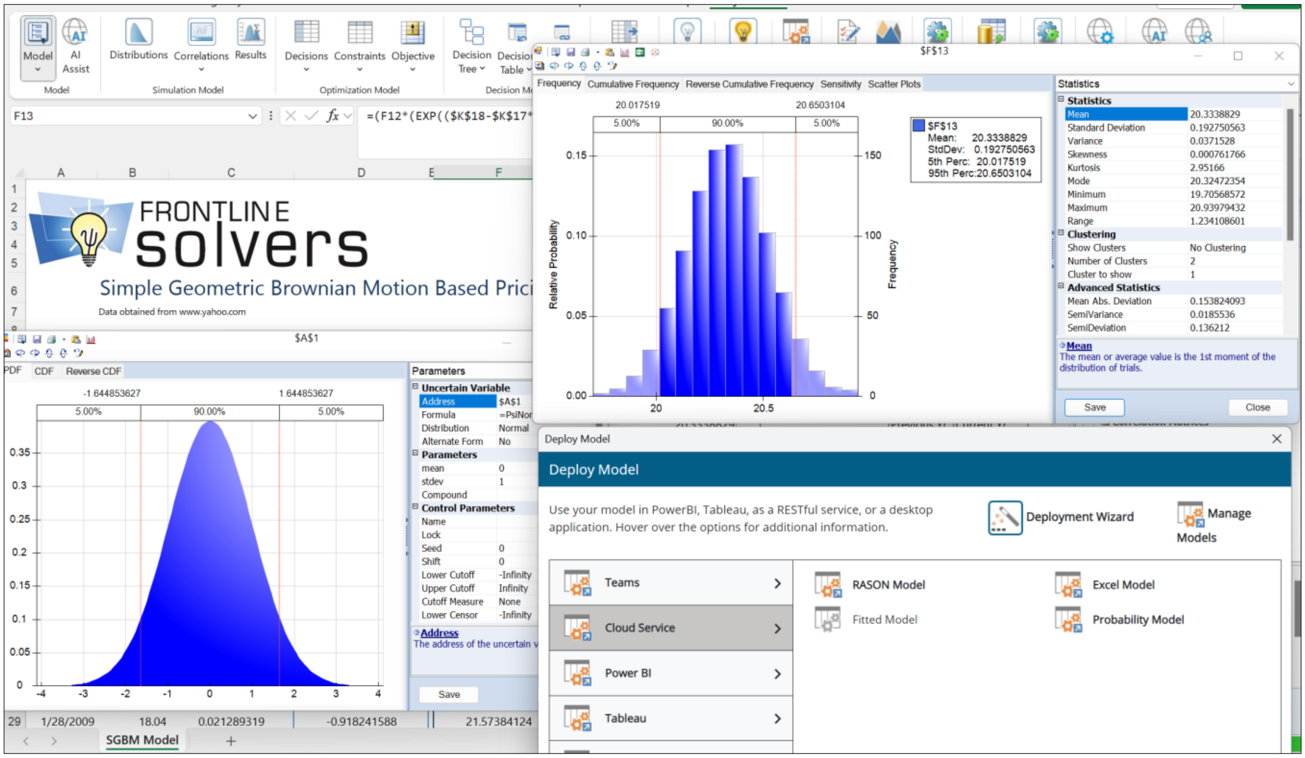
Version 2025 Q1

For Use With Excel 2013-2019

FRONTLINE solvers

Analytic Solver Optimization
Analytic Solver Simulation
User Guide



Statistics	
Statistics	
Mean	20.3338829
Standard Deviation	0.192750563
Variance	0.0371528
Skewness	0.000761766
Kurtosis	2.95166
Mode	20.32472354
Minimum	19.70568572
Maximum	20.93979432
Range	1.234108601
Clustering	
Show Clusters	No Clustering
Number of Clusters	2
Cluster to show	1
Advanced Statistics	
Mean Abs. Deviation	0.153824093
SemiVariance	0.0185536
SemiDeviation	0.136212

Mean
The mean or average value is the 1st moment of the distribution of trials.

Parameters	
Uncertain Variable	
Address	$A$1
Formula	=PsiNor
Distribution	Normal
Alternate Form	No
Parameters	
mean	0
stdev	1
Compound	
Control Parameters	
Name	
Lock	
Seed	0
Shift	0
Lower Cutoff	-Infinity
Upper Cutoff	Infinity
Cutoff Measure	None
Lower Censor	-Infinity

Address
The address of the uncertain v

$F$13
Mean: 20.3338829
StdDev: 0.192750563
5th Perc: 20.017519
95th Perc: 20.6503104

Deploy Model

Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Copyright

Software copyright 1991-2025 by Frontline Systems, Inc.
User Guide copyright 2025 by Frontline Systems, Inc.
GRG/LSGRG Solver: Portions copyright 1989 by Optimal Methods, Inc. SOCP Barrier Solver: Portions copyright 2002 by Masakazu Muramatsu. LP/QP Solver: Portions copyright 2000-2010 by International Business Machines Corp. and others. Neither the Software nor this User Guide may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without the express written consent of Frontline Systems, Inc., except as permitted by the Software License agreement below.

Trademarks

Frontline Solvers®, XLMiner®, Analytic Solver®, Risk Solver®, Premium Solver®, Solver SDK® and Rason® are trademarks of Frontline Systems, Inc. Windows and Excel are trademarks of Microsoft Corp. Gurobi is a trademark of Gurobi Optimization, Inc. Knitro is a trademark of Artelys. MOSEK is a trademark of MOSEK ApS. OptQuest is a trademark of OptTek Systems, Inc. Xpress^{MP} is a trademark of FICO, Inc. @RISK is a trademark of Palisade Corp.

Patent Pending

Systems and Methods for Automated Risk Analysis of Machine Learning Models.

Acknowledgements

Thanks to Dan Fylstra and the Frontline Systems development team for a 25-year cumulative effort to build the best possible optimization and simulation software for Microsoft Excel. Thanks to Frontline's customers who have built many thousands of successful applications, and have given us many suggestions for improvements.

Earlier versions of Risk Solver Pro and Risk Solver Platform have benefited from reviews, critiques, and suggestions from several risk analysis experts:

- Sam Savage (Stanford Univ. and AnalyCorp Inc.) for Probability Management concepts including SIPs, SLURPs, DISTs, and Certified Distributions.
- Sam Sugiyama (EC Risk USA & Europe LLC) for evaluation of advanced distributions, correlations, and alternate parameters for continuous distributions.
- Savvakis C. Savvides for global bounds, censor bounds, base case values, the Normal Skewed distribution and new risk measures.

How to Order

Contact Frontline Systems, Inc., P.O. Box 4288, Incline Village, NV 89450.
Tel (775) 831-0300 Fax (775) 831-0314 Email info@solver.com Web http://www.solver.com

Contents

Start Here: V2025 Q1 Essentials 14

Getting the Most from This User Guide 14
    License and Upgrade Options........ 14
    Desktop and Cloud Versions........ 14
    Installing the Software 14
    Getting Help Quickly........ 14
    Finding the Examples 15
    Using Existing Models........ 15
    Using Large-Scale Solver Engines 15
    Getting Started with Tutorials 15
    Getting and Interpreting Results........ 15
    Mastering Optimization and Simulation Concepts 16
    Using the Traditional Solver Parameters Dialog 16
    Automating Your Model with VBA 16

Installation and Add-Ins 17

What You Need........ 17
Installing the Software........ 17
    Installing Analytic Solver Cloud 17
    Installing Analytic Solver Desktop........ 19
Logging in the First Time 23
Uninstalling the Software 24
Activating and Deactivating the Software 24
If Something Goes Wrong 25
Cloud Versions........ 26
    Security and Privacy Considerations........ 27
    Analytic Solver Cloud........ 27
Using Solver Server to Solve Models........ 27
    Adding a Server........ 28
    Public RASON Server 29
    Private RASON Server (container)........ 30
    Solver SDK Server 31
    Selecting your Solver Server........ 34
    Solving Your Model on a Server........ 34

Analytic Solver Overview 36

Analytic Solver Product Line........ 36
    Desktop and Cloud versions........ 36
    Analytic Solver Academy 37
    Analytic Solver Upgrade........ 37
    Analytic Solver Optimization........ 38
    Analytic Solver Simulation 38
    Analytic Solver Data Science........ 38
    Analytic Solver Comprehensive........ 39
Enhancements in Recent Years........ 39
2017: Power BI, Analytic Solver Basic, and More........ 40

- 2018: Tableau, Data Science Workflows, and More 41
- 2019: Analytic Solver Cloud 42
  - Business Rules and DMN Decision Tables 43
- 2020: Simulation and Solver Engine Enhancements 43
  - Faster Solver Engines, Using Multiple Cores 43
  - Monte Carlo Simulation Enhancements 44
- What’s New in Analytic Solver V2020.5 44
  - Easily Deploy Your Model as a Cloud Service 44
  - More About RASON Decision Services 45
  - How You Can Use RASON 46
  - New Time Series Simulation Functions 47
  - New Optimization Result Functions 47
- What's New in Analytic Solver V2021 47
  - Lambda, Let and Box Functions 48
- What's New in Analytic Solver V2021.5 48
  - Automate Data Mining with Find Best Model 48
  - Better Simulation Models with Metalog Distributions and Fitting 49
  - Share Data Mining and Probability Models via RASON 49
  - And of Course, Optimization Enhancements 49
  - Analytic Solver Academy – Replaces Analytic Solver Basic 49
- What's New in Analytic Solver V2022 50
  - Faster Interaction, Faster Solves 50
- What’s New in Analytic Solver V2023 50
  - Automated Risk Analysis of Machine Learning Models 50
  - Synthetic Data Generation ‘For Free’ 51
- What’s New in Analytic Solver V2023 Q1 51
  - Faster LP/Quadratic Solver and Large-Scale LP/QP Solver Engine 51
  - More Plug-in Solver Engine Improvements 52
  - Greatly Improved “Deploy Your Model to Teams” Capability 52
  - Risk Analysis of Machine Learning Models Created in Other Software 52
- What’s New in Analytic Solver V2023 Q3 53
  - AI Agent: Ask for Help from ChatGPT “Trained on Analytic Solver” 53
  - Identify Inputs: Easily Set Up your Model for Data Updates and New Solves 54
- What’s New in Analytic Solver V2024 Q2 54
  - Solving Outside Excel 54
  - New Versions of Gurobi and OptQuest Solvers 55
  - PSI Interpreter Improvements 55
  - More About the PSI Interpreter: The Jacobian 55
  - Beyond the Jacobian: The Hessian 56
  - And There's (Much) More... 56
- What’s New in Analytic Solver V2024 Q3 57
  - New AI Assistant 57
  - Optimization Improvements 57
  - Simulation Improvements 58
- What’s New in Analytic Solver V2025 Q1 58
  - Usability: New Flexible Dialogs 58
  - Performance: New Nonlinear Solver Power 58
  - Standard Excel Formulas, New Solving Speed 59
- Optimization and Resource Allocation 60
- Simulation and Risk Analysis 61
- Optimal Solutions for Models with Uncertainty 61
- Automatic Model Transformation to RASON 62
- Sensitivity Analysis and Model Parameters 62
- Multiple Optimizations, Simulations or Calculations 62
- Decision Trees on the Spreadsheet 63
- Bringing Big Data into Excel using Apache Spark 63

Forecasting and Data Science 63
Large-Scale Solver Engines 65
Automatic Mode and Solution Time 66
Help, Guided Mode and Proactive Support 67

Help, Support, Licenses and Product Versions 68

Introduction 68
Working with Licenses V2025 Q1 68
    Licenses Tied to You, Not Your Computer 68
    Renamed Products Offering More 69
Getting Help 69
    Ask Question 70
    AI Assist 71
    Maximizing Benefits of AI Agent and AI Assist 78
    Analytic Solver Help Center 79
    Accessing Resources 79
    User Guides 80
    Example Models 80
    Knowledge Base 80
    Operating Mode 80
    Support Mode 81
    Submit a Support Ticket 81
    Solver Academy 82
    Video Tutorials/Live Webinars 82
    Learn more! 82
License Button 82
Product Selection Wizard 85
Using the Welcome Screen in Analytic Solver Desktop 87
License/Subscription in Analytic Solver Desktop 88
Context-Sensitive Help in Analytic Solver Desktop 89

Examples: Conventional Optimization 91

Introduction 91
A First Optimization Model 91
    The Model in Algebraic and Spreadsheet Form 92
    Defining and Solving the Optimization Model 93
    Using the Classical Solver Parameters Dialog 97
    Using Buttons on the Task Pane 98
    Exporting Data to Microsoft's Power BI 99
    Exporting Data to Tableau 102
    Tablea Data Extract 103
    Tableau Web Connector 104
Introducing the Standard Example Models 107
    Opening the Examples 107
    More Readable and Expandable Models 109
    Models, Worksheets and Workbooks 109
Linear Programming Examples 110
    Using the Output Tab and Creating Reports 110
    A Model with No Feasible Solution 113
    An ‘Accidentally’ Nonlinear Model 114
Nonlinear Optimization Examples 116
    Portfolio Optimization: Quadratic Programming 117
    Charts of the Objective and Constraints 118
    A Model with IF Functions 121

A Model with Cone Constraints 123
Solving an Optimization Model using Excel Online or Google Sheets 126

Examples: Simulation and Risk Analysis 135

Introduction 135
A First Simulation Example 135
A Business Planning Example 136
A What-If Spreadsheet Model 137
Defining a Simulation Model 137
Selecting Uncertain Functions 142
Running a Simulation 143
Interactive Simulation in Desktop Solver 147
Viewing the Full Range of Profit Outcomes 148
Analyzing Factors Influencing Net Profit 151
Interactive Simulation with Charts and Graphs 152
Charts and Graphs for Presentations 153
Exporting Data to Microsoft's Power BI 154
Exporting Data to Tableau 157
Tableau Web Connector 159
An Airline Revenue Management Model 162
A Single Simulation 162
Multiple Parameterized Simulations 165
Simulation Optimization 168
Using a Compound Distribution 170
Discrete Distribution passed to 1st Argument 171
Constant passed to 1st Argument 174
Publishing a Simulation Model to Excel Online or Google Sheets 176

Examples: Stochastic Optimization 189

Introduction 189
A Project Selection Model 189
Solving with Simulation Optimization 191
Solving Automatically 192
A Model with Chance Constraints 194
Solving with Robust Optimization 195
A Model with Recourse Decisions 198
Solving with Robust Optimization 199

Deploying Your Model 203

Introduction 203
Deploy Model Menu 204
Deployment Wizard 205
Deploy Model Menu 206
Conversion Exceptions 207
Manage Models 208
Teams 209
Teams Report 209
Cloud Service 213
Cloud Service - Rason 213
Cloud Service – Excel 228
Cloud Service Fitted Model 232
Cloud Service – Probability Model 233
Exceptions 243
Problem Limits 243

Exceptions.. 243
Power BI... 244
Installing Power BI 244
Power BI Managed Model 245
PsiDataSrc() Arguments.. 246
Deploy Model – Power BI – Embedded Model............................ 252
Power BI Fitted Model.. 257
Tableau 261
Tableau Managed Model... 262
Tableau Embedded Model... 267
Tableau Fitted Model 270
SolverSDK: Rason or Excel Model 278
RASON Model 278
Excel Model 282
Quick Test.. 284
Quick Test – Web Page... 285
Quick Test – Rason Model.. 287
Quick Test – Fitted Model... 288
Quick Test – Probability Model.. 291

Using Custom Functions 294

Introduction.. 294
LAMBDA Function Design... 294
LAMBDA Example Models 295
Optimization Example 295
Simulation Example... 296
Notes on the LAMBDA Function.. 298
Box Function Design.. 299
Execution of a Custom Function 302
Using a Box Iterator... 302
Box Function Loops... 305
Box Function Optimization Example.. 305
Simulation Example... 309
Implemented DMN functions for Conformance Level 3................ 314

Using Decision Tables 316

Introduction.. 316
Decision Table Structure.. 316
Additional Examples 327
Hit Policies 328
Merging Decision Table Results 334
Using Cascading Decision Tables 335
Using Decision Tables in Optimization and Simulation........................ 340
Optimization.. 340
Simulation 340
Using Decision Tables with Data Science 341
Input Parameters.. 341
Loan Strategy 342
More Information on Decision Tables.. 345

Creating Power BI Custom Visuals 347

Installing Power BI... 347
Creating a Custom Visual from an Optimization Model........................ 348
PsiDataSrc() Arguments.. 355

Troop Rotation Optimization Example 368
Using Optimization Parameters in Power BI 375
Simulation Example Model 376
Advanced Settings 381
Creating a Custom Visual from a Large Model 383
Error Messages 383

Creating Custom Extensions in Tableau 384

Introduction 384
Installing Tableau 384
Solving Optimization Models in Tableau 385
PsiDataSrc() Arguments 386
Deploying the Model to RASON 387
Deploying the Tableau Extension 388
Viewing the Tableau Extension File 390
Look, more inventory! 399
Gaining Control Over Your Data 401
Troop Rotation Optimization Example 408
Using Optimization Parameters in Tableau 416
Simulation Example Model 418
Error Messages 423

Examples: Parameters and Sensitivity Analysis 424

Introduction 424
Parameters and Results 424
Viewing Parameters in the Task Pane 424
Defining a Parameter 425
Automatic Parameter Identification 426
Defining Results 428
Identifying Inputs and Parameters 429
Using the Identify Input Tool 430
Sensitivity Analysis Reports and Charts 433
How Parameters are Varied 433
Creating Sensitivity Reports 434
Creating Sensitivity Charts 435
Optimization and Simulation Reports and Charts 436
When Optimizations and Simulations are Run 436

Examples: Decision Trees 437

Introduction 437
Creating Decision Trees 438
Creating Trees in Analytic Solver Desktop 438
Creating Trees in Analytic Solver Cloud 439
Creating and Editing Nodes 439
Creating and Editing Branches 440
Highlighting a Decision Strategy 441
Decision Trees in the Task Pane 443
Model Tab 443
Platform Tab 443

Getting Results: Optimization 445

Introduction 445
What Can Go Wrong, and What to Do About It 445

    Review Messages in the Output Tab 446
    Click the Solver Result Message for Help 450
    Choose Available Optimization Reports 451
    When Solving Takes a Long Time 452
    When the Solution Seems Wrong 455
    Problems with Poorly Scaled Models 455
    The Integer Tolerance Option 456
When Things Go Right: Getting Further Results 457
    Dual Values 457
    Multiple Solutions 458
    Multiple Parameterized Optimizations 459

Getting Results: Simulation 465

Introduction 465
What Can Go Wrong, and What to Do About It 465
    Review Messages in the Output Tab 466
    Click the Error Message for Help 466
    Role of the Random Number Seed 467
    When Simulation Takes a Long Time 467
    When Simulation Results Seem Wrong 470
When Things Go Right: Getting Further Results 470
    Using the Simulation Report 471
    Using the Uncertain Function Dialog 472
    Fitting a Classical Distribution to Simulation Results 473
    Fitting a Meta-Log Distribution to Simulation Results 478
    Charting Multiple Uncertain Functions 483
    Multiple Parameterized Simulations 483
    Time Series Forecasting 485
    Time Series Simulation 489
    Excel 2016 Forecast Functions 494

Getting Results: Stochastic Optimization 502

Introduction 502
What Can Go Wrong, and What to Do About It 502
    Review Messages in the Output Tab 503
    Click the Solver Result Message for Help 505
    Choose Available Optimization Reports 507
    When Solving Takes a Long Time 508
    When the Solution Seems Wrong 509
When Things Go Right: Getting Further Results 510

Dimensional Modeling 512

Introduction 512
Dimensions and Cubes 513
An Example Dimensional Model 514
Using Dimensional Modeling in Optimization 516
    Defining a Dimension 516
    Defining a Cube 519
    Cube Formulas: Operations and Dimensions 523
    Defining Outputs in Dimensional Models 526
    Building the Model 526
    Solving the Model 527
Additional Functions 529
Parametric Dimensions with Optimization 533

Solving with Parametric Dimensions 536
Dimension Options 539
Parameter Analysis 540
Active in Optimization, Simulation or Sensitivity 541
Using Dimensional Modeling in Simulation 544
Defining a Dimension 544
Defining a Cube 544
Defining an Output 545
Running a Simulation 546
Displaying Simulation Results 546
Displaying Statistic Values 549
Simulation Model with Parametric Dimensions 549
Using Dimensional Modeling with Pivot Tables 556
Using Sparse Cubes 560

Automating Optimization in VBA 569

Introduction 569
Why Use the Object-Oriented API? 569
Running Predefined Solver Models 570
Using the Macro Recorder 570
Adding a Reference in the VBA Editor 571
Analytic Solver Object Model 571
Using the VBA Object Browser 573
Programming the Object Model 573
Example VBA Code Using the Object Model 574
Evaluators Called During the Solution Process 574
Refinery.xlsx: Multiple Blocks of Variables and Functions 575
Adding New Variables and Constraints to a Model 576
CuttingStock.xlsx: Multiple Problems and Dynamically Generated Variables 577

Automating Simulation in VBA 582

Introduction 582
Adding a Reference in the VBA Editor 582
Activating Interactive Simulation 582
Using VBA to Control Analytic Solver 583
Analytic Solver Object Model 584
Using the VBA Object Browser 584
Using Analytic Solver Objects 585
Using Variable and Function Objects 586
Controlling Simulation Parameters in VBA 587
Evaluators Called During the Simulation Process 588
Working with Trials and Simulations in VBA 589
Displaying Normal or Error Trials 589
Using Multiple Simulations 589
Creating Uncertain Variables and SLURPs in VBA 590

Mastering Conventional Optimization Concepts 592

Introduction 592
Elements of Solver Models 592
Decision Variables and Parameters 592
The Objective Function 593
Constraints 593
Solutions: Feasible, “Good” and Optimal 594

More About Constraints.. 596
Functions of the Variables 599
    Convex Functions 600
    Linear Functions 601
    Quadratic Functions 602
    Nonlinear and Smooth Functions.. 603
    Discontinuous and Non-Smooth Functions 604
    Derivatives, Gradients, Jacobians, and Hessians 605
Optimization Problems and Solution Methods... 607
    Linear Programming 607
    Quadratic Programming 608
    Quadratically Constrained Programming 609
    Second Order Cone Programming 609
    Nonlinear Optimization... 610
    Global Optimization 611
    Non-Smooth Optimization 613
Integer Programming.. 615
    The Branch & Bound Method 616
    Cut Generation 616
    The Alldifferent Constraint 617
Looking Ahead to Models with Uncertainty 617

Mastering Simulation and Risk Analysis Concepts 619

Introduction.. 619
What Happens During Monte Carlo Simulation... 619
    Random Number Generation and Sampling 620
    The PSI Interpreter and Simulation... 622
Uncertain Functions, Statistics, and Risk Measures 623
    Measures of Central Tendency 623
    Measures of Variation... 623
    Risk Measures 624
    Quantile Measures 624
    Confidence Intervals 624
Uncertain Variables and Probability Distributions 624
    Discrete Vs. Continuous Distributions.. 625
    Metalog Distributions 625
    Bounded Vs. Unbounded Distributions... 625
    Analytic Vs. Custom Distributions 626
    Creating your Own Distributions When Past Data is Available 626
    When Past Data is Not Available... 627
    Using the Fit Feature... 627
    More Hints and Warnings 633
Dependence and Correlation 634
    Measuring Observed Correlation... 634
    Inducing Correlation Among Uncertain Variables 635
    Modeling Correlation Using Copulas 637
    Archimedean Copulas 638
    Elliptical Copulas 642
    Correlation Fitting 646
Using the Correlations Dialog.. 651
    Creating a Correlation Matrix.. 652
    Managing Copulas 655
    Removing a Correlation Matrix... 656
    Editing a Correlation Matrix 656
    Making a Matrix Consistent 657

Probability Management Concepts ... 660
    Analytic Distributions ... 660
    Certified Distributions and Stochastic Libraries ... 661
    Publishing and Using Certified Distributions ... 661
Stochastic Libraries: SIPs and SLURPs ... 662
    Using the PsiTSSip Function ... 663
    Creating Stochastic Libraries ... 664
    Using the DIST Feature ... 666
Creating and Using Certified Distributions ... 668
    Publishing Distributions with PsiCertify ... 668
    Using Distributions with PsiCertified ... 669
    Packaging Analytic Distributions ... 671

Mastering Stochastic Optimization Concepts 672

Introduction ... 672
    Certain and Uncertain Parameters ... 672
    Decision-Dependent Uncertainties ... 673
    Resolving Uncertainty and Recourse Decisions ... 673
    Uncertainty and Conventional Optimization ... 674
Elements of Solver Models ... 674
    Uncertain Variables ... 674
    Decision Variables ... 675
    Functions of the Variables and Uncertainties ... 675
    The Objective Function ... 676
    Constraints: Normal, Chance, Recourse ... 677
    Solutions: Feasible, Optimal, Well-Hedged ... 678
    More on Chance Constraints ... 679
    Diagnosing Your Model's Use of Uncertainty ... 682
Problems and Solution Methods ... 682
    Decision-Dependent Uncertainties ... 683
    Resolving Uncertainty and Recourse Decisions ... 683
Classes of Problems Involving Uncertainty ... 684
    One-Stage Problems ... 684
    Two-Stage Problems ... 685
Advanced Topics ... 687
    Bounds, Discretization, and Correlation ... 687
    Uncertainty Sets and Norms ... 687

Best Practices for Building Large-Scale Models 691

Introduction ... 691
Designing Large Solver Models ... 691
    Spreadsheet Modeling Hints ... 692
    Optimization Modeling Hints ... 693
    Using Multiple Worksheets and Data Sources ... 693
Quick Steps Towards Better Performance ... 694
Improving the Formulation of Your Model ... 695
    Convert MMULT Functions to DOTPRODUCT Functions ... 696
    Techniques Using Linear and Quadratic Functions ... 696
    Techniques Using Linear Functions and Binary Integer Variables ... 698
    Using Piecewise-Linear Functions ... 701
Organizing Your Model for Fast Solution ... 702
    Fast Problem Setup ... 702
    Using Array Formulas ... 704
    Using the Add-in Functions ... 705

References and Further Reading 708

Introduction 708
- Textbooks Using Analytic Solver 708
- Textbooks Using Solver or Premium Solver 709
- Other Textbooks 709
- Academic References 709

Start Here: V2025 Q1 Essentials

Getting the Most from This User Guide

License and Upgrade Options

Frontline Solvers V2025 Q1 feature a revised, simpler product line, based on **Analytic Solver**, that gives you access to **all** features, **all** the time for models of any size. It features a licensing system, tied to **you** and usable on more than one computer. Read about this in the chapter "Help, Support, Licenses and Product Versions."

Desktop and Cloud Versions

Analytic Solver V2025 Q1 comes in two versions: **Analytic Solver Desktop** – a traditional "COM add-in" that works only in Microsoft Excel for Windows PCs (desktops and laptops), and **Analytic Solver Cloud** – a modern "Office add-in" that works in Excel for Windows and Excel for Macintosh (desktops and laptops), and also in Excel for the Web using browsers such as Chrome, FireFox and Safari. Your license gives you access to both versions, and your Excel workbooks and optimization, simulation and data science models work in both versions, no matter where you save them (though OneDrive is most convenient). To get the most value from both versions, we *highly* recommend an **Office 365 subscription** to make your work easier and faster.

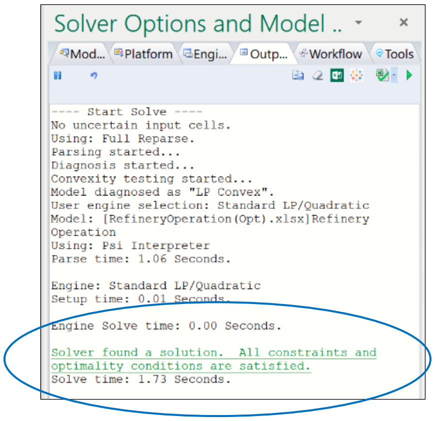
Installing the Software

Read the chapter "Installation and Add-Ins" for complete information on installing Analytic Solver Cloud and (if you wish) Analytic Solver Desktop. This chapter also explains how the Cloud and Desktop versions interact when both are installed, and how to install and uninstall both versions.

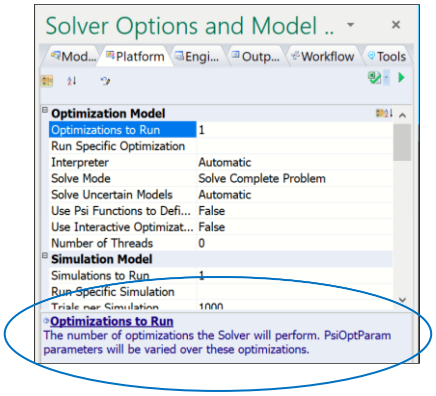
In brief, to add **Analytic Solver Cloud** version to your copy of Excel, you use the Excel Ribbon option **Insert – Get Add-ins** – no Setup program download or installation is required. To install **Analytic Solver Desktop** on Windows PCs, visit Solver.com, login using the email and password you used to register, then click Download on the top menu bar, and follow the instructions to download and run the SolverSetup program.

Getting Help Quickly

Choose Help on the Ribbon. You'll see several options, starting with **Help – Help Center**. Support Live Chat, Example Models, and User Guides are also available here. In Analytic Solver Desktop (only) you can also get quick online Help by clicking any <u>underlined</u> caption or message in the Task Pane.



```
---- Start Solve ----
No uncertain input cells.
Using: Full Reparse.
Parsing started...
Diagnosis started...
Convexity testing started...
Model diagnosed as "LP Convex".
User engine selection: Standard LP/Quadratic
Model: [RefineryOperation(Opt).xlsx]Refinery
Operation
Using: Psi Interpreter
Parse time: 1.06 Seconds.

Engine: Standard LP/Quadratic
Setup time: 0.01 Seconds.

Engine Solve time: 0.00 Seconds.

Solver found a solution.  All constraints and
optimality conditions are satisfied.
Solve time: 1.73 Seconds.
```



Optimization Model	
Optimizations to Run	1
Run Specific Optimization	
Interpreter	Automatic
Solve Mode	Solve Complete Problem
Solve Uncertain Models	Automatic
Use Psi Functions to Defi...	False
Use Interactive Optimizat...	False
Number of Threads	0
Simulation Model	
Simulations to Run	1
Run Specific Simulation	
Trials per Simulation	1000

Optimizations to Run

The number of optimizations the Solver will perform. PsiOptParam parameters will be varied over these optimizations.

Finding the Examples

Use Help – Examples on the Analytic Solver or Data Science Ribbon to open a list of example optimization and simulation models, and example data sets for data science, that you can open by clicking hyperlinks. See the chapter "Help, Support, Licenses and Product Subsets" for details. Some of these examples are used and described in the **Examples** chapters.

Using Existing Models

Open your existing workbook, developed in any previous version of Frontline Solvers or the standard Excel Solver. Your model should appear in the Task Pane; just click the Optimize or Simulate button. Read **Automatic Mode and Solution Time** in "Analytic Solver Overview" to understand how this mode can impact solution time.

Using Large-Scale Solver Engines

Read the Platform Solver Engines Guide to learn more about Frontline's eight large-scale Solver Engines for optimization, including their Solver Options and special Solver Result Messages. All Large-Scale Engines are pre-installed with both Analytic Solver Desktop and Analytic Solver Cloud; even if you don't yet have a paid license for one of them, you can use them in a "Test Run" as described in the chapter "Help, Support, Licenses and Product Versions."

Getting Started with Tutorials

To quickly gain a good grasp of Analytic Solver's optimization and simulation capabilities, work through the **Examples** chapters in this Guide. For Analytic Solver's data science capabilities, see the Data Science User Guide.

Getting and Interpreting Results

Learn how to interpret Analytic Solver's result messages, error messages, reports and charts, and how to run multiple parameterized optimizations and simulations in the **Getting Results** chapters.

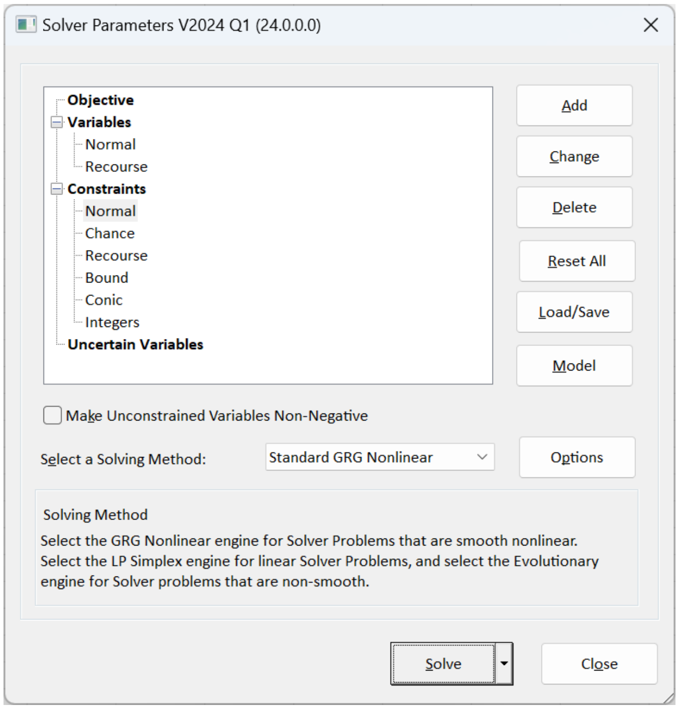
Mastering Optimization and Simulation Concepts

This guide can give you a professional education in simulation/risk analysis, conventional optimization, and stochastic optimization (with uncertainty). Go from beginner to expert and learn how to fully exploit the software by reading the **Mastering Concepts** chapters, and the Frontline Solvers Reference Guide.

Using the Traditional Solver Parameters Dialog

If you've used the **Solver in desktop Excel** (which Frontline Systems developed for Microsoft) and you're familiar with its Solver Parameters dialog, you can still use it in Analytic Solver Desktop: Click **Premium Solver** on the **Add-Ins tab** to display a slightly enhanced Solver Parameters dialog.

You can **go back and forth freely** between the Solver Parameters dialog and the new, modeless Task Pane and Ribbon. Within a few minutes, most users find the Task Pane and Ribbon faster and easier to use. If you've used the basic **Solver for Excel Online** (free in the Office Store), you're already familiar with use of the Task Pane, which is enhanced in Analytic Solver Desktop and Cloud.



☐ Make Unconstrained Variables Non-Negative

Select a Solving Method: Standard GRG Nonlinear

Solving Method

Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

Automating Your Model with VBA

VBA (Visual Basic for Applications) is available only in desktop Excel; it is not available in Excel for the Web. If you have Excel workbooks that use **macros** from the **Solver in desktop Excel** (such as SolverOK and SolverSolve) you can use them in Analytic Solver Desktop (**only**), provided that you use Tools **--** References in the VBA Editor to set or change the reference to **Analytic Solver V2025 Q1 Type Library**. You can use the code examples in the chapters "Automating Optimization in VBA" and "Automating Simulation in VBA" – but read about **RASON** in this Guide for a modern desktop + cloud approach.

Installation and Add-Ins

What You Need

You can use Analytic Solver Cloud in Excel for the Web through a browser (such as Edge, Chrome, Firefox or Safari), without installing anything else. This is the simplest and most flexible option, but it requires a constant Internet connection.

To use Analytic Solver Cloud in Excel Desktop on a PC or Mac, you must have a current version of Windows or iOS installed, and **you will need the latest Excel version installed via your Office 365 subscription** – older non-subscription versions, even Excel 2019, do not have all the features and APIs needed for modern Office add-ins like Analytic Solver Cloud.

To use Analytic Solver Desktop (Windows PCs only), you must have first installed Microsoft Excel 2013, 2016, 2019, or the latest Office 365 version on Windows 10, Windows 8, Windows 7, or Windows Server 2019, 2016 or 2012. (Windows Vista or Windows Server 2008 may work but are no longer supported.). It's not essential to have the standard Excel Solver installed.

Installing the Software

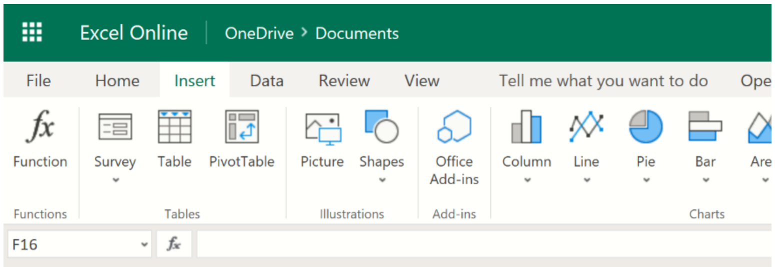
Installing Analytic Solver Cloud

Analytic Solver V2025 Q1 includes our next-generation offering, Analytic Solver Cloud – usable in the latest versions of Excel for Windows and Macintosh, and in Excel for the Web. Analytic Solver Cloud is divided into **two** add-ins that work closely together (since an Office add-in currently can have only one Ribbon tab): the **Analytic Solver** add-in builds optimization, simulation and decision table models, and the **Data Science** add-in builds data science or forecasting models.

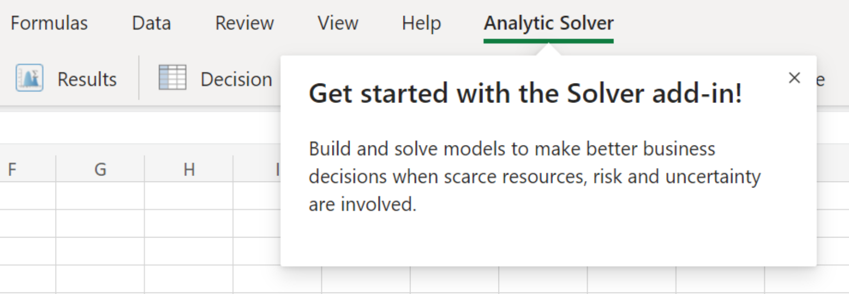
Both the Analytic Solver and Data Science add-ins support existing models created in previous versions of Analytic Solver. Your license for Analytic Solver allows you to use Analytic Solver Desktop in desktop Excel or Analytic Solver Cloud in either desktop Excel (latest version) *or* Excel for the Web.

To use the Analytic Solver and Data Science add-ins, **you must first "insert" them** for use in your copy of desktop Excel or Excel for the Web, while you are logged into your Office 365 account. Once you do this, the Analytic Solver and Data Science tabs will appear on the Ribbon in each new workbook you use.

To insert the add-ins for the first time, open desktop Excel (latest version) or Excel for the Web, click the **Insert** tab on the Ribbon, then click the button **Office Add-ins** or (if you see it) the smaller button **Get Add-ins**.



In the dialog box that appears, click the **Store** tab and type "Analytic Solver" into the Search box. Once you find the Analytic Solver add-in, click **Add**. After a moment, you should see the **Analytic Solver** tab appear on the Ribbon, with a note about how to "Get started with the Solver add-in!", as shown below.

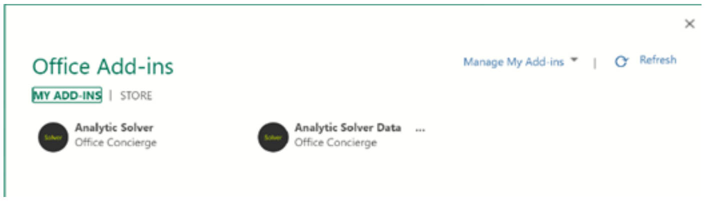


Get started with the Solver add-in!

Build and solve models to make better business decisions when scarce resources, risk and uncertainty are involved.

Repeat these steps to search for, locate and **Add** the Analytic Solver Data Science add-in. After a moment, you should see the **Data Science** tab appear on the Ribbon, with a similar "Get started" note.

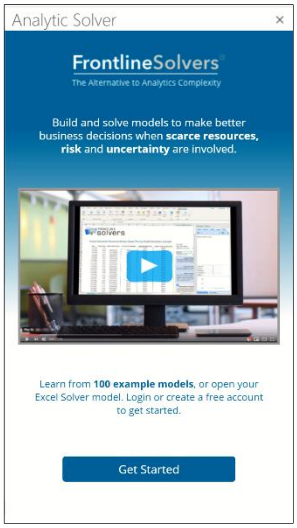
After you perform these steps (one time) to insert the Analytic Solver and Analytic Solver Data Science add-ins, they will appear under "My Add-ins". If you ever need to remove the add-ins, click the "**…**" symbol to the right of the add-in name, then click the **Remove** choice on the dropdown menu that appears.



Single Sign On Functionality

Analytic Solver Cloud includes Single Sign On functionality which automatically logs in users to their Analytic Solver Cloud account using their Microsoft 365 credentials. This means that if you've signed in to your Microsoft 365 account using the same email address you used to register on www.solver.com, then you will not be asked to login to Analytic Solver Cloud. Once you insert Analytic Solver cloud, you will have immediate access to all the features and functionality of Analytic Solver. As long as you remain signed in to your Microsoft 365 account, you'll never have to login to Analytic Solver Cloud again!

If you log out of Analytic Solver or your Office 365 credentials do not match your Solver credentials, you'll see the following welcome screen in the Solver task pane. Just click the Get Started button or click License – Login/Logout on the ribbon to login.



Build and solve models to make better business decisions when **scarce resources, risk** and **uncertainty** are involved.

Learn from **100 example models**, or open your Excel Solver model. Login or create a free account to get started.

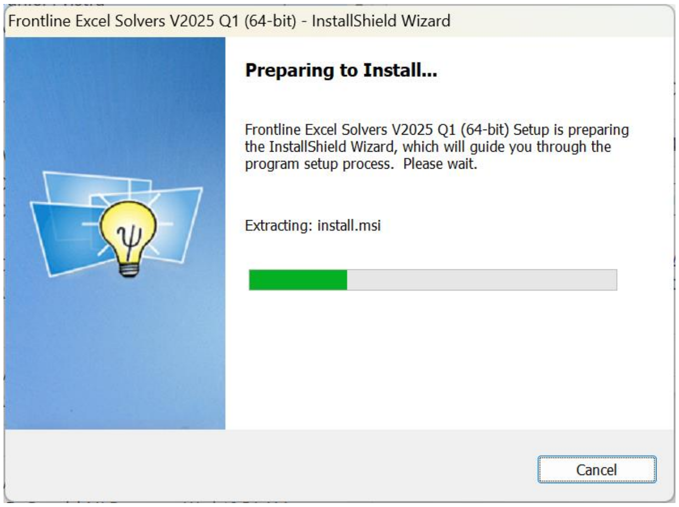
Installing Analytic Solver Desktop

To install Analytic Solver Desktop to work with any of the supported versions of Microsoft Excel (see above), simply run the program **SolverSetup.exe**, which installs all of the Solver program, Help and example files. SolverSetup.exe checks your system, detects what version of Office you are running (32-bit or 64-bit) and then downloads and runs the appropriate Setup program version.

Note that your copy of the Setup program will usually have a filename such as SolverSetup_12345.exe; the '12345' is your user account number on Solver.com.

When you run the Setup program, depending on your antivirus program or Windows security settings, you might be prompted with a message **"[reason such as new/unknown program]. Are you sure you want to run this software?"** You may safely click **Run** in response to this message.

You'll briefly see the standard Windows Installer dialog.

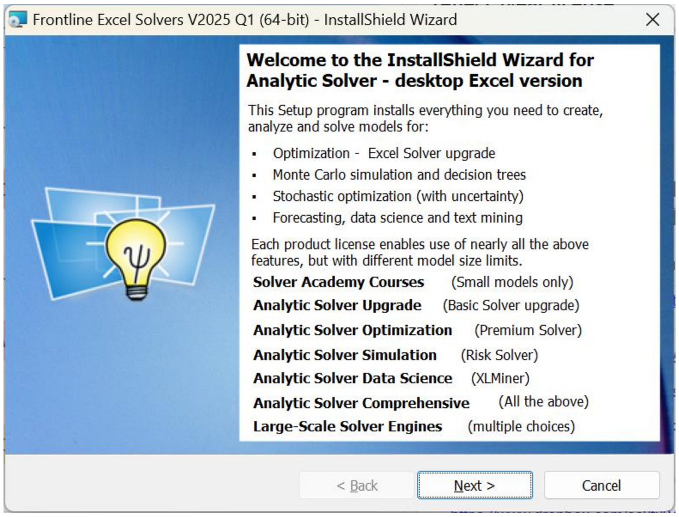

Frontline Excel Solvers V2025 Q1 (64-bit) - InstallShield Wizard

Preparing to Install...

Frontline Excel Solvers V2025 Q1 (64-bit) Setup is preparing the InstallShield Wizard, which will guide you through the program setup process. Please wait.

Extracting: install.msi

Then a dialog box like the one shown below should appear:


Frontline Excel Solvers V2025 Q1 (64-bit) - InstallShield Wizard

Welcome to the InstallShield Wizard for Analytic Solver - desktop Excel version

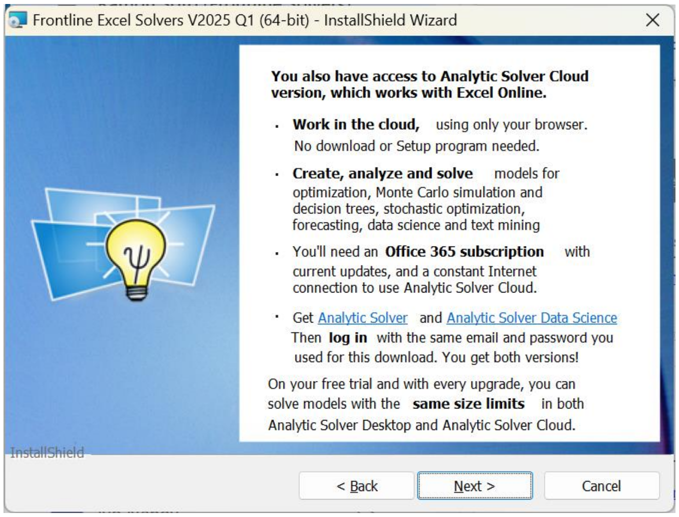
This Setup program installs everything you need to create, analyze and solve models for:

- Optimization - Excel Solver upgrade
- Monte Carlo simulation and decision trees
- Stochastic optimization (with uncertainty)
- Forecasting, data science and text mining

Each product license enables use of nearly all the above features, but with different model size limits.

Solver Academy Courses	(Small models only)
Analytic Solver Upgrade	(Basic Solver upgrade)
Analytic Solver Optimization	(Premium Solver)
Analytic Solver Simulation	(Risk Solver)
Analytic Solver Data Science	(XLMiner)
Analytic Solver Comprehensive	(All the above)
Large-Scale Solver Engines	(multiple choices)

Read this, so you know the *difference* between Analytic Solver Comprehensive and it's subsets. Then click **Next** to proceed – you'll see a dialog like the one below.


Frontline Excel Solvers V2025 Q1 (64-bit) - InstallShield Wizard

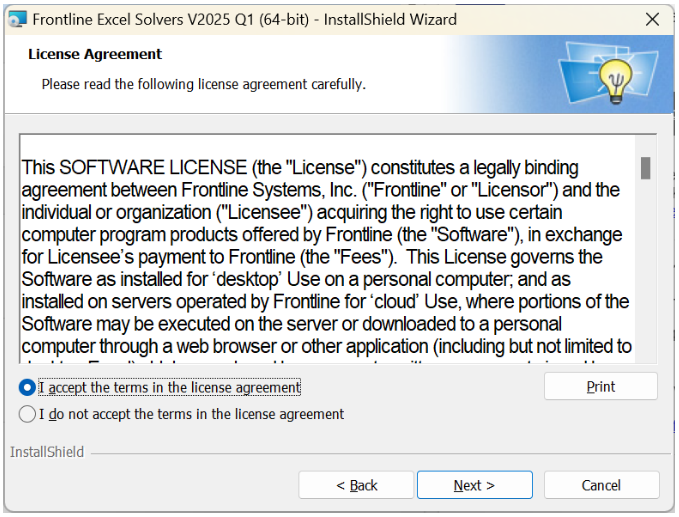
You also have access to Analytic Solver Cloud version, which works with Excel Online.

- **Work in the cloud,** using only your browser. No download or Setup program needed.
- **Create, analyze and solve** models for optimization, Monte Carlo simulation and decision trees, stochastic optimization, forecasting, data science and text mining
- You'll need an **Office 365 subscription** with current updates, and a constant Internet connection to use Analytic Solver Cloud.
- Get Analytic Solver and Analytic Solver Data Science Then **log in** with the same email and password you used for this download. You get both versions!

On your free trial and with every upgrade, you can solve models with the **same size limits** in both Analytic Solver Desktop and Analytic Solver Cloud.

Read this to learn how you can use Analytic Solver Cloud (our new Office add-in version) – "Excel Online" is another name for "Excel for the Web".

Next, the Setup program will ask if you accept Frontline's software license agreement. You must click "I accept" and **Next** in order to be able to proceed.


Frontline Excel Solvers V2025 Q1 (64-bit) - InstallShield Wizard

License Agreement

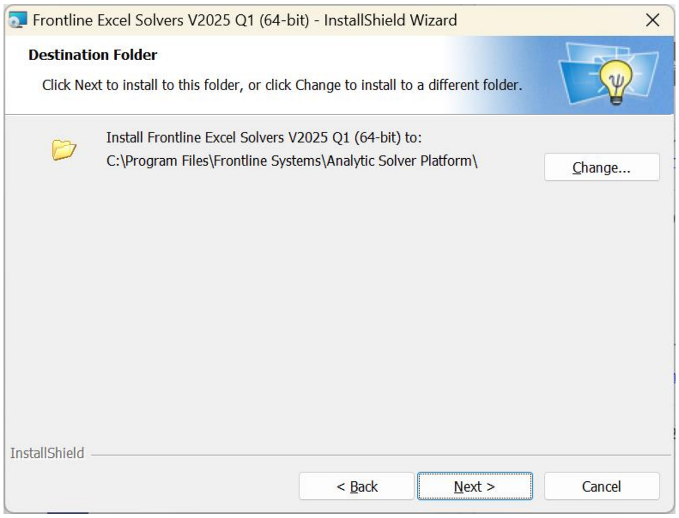
Please read the following license agreement carefully.

This SOFTWARE LICENSE (the "License") constitutes a legally binding agreement between Frontline Systems, Inc. ("Frontline" or "Licensor") and the individual or organization ("Licensee") acquiring the right to use certain computer program products offered by Frontline (the "Software"), in exchange for Licensee's payment to Frontline (the "Fees"). This License governs the Software as installed for 'desktop' Use on a personal computer; and as installed on servers operated by Frontline for 'cloud' Use, where portions of the Software may be executed on the server or downloaded to a personal computer through a web browser or other application (including but not limited to

● I accept the terms in the license agreement

○ I do not accept the terms in the license agreement

The Setup program then displays a dialog box like the one shown below, where you can select or confirm the folder to which files will be copied (normally C:\Program Files\Frontline Systems\Analytic Solver Platform (for 32-bit Excel on 64-bit Windows, C:\Program Files (x86)\Frontline Systems\Analytic Solver Platform). Click **Next** to proceed.

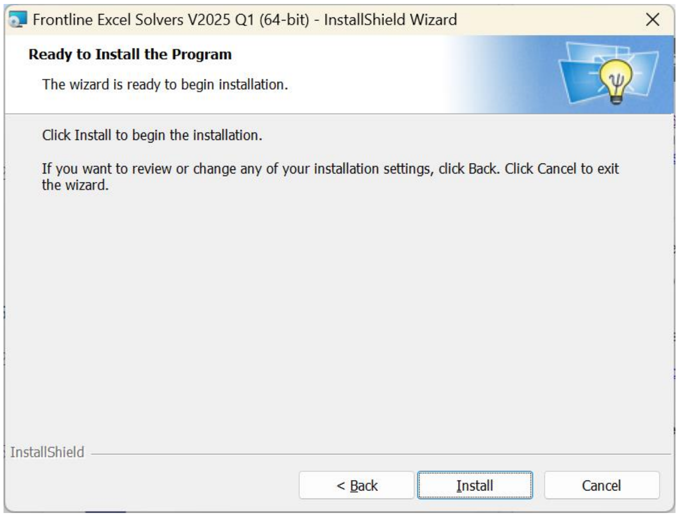

Frontline Excel Solvers V2025 Q1 (64-bit) - InstallShield Wizard

Destination Folder

Click Next to install to this folder, or click Change to install to a different folder.

Install Frontline Excel Solvers V2025 Q1 (64-bit) to:
C:\Program Files\Frontline Systems\Analytic Solver Platform\

You'll see a dialog confirming that the preliminary steps are complete, and the installation is ready to begin.



Frontline Excel Solvers V2025 Q1 (64-bit) - InstallShield Wizard

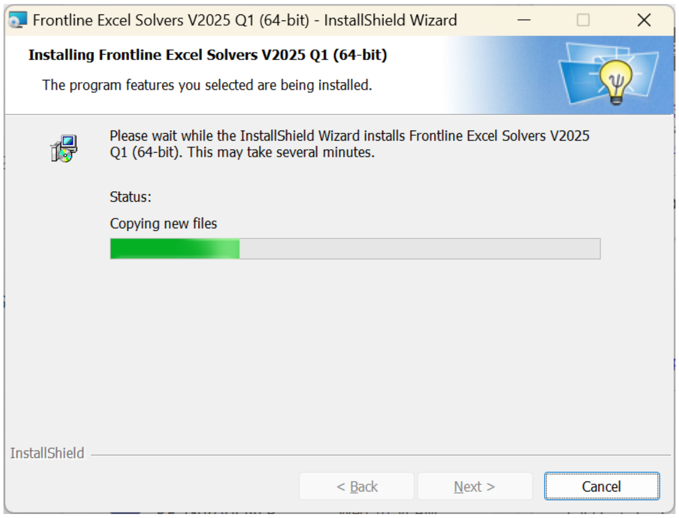
Ready to Install the Program

The wizard is ready to begin installation.

Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

After you click **Install**, the Analytic Solver files will be installed, and the program file RSPAddin.xll will be registered as a COM add-in (which may take some time). A progress dialog may appear; be patient, since this process could take longer than it has in previous Solver Platform releases.



Frontline Excel Solvers V2025 Q1 (64-bit) - InstallShield Wizard

Installing Frontline Excel Solvers V2025 Q1 (64-bit)

The program features you selected are being installed.

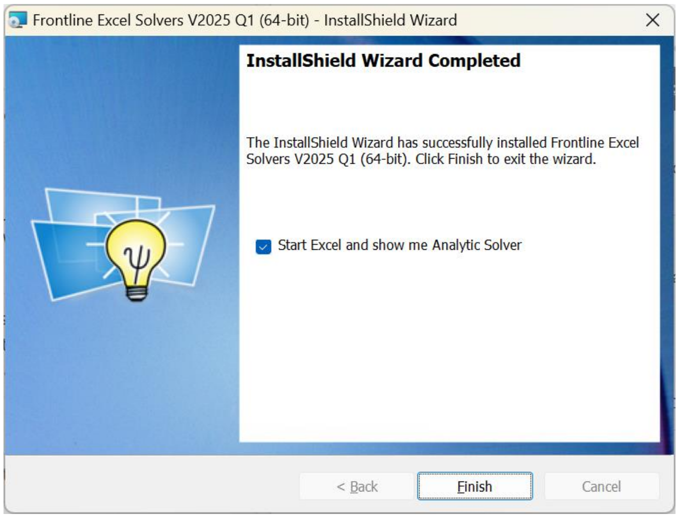
Please wait while the InstallShield Wizard installs Frontline Excel Solvers V2025 Q1 (64-bit). This may take several minutes.

Status:

Copying new files

When the installation is complete, you'll see a dialog box like the one below.

Click **Finish** to exit the installation wizard.



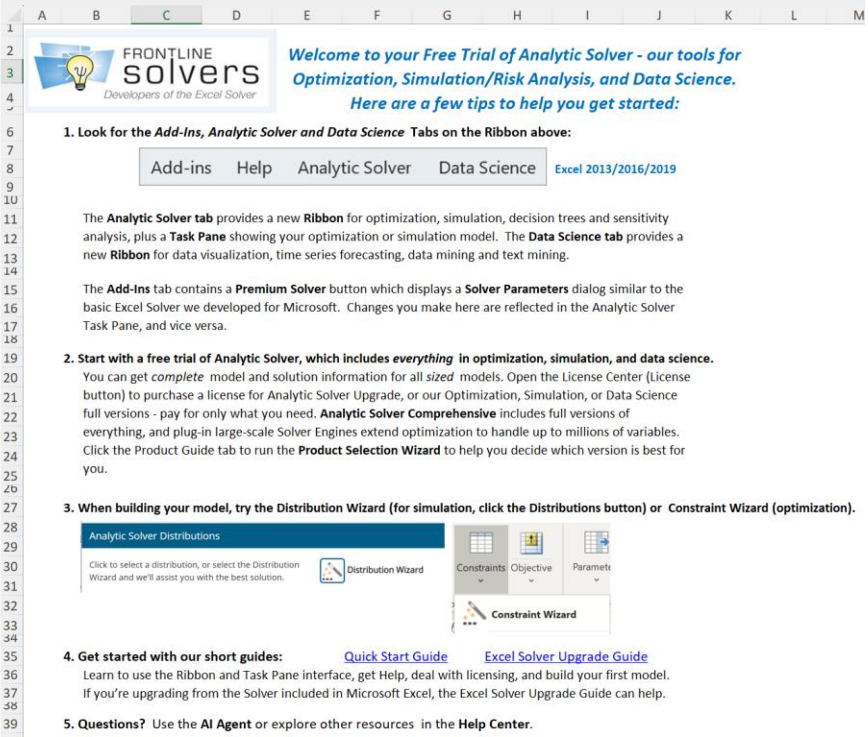
Frontline Excel Solvers V2025 Q1 (64-bit) - InstallShield Wizard

InstallShield Wizard Completed

The InstallShield Wizard has successfully installed Frontline Excel Solvers V2025 Q1 (64-bit). Click Finish to exit the wizard.

☑ Start Excel and show me Analytic Solver

The full Analytic Solver product family is now installed. With your trial and paid license, you can access every feature of the software, including forecasting and data science, simulation and risk analysis, and conventional and stochastic optimization. Simply click "Finish" and Microsoft Excel will launch with a Welcome workbook containing information to help you get started quickly.



Welcome to your Free Trial of Analytic Solver - our tools for Optimization, Simulation/Risk Analysis, and Data Science.
Here are a few tips to help you get started:

1. Look for the *Add-Ins, Analytic Solver and Data Science* Tabs on the Ribbon above:

The **Analytic Solver tab** provides a new **Ribbon** for optimization, simulation, decision trees and sensitivity analysis, plus a **Task Pane** showing your optimization or simulation model. The **Data Science tab** provides a new **Ribbon** for data visualization, time series forecasting, data mining and text mining.

The **Add-Ins** tab contains a **Premium Solver** button which displays a **Solver Parameters** dialog similar to the basic Excel Solver we developed for Microsoft. Changes you make here are reflected in the Analytic Solver Task Pane, and vice versa.

2. Start with a free trial of Analytic Solver, which includes *everything* in optimization, simulation, and data science.
You can get *complete* model and solution information for all *sized* models. Open the License Center (License button) to purchase a license for Analytic Solver Upgrade, or our Optimization, Simulation, or Data Science full versions - pay for only what you need. **Analytic Solver Comprehensive** includes full versions of everything, and plug-in large-scale Solver Engines extend optimization to handle up to millions of variables. Click the Product Guide tab to run the **Product Selection Wizard** to help you decide which version is best for you.

3. When building your model, try the Distribution Wizard (for simulation, click the Distributions button) or Constraint Wizard (optimization).

Click to select a distribution, or select the Distribution Wizard and we'll assist you with the best solution.

4. Get started with our short guides: Quick Start Guide Excel Solver Upgrade Guide
Learn to use the Ribbon and Task Pane interface, get Help, deal with licensing, and build your first model.
If you're upgrading from the Solver included in Microsoft Excel, the Excel Solver Upgrade Guide can help.

5. Questions? Use the **AI Agent** or explore other resources in the **Help Center**.

Logging in the First Time

In Analytic Solver V2025 Q1, your license is associated with **you**, and may be used on **more than one PC**. For example, you can run SolverSetup to install the desktop software on your office PC, your company laptop, and your PC at home. But only **you** can use Analytic Solver, and only on **one** of these computers **at a time**. It is unlawful to "share" your license with another human user.

The first time you run Analytic Solver (Desktop or Cloud) after installing the software on a new computer, when you next start Excel and visit the Analytic Solver tab on the Ribbon, **you will be prompted to login**. Enter the **email address** and **password** that you used to register on Solver.com. Once you've done this in Analytic Solver Desktop, your identity will be "remembered," so you won't have to login every time you start Excel and go to one of the Analytic Solver tabs. In Analytic Solver Cloud, you may be asked to login more frequently.

You can login and logout at any time, by clicking License – Login/Logout in both Analytic Solver Desktop and Analytic Solver Cloud. If you share use of a single physical computer with other Analytic Solver users, be careful to **login** with your own email and password, and **log out when you're done** – if you don't, other users could access private files in your cloud account, or use up your allotted CPU time or storage.

When you move from one computer to another, you should **log out** on one and **log in** on the other. As a convenience, if you log in to Analytic Solver on a new computer when you haven't logged out on the old computer, Analytic Solver will let you know, and offer to automatically log you out on the other computer.

Uninstalling the Software

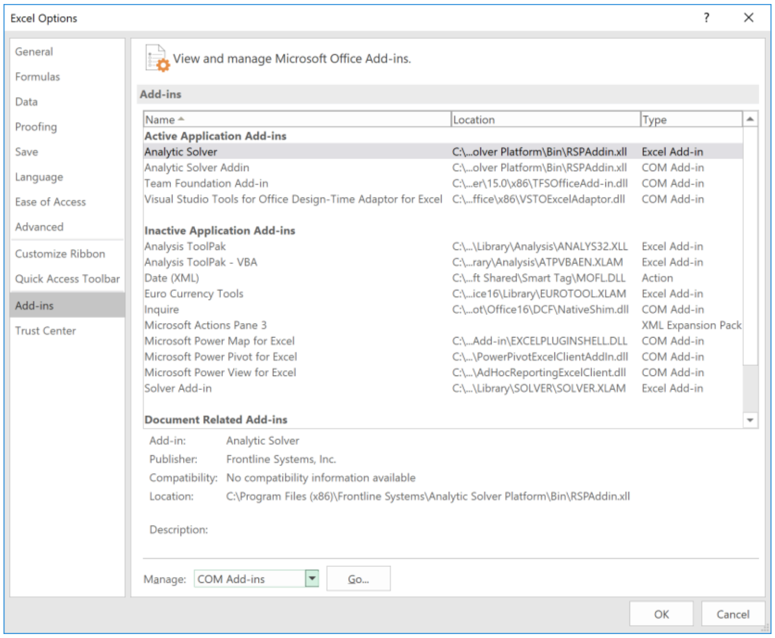
To uninstall Analytic Solver Desktop, just run the **SolverSetup** program as outlined above. You'll be asked to confirm that you want to remove the software.

You can also uninstall by choosing **Control Panel** from the **Start** menu, and double-clicking the **Programs and Features** or **Add/Remove Programs** applet. In the list box below "Currently installed programs," scroll down if necessary until you reach the line, "Frontline Excel Solvers V2025 Q1," and click the Uninstall/Change or Add/Remove… button. Click OK in the confirming dialog box to uninstall the software.

Activating and Deactivating the Software

Analytic Solver Desktop's main program file RSPAddin.xll is a COM add-in, an XLL add-in, and a COM server. A reference to the add-in **Solver.xla** is needed if you wish to use the "traditional" VBA functions to control Analytic Solver, instead of its new VBA Object-Oriented API.

In modern versions of Excel, you can manage all types of add-ins from one dialog, reached by clicking **File – Options -- Addins**.



View and manage Microsoft Office Add-ins.

Add-ins

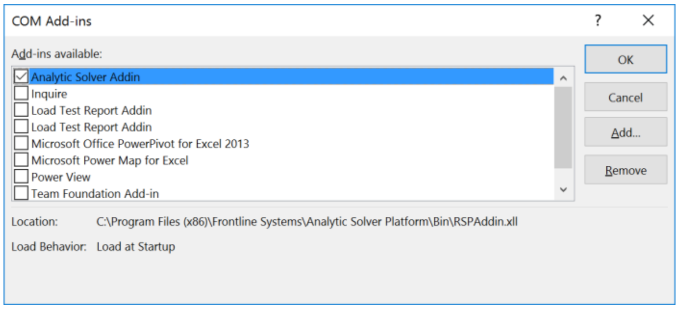
Name	Location	Type
Active Application Add-ins		
Analytic Solver	C:\...olver Platform\Bin\RSPAddin.xll	Excel Add-in
Analytic Solver Addin	C:\...olver Platform\Bin\RSPAddin.xll	COM Add-in
Team Foundation Add-in	C:\...er\15.0\x86\TFSOfficeAdd-in.dll	COM Add-in
Visual Studio Tools for Office Design-Time Adaptor for Excel	C:\...ffice\x86\VSTOExcelAdaptor.dll	COM Add-in
Inactive Application Add-ins		
Analysis ToolPak	C:\...\Library\Analysis\ANALYS32.XLL	Excel Add-in
Analysis ToolPak - VBA	C:\...rary\Analysis\ATPVBAEN.XLAM	Excel Add-in
Date (XML)	C:\...ft Shared\Smart Tag\MOFL.DLL	Action
Euro Currency Tools	C:\...ice16\Library\EUROTOOL.XLAM	Excel Add-in
Inquire	C:\...ot\Office16\DCF\NativeShim.dll	COM Add-in
Microsoft Actions Pane 3		XML Expansion Pack
Microsoft Power Map for Excel	C:\...Add-in\EXCELPLUGINSHELL.DLL	COM Add-in
Microsoft Power Pivot for Excel	C:\...\PowerPivotExcelClientAddIn.dll	COM Add-in
Microsoft Power View for Excel	C:\...\AdHocReportingExcelClient.dll	COM Add-in
Solver Add-in	C:\...\Library\SOLVER\SOLVER.XLAM	Excel Add-in
Document Related Add-ins		

Add-in: Analytic Solver
Publisher: Frontline Systems, Inc.
Compatibility: No compatibility information available
Location: C:\Program Files (x86)\Frontline Systems\Analytic Solver Platform\Bin\RSPAddin.xll

Description:

Manage: COM Add-ins

You can manage add-ins by selecting the type of add-in from the dropdown list at the bottom of this dialog. For example, if you select COM Add-ins from the dropdown list and clock the Go button, the dialog shown below appears.



Add-ins available:

☑ Analytic Solver Addin
☐ Inquire
☐ Load Test Report Addin
☐ Load Test Report Addin
☐ Microsoft Office PowerPivot for Excel 2013
☐ Microsoft Power Map for Excel
☐ Power View
☐ Team Foundation Add-in

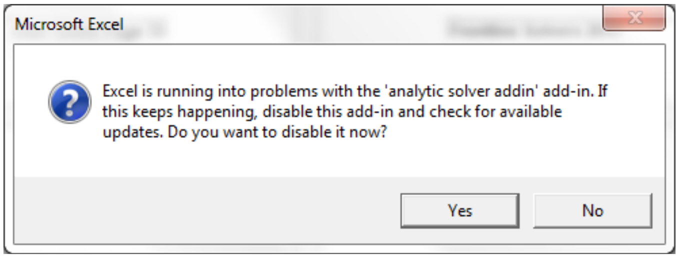
Location: C:\Program Files (x86)\Frontline Systems\Analytic Solver Platform\Bin\RSPAddin.xll
Load Behavior: Load at Startup

If you uncheck the box next to "Analytic Solver Addin" and click OK, you will deactivate the Analytic Solver COM add-in, which will remove the Analytic Solver tab from the Ribbon in desktop Excel, and also remove the PSI functions for optimization from the Excel Function Wizard.

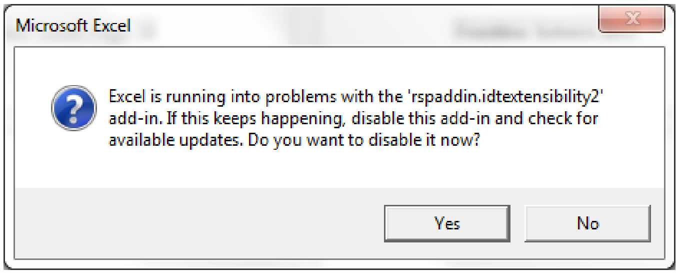
If Something Goes Wrong

Under certain circumstances, desktop Microsoft Excel can "crash" or shut down, so it must be restarted. This can happen for a variety of reasons, including bugs in Excel itself, in the Analytic Solver software, or in other add-ins.

When Excel restarts, you may see one of the following messages:



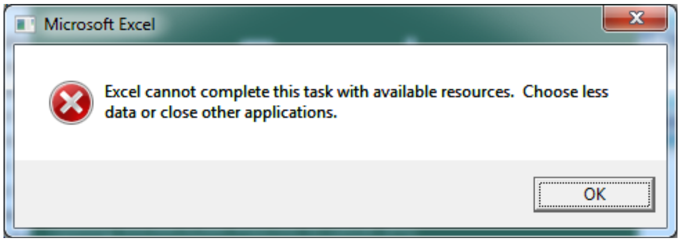
Excel is running into problems with the 'analytic solver addin' add-in. If this keeps happening, disable this add-in and check for available updates. Do you want to disable it now?



Excel is running into problems with the 'rspaddin.idtextensibility2' add-in. If this keeps happening, disable this add-in and check for available updates. Do you want to disable it now?

If you see these messages, you should usually **click the No button**. If you've experienced a problem while using Analytic Solver software, please contact Frontline Systems Technical Support as described below. If you click the Yes button, Excel will disable the Analytic Solver add-in, and the Analytic Solver and Data Science tabs will no longer appear on the Ribbon. To re-enable Analytic Solver and restore these Ribbon tabs, see the preceding section "Activating and Deactivating the Software:" you should select **File – Options**, then **Add-Ins**, then **Manage** COM Add-ins, **Go**. Then **check** the box next to "Analytic Solver Addin" and click OK.

In rare circumstances, Analytic Solver users have reported the following error message appearing upon opening of Excel.



Excel cannot complete this task with available resources. Choose less data or close other applications.

This is a generic error from Microsoft Excel that can occur if you have been running Excel for a long period of time without ever deleting *Microsoft Excel's* temp files. (Note: These temp files are not generated by Analytic Solver software; they are generated through Microsoft Excel.) To resolve this error, click OK on the error message, then using Windows Explorer browse to C:\Users\<username>\AppData\Roaming\Microsoft\Excel and delete all *files* in this folder. Restart Excel. The issue should be resolved.

Frontline Systems Technical Support team may be contacted via phone (888-831-0333), email (support@solver.com) or Live Chat (Help – Support Live Chart on the Analytic Solver ribbon).

Cloud Versions

With your free trial or paid license, you can use Analytic Solver in desktop Excel, and its cloud-based counterpart, Analytic Solver Cloud.

Analytic Solver Cloud is a "modern Office add-in" that works in Excel for the Web, Excel for Windows and Excel for Mac (latest versions).

- All versions offer a Ribbon user interface featuring nearly-identical buttons and menus, and a Task Pane that summarizes models and provides access to Platform and Engine options.
- All versions use the same modeling languages (Excel formulas and our RASON® modeling language, handled by our PSI Interpreter), and both use

the same algorithmic "engines" for mathematical optimization, Monte Carlo simulation and risk analysis, forecasting, data science and text mining.

- All versions can create or open existing optimization, simulation and data science models and datasets in Excel workbooks – and you can easily move such workbooks back and forth between desktop and cloud.

Security and Privacy Considerations

When you use Analytic Solver Desktop in Excel for Windows, your model is solved on the same computer, in the same memory and running process where Excel for Windows runs. You can save your workbook on your own computer, on Microsoft OneDrive “in the cloud”, or elsewhere.

When you use Analytic Solver Cloud, your workbook is stored, at least temporarily, “in the cloud”, and your model is **solved “in the cloud”**, using Frontline’s RASON servers on Microsoft Azure. While many steps are taken to ensure your security and privacy, you should understand and be comfortable with how the technology works:

When the browser running on your computer communicates with either Excel for the Web, all the information transmitted is encrypted using **Transport Layer Security (TLS) 1.2**, as is true for all “https” websites.

When you run or solve a data science, optimization or simulation model, a copy of your Excel workbook is transmitted to Frontline’s RASON servers, again using **TLS 1.2**. A copy of your workbook is stored temporarily on these Azure-based servers, but is always **encrypted** “at rest” and “in motion”. After the model is run or solved, all copies of your workbook are **deleted**; only a log of filename, model size and time taken to solve remains on the RASON servers.

Analytic Solver Cloud

Analytic Solver Cloud can be used with Excel for the Web, Excel for Windows and Excel for Macintosh (latest versions). Excel for the Web works the same way as desktop Excel so there's no learning curve – you can use it right away.

It's easy to move files between Analytic Solver Desktop and Analytic Solver Cloud products by simply saving your existing files to your Microsoft OneDrive account. Files saved on OneDrive may be opened in Microsoft Online or desktop Office. For Analytic Solver Cloud, **you will need the latest version installed via your Office 365 subscription** – older non-subscription versions, even Excel 2019, do not have all the features and APIs needed for modern Office add-ins like Analytic Solver Cloud.

Solver Home Tab Removed

The Solver Home tab was removed in Analytic Solver Desktop V2019. You can use the License menu to Login and Logout, browse to www.solver.com, start a Live Chat, etc. See the section below for more information.

Using Solver Server to Solve Models

With Analytic Solver Desktop (only), you also have an option to solve your optimization model, or run your simulation model on a corporate server, that may be more powerful than your desktop or laptop computer. Results appear in your spreadsheet, just as if you had solved the model on your own PC instead of the server. To do this, you use a separate software tool called Solver Server.

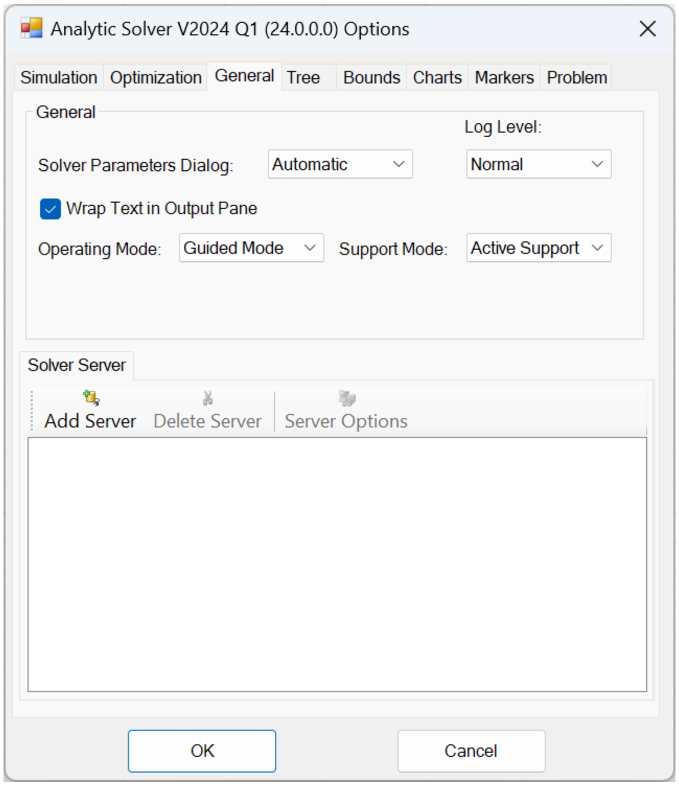
Solver Server is shipped as part of our Solver SDK product; a client for Solver Server is built into each copy of Analytic Solver software, and each copy of Solver SDK Comprehensive, Platform or Pro. You can also create your own client programs, even on mobile devices, using JavaScript and/or PHP.

If you would like more information on this service, please contact us at sales@solver.com. Note that Solver Server is a *separate* offering from our cloud version Analytic Solver Cloud, and from the Analytic Solver "Deploy Model" feature that translates your model into our RASON modeling language and may open it to be run at https://Rason.com.

Note: SolverServer is not used with, or applicable to Analytic Solver Cloud.

Adding a Server

To set up a connection to Solver Server from a client machine running Analytic Solver software, click the Options button on the Solver Ribbon, then click the General tab.



Solver Parameters Dialog: Automatic
Log Level: Normal
☑ Wrap Text in Output Pane
Operating Mode: Guided Mode
Support Mode: Active Support

Next, click **Add Server** to display the following dialog.

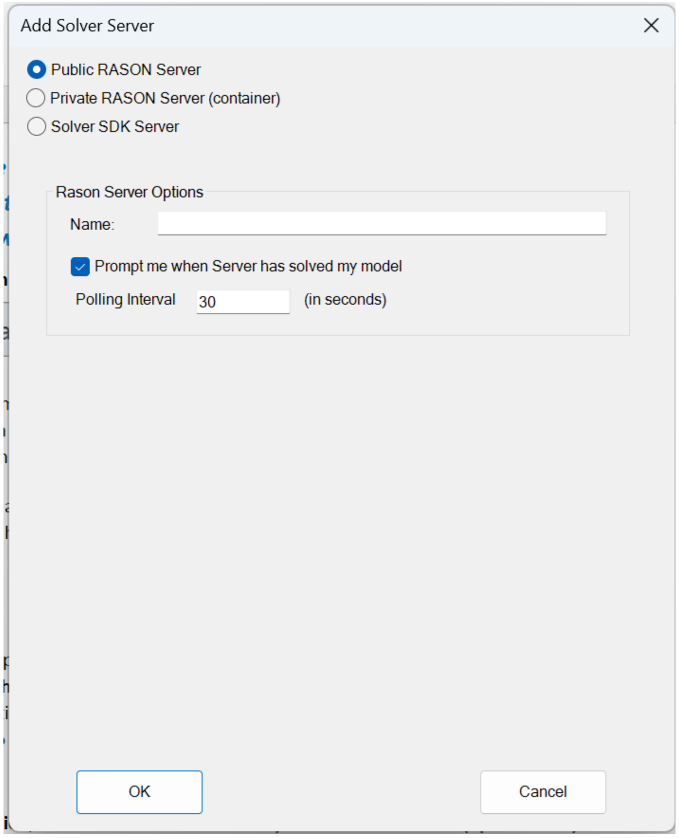
Select one of the following:

- Public RASON Server: (Default) Select this option to use Frontline's public RASON Server to solve your optimization, simulation or stochastic optimization model. (All licenses for Analytic Solver Comprehensive, Optimization, Simulation, and Data Science allow for a limited number of solves on Frontline's public server. For more information, see the Deploying Your Model chapter.)

- Private RASON Server (container): Select this option to use a Frontline RASON Container. For more information on Frontline's Container options, contact our Sales team at Sales@solver.com.
- Solver SDK Server: Select this option to use a Solver SDK Server to solve your model. For more information on Frontline's Solver SDK, contact our Sales team at Sales@solver.com.

Public RASON Server

When Public RASON Server is selected, the default setting, the following options appear on the bottom half of the Add Solver Server dialog. Choose this option to utilize Frontline's public RASON Server for solving your models. All licenses for Analytic Solver Comprehensive, Optimization, Simulation, and Data science include a restricted number of solves on this server. For further details, refer to the Deploying Your Model chapter.


Add Solver Server

● Public RASON Server
○ Private RASON Server (container)
○ Solver SDK Server

Rason Server Options

Name:

☑ Prompt me when Server has solved my model

Polling Interval 30 (in seconds)

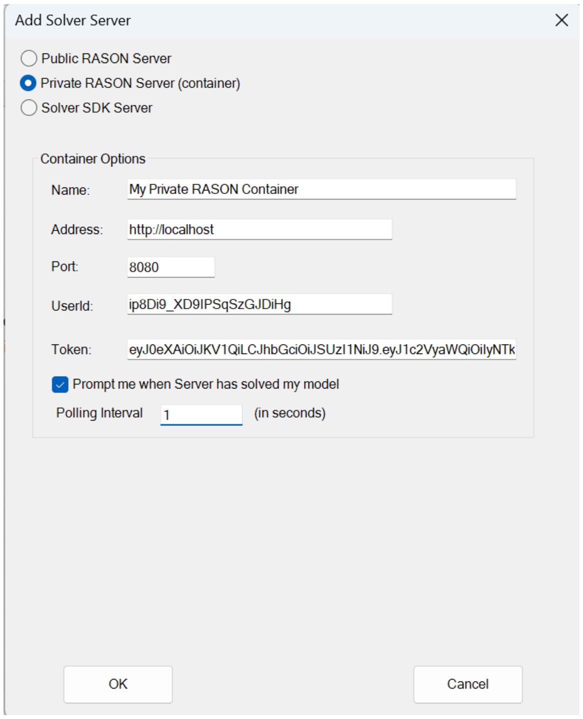
Name: Enter a convenient name for your server here. The name will appear as the name of this server in the Options dialog General tab display.

Select **Prompt me when Server has solved my model** in order to receive a message box indicating that the Solver has completed the solve.

Polling Interval: Please provide a positive number indicating the interval, in seconds, between server checks to determine if the solution process has finished.

Private RASON Server (container)

When Private RASON Server is selected, the following options appear on the bottom half of the Add Solver Server dialog. Select this option to use a Frontline RASON Container. For more information on Frontline's Container options, contact our Sales team at Sales@solver.com.



○ Public RASON Server
● Private RASON Server (container)
○ Solver SDK Server

Container Options
Name: My Private RASON Container
Address: http://localhost
Port: 8080
UserId: ip8Di9_XD9IPSqSzGJDiHg
Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJ1c2VyaWQiOiIyNTk
☑ Prompt me when Server has solved my model
Polling Interval 1 (in seconds)

Name: Enter a convenient name for your server here. The name will appear as the name of this server in the Options dialog General tab display.

Address: The Frontline container, by default, points to 80 and 443 to the external host ports 8080 and 8081. HTTP: "8080:80", HTTPS: "8081:443.

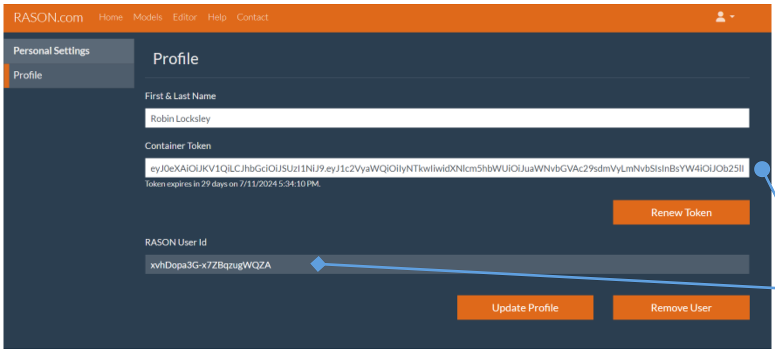
UserID: Enter the UserID as shown on the View Account page in the Frontline Client Container. (See screenshot below).

Token: Enter the Token as shown on the View Account page in the Frontline Client Container. (See screenshot below.) Note that this token will expire after a number of days. See your IT administrator or the Frontline Container Guide for more information.

Select **Prompt me when Server has solved my model** in order to receive a message box indicating that the Solver has completed the solve.

Polling Interval: Please provide a positive number indicating the interval, in seconds, between server checks to determine if the solution process has finished.

View Account in Frontline Container



Profile

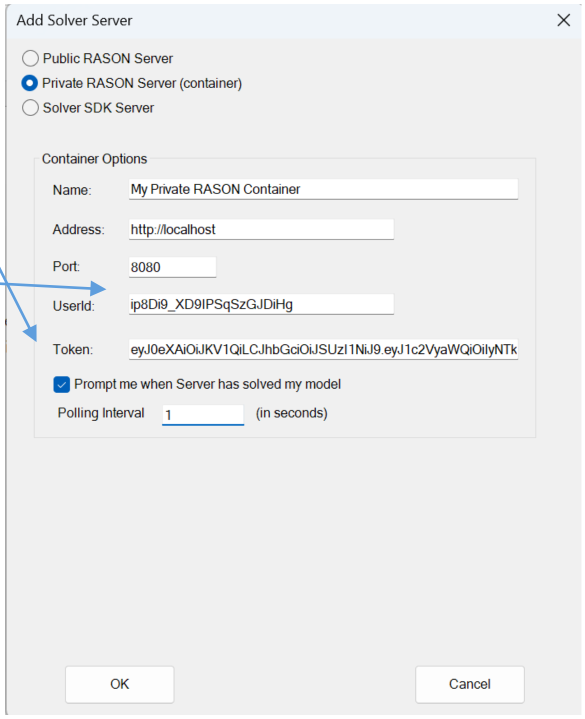
First & Last Name: Robin Locksley

Container Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJ1c2VyaWQiOiIyNTkwIiwidXNlcm5hbWUiOiJuaWNvbGVAc29sdmVyLmNvbSIsInBsYW4iOiJOb25lI

Token expires in 29 days on 7/11/2024 5:34:10 PM.

RASON User Id: xvhDopa3G-x7ZBqzugWQZA

Private RASON Container Add Solver Server dialog



○ Public RASON Server
● Private RASON Server (container)
○ Solver SDK Server

Name: My Private RASON Container

Address: http://localhost

Port: 8080

UserId: ip8Di9_XD9IPSqSzGJDiHg

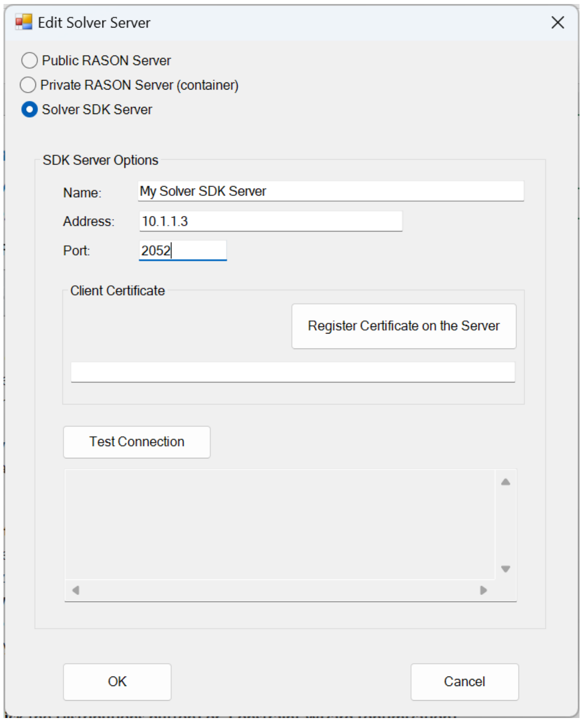
Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJ1c2VyaWQiOiIyNTk

☑ Prompt me when Server has solved my model

Polling Interval 1 (in seconds)

Solver SDK Server

Select this option to use a Solver SDK Server to solve your model. When Solver SDK Server is selected, the following options appear on the bottom half of the Add Solver Server dialog.



○ Public RASON Server
○ Private RASON Server (container)
● Solver SDK Server

SDK Server Options

Name: My Solver SDK Server
Address: 10.1.1.3
Port: 2052

Client Certificate

Name: Enter a convenient name for your server here. The name will appear as the name of this server in the Options dialog General tab display.

Address: Enter an IP address such as *10.1.1.3* or a public domain name such as *SolverServer.cloud.net* or a private domain or computer name such as *Frontline-PC\Machine1*.

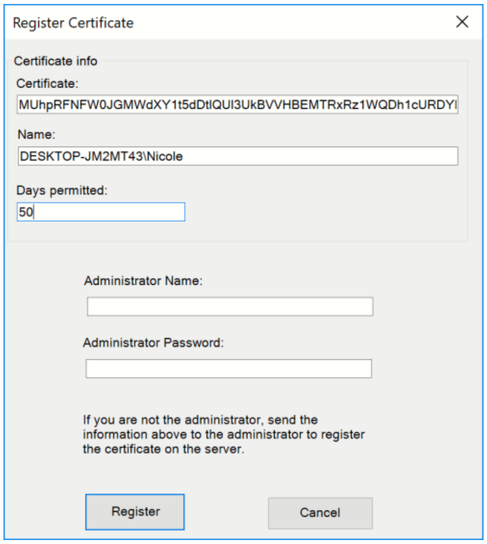
Port: To register with the server and obtain a certificate in Excel, set the client port to 2052. The TCP/IP port entered here must match the *client* port entered in the Solver Server application (SAdmin.exe). Solver Server is "listening" on this port. The default is 2050.

Note: Port 2050 is used for administrative functions, such as issuing certificates, while port 2052 is designated for client connections to solve models. These port numbers can be modified in the `SolverServer.ini` file or through the `SAdmin.exe` tool. The `SolverServer.bak` file serves as a backup in case you need to revert to previous settings. After making any changes, you must restart the server. For administrative adjustments, it is recommended to use `SAdmin.exe` to change the port number.

Client Certificate: This field holds the Solver Server Certificate. When connecting to Solver Server, the certificate will be inspected and if the same certificate is found on Solver Server, permission to solve the model will be granted. Click **Register Certificate** on the Server to open the Register Certificate dialog.

Note that if you are not the administrator, you should send the contents of your Certificate field to the server administrator, who can manually add the certificate on the server. This must be done within *24 hours* or the certificate will expire.

Solver Server includes tools for a server administrator to manage certificates for clients. But it's also possible for the administrator to manage certificates by "logging in" to the server from your Excel-based client software, using the dialog shown below.


Register Certificate

Certificate info

Certificate: MUhpRFNFW0JGMWdXY1t5dDtlQUI3UkBVVHBEMTRxRz1WQDh1cURDYl

Name: DESKTOP-JM2MT43\Nicole

Days permitted: 50

Administrator Name:

Administrator Password:

If you are not the administrator, send the information above to the administrator to register the certificate on the server.

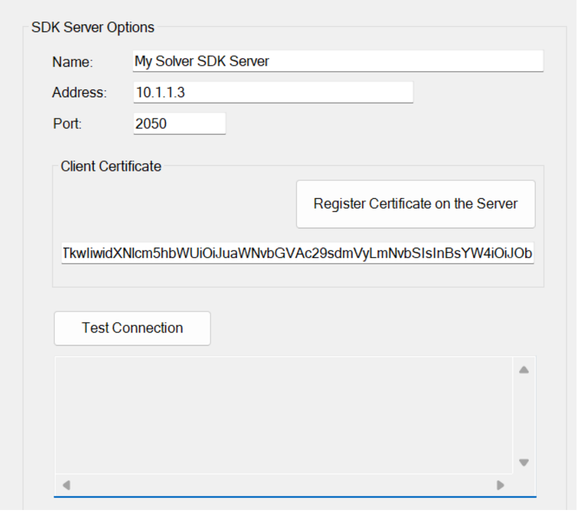
Days Permitted: An integer value, or 0 for a "permanent" certificate.

Administrator Name: Used by your server administrator.

Administrator Password: Used by your server administrator.

Click **Register** to register the Certificate.

Once back to the Add Solver Server dialog, click **Test Connection** to confirm a connection to the Server. If the connection is successful, click **Register Certificate on the Server** to display the following dialog. (If the test was not successful, please confirm that the Server Address and Port number are correct and that both machines (the client and server) are online.)


SDK Server Options

Name: My Solver SDK Server

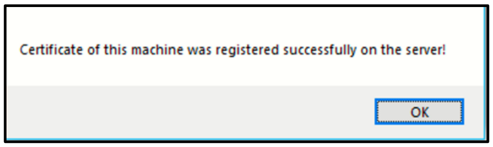
Address: 10.1.1.3

Port: 2050

Client Certificate

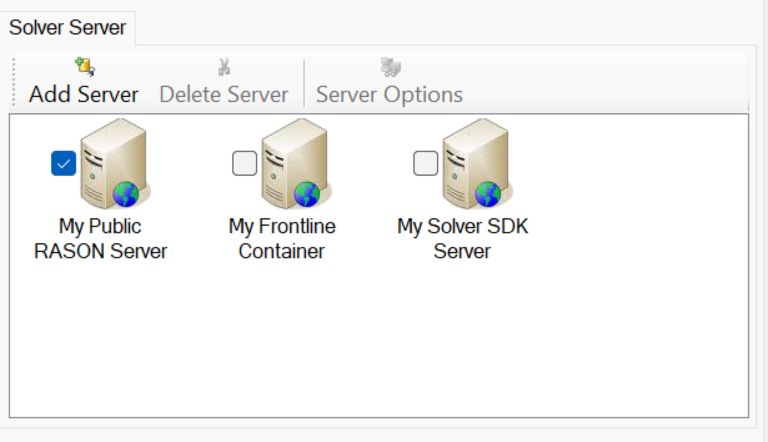
TkwliwidXNlcm5hbWUiOiJuaWNvbGVAc29sdmVyLmNvbSIsInBsYW4iOiJOb

If registration was successful, you will see the dialog below. Click *OK* to clear this dialog and *OK* again to clear the Add Solver Server dialog.


Certificate of this machine was registered successfully on the server!

Selecting your Solver Server

Regardless of the Server type selected, once **OK** is clicked on the Add Solver Server dialog, you will be returned to the General tab on the Options dialog. Your server(s) should appear as shown on the screenshot below.

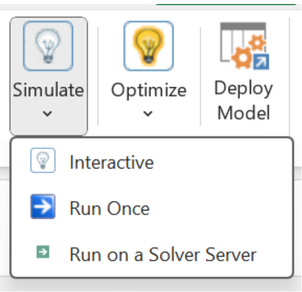


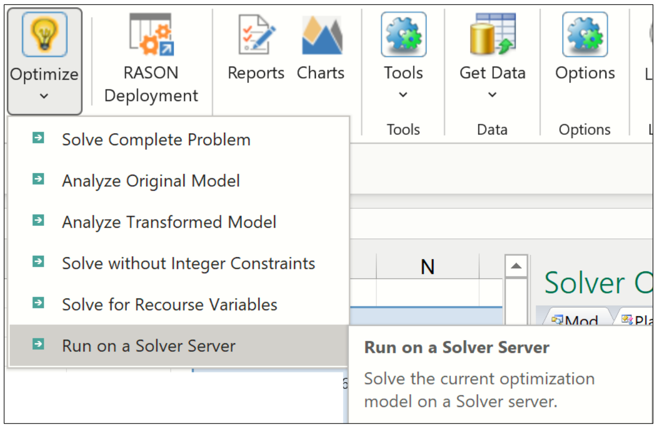
☑ My Public RASON Server
☐ My Frontline Container
☐ My Solver SDK Server

Select the checkbox to call the desired Server to solve your model. Click *OK* to close the Options dialog and return to your spreadsheet.

Solving Your Model on a Server

To solve a model on a server that you've set up previously as shown above, simply open your workbook, click the down arrow under the **Optimize** icon, and select **Run on a Solver Server** from the menu. If running a simulation, under the **Simulation** icon, click **Run on Solver Server**.

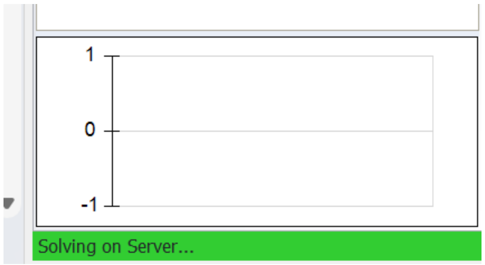




Run on a Solver Server

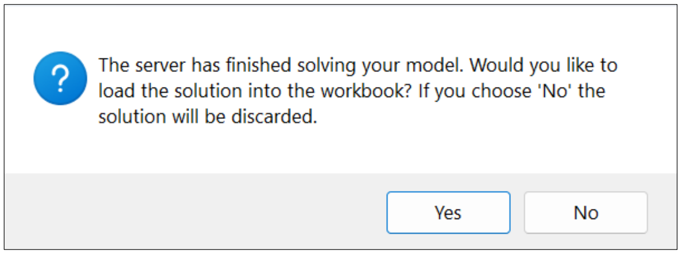
Solve the current optimization model on a Solver server.

When you first glance at the screen, you'll see the status bar indicating "Solving on Server..." as depicted in the screenshot below. This signals that the model has been dispatched to the Solver Server. While the solution process is ongoing, you're free to make adjustments to your spreadsheet, or even open a new workbook, but it's important to refrain from altering the location of the orginal model's decision variable cells or any uncertain variable/function formulas. If solving an optimization, cnce Solver completes its operation, the final variable values will be inserted into their original cell location. If you modify a formula within a constraint or uncertain function, such as changing from "Variable 1 + Variable 2" to "Variable 1 * Variable 2," please note that the spreadsheet will employ the current formulation, not the original one, when computing the constraint.



Solving on Server...

Once the solution process has been completed, a message box will appear signaling that the final solution is ready for input into the spreadsheet.



The server has finished solving your model. Would you like to load the solution into the workbook? If you choose 'No' the solution will be discarded.

Click **Yes** to proceed. The Output tab on the Solver Task Pane will be populated with the final result message and the final solution will be placed in the variable cells.

Example Output for an Optimization Model
Sending document to the server.
Downloading the results.
Solver found a solution. All constraints and optimality conditions are satisfied.

Example Output for a Simulation Model
Sending document to the server...
Simulation finished successfully.

For more information on Solver Server or if you are experiencing any problems with this service, please contact us at support@solver.com.

Analytic Solver Overview

Analytic Solver Product Line

This Guide shows you how to create and solve conventional optimization, Monte Carlo simulation, and stochastic optimization models using **Analytic Solver** – Frontline Systems' "secret weapon" for business analysts. The companion Data Science User Guide (formerly Data Science User Guide) shows you how to create and evaluate forecasting, data science and machine learning models using Analytic Solver.

Analytic Solver combines and integrates the features of Frontline's products for conventional optimization (formerly called **Premium Solver Pro** and **Premium Solver Platform**), Monte Carlo simulation and stochastic optimization (formerly **Risk Solver Pro** and **Risk Solver Platform**), and forecasting and data sciece (formerly **XLMiner Pro** and **XLMiner Platform**), in a common user interface that's available both in Excel (Desktop) and in your browser (Cloud).

Analytic Solver's **optimization** features are **fully compatible upgrades for the Solver** bundled with Microsoft Excel, which was developed by Frontline Systems for Microsoft. Your Excel Solver models and macros will work without changes. In Analytic Solver Desktop, you can use either the classical **Solver Parameters dialog**, or a newer Task Pane user interface to define optimization models.

Desktop and Cloud versions

The release of Analytic Solver V2025 Q1 includes our latest offering, Analytic Solver Cloud – a "modern Office add-in" usable in both Excel for the Web and latest versions of desktop Excel, for Windows and Macintosh. Analytic Solver Cloud handles optimization and simulation models, and forecasting and data science models, and is fully compatible with models created in previous versions of Analytic Solver. Your license for Analytic Solver will allow you to use Analytic Solver Desktop in desktop Excel or Analytic Solver Cloud in either desktop Excel or Excel for the Web. For example, a license for Analytic Solver Optimization in desktop Excel will also grant you a license for Analytic Solver Optimization in Analytic Solver Cloud. The overwhelming majority of features in Analytic Solver Desktop are also included in Analytic Solver Cloud. However, there will be some small differences between the two versions.

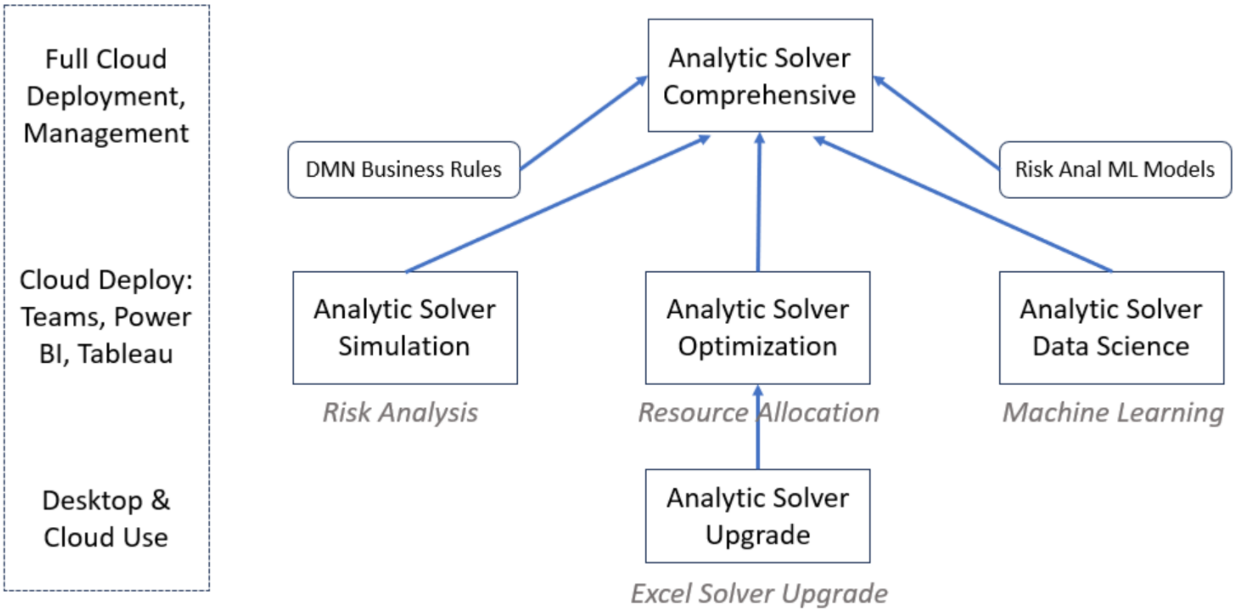
Analytic Solver for desktop or cloud may be purchased in several different ways starting with the basic version, Analytic Solver Upgrade, up to our most complete version, Analytic Solver Comprehensive. Continue reading to see which product will best meet your needs.

Analytic Solver Academy

Analytic Solver Academy allows access to Frontline Systems' Solver Academy courses: Optimization 101, Simulation 101, Data Science101 and Data Science 201. These four courses will quickly bring users up to speed on the complete functionality of Analytic Solver. If no other license exists, this product offers a basic course license with size limits for Monte Carlo simulation, data science and text mining sufficient to run all installed example models and complete all course work.

Product Licenses

Frontline Systems offers six annual licenses for our Desktop Excel and Excel Online product suite: Analytic Solver Upgrade, Analytic Solver Optimization, Analytic Solver Simulation, Analytic Solver Data Mining, and Analytic Solver Comprehensive. The graphic below illustrates how each of the subsets roll into Analytic Solver Comprehensive, our flagship product.


Full Cloud Deployment, Management
Cloud Deploy: Teams, Power BI, Tableau
Desktop & Cloud Use
Analytic Solver Comprehensive
DMN Business Rules
Risk Anal ML Models
Analytic Solver Simulation
Analytic Solver Optimization
Analytic Solver Data Science
Risk Analysis
Resource Allocation
Machine Learning
Analytic Solver Upgrade
Excel Solver Upgrade

Analytic Solver Upgrade

Frontline Systems is Frontline’s basic upgrade for the Excel Solver – enabling you to solve linear up to 10 times larger (up to 2,000 variables), quadratic (QPs) and quadratically constrained problems (QCPs) and nonlinear models 2.5 times larger (up to 500 variables). It includes faster versions of the LP/Quadratic, GRG Nonlinear, and Evolutionary Solvers), but it doesn’t support plug-in large-scale Solver Engines, decision tables, box functions, model deployment or the AI Agent. This product includes basic level support for simulation and data science models.

Analytic Solver Optimization, Simulation and Data Science

Users of the next three products are granted a license to share the results of their optimization or simulation model to Power BI, Tableau or Microsoft Teams via the Deploy Model button on the Analytic Solver tab. See the *Deploying Your Model* chapter within the Analytic Solver User Guide for more details.

Analytic Solver Optimization

Analytic Solver Optimization (formerly **Premium Solver Platform**) is Frontline's most powerful product for *conventional* optimization. It includes the PSI Interpreter, five built-in Solvers (LP/Quadratic, SOCP Barrier, GRG Nonlinear, Interval Global, and Evolutionary), solves linear, quadratic and quadratically constrained models up to 8,000 variables and nonlinear models up to 1,000 variables, and it supports plug-in large-scale Solver Engines to handle much larger models. When used with Analytic Solver Simulation, you can also solve models *with uncertainty* using simulation optimization, stochastic linear programming, and robust optimization. This product, alone, includes all the features of Analytic Solver Upgrade with limited size constraints.

Analytic Solver Simulation

Analytic Solver Simulation (***expanded*** from **Risk Solver Pro**) is Frontline's full-function product for Monte Carlo simulation and simulation optimization. It includes decision tree capabilities and the PSI Interpreter – which gives you the fastest Monte Carlo simulations available in any Excel-based products, unique interactive simulation capabilities, multiple parameterized simulations, and simulation optimization using the Evolutionary Solver. When coupled with Analytic Solver Optimization, you can also solve models with uncertainty using stochastic linear programming and robust optimization. This product, alone, includes all the features of Analytic Solver Upgrade with limited size constraints.

Together, Analytic Solver Optimization and Analytic Solver Simulation provide all the capabilities of Frontline's former product **Risk Solver Platform**.

Automated Risk Analysis of Machine Learning Models

Analytic Solver Data Science and Analytic Solver Comprehensive offer capability for **automated risk analysis** of **machine learning** models along with **synthetic data generation**, to augment the data you already have. The beauty of these new features is that **you don't have to do any work** to obtain a risk analysis of your model's performance, beyond a few mouse clicks on the **Simulation** tab to enable the analysis, which is entirely **automated**. You don't even have to be familiar with the features of Analytic Solver Simulation to use this capability for machine learning!

Analytic Solver Data Science

Analytic Solver Data Science (formerly XLMiner Platform and more recently Analytic Solver Data Mining) is Frontline's most powerful product for data mining, text mining, forecasting and predictive analytics. It includes data access and sampling, data exploration and visualization, text mining, data transformation, and feature selection capabilities; time series forecasting with ARIMA and exponential smoothing; and a wide range of data science methods for classification, prediction and affinity analysis, from multiple regression to neural networks. This product includes all the features of Analytic Solver Upgrade.

Analytic Solver Comprehensive

Analytic Solver Comprehensive (formerly **Analytic Solver Platform**) is Frontline's flagship product, combining the optimization capabilities of Analytic Solver Optimization, the simulation capabilities of Analytic Solver Simulation, and the data science capabilities of Analytic Solver Data Science. It includes the PSI Interpreter, five built-in Solvers (LP/Quadratic, SOCP Barrier, GRG Nonlinear, Interval Global, and Evolutionary) and it accepts a full range of plug-in large-scale Solver Engines. Analytic Solver Comprehensive includes support for DMN decision tables and custom functions when solving optimization, Monte Carlo simulation, simulation optimization, stochastic programming and robust optimization, and large-scale data science and forecasting models.

In addition, an Analytic Solver Comprehensive license allows the deployment of optimization and simulation model and data science workflows to RASON® – **R**estful **A**nalytic **S**olver® **O**bject **N**otation. RASON is Frontline's modeling language embedded in **JSON** and a REST API that makes it easy to create, test and deploy **decision services** powered by **analytic models** in **web** and **mobile** applications – using business rules and DMN decision tables, optimization, simulation, forecasting and machine learning. See the *Deploying Your Model* chapter for details.

Enhancements in Recent Years

Frontline Solver products have been rapidly and continually enhanced for more than two decades. Below is a brief summary of enhancements up to 2017.

- In our V10.x (2010) releases, we included faster algorithms, 64-bit versions, and new Windows HPC Server cluster computing capabilities.
- In our V11.x (2011) releases, we introduced tabu and scatter search methods in the Evolutionary Solver, Guided Mode in Premium Solver Pro, and the Distribution Wizard and Constraint Wizard.
- In our V12.0 (2012) release, we included Excel 2013 support, GPU support in the Evolutionary Solver, new stochastic decomposition, and solving on a corporate server with Solver Server.
- In V12.5 (2013) release, we introduced Analytic Solver Platform with data science capabilities, new data visualization features for both data science and Monte Carlo results, improved Guided Mode, and Support Live Chat integrated into Excel.
- In our V2014 release, we introduced a fundamental new way to build Excel-based optimization and simulation models: Dimensional Modeling, with concepts such as dimensions and cubes, and tools to build and solve larger scale, better structured, more maintainable models using these concepts.
- In our V2014-R2 release, we introduced a completely re-engineered, far more powerful data science and forecasting capability named XLMiner Platform. New data science algorithms were up to 100 times faster, constantly exploit multiple processor cores, and offer greater accuracy and numeric stability.

- Our V2015 release introduced a wide range of new features, including powerful text mining, ensemble methods for classification and prediction, feature selection, partitioning “on-the-fly,” ROC/RROC curves, and enhanced linear and logistic regression; extensive chart enhancements, distribution fitting, and new Six Sigma functions in Monte Carlo simulation; and support for “publishing” optimization and simulation models to Excel Online and Google Sheets.
- Our V2015-R2 release made it easy to share analytic model results in popular Business Intelligence software, including Microsoft Power BI and Tableau, both popular interactive data visualization tools, and it linked your Excel workbook with Big Data in compute clusters running Apache Spark.
- Our V2016 release introduced a new **Create App** feature that translates your Excel optimization or simulation model into Frontline’s new **RASON** modeling language – radically simplifying the path to create an application that can run in a **web browser**, or a **mobile app** for phones or tablets. New SQP-GS and Feasibility Pump methods greatly improved the performance of the Evolutionary Solver on challenging non-smooth models.
- Our V2016-R2 release added support for **compound distributions** and **correlation using copulas** (Gaussian, Student and Archimedean forms), in Monte Carlo simulation; new “GA methods’ for integer variables in the **Evolutionary Solver**, speed and memory improvements in **Dimensional Modeling**, and support for the **Web Data Connector** in Tableau 9.1.

2017: Power BI, Analytic Solver Basic, and More

In Frontline Solvers V2017, we introduced users to **AnalyticSolver.com**, a new (at the time) cloud-based platform for both predictive and prescriptive analytics models that you could use via a **web browser** – including all the optimization, simulation, and data science power found in the desktop version. The AnalyticSolver.com **user interface** worked just like our Excel user interface, with a Ribbon and Task Pane. In 2020, AnalyticSolver.com was superseded by Excel for the Web and our Analytic Solver Cloud version.

V2017 also uses a **new licensing system** that offers you more flexible ways to use the software, both desktop and cloud. Your license is associated with **you**, and may be used on **more than one PC**. For example, you can install the software on your office PC, your company laptop, and your PC at home. But only **you** can use Analytic Solver, and only on **one** of these computers **at a time**.

V2017 introduced **Analytic Solver Basic**, as described above, to give you access to **all** Analytic Solver features, **all** the time, for learning purposes using small models. It also includes a new **License/Subscription Manager** and a **Product Selection Wizard** that makes it much easier to upgrade or change your license subscription on a self-service basis, and a new **Test Run/Summary** feature that lets you see exactly how your model will run with an Analytic Solver upgrade, even a plug-in large-scale Solver Engine, before you purchase the upgrade – and do this any time, not just during a 15-day free trial.

Feature Enhancements in V2017

V2017 introduced major enhancements to **data mining**: Automatic support for **categorical variables** in many classification and prediction algorithms that ‘normally’ require continuous variables; **ensembles** that combine nearly any type of algorithm as a ‘weak learner’, not just (for example) classification and regression trees; general-purpose **Rescaling** as a new Data Transformation method that can also be applied 'on-the-fly' when training a model; greatly enhanced multilayer neural networks; ability to export models in PMML; and many report and chart enhancements.

The V2017 **Evolutionary Solver** includes another set of major enhancements in its handling of non-smooth models with integer variables – enough so that *most* such models will solve *significantly* faster.

And there's support for the Tableau Web Data Connector 2.0, and a new SolverSetup program that *automatically* installs the correct **32-bit or 64-bit** version of the software.

Feature Enhancements in V2017-R2

V2017-R2 introduced major enhancements to Monte Carlo simulation/risk analysis and optimization. It's now possible to **fit copula** parameters to historical data – a complement to distribution fitting that is sometimes called "correlation fitting." You can use a new family of probability distributions, called the **Metalog distributions**, even more general than the Pearson distributions – members of the family can be chosen based directly on historical data (even just a few observations), without a distribution fitting process.

The V2017-R2 PSI Interpreter includes major **speed enhancements** for large linear and nonlinear **optimization models** – users with large models are likely to see a dramatic speedup in "Setting Up Problem…" Also part of this release were new, higher performance versions of the Gurobi Solver Engine (based on Gurobi 7.5), the Xpress Solver Engine (based on Xpress 30.1), and the Knitro Solver Engine (based on Knitro 10.3).

Creating Power BI Custom Visuals

The most exciting new feature of V2017-R2 is the ability to turn your Excel-based optimization or simulation model into a **Microsoft Power BI Custom Visual**, with just a few mouse clicks! Where others must learn JavaScript (or TypeScript) programming and a whole set of Web development tools to even begin to create a Custom Visual, you'll be able to create one right away.

You simply select rows or columns of data to serve as changeable parameters, then choose **Create App – Power BI**, and save the file created by V2017-R2. You click the Load Custom Visual icon in Power BI, and select the file you just saved. What you get isn't just a chart – it's your *full optimization or simulation model*, ready to accept Power BI data**, run on demand** on the web, and display visual results in Power BI! You simply need to drag and drop appropriate Power BI datasets into the "well" of inputs to match your model parameters.

How does *that* work? The secret is that V2017-R2 translates your Excel model into **RASON®** (RESTful Analytic Solver Object Notation, embedded in JSON), then "wraps" a JavaScript-based Custom Visual around the RASON model. See the chapter "Creating Your Own Application" for full details!

2018: Tableau, Data Science Workflows, and More

Analytic Solver V2018 extended our forecasting and data science features with a new capability called **data mining workflows** that can save a lot of time and eliminate repetitive steps. You can combine nearly any of Analytic Solver's data retrieval, data transformation, forecasting and data science methods into a single, all-inclusive workflow, or pipeline.

Using the new Workflow tab in the Task Pane, you can either **"drag and drop"** icons onto a "canvas" to create a workflow diagram, or you can simply turn on a **workflow recorder**, carry out the steps as you've always done by choosing menu options and dialog selections, and the workflow diagram will be created automatically. Once the diagram or pipeline is created, you can "run" it in one step – each data science method in the workflow will be executed in sequence.

In previous releases, you could use the trained model from a *single* data science method (such as a Classification Tree or Neural Network) to "score" new data, by mapping features (columns) between the training set and new data set. In V2018, you can apply an *entire workflow* – including data transformations, partitioning, model training, and more – to a new dataset, by mapping features (columns) between the dataset used to create the workflow, and a new dataset.

Creating Tableau Dashboard Extensions

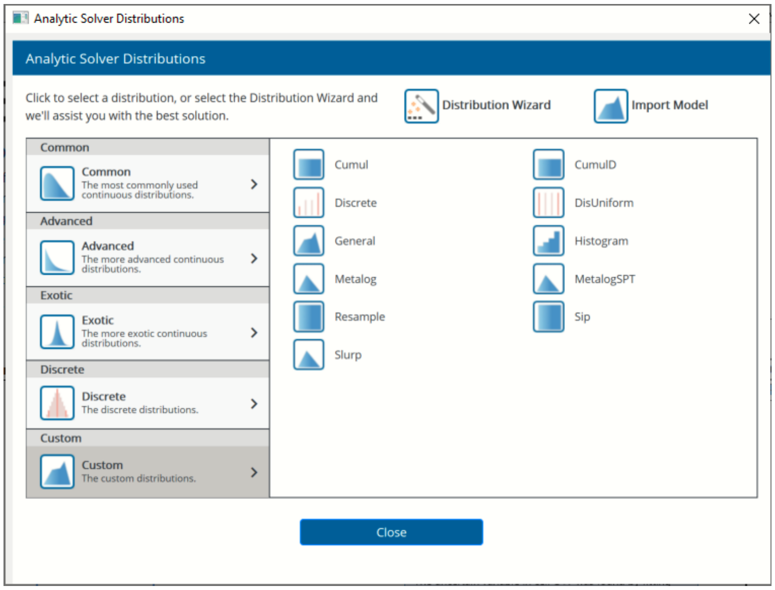
Another exciting new feature of V2018 is the ability to turn your Excel-based optimization or simulation model into a **Tableau Dashboard Extension**, with just a few mouse clicks! This is quite similar to the ability to create Power BI Custom Visuals introduced in Analytic Solver V2017-R2. It works (only) with Tableau version 2018.2 or later.

You simply select rows or columns of data to serve as changeable parameters, then choose **Create App – Tableau**, and save the file created by V2018. In Tableau, drag the **Extensions** object onto your dashboard, and choose the file you just saved. You'll be prompted to match the parameters your model needs with data in Tableau. What you get isn't just a chart – it's your *full optimization or simulation model*, ready to accept Tableau data**, run on demand** (using our **RASON** server), and display visual results in Tableau!

2019: Analytic Solver Cloud

In Frontline Solvers V2019, we introduced **Analytic Solver Cloud** – a next-generation product that's the result of **five years** of development using new cloud technologies, that we can now bring to you – since Microsoft has released a complete set of JavaScript APIs for new Excel features, such as functions like PsiNormal() and PsiMean() used in simulation and risk analysis.

In our first V2019 release, we focused on a **consistent user experience** between Analytic Solver Desktop and Analytic Solver Cloud. In a few cases, this has involved modifications to Analytic Solver Desktop. For example, older versions of Analytic Solver Desktop used "cascading submenus" to select probability distributions, and results in Monte Carlo simulations. Since an Office add-in cannot define or use cascading submenus, we have modified Analytic Solver Desktop *and* Analytic Solver Cloud so that the **Distributions** button on the Ribbon displays a dialog, rather than a cascading submenu, where you can select an appropriate probability distribution. Yet the order and layout of probability distributions remains the *same* as in previous Analytic Solver versions.


Analytic Solver Distributions
Analytic Solver Distributions
Distribution Wizard
Import Model
Common
Common
The most commonly used continuous distributions.
Advanced
Advanced
The more advanced continuous distributions.
Exotic
Exotic
The more exotic continuous distributions.
Discrete
Discrete
The discrete distributions.
Custom
Custom
The custom distributions.
Cumul
CumulD
Discrete
DisUniform
General
Histogram
Metalog
MetalogSPT
Resample
Sip
Slurp
Close

Click to select a distribution, or select the Distribution Wizard and we'll assist you with the best solution.

Business Rules and DMN Decision Tables

In Frontline Solvers V2019-R2, we introduced another completely new capability to Analytic Solver: Making it easy to capture rule-based decisions in the form of decision tables, with multiple columns for inputs, and multiple rows that express business rules. See the new chapter “Using Decision Tables”.

While **Business Rule Management Systems** (BRMS) have been around for 15 years, with the increasing use of predictive and prescriptive analytics (like those found in Analytic Solver), these systems have begun to give way to **Decision Management Systems** that support a mix of rules and analytics. Where older BRMS systems used proprietary rules languages, the new trend embraces an open standard called **DMN** (Decision Model and Notation), with a language called **FEEL** (Friendly Enough Expression Language) for business rules.

Analytic Solver V2019-R2 and V2020 make it easy for you to create and test DMN 1.2-compatible **decision tables with business rules** expressed in FEEL. And it’s equally easy to use Analytic Solver’s forecasting, machine learning, simulation and optimization capabilities to create analytic **inputs** for decision tables, or use to decision table **results** in simulation and optimization models.

2020: Simulation and Solver Engine Enhancements

With each new release, Analytic Solver gives you more! Our January 2020 enhancements apply to both Analytic Solver Desktop and Cloud versions.

Faster Solver Engines, Using Multiple Cores

In V2020, the LP/Quadratic Solver – probably the most-used Solver Engine in Analytic Solver – features significantly improved performance on linear mixed-integer models. Prior versions of this Solver would use only one processor core at a time, but V2020 will use *multiple processor cores* to speed your solution.

The plug-in large-scale Solver Engines in Analytic Solver V2020 also feature significantly improved performance (they continue to utilize multiple processor cores). These include the **Gurobi** Solver V9.0, with a new ability to solve non-convex quadratic models; the **Xpress** Solver V35, with a new Solution Refiner, and the Artelys **Knitro** Solver V12.1, with SOCP and MIP speedups.

Monte Carlo Simulation Enhancements

Analytic Solver V2020 includes 12 new probability distribution functions, *enhanced* property functions for the PSI Distribution functions, *new* property functions for the PSI Statistics functions, and *new* "theoretical" functions that return analytic moments of distributions. Full details are in the *Frontline Solvers Reference Guide*, but here's a partial list of new/enhanced functions:

PsiBurr
PsiDagum
PsiDblTriang
PsiFatigue
PsiFdist
PsiFrechet
PsiHypSecant
PsiJohnsonSB
PsiJohnsonUB
PsiKumaraswamy
PsiReciprocal
PsiLevy
PsiLevyAlt
PsiHypSecantAlt
PsiCumulD
PsiLaplace
PsiCauchy
PsiTruncate
PsiCensor
PsiLock
PsiOutput
PsiTheoMean
PsiTheoMedian
PsiTheoMode
PsiTheoMin
PsiTheoMax
PsiTheoVariance
PsiTheoStdDev
PsiTheoSkewness
PsiTheoKurtosis
PsiTheoRange
PsiTheoPercentile
PsiTheoPercentileD
PsiTheoTarget
PsiTheoTargetD
PsiCategory

These functions make it **even easier** to adapt risk analysis models developed with other popular Excel add-ins, such as Palisade's @RISK, to **work with Analytic Solver**. A simple Find and Replace of the function name prefix with 'Psi' is often all you need. And unlike those other Excel add-ins, with Analytic Solver you can easily run your model in the cloud with **Excel for the Web**, translate your model to RASON, and use it in **Power BI**, **Tableau**, or your own **web or mobile** application!

What's New in Analytic Solver V2020.5

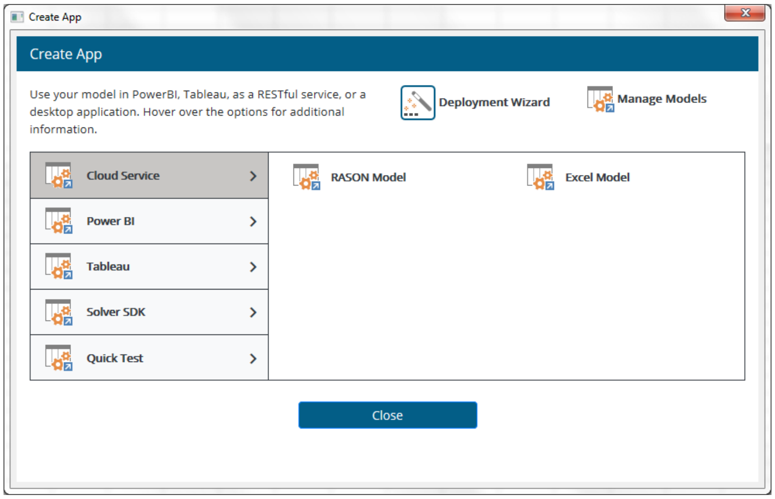
Analytic Solver V2020.5 includes significant enhancements to both Monte Carlo simulation and optimization – but the most exciting new feature is a greatly expanded **Create App** facility that makes it easy to **deploy** your Excel analytic model as a **cloud service** (thanks to RASON), usable from nearly any corporate, web or mobile application. What's more, you can manage, monitor and update your own cloud services, without ever leaving Excel!

Easily Deploy Your Model as a Cloud Service

We've realized for many years that developing and testing your analytic model in Excel is often just the first step: To gain the **real business value** from the decisions that can be made with your model, it's often necessary to get the model into the hands of **other people in the business** – in a form where they can easily ensure that it has **up-to-date data, re-run** the model's optimization, simulation, or data science process, and either **view** the results, or **plug them int**o another software application or process.

In our 2017 and 2018 Analytic Solver releases, we took the steps that were possible at that time, enabling users to get their models into the hands of **Power BI** and **Tableau** users. And we built a facility to ***translate*** simpler Excel models into the RASON modeling language, enabling them to be solved in our cloud platform (RASON is an acronym for RESTful Analytic Solver Object Notation). But up to this point, a typical Excel user would still need help from a web developer, or would need learn JavaScript and other web development skills, to make truly effective use of this facility.

Now in our V2020.5 release of both Analytic Solver and RASON, we've gone much further to **simplify** the process of **deploying** an Excel model as a cloud service, and **connecting** it to databases and cloud data sources. The RASON cloud service will now accept and run **Excel workbook** models "on a par" with models written in the RASON modeling language. With the **Create App** menu option, you can turn your Excel model into a cloud service in seconds.



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

As an Analytic Solver user, you can now create and test models, deploy them "to the cloud" – **point and click** – as full-fledged RESTful decision services, and even get reports of recent runs of your decision services, all without leaving Excel. Using our web portal at https://rason.com, you can go further –even embed your Excel workbook in a multi-stage "decision flow" that can combine SQL, RASON, Excel, and DMN models, passing results from stage to stage.

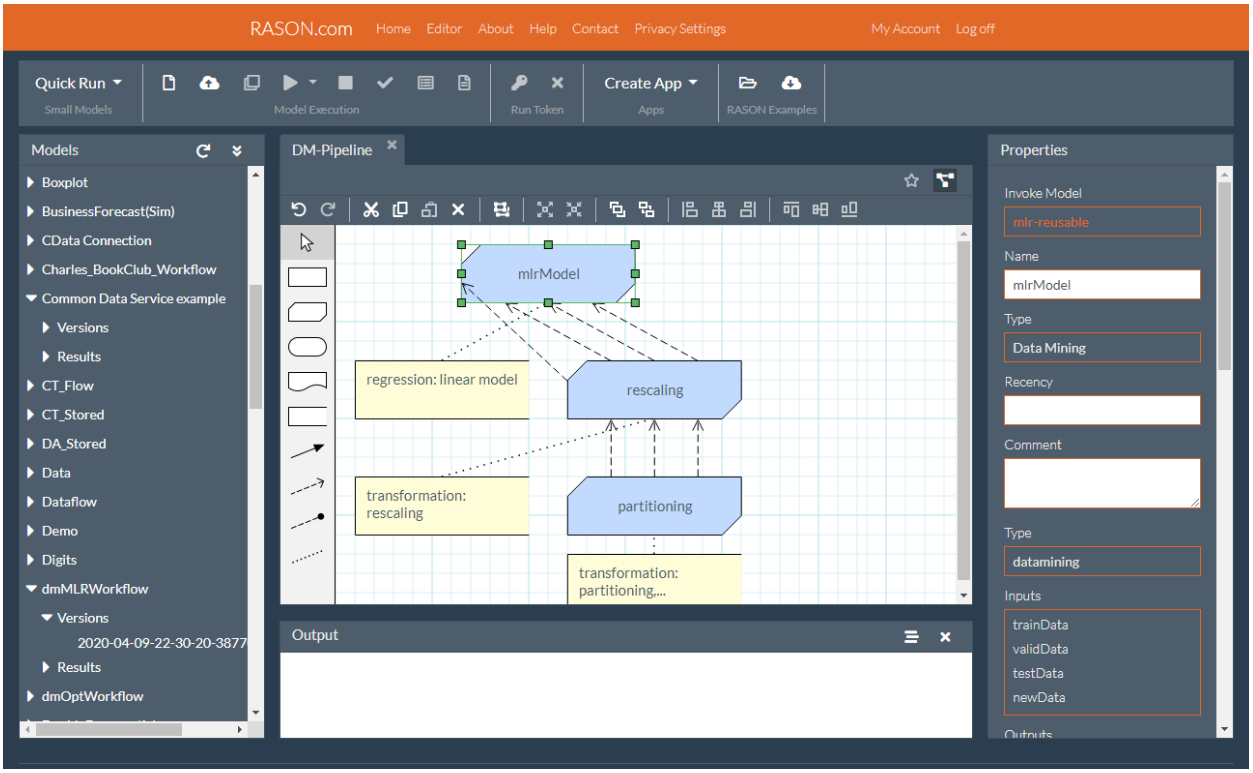
We start you out with a **RASON "basic" license**, so you can try out these new capabilities without purchasing anything else! (Of course, you may need an upgraded RASON license to deploy your model to many users, and re-solve it hundreds or thousands of times on our cloud servers.)

More About RASON Decision Services

RASON is an Azure-hosted cloud service that enables your company to easily embed 'intelligent decisions' in a custom application, manual or automated business process, applying the full range of analytics methods – from simple calculations and business rules to data science and machine learning, simulation and risk analysis, and conventional and risk-based optimization.

RASON Decision Services can be used from nearly **any application**, via a series of simple REST API requests to https://rason.net. To express the full range of analytic models, RASON includes a high-level, declarative modeling language, syntactically embedded in JSON (JavaScript Object Notation), the popular structured format almost universally used in web and mobile apps. RASON results appear in JSON, or as more structured OData JSON endpoints.

RASON Decision Services also includes comprehensive data access support for **Excel, SQL Server on Azure, Power BI, Power Apps, Power Automate** (aka Microsoft Flow) and **Dynamics 365**. And it includes powerful model management tools, such as tracking model versions including “champions and challengers”, monitoring model results, and automated scheduling of runs for both models and multi-stage decision flows.



Invoke Model: mlr-reusable
Name: mlrModel
Type: Data Mining
Recency:
Comment:
Type: datamining
Inputs: trainData, validData, testData, newData
Outputs

How You Can Use RASON

You can use RASON to quickly and easily create and solve optimization, simulation/risk analysis, data science, decision table, and decision flow models – instantly deployed as cloud services. You can learn RASON, create models, supply data and solve them, and even manage model versions and cloud data connections, “point and click” using https://rason.com, our “web portal” to the underlying REST API service.

If you’ve used another **modeling language** to build an analytic model, you’ll find the RASON language to be simple but powerful and expressive – and integrating RASON models into a larger application, especially a web or mobile app, is **much easier** than with other modeling languages. Excel users will find that RASON includes virtually the entire **Excel formula language** as a subset. If you’ve used tools based on the DMN (Decision Model and Notation) standard, you’ll find that RASON – and Analytic Solver, as shown in the chapter “Building Decision Tables” – fully support DMN and FEEL Level 2.

Unlike existing “heavyweight” Business Rule Management Systems, with year-long implementation schedules, six-figure budgets and limited analytics power, RASON Decision Services enables you to **get results in just weeks to months**, from building and testing models, to deploying them across an organization.

With RASON, you can build successful POCs (Proofs of Concept) without any IT or professional developer support – yet RASON is very “IT and developer friendly” when you’re ready to deploy your POC across your company.

New Time Series Simulation Functions

Analytic Solver V2020.5 includes another new set of PSI Distribution functions and related PSI property functions, focused around **time series simulation**. Earlier Analytic Solver versions supported time series simulation using functions such as PsiForecast() and PsiPredict(), and models fitted via Analytic Solver Data Mining – but V2020.5 goes further, to support time series functions found in other popular Excel add-ins, such as Palisade’s @RISK. Full details are in the *Frontline Solvers Reference Guide*, but here’s a partial list of new/enhanced functions:

PsiAR1
PsiAR2
PsiMA1
PsiMA2
PsiARMA11
PsiBMMR
PsiGBMJD
PsiARCH1
PsiGARCH11
PsiEGARCH11
PsiAPARCH11
PsiTSTransform
PsiTSIntegrate
PsiTSSeasonality
PsiTSSync

With these functions, virtually any risk analysis model developed with other popular Excel add-ins, such as Palisade’s @RISK, can be easily made to work with Analytic Solver. An appendix in the *Frontline Solvers Reference Guide,* “@Risk to Analytic Solver Psi Function Conversion Table”, explains the details. And with Analytic Solver, you can easily **deploy** your risk analysis model as a **cloud service** – usable from Tableau, Power BI, Power Apps, Power Automate, or virtually any corporate, web or mobile application!

New Optimization Result Functions

In every version of Analytic Solver (and its precedessors, such as Premium Solver and the Excel Solver), you could obtain all the properties of an optimal solution – such as initial and final values, dual values, and ranges for decision variables and constraints – via the Answer Report and Sensitivity Report, which are inserted into your Excel workbook as new worksheets. But what if you want only ***select subsets*** of these values – and you’d like to have them on the **same workshee**t as your model? That’s now possible in Analytic Solver V2020.5. Just type these new functions into cells, or use the Function Wizard in Excel.

PsiInitialValue
PsiFinalValue
PsiSlackValue
PsiDualValue
PsiDualLower
PsiDualUpper
PsiCalcValue
PsiOptStatus
PsiModelDesc

These new functions have another purpose in V2020.5, when you use Analytic Solver’s enhanced **Create App** facility to deploy your model as a cloud service: You can use them to determine a select subset of values from the solution that you want to ***return from your cloud service*** to a calling application.

What's New in Analytic Solver V2021

Analytic Solver V2021 features faster startup and better support for 4K monitors in Analytic Solver Desktop, improved support for decision trees "on the spreadsheet" in Analytic Solver Cloud, and a new, faster version of the Xpress Solver (V37.1.3) – but the most exciting new features in this release give you new ways to define your own **custom functions**, in a way that works in both Excel Desktop and Excel for the Web, as well as our cloud platform RASON.

Lambda, Let and Box Functions

In years past, Excel-savvy analysts used VBA (Visual Basic for Applications) to define their own custom functions. While this still works in Excel Desktop using COM (28-year-old Component Object Model), VBA functions are *not* supported in Excel for the Web – and according to Microsoft, VBA and COM will *never* move to the cloud. If you want your custom functions to work in *both* desktop and cloud, your options have been limited – until now:

1) Microsoft has introduced new Excel functions LAMBDA and LET. These are very special because you can use them in Excel formulas to define your own custom functions. The Excel community has expressed much excitement over these new functions, since they effectively make Excel a “complete programming language”. (In Q1 2021, these functions are being rolled out across the different Office update channels.)

- On another front, there’s the open standard known as DMN (Decision Model and Notation) – a business user-friendly “formula language” used to define business rules and decision tables, supported in “decision management” platforms from various vendors, and in Analytic Solver and RASON since 2019. DMN – now in version 1.3 – offers a way to define your own custom functions, known as “Box functions”.

Analytic Solver V2021 includes support for *both* Excel’s LAMBDA and LET functions, and for DMN-compatible Box functions. You’ll find a new chapter in this User Guide, “Using Custom Functions”, that explains how to use both approaches. Even better, LAMBDA and LET, and DMN Box functions enjoy full support from our PSI Interpreter – which means that our full range of Solver Engines, and our high-speed Monte Carlo simulation engine “understand” and take full advantage of custom functions that you define this way. This can yield better results than you’ve ever had with VBA-based functions that are embedded in an optimization or simulation model.

What's New in Analytic Solver V2021.5

Automate Data Mining with Find Best Model

Analytic Solver, our SDKs and RASON include comprehensive, powerful support for data mining and machine learning. Using these tools, you can “train” or fit your data to a wide range of statistical and machine learning models: Classification and regression trees, neural networks, linear and logistic regression, discriminant analysis, naïve Bayes, k-nearest neighbors and more. But the task of choosing and comparing these models, and selecting parameters for each one was up to you.

With the new Find Best Model options in V2021.5, you can automate this work as well! Find Best Model uses methods similar to those in (expensive high-end) tools like DataRobot and RapidMiner, to automatically choose types of ML models and their parameters, validate and compare them according to criteria that you choose, and deliver the model that best fits your data.

Better Simulation Models with Metalog Distributions and Fitting

Analytic Solver, our SDKs and RASON support over 60 “classical” probability distributions for Monte Carlo simulation. Since mid-2017, they’ve also supported the increasingly popular Metalog family of distributions, created by Dr. Tom Keelin, and recently popularized by the nonprofit Probability Management group. Metalog distributions can closely approximate virtually any classical continuous distribution, and often they can better fit user data than classical distributions. In V2021.5 we’ve brought Metalog distributions to the fore, with a powerful new facility to automatically fit user data to the full range of possible (bounded and unbounded, multi-term) Metalog distributions. It’s never been easier to get an accurate probability distribution that fits a real-world phenomenon.

Share Data Mining and Probability Models via RASON

In recent years, our Azure-hosted RASON Decision Services platform has offered increasingly powerful facilities to deploy models to the cloud and share them with other users – culminating in Frontline Solvers V2020.5, when we enabled deployment, sharing, versioning and management of Excel models as well as native RASON models, plus support for multi-stage decision flows, encompassing and going beyond traditional data science workflows.

In V2021.5 we’ve gone further: You can now deploy and share data mining and machine learning models, trained in Analytic Solver or RASON, to the Azure cloud, and use them directly for classification and prediction (without needing auxiliary “code” in R or Python, RASON or Excel). You can also deploy and share probability models, following the open Probability Management 3.0 standard. Using these Shared Information Probability resources (SIPs, also known as “Stochastic Information Packets”), you can ensure that your group or organization uses consistent data about uncertain/risky variables across simulation or decision models, enabling model results to be meaningfully compared.

And of Course, Optimization Enhancements

Analytic Solver, our SDKs and RASON have always offered rich support for conventional and stochastic optimization, improved in every new release. V2021.5 is no exception: We’ve made PSI Interpreter enhancements to better utilize main memory in large optimization models (with 1 million or more decision variables) in all three product lines. In our V2021.5 release, we’re also shipping the latest Gurobi Solver 9.1, Xpress Solver 37.1.3, and new KNITRO Solver V12.4 with each of our products.

Analytic Solver Academy – Replaces Analytic Solver Basic

We’ve found that many purchasers of Analytic Solver Basic, who lacked a previous analytics background and didn’t also purchase learning aids such as our Solver.Academy courses, experience limited success and often don’t renew their licenses. In V2021.5 we are replacing Analytic Solver Basic with a bundle of all four Solver.Academy courses plus a license for Analytic Solver (basic limits only), if no other Solver license has been purchased.

What's New in Analytic Solver V2022

Faster Interaction, Faster Solves

In Analytic Solver V2022, Excel startup is **faster**, and your ‘regular interaction’ with Excel is **faster** when Analytic Solver Desktop is loaded. Dialog rendering is improved for users with very-high-resolution monitors, as well as monitors with limited vertical depth.

Most impactful, the process of interpreting the model – when the model type is diagnosed and when “Setting Up Problem…” appears in the Task Pane status bar – is **faster** for most models, and *significantly* **faster** for larger models, especially on models with very deep “chains” of formulas that depend on other formulas. We’re also including new, **faster** versions of the Gurobi Solver for linear mixed-integer models, and the Knitro Solver for nonlinear models.

V2022 re-introduces the **Freeze** and **Thaw** options (now located on the Task Pane Tools tab), which allow you to share Analytic Solver models containing PSI function calls with users who don’t have Analytic Solver installed. (“Freeze” will save PSI function call formulas in cell comments, and “Thaw” will restore them later as formulas.)

Deploy Model functionality in V2022 is enhanced on our RASON cloud server, and **Microsoft Teams** messaging is improved for several account types.

In case you didn’t know: Recent Analytic Solver releases, including V2022, allow you to **Test Run** models that *exceed* the limits of your current license. (You don’t get full results for all variables and constraints, but you do get the final objective value and solution time.) You can even use optional large-scale Solver Engines, like the Gurobi and Knitro Solvers, in a “Test Run”. To make it clear what your current license does and doesn’t include, in V2022 you’ll see “(Test Run)” next to the names of optional Solver Engines in the Task Pane.

What’s New in Analytic Solver V2023

Automated Risk Analysis of Machine Learning Models

Besides our “usual enhancements” such as faster Solver Engines and new PSI functions like PsiCalcParam(), Analytic Solver V2023, our latest release, features an innovative (and patent pending) new capability for **automated risk analysis** of **machine learning** models on the Data Mining tab.

A further benefit of this new feature is a general-purpose, easy to use facility for **synthetic data generation**, to augment the data you already have. Analytic Solver Data Mining and Comprehensive users are able to use these new features with size limits constrained only by memory, but *all* Analytic Solver users have access to these features with “Basic size limits” for your datasets.

Until Analytic Solver V2023, data science and machine learning (DSML) tools – including ours – had no facility for **risk analysis** of machine learning (ML) models, prior to their production use. Most tools (including ours) had facilities for ‘training’ the model on one set of data, ‘validating’ its performance on another set of data, and ‘testing’ it versus other ML models on a third set of data. But this is not *risk analysis*.

Synthetic Data Generation 'For Free'

Synthetic data generation has come into use in recent years to augment available datasets, when the available data is limited, or is restricted by law or regulation, such as with personal health information (PHI). In Analytic Solver V2023, you have a powerful, general-purpose tool for synthetic data generation, using the new **Generate Data** button on the Data Mining Ribbon tab. So far, we're keeping Analytic Solver current with the "state of the art".

But with the new **Simulation** tab that you'll find in every ML model training dialog in Analytic Solver, we've gone *beyond* the "state of the art", to bring you a new way to assess your trained ML model's performance – not to determine how well it **has performed** on data you have, but to quantify how it **may perform differently** on data it will encounter in the future. We generate synthetic data "on the fly" and use it in a Monte Carlo simulation of your ML model's performance – and we highlight **differences** in model performance in training versus simulated production use.

The beauty of this approach is that **you don't have to do any work** to obtain a risk analysis of your model's performance, beyond a few mouse clicks on the **Simulation** tab to enable the analysis, which is entirely **automated**. You don't even have to be familiar with the features of Analytic Solver Simulation to use this capability for machine learning!

(You *can* adjust several settings that involve distribution fitting, correlation and copulas, Monte Carlo sample generation, etc. – but the default settings work very well.) Typically, the analysis adds only **seconds** to a perhaps a **minute** to the time taken to train and validate your ML model. So you can make this a routine **part of the process** of training and assessing new ML models.

As noted earlier (opposite the title page) in this User Guide, we have a patent application pending titled "Automated Risk Analysis of Machine Learning Models". But as an Analytic Solver (or Solver SDK or RASON) user, you gain first access to this new capability at no extra cost.

What's New in Analytic Solver V2023 Q1

We released Analytic Solver V2023 in September 2022, with the innovative enhancements described above – but we surprised even ourselves by how much more we could offer our customers by December 2022! So we've christened this new release "Analytic Solver V2023 Q1".

Faster LP/Quadratic Solver and Large-Scale LP/QP Solver Engine

Our "headline feature" for this release is a new, higher performance version of the built-in LP/Quadratic Solver, as well as its "cousin" the Large-Scale LP/QP Solver Engine which has much higher problem size limits. We expect most users will see **faster – sometimes *much* faster** solutions with this new version, both for LP and QP (linear programming and quadratic programming) models, and for LP and QP models with integer constraints.

And there's more: Now you can solve models with (convex) **quadratic constraints**, with or without a quadratic objective, using this Solver Engine.

(There are always tradeoffs: We've found that for about 10% of models we've tested, the new version is not faster, and can even be *slower* for models with integer constraints. If you encounter this, just select the **Classic Search** option to get exactly the same performance you were getting before.)

More Plug-in Solver Engine Improvements

V2023 Q1 also includes new versions of other large-scale Solver Engines. The **Gurobi Solver Engine** is upgraded (to their V10.0) with a range of enhancements, yielding solution times faster by 3-10% to 25% on a wide range of linear and quadratic mixed-integer models. The **Xpress Solver Engine** is upgraded (to their V9.0) with enhancements to strong branching, separation of cutting planes, and a new heuristic method, run at the branch & bound root. The **KNITRO Solver Engin**e is upgraded (to their V13.2) with a range of enhancements for smooth nonlinear mixed-integer models, including new presolve, cut selection and heuristic methods.

Greatly Improved "Deploy Your Model to Teams" Capability

We learned from surveys that a large majority of our customers work in companies using **Microsoft Teams –** so we've significantly enhanced a feature introduced in Analytic Solver V2022, that makes it easy to **share** your Excel model results with colleagues in your company, using Teams. The V2022 feature was designed to share your (entire) model, but we realize that many users want or need to share just the **model results** – not the full model with all its optimization and/or simulation model elements.

In V2023 Q1, when you choose **Deploy Model** from the Ribbon and click **Teams – Teams Report**, Analytic Solver will automatically create a new workbook holding only **model results**, with external links to your **model** workbook. You can choose exactly which optimization and/or simulation results you want to include in this workbook. The new workbook will be saved online, and made available to the users you want, through a "Teams channel" that you select. Your colleagues, using **just Teams**, will be able to open the workbook and view, copy or work with the results you're providing. And perhaps the best part: When you **re-run** your optimization or simulation model with new data, the workbook in Teams will be **automatically updated** (via those external links) with the latest model results! See the chapter "Deploying Your Model" in this Guide for full details.

Risk Analysis of Machine Learning Models Created in Other Software

Analytic Solver V2023 introduced an innovative (and patent pending) new capability for **automated risk analysis** of **machine learning** models – see "What's New in Analytic Solver V2023" above for a complete summary, and see our Data Mining User Guide for full details. But we realize that many people create and test machine learning models using other software. Those folks just don't have the ability to quantify how their ML models **may perform differently** on data they will encounter in the future … until now. But **you can help them**, using Analytic Solver V2023 Q1 (or using RASON V2023 Q1, if they prefer to use a cloud platform).

In V2023 Q1, we've made it easy to perform **automated risk analysis** of models **created in other software** and saved in **PMML** (Predictive Modeling Markup Language) format. PMML is an open standard that is widely supported by software for machine learning. Analytic Solver and RASON will also save a trained machine learning model in PMML form. But now you can bring a PMML model into Analytic Solver, plus some of the data you used to train the model (just copy the PMML text and the data onto worksheets) – then use simple menu options to quickly get insights into the future performance of this ML model, from our **automated risk analysis** methods.

There are plenty of other small enhancements and fixes in V2023 Q1 – and we have more new features "in the works". But you can see why we felt we were ready to deliver another major version to our customers – in time for Christmas!

What's New in Analytic Solver V2023 Q3

In Analytic Solver V2023 Q3, we've made it easier than ever to create analytic models, and easier to deploy them for ongoing use, through two major new features: our new conversational "AI Agent", with a button next to "Help" on the Analytic Solver Ribbon, and a new "Identify Inputs" feature that simplifies the task of updating key input parameters when it's time to re-run your model.

AI Agent: Ask for Help from ChatGPT "Trained on Analytic Solver"

By now, nearly everyone who's been following developments in software is aware of ChatGPT, the conversational agent using "Generative AI" methods, developed by the OpenAI nonprofit closely affiliated with Microsoft. You can ask ChatGPT questions on almost any topic – including analytic methods – and get meaningful and interesting (though not always 100% correct!) answers.

That's great – but what if you could have a ChatGPT "technical support agent" that had studied all 2,300 pages of Frontline's User Guides (including this one), Reference Guides and QuickStart Guides – everything about optimization, Monte Carlo simulation, data science and machine learning, and business rules – and was instructed to use its knowledge to answer your questions? Well, that's what you have in Analytic Solver V2023 Q3!

Actually making this work involves a fair amount of software engineering: Generative AI tools need the right "context" to respond to your question – so we've built an online resource where all those 2,300 pages of Guides are represented via "vector embeddings" that enable searches by meaning, not just "keyword matches". Our AI Agent automatically searches this online resource to create "context' for your queries to ChatGPT – and you can also search this online resource directly.

We're using the "real" online ChatGPT 3.5 Turbo version for its full conversational capabilities, which costs us money for every query – so you will find some limits placed on query length and number of queries per month, depending on your software license type. But we expect you'll be able to use our AI Agent to amplify your own efforts, ultimately building more capable and effective analytic models. We look forward to your feedback!

Identify Inputs: Easily Set Up your Model for Data Updates and New Solves

When you're first building and solving a model in Analytic Solver, usually your focus is on getting a solution for a specific instance of a business problem, with data you have today – gathered from one or several external sources into Excel. After some effort, you're now getting solutions from optimization, simulation, or data science and machine learning. That's a success by itself … but it leads to a desire to "do it again (and again) in the future."

But … those data and parameter values that *should be updated* for a future run may be scattered around your Excel spreadsheet, which also includes cells with calculated formulas, cells with constant data that doesn't need updating, and cells with data that you want to be visible, but that doesn't actually affect or "participate in" the analytic model. How do you find *just* the parameter values that *should* be updated – and then, how do you *actually update* them?

Now you can choose **Tools – Identify Inputs** and get help doing exactly that. This tool uses our PSI Interpreter to scan your model formulas, identify and list only those cell ranges that are candidates for data updates that will affect the model when solved. You can then pick and choose from a list of cell ranges, which ones you actually need to update for a new run.

Then you can either (i) let Analytic Solver "highlight" those cells with colors and backgrounds, or even better, (ii) automatically add calls to our PsiInput() function that reference those cells. These cells will automatically appear in the Task Pane as part of your model (under the **Input Data** heading) – but when you deploy your model for use *beyond Excel*, via our RASON and Solver SDK tools, you'll have easy ways to supply new values for exactly those input cells.

If you've planned in advance for data updates and re-solves, Tools – Identify Inputs can help you check and validate your work. But if – like most of us – you were focused on getting a first-time solution and 'put off' the task of data updates and re-solves, Tools – Identify Inputs can be a huge time-saver!

What's New in Analytic Solver V2024 Q2

In **Analytic Solver V2024 Q2**, our overall theme is "Core Performance Improvements" -- indeed we think every user with a model of nontrivial size and complexity should **upgrade** (at no extra cost), since you're likely to see performance improvements -- especially faster **"Setting Up Problem..."**.

Solving Outside Excel

The most visible change -- though *not* the one impacting the most users -- is on the menu for the **Optimize** button on the Ribbon: the last dropdown choice, **Solve on Solver Server**, has been greatly extended. In past Analytic Solver releases, this choice allowed you to solve your model (actually perform the optimization) **outside** the running Excel program, in a separate program called Solver Server (part of our Solver SDK product), that can run on your PC or on another PC on your local area network (LAN). But the Excel-based Analytic Solver add-in continued to "monitor" the solving process, which meant that you couldn't do *something else* with Excel, while waiting for a "long solve" to complete.

In Analytic Solver V2024 Q2, this choice allows you to solve your model (again, actually perform the optimization) **outside** of Excel, on any of three alternatives: the SDK-based **Solver Server**, our public, Azure-based **RASON server**, or on a new offering: our **Containerized RASON Server --** which brings all the capabilities of RASON to your own PC, another physical or virtual PC on your LAN, or your own cloud account (we'll say more about this in an upcoming blog post). And unlike in previous releases, V2024 Q2 is able to "monitor" the solving process **without** blocking your use of Excel for *something else*, even working on a different workbook. When the final solution for your long-running model is available, you'll be notified in Excel, and your workbook containing the original model will be automatically updated, without disturbing anything else.

New Versions of Gurobi and OptQuest Solvers

Analytic Solver V2024 Q2 also includes new (minor) releases of the **Gurobi** Solver and **OptQuest** Solver (again at no extra cost), that improve performance on a range of models using these plug-in Solver Engines. The Gurobi Solver (their version 11.0.2) has typical speed improvements of 8% to 18% compared to pre-V11 versions. The OptQuest Solver (their version 9.1.2.9) includes an improved "diversity search" algorithm, among other enhancements.

PSI Interpreter Improvements

In Analytic Solver V2024 Q2, we expect the biggest impact on performance for most users will come from some **deep** improvements in the **PSI Interpreter**, our "Polymorphic Spreadsheet Interpreter". This is the part of Analytic Solver (and RASON) that takes your model as expressed in Excel or RASON formulas, analyzes the model, and converts it into a form **usable by** the Solver Engines **--** whether it's linear or quadratic (for the LSLP or Gurobi Solvers), nonlinear (for the LSGRG, LSSQP and KNITRO Solvers), or "non-smooth / arbitrary" (for the Evolutionary and OptQuest Solvers). The PSI Interpreter is hard at work when you see "Setting Up Problem..." in the Task Pane, but also **during the solution** of nonlinear and non-smooth models.

In Analytic Solver V2024 Q2, the PSI Interpreter has been re-engineered to use "sparse methods in automatic differentation" (take a look at the Wikipedia article we've linked, if you're curious about the methods from algebra and calculus). The practical impact for users with large models is **faster** end-to-end solution times for linear models, less time spent in **"Setting Up Problem"**, and big savings in **memory use** (which translates to faster solutions, or solutions for models that exhausted memory in the past).

All you really need to know is "much better performance", but if you're curious, the rest of this post will seek to explain just some of what the PSI Interpreter does for you.

More About the PSI Interpreter: The Jacobian

You might be surprised to learn that the Solver Engines, or optimization algorithms, don't work with your Excel formulas at all. They require as input *tables of numbers* (constant or changing) that describe your model, at a rather low level. The PSI Interpreter reads, parses, and interprets your formulas and produces those tables of numbers.

The most fundamental such table is called the **Jacobian matrix --** see the Wikipedia article (rather technical) or "A Gentle Introduction to the Jacobian" which comes from machine learning, where the same matrix arises. In optimization models, this matrix has a row for the objective and each constraint, and a column for each decision variable **--** so its size grows with variables *times* constraints. Each matrix element is the **partial derivative** (or *rate of change*) of one constraint (or the objective) with respect to one decision variable.

In linear models, the partial derivatives are all **constant** numbers and are referred to as "LP coefficients" **--** but these numbers may not appear explicitly anywhere in your Excel model. They are computed by the PSI Interpreter. In nonlinear and non-smooth models, the partial derivatives are **not constant --** their values depend on the current values of (potentially many) decision variables. So the PSI Interpreter must **re-compute** them, and re-supply them to the Solver Engine, each time the Solver Engine tests a new set of values for the decision variables **--** and this can happen thousands or even millions of times while a model is being solved. So the PSI Interpreter has a lot of work to do.

Let's say your model has 32,000 variables and 32,000 constraints (the upper limits of our *Standard* Large-Scale LP/QP Solver **--** the *Extended* version removes the limits). That's not very large by our customers' standards **--** but it means the Jacobian matrix has 32,000 x 32,000 = 1,024,000,000 (just over **one billion**) elements. It takes a *lot* of memory, and a *lot* of computing to calculate a billion different values from your Excel model! The saving grace is that in *most* models, *most* of these matrix elements will be **zero**: A typical constraint will depend on a **small subset** of the decision variables, so its rate of change is zero with respect to all the other decision variables. The challenge for the PSI Interpreter is to **figure out which** elements will be zero (the "sparsity pattern"), **without** consuming a lot of memory and time just doing this. In V024 Q2, the PSI Interpreter is far better at this!

Beyond the Jacobian: The Hessian

When your model goes "beyond linear", even with just a quadratic objective (as in portfolio optimization), the Solver Engines will typically require a further large table of numbers, called the **Hessian matrix** (again see the Wikipedia article if you're interested). The Hessian has as many rows and as many columns as the number of **decision variables** in your model; it is typically computed for the objective function (at least). Each matrix element is the **second partial derivative** of one function (usually the objective) with respect to **a pair** of decision variables. To compute this, the PSI Interpreter must do even more work, and it must deal with the same issues of memory and computing time, for perhaps a billion values. Again *some* of these elements will be **zero**, since the function value may not depend on *all possible pairs* of variable values **--** the challenge is figuring out **which ones!** And again, in V2024 Q2 we've made major progress on this. We've seen some large customer models that formerly would always exhaust memory after running for hours, suddenly solve in a minute or less!

And There's (Much) More...

There's much more to the PSI Interpreter, that we don't have time or space to fully describe here. Whenever Microsoft adds **new built-in functions** or formula features (such as "spilled arrays") to Excel, we've been hard at work supporting those new features in the PSI Interpreter. Whenever you use menu options such as "Analyze Original Model" or "Analyze Transformed Model", notice your **Model Type** (LP for linear, QP for quadratic, etc.) or the counts of model **Dependencies** in the Task Pane, or create a Linearity Report, that's the PSI Interpreter at work. And the PSI Interpreter is also the key to **super-fast Monte Carlo simulation** in Analytic Solver, usually ten times faster (or more) than other Excel add-ins for simulation. The PSI Interpreter also parses and interprets **DMN** (Decision Model and Notation) for business rules and decision tables.

You might be surprised to learn how many of our competitors offering Excel add-ins (especially firms that have been acquired by other companies) have been "resting on their laurels" for a decade or more. At Frontline Solvers, ***we never rest!*** We spend millions each year on real R&D, to bring you the very best in modeling and analytic software tools.

What’s New in Analytic Solver V2024 Q3

In **Analytic Solver V2024 Q3**, we’ve gone beyond our conversational “AI Agent”, introduced one year earlier in Analytic Solver V2023 Q3: Our new “AI Assistant” doesn’t just consult our User Guides and publicly available info – it seeks to “understand” the *model you are working on*, so it can assist you in improving your model, or fixing problems with it!

But we don’t just focus on the latest AI methods – we also improve the usability and performance of the optimization and simulation solvers you use every day.

New AI Assistant

The most visible change in V2024 Q3 is on the Analytic Solver Ribbon, which now sports two “AI icons”: On the far right next to the **Help** icon is **AI Ask a Question**, similar to the capability we introduced in Analytic Solver V2023 Q3 (but enhanced): Ask it a question, and it will answer conversationally, with links to our example models, User Guides and Reference Guides. On the far left next to the **Model** icon is a new **AI Assist** icon: When you click this icon, the AI Assistant will examine and use “as context” your currently-open model in Excel (this also works in our cloud platform RASON): You can then ask a question and engage in a dialog with the Assistant about *your model*. For example, you can ask “is my model missing an essential element?” or “my model is building products from parts in inventory, how can I add another part?”

Optimization Improvements

In this release, we’ve included the latest versions of the **Gurobi Solver** (their 11.0.3), the **XPRESS Solver** (their 43.1.3.0), and the **MOSEK Solver** (their 10.2.0.3). For all the Solver Engines, we’ve expanded “Verbose” logging (enabled via the Task Pane Platform tab **General – Log Level** option) to include all the solution progress information that each Solver can provide. And in the case of the Gurobi Solver working on a non-integer, non-convex QCP (Quadratically Constrained Problem), verbose logging now provides “branch level” solution progress info that just wasn’t available before.

Simulation Improvements

In Monte Carlo simulation, we've added new "alternative parameter" distribution functions named PsiFatigueLifeAlt() and PsiFrechetAlt(). The first of these, the "fatigue life" or Birnbaum–Saunders distribution, is extensively used in reliability problems to model failure times. The second, the Frechet or inverse Weibull distribution, is used to model extreme events, for example in weather forecasting and hydroelectric power application.

What's New in Analytic Solver V2025 Q1

We've just released **Analytic Solver V2025 Q1**, with new versions of Analytic Solver Desktop and Cloud (for Excel users), Solver SDK (for developers), and our cloud platform RASON. This release features both **usability** improvements and a powerful new way to solve a **wide range** of **nonlinear** optimization models using the **Gurobi Solver** Engine **--** traditionally known for solving linear (and quadratic) mixed-integer models. This new approach yields **globally optimal** solutions **every time --** not possible with other nonlinear Solvers **--** and often **much faster** solutions.

Usability: New Flexible Dialogs

Analytic Solver has always offered an easy user interface with an Office-style Ribbon, Task Pane, and dialogs that you can move around and resize. But the **content** of many dialogs used fixed-size fonts and graphics. To help with both **accessibility**, and better use of **larger, high-resolution screens** when you're "consuming" this content, in Analytic Solver V2025 Q1 we've revised the Task Pane and scores of other dialogs with **"zoomable"** content. You can now press the CTRL key, then use the mouse wheel to enlarge or shrink the text and graphics in almost any dialog or pane of Analytic Solver. At the bottom of this page is an example, where we've enlarged the text in the top and bottom parts of the Task Pane, but left the middle part (Model Diagnosis) unchanged – so you can see the difference.

Performance: New Nonlinear Solver Power

Analytic Solver V2025 Q1 features a *very special* level of support for the just-released (November 19) **Gurobi Solver 12.0**. This popular Solver again improves performance on linear programming (about +4%) and mixed-integer programming (about 13%) models, and even better performance (about +28%) on non-convex mixed-integer quadratic models. But most interesting for our Excel users is Gurobi's improved support for (often non-convex) **mixed-integer nonlinear** optimization problems. Such problems arise in specialty chemicals, hydroelectric power, and some aeronautics and space applications.

Unlike other nonlinear Solvers (such as our LSGRG, LSSQP and KNITRO Solvers), Gurobi takes a different approach, that can yield **globally optimally** solutions **IF** you can formulate your model with (only) certain algebraic expressions **--** defined via special Python API calls **--** that use just over a dozen mathematical operators (plus, minus, times, divide, square, square root, exponential and logarithm, etc.). If you're really good at Python coding and really good with algebraic manipulation, you can code this yourself. But in Analytic Solver V2025 Q1, **you don't have to do any extra work** to define your nonlinear model for the Gurobi Solver.

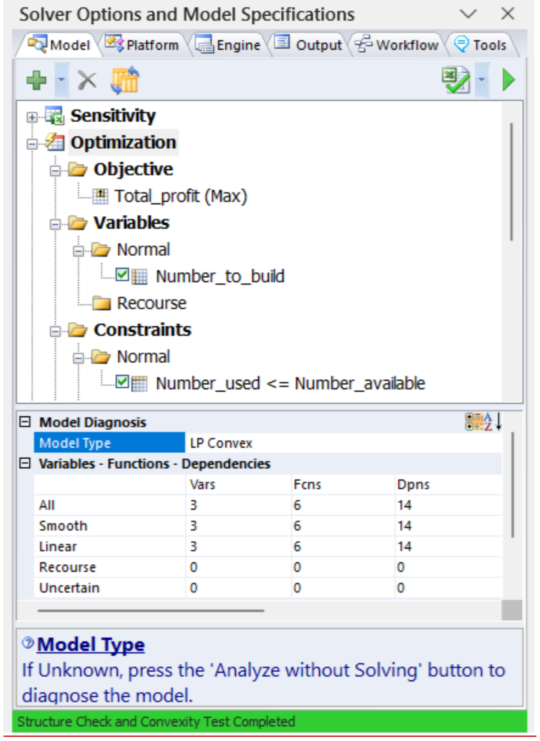
Standard Excel Formulas, New Solving Speed

You can write your model in Excel, using any of Excel's mathematical operators and any of Excel's 470+ built-in functions. Analytic Solver's **PSI Interpreter** will **analyze** your formulas (including formulas that depend on other formulas, to any level), **diagnose** your model as linear, quadratic, smooth nonlinear or non-smooth, and -- In V2025 Q1 -- tell you if your model is "nonlinear and **solvable via the new Gurobi Solver**". Analytic Solver can generate the internal "expression trees" that the Gurobi Solver requires.

Analytic Solver will even expand various Excel functions, based on their mathematical definitions, into chains of "Gurobi-supported primitive operators". For example, you can just go ahead and use the STDEV Excel function -- Analytic Solver will recognize this as "Gurobi-eligible" and create (internally) an expression tree using square, plus and square root operators, based on the definition of "standard deviation". The bottom line: With Analytic Solver, you can use the Gurobi 12.0 Solver to solve a much **wider range** of nonlinear models, much **more quickly and easily**, than you could any other way.

For "eligible" nonlinear optimization models, the Gurobi 12.0 Solver is often **faster** than other nonlinear Solvers, especially when **integer variables** are also used in the model, and when you require a **globally optimal** solution. This isn't always true however -- occasionally Gurobi is (much) slower -- so it's good that you have alternative nonlinear Solvers available to you, even in the basic Analytic Solver Optimization product. We do hope and expect to see even better performance in future versions of the Gurobi Solver.

Analytic Solver V2025 Q1 is a **free upgrade** for anyone with a paid Analytic Solver Optimization (or higher) license. If you've been a customer for years, you're probably accustomed to such dramatic enhancements -- but we're always working hard to bring you **more value, sooner** than anyone else in our "world" of advanced analytics.



Model Diagnosis

Model Type: LP Convex

Variables - Functions - Dependencies

	Vars	Fcns	Dpns
All	3	6	14
Smooth	3	6	14
Linear	3	6	14
Recourse	0	0	0
Uncertain	0	0	0

Model Type

If Unknown, press the 'Analyze without Solving' button to diagnose the model.

Structure Check and Convexity Test Completed

Optimization and Resource Allocation

Analytic Solver includes five bundled Solver Engines to find solutions for the full spectrum of optimization problems, up to certain size limits:

The **nonlinear GRG Solver** in Analytic Solver Comprehensive, Analytic Solver Optimization and Analytic Solver Upgrade handles large-scale smooth nonlinear (NLP) problems and includes Multistart or “clustering” methods for global optimization that can utilize multiple processor cores.

In Analytic Solver Upgrade, this engine is an upgraded version of the nonlinear GRG engine included in the Microsoft Excel Solver, and can solve NLP problems with up to 500 variables and 250 constraints, plus bounds on the variables. In Analytic Solver Comprehensive and Analytic Solver Optimization, this engine can handle NLP problems with up to 1,000 decision variables and 1,000 constraints, plus bounds on the variables.

The **LP/Quadratic Solver** in Analytic Solver Comprehensive, Analytic Solver Optimization and Analytic Solver Upgrade handles both linear programming (LP) and quadratic programming (QP) problems. In Analytic Solver Comprehensive and Analytic Solver Optimization, this engine can handle models with up to 8,000 decision variables and 8,000 constraints – four times the size of Analytic Solver Upgrade, and *40 times* the size of Excel Solver. In Analytic Solver Upgrade, this engine can solve LPs and QPs with up to 2,000 variables and 8,000 constraints. Advanced methods for LP/MIP problems yield solutions as much as hundreds of times faster than the Excel Solver, and 5 to 20 times faster than Analytic Solver Upgrade.

The **Evolutionary Solver** (in all products with optimization capabilities) uses a combination of genetic and evolutionary algorithms, tabu and scatter search methods, and classical optimization methods. In Analytic Solver Upgrade, it handles non-smooth (NSP) problems of up to 500 variables and 250 constraints, plus bounds on the variables. In Analytic Solver Comprehensive and Analytic Solver Optimization, an **enhanced version** of this Solver handles up to 1,000 variables and 1,000 constraints, exploits multiple processor cores in many of its algorithms, and exploits GPU processors in its local search.

The **Interval Global Solver** in (in all products with optimization capabilities) uses state-of-the-art interval methods to find the globally optimal solution to a nonlinear optimization problem, all real solutions to a system of nonlinear equations, or an “inner solution” to a system of nonlinear inequalities. In Analytic Solver Upgrade, this engine handles smooth nonlinear (NLP) problems of up to 200 decision variables and 100 constraints, plus bounds on the variables. In Analytic Solver Comprehensive and Analytic Solver Optimization, it handles smooth nonlinear problems of up to 500 decision variables and 250 constraints, plus bounds on the variables.

The **SOCP Barrier Solver** in (in all products with optimization capabilities) uses an interior point method to solve linear (LP), quadratic (QP), quadratically constrained (QCP), and second order cone programming (SOCP) problems – the *natural generalization* of linear and quadratic programming – with up to 2,000 variables and 8,000 constraints in Analytic Solver Comprehensive and Analytic Solver Optimization. In Analytic Solver Upgrade this engine has limits of 200 variables and 100 constraints. In Analytic Solver Comprehensive and Analytic Solver Optimization with Analytic Solver Simulation, the SOCP Barrier Solver can be used to solve linear programming models *with uncertainty* via robust optimization with the L2 norm.

Simulation and Risk Analysis

Analytic Solver includes a full-featured Monte Carlo simulation capability. Unlike other Excel-based simulation products, this feature enables you to play 'what if' with uncertain values as easily as you do with ordinary numbers. Each time you run a simulation, thousands of trials are executed, and a full range of simulation results and statistics will be displayed *on the spreadsheet*. It uses Frontline's *Polymorphic Spreadsheet Interpreter* technology to achieve breakthrough simulation speeds – up to **100 times faster** than normal Excel-based Monte Carlo simulation.

Analytic Solver supports nearly 80 analytic and custom distributions (continuous and discrete), and nearly 80 statistics, risk measures, Six Sigma functions, rank order correlation, distribution fitting, multiple random number generators and sampling methods with variance reduction, and powerful capabilities for multiple parameterized simulations.

In addition, Analytic Solver's **graphics** help you assess uncertainty, including charts of probability distributions, output frequency charts, sensitivity ("Tornado") charts, scatter plots, and Overlay, Trend, and Box-Whisker charts, in two or three dimensions. You can customize chart colors, titles, legends, grid lines, and markers, resize and rotate charts, and print charts or use them in PowerPoint presentations.

Optimal Solutions for Models with Uncertainty

Analytic Solver Comprehensive, Analytic Solver Simulation, Analytic Solver Upgrade and Analytic Solver Optimization with Analytic Solver Simulation support several powerful methods for finding robust solutions to optimization problems with uncertainty:

- High-speed **simulation optimization** methods for more general (non-linear, non-smooth or non-convex) problems, where uncertainties may depend on the first-stage decisions. Normal and chance constraints may be used, and a wide range of statistical aggregates can be used to summarize uncertainty in the objective and constraints. A simulation is performed on each major iteration of the optimization. This method is very general, but computationally very expensive, and usually *not scalable* to large size problems.
- **Robust optimization** (RO) methods for linear programming problems with uncertainties affecting the objective and constraints. Chance constraints specify a probability of satisfaction, which is converted to a 'budget of uncertainty.' Monte Carlo simulation is used to obtain bound and shape information for the uncertainties. This information is used to *automatically* create a robust counterpart problem, which is then solved. This method is *scalable* to large size models.
- **Robust optimization** methods and **stochastic programming** (SP) methods for *two-stage* stochastic linear programming problems with recourse ("wait and see") decisions. Scenarios are automatically created via built-in Monte Carlo simulation, or they may be drawn from user-defined cell ranges of sample values on the spreadsheet. With the benefit of second stage or *recourse* decisions, solutions are typically 'well-hedged' but not overly conservative. The RO and SP methods are relatively *scalable* to large size. A state-of-the-art implementation of **stochastic decomposition**, the most powerful and scalable method

known for solving two-stage stochastic linear programming problems, is also included. Note: Stochastic decomposition is supported only in Analytic Solver Desktop.

Automatic Model Transformation to RASON

A license for Analytic Solver Comprehensive includes facilities for automatic transformation of an existing model into the RASON modeling language.

RASON (which stands for Restful Analytic Solver Object Notation) is Frontline Systems modeling language embedded in JSON and a REST API that makes it easy to create, test and deploy analytic models using optimization, simulation, and data science, in web and mobile applications.

Users of Frontline's Analytic Solver Comprehensive will find that converting an existing Solver model in Excel to RASON is as easy as clicking a button. But we didn't stop there, in V2016, we created the ability to create an entire custom web application *in seconds*! Months of development work have been reduced to a single click of your mouse.

Web Application Developers who are familiar with AJAX and the use of REST API's, can immediately see first-hand how exceptionally easy it is to embed RASON models as JSON by viewing the automatically generated code.   In just a few seconds, you can see how to solve the models using Frontline's RASON server, which exposes a simple REST API that's scalable to handle very large, compute-intensive analytic models.

Sensitivity Analysis and Model Parameters

All Analytic Solver products include facilities for sensitivity analysis of your Excel model, that can be used before even starting an optimization, simulation or calculation model. It is especially easy to identify the model parameters with the most impact on your computed results – you can simply select *any formula cell* and choose **Parameters – Identify** from the Ribbon to quickly find the input cells with the greatest impact on this formula, ranked and shown in a Tornado chart.

You can choose some of these input cells to serve as Sensitivity Parameters, and then produce reports and charts that show the impact on computed results of varying these parameters over a range you specify. You can also turn these parameters into (or define other cells as) Simulation, Optimization or Calculation Parameters and produce reports and charts of simulation and optimization results as your parameters are automatically varied.

Multiple Optimizations, Simulations or Calculations

All Analytic Solver Products (except Analytic Solver Data Science) have powerful capabilities to perform multiple optimizations, simulations or calculations, automatically varying the values of parameters you specify, and collecting optimization, simulation or calculation results, which may be displayed on the spreadsheet, or summarized in a variety of built-in reports and charts. Many operations that would require a programming language in other systems, or VBA in the Excel Solver, can be easily performed without programming.

The **Dimensional Modeling** capability greatly enriches the scope of multiple parameterized optimizations and simulations. You can define multiple dimensions and multiple parameters indexed over each dimension, run an optimization or simulation for various combinations of these parameter values, and create tabular reports, Pivot Tables, and charts of the results.

Decision Trees on the Spreadsheet

Analytic Solver Comprehensive includes a facility to create decision trees on your Excel spreadsheet. Using the Ribbon, you can easily create decision nodes and branches, event nodes and branches, and terminal nodes. The tree is drawn in graphic form on the spreadsheet; standard Excel worksheet formulas compute ‘rollback’ values at each node, and the best-choice value at the root node, based on either expected value or utility function (certainty equivalent) criteria. With a Ribbon choice, you can graphically highlight the optimal path through the tree.

Since all computations for decision trees are performed via standard Excel worksheet formulas, you can use decision trees in your simulation and optimization models. You must make discrete choices for decisions, and define discrete alternatives for events, but you can, for example, define distributions for event outcomes and/or costs incurred for decisions, and view the composite distribution of outcomes at the root node.

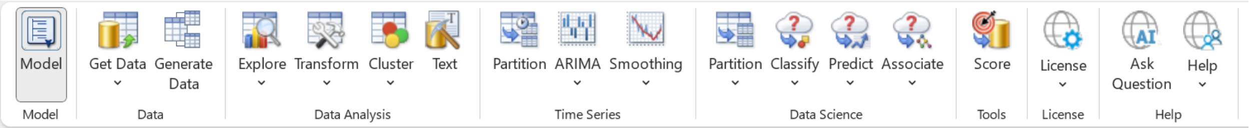
Bringing Big Data into Excel using Apache Spark

All Analytic Solver products include the ability to compute summary measures (sum, average, standard deviation, minimum or maximum) for variables in a dataset with up to billions of rows, stored across many hard disks in an external compute cluster running Apache Spark (https://spark.apache.org/), by clicking the **Get Data** icon. The results, which can be obtained immediately upon completion or at a later time, will include the number of variables included in the dataset and their names along with data counts for categorical variables. This kind of summary data is often what you need as input parameters to an optimization or simulation model. See the Analytic Solver Data Science User Guide for more help on this new feature.

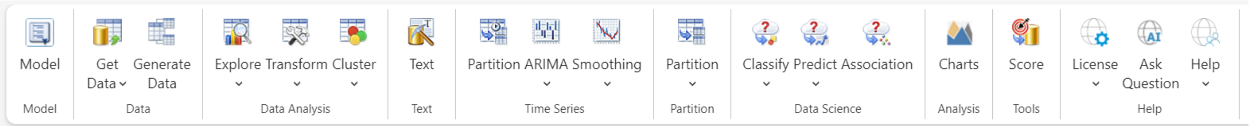
Forecasting and Data Science

Analytic Solver Comprehensive, Analytic Solver Data Science (formerly Analytic Solver Data Mining) and the Data Science Cloud (formerly Data Mining Cloud) app include powerful features for data visualization, forecasting and data science. You can explore these features using Analytic Solver Comprehensive or Data Science– just click the Data Science tab on the Excel Ribbon, and consult the Analytic Solver Data Science User Guide:

Desktop Analytic Solver Data Science



Data Science Cloud



- Click the Model button to display the Solver Task Pane. This feature (added in V2016) allows you to quickly navigate through datasets and worksheets containing Analytic Solver Data Science results.
- Click the **Get Data** button to draw a random sample of data, or summarize data from a (i) an Excel worksheet, (ii) the PowerPivot “spreadsheet data model” which can hold 10 to 100 million rows of data in Excel, (iii) an external SQL database such as Oracle, DB2 or SQL Server, or (iv) a dataset with up to billions of rows, stored across many hard disks in an external **Big Data** compute cluster running Apache Spark (https://spark.apache.org/).
- You can use the **Data Analysis** group of buttons to explore your data, both visually and through methods like cluster analysis, transform your data with methods like Principal Components, Missing Value imputation, Binning continuous data, and Transforming categorical data, or use the **Text Mining** feature to extract information from text documents.
- Use the **Time Series** group of buttons for time series forecasting, using both Exponential Smoothing (including Holt-Winters) and ARIMA (Auto-Regressive Integrated Moving Average) models, the two most popular time series forecasting methods from classical statistics. These methods forecast a single data series forward in time.
- The **Data Science** group of buttons give you access to a broad range of methods for prediction, classification and affinity analysis, from both classical statistics and data mining including Ensemble Methods for use with Classification Trees, Regression Trees, and Neural Networks.

  Find Best Model uses methods similar to those in (expensive high-end) tools like DataRobot and RapidMiner, to automatically choose types of ML models and their parameters, validate and compare them according to criteria that you choose, and deliver the model that best fits your data. These methods use multiple input variables to predict an outcome variable or classify the outcome into one of several categories.
- Use the **Predict** button to build prediction models using Multiple Linear Regression (with variable subset selection and diagnostics), k-Nearest Neighbors, Regression Trees, and Neural Networks. Use Ensemble Methods with Regression Trees and Neural Networks to create more accurate prediction models. Use Find Best Model Predict to automatically compare the performance of several ML models on your dataset.
- Use the **Classify** button to build classification models with Discriminant Analysis, Logistic Regression, k-Nearest Neighbors, Classification Trees,

Naïve Bayes, and Neural Networks. Use Ensemble Methods with Classification Trees and Neural Networks to create more accurate classification models. Use Find Best Model Classification to automatically compare the performance of several ML models on your dataset.

- Use the **Associate** button to perform affinity analysis ("what goes with what" or market basket analysis) using Association Rules.
- Use the **Score** button to score new data using a fitted forecasting, classification or prediction model.
- If forecasting and data science are new for you, don't worry – you can learn a lot about them by consulting our **AI Agent**, Frontline's artificial intelligence technical support assistant. AI Agent is designed to provide assistance and support for users of Frontline Solvers' Analytic Solver and Analytic Solver Data Science software. The AI Agent is knowledgeable about the functionality and features of the software, as well as the concepts and processes involved in optimization, simulation and data science/forecasting. Just enter a topic or question such as "What classification algorithms are supported in Analytic Solver Data Science?" and click Submit Query to get started.
- Use the **License** button to manage your account and licenses.
- Use the **Help** button to open example models, open the Help Center, where you can find pre-recorded webinars or access our Knowledge Base or explore our User Guides.

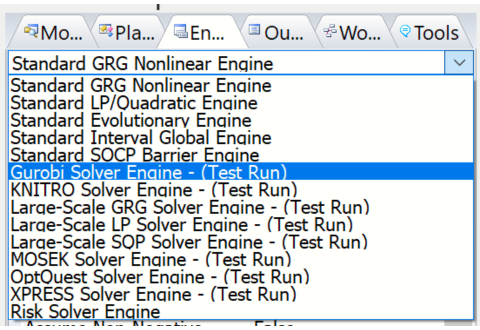
If you'd like to learn more and get started as a 'data scientist,' consult the excellent book *Data Mining for Business Intelligence*, which was written by the original Data Science (formally known as XLMiner and more recently Analytic Solver Data Mining) designers and early academic users. You'll be able to run all the Data Science examples and exercises in Analytic Solver.

Analytic Solver Data Science, along with the Data Science Cloud app, can be purchased as a stand-alone product. A stand-alone license for Analytic Solver Data Science includes all of the data analysis, time series data capabilities, classification and prediction features available in Analytic Solver Comprehensive and supports optimization and simulation models at limited size constraints. See Data Specifications, within Analytic Solver Data Science User Guide, for more information on each product.

Large-Scale Solver Engines

Frontline's Analytic Solver Comprehensive and Analytic Solver Optimization (in both Desktop and Cloud apps) support multiple optional, plug-in Solver engines, in addition to their "bundled" Solver engines. Such Solver engines are licensed as separate products; they provide additional power and capacity to solve problems much larger and/or more difficult than the problems handled by the bundled Solver engines. Unlike most other optimization software, a license for one of Frontline's Solver engines enables you to use that Solver in desktop Excel or Excel Online. In addition, if using Solver SDK, you may call the engine through C/C++, C#, Visual Basic, VB.NET, as MATLAB, Java, PHP, JavaScript and other languages.

Frontline Systems allows you to take these engines on a **Test Run**. (You don't get full results for all variables and constraints, but you do get the final objective value and solution time.) To make it clear what your current license does and doesn't include, you'll see "(Test Run)" next to the names of optional Solver Engines in the Task Pane.


Mo...
Pla...
En...
Ou...
Wo...
Tools
Standard GRG Nonlinear Engine
Standard GRG Nonlinear Engine
Standard LP/Quadratic Engine
Standard Evolutionary Engine
Standard Interval Global Engine
Standard SOCP Barrier Engine
Gurobi Solver Engine - (Test Run)
KNITRO Solver Engine - (Test Run)
Large-Scale GRG Solver Engine - (Test Run)
Large-Scale LP Solver Engine - (Test Run)
Large-Scale SQP Solver Engine - (Test Run)
MOSEK Solver Engine - (Test Run)
OptQuest Solver Engine - (Test Run)
XPRESS Solver Engine - (Test Run)
Risk Solver Engine

These Solver engines are seamlessly integrated – to use one, you simply select the Solver engine by name in the dropdown list that appears in the Task Pane Engine tab. They produce reports as Excel worksheets, like the bundled Solver engines; their options can be set in the Task Pane, or, if using Analytic Solver Desktop, in Solver Options dialogs; and they can be controlled by VBA code in your custom applications. The Solver engines are also seamlessly integrated into the Solver Platform SDK. Trial licenses for these Solver engines are available, allowing you to evaluate how well they perform on a challenging Solver model that you've developed.

The large-scale Solver Engines are more valuable than ever in our Analytic Solver products, where you can create large, structured models more easily with **Dimensional Modeling**, and transform linear programming models *with uncertainty* into larger, *conventional* linear programming or second order cone programming models, using stochastic programming and robust optimization. Note: Dimensional Modeling is not supported in Analytic Solver Cloud.

Automatic Mode and Solution Time

Analytic Solver has many performance enhancements, ranging from faster startup and faster switching among workbooks to faster Solver engines, and a deeply parallelized design that exploits multi-core processors. But its Automatic mode can sometimes *increase* solution time.

If using Analytic Solver Desktop, **Automatic mode** refers to the Automatic choice available for several Platform options, which formerly required some thought and manual setting by the user. (Since Analytic Solver Cloud always uses Automatic mode, this option is not applicable in Analytic Solver Cloud.)

In the desktop products, Automatic is the *default* choice for these options in newly created models. The options (found on the Task Pane Platform tab) include:

Optimization Model Interpreter

Solve Uncertain Models

Nonsmooth Model Transformation

Stochastic Transformation

Simulation Model Interpreter

Supply Engine With

Automatically Select Engine
(on the Task Pane Engine tab)

Each Automatic choice means that Analytic Solver can analyze your model and automatically choose the best setting for these options. This normally results in better solutions and faster times; however, the process of *analyzing* your model and *trying* certain alternatives can take *extra* time, especially for large models.

To improve solution time when you re-run your model, you can inspect the solution log in the Task Pane Output tab to see what choices were made automatically, **manually set the Platform options** to these same choices, and save your workbook. This will save time when you next run the model.

The desktop version of Analytic Solver Upgrade includes Automatic Mode for analyzing the model and selecting the best available engine. However, this product does not include the ability to automatically *transform* models.

Help, Guided Mode and Proactive Support

Frontline's Solvers include User Guides, Reference Guides and over a hundred example models – but we've gone much further to provide help. If using a desktop product, the in-product help will open whenever you click any hyperlinked element in the Task Pane or Help icon in the Ribbon

Our **Wizards** can help you with the hardest parts of defining a simulation model (probability distributions) or an optimization model (constraints). **Guided Mode** helps you step-by-step with optimization solutions, when you're first getting started. **Auto-Help Mode**, the default setting, is something you can use all the time, since it displays explanatory dialogs only when you encounter a problem!

Support Live Chat enables you to connect instantly to Frontline's technical support personnel from inside Excel, send your last error message, model diagnosis and other information , and get help from a live person, and **Proactive Support**, which automatically creates a support ticket in our Help Desk when you're struggling with repeated error messages or modeling problems.

Help, Support, Licenses and Product Versions

Introduction

This chapter summarizes the Frontline Solvers product line, explains the licensing system and how it benefits you, and describes how to use the newly introduced Cloud apps and the License and Help Centers.

Working with Licenses V2025 Q1

A **license** is a grant of rights, from Frontline Systems to you, to use our software in specified ways. As we learned in the last chapter, when you run the SolverSetup program to install Analytic Solver Desktop, software for the full Analytic Solver product family is installed. However, the product features you see in Analytic Solver Desktop and Analytic Solver Cloud, and the size of models you can solve, depends on the license you have. Frontline Solvers V2025 Q1 uses a **licensing system**, developed in V2017-R2, that offers you more flexible ways to use the software, both desktop and cloud.

Licenses Tied to You, Not Your Computer

Information about a license, including the features and size limits it enables and its expiration date, is maintained on Frontline's cloud servers, associated with your **user account ID**, **email address** and **password** on Solver.com. When you're using Analytic Solver on your desktop, license information is periodically downloaded or 'refreshed' from the cloud servers, but is stored on your PC for short periods, to enable you to run even when not connected to the Internet.

In Analytic Solver V2025 Q1, your license is associated with **you**, and may be used on **more than one PC**. For example, you can run SolverSetup to install the software on your office PC, your company laptop, and your PC at home. But only **you** can use Analytic Solver, and only on **one** of these computers **at a time**.

Using Your License: Login and Logout

When you first use Analytic Solver either in the cloud or on a new computer, **you will be prompted to login**. Enter the **email address** and **password** that you used to register on Solver.com. Once you've done this in Analytic Solver Desktop, your identity will be "remembered," so you won't have to login every time you start Excel, and go to one of the Analytic Solver tabs. If you are using Analytic Solver Cloud, you will need to login at the beginning of each session.

You can login and logout at any time, using **Login/Logout** on the License menu. If you share use of a single physical computer with other Analytic Solver users, be careful to **login** with your own email and password, and **log out when**

you're done – if you don't, other users could access private files in your cloud account, or use up your allotted CPU time or storage.

When you move from one computer to another, you should **log out** on one and **log in** on the other. As a convenience, if you log in to Analytic Solver on a new computer when you haven't logged out on the old computer, Analytic Solver will let you know, and offer to automatically log you out on the other computer.

Upgrading from Earlier Desktop Versions

A new license will not be required for V2025 Q1 if you're upgrading from Frontline Solvers V2024 Q3 thru V2018. Old license codes for V2016 and earlier have no negative effect in the current version. If they exist in the obsolete Solver.lic file (located at C:\ProgramData\Frontline Systems), they will be ignored. A new license will be issued at the time of purchase.

Renamed Products Offering More

Frontline Systems has simplified the names of products that are subsets of full Analytic Solver Platform, which is now called **Analytic Solver Comprehensive**:

- **Analytic Solver Upgrade** – Formerly Premium Solver Pro, our basic 'Excel Solver Upgrade'.
- **Analytic Solver Optimization** – Formerly Premium Solver Platform, full-size conventional optimization
- **Analytic Solver Simulation** – Full size Monte Carlo simulation, risk analysis and decision trees, plus simulation optimization (only)
- **Analytic Solver Data Science** – Formerly XLMiner Platform and more recently Analytic Solver Data Mining.
- **Analytic Solver Comprehensive** – Formerly Analytic Solver Platform
- **Analytic Solver Large-Scale** – A special bundle of Analytic Solver Comprehensive, Large-Scale LP/QP Solver, and Large-Scale GRG Solver

Analytic Solver Optimization, Analytic Solver Simulation and Analytic Solver Data Science include all supported features of Analytic Solver Upgrade. This means, for example, that when you upgrade to Analytic Solver Upgrade Analytic Solver Optimization, you still have access to all of Analytic Solver's simulation and data science features, for small models and datasets.

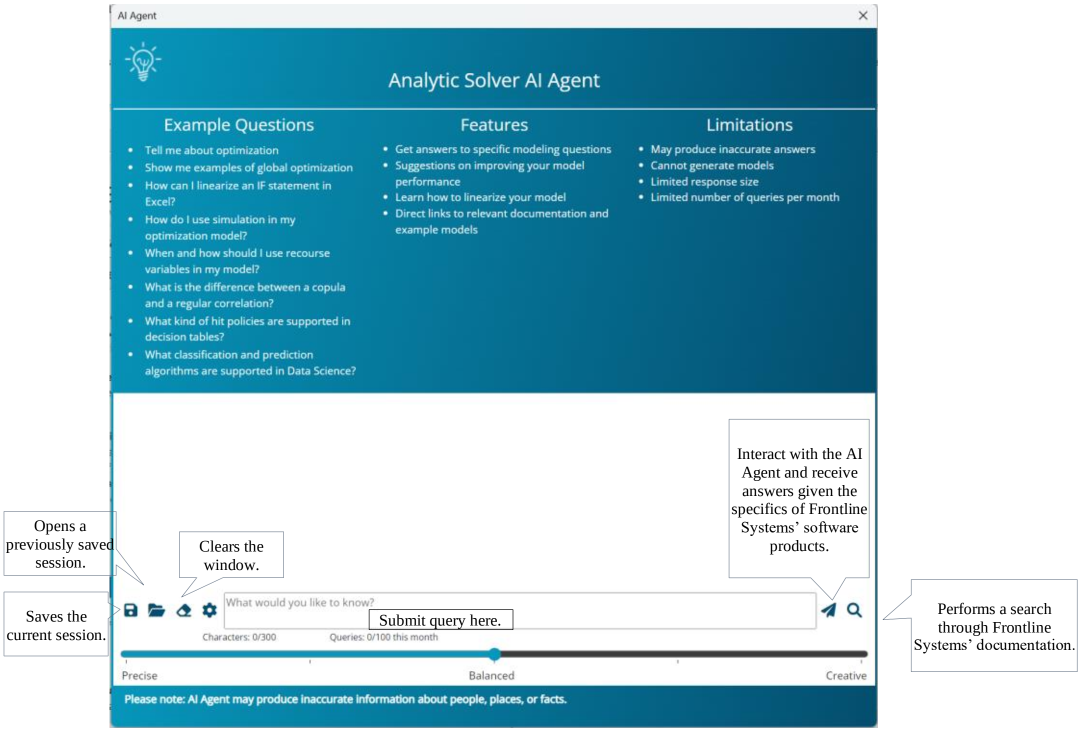
Getting Help

The AI Help features in Analytic Solver are designed to help users of Frontline Systems' software, including Analytic Solver and Analytic Solver Data Science, by providing technical support and guidance.

- Click AI Assist (on the far left of the Analytic Solver ribbon) while your model is open to allow AI Assist to analyze your model and check the formulation for correctness.
- Click Ask Question (on the far right of the Analytic Solver and Data Science ribbons) to open the AI Agent. AI Agent is knowledgeable about the functionalities and features of the software, as well as the underlying concepts and processes involved in optimization, simulation, forecasting, and data science.

Ask Question

If you are new to Analytic Solver and/or optimization, simulation, forecasting and data science, don't worry – Frontline's AI technical support assistant, AI Agent, is here to help. AI Agent is designed to provide assistance and support for users of Frontline Solvers' Analytic Solver and Analytic Solver Data Science software. The AI Agent is knowledgeable about the functionality and features of the software, as well as the concepts and processes involved in optimization, simulation and data science/forecasting. Just enter a topic or question such as "What classification algorithms are supported in Analytic Solver Data Science?" and click Submit Query to get started.



Analytic Solver AI Agent

Example Questions

- Tell me about optimization
- Show me examples of global optimization
- How can I linearize an IF statement in Excel?
- How do I use simulation in my optimization model?
- When and how should I use recourse variables in my model?
- What is the difference between a copula and a regular correlation?
- What kind of hit policies are supported in decision tables?
- What classification and prediction algorithms are supported in Data Science?

Features

- Get answers to specific modeling questions
- Suggestions on improving your model performance
- Learn how to linearize your model
- Direct links to relevant documentation and example models

Limitations

- May produce inaccurate answers
- Cannot generate models
- Limited response size
- Limited number of queries per month

What would you like to know?

Characters: 0/300

Queries: 0/100 this month

Please note: AI Agent may produce inaccurate information about people, places, or facts.

Opens a previously saved session.

Saves the current session.

Clears the window.

Submit query here.

Interact with the AI Agent and receive answers given the specifics of Frontline Systems' software products.

Performs a search through Frontline Systems' documentation.

Toggle between Precise, Balanced, and Creative to determine the type of answers returned. If Precise is selected, AI Agent will attempt to be as exact and deterministic as possible while Creative will usually result in more original, uncertain and non-repetitive answers. Use Balanced (the default) for the best of both worlds.

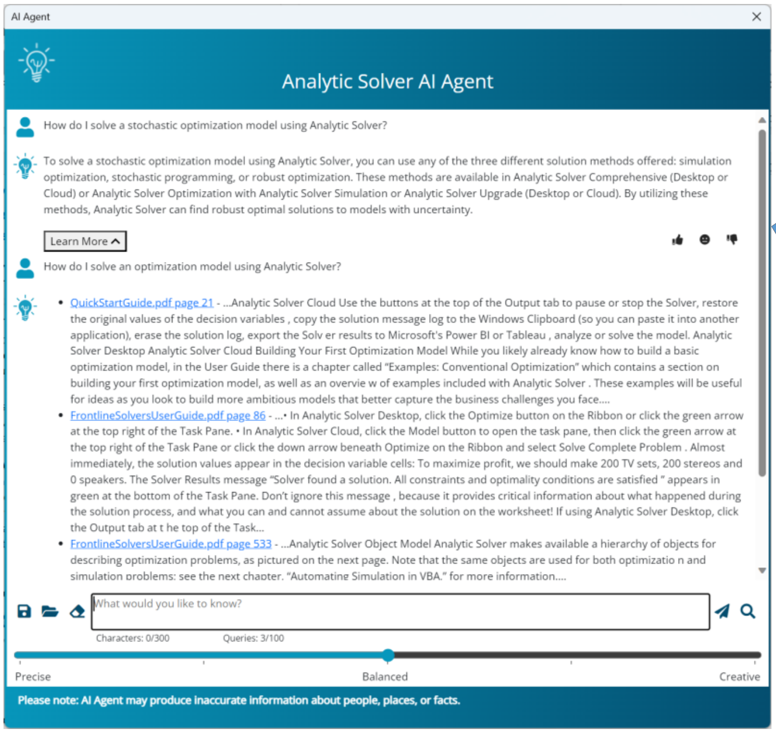
Click the **Save** file icon to save the current session. The file, transcript.json, will be downloaded. Click the **Open** file icon to open and restore a previously saved session. Click the **Erase** icon, to clear the current search window and start a new topic.

After typing a query,

- Click **Submit Query** to interact with the AI Agent and receive answers given the specifics of Frontline Systems' software products. On each new question, the AI Agent will take into account the chat history that is present in the current session. Click *Learn More* to view all extracts of source documents that are most relevant to the query. Each extract contains a link to the appropriate page in one of the official documents.
- Click **Search Documentation** (for faster search results) to perform a semantic/relevance search through all Analytic Solver and RASON

documentation. The resulting response will be similar to the “Learn More” section as described above.

The screenshot below illustrates the difference between the two different types of queries. The first query used AI technology by using Submit Query. The 2nd query was performed using Search Documentation.



How do I solve a stochastic optimization model using Analytic Solver?

To solve a stochastic optimization model using Analytic Solver, you can use any of the three different solution methods offered: simulation optimization, stochastic programming, or robust optimization. These methods are available in Analytic Solver Comprehensive (Desktop or Cloud) or Analytic Solver Optimization with Analytic Solver Simulation or Analytic Solver Upgrade (Desktop or Cloud). By utilizing these methods, Analytic Solver can find robust optimal solutions to models with uncertainty.

How do I solve an optimization model using Analytic Solver?

- QuickStartGuide.pdf page 21 - ...Analytic Solver Cloud Use the buttons at the top of the Output tab to pause or stop the Solver, restore the original values of the decision variables , copy the solution message log to the Windows Clipboard (so you can paste it into another application), erase the solution log, export the Solv er results to Microsoft's Power BI or Tableau , analyze or solve the model. Analytic Solver Desktop Analytic Solver Cloud Building Your First Optimization Model While you likely already know how to build a basic optimization model, in the User Guide there is a chapter called “Examples: Conventional Optimization” which contains a section on building your first optimization model, as well as an overvie w of examples included with Analytic Solver . These examples will be useful for ideas as you look to build more ambitious models that better capture the business challenges you face....
- FrontlineSolversUserGuide.pdf page 86 - ...• In Analytic Solver Desktop, click the Optimize button on the Ribbon or click the green arrow at the top right of the Task Pane. • In Analytic Solver Cloud, click the Model button to open the task pane, then click the green arrow at the top right of the Task Pane or click the down arrow beneath Optimize on the Ribbon and select Solve Complete Problem . Almost immediately, the solution values appear in the decision variable cells: To maximize profit, we should make 200 TV sets, 200 stereos and 0 speakers. The Solver Results message “Solver found a solution. All constraints and optimality conditions are satisfied ” appears in green at the bottom of the Task Pane. Don’t ignore this message , because it provides critical information about what happened during the solution process, and what you can and cannot assume about the solution on the worksheet! If using Analytic Solver Desktop, click the Output tab at t he top of the Task...
- FrontlineSolversUserGuide.pdf page 533 - ...Analytic Solver Object Model Analytic Solver makes available a hierarchy of objects for describing optimization problems, as pictured on the next page. Note that the same objects are used for both optimizatio n and simulation problems: see the next chapter. “Automating Simulation in VBA.” for more information....

What would you like to know?

Please note: AI Agent may produce inaccurate information about people, places, or facts.

Be sure to rate answers to improve future performance!

AI Assist

Frontline Systems' new AI assist is a feature designed to enhance model formulation for optimization, simulation, and stochastic models. AI Assist helps users by:

1. **Analyzing Model Formulation**: AI Assist incorporates artificial intelligence to analyze your existing model and giving you tips on how to expand and improve your current model formulation.
2. **Providing Recommendations**: The AI can offer suggestions for improving model performance or adjusting parameters to achieve better results.
3. **Enhancing User Experience**: It aims to simplify the process for users by offering intuitive guidance and insights, making advanced analytics more accessible.

This feature is part of Frontline Systems' broader effort to integrate AI into their suite of decision support tools, making it easier for users to leverage powerful optimization and analytics capabilities.

AI Assist Example

To demonstrate this exciting new feature, open the Product Mix example model by clicking Help – Examples – Optimization Models. While the model worksheet is open, click AI Assist. AI Assist opens and automatically enters the prompt, "Please describe the model and check for correctness."

AI Assist first translates your model to a RASON model, Frontline's modeling language, then diagnoses the model type and examines the model for correct formulation.

```
Please describe the model and check for correctness

### Description of the Model
```

Model diagnosis

```
The provided model is an optimization type,
specifically focused on determining the optimal mix
of products a company should manufacture in order to
maximize gross profit. Here's a detailed breakdown:
```

Model name & description

```
#### General Information
- **Model Name**: ProductMix(Opt)
- **Model Description**: Not provided, likely a
version of a Product Mix optimization problem.
- **Worksheets**: The model uses two worksheets named
"Example 1" and "Example 2", but it is set to use
"Example 1" as the active sheet.
```

Engine settings

```
#### Engine Settings
- **Engine**: LP/Quadratic (Linear
Programming/Quadratic)
- **Scaling**: -1 (Automatic scaling is disabled)
```

Variables

```
#### Variables
- **Alias Name**: Number_to_build
- **Initial Value**: 0
- **Lower Bound**: 0
- **Final Value**: []

These variables represent the number of different
products to be manufactured.
```

Description of data used in model.

```
#### Data
- **Production Coefficients**: `c18:e22` represents
the production coefficients matrix, which is a 2D
array of the resources consumed by each product. The
values are:
- `[[1, 1, 0], [1, 0, 0], [2, 2, 1], [1, 1, 0], [2,
1, 1]]`

- **Inventory/Resources Available**: `h18:h22`
represents the available quantity of each
part/resource. The values are:
- `[450, 250, 800, 450, 600]`, referred to as
"Number_available".

- **Profit per Unit**: `c24:e24` indicates the profit
per unit of each product. The values are:
- `[[75, 50, 35]]`
```

Description of constraints

```
#### Constraints
- **Alias Name**: Number_used
- **Formula**: For each part/resource, the total used
is computed via the SUMPRODUCT of production
coefficients and the variables (Number_to_build).
- **Upper Bound**: "Number_available"

The constraints ensure that the total number of each
part/resource used to produce the products does not
exceed the available quantities.
```

Description of objective

```
#### Objective
- **Alias Name**: Total_profit
- **Formula**: The objective function is to maximize
the total profit calculated as the SUMPRODUCT of the
profit per unit and the number of units to be built.
- **Type**: Max (Maximize the total profit)
- **Final Value**: []
```

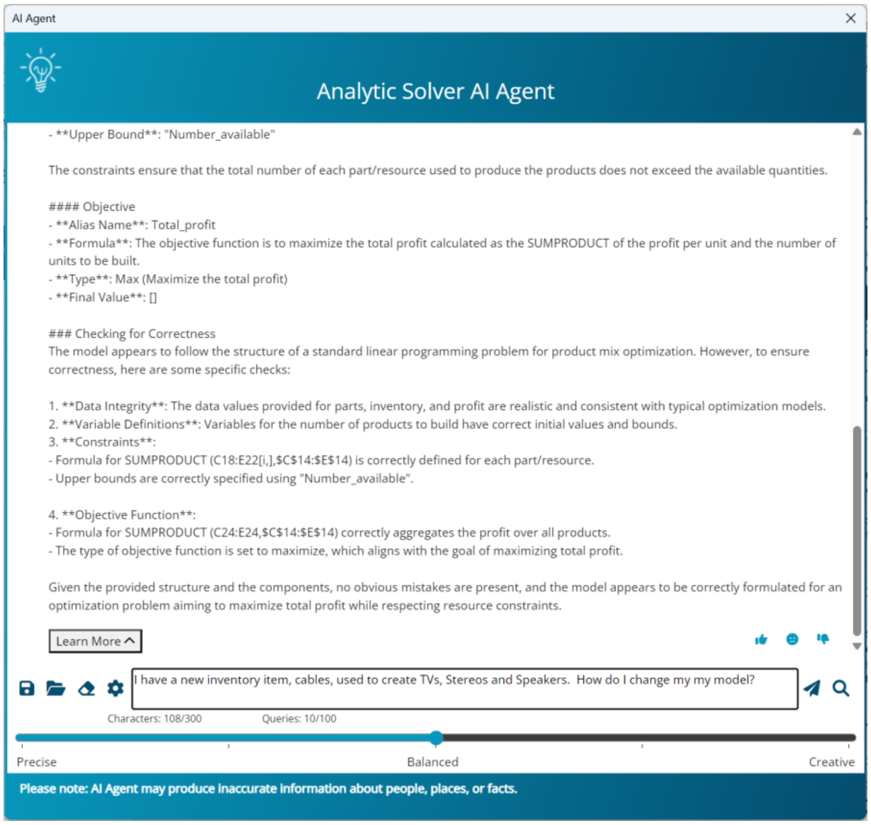
Results of analysis

```
### Checking for Correctness
The model appears to follow the structure of a
standard linear programming problem for product mix
optimization. However, to ensure correctness, here
are some specific checks:

1. **Data Integrity**: The data values provided for
parts, inventory, and profit are realistic and
consistent with typical optimization models.
2. **Variable Definitions**: Variables for the number
of products to build have correct initial values and
bounds.
3. **Constraints**:
- Formula for SUMPRODUCT (C18:E22[i,],$C$14:$E$14) is
correctly defined for each part/resource.
- Upper bounds are correctly specified using
"Number_available".

4. **Objective Function**:
- Formula for SUMPRODUCT (C24:E24,$C$14:$E$14)
correctly aggregates the profit over all products.
- The type of objective function is set to maximize,
which aligns with the goal of maximizing total
profit.
```

Conclusions of model analysis.

```
Given the provided structure and the components, no
obvious mistakes are present, and the model appears
to be correctly formulated for an optimization
problem aiming to maximize total profit while
respecting resource constraints.
```

Assume that you have a new part that must be incorporated into the production of all products, but you aren't sure how to change the model formulation to reflect this new part. Simply ask AI Assist!



```
- **Upper Bound**: "Number_available"

The constraints ensure that the total number of each part/resource used to produce the products does not exceed the available quantities.

#### Objective
- **Alias Name**: Total_profit
- **Formula**: The objective function is to maximize the total profit calculated as the SUMPRODUCT of the profit per unit and the number of units to be built.
- **Type**: Max (Maximize the total profit)
- **Final Value**: []

### Checking for Correctness
The model appears to follow the structure of a standard linear programming problem for product mix optimization. However, to ensure correctness, here are some specific checks:

1. **Data Integrity**: The data values provided for parts, inventory, and profit are realistic and consistent with typical optimization models.
2. **Variable Definitions**: Variables for the number of products to build have correct initial values and bounds.
3. **Constraints**:
- Formula for SUMPRODUCT (C18:E22[i,],$C$14:$E$14) is correctly defined for each part/resource.
- Upper bounds are correctly specified using "Number_available".

4. **Objective Function**:
- Formula for SUMPRODUCT (C24:E24,$C$14:$E$14) correctly aggregates the profit over all products.
- The type of objective function is set to maximize, which aligns with the goal of maximizing total profit.

Given the provided structure and the components, no obvious mistakes are present, and the model appears to be correctly formulated for an optimization problem aiming to maximize total profit while respecting resource constraints.
```

I have a new inventory item, cables, used to create TVs, Stereos and Speakers. How do I change my my model?

Characters: 108/300
Queries: 10/100

Please note: AI Agent may produce inaccurate information about people, places, or facts.

Notice that the prompt, "I have a new inventory item, cables, used to create TV, Stereos and Speakers. How do I change my model?", gives specific information. The more detailed the context provided in the prompt, the better the response will be. Click OK to view the response.

The response includes a step-by-step guide on all the changes that must be incorporated into your model in order to add a new part to the production requirements.

1. A new inventory item must be added.
2. Data containing the number of cables to be used in the product of each product must be updated.
3. A new constraint must be added to ensure that the inventory is not exceeded.

~~~
To include a new inventory item like cables in your
optimization model for manufacturing TVs, stereos, and
speakers, you'll need to update several components of the
model. Here's a step-by-step guide on how to modify the
model:

### Step-by-Step Guide:

1. **Add New Inventory Item:**
- In your parts inventory section, add a new row for
cables.
- Specify the initial amount of cables available in the
inventory.

Updated Inventory Table:
```plaintext
Part Name | Inventory | No. Used - TV | No. Used - Stereo |
No. Used - Speakers
-----------------------------------------------------------
-------------------
Chassis | 450 | 1 | 1 | 0
Screen | 250 | 1 | 0 | 0
Speaker | 800 | 2 | 2 | 1
Power Supply | 450 | 1 | 1 | 0
Electronics | 600 | 2 | 1 | 1
**Cables** | [Specify] | [Specify] | [Specify] | [Specify]
```

2. **Specify Part Requirement by Product:**
- Determine how many cables are used for each product (TVs,
stereos, and speakers).
~~~

~~~~plaintext
- Fill in these values under the "No. Used" columns for TV,
Stereo, and Speakers.

Example (Assuming the usage is):
```plaintext
Cables | 600 | 2 | 1 | 1
```

3. **Update the Constraints:**
- The constraints need to be updated to include the cable
usage and inventory.

Updated Constraints in Pseudocode:
```plaintext
Chassis_Required <= Chassis_Inventory
Screen_Required <= Screen_Inventory
Speaker_Required <= Speaker_Inventory
PowerSupply_Required <= PowerSupply_Inventory
Electronics_Required <= Electronics_Inventory
Cables_Required <= Cables_Inventory
```

In Excel, add a new constraint for the cables in the
constraints section:
```plaintext
Used_Cables = SUM(No. Used - TV * TVs_Built + No. Used -
Stereo * Stereos_Built + No. Used - Speakers *
Speakers_Built) <= Available_Cables
```

4. **Adjust Objective Function if Necessary:**
- Generally, the addition of a new inventory item does not
change your objective function (e.g., maximizing profit),
but ensure your objective function is calculated based on
the products built.

5. **Update the Excel Model (or other LP modeling tool):**
- Add the new “Cables” row in the appropriate section for
parts.
- Define the usage of cables for each product.
- Ensure the constraint formulas in Excel or elsewhere
include the new cables resource.

6. **Formulate the Model:**
- Double-check if all parts and their respective
constraints are properly entered.
- Ensure the Solver (or other optimization tool) is
configured correctly to consider the new constraints.

### Updating the Model Example in Excel:
Assuming you are using Excel, below is a simplified
example:
1. **New Inventory Row:**
- Add to the Inventory section, say cell `B6` for cables
inventory:
```plaintext
Cables | 600
```

2. **Usage per Product Row:**
- TV, Stereo, and Speakers in respective locations (e.g.,
row 12-14 for TV, row 15-17 for Stereo, etc.):
```plaintext
Cables Usage | 2 | 1 | 1
```

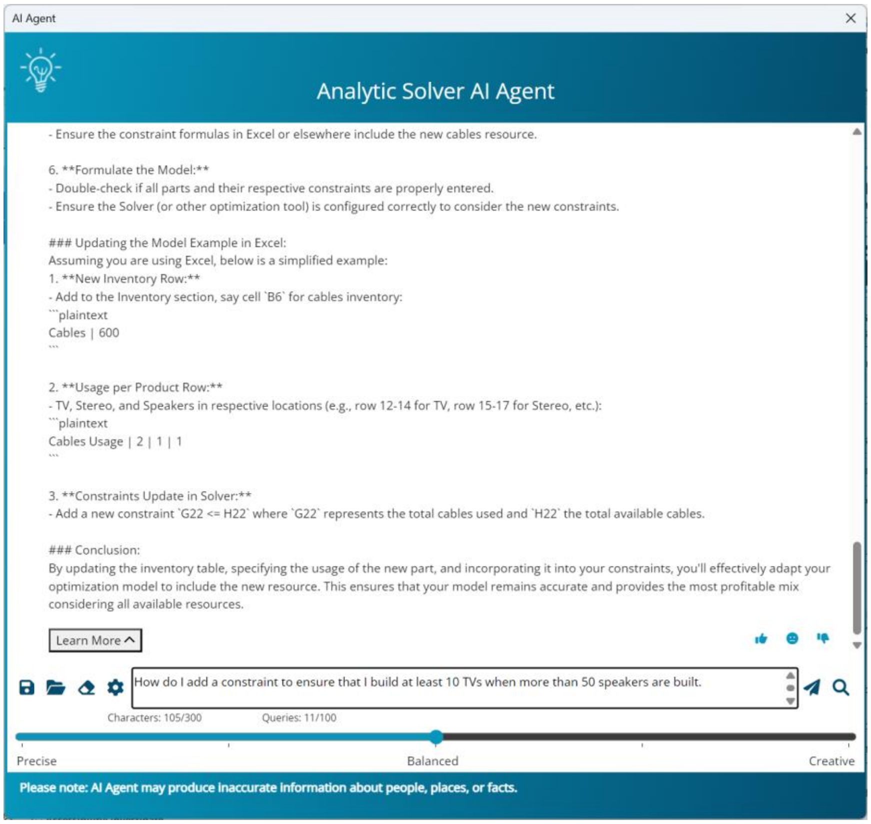
3. **Constraints Update in Solver:**
- Add a new constraint `G22 <= H22` where `G22` represents
the total cables used and `H22` the total available cables.

### Conclusion:
~~~~

```
By updating the inventory table, specifying the usage of
the new part, and incorporating it into your constraints,
you'll effectively adapt your optimization model to include
the new resource. This ensures that your model remains
accurate and provides the most profitable mix considering
all available resources.
```

Assume now that for every 50 speakers manufactured, 10 TVs must also be produced. These kinds of real-world "IF Statements" frequently appear in user models, and their conditions are often not intuitive to implement in a linear fashion. Enter AI Assist!

Enter the promp, "How do I add a constraint to ensure that I build at least 10 TVs when more than 50 speakers are built."



~~~
- Ensure the constraint formulas in Excel or elsewhere include the new cables resource.

6. **Formulate the Model:**
- Double-check if all parts and their respective constraints are properly entered.
- Ensure the Solver (or other optimization tool) is configured correctly to consider the new constraints.

### Updating the Model Example in Excel:
Assuming you are using Excel, below is a simplified example:
1. **New Inventory Row:**
- Add to the Inventory section, say cell `B6` for cables inventory:
```plaintext
Cables | 600
```

2. **Usage per Product Row:**
- TV, Stereo, and Speakers in respective locations (e.g., row 12-14 for TV, row 15-17 for Stereo, etc.):
```plaintext
Cables Usage | 2 | 1 | 1
```

3. **Constraints Update in Solver:**
- Add a new constraint `G22 <= H22` where `G22` represents the total cables used and `H22` the total available cables.

### Conclusion:
By updating the inventory table, specifying the usage of the new part, and incorporating it into your constraints, you'll effectively adapt your optimization model to include the new resource. This ensures that your model remains accurate and provides the most profitable mix considering all available resources.
~~~

How do I add a constraint to ensure that I build at least 10 TVs when more than 50 speakers are built.

Please note: AI Agent may produce inaccurate information about people, places, or facts.

AI Assist Result:

```
To add a constraint ensuring that you build at least 10 TVs
(x1) when more than 50 speakers (x3) are built in your
linear programming model, you can use a combination of
binary variables and a large constant (M). This is a common
approach for creating conditional constraints in linear
programming models.

Here's the step-by-step method to add this constraint:

1. **Introduce a Binary Variable (y):**
- Let \( y \) be a binary variable that indicates whether
more than 50 speakers are being built. \( y \) will be 1 if
more than 50 speakers are built and 0 otherwise.

2. **Add a Big M Constraint:**
- Use a sufficiently large constant \( M \) to enforce the
conditional logic. \( M \) should be larger than the
possible number of speakers that could be built (e.g., 100
here).

3. **Formulate the Conditional Constraint:**
- The constraint that ensures at least 10 TVs are built
```

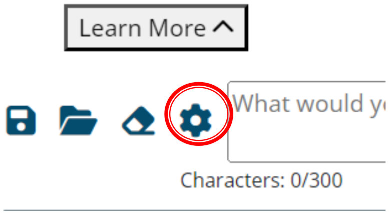
The response includes a step-by-step guide on all the changes that must be incorporated into the model in order to add the new constraint.

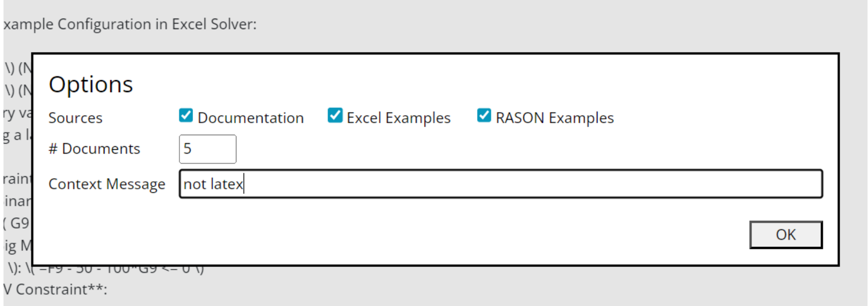
4. Introduce a new binary variable.
5. Add a new constraint that ensures that if the number of speakers built is greater than 50, then the binary variable is 1 and 0 otherwise.

```
when more than 50 speakers are built can be written as:
\[
x1 \geq 10y
\]
- \( x3 - 50 \leq My \)
- This ensures that \( y \) must be 1 when more than 50
speakers are built, triggering the \( x1 \geq 10 \)
constraint.

4. **Implement in Your Solver Model:**
- Add the binary variable \( y \) and the constraints in
your Solver tool.

Here is how you can structure these constraints in general
terms:

1. **Binary Variable Constraint:**
- Add a binary variable constraint:
\[
y \in \{0, 1\}
\]

2. **Relationship between \( y \) and \( x3 \):**
\[
x3 - 50 \leq My
\]

3. **Ensuring At Least 10 TVs if \( y = 1 \):**
\[
x1 \geq 10y
\]

### In Excel or Solver Model Interface
Assuming you are using Excel Solver, follow these steps:

1. **Define a Binary Variable:**
- Assume you add it in cell G9, and it will be called \( y
\).

2. **Add the Constraints:**
a. For \( y \):
- **Cell**: G9
- **Constraint**: Bin (Binary)

b. For the relationship between \( y \) and \( x3 \):
- **Formula Cell**: H9 (Formula: `=F9 - 50 - M*G9 <= 0`)

c. For ensuring at least 10 TVs:
- **Formula Cell**: I9 (Formula: `=D9 - 10*G9 >= 0`)

3. **Integrate into Solver:**
- Go to the Solver Parameters dialog box and add the
constraints created in steps 2a-c.

### Example Configuration in Excel Solver:
If:
- \( x1 \) (Number to Build: TVs) is in cell D9,
- \( x3 \) (Number to Build: Speakers) is in cell F9,
- Binary variable \( y \) is in cell G9,
- Using a large number \( M = 100 \)

Constraints added in Solver:
1. **Binary Constraint for \( y \) (Cell G9)**:
- Set \( G9 \) to be a binary variable.
2. **Big M Constraint**:
- \( H9 \): \( =F9 - 50 - 100*G9 <= 0 \)
3. **TV Constraint**:
- \( I9 \): \( =D9 - 10*G9 >= 0 \)

This configuration ensures that your linear programming
```

```
model will enforce the requirement of building at least 10
TVs if more than 50 speakers are built.
```

Note that the syntax in the results is LaTeX. To turn off this syntax, click the gear icon to open the Options dialog, type "not latex" into the Context Message field and then click OK.





Options

Sources ☑ Documentation ☑ Excel Examples ☑ RASON Examples

Documents: 5

Context Message: not latex

Click OK to view the results in a more readable format.

```
To accommodate a new inventory item,
specifically cables, in your existing linear
programming model, you will need to follow
these steps:

1. **Add the New Inventory Item (Cables) to the
Model:**
- Determine the initial inventory level for
cables.
```

...

As with the AI Agent, toggle between Precise, Balanced, and Creative to determine the type of answers returned. If Precise is selected, AI Agent will attempt to be as exact and deterministic as possible while Creative will usually result in more original, uncertain and non-repetitive answers. Use Balanced (the default) for the best of both worlds.

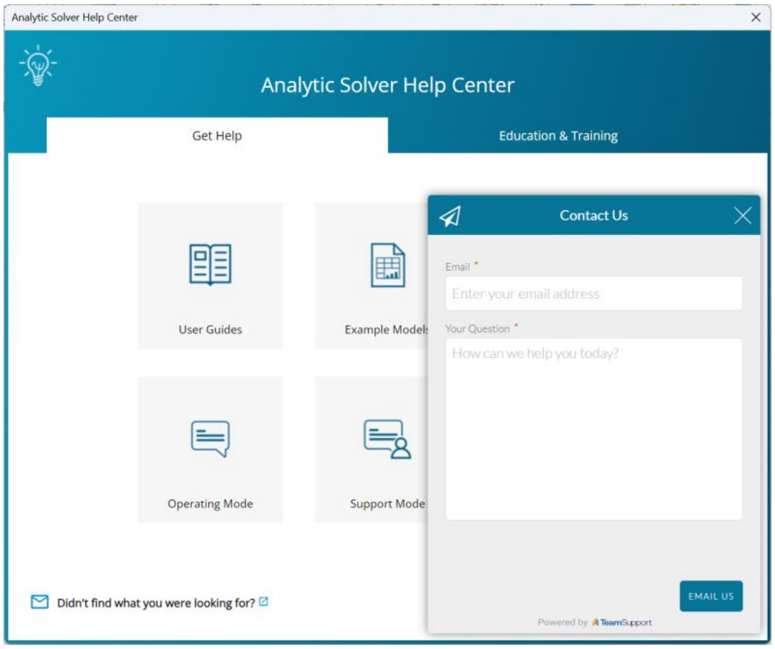
Maximizing Benefits of AI Agent and AI Assist

- Use “New Topic” frequently if your question does not require chat history. Results will be faster and more precise.
- Be as concise as possible in your question, i.e. “How to use RASON?” and “Give me step-by-step instructions on how to use RASON” will produce different results.
- Analytic Solver Upgrade or Solver Academy licenses do not include support for AI Agent. This button will be disabled.

Note: The AI Agent in Frontline Solvers software uses “generative artificial intelligence” methods, which can at times produce erroneous information about people, places or facts, including incorrect information about Frontline’s software products. Users are advised to take this into account, and use “common sense and human intelligence” when conversing with the AI Agent. Frontline Systems Inc. assumes no responsibility for inaccuracies in the AI Agent’s responses.

Analytic Solver Help Center

Click Help – Help Center to open Analytic Solver’s Help Center. Click *Support Live Chat*, in the bottom right hand corner, to open a Live Chat window. If you run into any issues when using the software, the best way to get help is to start a Live Chat with our support specialists This will start a Live Chat during our business hours (or send us a message at other hours), just as if you were to start a Live Chat on www.solver.com – but it saves you *and* our tech support rep a lot of time – because the software reports your latest error message, model diagnosis, license issue or other problem, without you having to type anything or explain verbally what’s happened. You’ll see a dialog like this:

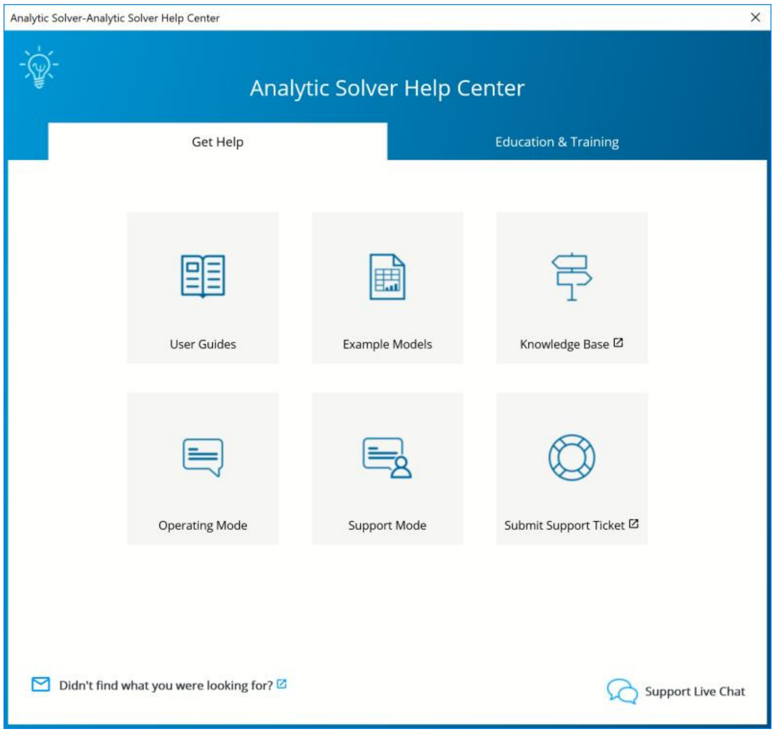


Since the software automatically sends diagnostic information to Tech Support, we can usually identify and resolve the problem faster. (Note: No contents from your actual spreadsheet model is sent, only information such as the number of variables and constraints, last error message, and Excel and Windows version.)

Note: If Support Live Chat is disabled, click the down arrow beneath Help and select *Support Mode – Active Support*.

Accessing Resources

To open Analytic Solver Help, simply click the Help icon on the Analytic Solver ribbon in either App to gain access to video demos, User Guides, online Help, example models, and Website support pages to learn how to use our software tools, and how to build an effective model.



User Guides

Click the User Guides menu choice to open PDF files of the Analytic Solver Optimization and Simulation User and Reference Guides, Analytic Solver Data Science User or Reference Guides, or our Quick Start Guides.

Example Models

Clicking this menu item will open a browser pointing to the workbook, Frontline Example Models Overview.xlsx, located at C:\ProgramData\Frontline Systems\ Examples. See the table below for a description of each example dataset.

By going to **Help – Welcome Screen** you can easily access a range of support and training resources.

Knowledge Base

Click Knowledge Base to peruse a multitude of online articles related to support and installation issues or to locate articles that will help you to quickly build accurate, efficient optimization, simulation, and data science models.

Operating Mode

Click Operating Mode to switch between three different levels of help. The Excel formulas and functions you use in your model have a huge impact on how fast it runs and how well it solves. If you learn more about this, you can get better results, but if you don't, your results will be limited. Guided Mode can help you learn.

- Guided Mode prompts you step-by-step when solving, with dialogs.

- Auto-Help Mode shows dialogs or Help only when there’s a problem or error condition.
- Expert Mode provides only messages in the Task Pane Output tab. (This mode not supported when using a trial license.)

Support Mode

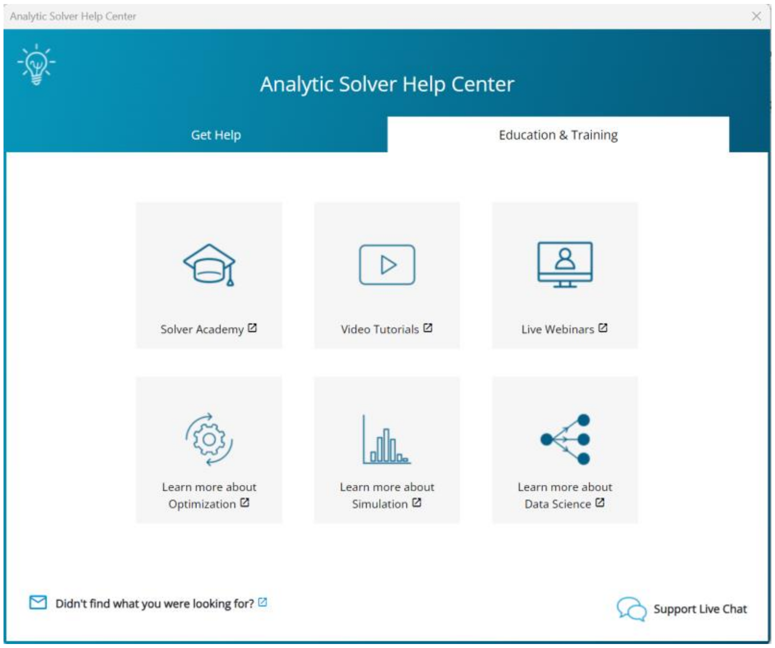
Click Support Mode to switch between three different levels of support. No information (cell contents etc.) from your Excel model is ever reported automatically to Frontline Systems, in any of these Support Modes. Only events in Frontline's software, such as menu selections, Solver Result messages, or error messages are reported.

- **Active Support** automatically reports events, errors and problems to Frontline Support, receives and displays messages to you from Support, and allows you to start a Live Chat with Support while working in Excel (**Recommended**).Auto-Help Mode shows dialogs or Help only when there’s a problem or error condition.
- **Standard Support** automatically reports events, errors and problems anonymously (not associated with you) to Frontline Support, but does not provide a means to receive messages or start a Live Chat with Support.
- **Basic Support** provides no automatic connection to Frontline Support. You will have to contact Frontline Support manually via email, website or phone if you need help.

Submit a Support Ticket

If you're having installation, technical, or modeling issues, submit a Support Ticket to open an online support request form. Submit your email address and a short, concise description of the issue that you are experiencing. You'll receive a reply from one of Frontline's highly trained Support Specialists within 24 hours, and generally much sooner.

Our technical support service is designed to supplement your own efforts: Getting you over stumbling blocks, pointing out relevant sections of our User Guides or example models, helping you fix a modeling error, or -- in rare cases -- working around an issue with our software (always at our expense).


Analytic Solver Help Center
Analytic Solver Help Center
Get Help
Education & Training
Solver Academy
Video Tutorials
Live Webinars
Learn more about Optimization
Learn more about Simulation
Learn more about Data Science
Didn't find what you were looking for?
Support Live Chat

Solver Academy

Solver Academy is Frontline Systems' own learning platform. It's the place where business analysts can gain expertise in advanced analytics: forecasting, data science, text mining, mathematical optimization, simulation and risk analysis, and stochastic optimization.

Video Tutorials/Live Webinars

Click Video Tutorials to be directed to Frontline's You Tube Channel. Browse videos on how to create an optimization or simulation model or construct a data science or prediction model using Analytic Solver.

Click Live Webinars to be redirected to www.solver.com to join a live or pre-recorded webinar. Topics include *Using Analytic Solver Data* Science *to Gain Insights from your Excel Data*, *Overview of Monte Carlo Simulation Applications*, *Applications of Optimization in Excel*, etc.

Learn more!

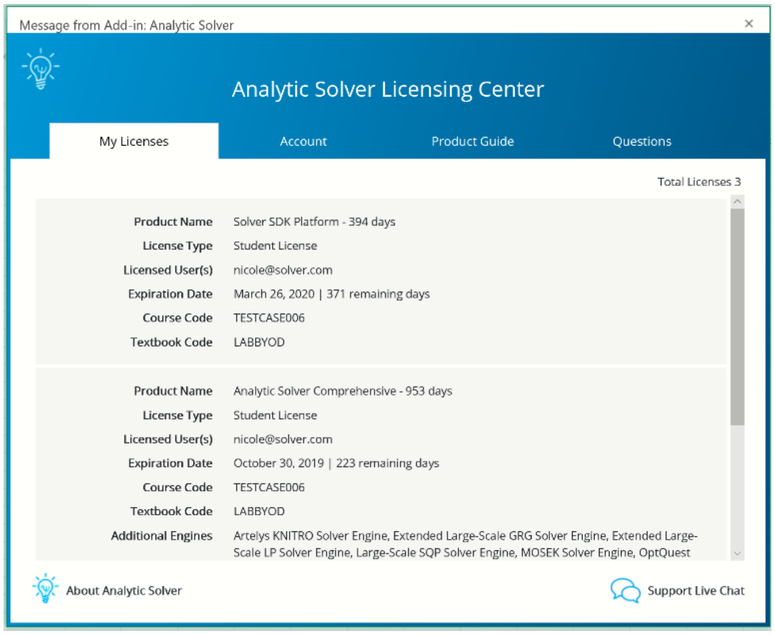
Click any of the three Learn More buttons to learn more about how you can solve large-scale optimization, simulation, and data science models, reduce costs, quantify and mitigate risk, and create forecasting, data science and text mining models using Analytic Solver.

License Button

Click the License button to open the License Manager where you can manage your current licenses and accounts, open our Product Selection Wizard, connect to Live Chat or peruse through a list of FAQs.

Click License – Manage License/Manage Account to open the Licensing Center.

The "MyLicenses" tab displays your current license and license type, along with the expiration date.

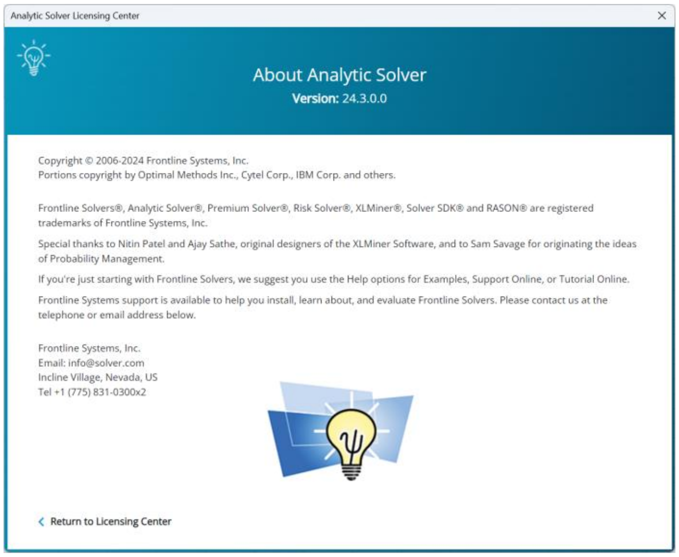


Total Licenses 3

Product Name	Solver SDK Platform - 394 days
License Type	Student License
Licensed User(s)	nicole@solver.com
Expiration Date	March 26, 2020 | 371 remaining days
Course Code	TESTCASE006
Textbook Code	LABBYOD

Product Name	Analytic Solver Comprehensive - 953 days
License Type	Student License
Licensed User(s)	nicole@solver.com
Expiration Date	October 30, 2019 | 223 remaining days
Course Code	TESTCASE006
Textbook Code	LABBYOD
Additional Engines	Artelys KNITRO Solver Engine, Extended Large-Scale GRG Solver Engine, Extended Large-Scale LP Solver Engine, Large-Scale SQP Solver Engine, MOSEK Solver Engine, OptQuest

Click *About Analytic Solver* to open the following dialog containing information on this release.



Version: 24.3.0.0

Copyright © 2006-2024 Frontline Systems, Inc.
Portions copyright by Optimal Methods Inc., Cytel Corp., IBM Corp. and others.

Frontline Solvers®, Analytic Solver®, Premium Solver®, Risk Solver®, XLMiner®, Solver SDK® and RASON® are registered trademarks of Frontline Systems, Inc.

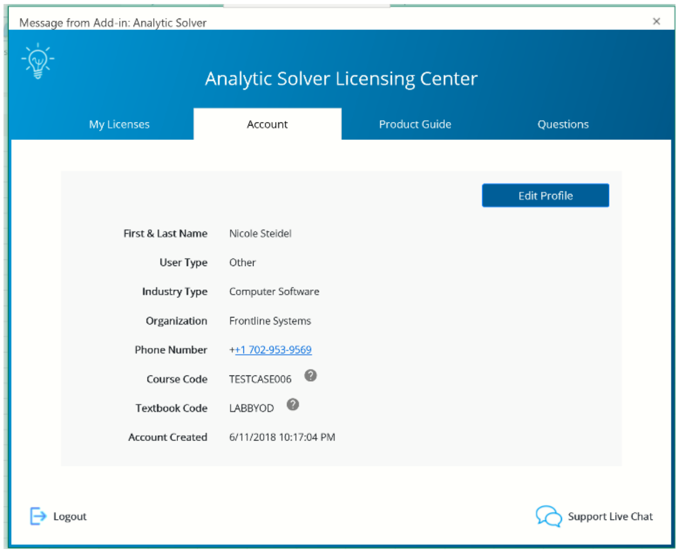
Special thanks to Nitin Patel and Ajay Sathe, original designers of the XLMiner Software, and to Sam Savage for originating the ideas of Probability Management.

If you're just starting with Frontline Solvers, we suggest you use the Help options for Examples, Support Online, or Tutorial Online.

Frontline Systems support is available to help you install, learn about, and evaluate Frontline Solvers. Please contact us at the telephone or email address below.

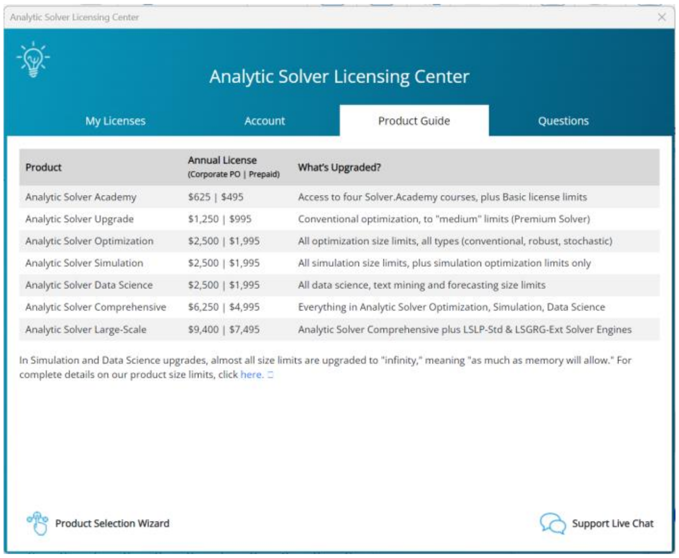
Frontline Systems, Inc.
Email: info@solver.com
Incline Village, Nevada, US
Tel +1 (775) 831-0300x2

Click the Account tab to view your account on www.solver.com. Click Edit Profile to edit the information. Click Live Chat to open a Live Chat window or Log Out to log out of the product.



First & Last Name	Nicole Steidel
User Type	Other
Industry Type	Computer Software
Organization	Frontline Systems
Phone Number	++1 702-953-9569
Course Code	TESTCASE006
Textbook Code	LABBYOD
Account Created	6/11/2018 10:17:04 PM

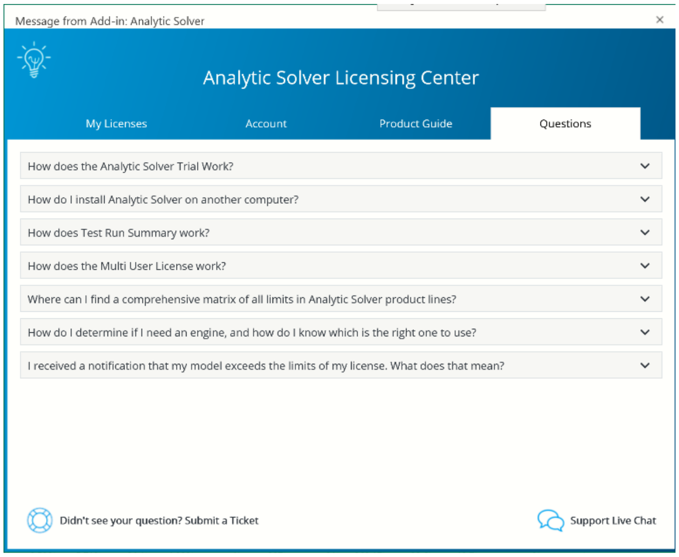
Click the Product Guide tab to view a list of products and pricing information. Click Product Selection Wizard to open the Product Selection Wizard. See the next section for information on this feature.



Product	Annual License (Corporate PO | Prepaid)	What's Upgraded?
Analytic Solver Academy	$625 | $495	Access to four Solver.Academy courses, plus Basic license limits
Analytic Solver Upgrade	$1,250 | $995	Conventional optimization, to "medium" limits (Premium Solver)
Analytic Solver Optimization	$2,500 | $1,995	All optimization size limits, all types (conventional, robust, stochastic)
Analytic Solver Simulation	$2,500 | $1,995	All simulation size limits, plus simulation optimization limits only
Analytic Solver Data Science	$2,500 | $1,995	All data science, text mining and forecasting size limits
Analytic Solver Comprehensive	$6,250 | $4,995	Everything in Analytic Solver Optimization, Simulation, Data Science
Analytic Solver Large-Scale	$9,400 | $7,495	Analytic Solver Comprehensive plus LSLP-Std & LSGRG-Ext Solver Engines

In Simulation and Data Science upgrades, almost all size limits are upgraded to "infinity," meaning "as much as memory will allow." For complete details on our product size limits, click here.

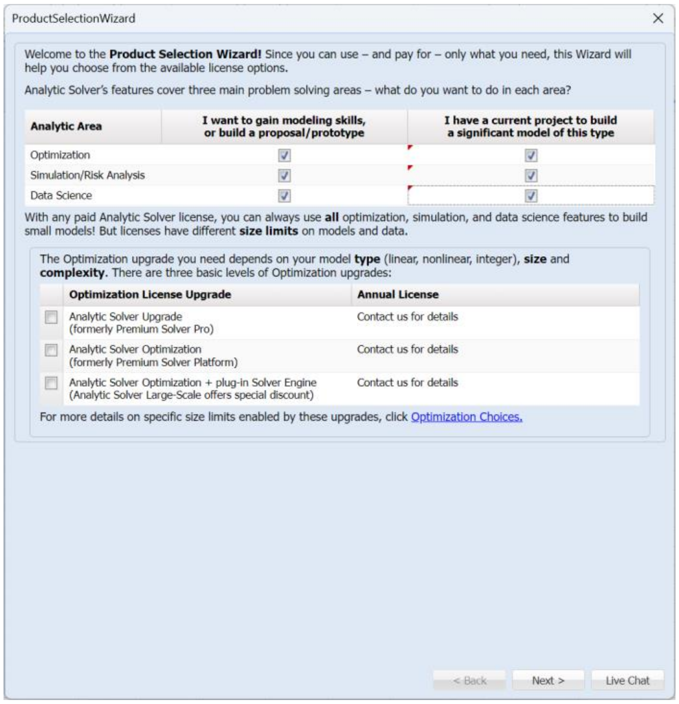
Click the Questions tab to review a list of FAQs, submit a support ticket or start a live chat.


Message from Add-in: Analytic Solver
Analytic Solver Licensing Center
My Licenses
Account
Product Guide
Questions

How does the Analytic Solver Trial Work?
How do I install Analytic Solver on another computer?
How does Test Run Summary work?
How does the Multi User License work?
Where can I find a comprehensive matrix of all limits in Analytic Solver product lines?
How do I determine if I need an engine, and how do I know which is the right one to use?
I received a notification that my model exceeds the limits of my license. What does that mean?

Product Selection Wizard

Select **Product Selection Wizard** from the Product Guide tab in the Licensing Center to open a series of dialogs that will help you determine which product will best meet your needs based on your recent pattern of use.



Welcome to the **Product Selection Wizard!** Since you can use – and pay for – only what you need, this Wizard will help you choose from the available license options.

Analytic Solver's features cover three main problem solving areas – what do you want to do in each area?

Analytic Area	I want to gain modeling skills, or build a proposal/prototype	I have a current project to build a significant model of this type
Optimization	☑	☑
Simulation/Risk Analysis	☑	☑
Data Science	☑	☑

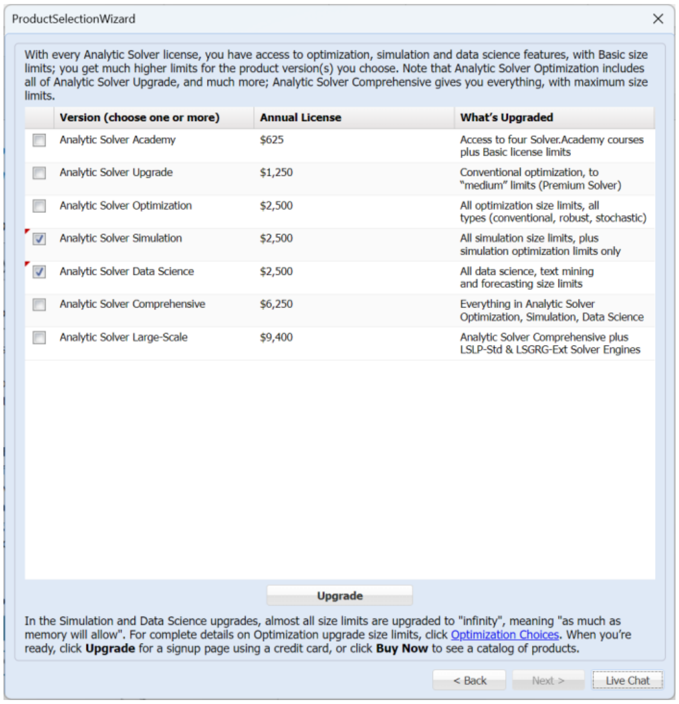
With any paid Analytic Solver license, you can always use **all** optimization, simulation, and data science features to build small models! But licenses have different **size limits** on models and data.

The Optimization upgrade you need depends on your model **type** (linear, nonlinear, integer), **size** and **complexity**. There are three basic levels of Optimization upgrades:

	Optimization License Upgrade	Annual License
☐	Analytic Solver Upgrade (formerly Premium Solver Pro)	Contact us for details
☐	Analytic Solver Optimization (formerly Premium Solver Platform)	Contact us for details
☐	Analytic Solver Optimization + plug-in Solver Engine (Analytic Solver Large-Scale offers special discount)	Contact us for details

For more details on specific size limits enabled by these upgrades, click Optimization Choices.

Select the Product that you'd like to purchase and then click **Next**. Click the *Optimization Choices* link to learn more about Analytic Solver products that can solve optimization models and to find more information on speed, memory, and the use of plug-in Solver Engines.



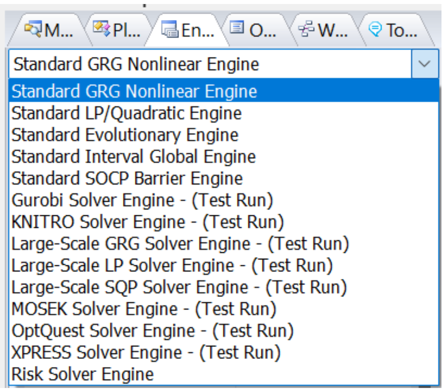
With every Analytic Solver license, you have access to optimization, simulation and data science features, with Basic size limits; you get much higher limits for the product version(s) you choose. Note that Analytic Solver Optimization includes all of Analytic Solver Upgrade, and much more; Analytic Solver Comprehensive gives you everything, with maximum size limits.

	Version (choose one or more)	Annual License	What's Upgraded
☐	Analytic Solver Academy	$625	Access to four Solver.Academy courses plus Basic license limits
☐	Analytic Solver Upgrade	$1,250	Conventional optimization, to "medium" limits (Premium Solver)
☐	Analytic Solver Optimization	$2,500	All optimization size limits, all types (conventional, robust, stochastic)
☑	Analytic Solver Simulation	$2,500	All simulation size limits, plus simulation optimization limits only
☑	Analytic Solver Data Science	$2,500	All data science, text mining and forecasting size limits
☐	Analytic Solver Comprehensive	$6,250	Everything in Analytic Solver Optimization, Simulation, Data Science
☐	Analytic Solver Large-Scale	$9,400	Analytic Solver Comprehensive plus LSLP-Std & LSGRG-Ext Solver Engines

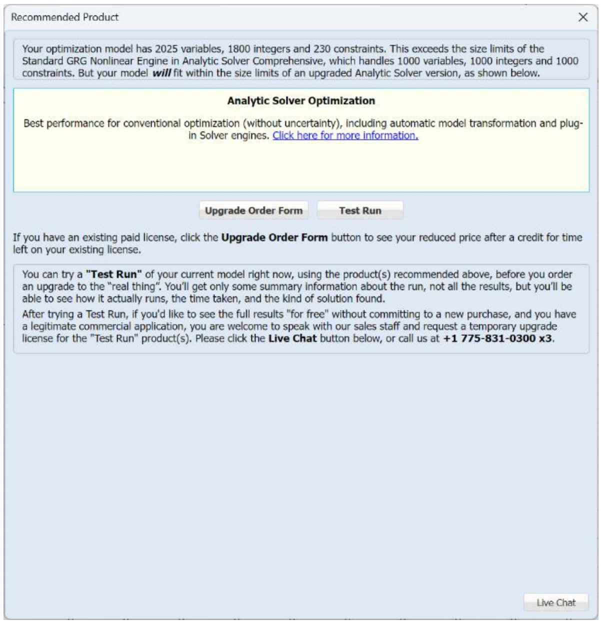
In the Simulation and Data Science upgrades, almost all size limits are upgraded to "infinity", meaning "as much as memory will allow". For complete details on Optimization upgrade size limits, click Optimization Choices. When you're ready, click **Upgrade** for a signup page using a credit card, or click **Buy Now** to see a catalog of products.

On this screen, the Product Selection Wizard will recommend a product or products based on your answers on the previous screens. Click Upgrade to purchase the recommended product. Click the *Optimization Choices* link to learn more about Analytic Solver products that can solve optimization models. If at any time you'd like to chat with a member of our Technical Support staff, click **Live Chat**. Or if you'd like to amend your answers on a previous dialog, click **Back**.

When you run a simulation or optimization model that contains too many decision/uncertain variables or constraints/uncertain functions for the selected engine, the Product Wizard will automatically appear and recommend a product that *can* solve your model.



M...
Pl...
En...
O...
W...
To...
Standard GRG Nonlinear Engine
Standard GRG Nonlinear Engine
Standard LP/Quadratic Engine
Standard Evolutionary Engine
Standard Interval Global Engine
Standard SOCP Barrier Engine
Gurobi Solver Engine - (Test Run)
KNITRO Solver Engine - (Test Run)
Large-Scale GRG Solver Engine - (Test Run)
Large-Scale LP Solver Engine - (Test Run)
Large-Scale SQP Solver Engine - (Test Run)
MOSEK Solver Engine - (Test Run)
OptQuest Solver Engine - (Test Run)
XPRESS Solver Engine - (Test Run)
Risk Solver Engine



Your optimization model has 2025 variables, 1800 integers and 230 constraints. This exceeds the size limits of the Standard GRG Nonlinear Engine in Analytic Solver Comprehensive, which handles 1000 variables, 1000 integers and 1000 constraints. But your model ***will*** fit within the size limits of an upgraded Analytic Solver version, as shown below.

Analytic Solver Optimization

Best performance for conventional optimization (without uncertainty), including automatic model transformation and plug-in Solver engines. Click here for more information.

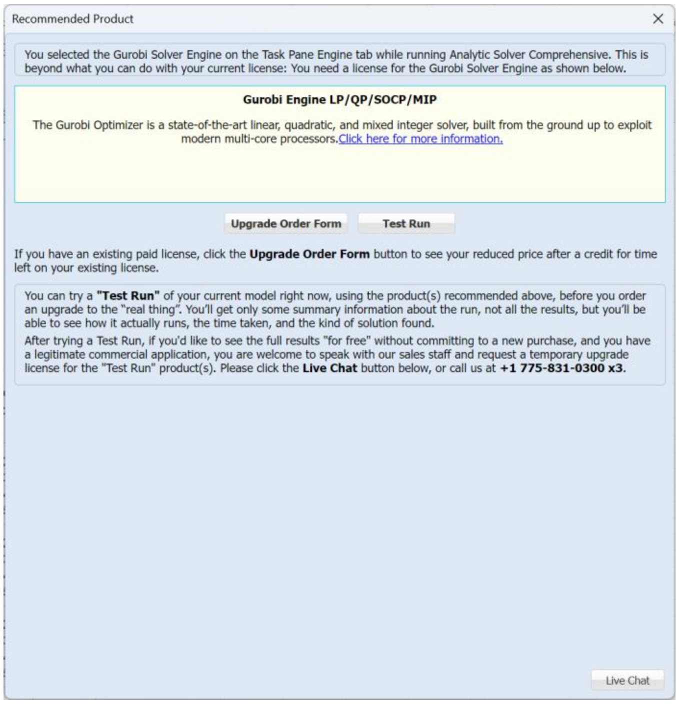
If you have an existing paid license, click the **Upgrade Order Form** button to see your reduced price after a credit for time left on your existing license.

You can try a **"Test Run"** of your current model right now, using the product(s) recommended above, before you order an upgrade to the "real thing". You'll get only some summary information about the run, not all the results, but you'll be able to see how it actually runs, the time taken, and the kind of solution found.

After trying a Test Run, if you'd like to see the full results "for free" without committing to a new purchase, and you have a legitimate commercial application, you are welcome to speak with our sales staff and request a temporary upgrade license for the "Test Run" product(s). Please click the **Live Chat** button below, or call us at **+1 775-831-0300 x3**.

When you click “Test Run”, the Product Wizard will immediately run the optimization or simulation model using the recommended product.   (Only summary information will be available.) At this point, you can purchase the recommended product(s), or close the dialog.

This same behavior will also occur when solving smaller models, if you select a specific external engine, from the Engine drop down menu on the Engine tab of the Solver Task Pane, for which you do not have a license. The Product Wizard will recommend the selected engine, and allow you to solve your model using this engine. Once Solver has finished solving, you will have the option to purchase the product.



You selected the Gurobi Solver Engine on the Task Pane Engine tab while running Analytic Solver Comprehensive. This is beyond what you can do with your current license: You need a license for the Gurobi Solver Engine as shown below.

Gurobi Engine LP/QP/SOCP/MIP

The Gurobi Optimizer is a state-of-the-art linear, quadratic, and mixed integer solver, built from the ground up to exploit modern multi-core processors.Click here for more information.

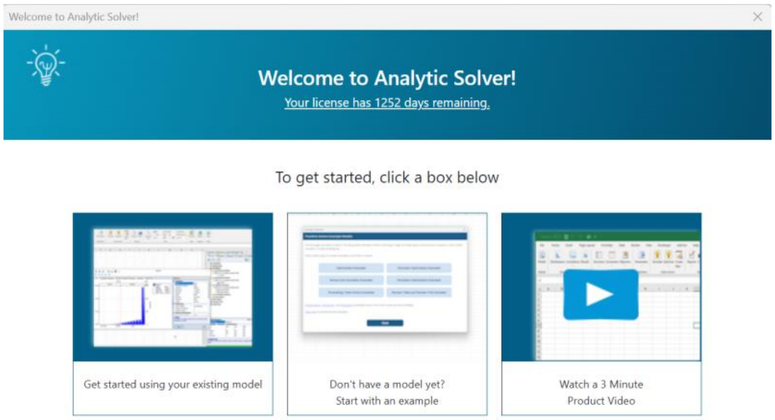
If you have an existing paid license, click the **Upgrade Order Form** button to see your reduced price after a credit for time left on your existing license.

You can try a **"Test Run"** of your current model right now, using the product(s) recommended above, before you order an upgrade to the "real thing". You'll get only some summary information about the run, not all the results, but you'll be able to see how it actually runs, the time taken, and the kind of solution found.

After trying a Test Run, if you'd like to see the full results "for free" without committing to a new purchase, and you have a legitimate commercial application, you are welcome to speak with our sales staff and request a temporary upgrade license for the "Test Run" product(s). Please click the **Live Chat** button below, or call us at **+1 775-831-0300 x3**.

Using the Welcome Screen in Analytic Solver Desktop

When using Analytic Solver Desktop, the first time you click the Analytic Solver tab on the Ribbon, you’ll see the Welcome Screen, pictured on the next page. You can display the Welcome Screen manually by choosing **Help – Welcome Screen** from the Ribbon.

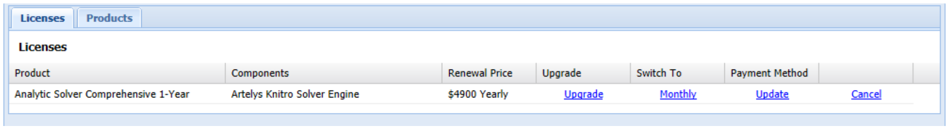

Welcome to Analytic Solver!
Welcome to Analytic Solver!
Your license has 1252 days remaining.
To get started, click a box below
Get started using your existing model
Don't have a model yet? Start with an example
Watch a 3 Minute Product Video

The Welcome Screen was considerably enhanced in V2017-R2. As in earlier releases, it reports the status of your license, and provides links to Example Models and a quick 3 minute instructional video. The **license link** takes you directly to the License / Subscription Manager (also available from the License drop-down menu).

License/Subscription in Analytic Solver Desktop

Since Frontline Solvers V2015, our products have been offered on an annual subscription basis, including the software license and Annual Support (technical support and software upgrades). But in order to renew your subscription, or upgrade or change the product you were using, you had to talk to someone in Frontline Systems sales or customer service, who would make the change, and email you a new license code that you'd copy and paste into a dialog box.

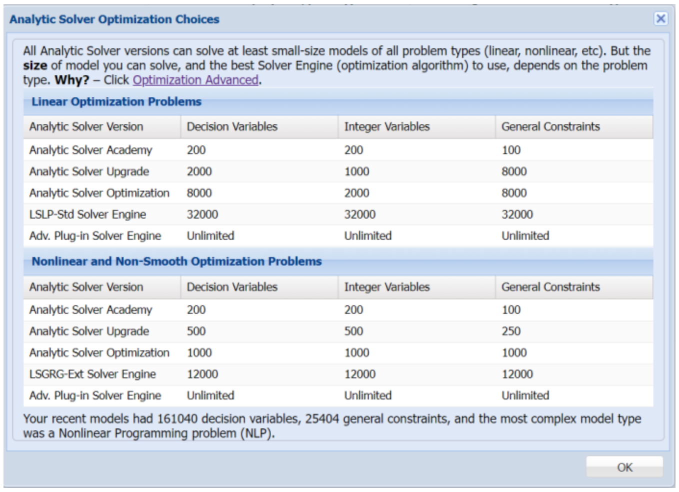
Starting in Frontline Solvers V2018, all those steps have been *automated*: You can review your subscription status, renew, upgrade or downgrade your subscription, or add plug-in Solver Engines, all 'self-service' using the License / Subscription Manager – accessible via the blue button in the Welcome Screen, **or Options - License / Subscription** on the Analytic Solver Ribbon. The next page shows a example of the License / Subscription Manager dialog.



Licenses

Product	Components	Renewal Price	Upgrade	Switch To	Payment Method	
Analytic Solver Comprehensive 1-Year	Artelys Knitro Solver Engine	$4900 Yearly	Upgrade	Monthly	Update	Cancel

The **Products** tab in the License / Subscription Manager dialog shows you a list of the products that are 'upgrade options' for your current subscription. You can click the hyperlinks for more information about the products on Solver.com, or click the buttons to go step-by-step through the upgrade process.



All Analytic Solver versions can solve at least small-size models of all problem types (linear, nonlinear, etc). But the **size** of model you can solve, and the best Solver Engine (optimization algorithm) to use, depends on the problem type. **Why?** – Click Optimization Advanced.

Linear Optimization Problems

Analytic Solver Version	Decision Variables	Integer Variables	General Constraints
Analytic Solver Academy	200	200	100
Analytic Solver Upgrade	2000	1000	8000
Analytic Solver Optimization	8000	2000	8000
LSLP-Std Solver Engine	32000	32000	32000
Adv. Plug-in Solver Engine	Unlimited	Unlimited	Unlimited

Nonlinear and Non-Smooth Optimization Problems

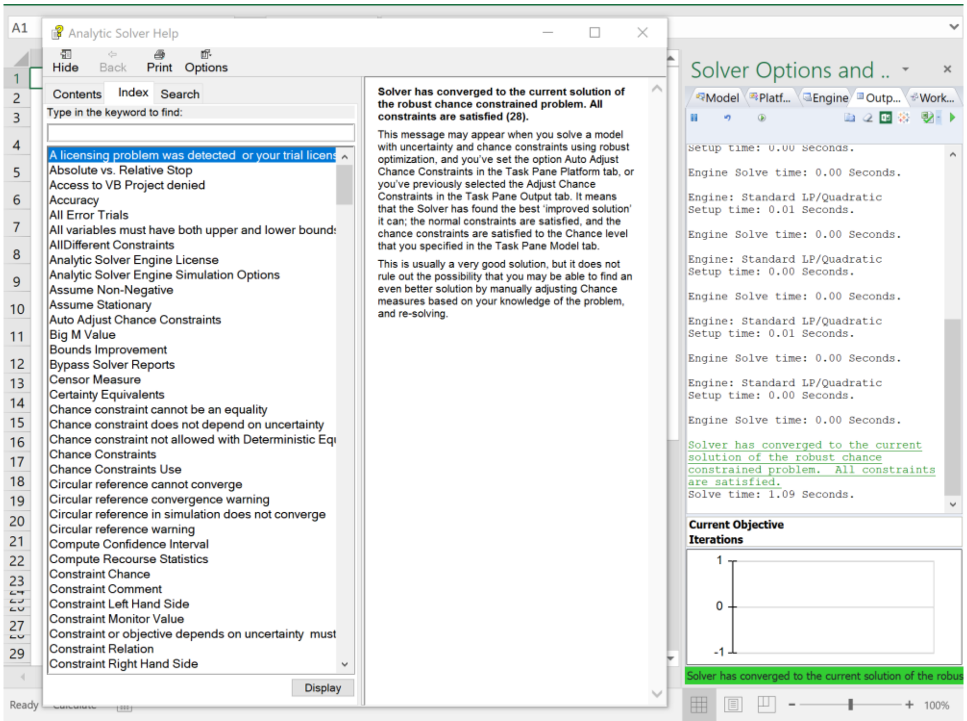
Analytic Solver Version	Decision Variables	Integer Variables	General Constraints
Analytic Solver Academy	200	200	100
Analytic Solver Upgrade	500	500	250
Analytic Solver Optimization	1000	1000	1000
LSGRG-Ext Solver Engine	12000	12000	12000
Adv. Plug-in Solver Engine	Unlimited	Unlimited	Unlimited

Your recent models had 161040 decision variables, 25404 general constraints, and the most complex model type was a Nonlinear Programming problem (NLP).

Context-Sensitive Help in Analytic Solver Desktop

Analytic Solver Desktop has a significant amount of context-sensitive Help. You can quickly access Help on any Solver Result message that appears in the Task Pane Output tab, any option that appears on the Engine tab or Platform tab, or any element of your model that appears on the Model tab.

Solver Result messages appear as underlined links – you can simply click them. For example, here's the Help that appears when you click the Solver Results message for the Gas Company Chance example in StochasticExamples.xlsx:


A1
Analytic Solver Help
Hide
Back
Print
Options
Contents
Index
Search
Type in the keyword to find:
A licensing problem was detected or your trial licens
Absolute vs. Relative Stop
Access to VB Project denied
Accuracy
All Error Trials
All variables must have both upper and lower bound
AllDifferent Constraints
Analytic Solver Engine License
Analytic Solver Engine Simulation Options
Assume Non-Negative
Assume Stationary
Auto Adjust Chance Constraints
Big M Value
Bounds Improvement
Bypass Solver Reports
Censor Measure
Certainty Equivalents
Chance constraint cannot be an equality
Chance constraint does not depend on uncertainty
Chance constraint not allowed with Deterministic Eq
Chance Constraints
Chance Constraints Use
Circular reference cannot converge
Circular reference convergence warning
Circular reference in simulation does not converge
Circular reference warning
Compute Confidence Interval
Compute Recourse Statistics
Constraint Chance
Constraint Comment
Constraint Left Hand Side
Constraint Monitor Value
Constraint or objective depends on uncertainty must
Constraint Relation
Constraint Right Hand Side
Display
Solver Options and ..
Model
Platf...
Engine
Outp...
Work...
Current Objective
Iterations
1
0
-1
Ready
100%

Solver has converged to the current solution of the robust chance constrained problem. All constraints are satisfied (28).

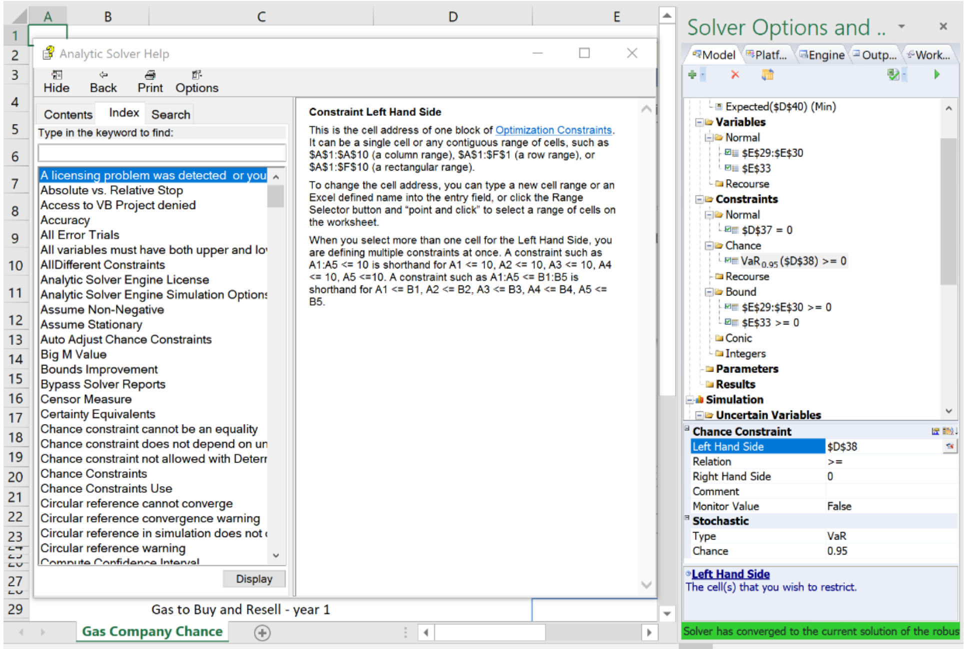
This message may appear when you solve a model with uncertainty and chance constraints using robust optimization, and you've set the option Auto Adjust Chance Constraints in the Task Pane Platform tab, or you've previously selected the Adjust Chance Constraints in the Task Pane Output tab. It means that the Solver has found the best 'improved solution' it can; the normal constraints are satisfied, and the chance constraints are satisfied to the Chance level that you specified in the Task Pane Model tab.

This is usually a very good solution, but it does not rule out the possibility that you may be able to find an even better solution by manually adjusting Chance measures based on your knowledge of the problem, and re-solving.

```
Setup time: 0.00 Seconds.

Engine Solve time: 0.00 Seconds.

Engine: Standard LP/Quadratic
Setup time: 0.01 Seconds.

Engine Solve time: 0.00 Seconds.

Engine: Standard LP/Quadratic
Setup time: 0.00 Seconds.

Engine Solve time: 0.00 Seconds.

Engine: Standard LP/Quadratic
Setup time: 0.01 Seconds.

Engine Solve time: 0.00 Seconds.

Engine: Standard LP/Quadratic
Setup time: 0.00 Seconds.

Engine Solve time: 0.00 Seconds.

Solver has converged to the current
solution of the robust chance
constrained problem.  All constraints
are satisfied.
Solve time: 1.09 Seconds.
```

Solver has converged to the current solution of the robus

To access Help on any Platform option, Engine option, or Model element, click the underlined link in the brief explanation that appears at the bottom of the Task Pane. For example, here's the Help that appears when you click the Type field link for a chance constraint in your model:



Constraint Left Hand Side

This is the cell address of one block of Optimization Constraints. It can be a single cell or any contiguous range of cells, such as $A$1:$A$10 (a column range), $A$1:$F$1 (a row range), or $A$1:$F$10 (a rectangular range).

To change the cell address, you can type a new cell range or an Excel defined name into the entry field, or click the Range Selector button and "point and click" to select a range of cells on the worksheet.

When you select more than one cell for the Left Hand Side, you are defining multiple constraints at once. A constraint such as A1:A5 <= 10 is shorthand for A1 <= 10, A2 <= 10, A3 <= 10, A4 <= 10, A5 <=10. A constraint such as A1:A5 <= B1:B5 is shorthand for A1 <= B1, A2 <= B2, A3 <= B3, A4 <= B4, A5 <= B5.

Chance Constraint	
Left Hand Side	$D$38
Relation	>=
Right Hand Side	0
Comment	
Monitor Value	False
Stochastic	
Type	VaR
Chance	0.95

Left Hand Side
The cell(s) that you wish to restrict.

Solver has converged to the current solution of the robus

To access help in Analytic Solver Cloud, click Help – Help Center.

Examples: Conventional Optimization

Introduction

This chapter introduces conventional optimization (with no uncertainty) in both the Desktop and Cloud apps using Analytic Solver Comprehensive, Analytic Solver Optimization and Analytic Solver Upgrade with a series of examples. Except as noted in specific sections, all of these examples can be used in any of the above product subsets.

- “A First Optimization Model” takes you step-by-step through the process of creating an optimization model from scratch. It illustrates use of the Ribbon and Task Pane, and its relationship to the Excel Solver Parameters dialog.
- “Introducing the Standard Examples,” “Linear Programming Examples” and “Nonlinear Optimization Examples” takes you through the workbook **StandardExamples.xlsx**, with seven examples that illustrate a number of Analytic Solver features.

A First Optimization Model

This section is meant for you if you’re familiar with optimization, but you’ve never used Excel Solver or Analytic Solver. It shows how you can translate from the algebraic statement of an optimization problem to a spreadsheet Solver model. If *optimization* is new to you, consult the chapter “Mastering Conventional Optimization Concepts.”

Note: This first step-by-step example is a ‘quick and dirty’ approach that can be used to solve the example problem, but is not well documented or easy to maintain. If using Analytic Solver Desktop or Cloud you'll have access to Microsoft Excel's many features that can help you organize and display the structure of your model, through tools such as defined names, formatting and outlining. The models in the **StandardExamples.xlsx** workbook illustrate some of these features. This workbook is normally installed into C:\ProgramData\Frontline Systems\Examples. As models become larger, the problems of managing data for constraints, coefficients, and so on become more significant, and a properly organized spreadsheet model can help manage this complexity.

Setting Up a Model

To set up an optimization model in a spreadsheet either in Microsoft Excel or Excel Online, you will follow these essential steps:

1. Reserve a cell to hold the value of each decision variable. We’ll use A1:A3 in the example below.
2. Pick a cell to represent the objective function, and enter a formula that calculates the objective function value in this cell. We’ll use A4 in the example below.

3. Pick other cells and use them to enter the formulas that calculate the left hand sides of the constraints. We'll use B1:B5 in the example below.
4. The constraint right hand sides can be entered as numbers in other cells, or entered directly in the Solver's Add Constraint dialog. We'll use C1:C5 for this purpose in the example below.

Within this overall framework, you have a great deal of flexibility in choosing cells to use, and calculating the objective and constraints. For example, the objective function will ultimately depend on the decision variable cells, but you don't have to calculate the entire function in one formula cell. You can use any number of formula cells to compute intermediate results, and use these to calculate the objective function or the constraints.

In desktop Excel and Excel Online, you can write a linear expression easily with Excel's SUMPRODUCT function, or you can use + and * operators, as shown below. You can use Excel's *array* formulas, and Excel functions that return vector and matrix results. And you can use Excel's rich facilities to access data in external text files, Web pages, and relational and multidimensional databases to 'populate' your model.

The Model in Algebraic and Spreadsheet Form

Consider the following problem. Our factory is building three products: TV sets, stereos and speakers. Each product is assembled from parts in inventory, and there are five types of parts: chassis, picture tubes, speaker cones, power supplies and electronics units. Our goal is to produce the mix of products that will maximize profits, given the inventory of parts on hand. This is a simple linear programming problem, also used to illustrate other ideas later in "Linear Programming Examples."

The Algebraic Form

The problem can be described in algebraic form as follows. The decision variables are the number of products of each type to build: x_1 for TV sets, x_2 for stereos and x_3 for speakers. There is a fixed profit per unit for each product, so the objective function (the quantity we want to maximize) is:

Maximize $75 x_1 + 50 x_2 + 35 x_3$ (Profit)

Building each product requires a certain number of parts of each type. For example, TV sets and stereos each require one chassis, but speakers don't use one. The number of parts used depends on the mix of products built (the left hand side of each constraint), and we have a limited number of parts of each type on hand (the corresponding constraint right hand side):

Subject to $1 x_1 + 1 x_2 + 0 x_3 <= 400$ (Chassis)
$1 x_1 + 0 x_2 + 0 x_3 <= 200$ (Picture tubes)
$2 x_1 + 2 x_2 + 1 x_3 <= 800$ (Speaker cones)
$1 x_1 + 1 x_2 + 0 x_3 <= 400$ (Power supplies)
$2 x_1 + 1 x_2 + 1 x_3 <= 600$ (Electronics)

Since the number of products built must be nonnegative, we also have the constraints $x_1, x_2, x_3 >= 0$. Note that terms like $0 x_3$ are included purely to show the structure of the model – they can be either omitted or included when entering formulas in Excel.

The Spreadsheet Formulas

The fastest (though not necessarily the best) way to lay out this problem on the spreadsheet is to pick (for example) cell A1 for x_1, cell A2 for x_2 and cell A3 for x_3. Then the objective can be entered in cell A4, much like the algebraic form above:

A4: =75*A1+50*A2+35*A3

We can go on to enter a formula in (say) cell B1 for the first constraint left hand side (Chassis) as =1*A1+1*A2+0*A3, or the simpler form:

B1: =A1+A2

Similarly, we can use cell B2 for the formula =A1 (Picture tubes), B3 for the formula =2*A1+2*A2+A3 (Speaker cones), B4 for the formula =A1+A2 (Power supplies), and B5 for the formula =2*A1+A2+A3 (Electronics):

B2: =A1
B3: =2*A1+2*A2+A3
B4: =A1+A2
B5: =2*A1+A2+A3

Finally, we'll enter the constraint right hand side values in cells C1:C5:

C1: 400
C2: 200
C3: 800
C4: 400
C5: 600

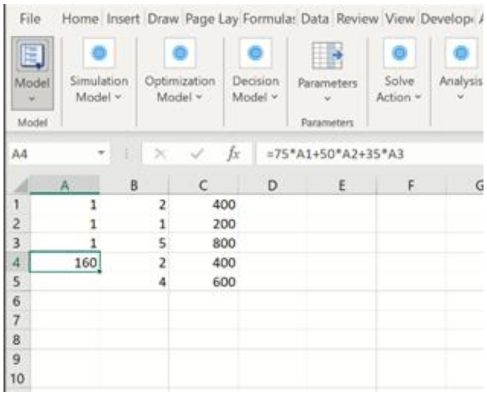
We now have a simple spreadsheet model that we can use to play 'what if.' For any values we enter for the decision variables in cells A1, A2 and A3, the objective (Total Profit) and the corresponding values of the constraint left hand sides (the numbers of parts used) will be calculated.

We want to turn this 'what if' spreadsheet model into an optimization model, where the Solver will automatically find *optimal* values for the cells A1:A3, so that the objective function at A4 is maximized, and the constraints are satisfied.

Defining and Solving the Optimization Model

To begin, click the Analytic Solver tab on the Ribbon. Your worksheet with an empty **Task Pane** ("Solver Options and Model Specifications") **Model tab** should appear, as shown on the next page.

To define the optimization problem, we'll use the Ribbon to point out to the Solver (i) the cells that we've reserved for the decision variables, (ii) the cell that calculates the value of the objective function, and (iii) the cells that calculate the constraint left hand sides. We'll also enter values for the constraint right hand sides, and non-negativity constraints on the variables. We can perform these steps in any order.

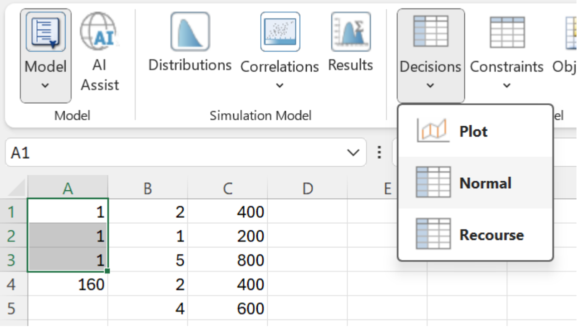


=75*A1+50*A2+35*A3

	A	B	C
1	1	2	400
2	1	1	200
3	1	5	800
4	160	2	400
5		4	600

Defining Decision Variables

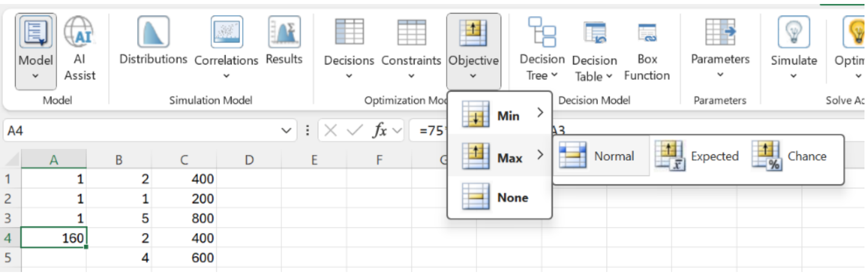
To define the **decision variables**, use your mouse to select the range A1:A3 on the worksheet. Then click the **Decisions** button to open the dropdown gallery (shown below), and click **Normal** to define A1:A3 as normal decision variables or, if using Analytic Solver Desktop, simply click the Decisions button to define A1:A3 as normal decision variables.



	A	B	C
1	1	2	400
2	1	1	200
3	1	5	800
4	160	2	400
5		4	600

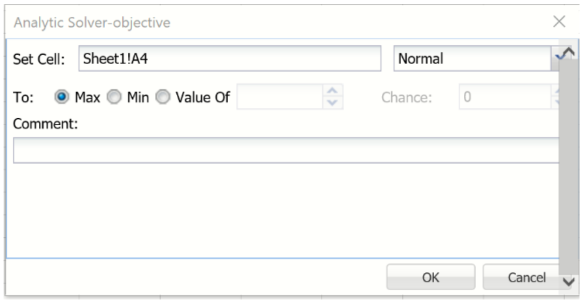
Defining the Objective

To define the **objective**, use your mouse to select cell A4 on the worksheet. In Analytic Solver Desktop, click the down arrow under **Objective**, then select **Max** and **Normal** from the menu to define A4 as a normal objective to be maximized – as shown on the next page. Alternatively, you can click the **Objective** button to open the Add Objective dialog. Click OK to accept the defaults and to define A4 as a normal objective to be maximized.



	A	B	C
1	1	2	400
2	1	1	200
3	1	5	800
4	160	2	400
5		4	600

In Analytic Solver Cloud, click the down arrow under Objective and select Add/Change Objective. Click OK to accept the defaults and to define A4 as a normal objective to be maximized.



Set Cell: Sheet1!A4 Normal

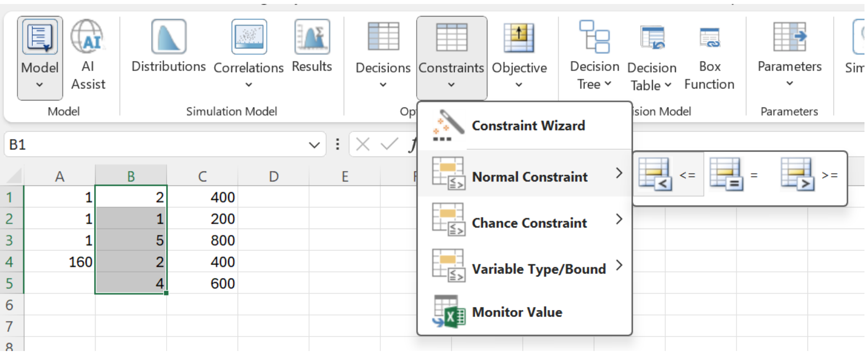
To: ● Max ○ Min ○ Value Of Chance: 0

Comment:

Defining Constraints

To define the **constraints**, use your mouse to select the range B1:B5 on the worksheet. Then, if using Analytic Solver Desktop, click the **Constraints** button to open the Add Constraint dialog.

Alternatively, you could also click the down arrow under Constraints, then click **Normal Constraint** and <= to define B1:B5 as the left hand sides of five <= constraints.

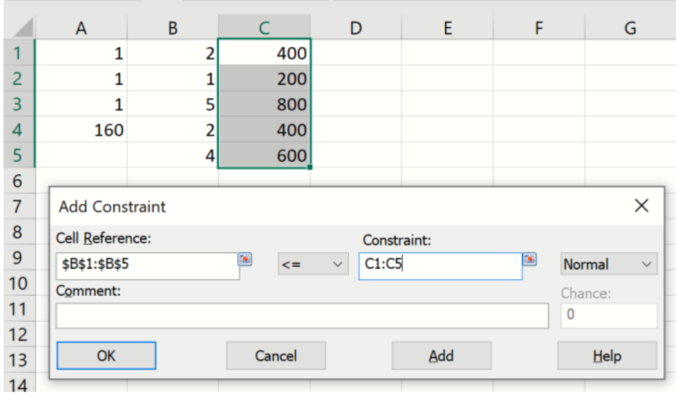


B1

	A	B	C
1	1	2	400
2	1	1	200
3	1	5	800
4	160	2	400
5		4	600

If using Analytic Solver Cloud, click the down arrow under Constraints, then click Add Constraint to open the Add Constraint dialog.

Within this dialog, you can enter the right hand side(s) as a constant number, a cell range, or a defined name. Click in the **Constraint** edit box, or click the **range selector icon** to its right, and use your mouse to select the cell range C1:C5 for the right hand sides:



	A	B	C
1	1	2	400
2	1	1	200
3	1	5	800
4	160	2	400
5		4	600

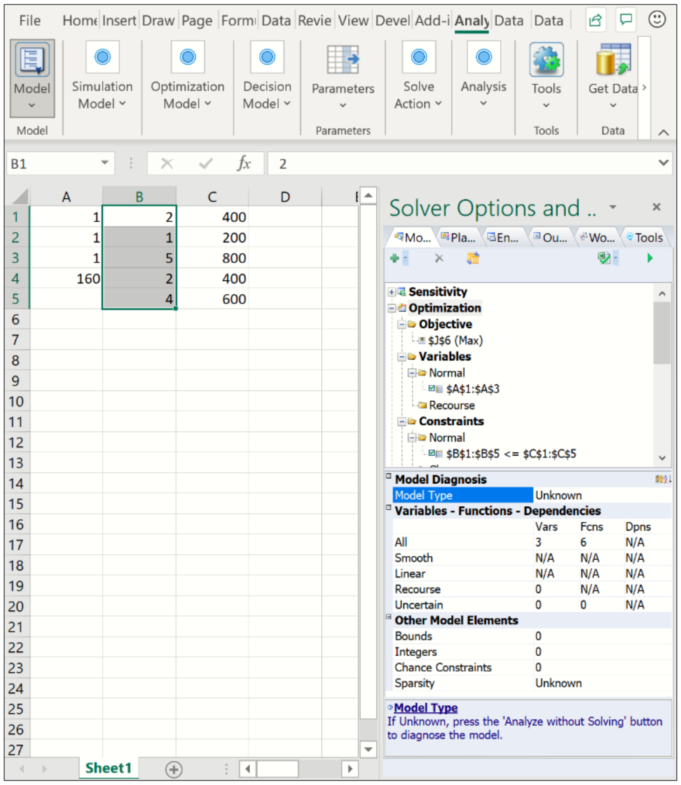
Cell Reference: $B$1:$B$5 <= Constraint: C1:C5 Normal

Comment:

Chance: 0

Then click OK. This will define five constraints: B1 <= C1, B2 <= C2, B3 <= C3, B4 <= C4, and B5 <= C5.

The Task Pane Model tab now shows all the elements of the optimization model you've just defined in outline form.



	A	B	C	D
1	1	2	400	
2	1	1	200	
3	1	5	800	
4	160	2	400	
5		4	600	

- Sensitivity
- Optimization
  - Objective
    - $J$6 (Max)
  - Variables
    - Normal
      - $A$1:$A$3
    - Recourse
  - Constraints
    - Normal
      - $B$1:$B$5 <= $C$1:$C$5

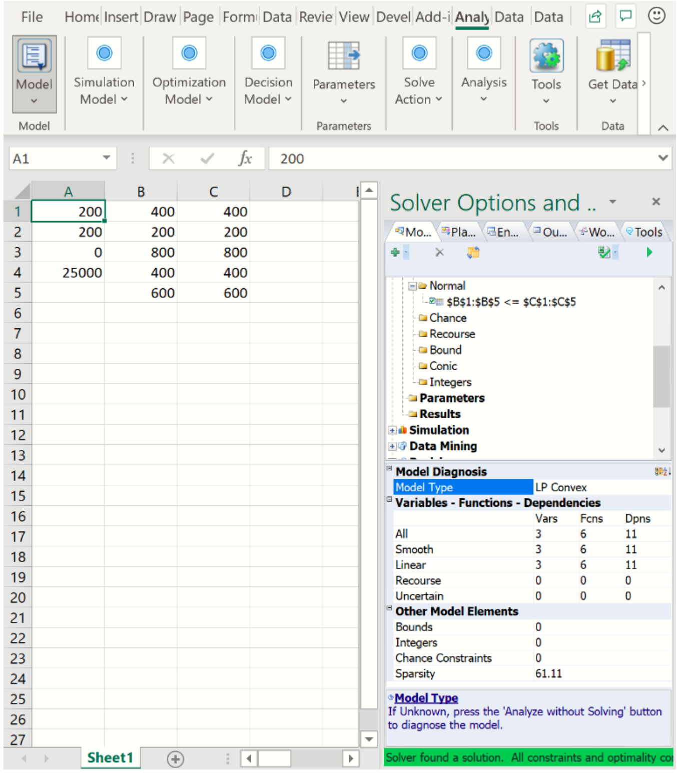
Model Diagnosis

Model Type	Unknown		
Variables - Functions - Dependencies			
	Vars	Fcns	Dpns
All	3	6	N/A
Smooth	N/A	N/A	N/A
Linear	N/A	N/A	N/A
Recourse	0	N/A	N/A
Uncertain	0	0	N/A
Other Model Elements			
Bounds	0		
Integers	0		
Chance Constraints	0		
Sparsity	Unknown		

Model Type

If Unknown, press the 'Analyze without Solving' button to diagnose the model.

You can solve the model immediately by clicking the **Optimize** button on the Ribbon, or by clicking the **green arrow** at the top right of the Task Pane.



	A	B	C	D
1	200	400	400	
2	200	200	200	
3	0	800	800	
4	25000	400	400	
5		600	600	

Model Diagnosis

Model Type	LP Convex		

Variables - Functions - Dependencies

	Vars	Fcns	Dpns
All	3	6	11
Smooth	3	6	11
Linear	3	6	11
Recourse	0	0	0
Uncertain	0	0	0

Other Model Elements

Bounds	0
Integers	0
Chance Constraints	0
Sparsity	61.11

Model Type

If Unknown, press the 'Analyze without Solving' button to diagnose the model.

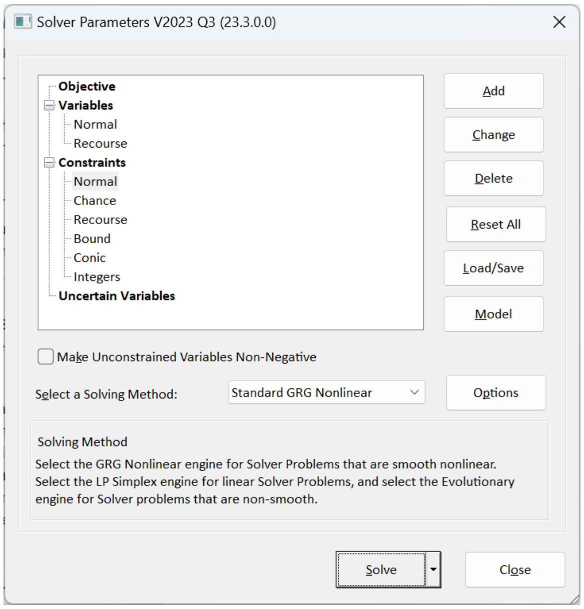
Solver found a solution. All constraints and optimality co

The optimal values of the decision variables at A1:A3, the objective at A4, and the constraints at B1:B5 appear on the worksheet, and a **Solver Result message** ("Solver found a solution. All constraints and optimality conditions are satisfied") appears in green at the bottom of the Task Pane.

You can view a log of solution messages on the Task Pane Output tab in Analytic Solver Desktop, and you can produce optimization reports by selecting **Reports – Optimization** from the Ribbon. For more information about these features, keep reading!

Using the Classical Solver Parameters Dialog

When using Analytic Solver Desktop, you can also view – and create or edit – this model through the classical Solver Parameters dialog (as used in the standard Excel Solver). To do so simply click **Premium Solver** on the **Add-Ins** tab on the Ribbon to display the Solver Parameters dialog. Note: This dialog is not supported in Analytic Solver Cloud.



☐ Make Unconstrained Variables Non-Negative

Select a Solving Method: Standard GRG Nonlinear

Solving Method

Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

Comparing the Task Pane and Solver Parameters Dialog

As you can see, the Task Pane Model tab and the Solver Parameters dialog contain the same information. But where the Solver Parameters dialog is *modal* (moving the mouse outside the dialog displays a wait cursor – you must close the dialog to do anything else), the Task Pane is *modeless*: You can move the mouse outside the pane, edit formulas on the worksheet, or use other commands.

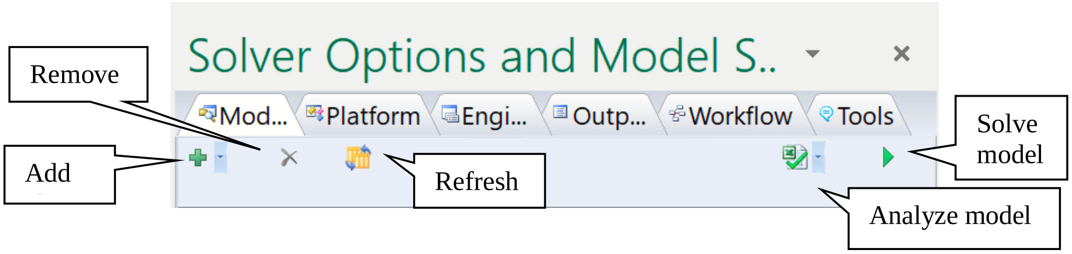
The Task Pane is initially docked to the right side of the Excel window, but you can select its title bar with your mouse, **drag** it to another position, and **resize** it, as shown on the next page. To "re-dock" the Task Pane, select its title bar with the mouse, drag to a position just beyond the right edge of the Excel window, then release the mouse.

Use the **Model tab** to view your model in outline form, and optionally edit model elements in-place. Use the **Platform tab** to view or change Platform options, such as the number of optimizations or simulations to run, or default bounds on decision variables or uncertain variables. Use the **Engine tab** to select a Solver Engine and view or change its options. Use the **Output tab** to view a log of solution messages, or a chart of the objective values.

Using Buttons on the Task Pane

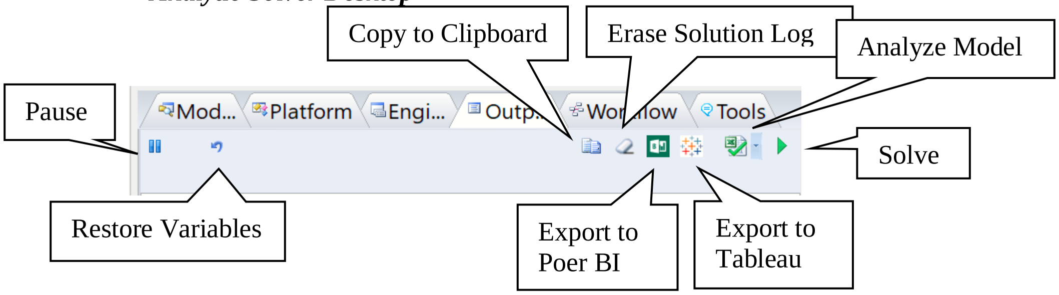
Use the **buttons** at the top of the **Model tab** to add or remove model elements (you can also use the Ribbon options to do this, as shown earlier), refresh the model outline when you've made unusual changes to the worksheet, analyze the structure of your model, or solve (run) the optimization or simulation model:

Analytic Solver Desktop and Cloud

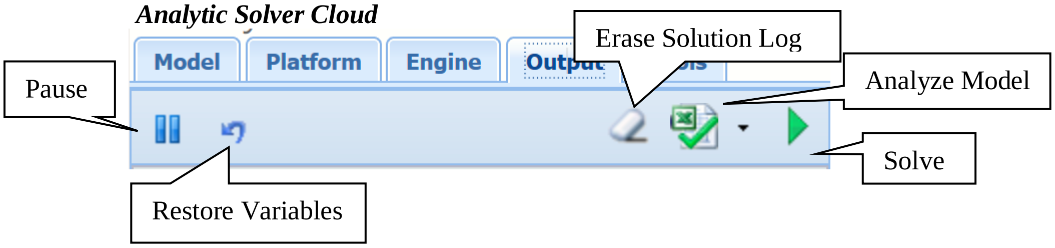


Use the **buttons** at the top of the **Output tab** to pause or stop the Solver, restore the original values of the decision variables, copy the solution message log to the Windows Clipboard (so you can paste the text into another application – Analytic Solver Desktop only), erase the solution log, analyze/solve the model and upload your results to Microsoft's Power BI or Tableau (see step by step instructions below – Analytic Solver Desktop only).

Analytic Solver Desktop



Analytic Solver Cloud



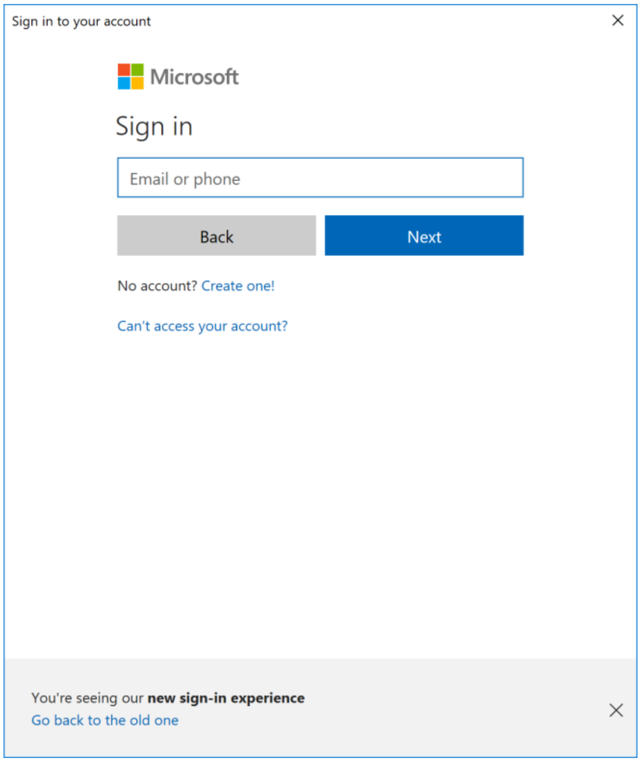
Exporting Data to Microsoft's Power BI

Microsoft's POWER BI, for use with Office 365, is a cloud-based service that works with Excel to help you visualize your data using various charts and reports. Analytic Solver Desktop includes the ability to export a fixed set of data from your optimization model, including final variable values, final constraint values, final objective function values, variable shadow prices or constraint reduced costs, etc., directly into a dataset in Power BI.

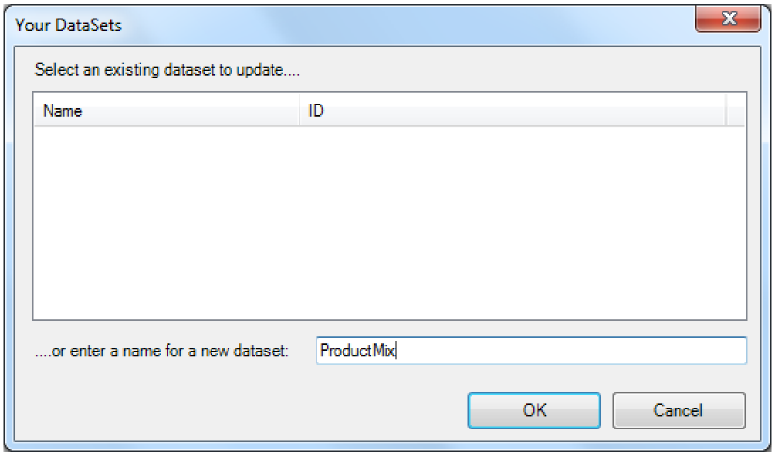
Note: Exporting your model to Power BI via the Output Pane on the Solver Task Pane is not supported in Analytic Solver Cloud. However, it is possible to upload your model to Power BI using the Deploy Model menu. See the chapter, Creating Power BI Custom Visuals, that appears later on in this guide for more information.

Once Solver has stopped with a final result message (even if a solution was not found), click the icon on the Output tab to upload the model to the Power BI dashboard. If this is the first time that the icon has been clicked within the current Excel instance, you will be asked to log in to Power BI.

Note: If an error appears immediately after clicking the Power BI icon, you'll need to upgrade your current version of TLS (Transport Layer Security) to V1.2.

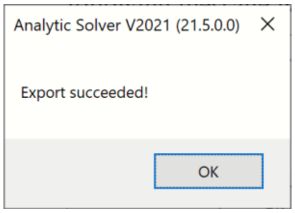


Once logged in, you will be asked to either update an existing dataset or create a new dataset.



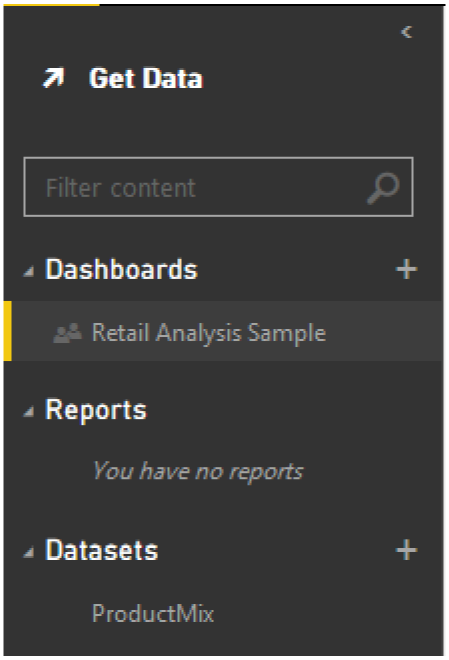
....or enter a name for a new dataset: ProductMix

In the screenshot above, we have created a new dataset, ProductMix, which will contain the objective, variable and constraint final values, shadow prices, dual values, etc from our optimization model. Once the upload is complete, the following message will appear.

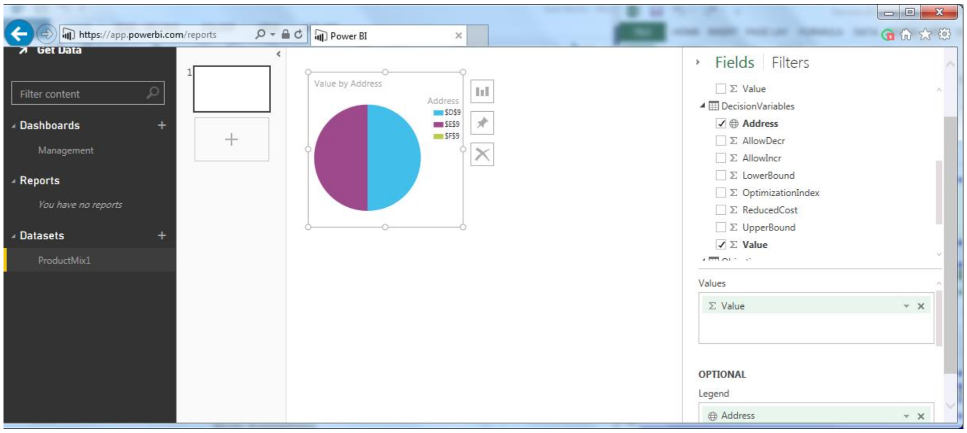


Export succeeded!

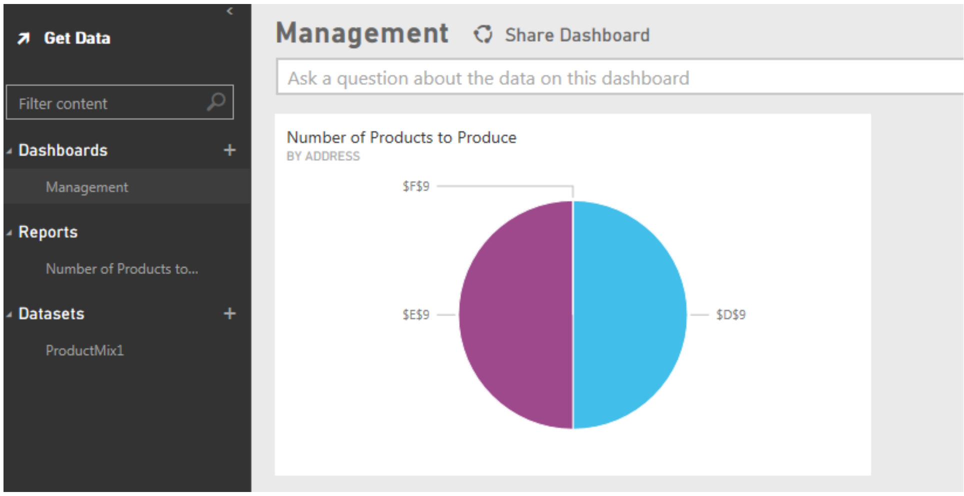
Logon to Power BI (http://powerbi.microsoft.com/). The newly created dataset will be listed under Datasets.



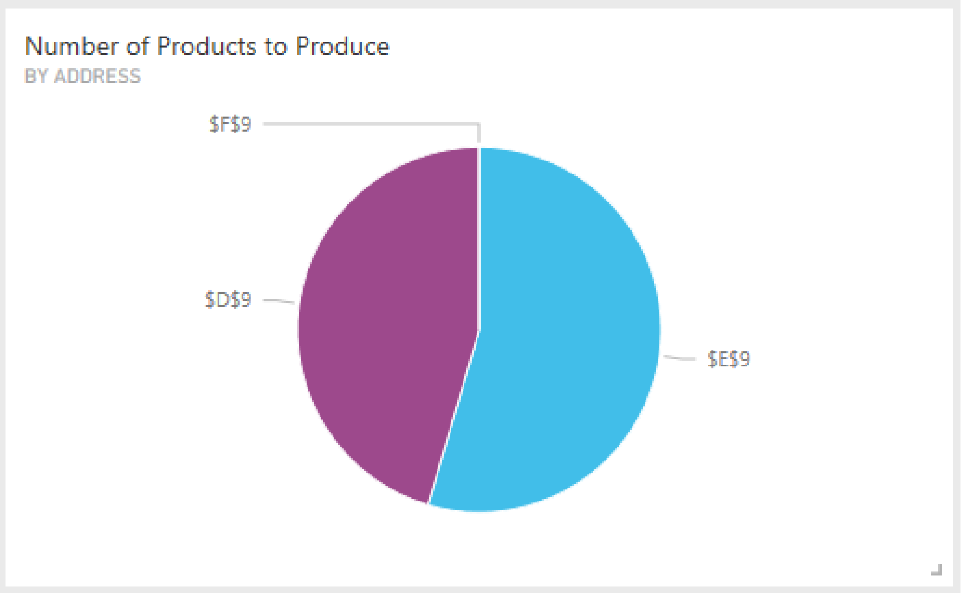
Select ProductMix, then determine the components to be included in the graph. In the screenshot below we have created a pie chart displaying the final variable values by clicking the 📊 icon to the right of the graph.



Use the 📌 icon to pin this graph to the Dashboard.



Now, each time the model is solved, the results may be uploaded to your Power BI dashboard. Click back to Excel and change the number of Speaker Cones in inventory from 800 to 825, then resolve either by clicking the green arrow on the Model tab in the Solver Task Pane or by clicking the Optimize icon on the Analytic Solver ribbon. Once Solver stops with the final result message, "Solver found a solution. All constraints and optimality conditions are satisfied.", click the icon on the Output tab to upload the most recent results. (Note: We are not asked to log in to the Power BI site a second time since we are using the same instance of Excel.) Click back to Power BI in your browser and refresh, the chart will update automatically with the new final variable values, as shown below.



Exporting Data to Tableau

Tableau is a popular interactive software package that allows you to visually explore and analyze your data. Tableau can import data from a wide range of sources, including Excel workbooks, and it is often used in conjunction with Excel. Because Tableau is designed to import data in table form, it hasn't been easy to import the *results* of an optimization model (such as final values of the decision variables, constraints and objective) into Tableau, unless those model elements occur in table form by themselves in your spreadsheet (which usually isn't the case).

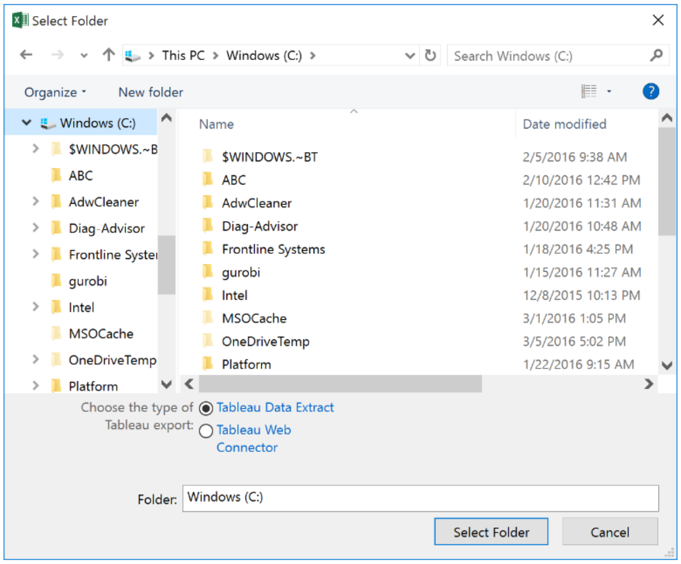
Analytic Solver simplifies this process considerably. With a single click, you can convert the results of your optimization model into a set of Tableau Data Extract (.tde) files or HTML files using the Tableau Web Connector, open them directly in Tableau, and visualize them with a few clicks.

Note: Exporting your model to Tableau via the Output Pane on the Solver Task Pane is not supported in Analytic Solver Cloud. However, it is possible to upload your model to Tableau using the Deploy Model menu. See the chapter, Creating Custom Extensions in Tableau, that appears later on in this guide for more information.

Click the green arrow on the Output Tab to solve your optimization model.

Once Solver has stopped with a final result message, click the icon on the Task Pane Output tab to save the values for the objective function, constraints and variables to *.tde files or HTML files. You will be prompted to select a folder where the Tableau files will be saved along with the type of Tableau export desired: Tableau Data Extract or Tableau Web Connector.

Note: If exporting to an existing .tde file, data will be appended rather than overwritten. As a result, when exporting to an existing .tde file, all data must be of the same structure as when the .tde file was first created.



Name	Date modified
$WINDOWS.~BT	2/5/2016 9:38 AM
ABC	2/10/2016 12:42 PM
AdwCleaner	1/20/2016 11:31 AM
Diag-Advisor	1/20/2016 10:48 AM
Frontline Systems	1/18/2016 4:25 PM
gurobi	1/15/2016 11:27 AM
Intel	12/8/2015 10:13 PM
MSOCache	3/1/2016 1:05 PM
OneDriveTemp	3/5/2016 5:02 PM
Platform	1/22/2016 9:15 AM

Choose the type of Tableau export: ● Tableau Data Extract ○ Tableau Web Connector

Folder: Windows (C:)

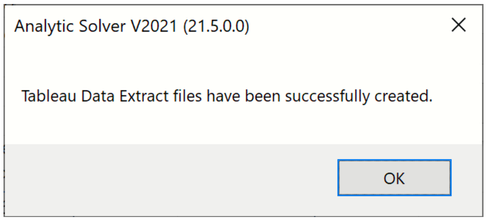
Tablea Data Extract

If Tableau Data Extract is selected, static data is exported to three files: Constraints.tde, Objective.tde and Variables.tde. (Although .tde files are designed to hold multiple tables, currently Tableau's software allows only one table per file.) When you import this data into Tableau, Variables.tde will have one row for each decision variable, Constraints.tde will have one row for each constraint and Objective.tde will have just one row for the objective.

- Each row of Variables.tde will contain the Excel cell address, the values of the variables at the time of extraction, the lower and upper bound of each variable, the optimization index and sensitivity information such as the reduced cost and allowable increase/decrease.
- Each row of Constraints.tde will contain the Excel cell address where the constraint is located, the value of each constraint at time of extraction, the constraint lower and upper bounds, and information that would appear on a sensitivity report such as the shadow price, slack value, a 1 or 0 to indicate if the constraint was binding (1) or not (0), and the allowable increase/decrease.
- Objective.tde will contain the Excel cell address where the objective is located, the value of the objective function at the time of extraction, and the optimization index.

Note: If exporting to an existing .tde file, data will be appended rather than overwritten. As a result, when exporting to an existing .tde file, all data must be of the same structure as when the .tde file was first created.

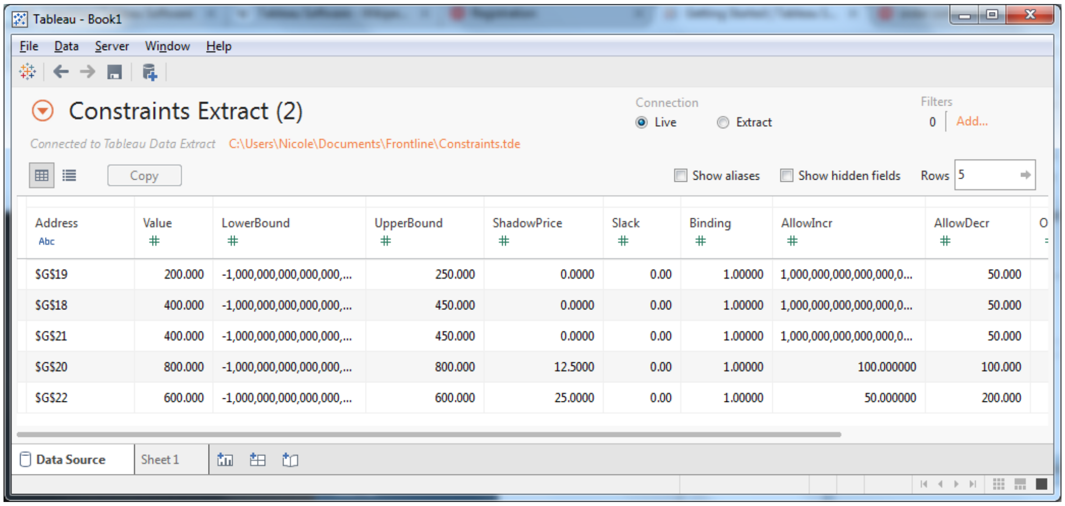
Once you select a folder, the *.tde files will be created, and the following message will appear.



Analytic Solver V2021 (21.5.0.0)

Tableau Data Extract files have been successfully created.

To open the files in Tableau, either double click each file (if using Desktop Tableau) or click **Other Files** under *Connect* and open the desired file(s).



Connected to Tableau Data Extract C:\Users\Nicole\Documents\Frontline\Constraints.tde

Connection ● Live ○ Extract

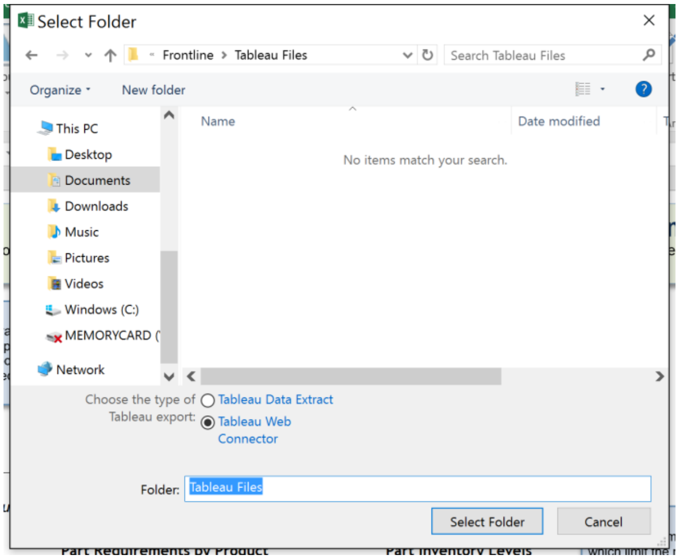
Filters 0 Add...

☐ Show aliases ☐ Show hidden fields Rows 5

Address	Value	LowerBound	UpperBound	ShadowPrice	Slack	Binding	AllowIncr	AllowDecr
$G$19	200.000	-1,000,000,000,000,000,...	250.000	0.0000	0.00	1.00000	1,000,000,000,000,000,0...	50.000
$G$18	400.000	-1,000,000,000,000,000,...	450.000	0.0000	0.00	1.00000	1,000,000,000,000,000,0...	50.000
$G$21	400.000	-1,000,000,000,000,000,...	450.000	0.0000	0.00	1.00000	1,000,000,000,000,000,0...	50.000
$G$20	800.000	-1,000,000,000,000,000,...	800.000	12.5000	0.00	1.00000	100.000000	100.000
$G$22	600.000	-1,000,000,000,000,000,...	600.000	25.0000	0.00	1.00000	50.000000	200.000

Tableau Web Connector

The Tableau Web Connector offers much more flexibility over Tableau Data Extract by allowing you to refresh your data dynamically inside of Tableau.

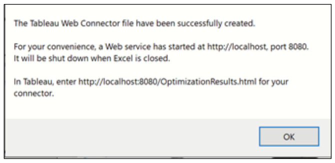


Choose the type of Tableau export: ○ Tableau Data Extract ● Tableau Web Connector

Folder: Tableau Files

If Tableau Web Connector is selected, you will be prompted to select a folder in which to save OptimizationResults.html. This file will hold all contents described above for Constraints.tde, Objective.tde and Variables.tde. The following message will appear.

Note: If exporting to an existing .tde file, data will be appended rather than overwritten. As a result, when exporting to an existing .tde file, all data must be of the same structure as when the .tde file was first created.

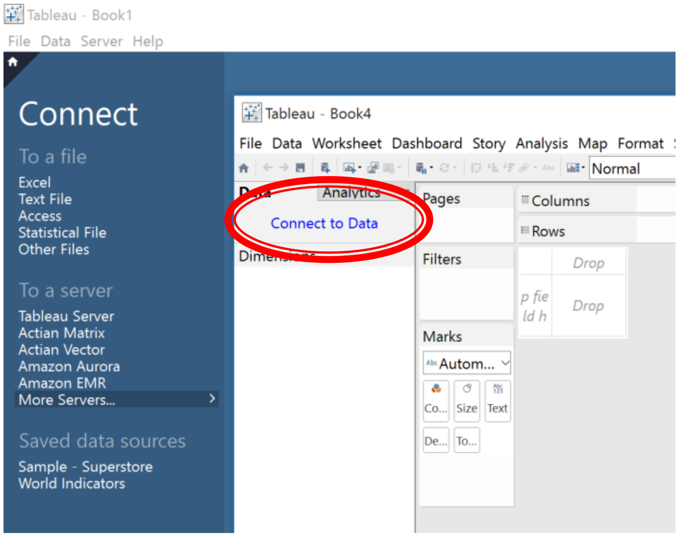


The Tableau Web Connector file have been successfully created.

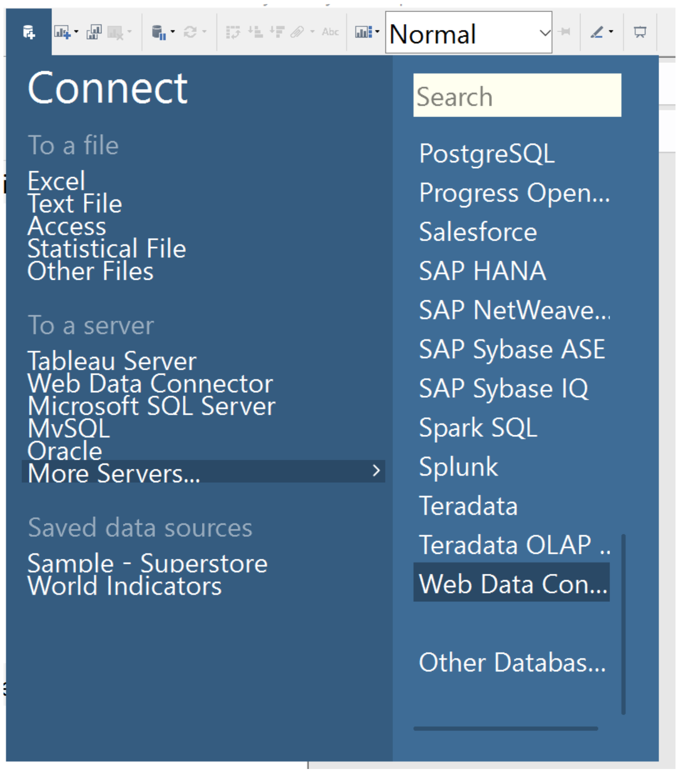
For your convenience, a Web service has started at http://localhost, port 8080. It will be shut down when Excel is closed.

In Tableau, enter http://localhost:8080/OptimizationResults.html for your connector.

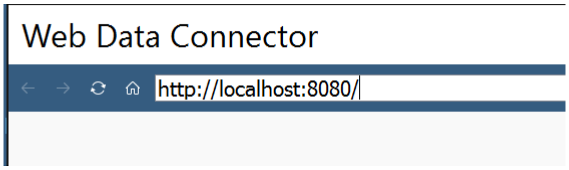
To open the files in Tableau, open a new workbook in Tableau and click **Connect to Data**.



On the connect menu, select **More Servers – Web Data Connector** on the Connect menu.



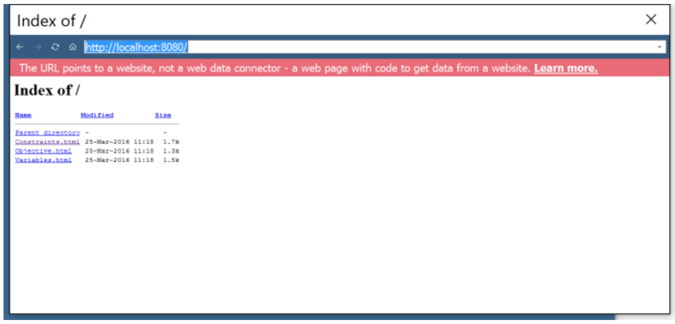
On the Web Data Connector dialog, enter the location displayed on the dialog shown above (i.e., http://localhost:8080/) and press **Enter**.


Web Data Connector

http://localhost:8080/

When the following dialog appears, click Constraints.html.

Note: The error highlighted in red is a simple warning that the URL is not pointing to a Tableau Web Data Connector file. If we had entered a file name, such as http://localhost:8080/Constraints.html, the results would have immediately been uploaded to Tableau.


Index of /

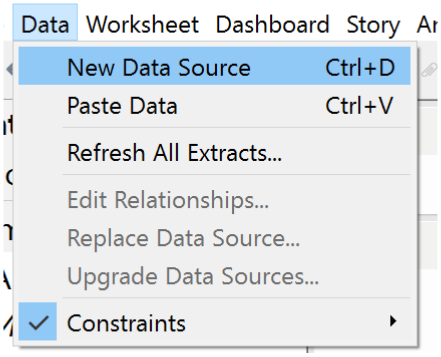
http://localhost:8080/

The URL points to a website, not a web data connector - a web page with code to get data from a website. Learn more.

Index of /

Name	Modified	Size
Parent directory	-	-
Constraints.html	25-Mar-2016 11:18	1.7k
Objective.html	25-Mar-2016 11:18	1.3k
Variables.html	25-Mar-2016 11:18	1.5k

To add more data, click **Data – New Data Source** on the Tableau ribbon, then repeat the actions described above.



If your Solver results have changed, you'll need to refresh the results within the Tableau Web Connector HTML files. To do so:

1. In ASP, re-run your model
2. Click the Tableau icon on the Output tab in the Solver Task Pane.
3. Select **Tableau Web Connector** and the folder where the files should be saved.
4. Click **OK**.
5. In Tableau, click **Data – Refresh All Extracts** to update your data.

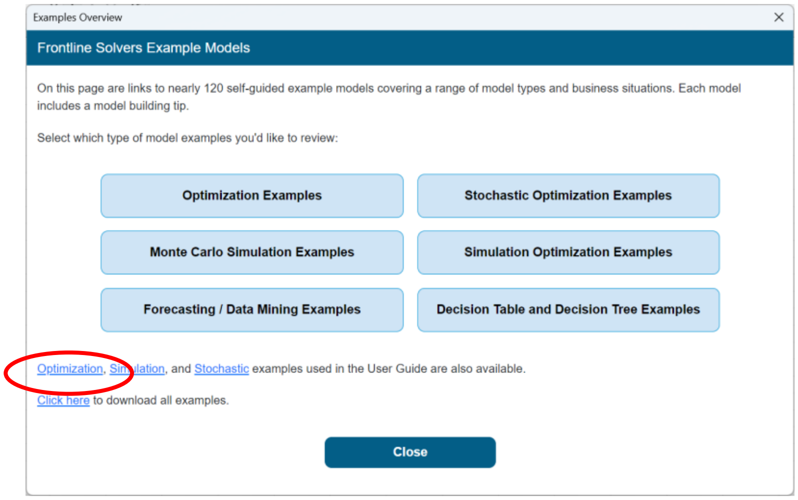
For more information on using Tableau, please refer to the Tableau documentation found at http://www.tableau.com/.

Introducing the Standard Example Models

The approach outlined above in "From Algebra to Spreadsheets" is a 'quick and dirty' way to translate from a model in algebraic form to an equivalent spreadsheet model, ready for optimization. However, that approach will soon prove to be short-sighted when you wish to change the data (for example unit profits or parts on hand), expand the model to include more products or parts, or show the model to someone unfamiliar with the problem or uncomfortable with algebraic notation.

Opening the Examples

To see a better approach to defining this model, click **Help – Examples** on the Ribbon, which opens the workbook, Frontline Example Models Overview.xlsx. Click Optimization on the Overview tab.



Examples Overview

Frontline Solvers Example Models

On this page are links to nearly 120 self-guided example models covering a range of model types and business situations. Each model includes a model building tip.

Select which type of model examples you'd like to review:

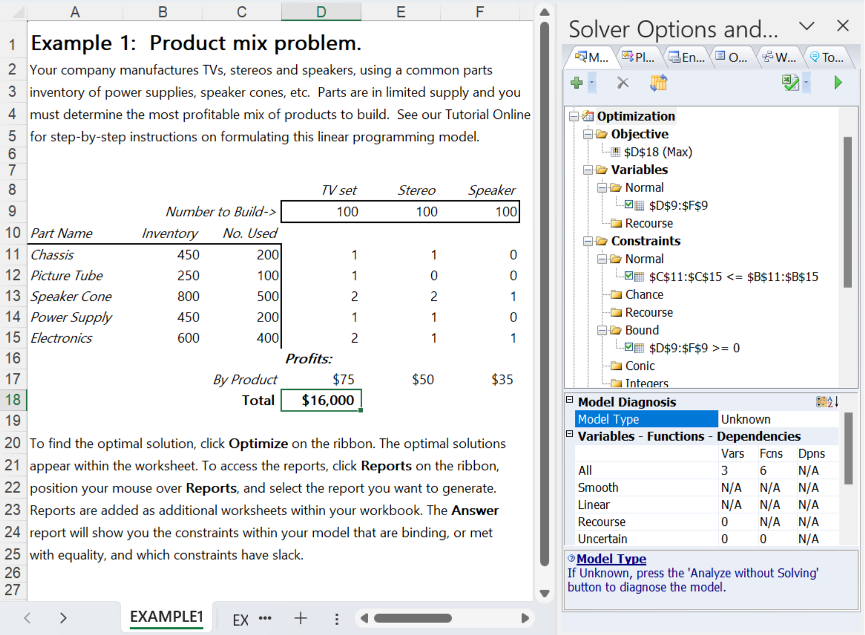
Optimization Examples
Stochastic Optimization Examples
Monte Carlo Simulation Examples
Simulation Optimization Examples
Forecasting / Data Mining Examples
Decision Table and Decision Tree Examples

Optimization, Simulation, and Stochastic examples used in the User Guide are also available.

Click here to download all examples.

Then click the link **StandardExamples.xlsx** to open the workbook, showing the first worksheet EXAMPLE1.

An optimization model – like the one we created 'from scratch' in the previous section – is already defined on this worksheet, and appears in the Task Pane Model tab, as shown below. We could *immediately* solve this model by clicking the green arrow on the Model tab of the Solver Task pane.



Example 1: Product mix problem.

Your company manufactures TVs, stereos and speakers, using a common parts inventory of power supplies, speaker cones, etc. Parts are in limited supply and you must determine the most profitable mix of products to build. See our Tutorial Online for step-by-step instructions on formulating this linear programming model.

			TV set	Stereo	Speaker
	Number to Build->		100	100	100
Part Name	Inventory	No. Used			
Chassis	450	200	1	1	0
Picture Tube	250	100	1	0	0
Speaker Cone	800	500	2	2	1
Power Supply	450	200	1	1	0
Electronics	600	400	2	1	1
			Profits:		
		By Product	$75	$50	$35
		Total	$16,000		

To find the optimal solution, click **Optimize** on the ribbon. The optimal solutions appear within the worksheet. To access the reports, click **Reports** on the ribbon, position your mouse over **Reports**, and select the report you want to generate. Reports are added as additional worksheets within your workbook. The **Answer** report will show you the constraints within your model that are binding, or met with equality, and which constraints have slack.

Optimization
- **Objective**
  - $D$18 (Max)
- **Variables**
  - Normal
    - ☑ $D$9:$F$9
  - Recourse
- **Constraints**
  - Normal
    - ☑ $C$11:$C$15 <= $B$11:$B$15
  - Chance
  - Recourse
  - Bound
    - ☑ $D$9:$F$9 >= 0
  - Conic
  - Integers

Model Diagnosis

Model Type: Unknown

Variables - Functions - Dependencies

	Vars	Fcns	Dpns
All	3	6	N/A
Smooth	N/A	N/A	N/A
Linear	N/A	N/A	N/A
Recourse	0	N/A	N/A
Uncertain	0	0	N/A

Model Type
If Unknown, press the 'Analyze without Solving' button to diagnose the model.

How did this optimization model appear pre-defined in the workbook, you ask? We simply used the Ribbon (or the Solver Parameters dialog) to select the decision variables, objective and constraints, and the Solver engine (and any options or settings it requires). Then we saved the workbook, in the usual way, with Excel's Save command. Models created in the standard Excel Solver, or an earlier version of Analytic Solver, Premium Solver, or Risk Solver can also be saved this way, and opened and immediately solved in Analytic Solver.

More Readable and Expandable Models

Worksheet EXAMPLE1 models the same problem we described in the previous section “From Algebra to Spreadsheets,” but the model is organized and laid out for easier readability and expandability.

The decision variables (A1:A3 in the previous section) are in cells D9:F9, and nearby labels show their meaning. The left hand sides of the constraints (B1:B5 in the previous section) are in cells C11:C15, and the right hand sides are in cells B11:B15 – since these constant values can be in any position on the worksheet.

Instead of using constants such as 75, 50 and 35 directly in the formula for the objective – as we did in the previous section – we’ve put these numbers in cells D17:F17. The formula for the objective is =SUMPRODUCT(D17:F17,D9:F9) in cell D18. This makes it easier to change product selling prices, or to obtain this data from an external data source, such as a database or accounting data file.

Similarly, we’ve placed the *coefficients* of the constraints in cells D11:F15. These are exactly the constant values we saw in the algebraic statement of the problem in the previous section:

Subject to $1 x_1 + 1 x_2 + 0 x_3 <= 450$ (Chassis)
$1 x_1 + 0 x_2 + 0 x_3 <= 250$ (Picture tubes)
$2 x_1 + 2 x_2 + 1 x_3 <= 800$ (Speaker cones)
$1 x_1 + 1 x_2 + 0 x_3 <= 450$ (Power supplies)
$2 x_1 + 1 x_2 + 1 x_3 <= 600$ (Electronics)

In EXAMPLE1, the formula at cell C11 – the left hand side of the first constraint – is =SUMPRODUCT(D11:F11,$D$9:$F$9). The formulas at C12:C15 were created by copying C11 downward.

To enhance readability, borders and labels have been used to draw attention to the decision variables, the constraints, and the objective function. If you haven’t used these Excel features before, select one or more cells, then choose the Home tab in Excel to see how to do this.

EXAMPLE1 is also much easier to maintain and expand than a model constructed with ‘hardwired’ formulas, as in the previous section. You can add products by inserting new columns, and add more parts by inserting rows, and copying any one of the existing left hand side formulas to the new rows; Excel will automatically adjust the formula cell references.

In the next section, we’ll solve EXAMPLE1, and create and examine the Answer Report and Sensitivity Report for this model.

Models, Worksheets and Workbooks

There are several worksheets (EXAMPLE1, EXAMPLE2, etc.) in the StandardExamples.xlsx workbook. As we proceed through this chapter, you’ll see that we can simply click a worksheet tab to display a new and different model in the Task Pane on that worksheet. Each of these models was created in the same way, and all of them are saved when you save the Excel workbook.

In StandardExamples.xlsx, each model is entirely contained on one worksheet. But this is not required – one model can include cells from *many* different worksheets in the same workbook, and it can refer to constant information in other workbooks. By default, models have the same names as the worksheets on which they are created, but you can give *your own names* to models, and create as many of them as you like. This is described in the Frontline Solvers Reference Guide.

Linear Programming Examples

This section takes you on a quick tour of Analytic Solver features for linear programming (LP) problems. We highly recommend that you ‘follow along’ by opening and actually solving the examples in this section.

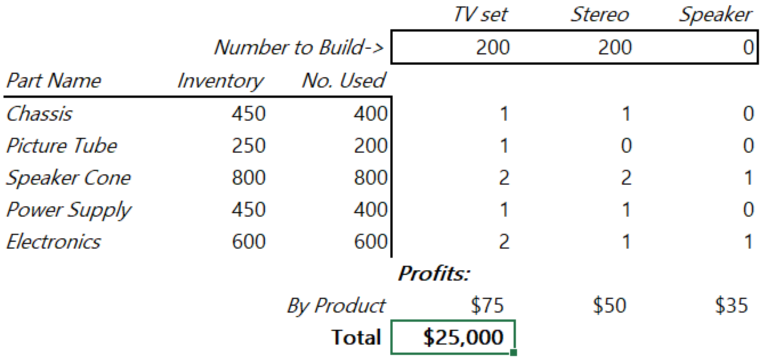
The first three examples are variants of the Product Mix model introduced earlier in this chapter. Our factory is building three products: TV sets, stereos and speakers. Each product is assembled from parts in inventory, and there are five types of parts: chassis, picture tubes, speaker cones, power supplies and electronics units. Our goal is to produce the mix of products that will maximize profits, given the inventory of parts on hand. Be sure to read “Introducing the Standard Example Models” above.

Using the Output Tab and Creating Reports

Click the tab to display the EXAMPLE1 worksheet. To solve this linear programming problem;

- In Analytic Solver Desktop, click the **Optimize** button on the Ribbon or click the **green arrow** at the top right of the Task Pane.
- In Analytic Solver Cloud, click the **Model** button to open the task pane, then click the **green arrow** at the top right of the Task Pane or click the **down arrow** beneath Optimize on the Ribbon and select **Solve Complete Problem**.

Almost immediately, the solution values appear in the decision variable cells:

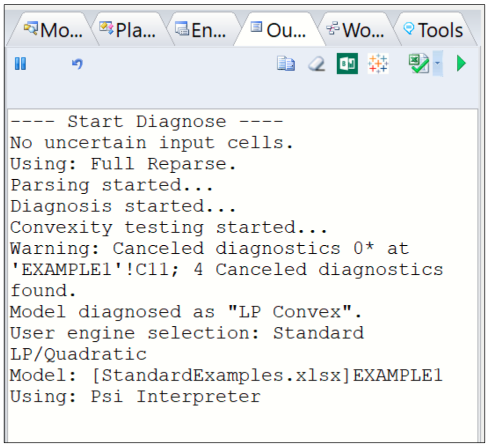


			TV set	Stereo	Speaker
		Number to Build->	200	200	0
Part Name	Inventory	No. Used			
Chassis	450	400	1	1	0
Picture Tube	250	200	1	0	0
Speaker Cone	800	800	2	2	1
Power Supply	450	400	1	1	0
Electronics	600	600	2	1	1
			Profits:		
		By Product	$75	$50	$35
		Total	**$25,000**		

To maximize profit, we should make 200 TV sets, 200 stereos and 0 speakers.

The Solver Results message “Solver found a solution. All constraints and optimality conditions are satisfied” appears in green at the bottom of the Task Pane. **Don’t ignore this message**, because it provides critical information about what happened during the solution process, and what you *can* and *cannot* assume about the solution on the worksheet!

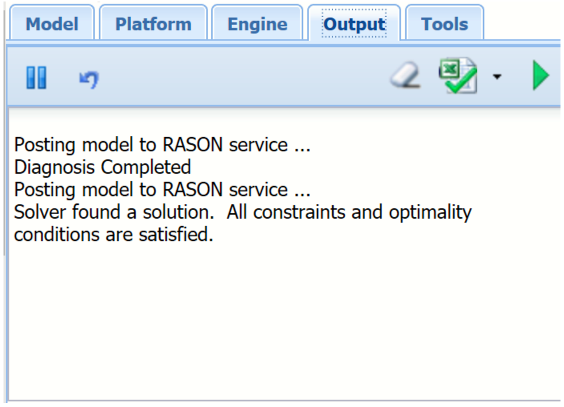
If using Analytic Solver Desktop, click the **Output** tab at the top of the Task Pane to learn more about the solution process. The solution log in the upper part of this pane reports what happened during the solution process. If you are puzzled by the Solver Result message, simply **click the underlined message** in the log to open online Help to a full explanation of that message.



```
---- Start Diagnose ----
No uncertain input cells.
Using: Full Reparse.
Parsing started...
Diagnosis started...
Convexity testing started...
Warning: Canceled diagnostics 0* at
'EXAMPLE1'!C11; 4 Canceled diagnostics
found.
Model diagnosed as "LP Convex".
User engine selection: Standard
LP/Quadratic
Model: [StandardExamples.xlsx]EXAMPLE1
Using: Psi Interpreter
```

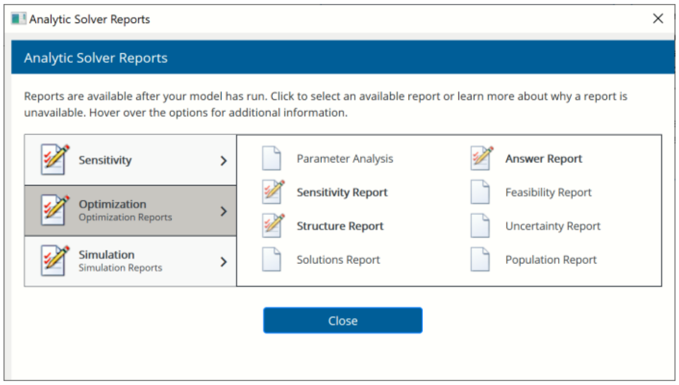
If solving takes more than a few seconds, the Output tab automatically appears, and current **best objective** and other information, plus a **running chart** of the objective value, appears in the bottom part of the Task Pane.

If using Analytic Solver Cloud, the output pane will display the status of the model on the Solver Server as shown below.



```
Posting model to RASON service ...
Diagnosis Completed
Posting model to RASON service ...
Solver found a solution.  All constraints and optimality
conditions are satisfied.
```

After solving, you can produce reports (such as the Answer Report and Sensitivity Report) by selecting **Reports – Optimization** from the Ribbon. Unlike the standard Solver, you don't have to choose reports *immediately* in the Solver Results dialog – you can select them later (until you perform another operation that creates new reports).



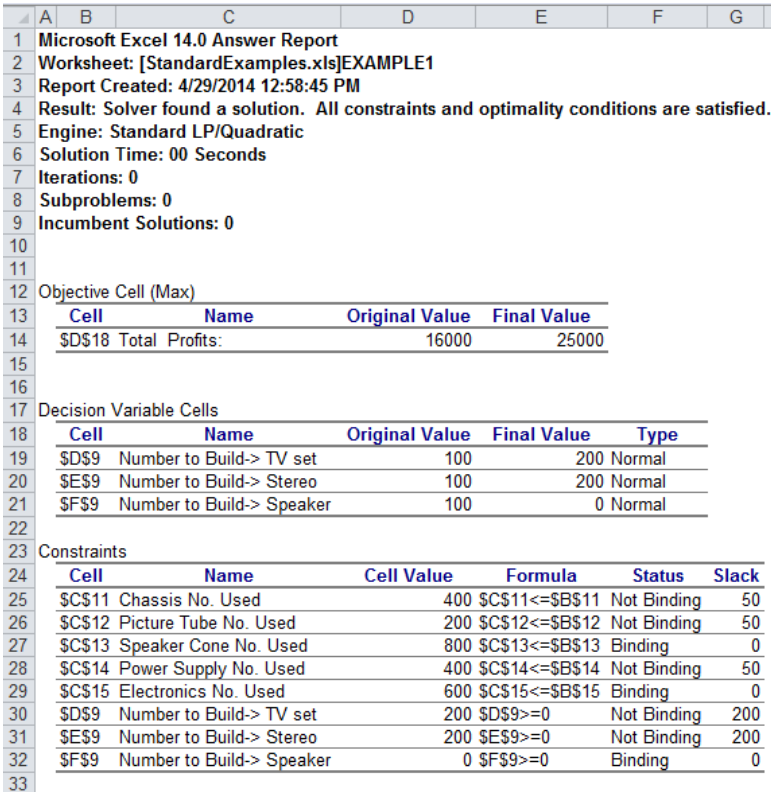
Reports are available after your model has run. Click to select an available report or learn more about why a report is unavailable. Hover over the options for additional information.

The Answer and Sensitivity Reports appear as new worksheets, inserted into the workbook just to the left of the EXAMPLE worksheet, as shown on the next page. The Sensitivity Report, which provides *dual values* (shadow prices and reduced costs) and associated *ranges*, is described in the Frontline Solvers Reference Guide.

The Answer Report shows the message from the Task Pane Output tab, the Solver engine used, and statistics about the solution process. It shows the beginning and ending values of the objective function and the decision variables. For the constraints, it shows the final cell value and formula, a status (whether the constraint was "binding" – satisfied with equality – at the solution, and a "slack value" for non-binding constraints.

If using Analytic Solver Desktop, you can ask for **outlined** reports by clicking the "Reports are not outlined" choice on the Ribbon to change it to "Reports are outlined," as shown above. In this case, each block of decision variables and constraints appears in its own outline group. Click the + symbol at the left edge of the report worksheet to show or hide each group in the outline. Any comments that you enter when defining decision variables or constraints appear for each group in the report. These features help you quickly find the information you want in a report created for a large model.

Analytic Solver Desktop



Microsoft Excel 14.0 Answer Report
Worksheet: [StandardExamples.xls]EXAMPLE1
Report Created: 4/29/2014 12:58:45 PM
Result: Solver found a solution. All constraints and optimality conditions are satisfied.
Engine: Standard LP/Quadratic
Solution Time: 00 Seconds
Iterations: 0
Subproblems: 0
Incumbent Solutions: 0

Objective Cell (Max)

Cell	Name	Original Value	Final Value
$D$18	Total Profits:	16000	25000

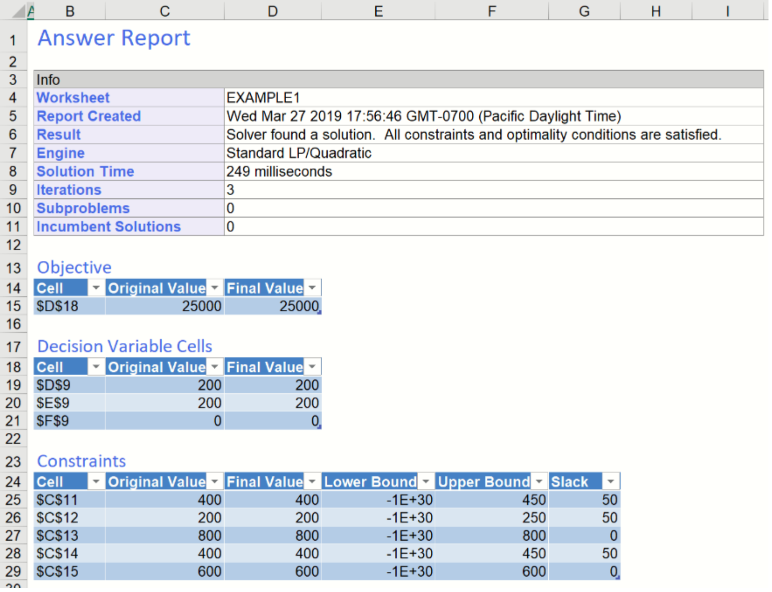
Decision Variable Cells

Cell	Name	Original Value	Final Value	Type
$D$9	Number to Build-> TV set	100	200	Normal
$E$9	Number to Build-> Stereo	100	200	Normal
$F$9	Number to Build-> Speaker	100	0	Normal

Constraints

Cell	Name	Cell Value	Formula	Status	Slack
$C$11	Chassis No. Used	400	$C$11<=$B$11	Not Binding	50
$C$12	Picture Tube No. Used	200	$C$12<=$B$12	Not Binding	50
$C$13	Speaker Cone No. Used	800	$C$13<=$B$13	Binding	0
$C$14	Power Supply No. Used	400	$C$14<=$B$14	Not Binding	50
$C$15	Electronics No. Used	600	$C$15<=$B$15	Binding	0
$D$9	Number to Build-> TV set	200	$D$9>=0	Not Binding	200
$E$9	Number to Build-> Stereo	200	$E$9>=0	Not Binding	200
$F$9	Number to Build-> Speaker	0	$F$9>=0	Binding	0

Analytic Solver Cloud



Answer Report

Info	
Worksheet	EXAMPLE1
Report Created	Wed Mar 27 2019 17:56:46 GMT-0700 (Pacific Daylight Time)
Result	Solver found a solution. All constraints and optimality conditions are satisfied.
Engine	Standard LP/Quadratic
Solution Time	249 milliseconds
Iterations	3
Subproblems	0
Incumbent Solutions	0

Objective

Cell	Original Value	Final Value
$D$18	25000	25000

Decision Variable Cells

Cell	Original Value	Final Value
$D$9	200	200
$E$9	200	200
$F$9	0	0

Constraints

Cell	Original Value	Final Value	Lower Bound	Upper Bound	Slack
$C$11	400	400	-1E+30	450	50
$C$12	200	200	-1E+30	250	50
$C$13	800	800	-1E+30	800	0
$C$14	400	400	-1E+30	450	50
$C$15	600	600	-1E+30	600	0

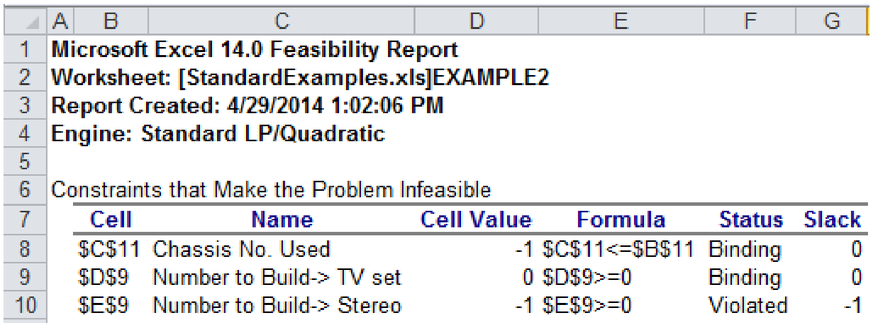
A Model with No Feasible Solution

What does it mean when the Solver can find no feasible solution – no combination of values for the decision variables that satisfies all the constraints? It may be that the underlying business problem has no feasible solution – if so, you'd like to focus on the 'hardest-to-satisfy' constraints, to see if you can do anything about them. Or you might have just made a mistake setting up the model – for example choosing <= when you meant to choose >=.

But in a large model, with thousands of constraints, how do you find your mistake, or focus on the 'hardest-to-satisfy' constraints? Analytic Solver can help, by producing a Feasibility Report.

Click the tab to display the EXAMPLE2 worksheet. This is the same Product Mix model as in EXAMPLE1, but the supply of Chassis parts at B11 has been changed from 450 to -1. Click the **Optimize** button in Analytic Solver Desktop or **Optimize** then **Solve Complete Problem** in Analytic Solver Cloud. The message "Solver could not find a feasible solution" appears in red at the bottom of the Task Pane.

Still TBD Click **Reports – Optimization – Feasibility** on the Ribbon. A Feasibility Report worksheet is inserted into the workbook, immediately to the left of EXAMPLE2 – with the contents shown on the next page.



Microsoft Excel 14.0 Feasibility Report
Worksheet: [StandardExamples.xls]EXAMPLE2
Report Created: 4/29/2014 1:02:06 PM
Engine: Standard LP/Quadratic

Constraints that Make the Problem Infeasible

Cell	Name	Cell Value	Formula	Status	Slack
$C$11	Chassis No. Used	-1	$C$11<=$B$11	Binding	0
$D$9	Number to Build-> TV set	0	$D$9>=0	Binding	0
$E$9	Number to Build-> Stereo	-1	$E$9>=0	Violated	-1

Notice that the report highlights the constraint on Chassis parts used, and the non-negativity constraints on products that are built from Chassis parts. To satisfy the constraint C11 <= B11 where B11 is -1, we'd have to build a negative number of stereos (or TV sets). In a large model with thousands of constraints with no feasible solution, this report can focus your attention on the "trouble spots" – the constraints that make the problem infeasible. The set of constraints shown is called an IIS or *Irreducibly Infeasible Subset* of all the constraints. For more information, please see the Frontline Solvers Reference Guide.

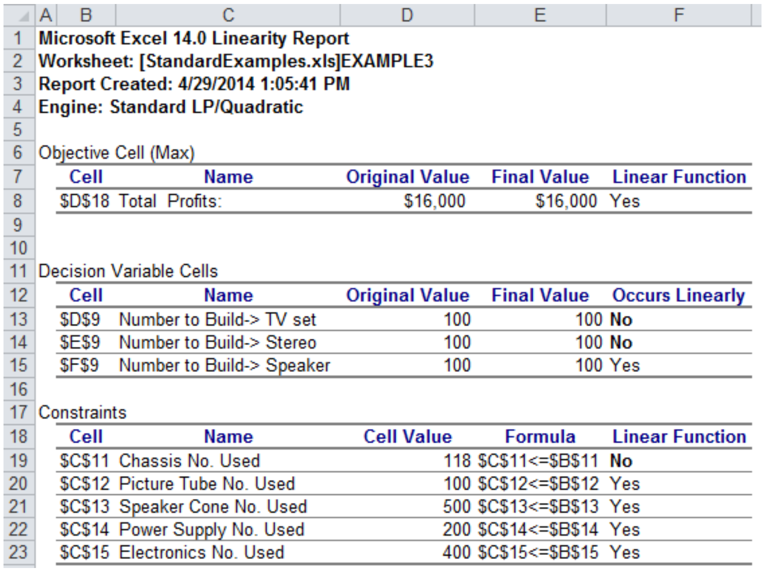
An 'Accidentally' Nonlinear Model

What if you intended to build a linear programming model, but you made a mistake and introduced a nonlinear function into the model? Even if you understand the requirements of linear functions (explained in the chapter "Mastering Conventional Optimization Concepts"), in a large model with thousands of variables and constraints, where each constraint is computed by a chain of formulas, an accidentally introduced nonlinearity may be difficult to find. Sometimes, a first-generation LP model is modified to make a previously fixed parameter into a decision variable – and this leads to an accidental multiplication of two variables.

Click the tab to display the EXAMPLE3 worksheet. This is the same Product Mix model as in EXAMPLE1, but the constraint left hand side formula for Chassis parts at C11 has been changed to read =SUMPRODUCT(D11:F11,D9:F9)^0.9. (The power function clearly makes the constraint nonlinear.) Click the **Optimize** button, or the down arrow beneath **Optimize** then **Solve Complete Problem**. The message "The linearity conditions required by this Solver Engine are not satisfied" appears in red at the bottom of the Task Pane. (Note that we're solving this model with the LP/Quadratic Solver.)

If you are using Analytic Solver Desktop, click **Reports – Optimization – Linearity** on the Ribbon. A Linearity Report worksheet is inserted into the workbook, immediately to the left of EXAMPLE3 – with the contents shown on the next page. Note: The Linearity Report is not supported in Analytic Solver Cloud.

The report highlights the constraint at C11 as a nonlinear function, and variables D9 and E9 that participate in calculation of this function. (F9 does not appear, because its coefficient at F11 is zero.) In a large model, this report can help you quickly find the source of an unintended nonlinearity.

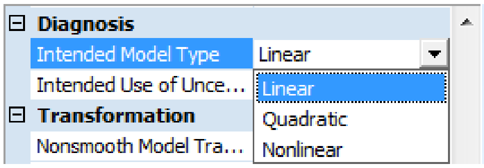


	A	B	C	D	E	F
1	**Microsoft Excel 14.0 Linearity Report**					
2	**Worksheet: [StandardExamples.xls]EXAMPLE3**					
3	**Report Created: 4/29/2014 1:05:41 PM**					
4	**Engine: Standard LP/Quadratic**					
5						
6	Objective Cell (Max)					
7		**Cell**	**Name**	**Original Value**	**Final Value**	**Linear Function**
8		$D$18	Total Profits:	$16,000	$16,000	Yes
9						
10						
11	Decision Variable Cells					
12		**Cell**	**Name**	**Original Value**	**Final Value**	**Occurs Linearly**
13		$D$9	Number to Build-> TV set	100	100	**No**
14		$E$9	Number to Build-> Stereo	100	100	**No**
15		$F$9	Number to Build-> Speaker	100	100	Yes
16						
17	Constraints					
18		**Cell**	**Name**	**Cell Value**	**Formula**	**Linear Function**
19		$C$11	Chassis No. Used	118	$C$11<=$B$11	**No**
20		$C$12	Picture Tube No. Used	100	$C$12<=$B$12	Yes
21		$C$13	Speaker Cone No. Used	500	$C$13<=$B$13	Yes
22		$C$14	Power Supply No. Used	200	$C$14<=$B$14	Yes
23		$C$15	Electronics No. Used	400	$C$15<=$B$15	Yes

The Linearity Report, which is available in all of the Analytic Solver Desktop subsets that perform optimization, is very useful, but it does have two drawbacks: (i) It highlights only a *constraint* that behaves nonlinearly – but if this constraint is calculated through a chain of formulas, it may not highlight the exact *cell formula* where the nonlinearity first occurs. (ii) It is based on a *numerical* test for linear or nonlinear behavior – a test that can be 'fooled' by a poorly scaled model. (A 'poorly scaled model' is one that computes values of the objective, constraints, or intermediate results that differ by many orders of magnitude, which can lead to inaccuracy in computer arithmetic.)

Using the PSI Interpreter to Check for Linearity

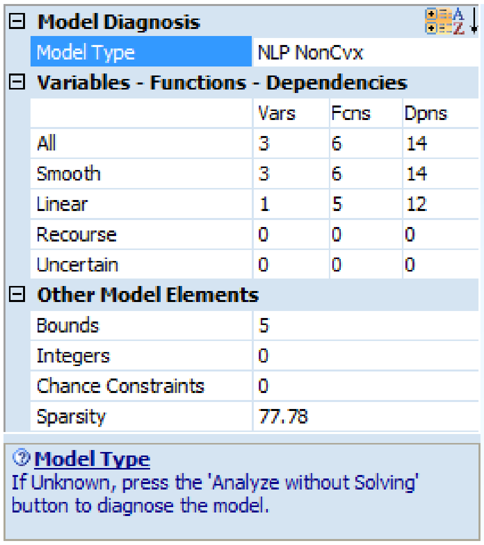
If you have Analytic Solver Comprehensive, Analytic Solver Optimization, or Analytic Solver Upgrade in Analytic Solver Desktop or Cloud, you can use the Polymorphic Spreadsheet Interpreter to perform a *symbolic* test for linearity that cannot be 'fooled' by scaling problems. Click the tab to display the EXAMPLE3 worksheet. On the Task Pane Platform tab, in the Diagnosis group of options, and ensure that the **Intended Model Type** is set to **Linear**:



⊟ **Diagnosis**
Intended Model Type: Linear

Now click the **Analyze** button in the Task Pane, or select the dropdown choice **Analyze Original Problem** below the **Optimize** button on the Ribbon. The Model Diagnosis portion of the Task Pane will pop up, showing the Model Type and related statistics, as shown on the next page.

Notice that the Model Type is "NLP NonCvx". If you're not sure about the meaning of any of the elements shown here, and you are using Analytic Solver Desktop, **click the underlined link** at the bottom of the Task Pane to display Help on that model element.



Model Diagnosis

Model Type	NLP NonCvx

Variables - Functions - Dependencies

	Vars	Fcns	Dpns
All	3	6	14
Smooth	3	6	14
Linear	1	5	12
Recourse	0	0	0
Uncertain	0	0	0

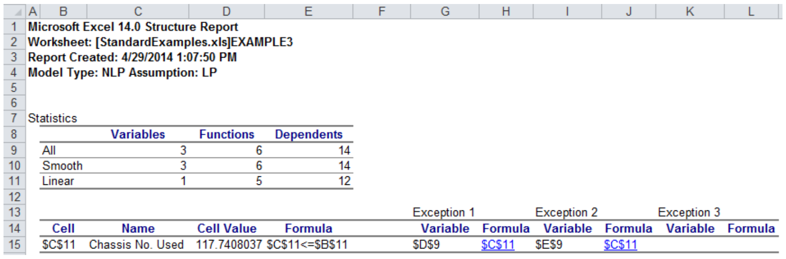
Other Model Elements

Bounds	5
Integers	0
Chance Constraints	0
Sparsity	77.78

Model Type

If Unknown, press the 'Analyze without Solving' button to diagnose the model.

Now click **Reports – Optimization – Structure** on the Ribbon. A Structure Report is inserted into the workbook, just to the left of EXAMPLE3. This report highlights the formula cells in your worksheet that violate the requirements of your **Intended Model Type** (in this case Linear). In other words, this report shows the formulas that make your model nonlinear.



Microsoft Excel 14.0 Structure Report
Worksheet: [StandardExamples.xls]EXAMPLE3
Report Created: 4/29/2014 1:07:50 PM
Model Type: NLP Assumption: LP

Statistics

	Variables	Functions	Dependents
All	3	6	14
Smooth	3	6	14
Linear	1	5	12

				Exception 1		Exception 2		Exception 3	
Cell	Name	Cell Value	Formula	Variable	Formula	Variable	Formula	Variable	Formula
$C$11	Chassis No. Used	117.7408037	$C$11<=$B$11	$D$9	$C$11	$E$9	$C$11		

For EXAMPLE3, this report again highlights the constraint at C11 and the two variables D9 and E9 (E9 in columns I and J, not shown to save space). But when a constraint or objective is calculated through a chain of formulas, this report normally finds the formula cell that first introduces a nonlinear relationship (such as a power operator, a function such as LOG, or multiplication of two cells that both depend on decision variables). EXAMPLE3!$C$11 in this report can be clicked to jump directly to that cell, so you can quickly inspect its formula.

Note: The Structure Report in Analytic Solver does not include the Name or Cell Value columns.

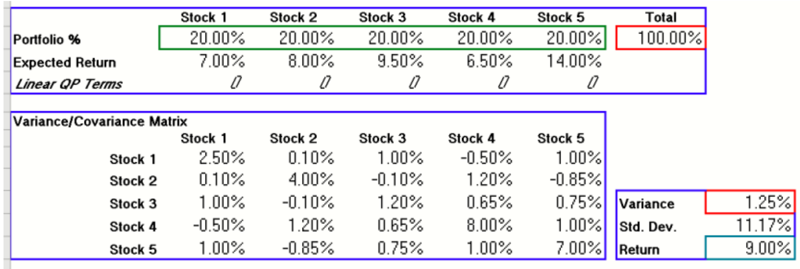
Nonlinear Optimization Examples

This section takes you on a quick tour of Analytic Solver features for nonlinear programming (NLP) and non-smooth optimization (NSP) problems, using the SOCP Barrier Solver, GRG Nonlinear Solver, and Evolutionary Solver. We highly recommend that you 'follow along' by opening and actually solving the examples in this section.

Portfolio Optimization: Quadratic Programming

The simplest kind of nonlinear optimization model is a quadratic programming (QP) model, with a quadratic objective and all linear constraints. Such a problem can be solved with a quadratic extension to the linear programming Simplex method – as in the LP/Quadratic Solver in Analytic Solver Comprehensive or Analytic Solver Optimization – or with the GRG Nonlinear Solver included in Analytic Solver Upgrade.

A classic example of this kind of model, shown in EXAMPLE4 in **StandardExamples.xlsx**, is a Markowitz-style portfolio optimization problem. In this model, we want to choose the mix of stocks to form an ‘efficient portfolio’ with the lowest possible risk (measured by portfolio variance) for a specified target rate of return. Click the EXAMPLE4 tab to display the model.

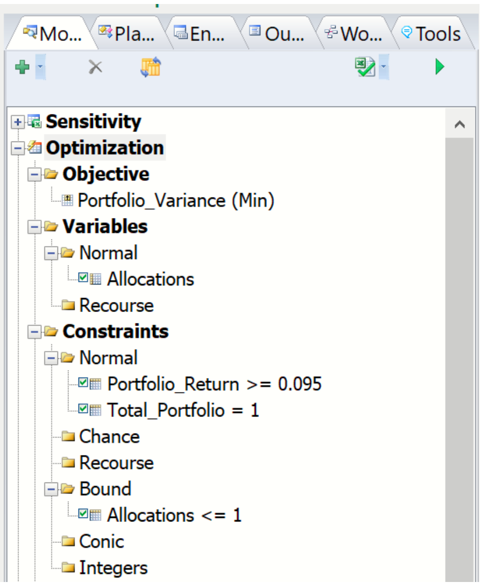


	Stock 1	Stock 2	Stock 3	Stock 4	Stock 5	Total
Portfolio %	20.00%	20.00%	20.00%	20.00%	20.00%	100.00%
Expected Return	7.00%	8.00%	9.50%	6.50%	14.00%	
Linear QP Terms	*0*	*0*	*0*	*0*	*0*	

Variance/Covariance Matrix	Stock 1	Stock 2	Stock 3	Stock 4	Stock 5
Stock 1	2.50%	0.10%	1.00%	-0.50%	1.00%
Stock 2	0.10%	4.00%	-0.10%	1.20%	-0.85%
Stock 3	1.00%	-0.10%	1.20%	0.65%	0.75%
Stock 4	-0.50%	1.20%	0.65%	8.00%	1.00%
Stock 5	1.00%	-0.85%	0.75%	1.00%	7.00%

Variance	1.25%
Std. Dev.	11.17%
Return	9.00%

This model uses Excel **defined names** for certain cell ranges: The objective cell I17 has the name Portfolio_Variance, which appears to the left of the formula bar when I17 is selected. Cell I19 has the name Portfolio_Return. The decision variables – the percentage of funds to invest in each of five stocks – are cells B9:F9, and this cell range has the name Allocations. Cell H9 has the name Total_Portfolio. These defined names appear in the Task Pane Model tab, as shown on the next page.



- **Sensitivity**
- **Optimization**
  - **Objective**
    - Portfolio_Variance (Min)
  - **Variables**
    - Normal
      - Allocations
    - Recourse
  - **Constraints**
    - Normal
      - Portfolio_Return >= 0.095
      - Total_Portfolio = 1
    - Chance
    - Recourse
    - Bound
      - Allocations <= 1
    - Conic
    - Integers

We can see immediately that this model will find values for the stock allocations that minimize portfolio variance, subject to constraints that (i) all funds are used (the sum of the stock allocations is 100% or 1.0) and (ii) the portfolio return is at least 9.5%.

Cell I17 computes the portfolio variance using a special function QUADPRODUCT, which is defined by Analytic Solver. This function can be used to compute any 'quadratic form' such as $x^TQx + cx$. In EXAMPLE4, the elements of the matrix Q are the covariances of returns of pairs of stocks, and the elements of the vector c (at B11:F11) are all zero. You could instead compute portfolio variance using a series of Excel formulas using multiplication and addition, but it's more convenient to use QUADPRODUCT.

Charts of the Objective and Constraints

Analytic Solver can help you visualize the shape of your problem functions (the objective and constraints). If you haven't been sure of the difference between a linear, quadratic, smooth nonlinear, and non-smooth function, you can use Analytic Solver to plot it.

As an example, select cell I19 (the Portfolio Return) on the worksheet, and choose **Decisions – Plot** from the Ribbon. This will plot I19 (whose formula is =SUMPRODUCT(Allocations,Stock_Returns)) as a function of the decision variables for Stock 1 through Stock 5, as shown on the next page.

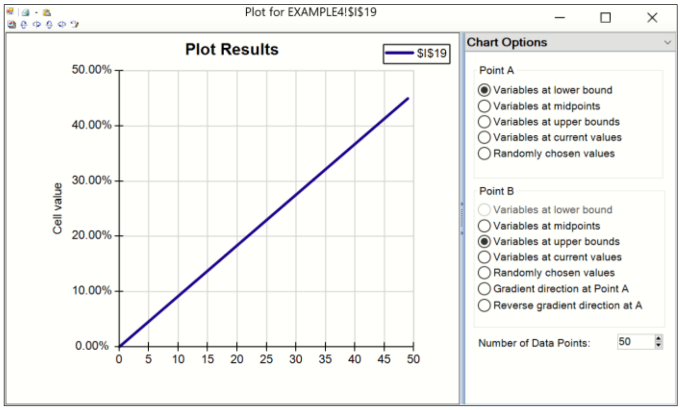


Chart Options

Point A
● Variables at lower bound
○ Variables at midpoints
○ Variables at upper bounds
○ Variables at current values
○ Randomly chosen values

Point B
○ Variables at lower bound
○ Variables at midpoints
● Variables at upper bounds
○ Variables at current values
○ Randomly chosen values
○ Gradient direction at Point A
○ Reverse gradient direction at A

Number of Data Points: 50

The SUMPRODUCT function is a linear function, so unsurprisingly, Portfolio Return plots as a straight. In contrast, select cell I17 (Portfolio Variance, the QUADPRODUCT function), and choose **Decisions – Plot** from the Ribbon. This function plots as a parabola – as expected for a quadratic function – showing only the positive values, since the decision variables are non-negative.

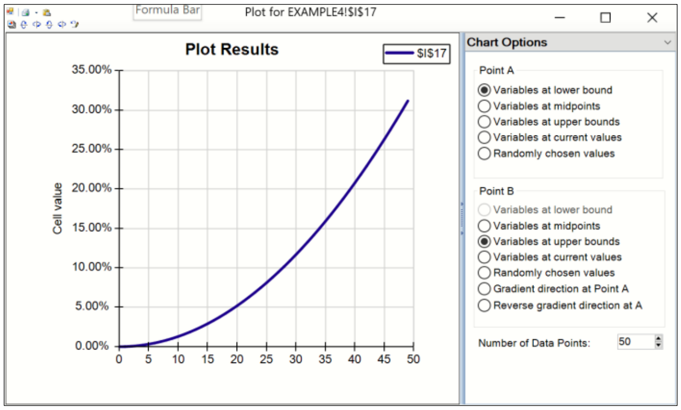


Chart Options

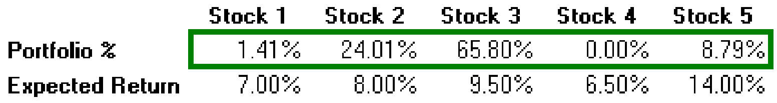
Point A
● Variables at lower bound
○ Variables at midpoints
○ Variables at upper bounds
○ Variables at current values
○ Randomly chosen values

Point B
○ Variables at lower bound
○ Variables at midpoints
● Variables at upper bounds
○ Variables at current values
○ Randomly chosen values
○ Gradient direction at Point A
○ Reverse gradient direction at A

Number of Data Points: 50

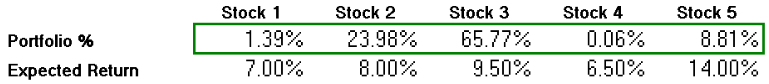
Both of these plots are actually projections of a function in five dimensions (Stock 1 through Stock 5) onto two dimensions, along a vector from Point A to Point B, shown in the side panel. You have many choices for Points A and B.

To find an optimal allocation of funds to stocks, just click the **Optimize** button. In an instant, the message “Solver found a solution. All constraints and optimality conditions are satisfied” appears in green at the bottom of the Task Pane, the Portfolio Variance is 0.85% and the Portfolio return is 9.5%, and the decision variables contain:



	Stock 1	Stock 2	Stock 3	Stock 4	Stock 5
Portfolio %	1.41%	24.01%	65.80%	0.00%	8.79%
Expected Return	7.00%	8.00%	9.50%	6.50%	14.00%

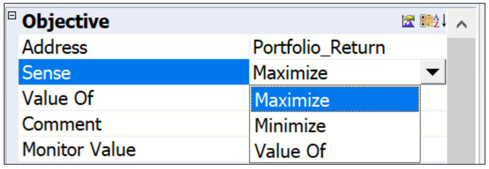
Solving with the GRG Nonlinear Solver yields the same values (to two decimal places) for the decision variables. If you have Analytic Solver Comprehensive, Analytic Solver Optimization or Analytic Solver Upgrade, you can also use the Standard SOCP Barrier Solver – just select it from the dropdown list at the top of the Task Pane Engine tab and click the **Optimize** button again. This yields nearly the same values for the decision variables, Portfolio Return at I19, and Portfolio Variance at I17.



	Stock 1	Stock 2	Stock 3	Stock 4	Stock 5
Portfolio %	1.39%	23.98%	65.77%	0.06%	8.81%
Expected Return	7.00%	8.00%	9.50%	6.50%	14.00%

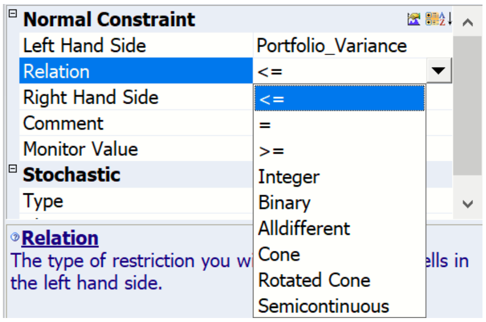
We can also “turn this problem around” and seek to maximize Portfolio Return, subject to a constraint that Portfolio Variance is (say) no more than 1%. To do this in Analytic Solver Desktop, we can edit the optimization model directly in the Task Pane:

1. Select “Portfolio_Variance (Min)” under Objective in the outline.
2. In the Objective Properties area below the outline, click the **Address** property and type or select cell I19 (or type “Portfolio_Return”). Change the **Sense** dropdown choice from Minimize to Maximize.



Objective	
Address	Portfolio_Return
Sense	Maximize
Value Of	
Comment	
Monitor Value	

3. Select “Portfolio_Return >= 0.095” under Constraints in the outline.
4. In the Normal Constraint Properties area below the outline, click the **Address** property and type or select cell I17 (or type “Portfolio_Variance”), change the Relation dropdown choice to <=, and type 0.01 as the Right Hand Side) value.

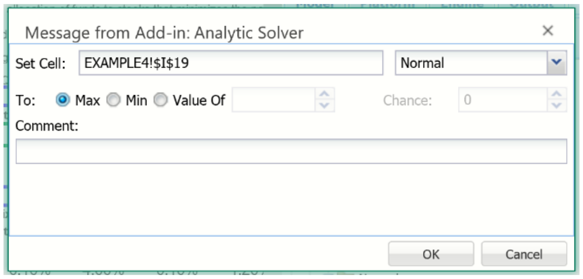


Normal Constraint	
Left Hand Side	Portfolio_Variance
Relation	<=
Right Hand Side	
Comment	
Monitor Value	
Stochastic	
Type	

Relation
The type of restriction you w ells in the left hand side.

Since the Task Pane is not editable in Analytic Solver Cloud, we must perform a few different steps to edit the model when using either of these products.

1. Double click Portfolio_Variance (Min) under Objective, change Set Cell to I19 and set the "sense" of the objective to Max, then click OK.



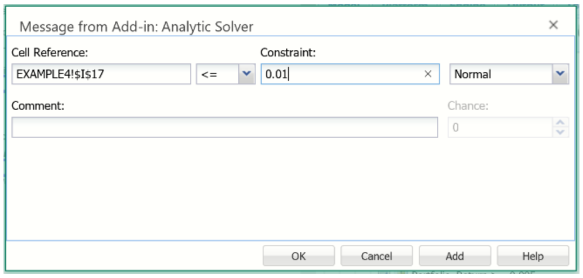
Message from Add-in: Analytic Solver

Set Cell: EXAMPLE4!$I$19 Normal

To: ● Max ○ Min ○ Value Of Chance: 0

Comment:

2. Double click Portfolio_Return >= 0.095 under Constraints, change Cell Reference to I17, change the constraint sense to <= and type 0.01 as the Right Hand Side Constraint value. Then click OK.



Message from Add-in: Analytic Solver

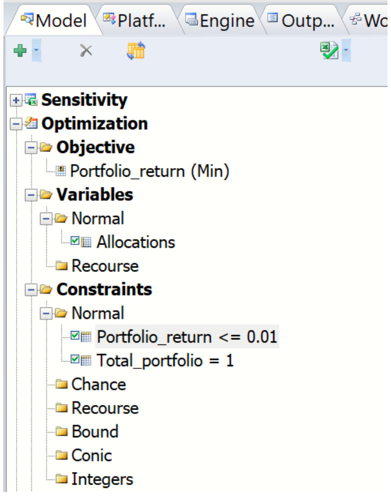
Cell Reference: EXAMPLE4!$I$17 <=

Constraint: 0.01 Normal

Comment:

Chance: 0

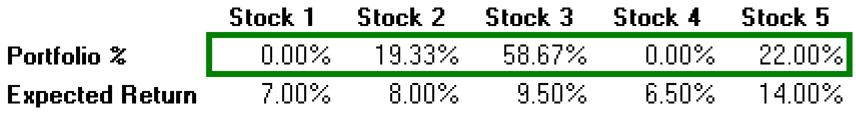
The Task Pane Model tab should now appear similar to the one shown below.



- Sensitivity
- Optimization
  - Objective
    - Portfolio_return (Min)
  - Variables
    - Normal
      - ☑ Allocations
    - Recourse
  - Constraints
    - Normal
      - ☑ Portfolio_return <= 0.01
      - ☑ Total_portfolio = 1
    - Chance
    - Recourse
    - Bound
    - Conic
    - Integers

This problem now has a *linear* objective and a *quadratic* constraint; it is called a *quadratically constrained* or QCP problem. It can no longer be solved by the methods used in the LP/Quadratic Solver (a quadratic extension of the Simplex method), but it can be solved by the GRG Nonlinear Solver, or (more efficiently) by the SOCP Barrier Solver.

Solving with the SOCP Barrier Solver yields a Portfolio Return of 10.2%, a Portfolio Variance of 1% as expected, and the following values for the decision variables:



	Stock 1	Stock 2	Stock 3	Stock 4	Stock 5
Portfolio %	0.00%	19.33%	58.67%	0.00%	22.00%
Expected Return	7.00%	8.00%	9.50%	6.50%	14.00%

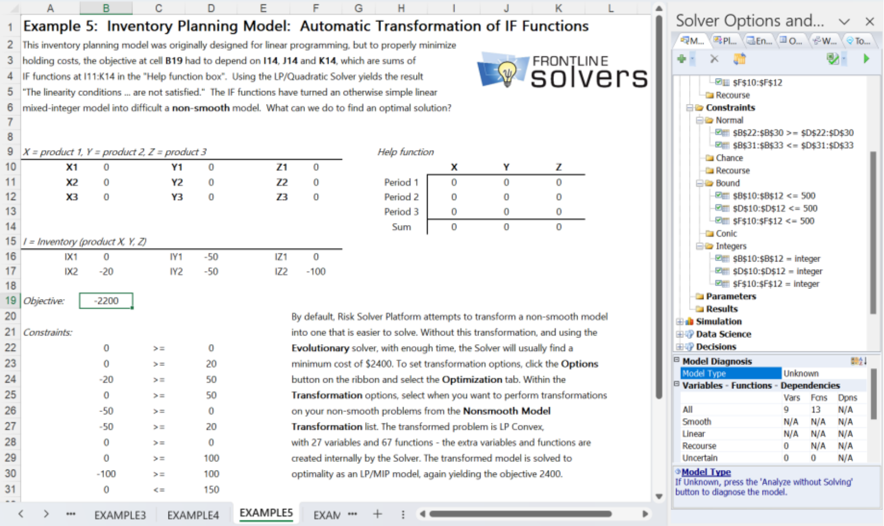
In "Multiple Parameterized Optimizations" in the chapter "Getting Results: Optimization," we return to EXAMPLE4 and show how to automatically vary the Portfolio Return threshold, solve multiple instances of this portfolio optimization model, and create a chart of the efficient frontier.

The original EXAMPLE4 QP model can be 'scaled up' to portfolios of many thousands of stocks, and solved quickly to optimality by the Large-Scale LP/QP Solver, the XPRESS Solver and the MOSEK Solver. The modified QCP model can also be scaled up to portfolios of thousands of stocks, and solved quickly to optimality by the MOSEK Solver.

A Model with IF Functions

EXAMPLE5 in the **StandardExamples.xlsx** workbook was adapted from an actual user model, and illustrates a common situation seen by Frontline Systems in technical support. This model was *meant* to be a linear mixed-integer (LP/MIP) model, but the user calculated the objective function based on several IF functions. IF functions are certainly not linear functions – in fact they aren't even smooth nonlinear functions!

One of the remarkable things about Analytic Solver is that, in Automatic mode, it will transform this model to a linear mixed-integer model, and solve it to optimality in seconds, without any user intervention. But in this Guide, we'll first see how Analytic Solver behaves *without* Automatic mode. Click the EXAMPLE5 tab to display the model.



Example 5: Inventory Planning Model: Automatic Transformation of IF Functions

This inventory planning model was originally designed for linear programming, but to properly minimize holding costs, the objective at cell **B19** had to depend on **I14**, **J14** and **K14**, which are sums of IF functions at I11:K14 in the "Help function box". Using the LP/Quadratic Solver yields the result "The linearity conditions ... are not satisfied." The IF functions have turned an otherwise simple linear mixed-integer model into difficult a **non-smooth** model. What can we do to find an optimal solution?

X = product 1, Y = product 2, Z = product 3

X1	0	**Y1**	0	**Z1**	0
X2	0	**Y2**	0	**Z2**	0
X3	0	**Y3**	0	**Z3**	0

Help function

	X	Y	Z
Period 1	0	0	0
Period 2	0	0	0
Period 3	0	0	0
Sum	0	0	0

I = Inventory (product X, Y, Z)

IX1	0	IY1	-50	IZ1	0
IX2	-20	IY2	-50	IZ2	-100

Objective: -2200

Constraints:

0	>=	0
0	>=	20
-20	>=	50
0	>=	50
-50	>=	0
-50	>=	20
0	>=	0
0	>=	100
-100	>=	100
0	<=	150

By default, Risk Solver Platform attempts to transform a non-smooth model into one that is easier to solve. Without this transformation, and using the **Evolutionary** solver, with enough time, the Solver will usually find a minimum cost of $2400. To set transformation options, click the **Options** button on the ribbon and select the **Optimization** tab. Within the **Transformation** options, select when you want to perform transformations on your non-smooth problems from the **Nonsmooth Model Transformation** list. The transformed problem is LP Convex, with 27 variables and 67 functions - the extra variables and functions are created internally by the Solver. The transformed model is solved to optimality as an LP/MIP model, again yielding the objective 2400.

Model Diagnosis

Model Type: Unknown

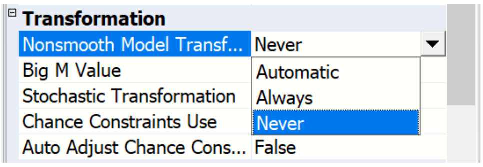
Variables - Functions - Dependencies

	Vars	Fcns	Dpns
All	9	13	N/A
Smooth	N/A	N/A	N/A
Linear	N/A	N/A	N/A
Recourse	0	N/A	N/A
Uncertain	0	0	N/A

Model Type
If Unknown, press the 'Analyze without Solving' button to diagnose the model.

The model has 9 decision variables, all constrained to be integer. The general constraints are linear functions of the decision variables. But the objective depends on I14, J14 and K14, each of which is a sum of three IF functions.

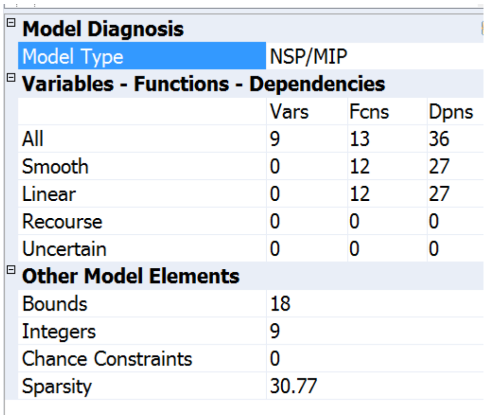
To turn off Automatic mode, in the Task Pane Platform tab **Transformation** group of options, set the **Nonsmooth Model Transformation** option to **Never**.



Transformation	
Nonsmooth Model Transf...	Never
Big M Value	
Stochastic Transformation	
Chance Constraints Use	
Auto Adjust Chance Cons...	False

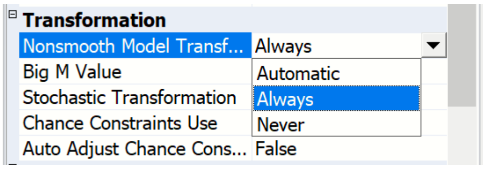
If we now click the **Optimize** button with the LP/Quadratic Solver selected, the message "The linearity conditions required by this Solver engine are not satisfied" appears in red at the bottom of the Task Pane.

If we click the **Analyze** button in the Task Pane, or select the dropdown choice **Analyze Original Problem** below the **Optimize** button on the Ribbon, Analytic Solver reports that the model is nonsmooth with integers (NSP/MIP), as shown on the next page.



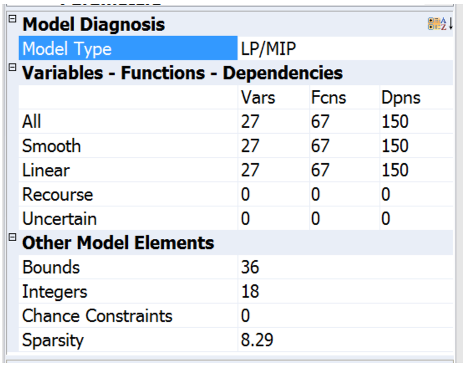
Model Diagnosis			
Model Type	NSP/MIP		
Variables - Functions - Dependencies			
	Vars	Fcns	Dpns
All	9	13	36
Smooth	0	12	27
Linear	0	12	27
Recourse	0	0	0
Uncertain	0	0	0
Other Model Elements			
Bounds	18		
Integers	9		
Chance Constraints	0		
Sparsity	30.77		

This type of model commonly arises, but is difficult to solve with the standard Excel Solver, or with Analytic Solver Upgrade. But Analytic Solver Comprehensive, Analytic Solver Optimization or Analytic Solver Upgrade in Analytic Solver Desktop or Cloud can *transform* the model to *eliminate* the IF functions, replacing them with additional binary and continuous variables and linear constraints that have the *same effect* on the model as the IF functions. The additional variables and constraints don't appear on the worksheet, but they are handled internally by the PSI Interpreter and the selected Solver engine. To do this, we simply set the **Nonsmooth Model Transformation** option to Automatic:



Transformation	
Nonsmooth Model Transf...	Always
Big M Value	
Stochastic Transformation	
Chance Constraints Use	
Auto Adjust Chance Cons...	False

If we again analyze the model by clicking the **Analyze** button in the Task Pane, or by selecting the dropdown choice **Analyze Transformed Problem** below the **Optimize** button on the Ribbon, Analytic Solver reports that the the *transformed* model has 27 variables (18 of them integer) and 67 constraints, as shown on the next page. But it's now a linear programming (LP) model – so it can be solved by the LP/Quadratic Solver. By solving this transformed model, we can find a solution to the original model on the EXAMPLE5 worksheet.



Model Diagnosis

Model Type	LP/MIP

Variables - Functions - Dependencies

	Vars	Fcns	Dpns
All	27	67	150
Smooth	27	67	150
Linear	27	67	150
Recourse	0	0	0
Uncertain	0	0	0

Other Model Elements

Bounds	36
Integers	18
Chance Constraints	0
Sparsity	8.29

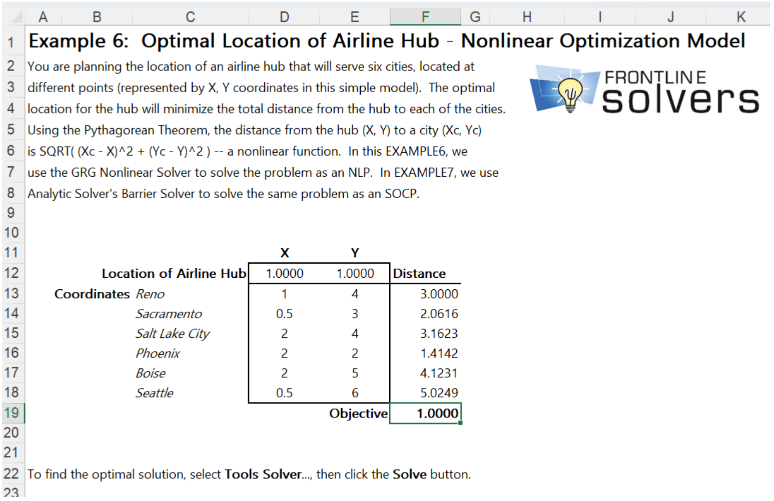
When we now click the **Optimize** button, the message "Solver found a solution. All constraints and optimality conditions are satisfied" appears in green at the bottom of the Task Pane, and the objective function value at cell B19 is 2400. The message indicates that the Solver has found a **proven optimal** solution for the original problem with IF functions.

The PSI Interpreter can automatically transform models containing IF, AND, OR, NOT, ABS, MIN and MAX, and relations <, <=, >= and > into equivalent models where these non-smooth functions are eliminated. When this transformation is sufficient to make the overall model linear – as it was in this case – you can use the LP/Quadratic Solver or a large-scale LP/MIP Solver engine to solve the problem.

A Model with Cone Constraints

In the EXAMPLE6 and EXAMPLE7 worksheets, we wish to solve a model to optimize the location of an airline hub. The hub will serve six cities, each of which is located at some latitude and longitude (in these models, we use simple X, Y coordinates). We want to choose a location (X, Y coordinate) for the airline hub that will minimize the maximum distance from the hub to any of the six cities. The distance from the hub (X, Y) to a city (Xc, Yc) is SQRT((Xc - X)^2 + (Yc - Y)^2). The model is pictured below. In EXAMPLE6, we model and solve the problem using the GRG Nonlinear Solver. In EXAMPLE7, which requires Analytic Solver Comprehensive, Analytic Solver Optimization or Analytic Solver Upgrade, we model the problem using **cone constraints**, and solve it using the SOCP Barrier Solver.

Modeling the Problem with Nonlinear Functions



Example 6: Optimal Location of Airline Hub - Nonlinear Optimization Model

You are planning the location of an airline hub that will serve six cities, located at different points (represented by X, Y coordinates in this simple model). The optimal location for the hub will minimize the total distance from the hub to each of the cities. Using the Pythagorean Theorem, the distance from the hub (X, Y) to a city (Xc, Yc) is SQRT((Xc - X)^2 + (Yc - Y)^2) -- a nonlinear function. In this EXAMPLE6, we use the GRG Nonlinear Solver to solve the problem as an NLP. In EXAMPLE7, we use Analytic Solver's Barrier Solver to solve the same problem as an SOCP.

		X	Y	
	Location of Airline Hub	1.0000	1.0000	**Distance**
Coordinates	*Reno*	1	4	3.0000
	Sacramento	0.5	3	2.0616
	Salt Lake City	2	4	3.1623
	Phoenix	2	2	1.4142
	Boise	2	5	4.1231
	Seattle	0.5	6	5.0249
			Objective	**1.0000**

To find the optimal solution, select **Tools Solver...**, then click the **Solve** button.

In EXAMPLE6, our decision variables are the (X, Y) coordinates of the airline hub, at cells D12:E12, and a third decision variable at F19. Cell F19 also serves as the objective function, to be minimized. We calculate the distance from the hub to each of the six cities in cells F13:F18. We define a block of constraints F13:F18 <= F19 – this means that the distance from the hub to each city must be less than or equal to the decision variable F19, which the Solver will minimize. The effect of these constraints is to cause the Solver to find values for D12:E12 – the (X, Y) coordinates of the hub – that minimize the maximum distance to any of the cities.

Click the EXAMPLE6 tab to display the model, and click Analyze – Analytic Original Problem. We see that the model is diagnosed as smooth nonlinear (NLP), with 3 variables and 7 functions. Notice that the Standard GRG Nonlinear Solver is selected.

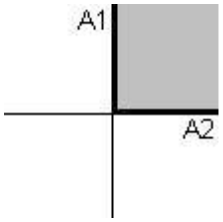
Click the **Optimize** button. The message "Solver found a solution" appears in green at the bottom of the Task Pane. The optimal location for the airline hub is at X = 1.25, Y = 4.0, and the maximum distance from this location to any of the six cities is 2.1360 units.

Modeling the Problem with Cone Constraints

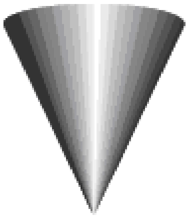
In EXAMPLE4, we solved the simplest kind of nonlinear optimization problem, a portfolio optimization QP model with a quadratic objective and all linear constraints, and a variation that was a QCP model, with a quadratic constraint . We saw that we could solve the problem with the GRG Nonlinear Solver, but we could also solve both variations of the problem with the **SOCP Barrier Solver** – and we learned that we could more easily 'scale up' the SOCP problem to large size, with thousands of stocks or more.

We can do the same thing with the airline hub problem. The model is a little more complex, with more variables and constraints – but this allows the Solver to exploit the structure of the model, and solve it more quickly and reliably. To do this, we will explicitly use cone constraints.

We have actually used a simple kind of cone constraint before – in EXAMPLE1, our first linear programming problem, at the beginning of this chapter. In that model, we used an objective and constraints that were linear functions of the decision variables, and *non-negativity* constraints on the variables. These constraints specified that the variables must lie within a simple kind of cone, called the *non-negative orthant*. This first order cone places a bound on the *L1*-norm of the vector of decision variables:



In our reformulation of the airline hub problem in EXAMPLE7, we'll specify that groups of variables must lie within a *second order cone* (also called a Lorentz cone, or "ice cream cone") – a ***convex*** set that looks like this:



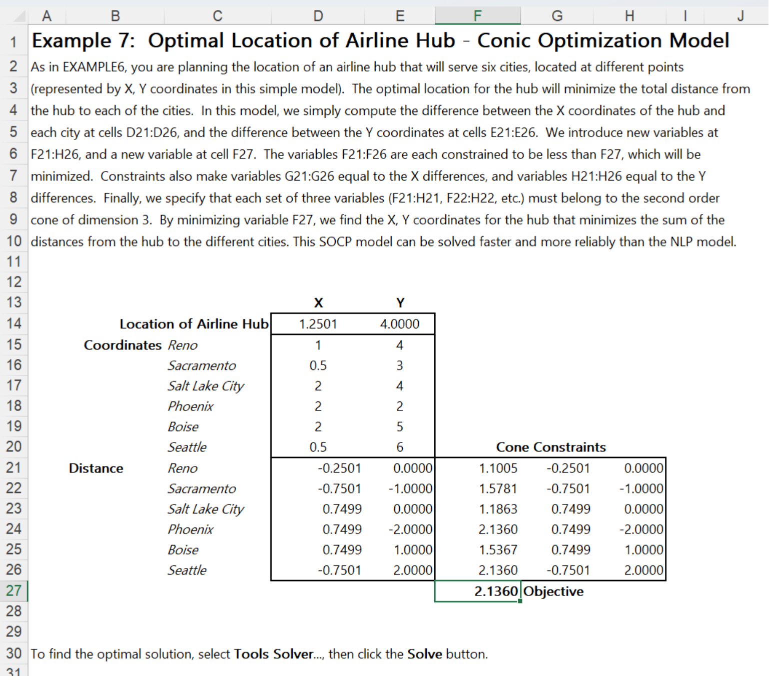
This cone places a bound on the *L2*-norm of the vector of decision variables. If A1:A3 are variables that lie within this cone, then

A1 >= SQRT(SUMSQ(A2:A3)) must hold. A problem with a linear objective and linear or second order cone (SOC) constraints is called a *second order cone programming* (SOCP) problem; it is always a ***convex*** optimization problem.

Second order cone programming is the *natural generalization* of linear programming. It offers the same advantages of convexity and scalability to large problems offered by linear programming – but for a broader class of models. For history buffs, Premium Solver Platform V6.0 was the first commercial software product to offer broad support for second order cone programming.

The EXAMPLE7 worksheet is shown below. In this model, we compute the simple *differences* Xc – X and Yc – Y between the airline hub coordinates and the coordinates of each city. We define new variables F21:F26, G21:G26, and H21:H26. We use constraints to make G21:G26 equal to the Xc – X differences, and H21:H26 equal to the Yc – Y differences.

Then we specify that each set of three variables (for example F21, G21, H21) must belong to a *second order cone* – hence F21 >= SQRT(SUMSQ(G21:H21)) – and likewise for F22, G22, H22, etc. Finally, we use one more variable F27 as the objective to be minimized, and we add a block of constraints F21:F26 <= F27 – much as we did in EXAMPLE6. The effect of these constraints is to cause the Solver to find values for the (X, Y) coordinates of the hub – now at D14:D15 – that minimize the maximum distance to any of the cities.



Example 7: Optimal Location of Airline Hub - Conic Optimization Model

As in EXAMPLE6, you are planning the location of an airline hub that will serve six cities, located at different points (represented by X, Y coordinates in this simple model). The optimal location for the hub will minimize the total distance from the hub to each of the cities. In this model, we simply compute the difference between the X coordinates of the hub and each city at cells D21:D26, and the difference between the Y coordinates at cells E21:E26. We introduce new variables at F21:H26, and a new variable at cell F27. The variables F21:F26 are each constrained to be less than F27, which will be minimized. Constraints also make variables G21:G26 equal to the X differences, and variables H21:H26 equal to the Y differences. Finally, we specify that each set of three variables (F21:H21, F22:H22, etc.) must belong to the second order cone of dimension 3. By minimizing variable F27, we find the X, Y coordinates for the hub that minimizes the sum of the distances from the hub to the different cities. This SOCP model can be solved faster and more reliably than the NLP model.

		X	Y			
Location of Airline Hub		1.2501	4.0000			
Coordinates	*Reno*	1	4			
	Sacramento	0.5	3			
	Salt Lake City	2	4			
	Phoenix	2	2			
	Boise	2	5			
	Seattle	0.5	6	**Cone Constraints**		
Distance	*Reno*	-0.2501	0.0000	1.1005	-0.2501	0.0000
	Sacramento	-0.7501	-1.0000	1.5781	-0.7501	-1.0000
	Salt Lake City	0.7499	0.0000	1.1863	0.7499	0.0000
	Phoenix	0.7499	-2.0000	2.1360	0.7499	-2.0000
	Boise	0.7499	1.0000	1.5367	0.7499	1.0000
	Seattle	-0.7501	2.0000	2.1360	-0.7501	2.0000
				2.1360	**Objective**	

To find the optimal solution, select **Tools Solver...**, then click the **Solve** button.

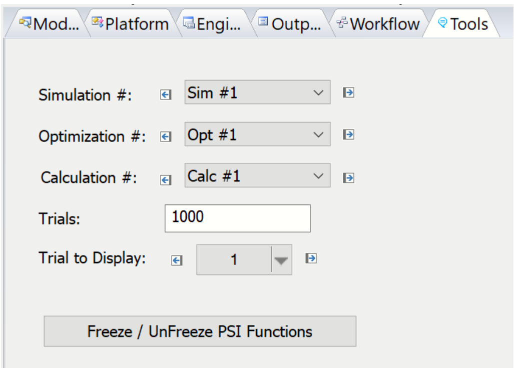
Click the EXAMPLE7 tab to display the model, and click **Analyze – Analyze Original Model**. We see that this formulation of the model is diagnosed as second order cone programming problem (SOCP), with 21 variables and 19 functions. Notice that the SOCP Barrier Solver is selected to solve the problem.

Click the **Optimize** button. After a moment, the same optimal solution as in EXAMPLE6 is found. In this simple example, both the Standard GRG Nonlinear Solver and the SOCP Barrier Solver solve the problem in sub-second time. But if we were to **scale up** this model to large size – say with 1,000 cities – the SOCP Barrier Solver would likely show a speed advantage.

Solving an Optimization Model using Excel Online or Google Sheets

In late 2014, Frontline Systems released Solver App for use with Excel 2013 and Solver Add-on for use with Google Sheets. With the Solver App you can define and solve optimization problems in your Excel workbook, using Excel Online in Office 365, the Excel Web App in SharePoint 2016 or 2013, or desktop Excel 2016 or 2013. With the Solver Add-on, you can define and solve optimization problems in Google Sheets. You can create and solve models on tablets, phones, or anywhere a Web browser can be used. Solver models that you may already have created in old or new versions of Microsoft Excel, are automatically recognized by the Solver App and, after the existing worksheet is published, with the Solver Add-on.

In Analytic Solver Desktop, the **Freeze** and **Thaw** buttons were combined into a single button which now appears on the Tools tab of the Solver Task Pane, as shown in the screenshot below. (Freeze/Unfreeze is not supported in Analytic Solver Cloud.)



Simulation #: Sim #1

Optimization #: Opt #1

Calculation #: Calc #1

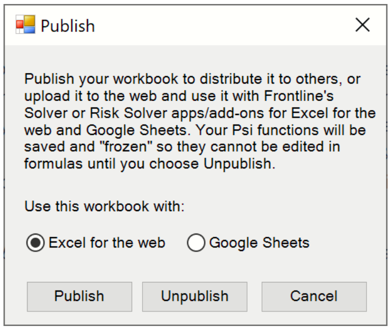
Trials: 1000

Trial to Display: 1

The “Freeze/Unfreeze” button can be used to prepare a workbook for distribution to other users who don’t have Analytic Solver or its subsets installed: All formulas containing Psi function calls (which would yield #NAME? for other users) are replaced by their values, and the formulas are saved, so they can be restored later by choosing “UnFreeze” (equivalent to the old “Thaw”).

But the major use of the **Freeze/Unfreeze** button in ASP and later, is to prepare a workbook for use with our Solver/Risk Solver Apps for Excel Online and our Solver/Risk Solver Add-on for Google Sheets (the online spreadsheets have limited or no support for user-defined functions).

Let’s go back to the model on the Example1 tab used in the section above, click the Freeze/UnFreeze button on the Tools tab of the Solver Task Pane to display the following dialog.



Publish your workbook to distribute it to others, or upload it to the web and use it with Frontline's Solver or Risk Solver apps/add-ons for Excel for the web and Google Sheets. Your Psi functions will be saved and "frozen" so they cannot be edited in formulas until you choose Unpublish.

Use this workbook with:

● Excel for the web ○ Google Sheets

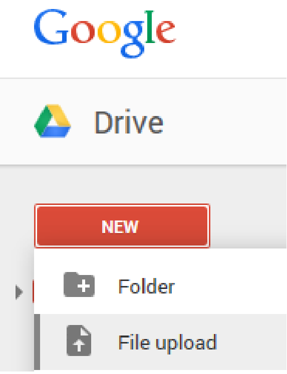
Select **Google Sheets** then **Publish**. (If you are a user of Excel Online, please see the example below.) Make sure to **save** the workbook before uploading to Google Drive.

Note: Only the Active Worksheet is published. To solve a model on a different worksheet within the same Workbook (say Example2), you’ll need to publish that worksheet separately. This is important when solving an optimization model. If solving a simulation model, this does not apply. (For an example on how to solve a simulation model with Excel Online or Google Sheets, please see the next chapter.)

Log on to your Google Drive account and upload Standard Example.xlsx, then open this file in Google Sheets.

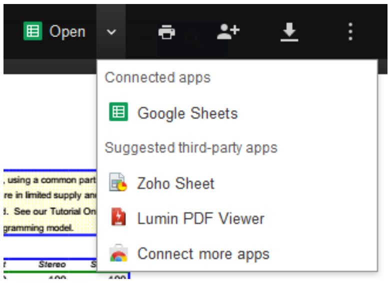
Uploading and Opening Files in Google Sheets

Click the orange New button then select File Upload to upload the StandardExamples.xlsx spreadsheet to Google Drive.

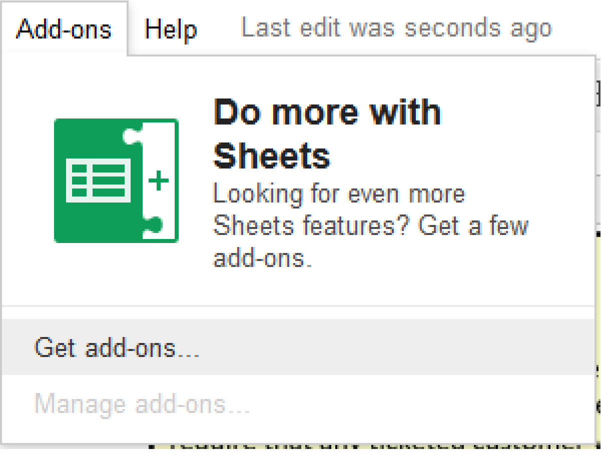


Browse to the location of the file, typically C:\ProgramData\Frontline Systems\Examples. StandardExamples.xlsx will appear under My Drive.

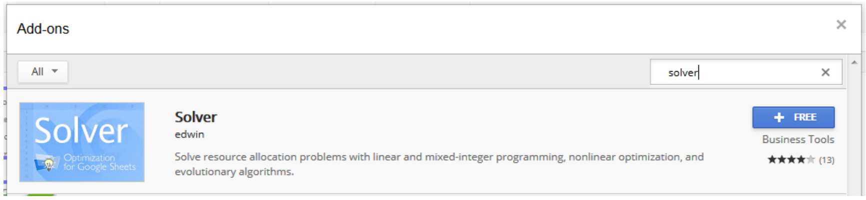
Double click the file to open in Google Drive Preview Viewer (shown below). Click Open – Google Sheets.



To add the Solver Add-on, choose the menu option Add-ons - Get Add-ons...


Add-ons
Help
Last edit was seconds ago
Do more with Sheets
Looking for even more Sheets features? Get a few add-ons.
Get add-ons...
Manage add-ons...

In the Add-ons dialog, scroll until you see *Solver*. Click to select and install.



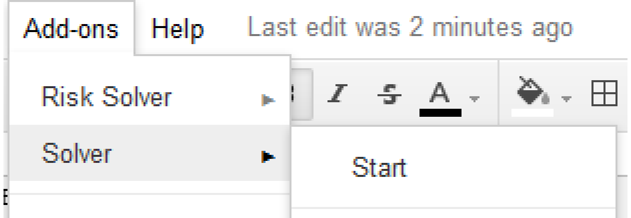
Search: solver

Solver

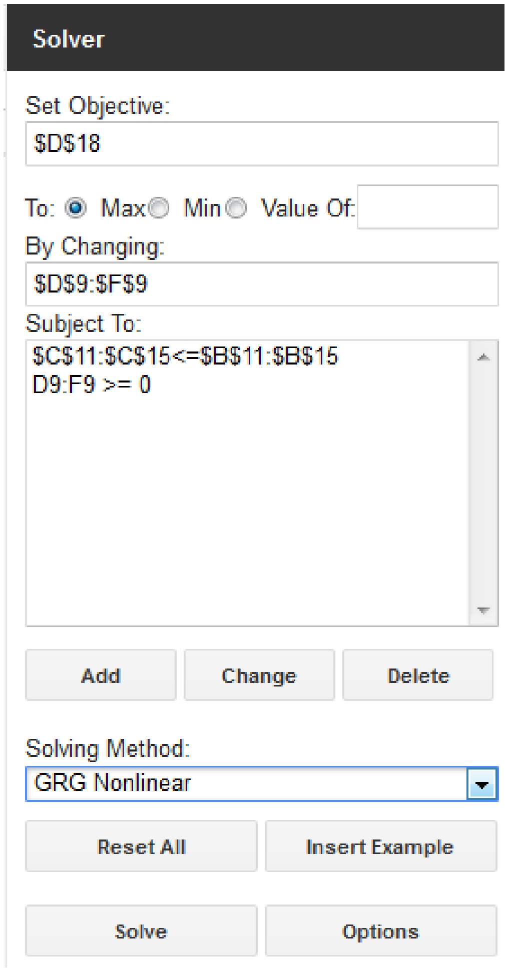
edwin

Solve resource allocation problems with linear and mixed-integer programming, nonlinear optimization, and evolutionary algorithms.

Once these steps are finished, Solver should now appear on the Add-ons menu.



Choose Add-ons - Solver - Start. The Solver Task Pane (Google calls this a Sidebar) will appear containing the Solver model: objective, variables, and constraints.



Set Objective: $D$18

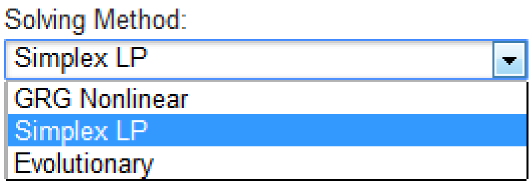
To: ● Max ○ Min ○ Value Of:

By Changing: $D$9:$F$9

Subject To:
$C$11:$C$15<=$B$11:$B$15
D9:F9 >= 0

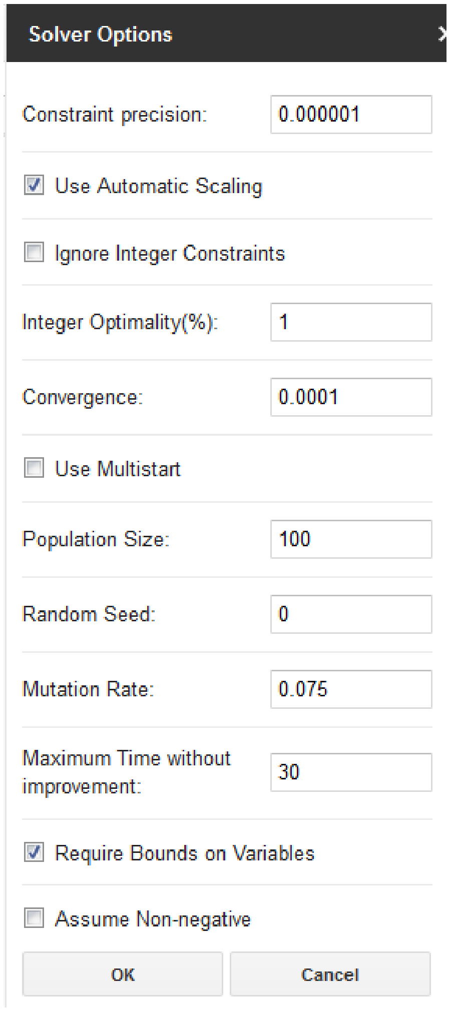
Solving Method: GRG Nonlinear

Since this is a linear model, click the down arrow to the right of *GRG Nonlinear* under *Solving Method* to select the **LP/Quadratic** engine.


Solving Method: Simplex LP

Click Solve to run the Simplex LP engine to solve the linear model. The Solver Add-on solves the model, inserting the final variable values of 200/200/0 in cells D9:F9 and finding an objective equal to $25,000. Note: Reports, such as Answer Report, Sensitivity Report, and/or Limits reports, are not supported in the Solver Add-on.

The Solver Sidebar behaves much like the Excel Solver dialog. To change a constraint, simply highlight the desired constraint and click *Change*. To delete a constraint, highlight the desired constraint and click *Delete*. Click *Reset All* to clear the objective, all variables and constraints, and all options settings. Select *Options* to change an engine option such as Constraint Precision, Automatic Scaling, Convergence, etc. For more information on each of these options, please see the *Engine Option* chapter in *Frontline Solvers Reference Guide*.


Solver Options

Constraint precision: 0.000001

☑ Use Automatic Scaling

☐ Ignore Integer Constraints

Integer Optimality(%): 1

Convergence: 0.0001

☐ Use Multistart

Population Size: 100

Random Seed: 0

Mutation Rate: 0.075

Maximum Time without improvement: 30

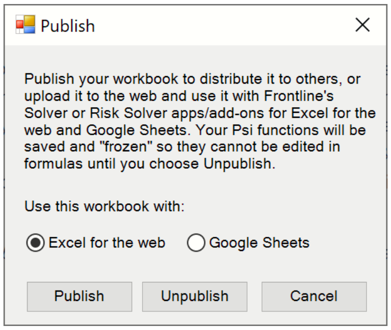
☑ Require Bounds on Variables

☐ Assume Non-negative

Now let’s Publish this same model to Excel Online. Let’s go back to Excel and StandardExamples.xlsx. Click the Freeze/UnFreeze button on the Tools tab of the Solver Task Pane. Note: When using Publish/Unpublish, the worksheet may not contain CTRL+SHIFT+ENTER arrays.

Confirm that *Excel Online* is selected. Click **Publish** to publish the worksheet to Excel Online.

Note: Only the Active Worksheet is published. To solve a model on a different worksheet within the same workbook (say Example2), you'll need to publish that worksheet separately. This is important when solving an optimization model. If solving a simulation model, this does not apply. (For an example on how to solve a simulation model with Excel Online or Google Sheets, please see the next chapter.)



Publish your workbook to distribute it to others, or upload it to the web and use it with Frontline's Solver or Risk Solver apps/add-ons for Excel for the web and Google Sheets. Your Psi functions will be saved and "frozen" so they cannot be edited in formulas until you choose Unpublish.

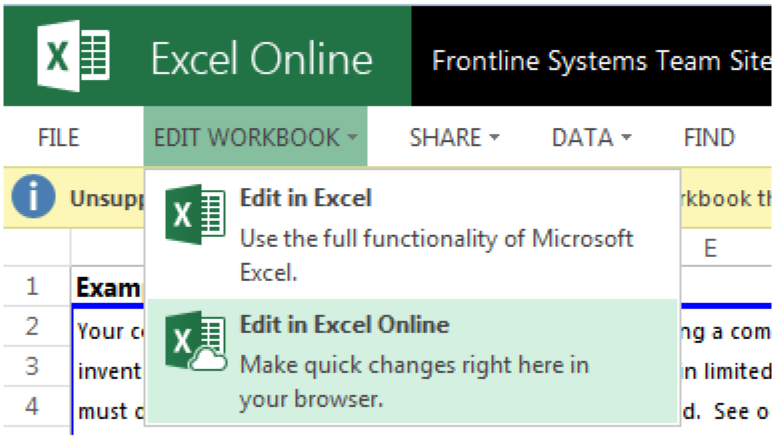
Use this workbook with:

● Excel for the web ○ Google Sheets

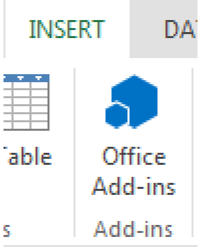
Now **Save** the file as **StandardExamples**.**xlsx** on your *SharePoint* or *Office 365* site.

Note: In order to use the Solver App, your file must be saved on SharePoint or Office 365 site. This app cannot solve models saved on your local hard drive.

Open the workbook using Excel Online, then click Edit Workbook – Edit in Excel Online.

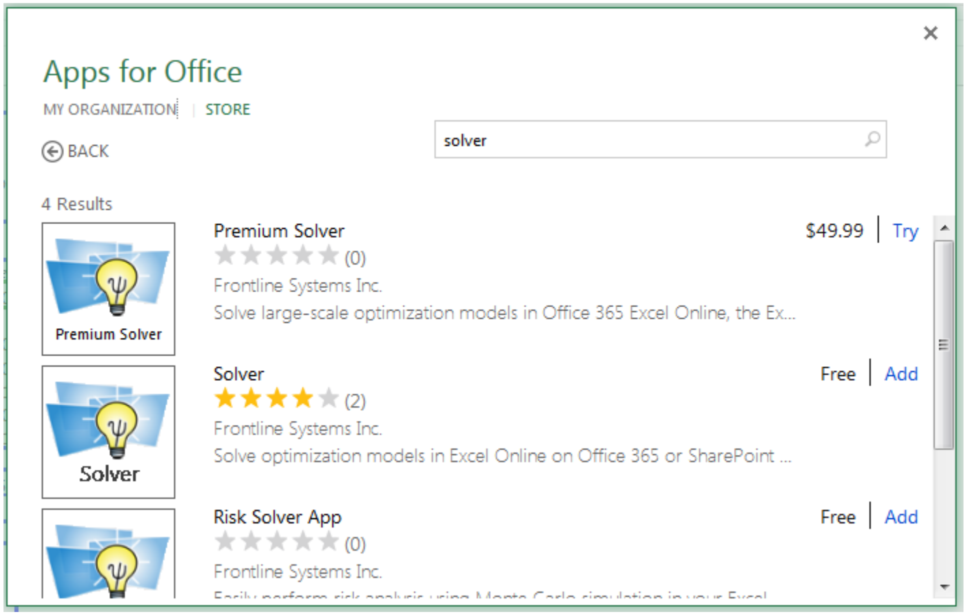

Excel Online
Frontline Systems Team Site
FILE
EDIT WORKBOOK
SHARE
DATA
FIND
Edit in Excel
Use the full functionality of Microsoft Excel.
Edit in Excel Online
Make quick changes right here in your browser.

Click Insert – Office Add-ins.



Select the Store tab and search for Solver, then select (2nd app in the list) and Install.

Please see the next chapter for an example on the use of the Risk Solver App.



Premium Solver

Frontline Systems Inc.

Solve large-scale optimization models in Office 365 Excel Online, the Ex...

Solver

Frontline Systems Inc.

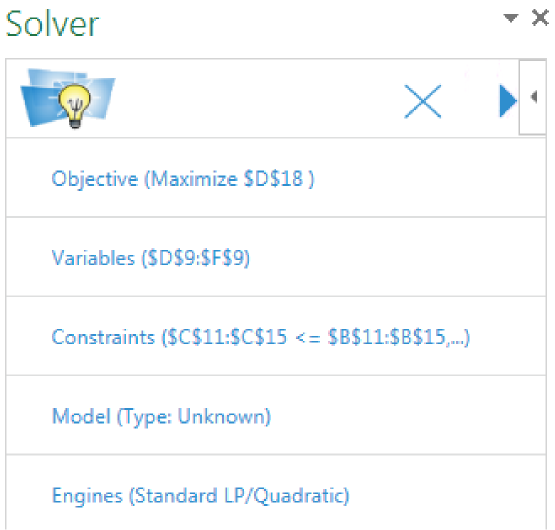
Solve optimization models in Excel Online on Office 365 or SharePoint ...

Risk Solver App

Frontline Systems Inc.

Easily perform risk analysis using Monte Carlo simulation in your Excel

The Solver App task pane opens to the right on your screen.



Objective (Maximize $D$18)

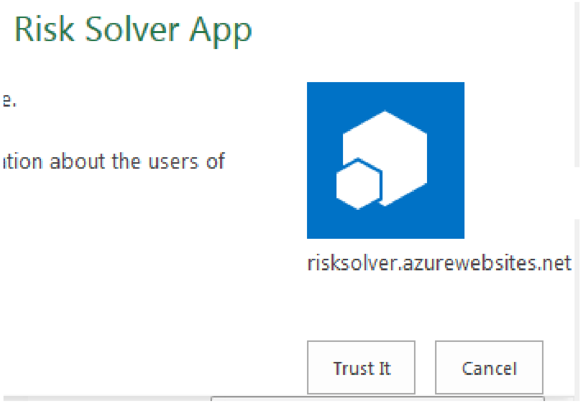
Variables ($D$9:$F$9)

Constraints ($C$11:$C$15 <= $B$11:$B$15,...)

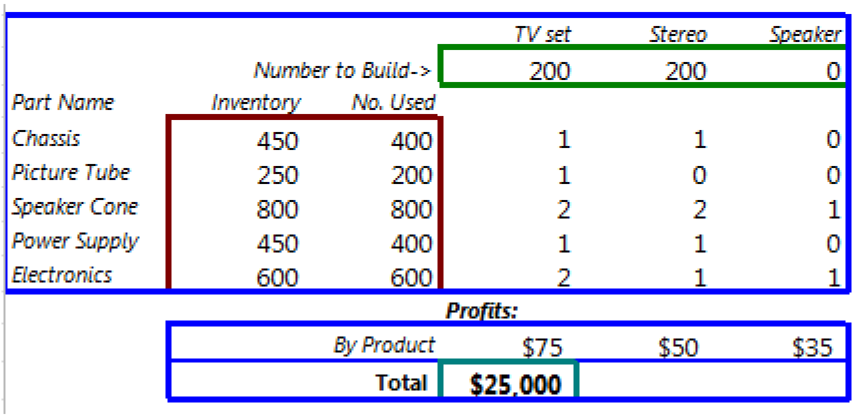
Model (Type: Unknown)

Engines (Standard LP/Quadratic)

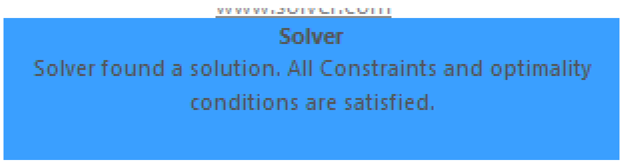
Simplex LP has already been selected for Solving Method. Click *Solve* to run this engine to solve this simple linear model. If running for the first time, you will be asked to **Trust** the App.



Solver inserts the final variable values 200/200/0 into cells D9:F9 and adds the final result message “Solver found a solution. All Constraints and optimality conditions are satisfied.” to the bottom of the Solver pane.



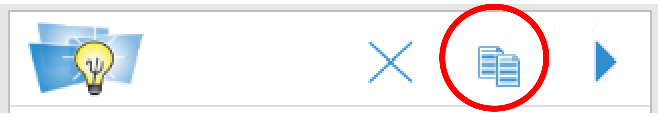
			TV set	Stereo	Speaker
		Number to Build->	200	200	0
Part Name	Inventory	No. Used			
Chassis	450	400	1	1	0
Picture Tube	250	200	1	0	0
Speaker Cone	800	800	2	2	1
Power Supply	450	400	1	1	0
Electronics	600	600	2	1	1
		Profits:			
		By Product	$75	$50	$35
		Total	$25,000		



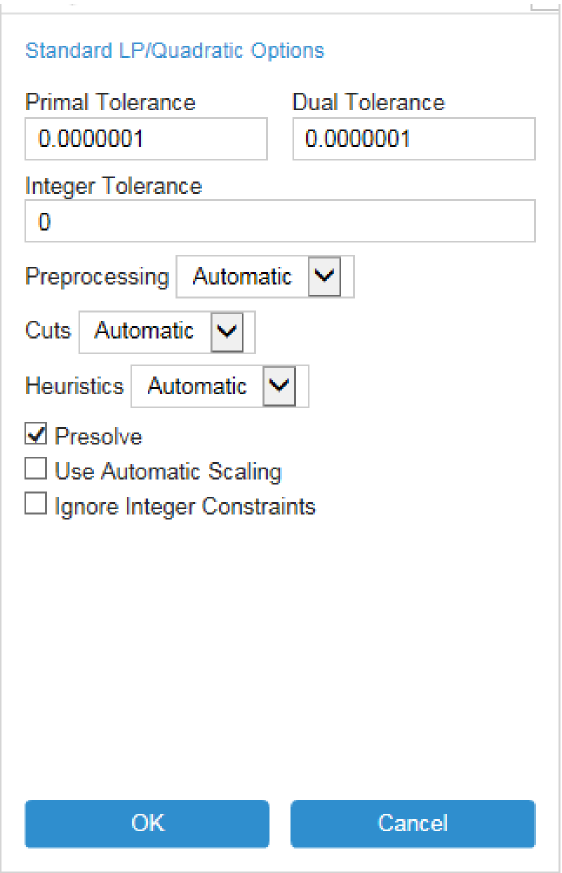
Solver

Solver found a solution. All Constraints and optimality conditions are satisfied.

Note: The Answer Report and Sensitivity Report are *only* supported when using the Solver App within Excel 2016. You can create the reports after Solver has found a solution by clicking the Create Report icon located in the title bar of the Solver App. If using Excel 2013, Excel for iPad & iPhone or Excel Online, you will not have the option to create reports.



This task pane behaves much like the Excel Solver. To Add a constraint, click *Constraints*, then *Add*. To Change or Delete a constriant, select the desired constraint and click *Change* or *Delete*. To change an engine option, click *Engine* then *Options*. For more information on each of these options, please see the *Engine Option* chapter in *Frontline Solvers Reference Guide*.



Standard LP/Quadratic Options

Primal Tolerance: 0.0000001

Dual Tolerance: 0.0000001

Integer Tolerance: 0

Preprocessing: Automatic

Cuts: Automatic

Heuristics: Automatic

☑ Presolve

☐ Use Automatic Scaling

☐ Ignore Integer Constraints

Examples: Simulation and Risk Analysis

Introduction

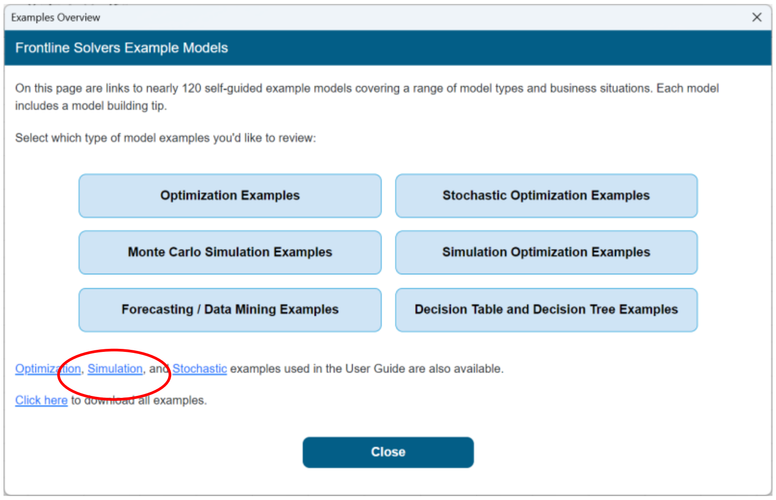
This chapter introduces simulation and risk analysis in Analytic Solver, with a series of examples. Each of these examples with the exception of the last one can be used in Analytic Solver Simulation.

- "A First Simulation Example" takes you step-by-step through the process of building and analyzing a risk analysis model using Monte Carlo simulation.
- "Airline Revenue Management Example" takes you through three versions of an airline yield management problem, to illustrate a single simulation, multiple parameterized simulations, and simulation optimization.

A First Simulation Example

Building a simulation model in Analytic Solver is straightforward: You simply build a conventional spreadsheet model, designed for 'what-if' analysis. Next, you identify the ***inputs*** to your model that are uncertain, and use PSI Distribution functions to describe the uncertainty. Then, you identify the ***outputs*** of special interest (such as Net Profit), and use PSI Statistics functions to examine or summarize how they behave in light of the uncertainty.

To open this example in Analytic Solver Desktop or Analytic Solver Cloud, click **Help – Examples** on the Ribbon, which opens a workbook Frontline Example Models Overview.xlsx. Click the Simulation link.


Examples Overview
Frontline Solvers Example Models
Optimization Examples
Stochastic Optimization Examples
Monte Carlo Simulation Examples
Simulation Optimization Examples
Forecasting / Data Mining Examples
Decision Table and Decision Tree Examples
Close

On this page are links to nearly 120 self-guided example models covering a range of model types and business situations. Each model includes a model building tip.

Select which type of model examples you'd like to review:

Optimization, Simulation, and Stochastic examples used in the User Guide are also available.

Click here to download all examples.

We'll build the simulation model step-by-step in this section using the model in the worksheet, **BusinessForecast**. To view the completed model, see the 2nd tab in the workbook, **BusinessForecastPsi**.

Uncertain Variables

In any problem, there are factors or inputs that you can control – for example, the price you set for a product, and factors or inputs that you *cannot* control – for example, customer demand, interest rates, etc. Analytic Solver uses **uncertain variables** (*random variables* in mathematics) to represent inputs that are uncertain and beyond your control. (It uses decision variables to represent factors or inputs that you *can* control.)

Uncertain Functions

You will also have outputs or results of interest – such as Net Profit – that you can compute, using formulas that depend on the factors influencing the problem – possibly both decision variables and uncertain variables. We'll use the term **uncertain functions** for quantities whose calculation depends on uncertain variables (in mathematics these are called *functions of random variables*).

A Business Planning Example

We'll illustrate the process of building a simulation model step by step, using a simple business planning example. Imagine you are the marketing manager for a firm that is planning to introduce a new product. You need to estimate the first year profit from this product, which will depend on:

- Sales in units
- Price per unit sold
- Unit manufacturing cost
- Fixed costs and overhead

Profit will be calculated as **Profit = Sales * (Price - Unit cost) - Fixed costs**.

Fixed costs are known to be **$120,000**. But the other factors all involve some uncertainty. Sales in units can cover quite a range, and the selling price per unit will depend on competitor actions. Unit manufacturing costs will also vary depending on vendor prices and production experience.

Uncertain Variables: Sales and Price

Based on your market research, you believe that there are equal chances that the market demand will be Slow, OK, or Hot for this product:

- In the Slow market demand scenario, you expect to sell 5**0,000** units at an average price of **$11.00** per unit.
- In the OK market scenario, you expect to sell **75,000** units, but you'll likely realize a lower average selling price of **$10.00** per unit.
- In the Hot market scenario, you expect to sell **100,000** units, but this will bring in competitors who will drive down the average selling price to **$8.00** per unit.

Since the scenarios are equally likely, your *average* volume is **75,000** units, and your *average* price per unit is **$9.67**. But think: How likely is this *average* case? (Will it ever actually occur?)

Uncertain Variables: Unit Cost

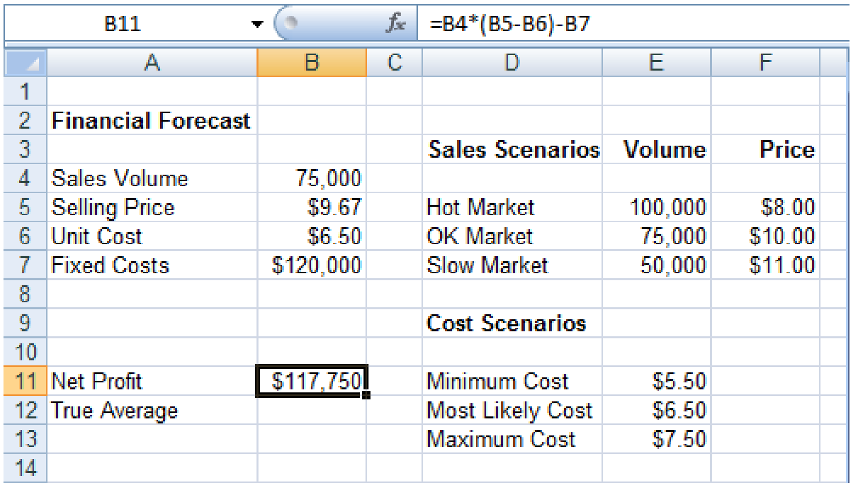
Your firm’s production manager advises you that unit costs may be anywhere from **$5.50** to **$7.50**, with a most likely cost of **$6.50**. The most likely cost is also the *average* cost.

Uncertain Function: Net Profit

Net Profit is calculated as **Profit = Sales * (Price - Unit cost) -Fixed costs**. Sales, Price and Unit costs are all uncertain variables, so Net Profit is an uncertain function.

A What-If Spreadsheet Model

At this point, you can summarize the problem in the Excel model pictured below, which calculates Net Profit based on *average* sales, price, and unit cost.



B11 =B4*(B5-B6)-B7

	A	B	C	D	E	F
1						
2	**Financial Forecast**					
3				**Sales Scenarios**	**Volume**	**Price**
4	Sales Volume	75,000				
5	Selling Price	$9.67		Hot Market	100,000	$8.00
6	Unit Cost	$6.50		OK Market	75,000	$10.00
7	Fixed Costs	$120,000		Slow Market	50,000	$11.00
8						
9				**Cost Scenarios**		
10						
11	Net Profit	$117,750		Minimum Cost	$5.50	
12	True Average			Most Likely Cost	$6.50	
13				Maximum Cost	$7.50	
14						

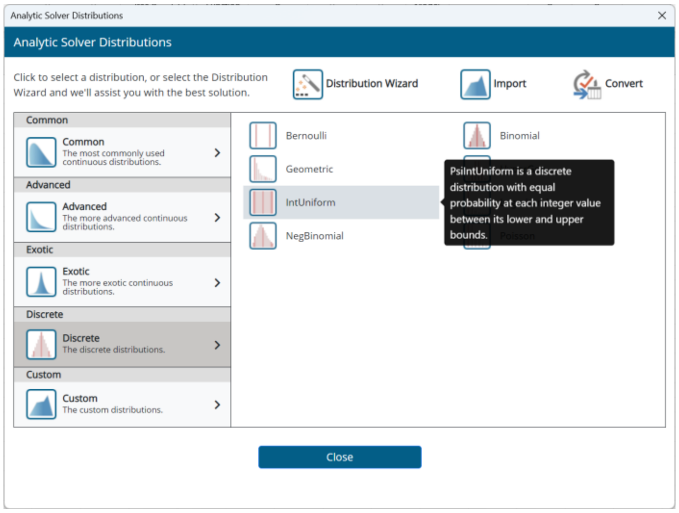
The Net Profit figure of $117,750 (=B4*(B5-B6)-B7) calculated by this model, based on *average* values for the uncertain factors, is quite misleading, as we’ll see in a moment. The ***true average*** Net Profit is closer to $93,000! In the spirit of Prof. Sam Savage’s book *The Flaw of Averages*, we’ll refer to the model above as the Flawed Average model.

Defining a Simulation Model

To “stress test” this model, we need to replace the fixed Sales, Price and Unit cost amounts with *variable* amounts that reflect their uncertainty.

Since there are equal chances that the market will be Slow, OK, or Hot, we want to create an uncertain variable that selects among these three possibilities, by drawing a random number – say 1, 2 or 3 – with equal probability. We can do this easily in Analytic Solver using an **integer uniform** probability distribution. We’ll then base our Sales Volume and Selling Price on this uncertain variable.

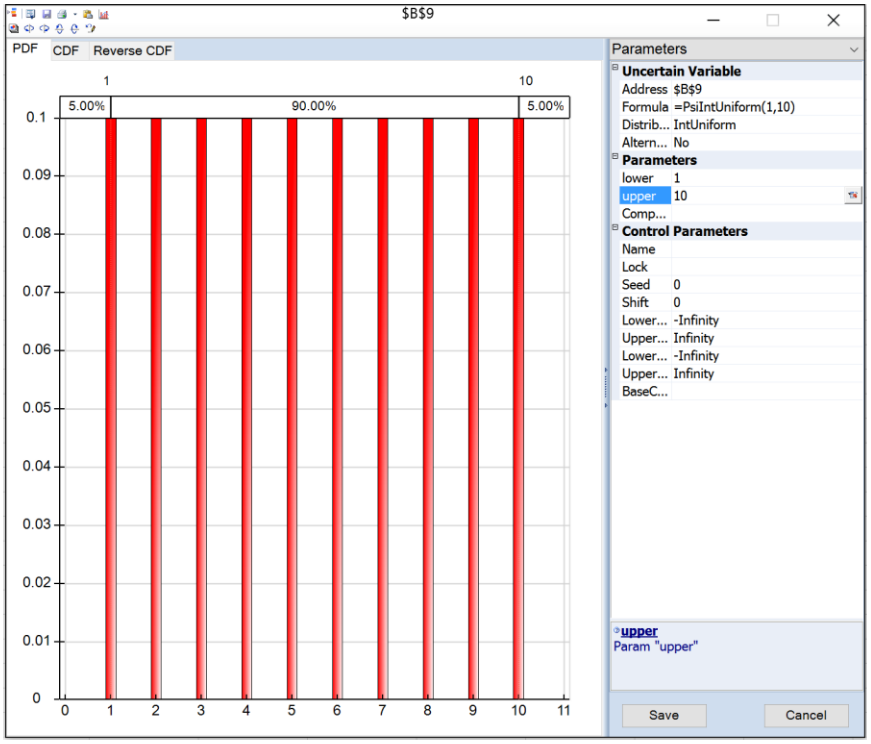
If using Analytic Solver Desktop, we’ll choose a currently empty cell, B9, to hold this distribution by clicking on it and selecting **Distributions - Discrete** on the Ribbon. This displays a cascading menu, as shown on the next page.


Analytic Solver Distributions
Analytic Solver Distributions
Distribution Wizard
Import
Convert
Common
Common
The most commonly used continuous distributions.
Advanced
Advanced
The more advanced continuous distributions.
Exotic
Exotic
The more exotic continuous distributions.
Discrete
Discrete
The discrete distributions.
Custom
Custom
The custom distributions.
Bernoulli
Binomial
Geometric
IntUniform
NegBinomial
Close

Click to select a distribution, or select the Distribution Wizard and we'll assist you with the best solution.

PsiIntUniform is a discrete distribution with equal probability at each integer value between its lower and upper bounds.

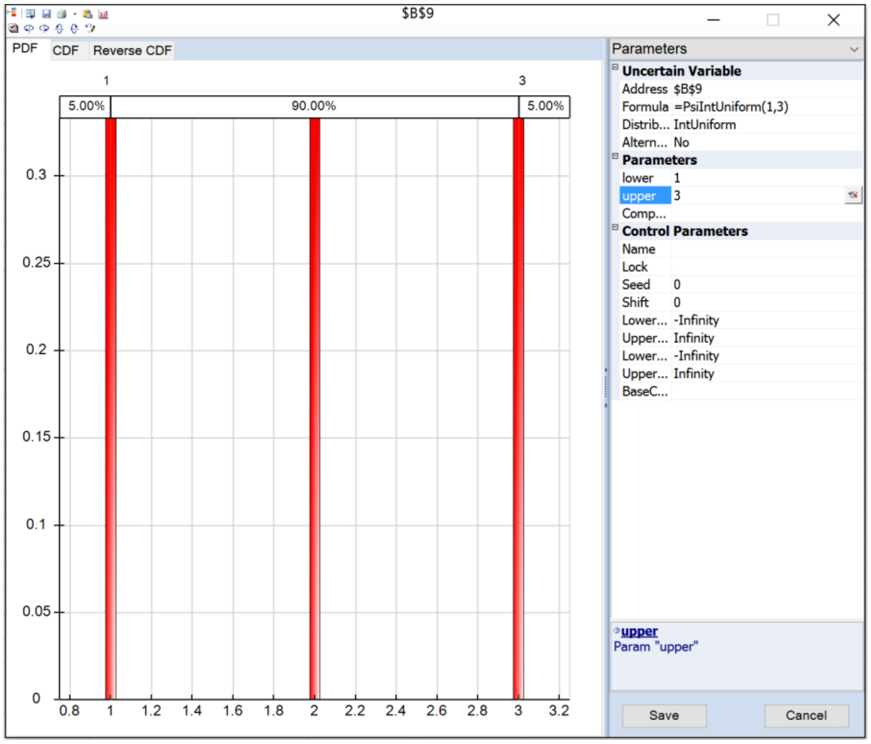
A sample drawn from a *discrete* distribution is always one of a set of discrete values, such as integer numbers. We click to choose **IntUniform** from the gallery or menu. Analytic Solver displays the Uncertain Variable dialog for a integer uniform distribution, initially with parameters lower 0 and upper 10:



Uncertain Variable
Address $B$9
Formula =PsiIntUniform(1,10)
Distrib... IntUniform
Altern... No
Parameters
lower 1
upper 10
Comp...
Control Parameters
Name
Lock
Seed 0
Shift 0
Lower... -Infinity
Upper... Infinity
Lower... -Infinity
Upper... Infinity
BaseC...

upper
Param "upper"

We click in the Value column and change the parameters to read **lower** 1 and **upper** 3. This means that on each trial, we'll draw a number 1, 2 or 3 from this distribution. As we do, the chart of probability mass (density) is updated:



Uncertain Variable
Address $B$9
Formula =PsiIntUniform(1,3)
Distrib... IntUniform
Altern... No
Parameters
lower 1
upper 3
Comp...
Control Parameters
Name
Lock
Seed 0
Shift 0
Lower... -Infinity
Upper... Infinity
Lower... -Infinity
Upper... Infinity
BaseC...

upper
Param "upper"

When we click the **Save** icon in the dialog toolbar in Analytic Solver Desktop or the Save button at the bottom of the dialog in Analytic Solver Cloud, a formula **=PsiIntUniform(1,3)** is written to B9. B9 is now an *uncertain variable*. For each trial of the Monte Carlo simulation, a different value – either 1, 2 or 3 – will appear in cell B9. When we run a Monte Carlo simulation of (say) 1,000 trials, 1,000 different values of 1, 2 or 3 will be sampled for this cell's value.

Now, we need to select one of the three sales scenarios in formulas for Sales Volume and Selling Price. With cell B4 selected, we enter the formula:

=CHOOSE(B9,E5,E6,E7) for Sales Volume

This will cause B4 to return 100,000, 75,000, or 50,000, depending on the value in B9. Next, with cell B5 selected, we enter the formula:

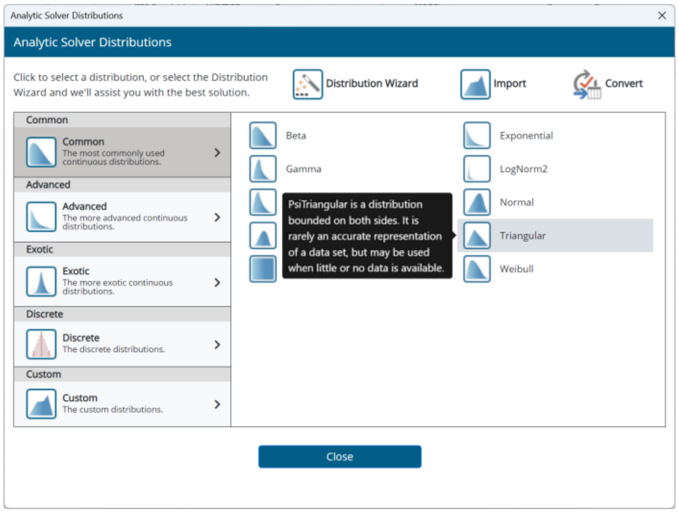
=CHOOSE(B9,F5,F6,F7) for Selling Price

This will cause B5 to return $8, $10 or $11, depending on the value in B9.

Notice that the values returned by B4 and B5 are related, or ***correlated***: Higher sales volume is accompanied by lower selling prices, and vice versa. If we had scenarios with 100,000 units sold at $11 each, our model would be unrealistic. Analytic Solver has more versatile ways to specify correlation between uncertain variables, but this approach is easy to understand in this example.

Next, we'll deal with Unit Cost. We have not just three, but *many* possible values for this variable: It can be anywhere from $5.50 to $7.50, with a most likely cost of $6.50. A crude but effective way to model this is to use a *triangular* distribution. Analytic Solver provides a function called **PsiTriangular()** for this distribution.

With cell B6 selected, we select **Distributions – Common – Triangular** from the Ribbon.



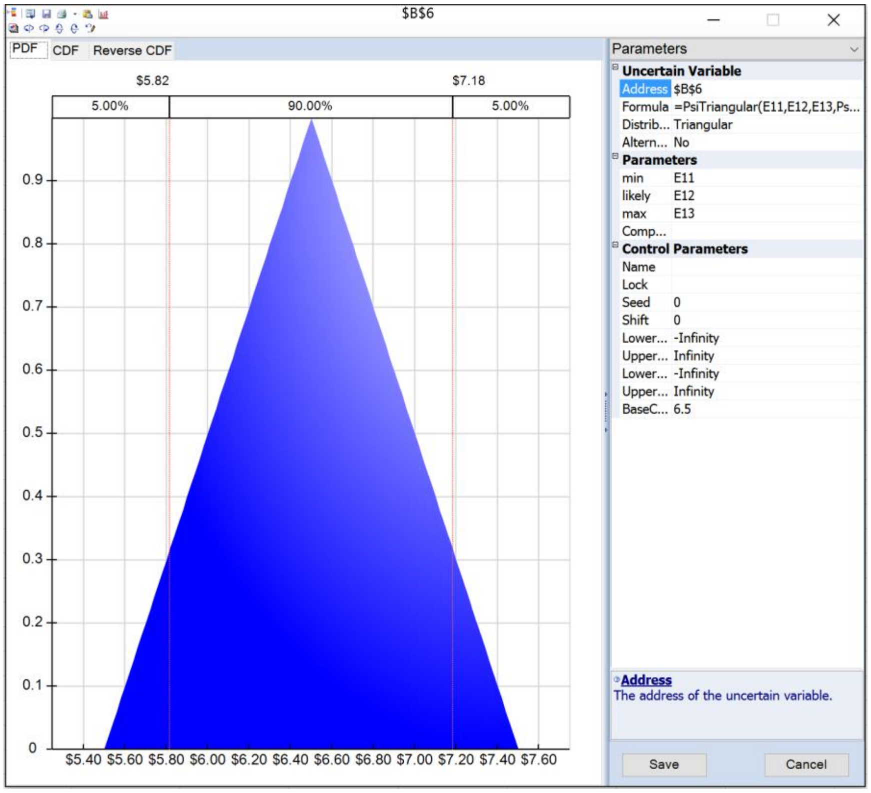
Analytic Solver Distributions

Click to select a distribution, or select the Distribution Wizard and we'll assist you with the best solution.

Distribution Wizard
Import
Convert
Common
Common
The most commonly used continuous distributions.
Advanced
Advanced
The more advanced continuous distributions.
Exotic
Exotic
The more exotic continuous distributions.
Discrete
Discrete
The discrete distributions.
Custom
Custom
The custom distributions.
Beta
Exponential
Gamma
LogNorm2
Normal
Triangular
Weibull
Close

PsiTriangular is a distribution bounded on both sides. It is rarely an accurate representation of a data set, but may be used when little or no data is available.

(Unlike a discrete distribution, a sample drawn from a *continuous* distribution can be any numeric value, such as 5.8 or 6.01, in a range.) Analytic Solver displays the Uncertain Variable dialog with a chart of the triangular distribution, initially from 0 to 3, peaking at 1.



Uncertain Variable
Address $B$6
Formula =PsiTriangular(E11,E12,E13,Ps...
Distrib... Triangular
Altern... No

Parameters
min E11
likely E12
max E13
Comp...

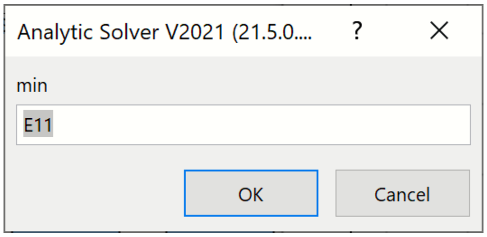
Control Parameters
Name
Lock
Seed 0
Shift 0
Lower... -Infinity
Upper... Infinity
Lower... -Infinity
Upper... Infinity
BaseC... 6.5

Address
The address of the uncertain variable.

To see more of the features of this dialog, try clicking on the Parameters drop down at the top of the right panel. See Frontline Solvers Reference Guide for more information.

The minimum, most likely, and maximum values for Unit Cost are in cells E11, E12 and E13. If we enter these *cell references* – not just the numbers 5.50, 6.50 and 7.50 – into our PsiTriangular() function call, we'll have a more flexible model. To start, we simply click in the Value field for the **min** parameter.

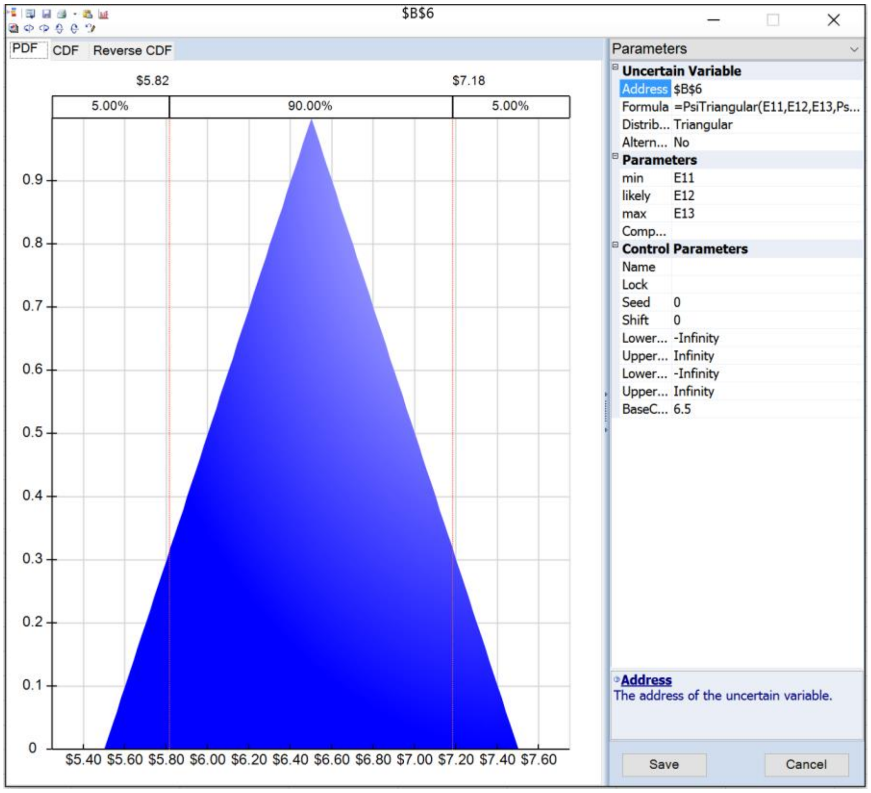
Now we can either type in a number or cell reference, or we can click the button at the right edge of the field: When we do this, a small dialog appears. Type “E11”, then click OK.



min: E11

If using Analytic Solver Cloud, simply type the cell address into the **min** field.

Now E11 appears as the **min (a)** parameter. Since at this moment, the **min (a)** value of 5.50 is greater than the **max (b)** value of 3, the error “The distribution is invalid with the current values of the parameters” is displayed instead of the chart. This is a hint that the parameters are inconsistent. (If using Analytic Solver Cloud, click OK to clear the error and continue editing the Parameter fields.) The chart reappears once we enter references to cell E12 for **likely (c)** and E13 for **max (b)**. When we click the **Save** icon, **=PsiTriangular(E11,E12,E13)** is inserted into cell B6.



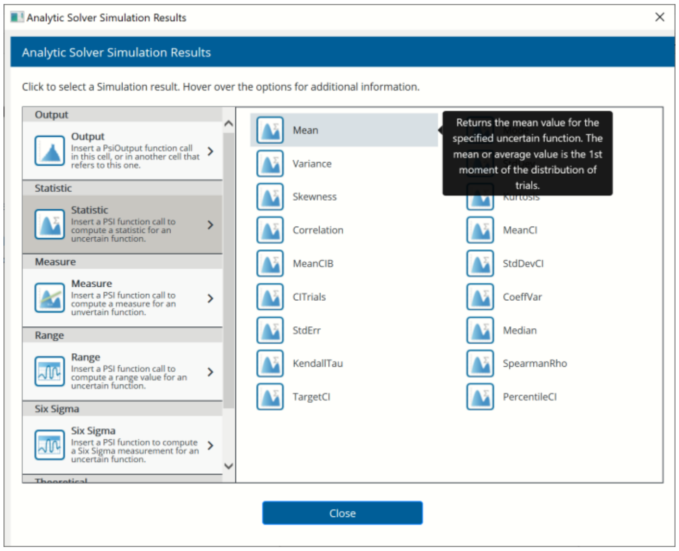
Uncertain Variable	
Address	$B$6
Formula	=PsiTriangular(E11,E12,E13,Ps...
Distrib...	Triangular
Altern...	No
Parameters	
min	E11
likely	E12
max	E13
Comp...	
Control Parameters	
Name	
Lock	
Seed	0
Shift	0
Lower...	-Infinity
Upper...	Infinity
Lower...	-Infinity
Upper...	Infinity
BaseC...	6.5

Address
The address of the uncertain variable.

We’ve now defined the **uncertain variables** in our model. Anything calculated from these uncertain variables is an **uncertain function**, but usually we’re interested only in specific results such as Net Profit at cell B11. When we run a simulation, B11 will effectively hold an **array of values**, each one calculated from different values sampled for B4, B5, B6 and B7.

What would we like to know about the array of values for Net Profit at cell B11? The simplest summary result is the average (or mean) Net Profit. Note that this will be the ***true average*** of Net Profit across 1,000 or more scenarios or trials – not a single calculation from *average* values of the *inputs*.

Click Results to open the Simulation Results dialog then select Statistics from the left and Mean from the right. Enter the output cell and the cell where the statistic is to be placed, then click Insert.



Click to select a Simulation result. Hover over the options for additional information.

Insert a PsiOutput function call in this cell, or in another cell that refers to this one.

Insert a PSI function call to compute a statistic for an uncertain function.

Insert a PSI function call to compute a measure for an unvertain function.

Insert a PSI function call to compute a range value for an uncertain function.

Insert a PSI function to compute a Six Sigma measurement for an uncertain function.

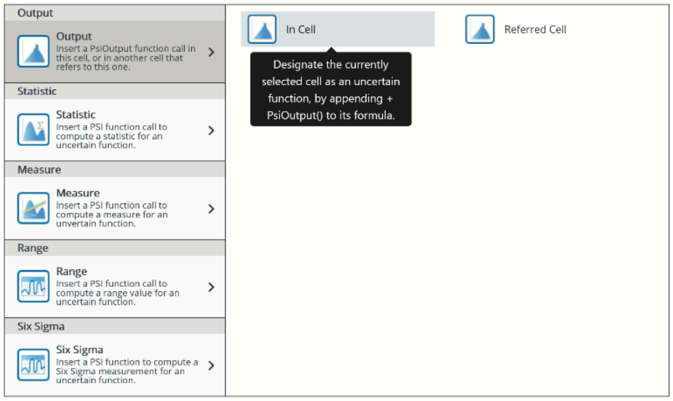
Returns the mean value for the specified uncertain function. The mean or average value is the 1st moment of the distribution of trials.

Selecting Uncertain Functions

When we define a summary statistic, such as =PsiMean(B11), we've implicitly designated cell B11 as an **uncertain function**. Analytic Solver will keep track of the full range of trial values for B11 during a simulation, and will display frequency and sensitivity charts, statistics and percentiles for it on demand. As noted above, in principle anything calculated from the uncertain variables is an "uncertain function" – but to save time and memory, Analytic Solver keeps track of trial values only for the formula cells that we ***designate*** as uncertain functions.

What if we want to designate cell B11 as an uncertain function without calculating any summary statistic for it on the worksheet?

With cell B11 selected, select Results to open the Result Dialog, then click OK to accept Output for Category and In Cell for Function.



Insert a PsiOutput function call in this cell, or in another cell that refers to this one.

Insert a PSI function call to compute a statistic for an uncertain function.

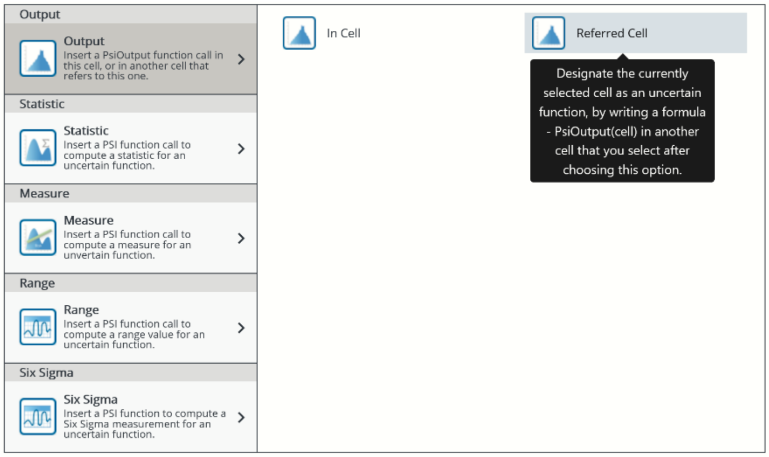
Insert a PSI function call to compute a measure for an unvertain function.

Insert a PSI function call to compute a range value for an uncertain function.

Insert a PSI function to compute a Six Sigma measurement for an uncertain function.

Designate the currently selected cell as an uncertain function, by appending + PsiOutput() to its formula.

This will modify cell B11’s formula =B4*(B5-B6)-B7 to read **=B4*(B5-B6)-B7 + PsiOutput()**. If we don’t want to modify this formula, we can instead choose **Results – Output – Referred Cell**.



Output

Output

Insert a PsiOutput function call in this cell, or in another cell that refers to this one.

Statistic

Statistic

Insert a PSI function call to compute a statistic for an uncertain function.

Measure

Measure

Insert a PSI function call to compute a measure for an unvertain function.

Range

Range

Insert a PSI function call to compute a range value for an uncertain function.

Six Sigma

Six Sigma

Insert a PSI function to compute a Six Sigma measurement for an uncertain function.

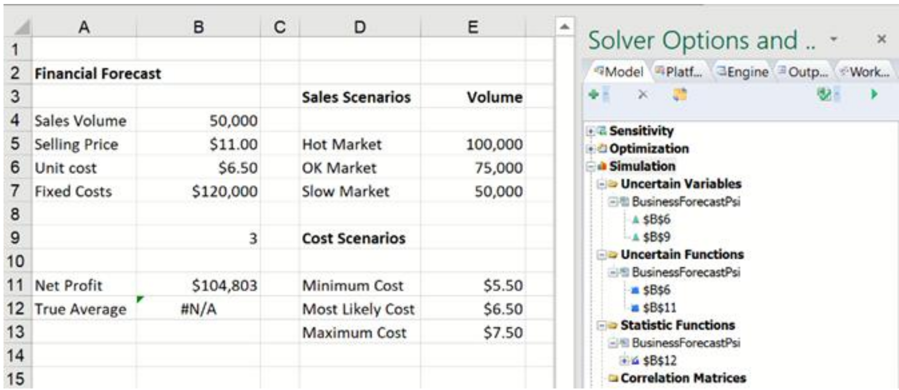
In Cell

Referred Cell

Designate the currently selected cell as an uncertain function, by writing a formula - PsiOutput(cell) in another cell that you select after choosing this option.

If we click in cell B12, this cell will contain **=PsiOutput(B11)**. Any **one** of these three actions – defining a summary statistic such as PsiMean() on B11, add PsiOutput() to the formula in cell B11, or placing =PsiOutput(B11) in another cell – is sufficient to designate cell B11 as an uncertain function.

Our model now looks like the one pictured below. Notice that the Task Pane **Model tab** shows our simulation model in outline form, and cell B12, where we placed the **PsiMean()** function, displays as #N/A – because a simulation has not yet been executed.

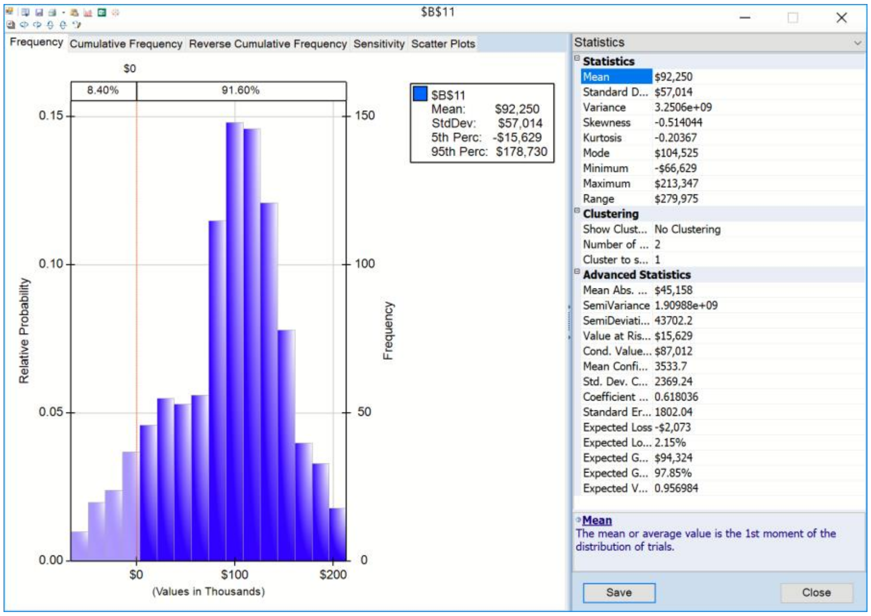


	A	B	C	D	E
1					
2	Financial Forecast				
3				Sales Scenarios	Volume
4	Sales Volume	50,000			
5	Selling Price	$11.00		Hot Market	100,000
6	Unit cost	$6.50		OK Market	75,000
7	Fixed Costs	$120,000		Slow Market	50,000
8					
9		3		Cost Scenarios	
10					
11	Net Profit	$104,803		Minimum Cost	$5.50
12	True Average	#N/A		Most Likely Cost	$6.50
13				Maximum Cost	$7.50
14					
15					

Running a Simulation

Click the green arrow on the Model tab to start a simulation. With an old-fashioned simulation software package, you’d press a button to start a simulation, then perhaps get a cup of coffee. Because simulations ran slowly, software packages were designed for “batch” operation: You’d spend time getting everything set up just right, run a simulation and wait (sometimes quite a while), then spend time analyzing the results. But with Analytic Solver, you'll find that in the blink of an eye, your first Monte Carlo simulation is complete!

Since there is one uncertain function, and this is the first time a simulation has been performed, the following dialog opens. Subsequent simulations will not produce this report. However, it is possible to reopen this frequency chart simply by double clicking cell B11. (See below for more information on this chart.)



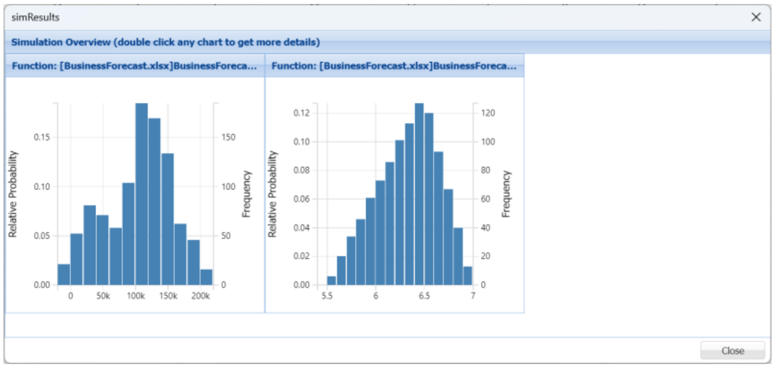
Statistics

Mean	$92,250
Standard D...	$57,014
Variance	3.2506e+09
Skewness	-0.514044
Kurtosis	-0.20367
Mode	$104,525
Minimum	-$66,629
Maximum	$213,347
Range	$279,975
Clustering	
Show Clust...	No Clustering
Number of ...	2
Cluster to s...	1
Advanced Statistics	
Mean Abs. ...	$45,158
SemiVariance	1.90988e+09
SemiDeviati...	43702.2
Value at Ris...	$15,629
Cond. Value...	$87,012
Mean Confi...	3533.7
Std. Dev. C...	2369.24
Coefficient ...	0.618036
Standard Er...	1802.04
Expected Loss	-$2,073
Expected Lo...	2.15%
Expected G...	$94,324
Expected G...	97.85%
Expected V...	0.956984

Mean

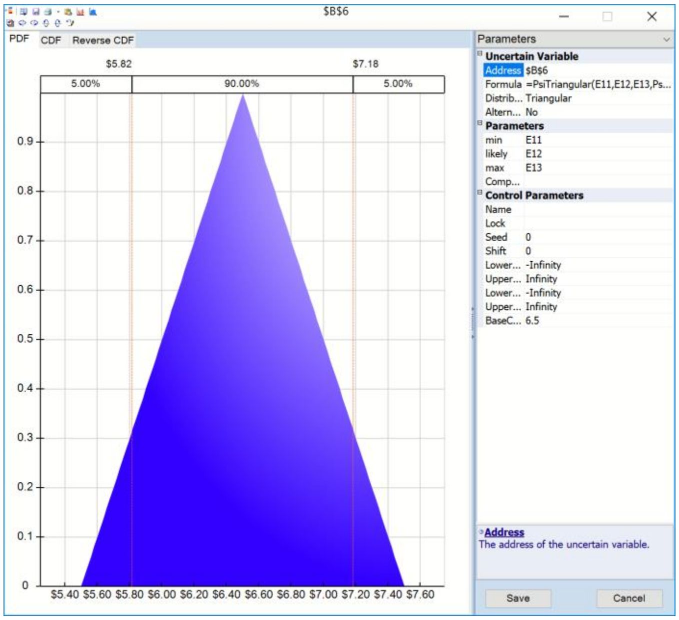
The mean or average value is the 1st moment of the distribution of trials.

If your simulation model contains more than one output function, then a chart containing frequency graphs of up to 9 output functions and uncertain variables will appear. For example, if the uncertain variable in cell B6 was also selected to be an output cell (B6 = PsiTriangular(0, 1, 3) + PsiOutput()) and a simulation was performed, the following graph would appear at the end of the simulation.



Simulation Overview (double click any chart to get more details)

Double clicking the 2nd chart will open the B6 uncertain variable dialog, as shown below. To open the uncertain function dialog, click the Show Output icon on the top left of the title bar.



Uncertain Variable

Address	$B$6
Formula	=PsiTriangular(E11,E12,E13,Ps...
Distrib...	Triangular
Altern...	No

Parameters

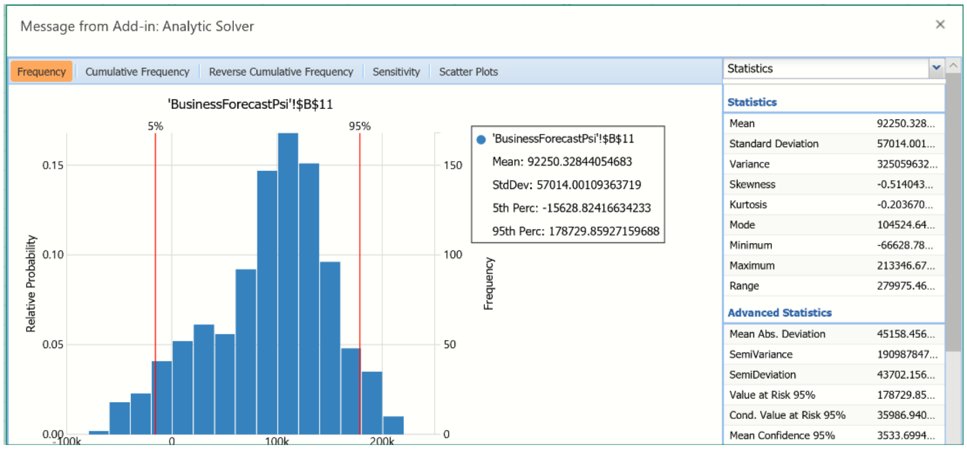
min	E11
likely	E12
max	E13
Comp...	

Control Parameters

Name	
Lock	
Seed	0
Shift	0
Lower...	-Infinity
Upper...	Infinity
Lower...	-Infinity
Upper...	Infinity
BaseC...	6.5

Address
The address of the uncertain variable.

Double clicking "BusinessForecastPsi!B6" in Analytic Solver Cloud will open the uncertain function dialog, as shown below.



'BusinessForecastPsi'!$B$11
Mean: 92250.32844054683
StdDev: 57014.00109363719
5th Perc: -15628.82416634233
95th Perc: 178729.85927159688

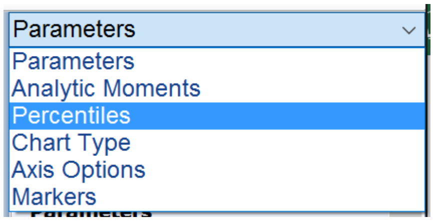
Statistics

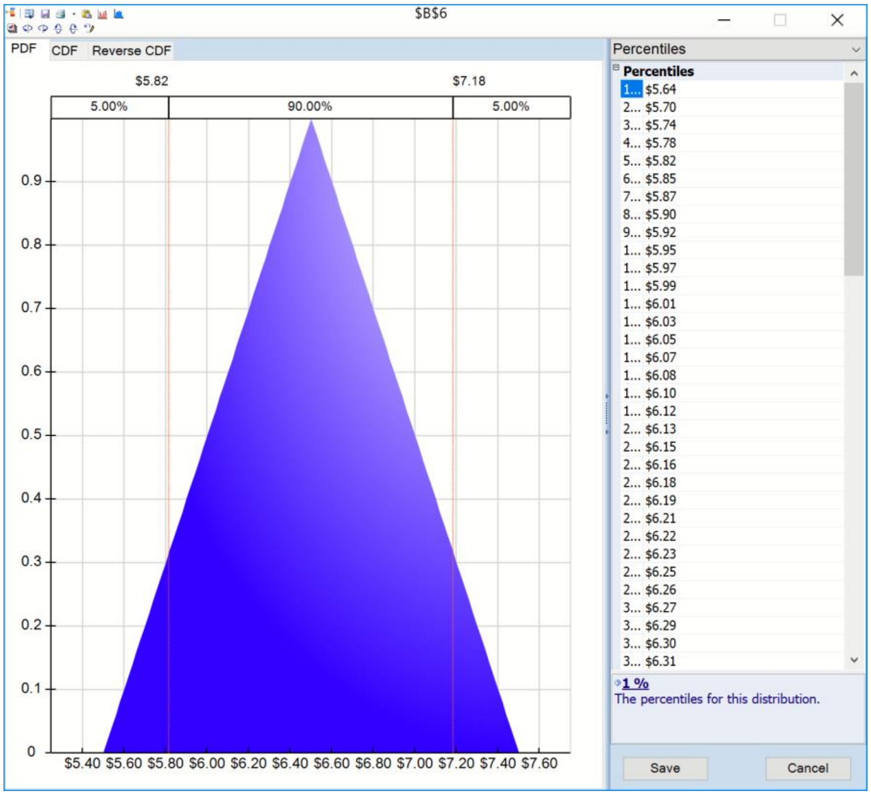
Mean	92250.328...
Standard Deviation	57014.001...
Variance	325059632...
Skewness	-0.514043...
Kurtosis	-0.203670...
Mode	104524.64...
Minimum	-66628.78...
Maximum	213346.67...
Range	279975.46...

Advanced Statistics

Mean Abs. Deviation	45158.456...
SemiVariance	190987847...
SemiDeviation	43702.156...
Value at Risk 95%	178729.85...
Cond. Value at Risk 95%	35986.940...
Mean Confidence 95%	3533.6994...

If you select Percentiles from the drop down menu on the top right, you'll notice that the the vertical red lines displayed on the chart denote the 5% and the 95% percentiles values of 5.82 and 7.18, respectively.



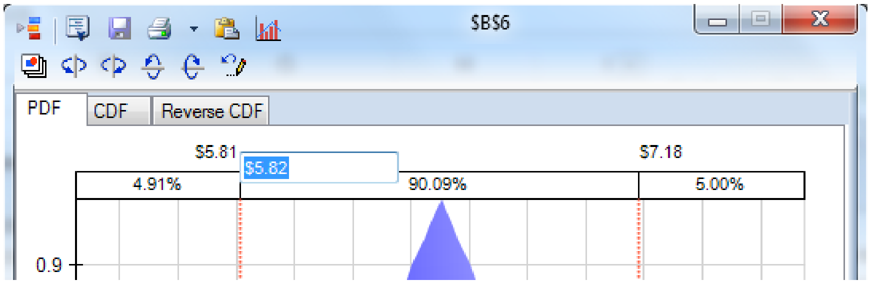


Percentiles

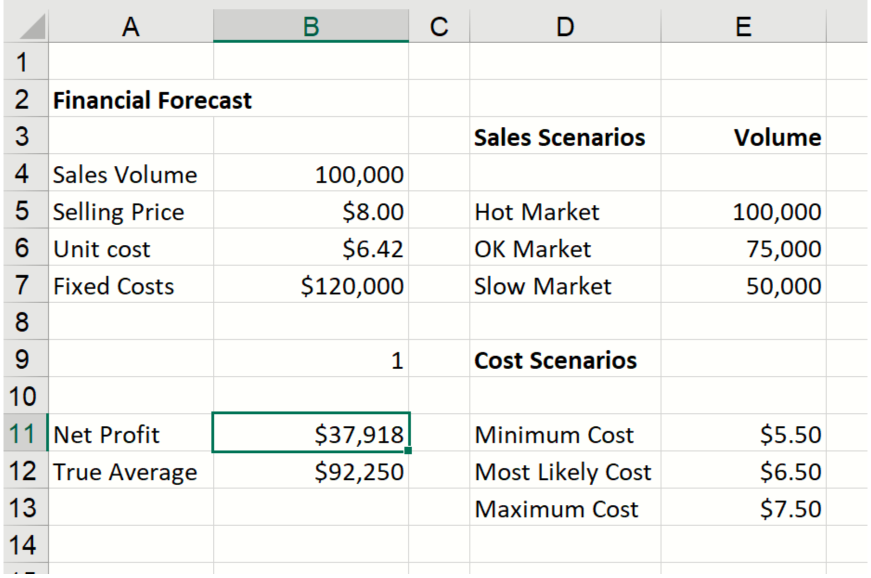
1... $5.64
2... $5.70
3... $5.74
4... $5.78
5... $5.82
6... $5.85
7... $5.87
8... $5.90
9... $5.92
1... $5.95
1... $5.97
1... $5.99
1... $6.01
1... $6.03
1... $6.05
1... $6.07
1... $6.08
1... $6.10
1... $6.12
2... $6.13
2... $6.15
2... $6.16
2... $6.18
2... $6.19
2... $6.21
2... $6.22
2... $6.23
2... $6.25
2... $6.26
3... $6.27
3... $6.29
3... $6.30
3... $6.31

1 %
The percentiles for this distribution.

Each of the 5 values at the top of the chart can be edited in Analytic Solver Desktop. For example, the red lines can be moved to the left or right until the desired percentile is achieved, or you can directly edit the percentile or the percentile's value by clicking the value to be changed. Note: This functionality is not supported in Analytic Solver Cloud. The red lines denoting the 5th and 95th percentiles are both stationary in these products.

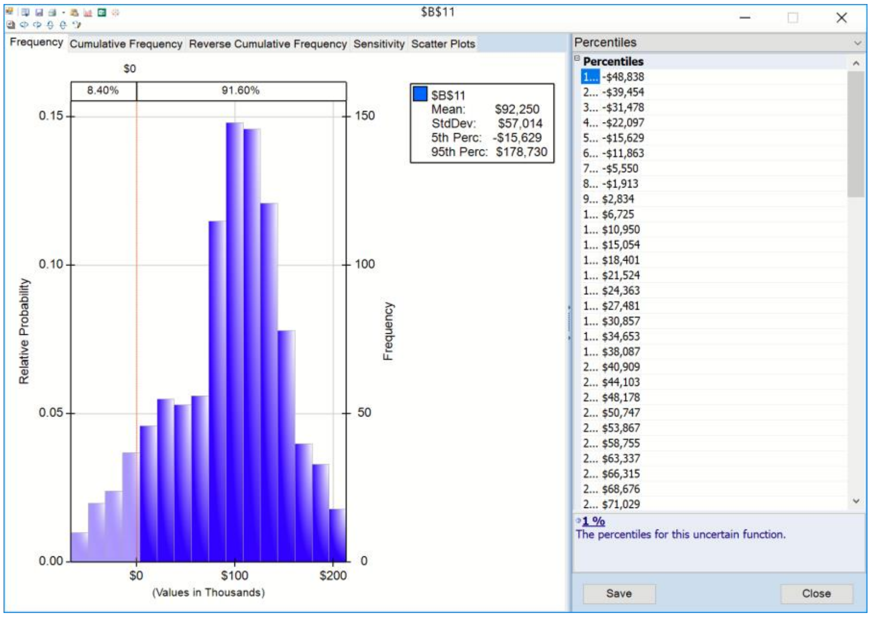


If you close the uncertain variable dialog for cell B6, (by clicking the X in the top right corner) you will see that cell B12 displays the ***true average*** for Net Profit across these trials – as shown below.



	A	B	C	D	E
1					
2	**Financial Forecast**				
3				**Sales Scenarios**	**Volume**
4	Sales Volume	100,000			
5	Selling Price	$8.00		Hot Market	100,000
6	Unit cost	$6.42		OK Market	75,000
7	Fixed Costs	$120,000		Slow Market	50,000
8					
9		1		**Cost Scenarios**	
10					
11	Net Profit	$37,918		Minimum Cost	$5.50
12	True Average	$92,250		Most Likely Cost	$6.50
13				Maximum Cost	$7.50
14					

The result of "shaking the ladder" is striking: Our ***true average*** Net Profit for these 1,000 trials is $92,250 – quite a bit less than the "Flawed Average" Model figure of $117,750! And we also see that we can **lose money**. Notice that the first eight percentiles are negative in the screenshot below.



$B$11
Mean: $92,250
StdDev: $57,014
5th Perc: -$15,629
95th Perc: $178,730

Percentiles

1...	-$48,838
2...	-$39,454
3...	-$31,478
4...	-$22,097
5...	-$15,629
6...	-$11,863
7...	-$5,550
8...	-$1,913
9...	$2,834
1...	$6,725
1...	$10,950
1...	$15,054
1...	$18,401
1...	$21,524
1...	$24,363
1...	$27,481
1...	$30,857
1...	$34,653
1...	$38,087
2...	$40,909
2...	$44,103
2...	$48,178
2...	$50,747
2...	$53,867
2...	$58,755
2...	$63,337
2...	$66,315
2...	$68,676
2...	$71,029

1 %
The percentiles for this uncertain function.

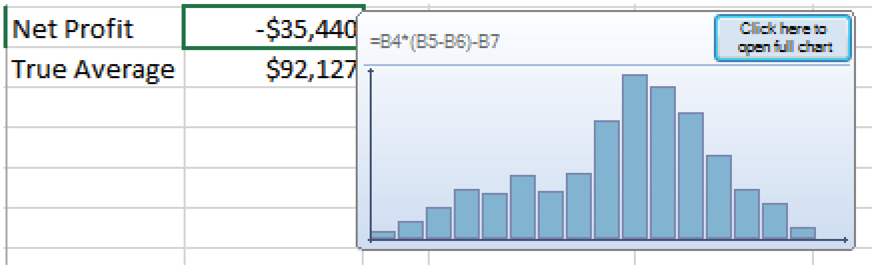
In Analytic Solver Desktop you can try pressing F9 (the Excel recalculate key) on this model: Each time you do, another 1,000 Monte Carlo trials are run, and a slightly different true average Net Profit figure will be displayed – but normally much less than $117,750.

Interactive Simulation in Desktop Solver

We've just seen that the mean or 'True Average' Net Profit, over 1,000 different simulated outcomes, is less than we expected from our naïve "Flawed Average" model. We've also seen that in some outcomes, our Net Profit is actually a loss. A quick look at the dropdown galleries for **Results – Statistic**, **Measure** and

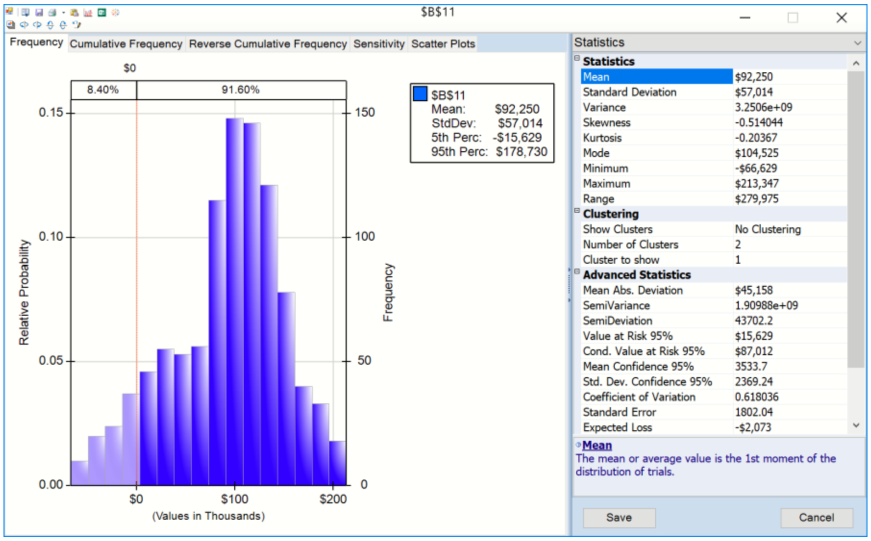
Range on the Ribbon suggests that we can easily compute and view many other statistics about Net Profit. But we'd really like to see the *full range of outcomes* in this model. This is *very* easy to do in *Analytic Solver Desktop* using Interactive Simulation. Note: Interactive Simulation is not supported in Analytic Solver Cloud.

With Interactive Simulation turned on, simply **move the mouse pointer to B11** and wait about 1 second. A ***miniature***, ***live*** frequency distribution chart of the simulation trial values for cell B11 appears automatically:



Net Profit	-$35,440
True Average	$92,127

To see and do more, just **double-click on B11**, the output cell calculating Net Profit, to display the same frequency chart that was automatically opened after our first simulation run.



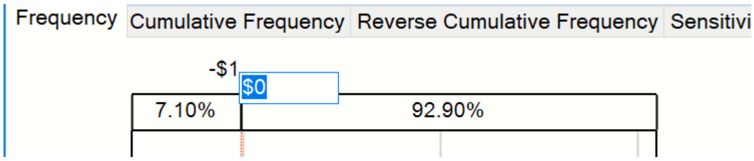
$B$11
Mean: $92,250
StdDev: $57,014
5th Perc: -$15,629
95th Perc: $178,730

Statistics	
Mean	$92,250
Standard Deviation	$57,014
Variance	3.2506e+09
Skewness	-0.514044
Kurtosis	-0.20367
Mode	$104,525
Minimum	-$66,629
Maximum	$213,347
Range	$279,975
Clustering	
Show Clusters	No Clustering
Number of Clusters	2
Cluster to show	1
Advanced Statistics	
Mean Abs. Deviation	$45,158
SemiVariance	1.90988e+09
SemiDeviation	43702.2
Value at Risk 95%	$15,629
Cond. Value at Risk 95%	$87,012
Mean Confidence 95%	3533.7
Std. Dev. Confidence 95%	2369.24
Coefficient of Variation	0.618036
Standard Error	1802.04
Expected Loss	-$2,073

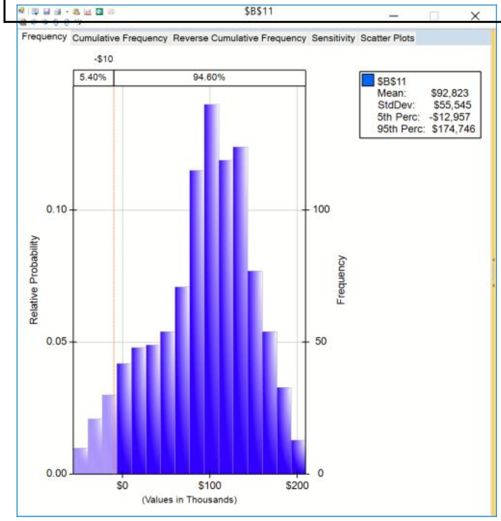
Mean
The mean or average value is the 1st moment of the distribution of trials.

Viewing the Full Range of Profit Outcomes

We see immediately that in some outcomes, we can lose of lot of money – over $60,000! We also see that we make a profit in *most* outcomes – but how many exactly? Analytic Solver automatically sets a **Lower-Cutoff** of "0" in the Frequency chart by displaying a vertical red bar. The probabilities are automatically shown above the chart. In this case there is an 8.4% chance of a loss and a 91.60% chance of a profit. In Analytic Solver Desktop, the lower cutoff can be changed by clicking the $0 at the top of the chart. The percentile values can be edited in the same way.



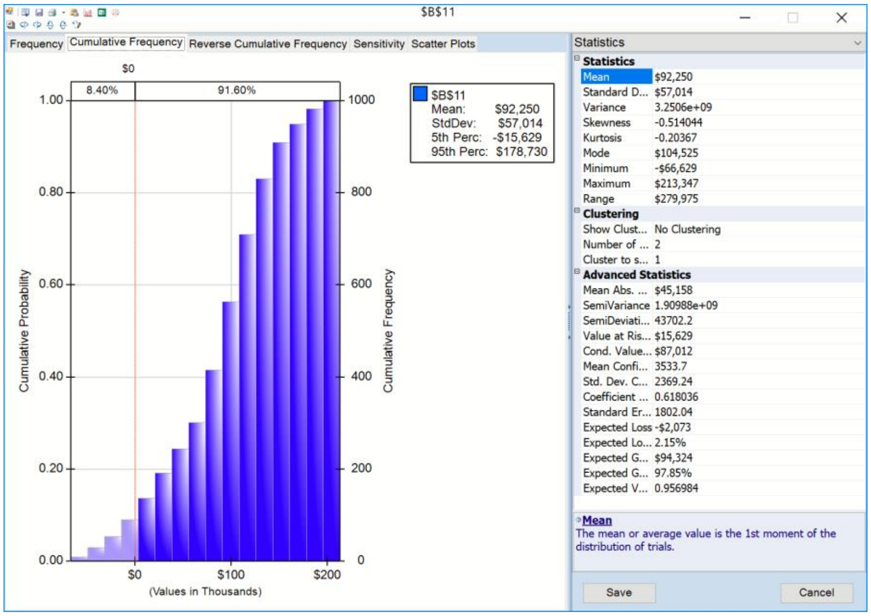
Note: If you change the lower cutoff from 0 to -10, you'll notice the red vertical line denoting the lower cutoff of $0 has now moved to -$10 and the percentile values have also changed accordingly. Percentile values can be edited in the same way or by simply moving the red vertical line to the left or right.



$B$11
Mean: $92,823
StdDev: $55,545
5th Perc: -$12,957
95th Perc: $174,746

Looking at the right panel you can see the Statistics view, which includes summary statistics for the full range of Net Profit outcomes. We can see that the worst case outcome (Minimum) of this simulation was -$66,629, and the best case outcome (Maximum) was +$213,347. Value at Risk 95% shows that we have a 5% chance of making $15,629 or less. The Conditional Value at Risk 95% value indicates that the average profit we will see (up to the 95% percentile) is $87,012.

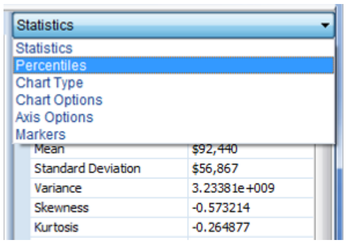
Another view of the full range of outcomes is shown on the Cumulative Frequency tab.



Statistics	
Mean	$92,250
Standard D...	$57,014
Variance	3.2506e+09
Skewness	-0.514044
Kurtosis	-0.20367
Mode	$104,525
Minimum	-$66,629
Maximum	$213,347
Range	$279,975
Clustering	
Show Clust...	No Clustering
Number of ...	2
Cluster to s...	1
Advanced Statistics	
Mean Abs. ...	$45,158
SemiVariance	1.90988e+09
SemiDeviati...	43702.2
Value at Ris...	$15,629
Cond. Value...	$87,012
Mean Confi...	3533.7
Std. Dev. C...	2369.24
Coefficient ...	0.618036
Standard Er...	1802.04
Expected Loss	-$2,073
Expected Lo...	2.15%
Expected G...	$94,324
Expected G...	97.85%
Expected V...	0.956984

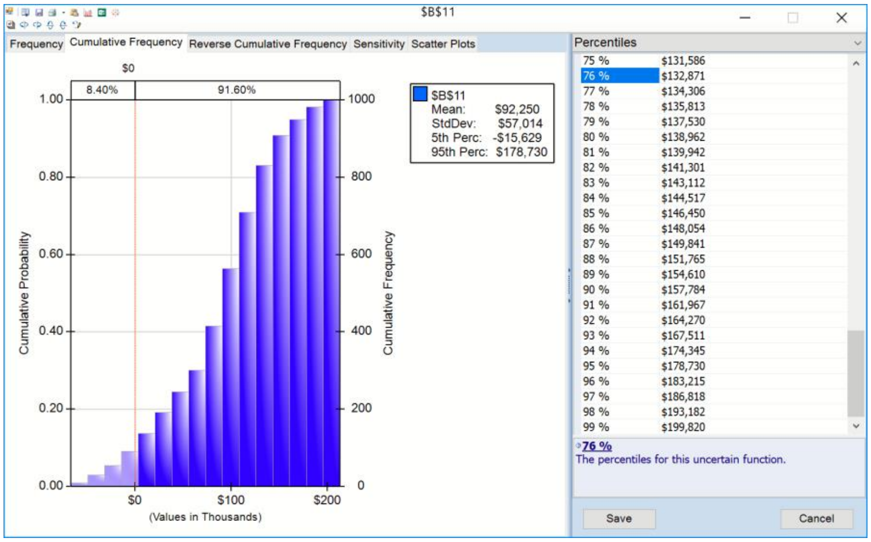
Mean
The mean or average value is the 1st moment of the distribution of trials.

Click on the Statistics drop down at the top of the right panel and choose Percentiles from the list.



Mean	$92,440
Standard Deviation	$56,867
Variance	3.23381e+009
Skewness	-0.573214
Kurtosis	-0.264877

The Percentiles tab shows the same information as the Cumulative Frequency tab, but in numeric form. Below, we've shown the 75th through 99th percentiles:



Percentiles	
75 %	$131,586
76 %	$132,871
77 %	$134,306
78 %	$135,813
79 %	$137,530
80 %	$138,962
81 %	$139,942
82 %	$141,301
83 %	$143,112
84 %	$144,517
85 %	$146,450
86 %	$148,054
87 %	$149,841
88 %	$151,765
89 %	$154,610
90 %	$157,784
91 %	$161,967
92 %	$164,270
93 %	$167,511
94 %	$174,345
95 %	$178,730
96 %	$183,215
97 %	$186,818
98 %	$193,182
99 %	$199,820

76 %
The percentiles for this uncertain function.

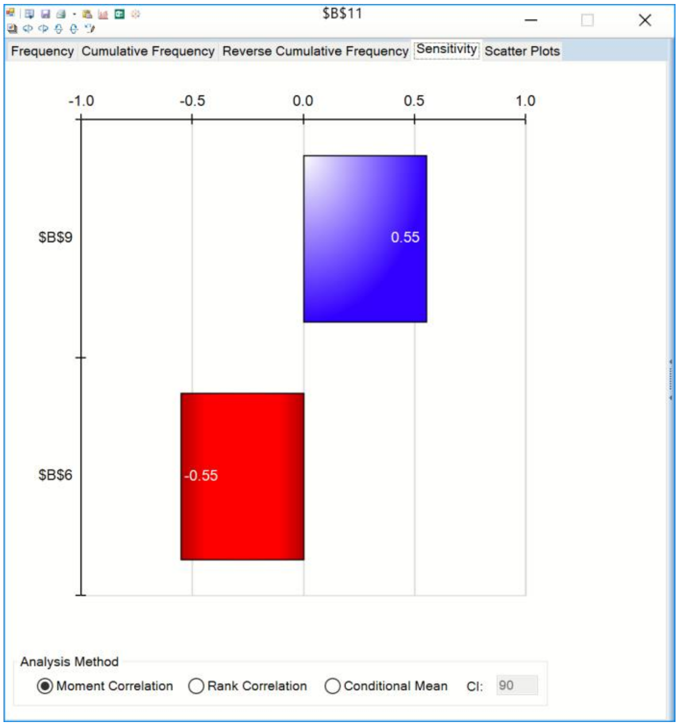
Note that you can close and open the right hand panel simply by clicking on the two arrows on the vertical bar between the chart and the right hand panel.

Analyzing Factors Influencing Net Profit

Other tabs on the Uncertain Function dialog can help us understand how the uncertain variables in our model influence our Net Profit.

Sensitivity Tab

The **Sensitivity tab** displays a "Tornado chart" that shows you how much Net Profit changes with a change in the uncertain variables – the Triangular distribution for Unit Cost at cell B6, and the integer uniform distribution at cell B9. In this model there are only two uncertain variables, but in a large model with many such variables, it's usually not obvious which ones have the greatest impact on outcomes such as Net Profit. A Tornado chart highlights the key variables: B6 has a negative correlation with Net Profit, whereas B9 has a positive correlation.



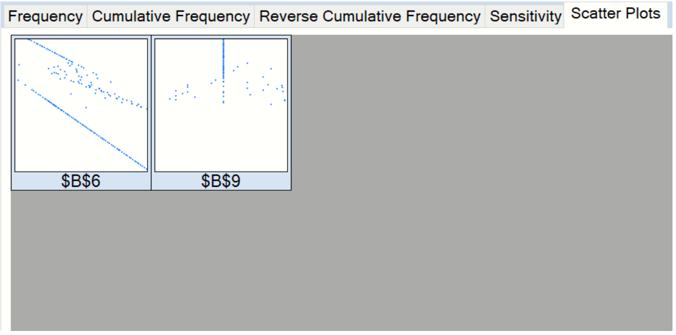
The fact that Net Profit goes down as our Unit Cost at B6 goes up is expected. But why does Net Profit go up as cell B9 goes up? Note that B9 = 1 corresponds to the Hot market scenario, 2 corresponds to the OK market, and 3 corresponds to the Slow market scenario. The fact that B9 is positively correlated with Net Profit is telling us that we make *higher* profits when the market is *slow*, not when it's hot. Our typical Selling Price is lower when the market is hot, and the increased Sales Volume doesn't make up for this.

Scatter Plots Tab

The **Scatter Plots tab**, shown on the next page, gives us a different view of the relationship of Net Profit to our uncertain variables: Unit Cost at cell B6, and the integer uniform distribution at cell B9.

Notice that the scatter plot against Unit Cost has three downward-sloping lines: Each line corresponds to a different market scenario (Hot, OK or Slow), and shows that as Unit Cost rises, our Net Profit drops – dropping fastest in the Hot Market scenario, because our profit margin is so narrow.

The scatter plot against the integer uniform distribution shows three high-low ranges for Net Profit, again corresponding to a different market scenario – Hot, OK, or Slow. We see again that the range of Net Profit is lowest in the Hot Market scenario.

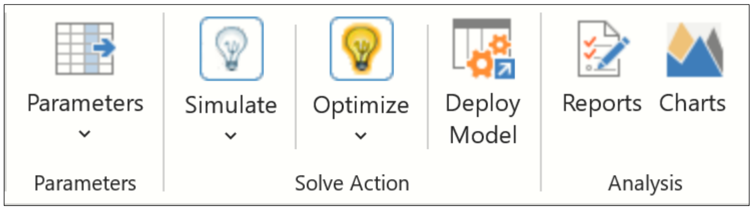


Interactive Simulation with Charts and Graphs

With an old-fashioned simulation software package, you'd press a button to start a simulation, then perhaps get a cup of coffee. Because simulations ran slowly, software packages were designed for "batch" operation: You'd spend time getting everything set up just right, run a simulation and wait (sometimes quite a while), then spend time analyzing the results. But with Analytic Solver, simulations run so fast that *fully Interactive Simulation* is practical.

Now 1,000 Monte Carlo simulation trials (the default number) will be executed *each time we change the spreadsheet*. Since there is one uncertain function, and this is the first time a simulation has been performed, the following dialog opens. Subsequent simulations will not produce this report. However, it is possible to reopen this frequency chart simply by double clicking cell B11. (See below for more information on this chart.)

To turn on **Interactive Simulation**, simply click the **light bulb** on the Ribbon. It will "light up," as shown below.

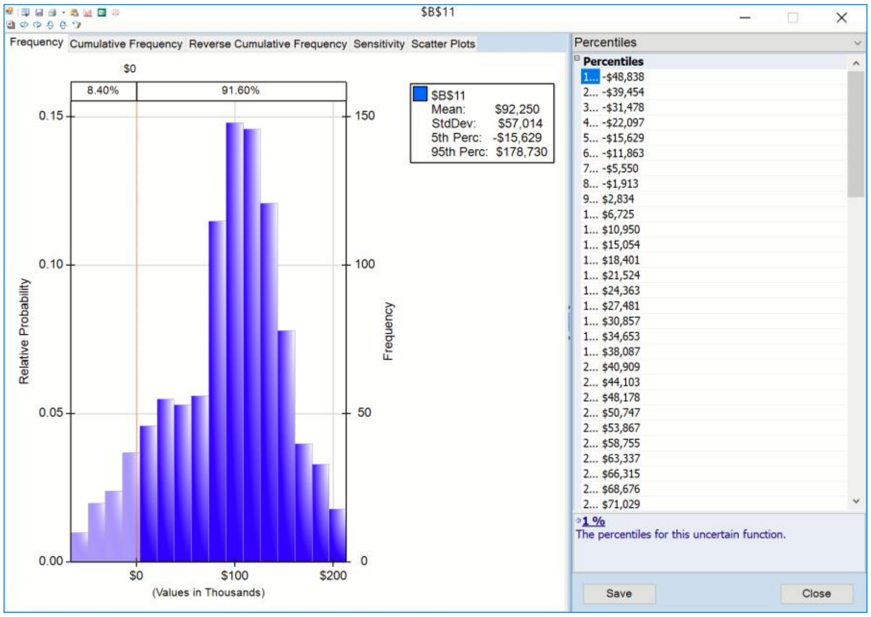


Interactive Simulation makes Analytic Solver Simulation fundamentally different from other Monte Carlo simulation tools for Excel. The kinds of charts we've just seen can be produced by other tools, but only *at the end* of a "simulation run." In contrast, Analytic Solver makes these charts ***live as you play what-if*** with your model.

After seeing this model, your production manager might think of a way to reduce the maximum Unit Cost to $7.00 instead of $7.50. What would be the impact of this change on Net Profit, over the *full range outcomes*? With Analytic Solver Comprehensive or Analytic Solver Simulation, this is as easy as any other 'what-if' question in Excel:

Click the Frequency tab to re-display the frequency chart of outcomes for cell B11. Then simply change the number in cell E13 from 7.50 to 7. ***Immediately***, a thousand Monte Carlo trials are performed, and the chart is updated. The effect is striking: We have a 97.7% chance of making a profit, and – checking the Percentiles tab – we see that instead of a 1% chance of losing about $48,000, we have a 1% chance of losing just $5,285!

But Analytic Solver Simulation is even faster than this: Click the Options button on the Ribbon, and change the number of Monte Carlo trials from 1,000 to 100,000 (or more). In a second or less on most modern PCs, ***one hundred thousand*** Monte Carlo trials are performed, and the chart is updated!



$B$11
Mean: $92,250
StdDev: $57,014
5th Perc: -$15,629
95th Perc: $178,730

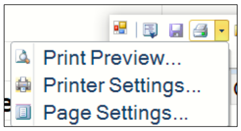
Percentiles
1... -$48,838
2... -$39,454
3... -$31,478
4... -$22,097
5... -$15,629
6... -$11,863
7... -$5,550
8... -$1,913
9... $2,834
1... $6,725
1... $10,950
1... $15,054
1... $18,401
1... $21,524
1... $24,363
1... $27,481
1... $30,857
1... $34,653
1... $38,087
2... $40,909
2... $44,103
2... $48,178
2... $50,747
2... $53,867
2... $58,755
2... $63,337
2... $66,315
2... $68,676
2... $71,029

1 %
The percentiles for this uncertain function.

The Flawed Average model presented a limited and misleading picture of this business situation. In contrast, the Analytic Solver model has illuminated the situation considerably. We can see what can go *right*, and what can go *wrong*. We can make an informed decision about whether the reward is worth the risk. And – most important – we can interactively explore ways to *improve* the reward and *reduce* the risk. This is risk analysis at work.

Charts and Graphs for Presentations

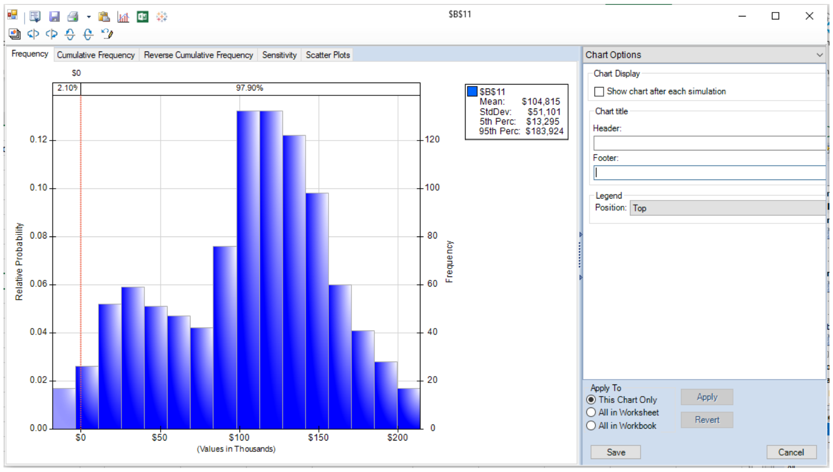
Often, you may be called up to present your results to others. One great way to do this is in *Excel itself, live!* But at times you may need to print a chart, or copy it into a Word document or PowerPoint presentation. This is *very* easy to do in Analytic Solver Desktop, with the toolbar buttons in the title bar of the Uncertain Variable or Uncertain Function dialog. Note: This functionality is not supported in Analytic Solver Cloud.



Click the **Clipboard** icon to copy the currently displayed chart to the Windows Clipboard. You can then choose Edit Paste in Word, Excel, PowerPoint and many other applications to paste the chart image into your document. (Choosing Edit Paste in Excel inserts a static, non-updating chart image in the worksheet.)

Click the **Print** icon to immediately print the currently displayed chart on your default printer, or click the down arrow next to this icon to display the menu choices shown above: Print Preview, Printer Settings and Page Settings. You can choose a printer and set printer options, set page margins, and preview your output using these menu choices.

Analytic Solver also makes it easy to control the format of your charts. Below, we've re-opened the right hand panel and clicked on the drop down menu to choose **Chart Type**.



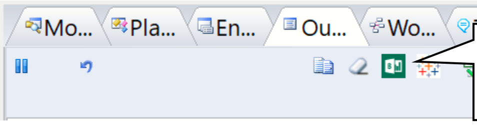
You can control the chart type, color, dimensionality and transparency, bin density, titles and legends, axis labels and number formats, horizontal axis scaling, and more. As you change chart options in the right pane, the chart is *immediately* updated so you can see the results (unlike some other simulation products for Excel). When you're satisfied with the chart format, you can save and apply it to just this chart, to all charts on this worksheet, or to all charts in the workbook when you click the Apply button at the bottom of the right pane.

Exporting Data to Microsoft's Power BI

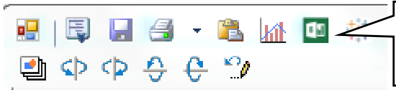
Microsoft's POWER BI, for use with Office 365, is a cloud-based service that works with Excel to help you visualize your data using various charts and reports. Analytic Solver Desktop includes the ability to export a fixed set of data from your simulation model, including all trial values, output function statistics, percentiles, etc., directly into a dataset in Power BI. This functionality is not present in Analytic Solver Cloud. See the chapter, Creating Power BI Custom Visuals, that appears later on in this guide for more information on using Power BI with these two products.

Once Solver has stopped with a final result message, click the icon on the Output tab *or* on the Uncertain Function dialog to upload the model results to the Power BI dashboard. If this is the first time that the icon has been clicked within the current Excel instance, you will be asked to log in to Power BI.

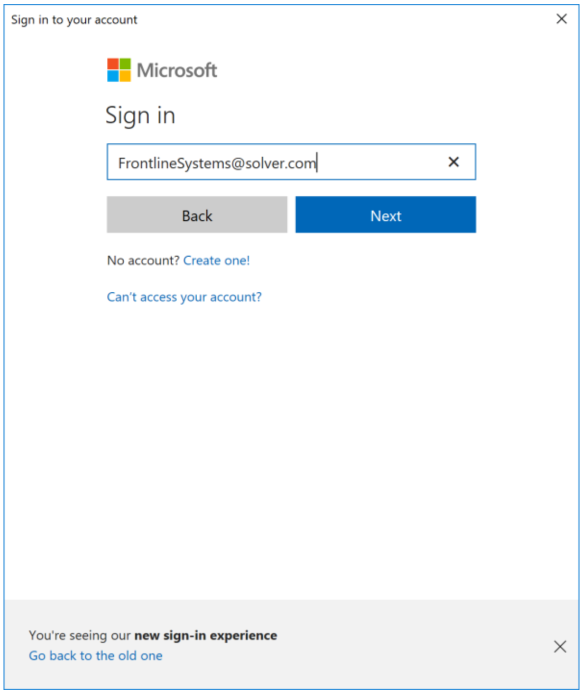
Note: If an error appears immediately after clicking the Power BI icon on the task pane or on the uncertain function dialog, you'll need to upgrade your current version of TLS (Transport Layer Security) to V1.2.



Power BI Icon as shown on the Output tab on the Solver Task Pane.



Power BI Icon as shown on the Uncertain Function dialog



Sign in to your account

Sign in

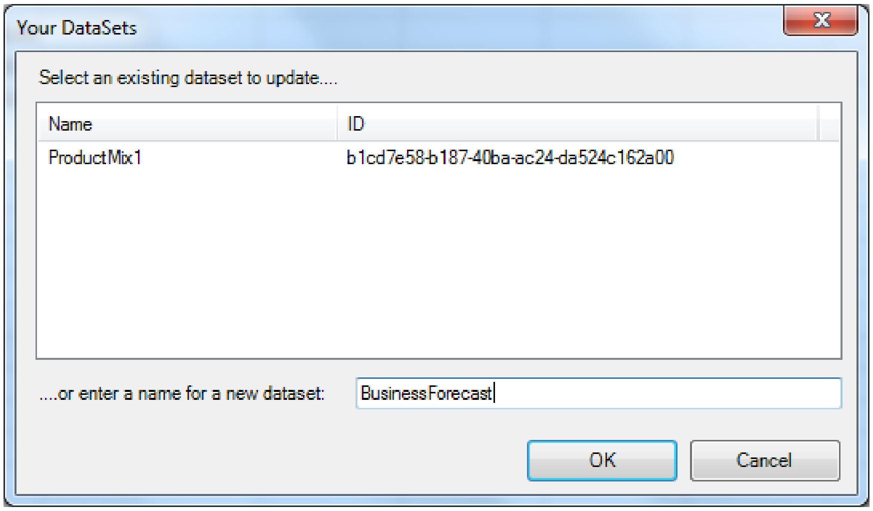
FrontlineSystems@solver.com

No account? Create one!

Can't access your account?

You're seeing our **new sign-in experience**
Go back to the old one

Once logged in, you will be asked to either update an existing dataset or create a new dataset.



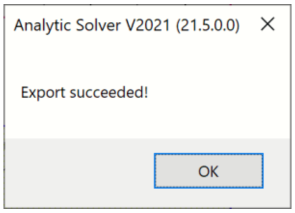
Your DataSets

Select an existing dataset to update....

Name	ID
ProductMix1	b1cd7e58-b187-40ba-ac24-da524c162a00

....or enter a name for a new dataset: BusinessForecast

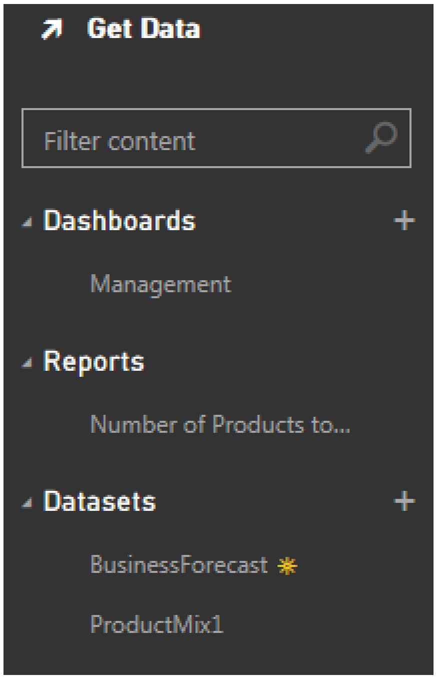
In the screenshot above, we have created a new dataset, BusinessForecast, which will contain the uncertain variable and uncertain function trial values, statistics and percentiles, etc from our optimization model. Once the upload is complete, the following message will appear.



Analytic Solver V2021 (21.5.0.0)

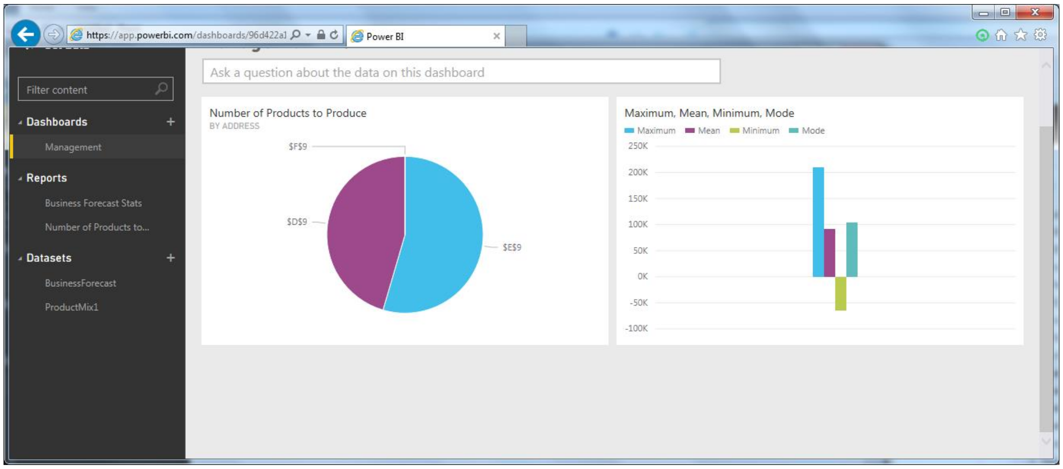
Export succeeded!

Logon to Power BI (http://powerbi.microsoft.com/). The newly created dataset will be listed under Datasets.

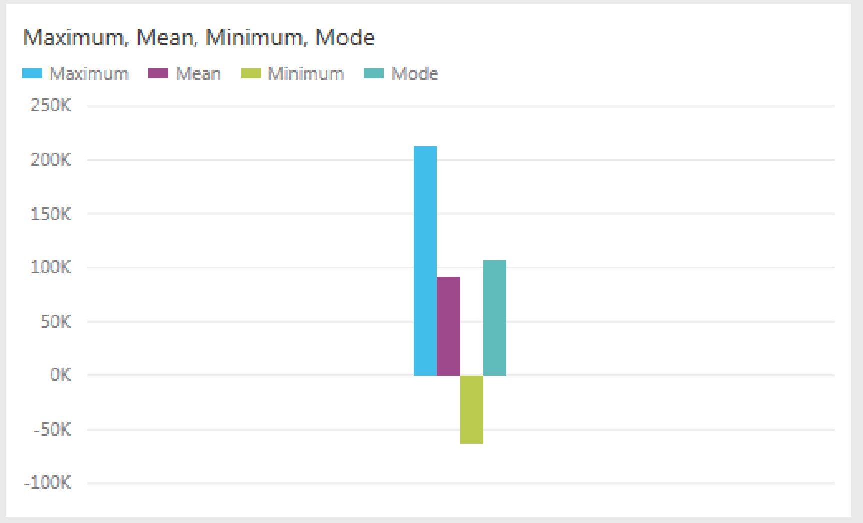


Select BusinessForecast, then determine the components to be included in the graph. In this example, we have created a bar chart by clicking (to the right of the graph) and selecting the Maximum, Mean, Minimum, and Mode Output Statistics.

Use the icon to pin this graph to the Dashboard.



Now, each time the model is solved, the results may be uploaded to your Power BI dashboard. Click back to Excel and change the Price of the Slow Market from $11 to $15, then run a 2nd simulation either by clicking the green arrow on the Model tab in the Solver Task Pane or by clicking the Simulate icon on the Analytic Solver ribbon. Once Solver stops, click the icon on either the Output tab *or* the Uncertain function dialog to upload the most recent results. Note: We are not asked to log in to the Power BI site a second time since we are using the same instance of Excel. However, we are asked if we would like to select an existing dataset to update. Select BusinessForecast and then click OK. Click back to Power BI in your browser and refresh, the chart will update automatically with the new final statistic values, as shown below.



Exporting Data to Tableau

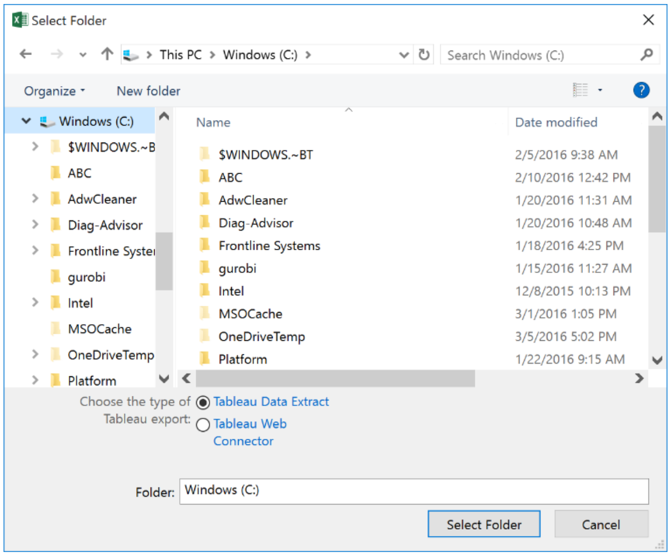
Tableau is a popular interactive software package that allows you to visually explore and analyze your data. Tableau can import data from a wide range of sources, including Excel workbooks, and it is often used in conjunction with Excel. Because Tableau is designed to import data in table form, it hasn't been easy to import the *results* of a simulation model (such as final values of the uncertain functions, uncertain variables, percentiles and statistic values) into Tableau, unless those model elements occur in table form by themselves in your spreadsheet (which usually isn't the case).

Analytic Solver Desktop simplifies this process considerably. With a single click, you can convert the results of your simulation model into a set of Tableau Data Extract files (*.tde), open them directly in Tableau, and visualize them with a few clicks.

This functionality is not present in Analytic Solver Cloud. See the chapter, Creating Custom Extensions in Tableau, that appears later on in this guide for more information on using Tableau with these two products.

Click the green arrow on the Output tab in the Solver Task Pane to run a simulation. Once Solver has stopped with a final result message, click the icon on the Task Pane Output tab or Uncertain Function dialog to save the values for the uncertain functions, uncertain variables, percentiles and statistics to *.tde files. You will be prompted to select a folder where the Tableau files will be saved along with the type of Tableau export desired: Tableau Data Extract or (*introduced in V2016-R2*) Tableau Web Connector.

Note: If exporting to an existing .tde file, data will be appended rather than overwritten. As a result, when exporting to an existing .tde file, all data must be of the same structure as when the .tde file was first created.



Choose the type of Tableau export: ● Tableau Data Extract ○ Tableau Web Connector

Folder: Windows (C:)

If Tableau Data Extract is selected, static data is exported to files saved on your hard drive.

Once you select a folder, the .tde files will be created, and the following message will appear.

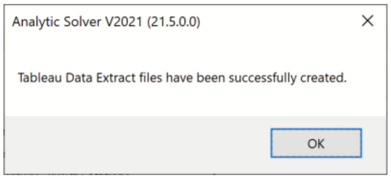


Tableau Data Extract files have been successfully created.

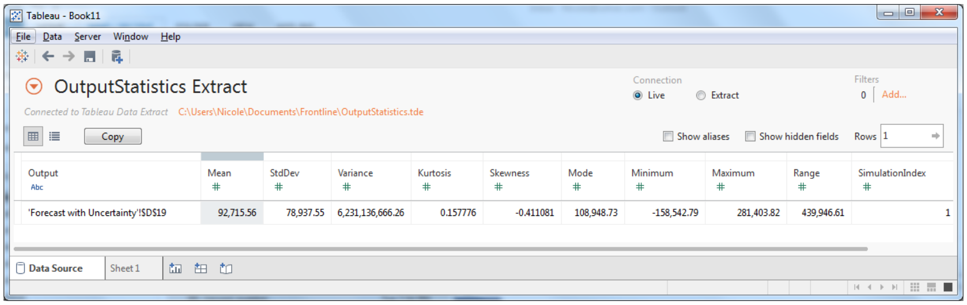
Four files will be saved in the folder you selected: InputTrials.tde, OutputStatistics.tde, OutputTrials.tde, and Percentiles.tde. (Although .tde files are designed to hold multiple tables, currently Tableau's software allows only one table per file.) When you import this data into Tableau, InputTrials.tde and OutputTrials.tde will have one row for each trial, Percentiles.tde will have one row for percentile and OutputStatistics.tde will have one row for each output function.

- Each row of InputTrials.tde will contain the trial value for each uncertain variable and the simulation index. The number of rows will equal the number of trials.
- Each row of OutputTrials.tde will contain the trial value for each uncertain function and the simulation index. The number of rows will equal the number of trials.
- Each row of Percentiles.tde will contain the percentile value for each uncertain function and the simulation index. The number of rows will equal 99.
- Each row of OutputStatistics.tde will contain the Excel cell address where the uncertain function is located along with 9 statistical functions

(mean, standard deviation, variance, kurtosis, skewness, mode, minimum, maximum, and range) and the simulation index.

Note: If exporting to an existing .tde file, data will be appended rather than overwritten. As a result, when exporting to an existing .tde file, all data must be of the same structure as when the .tde file was first created.

To open the files in Tableau, either double click each file (if using Desktop Tableau) or click **Other Files** under *Connect* and open the desired file(s).



OutputStatistics Extract

Connected to Tableau Data Extract C:\Users\Nicole\Documents\Frontline\OutputStatistics.tde

Connection: ● Live ○ Extract

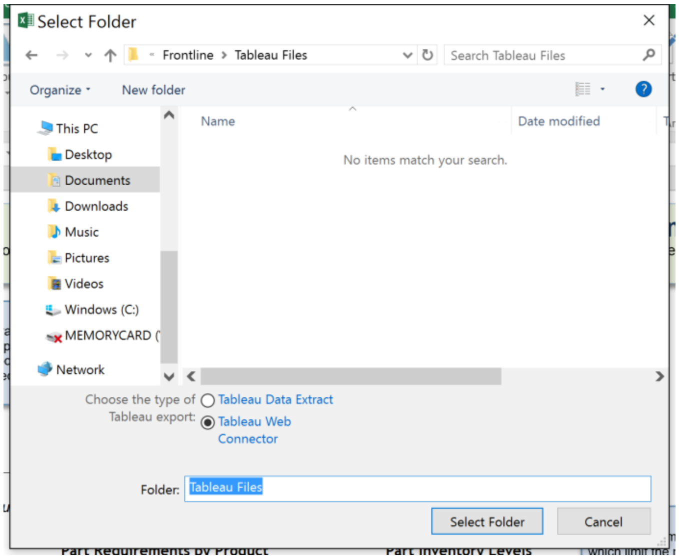
Filters: 0 Add...

☐ Show aliases ☐ Show hidden fields Rows 1

Output	Mean	StdDev	Variance	Kurtosis	Skewness	Mode	Minimum	Maximum	Range	SimulationIndex
'Forecast with Uncertainty'!$D$19	92,715.56	78,937.55	6,231,136,666.26	0.157776	-0.411081	108,948.73	-158,542.79	281,403.82	439,946.61	1

Tableau Web Connector

The Tableau Web Connector offers much more flexibility over Tableau Data Extract by allowing you to refresh your data dynamically inside of Tableau.



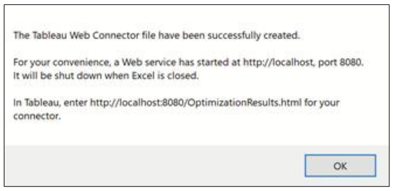
No items match your search.

Choose the type of Tableau export: ○ Tableau Data Extract ● Tableau Web Connector

Folder: Tableau Files

If Tableau Web Connector is selected, you will be prompted to select a folder in which to save SimulationResults.html. This file will hold all contents described above for InputTrials.tde, OutputStatistics.tde, OutputTrials.tde, and Percentiles.tde. The following message will appear.

Note: If exporting to an existing .tde file, data will be appended rather than overwritten. As a result, when exporting to an existing .tde file, all data must be of the same structure as when the .tde file was first created.

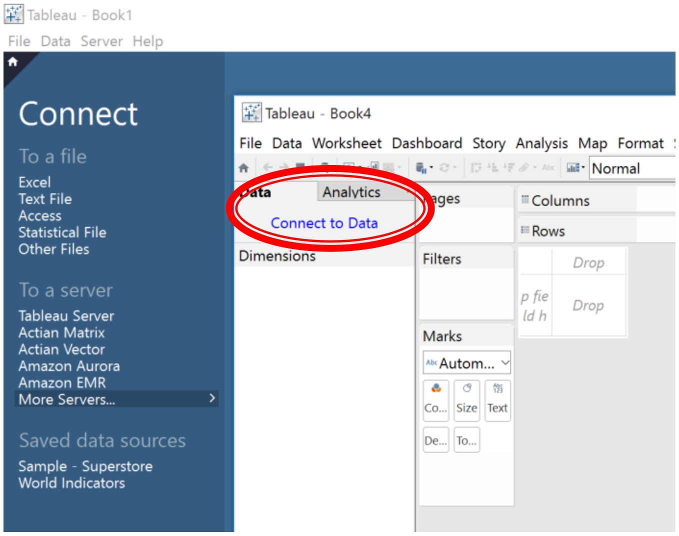


The Tableau Web Connector file have been successfully created.

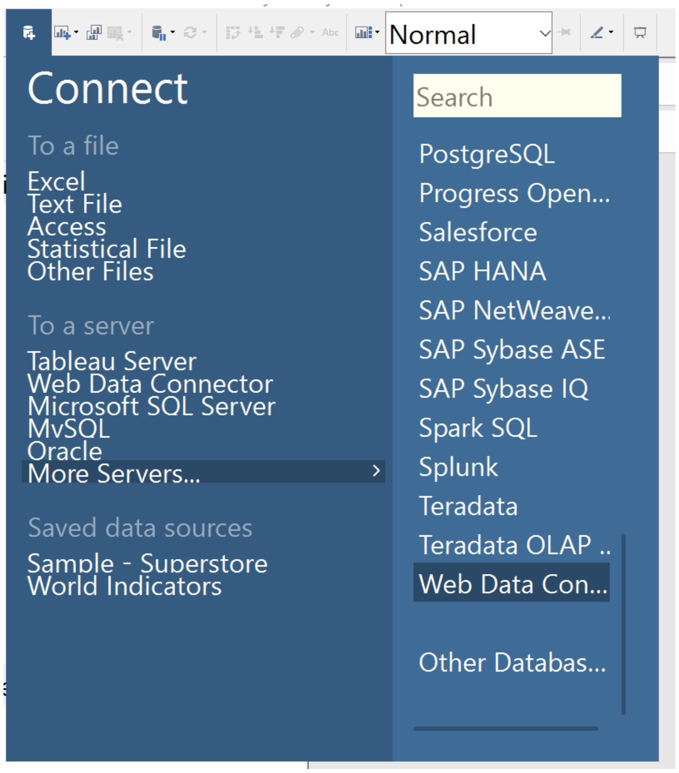
For your convenience, a Web service has started at http://localhost, port 8080. It will be shut down when Excel is closed.

In Tableau, enter http://localhost:8080/OptimizationResults.html for your connector.

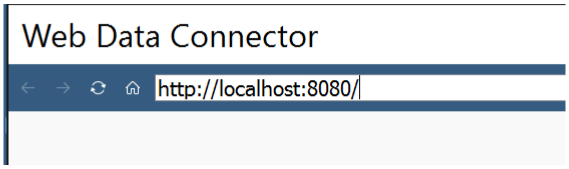
To open the files in Tableau, open a new workbook in Tableau and click **Connect to Data**.



On the Connect menu, select **More Servers – Web Data Connector**.



On the Web Data Connector dialog, enter the location displayed on the dialog shown above, i.e. http://localhost:8080/, and press **Enter**.

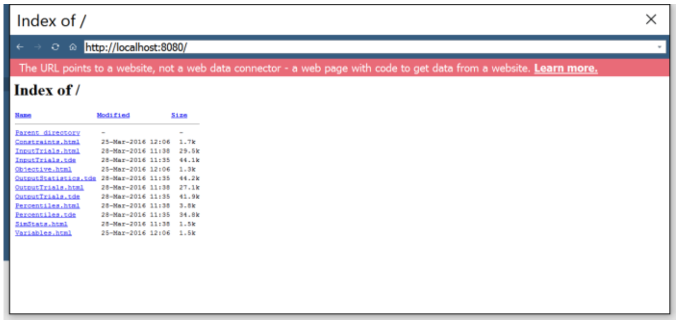


Web Data Connector

http://localhost:8080/

When the following dialog appears, click Constraints.html.

Note: The error highlighted in red is a simple warning that the URL is not pointing to a Tableau Web Data Connector file. If we had entered a file name, such as http://localhost:8080/InputTrials.html, the results would have immediately been uploaded to Tableau.



Index of /

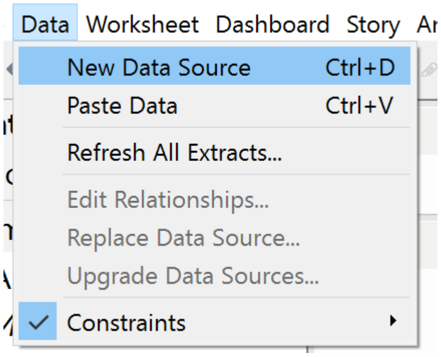
http://localhost:8080/

The URL points to a website, not a web data connector - a web page with code to get data from a website. Learn more.

Index of /

Name	Modified	Size
Parent directory	-	-
Constraints.html	25-Mar-2016 12:06	1.7k
InputTrials.html	28-Mar-2016 11:38	29.5k
InputTrials.tde	28-Mar-2016 11:35	44.1k
Objective.html	25-Mar-2016 12:06	1.3k
OutputStatistics.tde	28-Mar-2016 11:35	44.2k
OutputTrials.html	28-Mar-2016 11:38	27.1k
OutputTrials.tde	28-Mar-2016 11:35	41.9k
Percentiles.html	28-Mar-2016 11:38	3.8k
Percentiles.tde	28-Mar-2016 11:35	34.8k
SimStats.html	28-Mar-2016 11:38	1.5k
Variables.html	25-Mar-2016 12:06	1.5k

To add more data, click **Data – New Data Source** on the Tableau ribbon, then repeat the actions described above.



If your Solver results have changed, you'll need to refresh the results within the Tableau Web Connector HTML files. To do so:

1. In ASP, re-run your model
2. Click the Tableau icon on the Output tab in the Solver Task Pane.
3. Select **Tableau Web Connector** and the folder where the files should be saved.
4. Click **OK**.
5. In Tableau, click **Data – Refresh All Extracts** to update your data.

For more information on using Tableau, please refer to the Tableau documentation found at http://www.tableau.com/.

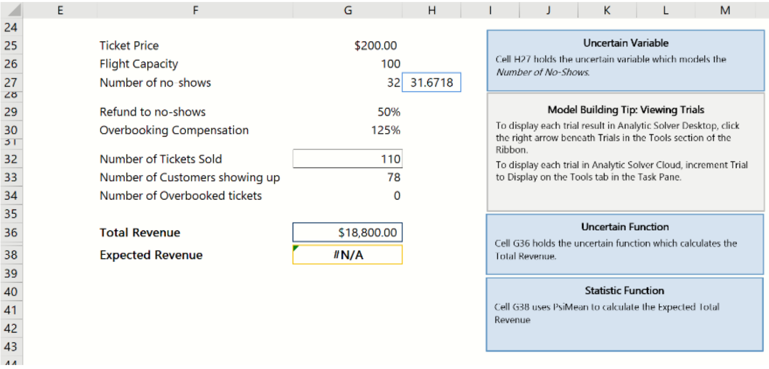
An Airline Revenue Management Model

In this section, we'll explore a simple airline revenue management model, also known as a yield management model. We'll start with a simple simulation model, like the one in the previous section. But in this section, we'll see if we can answer further questions by running *multiple parameterized simulations* and by solving a *simulation optimization* model (not available in Analytic Solver Simulation).

To open this example, click **Help – Examples** on the Ribbon, which opens the workbook, Frontline Example Models Overview.xlsx. Click the **Simulation tab**, then the **Yield Management Model 1** (a simple simulation model) and **Yield Management Model 2** (a model with multiple parameterized simulations) links. Click the **Simulation Optimization** tab to open **Yield Management Model 3**, a simulation optimization model.

A Single Simulation

The model YieldManagement1.xlsx opened in Analytic Solver Desktop is shown on the next page.



	F	G	H
25	Ticket Price	$200.00	
26	Flight Capacity	100	
27	Number of no shows	32	31.6718
29	Refund to no-shows	50%	
30	Overbooking Compensation	125%	
32	Number of Tickets Sold	110	
33	Number of Customers showing up	78	
34	Number of Overbooked tickets	0	
36	**Total Revenue**	$18,800.00	
38	**Expected Revenue**	#N/A	

Uncertain Variable

Cell H27 holds the uncertain variable which models the *Number of No-Shows.*

Model Building Tip: Viewing Trials

To display each trial result in Analytic Solver Desktop, click the right arrow beneath Trials in the Tools section of the Ribbon.

To display each trial in Analytic Solver Cloud, increment Trial to Display on the Tools tab in the Task Pane.

Uncertain Function

Cell G36 holds the uncertain function which calculates the Total Revenue.

Statistic Function

Cell G38 uses PsiMean to calculate the Expected Total Revenue

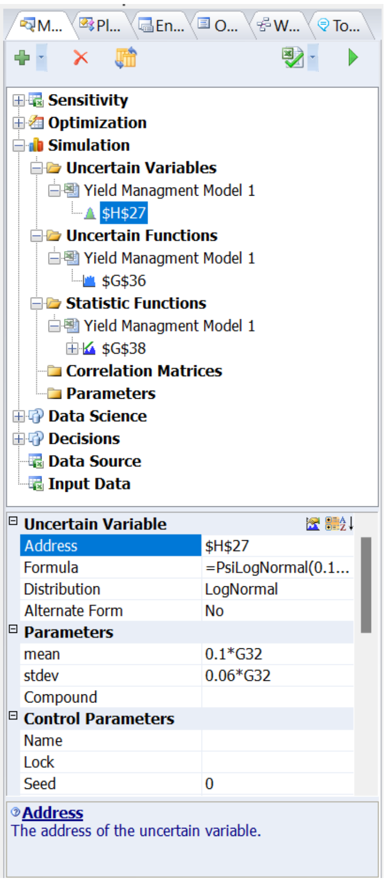
The model depicts a hypothetical airline flight from San Francisco to Seattle. The flight has 100 seats, and tickets are $200 per seat. Some passengers who purchase tickets are "no-shows" whose seats will be empty; in this example we assume that such passengers receive a refund of 50% of their purchase price.

To utilize their 'perishable inventory' of seats, the airline would like to sell more than 100 tickets for each flight. But we assume that Federal regulations require that any ticketed passenger who is unable to board the flight due to overbooking is entitled to compensation of 125% of the ticket price.

The airline would like to know how much revenue it will generate from each flight, less refunds for no-shows and compensation for 'bumped' passengers. As shown above, this total revenue amount is calculated at cell C31, for any specific number of tickets sold (110 above) and number of no-shows (6 above).

The uncertain quantity in this model is the number of no-shows; hence we should model this with an *uncertain variable*. We quickly realize that the **number of no-shows** will depend on the **number of tickets sold**. After some research, we decide that we can use a LogNormal distribution for the number of no-shows: Cell H27 contains the formula **=PsiLogNormal(0.1*G32,0.06*G32)**. Cell G27 contains =ROUND(H27,0) to ensure that the number of no-shows is an integer value – and this is used to compute Total Revenue at cell G36.

Note that the bottom area of the Task Pane is showing the properties of cell H27: for example, the *mean* of the distribution is 10% of the number of tickets sold.



Uncertain Variable

Address	$H$27
Formula	=PsiLogNormal(0.1…
Distribution	LogNormal
Alternate Form	No
Parameters	
mean	0.1*G32
stdev	0.06*G32
Compound	
Control Parameters	
Name	
Lock	
Seed	0

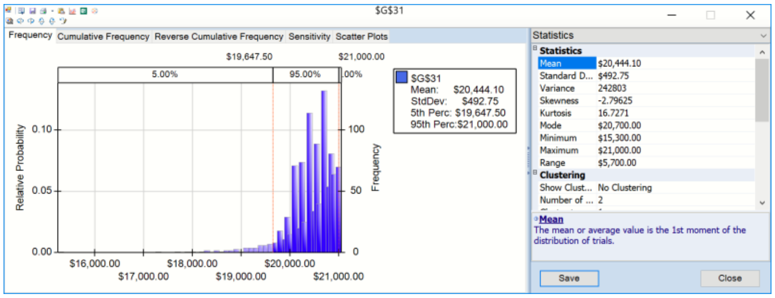
Address
The address of the uncertain variable.

See the chapter “Mastering Simulation and Risk Analysis Concepts” for advice on choosing distributions for your uncertain variables.

As shown in the Task Pane, cell G36 is an uncertain function; it is defined as such because cell G38 contains =PsiMean(G36) to compute the expected (or average) revenue across all Monte Carlo trials.

Performing the Simulation

To perform a single simulation (with 1,000 Monte Carlo trials), we click the green arrow on the Model tab of the Solver Task Pane. At the end of the simulation the following dialog appears displaying the distribution of outcomes in the Uncertain Function dialog:

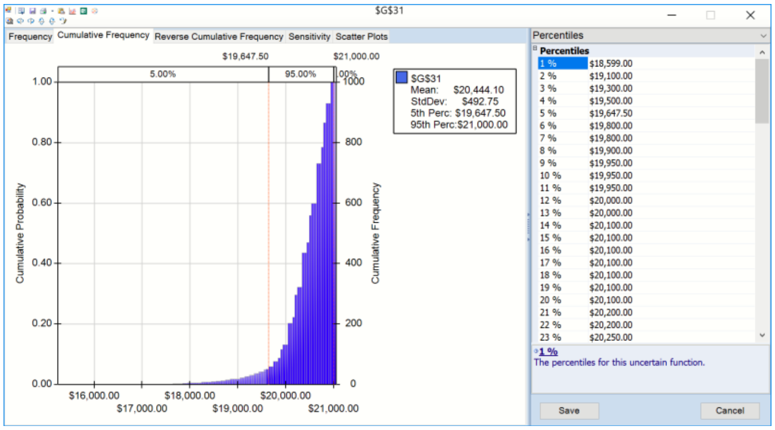


$G$31
Mean: $20,444.10
StdDev: $492.75
5th Perc: $19,647.50
95th Perc:$21,000.00

Statistics	
Mean	$20,444.10
Standard D...	$492.75
Variance	242803
Skewness	-2.79625
Kurtosis	16.7271
Mode	$20,700.00
Minimum	$15,300.00
Maximum	$21,000.00
Range	$5,700.00
Clustering	
Show Clust...	No Clustering
Number of ...	2

Mean
The mean or average value is the 1st moment of the distribution of trials.

The right hand panel shows the statistics from cell G36. We see that the mean or expected net revenue is about $20,000. From the **Cumulative Frequency tab** over the chart or by clicking on the drop down menu in the right hand panel (currently showing "Statistics") and selecting **Percentiles**, we can see that the 10% percentile is $19,950 – if we sell 110 tickets, we'll earn approximately as much revenue as a full flight (100 seats * $200), 90% of the time. You can see why airlines find that overbooking makes sense as a policy!

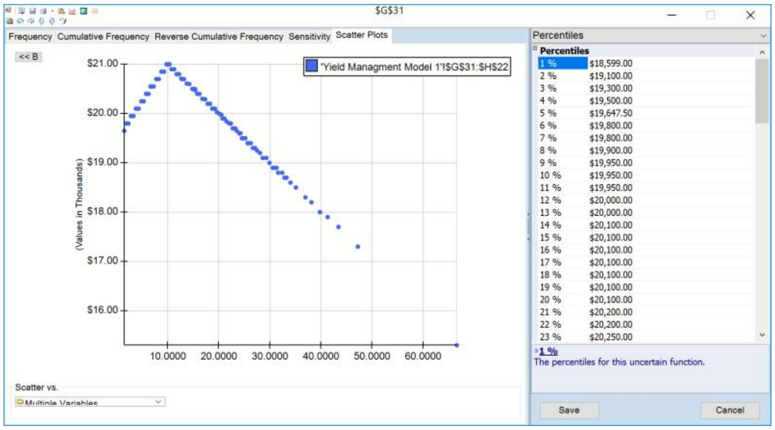


$G$31
Mean: $20,444.10
StdDev: $492.75
5th Perc: $19,647.50
95th Perc:$21,000.00

Percentiles	
1 %	$18,599.00
2 %	$19,100.00
3 %	$19,300.00
4 %	$19,500.00
5 %	$19,647.50
6 %	$19,800.00
7 %	$19,800.00
8 %	$19,900.00
9 %	$19,950.00
10 %	$19,950.00
11 %	$19,950.00
12 %	$20,000.00
13 %	$20,000.00
14 %	$20,100.00
15 %	$20,100.00
16 %	$20,100.00
17 %	$20,100.00
18 %	$20,100.00
19 %	$20,100.00
20 %	$20,100.00
21 %	$20,200.00
22 %	$20,200.00
23 %	$20,250.00

1 %
The percentiles for this uncertain function.

The **Scatter Plots tab** (click the graph to enlarge) shows you a quick scatter plot of this uncertain function against one or more uncertain variables, or other uncertain functions. In this example – shown on the next page – it shows a scatter plot of G36 (net revenue) against H27 (number of no-shows).



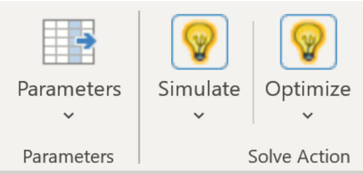
Scatter vs.
Multiple Variables

Percentiles	
1 %	$18,599.00
2 %	$19,100.00
3 %	$19,300.00
4 %	$19,500.00
5 %	$19,647.50
6 %	$19,800.00
7 %	$19,800.00
8 %	$19,900.00
9 %	$19,950.00
10 %	$19,950.00
11 %	$19,950.00
12 %	$20,000.00
13 %	$20,000.00
14 %	$20,100.00
15 %	$20,100.00
16 %	$20,100.00
17 %	$20,100.00
18 %	$20,100.00
19 %	$20,100.00
20 %	$20,100.00
21 %	$20,200.00
22 %	$20,200.00
23 %	$20,250.00

1 %
The percentiles for this uncertain function.

Using Interactive Simulation

If using Analytic Solver Desktop, we can explore the question of how many tickets to sell with Interactive Simulation: We simply click the Simulate button to "turn on the lights."

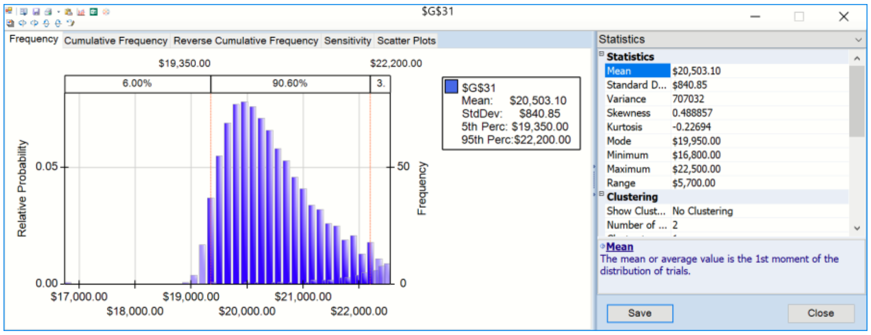


Now we can simply type a new number of Tickets Sold, such as **125**, into cell G32. Instantly, Analytic Solver (Desktop) performs a new simulation of 1,000

Monte Carlo trials, and updates the Uncertain Function dialog and the worksheet with the results – as shown on the next page.

The new distribution of outcomes is quite different – we have more high-revenue data points – but the mean or expected revenue is not so different -- $20,503. Selling 125 tickets does seem to be better than selling 110 tickets, but we're not yet sure that this is the *best* number of tickets to sell. (We'll answer that question in the next two sections.)

In the meantime, however, Interactive Simulation allows us to "play what-if" with this model, changing the parameter that we *can* control (the number of tickets sold), while properly modeling the behavior of the parameter we *cannot* control (the number of no-shows). As Prof. Sam Savage says, "Interactive Simulation does for uncertainty what the spreadsheet did for numbers."



Statistics	
Mean	$20,503.10
Standard D...	$840.85
Variance	707032
Skewness	0.488857
Kurtosis	-0.22694
Mode	$19,950.00
Minimum	$16,800.00
Maximum	$22,500.00
Range	$5,700.00
Clustering	
Show Clust...	No Clustering
Number of ...	2

Mean
The mean or average value is the 1st moment of the distribution of trials.

Note: Interactive Simulation is not supported in Analytic Solver Cloud. However, you can perform the same type of analysis in either product by simply running a new simulation after making the change to cell G32.

Multiple Parameterized Simulations

After playing "what-if" with Interactive Simulation, we can move to ask Analytic Solver to automate this process: Vary a parameter (the number of tickets sold) over a range, performing a simulation for each different number of tickets sold, and summarize the results.

Choose **Help – Examples** on the Ribbon, click the **Monte Carlo Simulation Examples** button, then scroll down to click the **Yield Management Model 2** link to open YieldManagement2.xlsx.

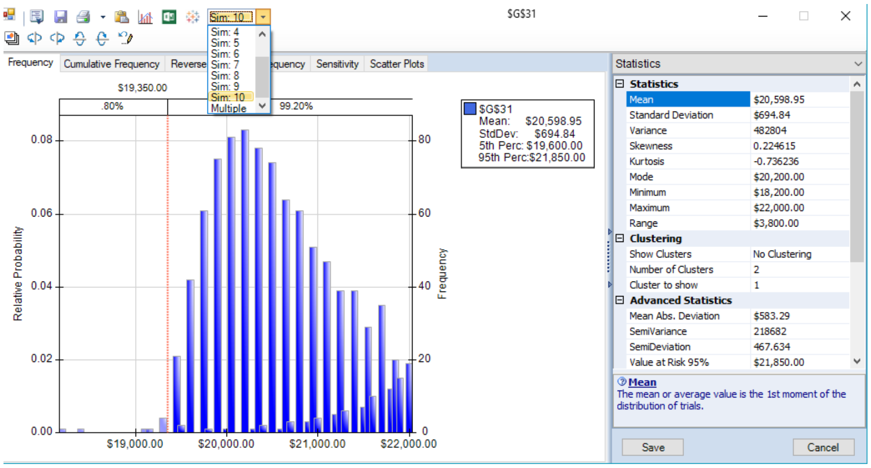
We can create YieldManagement2.xlsx from YieldManagement1.xlsx in just three steps:

1. Select cell G32, choose **Parameters – Simulation** from the Ribbon, and enter a lower limit of 102 and upper limit of 120 in the dialog. This creates a formula =PsiSimParam(102,120) in cell G32.
2. In the Task Pane Platform tab Simulation group, set the **Number of Simulations** to 10 (the maximum number of simulations allowed by Analytic Solver Upgrade). Cell G32 will take on the value 102 on the first simulation, 104 on the second simulation, 106 on the third simulation and so on, through 120 on the 10th simulation.
3. Select cell G36, choose **Results – Statistic – Mean**, and select a *range* of 10 cells, say G40:G50. Analytic Solver will place =PsiMean(G36,1) in the first cell, =PsiMean(G36,2) in the second cell, and so on for as many cells as you specify.

Now, when you run a simulation either by clicking the **green arrow** on the Model task pane or by using the **Simulate** icon on the Ribbon, Analytic Solver will perform 10 simulations, each with 1,000 Monte Carlo trials, and give you access to *all* of the results. On the worksheet, you'll see the expected revenue for 100, 102, …, 120 tickets sold. And you can use the Uncertain Function dialog to display the full distribution of outcomes for any of the 10 simulations.

Examining Results Across Simulations

Once the simulation finishes, the uncertain function dialog for cell G36 will open automatically or you can double-click the cell to open.



$G$31
Mean: $20,598.95
StdDev: $694.84
5th Perc: $19,600.00
95th Perc:$21,850.00

Statistics

Mean	$20,598.95
Standard Deviation	$694.84
Variance	482804
Skewness	0.224615
Kurtosis	-0.736236
Mode	$20,200.00
Minimum	$18,200.00
Maximum	$22,000.00
Range	$3,800.00

Clustering

Show Clusters	No Clustering
Number of Clusters	2
Cluster to show	1

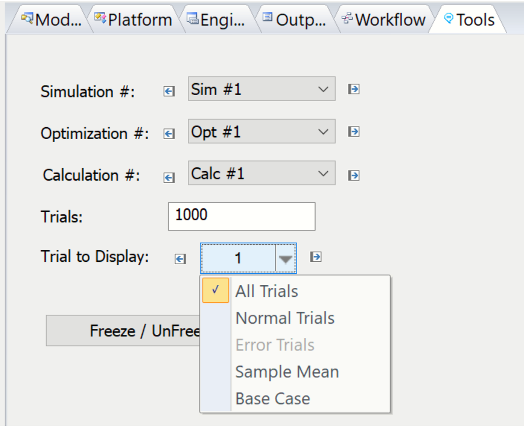
Advanced Statistics

Mean Abs. Deviation	$583.29
SemiVariance	218682
SemiDeviation	467.634
Value at Risk 95%	$21,850.00

Mean
The mean or average value is the 1st moment of the distribution of trials.

The dialog now includes a dropdown list that allows you to select a simulation to display. You can examine the Frequency, Cumulative Frequency, Reverse Cumulative Frequency, Sensitivity, or Scatter Plots tabs for any simulation.

The worksheet will initially display values for the last Monte Carlo trial of the last simulation performed (except for the cells containing PSI Statistics function calls such as PsiMean()). But you can display any trial from any simulation, by changing the selections in the Sim # and Trial # controls on the Tools tab of the Analytic Solver Task Pane:



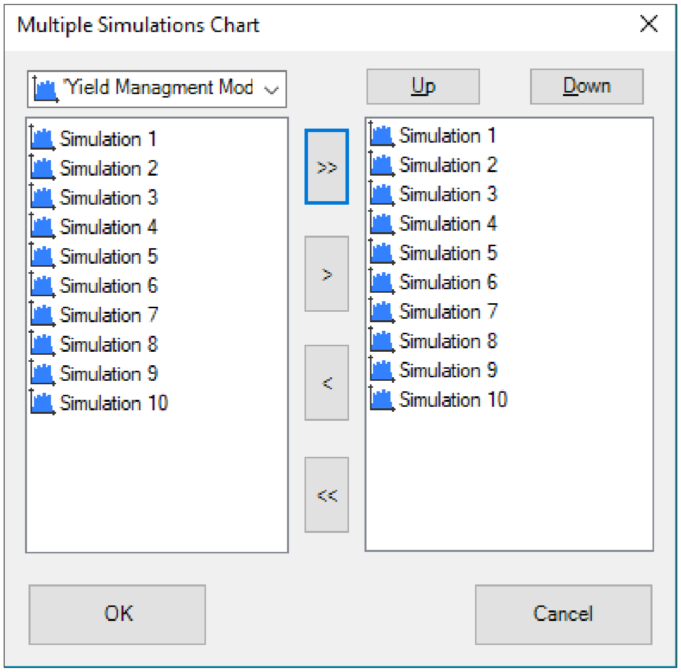
Simulation #: Sim #1
Optimization #: Opt #1
Calculation #: Calc #1
Trials: 1000
Trial to Display: 1

By examining this data, you can get a better idea of how many tickets to sell. But Analytic Solver can help your further.

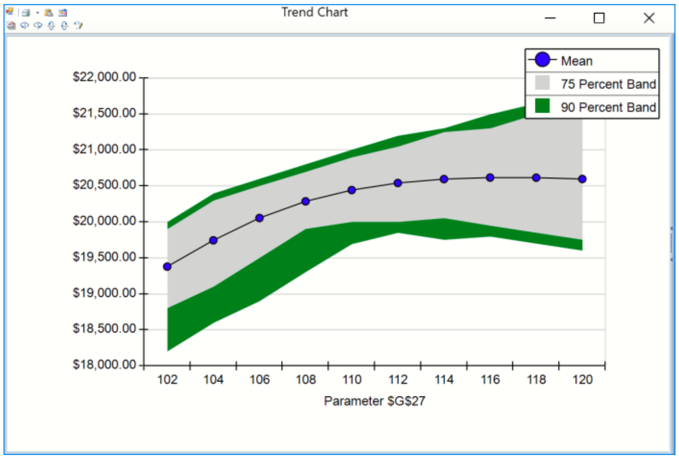
Reports and Charts of Multiple Simulations

Analytic Solver Comprehensive and Analytic Solver Optimization with Analytic Solver Simulation, in Analytic Solver Desktop or Analytic Solver Cloud, has rich facilities for reports and charts across multiple simulations. We'll cover just two chart examples; see the Frontline Solvers Reference Guide for further information.

First, select **Charts – Multiple Simulations – Trend** from the Ribbon. Analytic Solver displays a dialog where you can select one or more simulations to include in the chart. Click the >> button to select all 10 simulations:

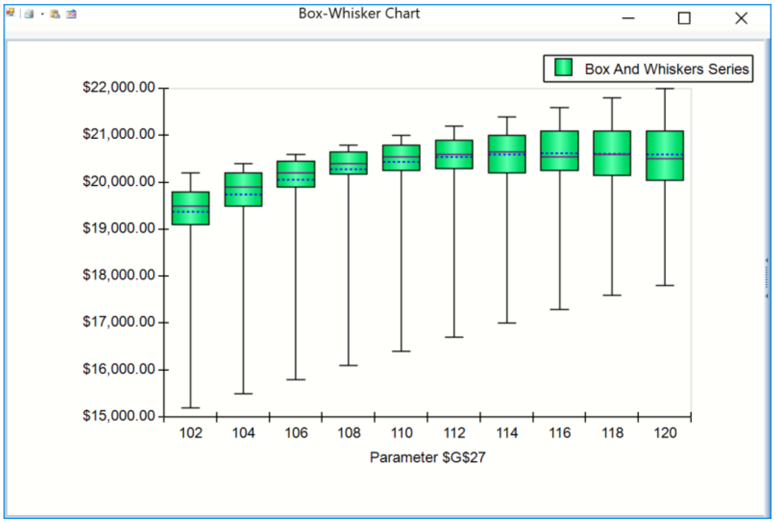


Analytic Solver then draws the Trend Chart:



The number of tickets sold is on the horizontal axis. Analytic Solver shows the mean, 25th and 75th percentile, and 10th and 90th percentile values of net revenue at G36 (this was the only choice in the dropdown list at the top left of the Multiple Simulations dialog). We can see that the mean or expected net revenue peaks at about 116 or 177 tickets sold.

For another view of this information, select **Charts – Multiple Simulations – Box-Whisker** from the Ribbon, and again select all 10 simulations to be included in the chart. Analytic Solver Comprehensive draws the following chart:



This shows a Box-Whisker diagram for each of the 10 simulations, with the mean, median, 25th and 75th percentiles, and the minimum and maximum value for net revenue on each one. In Analytic Solver Desktop, you can click the right edge of the Trend and Box-Whisker charts to open a right panel with options for customizing these charts. You can use the icons on the title toolbar to print the chart, or copy it to the Clipboard, where it can be pasted into another application such as PowerPoint.

Simulation Optimization

Running multiple parameterized simulations and examining the results on the worksheet and in chart form, has given us a good deal of intuition about the behavior of this simulation model. But Analytic Solver can *directly* answer the question “How many tickets should we sell to realize the *maximum expected* net revenue?”

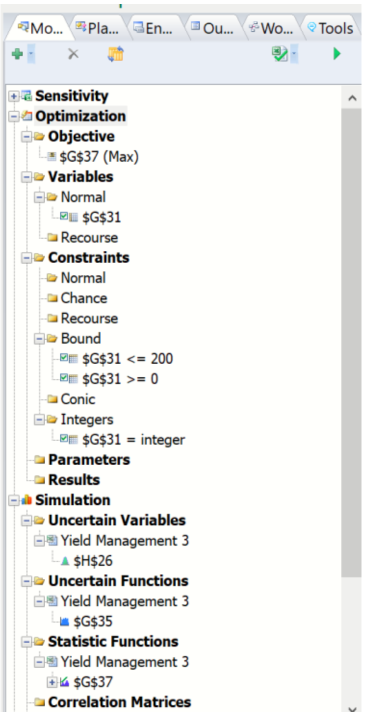
You can see this by choosing **Help – Examples** on the Ribbon, then clicking the Simulation Optimization button. Click the link to **Yield Management Model 3** in the list of example workbooks. This example will create YieldManagement3.xlsx from the existing YieldManagement1.xlsx, used in the previous two examples, in just five steps:

1. Select cell G32 and click **Decisions** on the Ribbon. This makes the number of tickets sold a decision variable for optimization. Alternatively, click the down arrow under Decisions, then select Normal to make the number of tickets sold a decision variable.
2. With G32 still selected, click the down arrow under **Constraints**, then choose **Variable Type/Bound – Integer** from the menu. This specifies that G32 must have an integer value at the optimal solution.
3. With G32 still selected, click the down arrow under **Constraints**, then choose **Variable Type/Bound – <=** from the menu, and enter 200 for the constraint left hand side. This specifies that G32 must have an upper bound of 200.

4. With G32 still selected, click the down arrow under **Constraints**, then choose **Variable Type/Bound – <=** from the menu, and enter 0 for the constraint left hand side. This specifies that G32 must have lower bound of 0.

5. Select cell G38, click the down arrow under **Objective**, then choose **Max – Normal** from the menu. The optimizer will maximize G38: = PsiMean(G31), the expected value of net revenue.

The model in Task Pane will now look like the one pictured below.

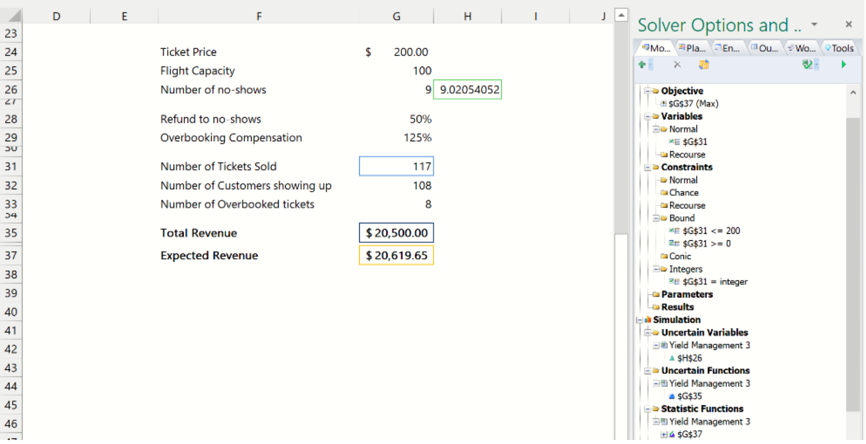


- Sensitivity
- Optimization
  - Objective
    - $G$37 (Max)
  - Variables
    - Normal
      - $G$31
    - Recourse
  - Constraints
    - Normal
    - Chance
    - Recourse
    - Bound
      - $G$31 <= 200
      - $G$31 >= 0
    - Conic
    - Integers
      - $G$31 = integer
  - Parameters
  - Results
- Simulation
  - Uncertain Variables
    - Yield Management 3
      - $H$26
  - Uncertain Functions
    - Yield Management 3
      - $G$35
  - Statistic Functions
    - Yield Management 3
      - $G$37
  - Correlation Matrices

In YieldManagement3.xlsx, we've made just these selections, and we've also chosen the GRG Nonlinear Solver from the dropdown list on the Engine tab, to solve the problem. (Other choices are possible, but this one is simplest for this example problem.)

Now we can simply click the **green arrow** on the Task Pane. After a moment of solving, Analytic Solver returns the solution of 116 or 117 tickets sold appears in G32 (the exact number depends on the simulation Random Seed), and the expected value of net revenue for this number of tickets appears at G32 (about $20,500). "Solver found a solution. All constraints and optimality conditions are satisfied" appears in green in the Output tab of the Task Pane, as shown on the next page.

Note that this required less than a page to explain how you can get answers to questions like "How many tickets should we sell to realize the *maximum expected* net revenue?"



Ticket Price	$ 200.00	
Flight Capacity	100	
Number of no-shows	9	9.02054052
Refund to no-shows	50%	
Overbooking Compensation	125%	
Number of Tickets Sold	117	
Number of Customers showing up	108	
Number of Overbooked tickets	8	
Total Revenue	$ 20,500.00	
Expected Revenue	$ 20,619.65	

- **Objective**
  - $G$37 (Max)
- **Variables**
  - Normal
    - $G$31
  - Recourse
- **Constraints**
  - Normal
  - Chance
  - Recourse
  - Bound
    - $G$31 <= 200
    - $G$31 >= 0
  - Conic
  - Integers
    - $G$31 = integer
- **Parameters**
- **Results**
- **Simulation**
  - **Uncertain Variables**
    - Yield Management 3
      - $H$26
  - **Uncertain Functions**
    - Yield Management 3
      - $G$35
  - **Statistic Functions**
    - Yield Management 3
      - $G$37

Using a Compound Distribution

Starting with V2016-R2, Analytic Solver Comprehensive and Analytic Solver Simulation include the ability to sum multiple independent random variables using a compound distribution. A compound distribution generates values for the sum of N independent identically distributed uncertain variables. In other words, a compound distribution models the probability that two independent events occur together.

There are many common applications for compound distributions in industries across the board. For example, in the insurance industry, compound distributions may be used to estimate the amount of claim payouts to customers or a portfolio of policies over a period of time by modeling the **number** of claims paid, along with the **size** of each claim.

In healthcare, compound distributions can be utilized for many reasons including to estimate the total amount of demand for a specific drug over a period of time, the number of days a patient will spend in hospital or to estimate the total amount of exposure to a certain toxin (i.e., radiation or radon) that a group of people might encounter in their lifetime. In addition, compound distributions are used in the food industry to estimate the total number of patients that will become sick due to a food-borne illness.

The incorrect implementation of creating a compound distribution manually is a primary cause of errors in a simulation model. In an effort to reduce these errors, Frontline Systems has implemented compound distributions with the introduction of a single property, PsiCompound(). This property supports any existing Psi Distribution function except the Psi multi-variate functions (PsiMVLogNormal, PsiMVNormal, PsiMVResample, and PsiMVShuffle), PsiSip, and PsiSlurp. PsiCompound is amended to a Psi Distribution, i.e., PsiBeta(3,2, PsiCompound(100)) The signature of this property is:

```
PsiCompound(number_cell, deduction, limit)
```

A compound distribution is made up of a "severity" distribution and a "frequency" distribution. Assume the following compound distribution, =PsiBeta(3, 2, PsiCompound(A2)), where A2 = PsiPoisson(100). PsiBeta(3,2) is referred to as the "severity" distribution. The severity distribution is the distribution to be added N times. PsiPoisson(100) is referred to as the "frequency" distribution. The frequency distribution determines the size N of the sum (i.e, how many PsiBeta to sum). N can be a constant but can also be computed at each trial by drawing from a discrete distribution.

- The **number_cell** argument passes the number of random trial values to be summed. Number_cell can be an integer, a cell containing an integer, a formula evaluating to an integer, or a cell containing a discrete distribution.

  If a fractional value is passed directly or indirectly (by using a formula) or a continuous distribution is passed to this argument, the result will be rounded down to the nearest integer.

  Note: If a discrete distribution is passed to the number_cell argument, the frequency distribution must be formulated in such a way that the trial values generated by the distribution must be greater than 1. If not, trial values < 1 will be set equal to 1.
- The value passed to the **deduction** argument is subtracted from every term of the compound sum which results in a shift of the compound distribution by -N * *deduction*.
- If a trial value is larger than a specified **limit**, then the trial value is reset to the limit.

Depending on what is passed to the number_cell argument, PsiCompound() automatically selects from several different compounding methods.

  - First, ASP tries to compute the distribution analytically. For example "=PsiExponential(par, PsiCompound(N))" can be computed as PsiGamma(N, par). Analytic Solver can immediately sample from this "shortcut" distribution.
  - If Analytic Solver is unable to compute a compound distribution analytically, but the frequency of the severity function (N) is greater than the value for the CLT Threshold option, then the distribution will be computed according to the Central Limit Theorem as PsiNormal(m, s). The parameters m and s will be computed analytically from the corresponding analytical moments of the severity distribution. The maximum value allowed for the CLT Threshold option is 1000 while the minimum value allowed is 1. The default setting is 100.
  - Otherwise, the compound distribution will be computed using Monte Carlo simulation to sum up N independent variates of the severity distribution.

The following examples illustrate how to use the PsiCompound() property when a discrete distribution or a constant value is passed to the number_cell argument

Discrete Distribution passed to 1st Argument

To follow along, open the InsuranceLossModel(Sim).xlsx simulation example by clicking Help – Example Models – Monte Carlo Simulation Examples – Insurance Loss Model. This example includes two tabs: Insurance Loss Model – Original and Insurance Loss Model – Compound.

Insurance Loss Model – Original tab: Imagine an insurance policy has two basic classes of insured members, class A and class B; there are 34 class A members and 10 class B members. The number of claims from class A members (in cell I24) follows a Negative Binomial distribution with parameters (5, 0.25), and the number of claims from class B members (in cell I25) follows a geometric distribution with parameter 0.25. The size of a claim from class A members follows an exponential distribution with mean $40,000 (see cells E25:E58), and the claim size from class B members is distributed with a Pearson

5 distribution with parameters (10, 2000000), see cells F25:F34. In other words, if there are six Class A claims, the size of the payout for each of the six claims will follow the PsiExponential(40000) distribution. The goal of this model is to calculate the expected claim payment for both classes.

The first tab formulates the model by simply adding up all the claims for each of the classes; cell I29 totals the claims for Class A and cell I30 totals the claims for Class B. The total claim amount is calculated in cell I32 and the expected total claim amount is calculated in cell I33.

Insurance Loss Model – Compound tab: The model on this tab has been reformulated using two compound distributions in cells D25 and D26 to replace the uncertain variables in cells E25:E58 and F25:F34, respectively.

Total Amount Paid for Class A Claims: Recall that the *number* of Class A claims follows a Negative Binomial distribution using a success parameter = 5 and probability parameter = 0.25 while the *size* of a Class A claim follows an exponential distribution with mean $40,000.

See the compound distribution in cell D24. This compound distribution replaces the sum of the uncertain variables in cell Insurance Loss Model – Original!CI29. In this formulation, PsiExponential(40000) is the severity distribution. This is the distribution that will be added together. If there are six claims, then PsiExponential(40000) will be compounded six times. The frequency distribution is the number of Class A claims, =PsiNegBinomial(5, 0.25, PsiTruncate(0, 34)).

Note: Since there are 34 class A members, the PsiTruncate property is used to ensure that sample values *always* lie within the range from 0 to 34.

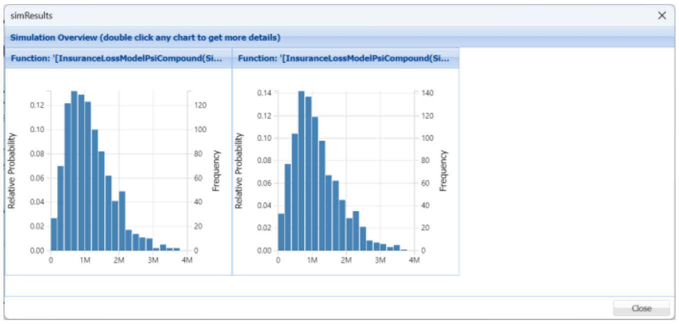
If D21 = PsiNegBinomial(5, 0.25, PsiTruncate(0, 34), the full compounding formula for the amount paid to Class A claims can be entered into cell D24 as "=PsiExponential(40000, PsiCompound(I21))".

Total Amount Paid for Class B Claims: The second compound distribution may be formulated in the same manner. PsiPearson5(10, 2000000) is the severity distribution while =PsiGeometric(0.25, PsiTruncate(0, 10)) is the frequency function (in cell D22). Since there are 10 class B members, PsiTruncate is used to ensure that sample values lie within the range 0 to 10.

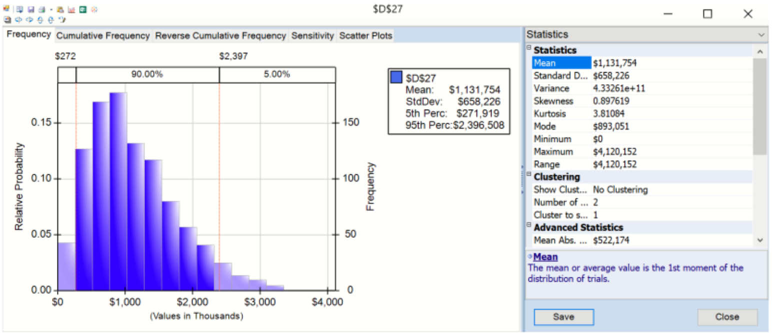
If D22 = PsiGeometric(0.25, PsiTruncate(0, 10), the full compounding formula for the amount paid to Class B claims can be entered into cell D25 as "= PsiPearson5(10,2000000, PsiCompound(D22))".

The Total Amount Paid to both classes is calculated in cell D27 and the Expected Total Amount Paid is calculated in cell D28 using the PsiMean function.

Run a simulation by clicking the green arrow on the Output tab on the Solver Task Pane. Instantly, Analytic Solver performs a new simulation of 1,000 Monte Carlo trials, and updates the Uncertain Function dialog and the worksheet with the results – as shown on the next page.


simResults
Simulation Overview (double click any chart to get more details)
Function: '[InsuranceLossModelPsiCompound(Si...
Function: '[InsuranceLossModelPsiCompound(Si...
Relative Probability
Frequency
0.12
0.10
0.08
0.06
0.04
0.02
0.00
120
100
80
60
40
20
0
0
1M
2M
3M
4M
0.14
140
Close

Double click the first distribution to open the uncertain function dialog for the Total Amount Paid.

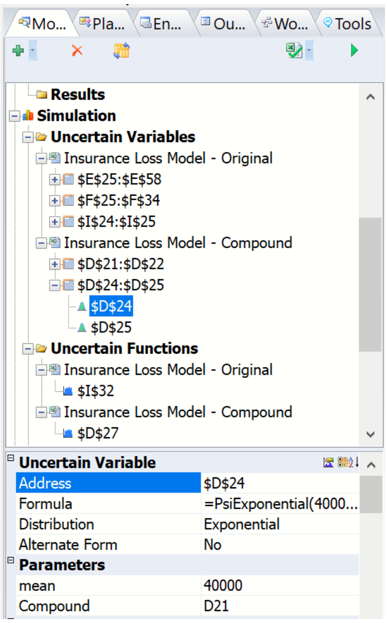


$D$27
Mean: $1,131,754
StdDev: $658,226
5th Perc: $271,919
95th Perc:$2,396,508

Statistics	
Mean	$1,131,754
Standard D...	$658,226
Variance	4.33261e+11
Skewness	0.897619
Kurtosis	3.81084
Mode	$893,051
Minimum	$0
Maximum	$4,120,152
Range	$4,120,152
Clustering	
Show Clust...	No Clustering
Number of ...	2
Cluster to s...	1
Advanced Statistics	
Mean Abs. ...	$522,174

Mean
The mean or average value is the 1st moment of the distribution of trials.

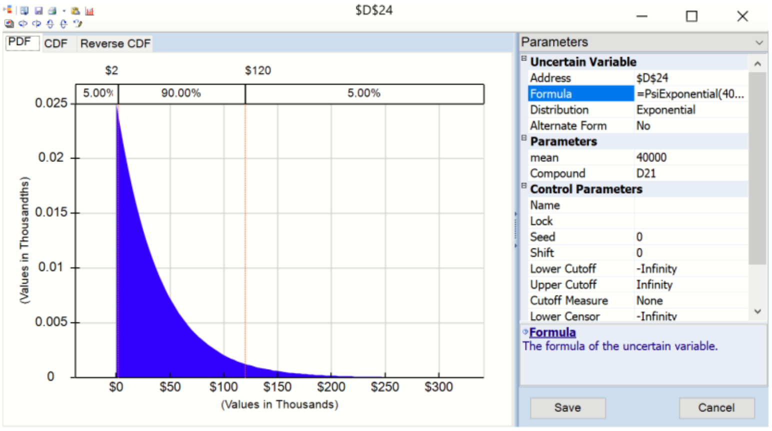
Close both dialogs by clicking X in the upper right hand corner. Then double click cell D24 on the Model tab of the Solver Task Pane to open the Uncertain Variable dialog.



- Results
- Simulation
  - Uncertain Variables
    - Insurance Loss Model - Original
      - $E$25:$E$58
      - $F$25:$F$34
      - $I$24:$I$25
    - Insurance Loss Model - Compound
      - $D$21:$D$22
      - $D$24:$D$25
        - $D$24
        - $D$25
  - Uncertain Functions
    - Insurance Loss Model - Original
      - $I$32
    - Insurance Loss Model - Compound
      - $D$27

Uncertain Variable	
Address	$D$24
Formula	=PsiExponential(4000...
Distribution	Exponential
Alternate Form	No
Parameters	
mean	40000
Compound	D21

If an uncertain variable uses the PsiCompound() property to create a compound distribution, you will see added fields on both the Parameter and Analytic Moments menus. Curves displayed in the Uncertain Variable dialog and statistics listed under the Parameters tab are related to the "severity" distribution.

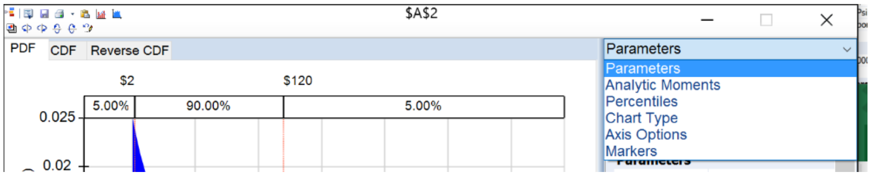


Uncertain Variable	
Address	$D$24
Formula	=PsiExponential(40...
Distribution	Exponential
Alternate Form	No
Parameters	
mean	40000
Compound	D21
Control Parameters	
Name	
Lock	
Seed	0
Shift	0
Lower Cutoff	-Infinity
Upper Cutoff	Infinity
Cutoff Measure	None
Lower Censor	-Infinity

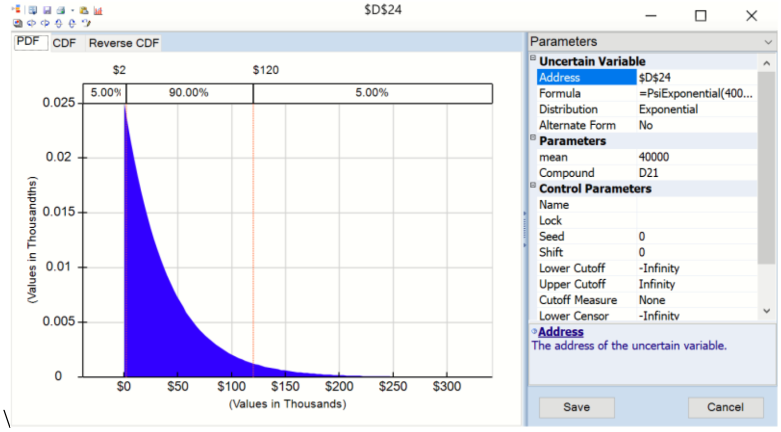
Formula
The formula of the uncertain variable.

On the Parameter tab, the Compound field will appear under Parameters. (If the distribution is not a compound distribution, this field will be blank.) This field holds the arguments passed to the PsiCompound property.

Click the down arrow next to Parameters and select Analytic Moments from the menu.



On the Analytic Moments menu, four additional statistics relating to the compound distribution appear under Compound Analytic Moments: Mean, Variance, Standard Deviation and Skewness. Statistics listed under Analytic Moments relate to the severity distribution.



Uncertain Variable	
Address	$D$24
Formula	=PsiExponential(400...
Distribution	Exponential
Alternate Form	No
Parameters	
mean	40000
Compound	D21
Control Parameters	
Name	
Lock	
Seed	0
Shift	0
Lower Cutoff	-Infinity
Upper Cutoff	Infinity
Cutoff Measure	None
Lower Censor	-Infinity

Address
The address of the uncertain variable.

\

Constant passed to 1st Argument

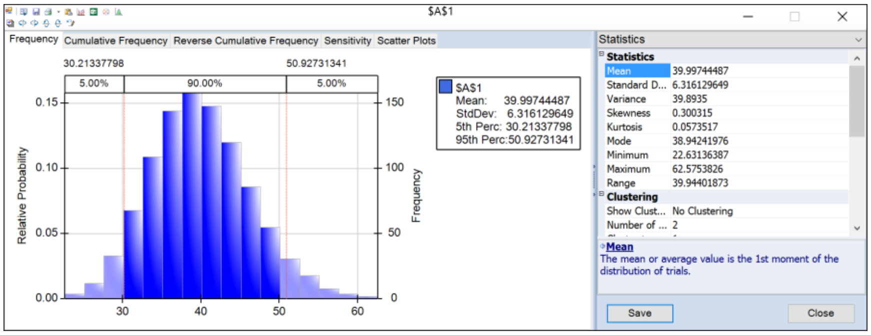
Recall that it is also possible to pass an integer, a cell containing an integer or a formula evaluating to an integer to the number_cell argument, the first argument of PsiCompound().

Imagine a department store manager predicting demand for snowboards for the coming winter season. From past analysis, she knows that the purchasing behavior of customers that visit her store follow the PsiGamma distribution with parameters of shape = 4 and scale = 1. She expects 1000 customers in the month of October. Using a compound distribution, calculate the expected number of snowboard purchases.

The severity distribution in this example is PsiGamma(4,1). Since N = 1000, a constant, no frequency distribution exists. Therefore, the syntax for this compound distribution would be =PsiGamma(4, 1, PsiCompound(1000).

Select the cell containing the compound distribution, then click **Results – Output – In Cell** on the Analytic Solver ribbon; "+ PsiOutput()" will be amended to the existing formula.

Again, run a simulation by clicking the green arrow on the Output tab of the Solver Task Pane. Instantly, Analytic Solver Simulation performs a new simulation of 1,000 Monte Carlo trials, and updates the Uncertain Function dialog and the worksheet with the results. If the worksheet only contains this one output function, the following dialog appears.

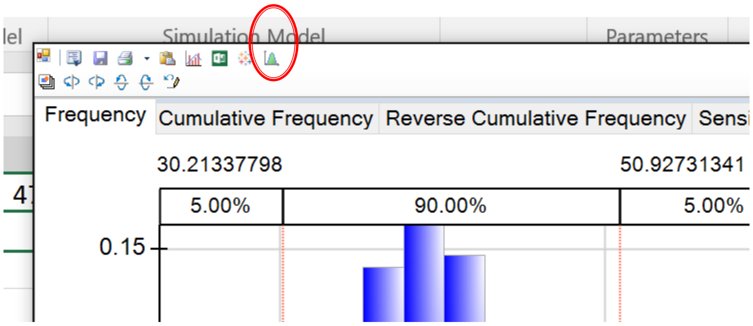


$A$1
Mean: 39.99744487
StdDev: 6.316129649
5th Perc: 30.21337798
95th Perc: 50.92731341

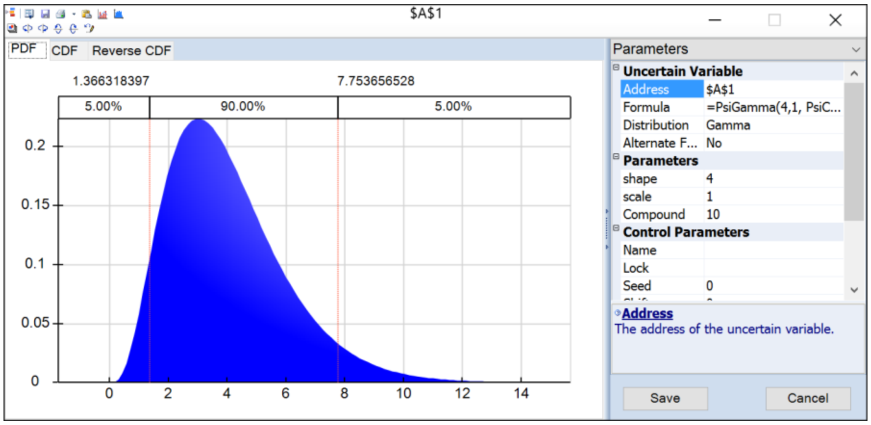
Statistics	
Mean	39.99744487
Standard D...	6.316129649
Variance	39.8935
Skewness	0.300315
Kurtosis	0.0573517
Mode	38.94241976
Minimum	22.63136387
Maximum	62.5753826
Range	39.94401873
Clustering	
Show Clust...	No Clustering
Number of ...	2

Mean
The mean or average value is the 1st moment of the distribution of trials.

To view the compound distribution results, click the Show Input icon, circled below.



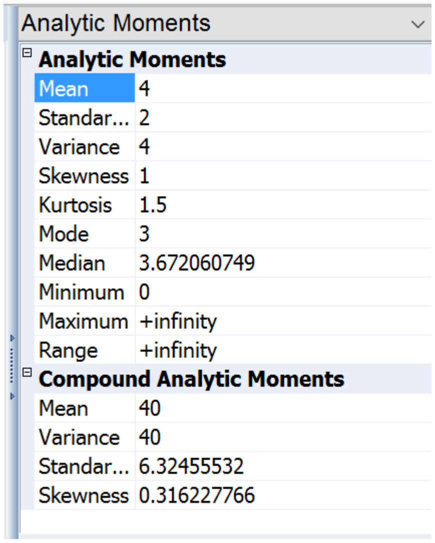
The Uncertain Variable dialog opens. Again, we see the Compound field, but this time, the constant 10 appears here.



Uncertain Variable	
Address	$A$1
Formula	=PsiGamma(4,1, PsiC...
Distribution	Gamma
Alternate F...	No
Parameters	
shape	4
scale	1
Compound	10
Control Parameters	
Name	
Lock	
Seed	0

Address
The address of the uncertain variable.

Click the down arrow next to Parameters to view the Analytic Moments for both the severity distribution (under Analytic Moments) and the compound distribution (under Compound Analytic Moments).

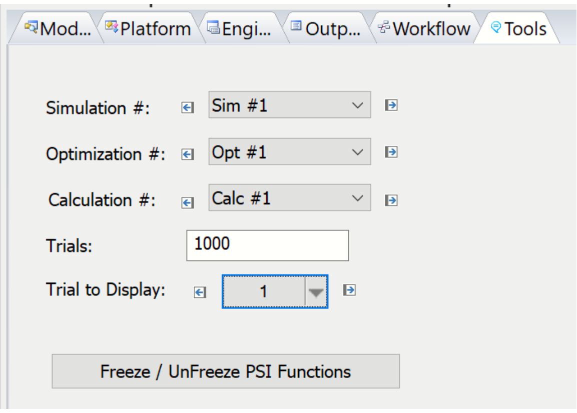

Analytic Moments

Analytic Moments	
Mean	4
Standar...	2
Variance	4
Skewness	1
Kurtosis	1.5
Mode	3
Median	3.672060749
Minimum	0
Maximum	+infinity
Range	+infinity
Compound Analytic Moments	
Mean	40
Variance	40
Standar...	6.32455532
Skewness	0.316227766

Publishing a Simulation Model to Excel Online or Google Sheets

In late 2014, Frontline Systems released Solver App for use with Excel 2013 and Solver Add-on for use with Google Sheets. With the Solver App you can define and solve simulation problems in your Excel workbook, using Excel Online in Office 365, the Excel Web App in SharePoint 2016 or 2013, or desktop Excel 2016 or 2013. With the Solver Add-on, you can define and solve simulation problems in Google Sheets. You can create and solve models on tablets, phones, or anywhere a Web browser can be used. Solver models that you may already have created in old or new versions of Microsoft Excel, are automatically recognized by the Solver App and , after the existing worksheet is published, with the Solver Add-on.

In Analytic Solver Desktop, the **Freeze** and **Thaw** buttons were combined into a single button which now appears on the Tools tab of the Solver Task Pane, as shown in the screenshot below. (Freeze/Unfreeze is not supported in Premium Solver or Analytic Solver Cloud.)



Simulation #: Sim #1

Optimization #: Opt #1

Calculation #: Calc #1

Trials: 1000

Trial to Display: 1

The “Freeze/UnFreeze” button can be used to prepare a workbook for distribution to other users who don’t have Analytic Solver or its subsets installed: All formulas containing Psi function calls (which would yield #NAME? for other users) are replaced by their values, and the formulas are saved, so they can be restored later by choosing “Unpublish” (equivalent to the old “Thaw”).

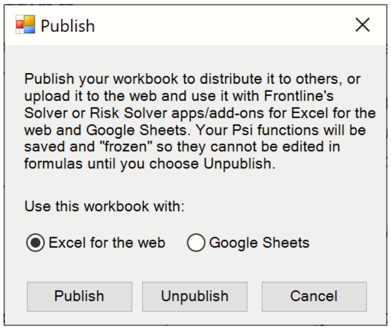
But the major use of this button is to prepare a workbook for use with our Risk Solver App for Excel Online and our Risk Solver Add-on for Google Sheets (the online spreadsheets have limited or no support for user-defined functions).

In *Analytic Solver*, when a model is published, the limits for Solver App (for Excel Online) and Solver Add-on (for Google Sheets) will be automatically adjusted to match the problem limits of your license. For example, if you purchased a license for Analytic Solver, then you will be able to run simulation models with an unlimited number of uncertain variables and functions, *if* you publish your model first by clicking the Publish button on the Analytic Solver ribbon.

Note: When using Publish/Unpublish, the worksheet may not contain CTRL+SHIFT+ENTER arrays.

Note: When using Publish/Unpublish, the worksheet may not contain CTRL+SHIFT+ENTER arrays.

Let’s go back to the YieldManagement1.xlsx example used in the section above, click the Freeze/Unfreeze button on the Tools tab of the Solver Task Pane to display the following dialog.

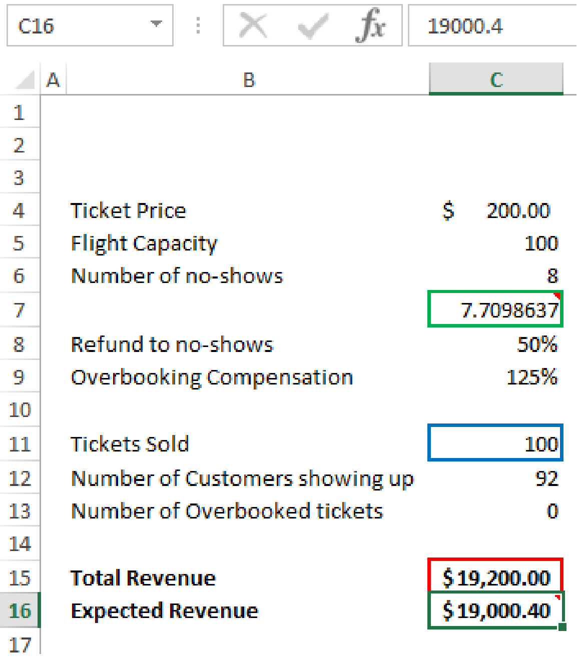


Publish your workbook to distribute it to others, or upload it to the web and use it with Frontline's Solver or Risk Solver apps/add-ons for Excel for the web and Google Sheets. Your Psi functions will be saved and "frozen" so they cannot be edited in formulas until you choose Unpublish.

Use this workbook with:

● Excel for the web ○ Google Sheets

Select **Google Sheets** then **Publish**. (If you are a user of Excel Online, please see the example below.) If you inspect the worksheet, you’ll notice that all Psi functions have been replaced with actual values. For example, cell C16 which used to contain the PsiMean() function, now contains a value of $19,00.40. Make sure to **save** the workbook before uploading to Google Drive.



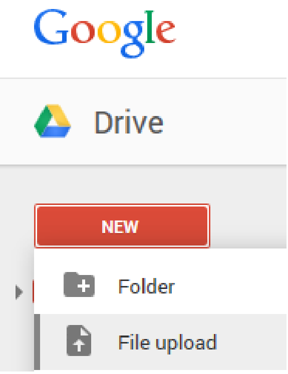
C16 19000.4

	A	B	C
1			
2			
3			
4		Ticket Price	$ 200.00
5		Flight Capacity	100
6		Number of no-shows	8
7			7.7098637
8		Refund to no-shows	50%
9		Overbooking Compensation	125%
10			
11		Tickets Sold	100
12		Number of Customers showing up	92
13		Number of Overbooked tickets	0
14			
15		**Total Revenue**	**$ 19,200.00**
16		**Expected Revenue**	**$ 19,000.40**
17			

Log on to your Google Drive account and upload YieldManagement1.xlsx, then open this file in Google Sheets.

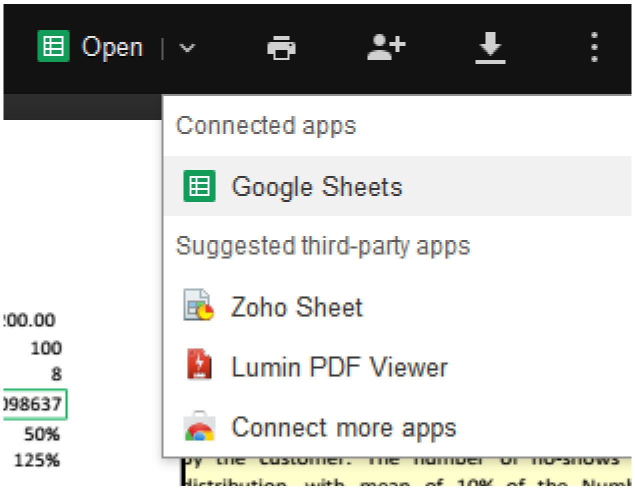
Uploading and Opening Files in Google Sheets

Click the orange New button then select File Upload to upload the YieldManagement1.xlsx spreadsheet to Google Drive.

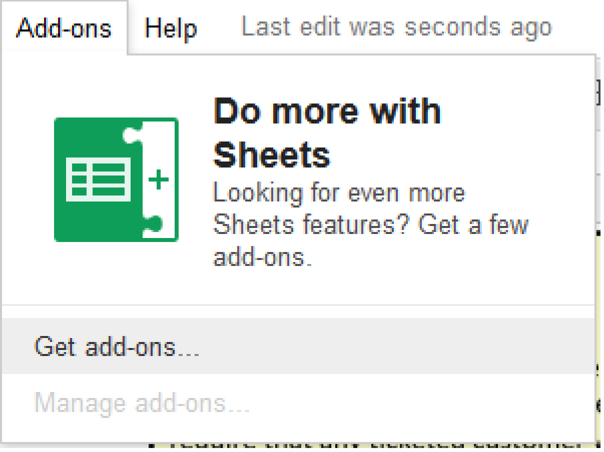


Browse to the location of the file, typically C:\ProgramData\Frontline Systems\ Examples. The YieldManagementModel1.xlsx will appear under My Drive.

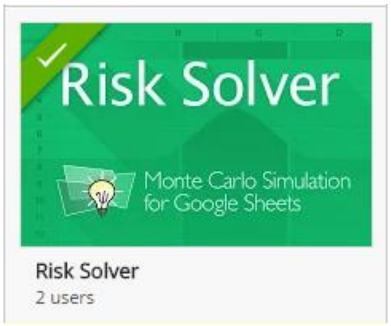
Double click YieldManagementModel1.xlsx, the file opens in Google Drive Preview Viewer (shown below). Click Open – Google Sheets.



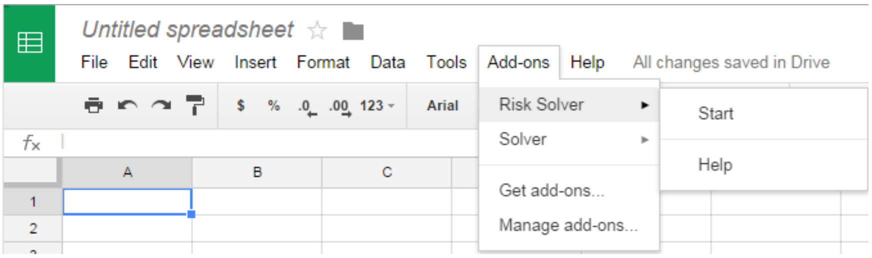
To add the Solver Add-on, choose the menu option Add-ons - Get Add-ons...


Add-ons
Help
Last edit was seconds ago
Do more with Sheets
Looking for even more Sheets features? Get a few add-ons.
Get add-ons...
Manage add-ons...

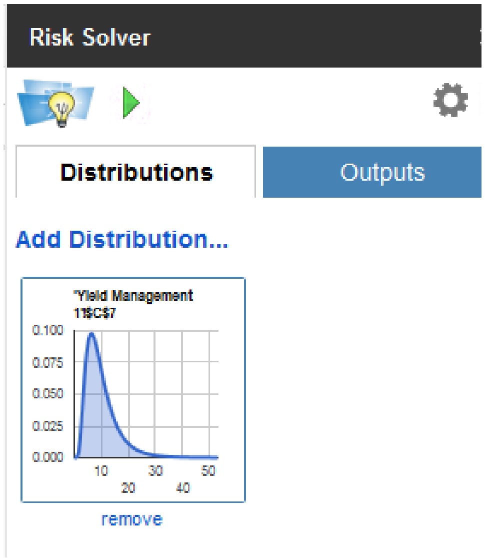
In the Add-ons dialog, scroll until you see Risk Solver. Click to select and install.



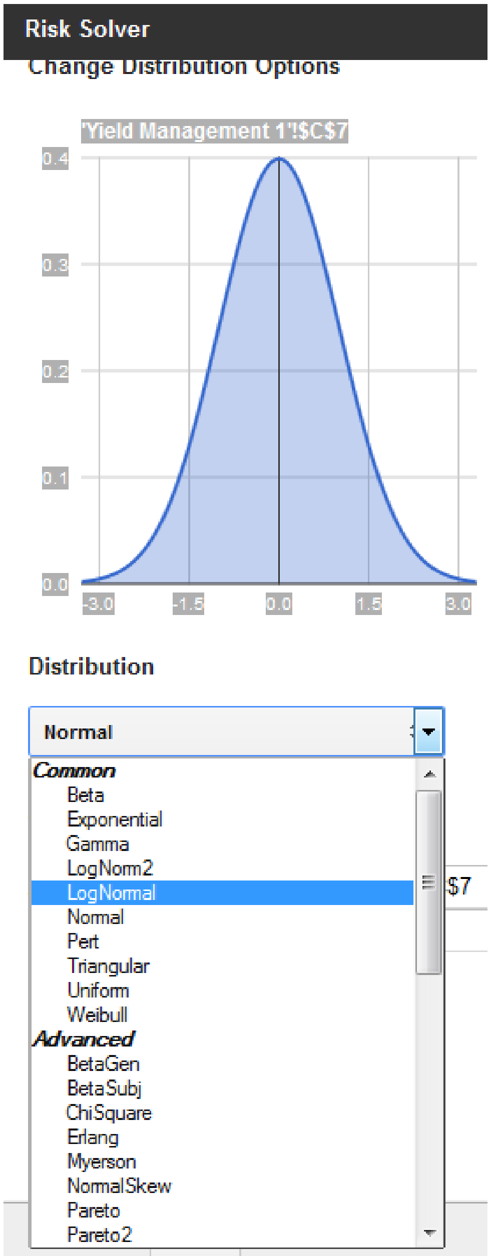
Once these steps are finished, Risk Solver should now appear on the Add-ons menu.



Choose Add-ons - Risk Solver - Start. The Risk Solver Task Pane (Google calls this a Sidebar) will appear. The uncertain variable that appears in cell C7 is displayed in this tab.



Click the distribution to change the distribution type and parameters or to view the cell formula and address.



Risk Solver

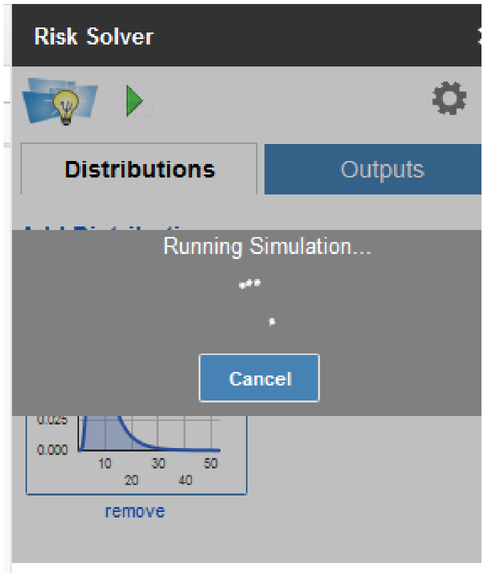
Change Distribution Options

Distribution

Normal

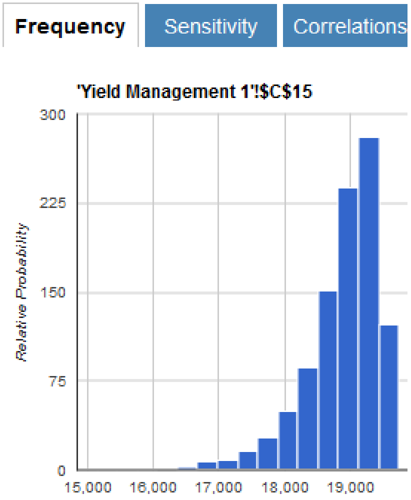
Click Cancel to leave everything “as is” or OK for Risk Solver to accept any changes. For this example, we’ll click the Cancel button to leave the Distribution as PsiLogNormal.

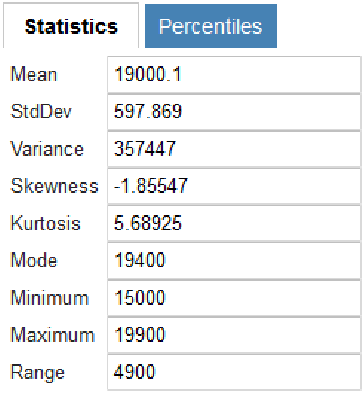
Click the green arrow at the top of the sidebar to run a simulation. (Click the Cancel button to stop the simulation before finishing.)



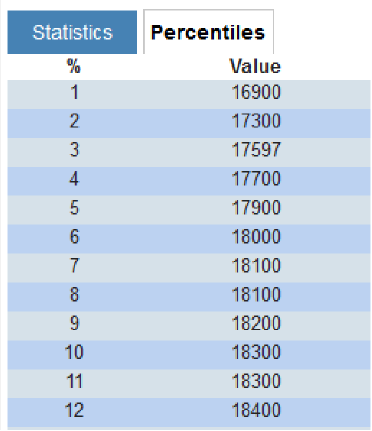
Running Simulation...

In the sidebar, the Output tab appears with a small frequency chart containing the uncertain function in cell C15. Click the chart to enlarge it. Click the Frequency tab to view statistics (mean, standard deviation, variance, skewness, kurtosis, mode, minimum, maximum, and range) or Percentiles (1 – 100).





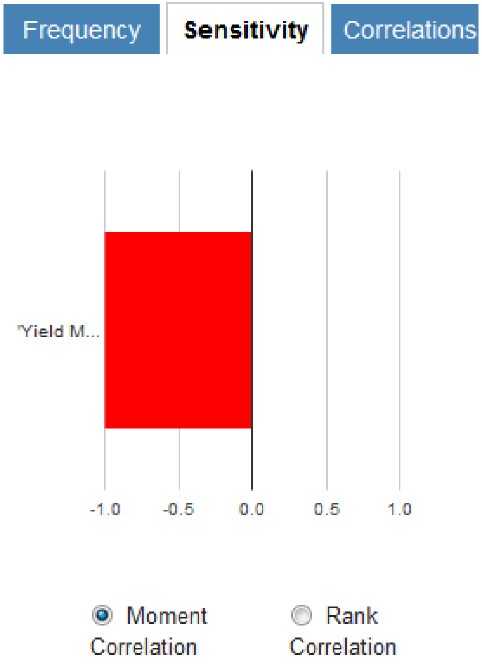
Mean	19000.1
StdDev	597.869
Variance	357447
Skewness	-1.85547
Kurtosis	5.68925
Mode	19400
Minimum	15000
Maximum	19900
Range	4900



%	Value
1	16900
2	17300
3	17597
4	17700
5	17900
6	18000
7	18100
8	18100
9	18200
10	18300
11	18300
12	18400

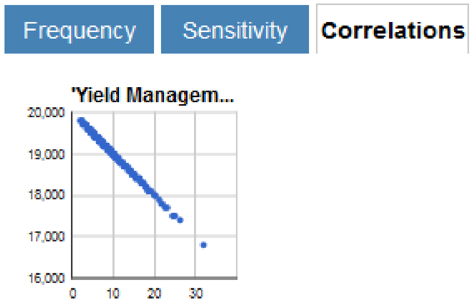
For more information on the statistics, please see the corresponding Psi function (PsiMean, PsiMean, PsiStdDev (Standard Deviation), PsiVariance, PsiSkewness, PsiKurtosis, PsiMode, PsiMin, PsiMax or PsiRange) in *the Psi Function Reference* chapter in *Frontline Solvers Reference Guide*.

Click the Sensitivity tab to view a “Tornado chart” that shows you how much Total Revenue changes with a change in the uncertain variable in cell C7. In this model there is only one uncertain variable. This chart shows a negative correlation between the Number of No-Shows and Total Revenue. For more information on this tab, please see the section, “A First Simulation Example” above.

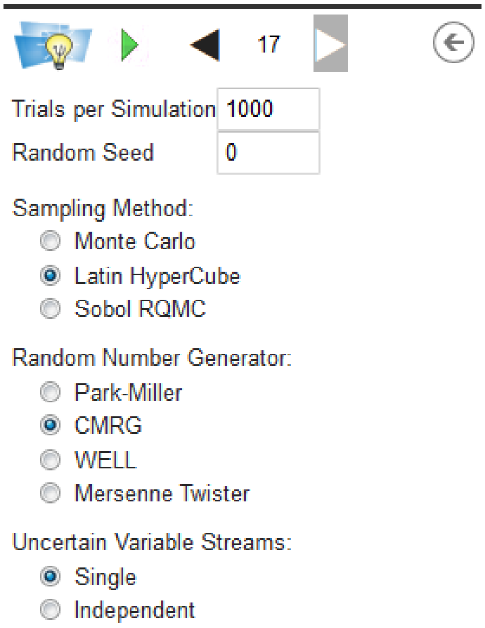


● Moment Correlation ○ Rank Correlation

Select the Correlations tab to view the same negative correlation between the uncertain variable in cell C7 and the uncertain function in cell C15. For more information on this tab, please see the section above.



Click the ← icon to go back to the Output tab or click the ◀ 1 ▶ left and right arrow icons to increment through each of the simulation trials. (The default is 1000.) To change the number of trials per simulation or other option settings, click the ⚙ icon in the upper right corner of the Sidebar. For more information on these options, please see the chapter *Platform Solver Options* in the *Frontline Solvers Reference Guide*.



Trials per Simulation 1000

Random Seed 0

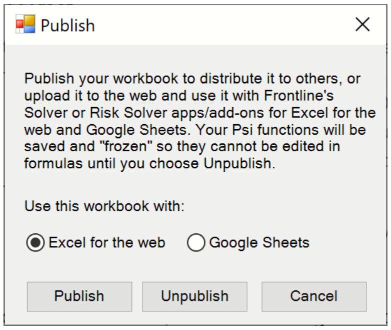
Sampling Method:
- ○ Monte Carlo
- ● Latin HyperCube
- ○ Sobol RQMC

Random Number Generator:
- ○ Park-Miller
- ● CMRG
- ○ WELL
- ○ Mersenne Twister

Uncertain Variable Streams:
- ● Single
- ○ Independent

Now let's Publish this same model to Excel Online. Let's go back to Excel and YieldManagement1.xlsx. Afterwards, click **Publish – Unpublish**. Immediately, all PsiFunctions are restored in cells C7 and C16.

Click **Publish – Publish** again, but this time select **Excel Online**, then click **Publish.**



Publish your workbook to distribute it to others, or upload it to the web and use it with Frontline's Solver or Risk Solver apps/add-ons for Excel for the web and Google Sheets. Your Psi functions will be saved and "frozen" so they cannot be edited in formulas until you choose Unpublish.

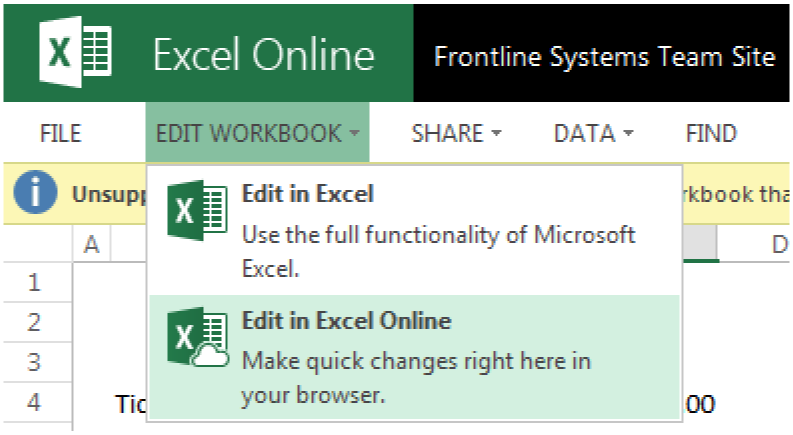
Use this workbook with:

● Excel for the web ○ Google Sheets

Click **File – Save As** and save this file as **YieldManagement1.xlsx** on your *SharePoint* or *Office 365* site.

Note: In order to use the Risk Solver App, your file must be saved on SharePoint or Office 365 site. This app cannot solve models saved on your local hard drive.

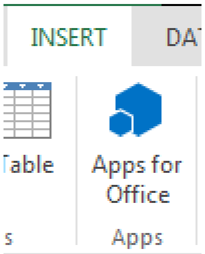
Open the workbook using Excel Online, then click **Edit Workbook – Edit in Excel Online**.



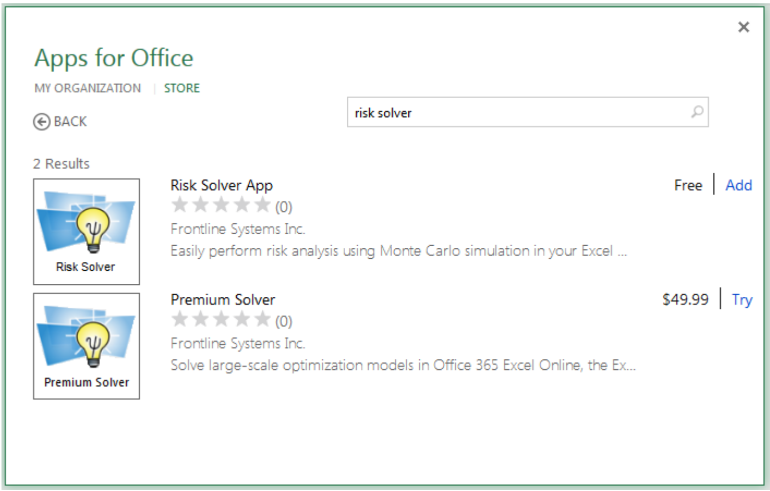
Edit in Excel
Use the full functionality of Microsoft Excel.

Edit in Excel Online
Make quick changes right here in your browser.

Click Insert – Apps for Office.



Select the Store tab and search for Risk Solver, then select (1st app in the list) and Install.

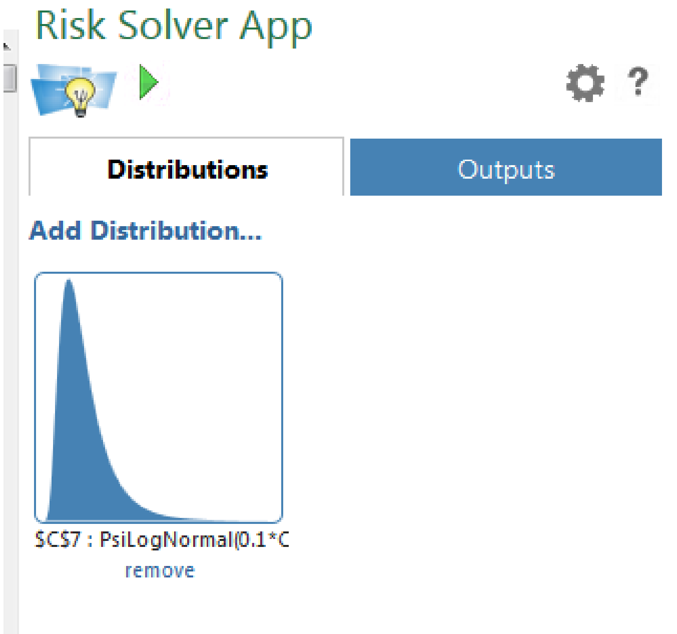


2 Results

Risk Solver App (0) Free | Add
Frontline Systems Inc.
Easily perform risk analysis using Monte Carlo simulation in your Excel ...

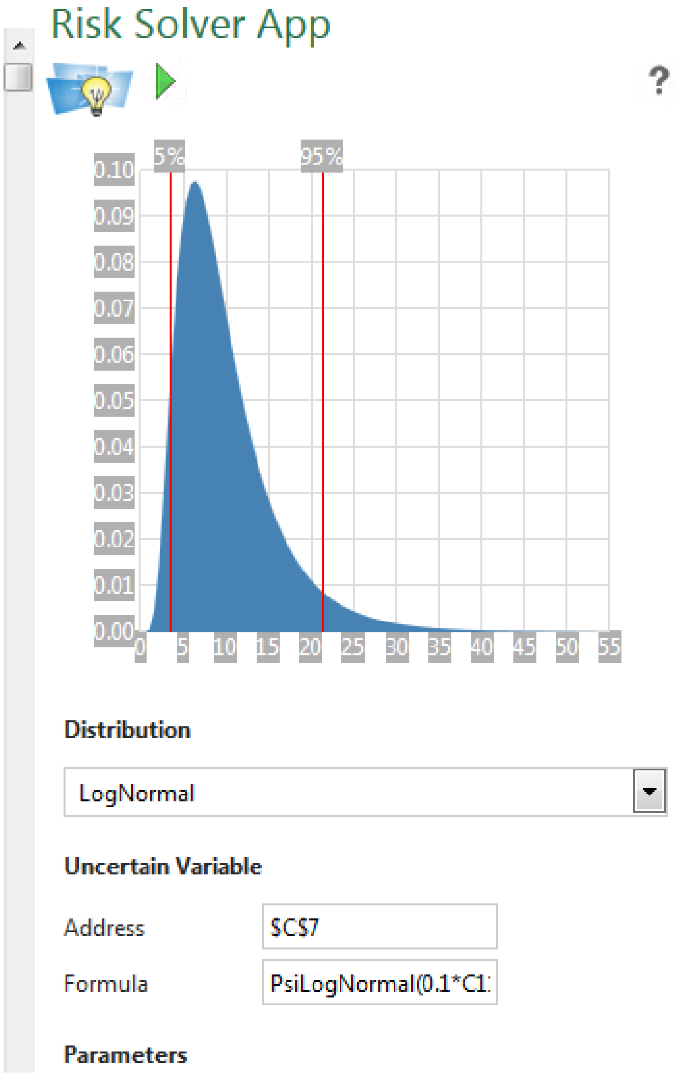
Premium Solver (0) $49.99 | Try
Frontline Systems Inc.
Solve large-scale optimization models in Office 365 Excel Online, the Ex...

The Risk Solver App task pane opens on the right of your screen.



The uncertain variable that appears in cell C7 is displayed in this tab. Click the distribution to change the distribution type and parameters or to view the cell formula and address as shown in the Google Sheets example above.

Note: In the Solver App red vertical lines have been drawn at the 5% and 95% percentiles.



Distribution

LogNormal

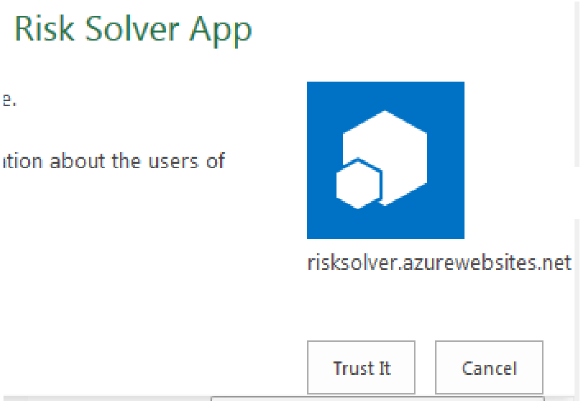
Uncertain Variable

Address $C$7

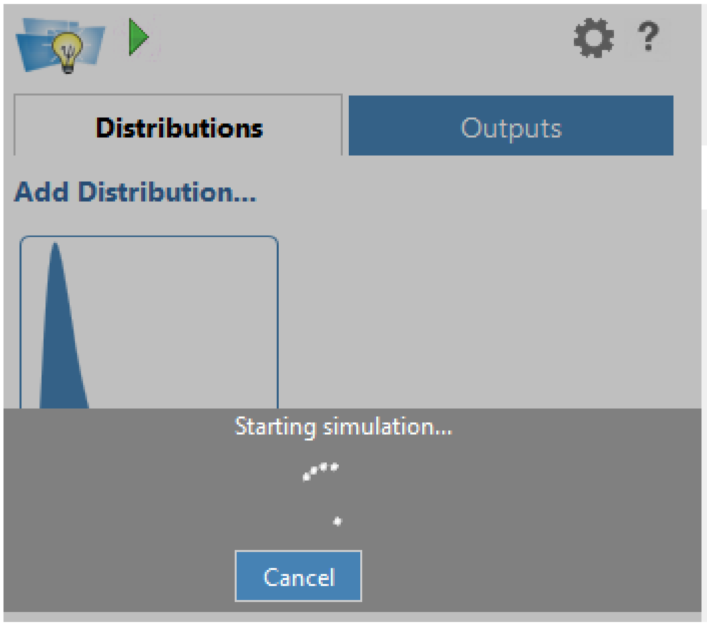
Formula PsiLogNormal(0.1*C1

Parameters

Click the green arrow at the top of the task pane to run a simulation. (Click the Cancel button to stop the simulation before finishing.) If running for the first time, you will be asked to "Trust" the app.



To cancel the simulation before completion, click Cancel.



Starting simulation...

In the sidebar, the Output tab appears with a small frequency chart containing the uncertain function in cell C15. Click the chart to enlarge it. Click the Frequency tab to view statistics (mean, standard deviation, variance, skewness, kurtosis, mode, minimum, maximum, and range) or Percentiles (1 – 100). Click the Sensitivity tab to view a "Tornado chart" which illustrates the negative correlation between the Number of no-shows (uncertain variables) and Total Revenue (uncertain function) and the Correlations tab to view the relationship between the uncertain variable in cell C7 and the uncertain function in cell C15 as a scatter chart. For more information on either of these tabs, please see the Google Sheets example above.

Click the ← icon to go back to the Output tab or click the ◀ 1 ▶ left and right arrow icons to increment through each of the simulation trials. (The default is 1000.) To change the number of trials per simulation or other option settings, click the ⚙ icon in the upper right corner of the Task Pane. For more information on these options, please see the chapter *Platform Solver Options* in the *Frontline Solvers Reference Guide*.

Examples: Stochastic Optimization

Introduction

This chapter introduces stochastic optimization in Analytic Solver Comprehensive (Desktop or Cloud) with a series of examples.

We use the term stochastic optimization to mean optimization of models that include *uncertainty*, using any solution method. Analytic Solver offers an exceptional level of power to find robust optimal solutions to models with uncertainty, using three different solution methods:

- Simulation optimization
- Stochastic programming
- Robust optimization

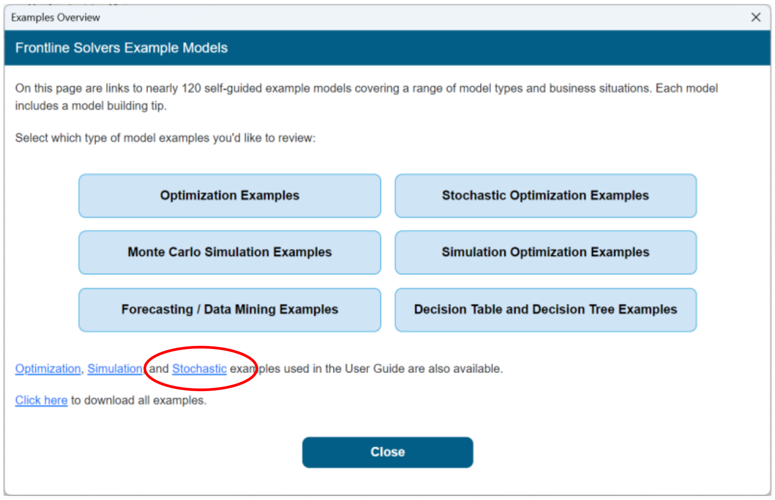
The first method, **simulation optimization**, is also available in Analytic Solver Optimization with Analytic Solver Simulation or Analytic Solver Upgrade (Desktop or Cloud), and is the only method in other software products such as Crystal Ball Professional and @RISK Industrial. It handles very general models, but it is not scalable to large models (with thousands of variables and constraints), and it doesn't support the important modeling concept of recourse decisions.

Stochastic programming and **robust optimization** can be applied only to linear and quadratic programming models with uncertainty, but they are scalable to large models. Either Analytic Solver Comprehensive, Analytic Solver Simulation with Analytic Solver Optimization or Analytic Solver Upgrade (Desktop or Cloud) are required to use these methods.

A Project Selection Model

To open the examples, click **Help – Examples** on the ribbon. Click the Stochastic link on the Overview tab to open the workbook StochasticExamples.xlsx which includes three worksheets.

Click the Project Selection tab within the StochasticExamples.xlsx workbook to open the first example, which may surprise you: It is a capital budgeting problem, where the projects being considered for funding have uncertain future cash flows. Models similar to **Project Selection** have been used with other software products for many years to illustrate how **simulation optimization** can be used to optimize models with uncertainty. We'll show how to do this in Analytic Solver Comprehensive, Analytic Solver Simulation with Analytic Solver Optimization or Analytic Solver Upgrade, using simulation optimization and the Evolutionary Solver – with much better performance. But when we use Analytic Solver to *analyze the structure* of this model, we'll see that this model *doesn't require simulation optimization at all.* We can solve this model in a fraction of a second – using the LP/Quadratic Solver!



Examples Overview
Frontline Solvers Example Models

On this page are links to nearly 120 self-guided example models covering a range of model types and business situations. Each model includes a model building tip.

Select which type of model examples you'd like to review:

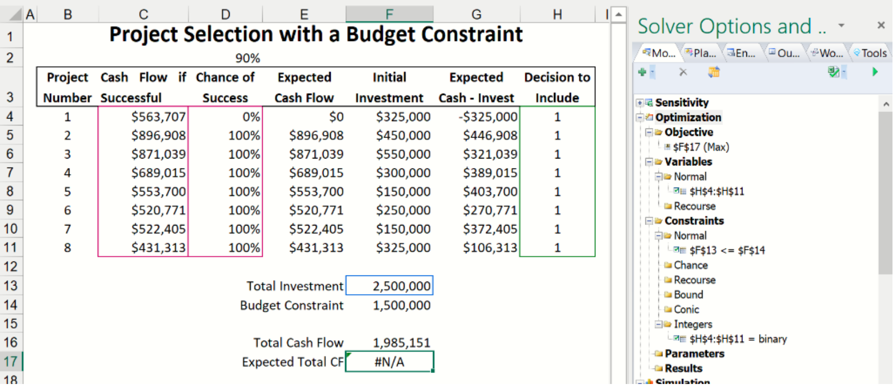
Optimization Examples
Stochastic Optimization Examples
Monte Carlo Simulation Examples
Simulation Optimization Examples
Forecasting / Data Mining Examples
Decision Table and Decision Tree Examples

Optimization, Simulation and Stochastic examples used in the User Guide are also available.

Click here to download all examples.

In **Project Selection**, eight different capital projects are proposed for funding. Each one has a known initial investment. Each project has a 90% chance of success, and if a project succeeds, it will have an uncertain (but positive) future cash flow. Funding all eight projects would require a total initial investment of $2.5 million, but our capital budget is only $1.5 million. Hence, we must choose a subset of the projects to fund that will maximize our expected total future cash flow, while ensuring that our total initial investment doesn't exceed our $1.5 million budget.

Click the tab for the Projection Selection model, shown below.



Project Selection with a Budget Constraint

Project Number	Cash Flow if Successful	Chance of Success	Expected Cash Flow	Initial Investment	Expected Cash - Invest	Decision to Include
		90%				
1	$563,707	0%	$0	$325,000	-$325,000	1
2	$896,908	100%	$896,908	$450,000	$446,908	1
3	$871,039	100%	$871,039	$550,000	$321,039	1
4	$689,015	100%	$689,015	$300,000	$389,015	1
5	$553,700	100%	$553,700	$150,000	$403,700	1
6	$520,771	100%	$520,771	$250,000	$270,771	1
7	$522,405	100%	$522,405	$150,000	$372,405	1
8	$431,313	100%	$431,313	$325,000	$106,313	1

Total Investment 2,500,000
Budget Constraint 1,500,000

Total Cash Flow 1,985,151
Expected Total CF #N/A

- Sensitivity
- Optimization
  - Objective
    - $F$17 (Max)
  - Variables
    - Normal
      - $H$4:$H$11
    - Recourse
  - Constraints
    - Normal
      - $F$13 <= $F$14
    - Chance
    - Recourse
    - Bound
    - Conic
    - Integers
      - $H$4:$H$11 = binary
  - Parameters
  - Results
- Simulation

To model the uncertainty in this problem, we use PSI functions that define *uncertain variables* with certain probability distributions. We can create these functions via the Distributions dropdown gallery and the Uncertain Variable dialog, as shown in the last chapter, or we can just type in the formulas.

To model the "Cash Flow if Successful" in column C, we use the **PsiTriangular()** function, specifying a minimum, most likely, and maximum cash flow. For example, the formula in C4 is =PsiTriangular(400000, 500000, 900000).

To model the "Chance of Success" in column D, we use **=PsiBinomial(1,D2) where D2=.9**. On each trial, this distribution returns 1 with probability 90% and 0 with probability 10%. In column E we multiply the "Cash Flow if Successful" and the "Chance of Success" to obtain the "Expected Cash Flow."

Column G computes the *net* cash flow – the "Expected Cash Flow" in column E minus the "Initial Investment" in column F. Column H holds our decision variables, which are constrained to be binary integer (0 or 1 at the solution).

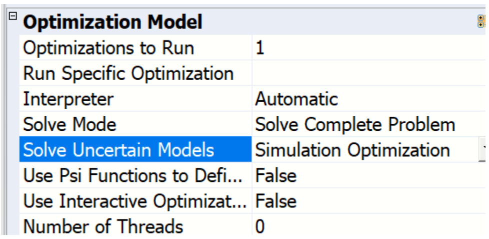
Cell F13 computes the total initial investment as the SUMPRODUCT of the project initial investments (column F) and the 0-1 variables in column H. In the model, F13 is constrained to be less than the $1.5 million budget in F14.

Cell F16 computes the total cash flow as the SUMPRODUCT of the project net cash flows and the 0-1 variables in column H. Cell F17 – unique to a model with uncertainty – contains =PsiMean(F16): It computes the mean (*expected*) value of total net cash flow. This is the objective function we want to maximize.

We're asking Analytic Solver to find the best combination of projects – by finding 0 or 1 values for the selection variables at H4:H11 – to maximize cell F17, the expected value of total net cash flow, subject to the constraint that the total initial investment doesn't exceed our $1.5 million budget.

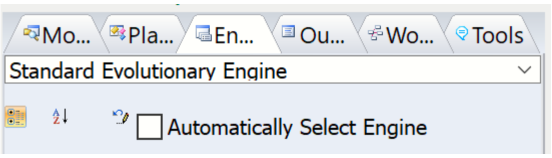
Solving with Simulation Optimization

We'll first solve this problem using **simulation optimization**. To do this, go to the Task Pane Platform tab, and confirm the **Solve Uncertain Models** option is set to **Simulation Optimization**. (We'll see later what happens when we allow Analytic Solver to choose the solution method automatically.)



Optimization Model	
Optimizations to Run	1
Run Specific Optimization	
Interpreter	Automatic
Solve Mode	Solve Complete Problem
Solve Uncertain Models	Simulation Optimization
Use Psi Functions to Defi...	False
Use Interactive Optimizat...	False
Number of Threads	0

On the Engine tab, and confirm that the **Evolutionary Solver** is selected.



Standard Evolutionary Engine

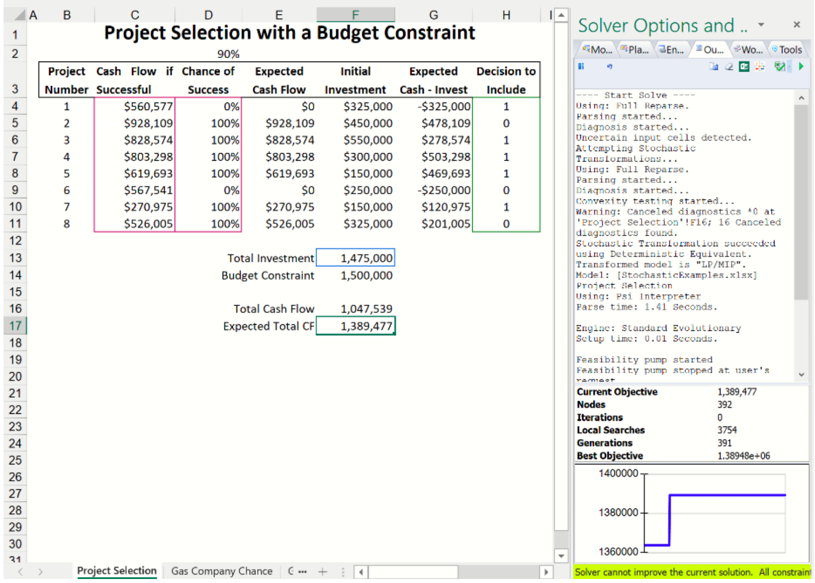
☐ Automatically Select Engine

Why do we choose these settings? In the case of other software products, it's because simulation optimization is the only technology they have – and "if your only tool is a hammer, every problem looks like a nail." Simulation optimization is also a very general approach that can handle nonlinear and non-smooth functions, and it's comparatively easy to understand.

The idea behind simulation optimization is straightforward: **For each** set of values for the decision variables considered by the optimizer, we perform **one simulation**, a compute user-specified *summary measures* – such as =PsiMean(F16) in the Project Selection model – for the constraints and/or objective that depend on uncertainty. The optimizer uses these summary measures to decide what set of values it should try *next* for the decision variables – and the process is repeated.

The great strength of simulation optimization is its **generality** – but this is also its weakness: It requires a new simulation **at each step** of the optimization, and because the method assumes no structure in the model, in general the **number of steps** can grow **exponentially** with the number of variables and constraints.

Now click the green arrow in the Task Pane to run a simulation. If you are using Analytic Solver Desktop, after a second or two (much faster than other software products), you can press ESC to stop the Evolutionary Solver and display the best solution found so far – as shown on the next page.



Project Selection with a Budget Constraint

90%

Project Number	Cash Flow if Successful	Chance of Success	Expected Cash Flow	Initial Investment	Expected Cash - Invest	Decision to Include
1	$560,577	0%	$0	$325,000	-$325,000	1
2	$928,109	100%	$928,109	$450,000	$478,109	0
3	$828,574	100%	$828,574	$550,000	$278,574	1
4	$803,298	100%	$803,298	$300,000	$503,298	1
5	$619,693	100%	$619,693	$150,000	$469,693	1
6	$567,541	0%	$0	$250,000	-$250,000	0
7	$270,975	100%	$270,975	$150,000	$120,975	1
8	$526,005	100%	$526,005	$325,000	$201,005	0

Total Investment 1,475,000
Budget Constraint 1,500,000

Total Cash Flow 1,047,539
Expected Total CF 1,389,477

```
---- Start Solve ----
Using: Full Reparse.
Parsing started...
Diagnosis started...
Uncertain input cells detected.
Attempting Stochastic
Transformations...
Using: Full Reparse.
Parsing started...
Diagnosis started...
Convexity testing started...
Warning: Canceled diagnostics *0 at
'Project Selection'!F16; 16 Canceled
diagnostics found.
Stochastic Transformation succeeded
using Deterministic Equivalent.
Transformed model is "LP/MIP".
Model: [StochasticExamples.xlsx]
Project Selection
Using: Psi Interpreter
Parse time: 1.41 Seconds.

Engine: Standard Evolutionary
Setup time: 0.01 Seconds.

Feasibility pump started
Feasibility pump stopped at user's
```

Current Objective	1,389,477
Nodes	392
Iterations	0
Local Searches	3754
Generations	391
Best Objective	1.38948e+06

Solver cannot improve the current solution. All constraint

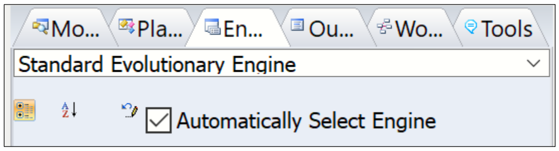
We've decided to fund projects #1, 3, 4, 5, and 7 for an expected total net cash flow of $1,389,477. That's a great result. But – *how much work* did we do to solve this model, and did we *need* to do that much work? It might not matter for a small model like this one – but for a 'scaled-up' model that might involve hundreds or thousands of projects, the solution might have taken far more time.

Solving Automatically

Let's allow Analytic Solver Comprehensive (or Analytic Solver Simulation with Analytic Solver Optimization) to choose the solution method and the Solver Engine automatically. To do this, go to the Task Pane Platform tab, and change the **Solve Uncertain Models** option to **Automatic**.

On the Engine tab, check the box to automatically select a Solver Engine in Analytic Solver Desktop or select Automatically select engine at the top of the menu for Analytic Solver Cloud.

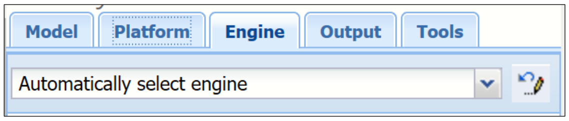
Analytic Solver Desktop



Standard Evolutionary Engine

☑ Automatically Select Engine

Analytic Solver Cloud



Automatically select engine

Now click the green arrow on the Task Pane. In a *fraction of a second*, a solution appears with the *same* objective value of $1,389,477 and the same projects selected, with the message “Solver found a solution. All constraints and optimality conditions are satisfied.” This means that Analytic Solver found a *proven* globally optimal solution – whereas with simulation optimization, we never know whether the solution we found is optimal. How did this happen?

If you examine the solution log, you’ll see these messages:

```
---- Start Solve ----
Using: Full Reparse.
Parsing started...
Diagnosis started...
Warning: Canceled diagnostics *0 at 'Project Selection'!F16;
6 Canceled diagnostics found.
Uncertain input cells detected.
Attempting Stochastic Transformations...
Using: Full Reparse.
Parsing started...
Diagnosis started...
Convexity testing started...
Warning: Canceled diagnostics *0 at 'Project Selection'!F16;
22 Canceled diagnostics found.
Stochastic Transformation succeeded using Deterministic
Equivalent.
Transformed model is "LP/MIP".
Automatic engine selection: Standard LP/Quadratic
Model: [StochasticExamples.xlsx]Project Selection
Using: Psi Interpreter
Parse time: 0.66 Seconds.

Engine: Standard LP/Quadratic
Setup time: 0.01 Seconds.

Engine Solve time: 0.33 Seconds.

Solver found a solution.  All constraints and optimality
conditions are satisfied.
Solve time: 1.56 Seconds.
```

Analytic Solver Comprehensive analyzed the model and diagnosed it as a Stochastic LP (linear programming) problem, with 8 variables, 2 functions (objective and budget constraint), and 16 uncertainties (the PsiTriangular() and PsiBinomial() functions) that affect the objective. As you’ll see in the chapter “Mastering Simulation and Risk Analysis Topics,” Analytic Solver can solve a Stochastic LP using two *other* methods besides simulation optimization – stochastic programming and robust optimization. For this model, it chose a transformation to Stochastic Programming Deterministic Equivalent form.

But this model is so simple that Analytic Solver determined that *no* transformations were needed to solve the problem. Despite the presence of uncertainty, this model *doesn’t require simulation optimization at all*. There was *no need* to run hundreds of Monte Carlo simulations (or many more, with some alternative software products). It was sufficient to run *one* Monte Carlo simulation at the beginning, to compute the *constant* linear coefficients of the objective function. Project Selection is simply an LP/MIP (linear mixed-integer programming) model!

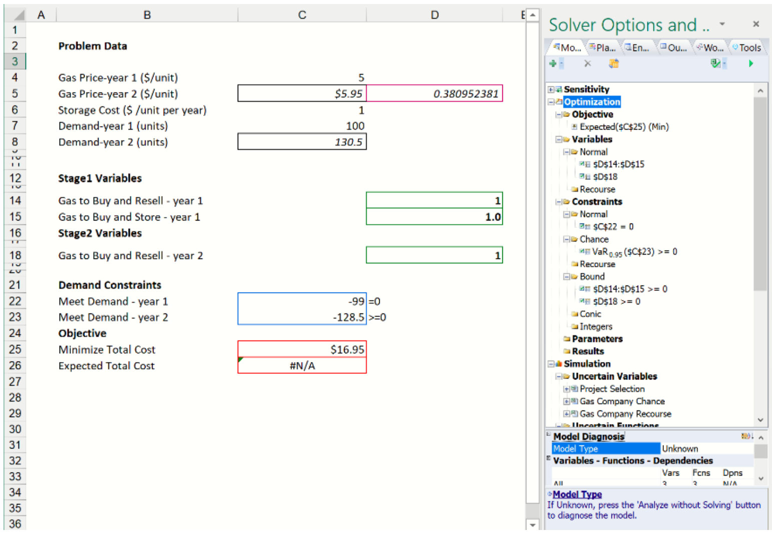
In this small model, we found the same solution within seconds using both methods. But in a ‘scaled-up’ model in a large company, with hundreds or even thousands of projects, Analytic Solver’s ability to *analyze the structure* of the model could make all the difference – between a solution of unknown quality, found after a long wait using the *wrong approach* – and a proven optimal solution, found within seconds using the *right approach.*

A Model with Chance Constraints

The worksheets Gas Company Chance and Gas Company Recourse in **StochasticExamples.xlsx** are related – they both describe the same problem, but with different assumptions about *when* a certain decision must be made. With Gas Company Chance, we'll show how a problem with uncertainty can be formulated with a *chance constraint* and solved using robust optimization methods. With Gas Company Recourse, we'll show how to use a "wait and see" or **recourse decision** – a capability *not available at all* with simulation optimization – to obtain a *better* optimal solution.

We hope these examples will motivate you to read the chapter "Mastering Simulation and Risk Analysis Topics," which will give you a *unified view* of the crucial characteristics of models with uncertainty, and the options of solving them with simulation optimization, stochastic programming, and robust optimization methods.

Click the **Gas Company Chance** tab to display the model, which is pictured on the next page.



	A	B	C	D
1				
2		**Problem Data**		
3				
4		Gas Price-year 1 ($/unit)	5	
5		Gas Price-year 2 ($/unit)	*$5.95*	*0.380952381*
6		Storage Cost ($ /unit per year)	1	
7		Demand-year 1 (units)	100	
8		Demand-year 2 (units)	*130.5*	
12		**Stage1 Variables**		
14		Gas to Buy and Resell - year 1		1
15		Gas to Buy and Store - year 1		**1.0**
16		**Stage2 Variables**		
18		Gas to Buy and Resell - year 2		1
21		**Demand Constraints**		
22		Meet Demand - year 1	-99	=0
23		Meet Demand - year 2	-128.5	>=0
24		**Objective**		
25		Minimize Total Cost	$16.95	
26		Expected Total Cost	#N/A	

In this model, a gas company purchases natural gas from suppliers and resells it to consumers in its service area. Consumer demand is known to be 100 units this year (cell C7) but is uncertain for next year (cell C8); the company must meet this demand with high probability, or it will encounter public relations and regulatory problems. The price of gas is known to be $5.00/unit this year (cell C4), but is uncertain for next year (cell C5): If the weather is very cold, demand may reach 180 units, and the price may reach $7.50/unit, but if the weather is warm, demand may remain at 100 units, and the price may remain at $5.00/unit. The company's goal is to minimize its total cost of purchased and stored gas (cell C25). It's decisions are:

- How much gas to purchase and resell to consumers *this year* (cell D14)
- How much gas to purchase *this year* and store (cell D15) , at a cost of $1 per unit, for use *next year*

- How much to purchase and resell to consumers *next year* (cell D18)

In Gas Company Chance, we assume that the gas company must commit to all of its gas purchases this year, *before* next year's demand is known: All three decision variables D14, D15 and D18 are normal or 'here and now' decisions. (In the next section, we'll consider Gas Company Recourse, where we assume that next year's gas purchases can be determined on a 'wait and see' basis.)

Gas Company Chance has a constraint that the company must have enough gas to meet next year's *uncertain* demand "with high probability." We'll model this in Analytic Solver Comprehensive with a ***chance constraint***. Our objective function – total cost, to be minimized – depends in part on next year's uncertain gas price. We will therefore seek to minimize the ***expected value* of total cost**.

The **objective** appears as "Expected($C$25) (Min)" – in Gas Company Chance, we used the Ribbon to define the objective interactively, rather than a PsiMean() function on the worksheet, as we did earlier in Project Selection. The decision variables are cells D14, D15 and D18, as explained earlier; each of these is constrained to be non-negative, as shown in the Bound section of the outline.

At C22 we calculate =D14-C7 (this year's gas purchases minus this year's demand), and at C23 we calculate =D15+D18-C8 (gas purchased and stored this year, plus gas purchased next year, minus next year's uncertain demand).

The **normal constraint** $C$22 = 0 specifies that we must meet this year's consumer demand exactly (forcing D14 to 100). The **chance constraint** **VaR**$_{0.95}$**($C$23) >= 0** requires that the 95th percentile of C23, over *all* realizations of the uncertainty, must be non-negative – in other words, we must meet demand with 95% probability.

We *could* express this chance constraint by placing =PsiPercentile(D15+D18-C8,0.95) in cell C23, and using the Ribbon to define a *normal* constraint C23 >= 0. But while PsiPercentile and other PSI Statistics functions are acceptable for simulation optimization, the transformation for *robust optimization* requires that we express the chance property directly in the constraint itself, not in a formula.

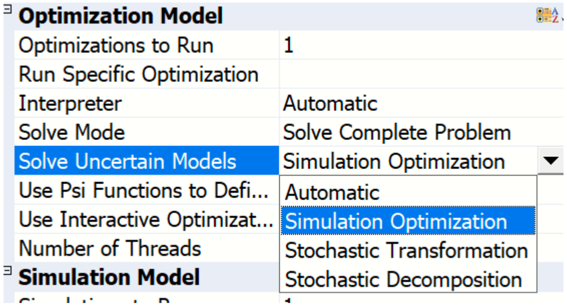
Solving with Robust Optimization

We could solve this model using *simulation optimization*, as we did for Project Selection in the previous section. But here, we want to show how Analytic Solver Comprehensive (or Analytic Solver Simulation with Analytic Solver Optimization) can automatically *transform* this stochastic linear programming problem – with its expected value objective and chance constraint – into a larger conventional optimization model, using the methods of ***robust optimization***.

The idea behind robust optimization is to *transform* an optimization problem with known *structure* – such as a stochastic linear programming problem – into a larger, *conventional* 'robust counterpart' problem that accounts for the impact of *bounded* uncertainty on the constraints and objective. Analytic Solver performs **one simulation** at the outset, to assess the impact of uncertainty and construct the robust counterpart problem. Solving the robust counterpart – a **single LP or SOCP problem** – gives us an approximate solution to the original stochastic problem.

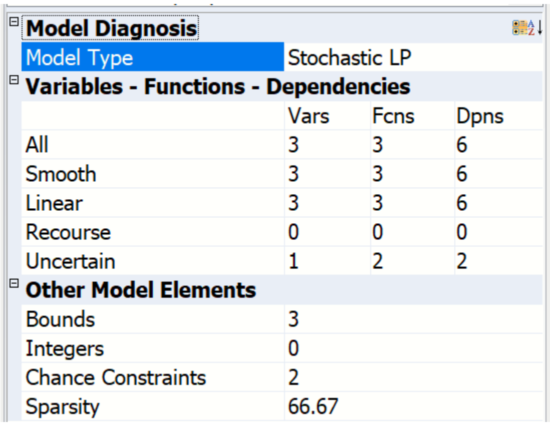
The strength of robust optimization is its **scalability** to very large problems – since only one simulation and one optimization of an LP or SOCP problem is required. But robust optimization cannot solve more general nonlinear, non-convex problems.

As saved, the model is ready to be solved using robust optimization. But to see the difference between the original model – a stochastic linear programming model – and the robust counterpart model that is the result of the transformation, we'll first set the **Solve Uncertain Models** option to **Simulation Optimization**:



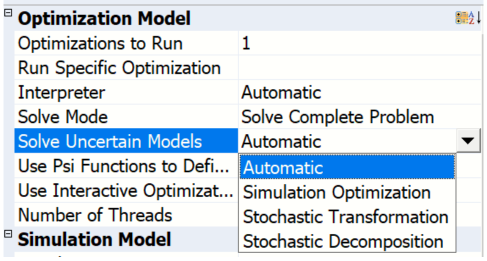
Optimization Model	
Optimizations to Run	1
Run Specific Optimization	
Interpreter	Automatic
Solve Mode	Solve Complete Problem
Solve Uncertain Models	Simulation Optimization
Use Psi Functions to Defi...	
Use Interactive Optimizat...	
Number of Threads	
Simulation Model	

and then click **Analyze – Analyze Transformed Model** (on the Model tab) to analyze the structure of the model. The Model Diagnosis section of the Task Pane Model tab "pops up" to show the results, as shown below.



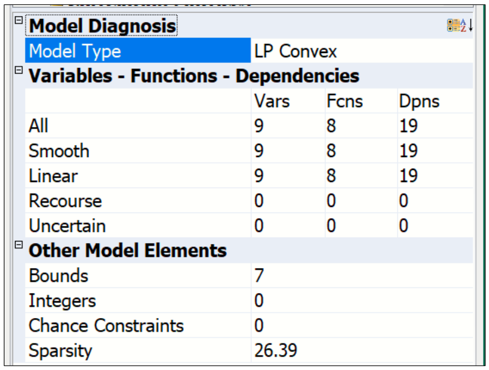
Model Diagnosis			
Model Type	Stochastic LP		
Variables - Functions - Dependencies			
	Vars	Fcns	Dpns
All	3	3	6
Smooth	3	3	6
Linear	3	3	6
Recourse	0	0	0
Uncertain	1	2	2
Other Model Elements			
Bounds	3		
Integers	0		
Chance Constraints	2		
Sparsity	66.67		

The model is diagnosed as a Stochastic LP, with 3 variables, 3 functions, and 1 uncertainty (the Beta distribution at cell D5, which models the weather). Now, we'll set the **Solve Uncertain Models** option to **Automatic**:



Optimization Model	
Optimizations to Run	1
Run Specific Optimization	
Interpreter	Automatic
Solve Mode	Solve Complete Problem
Solve Uncertain Models	Automatic
Use Psi Functions to Defi...	
Use Interactive Optimizat...	
Number of Threads	
Simulation Model	

and click the **Analyze – Analyze Transformed model** again – this will analyze the structure of the *transformed* robust counterpart model.

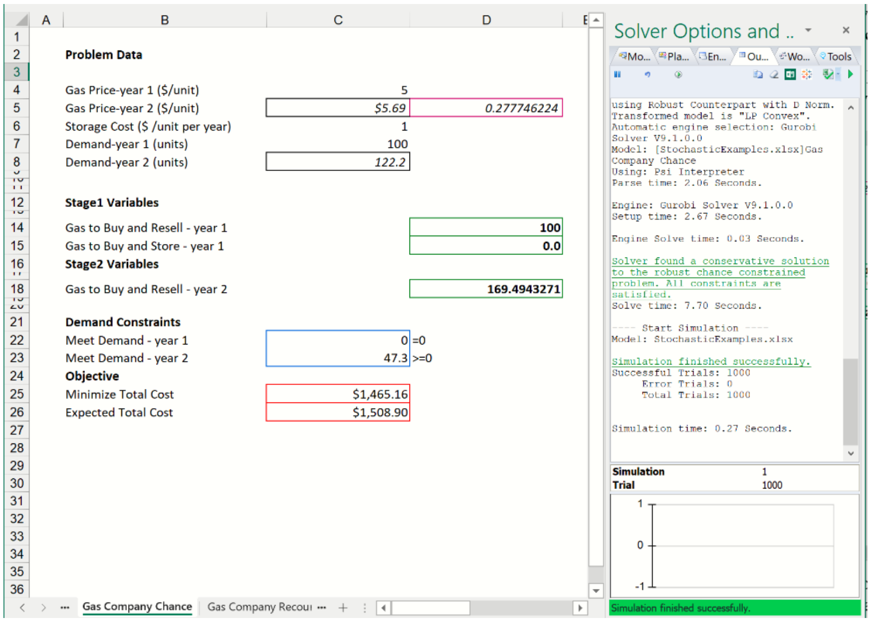


Model Diagnosis			
Model Type	LP Convex		
Variables - Functions - Dependencies			
	Vars	Fcns	Dpns
All	9	8	19
Smooth	9	8	19
Linear	9	8	19
Recourse	0	0	0
Uncertain	0	0	0
Other Model Elements			
Bounds	7		
Integers	0		
Chance Constraints	0		
Sparsity	26.39		

The *transformed* model has 9 variables and 8 functions (instead of 3 and 3), but it has *0 uncertainties* – it's a conventional linear programming model. Solving this robust counterpart model will give us an *approximate* (and somewhat conservative) solution to the original stochastic LP problem.

Technical note, explained in the chapter "Mastering Simulation and Risk Analysis Topics:" The robust counterpart model is an LP because the Platform tab Chance Constraints Use option is set to D Norm. If it had been set to L2 Norm, the robust counterpart model would have been an SOCP.

Now click the green arrow on the Task Pane. In a fraction of a second, the solution appears, with the message "Solver found a conservative solution to the robust chance constrained problem. All constraints are satisfied." You can **click this message to display Help** with a more complete explanation of what this means. To resolve the formula in cell C26, click the Simulate icon to run a simulation. The results are shown below.



	A	B	C	D
1				
2		**Problem Data**		
3				
4		Gas Price-year 1 ($/unit)	5	
5		Gas Price-year 2 ($/unit)	*$5.69*	*0.277746224*
6		Storage Cost ($ /unit per year)	1	
7		Demand-year 1 (units)	100	
8		Demand-year 2 (units)	*122.2*	
12		**Stage1 Variables**		
14		Gas to Buy and Resell - year 1		**100**
15		Gas to Buy and Store - year 1		**0.0**
16		**Stage2 Variables**		
18		Gas to Buy and Resell - year 2		**169.4943271**
21		**Demand Constraints**		
22		Meet Demand - year 1	0	=0
23		Meet Demand - year 2	47.3	>=0
24		**Objective**		
25		Minimize Total Cost	$1,465.16	
26		Expected Total Cost	$1,508.90	

```
using Robust Counterpart with D Norm.
Transformed model is "LP Convex".
Automatic engine selection: Gurobi
Solver V9.1.0.0
Model: [StochasticExamples.xlsx]Gas
Company Chance
Using: Psi Interpreter
Parse time: 2.06 Seconds.

Engine: Gurobi Solver V9.1.0.0
Setup time: 2.67 Seconds.

Engine Solve time: 0.03 Seconds.

Solver found a conservative solution
to the robust chance constrained
problem. All constraints are
satisfied.
Solve time: 7.70 Seconds.

---- Start Simulation ----
Model: StochasticExamples.xlsx

Simulation finished successfully.
Successful Trials: 1000
    Error Trials: 0
    Total Trials: 1000


Simulation time: 0.27 Seconds.
```

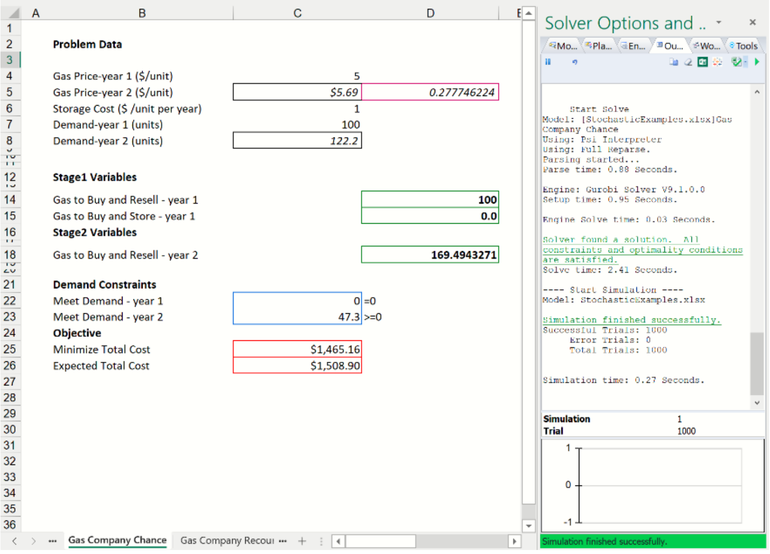
Simulation	1
Trial	1000

We run one simulation to see the expected value of the objective, which is about $1,509, lower than the initial value of $1,693. But our chance constraint is not only satisfied but *over*-satisfied – we meet consumer demand about 99% of the time, when we asked for only 95%. This solution is more conservative than necessary.

But Analytic Solver Comprehensive within Analytic Solver Desktop can *automatically improve* this solution, by adjusting the size of the uncertainty set for the chance constraint. To do this, click the special button to “Auto Adjust Chance Constraints” that appears at the top of the Task Pane. After a moment, a new solution appears with the message “Solver has converged to the current solution of the robust chance constrained problem. All constraints are satisfied.” To resolve cell C26, double click cell C25 and click "Yes" to run a simulation.

To find this same solution in Analytic Solver Cloud, click the Platform tab and set **Auto Adjust Chance Constraints** to True and then resolve by clicking the green arrow on the Model tab. To resolve cell C26, click the Simulation button to run a simulation.

The results are shown on the next page.



	A	B	C	D
2		**Problem Data**		
4		Gas Price-year 1 ($/unit)	5	
5		Gas Price-year 2 ($/unit)	*$5.69*	*0.277746224*
6		Storage Cost ($ /unit per year)	1	
7		Demand-year 1 (units)	100	
8		Demand-year 2 (units)	*122.2*	
12		**Stage1 Variables**		
14		Gas to Buy and Resell - year 1		**100**
15		Gas to Buy and Store - year 1		**0.0**
16		**Stage2 Variables**		
18		Gas to Buy and Resell - year 2		**169.4943271**
21		**Demand Constraints**		
22		Meet Demand - year 1	0	=0
23		Meet Demand - year 2	47.3	>=0
24		**Objective**		
25		Minimize Total Cost	$1,465.16	
26		Expected Total Cost	$1,508.90	

```
     Start Solve
Model: [StochasticExamples.xlsx]Gas
Company Chance
Using: Psi Interpreter
Using: Full Reparse.
Parsing started...
Parse time: 0.88 Seconds.

Engine: Gurobi Solver V9.1.0.0
Setup time: 0.95 Seconds.

Engine Solve time: 0.03 Seconds.

Solver found a solution.  All
constraints and optimality conditions
are satisfied.
Solve time: 2.41 Seconds.

---- Start Simulation ----
Model: StochasticExamples.xlsx

Simulation finished successfully.
Successful Trials: 1000
     Error Trials: 0
     Total Trials: 1000


Simulation time: 0.27 Seconds.
```

Simulation	1
Trial	1000

Simulation finished successfully.

We run one simulation to see the expected value of the objective, which is about $1,447. As noted earlier, this very simple model could have been solved via simulation optimization; but with robust optimization, we can *scale up* a model like this one to large size, and solve it efficiently using the LP/Quadratic Solver, or any of several large-scale Solver Engines.

A Model with Recourse Decisions

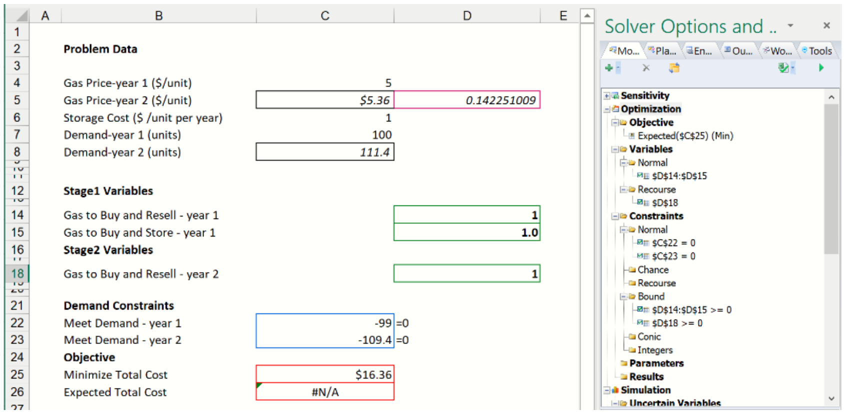
We now have a good solution to the Gas Company problem, found with either simulation optimization or robust optimization. But *we can do better than this* – we can reduce our expected total cost to $1,276, a significant improvement.

We can do this provided that *the business situation permits us* to make the decision of how much gas to purchase next year on a ‘wait and see’ basis, *after* the uncertainty of the weather is resolved – a so-called **recourse decision**. In this example, our gas supplier must accept our purchase order next year, and not require us to commit to a specific amount of gas this year for delivery next year.

One of the most important messages we can convey is this: **If the business situation permits you to make some decisions on a ‘wait and see’ basis, it’s crucial to *include* this in your optimization model.** If you *don’t* do this, you may find an optimal solution to *the wrong model* for the actual business problem.

We emphasize this because so many people have built models that *don't* include 'wait and see' decisions, because (i) they haven't learned this concept and (ii) their software hasn't allowed them to create models with recourse decisions. Simulation optimization is very popular, but as described in the technical literature and implemented in software, ***it has no concept of recourse decisions***.

In the next example, Gas Company Recourse, we'll assume that our decision of how much gas to purchase next year *can* be made on a 'wait and see' basis, *after* the uncertainty of the weather has been resolved. Click the **Gas Company Recourse** tab to display the model, which is pictured on the next page.

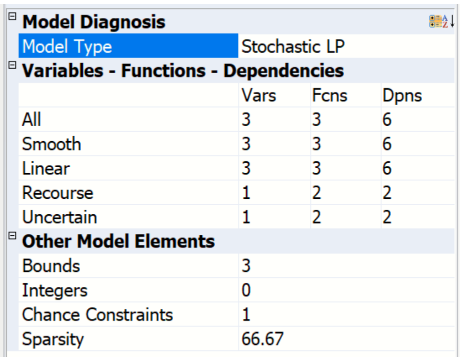


	A	B	C	D	E
1					
2		**Problem Data**			
3					
4		Gas Price-year 1 ($/unit)	5		
5		Gas Price-year 2 ($/unit)	*$5.36*	*0.142251009*	
6		Storage Cost ($ /unit per year)	1		
7		Demand-year 1 (units)	100		
8		Demand-year 2 (units)	*111.4*		
12		**Stage1 Variables**			
14		Gas to Buy and Resell - year 1		**1**	
15		Gas to Buy and Store - year 1		**1.0**	
16		**Stage2 Variables**			
18		Gas to Buy and Resell - year 2		**1**	
21		**Demand Constraints**			
22		Meet Demand - year 1	-99	=0	
23		Meet Demand - year 2	-109.4	=0	
24		**Objective**			
25		Minimize Total Cost	$16.36		
26		Expected Total Cost	#N/A		

This model has the same objective (C25), the same decision variables (D14, D15 and D18), and the same constraints (C22 and C23) as in EXAMPLE2. It differs in just two ways: (i) Cell D18 – the amount of gas to purchase next year – appears as a **recourse decision** variable. (ii) Cell C23 is **no longer a chance constraint** – it is a normal constraint, and in fact it's = rather than >= – we expect to satisfy next year's consumer demand *exactly*. The recourse decision variable allows us to do this – in each possible future scenario, once the weather uncertainty is resolved and consumer demand is known, we will purchase just the amount of gas we need to meet demand.

Solving with Robust Optimization

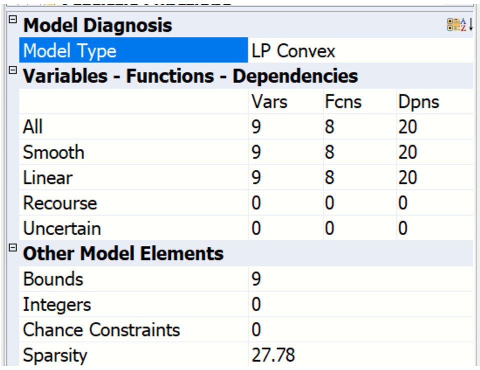
We cannot solve Gas Company Recourse with simulation optimization, since this method has no concept of recourse decisions. But we *can* use robust optimization methods to transform and solve Gas Company Recourse, much as we did with Gas Company Chance. We can proceed in much the same way. If we analyze the structure of the original model (by clicking Analyze – Analyze Transformed Model with the **Solve Uncertain Models** option set to **Simulation Optimization**), we find that it is a stochastic LP:



Model Diagnosis			
Model Type	Stochastic LP		
Variables - Functions - Dependencies			
	Vars	Fcns	Dpns
All	3	3	6
Smooth	3	3	6
Linear	3	3	6
Recourse	1	2	2
Uncertain	1	2	2
Other Model Elements			
Bounds	3		
Integers	0		
Chance Constraints	1		
Sparsity	66.67		

Again, the model has 3 variables, 3 functions, and 1 uncertainty – but now 1 of the 3 variables is a **recourse decision** variable. Note that the recourse variable plays a role in 2 functions – the objective, and the constraint that requires us to meet next year’s consumer demand. Our recourse decision variable will have ***many possible values* at the optimal solution** – *one for each realization* of the uncertain demand. With default settings in Analytic Solver Comprehensive (1,000 Monte Carlo trials in a simulation), the Solver will find 1,000 different values for D18.

If we analyze the structure of the robust counterpart model (by clicking Analyze **-** Analyze Transformed Model with the **Solve Uncertain Models** option set to **Automatic** or **Stochastic Transformation**, and the **Stochastic Transformation** option set to **Robust Counterpart**), the result is:



Model Diagnosis

Model Type	LP Convex		

Variables - Functions - Dependencies

	Vars	Fcns	Dpns
All	9	8	20
Smooth	9	8	20
Linear	9	8	20
Recourse	0	0	0
Uncertain	0	0	0

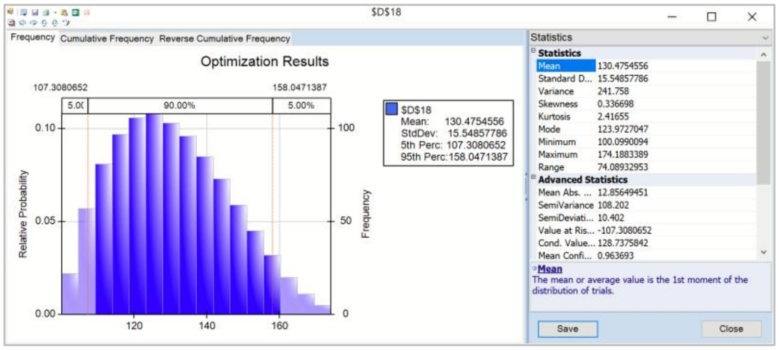
Other Model Elements

Bounds	9
Integers	0
Chance Constraints	0
Sparsity	27.78

The robust counterpart has 9 variables and 8 functions, but the uncertainty and the recourse decision have been eliminated, yielding a conventional linear programming model. As with Gas Company Chance, solving this robust counterpart model will give us an *approximate* solution to the original problem.

Now click the **Optimize** button, or the green arrow on the Task Pane. In a fraction of a second, the solution appears, with the message “Solver found a solution. All constraints and optimality conditions are satisfied.” As shown on the next page, the expected value of the objective is now about $1,365– a considerable improvement over the minimum cost of about $1,434 in Gas Company Chance.

As noted above, our recourse decision variable will have ***many possible values* at the optimal solution** – *one for each realization* of the uncertain demand. We can see these different realizations (there are 1,000 different values) by double clicking cell D18. A histogram displaying all 1,000 trial values for this recourse variable is displayed along with related statistics functions.



$D$18
Mean: 130.4754556
StdDev: 15.54857786
5th Perc: 107.3080652
95th Perc: 158.0471387

Statistics

Mean	130.4754556
Standard D...	15.54857786
Variance	241.758
Skewness	0.336698
Kurtosis	2.41655
Mode	123.9727047
Minimum	100.0990094
Maximum	174.1883389
Range	74.08932953

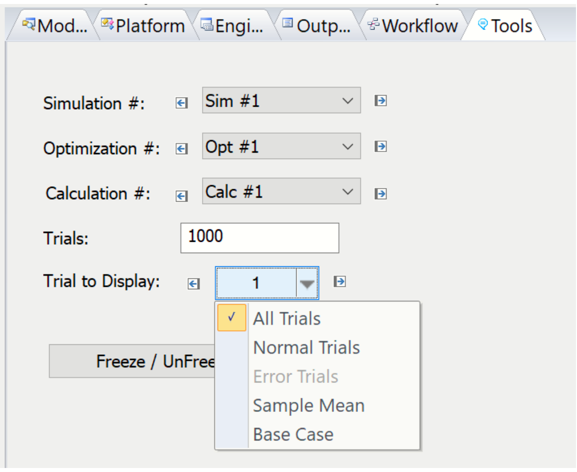
Advanced Statistics

Mean Abs. ...	12.85649451
SemiVariance	108.202
SemiDeviati...	10.402
Value at Ris...	-107.3080652
Cond. Value...	128.7375842
Mean Confi...	0.963693

Mean
The mean or average value is the 1st moment of the distribution of trials.

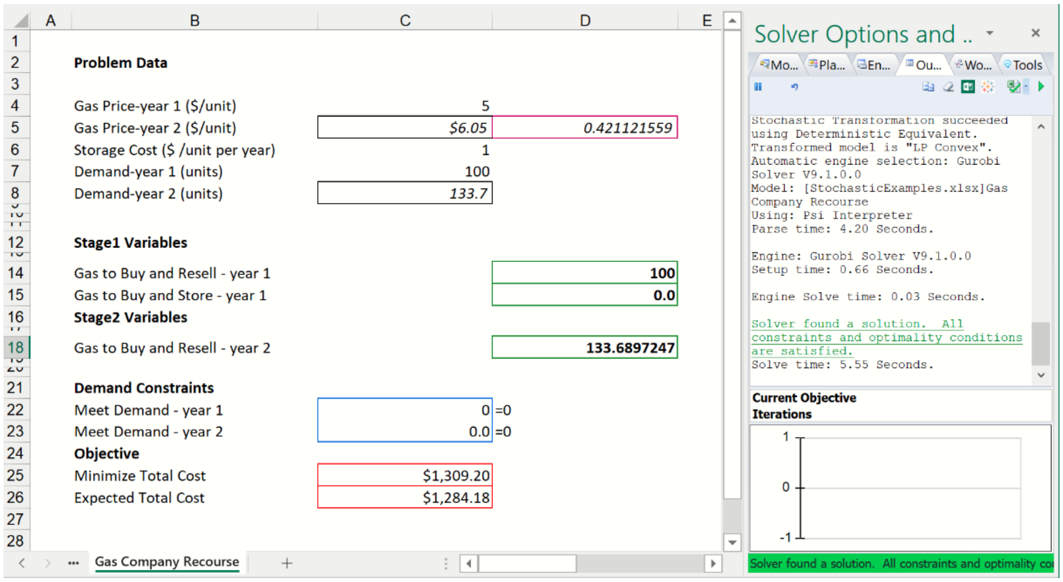
To see the value of the recourse variable at each trial, click the x in the upper right hand corner to close the dialog and then click the Tools tab on the Solver

Task pane. To increment the trial display don the worksheet, click the left and right arrows next to Trials to Display. Each time you click the left or right arrows, you'll notice that the variable values (cells D14, D15 and D18) change to the variable values found for each simulation trial.



Simulation #: Sim #1
Optimization #: Opt #1
Calculation #: Calc #1
Trials: 1000
Trial to Display: 1

If you examine the trials, you'll see the pattern: The optimal solution is to not store any gas in Year 1 for use in Year 2, but instead to buy just enough gas to meet the uncertain demand in Year 2. (With higher gas costs or lower storage costs, the optimal solution might change.) Click the Simulation icon to run a simulation and resolve the contents of cell C26.



	A	B	C	D
1				
2		**Problem Data**		
3				
4		Gas Price-year 1 ($/unit)	5	
5		Gas Price-year 2 ($/unit)	$6.05	0.421121559
6		Storage Cost ($ /unit per year)	1	
7		Demand-year 1 (units)	100	
8		Demand-year 2 (units)	133.7	
12		**Stage1 Variables**		
14		Gas to Buy and Resell - year 1		100
15		Gas to Buy and Store - year 1		0.0
16		**Stage2 Variables**		
18		Gas to Buy and Resell - year 2		133.6897247
21		**Demand Constraints**		
22		Meet Demand - year 1	0	=0
23		Meet Demand - year 2	0.0	=0
24		**Objective**		
25		Minimize Total Cost	$1,309.20	
26		Expected Total Cost	$1,284.18	
27				
28				

```
Stochastic Transformation succeeded
using Deterministic Equivalent.
Transformed model is "LP Convex".
Automatic engine selection: Gurobi
Solver V9.1.0.0
Model: [StochasticExamples.xlsx]Gas
Company Recourse
Using: Psi Interpreter
Parse time: 4.20 Seconds.

Engine: Gurobi Solver V9.1.0.0
Setup time: 0.66 Seconds.

Engine Solve time: 0.03 Seconds.

Solver found a solution.  All
constraints and optimality conditions
are satisfied.
Solve time: 5.55 Seconds.
```

Current Objective
Iterations

Solver found a solution. All constraints and optimality co

This solution offers a lower *expected* cost (the average cost over *all future* scenarios) because of the *flexibility* introduced by use of a recourse decision. And we were able to do this because we created the *right kind of model* for the business situation.

The 'moral of this story' is that Analytic Solver Comprehensive (Desktop or Cloud) gives you the power to solve such problems, using three different technologies – simulation optimization, stochastic programming, and robust optimization. But it's *up to you* to take advantage of the concept of recourse decisions – when they can be used – and create the *right kind of model* for your company’s business situation.

Deploying Your Model

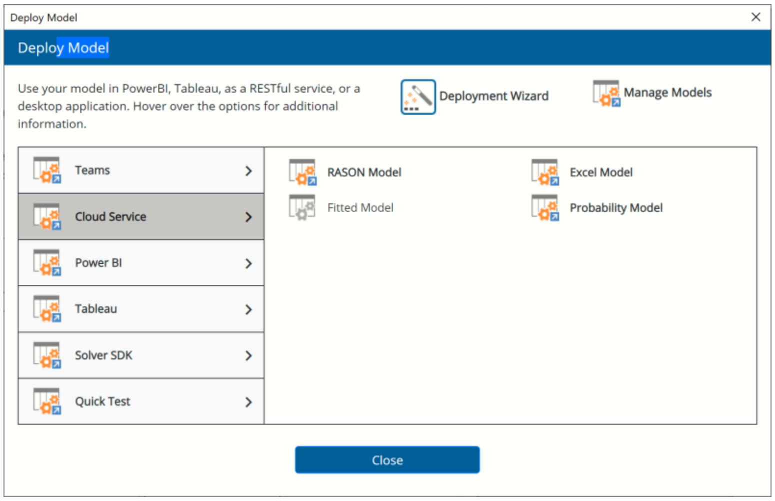
Introduction

When using Analytic Solver you have the ability to not only create, design and solve your optimization, simulation, simulation optimization, stochastic optimization and data science/forecasting models in Microsoft Excel, but you also have the ability to embed your model into your own application and solve it on the Web or within Power BI or Tableau by calling our RASON Server.

RASON (which stands for Restful Analytic Solver Object Notation) is Frontline Systems' modeling language embedded in JSON and a REST API that makes it easy to create, test and deploy analytic models using optimization, simulation and data science, in web and mobile applications.

Users of Frontline's Excel Solvers will find that 1. it's exceedingly easy to translate Excel models into RASON models 2. that your knowledge of Excel formulas and functions is immediately usable and 3. RASON models can be more flexibly "bound" to data from a variety of sources.

When you click the Deploy Model icon located on the Analytic Solver ribbon, you'll notice that your optimization or simulation model can be deployed to Cloud, Power BI, Tableau, or your own application on the Web.


Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Web App Developers will be able to immediately find how exceptionally easy it is to embed RASON models as JSON and solve them using Frontline's RASON server, which exposes a simple REST API that's scalable to handle very large, compute-intensive analytic models. Months of work, that would have previously been required, have been reduced to a single command button click!

Notes:

- A license for Analytic Solver Comprehensive allows access to the Deploy Model menu in its entirety. All functions of Deploy Model are supported with this license.
- Licenses for Analytic Solver Optimization, Analytic Solver Simulation and Analytic Solver Data Science do not support deployment to

RASON Decision Services, Import Probability Model functionality or access to Manage Models. These product licenses do support deployment to Teams, Power BI, Tableau and Quick Test (only Fitted Model and Probability Model are enabled).

- Analytic Solver Upgrade, Premium Solver and the basic course license do not support any Deploy Model functionality.

Deploy Model Menu

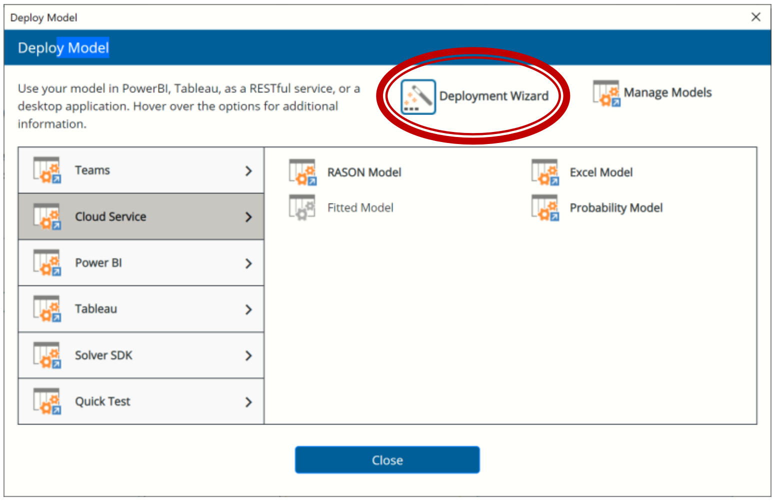
The purpose of Deploy Model is to enable your model to be used wherever/whenever it is needed – on the factory floor, on a salesperson's laptop or smart phone or in a call center custom application. On your own, without web development or IT help, you can get your model working as a cloud service. Once you've shown that it works, your developers or IT will find they can everything they'd like – customize code, connect to operational databases, apply security and governance best practices.

Rason makes your model accessible 24 x 7 via simply REST API and JSON responses – the common standard for web and mobile apps. Even better, it makes your model results available via OData – the standard widely supported by Microsoft apps including Power Platform. And Rason has built-in facilities to get the updated data your model needs directly from operational business systems. This is amazingly easy if you are using Power BI, Power Apps, Power Automate, or Dynamics 365.

Rason contains the entire Excel formula language as a subset, including virtually all of Excel's built-in functions. That makes translation possible and it allows you to see your Excel formulas embedded in Rason's syntax. To work well in web and mobile apps, Rason is embedded in JSON (JavaScript Object Notation) which allows it to work directly in Power BI, Tableau or a page on any website.

With RASON, you can create more flexible models, by working with Rason arrays and tables instead of fixed-size cell ranges. When products, regions, time periods and other "dimensions" change, your model can automatically adjust and keep running – without any extra work on your part.

When you first click the Deploy Model icon on the Analytic Solver ribbon, the new Deploy Model dialog appears.

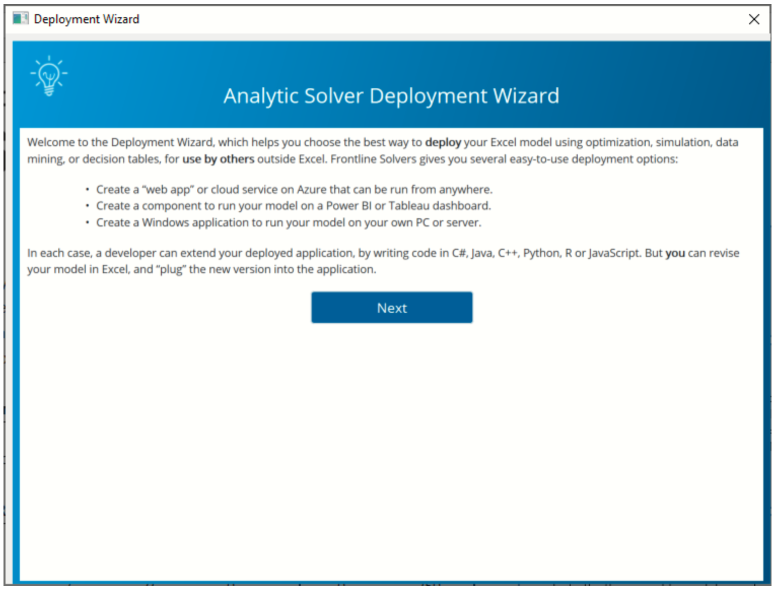


Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Follow the steps below to convert an optimization or simulation model into a RASON model and solve it in the Cloud, Tableau, Power BI, with Solver SDK or on an automatically created web page.

Deployment Wizard

If you just aren't sure where to start or what you need, use the handy deployment wizard to help guide you through the Deploy Model menu options.

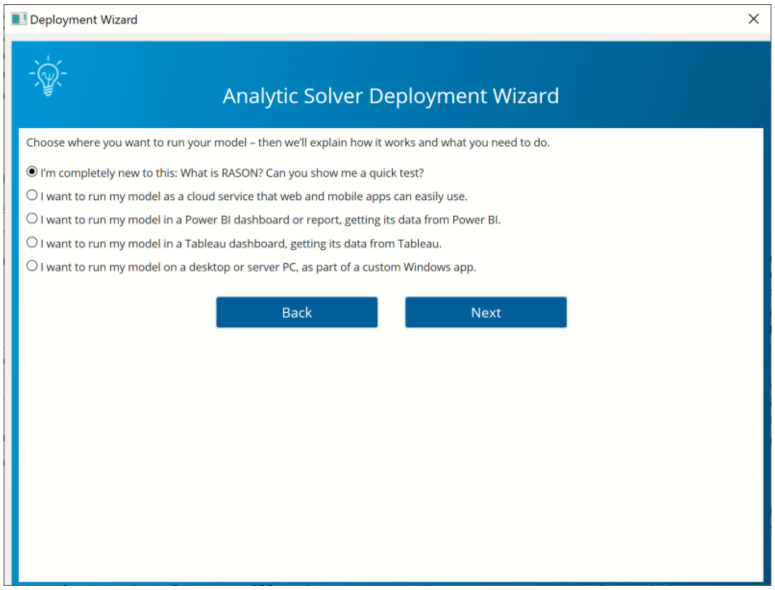


Welcome to the Deployment Wizard, which helps you choose the best way to **deploy** your Excel model using optimization, simulation, data mining, or decision tables, for **use by others** outside Excel. Frontline Solvers gives you several easy-to-use deployment options:

- Create a "web app" or cloud service on Azure that can be run from anywhere.
- Create a component to run your model on a Power BI or Tableau dashboard.
- Create a Windows application to run your model on your own PC or server.

In each case, a developer can extend your deployed application, by writing code in C#, Java, C++, Python, R or JavaScript. But **you** can revise your model in Excel, and "plug" the new version into the application.

When you click "Next" you'll be guided to select the best way to deploy your Excel model. In each case, you or a developer can extend your deployed application by writing C#, JAVA, C++, Python, R, JavaScript, or you can revise your model in Excel and "plug" the new version into the application.

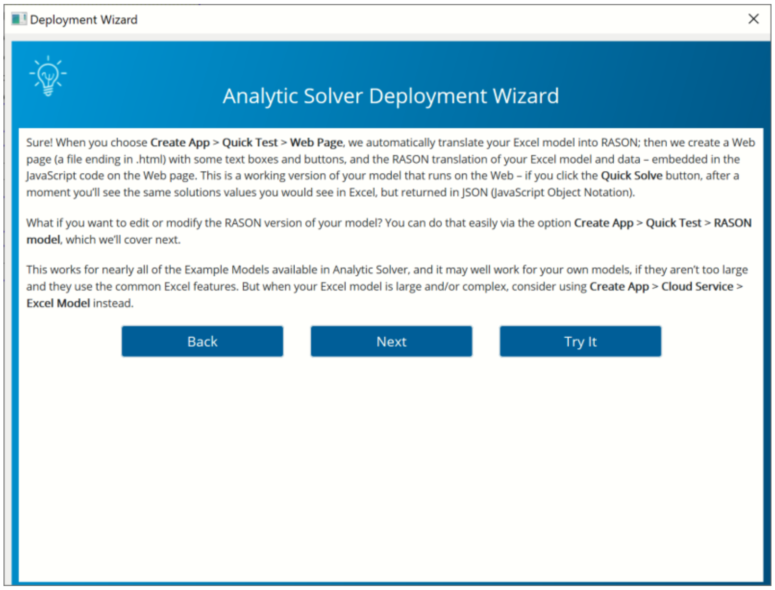


Choose where you want to run your model – then we'll explain how it works and what you need to do.

● I'm completely new to this: What is RASON? Can you show me a quick test?
○ I want to run my model as a cloud service that web and mobile apps can easily use.
○ I want to run my model in a Power BI dashboard or report, getting its data from Power BI.
○ I want to run my model in a Tableau dashboard, getting its data from Tableau.
○ I want to run my model on a desktop or server PC, as part of a custom Windows app.

Deploy Model Menu

The first option, "I'm completely new…" deploys your simulation, optimization, data science or decision table Excel Model to a Web application where your model can be solved. Click "Next" to learn how Analytic Solver can automatically transform your Excel model into a RASON model.

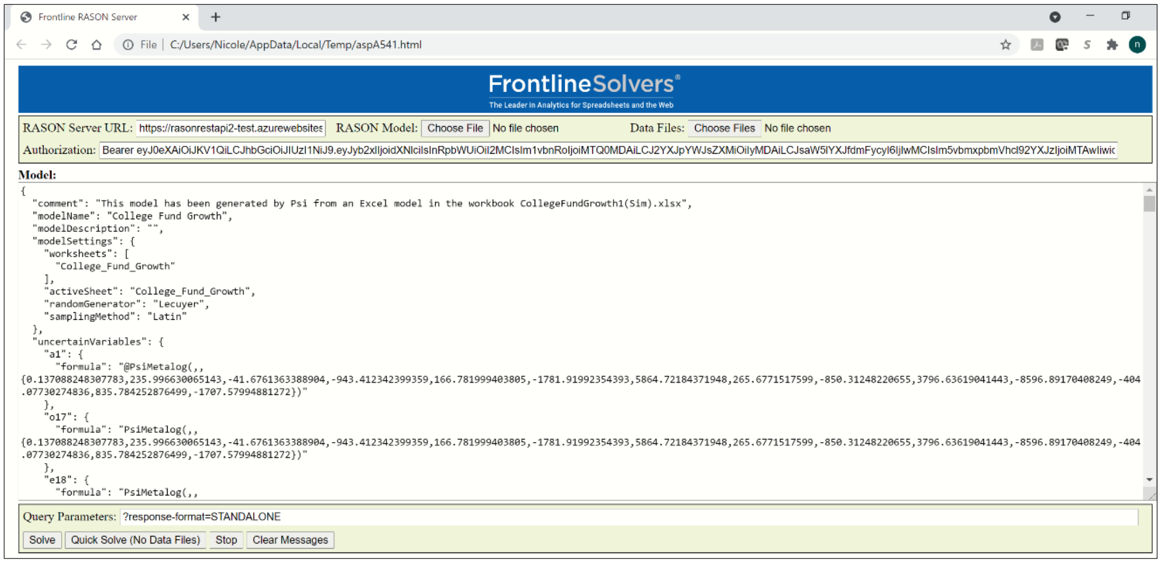


Sure! When you choose **Create App** > **Quick Test** > **Web Page**, we automatically translate your Excel model into RASON; then we create a Web page (a file ending in .html) with some text boxes and buttons, and the RASON translation of your Excel model and data – embedded in the JavaScript code on the Web page. This is a working version of your model that runs on the Web – if you click the **Quick Solve** button, after a moment you'll see the same solutions values you would see in Excel, but returned in JSON (JavaScript Object Notation).

What if you want to edit or modify the RASON version of your model? You can do that easily via the option **Create App** > **Quick Test** > **RASON model**, which we'll cover next.

This works for nearly all of the Example Models available in Analytic Solver, and it may well work for your own models, if they aren't too large and they use the common Excel features. But when your Excel model is large and/or complex, consider using **Create App** > **Cloud Service** > **Excel Model** instead.

Click "Try It" to deploy your simulation, optimization, data science or decision table Excel model to a Web application.



C:/Users/Nicole/AppData/Local/Temp/aspA541.html

RASON Server URL: https://rasonrestapi2-test.azurewebsites

RASON Model: No file chosen

Data Files: No file chosen

Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJyb2xlIjoidXNlciIsInRpbWUiOiI2MCIsIm1vbnRoIjoiMTQ0MDAiLCJ2YXJpYWJsZXMiOiIyMDAiLCJsaW5lYXJfdmFycyI6IjIwMCIsIm5vbmxpbmVhcl92YXJzIjoiMTAwIiwic

Model:

```
{
  "comment": "This model has been generated by Psi from an Excel model in the workbook CollegeFundGrowth1(Sim).xlsx",
  "modelName": "College Fund Growth",
  "modelDescription": "",
  "modelSettings": {
    "worksheets": [
      "College_Fund_Growth"
    ],
    "activeSheet": "College_Fund_Growth",
    "randomGenerator": "Lecuyer",
    "samplingMethod": "Latin"
  },
  "uncertainVariables": {
    "a1": {
      "formula": "@PsiMetalog(,,
{0.137088248307783,235.996630065143,-41.6761363388904,-943.412342399359,166.781999403805,-1781.91992354393,5864.72184371948,265.6771517599,-850.31248220655,3796.63619041443,-8596.89170408249,-404
.07730274836,835.784252876499,-1707.57994881272})"
    },
    "o17": {
      "formula": "PsiMetalog(,,
{0.137088248307783,235.996630065143,-41.6761363388904,-943.412342399359,166.781999403805,-1781.91992354393,5864.72184371948,265.6771517599,-850.31248220655,3796.63619041443,-8596.89170408249,-404
.07730274836,835.784252876499,-1707.57994881272})"
    },
    "e18": {
      "formula": "PsiMetalog(,,
```

Query Parameters: ?response-format=STANDALONE

Clicking the first option, "I want to run my model as a cloud service…" gives you four options, RASON Model, Excel Model, Fitted Model and Probability Model.

- **RASON Model** translates your optimization, simulation, data science and decision table model into Rason and POSTs the model to your RASON account.
- **Excel Model** POSTs the Excel workbook model to the RASON account.

- **Fitted Model** POSTs the data science or forecasting fitted model saved to an Excel workbook to the RASON Server.

- When you have created uncertain variables in your workbook, you can export any of these distributions, and their correlations, in the 3.0 SIPmath Standard by clicking **Probability Model**

From here you can maintain and solve your model and get the results on www.RASON.com.

Clicking the second option, Power BI, gives you the ability to turn your Excel-based optimization, simulation or data science fitted model into a **Microsoft Power BI Custom Visual**. First, you simply select rows or columns of data to serve as changeable parameters, next, you select **Deploy Model – Power BI – Managed Model/Embedded Model**, and thirdly, you save the file created by Analytic Solver. Afterwards, you click the Load Custom Visual icon in Power BI and select the file you just saved. What you get isn't just a chart – it's your *full optimization, simulation or data science fitted model*, ready to accept Power BI data**, run on demand** on the web, and display visual results in Power BI! You simply need to drag and drop appropriate Power BI datasets into the "well" of inputs to match your model parameters. This is possible because Analytic Solver translates your Excel model into **RASON**® then "wraps" a JavaScript-based Custom Visual around the RASON model. For more information on this feature, see the next chapter "Creating Power BI Custom Visuals".

Clicking the third option, Tableau, allows you to turn your Excel-based optimization, simulation or data science fitted model into a **Tableau Dashboard Extension**. You simply select rows or columns of data to serve as changeable parameters, then choose **Deploy Model – Tableau – Managed Model/Embedded Model**, and save the file created by Analytic Solver. In Tableau, you'll see the newly created file under **Extensions** on the left side of the dashboard, where you can drag it onto your dashboard. You'll be prompted to match the parameters your model needs, with data in Tableau. Much like with Power BI, what you get isn't just a chart – it's your *full optimization, simulation or data science fitted model*, ready to accept Tableau data**, run on demand** (using our **RASON** server), and display visual results in Tableau! Note: This feature works (only) with Tableau version 2018.2 or later. For more information on this feature, see the chapter "Creating Custom Extensions in Tableau".

The fourth option, Solver SDK – Rason Model translates your optimization, simulation or decision table Excel model to Rason and then opens an application that loads the RASON model. Solver SDK – Excel Model runs the SDK application and loads the RASON model.

Conversion Exceptions

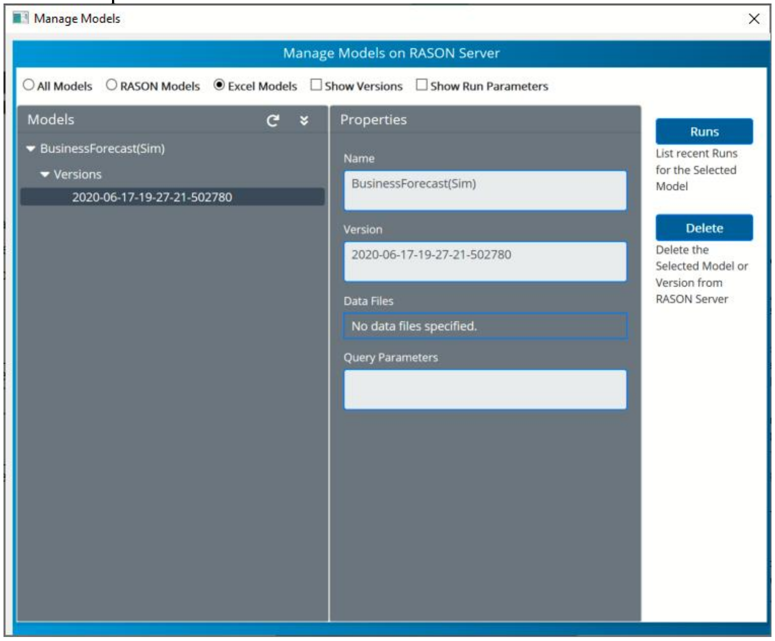
There are several types of models that Frontline Excel Solvers are not able to convert to RASON models, i.e. Excel models containing errors such as #NUM, #VALUE, etc. For all unsupported features, the conversion generator returns an appropriate message.

Note: When converting a simulation model into the RASON modeling language note that only statistics with explicit numeric arguments such as =PsiPercentile(cell, 0.50) or =PsiPercentile(cell, A1) where A1 = 0.05 are supported. Expressions passed to arguments such as =PsiPercentile(cell, A1+A2), where A1 = .1 and A2 = .3 are not supported.

Manage Models

Click Manage Models (top right) to display your models and model versions from your account on www.rason.com.

- Selecting a model name in the left column displays the model properties in the right.
- Select All Models to see all your models on your Rason account, Rason as well as Excel models.
- Click Show Versions to see all versions of each named model.
- Click Show Run Parameters to display query parameters.
- With a model name selected, clicking the Runs button displays a list of the most recent model instances in the right column.
- Click Show Run Parameters to display query parameters.
- Clicking Delete deletes the selected named model (all versions) or a specific version.



○ All Models ○ RASON Models ● Excel Models ☐ Show Versions ☐ Show Run Parameters

Name: BusinessForecast(Sim)

Version: 2020-06-17-19-27-21-502780

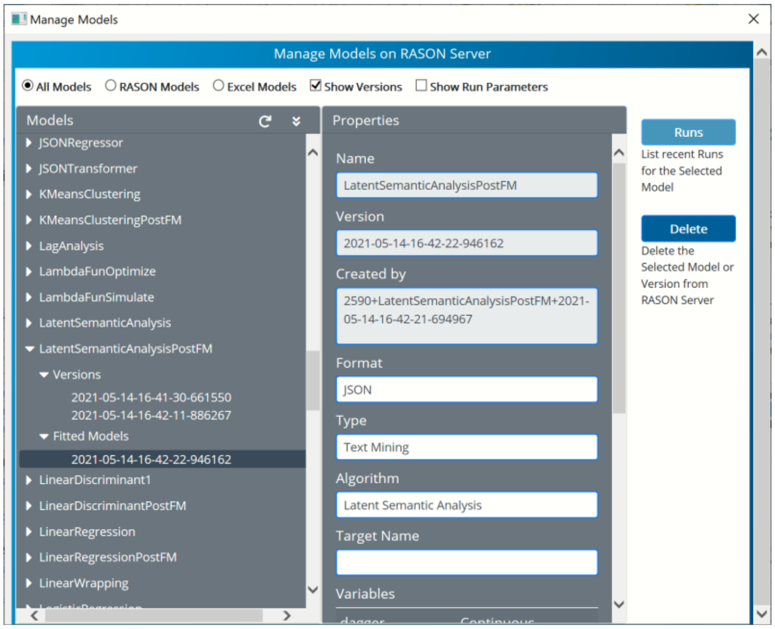
Data Files: No data files specified.

Query Parameters:

Runs: List recent Runs for the Selected Model

Delete: Delete the Selected Model or Version from RASON Server

The Manage Models dialog also supports Fitted Data Science Models. For more information on fitted models, please see the Data Science chapter within the RASON User Guide.



Name: LatentSemanticAnalysisPostFM

Version: 2021-05-14-16-42-22-946162

Created by: 2590+LatentSemanticAnalysisPostFM+2021-05-14-16-42-21-694967

Format: JSON

Type: Text Mining

Algorithm: Latent Semantic Analysis

Target Name:

Teams

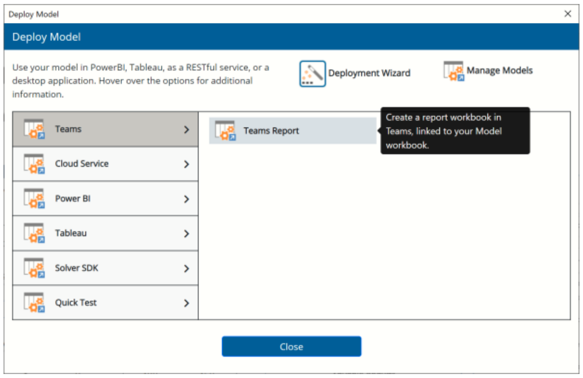
When you choose **Deploy Model** from the Ribbon and click **Teams – Teams Report**, Analytic Solver will automatically create a new workbook holding only **model results**, with external links to your **model** workbook. You can choose exactly which optimization and/or simulation results you want to include in this workbook. The new workbook will be saved online, and made available to the users you want, through a “Teams channel” that you select. Your colleagues, using **just Teams**, will be able to open the workbook and view, copy or work with the results you’re providing. And perhaps the best part: When you **re-run** your optimization or simulation model with new data, the workbook in Teams will be **automatically updated** (via those external links) with the latest model results! See the chapter “Deploying Your Model” in this Guide for full details.

Teams Report

Analytic Solver makes it easy to share your Excel model or your results with team members and colleagues using Microsoft Teams. Using the Deploy Model button, you can send a report through Microsoft Teams containing interactive links to your original workbook.

Note that your workbook must be saved to a location that is network accessible, such as in OneDrive or SharePoint.

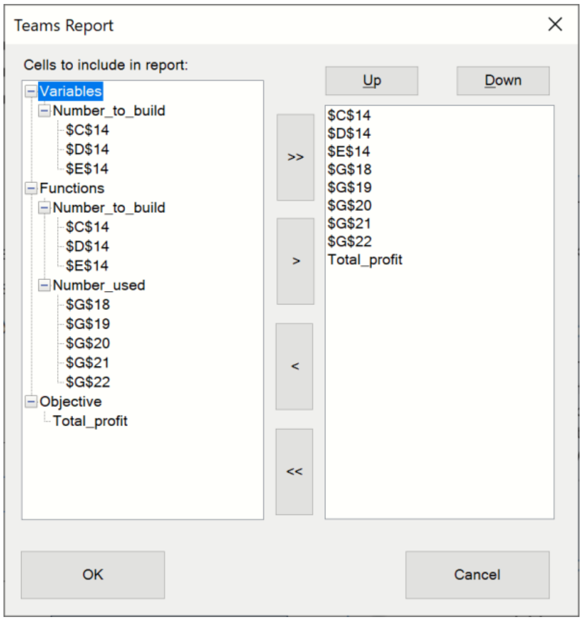
1. Once your optimization, simulation or stochastic optimization model has solved, the workbook must be saved. Afterwards, click Deploy Model on the Analytic Solver ribbon.
2. Click Teams – Teams Report to open the Optimization Chart dialog where you can select the model components to include in the Teams report.



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Create a report workbook in Teams, linked to your Model workbook.

3. Click the + buttons beside each model component to expand, then use the > or >> buttons to select the elements to be included in the report. In this example, the final values for all three varaibles, all five constraints and the objective function will be included in the report.



Cells to include in report:

- Variables
  - Number_to_build
    - $C$14
    - $D$14
    - $E$14
- Functions
  - Number_to_build
    - $C$14
    - $D$14
    - $E$14
  - Number_used
    - $G$18
    - $G$19
    - $G$20
    - $G$21
    - $G$22
- Objective
  - Total_profit

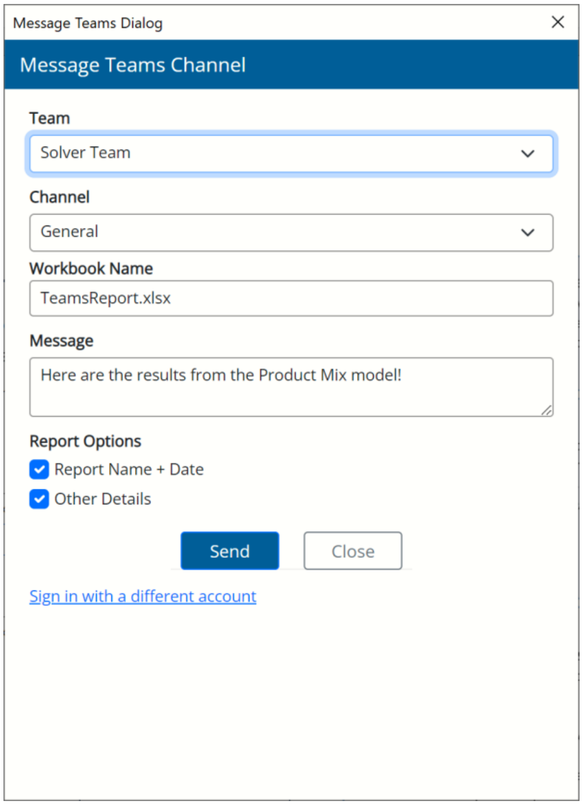
$C$14
$D$14
$E$14
$G$18
$G$19
$G$20
$G$21
$G$22
Total_profit

4. Clicking OK, opens the Message Teams Dialog.

   Select the **Team** and **Channel** where you'd like to share the results.

   To change the workbook name, simply type a new name under **Workbook Name** and also type an optional message to your teammates under **Message**.

   Select Report Name + Date and Other Details to print the name of the report, the date and other information onto the top of the report.



Message Teams Channel

Team: Solver Team

Channel: General

Workbook Name: TeamsReport.xlsx

Message: Here are the results from the Product Mix model!

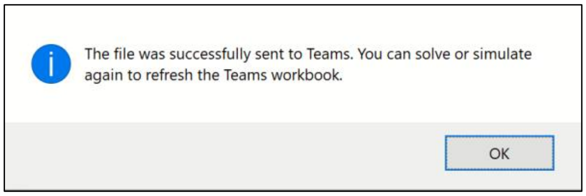
Report Options

☑ Report Name + Date

☑ Other Details

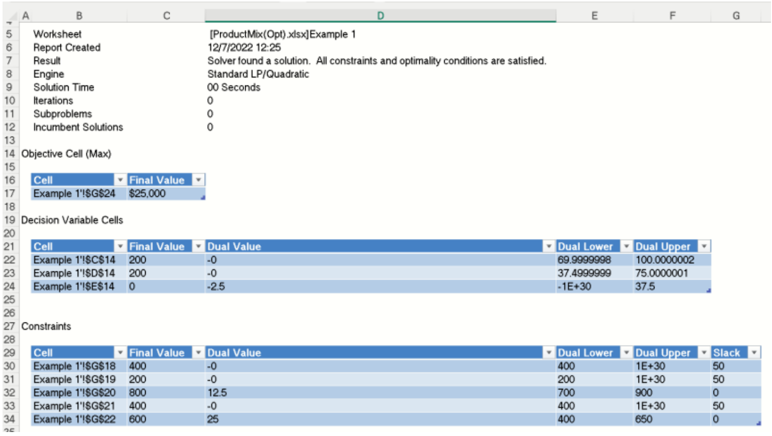
Note: If at this point an error appears or the "successful" message shown to the right does not appear, confirm that the current version of TLS is V1.2.

5. Click Send. A message will appear reporting that the report was sent successfully. Click OK.



The file was successfully sent to Teams. You can solve or simulate again to refresh the Teams workbook.

The OptTeamsReport (for an optimization model) or SimTeamsReport (for a simulation model) will be inserted to the left of the tab containing the model.



Worksheet	[ProductMix(Opt).xlsx]Example 1
Report Created	12/7/2022 12:25
Result	Solver found a solution. All constraints and optimality conditions are satisfied.
Engine	Standard LP/Quadratic
Solution Time	00 Seconds
Iterations	0
Subproblems	0
Incumbent Solutions	0

Objective Cell (Max)

Cell	Final Value
Example 1'!$G$24	$25,000

Decision Variable Cells

Cell	Final Value	Dual Value	Dual Lower	Dual Upper
Example 1'!$C$14	200	-0	69.9999998	100.0000002
Example 1'!$D$14	200	-0	37.4999999	75.0000001
Example 1'!$E$14	0	-2.5	-1E+30	37.5

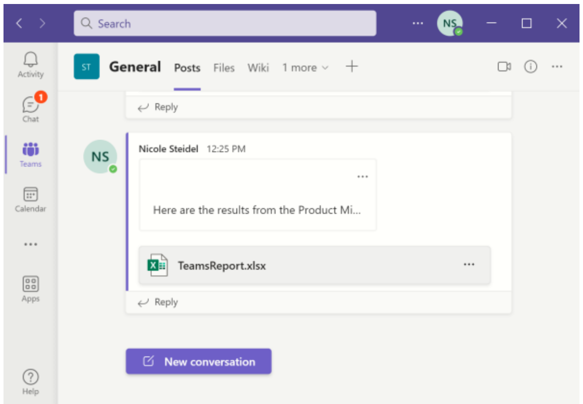
Constraints

Cell	Final Value	Dual Value	Dual Lower	Dual Upper	Slack
Example 1'!$G$18	400	-0	400	1E+30	50
Example 1'!$G$19	200	-0	200	1E+30	50
Example 1'!$G$20	800	12.5	700	900	0
Example 1'!$G$21	400	-0	400	1E+30	50
Example 1'!$G$22	600	25	400	650	0

Note: If exporting the results of a simulation or stochastic optimization model, report generation designates all model components included in the TEAMS report as output cells by ammending "+Psioutput()" to the existing formulas. As a result, these cells will display #N/A until the user performs a 2nd simulation.

Users are invited to customize this report in a design of their choice. (Just be sure not to break the links connecting the model to the report.) There is only a need to generate the Teams Report one time. Simply refresh the report to update after each solve. When generating a new Teams Report, you must give the model a unique name.

6. Open TEAMS and the proper channel to find the Report. Double click to open.

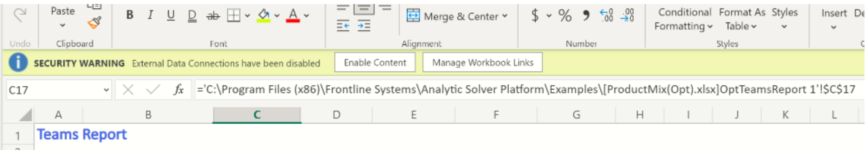


Nicole Steidel 12:25 PM

Here are the results from the Product Mi...

TeamsReport.xlsx

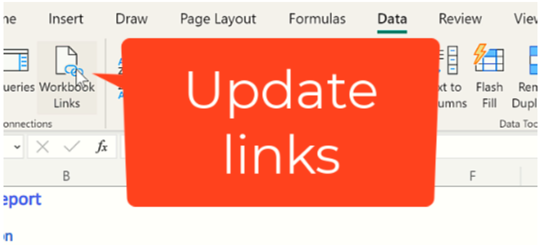
7. Click Enable Content to enable the content on the report or click Data – Workbook Links. Colleagues can now see your model results in Microsoft TEAMS!



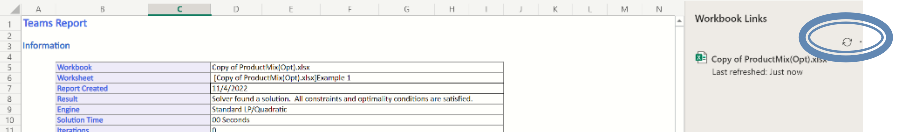
SECURITY WARNING External Data Connections have been disabled

C17 ='C:\Program Files (x86)\Frontline Systems\Analytic Solver Platform\Examples\[ProductMix(Opt).xlsx]OptTeamsReport 1'!$C$17

Teams Report



8. If a parameter changes on the original model worksheet, simply resolve the optimization or simulation model, then click the refresh icon on the Workbook Links pane. There is no need to deploy a new TEAMS report.



Teams Report

Information

Workbook	Copy of ProductMix(Opt).xlsx
Worksheet	[Copy of ProductMix(Opt).xlsx]Example 1
Report Created	11/4/2022
Result	Solver found a solution. All constraints and optimality conditions are satisfied.
Engine	Standard LP/Quadratic
Solution Time	00 Seconds
Iterations	0

Copy of ProductMix(Opt).xlsx
Last refreshed: Just now

Note for Analytic Solver Cloud Users

If working with Analytic Solver Cloud in either Online Excel or Desktop Excel, and a Frontline example model is opened from Help – Example Models, the example must first be saved to a network accessible location, then closed and reopened, before the model is deployed to Microsoft Teams. Once the model is deployed and opened within Teams, click “Enable Links” to populate the links.

Cloud Service

Clicking the second option on the Deploy Model dialog, Cloud Service, gives you four options, RASON Model, Excel Model, Fitted Model and Probability Model.

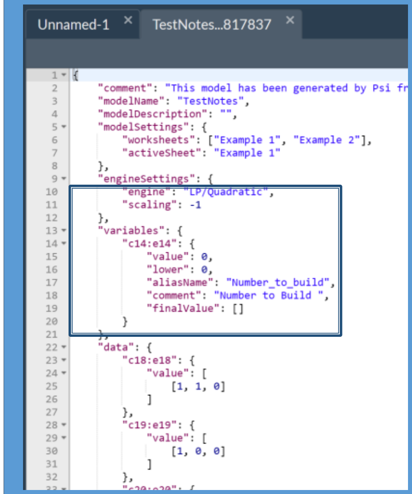
- *RASON Model* translates the Excel model into Rason and POSTs the model to your RASON account.
- *Excel Model* POSTs the Excel workbook model to the RASON account. This option is not applicable for data science/forecasting models.
- *Fitted Model* POSTs the fitted model contained within your Excel workbook to your RASON account.
- *Probablity Model* exports existing Psi distributions to your RASON account using the Probability Management 3.0 SIPmath Standard.

Cloud Service - Rason

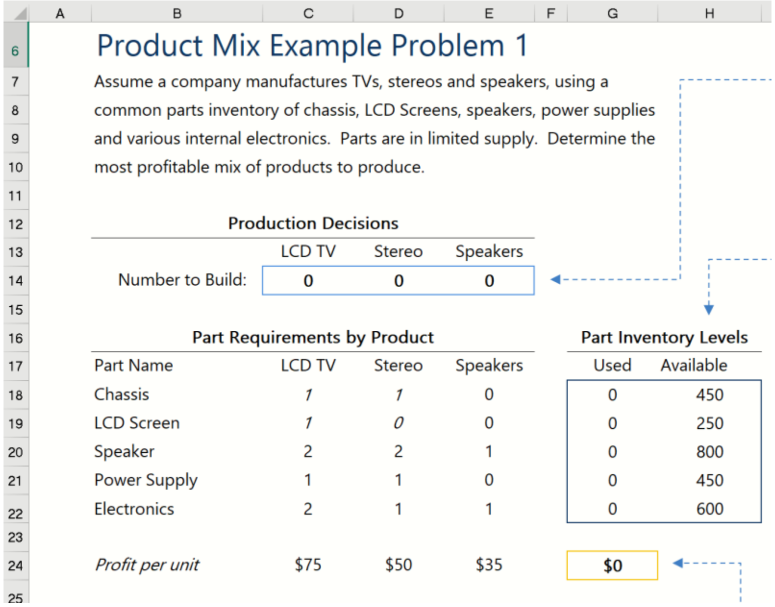
When you select Deploy Model – Cloud Service – Rason Model, Analytic Solver automatically translates your Excel workflow or optimization/simulation/decision table into Rason. The translated model is identical to your original Excel model. It has simply been translated into Frontline’s RASON modeling language. Your objective, variables, constraints and bounds are now defined in RASON, instead of in an Excel spreadsheet.

In *Analytic Solver Desktop,* open the Product Mix(RASON).xlsx model by clicking Help – Example Models on the ribbon, clicking Optimization Examples and then the Product Mix for RASON link. A similar version of this model was previously discussed in the Conventional Optimization chapter. Recall that this example model determines the optimal mix of products that a company should produce in order to maximize profits.

Excel notes will be deployed to RASON, along with the model, if they have been inserted into a model component cell. For example, in the Product Mix model to the right, if the text “Number to Build” is entered into a note for C14, the contents of the note will appear in the converted RASON model in the “comment” property for the variable block, c14:e14. Notes are supported for decision variables, uncertain variables, constraint left hand sides and right hand sides, uncertain functions, the objective, parameters (such as PsiSimParam/PsiOptParam) and Psi functions. In the case of a block of decision variables, uncertain variables, uncertain functions or constraints, the note must appear in the upper, most left cell. In this case, cell C14 is the upper, most left cell of the variable block C14:E14.



```
{
    "comment": "This model has been generated by Psi fr
    "modelName": "TestNotes",
    "modelDescription": "",
    "modelSettings": {
        "worksheets": ["Example 1", "Example 2"],
        "activeSheet": "Example 1"
    },
    "engineSettings": {
        "engine": "LP/Quadratic",
        "scaling": -1
    },
    "variables": {
        "c14:e14": {
            "value": 0,
            "lower": 0,
            "aliasName": "Number_to_build",
            "comment": "Number to Build ",
            "finalValue": []
        }
    },
    "data": {
        "c18:e18": {
            "value": [
                [1, 1, 0]
            ]
        },
        "c19:e19": {
            "value": [
                [1, 0, 0]
            ]
        },
        "c20:e20": {
```



Product Mix Example Problem 1

Assume a company manufactures TVs, stereos and speakers, using a common parts inventory of chassis, LCD Screens, speakers, power supplies and various internal electronics. Parts are in limited supply. Determine the most profitable mix of products to produce.

Production Decisions

	LCD TV	Stereo	Speakers
Number to Build:	0	0	0

Part Requirements by Product				Part Inventory Levels	
Part Name	LCD TV	Stereo	Speakers	Used	Available
Chassis	1	1	0	0	450
LCD Screen	1	0	0	0	250
Speaker	2	2	1	0	800
Power Supply	1	1	0	0	450
Electronics	2	1	1	0	600
Profit per unit	$75	$50	$35	$0	

The optimization model is included in the Model tab in the Solver Task Pane. This model written in algebraic form is below.

Maximize: Total Profit = $75x1 + $50x2 + $35x3

Subject To:

Chassis: 1x1 + 1x2 + 0x3 <= 450

LCD Screen: 1x1 + 0x2 + 0x3 <= 250

Speaker: 2x1 + 2x2 + 1x3

Power Supply: 1x1 + 1x2 + 0x3

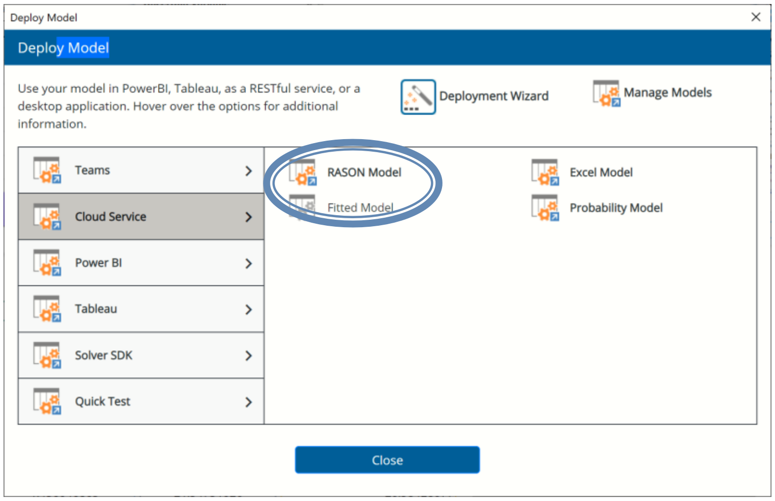
Electronics: 2x2 + 1x2 + 1x3

x1, x2, x3 > 0

Click Deploy Model on the ribbon, then select Cloud Service – RASON Model. Note that this action supports all model types: optimization, simulation, data science and decision table.

There is no need to solve the model first unless a change has been made to an already existing RASON model. If a change has been made to the workbook after a RASON model has been created, then you MUST either solve or diagnose the Excel model in order for the changes to be reflected in the new RASON model.

When creating a RASON model through desktop Analytic Solver, model elements (objective function and all variables and constraints) may exist on multiple worksheets within the same Excel workbook.

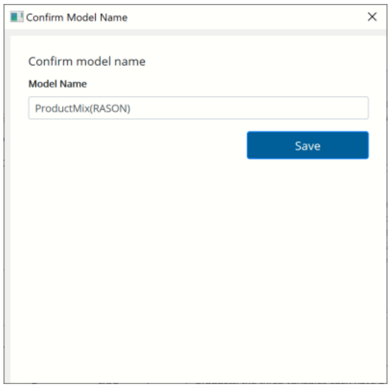


Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

On the Confirm Model Name dialog, you can accept the default name by clicking Save or you can enter a new model name and then click Save.

Note on Rason Subscriptions

If you have already registered on www.Solver.com, then you will automatically be registered on www.Rason.com with a trial subscription that expires in 15 days. If you are a subscriber to a premium plan, you will be subject to the problem limits and expiration date of the purchased plan.



Confirm model name

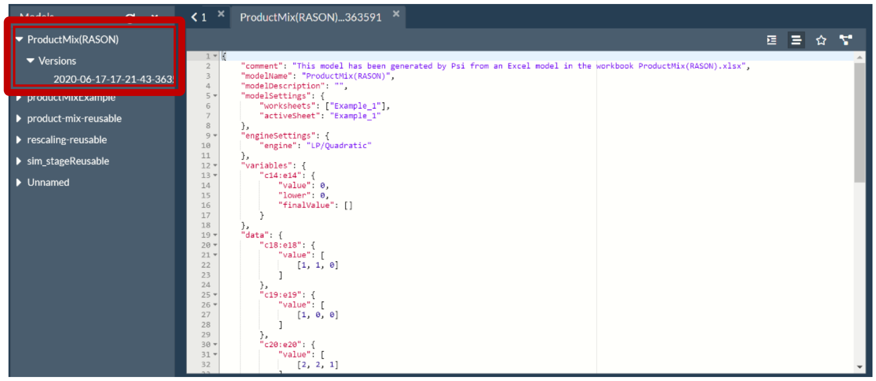
Model Name: ProductMix(RASON)

A Note on Model Versions

Each time you click Deploy Model – Cloud Service – Rason or Excel Model, you will create a new "version" of the of the model with a unique ID which will appear under the Model Name under "Versions".

For this example, click Save to accept the default model name. In a matter of seconds, a browser opens and navigates to the Web IDE on the Editor page at www.RASON.com. The Product Mix model, now written in the RASON modeling language, is displayed in the Model Editor.

Notice that the model ProductMix(RASON) is also listed in the Models pane to the left. This is because your Excel model, translated into the RASON modeling language, has automatically been posted to the Rason Server. Now you'll be able to solve the model and obtain the returns in JSON.

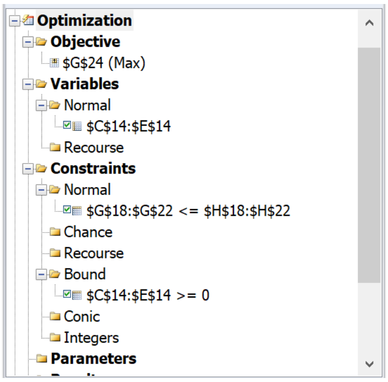


```
{
    "comment": "This model has been generated by Psi from an Excel model in the workbook ProductMix(RASON).xlsx",
    "modelName": "ProductMix(RASON)",
    "modelDescription": "",
    "modelSettings": {
        "worksheets": ["Example_1"],
        "activeSheet": "Example_1"
    },
    "engineSettings": {
        "engine": "LP/Quadratic"
    },
    "variables": {
        "c14:e14": {
            "value": 0,
            "lower": 0,
            "finalValue": []
        }
    },
    "data": {
        "c18:e18": {
            "value": [
                [1, 1, 0]
            ]
        },
        "c19:e19": {
            "value": [
                [1, 0, 0]
            ]
        },
        "c20:e20": {
            "value": [
                [2, 2, 1]
```

A RASON model begins with an opening { and closes with a closing }. Within the two braces, our model is made up of various "sections" – there are a total of eleven: "variables", "uncertain variables", "data", "dataSources", "engineSettings", "formulas", "modelSettings", "objective", "constraints", "indexSets" and "objective". Some optimization models will consist of just 3 sections: *variables* (where the decision variables will be defined), *constraints* (where the constraints will be defined) and *objective* (where the objective will be defined)) where other larger and more complex models might contain several additional segments such as: *engineSettings* (where the engine is chosen and an engine options are specified), `data` (where any arrays used in the calculation of the constraints or objective are defined), *formula* (where any intermediate calculations are performed) and/or *dataSources* (where any data is imported from an outside source such as a CSV file).

Our example is rather simple and only includes four: *variables*, *constraints*, *data* and *objective*. Let's compare our Excel model with our newly generated RASON model section by section, starting with *variables*.

If we click back to the Excel example, we can see the model in the Model tab in the Solver Task Pane.

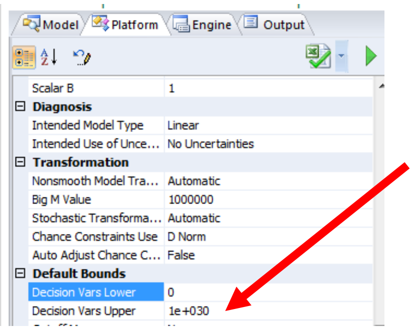


Under Variables, in the Model tab of the Solver Task Pane, we see the cell range: C14:E14. Now click back to your RASON model.

Within the *variables* section, we see the same cell range, "C14:E14". In our RASON model, C14:E14 is an array with three elements: C14, D14 and E14. (To refer to an individual element, use: "C14", "D14", or "E14".) The initial value specified for the *value* property in the RASON model equals the values contained in the Excel cell range C14:E14 at the time the model was converted (0 for all three cells). The *lower* property in the RASON model exists because a constraint specifying the variable lower bounds is present in the Excel model (C14:E14 >= 0). If an upper bound was present, the *upper* property would be inserted into the RASON model too. Upper and lower bounds may be present in the Excel model using explicit constraints, such as we have here, or they may be entered into the Platform tab of the Solver Task Pane for Decision Vars Lower and Upper. The properties "value", "lower" and "finalValue" are the default properties returned for a variable.

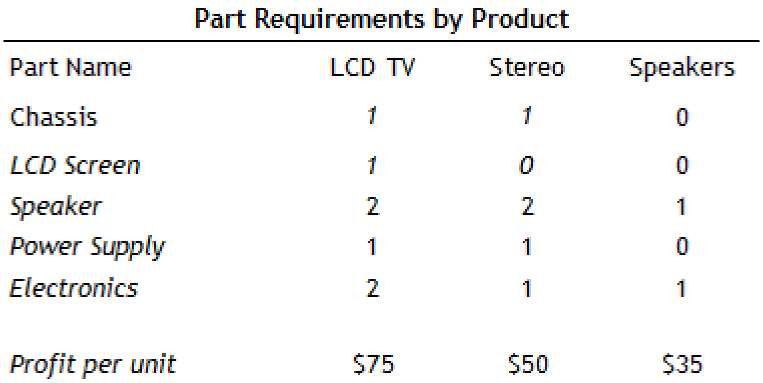
```
variables: {
    "c14:e14": {
      "value": 0,
      "lower": 0,
      "finalValue":[]
    }
}
```

Let's move on to the *data* section.



Scalar B	1
Diagnosis	
Intended Model Type	Linear
Intended Use of Unce…	No Uncertainties
Transformation	
Nonsmooth Model Tra…	Automatic
Big M Value	1000000
Stochastic Transforma…	Automatic
Chance Constraints Use	D Norm
Auto Adjust Chance C…	False
Default Bounds	
Decision Vars Lower	0
Decision Vars Upper	1e+030

The data is contained in the Excel model within the cell range: C18:E22 (constraint coefficients) and also C24:E24 (objective function coefficients).



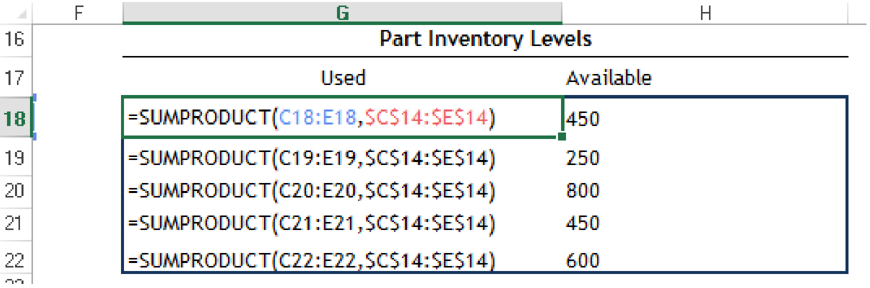
Part Requirements by Product

Part Name	LCD TV	Stereo	Speakers
Chassis	*1*	*1*	0
LCD Screen	*1*	*0*	0
Speaker	2	2	1
Power Supply	1	1	0
Electronics	2	1	1
Profit per unit	$75	$50	$35

The exact same values have been inserted into 6 arrays containing three elements each: C18:E18, C19:E19, C20:E20, C21:E21, C22:E22 and C24:E24. Again, to refer to individual elements of, say, the c18:e18 array, use: c18, d18, or e18.

```
data: {
    "c18:e18": {
      value: [
        [1, 1, 0]
      ]
    },
    "c19:e19": {
      value: [
        [1, 0, 0]
      ]
    },
    "c20:e20": {
      value: [
        [2, 2, 1]
      ]
    },
    "c21:e21": {
      value: [
        [1, 1, 0]
      ]
    },
    "c22:e22": {
      value: [
        [2, 1, 1]
      ]
    },
    "c24:e24": {
      value: [
      [75, 50, 35]
    ]
  }
},
```

Click back to the Excel model. Under Constraints in the Model tab of the Solver Task pane, we see the entry: G18:G22 <= H18:H22. If we inspect the first cell in the range, G18, we find the formula: =SUMPRODUCT(C18:E18, $C$14:$E$14) while H18=450. The other four cells, G19:G22, hold similar formulas.

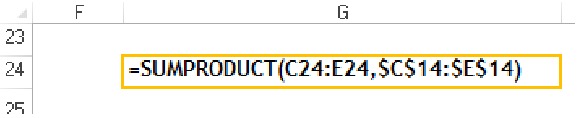


	F	G	H
16		Part Inventory Levels	
17		Used	Available
18		=SUMPRODUCT(C18:E18,$C$14:$E$14)	450
19		=SUMPRODUCT(C19:E19,$C$14:$E$14)	250
20		=SUMPRODUCT(C20:E20,$C$14:$E$14)	800
21		=SUMPRODUCT(C21:E21,$C$14:$E$14)	450
22		=SUMPRODUCT(C22:E22,$C$14:$E$14)	600

In the *constraint* section of the RASON model, we see five arrays, containing one element each, corresponding to the five constraints in our Excel model. The value for each upper bound is taken from cells H18:H22.

```
constraints: {
    "g18": {
      "formula":"SUMPRODUCT(C18:E18,$C$14:$E$14)",
      "upper": 450
    },
    "g19": {
      "formula":"SUMPRODUCT(C19:E19,$C$14:$E$14)",
      "upper": 250
    },
    "g20": {
      "formula":"SUMPRODUCT(C20:E20,$C$14:$E$14)",
      "upper": 800
    },
    "g21": {
      "formula":"SUMPRODUCT(C21:E21,$C$14:$E$14)",
      "upper": 450
    },
    "g22": {
      "formula": "SUMPRODUCT(C22:E22,$C$14:$E$14)",
      "upper": 600 }
    },
```

The objective function being maximized in the Excel model is located in cell G24 and contains the formula, =SUMPRODUCT(C24:E24,$C$14:$E$14).



	F	G
23		
24		=SUMPRODUCT(C24:E24,$C$14:$E$14)
25		

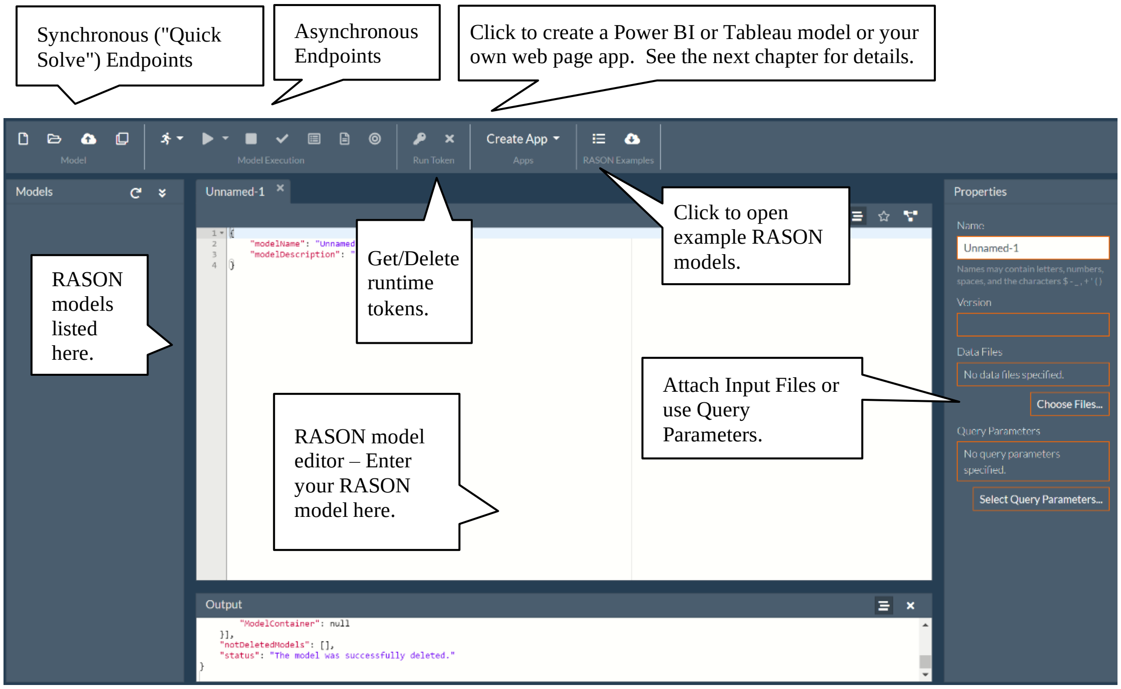
Looking back to the RASON model, we see the same formula for the G24 array. The type property is set to "max" to maximize the objective.

```
"objective": {
    "g24": {
      "formula":"SUMPRODUCT(C24:E24,$C$14:$E$14)",
      "type": "max",
      "finalValue": []
    }
}
```

At the end of this section, a closing } ends the RASON model. For information on what a Rason model looks like for a simulation, decision table or data science model, see the Rason User Guide.

Models entered into the Rason Model Editor in the Web IDE may be solved or analyzed using either a 'Quick Solve' REST API endpoint (also referred to as Synchronous Endpoints) or a standard REST API endpoint (also referred to as Asynchronous endpoints). Note: The endpoints displayed on these command buttons are the actual REST API endpoints that will be called when you create your own webpage in the next section. For more information on these endpoints, see the *RASON User Guide*.

RASON Model Editor


Synchronous ("Quick Solve") Endpoints
Asynchronous Endpoints
Click to create a Power BI or Tableau model or your own web page app. See the next chapter for details.
Model
Model Execution
Run Token
Create App
Apps
RASON Examples
Models
Unnamed-1
RASON models listed here.
Get/Delete runtime tokens.
Click to open example RASON models.
Properties
Name
Unnamed-1
Names may contain letters, numbers, spaces, and the characters $ - _ . + ' ( )
Version
Data Files
No data files specified.
Choose Files...
Query Parameters
No query parameters specified.
Select Query Parameters...
Attach Input Files or use Query Parameters.
RASON model editor – Enter your RASON model here.
Output

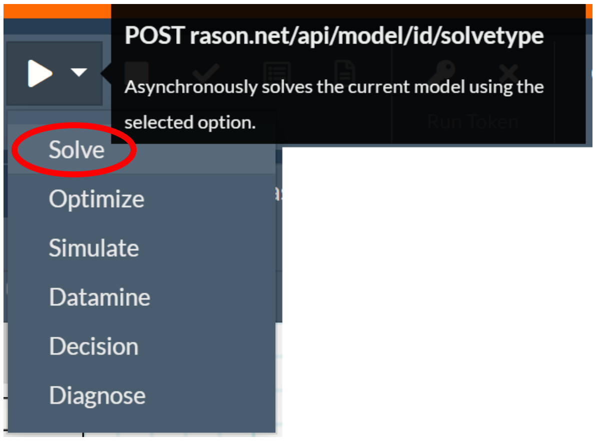
```
{
    "modelName": "Unnamed
    "modelDescription": "
}
```

```
        "ModelContainer": null
    }],
    "notDeletedModels": [],
    "status": "The model was successfully deleted."
}
```

There are two ways to solve a model in Rason, using the "Quick Solve" or Synchronous API calls or the Asynchronous API Calls. For this example, we will use a call to an asynchronous API endpoint. For more information on Quick Solve endpoints, see the RASON User Guide.

Click the arrow next to the ▶▾ icon and select Optimize (or Solve) from the Solve drop down menu to call the RASON endpoint, **POST rason.net/api/model/{nameorid}/optimize**. The model is submitted to the

RASON Server which solves the model and returns the result in the Output window, shown below.


POST rason.net/api/model/id/solvetype

Asynchronously solves the current model using the selected option.

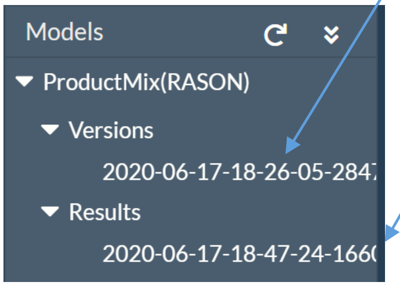
"Solve" calls the REST API endpoint POST rason.net/api/model/ProductMix(RASON)/solve which solves any problem type (optimization, simulation, decision table or data science) while Optimize calls the REST API endpoint POST rason.net/api/model/ProductMix(RASON)/optimize which solves only optimization models.

In the Output Window at the bottom of the screen you'll see similar output.

```
Executing asynchronous solve: POST
https://rason.net/api/model/2590+ProductMix(RASON)+2020-06-17-18-
26-05-284735/solve
{
    "ModelId": "2590+ProductMix(RASON)+2020-06-17-18-47-24-
     166065",
    "ModelName": "ProductMix(RASON)",
    "ModelDescr": "",
    "ModelFiles": [],
    "RuntimeToken": "",
    "ModelType": "Instance",
    "ModelKind": "Rason",
    "IsChampion": false,
    "ParentModelId": "2590+ProductMix(RASON)+2020-06-17-18-26-05-
    284735",
    "QueryString": ""

}
```

Model ID identifies this instance of the ProductMix(RASON) model. The ParentModelId is the model that was solved to create this instance. For more information on the Rason REST API, see the Rason User Guide downloadable from the Help page.

The results in JSON are inserted into the Models pane beneath "Results".



If we click the entry under Results, the Editor tab will open and display the results of the solve in JSON.

```
{
    "status": {
        "id": "2590+ProductMix(RASON)+2020-06-17-18-47-24-166065",
        "code": 0,
        "codeText": "Solver found a solution.  All constraints and
optimality conditions are satisfied.",
        "solveTime": 64
    },
    "results": {
        "c14:e14": ["finalValue"],
        "g24_": ["finalValue"]
    },
    "c14:e14": {
        "objectType": "dataFrame",
        "name": "c14:e14",
        "order": "col",
        "colNames": ["finalvalue"],
        "colTypes": ["double"],
        "indexCols": null,
        "data": [
            [200, 200, 0]
        ]
    },
    "g24_": {
        "objectType": "dataFrame",
        "name": "g24_",
        "order": "col",
        "colNames": ["finalvalue"],
        "colTypes": ["double"],
        "indexCols": null,
        "data": [
            [25000]
        ]
    }
}
```

The result "Solver found a solution. All constraints and optimality conditions are satisfied" means that no other solution exists that is better than the solution found. (For more information on this and all possible Solver Result messages, please see the *RASON Reference Guide*.) To maximize profit, we should make 200 TV sets, 200 stereos and 0 speakers. The final objective value of 25,000 is the maximum profit that we can attain.

Deploying a Simulation model to RASON

This section deploys a simulation model to RASON. Again, when you select Deploy Model – Cloud Service – Rason Model, Analytic Solver will automatically translate the simulation model into the Rason modeling language. The translated simulation model is identical to your original Excel simulation model. It has simply been translated into Frontline’s RASON modeling language. Your uncertain variables, uncertain functions and statistics functions are now defined in RASON, instead of in an Excel spreadsheet.

The model used in this example is the Yield Management model. A similar version of this model was previously discussed in the Simulation and Risk Analysis chapter. Recall that this example model determines the optimal number of tickets to sell in order to maximize revenue for a specific flight.

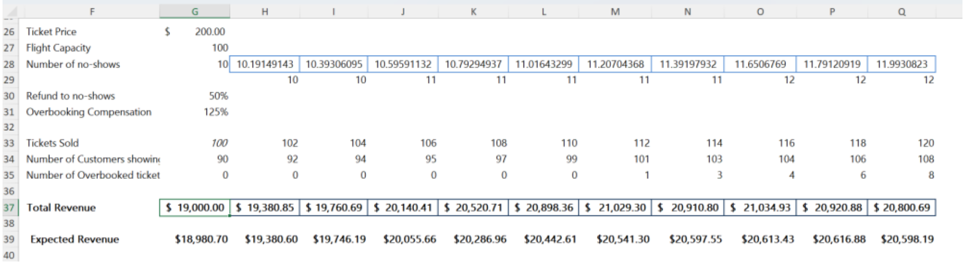
Uncertain Variables: The uncertain variables in the model are located in cells H28:Q28. Each of the 10 uncertain variables uses the PsiLogNormal distribution to model the number of passengers who do not “make” or show up to the flight. These missing passengers are referred to as “no-shows”.

Uncertain Functions: The uncertain variables in the model are located in cells G37:Q37. These functions calculate the Total Revenue for the flight based on the number of tickets sold (cells G33:Q33) minus the number of no-show passengers. Note the formula in each of these uncertain functions is of the form:

```
=(Tickets Sold * Ticket Price)-(No Shows Refund *
Ticket Price * Number of No Shows) - (Overbooking
Compensation * Ticket Price * Number of overbooked
customers)
```

Note that overbooked customers are customers who have been "bumped" from the flight and are therefore entitled to compensation of 125% of the ticket price. "No-show" passengers are entitled to compensation of 50% of the ticket price.

Statistics: Cells G39:Q39 contain the statistical function, PsiMean() which calculates the expected mean for Total Revenue.

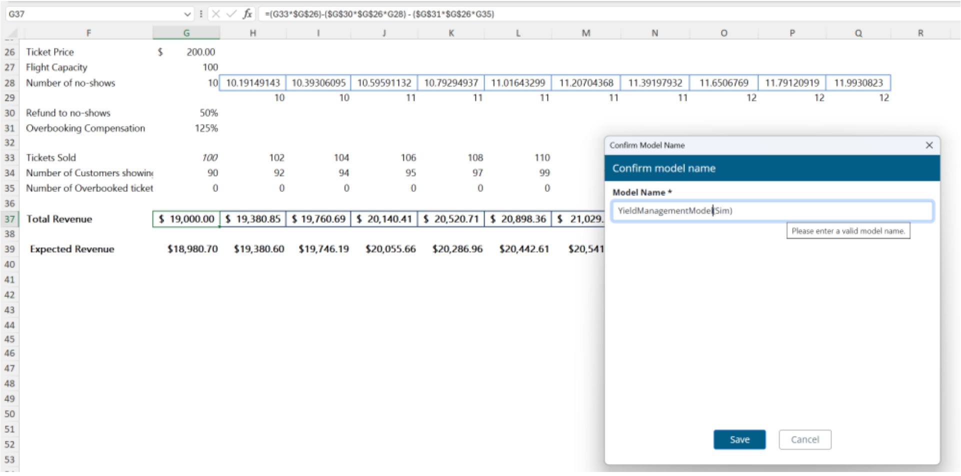


	F	G	H	I	J	K	L	M	N	O	P	Q
26	Ticket Price	$ 200.00										
27	Flight Capacity	100										
28	Number of no-shows	10	10.19149143	10.39306095	10.59591132	10.79294937	11.01643299	11.20704368	11.39197932	11.6506769	11.79120919	11.9930823
29			10	10	11	11	11	11	11	12	12	12
30	Refund to no-shows	50%										
31	Overbooking Compensation	125%										
32												
33	Tickets Sold	*100*	102	104	106	108	110	112	114	116	118	120
34	Number of Customers showin	90	92	94	95	97	99	101	103	104	106	108
35	Number of Overbooked ticket	0	0	0	0	0	0	1	3	4	6	8
36												
37	**Total Revenue**	$ 19,000.00	$ 19,380.85	$ 19,760.69	$ 20,140.41	$ 20,520.71	$ 20,898.36	$ 21,029.30	$ 20,910.80	$ 21,034.93	$ 20,920.88	$ 20,800.69
38												
39	**Expected Revenue**	$18,980.70	$19,380.60	$19,746.19	$20,055.66	$20,286.96	$20,442.61	$20,541.30	$20,597.55	$20,613.43	$20,616.88	$20,598.19
40												

Deploy the Model

Click Deploy Model – Cloud Service – RASON Model to deploy the simulation model to the RASON Server, viewable on the Editor tab at www.RASON.com.

Click Save on the Confirm model name dialog to confirm the model name YieldManagementModel(Sim).



G37 =(G33*$G$26)-($G$30*$G$26*G28) - ($G$31*$G$26*G35)

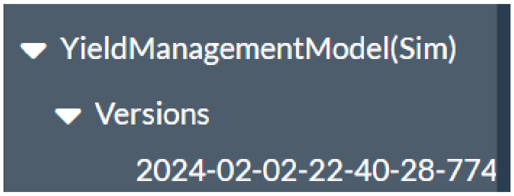
	F	G	H	I	J	K	L	M	N	O	P	Q	R
26	Ticket Price	$ 200.00											
27	Flight Capacity	100											
28	Number of no-shows	10	10.19149143	10.39306095	10.59591132	10.79294937	11.01643299	11.20704368	11.39197932	11.6506769	11.79120919	11.9930823	
29			10	10	11	11	11	11	11	12	12	12	
30	Refund to no-shows	50%											
31	Overbooking Compensation	125%											
32													
33	Tickets Sold	*100*	102	104	106	108	110						
34	Number of Customers showin	90	92	94	95	97	99						
35	Number of Overbooked ticket	0	0	0	0	0	0						
36													
37	**Total Revenue**	$ 19,000.00	$ 19,380.85	$ 19,760.69	$ 20,140.41	$ 20,520.71	$ 20,898.36	$ 21,029.					
38													
39	**Expected Revenue**	$18,980.70	$19,380.60	$19,746.19	$20,055.66	$20,286.96	$20,442.61	$20,541					
40													
41													
42													
43													
44													
45													
46													
47													
48													
49													
50													
51													
52													
53													

Model Name * YieldManagementModel(Sim)

Please enter a valid model name.

Immediately a browser opens displaying the converted model in the Editor tab of www.rason.com.

This model appears in the Model list. Note that each time a model is POSTed to the RASON server, a unique resource ID is created beneath the modelName, in this case YieldManagementModel(Sim).



YieldManagementModel(Sim)
  Versions
    2024-02-02-22-40-28-774

The converted RASON model is below.

The model begins with the modelName (name of the RASON model) , modelSettings (includes the worksheet name) and engineSettings (where engine options are set) properties.

The uncertainVariables section contains the uncertain variables in the model.

The data section contains the constant parameters in the model.

```
{
  "comment": "This model has been generated by Psi from an Excel model
  in the workbook YieldManagementModel2_1(Sim).xlsx",
    "modelName": "YieldManagementModel(Sim)",
    "modelDescription": "",
    "modelSettings": {
      "worksheets": ["Yield Management Model 2"],
      "activeSheet": "Yield Management Model 2"
    },
    "engineSettings": {
      "randomGenerator": "Lecuyer",
      "samplingMethod": "Latin"
    },
    "uncertainVariables": {
      "h28": {
        "formula": "PsiLogNormal(0.1*H33,0.06*H33)"
      },
      "i28": {
        "formula": "PsiLogNormal(0.1*I33,0.06*I33)"
      },
      "j28": {
        "formula": "PsiLogNormal(0.1*J33,0.06*J33)"
      },
      "k28": {
        "formula": "PsiLogNormal(0.1*K33,0.06*K33)"
      },
      "l28": {
        "formula": "PsiLogNormal(0.1*L33,0.06*L33)"
      },
      "m28": {
        "formula": "PsiLogNormal(0.1*M33,0.06*M33)"
      },
      "n28": {
        "formula": "PsiLogNormal(0.1*N33,0.06*N33)"
      },
      "o28": {
        "formula": "PsiLogNormal(0.1*O33,0.06*O33)"
      },
      "p28": {
        "formula": "PsiLogNormal(0.1*P33,0.06*P33)"
      },
      "q28": {
        "formula": "PsiLogNormal(0.1*Q33,0.06*Q33)"
      }
    },
    "data": {
      "h33": {
        "value": 102
      },
      "i33": {
        "value": 104
      },
      "j33": {
        "value": 106
      },
      "k33": {
        "value": 108
    },
    "l33": {
      "value": 110
    },
    "m33": {
      "value": 112
    },
    "n33": {
      "value": 114
    },
    "o33": {
      "value": 116
    },
    "p33": {
      "value": 118
    },
    "q33": {
      "value": 120
    },
    "h33:q33": {
      "value": [
        [102, 104, 106, 108, 110, 112, 114, 116, 118, 120]
      ]
    },
    "g26": {
      "value": 200
    },
    "g30": {
      "value": 0.5
    },
    "h28:q28": {
      "value": [
        [10.2, 10.4, 10.6, 10.8, 11.0, 11.2, 11.4, 11.6, 11.8, 12]
      ]
    },
    "g31": {
      "value": 1.25
    },
    "g27": {
      "value": 100
    },
    "h35:q35": {
      "value": [
        [0, 0, 0, 0, 0, 1, 3, 4, 6, 8]
      ]
    },
    "g33": {
      "value": 100
    }
  },
  "formulas": {
    "h29": {
          "formula": "ROUND(H28,0)"
    },
    "h34": {
          "formula": "H33-H29"
    },
```

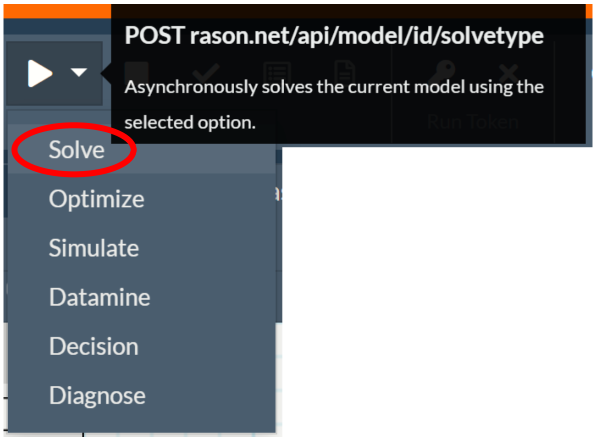
The formulas section calculates the intermediate formulas used in the uncertain functions.

```
"h35": {
      "formula": "MAX(0,H34-$G$27)"
},
"i29": {
      "formula": "ROUND(I28,0)"
},
"i34": {
      "formula": "I33-I29"
},
"i35": {
      "formula": "MAX(0,I34-$G$27)"
},
"j29": {
      "formula": "ROUND(J28,0)"
},
"j34": {
      "formula": "J33-J29"
},
"j35": {
      "formula": "MAX(0,J34-$G$27)"
},
"k29": {
      "formula": "ROUND(K28,0)"
},
"k34": {
      "formula": "K33-K29"
},
"k35": {
      "formula": "MAX(0,K34-$G$27)"
},
"l29": {
      "formula": "ROUND(L28,0)"
},
"l34": {
      "formula": "L33-L29"
},
"l35": {
      "formula": "MAX(0,L34-$G$27)"
},
"m29": {
      "formula": "ROUND(M28,0)"
},
"m34": {
    "formula": "M33-M29"
},
"m35": {
     "formula": "MAX(0,M34-$G$27)"
},
"n29": {
      "formula": "ROUND(N28,0)"
},
"n34": {
      "formula": "N33-N29"
},
"n35": {
      "formula": "MAX(0,N34-$G$27)"
},
    "o29": {
          "formula": "ROUND(O28,0)"
    },
    "o34": {
          "formula": "O33-O29"
    },
    "o35": {
          "formula": "MAX(0,O34-$G$27)"
    },
    "p29": {
          "formula": "ROUND(P28,0)"
    },
    "p34": {
          "formula": "P33-P29"
    },
    "p35": {
          "formula": "MAX(0,P34-$G$27)"
    },
    "q29": {
          "formula": "ROUND(Q28,0)"
    },
    "q34": {
          "formula": "Q33-Q29"
    },
    "q35": {
          "formula": "MAX(0,Q34-$G$27)"
    },
    "g28": {
          "formula": "ROUND(H28,0)"
    },
    "g34": {
          "formula": "G33-G28"
    },
    "g35": {
          "formula": "MAX(0,G34-$G$27)"
    }
  },
  "uncertainFunctions": {
    "for(i in 1..10)": {
      "h37:q37[i]": {
        "formula": "(H33:Q33[,i]*$G$26)-($G$30*$G$26*H28:Q28[,i]) -
        ($G$31*$G$26*H35:Q35[,i]) + PsiOutput()",
        "aliasName": "theOutputs",
        "trials": [],
        "mean": []
      }
    },
    "g37": {
      "formula": "(G33*$G$26)-($G$30*$G$26*G28) - ($G$31*$G$26*G35)",
      "mean": []
    }
  }
}
```

The uncertainFunction section holds the uncertain functions in the RASON model. Note that all 10 uncertain functions are generated using a for loop.

Note: When converting a simulation model into the RASON modeling language note that only statistics with explicit numeric arguments such as =PsiPercentile(cell, 0.50) or =PsiPercentile(cell, A1) where A1 = 0.05 are supported. Expressions passed to arguments such as =PsiPercentile(cell, A1+A2), where A1 = .1 and A2 = .3 are not supported.

Click the arrow next to the ▶▾ icon and select Simulate (or Solve) from the Solve drop down menu to call the RASON endpoint, **POST rason.net/api/model/{nameorid}/simulate**. The model is submitted to the RASON Server which solves the model and returns the result in the Output window, shown below.



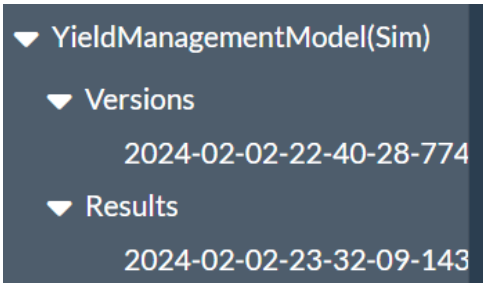
POST rason.net/api/model/id/solvetype

Asynchronously solves the current model using the selected option.

"Solve" calls the REST API endpoint POST rason.net/api/model/YieldManagement(Sim)/solve, which solves any problem type (optimization, simulation, decision table or data science) while Simulate calls the REST API endpoint POST rason.net/api/model/ YieldManagement(Sim)/simulate which solves only simulation models. Either one may be called to perform the simulation.

In the Output Window at the bottom of the screen you'll see a similar output.

The results, in JSON, are inserted into the Models pane beneath "Results".



Model ID identifies this instance of the YieldManagement(Sim)model. The ParentModelId is the model that was solved to create this instance. For more information on the Rason REST API, see the Rason User Guide downloadable from the Help page.

If we click the entry under Results, the Editor tab will open and display the results of the solve in JSON.

As you can see, Analytic Solver users have the ability to build an optimization/simulation/decision table model in Microsoft Excel and deploy it to Frontline's Rason server *without ever having to write a single line of code*. From here, users can publish, manage and govern analytic models as RESTful decision services, easily update them with new data and solve them, and use the results in any application that can consume JSON or OData.

Rason is the easiest way to transition into the world of web, mobile and cloud-base applications. For more information on using Frontline's Rason Services or to see how a Rason model looks for a simulation, decision table or data science model, see www.Rason.com or simply contact our Technical Support staff using support@solver.com.

Cloud Service – Excel

What if you had a large optimization, simulation or decision table model, or a workflow with a combination of each, with literally hundreds of thousands of decision variables/uncertain variables and constraints/uncertain functions? A translated Rason model could be too large and unwieldy to maintain, or maybe you just feel more comfortable maintaining your model in Excel? That's no problem because with Analytic Solver 20.5, it's possible to deploy your model to the Rason Server and maintain it in Excel. You can do so by using the Cloud Service – Excel option on the Deploy Model menu.

To get started, click back to Help – Example Models on the ribbon, click Monte Carlo Simulation Examples and then the Business Forecast link. Click the Forecast with Uncertainty tab. Recall this example model from the Simulation & Risk Analysis chapter which finds the True Average of a sales forecast.

This example includes just one Statistic function, PsiMean, in cell D20. As a result, our results will only contain one result, the Mean of cell D20. We could add additional Psi Statistics functions such as PsiPercentiles, PsiStdDev, etc., if desired in order to add these statistics to our results in Rason.

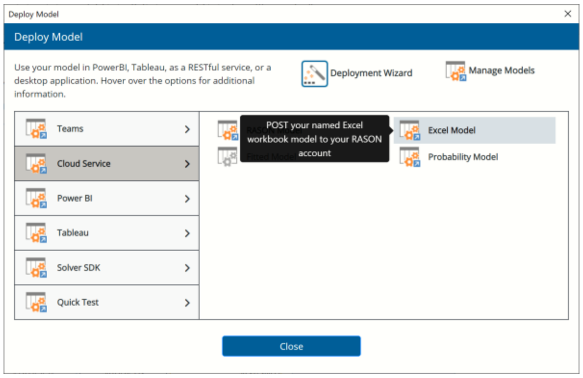
Before we click Deploy Model, let's use the PsiInput function to expose a few input parameters which we might expect will change later. For example, let's expose the product cost per scenario and also the maximum market demand, which, due to unforeseen circumstances, might change in the future.

To do so, click a blank cell on the worksheet, say G24 and enter =PsiInput(J16) and in cell G25 enter =PsiInput(H20:J20).

Note: You can also define cell J16 and cells H20:J20 using defined names such as MaxDemand (J16) and Cost (cells H20:J20) and then use PsiInput(MaxDemand) and PsiInput(Cost).

If running an optimization model, there are 6 output Psi functions available that you could enter into the spreadsheet: PsiFinalValue, PsiInitialValue, PsiSlackValue, PsiDualLower, PsiDualUpper and PsiDualValue. These may all be used to obtain the specified value of a constraint or variable in the result. For more information on these functions, see the Psi Functions chapter within the Analytic Solver Reference Guide.

Click Deploy Model – Cloud Service – Excel Model

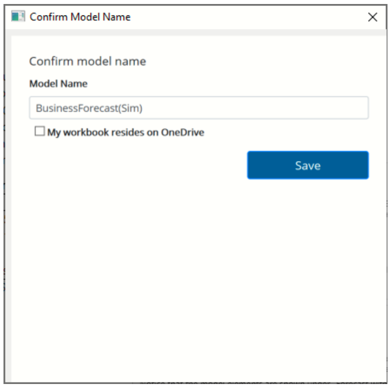


Deploy Model

Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

POST your named Excel workbook model to your RASON account

then click Save to use the default model name. This option will simply POST the Excel model to your Rason account. (If your model is saved on your OneDrive account, click the checkbox.)



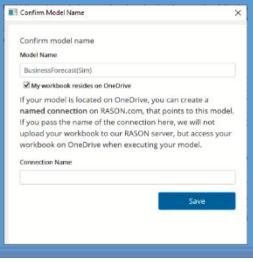
Confirm Model Name

Confirm model name

Model Name: BusinessForecast(Sim)

☐ My workbook resides on OneDrive

If your model is saved on your OneDrive account, click the "My workbook resides on OneDrive" checkbox to expand the dialog and enter your Connection Name (File name).



Confirm Model Name

Confirm model name

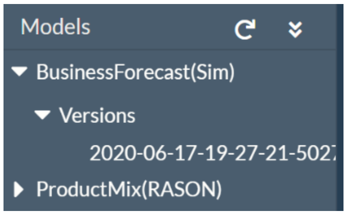
Model Name: BusinessForecast(Sim)

☑ My workbook resides on OneDrive

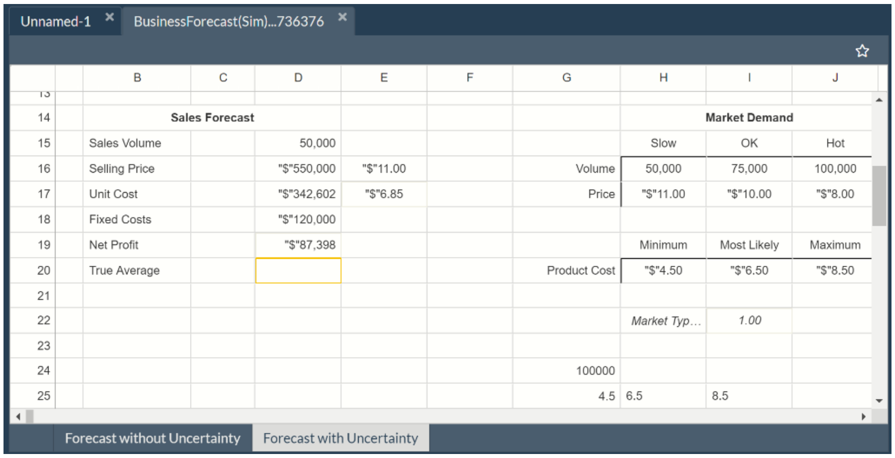
If your model is located on OneDrive, you can create a **named connection** on RASON.com, that points to this model. If you pass the name of the connection here, we will not upload your workbook to our RASON server, but access your workbook on OneDrive when executing your model.

Connection Name:

Notice that the BusinessForecast(Sim) model now appears in the Models pane on www.rason.com.



Models
- BusinessForecast(Sim)
  - Versions
    - 2020-06-17-19-27-21-502
- ProductMix(RASON)



Unnamed-1
BusinessForecast(Sim)...736376

	B	C	D	E	F	G	H	I	J
14		Sales Forecast						Market Demand	
15	Sales Volume		50,000				Slow	OK	Hot
16	Selling Price		"$"550,000	"$"11.00		Volume	50,000	75,000	100,000
17	Unit Cost		"$"342,602	"$"6.85		Price	"$"11.00	"$"10.00	"$"8.00
18	Fixed Costs		"$"120,000						
19	Net Profit		"$"87,398				Minimum	Most Likely	Maximum
20	True Average					Product Cost	"$"4.50	"$"6.50	"$"8.50
21									
22							Market Typ…	1.00	
23									
24						100000			
25						4.5	6.5	8.5	

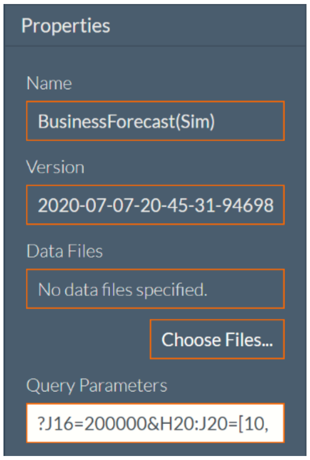
Forecast without Uncertainty
Forecast with Uncertainty

Under Properties on the left, we will use the Query Parameters edit box to change our Cost parameters to, say, Minimum = $10, Most Likely = $15 and Maximum = $20 and the Maximum sold to 200,000.

?J16=200000&H20:J20=[10, 15, 20]

Note:"?worksheet=Forecast with Uncertainty&J16=200000&H20:J20=[10, 15, 20]" is also supported.

Note that if we use PsiInput() to designate multiple cells as input parameters, it is not possible to change just 1 of the input parameters, you must pass all or none. If you want to change just one parameter, use PsiInput() for just the single parameter as we did in cell G24, = PsiInput(J16)



Name

BusinessForecast(Sim)

Version

2020-07-07-20-45-31-94698

Data Files

No data files specified.

Query Parameters

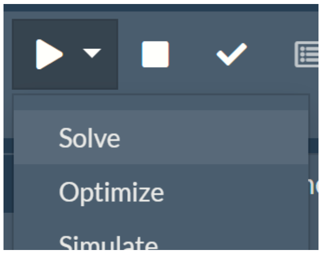
?J16=200000&H20:J20=[10,

Note: If we had used the defined names, our query would have been:

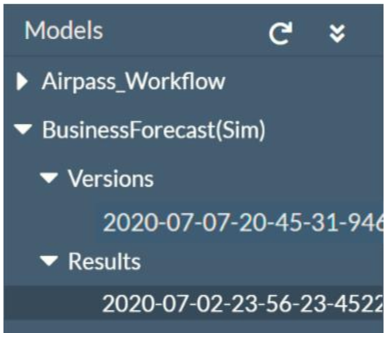
?MaxDemand=200000&Cost=[10,15,20] or

?worksheet=Forecast with Uncertainty&MaxDemand=200000&Cost=[10,15,20]

To run a new simulation with these updated parameters, click the down arrow next to the play button and click Solve.



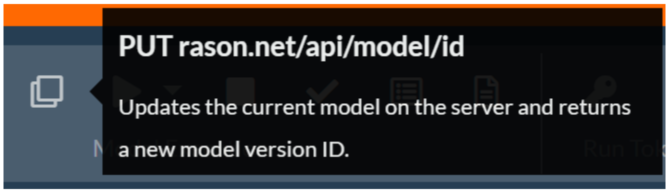
The results will appear under Results in the Models pane on the right.



Click the new result entry to view the new simulation results.

```
{
    "status": {
        "id": "2590+BusinessForecast(Sim)+2020-07-02-
         23-56-23-452286",
        "code": 0,
        "codeText": "Solver has completed the
         simulation.",
        "solveTime": 58
    },
    "results": {
        "$D$19.statistics": ["mean"]
    },
    "$D$19": {
        "statistics": {
            "objectType": "dataFrame",
            "name": "$D$19.statistics",
            "order": "col",
            "colNames": ["mean"],
            "colTypes": ["double"],
            "indexCols": null,
            "data": [
                [-781671.43164415448]
            ]
        }
    }
}
```

If you had multiple parameters to update, then you could enter the updated values into a spreadsheet or text file and then use the Choose Files button to upload the new data to the Rason Cloud Service. Simply click the Put icon to update your model with the attached data sheet and then click Solve to solve your model with the latest data.



PUT rason.net/api/model/id

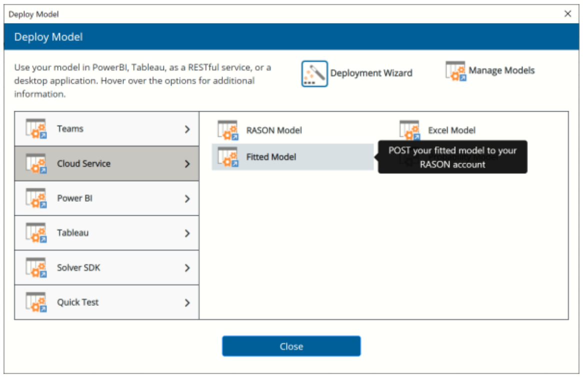
Updates the current model on the server and returns a new model version ID.

Cloud Service Fitted Model

The majority of Analytic Solver Data Science routines generate fitted, or stored, models which are used to score (classify, predict or forecast) new data points. Selecting Cloud Service – Fitted Model POSTs the fitted model to your account on the RASON server allowing users to utilize their fitted models to score new data on the RASON Server.

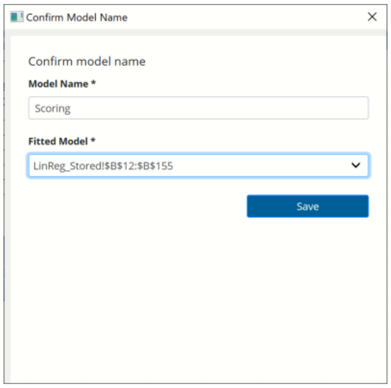
On the Analytic Solver ribbon, click Help – Example Models – Forecasting / Data Science Examples. Scroll down and click the Scoring example link to open Scoring.xlsx. This workbook contains a stored data science model, LinReg_Stored.

Click Deploy Model – Cloud Service – Fitted Model. The Confirm Model Name dialog opens.



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

POST your fitted model to your RASON account



Confirm model name

Model Name *

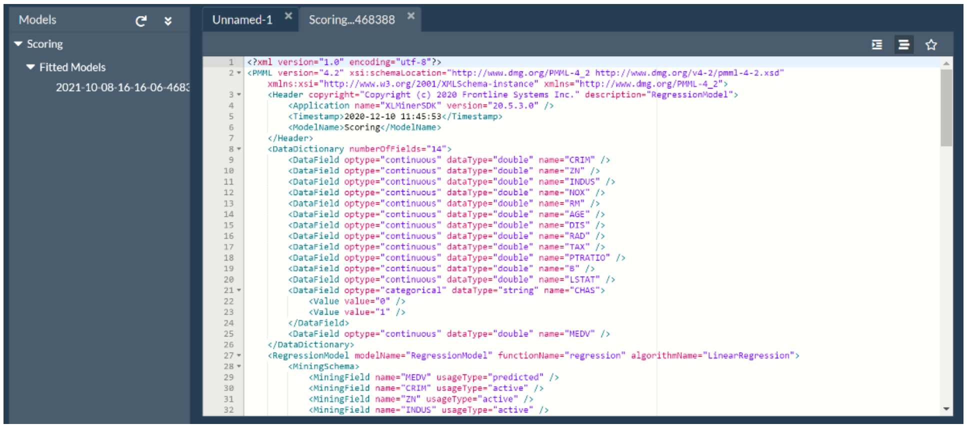
Scoring

Fitted Model *

LinReg_Stored!$B$12:$B$155

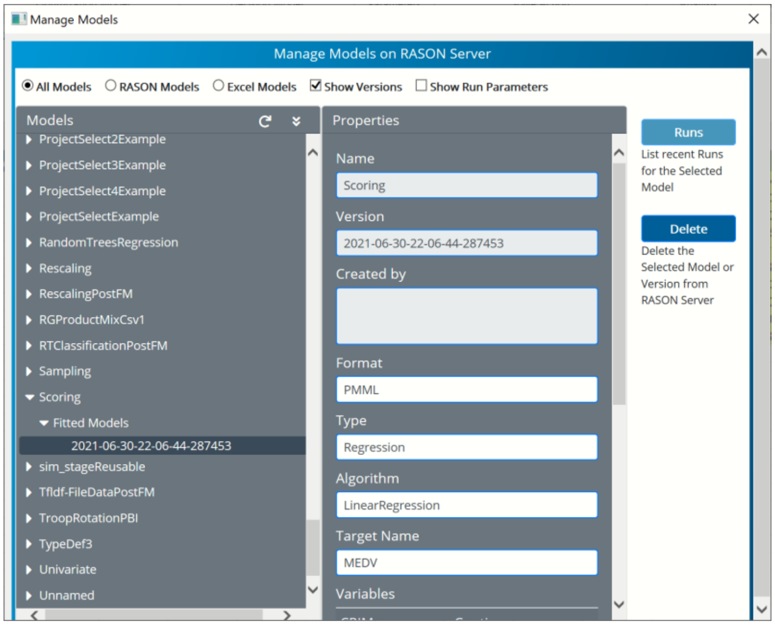
- Enter a Model name. This example will use the default, Scoring.
- Click the down arrow beneath Fitted Model to select the fitted model contained within the workbook, LinReg_Stored!$B$12:$B$155.
- Click Save to POST the fitted model to your RASON account.

From here you can maintain the fitted model through RASON Decision Services on RASON.com.



```xml
<?xml version="1.0" encoding="utf-8"?>
<PMML version="4.2" xsi:schemaLocation="http://www.dmg.org/PMML-4_2 http://www.dmg.org/v4-2/pmml-4-2.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.dmg.org/PMML-4_2">
    <Header copyright="Copyright (c) 2020 Frontline Systems Inc." description="RegressionModel">
        <Application name="XLMinerSDK" version="20.5.3.0" />
        <Timestamp>2020-12-10 11:45:53</Timestamp>
        <ModelName>Scoring</ModelName>
    </Header>
    <DataDictionary numberOfFields="14">
        <DataField optype="continuous" dataType="double" name="CRIM" />
        <DataField optype="continuous" dataType="double" name="ZN" />
        <DataField optype="continuous" dataType="double" name="INDUS" />
        <DataField optype="continuous" dataType="double" name="NOX" />
        <DataField optype="continuous" dataType="double" name="RM" />
        <DataField optype="continuous" dataType="double" name="AGE" />
        <DataField optype="continuous" dataType="double" name="DIS" />
        <DataField optype="continuous" dataType="double" name="RAD" />
        <DataField optype="continuous" dataType="double" name="TAX" />
        <DataField optype="continuous" dataType="double" name="PTRATIO" />
        <DataField optype="continuous" dataType="double" name="B" />
        <DataField optype="continuous" dataType="double" name="LSTAT" />
        <DataField optype="categorical" dataType="string" name="CHAS">
            <Value value="0" />
            <Value value="1" />
        </DataField>
        <DataField optype="continuous" dataType="double" name="MEDV" />
    </DataDictionary>
    <RegressionModel modelName="RegressionModel" functionName="regression" algorithmName="LinearRegression">
        <MiningSchema>
            <MiningField name="MEDV" usageType="predicted" />
            <MiningField name="CRIM" usageType="active" />
            <MiningField name="ZN" usageType="active" />
            <MiningField name="INDUS" usageType="active" />
```

The fitted model will also be listed in Manage Models.



Name: Scoring
Version: 2021-06-30-22-06-44-287453
Created by:
Format: PMML
Type: Regression
Algorithm: LinearRegression
Target Name: MEDV
Variables

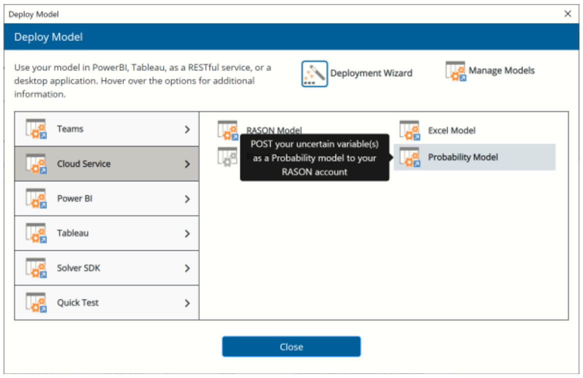
Cloud Service – Probability Model

As discussed in the previous chapter Examples: Simulation and Risk Analysis, simulation models attempt to model real life *uncertain* situations. Uncertainty is introduced to a model by the use of uncertain variables as uncertain input parameters. An example would be using the PsiTriangular() uncertain variable to model the economic state of the coming year, i.e. a good economy with lots of demand for a product or a slow economy with reduced demand.

Users can deploy and share probability models, following the open Probability Management 3.0 standard. Using these Shared Information Probability resources (SIPs, SLURPs and DISTs), users can ensure that their group or organization uses consistent data about uncertain/risky variables across simulation or decision models, enabling model results to be meaningfully compared.

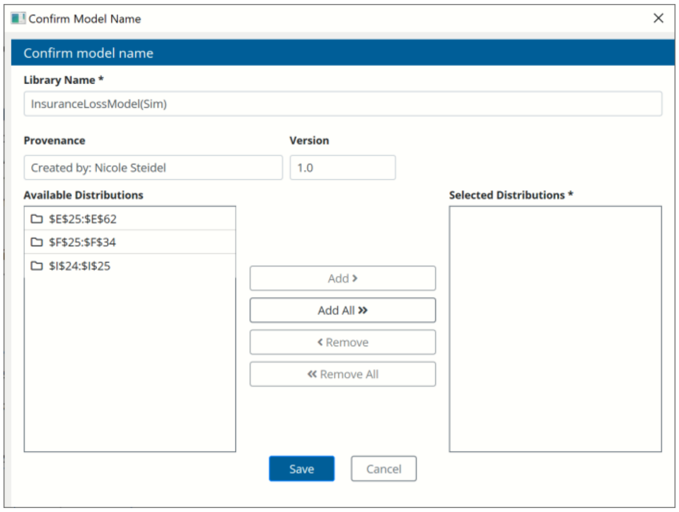
Open the Insurance Loss model (Help – Examples Models – Monte Carlo Simulation Examples). This model uses two different uncertain variables to model the financial impact of insurance claims for two different sets of customers. Insurance claims for the first set of customers follow an exponential distribution (PsiExponential) and claims for the second set of customers follow the Pearson (PsiPearson) distribution.

Click Deploy Model – Cloud Service **--** Probability Model to export both distributions to RASON using the 3.0 SIPmath standard.



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

POST your uncertain variable(s) as a Probability model to your RASON account



Library Name *: InsuranceLossModel(Sim)

Provenance: Created by: Nicole Steidel

Version: 1.0

Available Distributions:
- $E$25:$E$62
- $F$25:$F$34
- $I$24:$I$25

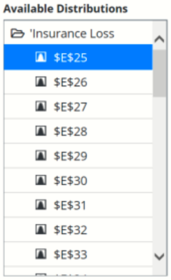
The Confirm Model Name dialog opens.

- Enter a new name for Library Name or keep the existing name. This is the name of the distribution library that will be exported into the RASON account.
- Leave the entries for "Provenance" and "Version" as is or enter new entries for one or both.
- All available uncertain distributions in the workbook appear under Available Distributions. Click the Add All button to add all uncertain distributions into the RASON distribution library.

  It's possible to add distributions into the library cell by cell. Select the first distribution in the list, Insurance Loss Model! E25:D62 to expand the range, select the desired cell containing the distribution and click the Add button. To add multiple distributions at once, hold down the CTRL key, select the desired cells, and click Add.

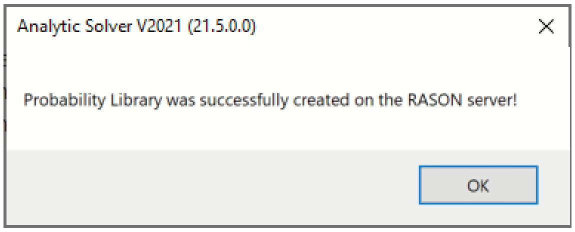
  To remove a distribution from the Selected Distributions window, highlight the cell, or cells, to be removed and click Remove.

  To remove all distributions from the Selected Distributions window, click the Remove All button.



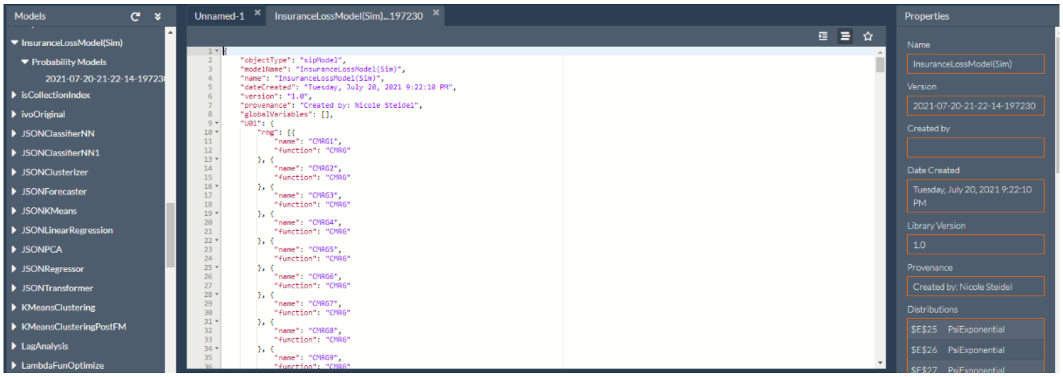
- Once the desired distributions have been added to the Selected Distributions window, click Save. (Click Cancel to close the dialog. No distributions will be added to the library.)

Immediately, a browser opens to www.RASON.com and the distribution library is POSTed to the RASON account. In Excel, you'll see the following message. Click OK to close.


Analytic Solver V2021 (21.5.0.0)

Probability Library was successfully created on the RASON server!

From here, these distributions may be used in a RASON or Excel model. To see an example of how to use this distribution library in a RASON model, see the RASON User Guide, downloaded from www.RASON.com on the Help tab.



```
{
    "objectType": "sipModel",
    "modelName": "InsuranceLossModel(Sim)",
    "name": "InsuranceLossModel(Sim)",
    "dateCreated": "Tuesday, July 20, 2021 9:22:10 PM",
    "version": "1.0",
    "provenance": "Created by: Nicole Steidel",
    "globalVariables": [],
    "U01": {
        "rng": [{
            "name": "CMRG1",
            "function": "CMRG"
        }, {
            "name": "CMRG2",
            "function": "CMRG"
        }, {
            "name": "CMRG3",
            "function": "CMRG"
        }, {
            "name": "CMRG4",
            "function": "CMRG"
        }, {
            "name": "CMRG5",
            "function": "CMRG"
        }, {
            "name": "CMRG6",
            "function": "CMRG"
        }, {
            "name": "CMRG7",
            "function": "CMRG"
        }, {
            "name": "CMRG8",
            "function": "CMRG"
        }, {
            "name": "CMRG9",
```

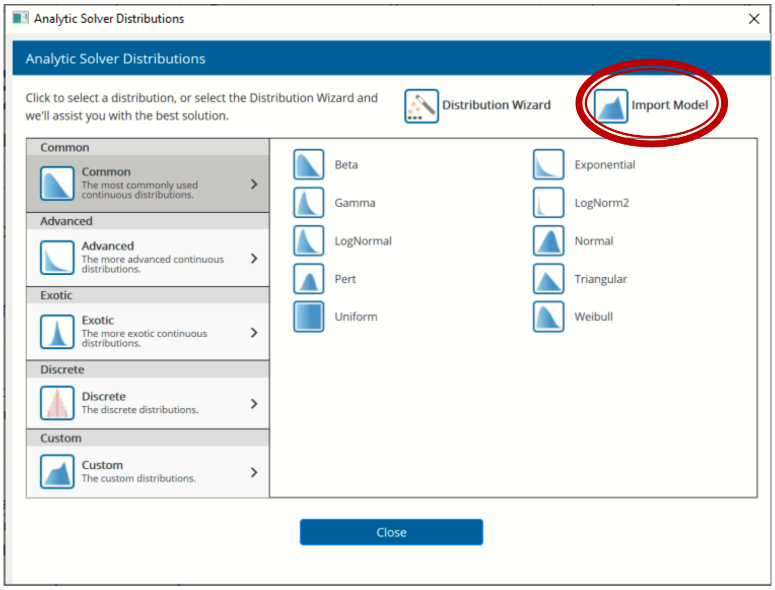
Name: InsuranceLossModel(Sim)
Version: 2021-07-20-21-22-14-197230
Created by:
Date Created: Tuesday, July 20, 2021 9:22:10 PM
Library Version: 1.0
Provenance: Created by: Nicole Steidel
Distributions:

$E$25	PsiExponential
$E$26	PsiExponential

Importing Distributions into Excel

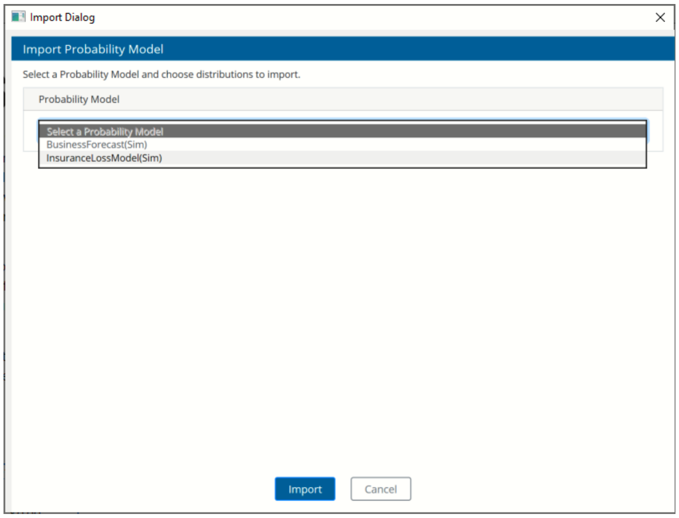
To import these distributions into a new Excel model, open a new worksheet and click Distributions on the Analytic Solver ribbon to open the Analytic Solver Distributions dialog.

Note: Only an Analytic Solver Comprehensive license supports the Import Model functionality. This button will be disabled for all other licenses: Analytic Solver Optimization/Simulation/Data Science, Analytic Solver Upgrade and Solver Academy.


Analytic Solver Distributions
Analytic Solver Distributions
Distribution Wizard
Import Model
Common
Common
The most commonly used continuous distributions.
Advanced
Advanced
The more advanced continuous distributions.
Exotic
Exotic
The more exotic continuous distributions.
Discrete
Discrete
The discrete distributions.
Custom
Custom
The custom distributions.
Beta
Exponential
Gamma
LogNorm2
LogNormal
Normal
Pert
Triangular
Uniform
Weibull
Close

Click to select a distribution, or select the Distribution Wizard and we'll assist you with the best solution.

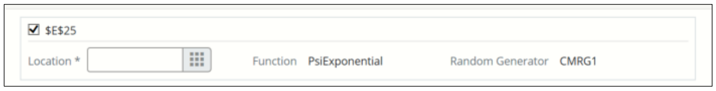
Click Import Model, top right, to open the Import Dialog. Select InsuranceLossModel(Sim) to import the distribution library from the model above. Click Import.



Select a Probability Model and choose distributions to import.

The Import Probability Model window opens displaying all the distributions that were exported to the InsuranceLossModel(Sim) distribution library.

Select the first distribution, E25. The entry expands to include a Location field.



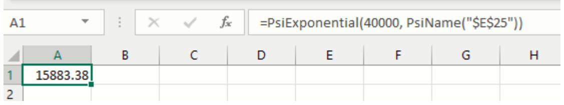
☑ $E$25

Location *

Function PsiExponential

Random Generator CMRG1

Enter the address of a blank cell, say cell A1, then scroll down to the bottom of the dialog and click Import.



=PsiExponential(40000, PsiName("$E$25"))

15883.38

The exported distribution for cell E25 from the Insurance Loss Model (Sim) is imported into cell A1 in the new worksheet.   This particular distribution is not very complicated, it uses the PsiExponential distribution with a mean of 40,000.

Open the College Fund Growth 1 model (Help – Example Models – Monte Carlo Simulation Examples) to see the export of a much more complicated distribution.

Click cell O17. This cell contains a PsiMetalog distribution with 14 terms! Exporting this function to a distribution library will prohibit the introduction of typos and mistakes when entering this distribution into a new model.

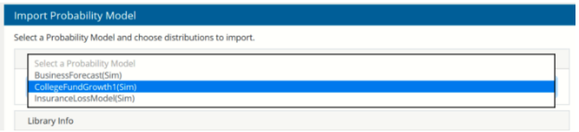
Click Deploy Model – Cloud Service – Probability Model to begin the export.

Exporting the Distribution

- On the Confirm Model Name dialog, leave Library Name, Provenance and Version fields at their defaults.
- Under Available Distributions, select College Fund Growth! O17 and click the Add button to add the distribution as a Selected Distribution.
- Click Save to export the distribution to the CollegeFundGrowth1(Sim) distribution library.

Importing the Distribution

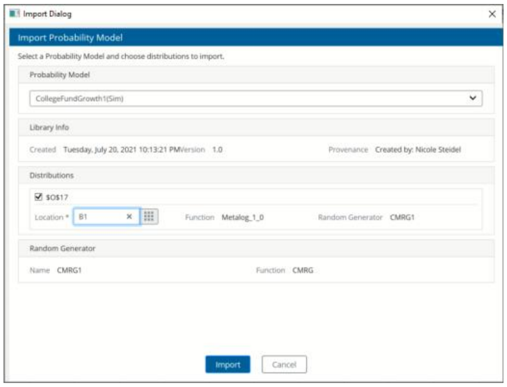
- Open a new worksheet and click Distributions – Import Model
- On the Import Probability Model dialog, select CollegeFundGrow1(Sim) from the drop down menu.



Import Probability Model

Select a Probability Model and choose distributions to import.

- Select O17 under Distributions and enter B1 for Location.



Import Dialog

Import Probability Model

Select a Probability Model and choose distributions to import.

Probability Model: CollegeFundGrowth1(Sim)

Library Info

Created Tuesday, July 20, 2021 10:13:21 PM Version 1.0 Provenance Created by: Nicole Steidel

Distributions

☑ $O$17

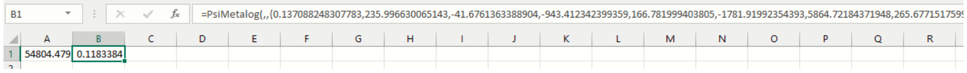
Location * B1 Function Metalog_1_0 Random Generator CMRG1

Random Generator

Name CMRG1 Function CMRG

- Then click Import.

The PsiMetalog distribution with all 14 terms is imported into cell B1.



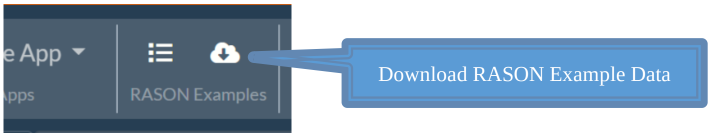
B1

=PsiMetalog(,,{0.137088248307783,235.996630065143,-41.6761363388904,-943.412342399359,166.781999403805,-1781.91992354393,5864.72184371948,265.677151759

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	54804.479	0.1183384																

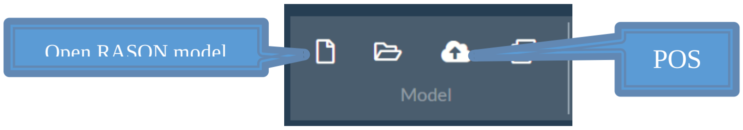
Importing Distributions from a Third Party

It's possible to import a third-party distribution library, saved in SipMath Standard 3.0, into a simulation or stochastic model. The distribution library, Lib_1_ Basic_w_Seed3_Param_for_Variable2.SIPmath created by [Probability Management](), can be downloaded from the Editor tab at www. RASON.com by clicking Download RASON Example Data.



Download RASON Example Data

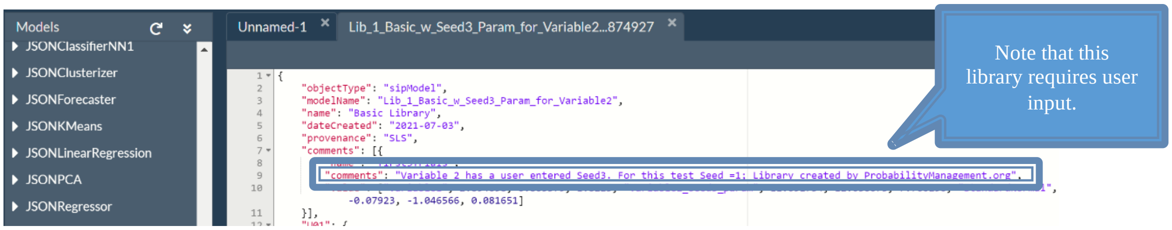
A zip file is downloaded when the cloud icon is clicked. Extract the contents of this file, then click the Open RASON icon, on the Editor tab ribbon, browse to the location of the extracted library, then click Open.



Open RASON model

POS

Afterwards, POST the distribution library to the RASON server by clicking the POST icon within the Model section of the Editor ribbon. The library appears on the Models pane, on the left.

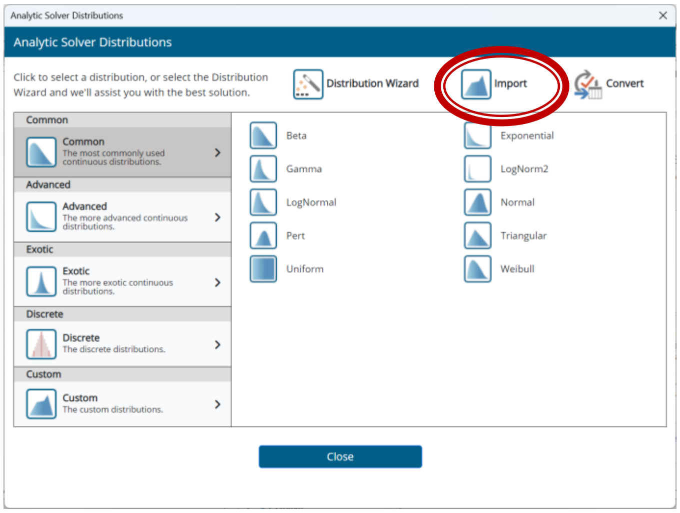


```
{
    "objectType": "sipModel",
    "modelName": "Lib_1_Basic_w_Seed3_Param_for_Variable2",
    "name": "Basic Library",
    "dateCreated": "2021-07-03",
    "provenance": "SLS",
    "comments": [{
      "comments": "Variable 2 has a user entered Seed3. For this test Seed =1; Library created by ProbabilityManagement.org",
          -0.07923, -1.046566, 0.081651]
    }],
```

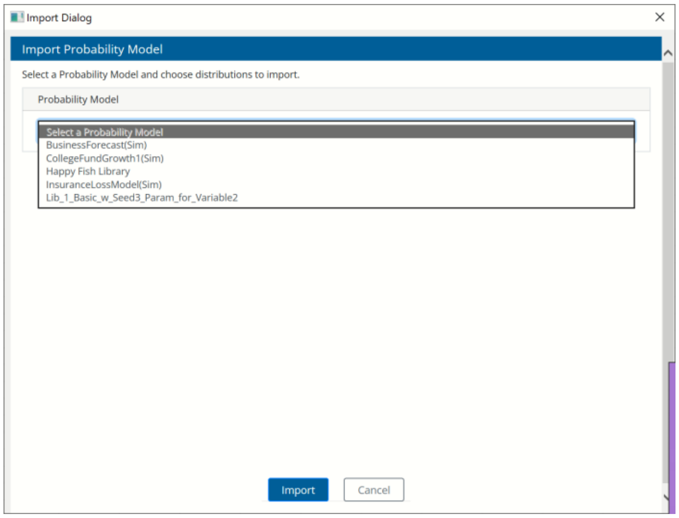
Note that this library requires user input.

Click back to Analytic Solver and open a new worksheet.

Click Distributions on the Analytic Solver Ribbon to open the Analytic Solver Distributions dialog.


Analytic Solver Distributions
Analytic Solver Distributions
Click to select a distribution, or select the Distribution Wizard and we'll assist you with the best solution.
Distribution Wizard
Import
Convert
Common
Common
The most commonly used continuous distributions.
Advanced
Advanced
The more advanced continuous distributions.
Exotic
Exotic
The more exotic continuous distributions.
Discrete
Discrete
The discrete distributions.
Custom
Custom
The custom distributions.
Beta
Exponential
Gamma
LogNorm2
LogNormal
Normal
Pert
Triangular
Uniform
Weibull
Close

Click Import Model on the Distributions dialog.



Select a Probability Model and choose distributions to import.

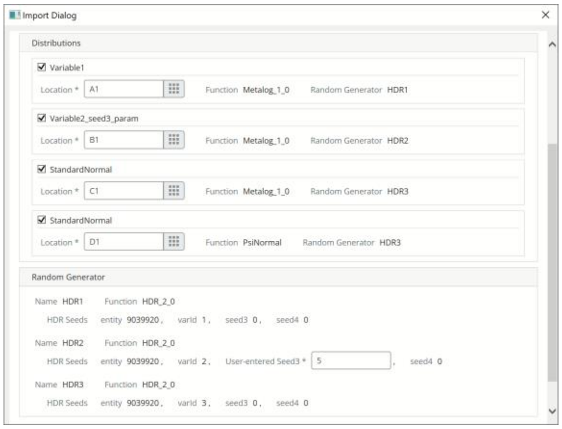
Probability Model

- Select a Probability Model
- BusinessForecast(Sim)
- CollegeFundGrowth1(Sim)
- Happy Fish Library
- InsuranceLossModel(Sim)
- Lib_1_Basic_w_Seed3_Param_for_Variable2

Click the down arrow and select Lib_1_Basic_w_Seed3_Param_for_Variable2. The dialog expands.

Select all distributions under Distributions: Variable1, Variable2_seed3_param and both StandardNormal. Enter a blank cell for each distribution as shown on the following dialog.

Beneath "Random Generator" enter "5" for "User-entered Seed 3". Then click Import.



☑ Variable1
Location * A1 Function Metalog_1_0 Random Generator HDR1

☑ Variable2_seed3_param
Location * B1 Function Metalog_1_0 Random Generator HDR2

☑ StandardNormal
Location * C1 Function Metalog_1_0 Random Generator HDR3

☑ StandardNormal
Location * D1 Function PsiNormal Random Generator HDR3

Name HDR1 Function HDR_2_0
HDR Seeds entity 9039920 , varId 1 , seed3 0 , seed4 0

Name HDR2 Function HDR_2_0
HDR Seeds entity 9039920 , varId 2 , User-entered Seed3 * 5 , seed4 0

Name HDR3 Function HDR_2_0
HDR Seeds entity 9039920 , varId 3 , seed3 0 , seed4 0

Use the PsiData() function to display all trials for each distribution. Enter the function as follows:

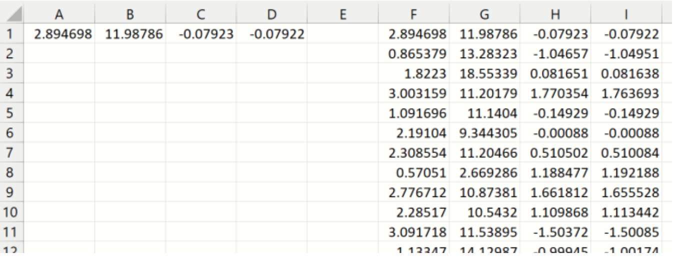
F1: =PsiData(A1)

G1: =PsiData(B1)

H1: = PsiData(C1)

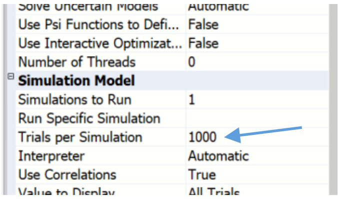
I1: =PsiData(D1)

The formulas will "spill" down to fill 1000 cells.



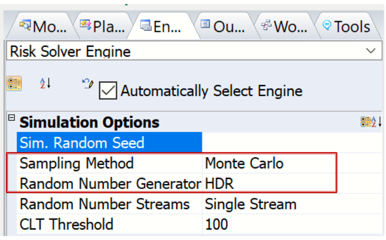
	A	B	C	D	E	F	G	H	I
1	2.894698	11.98786	-0.07923	-0.07922		2.894698	11.98786	-0.07923	-0.07922
2						0.865379	13.28323	-1.04657	-1.04951
3						1.8223	18.55339	0.081651	0.081638
4						3.003159	11.20179	1.770354	1.763693
5						1.091696	11.1404	-0.14929	-0.14929
6						2.19104	9.344305	-0.00088	-0.00088
7						2.308554	11.20466	0.510502	0.510084
8						0.57051	2.669286	1.188477	1.192188
9						2.776712	10.87381	1.661812	1.655528
10						2.28517	10.5432	1.109868	1.113442
11						3.091718	11.53895	-1.50372	-1.50085

You can change the number of trials on the Platform tab, under Simulation model.



Solve Uncertain Models	Automatic
Use Psi Functions to Defi...	False
Use Interactive Optimizat...	False
Number of Threads	0
Simulation Model	
Simulations to Run	1
Run Specific Simulation	
Trials per Simulation	1000
Interpreter	Automatic
Use Correlations	True
Value to Display	All Trials

Note that Analytic Solver will automatically change Sampling Method to "Monte Carlo" and Random Number Generator to "HDR" whenever Import Model functionality is invoked.

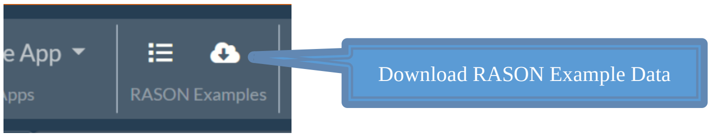


☑ Automatically Select Engine

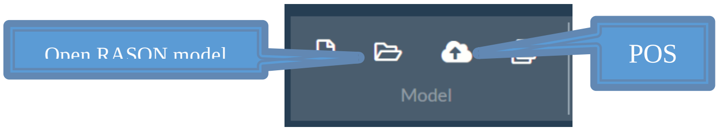
Simulation Options	
Sim. Random Seed	
Sampling Method	Monte Carlo
Random Number Generator	HDR
Random Number Streams	Single Stream
CLT Threshold	100

Third Party Distribution with Correlation Matrix

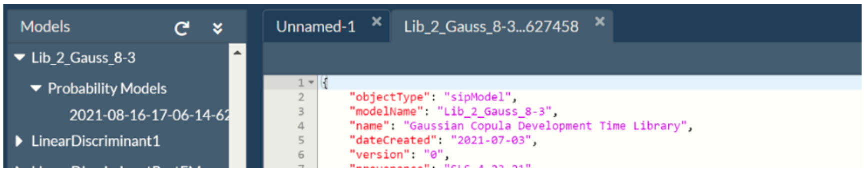
The distribution library, Lib_2_Gauss_8-3.SIPmath, also created by Probability Management, can be downloaded from the Editor tab at www. RASON.com by clicking Download RASON Example Data. This example probability model contains a correlation matrix.



As discussed earlier, a zip file is downloaded when the cloud icon is clicked. Extract the contents of this file, then click the Open RASON icon, on the Editor tab ribbon, browse to the location of the extracted library, then click Open.



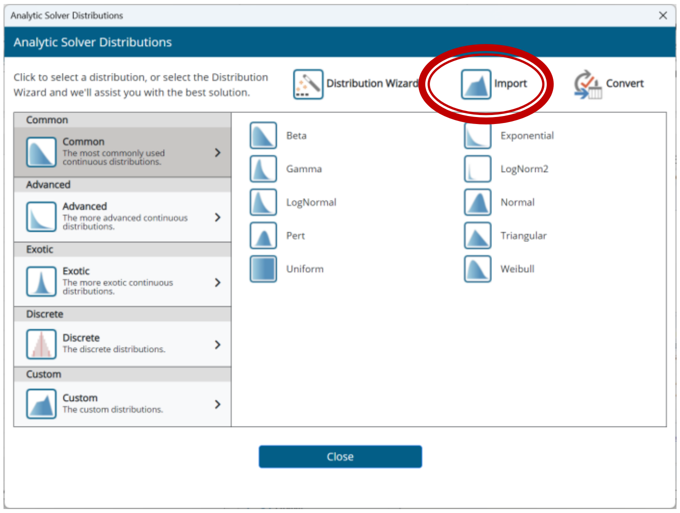
Afterwards, POST the distribution library to the RASON server by clicking the POST icon within the Model section of the Editor ribbon. The library appears on the Models pane, on the left.



```
{
    "objectType": "sipModel",
    "modelName": "Lib_2_Gauss_8-3",
    "name": "Gaussian Copula Development Time Library",
    "dateCreated": "2021-07-03",
    "version": "0",
```

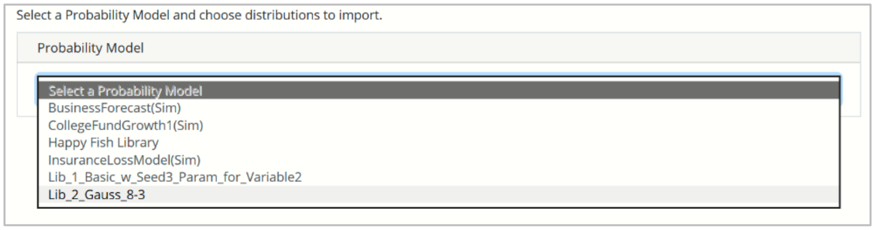
Click back to Analytic Solver and open a new worksheet.

Click Distributions on the Analytic Solver Ribbon to open the Analytic Solver Distributions dialog.


Analytic Solver Distributions
Analytic Solver Distributions
Distribution Wizard
Import
Convert
Common
Common
The most commonly used continuous distributions.
Advanced
Advanced
The more advanced continuous distributions.
Exotic
Exotic
The more exotic continuous distributions.
Discrete
Discrete
The discrete distributions.
Custom
Custom
The custom distributions.
Beta
Exponential
Gamma
LogNorm2
LogNormal
Normal
Pert
Triangular
Uniform
Weibull
Close

Click to select a distribution, or select the Distribution Wizard and we'll assist you with the best solution.

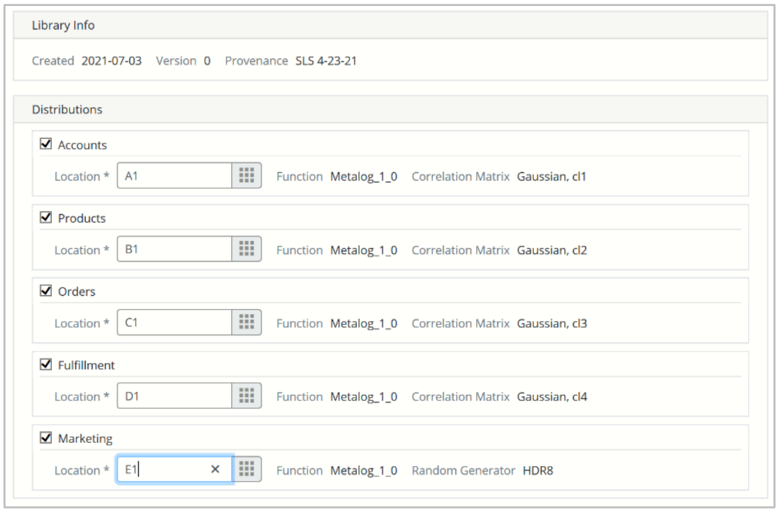
Click Import Model on the Distributions dialog.



Select a Probability Model and choose distributions to import.

Click the down arrow and select Lib_2_Gauss_8-3. The dialog expands.

Select all distributions under Distributions: Accounts, Products, Orders, Fulfillment and Marketing. Enter a blank cell for each distribution as shown on the following dialog.



Library Info

Created 2021-07-03 Version 0 Provenance SLS 4-23-21

Distributions

☑ Accounts

Location * A1 Function Metalog_1_0 Correlation Matrix Gaussian, cl1

☑ Products

Location * B1 Function Metalog_1_0 Correlation Matrix Gaussian, cl2

☑ Orders

Location * C1 Function Metalog_1_0 Correlation Matrix Gaussian, cl3

☑ Fulfillment

Location * D1 Function Metalog_1_0 Correlation Matrix Gaussian, cl4

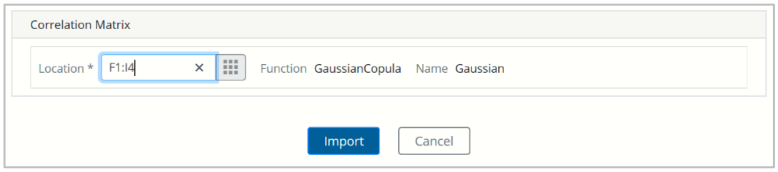
☑ Marketing

Location * E1 Function Metalog_1_0 Random Generator HDR8

Scroll down to the bottom of the dialog. Beneath "Correlation Matrix" enter a blank cell. (This is the cell where the correlation matrix will be imported.)

Enter the first cell, say "F1" and then a colon, i.e. "F1:" Analytic Solver will prefill the cell range to include a large enough location to include all elements of the matrix.

Since there are four uncertain variables in the correlation matrix (as shown in the RASON screenshot to the left) a 4 x 4 range is required. As a result, Analytic Solver prefills the range as "F1:I4".



Correlation Matrix

Location * F1:I4 Function GaussianCopula Name Gaussian

Then click Import.

To see all 1000 trial values for each distribution, use the PsiData() function. Enter the functions as follows:

A3: =PsiData(A1)

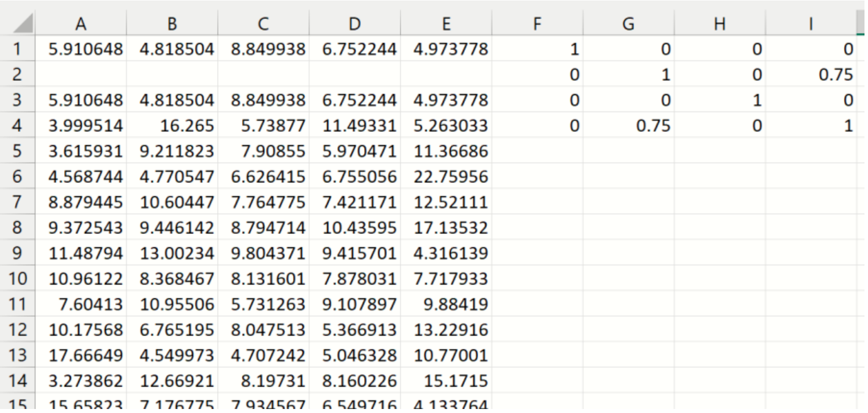
B3: =PsiData(B1)

C3: = PsiData(C1)

D3: =PsiData(D1)

E3 = PsiData(E1)

The formulas will "spill" down to fill 1000 cells. The correlation matrix is contained in cells F1:I4.



	A	B	C	D	E	F	G	H	I
1	5.910648	4.818504	8.849938	6.752244	4.973778	1	0	0	0
2						0	1	0	0.75
3	5.910648	4.818504	8.849938	6.752244	4.973778	0	0	1	0
4	3.999514	16.265	5.73877	11.49331	5.263033	0	0.75	0	1
5	3.615931	9.211823	7.90855	5.970471	11.36686				
6	4.568744	4.770547	6.626415	6.755056	22.75956				
7	8.879445	10.60447	7.764775	7.421171	12.52111				
8	9.372543	9.446142	8.794714	10.43595	17.13532				
9	11.48794	13.00234	9.804371	9.415701	4.316139				
10	10.96122	8.368467	8.131601	7.878031	7.717933				
11	7.60413	10.95506	5.731263	9.107897	9.88419				
12	10.17568	6.765195	8.047513	5.366913	13.22916				
13	17.66649	4.549973	4.707242	5.046328	10.77001				
14	3.273862	12.66921	8.19731	8.160226	15.1715				
15	15.65823	7.176775	7.934567	6.549716	4.133764				

Exceptions

There are several types of models that Frontline Excel Solvers are not able to convert to RASON models, i.e. Excel models containing errors such as #NUM, #VALUE, etc. For all unsupported features, the conversion generator returns an appropriate message.

Note: When converting a simulation model into the RASON modeling language note that only statistics with explicit numeric arguments such as =PsiPercentile(cell, 0.50) or =PsiPercentile(cell, A1) where A1 = 0.05 are supported. Expressions passed to arguments such as =PsiPercentile(cell, A1+A2), where A1 = .1 and A2 = .3 are not supported.

Problem Limits

Your Frontline Excel Solver license determines the size and type of model that can be solved with the RASON Web IDE. If you have a license for Analytic Solver Comprehensive, then the same problem limits will apply when solving a RASON model via the Rason Server, i.e. linear models with up to 8,000 variables (2,000 integers) and 8,000 constraints; smooth, nonlinear models with up to 1,000 variables and 1,000 constraints; nonsmooth models with up to 1,000 variables and 1,000 constraints and SOCP models with up to 2,000 variables and 8,000 constraints.

Exceptions

There are several types of models that Frontline Excel Solvers are not able to convert to RASON models, i.e. Excel models containing errors such as #NUM, #VALUE, etc. For all unsupported features, the conversion generator returns an appropriate message.

Note: When converting a simulation model into the RASON modeling language note that only statistics with explicit numeric arguments such as =PsiPercentile(cell, 0.50) or =PsiPercentile(cell, A1) where A1 = 0.05 are supported. Expressions passed to arguments such as =PsiPercentile(cell, A1+A2), where A1 = .1 and A2 = .3 are not supported.

Power BI

Clicking the thir0064 option on the Deploy Model dialog, Power BI, gives you three options, Managed Model, Embedded Model and Fitted Model.

- *Managed Model* translates the Excel optimization or simulation model to RASON and then POSTs this model to the RASON Server.
- *Excel Model* creates a named Excel optimization or simulation model that resides on the public RASON cloud service.
- *Fitted Model* POSTs the datamining or forecasting fitted model contained within your Excel workbook to your RASON account.

Note: This option only supports the creation of a Power BI Custom Visual with optimization and simulation models.

When you choose Deploy Model – Power BI – Managed Model, you must have previously used Deploy Model – Cloud Service to create a named model (either in Excel or Rason) that "lives" on the public Rason cloud service. You will be prompted for the name of that model. Analytic Solver will automatically create a "Custom Visual" that can be used on a Power BI dashboard or report. (This is saved as a file ending in .pbiviz, that you open in Power BI, adding it to your palette of visuals.) Re call that when Deploy Model – Cloud Service is

Analytic Solver offers a 2nd "quick" option that works for some (modest size) but not all Excel models, when you choose Deploy Model – Power BI – Embedded Model. Analytic Solver automatically translates your Excel model into Rason, our high-level modeling language that works in the cloud as well as on desktops and servers.

Analytic Solver then automatically creates a "Custom Power BI Visual" that can be used on a Power BI dashboard or report, where the RASON model is embedded inside the visual, in a .pbiviz file. You don't have to create a named model that "lives" on the public Rason cloud. The Custom Visual accepts the data your model needs from Power BI; whenever this data changes, it sends your Rason model plus this data to the public Rason cloud service where it is solved, gets back the results and displays them in bar chart form.

Installing Power BI

Power BI is Microsoft's desktop or cloud-based interactive data visualization business intelligence tool. In past versions of Analytic Solver, users have only been able to upload their model parameters (model data) to Power BI after solving an optimization or simulation model. Starting with V2017-R2, Analytic Solver can turn your Excel based optimization or simulation model into a Microsoft Power BI Custom Visual, *that is your full optimization or simulation model*, ready to accept power BI data, run on demand on the web, and display visual results in Power BI.

Note: Click this link to download Power BI for desktop.

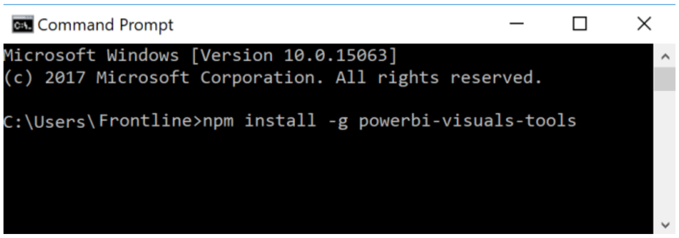
In order to use this new feature in Analytic Solver, you must first open desktop Power BI or the free cloud based version of Power BI. (Power BI is part of the office 365 suite.) For more information on this business tool, see the following website: https://powerbi.microsoft.com/en-us/documentation/powerbi-custom-visuals-getting-started-with-developer-tools/

It is important to note, that in order to create a custom visual in Power BI, you will first need to install Power BI's Developer Tools which is comprised of

NodeJS and the Power BI tools.   Follow the steps below to install both required items.

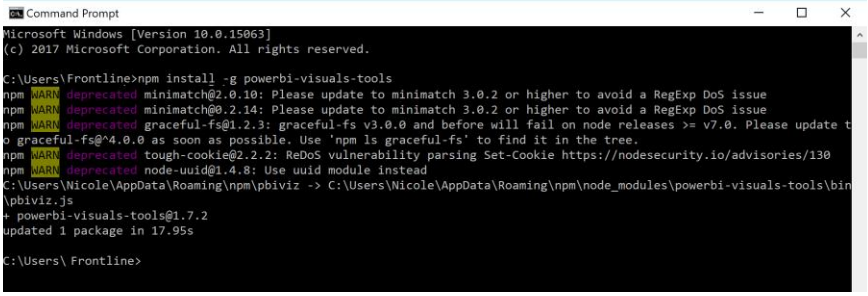
1. Click here to download NodeJS.
2. Click the Download button to download NodeJS for 64-bit Windows.
3. Once the NodeJS installer is downloaded, run the installer and follow the directions on the installer dialogs to install NodeJS onto your machine.
4. Install the command line tools by opening a command prompt and typing: "**npm install -g powerbi-visuals-tools**" as shown in the screenshot below.

   Note: If using V2017-R2, type: "npm install -g powerbi-visuals-tools@1.5".



```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Frontline>npm install -g powerbi-visuals-tools
```

After clicking Enter, you'll see the following results in the command prompt window.



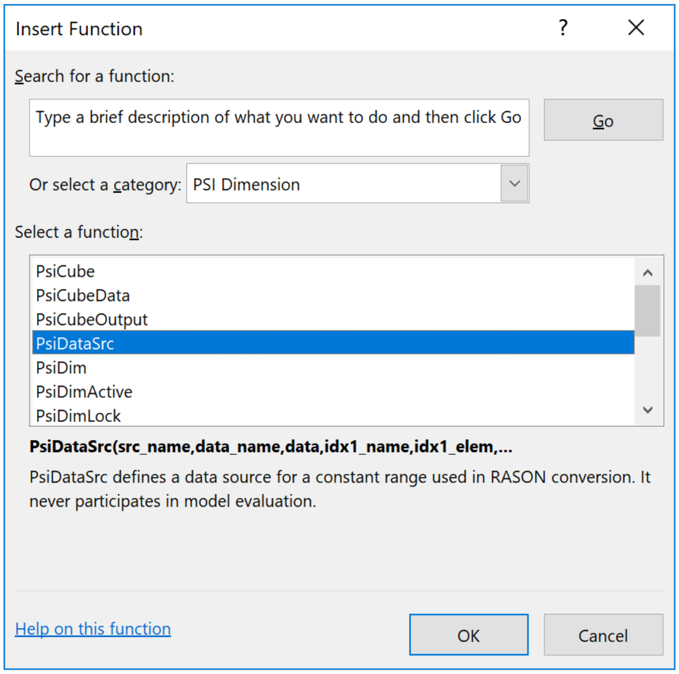
```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Frontline>npm install -g powerbi-visuals-tools
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated minimatch@0.2.14: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated graceful-fs@1.2.3: graceful-fs v3.0.0 and before will fail on node releases >= v7.0. Please update t
o graceful-fs@^4.0.0 as soon as possible. Use 'npm ls graceful-fs' to find it in the tree.
npm WARN deprecated tough-cookie@2.2.2: ReDoS vulnerability parsing Set-Cookie https://nodesecurity.io/advisories/130
npm WARN deprecated node-uuid@1.4.8: Use uuid module instead
C:\Users\Nicole\AppData\Roaming\npm\pbiviz -> C:\Users\Nicole\AppData\Roaming\npm\node_modules\powerbi-visuals-tools\bin
\pbiviz.js
+ powerbi-visuals-tools@1.7.2
updated 1 package in 17.95s

C:\Users\ Frontline>
```

Power BI Managed Model

Click back to the ProductMix(RASON).xlsx example model. Since we've already posted this model to the Rason cloud service within the previous section, we can immediately go to Deploy Model – Power BI – Managed Model. This Deploy Model option allows you to create a Data Well in Power BI which accepts input data.

Note: The menu choice Deploy Model – Cloud Service – Excel Model does not support the use of Data Wells in Power BI. You must first post your model to the RASON Server using Deploy Model – Cloud Service – Rason Model.

Select a blank cell, say J4. In Analytic Solver Desktop, click Formulas – Insert Function to open the Insert Function dialog, then select PSI Dimension for Category. Afterwards, highlight the function PsiDataSrc, and click OK.



Type a brief description of what you want to do and then click Go

Or select a category: PSI Dimension

PsiDataSrc(src_name,data_name,data,idx1_name,idx1_elem,...

PsiDataSrc defines a data source for a constant range used in RASON conversion. It never participates in model evaluation.

In Analytic Solver Desktop, you can simply type the formula directly into the cell.

The Function Arguments dialog opens displaying 5 arguments: Src_name, Val_col_names, Data, Idx1_name and Idx1_elem. We will use this dialog to create a PsiDataSrc() in cell J4 that will allow us the ability to change the inventory values and resolve the Product Mix example model not in Excel, but in Power BI.

PsiDataSrc() Arguments

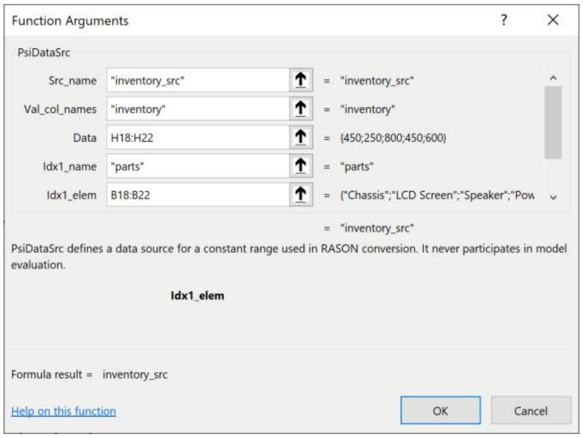
Src_name – This argument names the text file, created by the custom visual, which is submitted, along with the RASON model, to the RASON server. For this example, type: inventory_src.

Val_col_names – This is the name given to the "data well" that will appear in Power BI. This "data well" will hold the actual data. For this example, type: inventory.

Data – Enter the Excel range where the data is located on the spreadsheet for this argument. For this example, use your mouse to click the upward pointing arrow to the right of the Data argument and select cells H18:H22. You can also simply type this range into the edit box.

Idx1_name – Enter the name of the (1st) index set for the data specified in the Data argument. In this example, our Index set is simply the inventory available for each Part. Therefore, enter "parts" for this argument.

Idx1_elem – Enter the Excel range containing the data (or elements) for the first index set. For this example, either type or select cells: B18:B22



Function Arguments

PsiDataSrc

Src_name "inventory_src" = "inventory_src"
Val_col_names "inventory" = "inventory"
Data H18:H22 = {450;250;800;450;600}
Idx1_name "parts" = "parts"
Idx1_elem B18:B22 = {"Chassis";"LCD Screen";"Speaker";"Pow

= "inventory_src"

PsiDataSrc defines a data source for a constant range used in RASON conversion. It never participates in model evaluation.

Idx1_elem

Formula result = inventory_src

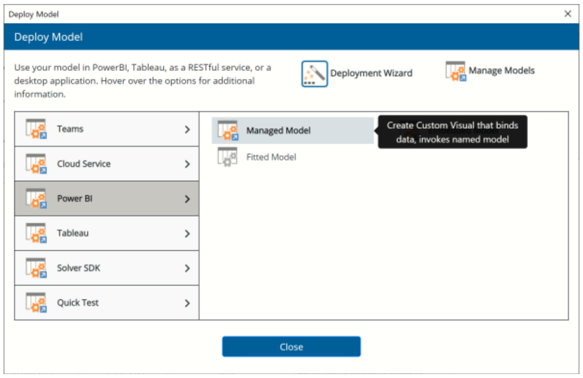
Since this index set is 1-dimensional, nothing further is needed and we can click OK to enter the PsiDataSrc() function into cell J4.

In Analytic Solver Cloud, you can type the formula directly into cell J4.

```
=PsiDataSrc("inventory_src", "inventory", 
Number_available, "parts", B18:B22)
```

Afterwards, cell J4 will display the value given to the Src_name argument, inventory_src.

Save the Excel model and then click Deploy Model – Power BI – Managed Model to create the Power BI Custom Visual.

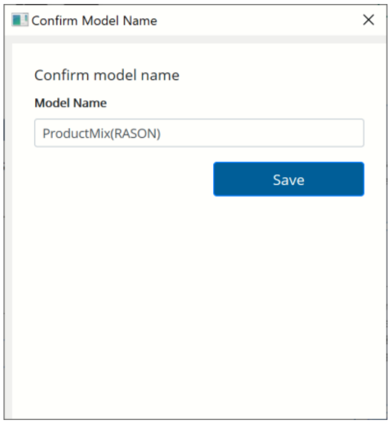


Deploy Model

Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Create Custom Visual that binds data, invokes named model

Confirm the name of the model by clicking Save on the Confirm Model Name dialog. This is the name of the model that will reside on the Rason Cloud Server.

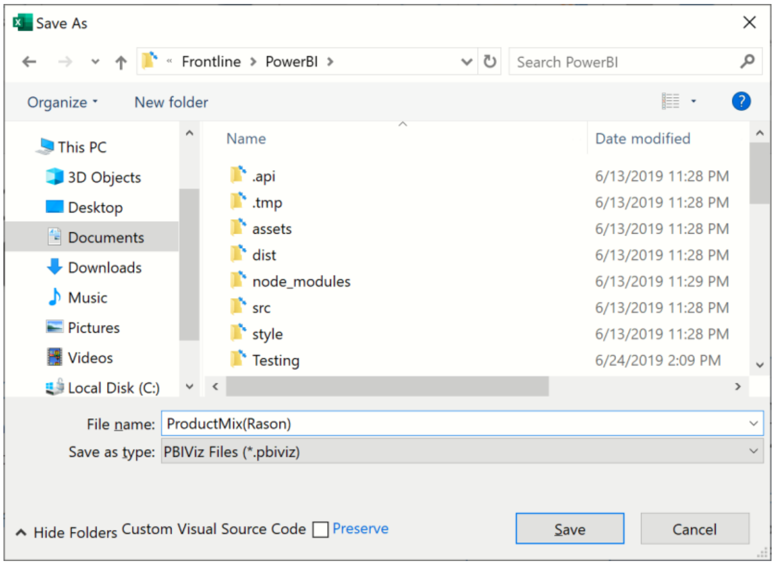


Confirm Model Name

Confirm model name

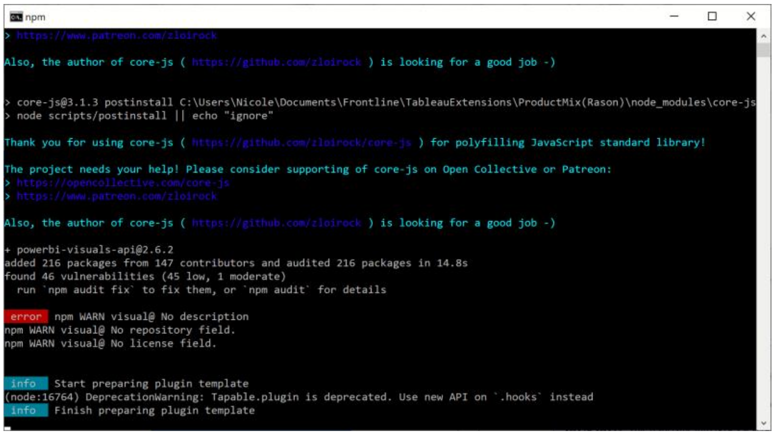
Model Name: ProductMix(RASON)

Browse to a location of your choice and enter a name for the PBIViz File. (This example uses ProductMix(Rason).



File name: ProductMix(Rason)
Save as type: PBIViz Files (*.pbiviz)

Then click Save to save the ProductMix(Rason).pbiviz file. A window appears as the file is being created.

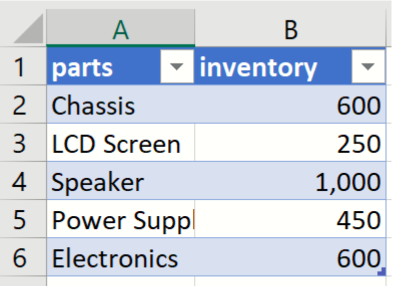


```
> https://www.patreon.com/zloirock

Also, the author of core-js ( https://github.com/zloirock ) is looking for a good job -)


> core-js@3.1.3 postinstall C:\Users\Nicole\Documents\Frontline\TableauExtensions\ProductMix(Rason)\node_modules\core-js
> node scripts/postinstall || echo "ignore"

Thank you for using core-js ( https://github.com/zloirock/core-js ) for polyfilling JavaScript standard library!

The project needs your help! Please consider supporting of core-js on Open Collective or Patreon:
> https://opencollective.com/core-js
> https://www.patreon.com/zloirock

Also, the author of core-js ( https://github.com/zloirock ) is looking for a good job -)

+ powerbi-visuals-api@2.6.2
added 216 packages from 147 contributors and audited 216 packages in 14.8s
found 46 vulnerabilities (45 low, 1 moderate)
  run `npm audit fix` to fix them, or `npm audit` for details

 error  npm WARN visual@ No description
npm WARN visual@ No repository field.
npm WARN visual@ No license field.


 info  Start preparing plugin template
(node:16764) DeprecationWarning: Tapable.plugin is deprecated. Use new API on `.hooks` instead
 info  Finish preparing plugin template
```

In order to provide our new data to Power BI, we must create a data file. This data file, containing the new inventory levels by part, will be imported into Power BI. This data file can be of any format supported by Power BI such as a CSV file, a text file, a database, etc. In this example, we will use an Excel workbook to hold the data. For more information on what types of files Power BI supports, please see the Power BI documentation at: https://powerbi.microsoft.com/en-us/documentation/powerbi-landing-page/

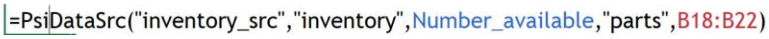
Open a blank worksheet and create a table with two columns. List the parts by name in the first column using the column heading “parts” and the inventory levels in the second column using the column heading “inventory”, as shown in the screenshot below. Make sure to pass a new inventory value of 1,000 Speakers.

Click Help - Example Models - Optimization Examples - Product Mix Data for Power BI - Tableau to open a complete data file. This table is included on the Parts Inventory tab.



	A	B
1	parts	inventory
2	Chassis	600
3	LCD Screen	250
4	Speaker	1,000
5	Power Suppl	450
6	Electronics	600

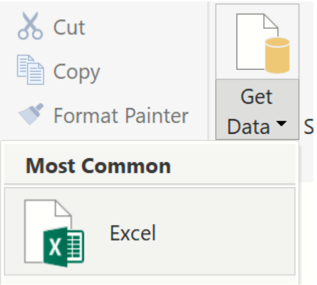
Note: Column headings ***must be identical*** *to the names of the Index Sets passed to the PsiDataSrc function. Make sure there are no spaces before or after column headings or row elements.*



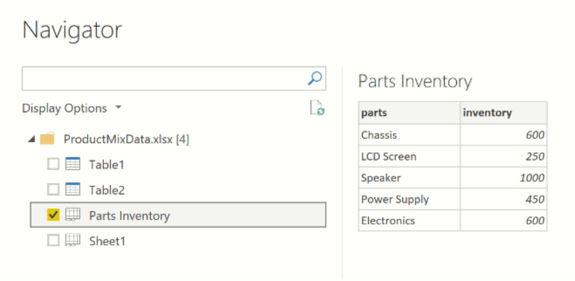
```
=PsiDataSrc("inventory_src","inventory",Number_available,"parts",B18:B22)
```

Recall that we labeled the IndexSet as “parts” (4th argument of PsiDataSrc) and Val_col_names as "inventory" (2nd argument of PsiDataSrc) so the title for column A must be "parts" and the title for column B must be "inventory". Now save the workbook to a desired location.

Open either desktop or cloud-based Power BI and click Get Data – Excel.

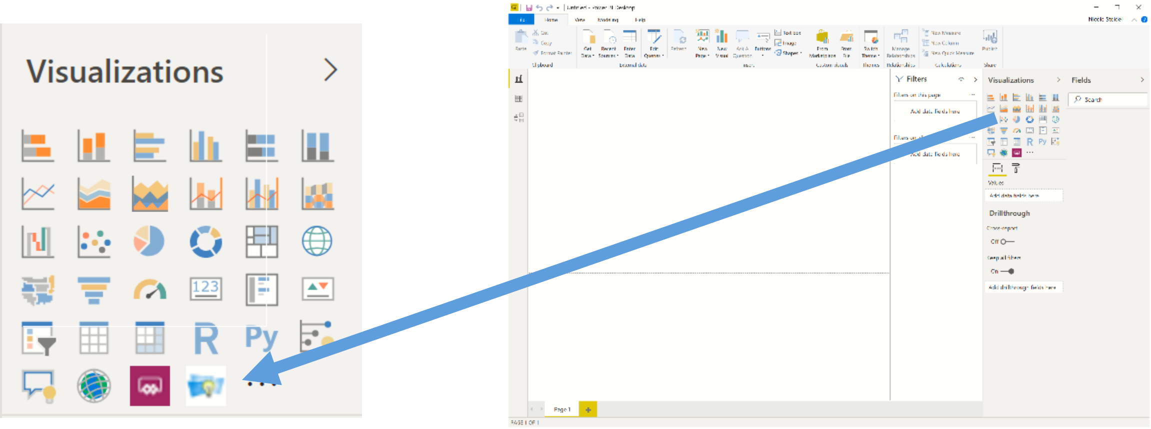


Browse to the location of the saved data file (C:\ProgramData\Frontline Systems\Examples) and click Open. Select the worksheet that contains your newly created data table and click Load.



parts	inventory
Chassis	600
LCD Screen	250
Speaker	1000
Power Supply	450
Electronics	600

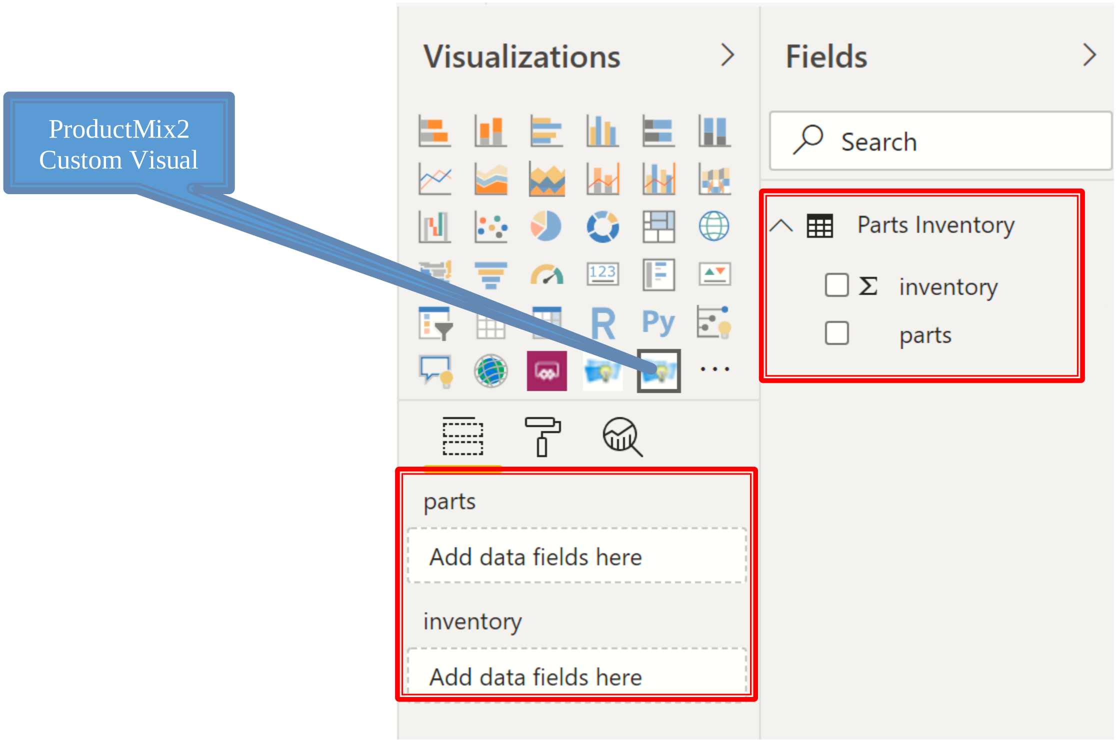
After the data table is uploaded to Power BI, click the icon containing three horizontal dots at the bottom of Visualizations, then select *Import from file* from the menu. Navigate to the location of the saved ProductMix(RASON).pbiviz file. Select the file, then click Open. If the import was successful, you will see a message indicating as such. Click OK to clear this dialog. A small icon appears bearing the Frontline Frontline Solvers logo is added under Visualizations.



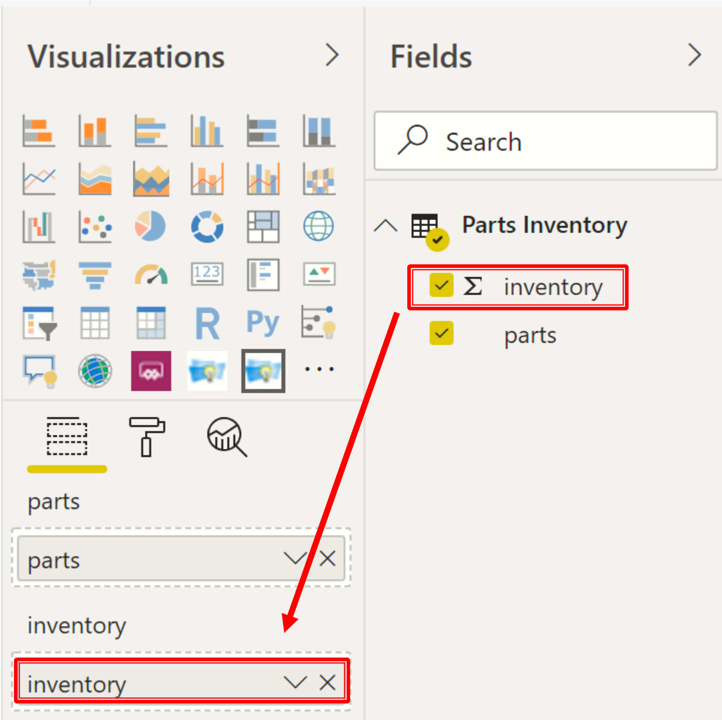
Click this icon to open Frontline Solvers’ custom visual.

Click the custom visual, the black and white chart is displayed. At this point you should see newly added items such as “parts” and “inventory” under Visualizations, on the task pane on the right, and the same under Fields. Recall that these are the same names that we passed to the PsiDataSrc() function within the Excel example model, ProductMixOpt.xlsx.

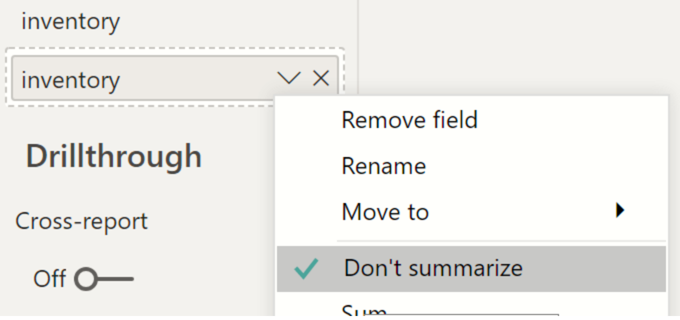
Two fields are added under Inventory in the Fields section of the Task Pane: inventory and parts.


ProductMix2 Custom Visual

Click parts Under Fields and move to parts under Visualizations where the words “Add data fields here” is located. Now drag inventory on the right to inventory on the left.

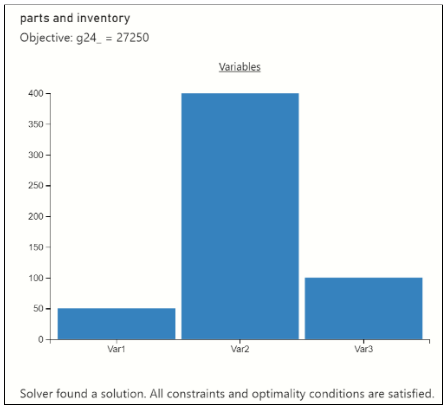


Click the down arrow next to both parts and inventory and select Don't summarize from the menu.



Immediately, the model is sent to the RASON server, solved, and the final values are returned. After enlarging the custom visual, your screen should appear similar to the screenshot below.

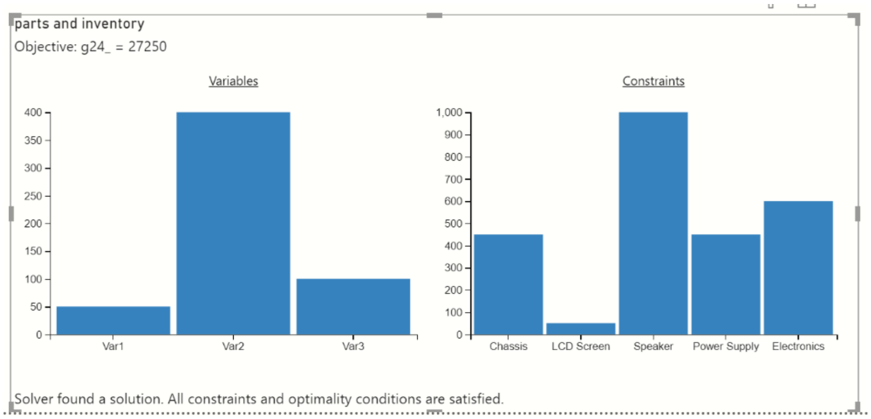
At the bottom of the custom visual, we find Solver's result message: Solver found a solution. All constraints and optimality conditions are satisfied. In the Variables chart, we see that final variable values are: Var1 (LCD TV) equal to 50, Var2 (Stereo) equal to 400 and Var3 (Speakers) equal to 100. These variable values result in an objective function value equal to 27,250. The payoff of increasing the available speakers from 800 to 1,000 is an added profit of $2,250 ($27,250 - $25,000).


parts and inventory

Objective: g24_ = 27250

Solver found a solution. All constraints and optimality conditions are satisfied.

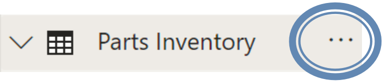
You can use the PsiFinalValue Psi function to add the constraints to the Custom Visual output, if desired, by entering PsiFinalValue(G18:G22) in a blank cell and then updating the model on the Rason Cloud Service (Deploy Model – Cloud Service – RASON Model) and recreating the Custom Visual (Deploy Model – Power BI – Managed Model).


parts and inventory

Objective: g24_ = 27250

Solver found a solution. All constraints and optimality conditions are satisfied.

You can easily update your inventory on the Parts Inventory tab of ProductMixData.xlsx, then click the three dots next to the data source within Power BI and select Refresh Data from the menu to resolve the model and display the new results in the Custom Visual.



You can update your named model (i.e. change a objective or constraint coefficient or a constraint right hand side) to a new version on the Rason Cloud service and still use the same Custom Visual in Power BI, as long as the input data definition your model requires has not changed. If the input data has changed, you'll need to use this menu option to create a new Custom Visual, to replace the old one.

Note: Continue reading to discover how to deploy a simulation model to Power BI or see the following chapter, Creating Power BI Custom Visuals, for more information on this feature.

Deploy Model – Power BI – Embedded Model

Another "quick" option that works for some (modest size) but not all Excel models, when you choose Deploy Model – Power BI – Embedded Model:

Analytic Solver automatically translates your Excel model into Rason and embeds your model into the Custom Visual.

Note: The menu choice Deploy Model – Cloud Service – Excel Model does not support the use of Data Wells in Power BI. You must first post your model to the RASON Server using Deploy Model – Cloud Service – Rason Model.

You don't have to create a named model that "lives" on the public Rason cloud. The Custom Visual accepts the data your model needs from Power BI; whenever this data changes, it sends your Rason model plus this data to the public Rason cloud service where it is solved, gets back results, and displays it in bar chart form.

Open the Business Forecast for Power BI – Tableau from Help – Example Models – Monte Carlo Simulation Examples. This model includes two PsiDataSrc functions in cells G24 and G25 to generate data wells for use in the Power BI Custom Visual.

G24 =PsiDataSrc(“market_src”, “market”, J17:L18, “parameters”, I17:I18, “mktdemand”, J16:L16)

G25=PsiDataSrc(“price_src”, “mktprice”, J21:L21, “price”, J20:L20)

Two Psi Functions, PsiPercentiles and PsiData, in order to draw the histogram in the Custom Visual.

Note that your model must also contain PsiPercentiles and PsiData to draw a histogram with 5th and 95th percentile markers in the Power BI Custom Visual when using Deploy Model – Power BI – Managed Model.

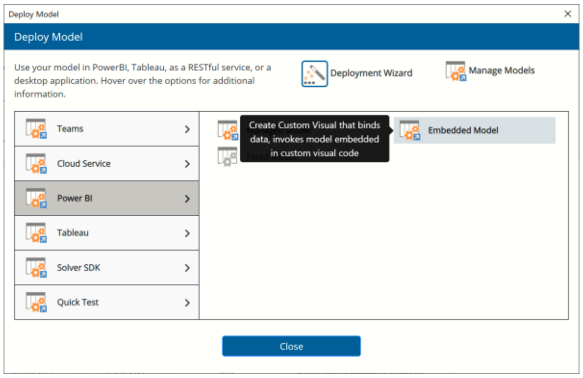
H28 = PsiPercentiles(F20)

G28 = PsiData(F20)

The PsiPercentiles function returns all percentiles (0.01-0.99) for the uncertain function in cell F20. The PsiData function returns all trial values for cell F20.

For more information on these formulas, see the Analytic Solver Reference Guide.

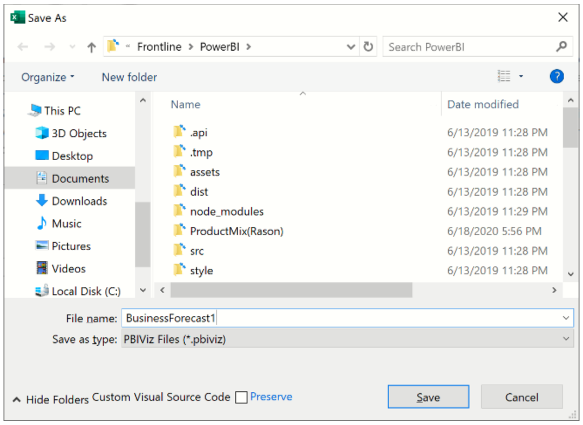
Click Deploy Model – Power BI – Embedded Model to create a new Power BI custom visual file.



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Create Custom Visual that binds data, invokes model embedded in custom visual code

When asked, provide a name for the Custom Visual.

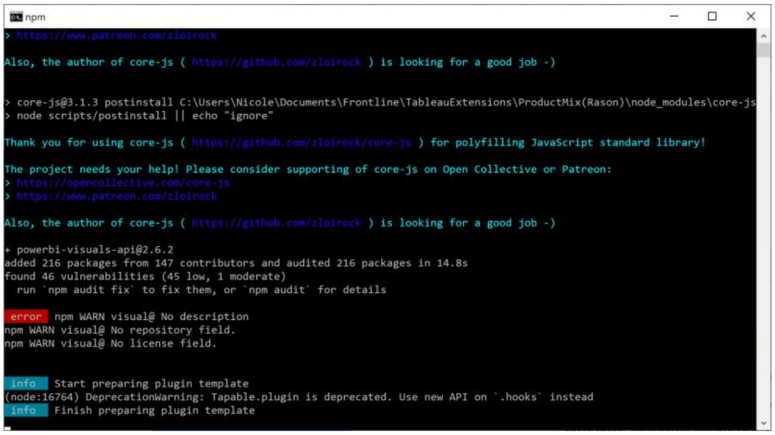


File name: BusinessForecast1

Save as type: PBIViz Files (*.pbiviz)

Custom Visual Source Code ☐ Preserve

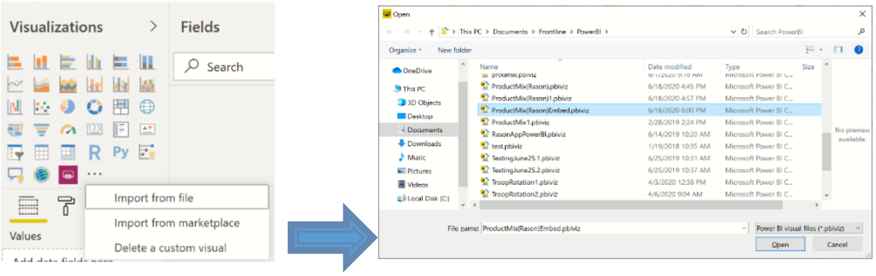
A window appears as the file is being created.



```
> https://www.patreon.com/zloirock

Also, the author of core-js ( https://github.com/zloirock ) is looking for a good job -)


> core-js@3.1.3 postinstall C:\Users\Nicole\Documents\Frontline\TableauExtensions\ProductMix(Rason)\node_modules\core-js
> node scripts/postinstall || echo "ignore"

Thank you for using core-js ( https://github.com/zloirock/core-js ) for polyfilling JavaScript standard library!

The project needs your help! Please consider supporting of core-js on Open Collective or Patreon:
> https://opencollective.com/core-js
> https://www.patreon.com/zloirock

Also, the author of core-js ( https://github.com/zloirock ) is looking for a good job -)

+ powerbi-visuals-api@2.6.2
added 216 packages from 147 contributors and audited 216 packages in 14.8s
found 46 vulnerabilities (45 low, 1 moderate)
  run `npm audit fix` to fix them, or `npm audit` for details

 error  npm WARN visual@ No description
npm WARN visual@ No repository field.
npm WARN visual@ No license field.


 info  Start preparing plugin template
(node:16764) DeprecationWarning: Tapable.plugin is deprecated. Use new API on `.hooks` instead
 info  Finish preparing plugin template
```

Analytic Solver automatically creates the Custom Visual that can be dragged onto the Power BI Dashboard, or a report. The Rason model will be embedded inside the Custom Visual.

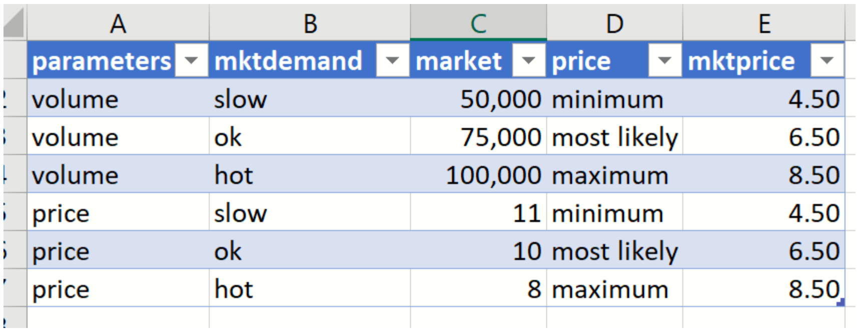
Once the npm window disappears, open Microsoft Power BI. Upon opening, click the three dots beneath Visualizations, and select Import from file.



File name: ProductMix(Rason)Embed.pbiviz

Power BI visual files (*.pbiviz)

Click back to Excel and open a new workbook to enter the following data **into an Excel table,** then save the workbook as BusinessForecastData.xlsx. (You can open the completed data file, BusinessForecastData.xlsx, from Help – Example Models – Monte Carlo Simulation Examples.)

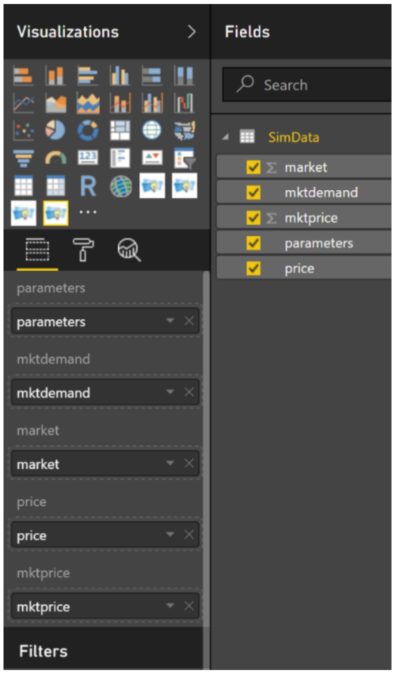


parameters	mktdemand	market	price	mktprice
volume	slow	50,000	minimum	4.50
volume	ok	75,000	most likely	6.50
volume	hot	100,000	maximum	8.50
price	slow	11	minimum	4.50
price	ok	10	most likely	6.50
price	hot	8	maximum	8.50

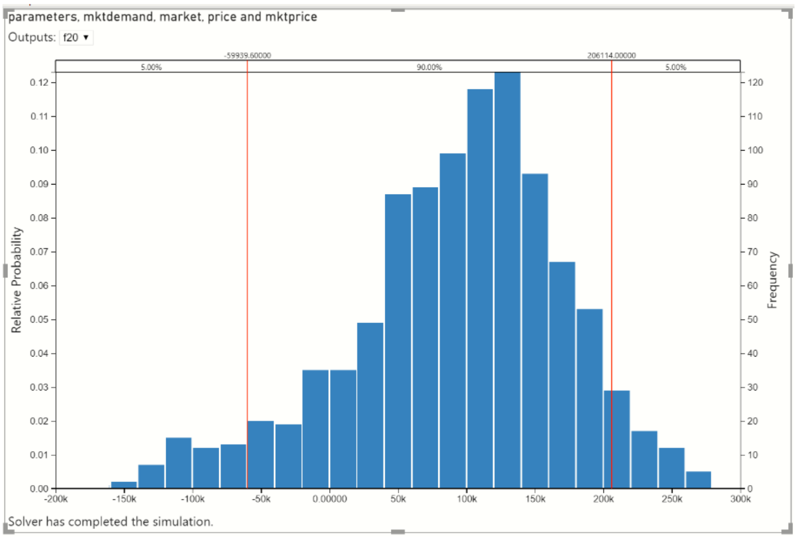
Open Power BI and add in the custom visual (Import from file – BusinessForcast1.pbiviz), then click Get Data – Excel to import the data. Afterwards, drag the data fields to the appropriate data wells as shown in the table below.

Fields	Data Wells
market	market
mktdemand	mktdemand
mktprice	mktprice
parameters	parameters
price	price

Your task pane should match the following screenshot. Make sure that none of the Fields are being summarized.



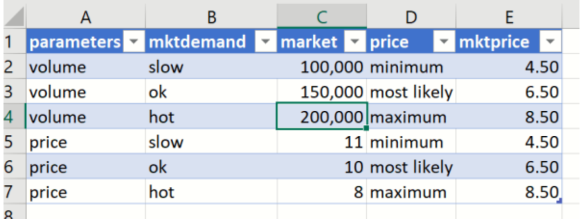
Immediately, once the data wells are filled with the appropriate data fields, a simulation will be run.



Solver has completed the simulation.

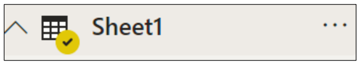
Again, note that as long as your PsiDataSrc functions remain the same, you will not need to create a new custom visual.

If your data changes, say your demand explodes, you can simply edit your data file.

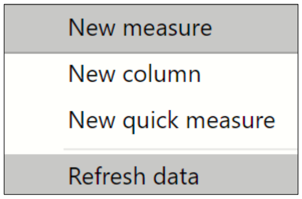


	A	B	C	D	E
1	parameters	mktdemand	market	price	mktprice
2	volume	slow	100,000	minimum	4.50
3	volume	ok	150,000	most likely	6.50
4	volume	hot	200,000	maximum	8.50
5	price	slow	11	minimum	4.50
6	price	ok	10	most likely	6.50
7	price	hot	8	maximum	8.50
8					

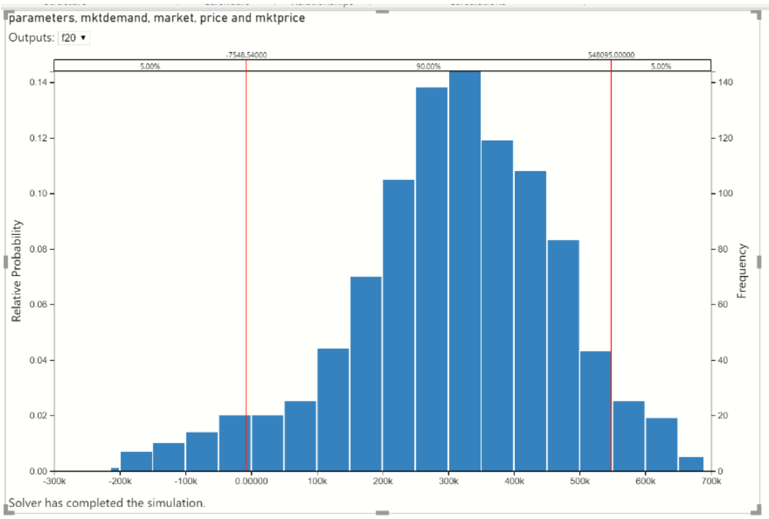
Click back to Power BI and click the three dots to the right of the data source…



and select Refresh Data from the menu.



The Custom Visual will automatically redraw using the new data.



parameters, mktdemand, market, price and mktprice

Outputs: f20 ▾

Solver has completed the simulation.

This option is simpler initially, but if you want to update your model in any way, you'll have to use this menu option to create a new Custom Visual. If you expect to revise the model over time, it's better to use the first option, Managed Model. For more information on creating a Power BI Custom Visual including how to use Data wells in Power BI, see the chapter Creating Power BI Custom Visuals.

Power BI Fitted Model

Users can deploy and share data science and machine learning models, trained in Analytic Solver or RASON, to the Azure cloud, and use them directly for classification and prediction (without needing auxiliary “code” in R or Python, RASON or Excel). (In the past, Analytic Solver Data Mining models could only be scored from within Excel, either online or desktop.) This example illustrates how one could use a fitted model created in Analytic Solver Data Science to score data residing in Power BI, Microsoft's data analytics platform.

Open the Scoring_Heart_Failure.xlsx example workbook by clicking Help – Example Models – Forecasting / Data Science Examples on the Analytic Solver ribbon. This example uses the Heart Failure Clinical Records Dataset[1], which contains thirteen variables describing 299 patients experiencing heart failure. The journal article referenced here discusses how the authors analyzed the dataset to first rank the features (variables) by significance and then used the Random Trees machine learning algorithm to fit a model to the dataset to be used to classify patients at risk of death due to heart failure.

DEATH_EVENT is the output variable in the Random Trees Classification model. This variable can either be a 1 or 0 . A patient classified as a 1 is predicted to perish due to complications of heart failure.

See the Feature Selection chapter within the Data Science User Guide for an example that attempts to emulate the results of that study. For now, this example dataset will illustrate how to score new data contained within Power BI

---

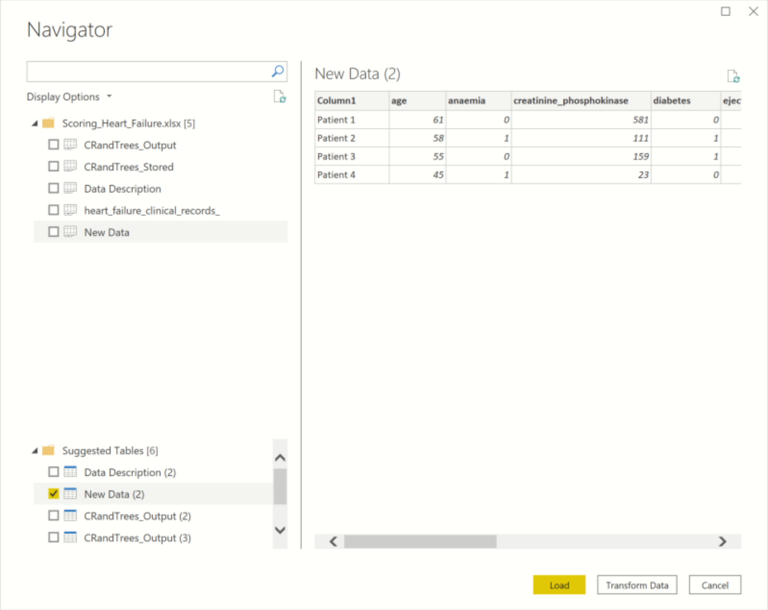
[1] Davide Chicco, Giuseppe Jurman: Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. BMC Medical Informatics and Decision Making 20, 16 (2020). (link)

using the Random Trees Classification fitted model saved in the workbook on the CRandTrees_Stored worksheet.

Click the New Data tab to open a list of new patient records which require classification using the stored model, CRandTrees_Stored. It is imperative that patients with a classification of "1" for the output variable, DEATH_EVENT, receive medical mitigation in order to reduce the chance of death from heart failure.

Open Power BI and import the New Data table. (Open Power BI and click Get Data – Excel and select the New Data table, then click Load.)

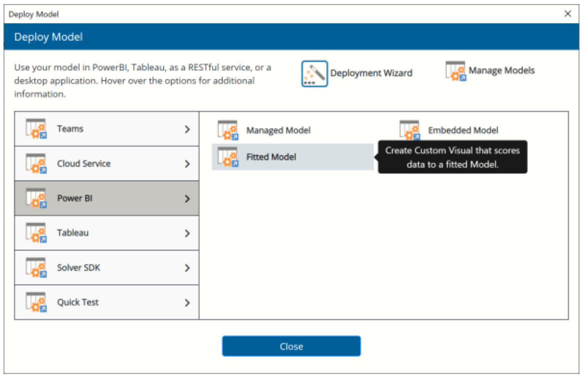
Note: This dataset includes four rows of new data but only 1 row is required for scoring.



New Data (2)

Column1	age	anaemia	creatinine_phosphokinase	diabetes	ejec
Patient 1	61	0	581	0	
Patient 2	58	1	111	1	
Patient 3	55	0	159	1	
Patient 4	45	1	23	0	

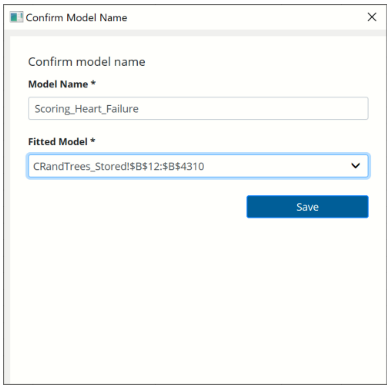
Click back to the CRandTrees_Stored worksheet in Excel and then click Deploy Model – Power BI – Fitted model to POST the fitted model, CRAndTrees_Stored, to the RASON Server.



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Create Custom Visual that scores data to a fitted Model.

This fitted model will be used to score the data that was just imported into Power BI.

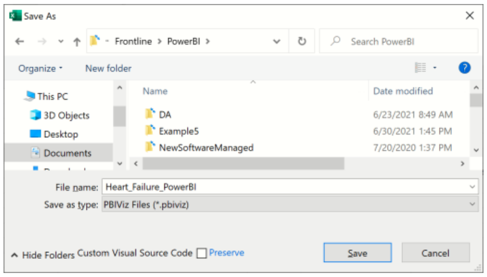


Model Name * Scoring_Heart_Failure

Fitted Model * CRandTrees_Stored!$B$12:$B$4310

Enter a Model Name on the Confirm Model Name dialog, then click the down arrow beneath Fitted Model and select the range CRandTrees_Stored!$B$12:$B$B4310. This is the range where the fitted model is saved on the CRandTrees worksheet.

Enter a file name for the Power BI custom visual file that will be created and saved. This example uses Heart_Failure_PowerBI



Name	Date modified
DA	6/23/2021 8:49 AM
Example5	6/30/2021 1:45 PM
NewSoftwareManaged	7/20/2020 1:37 PM

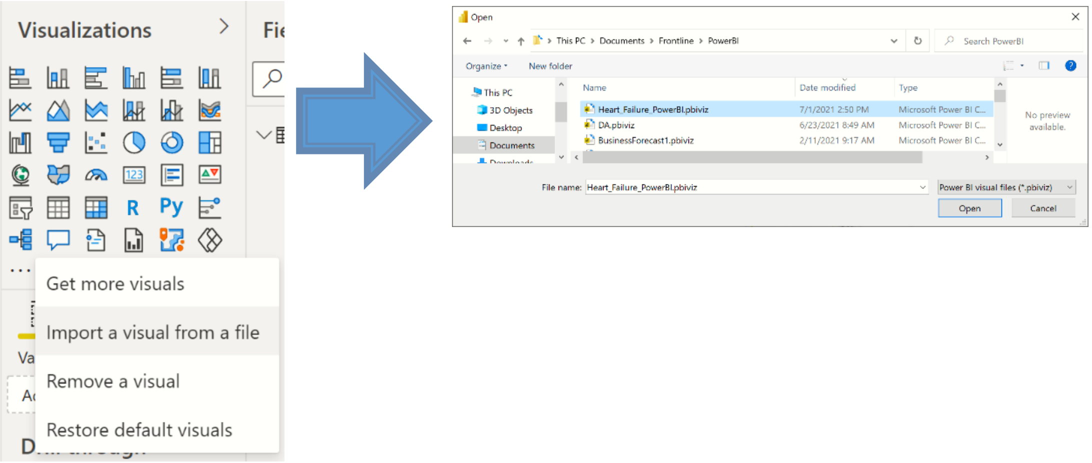
File name: Heart_Failure_PowerBI

Save as type: PBIViz Files (*.pbiviz)

Custom Visual Source Code ☐ Preserve

Click Save to save the Power BI Custom Visual. A command prompt window will appear on your screen and, after several moments, disappear. It is during this time that your Excel model is being translated into the **RASON**® modeling language, and your Power BI custom visual is created.

Click back to Power BI and import the Power BI custom visual by clicking the three dots beneath Visualizations, then browse to open Heart_Failure_PowerBI.pbiviz.



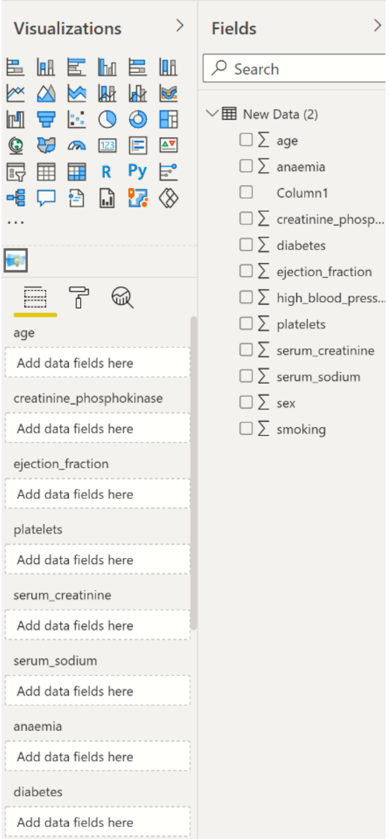
Name	Date modified	Type
Heart_Failure_PowerBI.pbiviz	7/1/2021 2:50 PM	Microsoft Power BI C...
DA.pbiviz	6/23/2021 8:49 AM	Microsoft Power BI C...
BusinessForecast1.pbiviz	2/11/2021 9:17 AM	Microsoft Power BI C...

File name: Heart_Failure_PowerBI.pbiviz

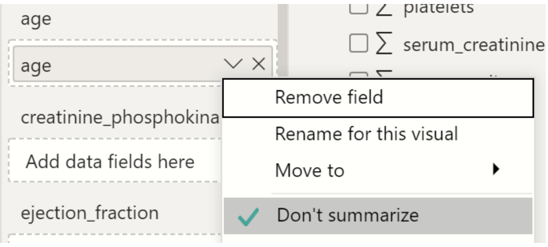
Power BI visual files (*.pbiviz)

Click the newly created Heart_Failure_PowerBI custom visual to display the custom visual data wells.

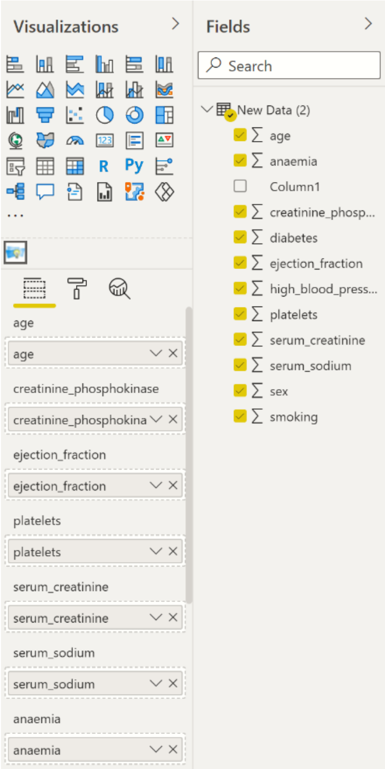
Expand New Data on the right to display the fields contained within the data table.



Match the data into the appropriate data well by dragging the New Data columns to the custom visual data wells, i.e. drag "age" to "age, "anaemia" to "anaemia", etc. Then click the down arrow next to each and select "Don't summarize".



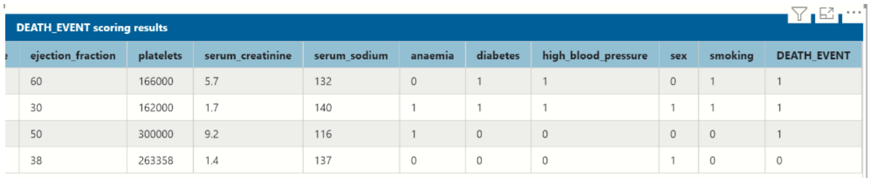
Power BI should look similar to the screenshot below after all fields are matched with the appropriate data wells.



age: age
creatinine_phosphokinase: creatinine_phosphokina
ejection_fraction: ejection_fraction
platelets: platelets
serum_creatinine: serum_creatinine
serum_sodium: serum_sodium
anaemia: anaemia

Once the last data column is match to the last data well, the custom visual will score the data and display the results in the Power BI dashboard, as shown in the screenshot below.

Output results contain the data to be scored plus the column containing the classification of the output variable, DEATH_EVENT.



DEATH_EVENT scoring results

ejection_fraction	platelets	serum_creatinine	serum_sodium	anaemia	diabetes	high_blood_pressure	sex	smoking	DEATH_EVENT
60	166000	5.7	132	0	1	1	0	1	1
30	162000	1.7	140	1	1	1	1	1	1
50	300000	9.2	116	1	0	0	0	0	1
38	263358	1.4	137	0	0	0	1	0	0

The scoring results indicate that the first three, out of the four patients, require some sort of medical mitigation in order to decrease the risk of death from heart failure.

Note that the order that Power BI scores the data may be slightly different than the order that Analytic Solver Data Science scores the data.

The .pbiviz file that is created by the Deploy Model tool may be manually altered to change the chart type created by Power BI. For more information on how to change the chart type created, contact Frontline Systems Support at support@solver.com.

Tableau

Clicking the fourth option on the Deploy Model dialog, Tableau, gives you three options, Managed Model, Embedded Model and Fitted Model.

- *Managed Model* translates the Excel optimization or simulation model to RASON and then POSTs this model to the RASON Server.

- *Excel Model* creates a named Excel optimization or simulation model that resides on the public RASON cloud service.
- *Fitted Model* POSTs the data science or forecasting fitted model contained within your Excel workbook to your RASON account.

Note: The first two options, Managed Model and Excel Model, only support the creation of a Tableau Extension with optimization and simulation models.

Tableau Managed Model

When you choose Deploy Model – Tableau – Managed Model, you must have previously used Deploy Model – Cloud Service (Rason or Excel) to create a named model that "lives" on the public Rason Cloud service. Recall that this model is identical to your original Excel model. It has simply been translated into Frontline’s RASON modeling language. Your objective, variables, constraints and bounds are now defined in RASON, instead of in an Excel spreadsheet.

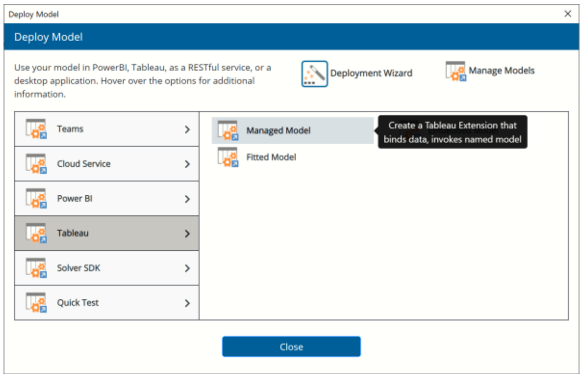
You will be prompted for the name of that model. Then Analytic Solver will automatically create a Tableau Dashboard Extension that can be used on a Tableau dashboard. (This is saved as a set of files, one ending in .trex that must be added to "My Extensions" in Tableau.)

The Dashboard Extension does three things: 1. It prompts you in Tableau to connect the input data your Excel or Rason model needs, for any Tableau source; 2. Whenever the Tableau data is updated, it automatically runs your model as a cloud service, getting optimization or simulation results; 3. It displays results in bar chart form, right where you placed the visual on your dashboard or report.

You can update your named model to a new version on the Rason cloud service, and still use the same Dashboard Extension, so long as the input data definition your model needs has not changed. If it has changed, you must return to Deploy Model – Tableau – Managed Model to create a new Custom Visual, to replace the old one.

Click back to the ProductMix(RASON).xlsx example model discussed in the Cloud Service section above. Since we've already posted this model to the Rason cloud service within the previous Cloud Service section, we can immediately go to Deploy Model – Tableau – Managed Model.

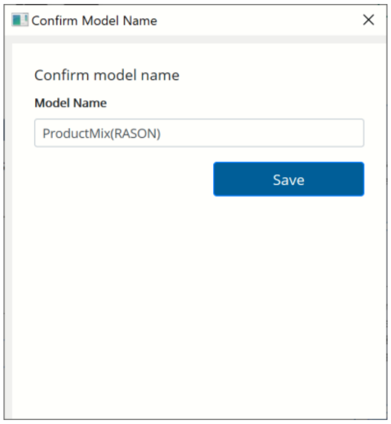
Note: The menu choice Deploy Model – Cloud Service – Excel Model does not support the use of Data Wells in Tableau. You must first post your model to the RASON Server using Deploy Model – Cloud Service – Rason Model.



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Create a Tableau Extension that binds data, invokes named model

Confirm the name of the model by clicking Save on the Confirm Model Name dialog. This is the name of the model that will reside on the Rason Cloud Server. To see a list of all models in your Rason account, click Manage Models (at the top of the Deploy Model dialog). For more information on the Manage Models dialog, see the Introduction to this chapter.

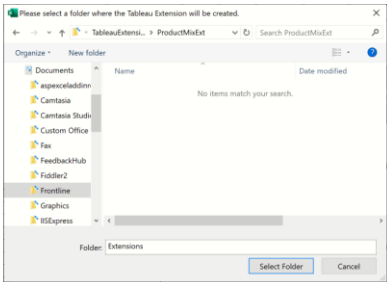


Confirm model name

Model Name

ProductMix(RASON)

Browse to a location of your choice and select a folder to save the Tableau Extension. (This example uses ProductMixExt.)



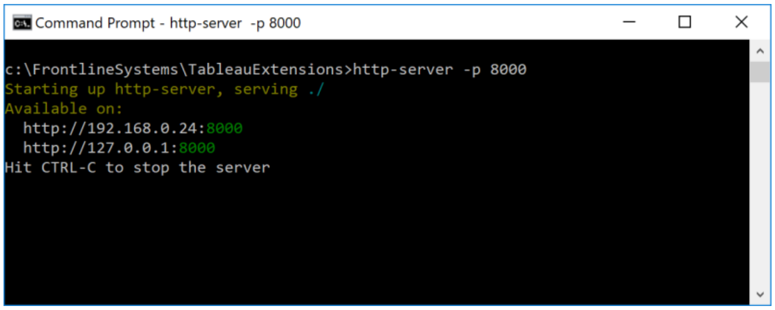
Please select a folder where the Tableau Extension will be created.

No items match your search.

Folder: Extensions

Starting up the Server

Since Tableau extensions are simply web pages, we will first need to start up a web server to serve our content. For this example, we will serve up the webpage to the default location. To do so, open a command prompt, navigate to the root of the extensions repository and run “http-server -p 8000”.



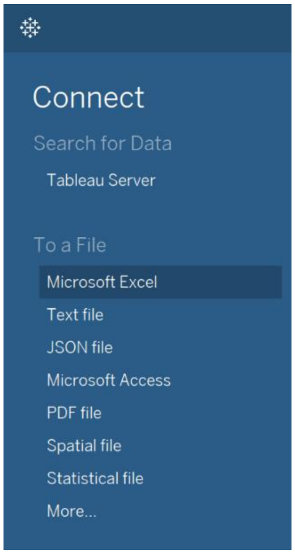
```
c:\FrontlineSystems\TableauExtensions>http-server -p 8000
Starting up http-server, serving ./
Available on:
  http://192.168.0.24:8000
  http://127.0.0.1:8000
Hit CTRL-C to stop the server
```

This command starts up a simple http server listening on port 8000.

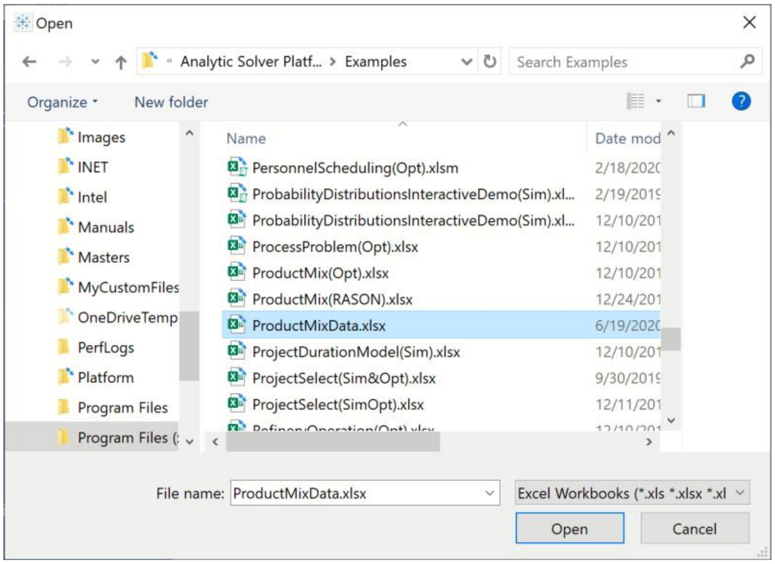
Opening a Tableau Extension

Open Tableau from the Desktop or on the Web. Upon opening, you'll be asked to connect to a data source. You can use ProductMixData.xlsx file saved locally at C:\ProgramData\Frontline Systems\Examples.

Click Connect – Microsoft Excel.

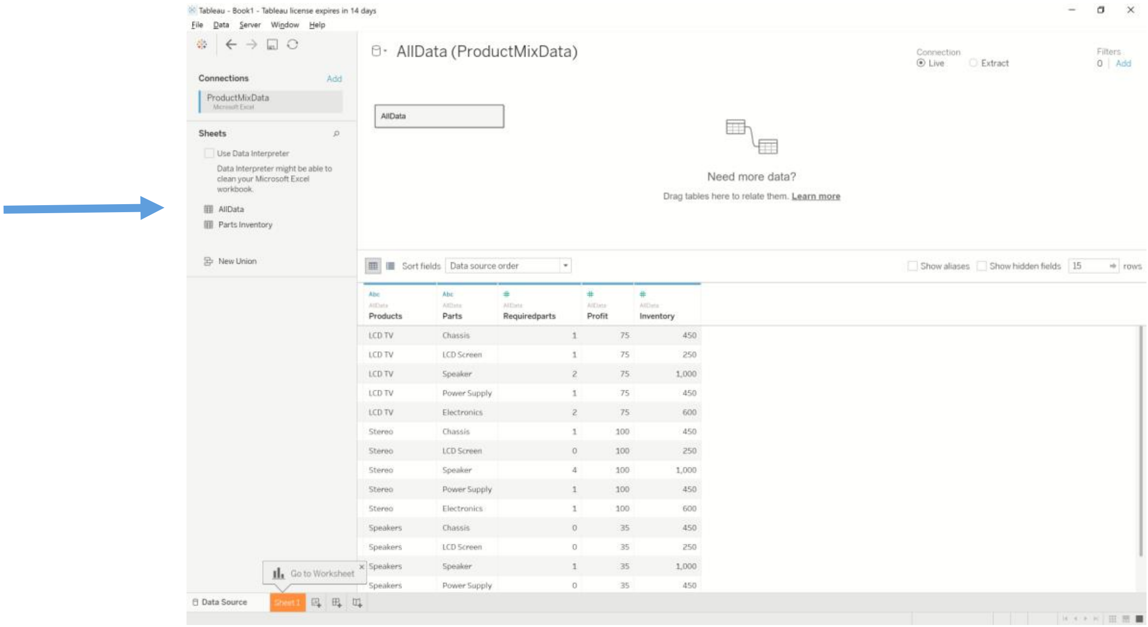


Browse to C:\ProgramData\Frontline Systems\Examples and open the file ProductMixData.xlsx. (If you are using Analytic Solver Cloud app, then you can download the file from Help – Example Models – Optimization Examples and save the file locally or to your OneDrive account.)


Open
Analytic Solver Platf... > Examples
Search Examples
Organize
New folder
Images
INET
Intel
Manuals
Masters
MyCustomFiles
OneDriveTemp
PerfLogs
Platform
Program Files
Program Files (
Name
Date mod
PersonnelScheduling(Opt).xlsm
ProbabilityDistributionsInteractiveDemo(Sim).xl...
ProbabilityDistributionsInteractiveDemo(Sim).xl...
ProcessProblem(Opt).xlsx
ProductMix(Opt).xlsx
ProductMix(RASON).xlsx
ProductMixData.xlsx
ProjectDurationModel(Sim).xlsx
ProjectSelect(Sim&Opt).xlsx
ProjectSelect(SimOpt).xlsx
Excel Workbooks (*.xls *.xlsx *.xl
Open
Cancel

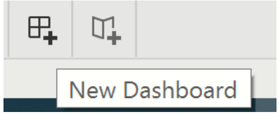
File name: ProductMixData.xlsx

Click Open. Tableau opens to the Data Source tab. Select the AllData Sheet to open the data table.

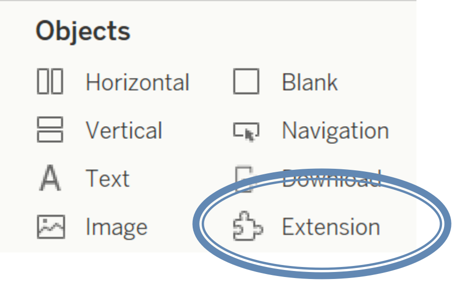


Products	Parts	Requiredparts	Profit	Inventory
LCD TV	Chassis	1	75	450
LCD TV	LCD Screen	1	75	250
LCD TV	Speaker	2	75	1,000
LCD TV	Power Supply	1	75	450
LCD TV	Electronics	2	75	600
Stereo	Chassis	1	100	450
Stereo	LCD Screen	0	100	250
Stereo	Speaker	4	100	1,000
Stereo	Power Supply	1	100	450
Stereo	Electronics	1	100	600
Speakers	Chassis	0	35	450
Speakers	LCD Screen	0	35	250
Speakers	Speaker	1	35	1,000
Speakers	Power Supply	0	35	450

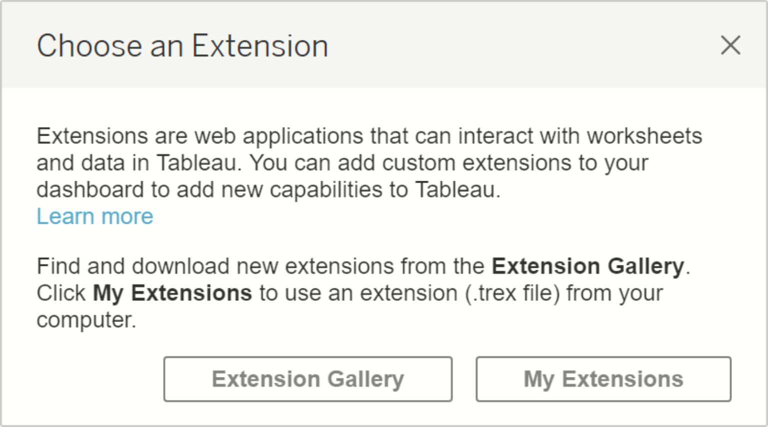
Click the New Dashboard icon at the bottom of the page to create a new Tableau dashboard.



Drag an Extension (from the Objects palette) to the dashboard.



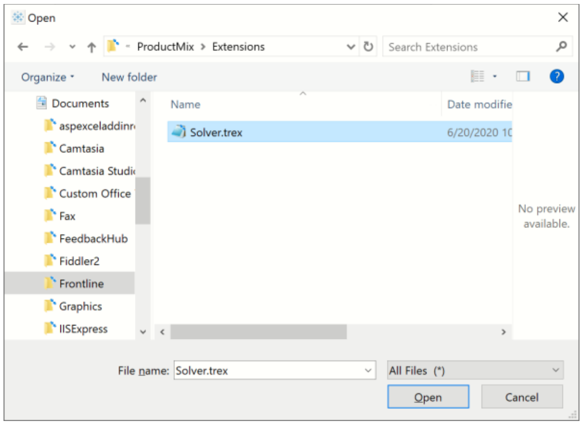
Click My Extensions…



Extensions are web applications that can interact with worksheets and data in Tableau. You can add custom extensions to your dashboard to add new capabilities to Tableau.
Learn more

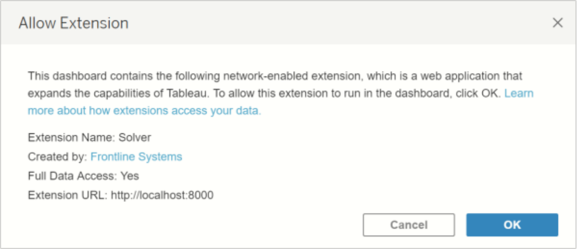
Find and download new extensions from the **Extension Gallery**. Click **My Extensions** to use an extension (.trex file) from your computer.

…browse to the location of the Tableau Extension.



(In this example, the Tableau Extension was saved to the ProductMix folder. Drill down to the Extensions folder to open the Solver.trex file.) Click Open.

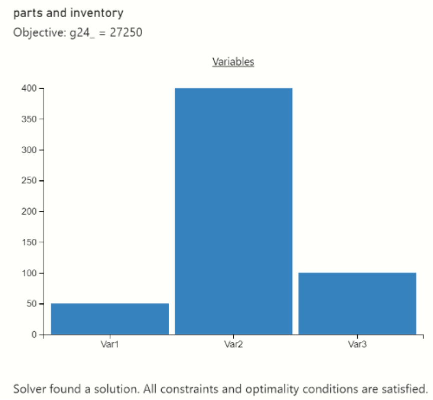
Click OK to allow the extension to be added to the dashboard.



Allow Extension

This dashboard contains the following network-enabled extension, which is a web application that expands the capabilities of Tableau. To allow this extension to run in the dashboard, click OK. Learn more about how extensions access your data.

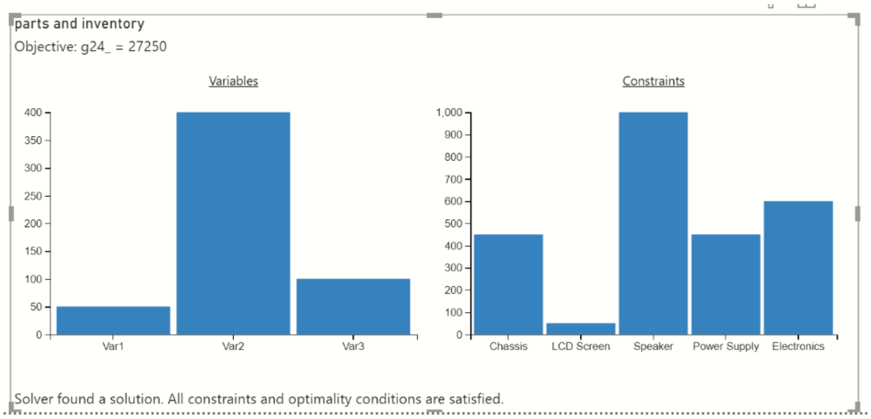
Extension Name: Solver
Created by: Frontline Systems
Full Data Access: Yes
Extension URL: http://localhost:8000



parts and inventory
Objective: g24_ = 27250

Solver found a solution. All constraints and optimality conditions are satisfied.

You can use the PsiFinalValue Psi function to add the constraints to the output, if desired, by entering PsiFinalValue(G18:G22) in a blank cell and then updating the model on the Rason Cloud Service (Deploy Model – Cloud Service – RASON Model) and recreating the Tableau Extension (Deploy Model – Tableau – Managed Model).

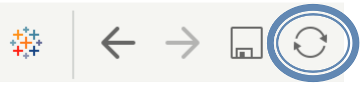


parts and inventory

Objective: g24_ = 27250

Solver found a solution. All constraints and optimality conditions are satisfied.

You can easily update your inventory on the Parts Inventory tab of ProductMixData.xlsx, then click back to the Data Source tab in Tableau and click the Refresh icon to refresh the data which will trigger an automatic resolve of the model. New results will be displayed in the Extension.



You can update your named model (i.e. change a objective or constraint coefficient or a constraint right hand side) to a new version on the Rason Cloud service and still use the same Tableau Extension, as long as the input data definition your model requires has not changed. If the input data has changed, you'll need to use this menu option to create a new Extension, to replace the old one.

See the following chapter Creating Custom Extensions in Tableau for more information on this feature.

Tableau Embedded Model

You have another "quick" option that works for some (modest size) but not all Excel models, when you choose Deploy Model – Tableau – Embedded Model. When this option is chosen, Analytic Solver translates your Excel model into Rason, and then automatically creates a "Dashboard Extension" that can be used on your Tableau dashboard.

This option is different than Tableau – Managed Model because with Tableau Embedded Model, your model is embedded inside the extensions' JavaScript file. *There is no need to create a named model that "lives" on the public Rason cloud.* The Dashboard Extension accepts the data your model needs from Tableau; whenever this data changes, it sends your Rason model plus this data to the public Rason cloud service where it is solved, obtains the results and displays them in chart form.

Although this option is initially simpler, if you want to update your model in any way, you'll have to use this menu option to create a new Dashboard Extension. If you expect to revise the model over time, it's better to use Deploy Model – Tableau – Managed Model.

Open the Business Forecast for Power BI – Tableau from Help – Example Models – Monte Carlo Simulation Examples. This model includes two

PsiDataSrc functions in cells G24 and G25 to generate data wells for use in the Tableau Extension.

G24 =PsiDataSrc("market_src", "market", J17:L18, "parameters", I17:I18, "mktdemand", J16:L16)

G25=PsiDataSrc("price_src", "mktprice", J21:L21, "price", J20:L20)

Two Psi Functions, PsiPercentiles and PsiData, in order to draw the histogram in the Tableau Extension.

H28 = PsiPercentiles(F20)

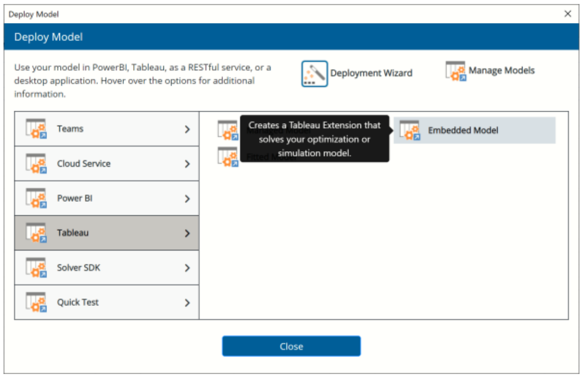
G28 = PsiData(F20)

The PsiPercentiles function returns all percentiles (0.01-0.99)e for the uncertain function in cell F20. The PsiData function returns all trial values for cell F20.

For more information on these formulas, see the Analytic Solver Reference Guide.

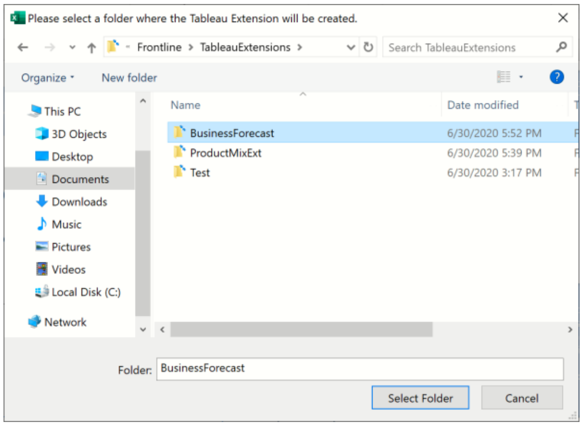
Click Deploy Model – Tableau – Embedded Model to create a new Tableau Extension.

Note: The menu choice Deploy Model – Cloud Service – Excel Model does not support the use of Data Wells in Tableau. You must first post your model to the RASON Server using Deploy Model – Cloud Service – Rason Model.


Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Creates a Tableau Extension that solves your optimization or simulation model.

Browse to a location of your choice and select a folder to save the Tableau Extension. (This example uses BusinessForecast.)


Please select a folder where the Tableau Extension will be created.

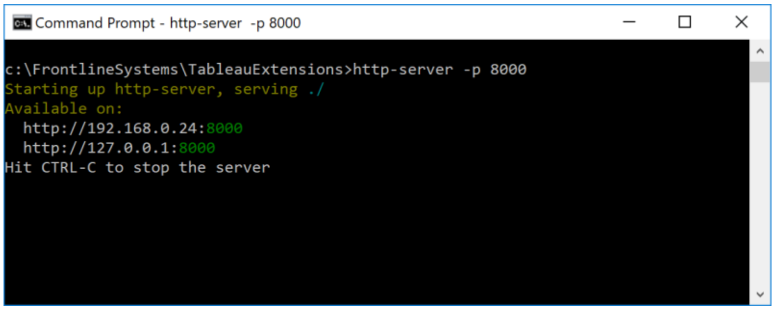
Name	Date modified
BusinessForecast	6/30/2020 5:52 PM
ProductMixExt	6/30/2020 5:39 PM
Test	6/30/2020 3:17 PM

Folder: BusinessForecast

A Note on Starting a Web Server

Recall that since Tableau extensions are simply web pages, a web server must first be started in order to serve the content of the extension. If a web server is currently running, there's no need to restart. However, if not, please follow the directions below.

For this example, we will serve up the webpage to the default location. To do so, open a command prompt, navigate to the root of the extensions repository and run "http-server -p 8000".

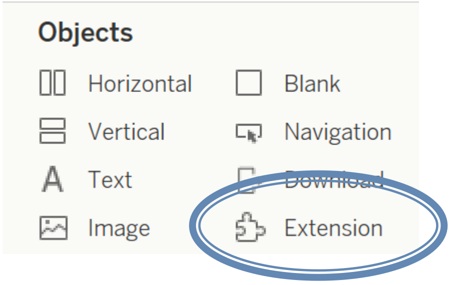


```
c:\FrontlineSystems\TableauExtensions>http-server -p 8000
Starting up http-server, serving ./
Available on:
  http://192.168.0.24:8000
  http://127.0.0.1:8000
Hit CTRL-C to stop the server
```

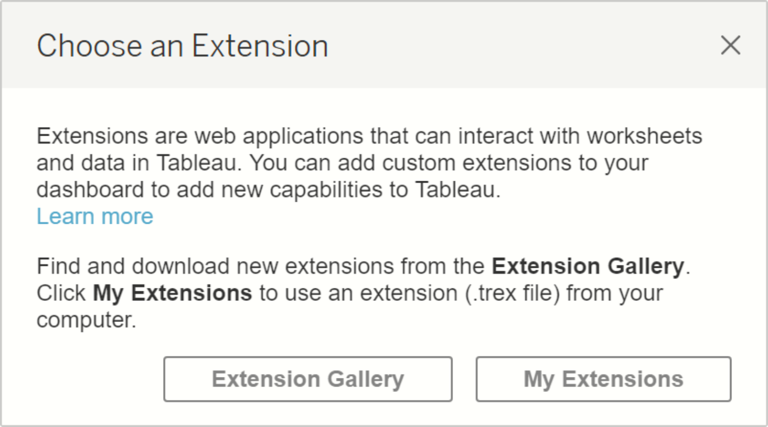
This command starts up a simple http server listening on port 8000.

Viewing the Extension in Tableau

Click back to Tableau and again drag the Extension icon to the dashboard.



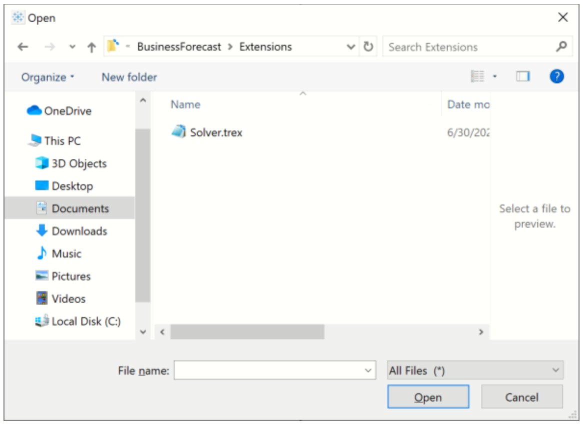
Click My Extensions…



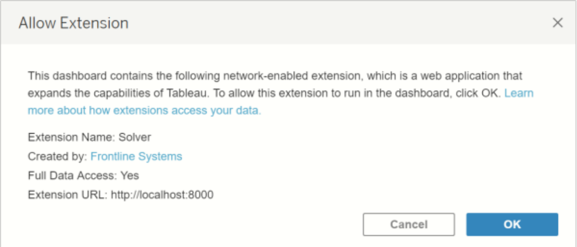
Extensions are web applications that can interact with worksheets and data in Tableau. You can add custom extensions to your dashboard to add new capabilities to Tableau.
Learn more

Find and download new extensions from the **Extension Gallery**. Click **My Extensions** to use an extension (.trex file) from your computer.

…browse to the location of the new Tableau Extension. (In this example …\BusinessForecast\Extensions.)



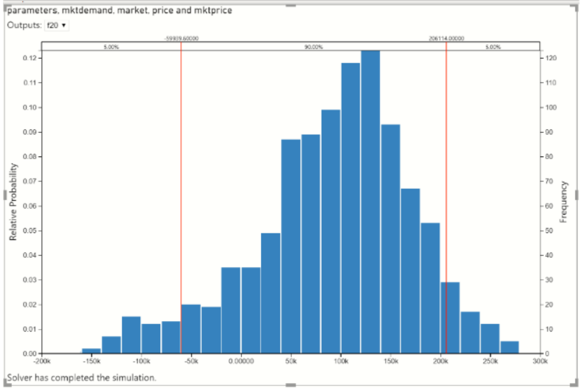
Click OK to allow the extension to be added to the dashboard.



Allow Extension

This dashboard contains the following network-enabled extension, which is a web application that expands the capabilities of Tableau. To allow this extension to run in the dashboard, click OK. Learn more about how extensions access your data.

Extension Name: Solver
Created by: Frontline Systems
Full Data Access: Yes
Extension URL: http://localhost:8000



Again, note that as long as your PsiDataSrc functions remain the same, the named model can be updated on the Rason Cloud Service and you will not need to create a new custom visual. However, if the PsiDataSrc() functions change, you will need to update both the Rason named model residing on the Rason Cloud Server and the custom visual.

Tableau Fitted Model

Users can deploy and share data science and machine learning models, trained in Analytic Solver or RASON, to the Azure cloud, and use them directly for classification and prediction (without needing auxiliary "code" in R or Python, RASON or Excel). (In the past, Analytic Solver Data Mining models could only be scored from within Excel, either online or desktop.) This example illustrates how one could use a fitted model created in Analytic Solver Data Science to score data residing in Tableau, a visualization platform. For more information on Tableau, see the chapter Creating Custom Extensions in Tableau that appears later in this guide.

Open the Scoring_Heart_Failure.xlsx example workbook by clicking Help – Example Models – Forecasting / Data Science Examples on the Analytic Solver ribbon. This example uses the Heart Failure Clinical Records Dataset[2], which contains thirteen variables describing 299 patients experiencing heart failure. The journal article referenced here discusses how the authors analyzed the dataset to first rank the features (variables) by significance and then used the Random Trees machine learning algorithm to fit a model to the dataset to be used to classify patients at risk of death due to heart failure.

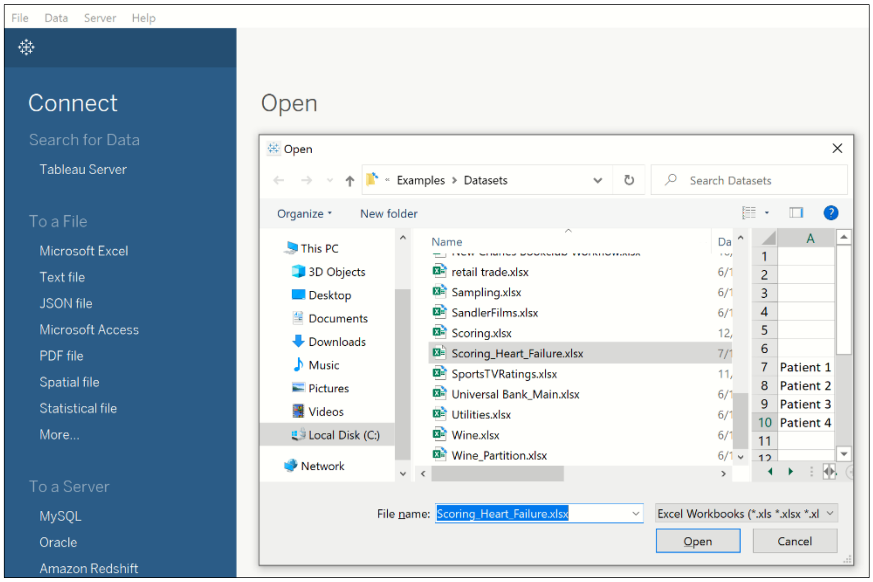
DEATH_EVENT is the output variable in the Random Trees Classification model. This variable can either be a 1 or 0. A patient classified as a 1 is predicted to perish due to complications of heart failure.

See the Feature Selection chapter within the Analytic Solver Data Science User Guide for an example that attempts to emulate the results of that study. For now, this example dataset will illustrate how to score new data contained within Tableau, using the Random Trees Classification fitted model saved in the workbook.

Click the New Data tab to open a list of new patient records which require classification using the stored model, CRandTrees_Stored. This classification is required in order to determine which patients require medical mitigation to reduce the chance of death from heart failure.

Open Tableau and import the New Data table.

Open Tableau and click Connect To a File: Microsoft Excel. Browse to the location of the Scoring_Heart_Failure.xlsx file typically C:\ProgramData\Frontline Systems\Datasets, then click Open.



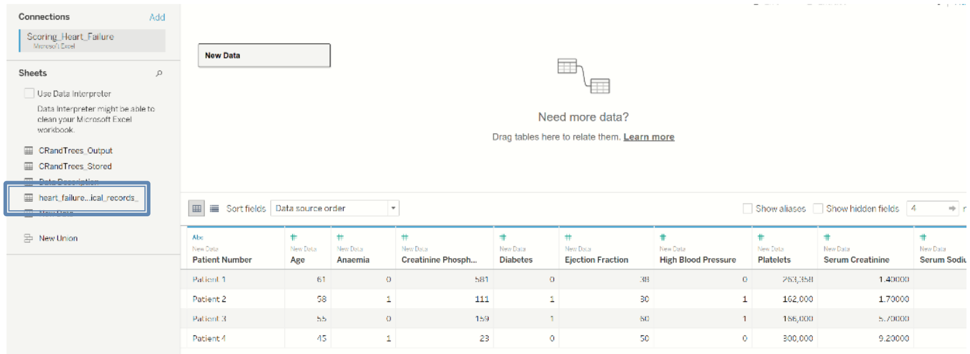
Click the New Data, on the bottom left, to open the table in Tableau. This table contains a list of new patient records which require classification using the stored model, CRandTrees_Stored. It is imperative that patients with a

---

[2] Davide Chicco, Giuseppe Jurman: Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. BMC Medical Informatics and Decision Making 20, 16 (2020). (link)

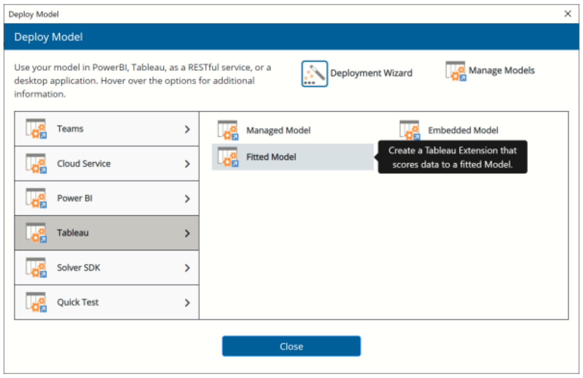
classification of "1" for the output variable, DEATH_EVENT, receive medical mitigation in order to reduce the chance of death from heart failure.

Note: This dataset includes four rows of new data but only 1 row is required for scoring.



Patient Number	Age	Anaemia	Creatinine Phosph...	Diabetes	Ejection Fraction	High Blood Pressure	Platelets	Serum Creatinine	Serum Sodiu
Patient 1	61	0	581	0	38	0	263,358	1.40000	
Patient 2	58	1	111	1	30	1	162,000	1.70000	
Patient 3	55	0	159	1	60	1	166,000	5.70000	
Patient 4	45	1	23	0	50	0	300,000	9.20000	

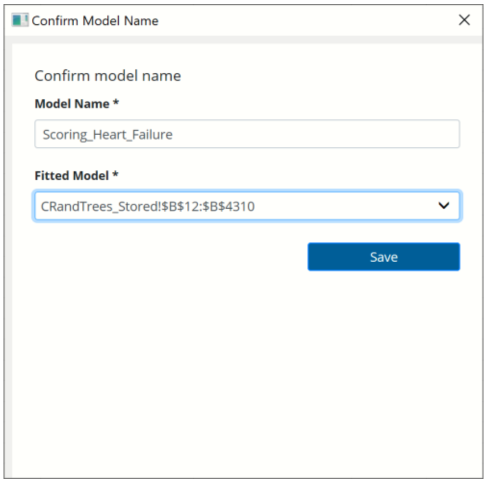
Click back to the CRandTrees_Stored worksheet in Excel and then click Deploy Model – Tableau– Fitted model to POST the fitted model, CRAndTrees_Stored, to the RASON Server.



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Create a Tableau Extension that scores data to a fitted Model.

This fitted model will be used to score the data that was just imported into Tableau



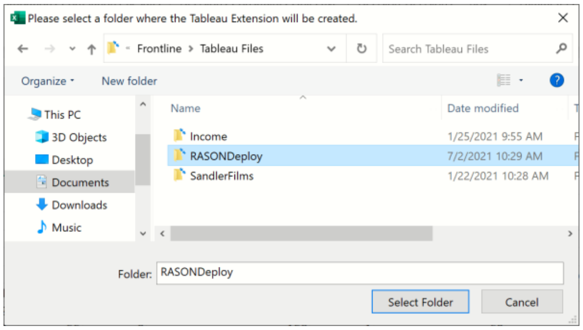
Confirm model name

Model Name * Scoring_Heart_Failure

Fitted Model * CRandTrees_Stored!$B$12:$B$4310

Enter a Model Name on the Confirm Model Name dialog, then click the down arrow beneath Fitted Model and select the range CRandTrees_Stored!$B$12:$B$B4310. This is the range where the fitted model is saved on the CRandTrees worksheet.

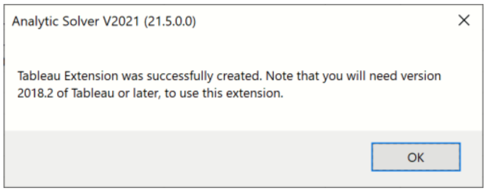
Select a folder to save the Tableau custom extension. This example uses the RASONDeploy folder.



Please select a folder where the Tableau Extension will be created.

Folder: RASONDeploy

Click Select Folder. It is during this time that your Excel model is being translated into the **RASON**® modeling language, and your Tableau custom extension is created. After a few moments, a dialog will appear indicating the Tableau extension has been created successfully.

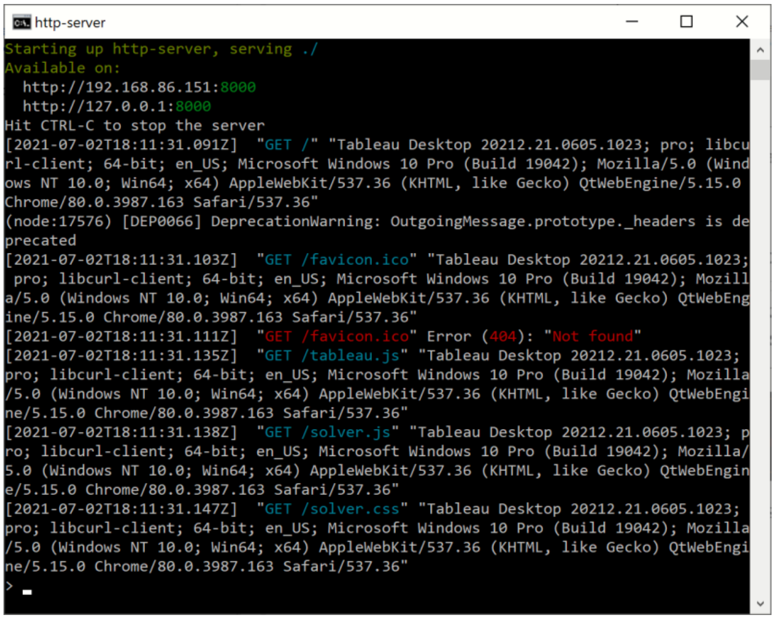


Analytic Solver V2021 (21.5.0.0)

Tableau Extension was successfully created. Note that you will need version 2018.2 of Tableau or later, to use this extension.

Starting up the Server

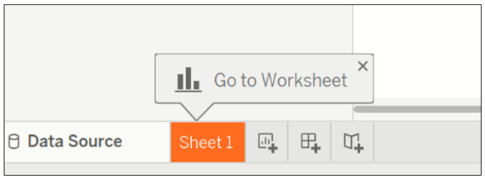
Since Tableau extensions are simply web pages, we will first need to start up a web server to serve our content. For this example, we will serve up the webpage to the default location. To do so, open a command prompt, navigate to the root of the extensions repository and run “http-server -p 8000”.

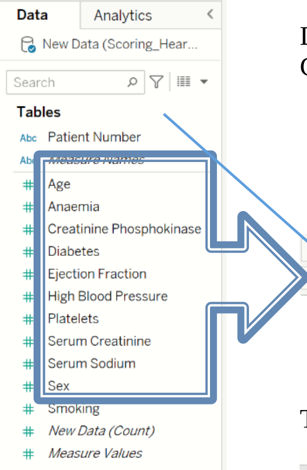


```
Starting up http-server, serving ./
Available on:
  http://192.168.86.151:8000
  http://127.0.0.1:8000
Hit CTRL-C to stop the server
[2021-07-02T18:11:31.091Z]  "GET /" "Tableau Desktop 20212.21.0605.1023; pro; libcu
rl-client; 64-bit; en_US; Microsoft Windows 10 Pro (Build 19042); Mozilla/5.0 (Wind
ows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) QtWebEngine/5.15.0
Chrome/80.0.3987.163 Safari/537.36"
(node:17576) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers is de
precated
[2021-07-02T18:11:31.103Z]  "GET /favicon.ico" "Tableau Desktop 20212.21.0605.1023;
 pro; libcurl-client; 64-bit; en_US; Microsoft Windows 10 Pro (Build 19042); Mozill
a/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) QtWebEng
ine/5.15.0 Chrome/80.0.3987.163 Safari/537.36"
[2021-07-02T18:11:31.111Z]  "GET /favicon.ico" Error (404): "Not found"
[2021-07-02T18:11:31.135Z]  "GET /tableau.js" "Tableau Desktop 20212.21.0605.1023;
pro; libcurl-client; 64-bit; en_US; Microsoft Windows 10 Pro (Build 19042); Mozilla
/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) QtWebEngi
ne/5.15.0 Chrome/80.0.3987.163 Safari/537.36"
[2021-07-02T18:11:31.138Z]  "GET /solver.js" "Tableau Desktop 20212.21.0605.1023; p
ro; libcurl-client; 64-bit; en_US; Microsoft Windows 10 Pro (Build 19042); Mozilla/
5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) QtWebEngin
e/5.15.0 Chrome/80.0.3987.163 Safari/537.36"
[2021-07-02T18:11:31.147Z]  "GET /solver.css" "Tableau Desktop 20212.21.0605.1023;
pro; libcurl-client; 64-bit; en_US; Microsoft Windows 10 Pro (Build 19042); Mozilla
/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) QtWebEngi
ne/5.15.0 Chrome/80.0.3987.163 Safari/537.36"
>
```

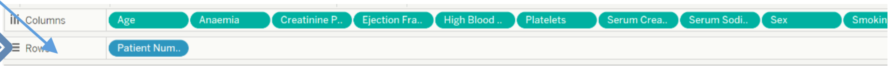
This command starts up a simple http server listening on port 8000.

Click back to Tableau and click Sheet 1.

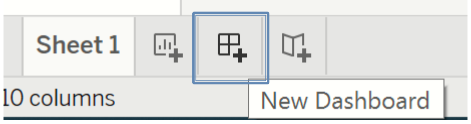




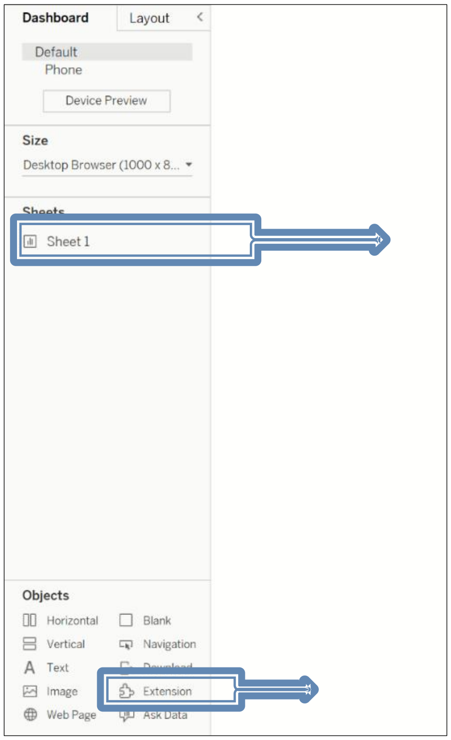
Drag Patient Number to Rows and then drag the 11 features (or variables) to Columns.



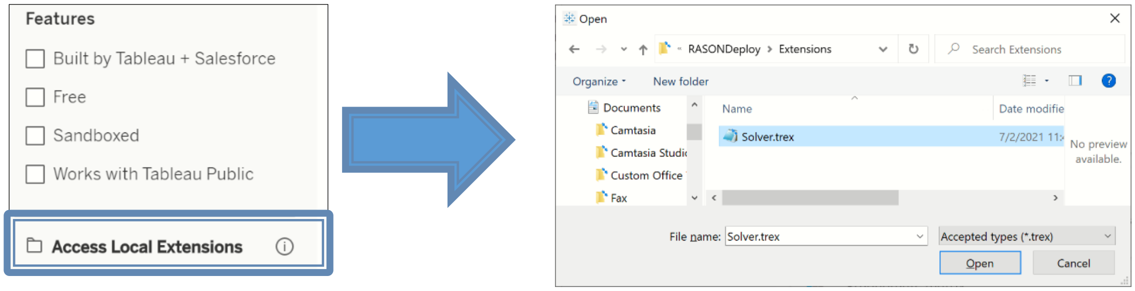
Then click New Dashboard to open a new Tableau dashboard.



Once opened, drag Sheet 1, under Sheets, and Extension, under Objects to the dashboard.

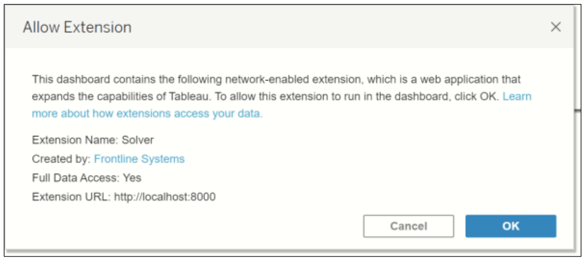


The "Add an Extension" dialog opens. On this dialog, click **Access Local Extensions** on the bottom left and browse to the location of the Tableau .trex file.



File name: Solver.trex

Click Open to open the Tableau extension. Then click OK to allow the extension to run the Tableau dashboard.



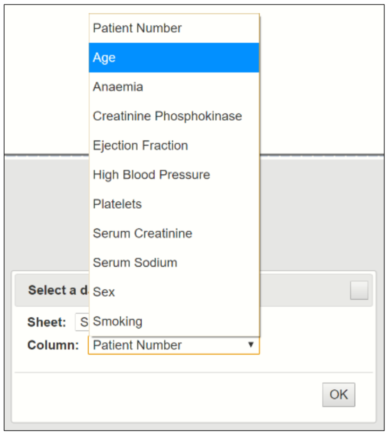
Allow Extension

This dashboard contains the following network-enabled extension, which is a web application that expands the capabilities of Tableau. To allow this extension to run in the dashboard, click OK. Learn more about how extensions access your data.

Extension Name: Solver
Created by: Frontline Systems
Full Data Access: Yes
Extension URL: http://localhost:8000

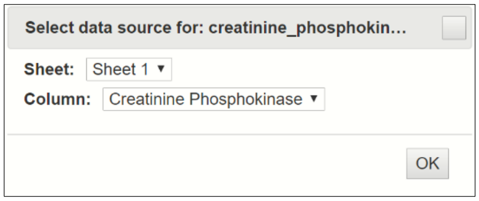
A series of dialogs will appear which allow the dimensions from the Tableau worksheet to be matched with the dimensions in the Solver custom Tableau extension (Solver.trex).

- Match the data source "age" dimension with the worksheet "age" dimension.



Sheet: S
Column: Patient Number

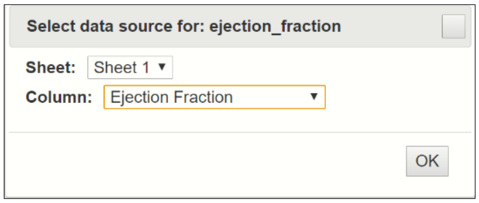
- Match the data source "creatinine_phosphokinase" dimension with the worksheet "Creatinine Phosphokinase" dimension.



Select data source for: creatinine_phosphokin...

Sheet: Sheet 1
Column: Creatinine Phosphokinase

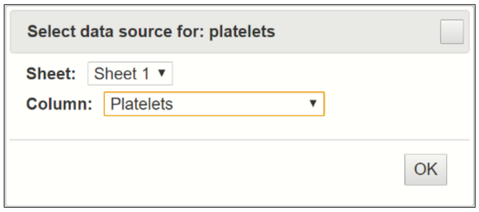
- Match the data source "ejection_fraction" dimension with the worksheet "Ejection Fraction" dimension.


Select data source for: ejection_fraction

Sheet: Sheet 1

Column: Ejection Fraction

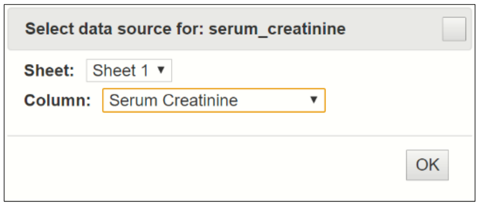
- Match the data source "Platelets" dimension with the worksheet "Platelets" dimension.


Select data source for: platelets

Sheet: Sheet 1

Column: Platelets

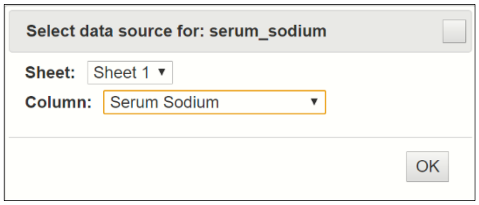
- Match the data source "serum_creatinine" dimension with the worksheet "Serum Creatinine" dimension.


Select data source for: serum_creatinine

Sheet: Sheet 1

Column: Serum Creatinine

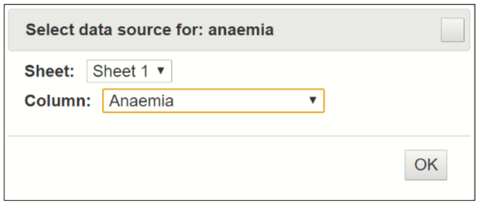
- Match the data source "serum_sodium" dimension with the worksheet "Serum Sodium" dimension.


Select data source for: serum_sodium

Sheet: Sheet 1

Column: Serum Sodium

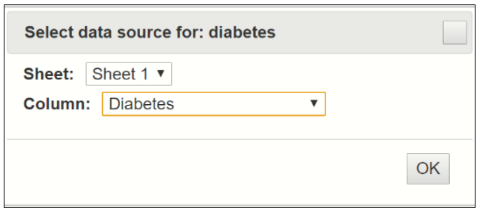
- Match the data source "anaemia" dimension with the worksheet "Anaemia" dimension.


Select data source for: anaemia

Sheet: Sheet 1

Column: Anaemia

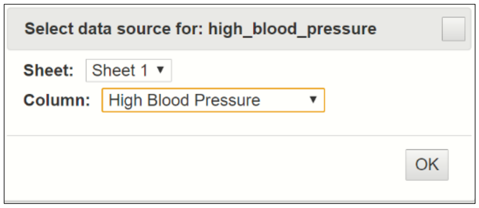
- Match the data source "diabetes" dimension with the worksheet "Diabetes" dimension.


Select data source for: diabetes

Sheet: Sheet 1

Column: Diabetes

- Match the data source "high_blood_pressure" dimension with the worksheet "High Blood Pressure" dimension.

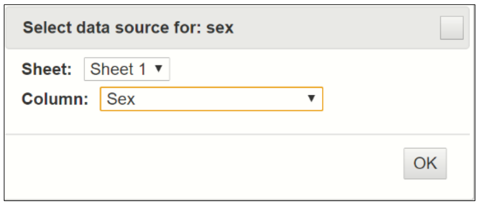


Select data source for: high_blood_pressure

Sheet: Sheet 1

Column: High Blood Pressure

- Match the data source "sex" dimension with the worksheet "Sex" dimension.

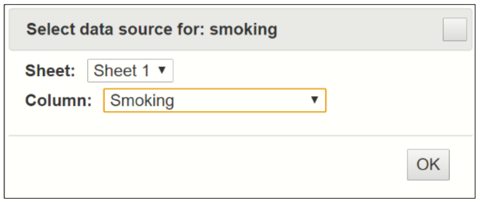


Select data source for: sex

Sheet: Sheet 1

Column: Sex

- Match the data source "smoking" dimension with the worksheet "Smoking" dimension.

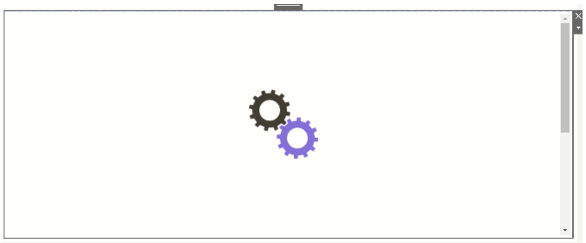


Select data source for: smoking

Sheet: Sheet 1

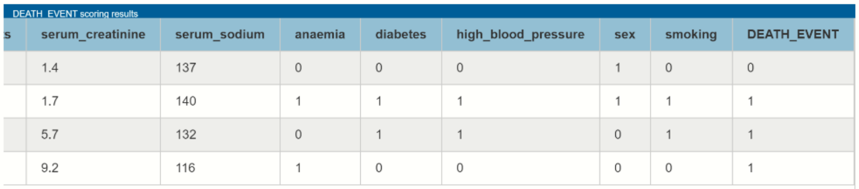
Column: Smoking

Once the last dimension is matched, Tableau immediately begins generating the custom extension.



The screenshot below displays the completed custom extension.

Output results contain the data to be scored plus the column containing the classification of the output variable, DEATH_EVENT.



DEATH_EVENT scoring results

s	serum_creatinine	serum_sodium	anaemia	diabetes	high_blood_pressure	sex	smoking	DEATH_EVENT
	1.4	137	0	0	0	1	0	0
	1.7	140	1	1	1	1	1	1
	5.7	132	0	1	1	0	1	1
	9.2	116	1	0	0	0	0	1

The scoring results indicate that the first three, out of the four patients, require some sort of medical mitigation in order to decrease the risk of death from heart failure.

SolverSDK: Rason or Excel Model

When you click Deploy Model – Solver SDK – Rason Model, Analytic Solver translates your Excel model into Rason and starts a simple Windows application that you can run right away, plus sample source code in C# that you can use to modify, compile and run the application. This time, when the app runs, it calls our Solver SDK asking it to load your Rason model from a JSON (text) file and solve it by performing an optimization or simulation. You can easily supply input data and pull results from the Rason model.

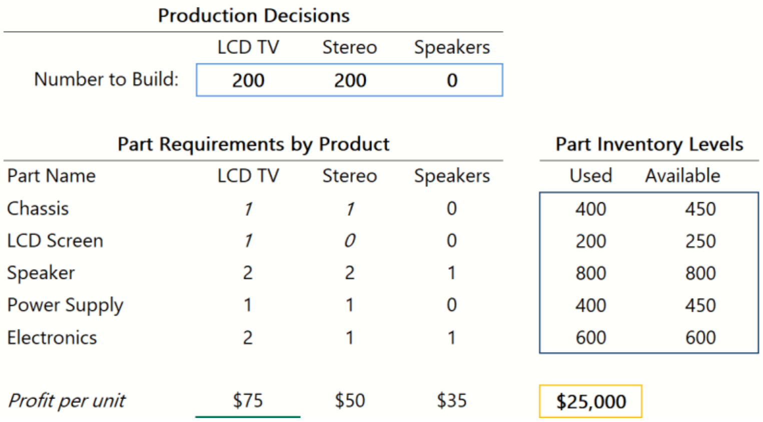
When you click Deploy Model – Solver SDK – Excel Model, Analytic Solver creates a simple Windows application plus sample source code in C# you can use to modify, compile (using Visual Studio) and run that Windows application. When the app runs, it calls our Solver SDK and asks it to load your Excel workbook model and solve it. Note: Data science models are not supported. Through the SDK's high-level, object-oriented API, users can obtain input data from any source available to your code and "plug" this data into your Excel model as loaded into memory – in cell ranges that you specify, "point and click" in Excel. Once solved, the app can easily pull results from cell ranges you choose in the model such as decision variables or the objective value.

RASON Model

Click Help – Example Models –Optimization Examples and click the Product Mix link.

Recall that this model determines the optimal mixture of products to manufacture given a set of parts.

To solve this example in Analytic Solver, we can of course simply click the green arrow on the task pane.



Production Decisions

	LCD TV	Stereo	Speakers
Number to Build:	200	200	0

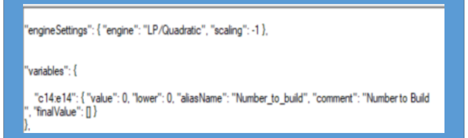
Part Requirements by Product				Part Inventory Levels	
Part Name	LCD TV	Stereo	Speakers	Used	Available
Chassis	*1*	*1*	0	400	450
LCD Screen	*1*	*0*	0	200	250
Speaker	2	2	1	800	800
Power Supply	1	1	0	400	450
Electronics	2	1	1	600	600
Profit per unit	$75	$50	$35	$25,000	

Notice the decision variable values for Number to Build: 200 TVs, 200 Stereos and no Speakers. The maximum profit is $25,000.

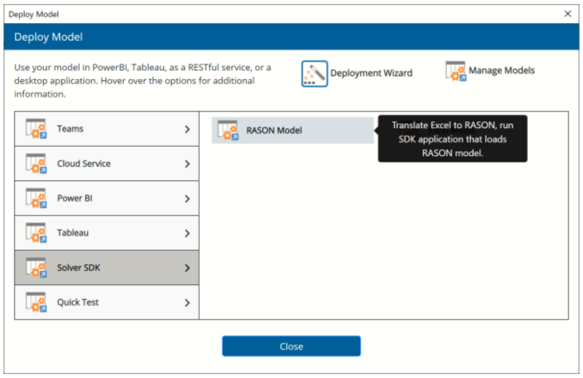
Now let's solve this same model using Frontline's Solver SDK, Frontline's Software Development Kit that allows you to solve your optimization and simulation models from within C++, C#, Java, Python or R.

Click Deploy Model – Solver SDK – RASON Model.

Excel notes will be deployed to RASON, along with the model, if they have been inserted into a model component cell. For example, in the Product Mix model to the right, if the text "Number to Build" is entered into a note for C14, the contents of the note will appear in the converted RASON model in the "comment" property for the variable block, c14:e14. Notes are supported for decision variables, uncertain variables, constraint left hand sides and right hand sides, uncertain functions, the objective, parameters (such as PsiSimParam/PsiOptParam) and Psi functions. In the case of a block of decision variables, uncertain variables, uncertain functions or constraints, the note must appear in the upper, most left cell. In this case, cell C14 is the upper, most left cell of the variable block C14:E14.



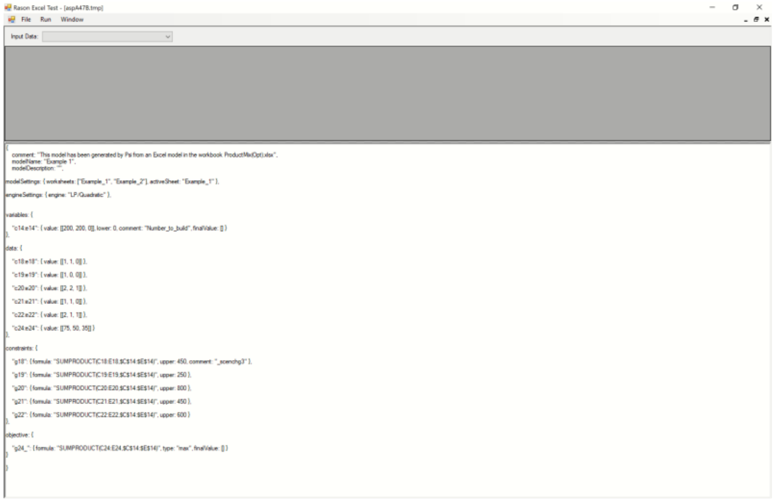
```
"engineSettings": { "engine": "LP/Quadratic", "scaling": -1 },

"variables": {

   "c14:e14": { "value": 0, "lower": 0, "aliasName": "Number_to_build", "comment": "Number to Build
", "finalValue": [] }
},
```



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

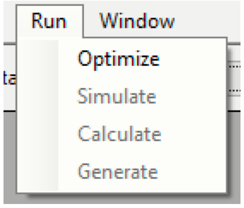
Translate Excel to RASON, run SDK application that loads RASON model.

Immediately, the Rason Run Test Windows application opens containing the Excel model translated into Rason code.

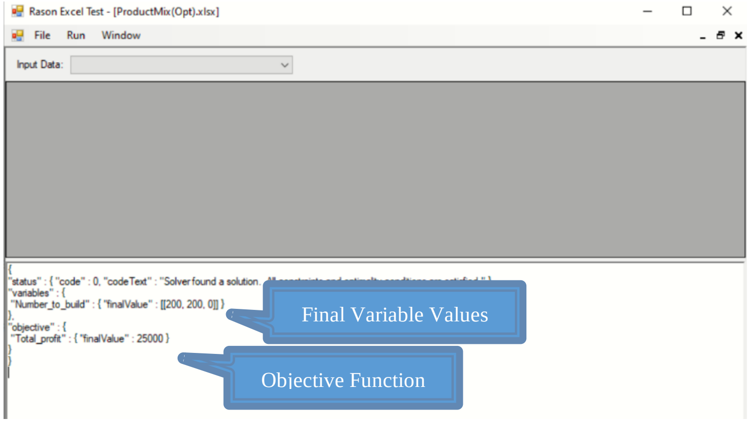


```
{
  comment: "This model has been generated by Psi from an Excel model in the workbook ProductMix(Opt).xlsx",
  modelName: "Example 1",
  modelDescription: "",

modelSettings: { worksheets: ["Example_1", "Example_2"], activeSheet: "Example_1" },

engineSettings: { engine: "LP/Quadratic" },


variables: {

  "c14:e14": { value: [[200, 200, 0]], lower: 0, comment: "Number_to_build", finalValue: [] }
},

data: {

  "c18:e18": { value: [[1, 1, 0]] },

  "c19:e19": { value: [[1, 0, 0]] },

  "c20:e20": { value: [[2, 2, 1]] },

  "c21:e21": { value: [[1, 1, 0]] },

  "c22:e22": { value: [[2, 1, 1]] },

  "c24:e24": { value: [[75, 50, 35]] }
},

constraints: {

  "g18": {formula: "SUMPRODUCT(C18:E18,$C$14:$E$14)", upper: 450, comment: "_scenchg3" },

  "g19": {formula: "SUMPRODUCT(C19:E19,$C$14:$E$14)", upper: 250 },

  "g20": {formula: "SUMPRODUCT(C20:E20,$C$14:$E$14)", upper: 800 },

  "g21": {formula: "SUMPRODUCT(C21:E21,$C$14:$E$14)", upper: 450 },

  "g22": {formula: "SUMPRODUCT(C22:E22,$C$14:$E$14)", upper: 600 }
},

objective: {

  "g24_": {formula: "SUMPRODUCT(C24:E24,$C$14:$E$14)", type: "max", finalValue: [] }
}

}
```

Solve the model by clicking Run – Optimize. If a simulation model had been exported, you would click Run Simulate and Run Calculate for a Decision Table Model.



The results are displayed in the output window. Notice the same final decision variable values of 200, 200 and 0 and the same objective function value of $25,000.



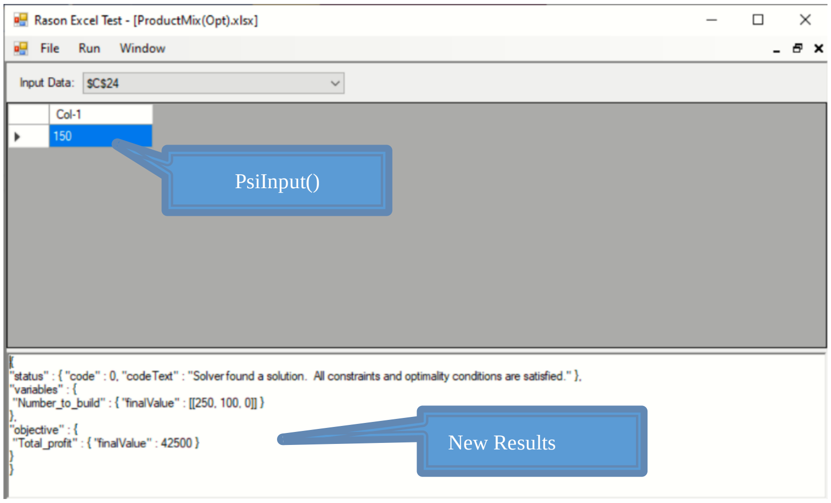
```
{
"status" : { "code" : 0, "codeText" : "Solver found a solution.
"variables" : {
 "Number_to_build" : { "finalValue" : [[200, 200, 0]] }
},
"objective" : {
 "Total_profit" : { "finalValue" : 25000 }
}
}
```

v

Now what if we wanted to change our selling price for a TV from $75 to $150. We can easily expose this parameter by using the PsiInput() function within the original excel model.

Click back to the Product Mix example and enter the following formula into a blank cell: =PsiInput(C24)

Now click Deploy Model – Solver SDK – Rason Model. Notice the grid at the top of the app. Enter 150 to change the price of a TV from $75 to $150. Then click Run – Optimize to solve the model with the new selling price.



```
{
"status" : { "code" : 0, "codeText" : "Solver found a solution.  All constraints and optimality conditions are satisfied." },
"variables" : {
 "Number_to_build" : { "finalValue" : [[250, 100, 0]] }
},
"objective" : {
 "Total_profit" : { "finalValue" : 42500 }
}
}
```

In order to view sensitivity information for the variables or constraints, or initial variable values, we could add PsiDualValue() and PsiInitialValue() to the original Excel model.

Click back to Excel and enter the following formulas into blank cells.

=PsiInitialValue(C14:E14) – Will report the variable initial values.

=PsiDualValue(G18:G22) – Will report the dual values for the constraints.

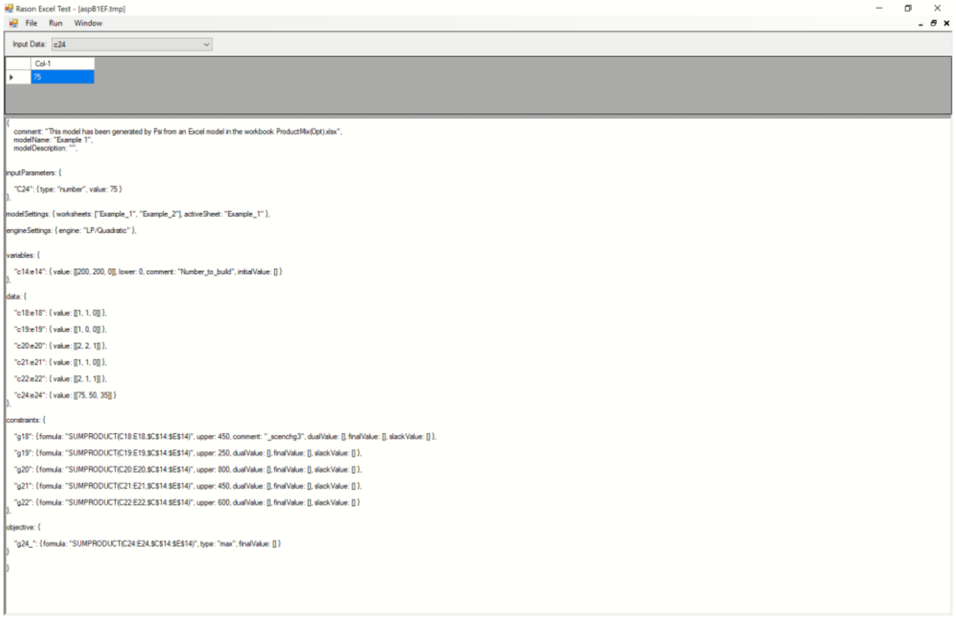
=PsiDualUpper(G18:G22) – Will report the dual upper values for the constraints.

=PsiDualLower(G18:G22) – Will report the dual lower values for the constraints.

=PsiFinalValue(G18:G22) – Will report the final values of the constraints.

=PsiSlackValue(G18:G22) – Will report the slack values of the constraints.

Now click Deploy Model – Solver SDK – RASON Model. Rason Excel Test opens and displays the RASON model.



```
{
  comment: "This model has been generated by Psi from an Excel model in the workbook ProductMix(Opt).xlsx",
  modelName: "Example 1",
  modelDescription: "",

inputParameters: {
  "C24": {type: "number", value: 75 }
},
modelSettings: { worksheets: ["Example_1", "Example_2"], activeSheet: "Example_1" },
engineSettings: { engine: "LP/Quadratic" },

variables: {
  "c14:e14": { value: [[200, 200, 0]], lower: 0, comment: "Number_to_build", initialValue: [] }
},
data: {
  "c18:e18": { value: [[1, 1, 0]] },
  "c19:e19": { value: [[1, 0, 0]] },
  "c20:e20": { value: [[2, 2, 1]] },
  "c21:e21": { value: [[1, 1, 0]] },
  "c22:e22": { value: [[2, 1, 1]] },
  "c24:e24": { value: [[75, 50, 35]] }
},
constraints: {
  "g18": {formula: "SUMPRODUCT(C18:E18,$C$14:$E$14)", upper: 450, comment: "_scenchg3", dualValue: [], finalValue: [], slackValue: [] },
  "g19": {formula: "SUMPRODUCT(C19:E19,$C$14:$E$14)", upper: 250, dualValue: [], finalValue: [], slackValue: [] },
  "g20": {formula: "SUMPRODUCT(C20:E20,$C$14:$E$14)", upper: 800, dualValue: [], finalValue: [], slackValue: [] },
  "g21": {formula: "SUMPRODUCT(C21:E21,$C$14:$E$14)", upper: 450, dualValue: [], finalValue: [], slackValue: [] },
  "g22": {formula: "SUMPRODUCT(C22:E22,$C$14:$E$14)", upper: 600, dualValue: [], finalValue: [], slackValue: [] }
},
objective: {
  "g24_": {formula: "SUMPRODUCT(C24:E24,$C$14:$E$14)", type: "max", finalValue: [] }
}
}
```

Now click Run – Optimize to solve the model and display the results.

```
{
  "status" : {
    "code" : 0,
    "codeText" : "Solver found a solution.  All constraints and
     optimality conditions are satisfied."
  },
  "variables" : {
     "c14:e14" : { "initialvalue" : [[200, 200, 0]] }
   },
   "objective" : {
     "g24_" : { "finalvalue" : 25000 }
   },
   "constraints" : {
      "g18" : {
        "duallower": 400,
        "dualupper": 1e+30,
        "dualvalue" : 0,
        "finalvalue" : 400,
        "slackvalue" : 50
      },
      "g19" : {
        "duallower": 400,
        "dualupper": 1e+30,
        "dualvalue" : 0,
        "finalvalue" : 200,
        "slackvalue" : 50
      },
      "g20" : {
        "duallower": 400,
        "dualupper": 1e+30,
        "dualvalue" : 12.5,
        "finalvalue" : 800,
        "slackvalue" : 0
      },
      "g21" : {
        "duallower": 400,
        "dualupper": 1e+30,
        "dualvalue" : 0,
        "finalvalue" : 400,
        "slackvalue" : 50
      },
      "g22" : {
        "duallower": 400,
        "dualupper": 1e+30,
        "dualvalue" : 25,
        "finalvalue" : 600,
        "slackvalue" : 0 }
      }
   }
```

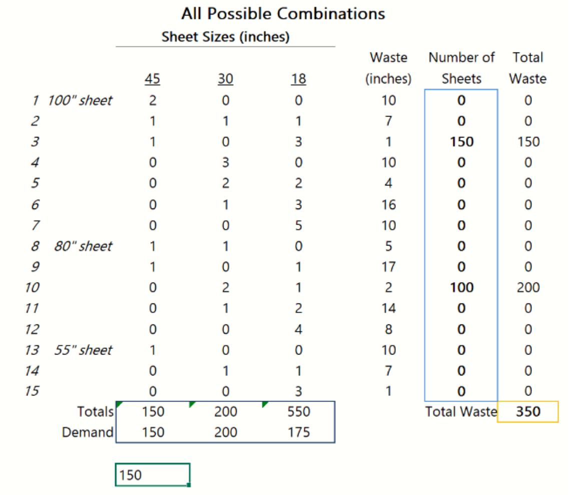
Excel Model

Click Help – Example Models –Optimization Examples and click the Cutting Stock link.

This model determines how to cut steel sheets, given steel sizes of 100, 80 and 55 inches, to meet demand in 3 sizes, 45, 30 and 18, while minimizing waste.

Enter "=PsiInput(D30)" in a blank cell to expose the demand for the 45" cut so that it can be changed within the SDK Windows app. This example uses cell D32.

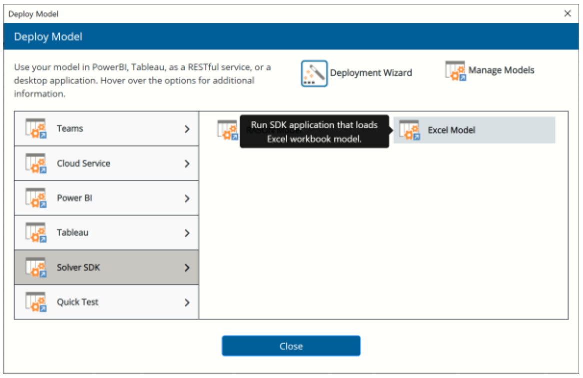
Click the green arrow on the task pane to solve the model. The number of sheets to cut are located under "Number of Sheets" and the cuts to make per sheet are listed under 45, 30 and 18, respectively.



All Possible Combinations

		Sheet Sizes (inches)			Waste	Number of	Total
		45	30	18	(inches)	Sheets	Waste
1	100" sheet	2	0	0	10	0	0
2		1	1	1	7	0	0
3		1	0	3	1	150	150
4		0	3	0	10	0	0
5		0	2	2	4	0	0
6		0	1	3	16	0	0
7		0	0	5	10	0	0
8	80" sheet	1	1	0	5	0	0
9		1	0	1	17	0	0
10		0	2	1	2	100	200
11		0	1	2	14	0	0
12		0	0	4	8	0	0
13	55" sheet	1	0	0	10	0	0
14		0	1	1	7	0	0
15		0	0	3	1	0	0
	Totals	150	200	550		Total Waste	350
	Demand	150	200	175			
		150					

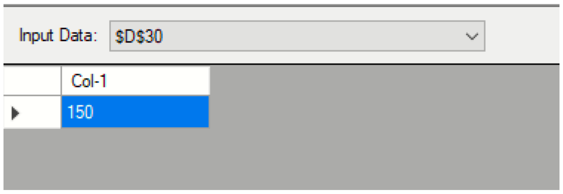
Click Deploy Model – Solver SDK – Excel Model.



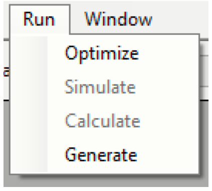
Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Run SDK application that loads Excel workbook model.

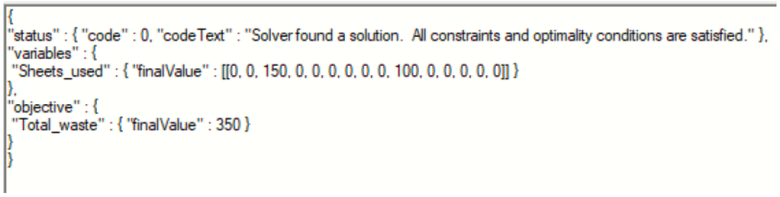
Immediately, the Rason Run Test Windows application opens. Notice the PsiInput parameters beneath Sheets_demand.



Click Run -- Optimize

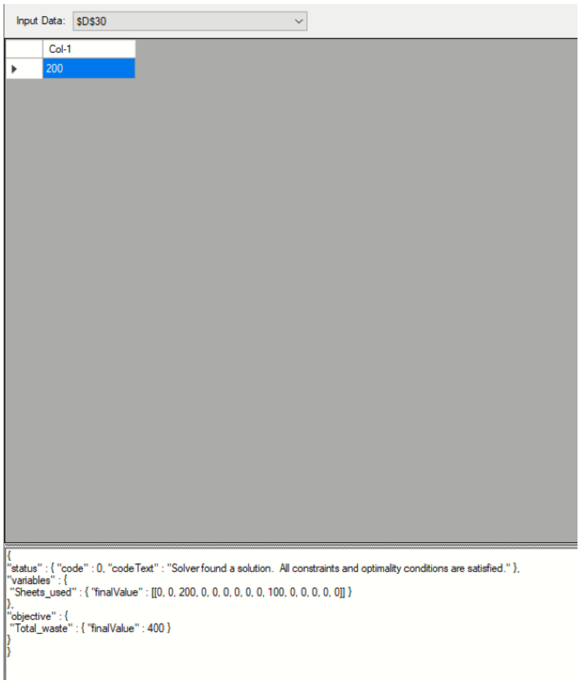


The results are displayed in the output window.



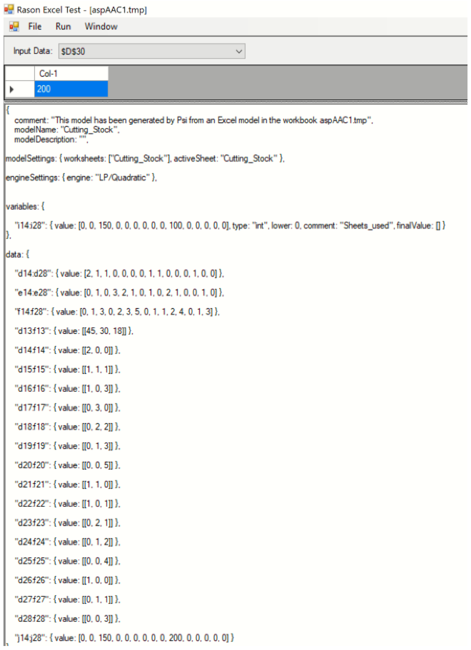
```
{
"status" : { "code" : 0, "codeText" : "Solver found a solution.  All constraints and optimality conditions are satisfied." },
"variables" : {
 "Sheets_used" : { "finalValue" : [[0, 0, 150, 0, 0, 0, 0, 0, 0, 100, 0, 0, 0, 0, 0]] }
},
"objective" : {
 "Total_waste" : { "finalValue" : 350 }
}
}
```

Now change the values of the input parameter (the demand for the 45" cut). Let's use: 200. Click Run Optimize again. The new results are displayed in the output window at the bottom of the app.



```
{
"status" : { "code" : 0, "codeText" : "Solver found a solution.  All constraints and optimality conditions are satisfied." },
"variables" : {
 "Sheets_used" : { "finalValue" : [[0, 0, 200, 0, 0, 0, 0, 0, 0, 100, 0, 0, 0, 0, 0]] }
},
"objective" : {
 "Total_waste" : { "finalValue" : 400 }
}
}
```

Click Run – Generate to generate the Rason model.



Input Data: $D$30

```
{
   comment: "This model has been generated by Psi from an Excel model in the workbook aspAAC1.tmp",
   modelName: "Cutting_Stock",
   modelDescription: "",

modelSettings: { worksheets: ["Cutting_Stock"], activeSheet: "Cutting_Stock" },

engineSettings: { engine: "LP/Quadratic" },


variables: {

   "i14:i28": { value: [0, 0, 150, 0, 0, 0, 0, 0, 0, 100, 0, 0, 0, 0, 0], type: "int", lower: 0, comment: "Sheets_used", finalValue: [] }
},

data: {

   "d14:d28": { value: [2, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0] },

   "e14:e28": { value: [0, 1, 0, 3, 2, 1, 0, 1, 0, 2, 1, 0, 0, 1, 0] },

   "f14:f28": { value: [0, 1, 3, 0, 2, 3, 5, 0, 1, 1, 2, 4, 0, 1, 3] },

   "d13:f13": { value: [[45, 30, 18]] },

   "d14:f14": { value: [[2, 0, 0]] },

   "d15:f15": { value: [[1, 1, 1]] },

   "d16:f16": { value: [[1, 0, 3]] },

   "d17:f17": { value: [[0, 3, 0]] },

   "d18:f18": { value: [[0, 2, 2]] },

   "d19:f19": { value: [[0, 1, 3]] },

   "d20:f20": { value: [[0, 0, 5]] },

   "d21:f21": { value: [[1, 1, 0]] },

   "d22:f22": { value: [[1, 0, 1]] },

   "d23:f23": { value: [[0, 2, 1]] },

   "d24:f24": { value: [[0, 1, 2]] },

   "d25:f25": { value: [[0, 0, 4]] },

   "d26:f26": { value: [[1, 0, 0]] },

   "d27:f27": { value: [[0, 1, 1]] },

   "d28:f28": { value: [[0, 0, 3]] },

   "j14:j28": { value: [0, 0, 150, 0, 0, 0, 0, 0, 0, 200, 0, 0, 0, 0, 0] }
```

Quick Test

When you choose **Deploy Model – Quick Test – Web Page**, Analytic Solver automatically translates your Excel model into Rason; then it creates a Web page (a file ending in .html) with some text boxes and buttons, and the Rason translation of your Excel model and data – embedded in the JavaScript code on the Web page. This is a working version of your model that runs on the Web – if you click the Quick Solve or Solve button, after a moment you'll see the same solution values that you would see in Excel, but returned in JSON (JavaScript Object Notation).

When you choose **Deploy Model – Quick Test – Rason Model**, Analytic Solver automatically translates your Excel model into Rason. To show you the new Rason model, Analytic Solver opens a new Web browser to www.Rason.com, logins as you, uploads the translated model text and displays it in Rason's Model Editor. You can solve the model using Rason's menu Quick Run – Optimize\Simulate\Decision\Solve endpoints.

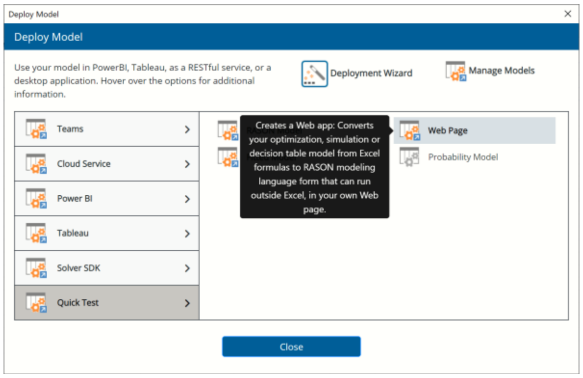
Whether you keep your model in Excel or translate it to Rason the big difference is that your model can be used wherever / whenever it is needed – on the factory floor, on a salesperson's laptop or smartphone, or in a call center custom application. And Rason can get the updated data your model needs directly from operational business systems. This is amazingly easy if you are using Power BI, Power Apps, Power Automate or Dynamics 365.

Use **Deploy Model – Quick Test – Fitted Model** to save a PMML (Predictive Model Markup Language) model to your local storage. PMML is an industry-standard XML based predictive model format supported by Analytic Solver Data Science for a full range of data transformation, forecasting, text mining or classification/prediction models. Analytic Solver will scan your workbook and allow you to choose the fitted model you wish to save. This saved PMML model can be used to score new data in either Analytic Solver Data Science or any environment that supports the PMML standard.

Select **Deploy Model – Quick Test – Probability Model** to deploy and share probability models, following the open Probability Management 3.0 standard. Using these Shared Information Probability resources (SIPs, also known as “Stochastic Information Packets”), you can ensure that your group or organization uses consistent data about uncertain/risky variables across simulation or decision models, enabling model results to be meaningfully compared. Analytic Solver will scan your workbook and allow you to choose the probability model you wish to save. This saved model can be imported into Analytic Solver or any environment that supports the Probability Management 3.0 standard.

Quick Test – Web Page

Analytic Solver includes the ability to bypass months of development work by creating a web application that solves the optimization, simulation, stochastic optimization, simulation optimization and/or decision table model in Excel. Click back to Excel, and the Product Mix example, then click Deploy Model – Quick Test -- Web Page.



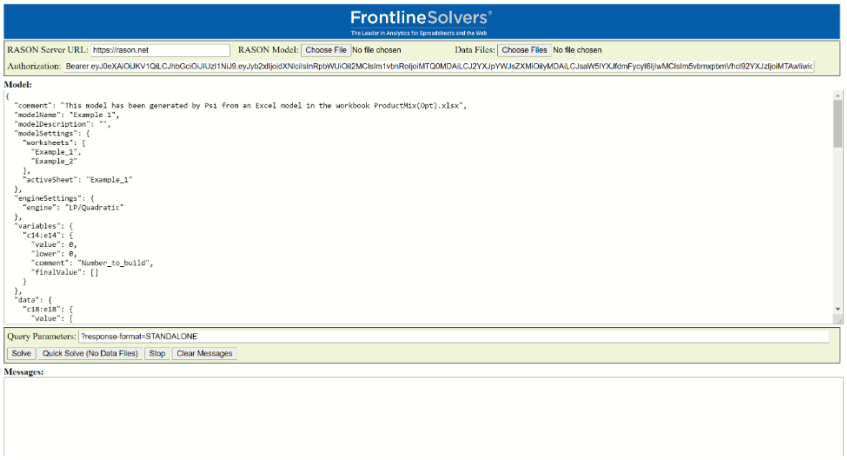
Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Creates a Web app: Converts your optimization, simulation or decision table model from Excel formulas to RASON modeling language form that can run outside Excel, in your own Web page.

NOTE: When creating a RASON model through desktop Analytic Solver, model elements (objective function, decision variables, constraints, uncertain variables and/or uncertain functions) may now appear on multiple worksheets within the same Excel workbook.

Immediately, a browser opens displaying the newly created Web Application.

See *Note on RASON Subscriptions* in the section above for log in and subscription information pertaining to solving models via the RASON Server.



RASON Server URL: https://rason.net

RASON Model: Choose File No file chosen

Data Files: Choose Files No file chosen

Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJyb2xlIjoidXNlciIsInRpbWUiOiI2MCIsIm1vbnRoIjoiMTQ0MDAiLCJ2YXJpYWJsZXMiOiIyMDAiLCJsaW5lYXJfdmFycyI6IjIwMCIsIm5vbmxpbmVhcl92YXJzIjoiMTAwIiwic

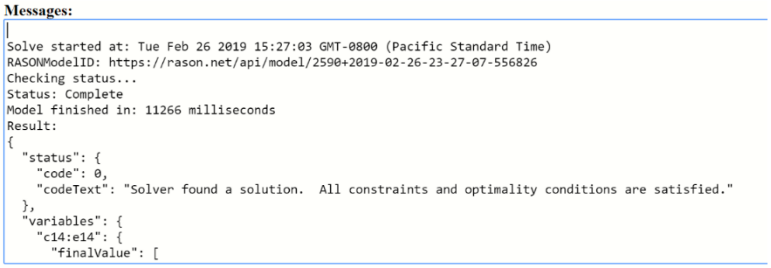
Model:

```
{
  "comment": "This model has been generated by Psi from an Excel model in the workbook ProductMix(Opt).xlsx",
  "modelName": "Example 1",
  "modelDescription": "",
  "modelSettings": {
    "worksheets": [
      "Example_1",
      "Example_2"
    ],
    "activeSheet": "Example_1"
  },
  "engineSettings": {
    "engine": "LP/Quadratic"
  },
  "variables": {
    "c14:e14": {
      "value": 0,
      "lower": 0,
      "comment": "Number_to_build",
      "finalValue": []
    }
  },
  "data": {
    "c18:e18": {
      "value": [
```

Query Parameters: ?response-format=STANDALONE

FrontlineSolvers
The Leader in Analytics for Spreadsheets and the Web
Solve
Quick Solve (No Data Files)
Stop
Clear Messages
Messages:

Click **Solve**.



```
Solve started at: Tue Feb 26 2019 15:27:03 GMT-0800 (Pacific Standard Time)
RASONModelID: https://rason.net/api/model/2590+2019-02-26-23-27-07-556826
Checking status...
Status: Complete
Model finished in: 11266 milliseconds
Result:
{
  "status": {
    "code": 0,
    "codeText": "Solver found a solution.  All constraints and optimality conditions are satisfied."
  },
  "variables": {
    "c14:e14": {
      "finalValue": [
```

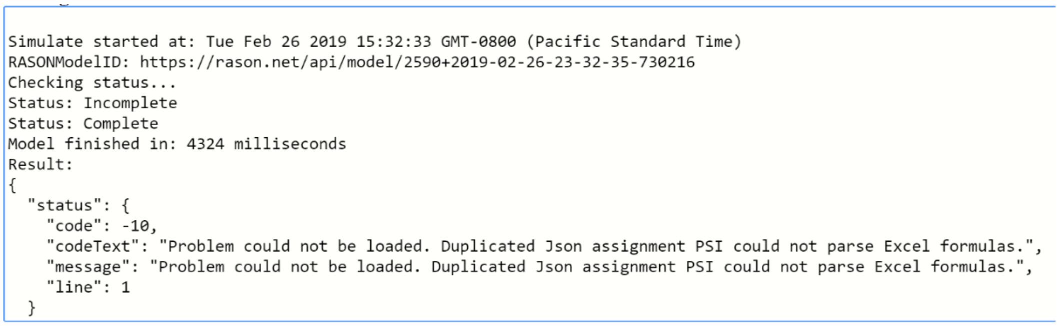
Automatically, the model is submitted to the RASON Server and the optimization is started using the RASON Rest API Endpoint *POST rason.net/api/model/id/solve*, the status is checked using the Endpoint *GET rason.net/api/model/id/status* and the results are obtained via the Endpoint *GET rason.net/api/model/id/result*. All this, at the click of a button.

Clicking the Quick Solve command button will call the Quick Solve endpoint. Try it for yourself – the results from the Quick Solve will be appended in the Result window.

Click Stop to stop an already executing optimization and Clear Messages to clear the output window.

Click back to Excel and the Business Forecast example, and click Deploy Model – Quick Test **--** Web Page. A new browser open displaying a new Web Application. This Web App solves the Business Forecast simulation model.

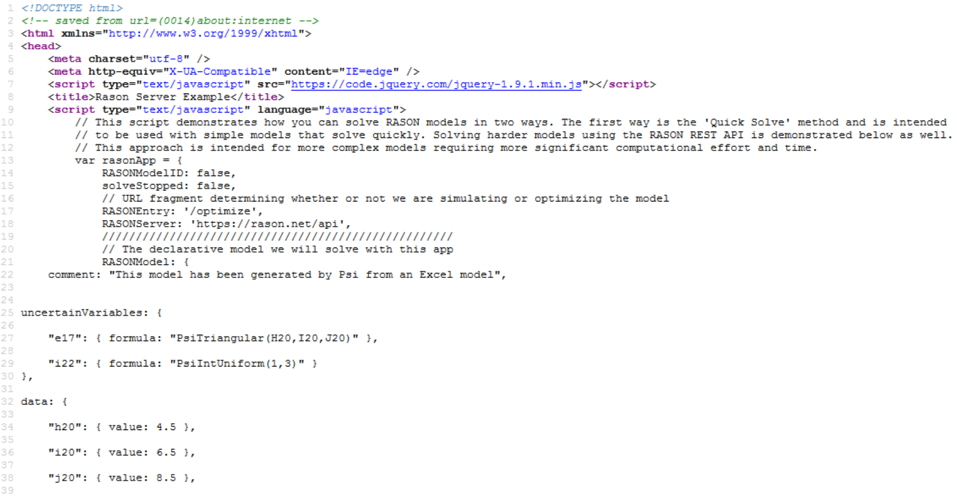
Click Solve. Automatically, the model is submitted to the RASON Server and the simulation is started using the RASON Rest API Endpoint *POST rason.net/api/model/id/solve*, the status is checked using the Endpoint *GET rason.net/api/model/id/status* and the results are obtained via the Endpoint *GET rason.net/api/model/id/result*. All this, at the click of a button.



```
Simulate started at: Tue Feb 26 2019 15:32:33 GMT-0800 (Pacific Standard Time)
RASONModelID: https://rason.net/api/model/2590+2019-02-26-23-32-35-730216
Checking status...
Status: Incomplete
Status: Complete
Model finished in: 4324 milliseconds
Result:
{
  "status": {
    "code": -10,
    "codeText": "Problem could not be loaded. Duplicated Json assignment PSI could not parse Excel formulas.",
    "message": "Problem could not be loaded. Duplicated Json assignment PSI could not parse Excel formulas.",
    "line": 1
  }
```

Clicking the Quick Solve command button will call the Quick Solve endpoint.

To view the code used in the Web Application, right click (on the shaded blue region at the top of the page) and select **View Page Source** (if using Mozilla Firefox or Google Chrome) or **View Source** (if using Internet Explorer). A browser will open displaying the code for the Web Application. This code gives examples of how to call all RASON REST API Endpoints. Please see the RASON User Guide for a description of each along with complete information about the RASON Rest API.

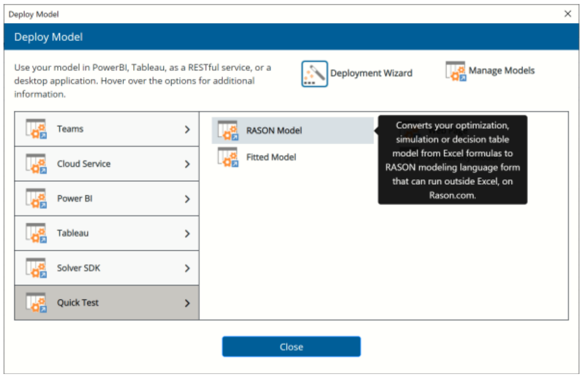


```
<!DOCTYPE html>
<!-- saved from url=(0014)about:internet -->
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <script type="text/javascript" src="https://code.jquery.com/jquery-1.9.1.min.js"></script>
    <title>Rason Server Example</title>
    <script type="text/javascript" language="javascript">
        // This script demonstrates how you can solve RASON models in two ways. The first way is the 'Quick Solve' method and is intended
        // to be used with simple models that solve quickly. Solving harder models using the RASON REST API is demonstrated below as well.
        // This approach is intended for more complex models requiring more significant computational effort and time.
        var rasonApp = {
            RASONModelID: false,
            solveStopped: false,
            // URL fragment determining whether or not we are simulating or optimizing the model
            RASONEntry: '/optimize',
            RASONServer: 'https://rason.net/api',
            ////////////////////////////////////////////////////
            // The declarative model we will solve with this app
            RASONModel: {
    comment: "This model has been generated by Psi from an Excel model",


uncertainVariables: {

    "e17": { formula: "PsiTriangular(H20,I20,J20)" },

    "i22": { formula: "PsiIntUniform(1,3)" }
},

data: {

    "h20": { value: 4.5 },

    "i20": { value: 6.5 },

    "j20": { value: 8.5 },

```

Quick Test – Rason Model

When you choose Deploy Model – Quick Test – Rason Model, Analytic Solver automatically translates your Excel model into Rason.

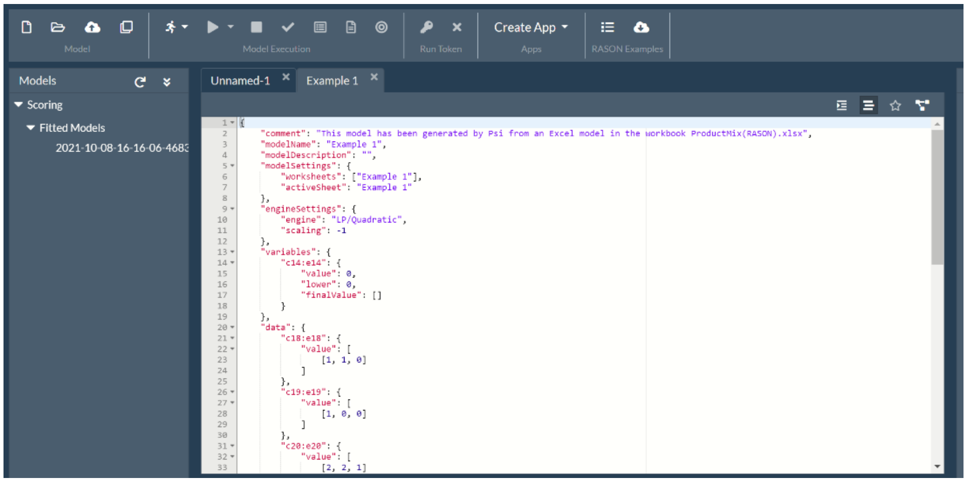
Click back to Excel and the Product Mix model and click Deploy Model – Quick Test – Rason Model.



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

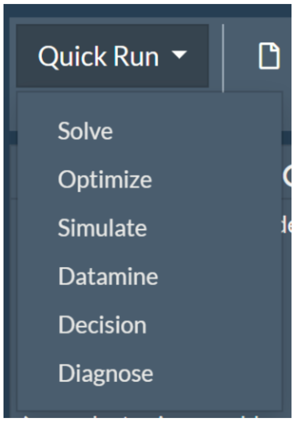
Converts your optimization, simulation or decision table model from Excel formulas to RASON modeling language form that can run outside Excel, on Rason.com.

To show you the new Rason model, Analytic Solver opens a new Web browser to www.Rason.com, logins as you, uploads the translated model text and displays it in Rason's Model Editor.

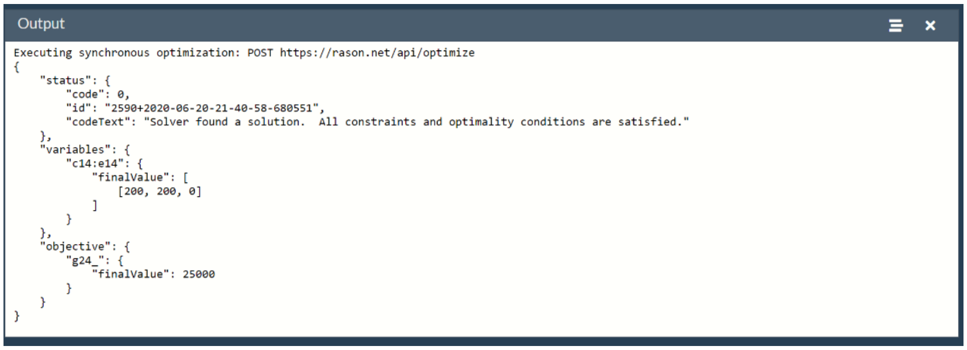


```
{
    "comment": "This model has been generated by Psi from an Excel model in the workbook ProductMix(RASON).xlsx",
    "modelName": "Example 1",
    "modelDescription": "",
    "modelSettings": {
        "worksheets": ["Example 1"],
        "activeSheet": "Example 1"
    },
    "engineSettings": {
        "engine": "LP/Quadratic",
        "scaling": -1
    },
    "variables": {
        "c14:e14": {
            "value": 0,
            "lower": 0,
            "finalValue": []
        }
    },
    "data": {
        "c18:e18": {
            "value": [
                [1, 1, 0]
            ]
        },
        "c19:e19": {
            "value": [
                [1, 0, 0]
            ]
        },
        "c20:e20": {
            "value": [
                [2, 2, 1]
```

You can solve the optimization model using Rason's menu Quick Run – Optimize or Quick Run – Solve endpoints.



Model results are displayed in the Output window.



```
Executing synchronous optimization: POST https://rason.net/api/optimize
{
    "status": {
        "code": 0,
        "id": "2590+2020-06-20-21-40-58-680551",
        "codeText": "Solver found a solution.  All constraints and optimality conditions are satisfied."
    },
    "variables": {
        "c14:e14": {
            "finalValue": [
                [200, 200, 0]
            ]
        }
    },
    "objective": {
        "g24_": {
            "finalValue": 25000
        }
    }
}
```

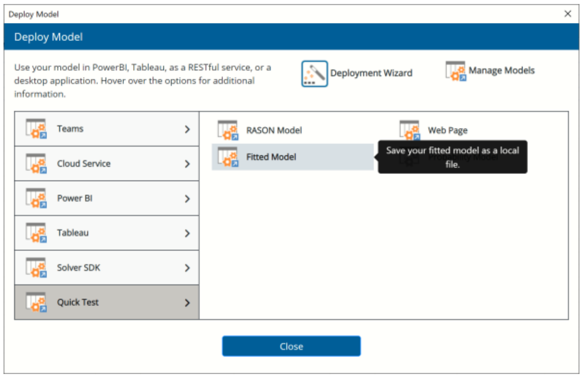
Rason is the easiest way to transition into the world of web, mobile and cloud-base applications. For more information on Frontline's Rason Cloud Service, visit www.rason.com.

Quick Test – Fitted Model

Use **Deploy Model – Quick Test – Fitted Model** to save a PMML (Predictive Model Markup Language) model to your local storage.

Open the Scoring data science example by clicking Help – Example Models – Forecasting / Data Science Examples. This example workbook contains a PMML model generated from the Linear Regression prediction method within the LinReg_Stored worksheet.

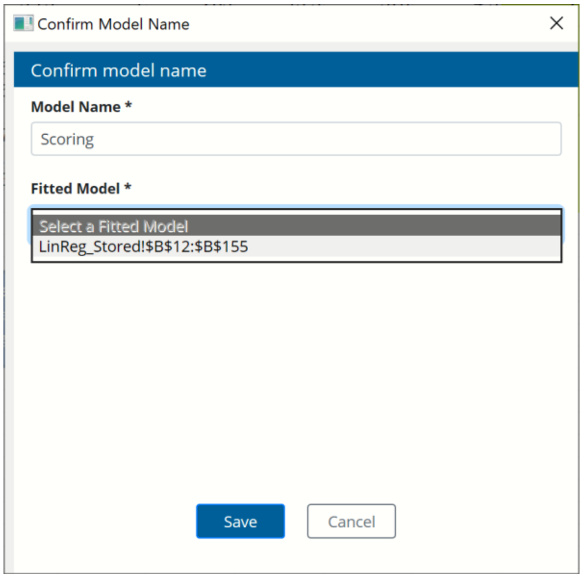
Click Deploy Model – Quick Test – Fitted Model…



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Save your fitted model as a local file.

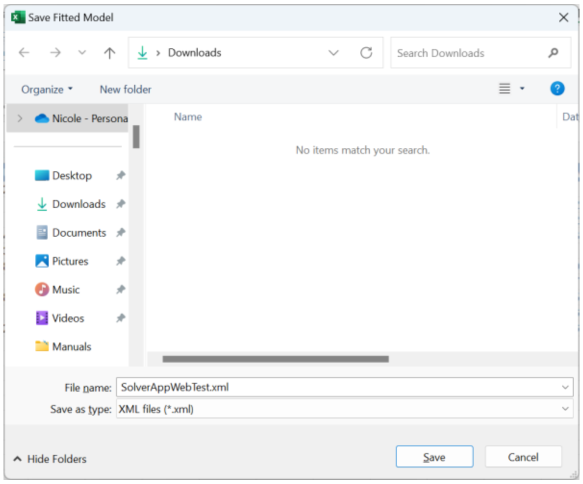
… then click the down arrow beneath Fitted Model and select LinReg_Stored!$B$12:$B$155. Click Save.



Model Name * Scoring

Fitted Model * Select a Fitted Model
LinReg_Stored!$B$12:$B$155

Enter a name for the XML file. This example uses the name SolverAppWebTest.xml. Then click Save.



No items match your search.

File name: SolverAppWebTest.xml

Save as type: XML files (*.xml)

That's it! Your PMML model, saved in XML format, may now be used to score new data. This saved model can be used to score new data in either Analytic Solver Data Science or any environment that supports the PMML standard.

Scoring in Analytic Solver Data Science

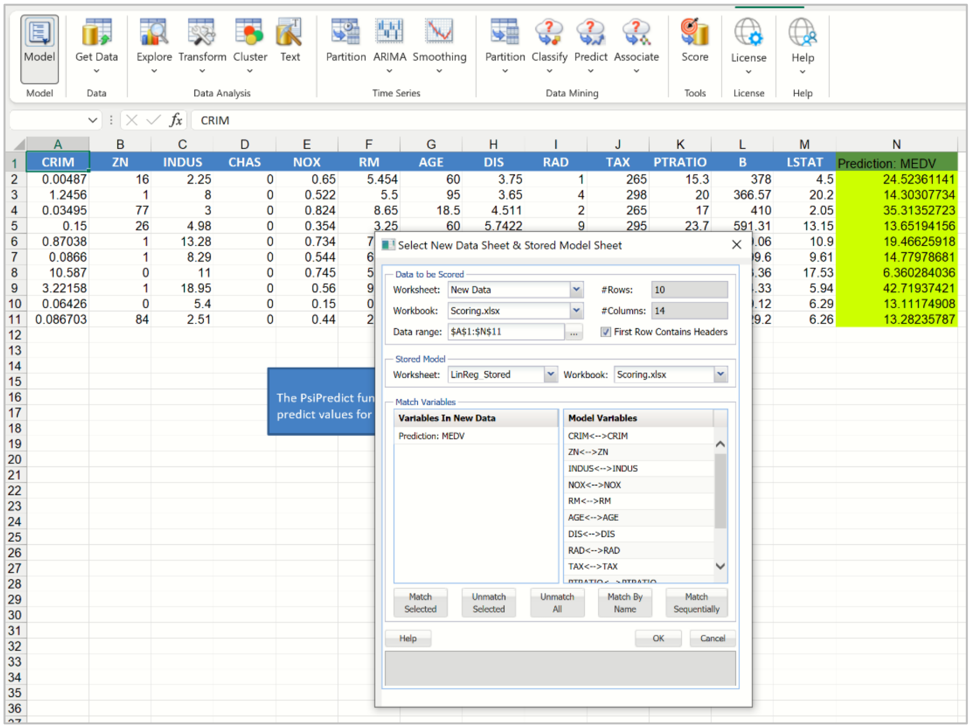
To use this PMML model in Analytic Solver Data Science, click back to the Scoring example in Excel and select the New Data worksheet. Click Data Science – Score to open the Scoring dialog.

Confirm that the Data to be Scored information and the Stored Model information matches what is shown on the screenshot below.

For the Data to be Scored information, the selected Worksheet should be New Data within the Scoring.xlsx workbook. Confirm that the data range is A1:N11.

For the Stored Model information, the selected Worksheet should be LinReg_Stored within the Scoring.xlsx workbook.

Click Match by Name to match the variables in the stored model (the PMML model, LinReg_Stored) with the variables on the New Data tab. Afterwards, click OK.



CRIM

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Prediction: MEDV
2	0.00487	16	2.25	0	0.65	5.454	60	3.75	1	265	15.3	378	4.5	24.52361141
3	1.2456	1	8	0	0.522	5.5	95	3.65	4	298	20	366.57	20.2	14.30307734
4	0.03495	77	3	0	0.824	8.65	18.5	4.511	2	265	17	410	2.05	35.31352723
5	0.15	26	4.98	0	0.354	3.25	60	5.7422	9	295	23.7	591.31	13.15	13.65194156
6	0.87038	1	13.28	0	0.734	7						.06	10.9	19.46625918
7	0.0866	1	8.29	0	0.544	6						9.6	9.61	14.77978681
8	10.587	0	11	0	0.745	5						.36	17.53	6.360284036
9	3.22158	1	18.95	0	0.56	9						.33	5.94	42.71937421
10	0.06426	0	5.4	0	0.15	0						.12	6.29	13.11174908
11	0.086703	84	2.51	0	0.44	2						9.2	6.26	13.28235787

Select New Data Sheet & Stored Model Sheet

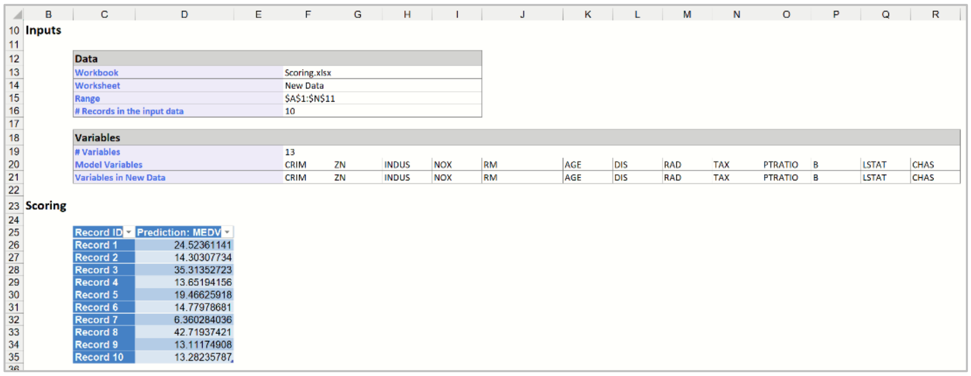
Data to be Scored
Worksheet: New Data
Workbook: Scoring.xlsx
Data range: $A$1:$N$11
#Rows: 10
#Columns: 14
☑ First Row Contains Headers

Stored Model
Worksheet: LinReg_Stored
Workbook: Scoring.xlsx

Match Variables

Variables In New Data	Model Variables
Prediction: MEDV	CRIM<-->CRIM
	ZN<-->ZN
	INDUS<-->INDUS
	NOX<-->NOX
	RM<-->RM
	AGE<-->AGE
	DIS<-->DIS
	RAD<-->RAD
	TAX<-->TAX

A new worksheet, Scoring_LinearRegression is inserted into the workbook to the left of the New Data worksheet. The new predictions (scores) for the data contained in the New Data worksheet is written to cells C25:D35.



Inputs

Data	
Workbook	Scoring.xlsx
Worksheet	New Data
Range	$A$1:$N$11
# Records in the input data	10

Variables													
# Variables	13												
Model Variables	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	CHAS
Variables in New Data	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	CHAS

Scoring

Record ID	Prediction: MEDV
Record 1	24.52361141
Record 2	14.30307734
Record 3	35.31352723
Record 4	13.65194156
Record 5	19.46625918
Record 6	14.77978681
Record 7	6.360284036
Record 8	42.71937421
Record 9	13.11174908
Record 10	13.28235787

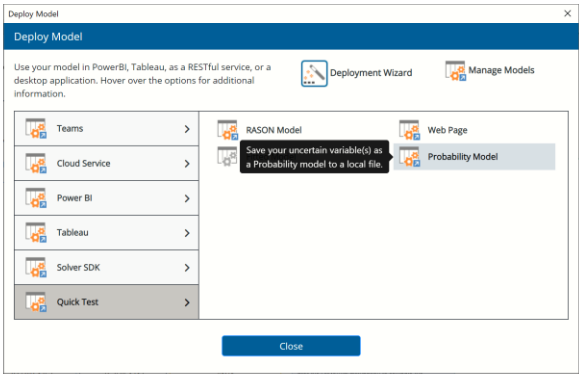
Notice that the Inputs section in cells C12:R21 contains a description of the scoring inputs such as the name of the workbook, worksheet, data range and the Model variables matched to the Variables in the New Data.

Quick Test – Probability Model

Select **Deploy Model – Quick Test – Probability Model** to deploy and share probability models, following the open Probability Management 3.0 standard.

Open the College Fund Growth 1 simulation example by clicking Help – Example Models – Monte Carlo Simulation Examples. This example workbook contains a simulation model that fits a distribution to past data, specifically S&P 500 returns from the 1926 to 2019. This example will export the fitted distribution from cell O17.

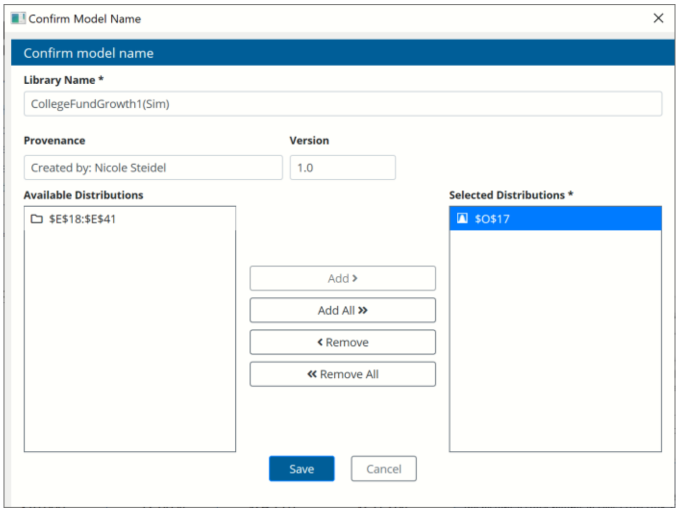
Click Deploy Model – Quick Test – Probability Model.



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Save your uncertain variable(s) as a Probability model to a local file.

Under Available Distributions select O17 and then click the Add button to select the distribution contained within this cell. Note: To add additional distributions, click E18:E41 under Available Distributions and then select the desired distributions or cell Add All to add all distributions as Selected Distributions. Click Save to create the PMML model.



Library Name *: CollegeFundGrowth1(Sim)

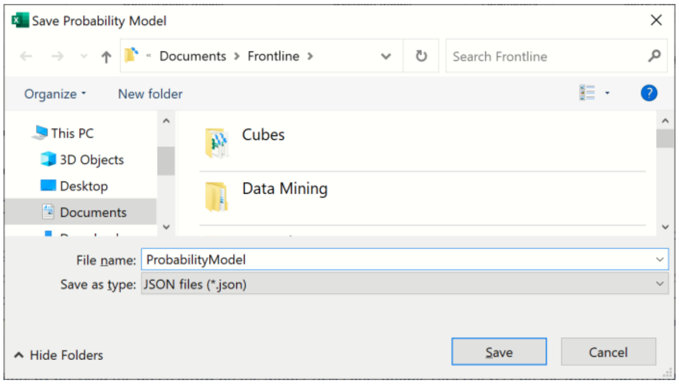
Provenance: Created by: Nicole Steidel

Version: 1.0

Available Distributions: $E$18:$E$41

Selected Distributions *: $O$17

Enter a name for the Probability Model file. This example uses the name ProbabilityModel.json. Click Save.



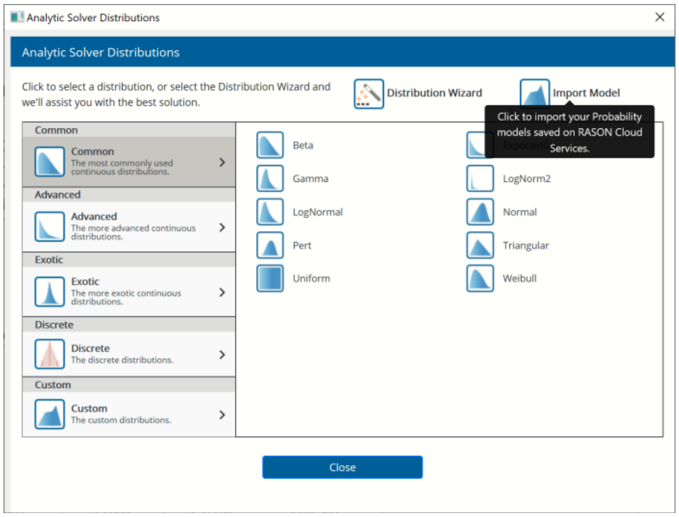
File name: ProbabilityModel

Save as type: JSON files (*.json)

That's it! The fitted distribution has now been saved to a JSON file. This saved probability model can be imported to any environment that supports the Probability Management 3.0 Standard.

Importing a Probability Model in Analytic Solver

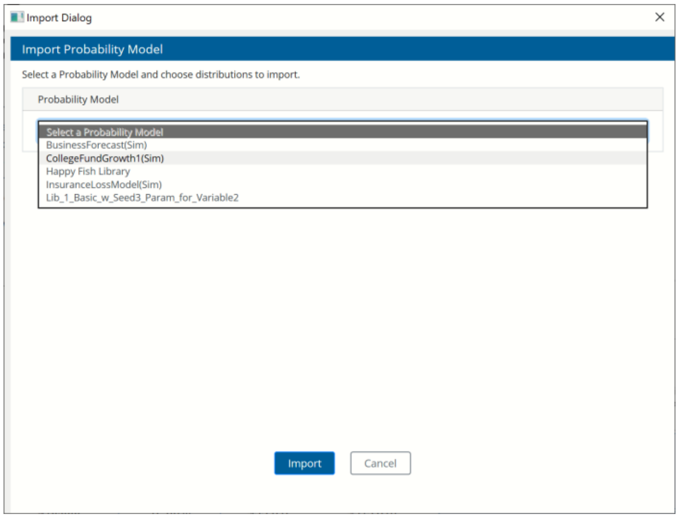
To import the Probability Model into Analytic Solver, open a new worksheet, and click Distributions on the Ribbon, then Import Model.


Analytic Solver Distributions
Analytic Solver Distributions
Distribution Wizard
Import Model
Common
Common
The most commonly used continuous distributions.
Advanced
Advanced
The more advanced continuous distributions.
Exotic
Exotic
The more exotic continuous distributions.
Discrete
Discrete
The discrete distributions.
Custom
Custom
The custom distributions.
Beta
Gamma
LogNormal
Pert
Uniform
LogNorm2
Normal
Triangular
Weibull
Close

Click to select a distribution, or select the Distribution Wizard and we'll assist you with the best solution.

Click to import your Probability models saved on RASON Cloud Services.

Select the appropriate model, in this case CollegeFundGrowth1(Sim), and click Import.

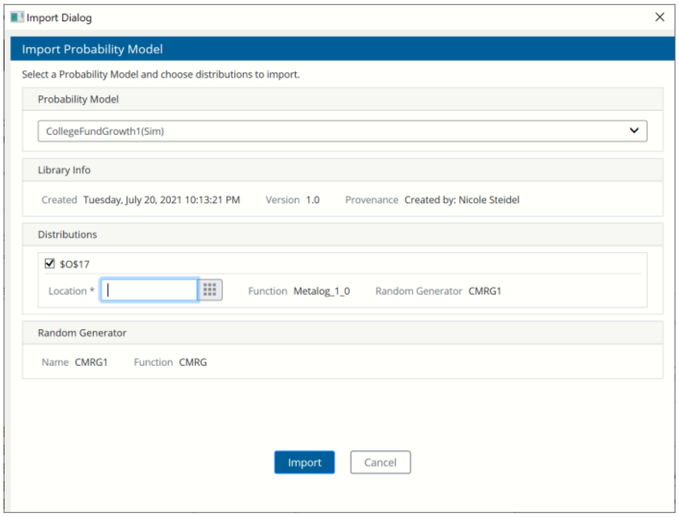


Select a Probability Model and choose distributions to import.

Probability Model

- Select a Probability Model
- BusinessForecast(Sim)
- CollegeFundGrowth1(Sim)
- Happy Fish Library
- InsuranceLossModel(Sim)
- Lib_1_Basic_w_Seed3_Param_for_Variable2

The dialog will expand to include the following new options.



Select a Probability Model and choose distributions to import.

Probability Model: CollegeFundGrowth1(Sim)

Library Info

Created Tuesday, July 20, 2021 10:13:21 PM Version 1.0 Provenance Created by: Nicole Steidel

Distributions

☑ $O$17

Location * Function Metalog_1_0 Random Generator CMRG1

Random Generator

Name CMRG1 Function CMRG

Select the O17 distribution and then enter the cell address of a blank cell for Location, then click Import.

The following fitted distribution will be inserted into the blank cell address. This is the PsiMetalog function from cell O17 that was fit to the historical S&P 500 returns in the College Fund Growth 1 example model.

=PsiMetalog(,,{0.137088248307783,235.996630065143,-41.6761363388904,-943.412342399359,166.781999403805,-1781.91992354393,5864.72184371948,265.6771517599,-850.31248220655,3796.63619041443,-8596.89170408249,-404.07730274836,835.784252876499,-1707.57994881272}, PsiName("$O$17"))

Using Custom Functions

Introduction

In years past, Excel-savvy analysts used VBA (Visual Basic for Applications) to define their own custom functions. While this still works in Excel Desktop using COM, VBA functions are *not* supported in Excel for the Web – and according to Microsoft, VBA and COM will *never* move to the cloud. If you want your custom functions to work in *both* desktop and cloud, your options have been limited – until now:

Microsoft has introduced new Excel functions LAMBDA and LET. These are very special because you can use them in Excel formulas to define your own custom functions. The Excel community has expressed much excitement over these new functions, since they effectively make Excel a “complete programming language”.

On another front, there’s the open standard known as DMN (Decision Model and Notation) – a business user-friendly “formula language” used to define business rules and decision tables, supported in “decision management” platforms from various vendors, and in Analytic Solver and RASON since 2019. DMN – now in version 1.3 – offers a way to define your own custom functions, known as “Box functions”.

Analytic Solver Comprehensive includes support for *both* Excel’s LAMBDA and LET functions, and for DMN-compatible Box functions. This chapter explains how to use both approaches. LAMBDA and LET, and DMN Box functions enjoy full support from our PSI Interpreter – which means that our full range of Solver Engines, and our high-speed Monte Carlo simulation engine “understand” and take full advantage of custom functions that you define this way. This can yield better results than you’ve ever had with VBA-based functions that are embedded in an optimization or simulation model.

Note: Custom Functions are only supported in Analytic Solver Comprehensive. The Box Function button on the ribbon will be disabled for users of Analytic Solver Optimization/Simulation/Data Science, Analytic Solver Upgrade and Solver Academy licenses.

LAMBDA Function Design

Microsoft Excel's LAMBDA function allows users to define a custom function written in Excel's formula language.

Suppose a user wanted to create a function that computes the Total Profit of a company that produces TV, Stereos and Speakers. (Recall this is exactly what the Product Mix example model does in the Optimization chapter that occurred previously in this guide.)

=LAMBDA(TV,Stereos,Speakers,75*TV + 50*Stereos + 35* Speakers)

The LET function can be used in conjunction with the LAMBDA function to give:

=LAMBDA(TV,Stereos,Speakers,LET(TV,TV*75Stereos,50*Stereos,Speakers, 35*Speakers,TV+Stereos+Speakers))

The function takes three arguments named TV, Stereos and Speakers, binds the value of 75 * TV to the name TV, the value of 50*Stereos to Stereos, the value of 35*Speakers to Speakers and TV + Stereos + Speakers as the result.

The Name Manager in Excel allows a name to be assigned to the formula. If this function is named TotalProfit, then the formula TotalProfit(200,200,0) would evaluate to 25,000.

Note: According to Microsoft's Support for the Lambda function, it's best to create the function within an Excel cell before adding to the Name Manager, in order to determine if the custom lambda function is calculating correctly.

Once the function is created, or named, the function can be called by that name thereby eliminating the need to enter this formula multiple times in the spreadsheet.

Note: Although in Excel a LAMBDA can be an argument to another LAMBDA, this behavior is not supported in Analytic Solver.

LAMBDA Example Models

Optimization Example

Open the Airport Box Function example (Help – Example Models – Optimization Models) to view an example of how a custom lambda function could be used in an optimization model.

Original Formulation Recap

As discussed in the earlier Optimization chapter, the original model formulation is in the first worksheet, Original Formulation. This example uses the distance formula in cells G14:G19 to find the optimal location (using X and Y coordinates) of a proposed airport. The airport's location should be as equidistant as possible between each of the six cities.

- The decision variables in this model are in cells E13, F13 and G20.
- The constraints in this model are G14:G19 <= G20. Cells G14:G19 use the distance formula (SQRT(Xc – X)^2 + (Yc – Y) ^2) to calculate the distance between the proposed airport location and each city.
- The objective to be Minimized is the decision variable, G20.

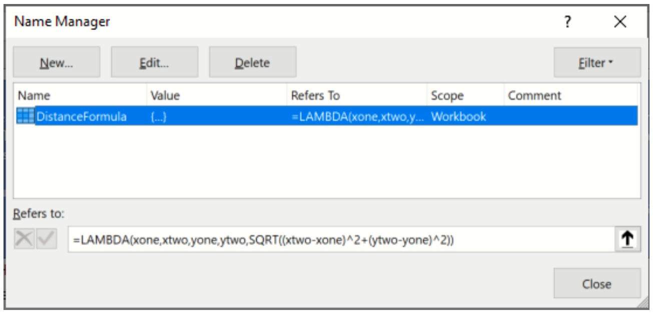
The objective in cell G20 is being minimized which "squishes" the values in cells G14:G19 to be as small as possible. This is what is referred to as a Mini-Max problem; G20 will equal the maximum of G14:G19 in the final solution.

LAMBDA Function Formulation

On the Lambda Function tab, the model has been reformulated to use the Microsoft Excel LAMBDA function to calculate the distance between the proposed airport location and each of the six cities in cells G14:G19. The LAMBDA function, named DistanceFormula, is defined within Excel's Name Manager as:

DistanceFormula=LAMBDA(Xone,Xtwo,Yone,Ytwo, SQRT((Xtwo-Xone)*2 + (Ytwo-Yone)^2))

Notice the order of the arguments, Xone, Xtwo, Yone, Ytwo. When invoked in the Excel cell, the arguments must be passed in this same order.



Name	Value	Refers To	Scope	Comment
DistanceFormula	{...}	=LAMBDA(xone,xtwo,y...	Workbook	

Refers to: =LAMBDA(xone,xtwo,yone,ytwo,SQRT((xtwo-xone)^2+(ytwo-yone)^2))

Invoking the LAMBDA Function

Rather than typing the distance formula into cells G14:G19, this formulation of the model calls the newly created DistanceFormula custom function to calculate the distance between the proposed airport site and each city.

In the Excel sheet, Cell G14 contains the formula:

=DistanceFormula($E$13, E14, $F$13, F14)

Following the order of the arguments in the LAMBDA function: Xone = E13, Xtwo = E14, Yone = F13 and Ytwo = F14.

This custom function in cell G14 calculates as:

SQRT((E14 - E13)^2 + (F14 - F13)^2)

The remaining functions in cells G15:G19 calculate similarly.

Click LAMBDA and LET for Microsoft Excel documentation.

Solving the Model

Click the green arrow on the task pane to solve the model. Notice that Analytic Solver is able to find an optimal solution with an objective equal to 3.5355 and final variable values equal to X = 3.5 and Y = 4.5. This is the same optimal solution found with the original formulation.

Note: Recall that according to Microsoft's Support for the Lambda function, it's best to create the function within an Excel cell before adding to the Name Manager, in order to determine if the custom lambda function is calculating correctly. This is exactly why you see the LAMBDA function in cells H14:H19.

Simulation Example

Open the GBM Simple Model Box Function example (Help – Example Models – Simulation Models) to view an example of how a custom box function could be used in an simulation model.

Original Formulation

As mentioned above, the original model formulation is in the first worksheet, Original Formulation. This example predicts the average daily closing price of a stock based on it's previous closing price.

- The uncertain variables in this model are in cells D13:D36.
- The uncertain functions in this model are in cells F13:F36. Recall that these cells are designated as uncertain functions by the statistic PsiMean that appears in cells H13:H36.
- Cells H13:H36 contain the PsiMean() function which finds the expected value of cells F13:F36.

In the original formulation of the model, cells F13 contains the formula:

```
=(F12*(EXP(($K$18-$K$17*$K$17*0.5)*(1/$K$13))+($K$17*SQRT(1/$K$13)*D13)))
```

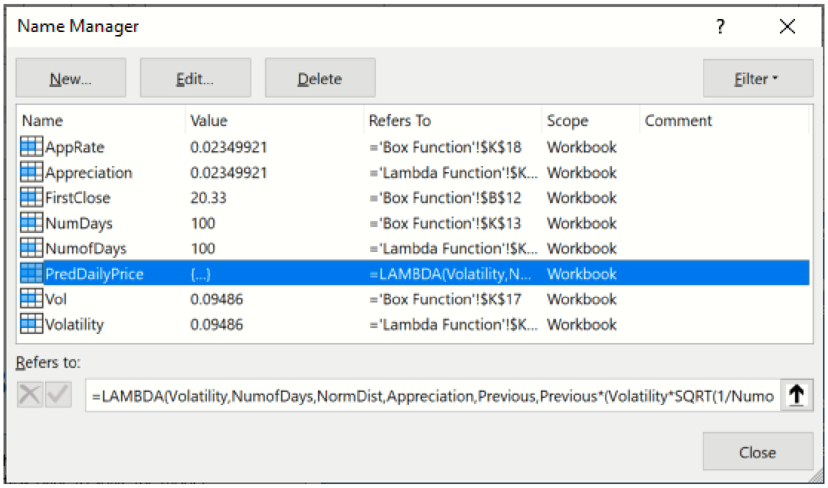
As mentioned previously, it's easy to imagine how easy it would be to accidentally introduce an error when entering this intricate formula. To alleviate this challenge, and the challenge of searching for an error in a spreadsheet with formulas such as this, Analytic Solver now fully supports the use of Microsoft's LAMBDA function which can be defined once and reused repeatedly. Custom functions reduce "bugs" in the spreadsheet by only requiring an complex function to be defined once.

LAMBDA Function Formulation

On the Lambda Function tab, the model has been reformulated to use the Microsoft Excel LAMBDA function to calculate the predicted Daily Price of a stock. The LAMBDA function, named PredDailyPrice, is defined within Excel's Name Manager (Formulas – Name Manager) as:

PredDailyPrice=LAMBDA(Volatility,NumofDays,NormDist,Appreciation,Previous,Previous*(Volatility*SQRT(1/NumofDays)*NormDist)+ EXP((Appreciation-Volatility^2*0.5)*(1/NumofDays)))

As with the previous optimization example, notice the order of the arguments: Volatility, NumofDays,NormDist,Appreciation, Previous. When invoked in the Excel cell, the arguments must be passed in this same order.



Name	Value	Refers To	Scope	Comment
AppRate	0.02349921	='Box Function'!$K$18	Workbook	
Appreciation	0.02349921	='Lambda Function'!$K...	Workbook	
FirstClose	20.33	='Box Function'!$B$12	Workbook	
NumDays	100	='Box Function'!$K$13	Workbook	
NumofDays	100	='Lambda Function'!$K...	Workbook	
PredDailyPrice	{...}	=LAMBDA(Volatility,N...	Workbook	
Vol	0.09486	='Box Function'!$K$17	Workbook	
Volatility	0.09486	='Lambda Function'!$K...	Workbook	

Refers to:

=LAMBDA(Volatility,NumofDays,NormDist,Appreciation,Previous,Previous*(Volatility*SQRT(1/Numo

Invoking the LAMBDA Function

In the Excel sheet, Cell F13 contains the formula:

=PredDailyPrice(Volatility,NumofDays,D13,Appreciation,F12)

Following the order of the arguments in the function: Volatility = Volatility, NumofDays = NumofDays, NormDist = D13, Appreciation = Appreciation, Previous = F12.

This custom function in cell F13 calculates as:

F13*(K17*SQRT(1/K13)*D13)+ EXP((K18-K17^2*0.5)*(1/K13)))

The remaining functions in cells F14:F36 calculate similarly.

Click LAMBDA and LET for Microsoft Excel documentation.

Note: Recall that according to Microsoft's Support for the Lambda function, it's best to create the function within an Excel cell before adding to the Name Manager, in order to determine if the custom lambda function is calculating correctly. This is exactly why you see the LAMBDA function in cells F13:F36.

Notes on the LAMBDA Function

Scope of variables within a LAMBDA function

When a LAMBDA expression is calculated, Excel/Analytic Solver will first search the local scope of the expression for the definition of a variable. If not found, Excel/Analytic Solver will proceed to search the parent scope and then will move to the global scope (cell/range names).

Assume an empty cell with the defined name "z" and the custom Lambda function below exist within the same workbook.

=LAMBDA (x, y, LET (z, x+1, y * z + b1))

In this instance, the local scope is the LET function, the parent scope is the LAMBDA function enclosing the LET function, and the global scope is all cell/range names in the workbook. The variable z =x+1 despite the existence of z in the global scope. The variable z inside LET and the z defined name in Excel are considered to be different variables.

Entering the LAMBDA/LET functions in Excel

It's possible to enter the LAMBDA function in an Excel in multiple lines by pressing ALT + ENTER. For example:

```
=LAMBDA(
   x,y, LET(
      y * z + b1
  )
)
```

Nesting LAMBDA/LET functions

Analytic Solver supports only nested LET functions. Nested LAMBDA functions are not supported by Analytic Solver.

Properties of LAMBDA/LET in PSI Interpreter

The following list contains important properties of both the LAMBDA and LET functions when used with Analytic Solver.

1. Reusability – LET definitions are not reusable while LAMBDA definitions, even ones containing LET functions, are resuable.
2. Nesting – LET definitions can be nested while LAMBDA definitions cannot.

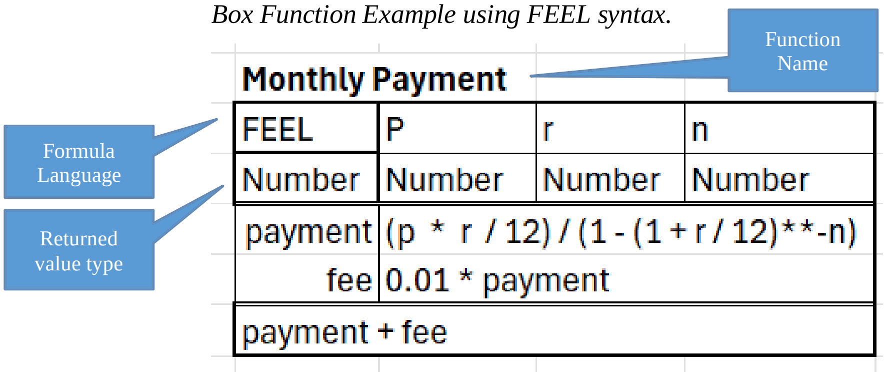
3. Recursion – Although the LAMBDA function is recursive in Excel, Analytic Solver does not support recursion with this function.
4. Threads – Analytic Solver supports both LET and LAMBDA functions running in multiple threads.
5. Polymorphic Evaluation – Analytic Solver supports the calculation of derivates, intervals, etc inside of a LAMBDA and LET function.
6. Analytical Transformations – Non-smooth trasnformations inside both LET and LAMBDA functions is restricted. Stochastic transformations allow both LET and LAMBDA functions.
7. Cube Evaluations: Analytic Solver does not support the use of dimensional modeling (cubes) within a LAMBDA or LET function.
8. Environment: Both LAMBDA and LET functions are supported in Analytic Solver, Solver SDK and RASON Decision Services.
9. Model Conversion: Both LET and LAMBDA functions are supported in model conversion from Excel to RASON (Deploy Model – RASON).
    - A LET function within an Excel model will appear within the "formulas" section within the RASON model, after conversion is complete.
    - The LAMBDA function within an Excel model will appear within a new section, "boxFunctions" within the RASON model, after the model conversion is complete.

    See the RASON User Guide for more information on both the LET and LAMBDA functions appearing within a RASON model.

Box Function Design

An example of a custom box function is below as implemented in an Excel workbook. Assume both appear in cells A3:D8 in the Excel workbook.

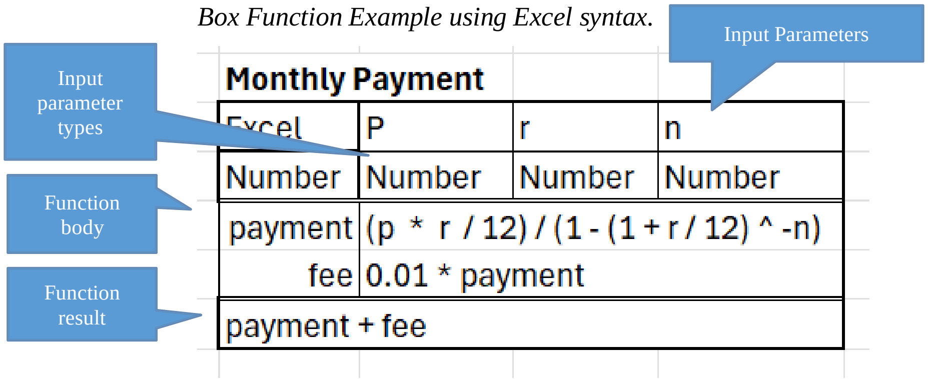
Box Function Example using FEEL syntax.



Monthly Payment

FEEL	P	r	n
Number	Number	Number	Number
payment	(p * r / 12) / (1 - (1 + r / 12)**-n)		
fee	0.01 * payment		
payment + fee			

Box Function Example using Excel syntax.



Monthly Payment

Excel	P	r	n
Number	Number	Number	Number
payment	(p * r / 12) / (1 - (1 + r / 12) ^ -n)		
fee	0.01 * payment		
payment + fee			

The first box function is written using the formula syntax for FEEL and the second table has been written using Excel syntax.

Note: Feel box functions have only the local scope of parameters and **no access** to the outer scope. DMN/Feel requires that if a variable from the global scope is referenced within a box function with FEEL, then the name is automatically treated as a string even when not surrounded by " ".

Box Function Component	**Description**
Function name	(Required) The name of the custom function appears above the top-left cell of the table layout.
Formula language	(Required) The formula language is specified in the **top-left** cell. Supported options are FEEL or EXCEL. If FEEL is entered as the formula language, FEEL syntax is expected. (See top screenshot.)
Returned value type	**Formula Language = Excel** **Array**: Any Excel cell reference, i.e. A1:C1. This can be used for a Box function, that, say, computes the SUMPRODUCT(A1:A3, B1:B3) where A1:A3 is a range for the first input parameter and B1:B3 is the range for the second input parameter. **Boolean:** The entered words TRUE and FALSE are interpreted as Boolean reserved words, not strings. **Empty**: Select "empty" if Formula Language = EXCEL and no Data Type is being specified. **Error**: Any Excel error such as #N/A, #Number, etc. **Number**: May be an integer or fraction. **String or Text**: Any string **Language = FEEL** **Boolean:** The entered words TRUE and FALSE are interpreted as Boolean reserved words, not strings. **Date:** Any valid date, such as 05-05-1964

	Duration: There are two formats for duration, one measuring periods in months and another measuring periods in seconds. For example, P1DT1H2M3S denotes: P for "period" 1D for 1 day T for "time" 1H for 1 hour 2M for 2 minutes and 3S for 3 seconds.
Input parameters	(Required) Input parameter values are specified on the first row starting in the 2nd column. Input Parameters may be changing (i.e. decision variables, recourse variable, uncertain variables) or unchanging (i.e constant values).
Input Parameter types	**Formula Language = Excel** **Array**: Any Excel cell reference, i.e. A1:C1. This can be used for a Box function, that, say, computes the SUMPRODUCT(A1:A3, B1:B3) where A1:A3 is a range for the first input parameter and B1:B3 is the range for the second input parameter. **Boolean:** The entered words TRUE and FALSE are interpreted as Boolean reserved words, not strings. **Empty**: Select "empty" if Formula Language = EXCEL and no Data Type is being specified. **Error**: Any Excel error such as #N/A, #Number, etc. **Number**: May be an integer or fraction. **String or Text**: Any string **Language = FEEL** **Boolean:** The entered words TRUE and FALSE are interpreted as Boolean reserved words, not strings. **Date:** Any valid date, such as 05-05-1964 **Duration:** There are two formats for duration, one measuring periods in months and another measuring periods in seconds. For example, P1DT1H2M3S denotes: P for "period" 1D for 1 day T for "time" 1H for 1 hour 2M for 2 minutes and 3S for 3 seconds.

Empty row	(Required) An empty row separates the function signature with the function body.
Function body	(Optional) The body of the function includes 2 columns. • The first column names the intermediary formula being calculated in each row. • The second column contains the intermediary formula. All formulas are executed in the order they are entered.
Function result	(Required) The last row in a single merged column holds the final formula.

The function in the examples above accepts three numbers, p, r and n, and returns a number. In order to calculate the function the first formula, *payment*, is calculated as: (p * r/12) / (1 – (1+ r/12)^-n). The 2nd formula, *fee*, is calculated as 0.01 * payment. Finally, the final result is calculated as: payment + fee.

Execution of a Custom Function

The new PSI function, PsiBoxFunction, executes the custom function. This function must be called with the cell address of the custom function and any input parameters.

=PsiBoxFunction(loc_range, input1, input2, input3, etc.)

To invoke the custom function in this example use:

=PsiBoxFunction(A3:D7, 60000, 0.0374, 360) where 60,000 = p, 0.0374 = r and 360 = n

It's also possible to create a defined name for the function location in cells A3:D7 and a defined name for each input parameter into a cell with a defined name, then pass the defined name as the argument, i.e.

=PsiBoxFunction(Function1, amount, rate, term)

Using a Box Iterator

A "box iterator" describes an iterator that encapsulates or "boxes" the iteration process in a neat, encapsulated manner. Just like how a physical box contains items, a box iterator contains the logic for iterating over a particular data structure or set of elements. This encapsulation makes the code more modular and easier to understand, especially when dealing with complex data structures or when the iteration logic needs to be reused in multiple places.

In earlier editions of Analytic Solver, loops were confined to usage within VBA or, more recently, with Python in Excel. When a function is authored in either VBA or Python in Excel, the Psi Interpreter remains inaccessible to the formulation. Instead, the Excel Interpreter must be employed to analyze and solve the model. However, this constraint can now be circumvented by introducing a box iterator.

As previously mentioned, when employing FEEL syntax, variables are restricted from being accessed beyond the confines of the box function environment, since FEEL syntax confines the scope exclusively to the box function. However, by opting for Excel syntax, the broader Excel scope becomes accessible, thereby facilitating the construction of loops.

A box iterator contains many of the same components of a box function, three examples of box iterators are shown below.

Example 1: Box Iterator Example using Excel Syntax

Name

EXCEL	number
for	i
in	1..rows(amount)
payment	(amount[i] * rate[i] / 12) / (1 - (1 + rate [i] / 12) ^ -term[i])
fee	0.01 * payment
return	payment + fee

*amount is a defined name in Excel encompassing 3 rows.

Example 2: Box Iterator Example using FEEL Syntax

Name

FEEL	number
for	r
in	[0.05, 0.06, 0.07]
payment	(600000 * r / 12) / (1 - (1 + r / 12) ** -360)
fee	0.01 * payment
return	payment + fee

Example 3: Box Iterator Example using FEEL Syntax

Name

FEEL	p	r	n
number	array	array	array
for I in 1..count(p) return ((p[i] * r[i] /12 / (1 - (1 + r[i] / 12)**-n[i])) * 1.01			

**p is a defined name in Excel encompassing three rows.*

Example 4: Box Iterator Example using 2 indexes

BoxIterTwoIndices

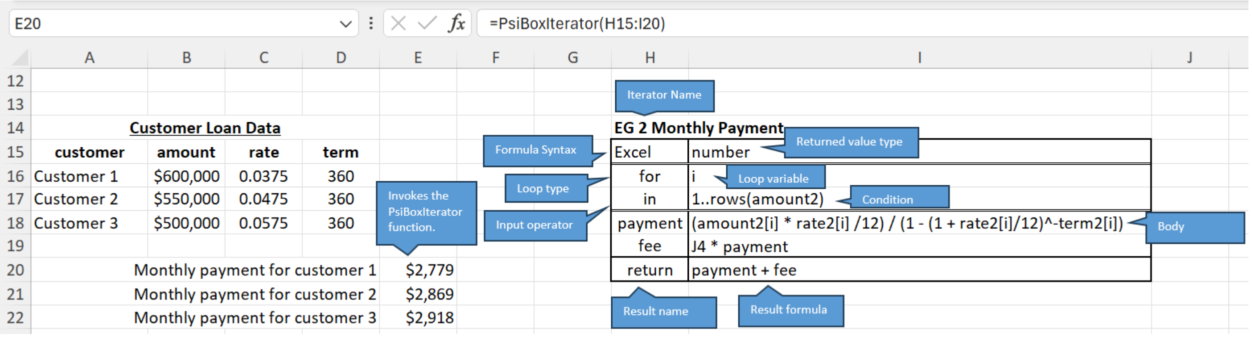
Excel	
for	*i, j*
in	*1..rows(mylist), mylist*
return	*i+j*

*mylist is a defined name in Excel encompassing three rows.

- The (required) name of the box iterator appears above the top-left cell of the table layout.
- The (required) formula syntax (FEEL or EXCEL) is passed in the top upper left hand corner. If syntax is FEEL, syntax must be compliant with DMN decision modeling specifications.

  Note: Box functions defined in FEEL syntax are limited to the internal box scope. Variables or defined names may not be referenced outside of the box iterator.
- The (optional) returned value type is specified to the right of the formula syntax.
- Below the formula syntax is the (required) loop type, in this case the loop type is “for”. Analytic Solver supports three loop types: for, every, and some.
  - A “for” loop behaves the same in a box iterator as in a programming language, i.e a condition is specified and the loop continues until the condition is no longer satisfied, changing the formula on each loop.
  - An “every” loop checks whether all elements in the loop are True. Computed result must be a single Boolean value.
  - A “some” loop checks whether at least one computed element of the loop is True. Computed elements must be Boolean.
- Loop variable (required) specifies the variable that will be changing.
- Below the loop type, the (optional) operator “in” (short for input) appears.
- To the right of “in”, the (required) index ranges are listed for each index either as a list/array or through the range operator “…”.
- The (optional) iterator body of formulas appears. This body is composed of both columns. The first names the formula and the 2nd contains the formula itself.
- The (required) last row specifies the result of the iterated formula. The first column names the result and the 2nd column calculates the result.

Use the PsiBoxIterator function to execute the box iterator. The PsiBoxIterator function must be the only Psi function contained in the cell. Multiple PsiBoxIterator functions may be contained within the same worksheet. Pass the address of the box iterator function starting with the cell containing the syntax and ending with the bottom right most cell containing the return formula. In the screenshot below, notice that the PsiBoxIterator function in cell E20 includes the cell address H15:I20).



=PsiBoxIterator(H15:I20)

Customer Loan Data

customer	amount	rate	term
Customer 1	$600,000	0.0375	360
Customer 2	$550,000	0.0475	360
Customer 3	$500,000	0.0575	360

Monthly payment for customer 1	$2,779
Monthly payment for customer 2	$2,869
Monthly payment for customer 3	$2,918

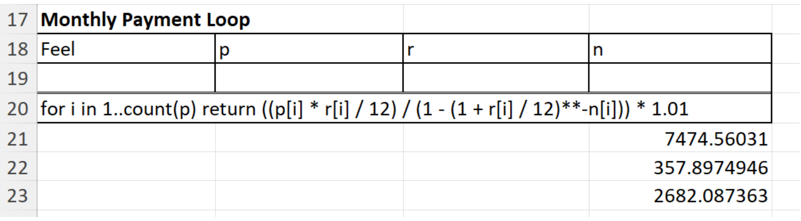
EG 2 Monthly Payment

Excel	number
for	i
in	1..rows(amount2)
payment	(amount2[i] * rate2[i] /12) / (1 - (1 + rate2[i]/12)^-term2[i])
fee	J4 * payment
return	payment + fee

See the Box Iterator Example at Help – Example Models – Decision Table and Decision Tree Examples for an illustration on the use of a box iterator.

Box Function Loops

It’s also possible to write a loop using a box function using FEEL language, as in the example below. This example also computes the monthly payment for three consumer loans given the loan principal, the loan interest rate and the loan term (or length).



17	**Monthly Payment Loop**			
18	Feel	p	r	n
19				
20	for i in 1..count(p) return ((p[i] * r[i] / 12) / (1 - (1 + r[i] / 12)**-n[i])) * 1.01			
21				7474.56031
22				357.8974946
23				2682.087363

The box function loop above has been created using FEEL language. The for loop appears in row 20:

for i in 1..count(p) return ((p[i] * r[i] /12) / (1- (1+ r[i] /12) **-n[i])) * 1.01

This loop accepts three inputs, p (loan principal), r (loan interest rate), and n (length of loan term).

The PSI function PsiBoxFunction is used to invoke this box function, as discussed in the Box Function section above, =PsiBoxFunction(A17:D20, F4:F6, G4:G6, H4:H6)

Since there are three elements in the first input array, F4:F6, the function in row 20 will loop 3 times, each time with a new input for p, r and n. This function will spill down to 3 cells, each cell containing the calculated monthly payment.

Box Function Optimization Example

Open the Airport Box Function example (Help – Example Models – Optimization Examples) to view an example of how a custom box function could be used in an optimization model.

Original Formulation

The original model formulation is in the first worksheet, Original Formulation. This example uses the distance formula in cells G14:G19 to find the optimal location (using X and Y coordinates) of a proposed airport. The airport's location should be as equidistant as possible between each of the six cities.

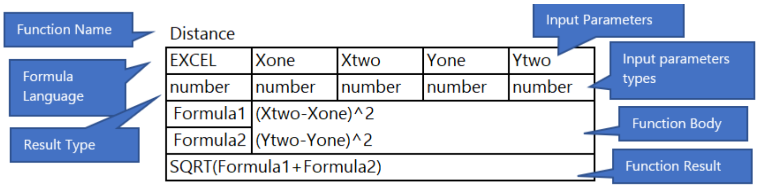
- The decision variables in this model are in cells E13, F13 and G20.

- The constraints in this model are G14:G19 <= G20. Cells G14:G19 use the distance formula (SQRT(Xc – X)^2 + (Yc – Y) ^2) to calculate the distance between the proposed airport location and each city.
- The objective to be Minimized is the decision variable, G20.

The objective in cell G20 is being minimized which "squishes" the values in cells G14:G19 to be as small as possible. This is what is referred to as a Mini-Max problem; G20 will equal the maximum of G14:G19 in the final solution.

Box Function Formulation

On the Box Function tab, the model has been reformulated to use a custom box function to calculate the distance between the proposed airport location and each of the six cities. The box function is located in cells E24:I28.



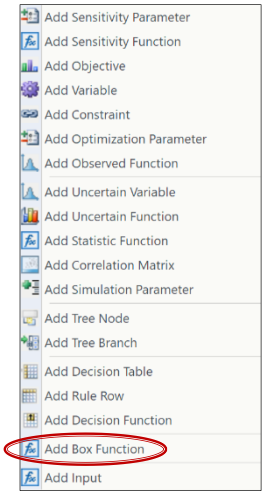
Distance

EXCEL	Xone	Xtwo	Yone	Ytwo
number	number	number	number	number
Formula1	(Xtwo-Xone)^2			
Formula2	(Ytwo-Yone)^2			
SQRT(Formula1+Formula2)				

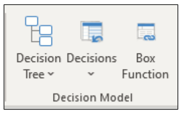
Creating a Custom Box Function

To quickly create a Box function on the worksheet, select an empty cell on the worksheet, click the down arrow next to the + icon on the Model tab of the Solver Task Pane and select Box Function at the bottom or click Decisions – Box Function on the ribbon.

Analytic Solver Task Pane



Analytic Solver Ribbon



The Define Your Table dialog appears.

For this example, enter the following:

- Function Name: Distance

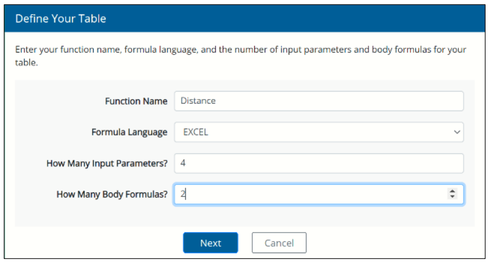
  A function name is required.
- Formula Language: Click the down arrow and select EXCEL. Recall that this setting determines the syntax used in the function. You must select either EXCEL or FEEL.

- How Many Input Parameters?: Enter 4

  At least one Input Parameter is required.
- How Many Body Formulas?: Enter 2 Recall that these are the intermediary formulas.

  Body formulas are optional; if none, enter 0.

The completed dialog should look similar as the one in the screenshot below.



Define Your Table

Enter your function name, formula language, and the number of input parameters and body formulas for your table.

Function Name: Distance

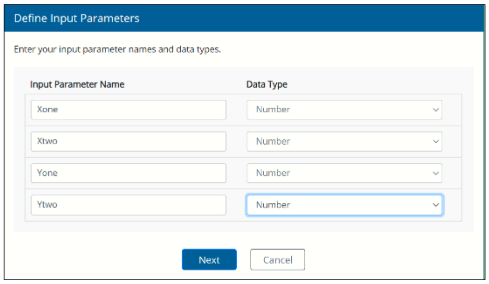
Formula Language: EXCEL

How Many Input Parameters? 4

How Many Body Formulas? 2

Click Next.

In the Define Input Parameters dialog, enter the four Input Parameters (Xone, Xtwo, Yone, Ytwo) and their types (number) as shown below. Recall that the Input Parameters are the arguments to the custom function. The Data Type argument is optional. Select "empty" if not specifying a data type.

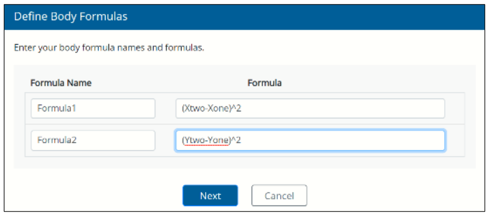


Define Input Parameters

Enter your input parameter names and data types.

Input Parameter Name	Data Type
Xone	Number
Xtwo	Number
Yone	Number
Ytwo	Number

Click Next. Enter the two intermediary formulas (if using) on the Define Body Formulas dialog. Formulas are executed in the order entered.



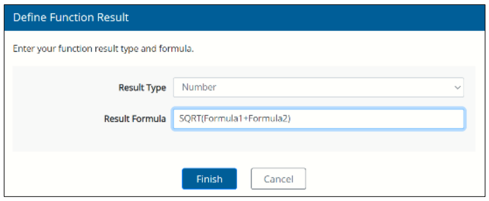
Define Body Formulas

Enter your body formula names and formulas.

Formula Name	Formula
Formula1	(Xtwo-Xone)^2
Formula2	(Ytwo-Yone)^2

Click Next. On the last dialog, the Define Function Result dialog,

- Select **Number** for *Result Type*. This argument is optional. Select "empty" if not specifying a data type.
- The formula, **SQRT(Formula1+Formula2)** for *Result Formula*. This field is required.



Enter your function result type and formula.

Result Type: Number

Result Formula: SQRT(Formula1+Formula2)

Click Finish.

Notice the form of the function.

The **Function name** (required) appears at the top left of the function, in this example, the function name appears in cell E23, Distance.

The **Formula Language** (required) appears in the cell immediately below the Function name. This setting determines what syntax is used in the function.

The **Input Parameters** (required) appear in the columns to the right of the Formula Language. This example uses four input parameters, Xone, Xtwo, Yone and Ytwo. Arguments must be passed to PsiBoxFunction() in this order.

The 2nd row includes the **Result** and **Input Parameter** types, (optional). Cell E25 contains the result type, which will be a number. The same is true for the Input Parameter types, which are all numbers. Although, the types are optional, this row must exist, even if empty.

The third row is the start of the **Function body**, (optional). The body is where intermediary formulas may be calculated. The *first column* of the body contains the formula name and the *2nd, merged column* contains the actual formula. In this example, the 1st formula calculated is Formula1, followed by Formula2.

The last merged row contains the **Function result**., (required). In this example, the function calculates as: SQRT((Xtwo-Xone)^2 + (Ytwo-Yone)^2). This is the same function used in cells G14:G19 in the original formulation.

Invoking the Box Function

The custom box function is invoked in cells G14:G19 using the PSI function, PsiBoxFunction.

G14=PsiBoxFunction($E$24:$I$28, $E$13,E14,$F$13,F14)

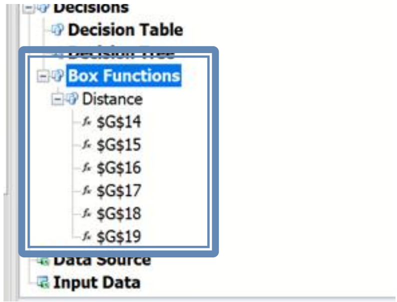
Notes:

- The "anchors" around the box function location and the E13 and F13 arguments are needed when copying this formula down to cell G19.
- The order of the arguments passed to PsiBoxFunction is the same as the order in the custom function, Xone, Xtwo, Yone, Ytwo.

This formula calculates as: SQRT((E14-E13)^2 + (F14-F13)^2)

This formulas in cells G15:G19 are similar.

The PsiBoxFunctions are displayed in the task pane under Decisions – Box Functions.



Decisions
Decision Table
Box Functions
Distance
$G$14
$G$15
$G$16
$G$17
$G$18
$G$19
Data Source
Input Data

To solve the model, click the green arrow in the Model Task Pane. Analytic Solver calls the box function to calculate the distance formula and solves the optimization model. Notice that the final variable values (X=3.5, Y = 4.5) are exactly the same as in the original formulation.

Continue to the next section to see this same model formulated using the Excel LAMBDA function.

Simulation Example

Open the BoxFunctionGBMSimpleModel example (Help – Example Models – Monte Carlo Simulation Examples) to view an example of how a custom box function could be used in a simulation model.

Original Formulation

The original model formulation is in the first worksheet, Original Formulation. This example predicts the average daily closing price of a stock based on it's previous closing price.

- The uncertain variables in this model are in cells D13:D36.
- The uncertain functions in this model are in cells F13:F36. Recall that these cells are designated as uncertain functions by the statistic PsiMean that appears in cells H13:H36.
- Cells H13:H36 contain the PsiMean() function which finds the expected value of cells F13.

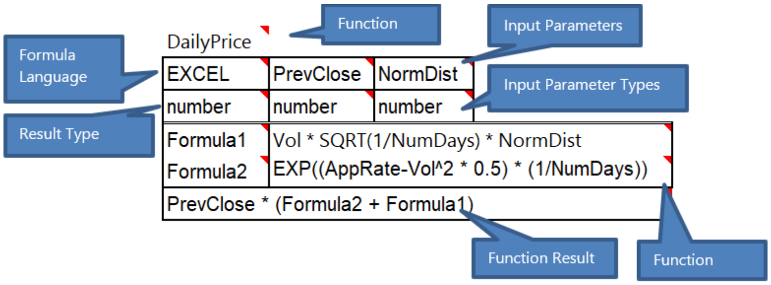
In the original formulation of the model, cells F13 contains the formula:

```
=(F12*(EXP(($K$18-$K$17*$K$17*0.5)*(1/$K$13))+($K$17*SQRT(1/$K$13)*D13)))
```

Due to the length and intricacy of this formula, one can imagine how easy it would be to accidentally introduce an error when entering this formula. Not to mention the effort required when searching for an error in the spreadsheet. To combat these challenges, Analytic Solver now fully supports the use of custom Box functions which can be defined once and reused repeatedly. Custom Box functions reduce "bugs" in the spreadsheet by only requiring an intricate function to be defined once.

Box Function Formulation

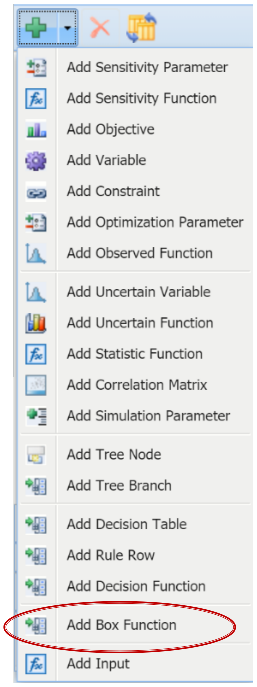
On the Box Function tab, the model has been reformulated to use a custom box function to calculate the predicted daily stock price. The box function is located in cells K23:O27.



Creating the Custom Box Function

To quickly create the Box function on the worksheet, select an empty cell on the worksheet, click the down arrow next to the + icon on the Model tab of the Solver Task Pane and select Box Function at the bottom.

Analytic Solver Cloud



The Define Your Table dialog appears.

For this example, enter the following:

- Function Name: DailyPrice

  A function name is required.

- Formula Language: Click the down arrow and select EXCEL. Recall that this setting determines the syntax used in the function. You must select either EXCEL or FEEL.

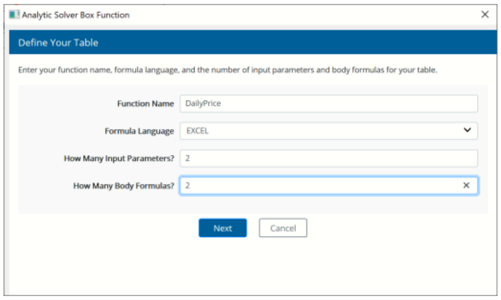
- How Many Input Parameters?: Enter 2

  At least one Input Parameter is required.

- How Many Body Formulas?: Enter 2   Recall that these are the intermediary formulas.

Body formulas are optional; if none, enter 0.

The completed dialog should look similar as the one in the screenshot below.

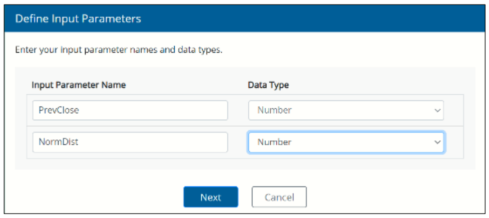


Enter your function name, formula language, and the number of input parameters and body formulas for your table.

Function Name: DailyPrice
Formula Language: EXCEL
How Many Input Parameters?: 2
How Many Body Formulas?: 2

Click Next.

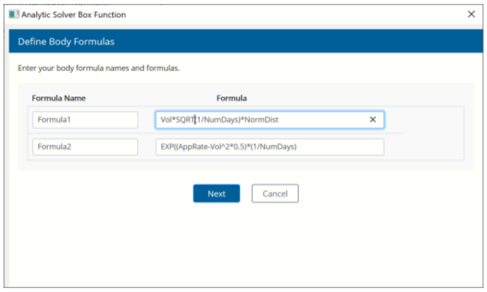
In the Define Input Parameters dialog, enter the two Input Parameters (PrevClose and NormDist) and their types (number) as shown below. Recall that the Input Parameters are the arguments to the custom function. The Data Type argument is optional. Select "empty" if not specifying a data type. See the table above for definitions of all Data Types.



Enter your input parameter names and data types.

Input Parameter Name	Data Type
PrevClose	Number
NormDist	Number

Click Next. Enter the three intermediary formulas (if using) on the Define Body Formulas dialog. Formulas are executed in the order entered.

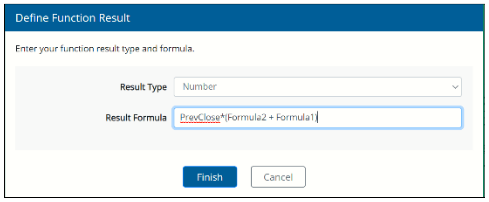


Enter your body formula names and formulas.

Formula Name	Formula
Formula1	Vol*SQRT(1/NumDays)*NormDist
Formula2	EXP((AppRate-Vol^2*0.5)*(1/NumDays))

Click Next. On the last dialog, the Define Function Result dialog,

- Select **Number** for *Result Type*. This argument is optional. Select "empty" if not specifying a data type. See the table in the introduction to this chapter for the definitions of all Result Types.
- The formula, **PrevClose*(Formula2+Formula1)** for *Result Formula*. This field is required.



Enter your function result type and formula.

Result Type: Number
Result Formula: PrevClose*(Formula2 + Formula1)

Click Finish.

The form of this function is:

- **Function name -** (Cell K22) Appears at the top left of the function, in this example, DailyPrice. (Required)
- **Formula Language** – (Cell K23) Appears in the cell immediately below the Function name. This entry can either be EXCEL or FEEL. This setting determines what syntax is used in the function. (Required)
- **Input Parameters** – (Cells (L23:M23) Appear in the columns to the right of the Formula Language. This example uses two input parameters, PrevClose and NormDist . Arguments must be passed to PsiBoxFunction() in this order. (Required)
- **Result** and **Input Parameter Types** – (Cells K24:M24) contains the result type for both the Result and Input Parameters, in this example a number. Although, the types are optional since Formula Language = Excel, this row must exist, even if empty. (Required if Formula Language = FEEL, otherwise optional)
- **Function Body** – (Cells K25:O26) The third row is the start of the Function body. The body is where intermediary formulas may be calculated. The *first column* of the body contains the formula name and the *2nd, merged column* contains the actual formula. In this example, the 1st formula calculated is Formula1, followed by Formula 2. (Optional)
- **Function Result** – (Cells K27:M27) The last merged row contains the Function result. In this example, the function calculates as:

  =PrevClose * (Formula2 + Formula1)

  =PrevClose * (Vol*SQRT(1/NumDays)*NormDist) + EXP((AppRate-Vol^2 * 0.5) * (1/NumDays))

  where

    - PrevClose = the previous days predicted daily rate (Box Function Input Parameter 1)
    - Vol = K17 = 0.09486
    - NumDays = K13 = 100
    - NormDist = PsiNormal() uncertain variable in column D (Box Function Input Parameter 2)
    - AppRate = K18 = 0.0234992

Invoking the Box Function

The custom Invoking box function is invoked in cells F13:F36 using the PSI function, PsiBoxFunction.

F13 = PsiBoxFunction($K$23:$M$27, FirstClose, D13)

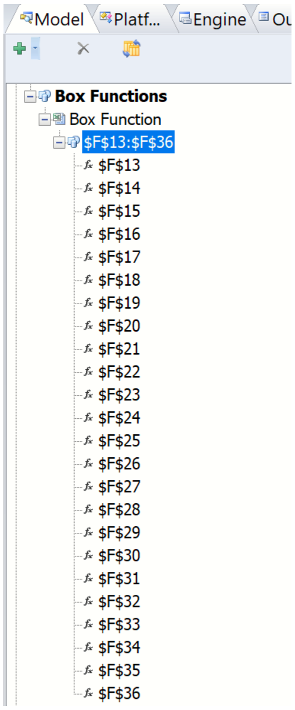
Notes:

- The "anchors" around the box function location are needed when copying this formula down to cell F36.
- The order of the arguments passed to PsiBoxFunction is the same as the order in the custom function, PrevClose, NormDist.

This formula in cell F13 calculates as: =F12 * (K17*SQRT(1/K13)*D13) + EXP((K18-K17^2 * 0.5) * (1/K13).

This formulas in cells F14:F36 calculate similarly.

The PsiBoxFunctions are displayed in the task pane under Decisions – Box Functions.

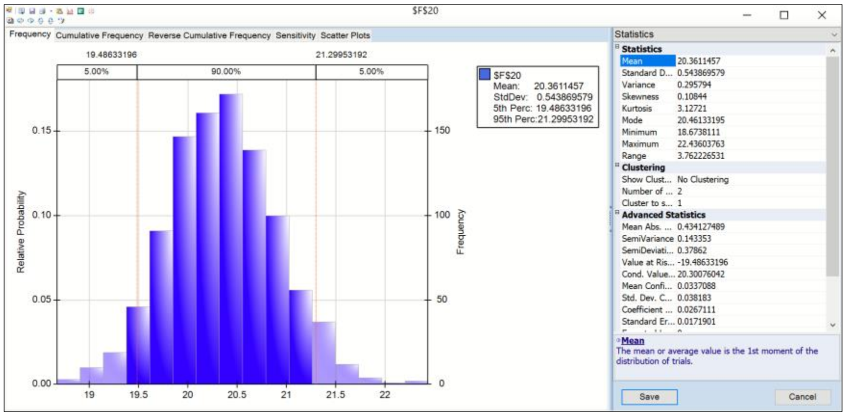


- **Box Functions**
  - Box Function
    - $F$13:$F$36
      - $F$13
      - $F$14
      - $F$15
      - $F$16
      - $F$17
      - $F$18
      - $F$19
      - $F$20
      - $F$21
      - $F$22
      - $F$23
      - $F$24
      - $F$25
      - $F$26
      - $F$27
      - $F$28
      - $F$29
      - $F$30
      - $F$31
      - $F$32
      - $F$33
      - $F$34
      - $F$35
      - $F$36

Solving the Model

To solve the model, click the green arrow in the Model Task Pane. Analytic Solver calls the box function to calculate the predicted daily price over 1000 trials.

To view a histogram of each predicted daily price, double click any cell in the F column. The uncertain function dialog below displays the distribution of trial values found for the custom function in cell F20. From this histogram, a user can determine that there is a 90% chance that the stock price will range from $19.49 to $21.30. For more information on how to interpret a histogram in Analytic Solver, see the Simulation chapter.



Statistics	
Mean	20.3611457
Standard D...	0.543869579
Variance	0.295794
Skewness	0.10844
Kurtosis	3.12721
Mode	20.46133195
Minimum	18.6738111
Maximum	22.43603763
Range	3.762226531
Clustering	
Show Clust...	No Clustering
Number of ...	2
Cluster to s...	1
Advanced Statistics	
Mean Abs. ...	0.434127489
SemiVariance	0.143353
SemiDeviati...	0.37862
Value at Ris...	-19.48633196
Cond. Value...	20.30076042
Mean Confi...	0.0337088
Std. Dev. C...	0.038183
Coefficient ...	0.0267111
Standard Er...	0.0171901

Mean
The mean or average value is the 1st moment of the distribution of trials.

Implemented DMN functions for Conformance Level 3

See the listed example functions below implemented for DMN CL3 for both type and non-typed lists. These functions and restricted to 1D and 2D array structures.

count(list) returns integer	count([1,2,5]) = 3
max(list) returns list elem type	max([1,2,5]) = 5
max(num1, num2,…) returns number	max(1,2,5) = 5
min(num1, num2,…) returns number	min(1,2,5) = 1
sum(list) returns number	sum([1,2,5]) = 8
sum(num1, num2,…) returns number	sum(1,2,5) = 8
roundup(n, scale) returns number	Returns n with given scale, rounded up. roundup(5.5, 0) = 6 roundup(-5.5, 0) = -6 roundup(1.121, 2) = 1.13 roundup(-1.126, 2) = -1.13.
rounddown(n, scale) returns number	Returns n with given scale, rounded down. rounddown(5.5, 0) = 5 rounddown(-5.5, 0) = -5 rounddown(1.121, 2) = 1.12 rounddown(-1.126, 2) = -1.12
roundHalfUp(n, scale) returns number	Returns n with given scale; rounded up. roundHalfUp(5.5, 0) = 6 round HalfUp(-5.5, 0) = -6 roundHalfUp(1.121, 2) = 1.12 roundHalfUp(-1.126, 2) = -1.
roundHalfDown(n, scale) returns number	Returns n with given scale; rounded down. roundHalfDown(5.5, 0) = 5 round HalfUp(-5.5, 0) = -5 roundHalfUp(1.121, 2) = 1.12 roundHalfUp(-1.126, 2) = -1.13

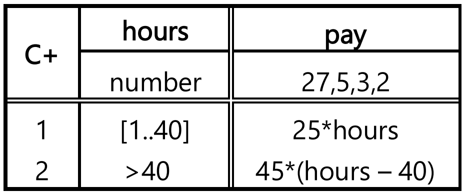
Signatures for other statistics functions, such as mean, median, mode, product, stdev, are the same as sum, i.e. mean(num1, num2, …).

and(list), **all**(list)	returns Boolean
and(bool1, bool2,…), **all**(bool1, bool2,…)	returns Boolean
or(list), **any**(list)	returns Boolean
or(bool1, bool2,…), **any**(bool1, bool2,…)	returns Boolean
sublist(list, start pos, [length]) returns list	sublist([1,2,5], 2) = [2,5]
append(list, elem1, elem2,…) returns list	append([1,2,5], 0) = [1,2,5,0]
concatenate(list1, list2,…) returns list	concatenate([1,2,5], [3,4]) = [1,2,5,3,4]
insertBefore(list, pos, elem) returns list	insertBefore([1,2,5], 3, 0) = [1,2,0,5]
listContains(list, elem) returns Boolean	listContains([1,2,5], 0) = false
remove(list, pos) returns list	removes([1,2,5], 3) = [1,2]
reverse(list) returns list	reverse([1,2,5]) = [5,2,1]
indexOf(list, match) returns list	indexOf([1,2,1,2,3], 2) = [2,4]
union(list1, list2,…) returns list	union([1,2,3], [1,2,5]) = [1,2,3,5]
distinctValues(list) returns list	distinctValues([1,2,1,2,3]) = [1,2,3]
flatten(list) returns list	flatten([1,2], [2,3]) = [1,2,2,3]
listReplace(list, position/match, newItem) returns list, number or Boolean function (item, newItem), any element including null.	listReplace ([2,4,7,8],3,6) = [2,4,6,8]

Using Decision Tables

Introduction

Decision Tables, introduced in Analytic Solver V2019, consist of a set of rules specifying an action based on specific conditions. Decision tables are particularly useful when there is a consistent number of rules, or conditions, to be evaluate followed by a specific actions triggered once a rule, or condition, is met. For example, the simple decision table below determines an employee's pay based on the number of hours worked.



C+	hours	pay
	number	27,5,3,2
1	[1..40]	25*hours
2	>40	45*(hours – 40)

If the employee worked from 1 to 40 hours in a week, the pay is $25 * hour. Any hours over 40, defined as overtime hours, are paid at an increased rate of $45 an hour.

In Analytic Solver Comprehensive, a decision table represents a specific implementation of a custom function. Each table includes a name, input/output parameters and a body that executes one or more logic rules. A decision table is invoked in the same way as a VBA function when computing one or more output parameters based on the input values. In Analytic Solver, a decision table is invoked using the Psi function, PsiDecTable(). The modeling language inside of Analytic Solver Comprehensive's Decision Table tool is S-FEEL, which extends standard conversion functions in FEEL. For more information on Decision Tables, we recommend consulting “DMN Method and Style” by Bruce Silver (Cody Cassidy Press, September 28, 2018) and “DMN Cookbook” by Bruce Silver & Edson Tirelli (Cody-Cassidy Press, April 4, 2018).

Notes

- Decision Trees are only available for users with an Analytic Solver Comprehensive license. The Decision Table button, on the Analytic Solver ribbon, will be disabled for all other product licenses.

Decision Table Structure

Open the example model, DT Structure.xlsx by clicking **Help – Examples – Decision Trees/Decision Tables** on the Ribbon. This table returns the number of holidays apportioned to an employee based on his/her age and number of years in service. This file defines the regions of the decision table as illustrated in the screenshot below.

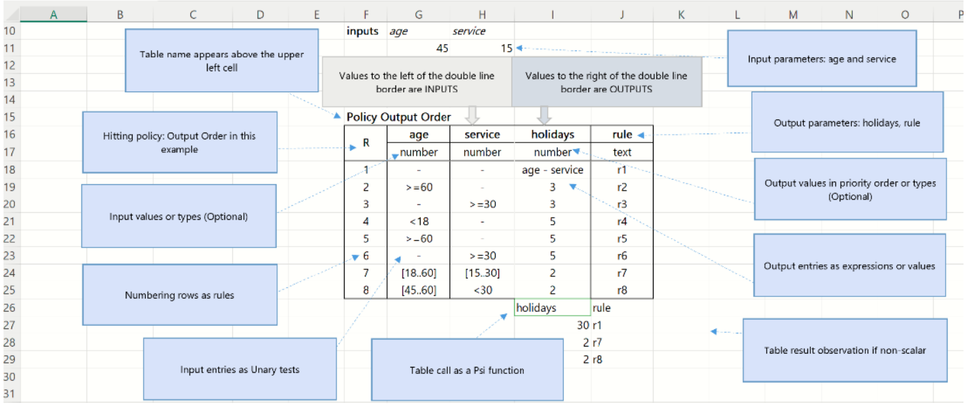


Table name appears above the upper left cell
Values to the left of the double line border are INPUTS
Values to the right of the double line border are OUTPUTS
Input parameters: age and service
Output parameters: holidays, rule
Hitting policy: Output Order in this example
Input values or types (Optional)
Output values in priority order or types (Optional)
Numbering rows as rules
Output entries as expressions or values
Input entries as Unary tests
Table call as a Psi function
Table result observation if non-scalar

inputs	age	service
	45	15

Policy Output Order

R	age	service	holidays	rule
	number	number	number	text
1	-	-	age - service	r1
2	>=60	-	3	r2
3	-	>=30	3	r3
4	<18	-	5	r4
5	>=60	-	5	r5
6	-	>=30	5	r6
7	[18..60]	[15..30]	2	r7
8	[45..60]	<30	2	r8

holidays	rule
30	r1
2	r7
2	r8

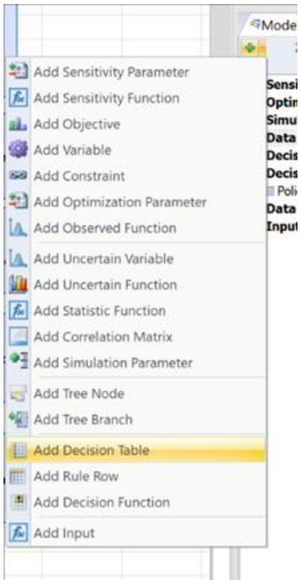
A Decision Table is created using the newly introduced PsiDecTable() function where the cell range of the table is passed as the first argument. See the PsiDecTable() function as it appears in cell I26, =PsiDecTable(F16:J25, {"holidays","rule"}, TRUE, G11, H11) where:

- F16:J25 - The cell address of the table is F16:J25. Note that the table name in cell F15 is not included in the table's cell range.
- "holidays", "rule" – The output will display the number of holidays each employee is entitled to, along with the rules that determine this number.
- True – Pass True to return a column heading for the output results.
- G11 and H11 – Input parameters: age (G11) and service (H11).

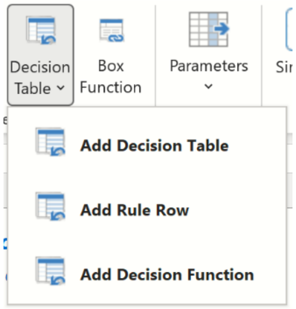
To help guide in the creation of a decision table, Analytic Solver Comprehensive has created a Decision Table wizard. Add a new worksheet to DT Stucture.xlsx and follow along with the steps below.

To open, click a blank cell on the worksheet and click the plus on the Solver Task pane (to the right of your screen), and select Add Decision Table from the menu. The Decision Table wizard may also be opened by clicking Decisions – Add Decision Table.

Analytic Solver Comprehensive Taskpane

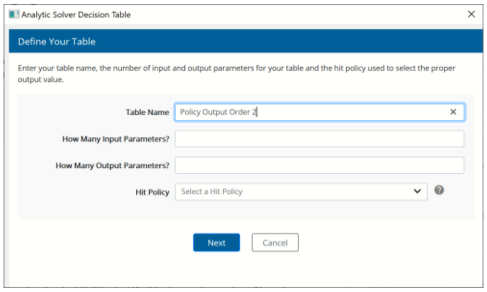


Analytic Solver Comprehensive Ribbon



Step 1: Enter a *unique* Name to define the table. This name will appear immediately above the upper left corner of the table cell range. (This argument creates the decision table object and identifies the table.) *In this example, the table name is specified as "Policy Output Order".* If deploying this model to RASON, the name component will be used to identify this specific table.

If using the wizard, be sure to enter a unique name such as "Policy Output Order 2".


Analytic Solver Decision Table
Define Your Table
How Many Input Parameters?
How Many Output Parameters?
Hit Policy
Select a Hit Policy
Next
Cancel

Enter your table name, the number of input and output parameters for your table and the hit policy used to select the proper output value.

Table Name: Policy Output Order 2

Step 2: Enter the number of Input Parameters. Decision table input parameters are the values or conditions that are evaluated by the decision table to determine which specific rules are triggered and, consequently, what actions are to be performed. Input parameters in decision tables are essential for determining the logic flow and output of the decision table's rules.

Since age and service are the input parameters in this example, increment the number of input parameters to “2”. Note: Input parameters belong to the local scope of a table.

Step 3: Enter the number of Output Parameters. Decision table output parameters are the parameters for which the decision table provides results after evaluating its rules. These outputs are derived based on the input conditions specified in the decision table.

In this example, “holidays” and “rule” are the output parameters. The “holidays” output will return the number of vacation days the employee is entitled to. The “rule” output will return the specific rule calculating the correct number of days. Asking for the rule to be returned in the output is an important debugging tool.

After calculation, a table returns a few selected or all output parameter values in the form of a dynamic array. If a table has a single output or a single output has been selected, the result will be a scalar value. Note: Output parameters belong to the local scope of this table

In the wizard, enter "2" for "How many Output Parameters?" and click Next.

Step 4: Select a Hit Policy from the drop down menu. Thie Hit Policy will be displayed in the left, upper-most cell in the table range. This policy specifies how the table will be evaluated when multiple rules are applicable and hence, multiple output values are returned. The entered value identifies the supported policies by a capital letter and an operator when applicable. If creating a decision table without the use of the Decision Table Wizard, you may enter the 1st letter of the policy or the whole word. The currently supported Hit types and their meanings are:

> **Unique (U):** A unique rule must "hit" evaluating to a unique result. If multiple rules are "hit", an error will be returned.
>
> **Any (A):** If rules overlap, but point to the same result, that unique result is returned.

Priority(P): If multiple rules are "hit" and multiple results collected, return only one result with the highest priority; the priorities are defined by the order of the output values.

First (F): Only 1 result is returned for this policy. Once a rule is evaluated successfully, or a hit occurs, the search stops.

Rule Order (**R**) **-** If multiple rules are hit, return the collection of results as created in the rule order.

Output Order (**O**) – If multiple rules are hit, return the collection of results in the priority order of the listed output values.

Collect (**C**) –

Note: If aggregating a date, a scalar is returned. If using an operator, output must not be a string, but only a numerical value.

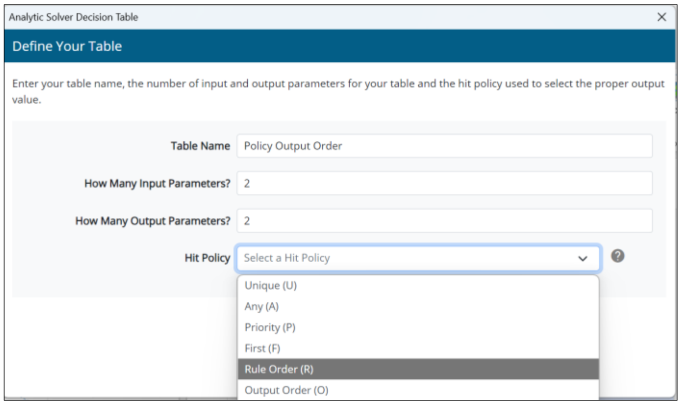
C+ **-** sum the matched output values

C< **-** return the min of matched output values

C> **-** return the max of matched output values

C# **-** return the number of matched output values

In the Wizard, click the down arrow next to Hit Policy and select Rule Order.



Enter your table name, the number of input and output parameters for your table and the hit policy used to select the proper output value.

Table Name: Policy Output Order

How Many Input Parameters? 2

How Many Output Parameters? 2

Hit Policy: Select a Hit Policy

Step 5: Click *Next.*

Step 6: Enter the Input Parameter Names in each field. In this example, the input parameters are “age” and “service”.

Step 7: Select the Input Parameter Data Types. Input Values may be a list of values separated by a commas (i.e. 27, 5, 3, 2), a list of unary tests (i.e. <10, >=20, [18..20]), or a data type (i.e. boolean, number, text, data, time, and duration). Input Parameter Types will be placed beneath each input parameter. In this example, both input parameters (age and service) are designated as “numbers”. See the allowable types defined below.

Data Types

Any: Use this option to allow any type of value or text.

Boolean: The entered words TRUE and FALSE are interpreted as Boolean reserved words, not strings.

Number: Maybe an integer or fraction. *This is the data type entered in the example above.*

Text: Any string

Date: Any valid date, such as 05-05-1964

Time: Any valid time

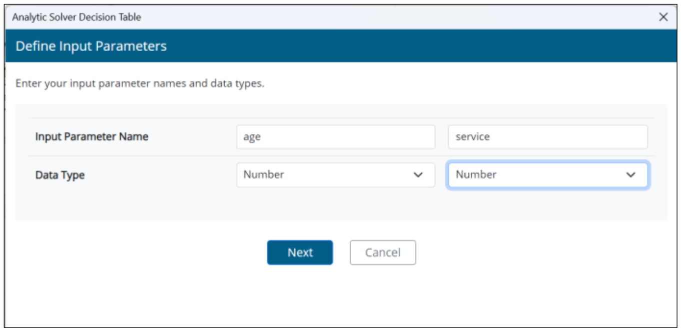
Duration: There are two formats for duration, one measuring periods in months and another measuring periods in seconds. For example, P1DT1H2M3S denotes:

- P for "period"
- 1D for 1 day
- T for "time"
- 1H for 1 hour
- 2M for 2 minutes and
- 3S for 3 seconds.

When testing a value against a list of values or unary tests, the OR operator is used. For example, in the first example above, the list of values is evaluated as 27 OR 5 OR 3 OR 2. Likewise, the list of unary tests is evaluated as <10 OR >=20. It's possible to negate a list as well. For example, NOT(27, 5, 3, 2) would result in a selection of a record that does not includes 27 OR 5 OR 3 OR 2. Similarly, NOT(<10, >=20) would equate to neither <10 OR >=20 being selected.

All input entries should cover the entered domain, otherwise, an error will be generated indicating that the table is not complete. For instance, if the word “the” is input for the age input parameter, an error will occur because "the" is not a numeric value. Note: If a data type is omitted, the completeness test is not performed.

In the wizard, enter "age" and "service" for Input Parameter Names and select "Number" from the Data Type menu for both parameters. Then click Next.



Enter your input parameter names and data types.

Input Parameter Name	age	service
Data Type	Number	Number

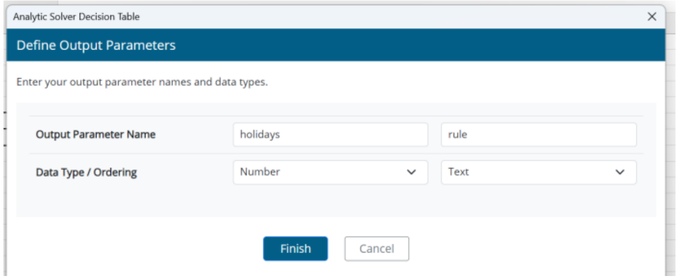
Step 8: Click Next.

Step 9: Enter the Output Parameter Names in each field. In this example, the output parameters are “holidays” and “rule”. Recall that the output parameter, “holidays”, will return the number of holidays, or vacation days, that the employee is entitled to and the output parameter, “rule” will return the specific rule(s) that were triggered, or activated.

Step 10: Select the Data Type / Ordering. Like Input Values, an Output value may be a list of values separated by commas (i.e. 27, 5, 3, 2), a list of unary tests (i.e. <10, >=20, [18..20]), or a data type (i.e. boolean, number, text, data, time, and duration). Similarly, to Input values, if a value appears in the table that does not match the output value, an error will be returned. See Step 7 above for supported types.

In this example, "Number" is selected for the *holidays* output and "Text" is selected for *rule* output. "Number" specifies that the output will be a number and “Text” specifies that the output will be text.

- *In the wizard, enter "holidays" for the first Output Parameter Name and then select Number for Data Type/Ordering.*
- *Enter "rule" for the second Output Parameter Name and then select "Text" for Data Type/Ordering.*

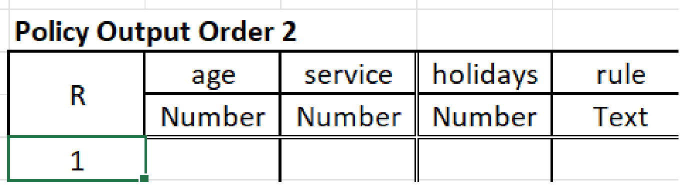


Define Output Parameters

Enter your output parameter names and data types.

Output Parameter Name	holidays	rule
Data Type / Ordering	Number	Text

Step 11: Click Finish. The top portion of the Policy Output 2 table will be inserted onto the new worksheet.



Policy Output Order 2

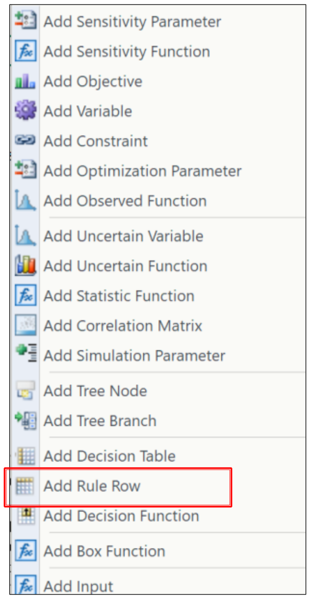
R	age	service	holidays	rule
	Number	Number	Number	Text
1				

Step 12: Enter the Decision Table Rules (or Unary tests) in each row below the double horizontal line by clicking the + on the task pane and then selecting Add Rule Row to add new rules to the table. This can also be done by clicking Decisions – Add Rule Row on the ribbon. A Decision Table must contain at least 1 rule. This example contains 8 rules.

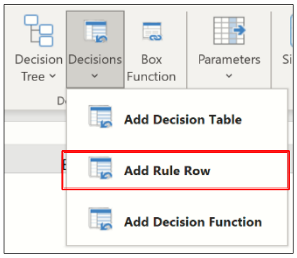
The unary test returns information (true or false) about the rule. Supported unary tests may have one of the following syntax forms: value(Boolean, number, text, date, time, duration), < value, > value, <= value, >= value, [value..value], (value..value], [value..value), (value..value), or "-". Note: Currently, rules *may only* be entered as rows.

The first test examines whether the value being tested is equal to the value inside the parenthesis. The forms, [value..value], (value..value], [value..value) and (value..value), are interval tests. The [ ] operators denote a closed interval while the ( ) operators denote an open interval. The last test, "-" returns TRUE against any value. Note: To return a default value for a table, simply use "-" for the unary tests.

Analytic Solver Comprehensive Taskpane



Analytic Solver Comprehensive Ribbon



Expressions

Variables and constants can be combined through operations called literal expressions. Literal expressions in S-FEEL are like formulas in Excel.

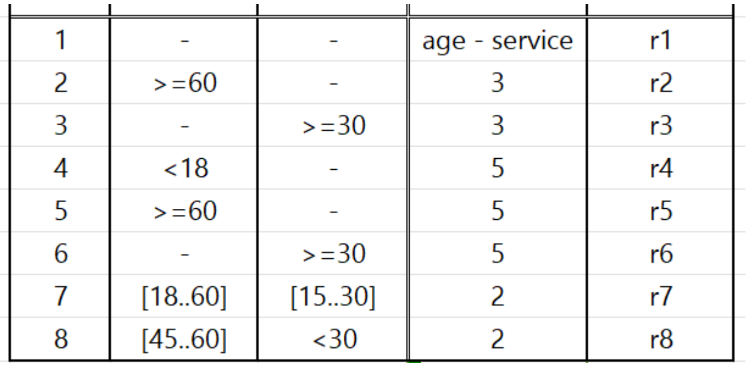
The following operators are supported in combination with decision table rules: addition (+), subtraction (-), multiplication (*), division (/) and exponentiation (**). Variables and constants can be combined using only these supported operators and parentheses. An example of an expression is: 2 * age – service where two variables, age and service, and a constant, 2, are linked by two arithmetic operations (* and **-**). Note that an expression is a FEEL expression, NOT an Excel formula. In this example, the expression "age – service" appears in the first rule where "age" and "service" refer to cells H11 and I11. This expression does not refer to any appearances of "age" or "service" outside of the scope of this table. For more information on supported conversion expressions, please see the Expressions heading at the end of this chapter.

Adding Rules to a Decision Table

Step A: The **rule number** is inserted into the first column under the Hit policy starting with "1". Although optional, this component of the table is strongly recommended.

Step B: Enter the Input entries in the input columns

Step C: Enter the Output entries in the output columns.



1	-	-	age - service	r1
2	>=60	-	3	r2
3	-	>=30	3	r3
4	<18	-	5	r4
5	>=60	-	5	r5
6	-	>=30	5	r6
7	[18..60]	[15..30]	2	r7
8	[45..60]	<30	2	r8

1st Rule: The 1st rule calculates the base number of holidays, or vacation days, that an employee is entitled to based on the calculation, age – service. Note that "-" appears for the two unary tests. As a result, this rule will be triggered for any employee, regardless of their age or years of service.

2nd Rule: The 2nd rule first tests if age is greater than or equal to 60. If so, then 3 more days are added to the employee's total number of vacation days.

3rd Rule: The 3rd rule tests if the number of years of service are greater than or equal to 30. If so, then 3 more days are added to the employee's total number of vacation days.

4th Rule: The 4th rule tests if age is less than 18. If so, then 5 more days are added to the employee's total number of vacation days.

5th Rule: The 5th rule tests if age is greater than or equal to 60. If so, then 5 more days are added to the employee's total number of vacation days.

6th rule: The 6th rule tests if service is greater than or equal to 30. If so, then 5 more days are added to the employee's total number of vacation days.

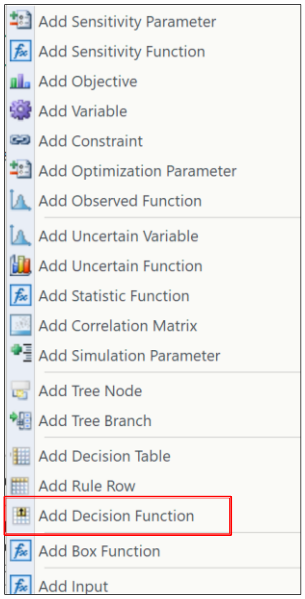
7th Rule: The 7th rule contains two unary tests. The first tests if age is between 18 and 60, inclusive *and* the 2nd tests if service is between 15 and 30, inclusive. If both of these conditions are true, then 2 more days are added to the total number of vacation days.

8th Rule: The last rule tests if age is between 45 and 60, inclusive and if service is less than 30. If both these tests are true, then 2 more days are added to the total number of vacation days.

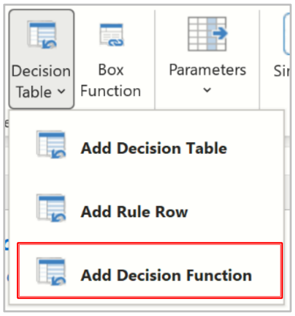
Step 13: Enter the Decision Table Inputs. Cells G11 and H11 have been designated as the input values for age (45) and service (15). These are the values of which the decision table depends.

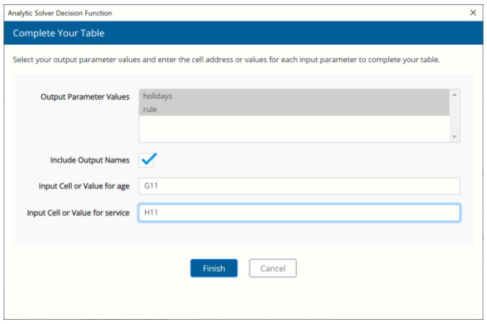
Step 14: Enter the PsiDecTable() function to the worksheet. Click Decisions – Add Decision Function on the Analytic Solver Comprehensive ribbon, click the + on the Solver Task Pane then Add Decision Function, or simply type the formula into cell I26.

Analytic Solver Comprehensive Taskpane



Analytic Solver Comprehensive Ribbon





Select your output parameter values and enter the cell address or values for each input parameter to complete your table.

Output Parameter Values: holidays, rule

Include Output Names ✓

Input Cell or Value for age: G11

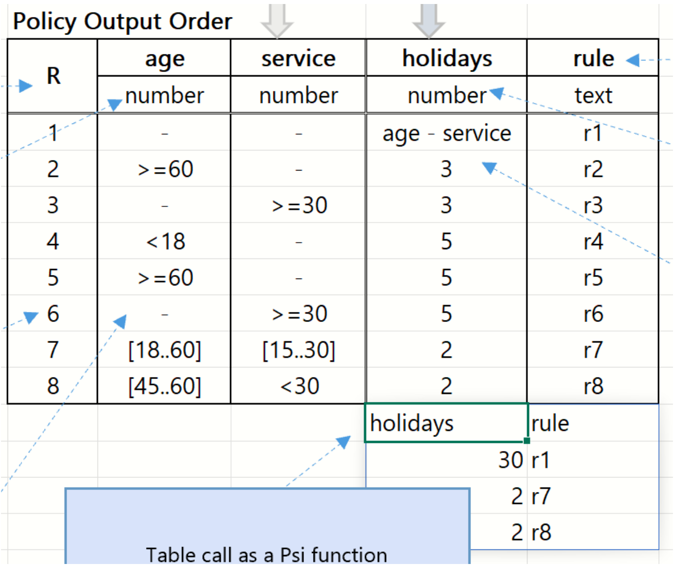
Input Cell or Value for service: H11

- Select both Output Parameters: "holidays" and "rule". Both the number of holidays the employee is entitled to and the rule that grants each holiday will be included in the output. To only receive 1 column of output, select only "holidays" or "rule".
- Select the Output Names checkbox. A column heading will be added to the top of each output column in the output.
- Enter G11 for the first Input Cell (Age)
- Enter H11 for the second Input Cell (Service)

The formula, =PsiDecTable(F16:J25,{"holidays","rule"},TRUE, G11, H11) w

Step 15: Click the green arrow on the Model tab of the Solver Task Pane to calculate the Decision Table.

If using Excel 2019 and later, the result will "spill" into 4 rows and 2 columns as shown in the screenshot above starting with cell I26. In the first column you see the rule output: 30, 2, and 2. The second column contains the rule numbers, r1, r7 and r8. If Include Output Names is not selected for Heading, then the output will spill into only 3 rows.



Policy Output Order

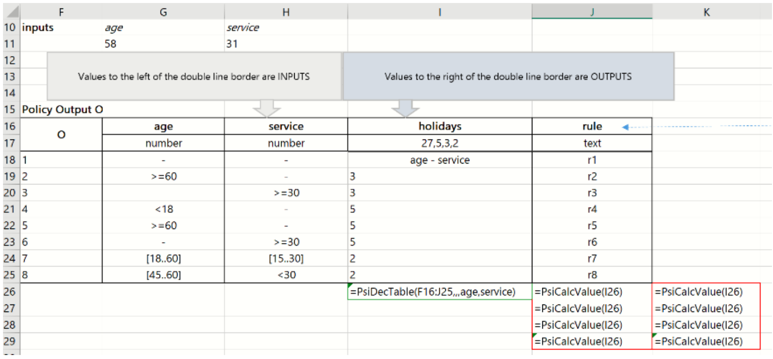
R	age	service	holidays	rule
	number	number	number	text
1	-	-	age - service	r1
2	>=60	-	3	r2
3	-	>=30	3	r3
4	<18	-	5	r4
5	>=60	-	5	r5
6	-	>=30	5	r6
7	[18..60]	[15..30]	2	r7
8	[45..60]	<30	2	r8

holidays	rule
30	r1
2	r7
2	r8

Table call as a Psi function

Note: If you are using an earlier version of Excel, which does not support Dynamic Arrays, you'll need to use the PsiCalcValues() function, entered as an array (CTRL + SHIFT + ENTER) in cells J26:K28, to view the results.

Note: It's important when using Dynamic arrays in Excel that you enter the array in a cell location where there are enough blank cells to contain the array. This example contains two output values: holidays and rule. Therefore, we automatically know that the dynamic array will contain two columns, one containing the number of holidays and one containing the rule label. The first column will start in the cell containing the PsiDecTable() function. However, you will not know in advance how many rows will be used. This is also true when entering PsiCalcValue as an array in earlier versions of Excel.



	F	G	H	I	J	K
10	inputs	*age*	*service*			
11		58	31			
12						
13		Values to the left of the double line border are INPUTS		Values to the right of the double line border are OUTPUTS		
14						
15	Policy Output O					
16	O	age	service	holidays	rule	
17		number	number	27,5,3,2	text	
18	1	-	-	age - service	r1	
19	2	>=60	-	3	r2	
20	3		>=30	3	r3	
21	4	<18	-	5	r4	
22	5	>=60	-	5	r5	
23	6	-	>=30	5	r6	
24	7	[18..60]	[15..30]	2	r7	
25	8	[45..60]	<30	2	r8	
26				=PsiDecTable(F16:J25,,,age,service)	=PsiCalcValue(I26)	=PsiCalcValue(I26)
27					=PsiCalcValue(I26)	=PsiCalcValue(I26)
28					=PsiCalcValue(I26)	=PsiCalcValue(I26)
29					=PsiCalcValue(I26)	=PsiCalcValue(I26)

During recalculation, each rule is evaluated. Both unary tests must be satisfied in order for a "hit" to occur and the results to be added to the result collection.

1. The first rule, under age, uses the unary tests age: "-" and service: "-", which, as you will recall, returns TRUE against any value. Therefore, the first "hit" occurs with R1, as age – service or 45 – 15 = 30.
2. The next rule tests whether age is >= 60 which is not successful as 45 is not greater than or equal to 60.
3. The next rule (rule 3) returns TRUE for age ("-") but FALSE for service (>= 30).
4. The fourth rule returns FALSE for age <= 18.

5. The fifth rule returns FALSE for age >= 60.
6. The sixth rule returns TRUE for "-" for age but FALSE for service >= 30.
7. The 7th rule returns TRUE for both the age interval [18..60] and the service interval [15..30].
8. The 8th rule returns TRUE for the age interval [45..60] but TRUE for service <30.

Three rules, r1, r7 and r8 are added to the result collection. The results for these three rules are 30, 2, and 2, respectively. After the results are updated, the HIT policy is checked. In this example, the hit policy is "R" or "Rule Order" which specifies that if multiple rules are hit, the collection of results will be returned in order of activation. Therefore, our results are returned as 30 (r1), 2 (r7) and 2 (r8).

If no rules are activated, or triggered, Analytic Solver will still recalculate the decision table but will issue a warning that no rules were triggered.

```
Recalculation finished successfully.
```

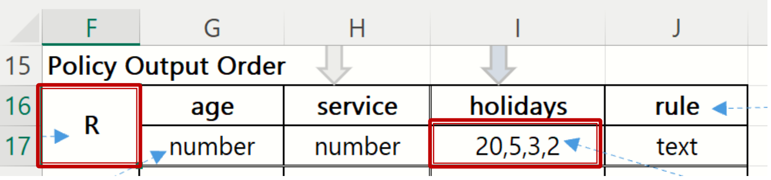
```
Warning: No hit for decision table Policy Output
Order at 'Decision Table'!I26; 2 No hits found.
```

Open the example DT Hit Policy Examples.xlsx to see how the different Hit policies work.

Note: During parse-time, each PsiDecTable() function is uniquely identified by it's cell address. A "completeness test" ensures that all input/output entries per column cover the domain of the input/output values in the same column. If so, these expressions are evaluated at run-time. Type verification is performed for those columns in which a data type is specified rather than an input/output value.

A Note About Specifying an Output Domain

Enter the domain, 20, 5, 3, 2, into cell I17 as shown below. When a domain is entered into the cell beneath an input or output parameter, Analytic Solver Comprehensive will return an error if a value outside of that domain is calculated.



	F	G	H	I	J
15	Policy Output Order				
16	R	age	service	holidays	rule
17		number	number	20,5,3,2	text

Click the green arrow on the Model tab on the Solver Task Pane to perform a recalculation of the decision table. Then click the output tab to view the error:

```
Output entry not found in the output values domain
PSI could not parse Excel formulas - Use Excel Interpreter.
Error occurred at range 'Decision Table'!$I$26
[tblPolicyOutputOrder]
```

This error was generated because "20" was not found in the output results. As you can see, it's important to take into account your range of output values when generating a domain for your decision table output.

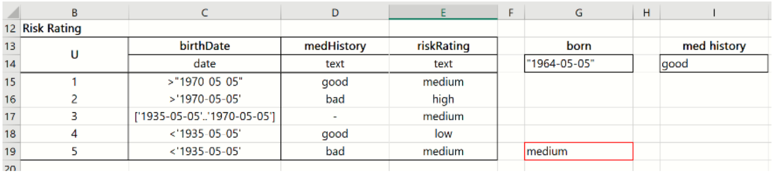
Additional Examples

This section presents a special class of tables, in which the specific Date, Time, and Duration data types are utilized.

Decision Table Containing Dates

Open the first tab within the workbook DT Date & Time Examples.xlsx. This table returns the credit risk rating of an individual based on their age and medical history. The PsiDecTable function appears in cell G19 = PsiDecTable(B13:E19,,, bd, mh) where

- B13:E19 is the location of the decision table,
- bd (cell G14 containing "1964-05-05") and mh (cell H14 containing "good") are the input variables.



	B	C	D	E	F	G	H	I
12	Risk Rating							
13	U	birthDate	medHistory	riskRating		born		med history
14		date	text	text		"1964-05-05"		good
15	1	>"1970 05 05"	good	medium				
16	2	>'1970-05-05'	bad	high				
17	3	['1935-05-05'..'1970-05-05']	-	medium				
18	4	<'1935-05-05'	good	low				
19	5	<'1935-05-05'	bad	medium		medium		

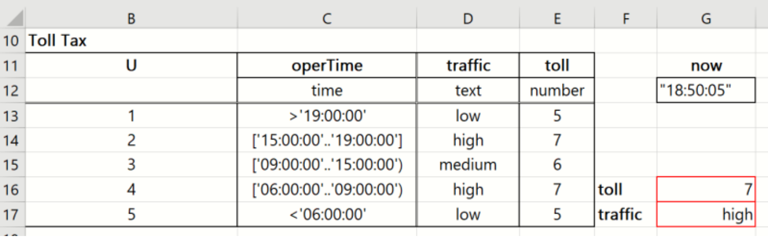
Notice that the string values in columns D (medHistory) and E (riskRating) in the table are entered without quotes. This is because Analytic Solver Comprehensive always assumes any output as an expression requiring parsing. If a single word or "token" is detected, the Psi Interpreter first checks if it is a local scope variable name. If not, the word-token is assumed to be a string. In the first example table the expression *age – service* existed in the "holidays" column. Since age and service are variables connected through the operator '**-**', the Psi Interpreter assumed the entry as an expression to be evaluated. If instead, we wanted to enter that expression as a string value, then we would need to surround the expression with quotation marks.

The situation with dates is however, different. While in Excel we may enter a date without " ", when a Date is part of a Decision table, we must surround the date with single ' ' or double " ". S-FEEL defines a single format for the **Date** type **'yyyy-mm-dd'**. Optionally a date may begin with the letter D, for example "D1964-05-05". Other date type formats in Excel are not valid in S-FEEL. When we pass a date as an input parameter to a Decision table, the date must be a string formatted as S-FEEL Date. If we enter an Excel date in that cell, it will be automatically converted into the date representing number and the Decision table will still recognize it as date. However, we strongly recommend using the S-FEEL format outside of Decision tables for readability of the model.

In this example,the unique rule which matches the input birthdate "1964-05-05" is rule #3, because "1964-05-05" is in the date interval ['1935-05-05'..'1970-05-05'].

Decision Table Containing Time

The Psi Interpreter will convert any time entries in a decision table into the Time format, "hh:mm:ss". Optionally, we could enter the times using the format beginning with the letter **T** such as in "**T**18:29:30". Usually this time format denotes local time. If we want to specify the UTC time, we can simply add **z** at the end of the format, for example "**T**18:29:30z". However, it's important to note that the z time format is just a readable format. The Psi Interpreter cannot effectively convert local and zulu times, because the location of the local time is not specified in the supported format.



	B	C	D	E	F	G
10	Toll Tax					
11	U	operTime	traffic	toll		now
12		time	text	number		"18:50:05"
13	1	>'19:00:00'	low	5		
14	2	['15:00:00'..'19:00:00']	high	7		
15	3	['09:00:00'..'15:00:00')	medium	6		
16	4	['06:00:00'..'09:00:00')	high	7	toll	7
17	5	<'06:00:00'	low	5	traffic	high

Click the 2nd tab in the workbook to open the example shown above which returns the traffic prediction (low, medium, or high) for a toll station based on the time of day.

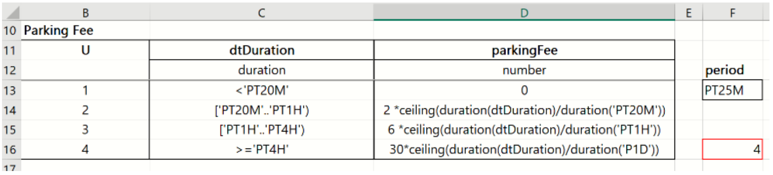
To avoid the downward spill of the result for the two outputs, we use the optional output argument in both PsiDecTable() functions in cells G16 and G17.

G16 = PsiDecTable(B14:E20, "toll",, byNow)

G17 = PsiDecTable(B14:E20, "traffic",, byNow)

Decision Table Containing Duration

Finally, let's take a look into an example using the data type, duration. Click the third tab in the workbook to open the decision table, shown below, which computes the output parking fee according to the input parking time. The table is invoked through the Excel formula in cell F16 = PsiDecTable(B11:D16, , , period).



	B	C	D	E	F
10	Parking Fee				
11	U	dtDuration	parkingFee		
12		duration	number		period
13	1	<'PT20M'	0		PT25M
14	2	['PT20M'..'PT1H')	2 *ceiling(duration(dtDuration)/duration('PT20M'))		
15	3	['PT1H'..'PT4H')	6 *ceiling(duration(dtDuration)/duration('PT1H'))		
16	4	>='PT4H'	30*ceiling(duration(dtDuration)/duration('P1D'))		4

There are two types of formats for Duration:

- **P1Y2M -** measures duration in months
- **P1DT1H20M10S** – measures duration in seconds

The individual components in these formats are optional. For example, if we have 0 hours to represent, we may skip the hour component. For example, we can represent a 20 minutes duration using either: PT0H20M0S or PT20M.

While we can skip a duration component in the format, the order of components in a format must be followed. For example, we cannot first enter minutes and then hours; PT20M1H is not a currently supported syntax.

Hit Policies

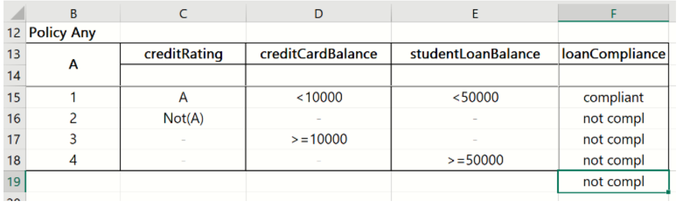
This section presents a collection of tables to illustrate each of the 7 Hit policies currently supported by Analytic Solver Comprehensive: Any, Collect, First, Rule Order, Output Order, Priority Order and Unique.   Open the example workbook, DT Hit Policy Examples.xlsx. This workbook contains 7 tabs, one tab for each example.

Hit Policy: Any

The decision table appearing on the Any tab returns the loan compliance for an individual given their credit rating, current card balance, and current student loan balance.

The PsiDecTable() function is located in cell F19. This function returns "compliant" or "not compliant" given a credit rating of "B", a credit card balance of $12,000, and a student loan balance of $75,000. The Hit policy Any will return only one result even if multiple rules overlap.

F19 = PsiDecTable(B13:F18, , , "B", 12000, 75000)

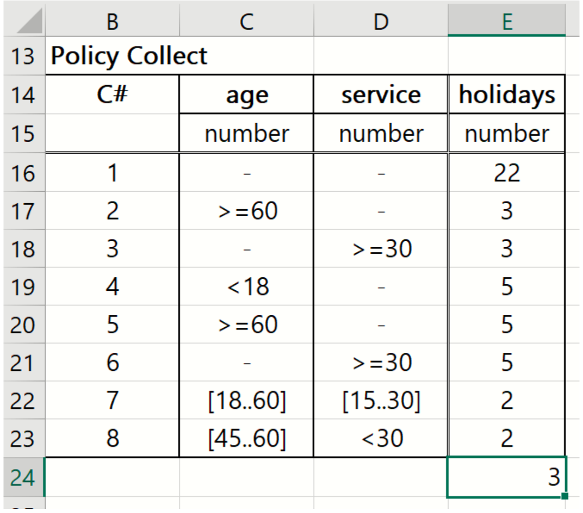


	B	C	D	E	F
12	Policy Any				
13	A	creditRating	creditCardBalance	studentLoanBalance	loanCompliance
14					
15	1	A	<10000	<50000	compliant
16	2	Not(A)	-	-	not compl
17	3	-	>=10000	-	not compl
18	4	-	-	>=50000	not compl
19					not compl

In this table, all rules except for rule 1 are evaluated successfully for a "hit". However, the result collection only includes 1 "not compl" entry.

Hit Policy: Collect

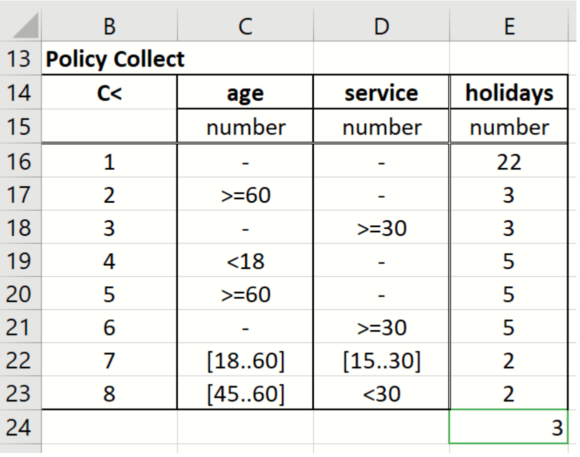
Click the Collect tab to display the 2nd Hit Policy example. This decision table returns the total number of holidays allotted to an employee based on their age and number of years in service. In the previous example, the Hit Policy was equal to Unique which resulted in all successfully hit rules returning their unique result. In this example, the Collect policy is used with the "+" operator. This operator returns the sum of the collection of results in rule order when multiple rules are applicable. The Collect Policy allows 4 operators, + (sum), < (return the minimum of matched output values), > (return the maximum of matched output values) and # (return the number of matched output values).



	B	C	D	E
13	Policy Collect			
14	C#	age	service	holidays
15		number	number	number
16	1	-	-	22
17	2	>=60	-	3
18	3	-	>=30	3
19	4	<18	-	5
20	5	>=60	-	5
21	6	-	>=30	5
22	7	[18..60]	[15..30]	2
23	8	[45..60]	<30	2
24				3

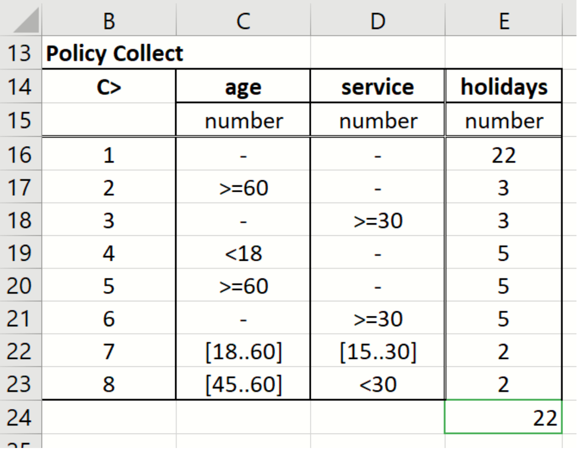
In this example, the PsiDecTable() function appears in cell E24 = PsiDecTable(B14:E23, , , 58, 31). As in the previous example, three rules, 1, 3 and 6, are included in the results returning the values, 22, 3 and 5, respecitvely. However, since this Hit policy sums the results, the answer 30 is returned.

Let's see what happens when we change the Hit Policy to C<. Simply select cell C+ and enter C<, then click the green arrow on the Model tab of the Solver Task Pane to recalculate the table. When the Hit Policy is equal to C<, PsiDecTable() returns the minimum result value, 3.



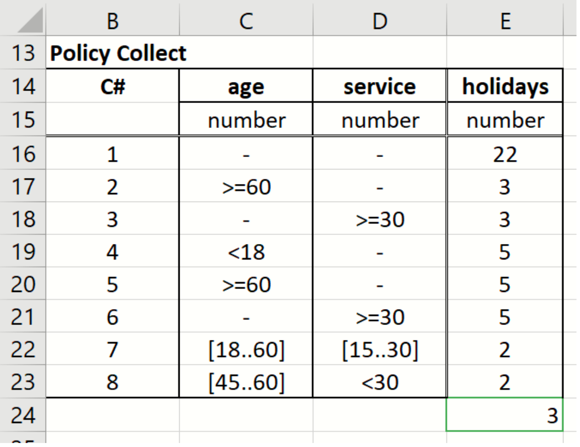
	B	C	D	E
13	**Policy Collect**			
14	**C<**	**age**	**service**	**holidays**
15		number	number	number
16	1	-	-	22
17	2	>=60	-	3
18	3	-	>=30	3
19	4	<18	-	5
20	5	>=60	-	5
21	6	-	>=30	5
22	7	[18..60]	[15..30]	2
23	8	[45..60]	<30	2
24				3

Let's repeat these steps for "C"> to find the maximum result value of 22,



	B	C	D	E
13	**Policy Collect**			
14	**C>**	**age**	**service**	**holidays**
15		number	number	number
16	1	-	-	22
17	2	>=60	-	3
18	3	-	>=30	3
19	4	<18	-	5
20	5	>=60	-	5
21	6	-	>=30	5
22	7	[18..60]	[15..30]	2
23	8	[45..60]	<30	2
24				22

and "C#" to find the number of returned results, 3.



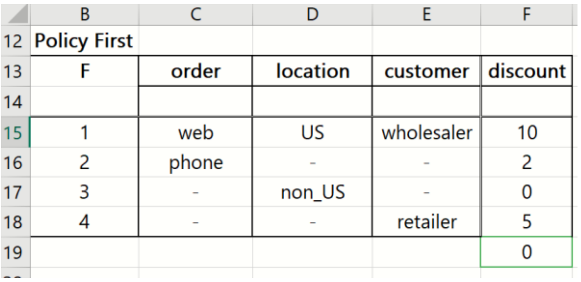
	B	C	D	E
13	**Policy Collect**			
14	**C#**	**age**	**service**	**holidays**
15		number	number	number
16	1	-	-	22
17	2	>=60	-	3
18	3	-	>=30	3
19	4	<18	-	5
20	5	>=60	-	5
21	6	-	>=30	5
22	7	[18..60]	[15..30]	2
23	8	[45..60]	<30	2
24				3

Hit Policy: First

Click the First tab to display the next Hit Policy example. In this example, a percentage discount is returned based on where an order originated, on the web or by phone, the location of the order, inside or outside the US, and customer type, wholesaler or retail according to the First Hit Policy. When this policy is in use, only the first entry in the result collection is returned.

The PsiDecTable() function appears in cell F19.

F19=PsiDecTable(B13:F18, 3,,, "web", "non_US", "retailer")

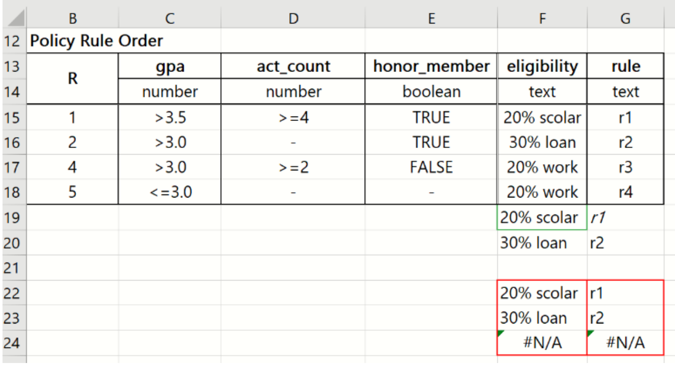


	B	C	D	E	F
12	Policy First				
13	F	order	location	customer	discount
14					
15	1	web	US	wholesaler	10
16	2	phone	-	-	2
17	3	-	non_US	-	0
18	4	-	-	retailer	5
19					0

Two rules are "hit", rule 3 and rule 4. However, since we are using the First Hit Policy, only the result for rule 3 is returned, 0.

Hit Policy: Rule Order

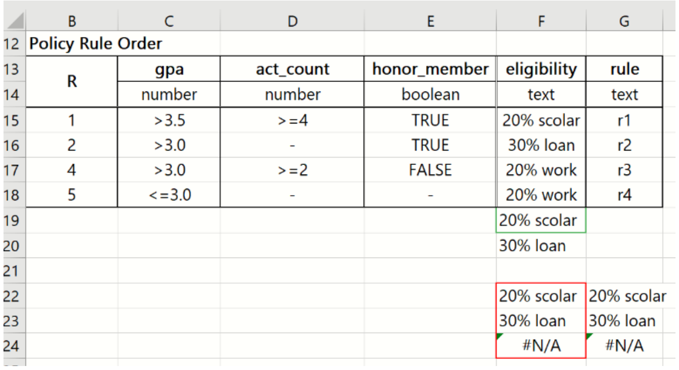
Click the next tab to display the Rule Order Hit Policy example. This table returns the amount of money available to a student based on their grade point average and number of times they took the ACT. Funds may be provided via a scholarship, loan, or work study program. The "Rule Order" Hit policy returns the collection of results in priority order of the listed output values. The PsiDecTable() function is located in cell F19 = PsiDecTable(B13:G18, "eligibility", , 3.6, 4, TRUE). Given an honors student with a gpa = 3.6 and an act_count = 4, this student would be eligible to receive a scholarship covering 20% of tuition (rule 1) and a student loan covering 30% of tuition (rule 2).



	B	C	D	E	F	G
12	Policy Rule Order					
13	R	gpa	act_count	honor_member	eligibility	rule
14		number	number	boolean	text	text
15	1	>3.5	>=4	TRUE	20% scolar	r1
16	2	>3.0	-	TRUE	30% loan	r2
17	4	>3.0	>=2	FALSE	20% work	r3
18	5	<=3.0	-	-	20% work	r4
19					20% scolar	*r1*
20					30% loan	r2
21						
22					20% scolar	r1
23					30% loan	r2
24					#N/A	#N/A

Rules 3 and 4 are not applicable because the student is an honor student (honor_member = TRUE) and the student's GPA is greater than 3.0.

Cells F22:G24 contain PsiCalcValue(F19). Use these values to return the contents of the result collection when using a version of Excel that does not support Dynamic Arrays. Note that if this function is entered into more than the number of cells required, #N/A is returned.

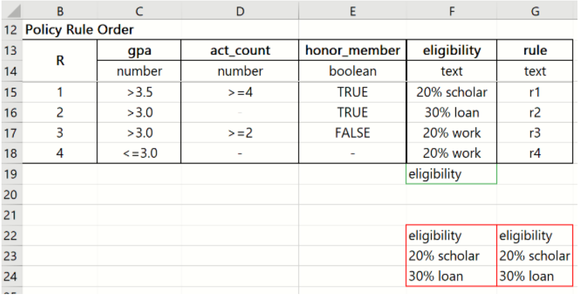


	B	C	D	E	F	G
12	Policy Rule Order					
13	R	gpa	act_count	honor_member	eligibility	rule
14		number	number	boolean	text	text
15	1	>3.5	>=4	TRUE	20% scolar	r1
16	2	>3.0	-	TRUE	30% loan	r2
17	4	>3.0	>=2	FALSE	20% work	r3
18	5	<=3.0	-	-	20% work	r4
19					20% scolar	
20					30% loan	
21						
22					20% scolar	20% scolar
23					30% loan	30% loan
24					#N/A	#N/A

Adding in the two optional arguments, output_names and return_header, returns the header "eligibility", and only the eligibility text.

F19 = PsiDecTable(B13:G18, "eligibility", TRUE, 3.6, 4, TRUE).

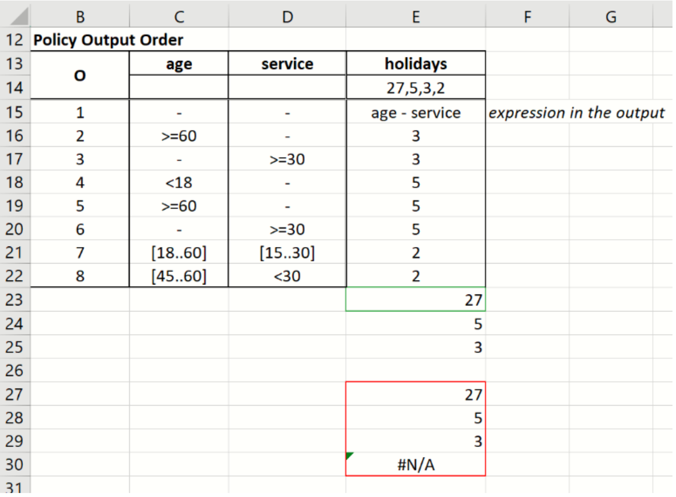
Notice that when only one column is returned, the PsiCalcValue array simply repeats the results in cells G22:G23.



	B	C	D	E	F	G
12	Policy Rule Order					
13	R	gpa	act_count	honor_member	eligibility	rule
14		number	number	boolean	text	text
15	1	>3.5	>=4	TRUE	20% scholar	r1
16	2	>3.0	-	TRUE	30% loan	r2
17	3	>3.0	>=2	FALSE	20% work	r3
18	4	<=3.0	-	-	20% work	r4
19					eligibility	
20						
21						
22					eligibility	eligibility
23					20% scholar	20% scholar
24					30% loan	30% loan

Hit Policy: Output Order

Click the next tab to display the Output Order Hit Policy example. This example returns to our original example where the number of holidays allotted to an employee is returned based on the employee's age and number of years of service. The output is returned in the order as given in the Output values header (cell D3). The PsiDecTable() function is located in cell E23 = PsiDecTable(B13:E22, , , 58, 31). According to the rules of the table, a 58 year old employee with 31 years of service is allotted 27 + 3 + 5 vacation days via rules 1, 3, and 6, respectively. However, since the Hit Policy specifies that the results must be returned priority order, the result collection is listed as 27, 5, and 3.

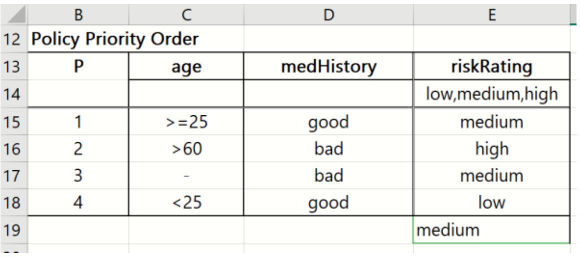


	B	C	D	E	F
12	Policy Output Order				
13	O	age	service	holidays	
14				27,5,3,2	
15	1	-	-	age - service	*expression in the output*
16	2	>=60	-	3	
17	3	-	>=30	3	
18	4	<18	-	5	
19	5	>=60	-	5	
20	6	-	>=30	5	
21	7	[18..60]	[15..30]	2	
22	8	[45..60]	<30	2	
23				27	
24				5	
25				3	
26					
27				27	
28				5	
29				3	
30				#N/A	

Hit Policy: Priority

Click the next tab to display the last Hit Policy example. According to the Priority Hit Policy, if multiple rules are "hit" and multiple results are collected, only one result will be returned. The priority of the returned result is defined by the order of the output values in the header.

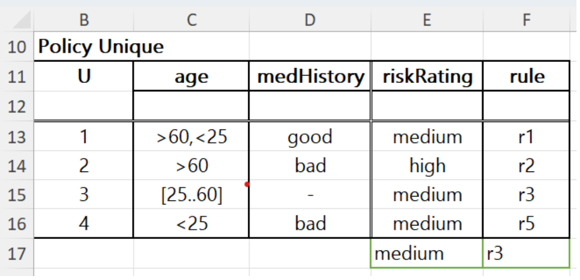
This example returns a risk of disease rating of low, medium, or high based on a person's age and medical history. The PsiDecTable() function in cell E18 (=PsiDecTable(B12:E17, , , 61, "bad")) returns "medium" for a 61 year old person with a "bad" medical history. Note that according to our criteria, two rules are returned as successful, rule 2 (age > 60 and medicalHistory = "bad") and rule 3 (age = "-" and medHistory = "medium"). Why isn't "high" returned? Because the priority list in cell B13 (low, medium, high) specifies that "medium" should have higher priority than "high".



	B	C	D	E
12	Policy Priority Order			
13	P	age	medHistory	riskRating
14				low,medium,high
15	1	>=25	good	medium
16	2	>60	bad	high
17	3	-	bad	medium
18	4	<25	good	low
19				medium

Hit Policy: Unique

Click the last tab to display the last Hit Policy example. According to the Unique Hit Policy, only one rule may "hit". If multiple rules are "hit" and multiple results are collected, an error will be returned.



	B	C	D	E	F
10	Policy Unique				
11	U	age	medHistory	riskRating	rule
12					
13	1	>60,<25	good	medium	r1
14	2	>60	bad	high	r2
15	3	[25..60]	-	medium	r3
16	4	<25	bad	medium	r5
17				medium	r3

E18 = PsiDecTable(B11:F16, "rule",, 54, "good")

This table uses the optional output_names argument to return a risk rating (only) for an individual based on their age and medical history (medHistory).   In this example, given an age or 54 and a "good" medical history, a "medium" risk rating is returned by rule 3.

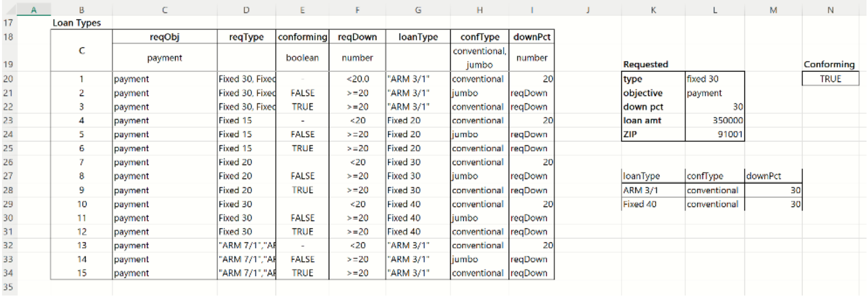
Merging Decision Table Results

The DT Loan Recommend Example illustrates how to use the PsiJoin() function to merge the results of one decision table with a 2nd table.

The decision table, in cells B18:I34, returns a loan or loans meeting various criteria. Cell K27 includes the PsiDecTable() formula. Two types of loans meet the 4 loan requirements (reqObj = payment, reqType = fixed 30, conforming = TRU and reqDown = 30): a conventional Arm 3\1 and a conventional Fixed 40. (See the results in cells J32:L34).

K27=PsiDecTable(B18:I34, , True, reqObj, reqType, conforming, reqDown, TRUE)

Notice that the optional "header" argument has been passed to PsiDecTable() in cell K27. This was done in preparation for the use of the PsiJoin() function to merge the results with the table on the Bank Rates tab.



Loan Types

C	reqObj payment	reqType	conforming boolean	reqDown number	loanType	confType conventional, jumbo	downPct number
1	payment	Fixed 30, Fixed	-	<20.0	"ARM 3/1"	conventional	20
2	payment	Fixed 30, Fixed	FALSE	>=20	"ARM 3/1"	jumbo	reqDown
3	payment	Fixed 30, Fixed	TRUE	>=20	"ARM 3/1"	conventional	reqDown
4	payment	Fixed 15	-	<20	Fixed 20	conventional	20
5	payment	Fixed 15	FALSE	>=20	Fixed 20	jumbo	reqDown
6	payment	Fixed 15	TRUE	>=20	Fixed 20	conventional	reqDown
7	payment	Fixed 20		<20	Fixed 30	conventional	20
8	payment	Fixed 20	FALSE	>=20	Fixed 30	jumbo	reqDown
9	payment	Fixed 20	TRUE	>=20	Fixed 30	conventional	reqDown
10	payment	Fixed 30		<20	Fixed 40	conventional	20
11	payment	Fixed 30	FALSE	>=20	Fixed 40	jumbo	reqDown
12	payment	Fixed 30	TRUE	>=20	Fixed 40	conventional	reqDown
13	payment	"ARM 7/1","AF	-	<20	"ARM 3/1"	conventional	20
14	payment	"ARM 7/1","AF	FALSE	>=20	"ARM 3/1"	jumbo	reqDown
15	payment	"ARM 7/1","AF	TRUE	>=20	"ARM 3/1"	conventional	reqDown

Requested

type	fixed 30
objective	payment
down pct	30
loan amt	350000
ZIP	91001

Conforming

TRUE

loanType	confType	downPct
ARM 3/1	conventional	30
Fixed 40	conventional	30

Click the Bank Rates tab to view the 2nd table in cells A2:I8. This table lists six banks offering loans in the area. Notice that no bank is offering the Fixed 40 loan.

The PsiJoin function has been entered as an array in cells A12:L15.

A12:L15{= PsiJoin('Loan Types'!K27:M29, A2:I8, "loanType = loanType, confType = confType, downPct >= 'minDown %'")}

This function joins results from the decision table function in Loan Types!K27 with the second table on the Bank Rates tab.

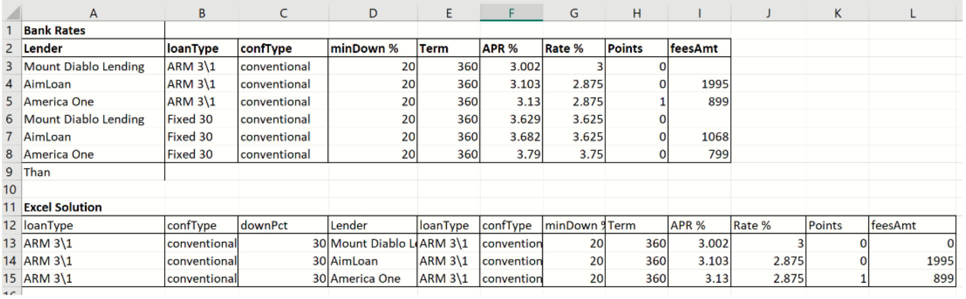
PsiJoin() takes three arguments, Table1, Table2, and Clause. The Table1 argument passes the first table, in this example, the table in Loan Types!B18:I34. The Table2 argument passes the second table, in this example, the table in Bank Rates!A2:I8. The third argument, Clause, passes the conditions for joining the table. In this example, the results and the table are joined via three columns: loanType, confType, and downPct. This argument must be surround by quotes.

Note that in this example, two conditions used to join the results from PsiDecTable() and the Bank Rates table are the same, i.e. loanType = loanType and confType = confType. This is not a requirement; table headers can be different.

The third condition, downPct >= 'minDown %' requires that the downPct in the Loan Types decision table must be greater than the "minDown %" in the Bank Rates table. In other words, the down payment percentage in the Loan Types table must be greater than the minimum percentage down in the Bank Rates table in order for the loan to be selected and displayed by PsiJoin().

Note: Recall that the PsiDecTable() function in cell Loan Types!K27 included the optional "header" argument. If the results had not included this argument, it would not have been possible to use the PsiJoin() function complete the merge.

All three columns are common to both the PsiDecTable() results and the Bank Rates table. Three banks offer a conventional ARM 3\1 loan: Mount Diablo, AimLoan, and America One. The first three columns (A12:L15) are the three columns passed to the Clause argument: loanType, confType, and downPct. Columns D through L contain the remaining information for each offered loan: Lender, loanType, confType, minDown%, Term, APR %, Rate %, Points, and feesAmt.

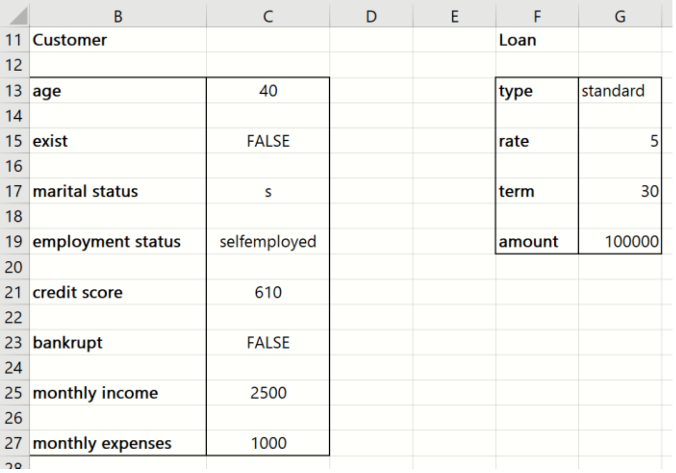


	A	B	C	D	E	F	G	H	I	J	K	L
1	**Bank Rates**											
2	**Lender**	**loanType**	**confType**	**minDown %**	**Term**	**APR %**	**Rate %**	**Points**	**feesAmt**			
3	Mount Diablo Lending	ARM 3\1	conventional	20	360	3.002	3	0				
4	AimLoan	ARM 3\1	conventional	20	360	3.103	2.875	0	1995			
5	America One	ARM 3\1	conventional	20	360	3.13	2.875	1	899			
6	Mount Diablo Lending	Fixed 30	conventional	20	360	3.629	3.625	0				
7	AimLoan	Fixed 30	conventional	20	360	3.682	3.625	0	1068			
8	America One	Fixed 30	conventional	20	360	3.79	3.75	0	799			
9	Than											
10												
11	**Excel Solution**											
12	loanType	confType	downPct	Lender	loanType	confType	minDown	Term	APR %	Rate %	Points	feesAmt
13	ARM 3\1	conventional	30	Mount Diablo L	ARM 3\1	convention	20	360	3.002	3	0	0
14	ARM 3\1	conventional	30	AimLoan	ARM 3\1	convention	20	360	3.103	2.875	0	1995
15	ARM 3\1	conventional	30	America One	ARM 3\1	convention	20	360	3.13	2.875	1	899

Using Cascading Decision Tables

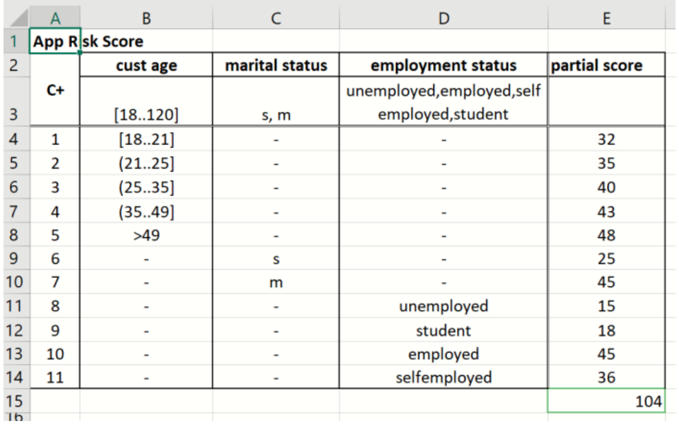
Open the DT Loan Strategy Model to learn how to use cascading decision tables where the output of one decision table is the input to another.

This Input worksheet contains the input parameters to the Loan Strategy model contained on the Loan Strategy and Post Bureau worksheets. Input parameters are all constant values.



	B	C	D	E	F	G
11	Customer				Loan	
12						
13	age	40			type	standard
14						
15	exist	FALSE			rate	5
16						
17	marital status	s			term	30
18						
19	employment status	selfemployed			amount	100000
20						
21	credit score	610				
22						
23	bankrupt	FALSE				
24						
25	monthly income	2500				
26						
27	monthly expenses	1000				

Click the Loan Strategy worksheet. This worksheet contains six cascading decision tables. The first table appears in cells A2:E14. This table returns a partial "risk of default" score based on a loan applicant's age, marital and employment status. Higher scores are given to older, employed, married applicants. Cell E15 contains the PsiDecTable() function.

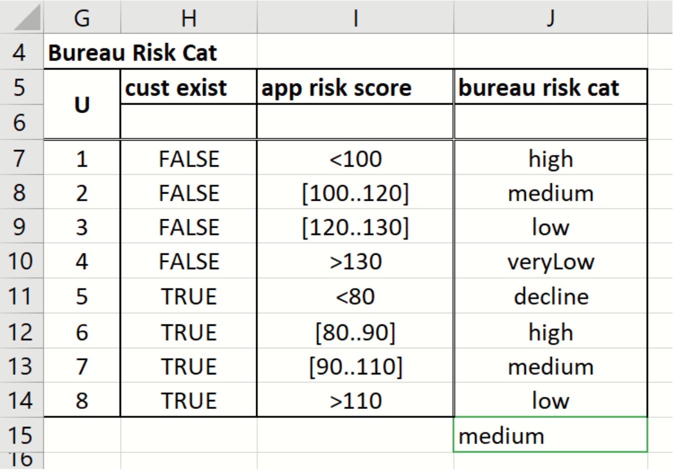


	A	B	C	D	E
1	App Risk Score				
2		cust age	marital status	employment status	partial score
3	C+	[18..120]	s, m	unemployed,employed,self employed,student	
4	1	[18..21]	-	-	32
5	2	(21..25]	-	-	35
6	3	(25..35]	-	-	40
7	4	(35..49]	-	-	43
8	5	>49	-	-	48
9	6	-	s	-	25
10	7	-	m	-	45
11	8	-	-	unemployed	15
12	9	-	-	student	18
13	10	-	-	employed	45
14	11	-	-	selfemployed	36
15					104

E15=PsiDecTable(A2:E14, , , custAge, maritalSt, employSt)

The input parameters from the Input worksheet are: custAge="40", maritalSt = "s", employSt = "selfemployed". The applicable rules for these input parameters are: 4, 6, and 11 resulting in partial scores of 43, 25, and 36, respectively. Since the Hit Policy is "C+" the result collection is added together, 43 + 25, + 36 = 104.

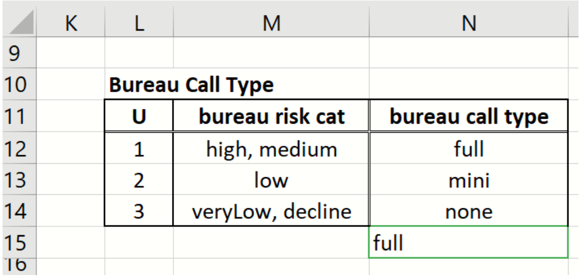
The next decision table is located in cells G5:J14, the Bureau Risk Cat table. This table returns a risk category of high, medium, or low based on their customer status (existing or not) and their application risk score. Non-existing customers are allowed a higher risk score. Cell J15 contains the PsiDecTable() function



	G	H	I	J
4	Bureau Risk Cat			
5	U	cust exist	app risk score	bureau risk cat
6				
7	1	FALSE	<100	high
8	2	FALSE	[100..120]	medium
9	3	FALSE	[120..130]	low
10	4	FALSE	>130	veryLow
11	5	TRUE	<80	decline
12	6	TRUE	[80..90]	high
13	7	TRUE	[90..110]	medium
14	8	TRUE	>110	low
15				medium

J15 =PsiDecTable(G5:J14, , , custExist, tblAppRiskScore) where custExist = False and tblAppRiskScore() refers to the value in cell E15, or 104. Based on these conditions, one rule is "hit", rule 2. This rule returns "medium" for the bureau risk category. Since the hit policy is equal to U (Unique) only one result is returned, even if multiple results are returned.

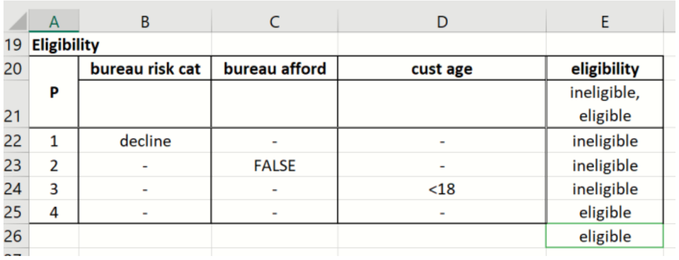
The next decision table, Bureau Call Type, is located in cells L11:N14. This table returns the bureau call type based on the bureau risk category found in the previous decision table, Bureau Risk Cat. Cell N15 contains the PsiDecTable() function.



	K	L	M	N
9				
10		**Bureau Call Type**		
11		**U**	**bureau risk cat**	**bureau call type**
12		1	high, medium	full
13		2	low	mini
14		3	veryLow, decline	none
15				full
16				

N15=PsiDecTable(L11:N14, , , tblBureauRiskCat) where tblBureauRiskCat is equal to the result of the PsiDecTable() function in cell J15, or "medium". Only 1 rule is "hit", rule 1, which returns "full".

The next decision table, Eligibility, is located in cells A20:E25. This table returns the applicant's eligibility based on three criteria: the bureau risk category, from cell J15, bureau afford, calculated in cell C36, and customer age, located in cell Input!C13. Since the Hit Policy is equal to P (Priority), the result collection will be returned in the order listed according to the list in cell E21. The PsiDecTable() function is located in cell E26.



	A	B	C	D	E
19	**Eligibility**				
20		**bureau risk cat**	**bureau afford**	**cust age**	**eligibility**
21	**P**				ineligible, eligible
22	1	decline	-	-	ineligible
23	2	-	FALSE	-	ineligible
24	3	-	-	<18	ineligible
25	4	-	-	-	eligible
26					eligible

E26 = PsiDecTable(A20:E25, , , tblBureauRiskCat, bureauAfford, custAge)

Intermediate Formulas

Cell C36 calculates the bureau afford statistic, =IF(disposIncome * tblCreditContFactor > monthInstall, TRUE, FALSE)

where disposIncome is calculated in cell C34, tblCreditContFactor is calculated in cell N26 (more on this later) and monthInstall is calculated in cell C32.

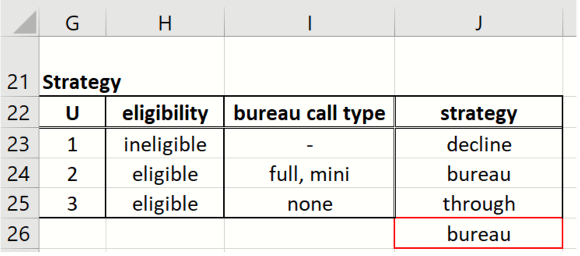
Cell 34 calculates an applicant's disposable income using the formula, =monthIncome **-** (monthRepay + monthExpenses) where monthIncome = 2500, monthRepay is calculated in cell C30 and monthExpenses = 1000.

Cell C30 calculates the monthly loan replayment amount using the formula, -PMT(loanRate%/12, loanTerm*12, loanAmnt) where loanRate is 5%, loanTerm = 30 and loanAmnt = 100,000.

Finally, cell C28 calculates the monthly fee using the IF statement, = IF(loanType = "standard", 20, IF(loanType = "special", 25, 0)), where loanType = standard.

Using the final input values of tblBureauRiskCat = medium, bureauAfford = True, and custAge = 40, the last rule, rule 4, is the only rule "hit", returning "eligible".

The next decision table is located in cells G22:J25. This table returns the loan strategy based on the applicant's eligibility and bureau call type. The PsiDecTable() function is located in cell J26.

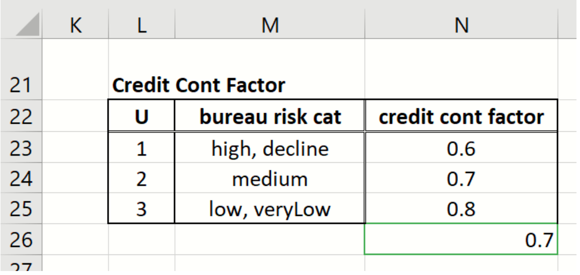


Strategy

U	eligibility	bureau call type	strategy
1	ineligible	-	decline
2	eligible	full, mini	bureau
3	eligible	none	through
			bureau

J26=PsiDecTable(G22:J25, , , tblEligibility, tblBureauCallType) where tblEligibility is calculated in cell E26 and tblBureauCallType is calculated in cell N15. Both of these inputs are outputs of earlier decision tables. Based on the results calculated by the previous decision tables, this PsiDecTable() function results in a "bureau" strategy.

Finally, the last decision table, Credit Cont Factor, is contained in cells L22:N25. The PsiDecTable() function is located in cell N26.



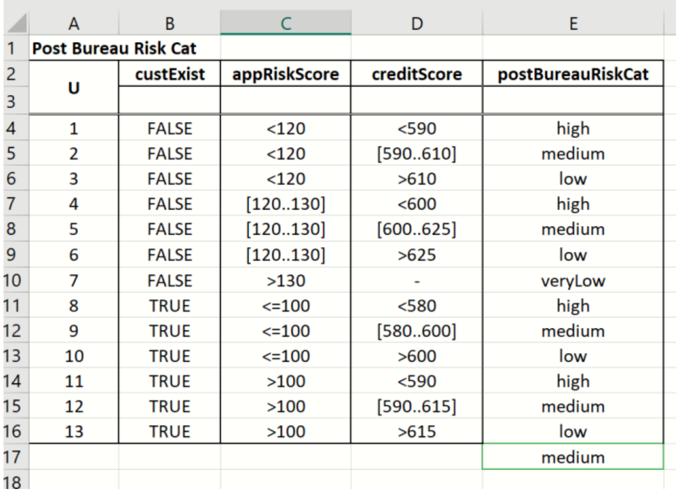
Credit Cont Factor

U	bureau risk cat	credit cont factor
1	high, decline	0.6
2	medium	0.7
3	low, veryLow	0.8
		0.7

N26 =PsiDecTable(L22:N25, , , tblBureauRiskCat) where tblBureauRiskCat is calculated in cell J15. Again, this input is an output from an earlier discussed decision table.

If the result of the PsiDecTable function in cell J26 would have been either "decline" or "through", then our work would be finished. We would either decline or accept the applicant, respectively. However, since "bureau" was returned, more analysis is required and on we must go. Click the Post Bureau worksheet to proceed.

The first decision table on the Post Bureau tab is the Post Bureau Risk Cat table. This table returns a Post Bureau Risk Category based on if the applicant is an existing customer, his/her application Risk Score and credit score. The PsiDecTable() function can be found in cell E17.

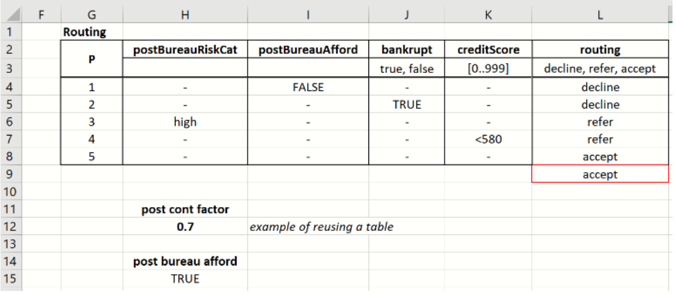


Post Bureau Risk Cat

U	custExist	appRiskScore	creditScore	postBureauRiskCat
1	FALSE	<120	<590	high
2	FALSE	<120	[590..610]	medium
3	FALSE	<120	>610	low
4	FALSE	[120..130]	<600	high
5	FALSE	[120..130]	[600..625]	medium
6	FALSE	[120..130]	>625	low
7	FALSE	>130	-	veryLow
8	TRUE	<=100	<580	high
9	TRUE	<=100	[580..600]	medium
10	TRUE	<=100	>600	low
11	TRUE	>100	<590	high
12	TRUE	>100	[590..615]	medium
13	TRUE	>100	>615	low
				medium

E17 = PsiDecTable(A2:E16, , , custExist, tblAppRiskScore, creditScore) where custExist is False, tblAppRiskScore (104) is calculated in cell E15 and creditScore = 610. Given these input values, 1 rule is "hit", rule 2.

Finally, we come to the Routing table in cells G2:L8. This table returns a final strategy based on two decision table results (postBureauRiskCat and postBureauAfford) and two input parameters, bankrupt and creditScore. The PsiDecTable() function in cell L9 = PsiDecTable(G2:L8, , , tblPostBureauRiskCat, postBureauAfford, bankrupt, creditScore) where tblPostBureauRiskCat = medium, postBureauAfford = True (calculated in cell H15), bankrupt = FALSE (never bankrupt) and creditScore = 610.



Routing

P	postBureauRiskCat	postBureauAfford	bankrupt	creditScore	routing
			true, false	[0..999]	decline, refer, accept
1	-	FALSE	-	-	decline
2	-	-	TRUE	-	decline
3	high	-	-	-	refer
4	-	-	-	<580	refer
5	-	-	-	-	accept
					accept

post cont factor

0.7 *example of reusing a table*

post bureau afford

TRUE

The input parameter postBureauAfford is calculated in cell H15 according to the formula, =IF(disposIncome * postCreditContFactor > monthInstall, TRUE, FALSE). Since the product of disposIncome and postCreditContFactor (963.18 * .7) is greater than the monthly installment (556.82), the IF function returns TRUE.

The input parameter postCreditContFactor is calculated in cell H12 = PsiDecTable('Loan Strategy'!L22:N25, , , tblPostBureauRiskCat). Note that this PsiDecTable() function refers to the App Risk Score decision table in cells Loan Strategy!L22:N25. Analytic Solver Comprehensive supports multiple PsiDecTable() functions that refer to the same table, as long as the input variables are different. Multiple PsiDecTable() functions with identical input variables are not supported and will result in an error. Two PsiDecTable() functions return to the App Risk Score decision table.

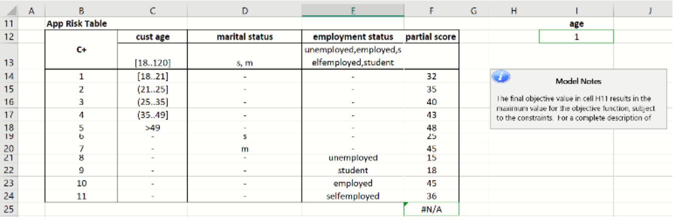
With these criteria, the last rule, rule 5, is hit, accept the loan application.

Using Decision Tables in Optimization and Simulation

It's also possible to build an optimization and simulation model using a decision table as input. See the Data Science User Guide for an example on how to use a decision tree in conjunction with a data science model saved in PMML format.

Optimization

Open DT Opt & Sim Examples.xlsx and click the Optimization tab. The decision table appearing in cells B12:F24 returns the collective score, due to the Hit Policy of "Collect +", of a set of partial scores based on a customer's age, marital status, and employment status. The PsiDecTable() function appearing in cell F25 = PsiDecTable(B12:F24, , , I12, maritalSt, employSt) where I12 currently refers to the age "1", "maritalSt" refers to cell Input!B15 ("s" for single) and "employSt" refers to cell Input!B17 ("selfemployed").



App Risk Table

C+	cust age	marital status	employment status	partial score
	[18..120]	s, m	unemployed,employed,selfemployed,student	
1	[18..21]	-	-	32
2	(21..25]	-	-	35
3	(25..35]	-	-	40
4	(35..49]	-	-	43
5	>49	-	-	48
6	-	s	-	25
7	-	m	-	45
8	-	-	unemployed	15
9	-	-	student	18
10	-	-	employed	45
11	-	-	selfemployed	36
				#N/A

age: 1

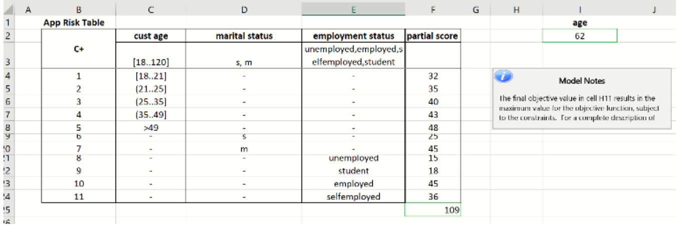
Model Notes

The final objective value in cell H11 results in the maximum value for the objective function, subject to the constraints. For a complete description of

The Model tab on the Solver Task Pane shows the optimization model which is maximizing the PsiDecTable() function in cell F25 by varying cell I12 subject to the variable bounds of 18 and 65. Click Optimize – Analyze Original Model. Analytic Solver Comprehensive will report the model type as "NSP" meaning "nonsmooth problem". The model is nonsmooth due to the use of the decision table. As a result, you will need to either select the Evolutionary Engine to solve the model or set Automatically Select Engine on the Engine tab. Then click the green arrow on the Model tab to solve the optimization model.

Almost immediately, the Evolutionary Engine stops with the result, "Solver has converged to the current solution. All constraints are satisfied." This result means that the Solver has been unable to find a new, better member of the population whose “fitness” represents a relative (percentage) improvement over the current best member’s fitness. For more information on this result, see the "Solver Result Messages" chapter within the Frontline Solvers Reference Guide.

The final variable age results in the largest collective score according to the decision table rules, based on the initial starting point. (For more information on optimizing with the Evolutionary Engine, see the Frontline Solvers Reference Guide.)



App Risk Table

C+	cust age	marital status	employment status	partial score
	[18..120]	s, m	unemployed,employed,selfemployed,student	
1	[18..21]	-	-	32
2	(21..25]	-	-	35
3	(25..35]	-	-	40
4	(35..49]	-	-	43
5	>49	-	-	48
6	-	s	-	25
7	-	m	-	45
8	-	-	unemployed	15
9	-	-	student	18
10	-	-	employed	45
11	-	-	selfemployed	36
				109

age: 62

Model Notes

The final objective value in cell H11 results in the maximum value for the objective function, subject to the constraints. For a complete description of

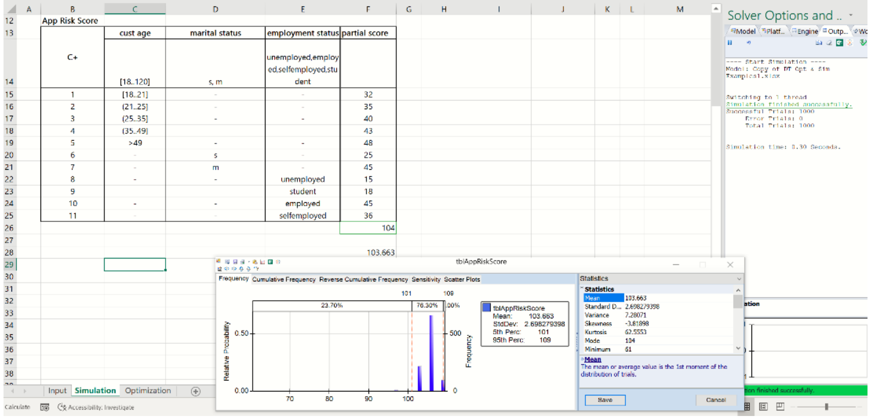
Simulation

Click the Simulation tab to open the simulation example. This example uses the same decision table as in the optimization example. However, in this example, the PsiDecTable() function in cell E26 = PsiDecTable(A13:F25, , custAge, maritalSt, employSt), where "custAge" refers to Input!C11 which holds a PsiNormal() function. This function will pull 1000 values from the PsiNormal() distribution, with a mean of 40 and a standard deviation of 7, during the simulation. The PsiMean() function in cell E28 will summarize the 1000 trial values into one mean value. In addition, the PsiMean function will also denote cell E26, the cell containing the PsiDecTable() function, as an output cell. For simulations, Analytic Solver Comprehensive tracks the values of all uncertain functions (simulation outputs, referenced in PsiOutput() or PSI Statistics function calls).

Click the green arrow on the Model tab to run a simulation. Immediately, the simulation finishes and the uncertain function dialog opens displaying a discrete distribution where we can see that over 70% of the time, the values for the collective sum of the partial scores is over 101 when the age parameter follows a normal distribution with mean 40 and standard deviation 7. For more information on simulation with Analytic Solver Comprehensive, see the Examples: Simulation chapter that appears previously in this guide.

Note: Using a PsiStatistic as an input to the PsiDecTable() is not supported.

For example, given the formula, =PsiDecTable(a1:f10, , , G1, G2) neither G1 nor G2 may contain a PsiStatistic, such as PsiMean, PsiMedian, etc.



App Risk Score

C+	cust age	marital status	employment status	partial score
	[18..120]	s, m	unemployed,employed,selfemployed,student	
1	[18..21]	-	-	32
2	(21..25]	-	-	35
3	(25..35]	-	-	40
4	(35..49]			43
5	>49	-	-	48
6	-	s	-	25
7	-	m	-	45
8	-	-	unemployed	15
9			student	18
10	-	-	employed	45
11	-	-	selfemployed	36
				104
				103.663

tblAppRiskScore

tblAppRiskScore
Mean: 103.663
StdDev: 2.698279398
5th Perc: 101
95th Perc: 109

Statistics

Statistics	
Mean	103.663
Standard D...	2.698279398
Variance	7.28071
Skewness	-3.81898
Kurtosis	62.5553
Mode	104
Minimum	61

Mean
The mean or average value is the 1st moment of the distribution of trials.

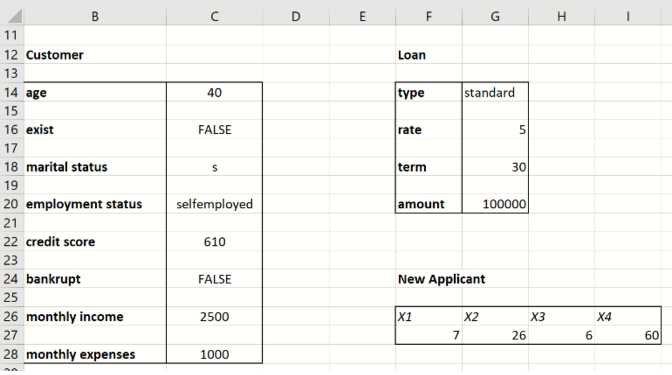
```
---- Start Simulation ----
Model: Copy of DT Opt & Sim
Examples1.xlsx

Switching to 1 thread
Simulation finished successfully.
Successful Trials: 1000
     Error Trials: 0
     Total Trials: 1000

Simulation time: 0.30 Seconds.
```

Using Decision Tables with Data Science

This DT Loan Strategy + Predictive.xlsx example expands on the DT Loan Strategy Model by adding an element to the APP Risk Score table on the Loan tab. The formula in cell Loan!E15 returns the minimum of 2 outputs. The first output is produced from a linear regression model scoring new data contained in cells Input!F26:I27. The second output is from the decision table.

Input Parameters

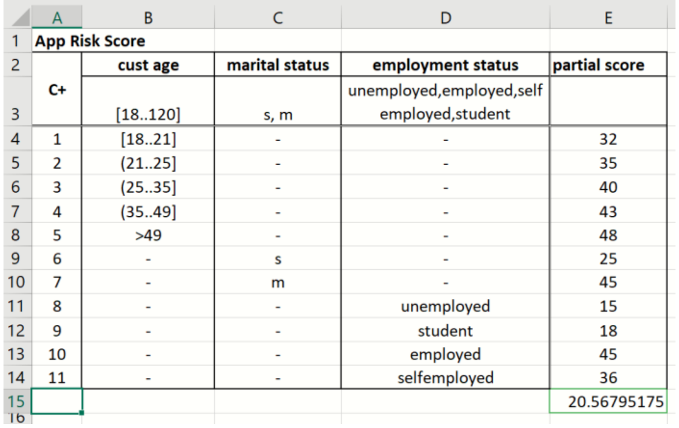
This Input worksheet contains the input parameters to the Loan Strategy model contained on the Loan Strategy and Post Bureau worksheets. Input parameters are all constant values.



	B	C	D	E	F	G	H	I
11								
12	**Customer**				**Loan**			
13								
14	**age**	40			**type**	standard		
15								
16	**exist**	FALSE			**rate**	5		
17								
18	**marital status**	s			**term**	30		
19								
20	**employment status**	selfemployed			**amount**	100000		
21								
22	**credit score**	610						
23								
24	**bankrupt**	FALSE			**New Applicant**			
25								
26	**monthly income**	2500			*X1*	*X2*	*X3*	*X4*
27					7	26	6	60
28	**monthly expenses**	1000						

Loan Strategy

Click the Loan Strategy worksheet. Like in the DT Loan Strategy Model.xlsx workbook, this worksheet contains six cascading decision tables. The first table appears in cells A2:E14. This table returns a partial "risk of default" score based on a loan applicant's age, marital and employment status. Higher scores are given to older, employed, married applicants. Cell E15 contains the PsiDecTable() function.



	A	B	C	D	E
1	**App Risk Score**				
2		**cust age**	**marital status**	**employment status**	**partial score**
3	**C+**	[18..120]	s, m	unemployed,employed,self employed,student	
4	1	[18..21]	-	-	32
5	2	(21..25]	-	-	35
6	3	(25..35]	-	-	40
7	4	(35..49]	-	-	43
8	5	>49	-	-	48
9	6	-	s	-	25
10	7	-	m	-	45
11	8	-	-	unemployed	15
12	9	-	-	student	18
13	10	-	-	employed	45
14	11	-	-	selfemployed	36
15					20.56795175

In this workbook, however, the formula in cell E15, has an added component. In this example, the Excel MIN function is used to return the minimum of a predicted value obtained from a linear regression model made to score on an applicant's data (see cells Input!F26:I28) and the decision table result given the applicant's name, marital status, and employment status.

=MIN(PsiPredict(Predictive_PMML_Model!B12:B44,Input!E15:H16),PsiDecTable(A2:E14, , , custAge, maritalSt, employSt))

The input parameters for the first term in the MIN function (PsiPredict(Predictive_PMML_Model!B12:B44,Input!E15:H16) are the linear regression model saved in PMML format on the Predict_PMML_Model tab in cells B12:B44 and the new data saved on the Input tab in cells E15:H16. The result from PsiPredict() is 20. 57 For a complete explanation of the PsiPredict function, see the Data Science Guide.

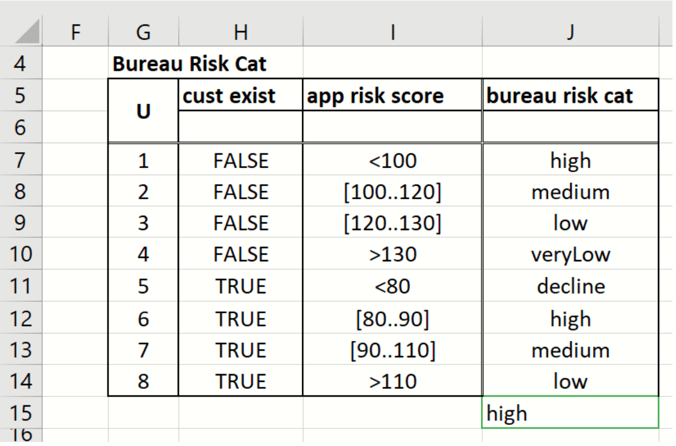
The input parameters from the Input worksheet are: custAge="40", maritalSt = "s", employSt = "selfemployed". The applicable rules for these input parameters are: 4, 6, and 11 resulting in partial scores of 43, 25, and 36, respectively. Since the Hit Policy is "C+" the result collection is added together, 43 + 25, + 36 =

104. As shown in the DT Loan Strategy Model, the output from the App Risk Score table is 104.

The minimum value returned is of course 20.57, or the output from the PsiPredict() function.

The remaining decision tables are calculated as in the DT Loan Strategy Model.

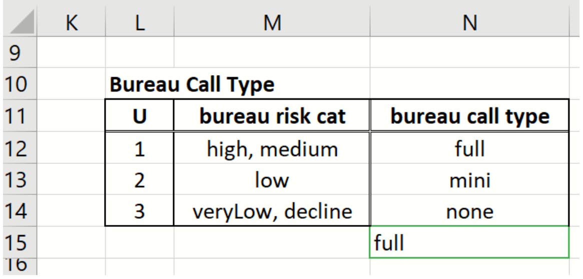
The next decision table is located in cells G5:J14, the Bureau Risk Cat table. This table returns a risk category of high, medium, or low based on their customer status (existing or not) and their application risk score. Non-existing customers are allowed a higher risk score. Cell J15 contains the PsiDecTable() function



	F	G	H	I	J
4		**Bureau Risk Cat**			
5		**U**	**cust exist**	**app risk score**	**bureau risk cat**
6					
7		1	FALSE	<100	high
8		2	FALSE	[100..120]	medium
9		3	FALSE	[120..130]	low
10		4	FALSE	>130	veryLow
11		5	TRUE	<80	decline
12		6	TRUE	[80..90]	high
13		7	TRUE	[90..110]	medium
14		8	TRUE	>110	low
15					high
16					

J15 =PsiDecTable(G5:J14, , , custExist, tblAppRiskScore) where custExist = False and tblAppRiskScore() refers to the value in cell E15, or 20.57.   Based on these conditions, one rule is "hit", rule 2. This rule returns "high" for the bureau risk category. Since the hit policy is equal to U (Unique) only one result is returned, even if multiple results are returned.

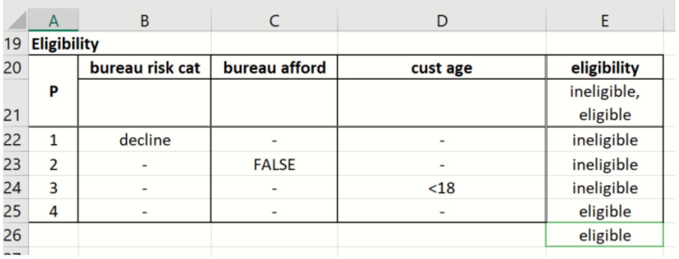
The next decision table, Bureau Call Type, is located in cells L11:N14. This table returns the bureau call type based on the bureau risk category found in the previous decision table, Bureau Risk Cat. Cell N15 contains the PsiDecTable() function.



	K	L	M	N
9				
10		**Bureau Call Type**		
11		**U**	**bureau risk cat**	**bureau call type**
12		1	high, medium	full
13		2	low	mini
14		3	veryLow, decline	none
15				full
16				

N15=PsiDecTable(L11:N14, , , tblBureauRiskCat) where tblBureauRiskCat is equal to the result of the PsiDecTable() function in cell J15, or "medium". Only 1 rule is "hit", rule 1, which returns "full" This means that a full evaluation by the bureau must be performed.

The next decision table, Eligibility, is located in cells A20:E25. This table returns the applicant's eligibility based on three criteria: the bureau risk category, from cell J15, bureau afford, calculated in cell C36, and customer age, located in cell Input!C13. Since the Hit Policy is equal to P (Priority), the result collection will be returned in the order listed according to the list in cell E21. The PsiDecTable() function is located in cell E26.



	A	B	C	D	E
19	**Eligibility**				
20		**bureau risk cat**	**bureau afford**	**cust age**	**eligibility**
21	**P**				ineligible, eligible
22	1	decline	-	-	ineligible
23	2	-	FALSE	-	ineligible
24	3	-	-	<18	ineligible
25	4	-	-	-	eligible
26					eligible

E26 = PsiDecTable(A20:E25, , , tblBureauRiskCat, bureauAfford, custAge)

Intermediate Formulas

Cell C36 calculates the bureau afford statistic, =IF(disposIncome * tblCreditContFactor > monthInstall, TRUE, FALSE)

where disposIncome is calculated in cell C34, tblCreditContFactor is calculated in cell N26 (more on this later) and monthInstall is calculated in cell C32.

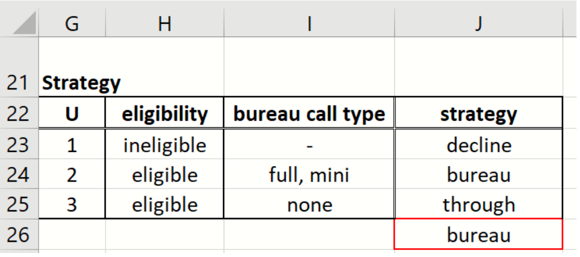
Cell 34 calculates an applicant's disposable income using the formula, =monthIncome **-** (monthRepay + monthExpenses) where monthIncome = 2500, monthRepay is calculated in cell C30 and monthExpenses = 1000.

Cell C30 calculates the monthly loan replayment amount using the formula, **-**PMT(loanRate%/12, loanTerm*12, loanAmnt) where loanRate is 5%, loanTerm = 30 and loanAmnt = 100,000.

Finally, cell C28 calculates the monthly fee using the IF statement, = IF(loanType = "standard", 20, IF(loanType = "special", 25, 0)), where loanType = standard.

Using the final input values of tblBureauRiskCat = high, bureauAfford = True, and custAge = 40, the last rule, rule 4, is the only rule "hit", returning "eligible".

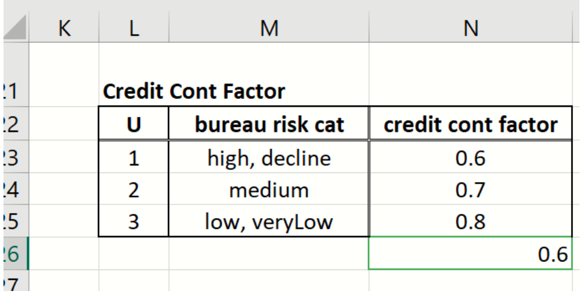
The next decision table is located in cells G22:J25. This table returns the loan strategy based on the applicant's eligibility and bureau call type. The PsiDecTable() function is located in cell J26.



	G	H	I	J
21	**Strategy**			
22	**U**	**eligibility**	**bureau call type**	**strategy**
23	1	ineligible	-	decline
24	2	eligible	full, mini	bureau
25	3	eligible	none	through
26				bureau

J26=PsiDecTable(G22:J25, , , tblEligibility, tblBureauCallType) where tblEligibility is calculated in cell E26 and tblBureauCallType is calculated in cell N15. Both of these inputs are outputs of earlier decision tables. Based on the results calculated by the previous decision tables, this PsiDecTable() function results in a "bureau" strategy.

Finally, the last decision table, Credit Cont Factor, is contained in cells L22:N25. The PsiDecTable() function is located in cell N26.



Credit Cont Factor

U	bureau risk cat	credit cont factor
1	high, decline	0.6
2	medium	0.7
3	low, veryLow	0.8
		0.6

N26 =PsiDecTable(L22:N25, , , tblBureauRiskCat) where tblBureauRiskCat is calculated in cell J15. Again, this input is an output from an earlier discussed decision table.

If the result of the PsiDecTable function in cell J26 would have been either "decline" or "through", then our work would be finished. We would either decline or accept the applicant, respectively. However, since "bureau" was returned, more analysis is required.

More Information on Decision Tables

Supported **conversion** functions are listed below.

- **date**(string *date*) returns a date serial number, the same as Excel DATEVALUE
- **date**(number *y*, number *m*, number *d*)
- **time**(string *time*) returns a time serial number, the same as Excel TIMEVALUE
- **time**(number *h*, number *m*, number *s*[, number *offset*]) the optional *offset* is duration in seconds, which can be used to model UTC
- **duration**(string *dur*) returns duration in months or seconds depending on format
- **yearsAndMonthsDuration**(string *from_date*, string *to_date*) return difference between two dates as a duration in months
- **number**(string num, string group_sep, string dec_sep) returns a number from a string
- **string**(float *num*) returns a float number as a string
- Supported **numeric** functions are:
- **ceiling**(number *num*) rounds up a number
- **decimal**(number *num*, number *decimals*) rounds a number to the given number of decimals
- **floor**(number *num*) rounds down a number
- Supported **string** functions:
- **substring**(string *str*, number *pos*, number num_*chars*)
- **stringLength**(string *str*)
- **upperCase**(string *str*)
- **lowerCase**(string *str*)

- **substringBefore**(string *str*, string *match*)
- **substringAfter**(string *str*, string *match*)
- **contains**(string *str*, string *match*)
- **startsWith**(string *str*, string *match*)
- **endsWith**(string *str*, string *match*)
- Supported **list** functions are:
- **min**(number n1, number n2,…) returns the minimum number
- **max**(number n1, number n2,…)
- **sum**(number n1, number n2,…)
- **mean**(number n1, number n2,…)
- **and**(bool b1, bool b2,…)
- **or**(bool b1, bool b2,…)

Creating Power BI Custom Visuals

In past versions of Analytic Solver and still in Version 20.5, it's possible to turn an Excel-based optimization or simulation model into a **Microsoft Power BI Custom Visual**. Now with **Version 20.5**, you can actually *embed* your optimization or simulation model into the Custom Visual which will accept the data your model needs from Power BI. Whenever this data changes, the Custom Visual will send your model, plus the data, back to Frontline's Cloud Service where it will re-solve the model, obtain the results and will display them in chart form in Power BI.

Where others must learn JavaScript (or TypeScript) programming and a whole set of Web development tools to even begin to create a Custom Visual in Power BI, after reading this section, Analytic Solver users will be able to create one right away.

In either case, users simply select rows or columns of data to serve as changeable parameters, then choose **Deploy Model – Power BI**, and save the file created by Analytic Solver. Afterwards, users click the Load Custom Visual icon in Power BI and select the file just saved. What is produced isn’t just a chart – it’s a *full optimization or simulation model*, ready to accept Power BI data**, run on demand** on the web, and display visual results in Power BI. Users simply need to drag and drop appropriate datasets into the “well” of inputs in Power BI to match the model parameters.

Analytic Solver translates the Excel model into **RASON**® (RESTful Analytic Solver Object Notation, embedded in JSON), then “wraps” a JavaScript-based Custom Visual around the RASON model. See the previous chapter “Creating Your Own Application” for more information on RASON.

Notes:

- Analytic Solver Upgrade and Solver Academy licenses do not support Deploy Model functionality. This button will be disabled on the Analytic Solver ribbon.

Installing Power BI

Power BI is Microsoft’s desktop or cloud-based interactive data visualization business intelligence tool. In past versions of Analytic Solver, users have only been able to upload their model parameters (model data) to Power BI after solving an optimization or simulation model. Starting with V2017-R2, Analytic Solver can turn your Excel based optimization or simulation model into a Microsoft Power BI Custom Visual, *that is your full optimization or simulation model*, ready to accept power BI data, run on demand on the web, and display visual results in Power BI.

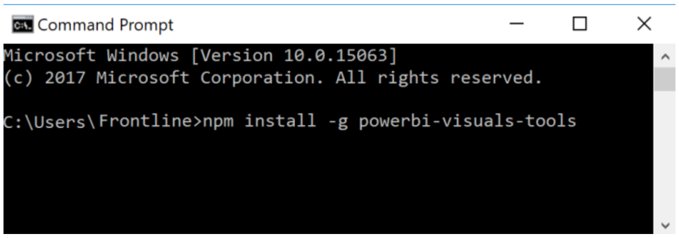
Note: Click this link to download Power BI for desktop.

In order to use this new feature in Analytic Solver, you must first open desktop Power BI or the free cloud based version of Power BI. (Power BI is part of the office 365 suite.) For more information on this business tool, see the following website: https://powerbi.microsoft.com/en-us/documentation/powerbi-custom-visuals-getting-started-with-developer-tools/

It is important to note, that in order to create a custom visual in Power BI, you will first need to install Power BI's Developer Tools which is comprised of NodeJS and the Power BI tools. Follow the steps below to install both required items.

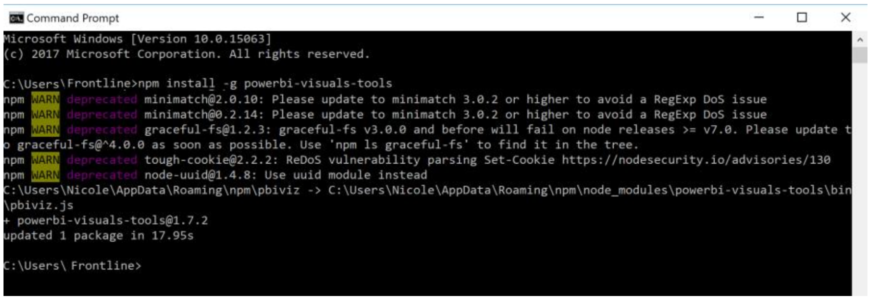
5. Click here to download NodeJS.
6. Click the Download button to download NodeJS for 64-bit Windows.
7. Once the NodeJS installer is downloaded, run the installer and follow the directions on the installer dialogs to install NodeJS onto your machine.
8. Install the command line tools by opening a command prompt and typing: "**npm install -g powerbi-visuals-tools@3.2.3**" as shown in the screenshot below.

   Note: If using V2017-R2, type: "npm install -g powerbi-visuals-tools@1.5".



```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Frontline>npm install -g powerbi-visuals-tools
```

After clicking Enter, you'll see the following results in the command prompt window.

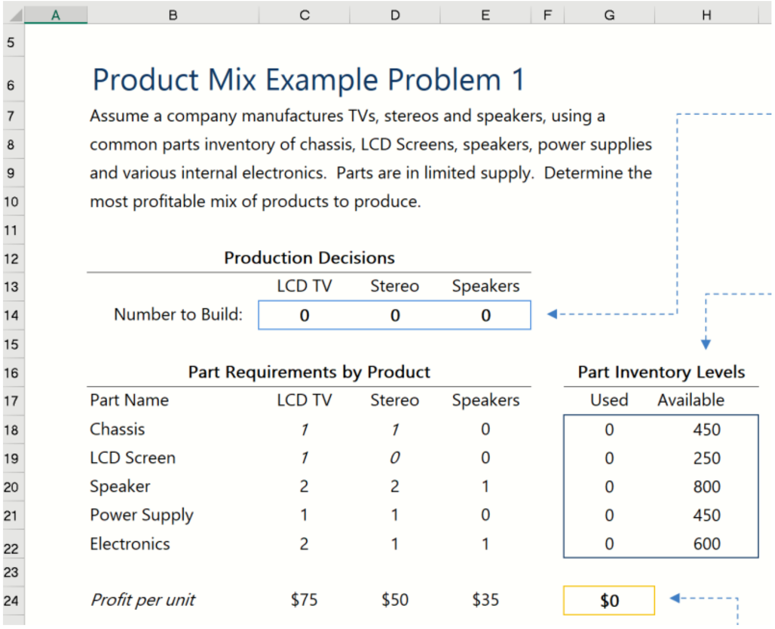


```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Frontline>npm install -g powerbi-visuals-tools
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated minimatch@0.2.14: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated graceful-fs@1.2.3: graceful-fs v3.0.0 and before will fail on node releases >= v7.0. Please update t
o graceful-fs@^4.0.0 as soon as possible. Use 'npm ls graceful-fs' to find it in the tree.
npm WARN deprecated tough-cookie@2.2.2: ReDoS vulnerability parsing Set-Cookie https://nodesecurity.io/advisories/130
npm WARN deprecated node-uuid@1.4.8: Use uuid module instead
C:\Users\Nicole\AppData\Roaming\npm\pbiviz -> C:\Users\Nicole\AppData\Roaming\npm\node_modules\powerbi-visuals-tools\bin
\pbiviz.js
+ powerbi-visuals-tools@1.7.2
updated 1 package in 17.95s

C:\Users\ Frontline>
```

Creating a Custom Visual from an Optimization Model

Now that the Developer Tools are installed, we can move on to a small optimization example. In this section, we will create a custom visual using the Product Mix example model.

Open the ProductMix(Opt).xlsx model that was previously discussed in the Conventional Optimization chapter by clicking Help – Example Models on the ribbon, clicking Optimization Examples and then the Product Mix link. Recall this example model determines the optimal mix of products that a company should produce in order to maximize profits.



Product Mix Example Problem 1

Assume a company manufactures TVs, stereos and speakers, using a common parts inventory of chassis, LCD Screens, speakers, power supplies and various internal electronics. Parts are in limited supply. Determine the most profitable mix of products to produce.

Production Decisions

	LCD TV	Stereo	Speakers
Number to Build:	0	0	0

Part Requirements by Product				Part Inventory Levels	
Part Name	LCD TV	Stereo	Speakers	Used	Available
Chassis	1	1	0	0	450
LCD Screen	1	0	0	0	250
Speaker	2	2	1	0	800
Power Supply	1	1	0	0	450
Electronics	2	1	1	0	600
Profit per unit	$75	$50	$35	$0	

The optimization model is included in the Model tab in the Solver Task Pane. This model written in algebraic form is below.

Maximize: Total Profit = $75x1 + $50x2 + $35x3

Subject To:

Chassis: 1x1 + 1x2 + 0x3 <= 450

LCD Screen: 1x1 + 0x2 + 0x3 <= 250

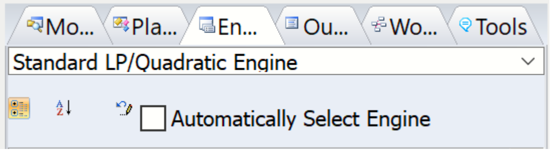
Speaker: 2x1 + 2x2 + 1x3 <= 800

Power Supply: 1x1 + 1x2 + 0x3 <= 450

Electronics: 2x1 + 1x2 + 1x3 <= 600

x1, x2, x3 > 0

The engine selected to solve this model is the Standard LP/Quadratic Engine as shown on the Engine tab in the Solver Task Pane.

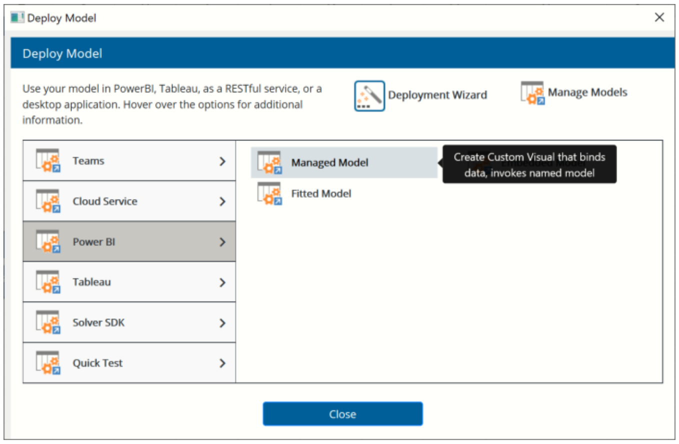


Standard LP/Quadratic Engine

☐ Automatically Select Engine

Click Deploy Model on the ribbon, then select Power BI – Managed Model. Recall that in order to use the Managed Model choice, your model must have already been posted to the Rason Cloud Service using Deploy Model – Cloud Service – RASON Model.

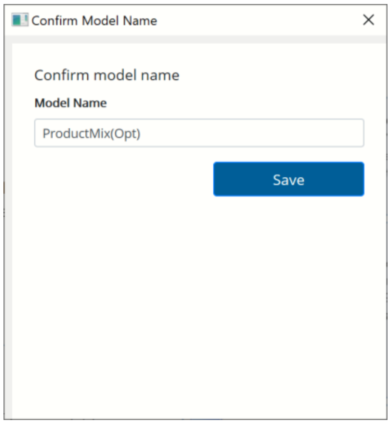
Note that it's also possible to embed your model into Power BI using Power BI – Embedded Model, however, if your model will change in any way, it's recommended to use Managed Model. For more formation on creating an embedded model, see the previous Deploying Your Model chapter.



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Create Custom Visual that binds data, invokes named model

Click Save to confirm the model name as "ProductMix(Opt)"



Model Name: ProductMix(Opt)

Model Versions

To create a new version of the model, enter a new name such as "ProductMix(Opt)1". You can manage your model versions by clicking the Manage Models icon on the Deploy Model dialog. For more information on this dialog, see the previous Deploying Your Model chapter

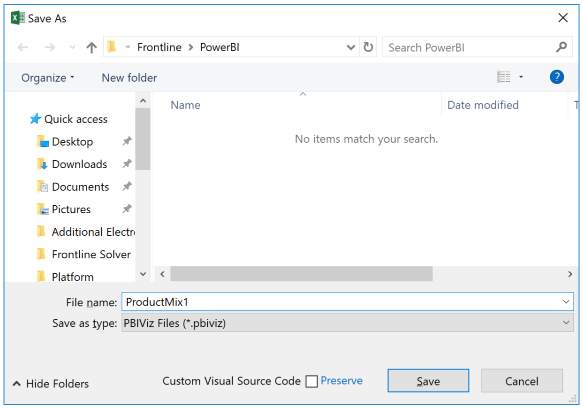
There is no need to solve the model first unless a change has been made to an already existing RASON model. If a change has been made to the workbook after a RASON model has been created, then you MUST either solve or diagnose the Excel model in order for the changes to be reflected in the new RASON model.

When creating a Power BI custom visual through desktop Analytic Solver, model elements (objective function and all variables and constraints) may appear across multiple worksheets within the same Excel workbook.

Immediately, the Save As dialog appears. Enter a file name in the location of your choice and then click Save.

- The screenshot below displays a file name of “ProductMix1” being saved into the Power BI folder, in Analytic Solver Desktop.

Analytic Solver Desktop


Save As

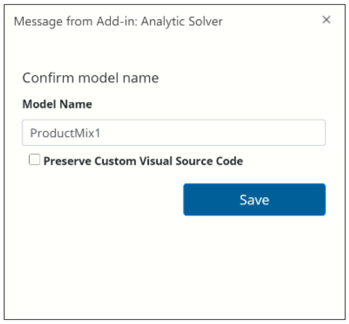
File name: ProductMix1

Save as type: PBIViz Files (*.pbiviz)

No items match your search.

Custom Visual Source Code ☐ Preserve

- In Analytic Solver Cloud, you'll first receive a dialog where you'll name the file to be exported.


Message from Add-in: Analytic Solver

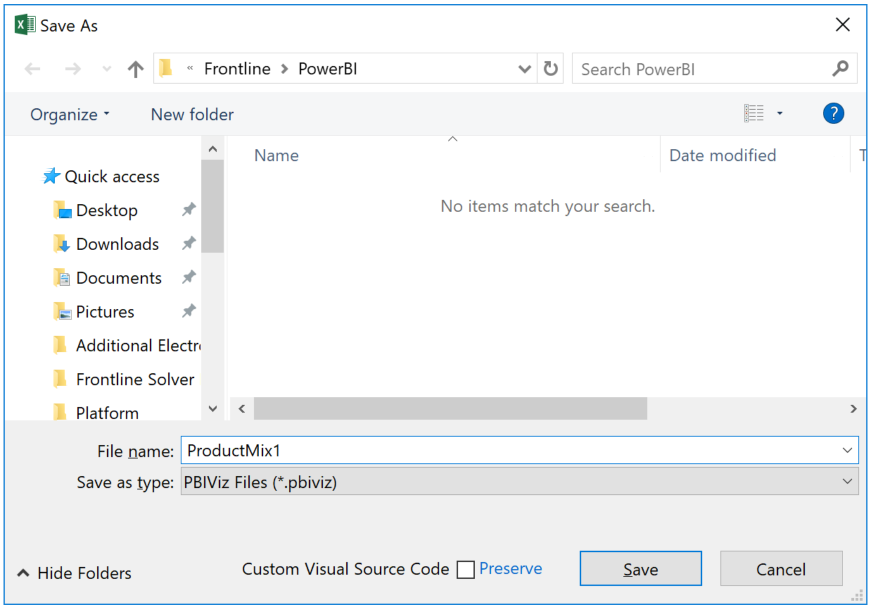
Confirm model name

Model Name

ProductMix1

☐ **Preserve Custom Visual Source Code**

After you click Save and the file downloads, you'll be asked if you'd like to open or save the file. Choose "Save As" to open the Save As dialog and browse to where you'd like to save the Power BI export.


Save As

File name: ProductMix1

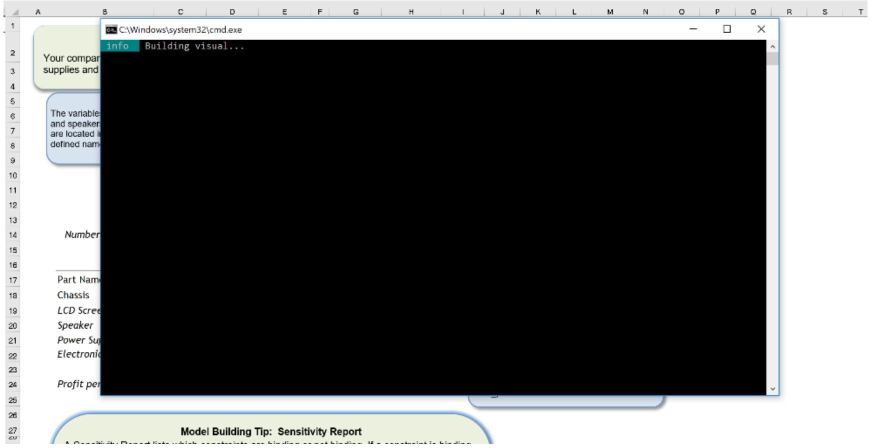
Save as type: PBIViz Files (*.pbiviz)

No items match your search.

Custom Visual Source Code ☐ Preserve

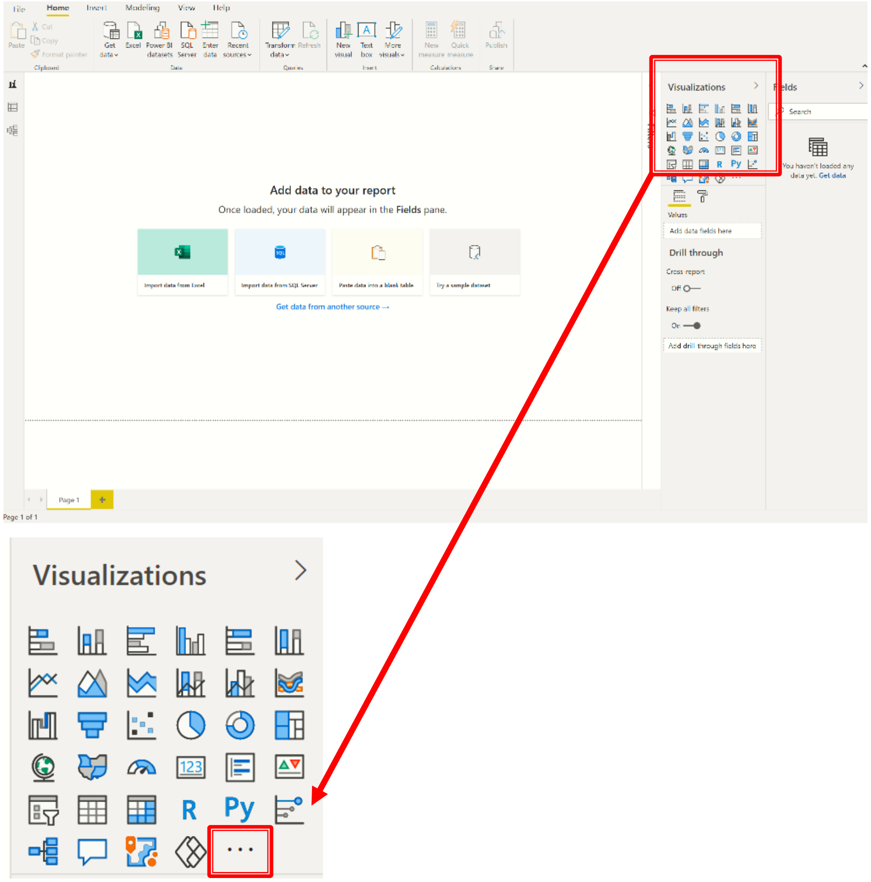
Note the Custom Visual Source Code checkbox at the bottom of the dialog. This feature will be explained a little later in this chapter.

Save ProductMix1.pbiviz. Two command prompt windows will appear on your screen. After several moments, both will disappear. It is during this time that your Excel model is being translated into the **RASON**® modeling language, and your Power BI custom visual is created.

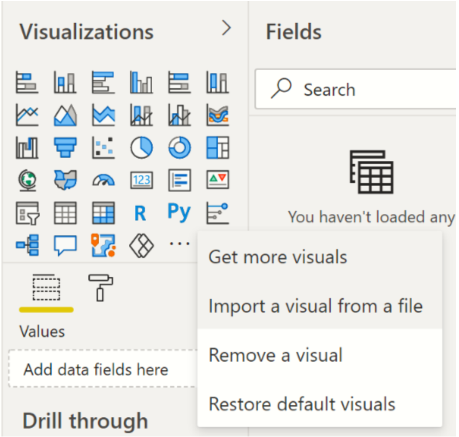


```
info  Building visual...
```

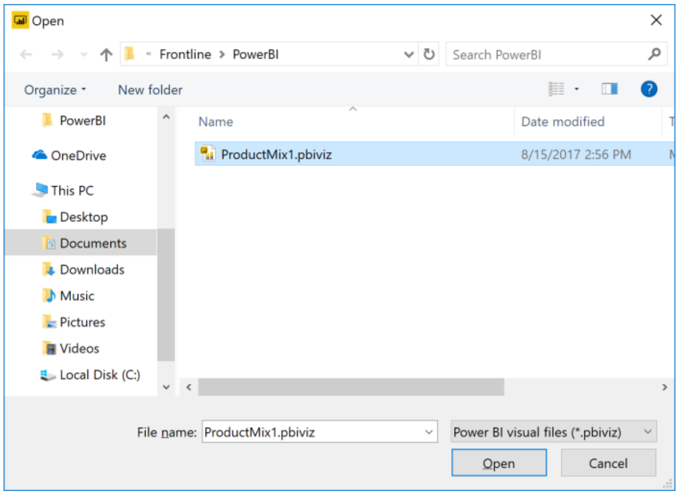
Open either desktop or cloud-based Power BI. The screenshot below depicts the opening screen of desktop Power BI. Click the icon containing three horizontal dots that appears at the bottom of Visualizations (in the top right-hand corner).



Then select *Import a visual from a file* from the menu.

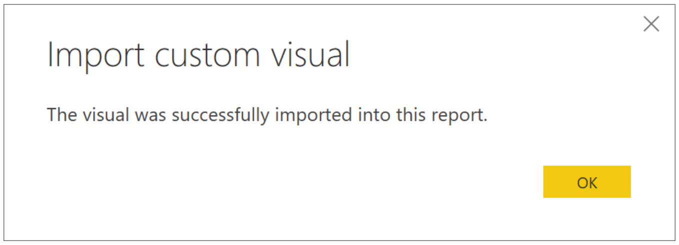


Navigate to the location of the saved ProductMix1.pbiviz file.



File name: ProductMix1.pbiviz

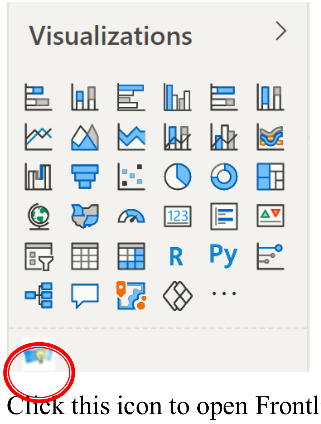
Then click Open. If the import was successful, you will see a message indicating as such. Click OK to clear this dialog.



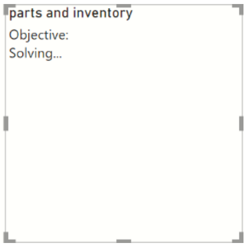
Import custom visual

The visual was successfully imported into this report.

A new icon, bearing the Frontline Solvers logo is added under Visualizations.

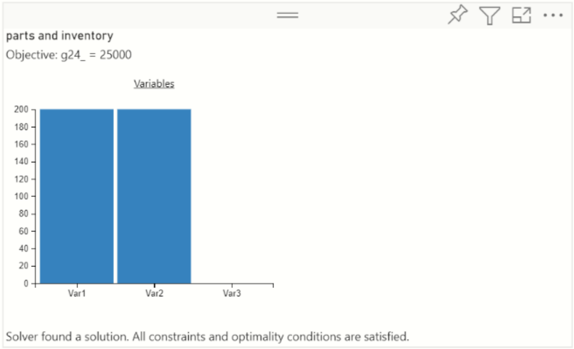


Click this icon to open Frontline Solvers' custom visual. If you glance at the custom visual, you'll notice "Solving…" is displayed beneath "Objective" which indicates that the model has been submitted to the RASON server. If the model is an optimization or stochastic optimization model, the RASON server will use the engine selected in the Excel model to solve the model. Recall that the LP/Quadratic engine was selected in the Product Mix example model, above. (If the model is a simulation model, Frontline Systems' Risk Solver engine will be used.)



parts and inventory
Objective:
Solving...

After the model is solved, values for the objective function, variables and constraints, or uncertain variables and uncertain functions, are imported back into Power BI and the results are displayed in the Power BI window. After enlarging the custom visual, your screen should appear similar to the screenshot below.



parts and inventory
Objective: g24_ = 25000

Solver found a solution. All constraints and optimality conditions are satisfied.

At the bottom of the custom visual, we find Solver's result message: Solver found a solution. All constraints and optimality conditions are satisfied. Recall that since we are solving a linear model, this message indicates that Solver has found the globally optimal solution: There is no other solution satisfying the constraints that has a better value for the objective. For more information on this Solver result, please see the chapter "Solver Results Messages" within the *Frontline Solvers Reference Guide.*

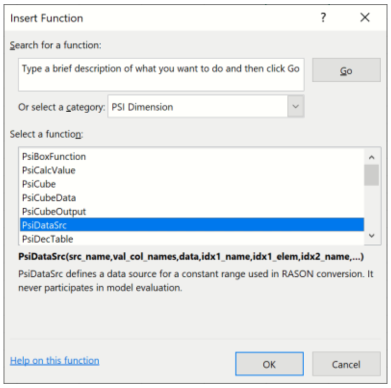
In the first chart, we see that final variable values are: Var1 (LCD TV) equal to 200, Var2 (Stereo) equal to 200 and Var3 (Speakers) equal to 0. These variable values result in an objective function value equal to $25,000.

Solving this model in Power BI could be useful and the chart is certainly informative. However, what if we wanted to change the inventory? What if Purchasing was able to obtain a discount on Speakers and our inventory increased from 800 to 1,000? In order to solve the new model with the updated inventory parameters, we would have to go back to the Excel model, update the chassis inventory, save the model, create a new .pbiviz file, Import the new .pbiviz file into Power BI and then finally open the new custom visual. That's a lot of effort to change just one model parameter!

Thankfully, there is an easier way. By using the Psi function, PsiDataSrc(), we are able to specify the parameters in the Excel model that could change in the future, for example, the amount of inventory on hand for each part, the profit point for each product or the number of parts that makeup each product.

Click back to the Excel model and select a blank cell, say J6. In Analytic Solver Desktop, click Formulas – Insert Function to open the Insert Function dialog, then select PSI Dimension for Category. Afterwards, highlight the function PsiDataSrc, and click OK.

Note: It is suggested that you delete the Constraints text box in order to create room for the added PsiDataSrc() functions, described below.



Or select a category: PSI Dimension

PsiDataSrc(src_name,val_col_names,data,idx1_name,idx1_elem,idx2_name,...)

PsiDataSrc defines a data source for a constant range used in RASON conversion. It never participates in model evaluation.

In Analytic Solver Desktop, you can simply type the formula directly into the cell.

The Function Arguments dialog opens displaying 5 arguments: Src_name, Val_col_names, Data, Idx1_name, and Idx1_elem. We will use this dialog to create a PsiDataSrc() in cell J6 that will allow us the ability to change the inventory values and resolve the Product Mix example model not in Excel, but in Power BI.

PsiDataSrc() Arguments

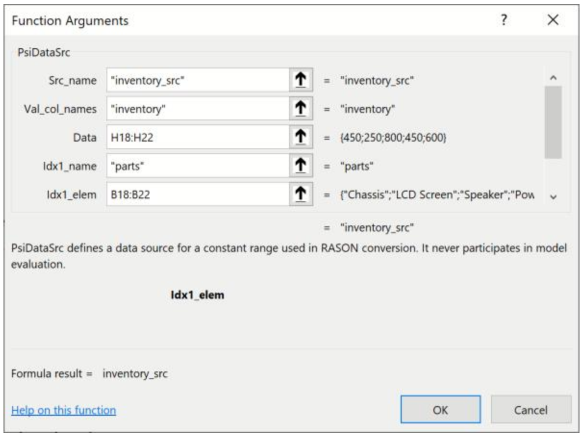
Src_name – This argument names the text file, created by the custom visual, which is submitted, along with the RASON model, to the RASON server. For this example, type: inventory_src.

Val_col_names – This is the name given to the "data well" that will appear in Power BI. This "data well" will hold the actual data. For this example, type: inventory.

Data – Enter the Excel range where the data is located on the spreadsheet for this argument. For this example, use your mouse to click the upward pointing arrow to the right of the Data argument and select cells H18:H22. You can also simply type this range into the edit box.

Idx1_name – Enter the name of the (1st) index set for the data specified in the Data argument. In this example, our Index set is simply the inventory available for each Part. Therefore, enter “parts” for this argument.

Idx1_elem – Enter the Excel range containing the data (or elements) for the first index set. For this example, either type or select cells: B18:B22



Src_name "inventory_src" = "inventory_src"
Val_col_names "inventory" = "inventory"
Data H18:H22 = {450;250;800;450;600}
Idx1_name "parts" = "parts"
Idx1_elem B18:B22 = {"Chassis";"LCD Screen";"Speaker";"Pow
= "inventory_src"

PsiDataSrc defines a data source for a constant range used in RASON conversion. It never participates in model evaluation.

Idx1_elem

Formula result = inventory_src

Since this index set is 1-dimensional, nothing further is needed and we can click OK to enter the PsiDataSrc() function into cell J6.

In Analytic Solver Cloud, you can type the formula directly into cell J6.

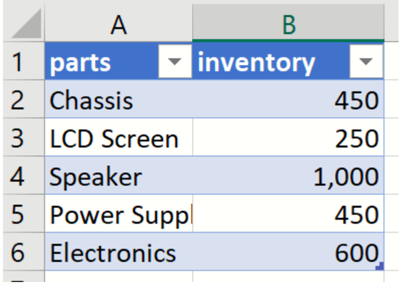
```
=PsiDataSrc("inventory_src", "inventory", H18:H22,
"parts", B18:B22)
```

Afterwards, cell J6 will display the value given to the Src_name argument, inventory_src. Save the Excel model and then click Deploy Model – Power BI – Managed Model to save a new file using a new name such as ProductMix2.pbiviz.

In order to provide our new data to Power BI, we must create a data file. This data file, containing the new inventory levels by part, will be imported into Power BI. This data file can be of any format supported by Power BI such as a CSV file, a text file, a database, etc. In this example, we will use an Excel workbook to hold the data. For more information on what types of files Power BI supports, please see the Power BI documentation at: https://powerbi.microsoft.com/en-us/documentation/powerbi-landing-page/

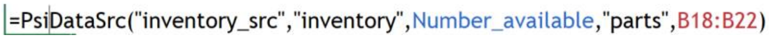
Open a blank worksheet and create a table with two columns. List the parts by name in the first column using the column heading “parts” and the inventory levels in the second column using the column heading “inventory”, as shown in the screenshot below. Make sure to pass the new inventory value of 1,000 Speakers.

Click Help – Example Models – Optimization Examples – Product Mix Data for Power BI-Tableau to open the complete data file. This table is included on the Parts Inventory tab.



	A	B
1	parts	inventory
2	Chassis	450
3	LCD Screen	250
4	Speaker	1,000
5	Power Suppl	450
6	Electronics	600

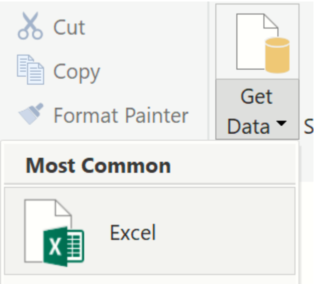
Note: Column headings ***must be identical*** *to the names of the Index Sets passed to the PsiDataSrc function. Make sure there are no spaces before or after column headings or row elements.*



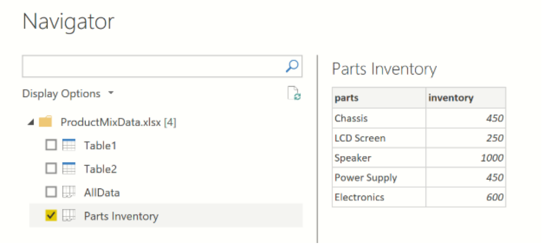
```
=PsiDataSrc("inventory_src","inventory",Number_available,"parts",B18:B22)
```

Recall that we labeled the IndexSet as “parts” (4th argument of PsiDataSrc) and Val_col_names as "inventory" (2nd argument of PsiDataSrc) so the title for column A must be "parts" and the title for column B must be "inventory". Now save the workbook to a desired location.

Open either desktop or cloud-based Power BI and click Get Data – Excel.



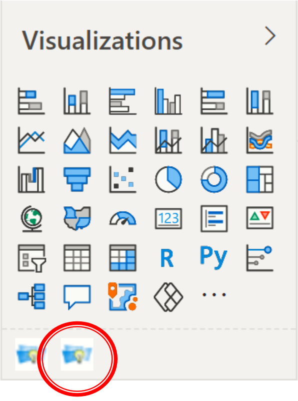
Browse to the location of the saved data file and click Open. Select the worksheet that contains your newly created data table and click Load.



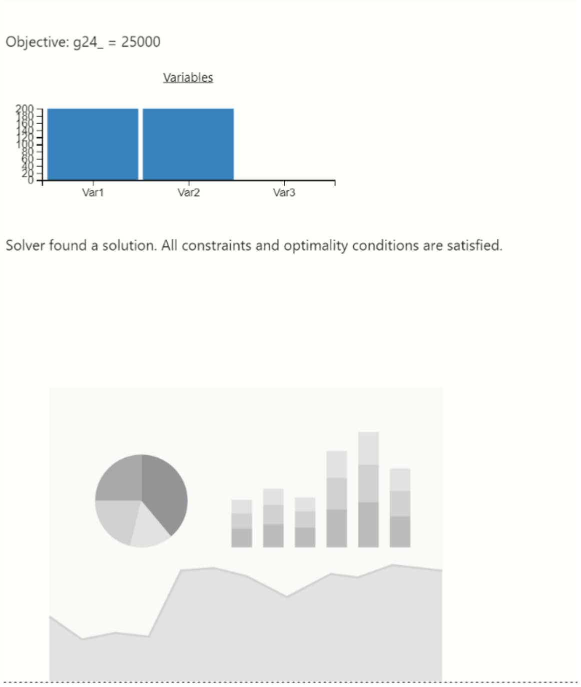
Parts Inventory

parts	inventory
Chassis	450
LCD Screen	250
Speaker	1000
Power Supply	450
Electronics	600

After the data table is uploaded to Power BI, click the icon containing three horizontal dots at the bottom of Visualizations, then select *Import from file* from the menu. Navigate to the location of the saved ProductMix2.pbiviz file. Select the file, then click Open. If the import was successful, you will see a message indicating as such. Click OK to clear this dialog. A new second icon, bearing the Frontline Solvers logo is added under Visualizations.



Click this icon to open Frontline Solvers' custom visual. Your Power BI screen should appear similar to the screenshot below.

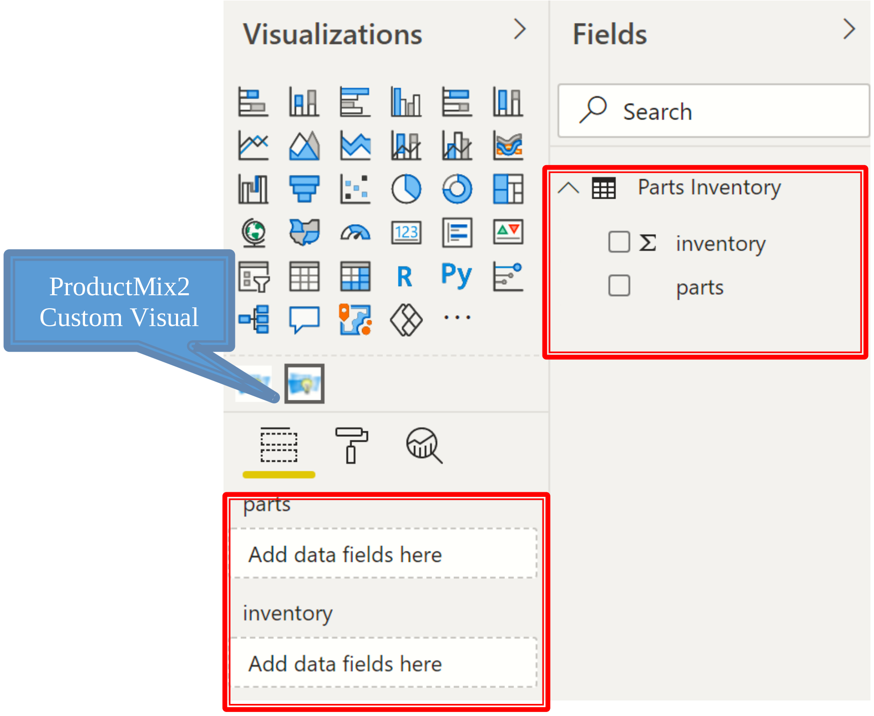


Objective: g24_ = 25000

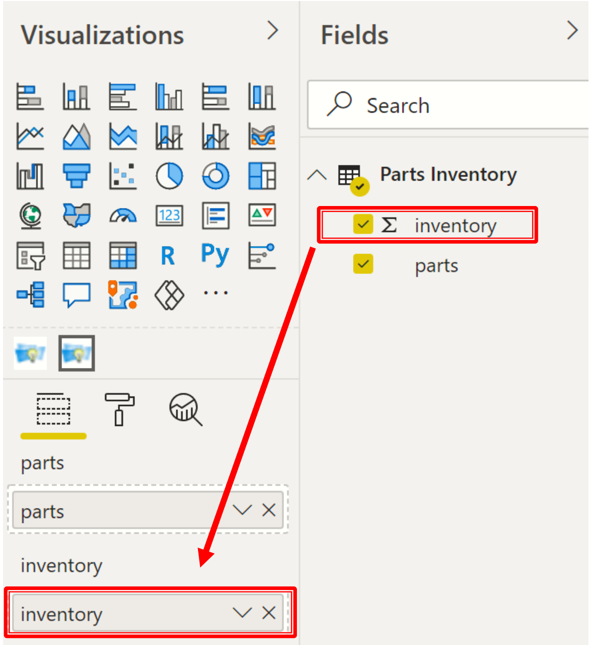
Solver found a solution. All constraints and optimality conditions are satisfied.

Click the custom visual, the black and white chart. At this point you should see newly added items such as "parts" and "inventory" under Visualizations, on the task pane on the right, and the same under Fields. Recall that these are the same names that we passed to the PsiDataSrc() function within the Excel example model, ProductMix(Opt).xlsx.

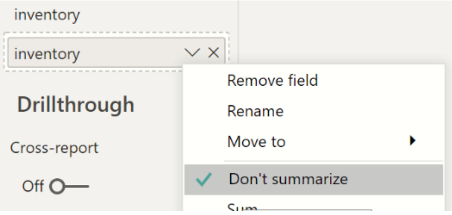
Two fields are added under Parts Inventory in the Fields section of the Task Pane: inventory and parts.



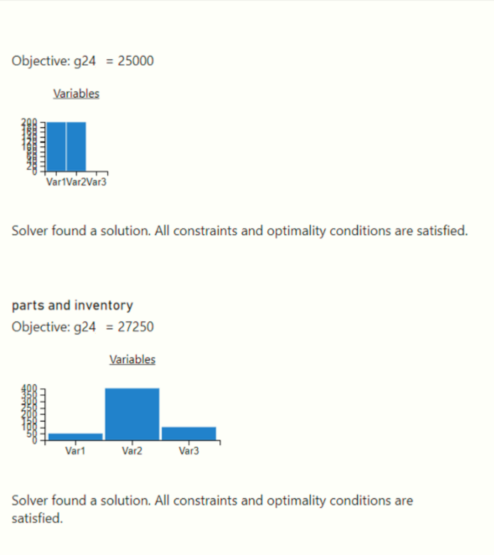
Click "parts" Under Fields and move to "parts" under Visualizations where the words “Drag data fields here” is located. Now drag "inventory" on the right to "inventory" on the left.



Click the down arrow next to both parts and inventory and select Don't Summarize from the menu.



Immediately, you should see the "Solving…" displayed on the custom visual while the model is sent to the RASON server, solved, and the final values are returned. After enlarging the custom visual, your screen should appear similar to the screenshot below.



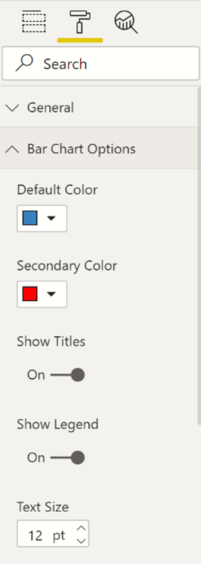
Objective: g24 = 25000

Solver found a solution. All constraints and optimality conditions are satisfied.

parts and inventory
Objective: g24 = 27250

Solver found a solution. All constraints and optimality conditions are satisfied.

In the latest version of Analytic Solver, the generated Power BI visual can be further customized by the user by clicking the Format icon (Paint roller icon). From this menu, chart titles, chart colors, etc can be changed to fine tune the chart to the user's specifications.



Default Color 🟦

Secondary Color 🟥

Show Titles On

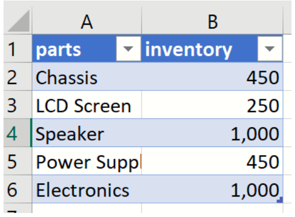
Show Legend On

Text Size 12 pt

At the bottom of the custom visual, we find Solver’s result message: Solver found a solution. All constraints and optimality conditions are satisfied. In the

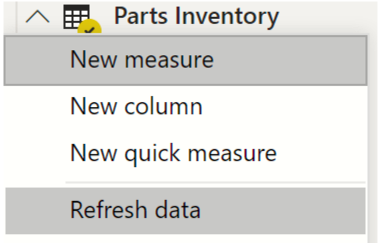
Variables chart, we see that final variable values are: Var1 (LCD TV) equal to 50, Var2 (Stereo) equal to 400 and Var3 (Speakers) equal to 100. These variable values result in an objective function value equal to 27,250. The payoff of increasing the available speakers from 800 to 1,000 is an added profit of $2,250 ($27,250 - $25,000).

Now, if you wanted to change another inventory level, let's say you were able to obtain 400 more electronic components, for a total of 1000 electronic components in inventory. You would simply update the number of electronic components in ProductMixData.xlsx,…

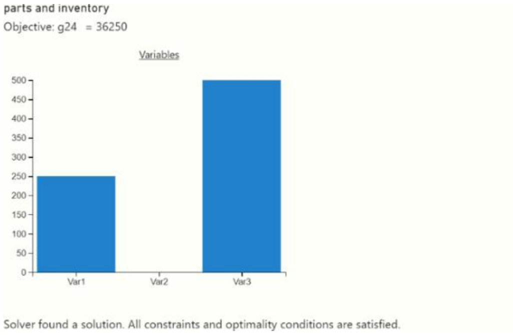


	A	B
1	parts	inventory
2	Chassis	450
3	LCD Screen	250
4	Speaker	1,000
5	Power Suppl	450
6	Electronics	1,000

refresh your data in Power BI….

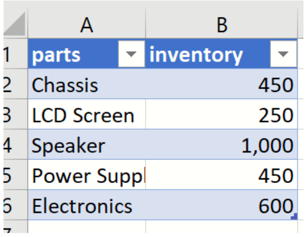


…and the Custom Visual will redraw with the correct objective function value, $36,250.



Solver found a solution. All constraints and optimality conditions are satisfied.

Return the number of electronics in inventory to 600 in ProductMixData.xlsx.



	A	B
1	parts	inventory
2	Chassis	450
3	LCD Screen	250
4	Speaker	1,000
5	Power Suppl	450
6	Electronics	600

Now let's go back to the Excel Model and see what happens when we change other parameters such as adding 2 more speakers to a Stereo and increasing the Stereo's selling price to $100. Click Help – Example Models – Optimization Examples – Product Mix for Power BI - Tableau to open the completed model.

First, we will use a 2nd PsiDataSrc() function to denote that new data (an increased selling price for Stereos) will be available in Power BI. Click back to the ProductMix(Opt) example model, select cell J7 and then click Formulas – Insert Formulas, select PSI Dimension for Category, select PsiDataSrc, then click OK.

Enter the following for the arguments appearing on the Function Arguments dialog.

Src_name – Enter “profit_src” to name the text file, created by the custom visual, which will be submitted, along with the RASON model, to the RASON server.

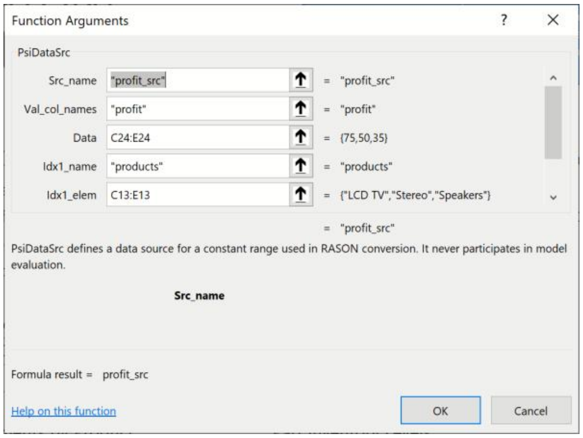
Val_col_names – Enter “profit” for the data well name.

Data – Enter the Excel range, C24:E24.

Idx1_name – Enter “products” which is the name of the 1st index set for the data specified in the Data argument. In this example, our Index set is simply the selling price for each product.

Idx1_elem – Enter the Excel range containing the data for the first index set, C13:E13.

Since this index set is 1-dimensional, nothing further is needed and we can click OK to enter the PsiDataSrc() function into cell J7. Afterwards, cell J7 will display the value given to the Src_name argument, profit_src.



Src_name "profit_src" = "profit_src"
Val_col_names "profit" = "profit"
Data C24:E24 = {75,50,35}
Idx1_name "products" = "products"
Idx1_elem C13:E13 = {"LCD TV","Stereo","Speakers"}
= "profit_src"

PsiDataSrc defines a data source for a constant range used in RASON conversion. It never participates in model evaluation.

Formula result = profit_src

In Analytic Solver Cloud, you can type the formula directly into cell J7.

```
=PsiDataSrc("profit_src", "profit", C24:E24,
"products", C13:E13)
```

We will use a 3rd PsiDataSrc() function to denote that new data (an increase in the number of speakers to be included in the building of a stereo) will be available in Power BI. This time, we will simply enter the formula directly into cell J8.

First type “=PsiDataSrc(“, then enter the arguments below.

Src_name – Enter “reqparts_src” to name the text file which will be submitted, along with the RASON model, to the RASON server.

Val_col_names – Enter “requiredparts” for the data well name.

Data – Enter the Excel range, C18:E22.

Idx1_name – Enter “parts” which is the name of the 1st index set for the data specified in the Data argument. In this example, there are two index sets. When data is 2-dimensional, we must enter the index set that describes the rows first.

Idx1_elem – Enter the Excel range containing the data for the first index set, B18:B22.

Idx2_name – Enter “products” which is the name of the 2nd index set for the data specified in the Data argument.

Idx2_elem – Enter the Excel range containing the data for the second index set, C13:E13.

Now we can type “)” to close the function, then press Enter to enter the PsiDataSrc() function into cell J8. Afterwards, cell J8 will display the value given to the Src_name argument, reqparts_src.

Your PsiDataSrc function should be similar to the screenshot below.

=PsiDataSrc("reqparts_src", "requiredparts", C18:E22, "parts", B18:B22, "products", C13:E13)

At this point, cell J6 displays “inventory_src”, cell J7 displays “profit_src” and cell J8 displays “reqparts_src”.

Finally, we will use a 4th PsiDataSrc() function to add our variable names to the Power BI chart. This data well will perform double duty by allowing a "starting value" to be passed to the custom visual. Again, we will simply enter the formula directly into cell J9.

First type “=PsiDataSrc(“, then enter the arguments below.

Src_name – Enter “vars_src” to name the text file which will be submitted, along with the RASON model, to the RASON server.

Val_col_names – Enter “initialvalues” for the data well name.

Data – Enter the Excel range, C14:E14.

Idx1_name – Enter “products” which is the name of the 1st index set for the data specified in the Data argument. In this example, there are two index sets. When data is 2-dimensional, we must enter the index set that describes the rows first.

Idx1_elem – Enter the Excel range containing the data for the first index set, C13:E13.

Now we can type “)” to close the function, then press Enter to enter the PsiDataSrc() function into cell J9. Afterwards, cell J9 will display the value given to the Src_name argument, vars_src.

Your PsiDataSrc function should be similar to the function below.

=PsiDataSrc("vars_src", "initialvalues", C13:E13, "products", C14:E14)

At this point, cell J6 displays “inventory_src”, cell J7 displays “profit_src”, cell J8 displays “reqparts_src” and J9 displays "vars_src".

Before we create our new Custom Visual, we will add a PsiFinalValue() function to the worksheet in order to add our constraint final values to the visual. In cell K18, enter: =PsiFinalValue(Number_used).

Then enter the PsiSlackValue() function to display the slack values in the visual. In cell L18, enter: =PsiSlackValue(Number_used).

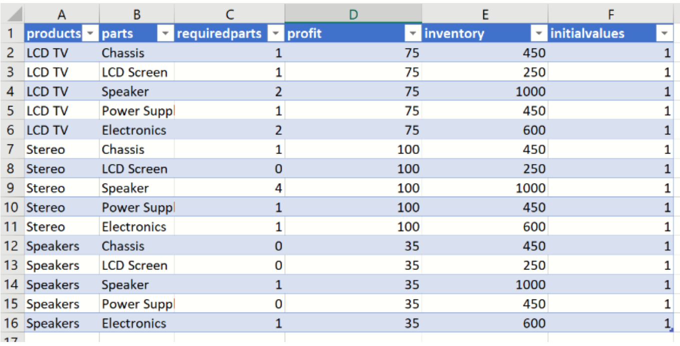
You'll need to update the model on Rason Cloud Services so save the Excel model and click Deploy Model – Cloud Service – Rason Model. You can reuse the previous model name. Then click Deploy Model – Power BI – Managed Model to create a new Custom Visual using a new name such as ProductMix3.pbiviz.

Now we are ready to add a new data table to the ProductMixData.xlsx file.

Open your data file, or the ProductMixData.xlsx workbook.

Important Note: **All data for a custom visual MUST be contained within the same Excel table or from within multiple tables with the same number of rows and columns.** As a result, we must create a new table in Excel and enter all data into this new table, including the information from the Inventory worksheet.

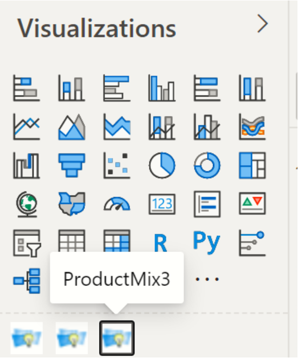
In a new Excel sheet, create a table (Insert – Table) with 16 rows and 6 columns, then list the products in the first column using the column heading “products”, the parts by name using the column heading “parts” in the 2nd column, the parts required per product (requiredparts) in the 3rd column, the selling price or “profit” per product in the 4th column, the part inventory for each product in the 5th column and the initial variable values in the last column. Make sure to pass the new selling price for Stereos ($100) and the increased number of speakers per stereo (4).



	A	B	C	D	E	F
1	products	parts	requiredparts	profit	inventory	initialvalues
2	LCD TV	Chassis	1	75	450	1
3	LCD TV	LCD Screen	1	75	250	1
4	LCD TV	Speaker	2	75	1000	1
5	LCD TV	Power Suppl	1	75	450	1
6	LCD TV	Electronics	2	75	600	1
7	Stereo	Chassis	1	100	450	1
8	Stereo	LCD Screen	0	100	250	1
9	Stereo	Speaker	4	100	1000	1
10	Stereo	Power Suppl	1	100	450	1
11	Stereo	Electronics	1	100	600	1
12	Speakers	Chassis	0	35	450	1
13	Speakers	LCD Screen	0	35	250	1
14	Speakers	Speaker	1	35	1000	1
15	Speakers	Power Suppl	0	35	450	1
16	Speakers	Electronics	1	35	600	1

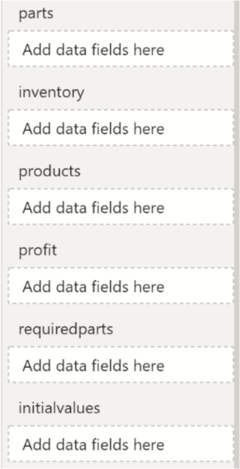
Notice that in order to list all of the required parts for each product, we must list each part and selling price for each part. In addition, each column heading matches the index sets used in the PsiDataSrc() functions. Rename the worksheet “AllData” then save the workbook. *If there are any leading or trailing spaces for the column headings or elements, Power BI will stop with an error.*

Open either desktop or cloud-based Power BI, click the icon containing three horizontal dots at the bottom of Visualizations, then select Import from file from the menu. Navigate to the location of the saved ProductMix3.pbiviz file. Select the file, then click Open. If the import was successful, you will see a message indicating as such. Click OK to clear this dialog. A new icon, bearing the Frontline Solvers logo and the name ProductMix3 is added under Visualizations.

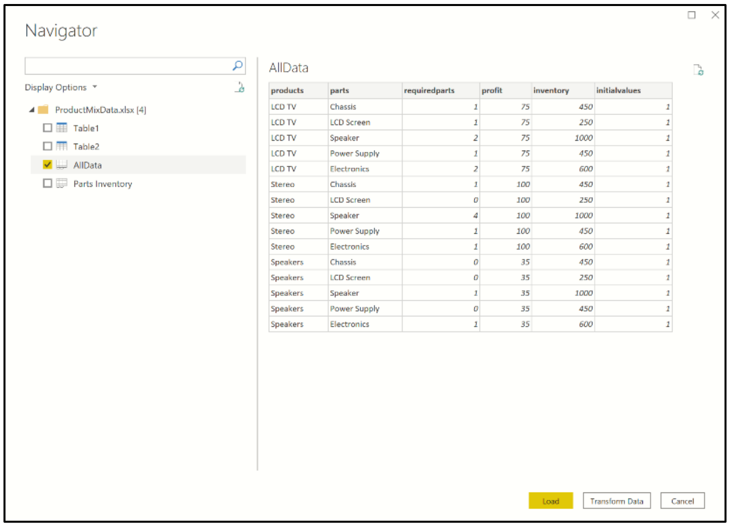


Click this icon to open Frontline Solvers' custom visual.

If you look to the right of the screen, you will see new data wells appearing under Visualizations. Recall that the "parts" and "inventory" data wells were created by the PsiDataSrc() function in the ProductMix(Opt) example model cell J6, inventory_src. Data wells "products" and "profit" were created by the PsiDataSrc() function in cell J7, profit_src. The PsiDataSrc() function in cell J8, reqparts_src, created the data well "requiredparts". The PsiDataSrc() function in cell J9, vars_src, created the data well "initialvalues".


parts
Add data fields here
inventory
Add data fields here
products
Add data fields here
profit
Add data fields here
requiredparts
Add data fields here
initialvalues
Add data fields here

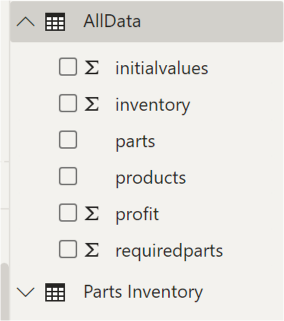
Now it is time to upload our data. Click back to Power BI, then click Get Data – Excel and open the ProductMixData.xlsx file.



products	parts	requiredparts	profit	inventory	initialvalues
LCD TV	Chassis	1	75	450	1
LCD TV	LCD Screen	1	75	250	1
LCD TV	Speaker	2	75	1000	1
LCD TV	Power Supply	1	75	450	1
LCD TV	Electronics	2	75	600	1
Stereo	Chassis	1	100	450	1
Stereo	LCD Screen	0	100	250	1
Stereo	Speaker	4	100	1000	1
Stereo	Power Supply	1	100	450	1
Stereo	Electronics	1	100	600	1
Speakers	Chassis	0	35	450	1
Speakers	LCD Screen	0	35	250	1
Speakers	Speaker	1	35	1000	1
Speakers	Power Supply	0	35	450	1
Speakers	Electronics	1	35	600	1

Select AllData on the left, then click Load on the right.

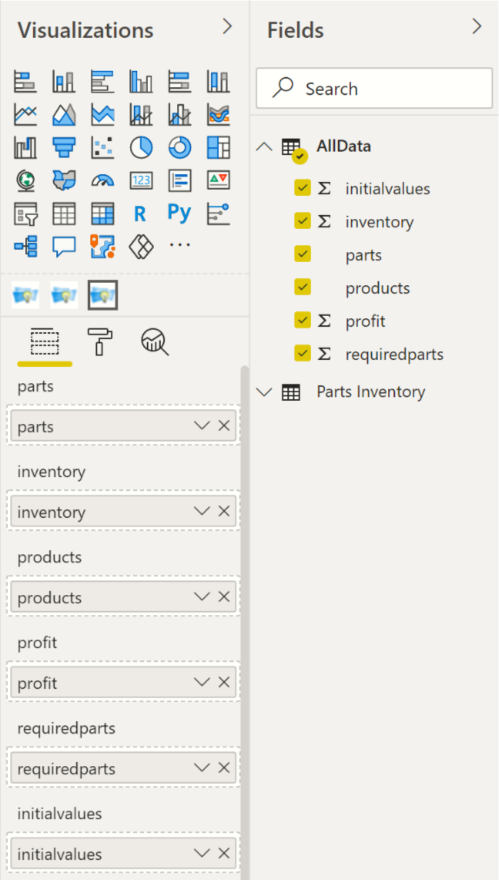
The table headings from the AllData worksheet are uploaded to the Fields task pane in Power BI.



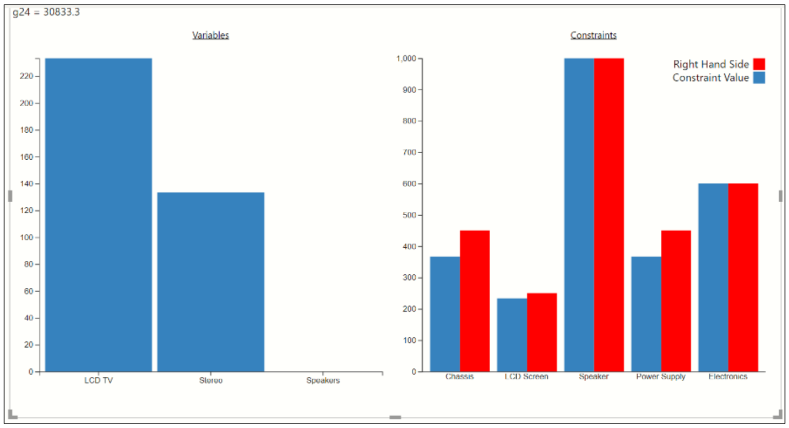
Drag the fields into the data wells according to the following table.

Field	Data Well
inventory	inventory
parts	parts
products	products
profit	profit
requiredparts	requiredparts
initialvalues	initialvalues

Afterwards, your task pane should match the following screenshot. Make sure to check that none are being summarized by clicking the down arrow beside each and selecting "Don't summarize" from the menu.

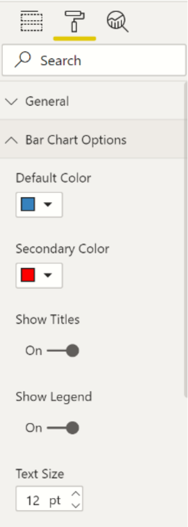


Immediately, the RASON model is submitted to the RASON Server, the model is solved and the final variable values are imported back into Power BI.



To update the data source, revise the data, refresh the data source in Power BI and watch as your Custom Visual is automatically redrawn.

As mentioned above, in latest version of Analytic Solver the generated Power BI visual can be further customized by the user by clicking the Format icon (Paint roller icon).



Search

General

Bar Chart Options

Default Color 🟦

Secondary Color 🟥

Show Titles: On

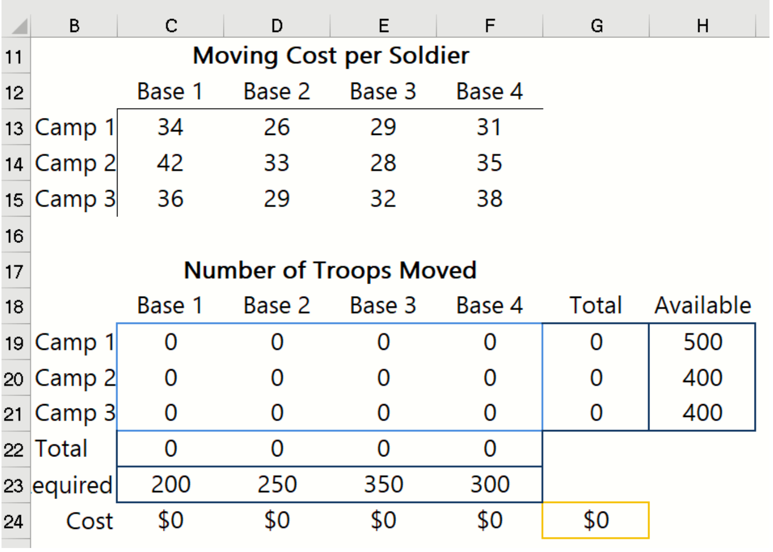
Show Legend: On

Text Size: 12 pt

The .pbiviz file that is created by the Deploy Model tool may be manually altered to change the chart type created by Power BI. For more information on how to change the chart type created, contact Frontline Systems Support at support@solver.com.

Troop Rotation Optimization Example

In this example, we will create a Power BI Custom Visual using a slightly more complex example, the Troop Rotation example. To open, click Help – Example Models on the Analytic Solver ribbon, then click Optimization Examples and scroll down to Troop Rotation for Power BI-Tableau. Click the hyperlink to open the example model. This example model illustrates the moving of army troops while minimizing cost.



	B	C	D	E	F	G	H
11		Moving Cost per Soldier					
12		Base 1	Base 2	Base 3	Base 4		
13	Camp 1	34	26	29	31		
14	Camp 2	42	33	28	35		
15	Camp 3	36	29	32	38		
16							
17		Number of Troops Moved					
18		Base 1	Base 2	Base 3	Base 4	Total	Available
19	Camp 1	0	0	0	0	0	500
20	Camp 2	0	0	0	0	0	400
21	Camp 3	0	0	0	0	0	400
22	Total	0	0	0	0		
23	equired	200	250	350	300		
24	Cost	$0	$0	$0	$0	$0	

The optimization model is included in the Model tab in the Solver Task Pane. This model written in algebraic form is below.

Maximize: Total Cost = 34 * x1 + 42 * x5 + 36 * x9 + 26 * x2 + 33 * x6 + 29 * x10 + 29 * x3 + 28 * x7 + 32 * x11 + 31 * x4 + 35 * x8 + 38 * x12

Subject To:

Number of troops from Camps 1, 2 and 3 moved to Bases 1, 2, 3 and 4 must not exceed the number of troops available

Camp1: x1 + x2 + x3 + x4 <= 500

Camp2: x5 + x6 + x7 + x8 <= 400

Camp3: x9 + x10 + x11 + x12 <= 400

Number of troops assigned per base must not be less than the amount required.

Base 1: x1 + x5 + x9 >= 200

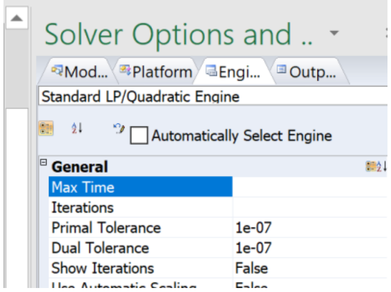
Base 2: x2 + x6 + x10 >= 250

Base 3: x3 + x7 + x11 >= 350

Base 4: x4 + x8 + x12 >= 300

x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12 > 0

The engine selected to solve this model is the Standard LP/Quadratic Engine as shown on the Engine tab in the Solver Task Pane.



Standard LP/Quadratic Engine

☐ Automatically Select Engine

General

Max Time	
Iterations	
Primal Tolerance	1e-07
Dual Tolerance	1e-07
Show Iterations	False

Next, we will enter four PsiDataSrc() functions into cells J13, J14, J15, and J16 in order to expose our data to the RASON Server and ultimately Power BI.

In J13, we will expose the moving costs per soldier according to the Base and Camp dimensions in cells C13:F15. To open the Function Arguments dialog, click Formulas – Insert Function, select PSI Dimension for Category, select PsiDataSrc, then click OK.

Src_name = Enter “cost_src” for this argument. This names the text file that will be submitted to the RASON server, along with the RASON model, when the Power BI custom visual is created.

Val_col_names – Enter “cost” for this argument. This will be the name of the Power BI data well associated with the data passed in the next argument.

Data – Enter C13:F15 which is the actual cells containing the moving price per soldier.

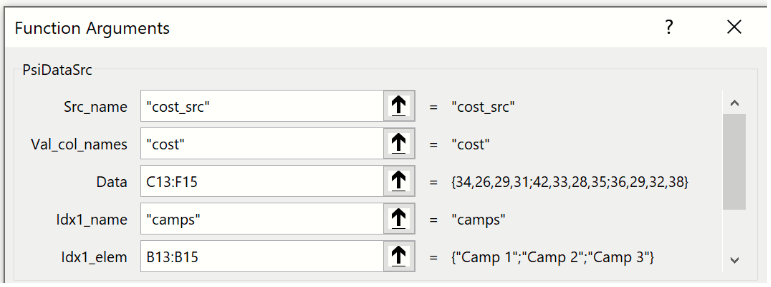
Idx1_name – Enter “camps” which is the name of the 1st dimension or index set, the row-wise index set. Remember, you must pass the row-wise index set before the column-wise index set.

Idx1_elem – Enter the elements of the “camps” index set. These appear in cells B13:B15.

Idx2_name – Enter “bases” which is the name of the 2nd dimension or index set, the column wise index set.

Idx2_elem – Enter the elements of the “bases” index set. These appear in cells C12:F12.

The Function Arguments dialog should now appear as shown in the screenshots below.



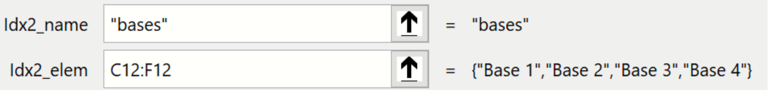
Src_name "cost_src" = "cost_src"

Val_col_names "cost" = "cost"

Data C13:F15 = {34,26,29,31;42,33,28,35;36,29,32,38}

Idx1_name "camps" = "camps"

Idx1_elem B13:B15 = {"Camp 1";"Camp 2";"Camp 3"}



Idx2_name "bases" = "bases"

Idx2_elem C12:F12 = {"Base 1","Base 2","Base 3","Base 4"}

Click OK to save this function to cell J13.

In Analytic Solver Cloud, you can type the formula directly into cell J13.

```
=PsiDataSrc("cost_src", "cost", C13:F15, "camps",
B13:B15, "bases", C12:F12)
```

Click cell J14 and open the Function Arguments dialog to enter the next PsiDataSrc() function.

In J14, we will expose the variables that indicate the number of troops moved by camp and base in cells C19:F21. Open the PsiDataSrc function (as explained above).

Src_name = Enter “vars_src” for this argument.

Val_col_names – Enter “troopsmoved” for this argument. This will be the name of the Data Well in Power BI that is associated with this data.

Data – Enter C19:F21 which is the actual cells containing the moving price per soldier. Note: The data range C19:F21 will automatically convert to “Troops_moved” which is the defined name for this Excel range.

Idx1_name – Enter “camps” which is the name of the 1st dimension or index set, the row-wise index set.

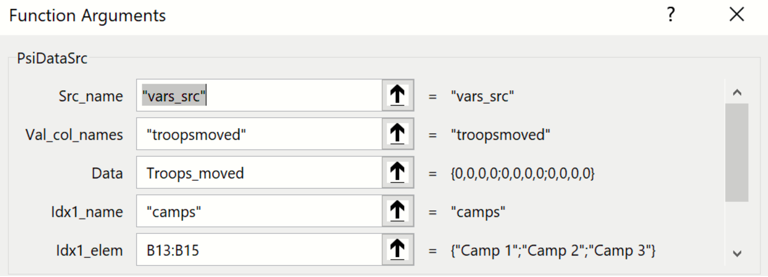
Idx1_elem – Enter the elements of the “camps” index set. These appear in cells B13:B15.

Note that it is important to re-use the location of the index sets. If we would have entered the elements for "camps" as B19:B21, rather than B13:B15, a new index set would be created.

Idx2_name – Enter “bases” which is the name of the 2nd dimension or index set, the column-wise index set.

Idx2_elem – Enter the elements of the “bases” index set. These appear in cells C12:F12.

The Function Arguments dialog should now appear as shown in the screenshots below.



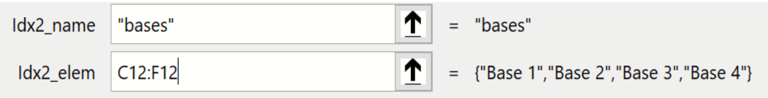
Src_name "vars_src" = "vars_src"

Val_col_names "troopsmoved" = "troopsmoved"

Data Troops_moved = {0,0,0,0;0,0,0,0;0,0,0,0}

Idx1_name "camps" = "camps"

Idx1_elem B13:B15 = {"Camp 1";"Camp 2";"Camp 3"}



Idx2_name "bases" = "bases"

Idx2_elem C12:F12 = {"Base 1","Base 2","Base 3","Base 4"}

Click OK to save this function to cell J14.

In Analytic Solver Cloud, you can type the formula directly into cell J14.

```
=PsiDataSrc("vars_src", "troopsmoved", Troops_moved,
"camps", B13:B15, "bases", C12:F12)
```

Let's enter the last two PsiDataSrc() functions by typing the formulas directly into the Excel cells, starting with the function in cell J15 that will expose the number of troops available for movement.

In cell J15, enter: "=PsiDataSrc("

First the first argument, Src_name, enter "available_src".

For the second argument, Val_col_names, enter: "availabletroops". The Power BI Data Well associated with this data will be given this name.

For the Data argument, the third argument, enter: Troops_available. This is the defined name for cells H19:H21. Alternatively, we could have typed the Excel range, H19:H21.

Type "camps" for the 4th argument, Idx1_name, which is the name given to the row-wise index set.

Enter the range for this index set, B13:B15, for Idx1_elem.

Your PsiDataSrc() function should match the one in the screenshot shown below.

```
=PsiDataSrc("available_src", "availabletroops",
Troops_available, "camps", B13:B15)
```

Click Enter to enter the PsiDataSrc() function into cell J15.

Finally, the last PsiDataSrc() function in cell J16 should be entered as:

=PsiDataSrc("required_src","requiredtroops",Troops_required,"bases",C12:F12)

Where:

Data_src = "required_src"

Val_col_names = "requiredtroops" This is the name that will appear for the Data Well associated with this data in Power BI.

Data = Troops_required This is the data associated with Val_col_names.

Idx1_name = "bases" This is the name of the index set that is associated with the data passed to the Data argument.

Idx1_elem – c12:f12 This is the Excel range where the elements for Idx1_name are located on the worksheet.

Now we are ready to create the custom visual.

In order to see our constraint final values in the Custom Visual, we will need to use the Psi function PsiFinalValue. (For more information on this function, see the Psi Function chapter in the Analytic Solver Reference Guide.)

- In cell K23, enter "=PsiFinalValue(Troops_per_camp)" to display the final value of the constraints in cells G19:G21 in the Custom Visual. In the COM addin, this function "spills" vertically to cells K23:K25, since the orientation of cells G19:G21 are vertical. In Analytic Solver Cloud, this function "spills" horizontally to cells K23:M23.
- In cell L22, enter "=PsiFinalValue(Troops_per_base)" to display the final value of the constraints in cells C22:F22 in the Custom Visual. This function will "spill" horizontally to cells L22:O22, since the orientation of cells C22:F22 are vertical. (This function spills horizontally in the Cloud app regardless of the orientation of cells C22:F22.)

We can also add PsiSlackValue functions to the workbook to display the constraint slack values in the Power BI Custom Visual. Note that in the COM addin, the PsiFinalValue function in cell K23 will spill down to cell K25 so in order not to prevent the spillage, we will enter the PsiSlackValue functions in row 29.

- In cell K29, enter "=PsiSlackValue(Troops_per_camp)" to see the slack values of the constraints in cells G19:G21
- In cell L28, enter "=PsiSlackValue(Troops_per_base)" to display the slack values of the constraints in cells C22:F22.

First, click Deploy Model on the ribbon, then select Cloud Service – RASON Model to post the model to RASON Cloud Services. Then click Deploy Model again and select Power BI – Managed Model.

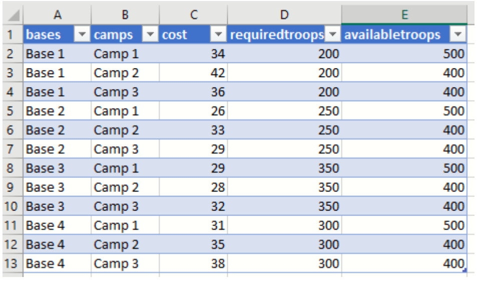
Since this is the first time we are creating a custom visual from this model, there is no need to solve the model first. If however, we make a change to the Troop Rotation model, then we would need to either diagnose or solve the model first to ensure that the changes are captured in the new custom visual.

Save the .pbiviz file as TroopRotation1.pbiviz in a location of your choice.

Next, we will create the data file. Open a new Excel workbook. Recall, that when supplying data to Power BI, all data must be contained in tables containing the exact same number of rows and columns. If you are unable to pass your data using Excel tables with the exact same number of rows and columns, then you must pass all data in one table.

In this example, since we would not be able to pass the data in multiple tables with the same number of rows, we will pass all data in one Excel table.

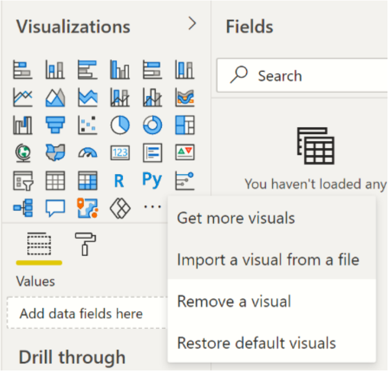
In a new Excel sheet, create a table (Insert – Table) with 13 rows and 5 columns, as shown in the screenshot below. Notice that in order to enter all 12 costs, labels for camps, bases, requiredtroops and availabletroops were repeated. In addition, each column heading matches the index sets used in the PsiDataSrc() functions. Rename the worksheet “AllData” then save the workbook as TroopsRotationData.xlsx.



	A	B	C	D	E
1	bases	camps	cost	requiredtroops	availabletroops
2	Base 1	Camp 1	34	200	500
3	Base 1	Camp 2	42	200	400
4	Base 1	Camp 3	36	200	400
5	Base 2	Camp 1	26	250	500
6	Base 2	Camp 2	33	250	400
7	Base 2	Camp 3	29	250	400
8	Base 3	Camp 1	29	350	500
9	Base 3	Camp 2	28	350	400
10	Base 3	Camp 3	32	350	400
11	Base 4	Camp 1	31	300	500
12	Base 4	Camp 2	35	300	400
13	Base 4	Camp 3	38	300	400

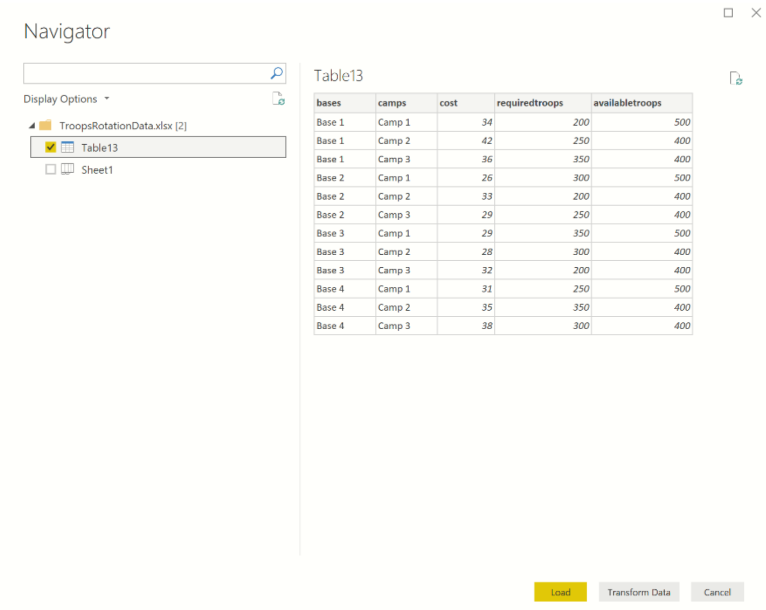
Now let's click back to Power BI, to import the new custom visual and the data.

Click the three dots under Visualizations and select Import from file.



Once the Frontline Solvers icon is inserted under Visualizations, click the icon to open the custom visual.

Then click Get Data – Excel. Browse to the folder where the TroopsRotationData.xlsx file was saved, and open the file. Select the table containing your data, then click Load. (In this example, select "Table 13", then Load.)



bases	camps	cost	requiredtroops	availabletroops
Base 1	Camp 1	34	200	500
Base 1	Camp 2	42	250	400
Base 1	Camp 3	36	350	400
Base 2	Camp 1	26	300	500
Base 2	Camp 2	33	200	400
Base 2	Camp 3	29	250	400
Base 3	Camp 1	29	350	500
Base 3	Camp 2	28	300	400
Base 3	Camp 3	32	200	400
Base 4	Camp 1	31	250	500
Base 4	Camp 2	35	350	400
Base 4	Camp 3	38	300	400

Now it is time to drag the data into the data wells. Drag:

availabletroops → availabletroops

bases → bases

camps → camps

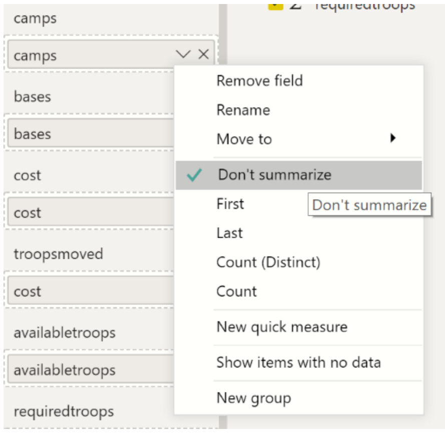
cost → cost

cost → troopsmoved

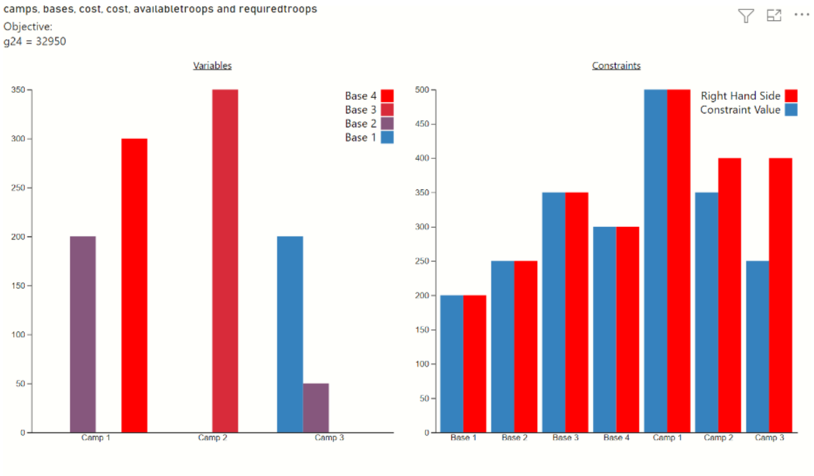
requiredtroops → requiredtroops

Note: Since “troopsmoved” holds the variable values, we can simply drag “cost”, which contains the same number of rows and columns as "troopsmoved", as a starting point for the algorithm. A starting point for this model is not needed because our model is linear, however, if our model was nonlinear or nonsmooth, this would be more important.

After dragging each field to its appropriate data well, click the down arrow next to each element and select "Don't summarize". This is selected in order that the data not be aggregated in the custom visual.



The resultant Power BI custom visual is shown in the screenshot below. Now each time new data is imported into Power BI, the Troop Rotation example data will be solved from within Power BI. However, please note that as long as your PsiDataSrc functions remain the same, the named model can be updated on the Rason Cloud Service and you will not need to create a new custom visual. However, if the PsiDataSrc() functions change, you will need to update the named model residing on the Rason Cloud service and the Custom Visual.


camps, bases, cost, cost, availabletroops and requiredtroops

Objective:
g24 = 32950

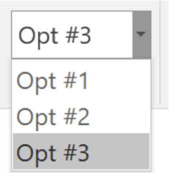
Notice that PsiSlackValues and the PsiFinalValues are both displayed on the right under Constraints.

Using Optimization Parameters in Power BI

Now what happens when we would like to see how our final variable values and objective function value change as the number of troops required changes. For this type of "What If" analysis, we can use the PsiOptParam() optimization parameter. Recall from the chapter, Examples: Parameters and Sensitivity Analysis, an **optimization parameter** (PsiOptParam) is automatically varied when you perform multiple optimizations. You do this by clicking the Optimize button on the Ribbon, or the green arrow in the Task Pane, with the Number of Optimizations set to a value greater than 1.

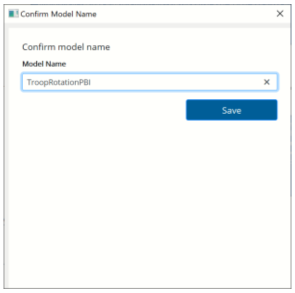
Let's go back to our example Excel model and enter an optimization parameter for the number of troops required in Base 1. In cell C13, enter "=PsiOptParam({34,24,14})", then click the Platform tab in the Solver Task pane and change the Number of Optimizations to "3". When I click Solve, Analytic Solver will solve 3 optimizations. In the first optimization, cell C13 will be set to 34, in the second, 24, and in the third, 14.

Click the Output tab, then click the green arrow to solve the model. After the solving process is over, we can select the three different optimizations in the Optimization drop down menu (in the Tools section of the Analytic Solver ribbon), to see how the objective function changes according to the value set for this parameter.



In the first optimization, cell C13 = 34 and the resultant objective function value was $32,950. In the second optimization, cell C13 = 24 and the resultant objective function value was $31,150 and in the third optimization cell C13 = 14 with an objective function value of 29,150.

Save the Excel file, then click Deploy Model – Cloud Service – RASON Model to update the model on RASON server.



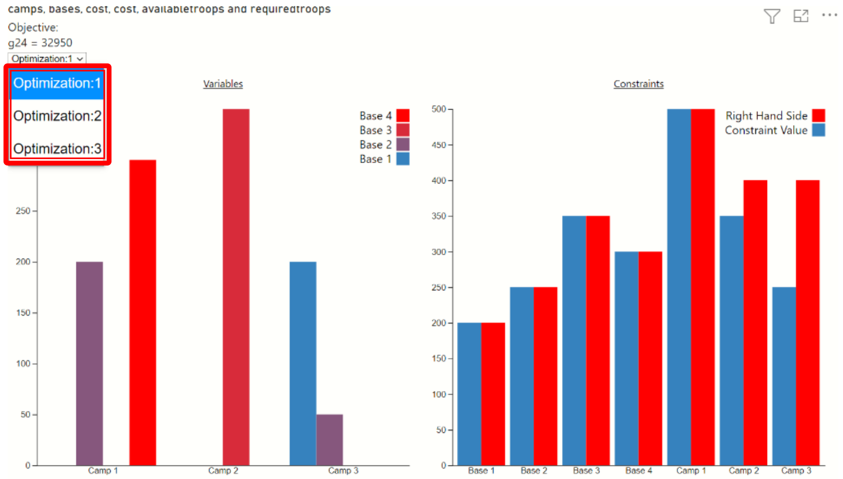
Confirm model name

Model Name: TroopRotationPBI

Then click Deploy Model – Power BI – Managed Model to create a new custom visual file. This time, we will name the custom visual file, TroopRotationOptParam.pbiviz.

Click back to Power BI and import the newly created custom visual file. Click the new Frontline Solvers icon to open the custom visual. Immediately, the model will solve and the results will be displayed in the Power BI window. To view each solution, click the down arrow within the newly added Multiple Optimization drop-down menu.

Note: When entering PsiOptParam manually (i.e. simply typing the function into an Excel cell) in Analytic Solver Cloud, the taskpane must be refreshed before performing a parameter analysis report.



camps, bases, cost, cost, availabletroops and requiredtroops

Objective:

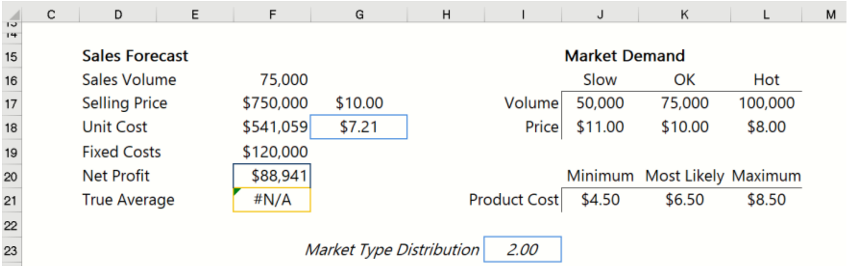
g24 = 32950

Simulation Example Model

Now let’s create a custom visual from a simulation example. This time, we will *embed* the model into the Power BI Custom Visual. This option is different than Power BI – Managed Model because with Power BI -- Embedded Model, your model is embedded inside the custom visuals' JavaScript file. *There is no need to create a named model that "lives" on the public Rason cloud*. The custom visual accepts the data your model needs from Power BI; whenever this data changes, it sends your Rason model plus this data to the public Rason cloud service where it is solved, obtains the results and displays them in chart form.

Note: Although this option is initially simpler, if you want to update your model in any way, you'll have to use this menu option to create a new Dashboard Extension. If you expect to revise the model over time, it's better to use Deploy Model – Power BI – Managed Model.

Open the Business Forecast Example by clicking Help – Example Models – Monte Carlo Simulation Examples , Business Forecast hyperlink to open the file, then click the Forecast with Uncertainty tab. To open a completed example, open the BusinessForecastPBI example that appears at the bottom of Simulation Models.



	C	D	E	F	G	H	I	J	K	L	M
15		**Sales Forecast**						**Market Demand**			
16		Sales Volume		75,000				Slow	OK	Hot	
17		Selling Price		$750,000	$10.00		Volume	50,000	75,000	100,000	
18		Unit Cost		$541,059	$7.21		Price	$11.00	$10.00	$8.00	
19		Fixed Costs		$120,000							
20		Net Profit		$88,941				Minimum	Most Likely	Maximum	
21		True Average		#N/A			Product Cost	$4.50	$6.50	$8.50	
22											
23				*Market Type Distribution*			*2.00*				

First, we will use a PsiDataSrc() function to generate data wells for use in the Power BI custom visual.

Click a blank cell, such as cell G24, then type: “=PsiDataSrc(“market_src”, “market”, J17:L18, “parameters”, I17:I18, “mktdemand”, J16:L16) where:

market_src – Is the name used to identify the text file that will be submitted to the RASON server.

market – Will be the name of the data well in Power BI.

J17:L18 – Is the data range for “market”.

parameters – Is the name of the 1st index set for the data specified in the “data” argument, market.

I17:I18 – Is the Excel range where the data for “parameters” is located.

mktdemand – Is the name of the 2nd index set for the data specified in the “data” argument.

J16:L16 – Is the Excel range where the data for “mktdemand” is located.

Type “)” to close the formula and then press the Enter key to enter the formula into the Excel cell.

Click a 2nd blank cell, such as G25, then enter the following function.

=PsiDataSrc(“price_src”, “mktprice”, J21:L21, “price”, J20:L20)

where:

price_src – Is the name used to identify the text file that will be submitted to the RASON server.

mktprice – Will be the name of the data well in Power BI.

J21:L21 – Is the data range for “mktprice”.

price – Is the name of the 2nd index set for the data specified in the Data argument, market.

J20:L20 – Is the Excel range where the data for “price” is located.

Two Psi Functions, PsiPercentiles and PsiData, must be added to the spreadsheet in order to draw the histogram in the Custom Visual.

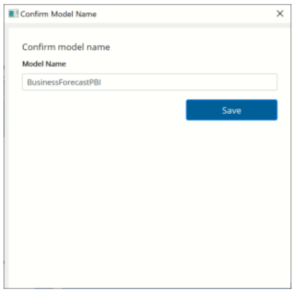
H28 = PsiPercentiles(F20)

G28 = PsiData(F20)

The PsiPercentile function returns all percentiles for the uncertain function in cell F20. The PsiData function returns all trial values for cell F20.

For more information on these formulas, see the Analytic Solver Reference Guide.

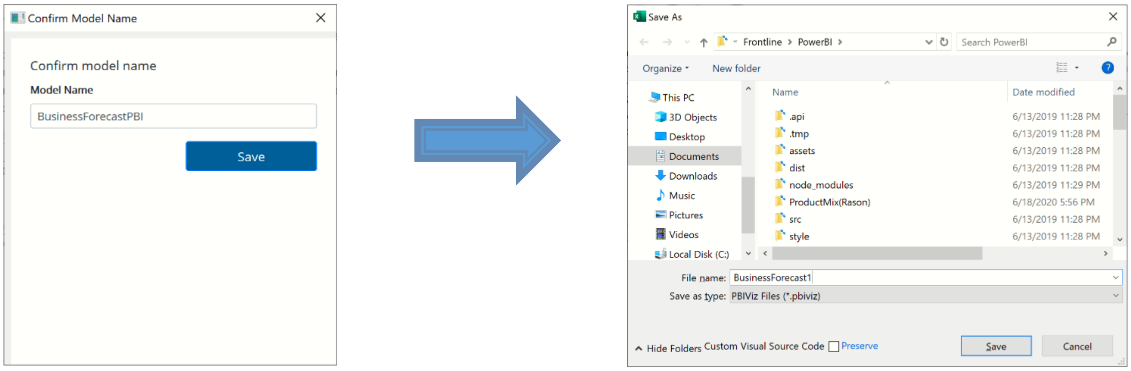
Click Deploy Model – Cloud Service – Rason Model to post the simulation model to the RASON server.



Confirm model name

Model Name: BusinessForecastPBI

Then click Deploy Model – Power BI -- Managed Model to create a new Power BI Custom Visual. Confirm that the name of the model on the Rason Cloud Service and then enter a name for the Custom Visual.



Confirm model name

Model Name: BusinessForecastPBI

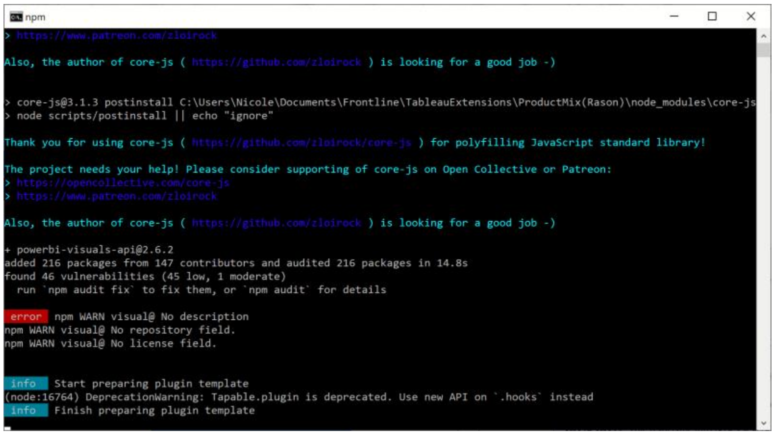
Name	Date modified
.api	6/13/2019 11:28 PM
.tmp	6/13/2019 11:28 PM
assets	6/13/2019 11:28 PM
dist	6/13/2019 11:28 PM
node_modules	6/13/2019 11:29 PM
ProductMix(Rason)	6/18/2020 5:56 PM
src	6/13/2019 11:28 PM
style	6/13/2019 11:28 PM

File name: BusinessForecast1

Save as type: PBIViz Files (*.pbiviz)

Custom Visual Source Code ☐ Preserve

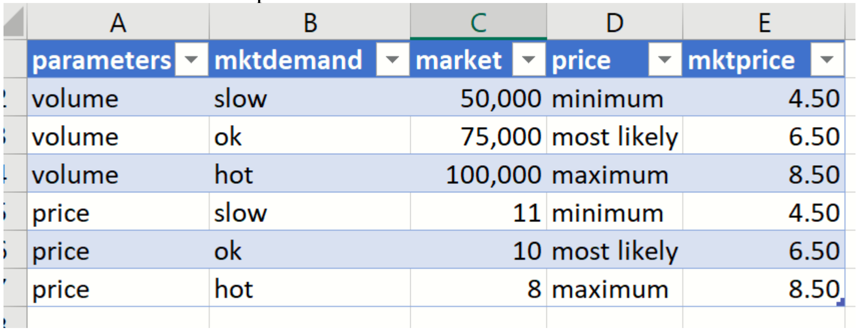
A window appears as the Custom Visual is being created.



```
> https://www.patreon.com/zloirock

Also, the author of core-js ( https://github.com/zloirock ) is looking for a good job -)


> core-js@3.1.3 postinstall C:\Users\Nicole\Documents\Frontline\TableauExtensions\ProductMix(Rason)\node_modules\core-js
> node scripts/postinstall || echo "ignore"

Thank you for using core-js ( https://github.com/zloirock/core-js ) for polyfilling JavaScript standard library!

The project needs your help! Please consider supporting of core-js on Open Collective or Patreon:
> https://opencollective.com/core-js
> https://www.patreon.com/zloirock

Also, the author of core-js ( https://github.com/zloirock ) is looking for a good job -)

+ powerbi-visuals-api@2.6.2
added 216 packages from 147 contributors and audited 216 packages in 14.8s
found 46 vulnerabilities (45 low, 1 moderate)
  run `npm audit fix` to fix them, or `npm audit` for details

 error  npm WARN visual@ No description
npm WARN visual@ No repository field.
npm WARN visual@ No license field.


 info  Start preparing plugin template
(node:16764) DeprecationWarning: Tapable.plugin is deprecated. Use new API on `.hooks` instead
 info  Finish preparing plugin template
```

Click back to Excel and open a new workbook to enter the following data **into an Excel table**. Then save the workbook as BusinessForecastData.xlsx.

You can open this completed data file from Help -- Example Models – Monte Carlo Simulation Examples – BusinessForecastData.xlsx



parameters	mktdemand	market	price	mktprice
volume	slow	50,000	minimum	4.50
volume	ok	75,000	most likely	6.50
volume	hot	100,000	maximum	8.50
price	slow	11	minimum	4.50
price	ok	10	most likely	6.50
price	hot	8	maximum	8.50

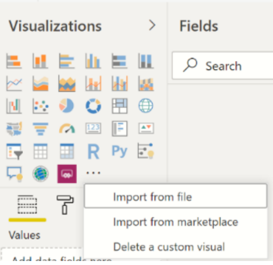
Open Power BI and add in the custom visual (Import from file – BusinessForcast1.pbiviz), then click Get Data – Excel to import the data. Afterwards, drag the data fields to the appropriate data wells as shown in the table below.

Fields	Data Wells

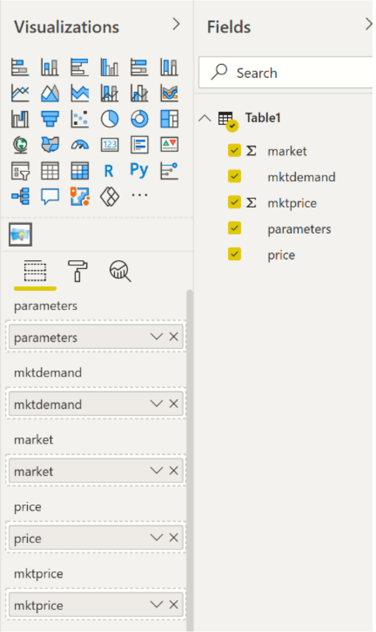
market	market
mktdemand	mktdemand
mktprice	mktprice
parameters	parameters
price	price

Analytic Solver automatically creates the Custom Visual, embedded with the RASON model, that can be dragged onto the Power BI Dashboard, or a report.

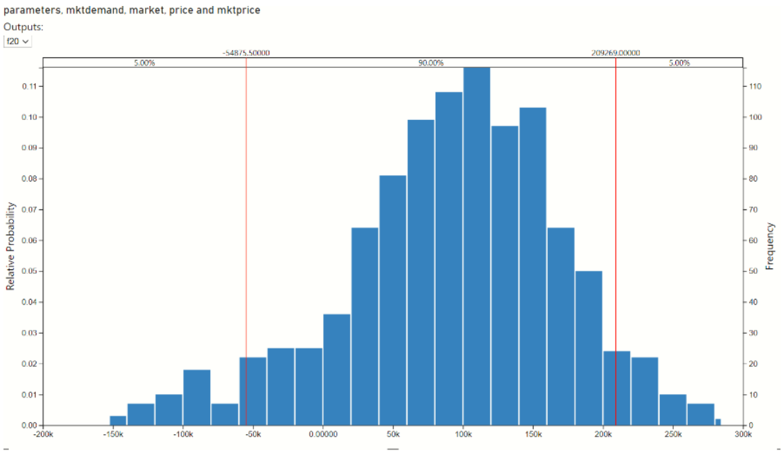
Once the npm window disappears, open Microsoft Power BI. Upon opening, click the three dots beneath Visualizations, and select Import from file.



Your task pane should match the following screenshot. Make sure that none of the Fields are being summarized.

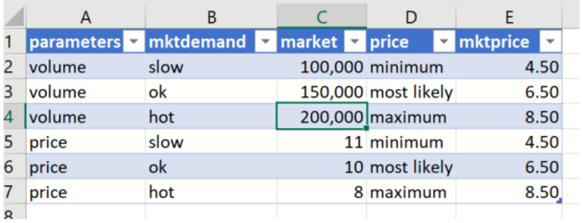


Immediately, once the data wells are filled with the appropriate data fields, a simulation will be run.



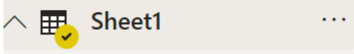
Again, note that as long as your PsiDataSrc functions remain the same, the named model can be updated on the Rason Cloud Service and you will not need to create a new custom visual. However, if the PsiDataSrc() functions change, you will need to create a new embedded custom visual.

If your data changes, say your demand explodes, you can simply edit your data file.

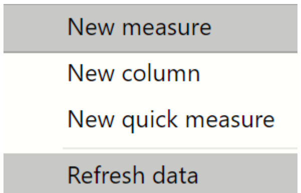


	A	B	C	D	E
1	parameters	mktdemand	market	price	mktprice
2	volume	slow	100,000	minimum	4.50
3	volume	ok	150,000	most likely	6.50
4	volume	hot	200,000	maximum	8.50
5	price	slow	11	minimum	4.50
6	price	ok	10	most likely	6.50
7	price	hot	8	maximum	8.50
8					

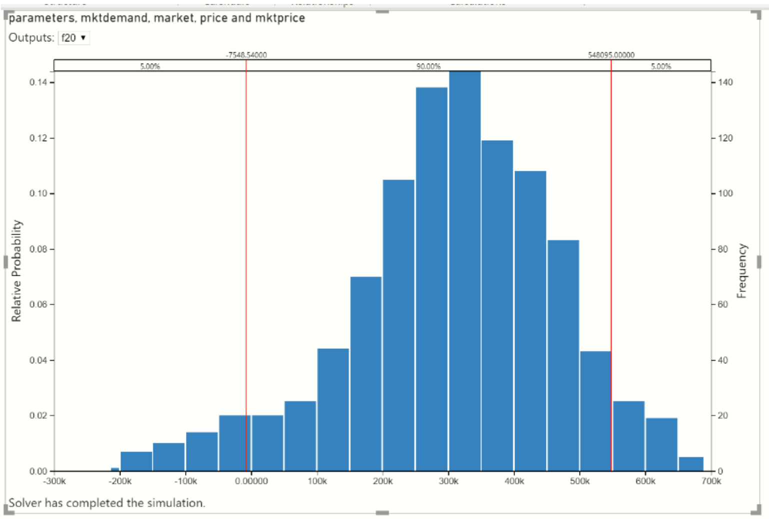
Click back to Power BI and click the three dots to the right of the data source…



and select Refresh Data from the menu.



The Custom Visual will automatically redraw using the new data.



parameters, mktdemand, market, price and mktprice

Outputs: f20 ▾

Solver has completed the simulation.

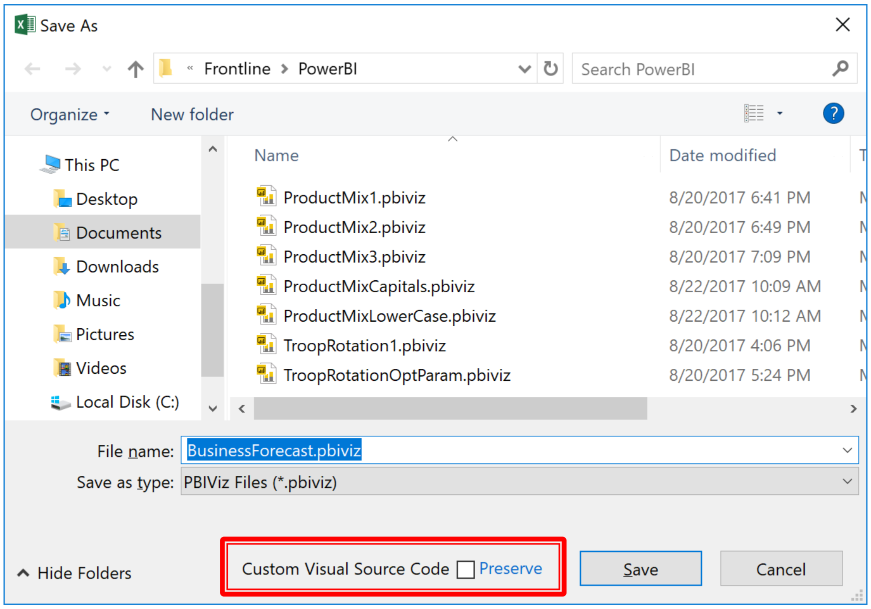
Advanced Settings

Recall from the examples above, once you click Deploy Model – Power BI – Managed Model in the Excel worksheet and confirm the model name, the Save As dialog appears.

At the bottom of this dialog is the Custom Visual Source Code Preserve checkbox. When this option is selected, a folder is created in the same location as the .pbiviz file. This folder will contain the source code for the custom visual.

In the screenshots below, new folders, BusinessForecast in Analytic Solver Desktop and ProductMix1 in Analytic Solver Cloud, will be created at the same time as the BusinessForecast.pbiviz and ProductMix1.pbiviz files.

Analytic Solver Desktop

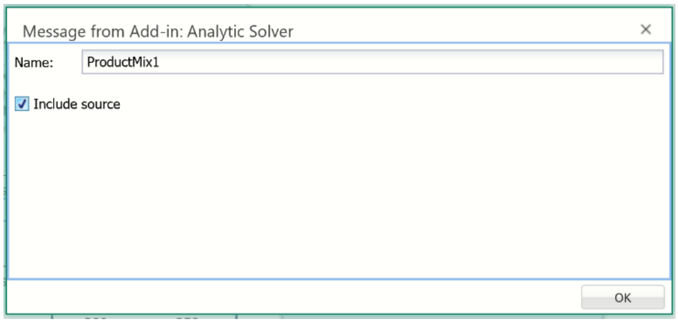


Name	Date modified
ProductMix1.pbiviz	8/20/2017 6:41 PM
ProductMix2.pbiviz	8/20/2017 6:49 PM
ProductMix3.pbiviz	8/20/2017 7:09 PM
ProductMixCapitals.pbiviz	8/22/2017 10:09 AM
ProductMixLowerCase.pbiviz	8/22/2017 10:12 AM
TroopRotation1.pbiviz	8/20/2017 4:06 PM
TroopRotationOptParam.pbiviz	8/20/2017 5:24 PM

File name: BusinessForecast.pbiviz

Save as type: PBIViz Files (*.pbiviz)

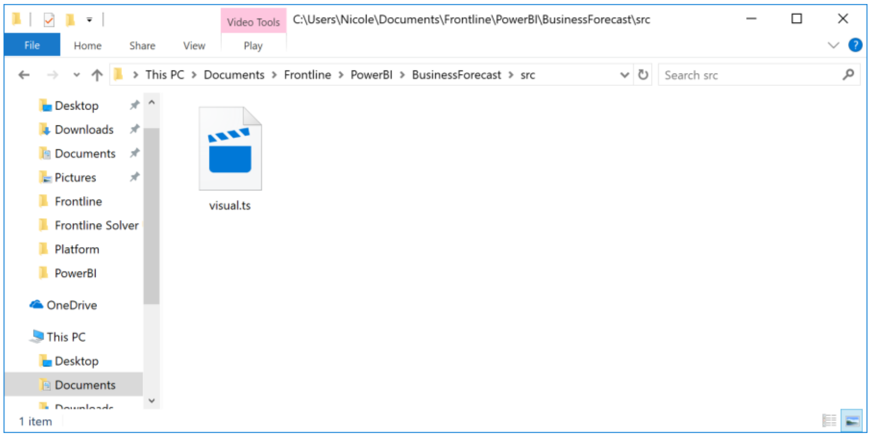
Custom Visual Source Code ☐ Preserve



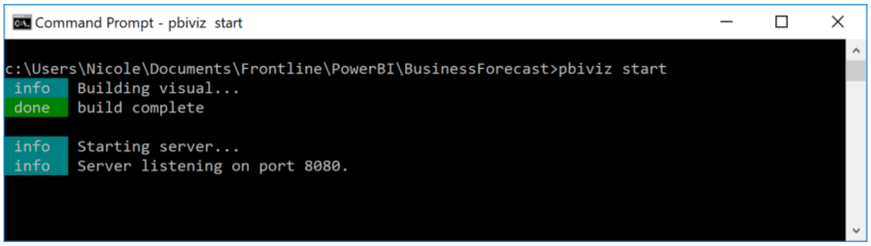
Name: ProductMix1

☑ Include source

You will find the type script file, visual.ts within the src folder. Users can manipulate this file to customize the look and feel of the custom visual, such as the face of the custom visual icon (within Visualizations), alter the RASON model, etc. For more information on customizing the visual.ts file, see the webpage: https://powerbi.microsoft.com/en-us/documentation/

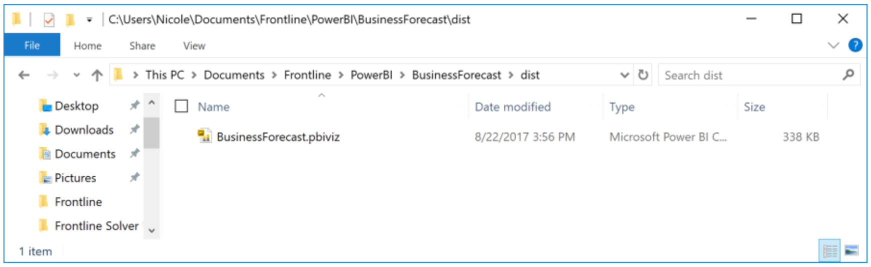


Note: If the visual.ts file is altered, a new pbiviz package must be created.   To do so, open a command prompt, change the directory to the root directory of the custom visual, and enter “pbiviz package”, as shown in the screenshot below.



```
c:\Users\Nicole\Documents\Frontline\PowerBI\BusinessForecast>pbiviz start
 info   Building visual...
 done   build complete

 info   Starting server...
 info   Server listening on port 8080.
```

The new custom visual file (.pbiviz) will be saved to the Dist folder.


C:\Users\Nicole\Documents\Frontline\PowerBI\BusinessForecast\dist
File
Home
Share
View
This PC > Documents > Frontline > PowerBI > BusinessForecast > dist
Search dist
Desktop
Downloads
Documents
Pictures
Frontline
Frontline Solver
1 item

Name	Date modified	Type	Size
BusinessForecast.pbiviz	8/22/2017 3:56 PM	Microsoft Power BI C...	338 KB

Creating a Custom Visual from a Large Model

For very large embedded models, node.js might run out of heap space when attempting to generate the pbiviz package. If all heap space is consumed, you'll receive the message, " Javascript out of heap space". In order to increase the heap space, simply create a new Environment Variable "NODE_OPTIONS" with value "—max-old-space-size=8192". Doing so will increase the heap space and allow the custom visual to be generated.

Error Messages

If you've made an error in your PsiDataSrc() function or in your data file, your custom visual will display an error such as "Inconsistent Data source definition PSI could not parse Excel formulas" or "Invalid binding source definition PSI could not parse Excel formulas".

If your custom visual is displaying either of these errors or a different error, first, check the PsiDataSrc() function in your Excel model to make sure that you have not mixed up the order of your index sets. In addition, check your data to make sure that it is 1. Contained within an Excel table, 2. That the column headings match the index sets defined in PsiDataSrc() and 3. That your model parameters match the elements in your data table.

For more information on an error message displayed in your custom visual, contact Frontline Systems Support Team by emailing support@solver.com or by calling 888-831-0333.

Creating Custom Extensions in Tableau

Introduction

Analytic Solver includes the ability to turn your Excel-based optimization or simulation model into a **Tableau Dashboard Extension**, with just a few mouse clicks! This is quite similar to the ability to create Power BI Custom Visuals as discussed in the previous chapter. *This feature is only compatible with versions of Tableau version 2018.2 or later.*

To start, simply select rows or columns of data to serve as changeable parameters, then choose **Deploy Model – Tableau – Managed Model**, and save the file created byAnalytic Solver. In Tableau, you'll see the newly-created file under **Extensions** on the left side of the dashboard, where you can drag it onto your dashboard. You'll be prompted to match the parameters your model needs with data in Tableau. What you get isn't just a chart – it's your *full optimization or simulation model,* ready to accept Tableau data**, run on demand** (using our **RASON** server), and display visual results in Tableau!

You can also use **Deploy Model – Tableau – Embedded Model** to embed your optimization or simulation model into a Tableau Extension. However, your extension will not allow Input Functions and is not recommended if your model will change at any time.

Notes:

- Analytic Solver Upgrade and Solver Academy licenses do not support Deploy Model functionality. This button will be disabled on the Analytic Solver ribbon.

Installing Tableau

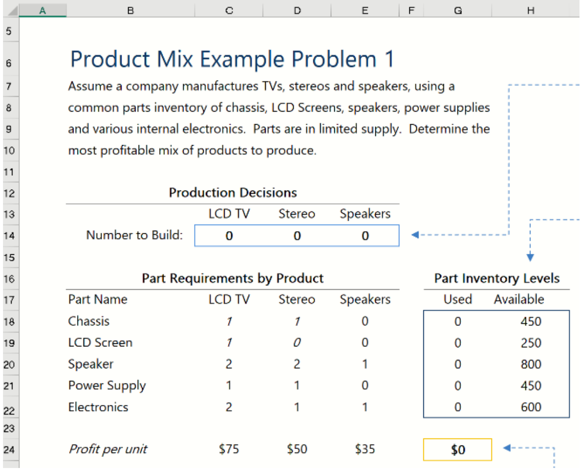
Tableau is an industry leading desktop and cloud-based interactive business intelligence tool that centers on data visualization, dashboarding and data discovery. In past versions of Analytic Solver, users have only been able to upload their model parameters (or their model data) to Tableau after solving an optimization or simulation model. However, starting with V2018, Analytic Solver now includes the ability to turn your Excel based optimization or simulation model into a Tableau Extension; your full optimization or simulation model, ready to accept Tableau data, run on demand on the web (using our RASON server) and display visual results in Tableau.

In order to use this new feature in Analytic Solver, you must first install Tableau V2018.2 or later or open the free cloud based version of Tableau. For more information on this business intelligence tool, see the following website: https://www.tableau.com/

Solving Optimization Models in Tableau

In this section, we will create a custom Tableau Extension using the Product Mix example model.

Open the ProductMix(Opt).xlsx model that was previously discussed in the Conventional Optimization chapter by clicking Help – Example Models - Optimization Examples on the ribbon, clicking Optimization Examples and then the Product Mix link. Recall this example model determines the optimal mix of products that a company should produce in order to maximize profits.



Product Mix Example Problem 1

Assume a company manufactures TVs, stereos and speakers, using a common parts inventory of chassis, LCD Screens, speakers, power supplies and various internal electronics. Parts are in limited supply. Determine the most profitable mix of products to produce.

Production Decisions

	LCD TV	Stereo	Speakers
Number to Build:	0	0	0

Part Requirements by Product				Part Inventory Levels	
Part Name	LCD TV	Stereo	Speakers	Used	Available
Chassis	1	1	0	0	450
LCD Screen	1	0	0	0	250
Speaker	2	2	1	0	800
Power Supply	1	1	0	0	450
Electronics	2	1	1	0	600
Profit per unit	$75	$50	$35	$0	

The optimization model is included in the Model tab in the Solver Task Pane. This model written in algebraic form is below.

Maximize: Total Profit = $75x1 + $50x2 + $35x3

Subject To:

Chassis: 1x1 + 1x2 + 0x3 <= 450

LCD Screen: 1x1 + 0x2 + 0x3 <= 250

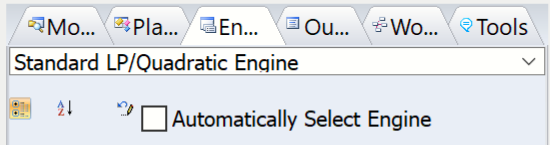
Speaker: 2x1 + 2x2 + 1x3 <= 800

Power Supply: 1x1 + 1x2 + 0x3 <= 450

Electronics: 2x2 + 1x2 + 1x3 <= 600

x1, x2, x3 > 0

The engine selected to solve this model is the Standard LP/Quadratic Engine as shown on the Engine tab in the Solver Task Pane. This is the engine that will be used to solve the model within Tableau.



Standard LP/Quadratic Engine

☐ Automatically Select Engine

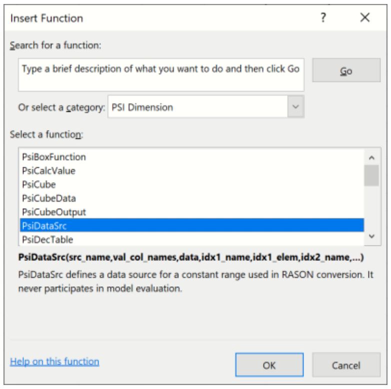
In past versions of Analytic Solver, you only had the option of uploading your final *solution values* (final variable, constraint and objective values) to Tableau. As a result, any time there was a change to the model, the model had to be solved in Excel first and then the data could be (again) uploaded to Tableau. All of this changed with V2018. Now you can change your model parameters and resolve your model *within* Tableau - without ever having to go back to Excel!

To start, we will use the PsiDataSrc() function to “flag” or specify the parameters in the Excel model that might possibly change in the future; for example,

1. the amount of inventory on hand for each part,
2. the profit point for each product
3. the number of parts that makeup each product.

Select a blank cell on the Product Mix Example 1 tab, say J6. Click Formulas – Insert Function to open the Insert Function dialog, then select PSI Dimension for Category. Under “Select a function”, highlight the function PsiDataSrc, and click OK.

Note: It is suggested that you delete the Constraints text box, in the example, in order to create room for the added PsiDataSrc() functions, described below.



Or select a category: PSI Dimension

PsiDataSrc(src_name,val_col_names,data,idx1_name,idx1_elem,idx2_name,...)

PsiDataSrc defines a data source for a constant range used in RASON conversion. It never participates in model evaluation.

In Analytic Solver Desktop, you can also simply type the formula directly into the cell.

PsiDataSrc() Arguments

The Function Arguments dialog opens displaying 5 arguments: Src_name, Val_col_names, Data, Idx1_name and Idx1_elem. We will use this dialog to create a PsiDataSrc() in cell J6 that will allow us the ability to change the inventory values and resolve the Product Mix example model not in Excel, but within Tableau.

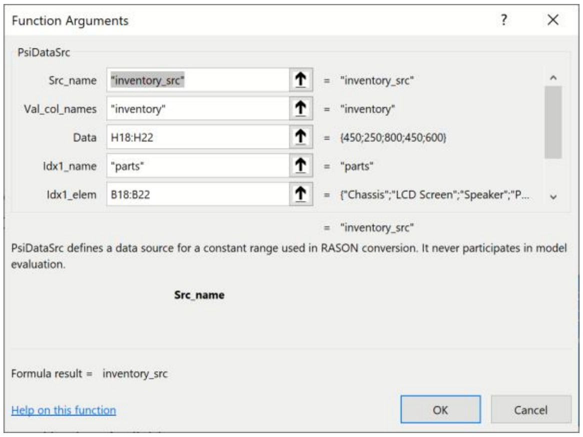
Src_name – This argument names the text file, created by the Extension, which is submitted, along with the RASON model, to the RASON server. For this example, type: inventory_src.

Val_col_names – The available data in Tableau must be mapped to the data source we create in the Excel workbook. This field provides the name of the data repository that will appear in Tableau. For this example, type: inventory. *Note: Names passed to Val_col_names are case sensitive.*

Data – Enter the Excel range where the data is located on the spreadsheet for this argument. For this example, use your mouse to click the upward pointing arrow to the right of the Data argument and select cells H18:H22. You can also simply type this range into the edit box.

Idx1_name – Enter the name of the (1st) index set for the data specified in the Data argument. In this example, our Index set is simply the inventory available for each Part. Therefore, enter “parts” for this argument. *Note: Index set names are case sensitive.*

Idx1_elem – Enter the Excel range containing the data (or elements) for the first index set. For this example, either type or select cells: B18:B22



Src_name "inventory_src" = "inventory_src"
Val_col_names "inventory" = "inventory"
Data H18:H22 = {450;250;800;450;600}
Idx1_name "parts" = "parts"
Idx1_elem B18:B22 = {"Chassis";"LCD Screen";"Speaker";"P...
= "inventory_src"

PsiDataSrc defines a data source for a constant range used in RASON conversion. It never participates in model evaluation.

Src_name

Formula result = inventory_src

Since this index set is 1-dimensional, nothing further is needed and we can click OK to enter the PsiDataSrc() function into cell J6.

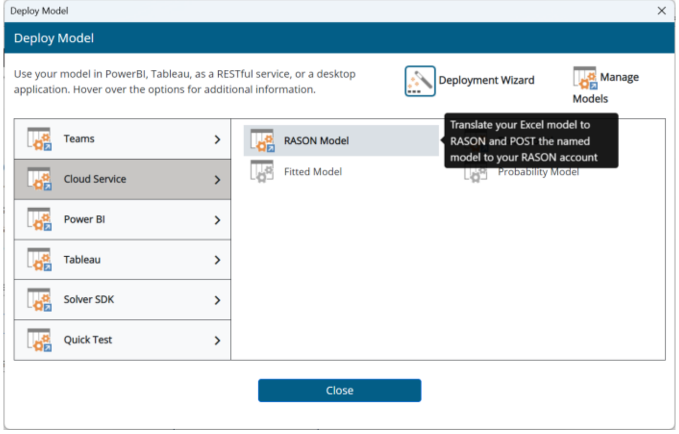
In Analytic Solver Cloud, you can type the formula directly into cell J6.

```
=PsiDataSrc("inventory_src", "inventory", H18:H22,
"parts", B18:B22)
```

Afterwards, cell J6 will display the value given to the Src_name argument, inventory_src. Save the Excel model.

Deploying the Model to RASON

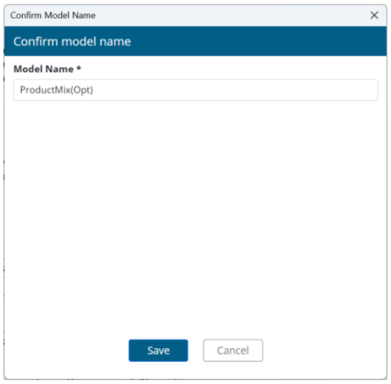
In order to deploy the model to the RASON Cloud Service, click Deploy Model – Cloud Service – RASON Model.



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

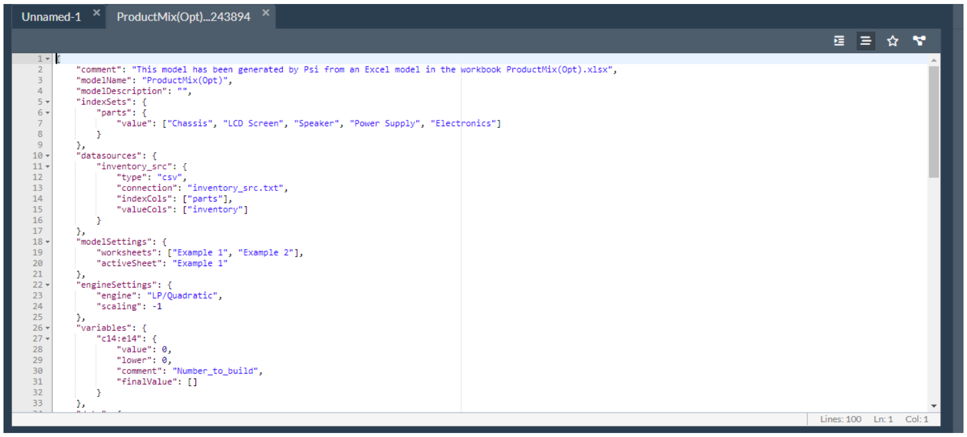
Translate your Excel model to RASON and POST the named model to your RASON account

The next screen will ask to confirm the model name. Click Save.



Model Name *: ProductMix(Opt)

A browser will open displaying the Editor tab at to www.RASON.com. Notice that the ProductMix(Opt) tab displays the algebraic representation of the Excel model written in Frontline's RASON Modeling language.

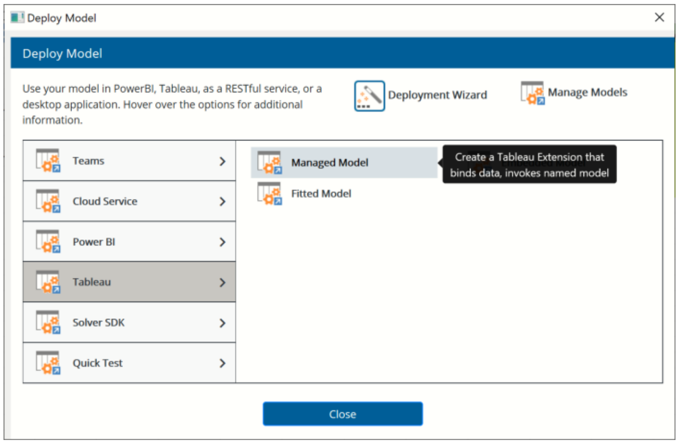


```
{
    "comment": "This model has been generated by Psi from an Excel model in the workbook ProductMix(Opt).xlsx",
    "modelName": "ProductMix(Opt)",
    "modelDescription": "",
    "indexSets": {
        "parts": {
            "value": ["Chassis", "LCD Screen", "Speaker", "Power Supply", "Electronics"]
        }
    },
    "datasources": {
        "inventory_src": {
            "type": "csv",
            "connection": "inventory_src.txt",
            "indexCols": ["parts"],
            "valueCols": ["inventory"]
        }
    },
    "modelSettings": {
        "worksheets": ["Example 1", "Example 2"],
        "activeSheet": "Example 1"
    },
    "engineSettings": {
        "engine": "LP/Quadratic",
        "scaling": -1
    },
    "variables": {
        "c14:e14": {
            "value": 0,
            "lower": 0,
            "comment": "Number_to_build",
            "finalValue": []
        }
    },
```

Deploying the Tableau Extension

Afterwards, go back to Excel and click Deploy Model on the ribbon, then select Tableau – Managed Model.

Note that it's also possible to embed your model into Tableau using Deploy Model -- Tableau – Embedded Model, however, if your model will change in any way, it's recommended to use Managed Model. For more formation on creating an embedded model, see the previous Deploying Your Model chapter.



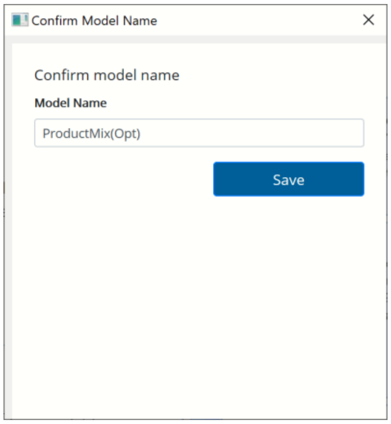
Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Create a Tableau Extension that binds data, invokes named model

Click Save to confirm the model name as "ProductMix(Opt)".

Model Versions

To create a new version of the model, enter a new name such as "ProductMix(Opt)1". You can manage your model versions by clicking the Manage Models icon on the Deploy Model dialog. For more information on this dialog, see the previous Deploying Your Model chapter.



Confirm Model Name

Confirm model name

Model Name: ProductMix(Opt)

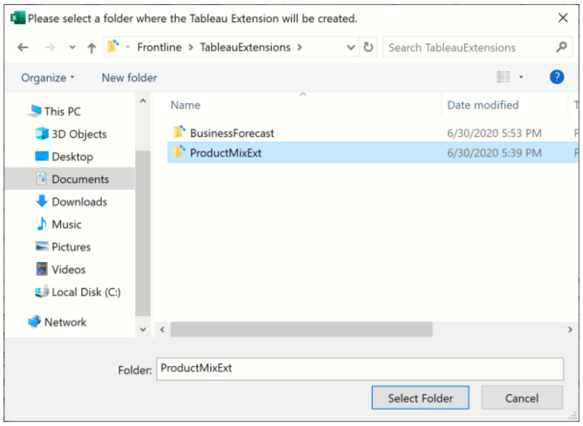
There is no need to solve the model first unless a change has been made to an already existing RASON model. If a change has been made to the workbook after a RASON model has been created, then you MUST either solve or diagnose the Excel model in order for the changes to be reflected in the new RASON model.

Note: When creating a Tableau extension through desktop Analytic Solver, model elements (objective function and all variables and constraints) may appear across multiple worksheets within the same Excel workbook.

Immediately, the Save As dialog appears. Enter a file name in the location of your choice and then click Save.

- Select a folder to save the Tableau extension files to, and then click Save. For example, the screenshot below displays a folder name of "ProductMixExt".

  Analytic Solver Desktop



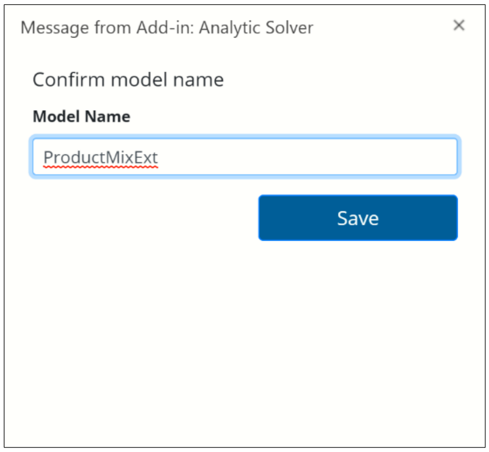
Please select a folder where the Tableau Extension will be created.

Frontline > TableauExtensions

Name	Date modified
BusinessForecast	6/30/2020 5:53 PM
ProductMixExt	6/30/2020 5:39 PM

Folder: ProductMixExt

- In Analytic Solver Cloud first, you will receive the following dialog. Type the filename of the folder, for example, TableauExtensions.

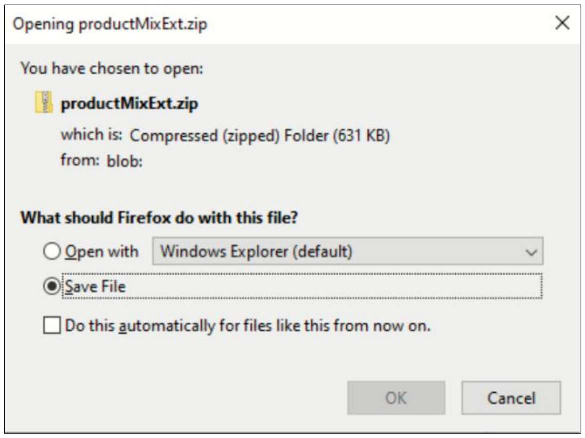


Message from Add-in: Analytic Solver

Confirm model name

Model Name: ProductMixExt

After the file is downloaded, you'll be asked if you'd like to open or save the file, choose Save As, then save the file in the desired location.



Opening productMixExt.zip

You have chosen to open:

productMixExt.zip

which is: Compressed (zipped) Folder (631 KB)

from: blob:

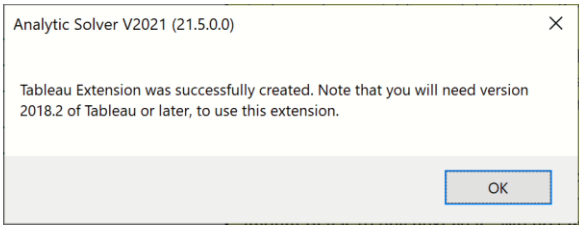
What should Firefox do with this file?

○ Open with Windows Explorer (default)

● Save File

☐ Do this automatically for files like this from now on.

Click Select Folder or Save. At this point your Excel model is translated into the **RASON**® modeling language, and your Tableau extension is created. After a moment, the following message appears.



Analytic Solver V2021 (21.5.0.0)

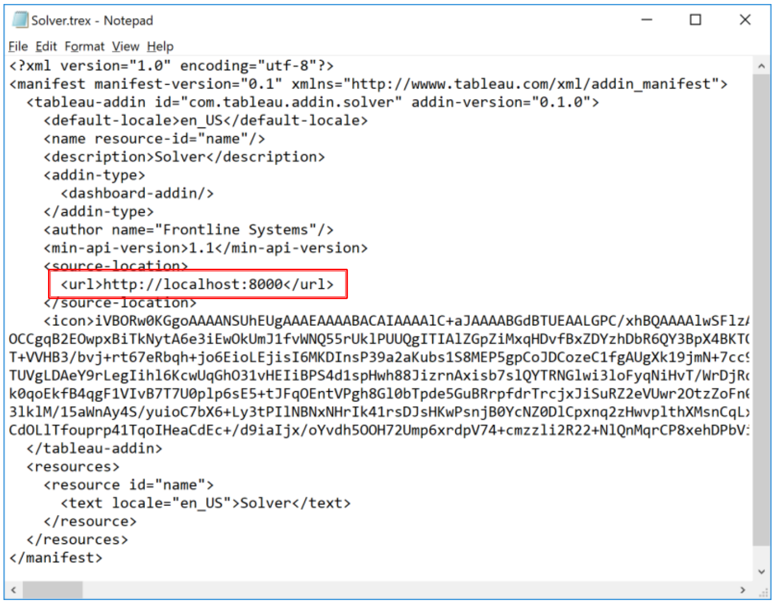
Tableau Extension was successfully created. Note that you will need version 2018.2 of Tableau or later, to use this extension.

Viewing the Tableau Extension File

Open File Explorer and browse to the folder where you saved the Tableau extension. Within that folder will be an Extensions folder. Click that folder and open the Solver.trex file using Notepad.

This file contains information such as the extensions name (or how it will appear in Tableau) and the url where the extension is hosted. To publish the html file to a website, simply replace http://localhost:8000 with the desired url.

For this example, you can leave as http://localhost:8000.

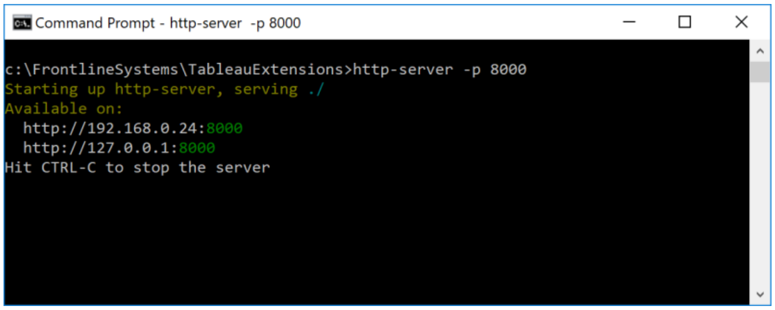


```
<?xml version="1.0" encoding="utf-8"?>
<manifest manifest-version="0.1" xmlns="http://wwww.tableau.com/xml/addin_manifest">
  <tableau-addin id="com.tableau.addin.solver" addin-version="0.1.0">
    <default-locale>en_US</default-locale>
    <name resource-id="name"/>
    <description>Solver</description>
    <addin-type>
      <dashboard-addin/>
    </addin-type>
    <author name="Frontline Systems"/>
    <min-api-version>1.1</min-api-version>
    <source-location>
      <url>http://localhost:8000</url>
    </source-location>
    <icon>iVBORw0KGgoAAAANSUhEUgAAAEAAAABACAIAAAAlC+aJAAAABGdBTUEAALGPC/xhBQAAAAlwSFlz
OCCgqB2EOwpxBiTkNytA6e3iEwOkUmJ1fvWNQ55rUklPUUQgITIAlZGpZiMxqHDvfBxZDYzhDbR6QY3BpX4BKT
T+VVHB3/bvj+rt67eRbqh+jo6EioLEjisI6MKDInsP39a2aKubs1S8MEP5gpCoJDCozeC1fgAUgXk19jmN+7cc
TUVgLDAeY9rLegIihl6KcwUqGhO31vHEIiBPS4d1spHwh88JizrnAxisb7slQYTRNGlwi3loFyqNiHvT/WrDjR
k0qoEkfB4qgF1VIvB7T7U0plp6sE5+tJFqOEntVPgh8Gl0bTpde5GuBRrpfdrTrcjxJiSuRZ2eVUwr2OtzZoFn
3lklM/15aWnAy4S/yuioC7bX6+Ly3tPIlNBNxNHrIk41rsDJsHKwPsnjB0YcNZ0DlCpxnq2zHwvplthXMsnCqL
CdOLlTfouprp41TqoIHeaCdEc+/d9iaIjx/oYvdh5OOH72Ump6xrdpV74+cmzzli2R22+NlQnMqrCP8xehDPbV
  </tableau-addin>
  <resources>
    <resource id="name">
      <text locale="en_US">Solver</text>
    </resource>
  </resources>
</manifest>
```

Starting up the Server

Since Tableau extensions are simply web pages, we will first need to start up a web server to serve our content. For this example, we will serve up the webpage to the default location. To do so, open a command prompt, navigate to the root of the extensions repository and run “http-server -p 8000”.

Note: If you do not have http-server installed on your machine, you can install using “npm install -g http-server”.



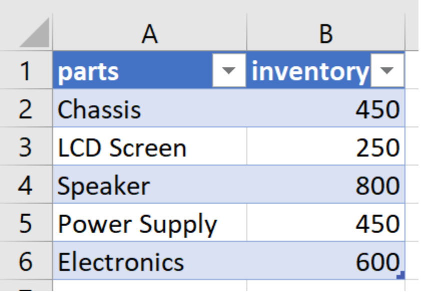
```
c:\FrontlineSystems\TableauExtensions>http-server -p 8000
Starting up http-server, serving ./
Available on:
  http://192.168.0.24:8000
  http://127.0.0.1:8000
Hit CTRL-C to stop the server
```

This command starts up a simple http server listening on port 8000.

Creating a Data File

In order to provide the data for the Product Mix example file to Tableau, we must create a data file. Within this file, we will pass the inventory levels by part. This data file can be of any format supported by Tableau such as a CSV file, a text file, a database, etc. In this example, we will use an Excel workbook to hold the data. For more information on the types of files supported by Tableau, please see the Tableau documentation at: http://onlinehelp.tableau.com/current/pro/desktop/en-us/default.html

Open a blank worksheet and create a table with two columns. List the parts by name in the first column using the column heading “parts” and the inventory levels in the second column using the column heading “inventory”, as shown in the screenshot below. You can open a completed data file by clicking Help – Example Models – Optimization Examples – Product Mix **Data** for Power BI – Tableau.



	A	B
1	parts	inventory
2	Chassis	450
3	LCD Screen	250
4	Speaker	800
5	Power Supply	450
6	Electronics	600

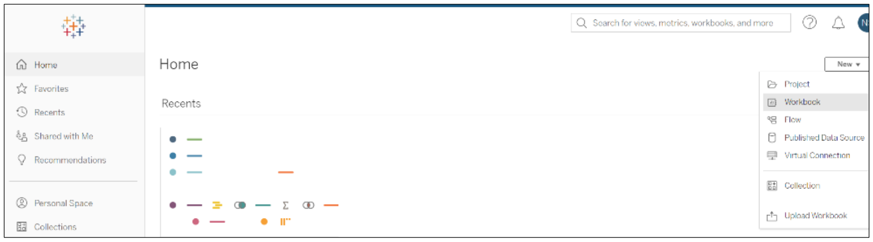
Note: Column headings ***must be identical*** *to the names of the Index Sets passed to the PsiDataSrc() function. Make sure there are no spaces before or after column headings or row elements.*

Recall that we labeled the IndexSet as “parts” (4th argument of PsiDataSrc) and Val_col_names as “inventory” (2nd argument of PsiDataSrc) so the title for column A must be “parts” and the title for column B must be “inventory”. Now save the workbook to a desired location. In this example, the data file is named Data1.xlsx.

Opening Extension within Tableau

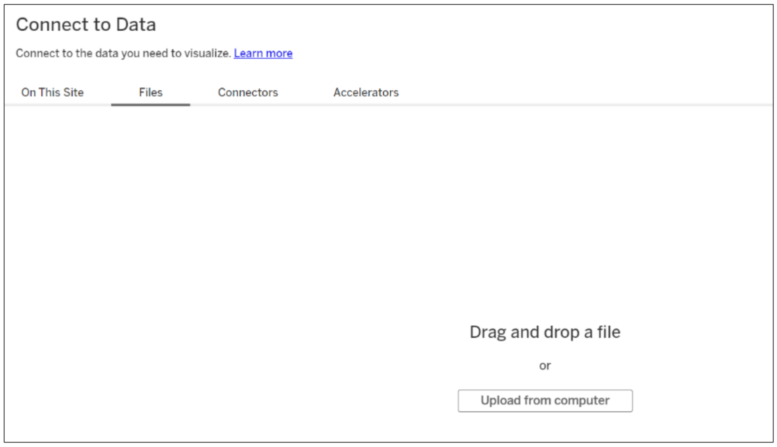
Open either desktop, server or cloud-based Tableau. The screenshot below depicts the opening screen of Tableau Cloud (as of Fall 2023).

Click New - Worksheet on the Home screen of Tableau.



Note: Tableau screens may change in future releases.

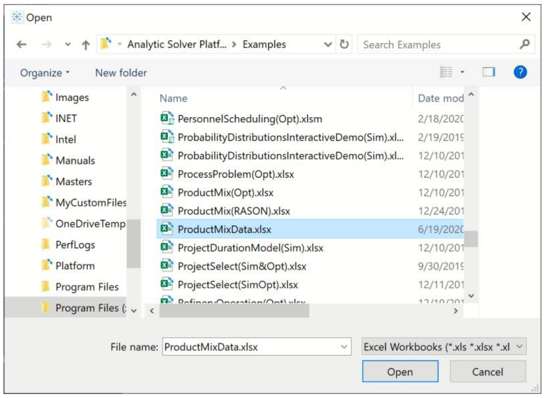
Click Files and then Upload from computer.



Connect to Data

Connect to the data you need to visualize. Learn more

An Open dialog opens.



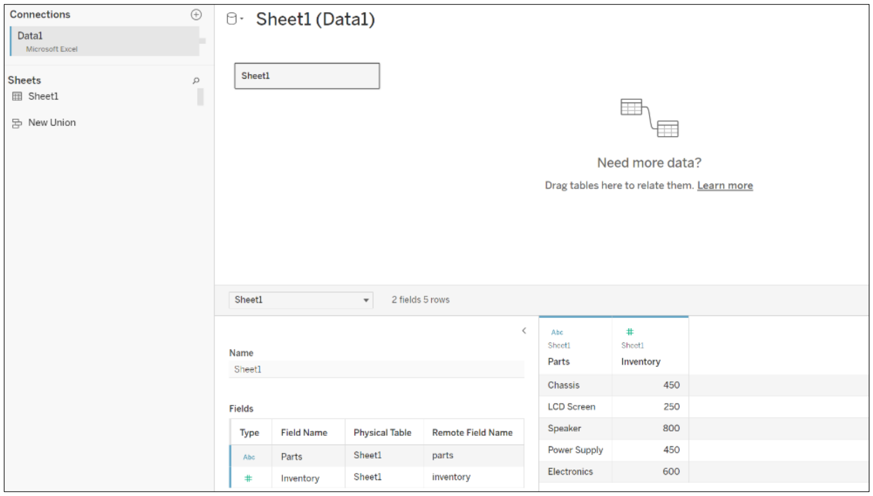
Name	Date mod
PersonnelScheduling(Opt).xlsm	2/18/2020
ProbabilityDistributionsInteractiveDemo(Sim).xl...	2/19/2019
ProbabilityDistributionsInteractiveDemo(Sim).xl...	12/10/201
ProcessProblem(Opt).xlsx	12/10/201
ProductMix(Opt).xlsx	12/10/201
ProductMix(RASON).xlsx	12/24/201
ProductMixData.xlsx	6/19/2020
ProjectDurationModel(Sim).xlsx	12/10/201
ProjectSelect(Sim&Opt).xlsx	9/30/2019
ProjectSelect(SimOpt).xlsx	12/11/201

File name: ProductMixData.xlsx

Excel Workbooks (*.xls *.xlsx *.xl

Browse to the location of your data file (C:\ProgramData\Frontline Systems\ Examples if using the completed data file) and click Open.

The data is imported to Tableau. Check the Data Source tab in Tableau, as shown in the screenshot below.



Name: Sheet1

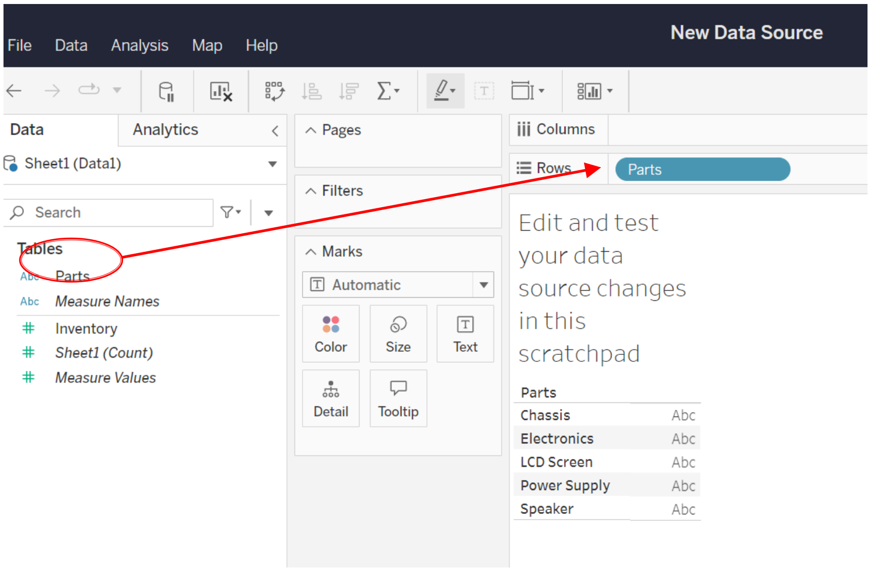
Fields

Type	Field Name	Physical Table	Remote Field Name
Abc	Parts	Sheet1	parts
#	Inventory	Sheet1	inventory

Parts	Inventory
Chassis	450
LCD Screen	250
Speaker	800
Power Supply	450
Electronics	600

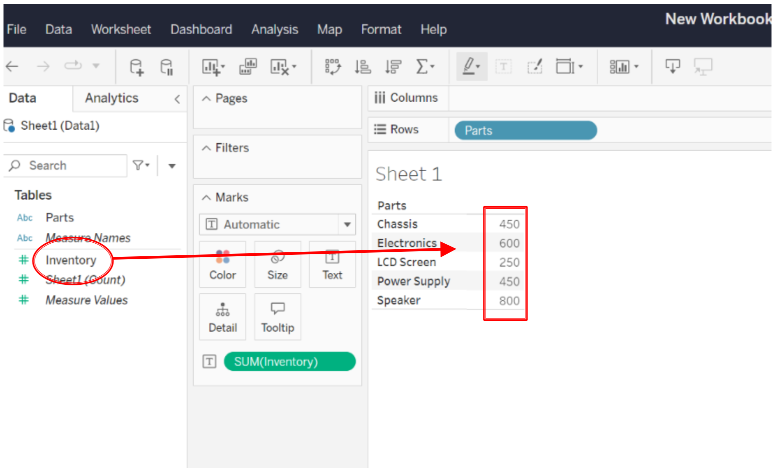
Click the Sheet 1 tab, at the bottom of the Tableau screen.

Drag Parts to Rows…



Parts	
Chassis	Abc
Electronics	Abc
LCD Screen	Abc
Power Supply	Abc
Speaker	Abc

… and then drag Inventory to the column in the Sheet 1 table.



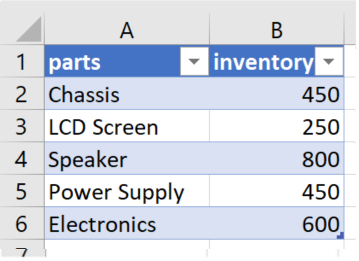
Sheet 1

Parts	
Chassis	450
Electronics	600
LCD Screen	250
Power Supply	450
Speaker	800

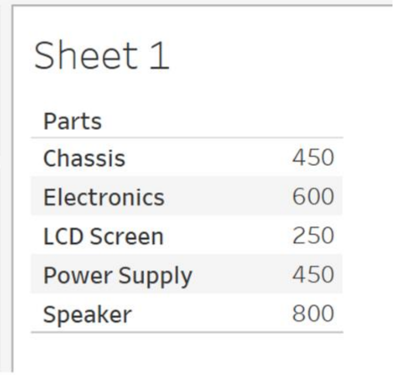
Hover over SUM(Inventory), click the down arrow that appears to the right, then select Dimensions from the menu.

Note that the information in ProductMixData.xlsx matches the data in Sheet 1.

ProductMixDataSource.xlsx **Tableau Sheet 1**



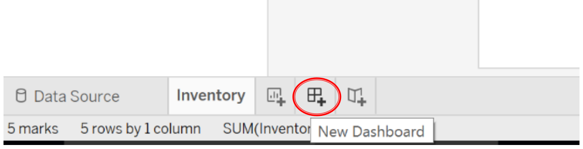
	A	B
1	parts	inventory
2	Chassis	450
3	LCD Screen	250
4	Speaker	800
5	Power Supply	450
6	Electronics	600



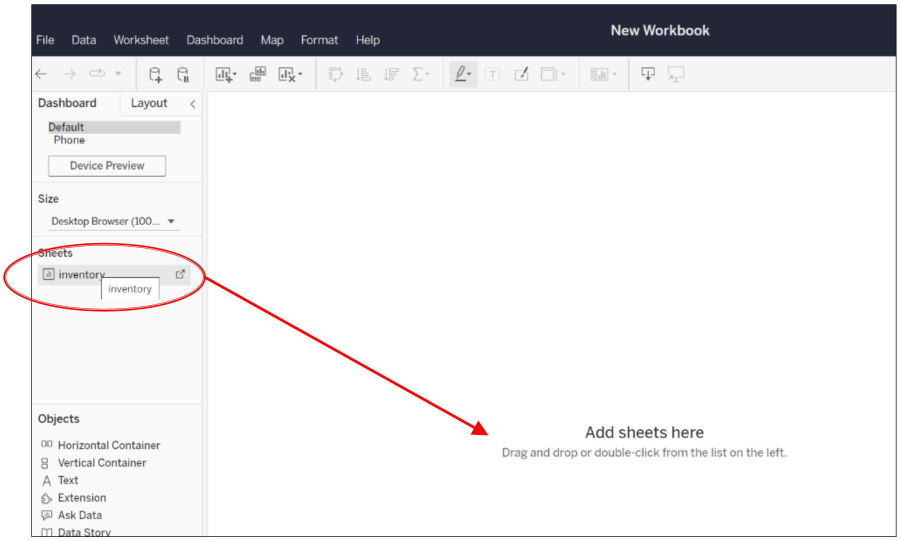
Sheet 1

Parts	
Chassis	450
Electronics	600
LCD Screen	250
Power Supply	450
Speaker	800

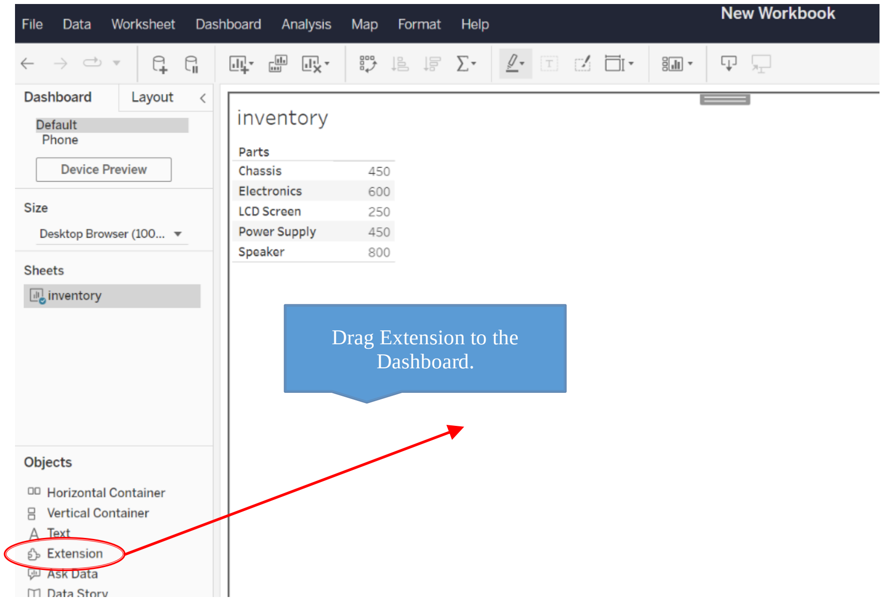
Right click the Sheet 1 tab and rename the worksheet to “inventory”, then click the New Dashboard icon at the bottom of Tableau to open a new dashboard.



First, drag the Inventory sheet onto the dashboard canvas.



Then drag the Extension icon, under Objects, onto the canvas.

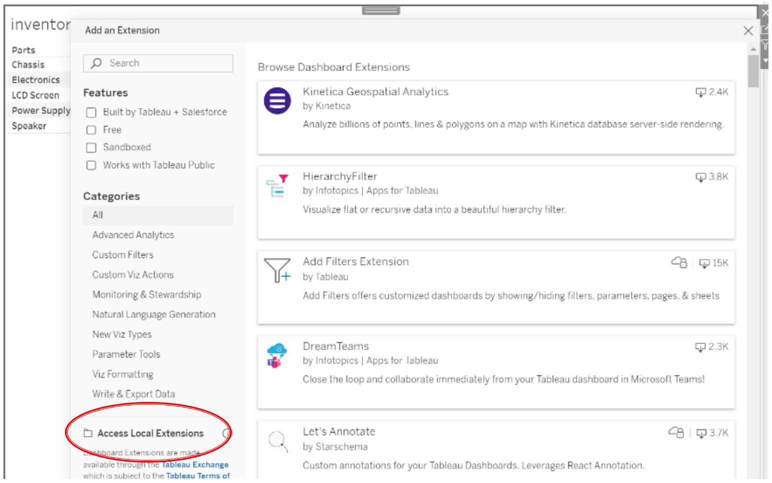


inventory

Parts	
Chassis	450
Electronics	600
LCD Screen	250
Power Supply	450
Speaker	800

Drag Extension to the Dashboard.

On the Add an Extension dialog, click the folder icon next to Access Local Extensions.



Analyze billions of points, lines & polygons on a map with Kinetica database server-side rendering.

Visualize flat or recursive data into a beautiful hierarchy filter.

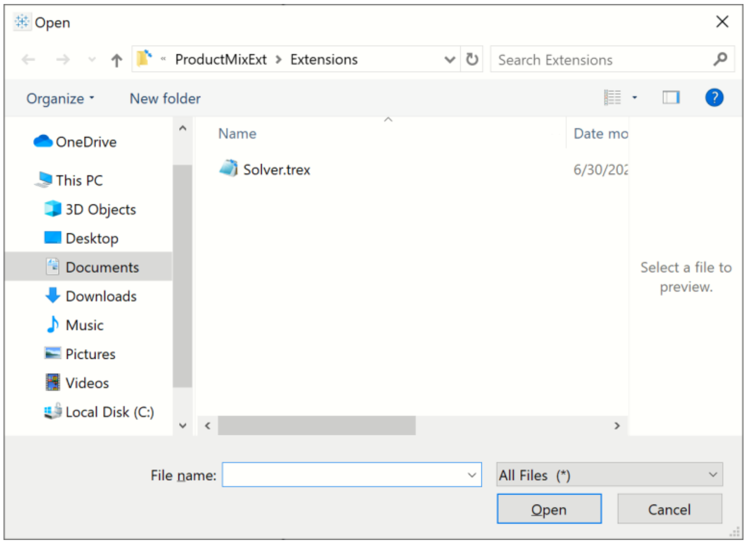
Add Filters offers customized dashboards by showing/hiding filters, parameters, pages, & sheets

Close the loop and collaborate immediately from your Tableau dashboard in Microsoft Teams!

Custom annotations for your Tableau Dashboards. Leverages React Annotation.

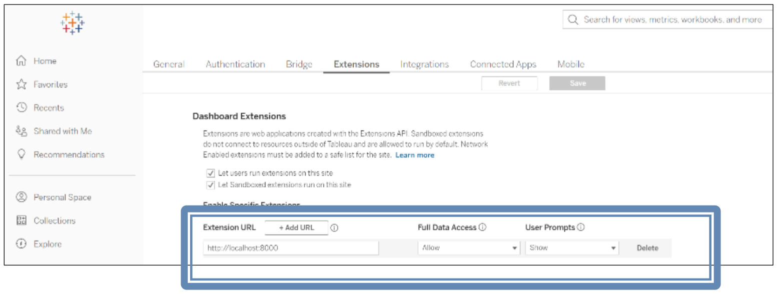
Dashboard Extensions are made available through the Tableau Exchange which is subject to the Tableau Terms of

…browse to the location of the Tableau Extension.



(In this example, the Tableau Extension was saved to the ProductMixExt folder. Drill down to the Extensions folder to open the Solver.trex file.) Click Open.

Note: You must add the Extension URL to Tableau Settings – Extension. Otherwise you will get an error that this extension is not allowed.

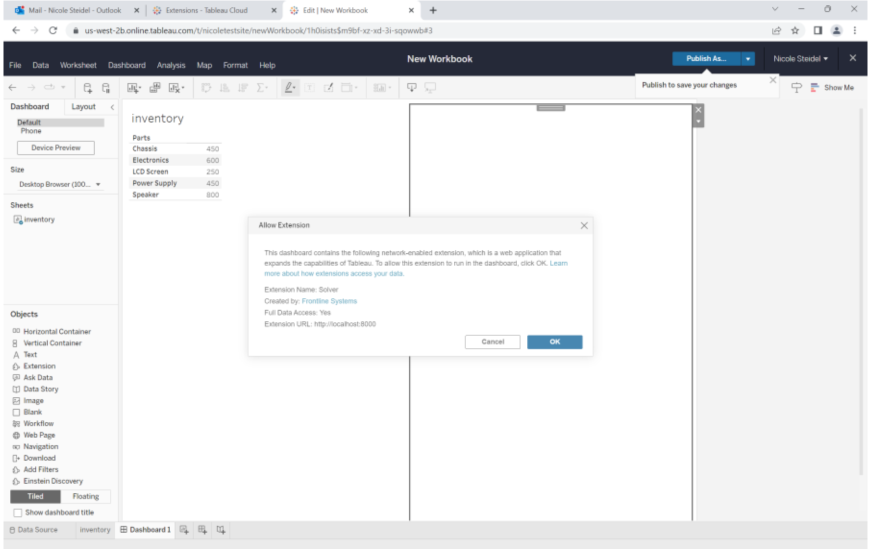


Extensions are web applications created with the Extensions API. Sandboxed extensions do not connect to resources outside of Tableau and are allowed to run by default. Network Enabled extensions must be added to a safe list for the site. Learn more

☑ Let users run extensions on this site
☑ Let Sandboxed extensions run on this site

Extension URL: http://localhost:8000
Full Data Access: Allow
User Prompts: Show

Click OK to allow the extension to be added to the dashboard.



inventory

Parts	
Chassis	450
Electronics	600
LCD Screen	250
Power Supply	450
Speaker	800

Allow Extension

This dashboard contains the following network-enabled extension, which is a web application that expands the capabilities of Tableau. To allow this extension to run in the dashboard, click OK. Learn more about how extensions access your data.

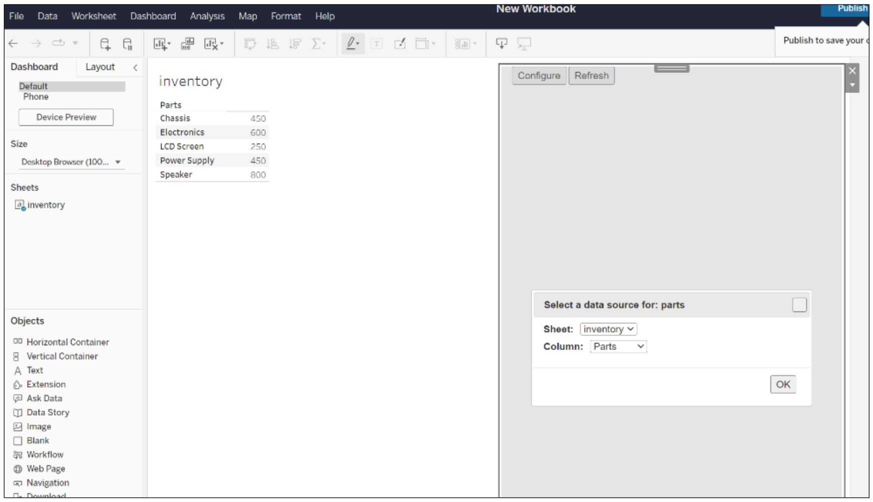
Extension Name: Solver

Created by: Frontline Systems

Full Data Access: Yes

Extension URL: http://localhost:8000

A dialog opens asking to select a data source for “parts”. Confirm that “inventory” is selected for “Sheet” and “Parts” is selected for “Column”. Note that these are the same names passed to the PsiDataSrc() function in the Excel Product Mix.xlsx workbook. Click OK.

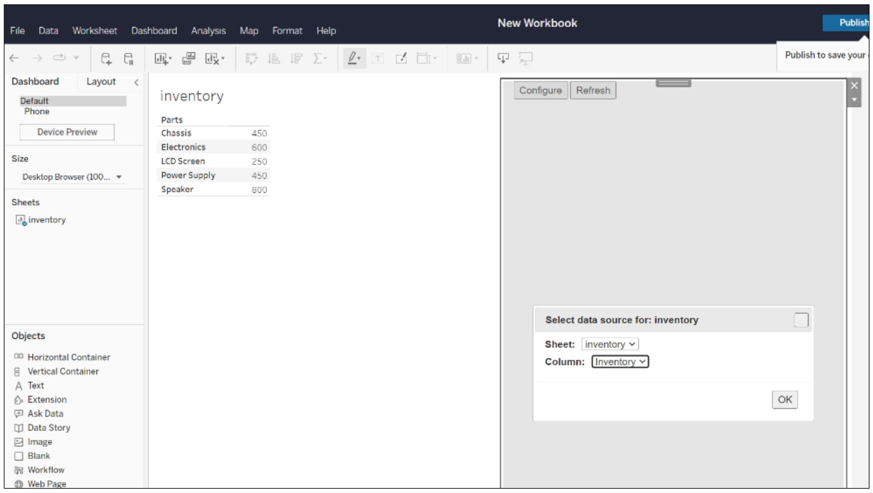


Select a data source for: parts

Sheet: inventory

Column: Parts

On the *Select data source for: inventory* dialog, select “inventory” for “Sheet” and “Inventory” for “Column”. Then click OK.



inventory

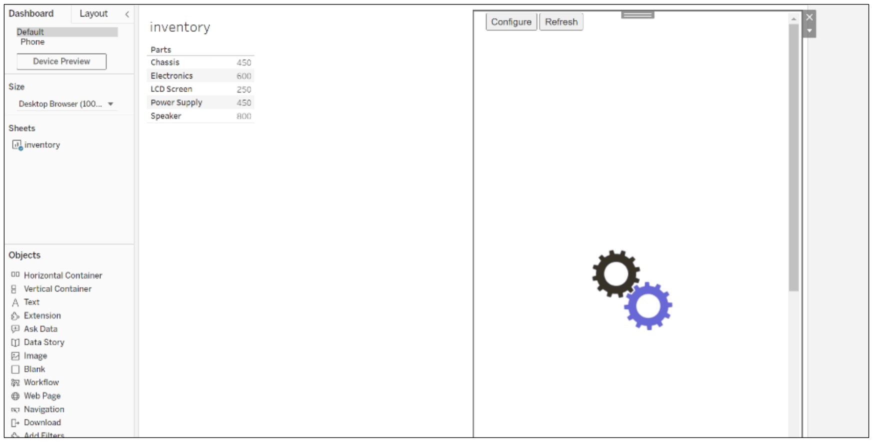
Parts	
Chassis	450
Electronics	600
LCD Screen	250
Power Supply	450
Speaker	800

Select data source for: inventory

Sheet: inventory

Column: Inventory

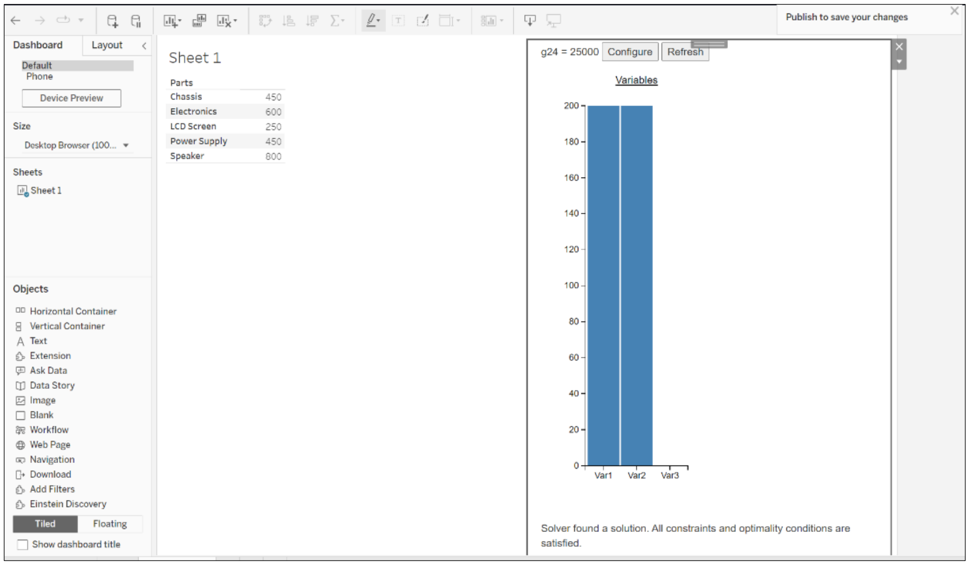
Immediately, two gears appear and start to turn while the model is sent to the RASON server, solved with the LP/Quadratic engine, and the final values are returned. Your screen should appear similar to the screenshot below.



inventory

Parts	
Chassis	450
Electronics	600
LCD Screen	250
Power Supply	450
Speaker	800

Once the RASON server has imported the final solution values into Tableau, Solver's result message, "Solver found a solution. All constraints and optimality conditions are satisfied", is displayed at the bottom of the Extension window. In the Variables chart, we see that the final variable values are: Var1 (LCD TV) equal to 200, Var2 (Stereo) equal to 200 and Var3 (Speakers) equal to 0. These variable values result in an objective function value equal to $25,000.



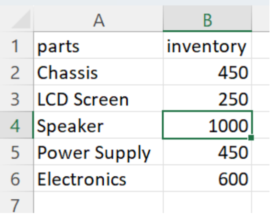
Parts	
Chassis	450
Electronics	600
LCD Screen	250
Power Supply	450
Speaker	800

Solver found a solution. All constraints and optimality conditions are satisfied.

Look, more inventory!

What happens if the number of speakers in inventory increases from 800 to 1000?

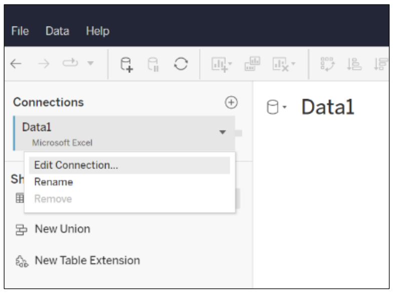
First, the inventory inside the Data1 worksheet must be changed. Notice that the inventory for Speakers has been changed to 1000.



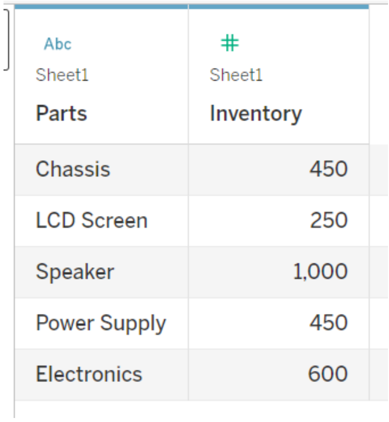
	A	B
1	parts	inventory
2	Chassis	450
3	LCD Screen	250
4	Speaker	1000
5	Power Supply	450
6	Electronics	600
7		

Secondly, the data source must be updated inside Tableau.

Click back to Tableau and the Data Source tab and refresh the data source by clicking the down arrow next to Data1 and selecting Edit Connection, then upload the new version of Data1.xlsx.



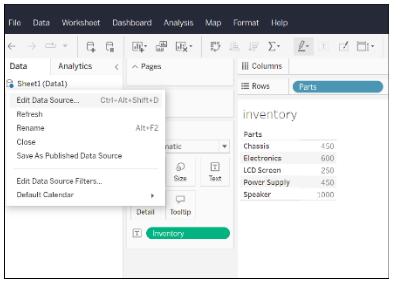
Notice that the Inventory for Speakers has changed to 1,000 in the Data Source window.



Abc Sheet1 Parts	# Sheet1 Inventory
Chassis	450
LCD Screen	250
Speaker	1,000
Power Supply	450
Electronics	600

The inventory table on the Inventory sheet will refresh once opened.

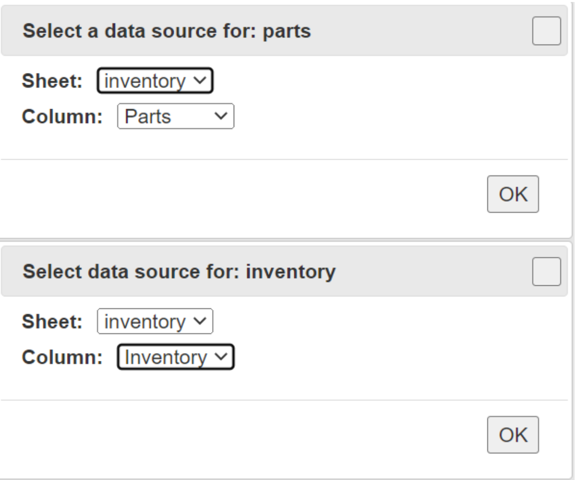
The sheet can be refreshed manually by clicking the down arrow next to Sheet 1 (Data1) and selecting Refresh.



inventory

Parts	
Chassis	450
Electronics	600
LCD Screen	250
Power Supply	450
Speaker	1000

Click the Dashboard tab to open the Dashboard. The Solver Extension will immediately ask to update the data fields. Simply select the Parts column for the parts data source and the Inventory column for the inventory data source, as shown in the screenshots below.



Select a data source for: parts ☐

Sheet: inventory

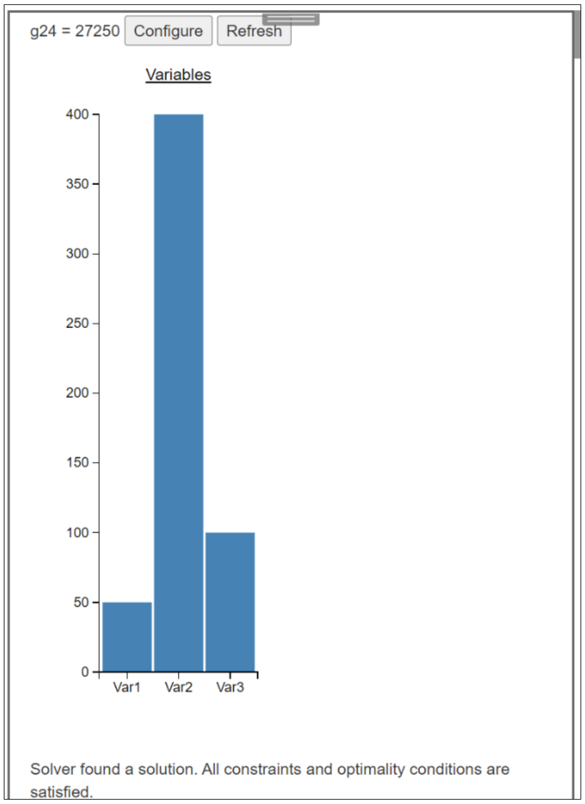
Column: Parts

Select data source for: inventory ☐

Sheet: inventory

Column: Inventory

Immediately, two gears appear and start to turn while the model is sent to the RASON server, solved with the LP/Quadratic engine, and the final values are returned. Notice that by increasing the number of speakers in inventory, our profit has increased by $2,250 ($27,250 = $25,000 (original solution) + $2,250).



Solver found a solution. All constraints and optimality conditions are satisfied.

Gaining Control Over Your Data

Now let's go back to the Excel Model and see what happens when we add three more PsiDataSrc() functions to:

1. Give us control over the number of parts required to manufacture a stereo
2. Allow us to increase the profit for stereos to $100.
3. Add variable names to the graphs.

First, we will use a 2nd PsiDataSrc() function to denote that new data (an increased selling price for Stereos) will be available in Tableau. Click back to the ProductMix(Opt) example model, select cell J7 and then click Formulas – Insert Function, select PSI Dimension for Category, select PsiDataSrc, then click OK.

Enter the following for the arguments appearing on the Function Arguments dialog.

Src_name – Enter "profit_src" to name the text file, created by the custom visual, which will be submitted, along with the RASON model, to the RASON server.

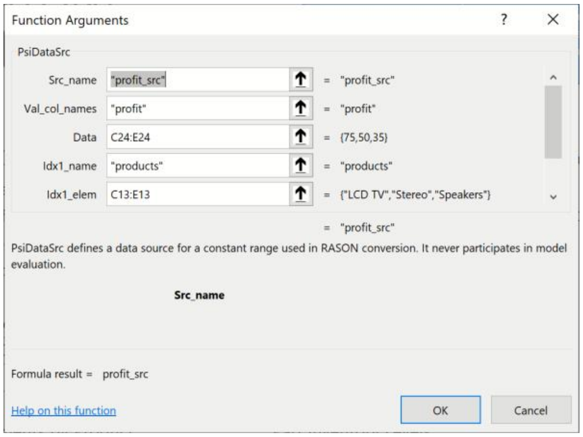
Val_col_names – Enter "profit" for the data repository name. *Note: Names passed to Val_col_names argument are case sensitive.*

Data – Enter the Excel range, C24:E24.

Idx1_name – Enter "products" which is the name of the 1st index set for the data specified in the Data argument. In this example, our Index set is simply the selling price for each product. *Note: Index set names are case sensitive.*

Idx1_elem – Enter the Excel range containing the data for the first index set, C13:E13.

Since this index set is 1-dimensional, nothing further is needed and we can click OK to enter the PsiDataSrc() function into cell J7. Afterwards, cell J7 will display the value given to the Src_name argument, profit_src.



Src_name "profit_src" = "profit_src"
Val_col_names "profit" = "profit"
Data C24:E24 = {75,50,35}
Idx1_name "products" = "products"
Idx1_elem C13:E13 = {"LCD TV","Stereo","Speakers"}

= "profit_src"

PsiDataSrc defines a data source for a constant range used in RASON conversion. It never participates in model evaluation.

Src_name

Formula result = profit_src

Help on this function

In Analytic Solver Cloud, you can type the formula directly into cell J7.

```
=PsiDataSrc("profit_src", "profit", C24:E24,
"products", C13:E13)
```

A 3rd PsiDataSrc() function will denote that new parts data (an increase in the number of speakers to be included in the building of a stereo) will be available in Tableau. This time, we will simply enter the formula directly into cell J8.

First type “=PsiDataSrc(“, then enter the arguments below.

Src_name – Enter “reqparts_src” to name the text file which will be submitted, along with the RASON model, to the RASON server.

Val_col_names – Enter “requiredparts” for the data repository name. *Note: Names passed here are case sensitive.*

Data – Enter the Excel range, C18:E22.

Idx1_name – Enter “parts” which is the name of the 1st index set for the data specified in the Data argument. In this example, there are two index sets. When data is 2-dimensional, we must enter the index set that describes the rows first. *Note: Index set names are case sensitive.*

Idx1_elem – Enter the Excel range containing the data for the first index set, B18:B22.

Idx2_name – Enter “products” which is the name of the 2nd index set for the data specified in the Data argument.

Idx2_elem – Enter the Excel range containing the data for the second index set, C13:E13. *Note: Index set names are case sensitive.*

Now we can type “)” to close the function, then press Enter to enter the PsiDataSrc() function into cell J8. Afterwards, cell J8 will display the value given to the Src_name argument, reqparts_src.

Your PsiDataSrc function should be similar to the screenshot below.

=PsiDataSrc("reqparts_src, "requiredparts", C18:E22, "parts", B18:B22, "products", C13:E13)

Finally, we will use a 4th PsiDataSrc() function to add the variable names to the chart and also supply initial variable values to the optimization model. Enter the formula directly into cell J9.

First type "=PsiDataSrc(", then enter the arguments below.

Src_name – Enter " vars_src" to name the text file which will be submitted, along with the RASON model, to the RASON server.

Val_col_names – Enter "initialvalues" for the data repository name. *Note: Names passed here are case sensitive.*

Data – Enter the Excel range, C14:E14.

Idx1_name – Enter "products" which is the name of the 1st index set for the data specified in the Data argument. In this example, there is just one set. *Note: Index set names are case sensitive.*

Idx1_elem – Enter the Excel range containing the data for the first, and only, index set, C13:E13.

Now we can type ")" to close the function, then press Enter to enter the PsiDataSrc() function into cell J9. Afterwards, cell J will display the value given to the Src_name argument, vars_src.

Your PsiDataSrc function should be similar to the screenshot below.

=PsiDataSrc("vars_src, "initialvalues", C14:E14, "products", C13:E13)

At this point, cell J6 displays "inventory_src", cell J7 displays "profit_src", cell J8 displays "reqparts_src" and cell J9 displays "vars_src".

Before we create a new Tableau Extension, we need to add two more functions to the worksheet: PsiFinalValue() and PsiSlackValue(). Adding these two functions will allow the final constraint values to be displayed in the Extension. PsiFinalValue will display the final values of the constraints and PsiSlackValue will display the right hand side of the constraints.

In cell K18 enter "=PsiFinalValue(G18:G22)".

In cell L18, enter: "=PsiSlackValue(G18:G22)".

Note that these formulas will both "spill" down the column.

Since we added the additional PsiDataSrc, PsiFinalValue and PsiSlackValue functions, you'll need to update the model on Rason Cloud Services. Save the Excel model and click Deploy Model – Cloud Service – Rason Model. You can reuse the previous model name. Then click Deploy Model – Tableau – Managed Model to create a new Tableau Extension.

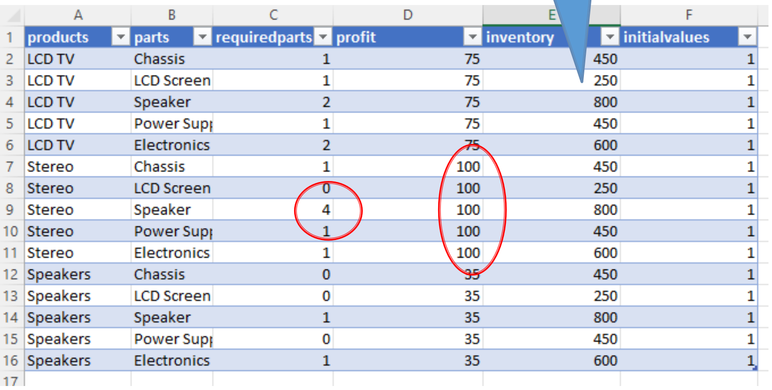
Now we are ready to add a new data table to the Data1.xlsx file.

Important Note: In Tableau, unlike when creating a Power BI custom visual, it is not necessary that all data be contained within the same Excel table or from within multiple tables with the same number of rows and columns. However, for ease of use, this example includes all data within the same Excel table.

In a new Excel sheet, create a table (Insert – Table) with 16 rows and 6 columns, then list the products in the first column using the column heading "products", the parts by name using the column heading "parts" in the 2nd column, the parts required per product (requiredparts) in the 3rd column, the selling price or "profit" per product in the 4th column, the part inventory for each product in the 5th column and the initial variable values in the last column (simply set to all

1’s). **Make sure to change the new sel** **ereos to $100 (cells D7:D11) and the increased number of** **eo to 4 (cell C9).**

Note: Speaker inventory has been reset to the original value, 80



Note that Speaker inventory has been reset to the original value, 800.

	A	B	C	D	E	F
1	products	parts	requiredparts	profit	inventory	initialvalues
2	LCD TV	Chassis	1	75	450	1
3	LCD TV	LCD Screen	1	75	250	1
4	LCD TV	Speaker	2	75	800	1
5	LCD TV	Power Supp	1	75	450	1
6	LCD TV	Electronics	2	75	600	1
7	Stereo	Chassis	1	100	450	1
8	Stereo	LCD Screen	0	100	250	1
9	Stereo	Speaker	4	100	800	1
10	Stereo	Power Supp	1	100	450	1
11	Stereo	Electronics	1	100	600	1
12	Speakers	Chassis	0	35	450	1
13	Speakers	LCD Screen	0	35	250	1
14	Speakers	Speaker	1	35	800	1
15	Speakers	Power Supp	0	35	450	1
16	Speakers	Electronics	1	35	600	1
17						

Notice that in order to list all of the required parts for each product, we must list each part and selling price for each part. In addition, each column heading matches the index sets used in the PsiDataSrc() functions. Rename the worksheet “AllData” then save the workbook. *If there are any leading or trailing spaces for the column headings or elements, Tableau will stop with an error.*

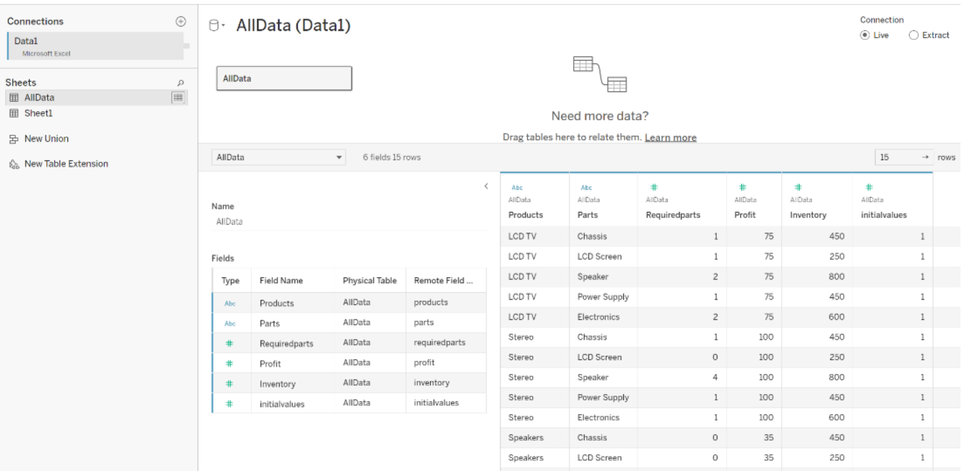
Note: Be sure to clear your browser’s cache before continuing.

Click back to Tableau and the Data Source tab and refresh the data source by

1. Clicking the down arrow next to Data1
2. Selecting Edit Connection and uploading Data1.xlsx.

You’ll now see two sheets under Data1.

Drag the AllData sheet to the window.

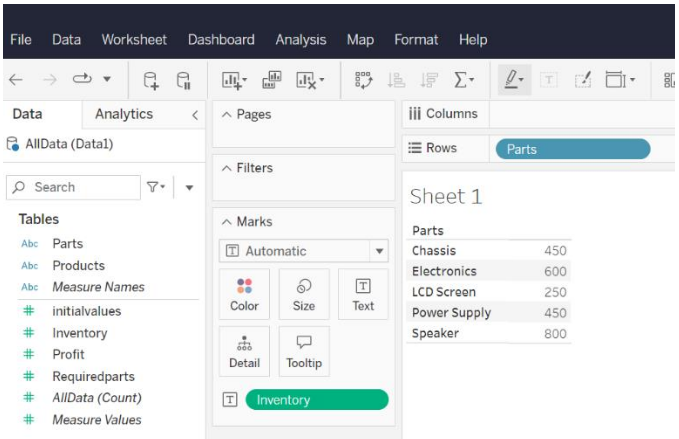


Type	Field Name	Physical Table	Remote Field ...
Abc	Products	AllData	products
Abc	Parts	AllData	parts
#	Requiredparts	AllData	requiredparts
#	Profit	AllData	profit
#	Inventory	AllData	inventory
#	initialvalues	AllData	initialvalues

Abc AllData Products	Abc AllData Parts	# AllData Requiredparts	# AllData Profit	# AllData Inventory	# AllData initialvalues
LCD TV	Chassis	1	75	450	1
LCD TV	LCD Screen	1	75	250	1
LCD TV	Speaker	2	75	800	1
LCD TV	Power Supply	1	75	450	1
LCD TV	Electronics	2	75	600	1
Stereo	Chassis	1	100	450	1
Stereo	LCD Screen	0	100	250	1
Stereo	Speaker	4	100	800	1
Stereo	Power Supply	1	100	450	1
Stereo	Electronics	1	100	600	1
Speakers	Chassis	0	35	450	1
Speakers	LCD Screen	0	35	250	1

Tableau displays information from the Excel table at the bottom of the window.

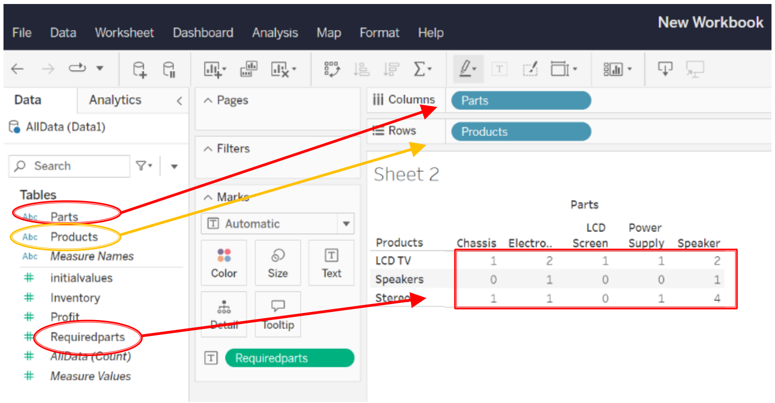
Click on the Inventory worksheet, the table automatically updates.



Parts	
Chassis	450
Electronics	600
LCD Screen	250
Power Supply	450
Speaker	800

Add a new worksheet and drag **“Parts” to “Columns”** and **“Products” to “Rows”**. Then drag “Requiredparts” to the table elements.

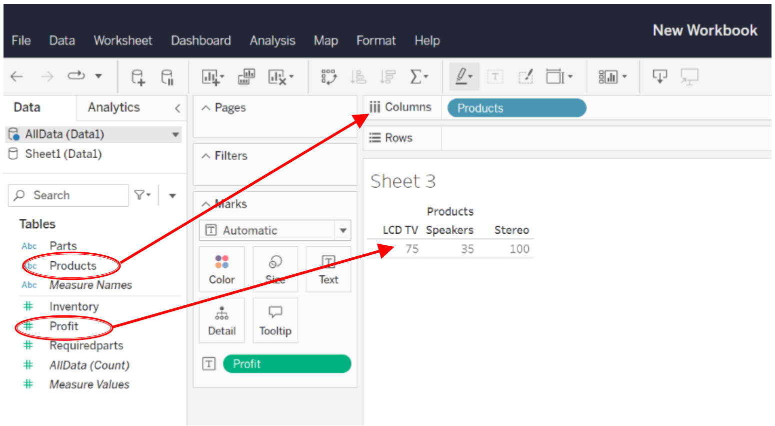
Hover over SUM(Required.., click the down arrow that appears to the right, and select Dimensions from the menu.



	Parts				
Products	Chassis	Electro..	LCD Screen	Power Supply	Speaker
LCD TV	1	2	1	1	2
Speakers	0	1	0	0	1
Stereo	1	1	0	1	4

Name the sheet “Required Parts” and then add a third sheet. In Sheet 3, drag “Products” to “Columns” and drag “Profit” to the column elements.

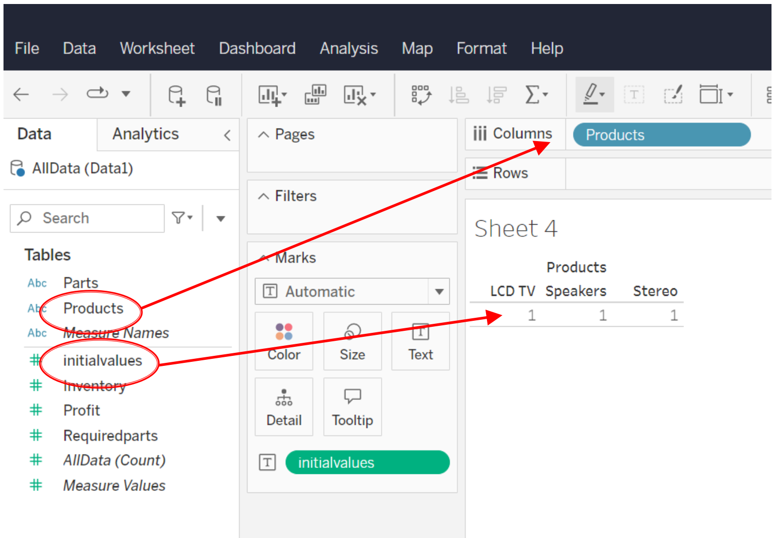
Hover over SUM(Profit) , click the down arrow that appears on the right then select Dimensions from the menu.



Products		
LCD TV	Speakers	Stereo
75	35	100

Add one last new worksheet and drag “Products” to “Columns”. Then drag “initialvalues” to the table elements.

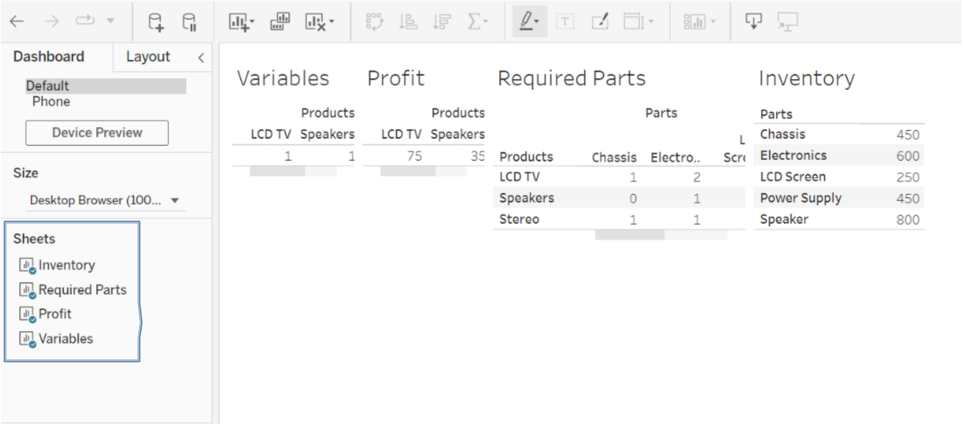
Hover over SUM(initialvalues), click the down arrow that appears to the right, and select Dimensions from the menu.



Sheet 4

Products		
LCD TV	Speakers	Stereo
1	1	1

Rename the sheet from Sheet 4 to Variables and then create a new dashboard. Drag all four sheets, Inventory, Required Parts, Profit and Variables, to the canvas.



Variables

Products	
LCD TV	Speakers
1	1

Profit

Products	
LCD TV	Speakers
75	35

Required Parts

	Parts		
Products	Chassis	Electro..	Scr
LCD TV	1	2	
Speakers	0	1	
Stereo	1	1	

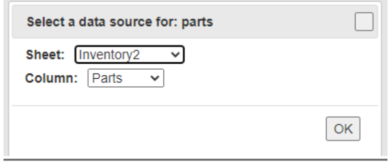
Inventory

Parts	
Chassis	450
Electronics	600
LCD Screen	250
Power Supply	450
Speaker	800

Then drag a new Extension to the canvas.

Open the Solver.trex file in …\TableauExtensions\Extensions.

At this point we will connect the dimensions created by the PsiDataSrc() function in Excel with the actual data in Tableau. A dialog appears asking to “Select a datasource for: parts”. Select “Inventory2” for Sheet and “Parts” for Column to match the “parts” dimension in the Excel sheet to the “Parts” dimension in the Tableau Inventory2 sheet.



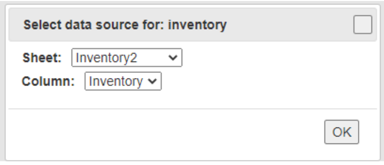
Select a data source for: parts ☐

Sheet: Inventory2

Column: Parts

Click OK.

On the “Select data source for: inventory” dialog, select “Inventory2” for Sheet and “Inventory” for Column to match the “inventory” dimension in the Excel sheet to the “Inventory” dimension in the Tableau Inventory sheet.



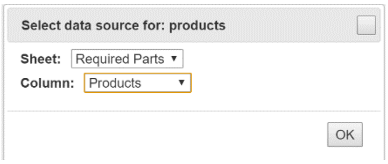
Select data source for: inventory

Sheet: Inventory2

Column: Inventory

Click OK.

On the “Select data source for: products” dialog, select “Required Parts” for Sheet and “Products” for Column to match the “products” dimension in the Excel sheet to the “Products” dimension in the Tableau Required Parts sheet.



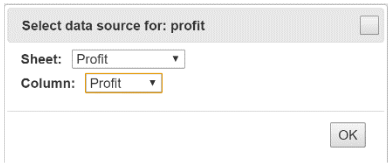
Select data source for: products

Sheet: Required Parts

Column: Products

Click OK.

On the “Select data source for: profit” dialog, select “Profit” for Sheet and “Profit” for Column to match the “profit” dimension in the Excel sheet to the “Profit” dimension in the Tableau Profit sheet.



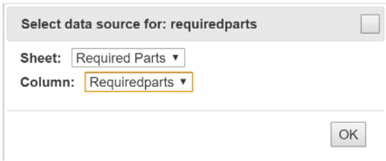
Select data source for: profit

Sheet: Profit

Column: Profit

Click OK.

On the “Select data source for: requiredparts” dialog, select “Required Parts” for Sheet and “Requiredparts” for Column to match the “requiredparts” dimension in the Excel sheet to the “Requiredparts” dimension in the Tableau Required Parts sheet.



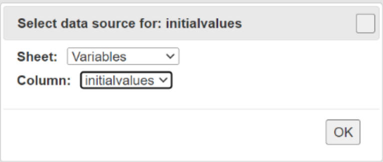
Select data source for: requiredparts

Sheet: Required Parts

Column: Requiredparts

Click OK.

On the “Select data source for: “initialvalues” dialog, select “Variables” for Sheet and “initialvalues” for Column to match the “initialvalues” dimension in the Excel sheet to the “initialvalues” dimension in the Tableau sheet.

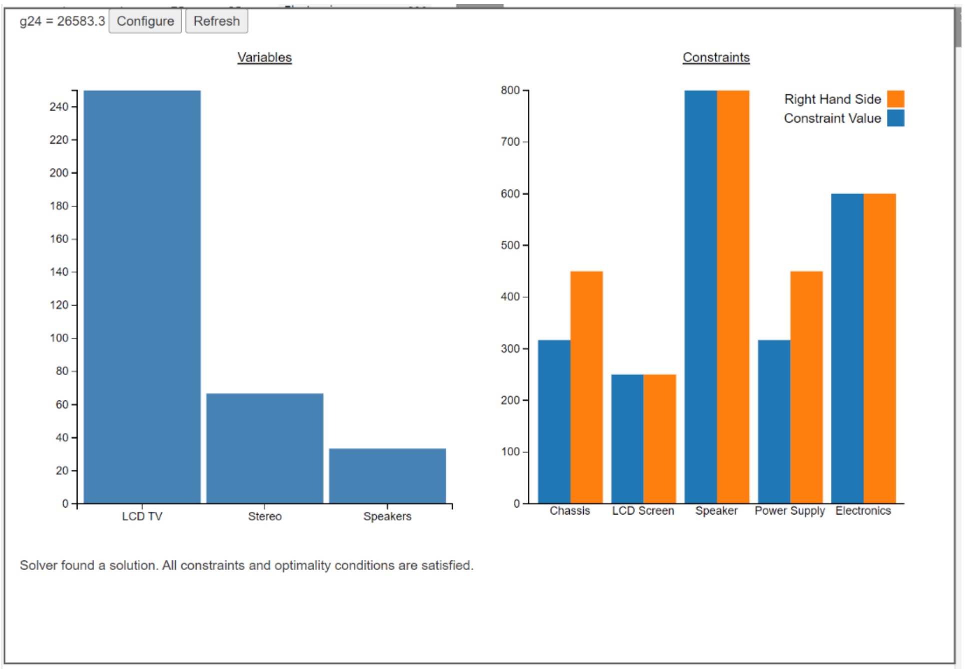


Select data source for: initialvalues

Sheet: Variables

Column: initialvalues

Immediately, the RASON model is submitted to the RASON Server, the model is solved with the LP/Quadratic engine and the final variable values are imported back into Tableau.



g24 = 26583.3

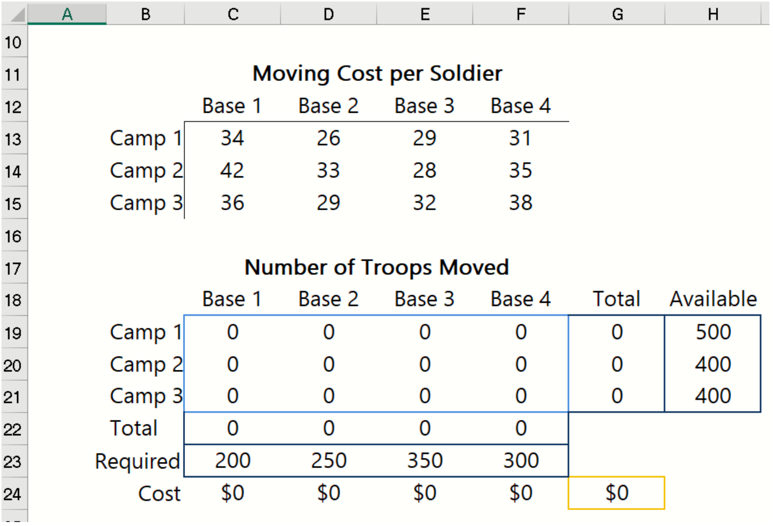
Solver found a solution. All constraints and optimality conditions are satisfied.

At the bottom of the extension, we find Solver's result message: Solver found a solution. All constraints and optimality conditions are satisfied. In the Variables chart, we see that the final variable values are: Var1 (LCD TV) equal to 250, Var2 (Stereo) equal to 67 and Var3 (Speakers) equal to 33. (Notice that the variables in the Variables graph are now listed with the actual product names rather than var 1, var 2 and var 3.) These variable values result in an objective function value equal to $26,583. In addition, a 2nd graph has been included displaying the final constraint values and the constraint right hand sides.

Troop Rotation Optimization Example

In this example, we will create a Tableau Extension using a slightly more complex example using the Troop Rotation example model. This example may be opened by clicking Help – Example Models on the Analytic Solver ribbon, then clicking Optimization Examples and scrolling down to Troop Rotation. This objective of this example is to minimize the costs associated with moving troops from Base to Base.

To open a completed version of this model click Help – Example Models – Optimization Examples, scroll down and click the link Troop Rotation for Power BI - Tableau.



Moving Cost per Soldier

	Base 1	Base 2	Base 3	Base 4
Camp 1	34	26	29	31
Camp 2	42	33	28	35
Camp 3	36	29	32	38

Number of Troops Moved

	Base 1	Base 2	Base 3	Base 4	Total	Available
Camp 1	0	0	0	0	0	500
Camp 2	0	0	0	0	0	400
Camp 3	0	0	0	0	0	400
Total	0	0	0	0		
Required	200	250	350	300		
Cost	$0	$0	$0	$0	$0	

The optimization model is included in the Model tab in the Solver Task Pane. This model written in algebraic form is below.

Minimize: Total Cost = $34 * x1 + 42 * x5 + 36 * x9 + 26 * x2 + 33 * x6 + 29 * x10 + 29 * x3 + 28 * x7 + 32 * x11 + 31 * x4 + 35 * x8 + 38 * x12$

Subject To:

Number of troops from Camps 1, 2 and 3 moved to Bases 1, 2, 3 and 4 must not exceed the number of troops available.

Camp1: $x1 + x2 + x3 + x4 <= 500$

Camp2: $x5 + x6 + x7 + x8 <= 400$

Camp3: $x9 + x10 + x11 + x12 <= 400$

Number of troops assigned per base must not be less than the amount required.

Base 1: $x1 + x5 + x9 >= 200$

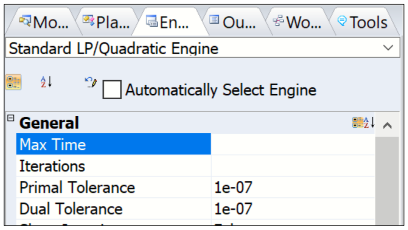
Base 2: $x2 + x6 + x10 >= 250$

Base 3: $x3 + x7 + x11 >= 350$

Base 4: $x4 + x8 + x12 >= 300$

$x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12 > 0$

The engine selected to solve this model is the Standard LP/Quadratic Engine as shown on the Engine tab in the Solver Task Pane. This is the engine that will be used to solve the model within Tableau.



Standard LP/Quadratic Engine

☐ Automatically Select Engine

General

Max Time	
Iterations	
Primal Tolerance	1e-07
Dual Tolerance	1e-07

Next, we will enter four PsiDataSrc() functions into cells J13:J16.

In cell J13, we will expose the moving costs per soldier according to the base and camp dimensions. To open the Function Arguments dialog, click Formulas

– Insert Function, select PSI Dimension for Category, select PsiDataSrc, then click OK.

Src_name = Enter “cost_src” for this argument. This names the text file that will be submitted to the RASON server, along with the RASON model.

Val_col_names – Enter “cost” for this argument. This will be the name of the data repository associated with the data passed in the next argument.

Data – Enter C13:F15 which is the actual cells containing the moving price per soldier.

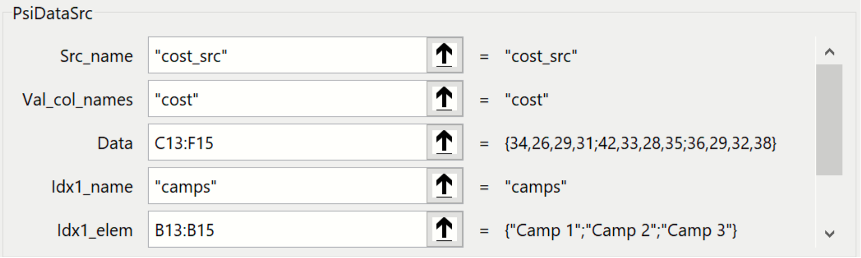
Idx1_name – Enter “camps” which is the name of the 1st index set, the row-wise index set. Remember, you must pass the row-wise index set before the column-wise index set.

Idx1_elem – Enter the elements of the “camps” index set. These appear in cells B13:B15.

Idx2_name – Enter “bases” which is the name of the 2nd index set, the column wise index set.

Idx2_elem – Enter the elements of the “bases” index set. These appear in cells C12:F12.

The Function Arguments dialog should now appear as shown in the screenshots below.


PsiDataSrc

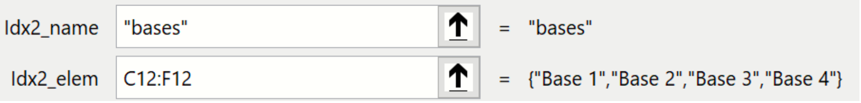
Src_name "cost_src" = "cost_src"

Val_col_names "cost" = "cost"

Data C13:F15 = {34,26,29,31;42,33,28,35;36,29,32,38}

Idx1_name "camps" = "camps"

Idx1_elem B13:B15 = {"Camp 1";"Camp 2";"Camp 3"}


Idx2_name "bases" = "bases"

Idx2_elem C12:F12 = {"Base 1","Base 2","Base 3","Base 4"}

Click OK to save this function to cell J13.

In Analytic Solver Cloud, you can type the formula directly into cell J13.

```
=PsiDataSrc("cost_src", "cost", C13:F15, "camps",
B13:B15, "bases", C12:F12)
```

Click cell J14 and open the Function Arguments dialog to enter the next PsiDataSrc() function.

In J14, we will expose the variables that indicate the number of troops moved by camp and base in cells C19:F21. Open the PsiDataSrc function (as explained above).

Src_name = Enter “vars_src” for this argument.

Val_col_names – Enter “troopsmoved” for this argument. This will be the name of the data repository associated with the data passed in the next argument.

Data – Enter C19:F21 which is the actual cells containing the moving price per soldier. Note: The range C19:F21 has been given the defined name “Troops_moved”.

Idx1_name – Enter “camps” which is the name of the 1st dimension or index set, the row-wise index set.

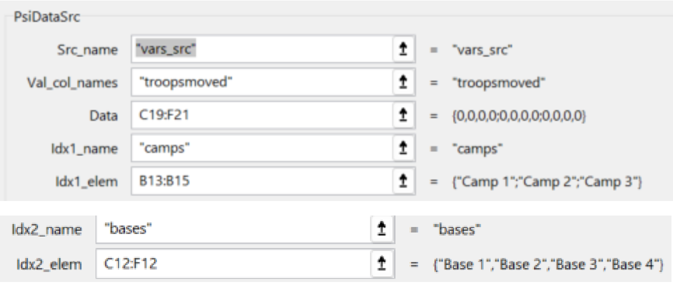
Idx1_elem – Enter the elements of the “camps” index set. These appear in cells B13:B15.

Note that it is important to re-use the location of the index sets. If we would have entered the elements for "camps" as B19:B21, rather than B13:B15, a new index set would be created.

Idx2_name – Enter “bases” which is the name of the 2nd dimension or index set, the column-wise index set.

Idx2_elem – Enter the elements of the “bases” index set. These appear in cells C12:F12.

The Function Arguments dialog should now appear as shown in the screenshots below.



PsiDataSrc

Src_name "vars_src" = "vars_src"
Val_col_names "troopsmoved" = "troopsmoved"
Data C19:F21 = {0,0,0,0;0,0,0,0;0,0,0,0}
Idx1_name "camps" = "camps"
Idx1_elem B13:B15 = {"Camp 1";"Camp 2";"Camp 3"}
Idx2_name "bases" = "bases"
Idx2_elem C12:F12 = {"Base 1","Base 2","Base 3","Base 4"}

Click OK to save this function to cell J14.

In Analytic Solver Cloud, you can type the formula directly into cell J14.

```
=PsiDataSrc("vars_src", "troopsmoved", Troops_moved,
"camps", B13:B15, "bases", C12:F12)
```

Click cell J15. Let’s enter this function by typing the formulas directly into the Excel cell. This next PsiDataSrc function exposes the number of troops available for movement.

In cell J15, enter: “=PsiDataSrc(“

First the first argument, Src_name, enter “available_src”.

For the second argument, Val_col_names, enter: “availabletroops”. The Data Well associated with this data will be given this name.

For the Data argument, the third argument, enter: “Troops_available”. This is the defined name for cells H19:H21. Alternatively, we could have typed the Excel range.

Type “camps” for the 4th argument, Idx1_name, which is the name given to the row-wise index set.

Enter the range for this index set, B13:B15, for Idx1_elem.

Your PsiDataSrc() function should match the one in the screenshot shown below.

```
=PsiDataSrc("available_src", "availabletroops",
Troops_available, "camps", B13:B15)
```

Click Enter to enter the PsiDataSrc() function into cell J15.

Finally, the last PsiDataSrc() function in cell J16 should be entered as:

```
=PsiDataSrc("required_src","requiredtroops",Troops_re
quired,"bases", C12:F12)
```

Where:

Data_src = “required_src”

Val_col_names = “requiredtroops” This is the name that will appear for the data repository associated with this data in Tableau.

Data = Troops_required This is the data associated with Val_col_names.

Idx1_name = “bases” This is the name of the index set that is associated with the data passed to the Data argument.

Idx1_elem – C12:F12 This is the Excel range where the elements for Idx1_name are located on the worksheet.

Now we are ready to create the Tableau Extension.

In order to see our constraint final values in the Tableau Extension, we will need to use the Psi function PsiFinalValue. (For more information on this function, see the Psi Function chapter in the Analytic Solver Reference Guide.) In cell J22, enter "=PsiFinalValue(Troops_per_camp)" to see the final value of the constraints in cells G19:G21 in the Tableau Extension. In cell L22, enter =PsiFinalValue(Troops_per_base)" to see the final value of the constraints in cells C22:F22 in the Tableau Extension.

In order to see the constraint right hand sides in the Tableau Extension, we will need to to use the Psi functon PsiSlackValue. (For more information on this function, see the Psi Function chapter in the Analytic Solver Reference Guide.) In cell K29, enter “=PsiSlackValue(Troops_per_camp) and in cell L28, enter “=PsiSlackValue(Troops_per_base).

First, click Deploy Model on the ribbon, then select Cloud Service – RASON Model to post the model to RASON Cloud Services. Then click Deploy Model again and select Tableau – Managed Model.

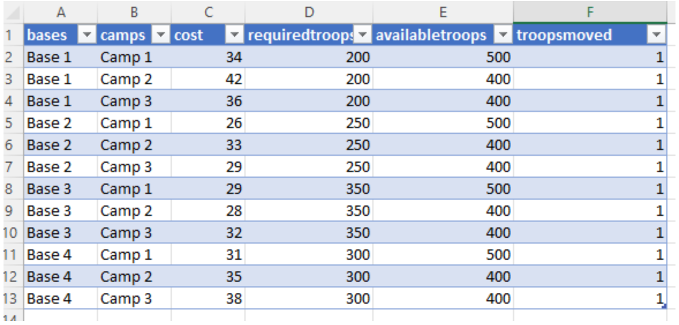
Since this is the first time we are creating an Extension from this model, there is no need to solve the model first. If, however, we make a change to the Troop Rotation model, then we would need to either diagnose or solve the model first to ensure that the changes are captured in the new extension.

Save the Solver.trex file (the Tableau Extension file) in a location of your choice.

Next, we will create the data file. Open a new Excel workbook.

For convenience sake, in this example, we will pass all data in one Excel table.

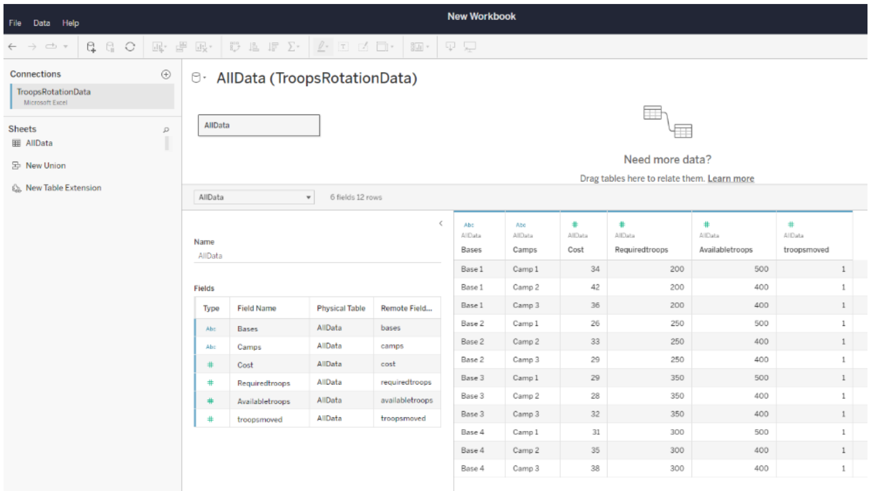
In a new Excel sheet, create a table (Insert – Table) with 13 rows and 6 columns, as shown in the screenshot below. Notice that in order to enter all 12 costs, labels for camps, bases, requiredtroops and availabletroops were repeated. The last column, troopsmoved, holds the initial starting values for the variables, all 1’s. Note that each column heading matches the index sets used in the PsiDataSrc() functions. Rename the worksheet “AllData” then save the workbook as TroopsRotationData.xlsx. (Note that you can find this completed data file by clicking Help -- Example Models – Optimization Examples and scroll down to click the Troop Rotation Data for Power BI – Tableau link.)



bases	camps	cost	requiredtroops	availabletroops	troopsmoved
Base 1	Camp 1	34	200	500	1
Base 1	Camp 2	42	200	400	1
Base 1	Camp 3	36	200	400	1
Base 2	Camp 1	26	250	500	1
Base 2	Camp 2	33	250	400	1
Base 2	Camp 3	29	250	400	1
Base 3	Camp 1	29	350	500	1
Base 3	Camp 2	28	350	400	1
Base 3	Camp 3	32	350	400	1
Base 4	Camp 1	31	300	500	1
Base 4	Camp 2	35	300	400	1
Base 4	Camp 3	38	300	400	1

Now let’s click back to the Data Source tab in Tableau, to import the new Extension and the data.

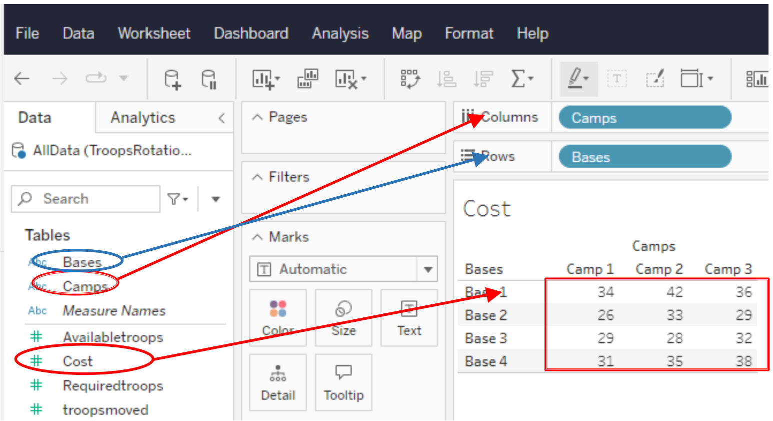
Open a new workbook and import the troop data. (Having trouble importing the data into Tableau? See the example above.)



Type	Field Name	Physical Table	Remote Field...
Abc	Bases	AllData	bases
Abc	Camps	AllData	camps
#	Cost	AllData	cost
#	Requiredtroops	AllData	requiredtroops
#	Availabletroops	AllData	availabletroops
#	troopsmoved	AllData	troopsmoved

Bases	Camps	Cost	Requiredtroops	Availabletroops	troopsmoved
Base 1	Camp 1	34	200	500	1
Base 1	Camp 2	42	200	400	1
Base 1	Camp 3	36	200	400	1
Base 2	Camp 1	26	250	500	1
Base 2	Camp 2	33	250	400	1
Base 2	Camp 3	29	250	400	1
Base 3	Camp 1	29	350	500	1
Base 3	Camp 2	28	350	400	1
Base 3	Camp 3	32	350	400	1
Base 4	Camp 1	31	300	500	1
Base 4	Camp 2	35	300	400	1
Base 4	Camp 3	38	300	400	1

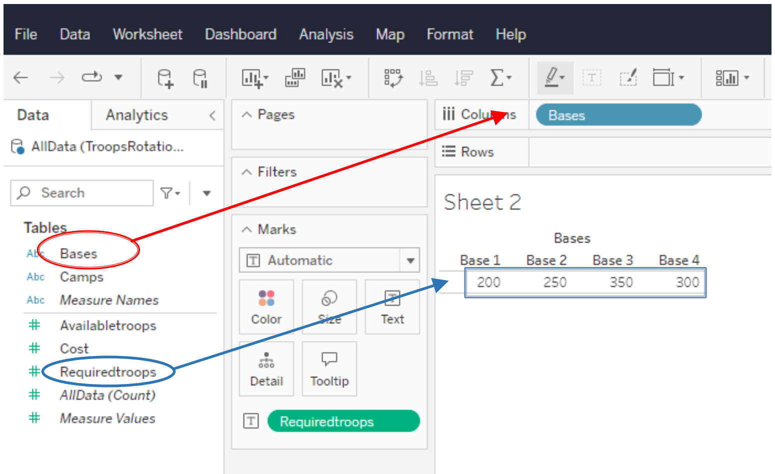
The AllData sheet appears under Sheets and the data from that sheet appears at the bottom of the Tableau window. Click Sheet 1. Under Dimensions, drag “Bases” to Rows and “Camps” to Columns. Then drag “Cost” to the table elements. Hover over SUM(Cost) (in the Marks section), click the down arrow that appears to the right, and select Dimensions from the menu.



Bases	Camp 1	Camp 2	Camp 3
Base 1	34	42	36
Base 2	26	33	29
Base 3	29	28	32
Base 4	31	35	38

Rename the worksheet to Cost and then add a new Sheet.

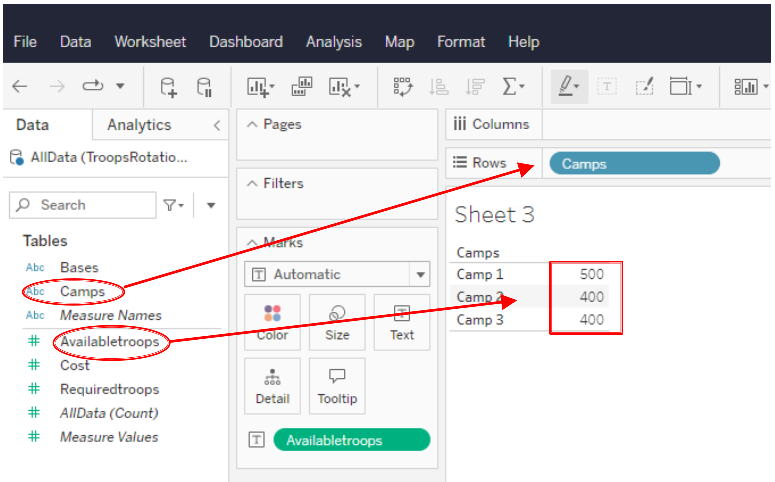
On Sheet 2, drag “Bases” to Columns and “Requiredtroops” to the table elements. Hover over SUM(Required.., click the down arrow that appears to the right, then select Dimensions from the menu.



Bases			
Base 1	Base 2	Base 3	Base 4
200	250	350	300

Rename the worksheet to Required Troops and then add a new Sheet.

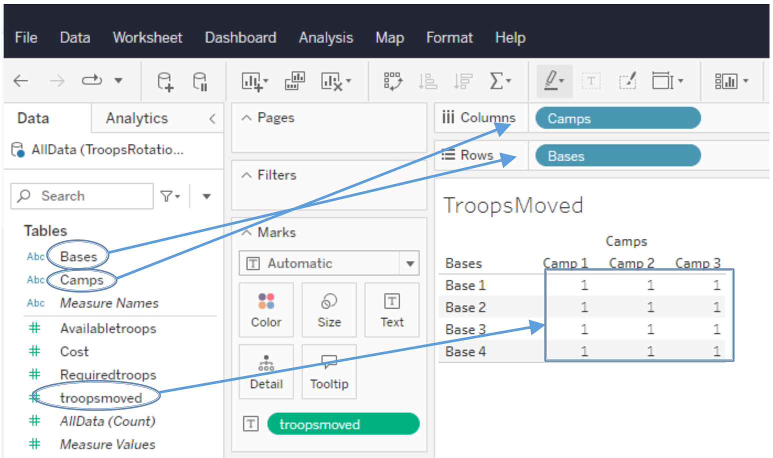
On Sheet 3, drag “Camps” to Rows and “Availabletroops” to the table elements. Hover over SUM(Available.., click the down arrow that appears to the right, then select Dimensions from the menu.



Camps	
Camp 1	500
Camp 2	400
Camp 3	400

Rename the sheet to Available Troops and then add a new Sheet.

On Sheet 4, drag “Camps” to Columns, and “Bases” to Rows, then “troopsmoved” to the table elements. Hover over SUM(troopsmoved), click the down arrow that appears to the right, then select Dimensions from the menu.



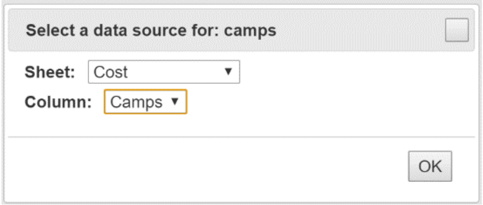
TroopsMoved

	Camps		
Bases	Camp 1	Camp 2	Camp 3
Base 1	1	1	1
Base 2	1	1	1
Base 3	1	1	1
Base 4	1	1	1

Rename the sheet to TroopsMoved and then open a new Dashboard.

Drag each of the four sheets (Cost, Required Troops, TroopsMoved and Available Troops) and the Extension to the Tableau canvas as described in the Product Mix example, above.

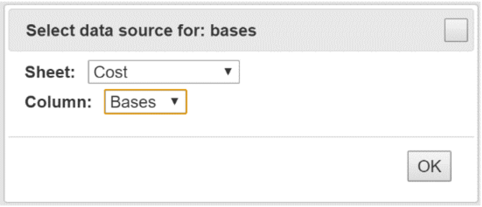
Match the dimensions from the Excel sheet to the dimensions and measures in the Tableau sheets as follows.



Select a data source for: camps

Sheet: Cost

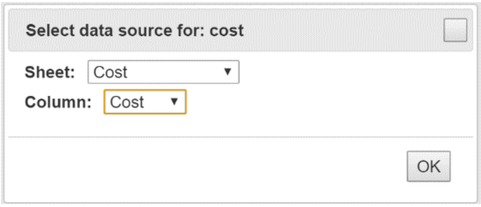
Column: Camps



Select data source for: bases

Sheet: Cost

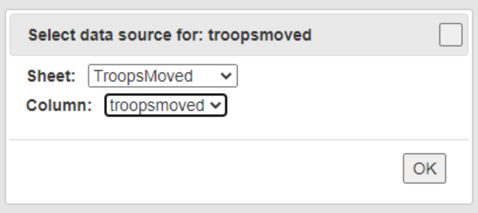
Column: Bases



Select data source for: cost

Sheet: Cost

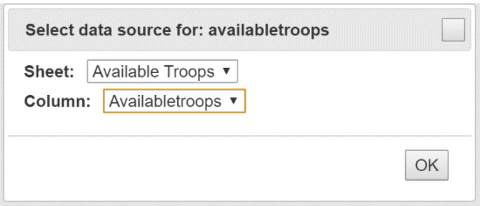
Column: Cost



Select data source for: troopsmoved

Sheet: TroopsMoved

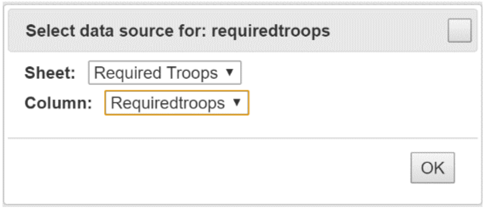
Column: troopsmoved



Select data source for: availabletroops

Sheet: Available Troops

Column: Availabletroops

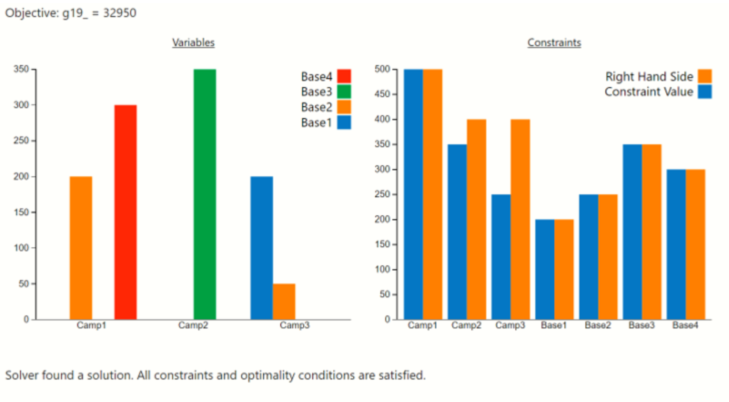


Select data source for: requiredtroops

Sheet: Required Troops

Column: Requiredtroops

The resultant Tableau Extension is shown in the screenshot below. Now each time new data is imported into Tableau, the Troop Rotation example data can be solved from within Tableau without the need to ever return to the original Excel workbook. Note that as long as your PsiDataSrc functions remain the same, the named model can be updated on the Rason Cloud Service and you will not need to create a new extension. However, if the PsiDataSrc() functions change, you will need to update the named model residing on the Rason Cloud service and the Tableau Extension.



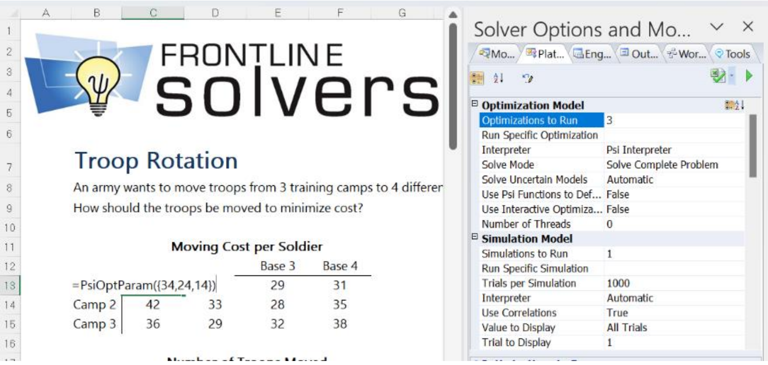
Objective: g19_ = 32950

Solver found a solution. All constraints and optimality conditions are satisfied.

Using Optimization Parameters in Tableau

What if we want to see how our final variable values and objective function value change as the number of troops required changes? For this type of “What If” analysis, we can use the PsiOptParam() optimization parameter. Recall from the chapter, Examples: Parameters and Sensitivity Analysis, an **optimization parameter** (PsiOptParam) is automatically varied when you perform multiple optimizations. You do this by clicking the Optimize button on the Ribbon, or the green arrow in the Task Pane, with the Number of Optimizations set to a value greater than 1.

Let’s go back to our example Troop Rotation Excel model and enter an optimization parameter for the number of troops required in Base 1. In cell C13, enter “=PsiOptParam({34,24,14})”, then click the Platform tab in the Solver Task pane and change the Number of Optimizations to “3”. When I click Solve, Analytic Solver will solve 3 optimizations. In the first optimization, cell C13 will be set to 34, in the second, 24, and in the third, 14.



Troop Rotation

An army wants to move troops from 3 training camps to 4 differer

How should the troops be moved to minimize cost?

Moving Cost per Soldier

			Base 3	Base 4
=PsiOptParam({34,24,14})			29	31
Camp 2	42	33	28	35
Camp 3	36	29	32	38

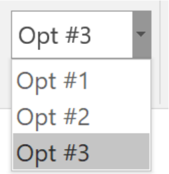
Optimization Model

Optimizations to Run	3
Run Specific Optimization	
Interpreter	Psi Interpreter
Solve Mode	Solve Complete Problem
Solve Uncertain Models	Automatic
Use Psi Functions to Def...	False
Use Interactive Optimiza...	False
Number of Threads	0

Simulation Model

Simulations to Run	1
Run Specific Simulation	
Trials per Simulation	1000
Interpreter	Automatic
Use Correlations	True
Value to Display	All Trials
Trial to Display	1

Click the Output tab, then click the green arrow to solve the model. After the solving process is over, we can select the three different optimizations in the Optimization drop down menu (in the Tools section of the Analytic Solver ribbon), to see how the objective function changes in cell G24, according to the value set for this parameter.



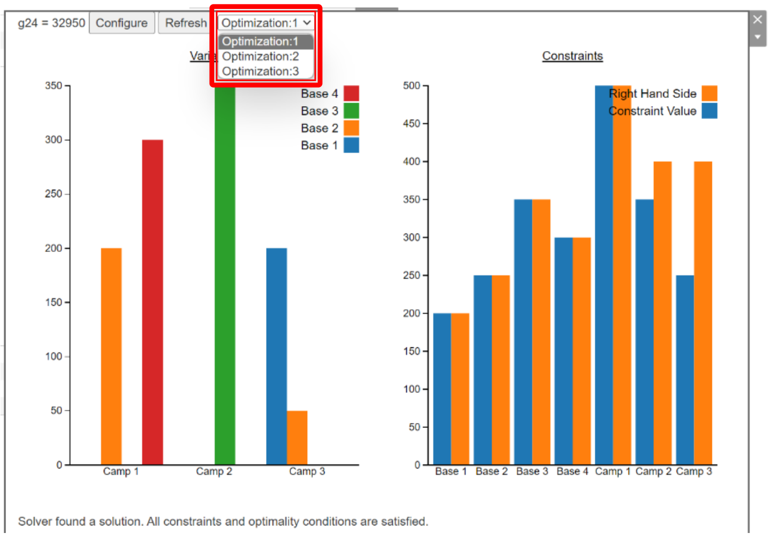
In the first optimization, cell C13 = 34 and the resultant objective function value was $32,950. In the second optimization, cell C13 = 24 and the resultant objective function value was $31,150 and in the third optimization cell C13 = 14 with an objective function value of 29,150.

Deploy a new Tableau Extension by clicking Deploy Model **--** Tableau – Managed Model. (There's no need to re-deploy the model to the RASON Cloud Service.)

Note: At this point, it's a good idea to clear your browser's cache.

Click back to Tableau and import the newly created Tableau Extension using the steps outlined above.

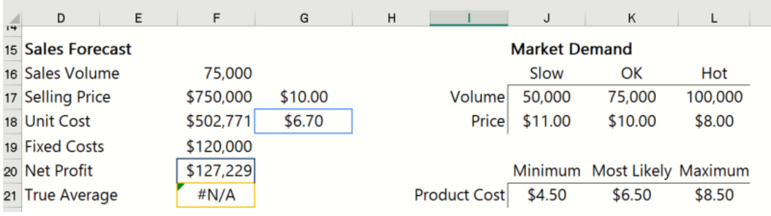
Immediately, the model solves and the results will be displayed on the Tableau canvas. To view each solution, click the down arrow within the newly added Multiple Optimization drop-down menu.



Solver found a solution. All constraints and optimality conditions are satisfied.

Simulation Example Model

Now let's create a Tableau extension using a simulation example. The process is exactly the same as when using an optimization model. Open the Business Forecast Example by clicking Help – Example Models – Monte Carlo Simulation Examples, click the Business Forecast hyperlink to open the file. To open a completed example, open the BusinessForecastPBI example that appears at the bottom of Simulation Models.



	D	E	F	G	H	I	J	K	L
15	Sales Forecast						Market Demand		
16	Sales Volume		75,000				Slow	OK	Hot
17	Selling Price		$750,000	$10.00		Volume	50,000	75,000	100,000
18	Unit Cost		$502,771	$6.70		Price	$11.00	$10.00	$8.00
19	Fixed Costs		$120,000						
20	Net Profit		$127,229				Minimum	Most Likely	Maximum
21	True Average		#N/A			Product Cost	$4.50	$6.50	$8.50

First, we will use a PsiDataSrc() function to generate data repositories for use in the Tableau Extension.

Click a blank cell, such as cell G24, then type: "=PsiDataSrc("market_src", "market", J17:L18, "parameters", I17:I18, "mktdemand", J16:L16)

where:

market_src – Is the name used to identify the text file that will be submitted to the RASON server.

market – Will be the name of the data repository in Tableau.

J17:L18 – Is the data range for "market".

parameters – Is the name of the 1st index set for the data specified in the "data" argument, market.

I17:I18 – Is the Excel range where the data for "parameters" is located.

mktdemand – The name of the 2nd index set for the data specified in the "data" argument.

J16:L16 – Is the Excel range where the data for "mktdemand" is located.

Type “)” to close the formula and then press the Enter key to enter the formula into the Excel cell.

Click a 2nd blank cell, such as G25, then enter the following function.

=PsiDataSrc(“price_src”, “mktprice”, J21:L21, “price”, J20:L20)

where:

price_src – Is the name used to identify the text file that will be submitted to the RASON server.

mktprice – Is the name of the data repository in Tableau.

J21:L21 – Is the data range for “mktprice”.

price – Is the name of the 2nd index set for the data specified in the Data argument, market.

J20:L20 – Is the Excel range where the data for “price” is located.

Two Psi Functions, PsiPercentiles and PsiData, must be added to the spreadsheet in order to draw the histogram in the Tableau Extension.

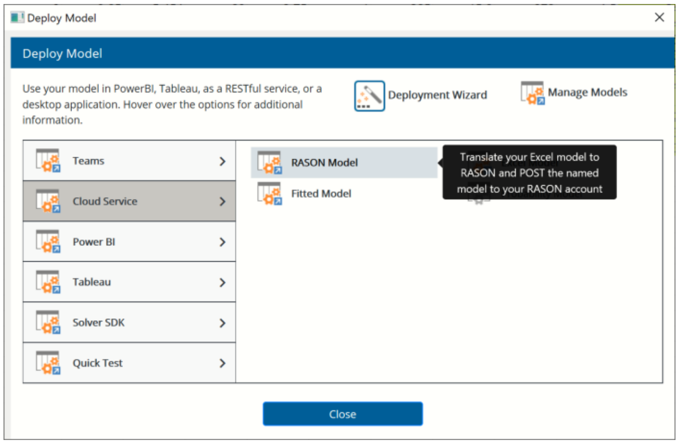
H28 = PsiPercentile(F20)

G28 = PsiData(F20)

The PsiPercentile function returns all percentiles for the uncertain function in cell F20. The PsiData function returns all trial values for cell F20.

For more information on these formulas, see the Analytic Solver Reference Guide.

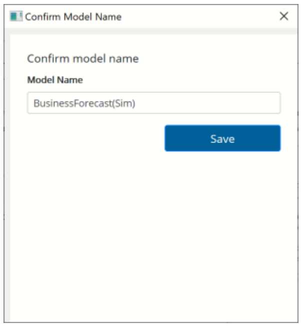
Click Deploy Model – Cloud Service – RASON Model to post the model to RASON Cloud Services.



Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Translate your Excel model to RASON and POST the named model to your RASON account

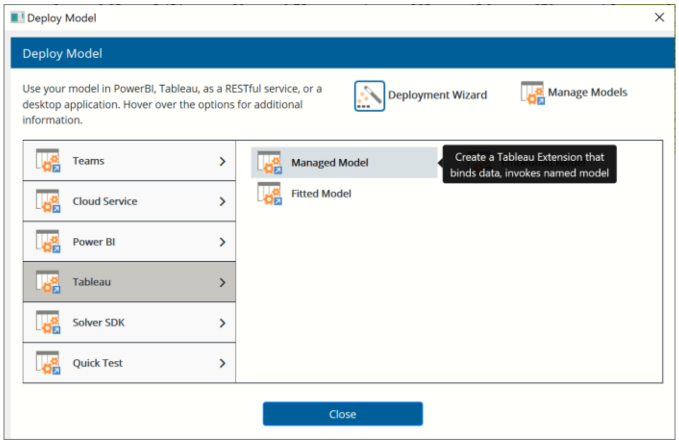
Confirm the model name as BusinessForecast(Sim).



Confirm model name

Model Name: BusinessForecast(Sim)

Then click Deploy Model – Tableau – Managed Model to create a new Tableau Extension, Solver.trex.



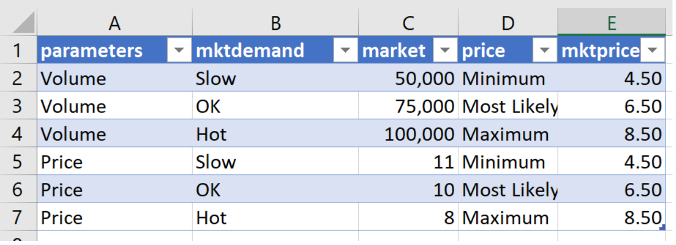
Deploy Model

Use your model in PowerBI, Tableau, as a RESTful service, or a desktop application. Hover over the options for additional information.

Create a Tableau Extension that binds data, invokes named model

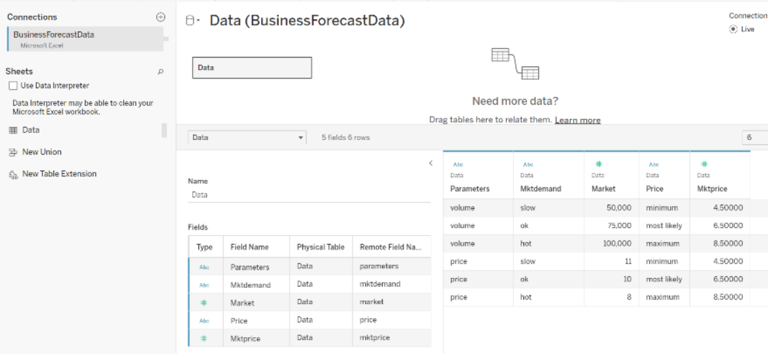
Then save the Tableau Extension to a desired location. Your entire simulation model (uncertain variables/functions) is now contained within this extension.

Click back to Excel and open a new workbook to enter the following data into an Excel table. Rename the worksheet as “Data” and save the workbook as BusinessForecastData.xlsx. (You can open this completed file by clicking Help – Example Models – Monte Carlo Simulation Examples, scroll down and open the link.)



	A	B	C	D	E
1	parameters	mktdemand	market	price	mktprice
2	Volume	Slow	50,000	Minimum	4.50
3	Volume	OK	75,000	Most Likely	6.50
4	Volume	Hot	100,000	Maximum	8.50
5	Price	Slow	11	Minimum	4.50
6	Price	OK	10	Most Likely	6.50
7	Price	Hot	8	Maximum	8.50

Open a new workbook in Tableau, then upload the data file.



Data (BusinessForecastData)

Connections

BusinessForecastData
Microsoft Excel

Sheets

☐ Use Data Interpreter

Data Interpreter may be able to clean your Microsoft Excel workbook.

Need more data?

Drag tables here to relate them. Learn more

Connection
● Live

Data 5 fields 6 rows

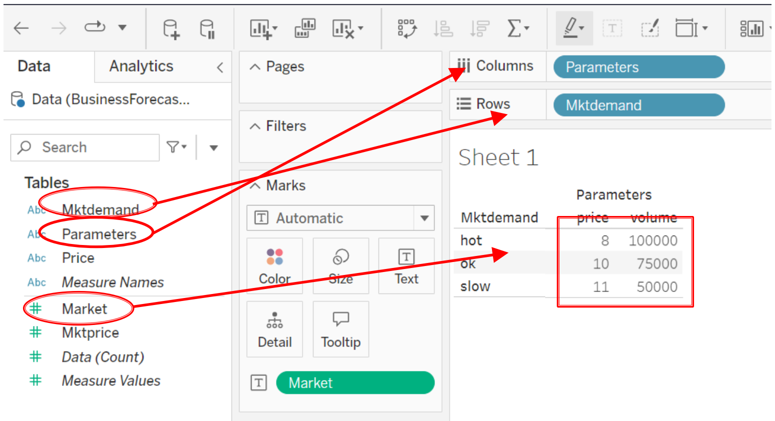
Name: Data

Fields

Type	Field Name	Physical Table	Remote Field Na...
Abc	Parameters	Data	parameters
Abc	Mktdemand	Data	mktdemand
#	Market	Data	market
Abc	Price	Data	price
#	Mktprice	Data	mktprice

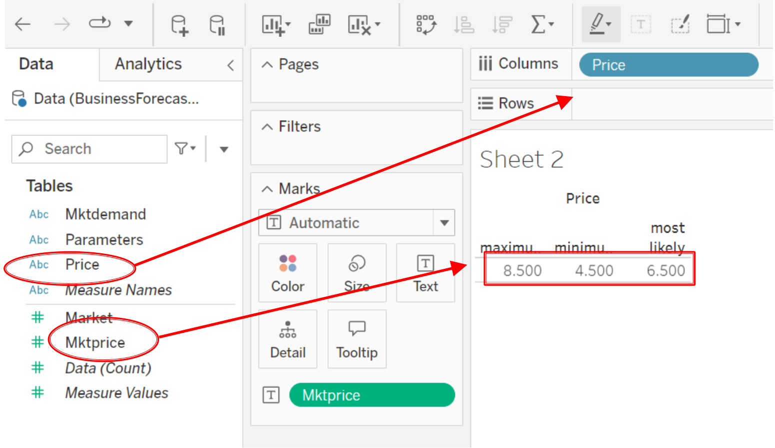
Parameters	Mktdemand	Market	Price	Mktprice
volume	slow	50,000	minimum	4.50000
volume	ok	75,000	most likely	6.50000
volume	hot	100,000	maximum	8.50000
price	slow	11	minimum	4.50000
price	ok	10	most likely	6.50000
price	hot	8	maximum	8.50000

Click on the Sheet 1 tab, under Dimensions, drag Parameters to Columns and Mktdemand to Rows, then drag Market to the table elements. Hover over SUM(Market) , click the down arrow that appears to the right and select Dimension.



	Parameters	
Mktdemand	price	volume
hot	8	100000
ok	10	75000
slow	11	50000

Rename Sheet 1 to Market and add a new worksheet. On Sheet 2, drag "Price" to Columns and "Mktprice" to the table elements. Hover over SUM(Mktprice), click the down arrow that appears to the right and select Dimension.

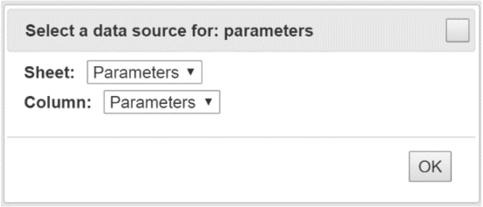


Price		
maximu	minimu	most likely
8.500	4.500	6.500

Rename Sheet 2 to Price and create a new dashboard.

Drag both the Parameters and Price Sheets to the Tableau canvas. Drag the Extension icon to the canvas, browse to where the Solver.trex file was saved and click Open.

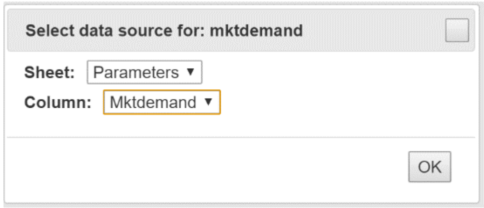
On the "Select data source for: parameters" dialog, select "Parameters" for both Sheet and Column to match the "parameters" dimension in the Excel sheet to the "Parameters" dimension in the Tableau Parameters sheet. Then click OK.


Select a data source for: parameters

Sheet: Parameters ▾

Column: Parameters ▾

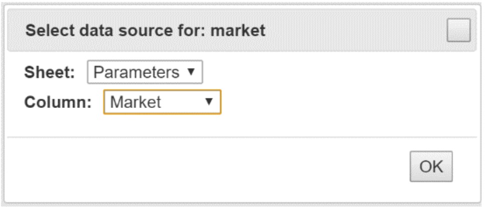
On the "Select data source for: mktdemand", select "Parameters" for Sheet and "Mktdemand" for Column to match the "mktdemand" dimension in the Excel sheet to the "Mktdemand" dimension in the Tableau Parameters sheet. Then click OK.


Select data source for: mktdemand

Sheet: Parameters

Column: Mktdemand

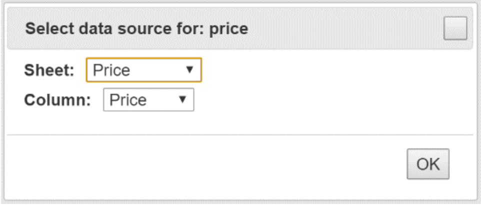
On the “Select data source for: market”, select “Parameters” for Sheet and “Market” for Column to match the “market” dimension in the Excel sheet to the “Market” measure in the Tableau Parameters sheet. Then click OK.


Select data source for: market

Sheet: Parameters

Column: Market

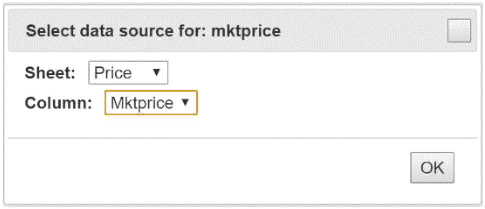
On the “Select data source for: price”, select “Price” for both Sheet and Column, then click OK to match the “price” dimension in the Excel sheet to the “Price” dimension in the Tableau Price sheet.


Select data source for: price

Sheet: Price

Column: Price

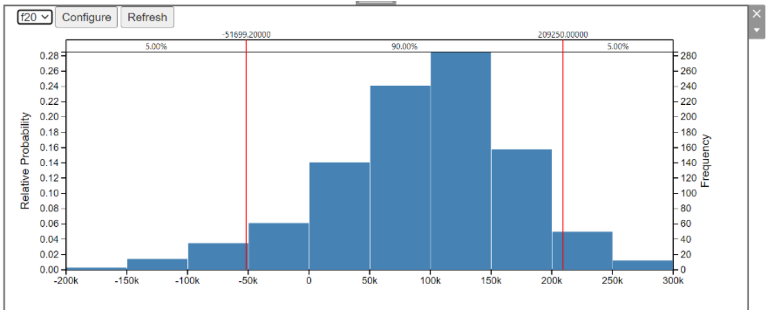
On the “Select data source for: mktprice”, select “Price” for Sheet and “Mktprice” for Column, then click OK to match the “mktprice” dimension in the Excel sheet to the “Mktprice” measure in the Tableau Price sheet.


Select data source for: mktprice

Sheet: Price

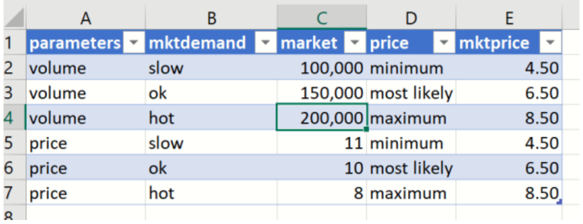
Column: Mktprice

Immediately, once the data repositories are filled with the appropriate data fields, a simulation will be run.



Again, note that as long as your PsiDataSrc functions remain the same, the named model can be updated on the Rason Cloud Service and you will not need to create a new extension. However, if the PsiDataSrc() functions change, you will need to update both the Rason named model residing on the Rason Cloud Server and the Tableau Extension.

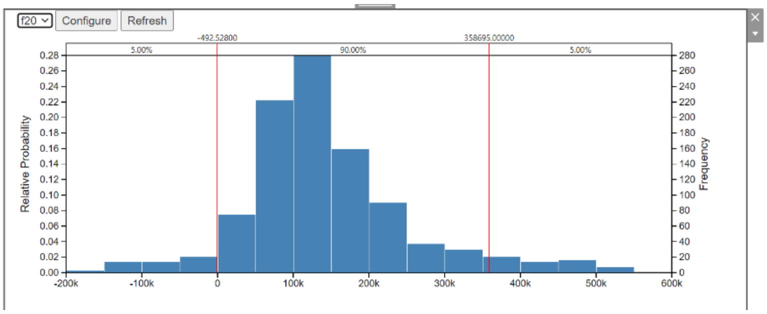
If your data changes, say your demand explodes, you can simply edit (and save) your data file…



parameters	mktdemand	market	price	mktprice
volume	slow	100,000	minimum	4.50
volume	ok	150,000	most likely	6.50
volume	hot	200,000	maximum	8.50
price	slow	11	minimum	4.50
price	ok	10	most likely	6.50
price	hot	8	maximum	8.50

…click back to Tableau and refresh the data source…

… and the Tableau Extension will automatically redraw using the new data.



Error Messages

If you've made an error in your PsiDataSrc() function or in your data file, your Tableau extension will display an error such as "Inconsistent Data source definition PSI could not parse Excel formulas" or "Invalid binding source definition PSI could not parse Excel formulas".

If your Tableau extension is displaying either of these errors or a different error, first, check the PsiDataSrc() function in your Excel model to make sure that you have not mixed up the order of your index sets. In addition, check your data to make sure that it is 1. Contained within an Excel table, 2. That the column headings match the index sets defined in PsiDataSrc() and 3. That your model parameters match the elements in your data table.

For more information on an error message displayed in your Tableau Extension, see the Frontline Solvers Reference Guide or contact Frontline Systems Support Team by emailing support@solver.com or by calling 888-831-0333.

Examples: Parameters and Sensitivity Analysis

Introduction

This chapter introduces the use of **parameters**, and functions for **sensitivity analysis** in Analytic Solver, with examples. The use of parameters and sensitivity analysis has also been extended to Analytic Solver Cloud.

Sensitivity analysis is designed to help you explore a conventional Excel spreadsheet model, often before you define an optimization or simulation model. The goal of sensitivity analysis is to identify the key input parameters that have the *greatest impact* on results of interest in your model (such as Net Profit), and to see the effect of varying these key input parameters over a range, in reports and graphs. Analytic Solver helps you *automatically identify* sensitivity parameters, and *automatically vary* them to produce reports and graphs.

Parameters and Results

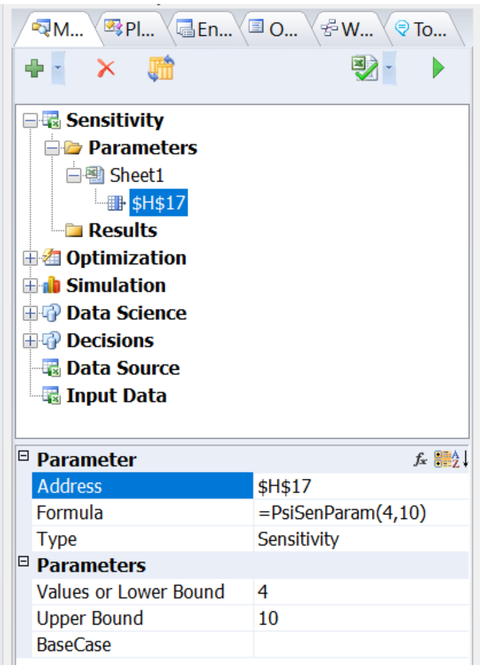
All Analytic Solver products support three kinds of parameters:

- A **sensitivity parameter** (PsiSenParam) is automatically varied when you perform a sensitivity analysis. You do this by selecting **Sensitivity** from the Reports or Charts galleries on the Ribbon.
- An **optimization parameter** (PsiOptParam) is automatically varied when you perform multiple optimizations. You do this by clicking the Optimize button on the Ribbon, or the green arrow in the Task Pane, with the Number of Optimizations set to a value greater than 1. (Not available in Analytic Solver Simulation.)
- A **simulation parameter** (PsiSimParam) is automatically varied when you perform multiple simulations. You do this by clicking the Simulate button on the Ribbon, or the green arrow in the Task Pane, with the Number of Simulations set to a value greater than 1. (Not available in Analytic Solver Optimization.)
- A **calculation parameter** (PsiCalcParam) is automatically varied when you perform multiple calculations. You do this by clicking the green arrow in the Task Pane, with the Number of Calculations set to a value greater than 1.

Note: When entering PsiOptParam, PsiSenParam, PsiCalcParam or, PsiSimParam manually (i.e. simply typing the function into an Excel cell) in Analytic Solver Cloud, the taskpane must be refreshed before performing a parameter analysis report.

Viewing Parameters in the Task Pane

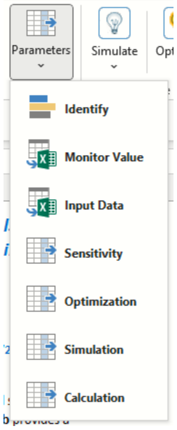
All four types of parameters appear in the Task Pane Model outline. In the example on the next page, a sensitivity parameter has been defined at cell H17 and selected in the task pane. Its properties are shown in the lower part of the Task Pane. It is possible to **change the type** of a parameter, by simply selecting from the dropdown list in the lower part of the Task Pane as shown: For example, to use H17 in a simulation model, automatically varied across multiple simulations, you would select **Simulation** from the dropdown list.

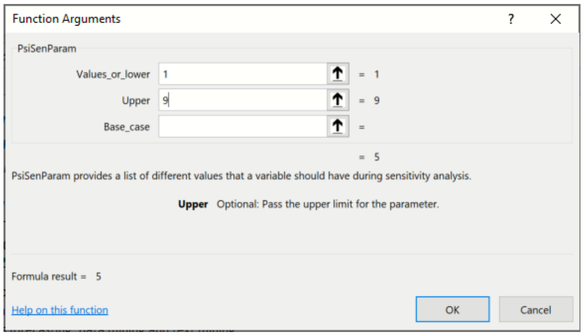


Parameter	
Address	$H$17
Formula	=PsiSenParam(4,10)
Type	Sensitivity
Parameters	
Values or Lower Bound	4
Upper Bound	10
BaseCase	

Defining a Parameter

You can manually define an input cell as a parameter by simply selecting the cell, choosing **Parameters** and **Sensitivity**, **Optimization, Simulation** or **Calculation** from the Ribbon, and entering a lower and upper limit on values for the parameter, or a list of explicit values for the parameter. In the example on the next page, we are defining a sensitivity parameter.





Values_or_lower 1 = 1

Upper 9 = 9

Base_case =

= 5

PsiSenParam provides a list of different values that a variable should have during sensitivity analysis.

Upper Optional: Pass the upper limit for the parameter.

Formula result = 5

You can also simply type a formula such as =PsiSenParam(1,9) in a cell, which has the same effect as the above steps. In both cases, the parameter appears in the Task Pane Model outline.

If you specify a lower and upper limit, the **step size** for successive values is determined when you perform a parameterized sensitivity analysis, or multiple parameterized optimizations or simulations. Analytic Solver will subdivide the range from the lower to the upper limit into equal-size intervals.

For example, for the above sensitivity parameter, you would select **Reports** or **Charts** – **Sensitivity** – **Parameter Analysis** and specify the number of points on the major axis. If you specified 18 points, the step size would be 0.5.

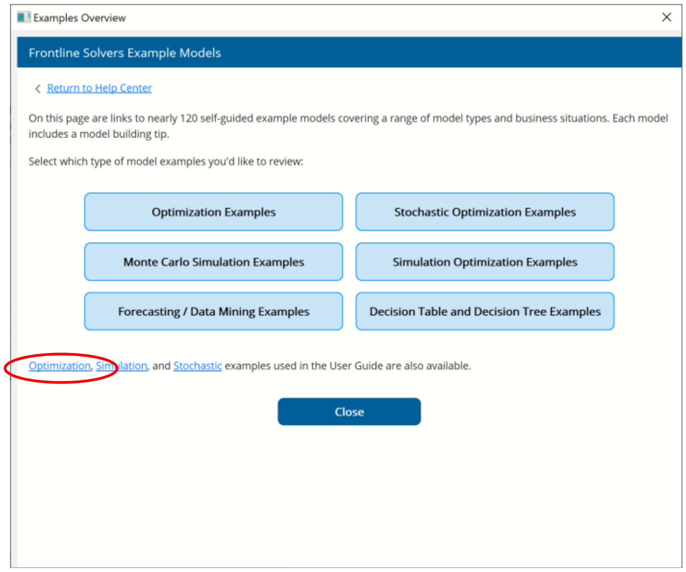
Automatic Parameter Identification

Analytic Solver can *automatically* find the input cells with the greatest impact on any selected formula cell. You can then choose one or more of these input cells to serve as sensitivity (or optimization, simulation or calculation) parameters.

To do this, **select any formula cell in an Excel model, and choose Parameters – Identify from the Ribbon**.

Analytic Solver traces through all ‘precedent’ formulas to find input cells (leaves of the formula tree) on which the selected formula cell depends. It then varies each input cell independently, computing its impact on the selected formula cell. The input cells are ranked in descending order by (absolute) impact, and the highest-impact cells are displayed in the Tornado chart.

In the example below, we’ve clicked **Help – Examples** on the ribbon to open the StandardExamples.xlsx.

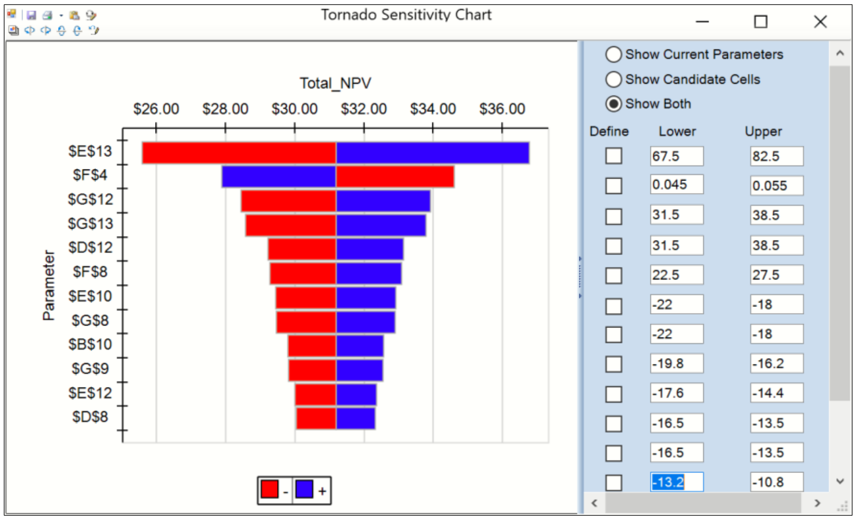

Examples Overview
Frontline Solvers Example Models
Return to Help Center
Optimization Examples
Stochastic Optimization Examples
Monte Carlo Simulation Examples
Simulation Optimization Examples
Forecasting / Data Mining Examples
Decision Table and Decision Tree Examples
Close

On this page are links to nearly 120 self-guided example models covering a range of model types and business situations. Each model includes a model building tip.

Select which type of model examples you'd like to review:

Optimization, Simulation, and Stochastic examples used in the User Guide are also available.

Click the Example 8 tab and set cells B17:G17 to 1. (If these cells are all 0, then no projects are selected and the Total NPV is always 0). To see which cells have the greatest impact on Total NPV, we simply select cell (I28) and choose **Parameters – Identify** from the Ribbon. After a moment, a Tornado chart like the one below appears.



○ Show Current Parameters
○ Show Candidate Cells
● Show Both

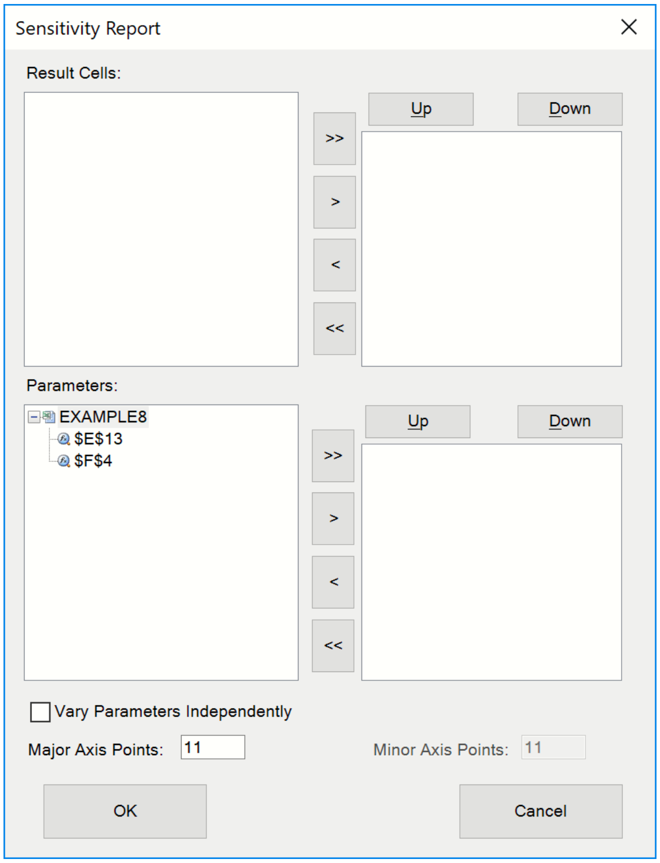
Define	Lower	Upper
☐	67.5	82.5
☐	0.045	0.055
☐	31.5	38.5
☐	31.5	38.5
☐	22.5	27.5
☐	-22	-18
☐	-22	-18
☐	-19.8	-16.2
☐	-17.6	-14.4
☐	-16.5	-13.5
☐	-16.5	-13.5
☐	-13.2	-10.8

The chart shows us, at a glance, that Total NPV is most sensitive to cell E13 – which is a large positive cash flow in Year 6 for Opportunity 4. It is next most sensitive (in the opposite direction, decreasing when the parameter increases) to the interest rate in cell F4.

The right panel of the chart shows the lower value and upper value that were used for each input cell when computing the formula value. An ordinary cell containing a number is set to its current value – *n*% and its current value + *n*%, where *n*% (initially 10%) is specified in the edit box in the lower right corner of the chart. If any input cell is already *defined as a parameter*, the lower and upper limit arguments of its PsiXxxParam() call are used instead.

By choosing the radio button corresponding to “Show Current Parameters”, “Show Candidate Cells”, or “Show both” at the top of the right panel, you can show either one or both types of input cells in the Tornado chart. By **checking the box on the same row** as the cell, you can *define* a candidate cell as a **sensitivity parameter** (or, if you change the selection in the dropdown list in the lower right corner of the chart, an **optimization**, **simulation or calculation** parameter). These definitions take effect when you click the Save icon at the upper left corner of the chart. For this example, we’ll check the boxes on the same rows as cells **E13** and **F4**; we’ll also set F4’s Lower value to **0.01** (1%) and the Upper Value to **0.10** (10%).

Now, when you click Charts – Sensitivity Analysis – Parameter Analysis, you'll see both parameters appear under Parameters in the Sensitivity Report dialog. Continue reading to discover how to create a parameter report using Analytic Solver.



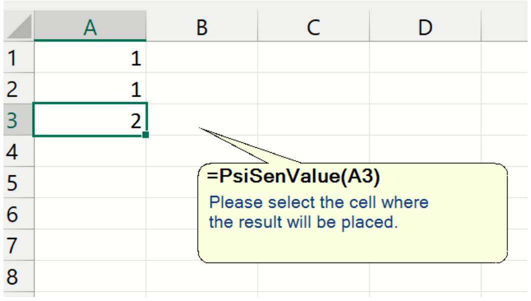
The simplest way to remove a parameter is to select the parameter cell on the worksheet and press the DEL key – which erases its cell formula – or type a number value that will overwrite the formula in the cell. The parameter will then disappear from the Task Pane.

Defining Results

When you perform a sensitivity analysis (which will automatically vary sensitivity parameters), or multiple optimizations or simulations (which will vary optimization, simulation or calculation parameters), Analytic Solver will keep track of one or more result cell values, corresponding to each parameter value.

- For optimizations, Analytic Solver tracks the final objective and decision variable values found by the Solver; you can also track a constraint value by setting its Monitor property in the Task Pane.
- For simulations, Analytic Solver tracks the values of all uncertain functions (simulation outputs, referenced in PsiOutput() or PSI Statistics function calls).
- For sensitivity analysis, Analytic Solver tracks results from the cell *currently selected* at the time you perform the analysis, and any other results you define.

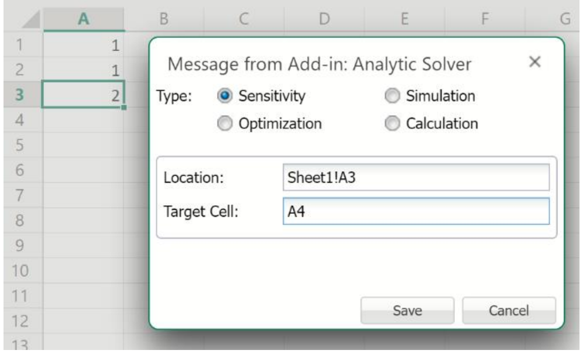
To define a sensitivity result cell in Analytic Solver, select the cell (such as A3 below – always a formula cell) on the worksheet, and choose ***Parameters – Monitor Value*** *from the Ribbon. A small "balloon" will appear, containing a function call such as =PsiSenValue(A3). You can move the mouse to "drag and drop" this formula into an* *empty cell. When you do this, cell A3 is defined as a sensitivity result cell, and appears in the Task Pane Model outline under Sensitivity Results.*



=PsiSenValue(A3)

Please select the cell where the result will be placed.

To define a sensitivity result cell in Analytic Solver Cloud, select the cell (such as A3 below) on the worksheet, and choose **Parameters – Monitor Value** from the Ribbon. A small dialog will appear, containing the location of the selected cell, a second Target field and three radio buttons. Select the type of parameter you would like to add, Sensitivity, Optimization, Simulation or Calculation, then enter an empty cell for the Target field, such as A4. Click Save to define cell A3 as a sensitivity result cell which will appear in the Task Pane Model outline under Sensitivity Results.


Message from Add-in: Analytic Solver

Type: ● Sensitivity ○ Simulation ○ Optimization ○ Calculation

Location: Sheet1!A3

Target Cell: A4

Identifying Inputs and Parameters

The new Identify Inputs tool (located on the Tools menu) identifies all constant parameters in an optimization, simulation or stochastic optimization model. If desired, Analytic Solver will apply formatting (cell color, border color/thickness) to those cells containing the parameters. RASON and Solver SDK users can utilize this feature to add input parameters to their model, providing the flexibility to change or update parameters without modifying the underlying code or structure of the model.

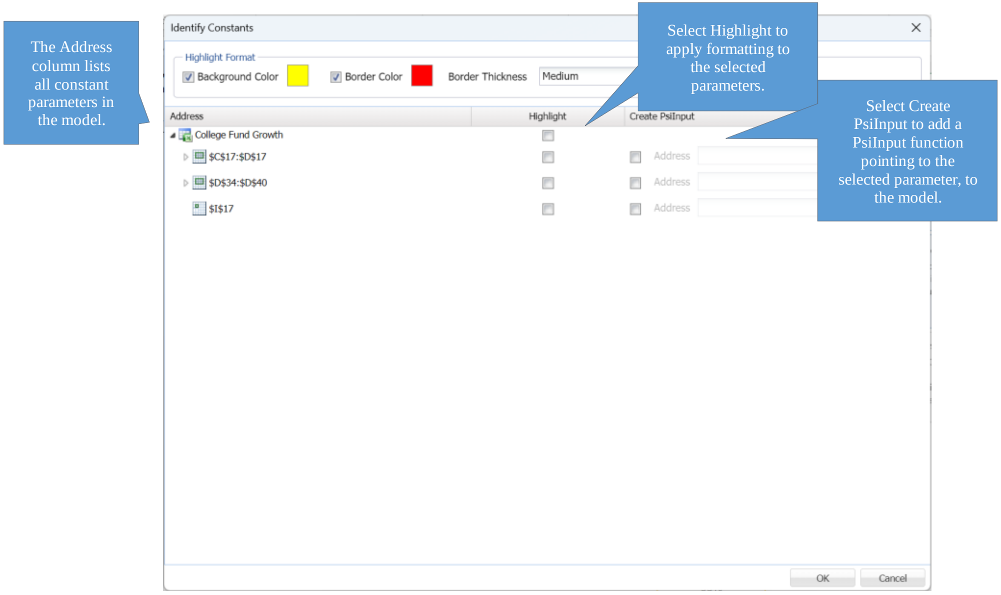
An input parameter is an argument or value that is passed into a function, model, or decision flow to provide input or information for its execution. Input parameters are used in Solver SDK and RASON Services to define the values or arrays that can be varied or modified by the user. They can be set or changed on the Editor tab of www.RASON.com by clicking the Select Query Parameters button on the Properties pane. These input parameters are then used in the model to perform calculations or analysis based on the provided values. Input parameters are essential for making RASON and SDK models more dynamic and adaptable to different scenarios or user requirements.

Using the Identify Input Tool

Open the College Fund Growth 2 example simulation model by clicking Help – Example Models – Monte Carlo Simulation Examples -- College Fund Growth 2. Recall that this example simulates a college fund account, set up for two children. This model opens with a yearly investment of $10,000 (D17) and a college loan interest rate of 8% (I17). Both are constant values.

1. Click Tools – Identify Inputs on the Analytic Solver ribbon to open the Identify Constants dialog. This dialog contains all constant parameters present in the model: C17:D17 (Beginning Balance and the first deposit), D34:D40 (Yearly Deposits) and I17 (Interest Rate).

   Note that if solving an optimization or stochastic optimization model, constant constraint right hand side values will appear on the Identify Constants dialog.



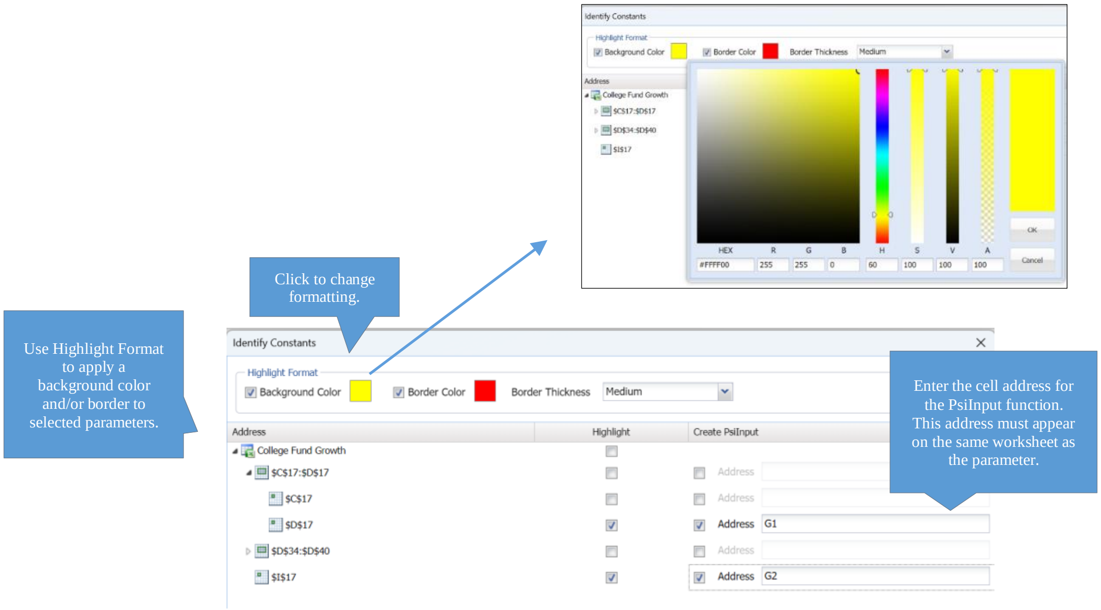
☑ Background Color 🟨 ☑ Border Color 🟥 Border Thickness Medium

The Address column lists all constant parameters in the model.

Select Highlight to apply formatting to the selected parameters.

Select Create PsiInput to add a PsiInput function pointing to the selected parameter, to the model.

2. Expand C17:D17. Click **Highlight** and **Create PsiInput** for both D17 and I17, then type a cell address for each, say **G1** for D17 and **G2** for I17.
   - Highlight: Analytic Solver will apply the Highlight Format to each cell.
   - Create PsiInput: Analytic Solver will create a PsiInput function in the provided cell address pointing to the parameter cell.
   - Click the color blocks next to Background Color and Border Color or the down arrow next to Border Thickness to change formatting from the defaults.



Use Highlight Format to apply a background color and/or border to selected parameters.

Click to change formatting.

Enter the cell address for the PsiInput function. This address must appear on the same worksheet as the parameter.

Identify Constants

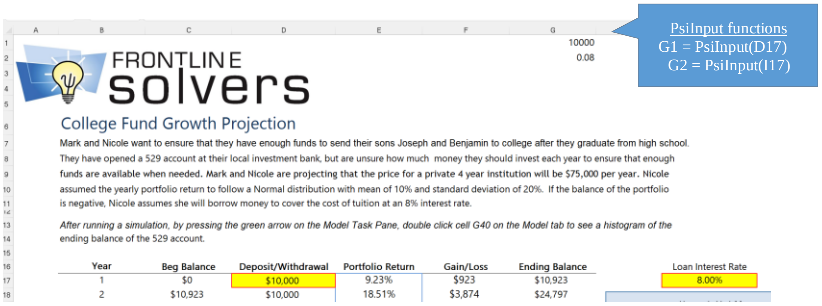
Highlight Format: ☑ Background Color, ☑ Border Color, Border Thickness: Medium

Address	Highlight	Create PsiInput
College Fund Growth	☐	
$C$17:$D$17	☐	☐ Address
$C$17	☐	☐ Address
$D$17	☑	☑ Address G1
$D$34:$D$40	☐	☐ Address
$I$17	☑	☑ Address G2

Click **OK** at the bottom of the dialog.

3. Formatting for cells D17 and I17 has been changed to a yellow background color with a red border. These cells now standout, allowing easy identification.

   Notice the two PsiInput functions in cells G1 and G2. These functions will allow easy access to both parameters when solving with either RASON or Solver SDK.



PsiInput functions
G1 = PsiInput(D17)
G2 = PsiInput(I17)

10000
0.08

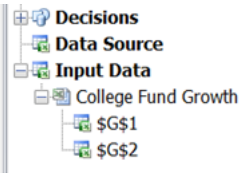
College Fund Growth Projection

Mark and Nicole want to ensure that they have enough funds to send their sons Joseph and Benjamin to college after they graduate from high school. They have opened a 529 account at their local investment bank, but are unsure how much money they should invest each year to ensure that enough funds are available when needed. Mark and Nicole are projecting that the price for a private 4 year institution will be $75,000 per year. Nicole assumed the yearly portfolio return to follow a Normal distribution with mean of 10% and standard deviation of 20%. If the balance of the portfolio is negative, Nicole assumes she will borrow money to cover the cost of tuition at an 8% interest rate.

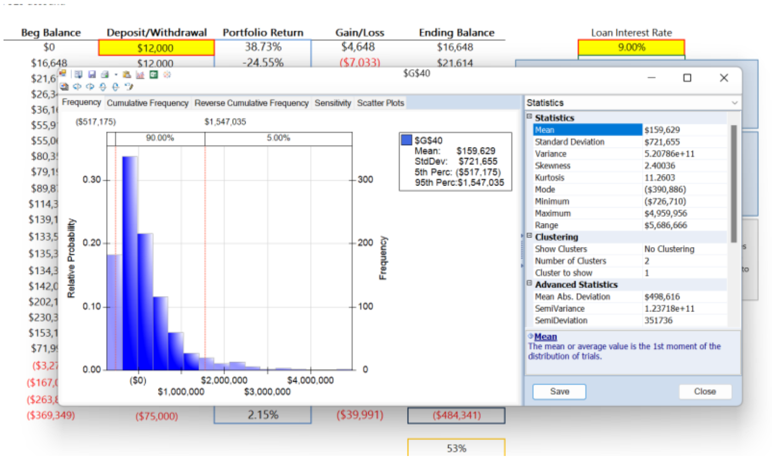
After running a simulation, by pressing the green arrow on the Model Task Pane, double click cell G40 on the Model tab to see a histogram of the ending balance of the 529 account.

Year	Beg Balance	Deposit/Withdrawal	Portfolio Return	Gain/Loss	Ending Balance	Loan Interest Rate
1	$0	$10,000	9.23%	$923	$10,923	8.00%
2	$10,923	$10,000	18.51%	$3,874	$24,797	

   Both Input functions are also listed on the Solver Task Pane under Input Data.



Enter 12,000 into cell D17 to change the yearly investment to $12,000 and enter .09 into cell I17 to change the interest rate to 9%. Click the Simulate button to run a new simulation.



Statistics	
Statistics	
Mean	$159,629
Standard Deviation	$721,655
Variance	5.20786e+11
Skewness	2.40036
Kurtosis	11.2603
Mode	($390,886)
Minimum	($726,710)
Maximum	$4,959,956
Range	$5,686,666
Clustering	
Show Clusters	No Clustering
Number of Clusters	2
Cluster to show	1
Advanced Statistics	
Mean Abs. Deviation	$498,616
SemiVariance	1.23718e+11
SemiDeviation	351736

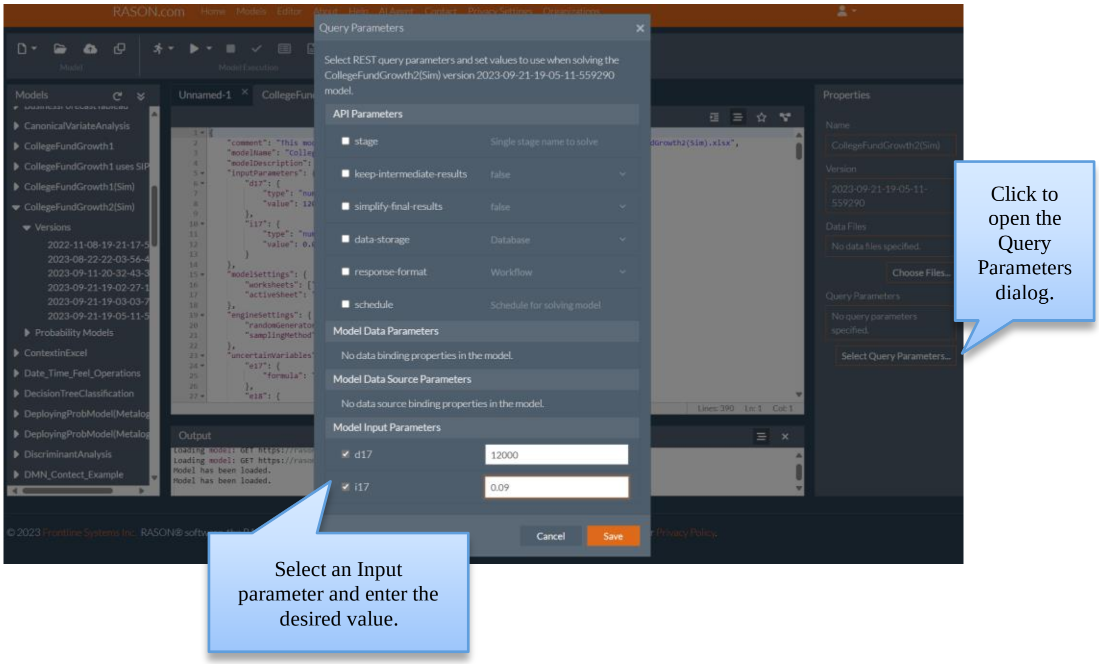
Mean
The mean or average value is the 1st moment of the distribution of trials.

Using a PsiInput Function with RASON and Solver SDK

Recall that two Input Parameters were created in cells G1 and G2. An Input Parameter provides easy access to constant values in a RASON or Solver SDK model. *Without* an Input Parameter, users must change the actual code in order to perform a "What if" analysis. *With* an Input Parameter, users can change the input *outside* of the coded model, thus preserving the original code.

4. Click Deploy Model – Cloud Service - Rason model to convert your model to a RASON model. A browser opens to the Editor tab of Frontline's RASON.com. On the Properties tab, click Query Parameters. Scroll to the bottom of the Query Parameters dialog to view Model Input Parameters. Select the checkboxes to enter a new value for each parameter. Input parameters provide a quick and easy way to change the value of each input parameter without the need to change the original code.

   For more information on solving a model in the RASON editor, see the Deploying your Model chapter that appears earlier in this guide.)



Query Parameters

Select REST query parameters and set values to use when solving the CollegeFundGrowth2(Sim) version 2023-09-21-19-05-11-559290 model.

API Parameters

☐ stage — Single stage name to solve
☐ keep-intermediate-results — false
☐ simplify-final-results — false
☐ data-storage — Database
☐ response-format — Workflow
☐ schedule — Schedule for solving model

Model Data Parameters

No data binding properties in the model.

Model Data Source Parameters

No data source binding properties in the model.

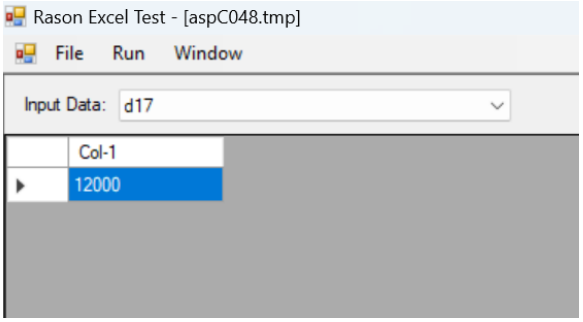
Model Input Parameters

☑ d17 — 12000
☑ i17 — 0.09

Click to open the Query Parameters dialog.

Select an Input parameter and enter the desired value.

5. Click Solver SDK – Excel Model to solve the model with Analytic Solver SDK. The SDK Test App opens displaying the first input parameter in cell D17. Click the down arrow next to Input Data to view the 2nd Input Parameter in cell I17.



Click the cell in the app and enter new values for each parameter, say $8,000 for D17 and 0.12 for I17, then click Run – Simulate (in the app). Immediately, a simulation is performed using 8,000 for the yearly investment and 12% for the interest rate. The results are printed at the bottom of the app.

For more information on the SDK Test App, see the Deploying Your Model chapter that appears earlier in this guide.

The Identify Input feature allows users to quickly identify constant model parameters in one spreadsheet or create an Input Parameter for a RASON or Solver SDK model. Use this feature to save time, easily perform what-if analyses and keep the SDK or RASON production model safely preserved.

Sensitivity Analysis Reports and Charts

The **Parameters – Identify** step just described performs a simple kind of sensitivity analysis; in some situations, this may be all that you need. But it is often useful to document the sensitivity of some computed result to systematic variations in the values of one or more parameters. Analytic Solver can produce both reports and charts to document these results for optimization, simulation and sensitivity. This feature is not supported for calculation.

How Parameters are Varied

In reports and charts, Analytic Solver can automatically vary your parameters' values in two ways:

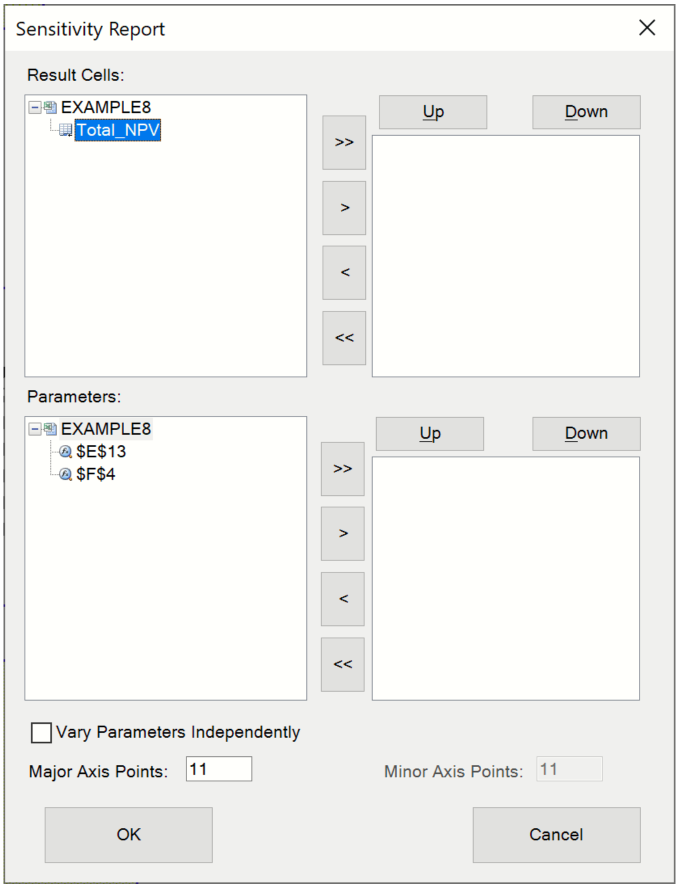
1. Varying **all parameters simultaneously**, from their respective lower to upper limits, for the number of steps that you specify.
2. Varying **two parameters independently**, from their lower to upper limits, computing a result for all combinations of the two parameters.

You select the second choice by checking the box "Vary Parameters Independently" at the bottom of the dialog shown on the next page. If you do this, you must select *exactly one* result cell, and *exactly two* parameter cells in the dialog below. A report will appear as a two dimensional table, and a chart will appear in 3-D with the two parameters on the axes.

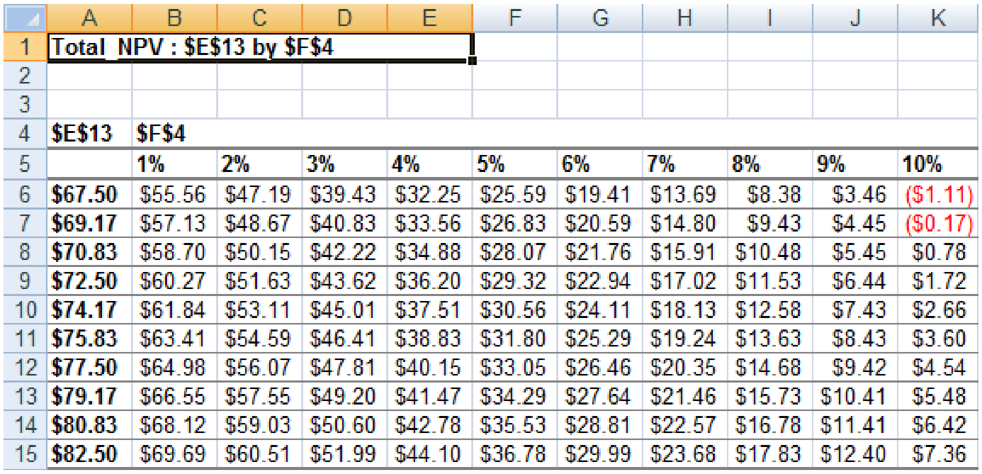
If you don't check this box – meaning that all parameters will be varied simultaneously – you can select one or more parameter cells, and one or more result cells in the dialog below. A report will contain a column for each parameter and each result, and a chart will contain a plot of each result cell.

Creating Sensitivity Reports

To create a sensitivity analysis report, select a formula cell on the worksheet (whose value should be computed in the report say, cell I28 (Total_NPV)), and choose **Reports – Sensitivity – Parameter Analysis** from the Ribbon. A dialog like the one shown below will appear.



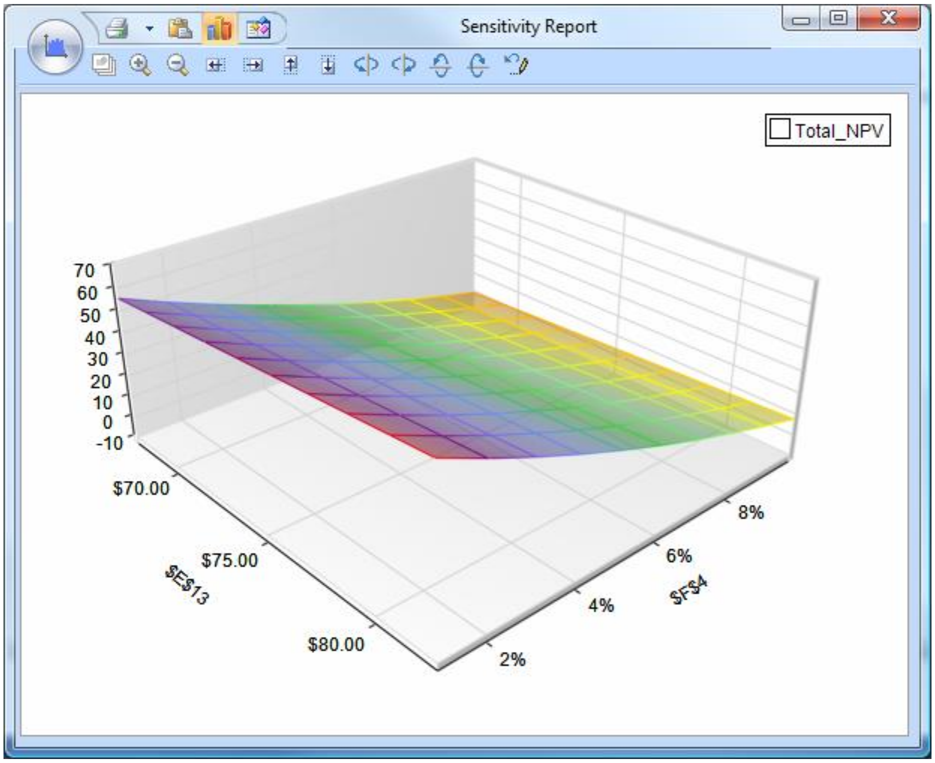
Use the arrow buttons in the top section to select one or more result cells (exactly one if you check the box “Vary Parameters Independently”). Use the arrow buttons in the bottom section to select one or more sensitivity parameter cells (exactly two if you check the box “Vary Parameters Independently”). Then click OK to produce the report, as a new worksheet in into your Excel workbook, just to the left of the active worksheet. An example is shown below.



	A	B	C	D	E	F	G	H	I	J	K
1	**Total_NPV : $E$13 by $F$4**										
2											
3											
4	**$E$13**	**$F$4**									
5		**1%**	**2%**	**3%**	**4%**	**5%**	**6%**	**7%**	**8%**	**9%**	**10%**
6	**$67.50**	$55.56	$47.19	$39.43	$32.25	$25.59	$19.41	$13.69	$8.38	$3.46	($1.11)
7	**$69.17**	$57.13	$48.67	$40.83	$33.56	$26.83	$20.59	$14.80	$9.43	$4.45	($0.17)
8	**$70.83**	$58.70	$50.15	$42.22	$34.88	$28.07	$21.76	$15.91	$10.48	$5.45	$0.78
9	**$72.50**	$60.27	$51.63	$43.62	$36.20	$29.32	$22.94	$17.02	$11.53	$6.44	$1.72
10	**$74.17**	$61.84	$53.11	$45.01	$37.51	$30.56	$24.11	$18.13	$12.58	$7.43	$2.66
11	**$75.83**	$63.41	$54.59	$46.41	$38.83	$31.80	$25.29	$19.24	$13.63	$8.43	$3.60
12	**$77.50**	$64.98	$56.07	$47.81	$40.15	$33.05	$26.46	$20.35	$14.68	$9.42	$4.54
13	**$79.17**	$66.55	$57.55	$49.20	$41.47	$34.29	$27.64	$21.46	$15.73	$10.41	$5.48
14	**$80.83**	$68.12	$59.03	$50.60	$42.78	$35.53	$28.81	$22.57	$16.78	$11.41	$6.42
15	**$82.50**	$69.69	$60.51	$51.99	$44.10	$36.78	$29.99	$23.68	$17.83	$12.40	$7.36

Creating Sensitivity Charts

To create a sensitivity analysis chart, select a formula cell on the worksheet (whose value should be computed in the chart), and choose **Charts – Sensitivity – Parameter Analysis** from the Ribbon. A dialog like the one on the previous page will appear. Follow the same steps as for sensitivity analysis reports, and click OK. A chart like the one below will appear:



You can use icons on the toolbar at the top of the chart to resize and rotate the chart, print the chart, or copy it to the Windows Clipboard (where you can paste it into other applications).

Optimization and Simulation Reports and Charts

Reports and charts for multiple parameterized optimizations and simulations work in much the same way as the reports and charts for sensitivity analysis shown in the previous section. But where for sensitivity analysis, Analytic Solver simply performs a **worksheet recalculation**, for these reports and charts, Analytic Solver performs a **complete optimization or simulation** for each set of parameter values.

To produce these reports and charts, select **Reports** or **Charts – Multiple Optimizations** from the Ribbon. For multiple parameterized simulations, select **Reports** or **Charts – Multiple Simulations** from the Ribbon.

When Optimizations and Simulations are Run

As shown in the earlier Examples chapters, optimizations are run when you click the Optimize button on the Ribbon, or the green arrow in the Task Pane, and simulations are run when you click the Simulate button on the Ribbon, or (assuming there's no optimization model) when you click the green arrow in the Task Pane. When you select Reports and Charts of Multiple Optimizations or Multiple Simulations *other than* Parameter Analysis, the results **cached in memory** from the last optimization or simulation run are used to create the reports or charts.

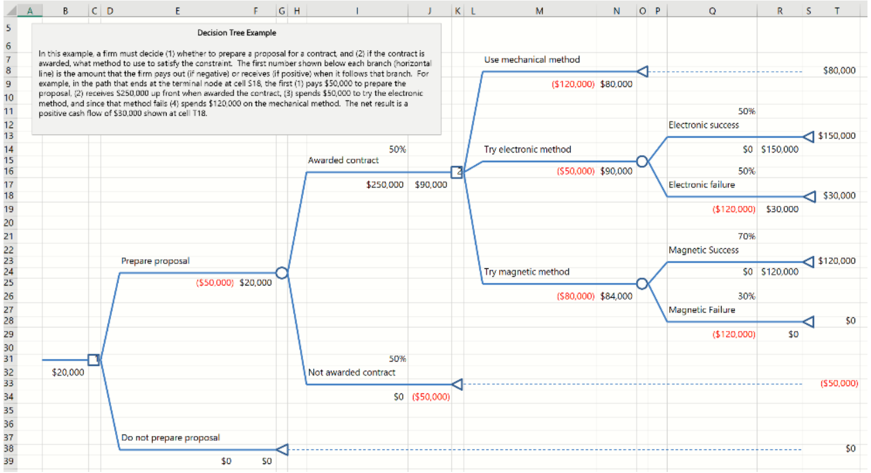
However, when you select Reports and Charts of Multiple Optimizations or Multiple Simulations with **Parameter Analysis**, which gives you greater control over how parameters are varied, Analytic Solver cannot use the results cached in memory. Instead, a new set of optimizations or simulations is run, varying the values of the parameters as you've specified; you can for example vary two parameters independently, yielding all combinations of these parameter values. The number of optimizations or simulations to run is independent of the settings you use for Number of Optimizations or Number of Simulations in the Task Pane Platform tab; it is the ***product*** of the number of **Major Axis Points** and **Minor Axis Points** you select in the Reports or Charts selection dialog. With the default value of 10 points each, 100 optimizations or simulations are required. This can take some time, especially for optimizations; a progress dialog is displayed during this process.

Examples: Decision Trees

Introduction

This chapter introduces **decision trees** in Analytic Solver. Such trees are a useful graphical and computational aid for problems that involve sequential decisions and events, where there are a small number of decision alternatives at each step, and a small number of alternative outcomes for each event.

In the example below, we've clicked **Help – Examples** on the ribbon, clicked **Decision Tree Examples** on the Overview Tab, and clicked on **TreeExample.xlsx**, which is based on an example by Prof. Mike Middleton.



Decision Tree Example

In this example, a firm must decide (1) whether to prepare a proposal for a contract, and (2) if the contract is awarded, what method to use to satisfy the constraint. The first number shown below each branch (horizontal line) is the amount that the firm pays out (if negative) or receives (if positive) when it follows that branch. For example, in the path that ends at the terminal node at cell S18, the first (1) pays $50,000 to prepare the proposal, (2) receives $250,000 up front when awarded the contract, (3) spends $50,000 to try the electronic method, and since that method fails (4) spends $120,000 on the mechanical method. The net result is a positive cash flow of $30,000 shown at cell T18.

In this example, a firm must decide (1) **whether to prepare a proposal** for a contract, and (2) if the contract is awarded, **what method to use** to satisfy the contract. The first number shown below each branch (horizontal line) is the amount the firm pays out (if negative) or receives (if positive) when it follows that branch. For example, in the path that ends at the terminal node at cell S18, the firm (1) pays $50,000 to prepare the proposal, (2) receives $250,000 up front when awarded the contract, (3) spends $50,000 to try the electronic method, and since that method fails, (4) spends $120,000 on the mechanical method. The net result is a positive cash flow of $30,000 shown at cell T18.

A decision tree is "solved" by Excel fomulas calculating each of the numbers you see on the worksheet. **Terminal values** such as T18 sum all the partial cash flows along the path leading to that terminal node. The tree is then **rolled back** by computing expected values (or certainty equivalents) at event nodes, and by maximizing (or minimizing) the alternative values at decision nodes. The rollback values appear below and just to the left of each node, showing the value of the subtree rooted at that node. The numbers "inside the boxes" at decision nodes show **which alternative is optimal** for that decision. In the example, the 1 in the decision node at C31 indicates that it is optimal to prepare the proposal, and the 2 in the decision node at K16 indicates the firm should try the electronic method because that alternative leads to a higher expected value ($90,000) than the mechanical method ($80,000).

Since all the computations are performed with Excel worksheet formulas, a decision tree is a ‘regular’ Excel worksheet model, where you can ask what-if, perform sensitivity analysis, or create simulation or optimization models if desired. For example, the formula at cell F25, for the event where we are or are not awarded the contract, is =IF(ABS(1-SUM($I$14,$I$31))<=0.00001,SUM($I$14*$J$17,$I$31*$J$34),NA()) – an expected value calculation – and the formula at B32, for the decision whether to prepare a proposal, is simply =MAX(F25,F39), since we are maximizing EV.

The Task Pane Platform tab includes a Decision Tree group of options, where you can choose to maximize or minimize values at decision nodes, and to compute expected values (which are risk neutral) or certainty equivalents (which reflect risk aversion) at each node. If you choose certainty equivalents, you can specify the parameters of an exponential utility function.

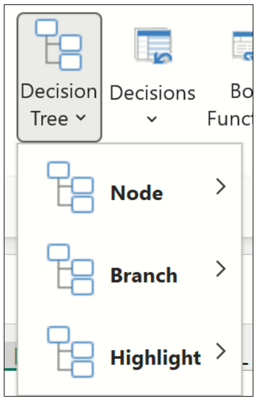
Creating Decision Trees

A decision tree consists of **nodes** and **branches**. Nodes may be either decision **nodes**, **event nodes**, or **terminal nodes**; branches represent alternative choices at decision nodes, and alternative outcomes at event nodes.

You create and edit a decision tree with the Decision Tree choice on the Ribbon.

Creating Trees in Analytic Solver Desktop

Click the Decision Tree icon on the Ribbon to open the menu choices.

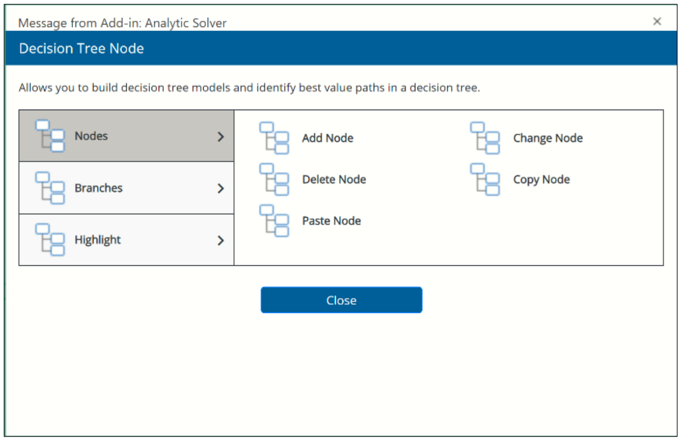


You can also create and edit a decision tree in Analytic Solver Desktop in the Task Pane Model tab, using the Add (+) and Remove (X) icons and the properties area at the bottom of the Task Pane; but you can highlight the best or worst decision strategy (see below) only from the Ribbon.

Note: When creating or modifying decision trees in either Analytic Solver Cloud or Desktop, it’s important to use the decision tree dialogs. These may be opened by clicking Decision Trees on the ribbon. Changes made directly to an Excel cell will be deleted once the tree is drawn or redrawn on the worksheet.

Creating Trees in Analytic Solver Cloud

If using Analytic Solver Cloud, clicking the Decision Tree icon on the Ribbon brings up the Decision Tree dialog which is where nodes and branches may be added/deleted/edited and the best and worst paths may be highlighted.

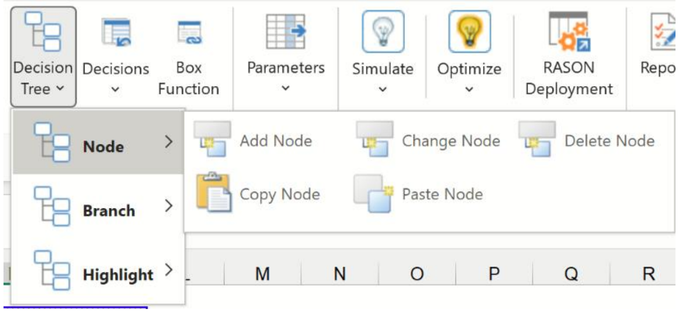

Message from Add-in: Analytic Solver

Decision Tree Node

Allows you to build decision tree models and identify best value paths in a decision tree.

Creating and Editing Nodes

Selecting **Node** in the dropdown presents choices for working with nodes:

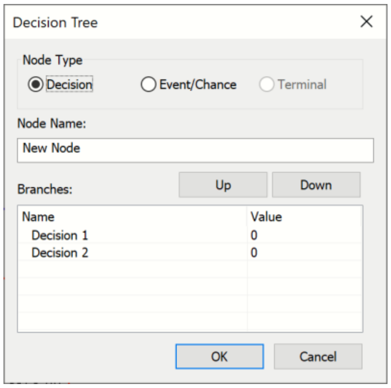


The **Add Node** and **Change Node** options display the dialog shown on the next page. To change a node, you must first *select* it on the Excel worksheet – you may select either the node graphic or one of the immediately adjacent cells.

The **Copy Node** and **Paste Node** choices can be used to copy a subtree (rooted at the selected node) and paste the copy at another position in the decision tree; this is very useful when there are identical choices at later stages in the tree.

In Analytic Solver Cloud

In Analytic Solver Cloud, click **Decisions** on the ribbon to open the Decision Tree dialog, then click **Add Node** to open the Add Node dialog.



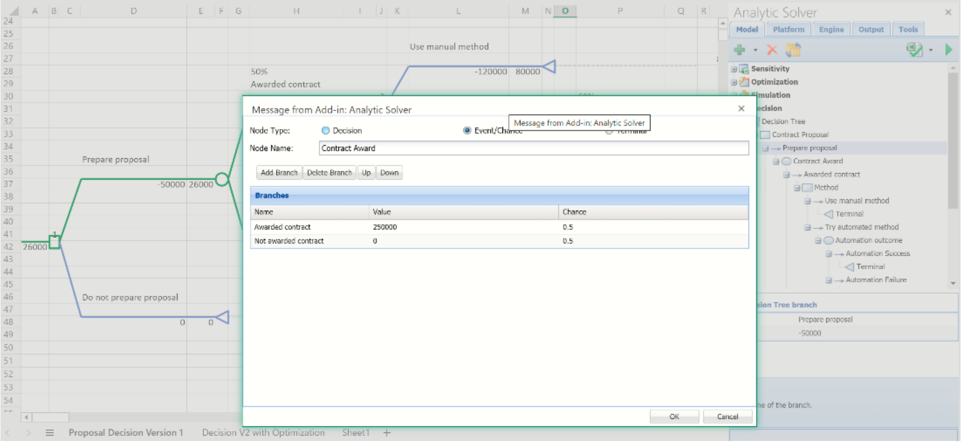
Node Type: ● Decision ○ Event/Chance ○ Terminal

Node Name: New Node

Branches:

Name	Value
Decision 1	0
Decision 2	0

To change a node, you must first select the desired node on the dialog. In the example below, the *Prepare proposal* node has been selected in the taskpane (on the right) before clicking **Decisions – Node -- Change Node**.



Node Type: ○ Decision ● Event/Chance

Node Name: Contract Award

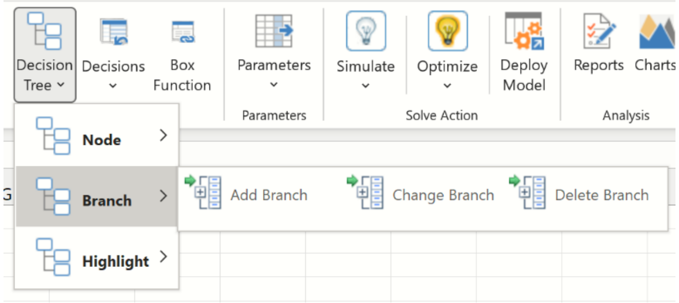
Branches

Name	Value	Chance
Awarded contract	250000	0.5
Not awarded contract	0	0.5

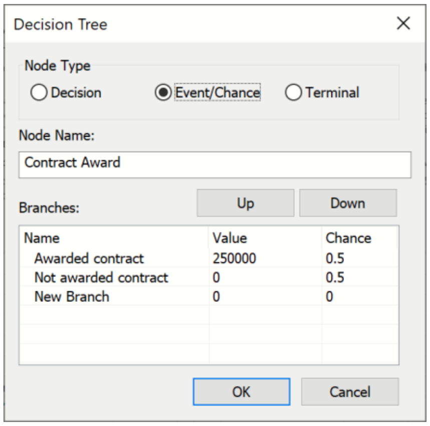
Creating and Editing Branches

In Analytic Solver Desktop

Selecting **Branch** in the dropdown presents choices for working with branches:



The **Add Branch** and **Change Branch** options display the Decision Tree dialog shown below. To change a branch, you must first *select* it on the Excel worksheet – you may select either the branch graphic or one of the immediately adjacent cells. Note that Add/Change Node and Add/Change Branch options display the *same* Decision Tree dialog: Branches are associated with the node to their left, and you can edit properties of the node and its branches in this dialog.



Node Type: ○ Decision ● Event/Chance ○ Terminal

Node Name: Contract Award

Branches:

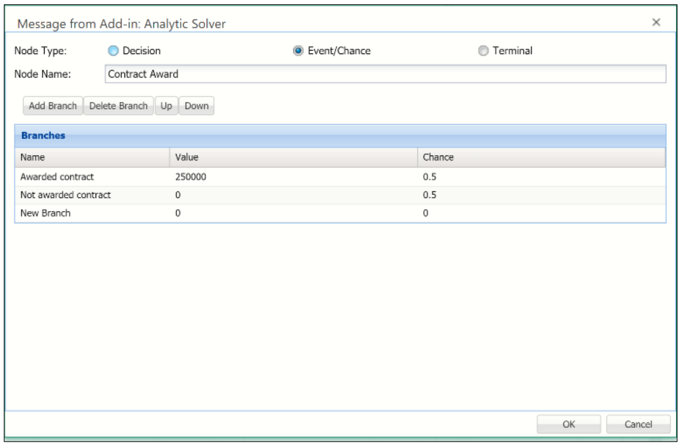
Name	Value	Chance
Awarded contract	250000	0.5
Not awarded contract	0	0.5
New Branch	0	0

In this dialog, you can change the type of the node, the name of the node, and you can add, edit, reorder, or remove branches for this node. Each branch for a **decision node** has a **value** (the cost or payoff of taking that decision); each branch for an **event node** has a **value** and a **probability** (the cost or payoff, and the probability of occurrence, of that outcome).

In Analytic Solver Cloud

In Analytic Solver Cloud, click Decisions to bring up the Decision Tree dialog, then click either the **Add Branch** or **Change Branch** option to bring up the Add/Change Branch dialog. As in the Analytic Solver Desktop, to change a branch, you must first *select* it on the Excel worksheet. Branches are associated with the node to their left, and you can edit properties of the node and its branches here.

In this dialog, as in Analytic Solver Desktop, you can change the type of the node, the name of the node, and you can add, edit, reorder, or remove branches for this node. Each branch for a **decision node** has a **value** (the cost or payoff of taking that decision); each branch for an **event node** has a **value** and a **probability** (the cost or payoff, and the probability of occurrence, of that outcome).



Node Type: ○ Decision ● Event/Chance ○ Terminal

Node Name: Contract Award

Branches

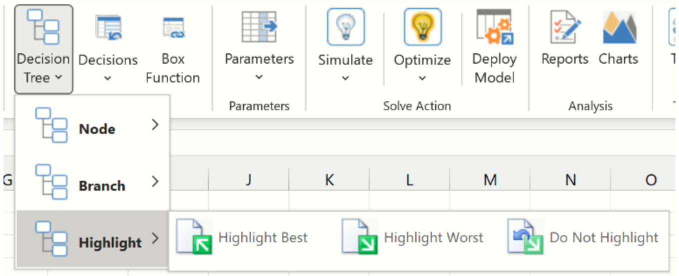
Name	Value	Chance
Awarded contract	250000	0.5
Not awarded contract	0	0.5
New Branch	0	0

Highlighting a Decision Strategy

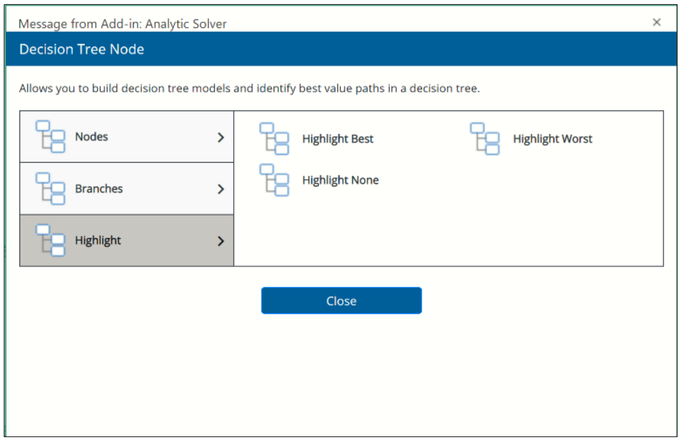
A *decision strategy* is a complete sequence of decisions that you can make through the decision tree. You can highlight the best or worst decision strategy by selecting **Highlight** in the Decision Tree dropdown list in Analytic Solver

Desktop or by clicking **Highlight Best** in the Decision Tree dialog in Analytic Solver Cloud, as shown on the next page. If you are maximizing, the best strategy maximizes EV or CE, and the worst strategy minimizes EV or CE. If you are minimizing, the opposite applies.

Analytic Solver Desktop



Analytic Solver Cloud

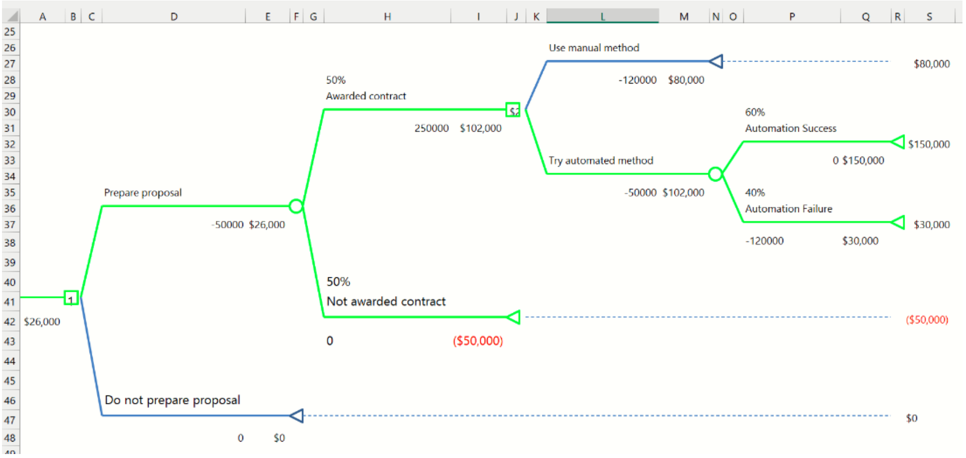

Message from Add-in: Analytic Solver

Decision Tree Node

Allows you to build decision tree models and identify best value paths in a decision tree.

The next page shows the example decision tree shown earlier, with Excel worksheet gridlines turned off, and the best decision strategy highlighted.

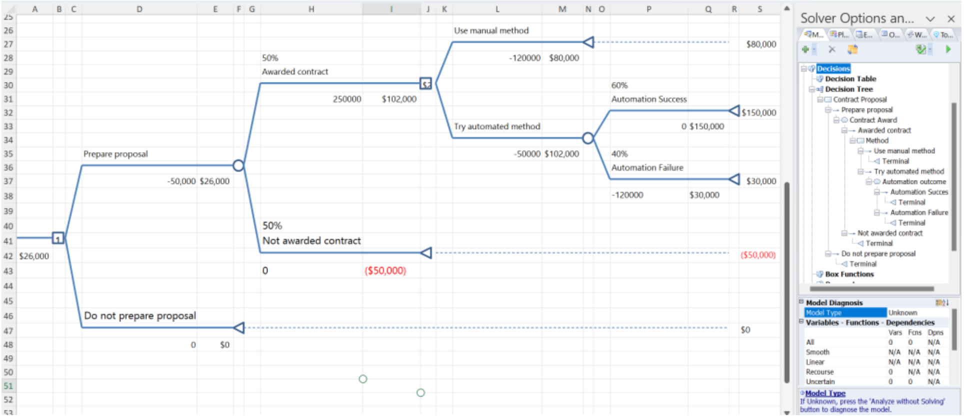
Analytic Solver Desktop



Decision Trees in the Task Pane

Model Tab

Decision trees may also be viewed and edited directly in the Task Pane Model tab, and choices for tree evaluation (see “Platform tab” below) can only be made in the Task Pane. When a node or branch is selected in the Task Pane outline, properties of that node or branch may be viewed and edited in the lower portion of the Task Pane, as shown on the next page. You can also **double-click a node** in the Model outline to display the same Decision Tree dialog that appears when you add or change a node from the Ribbon.



Model Type: Unknown

	Vars	Fcns	Dpns
All	0	0	N/A
Smooth	N/A	N/A	N/A
Linear	N/A	N/A	N/A
Recourse	0	N/A	N/A
Uncertain	0	0	N/A

Model Type
If Unknown, press the 'Analyze without Solving' button to diagnose the model.

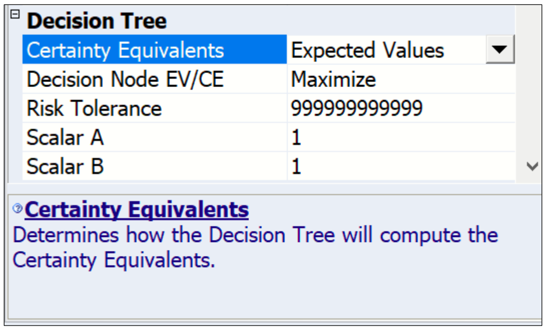
In this example, we’ve selected the decision node named Method, represented by the graphic symbol at cell K16 on the worksheet. The optimal choice at this node (given that we’re maximizing expected value or EV) is branch #2, and the rollback value of the subtree rooted at this node is 90,000.

Note the small **Copy** icon in the upper right corner of the Decision Tree Node properties pane: You can click this icon to **copy a subtree** rooted at this node. When you’ve selected a Terminal node in the Model outline, you’ll notice a small **Paste** icon in this same area: You can click this icon to **paste a subtree** at the position of the Terminal node.

Platform Tab

By default, rollback values for a decision tree are computed assuming that you want to Maximize Expected Value or EV. You can change this by setting options in the Decision Tree section of the Task Pane Platform tab, as shown on the next page.

You can select **Maximize** or **Minimize** in the dropdown list for the Decision Node EV/CE option. For the Certainty Equivalents option, you can choose either **Expected Value** or **Exponential Utility Function**. The remaining three options come into play only when you choose Exponential Utility Function – they set parameters of this function.



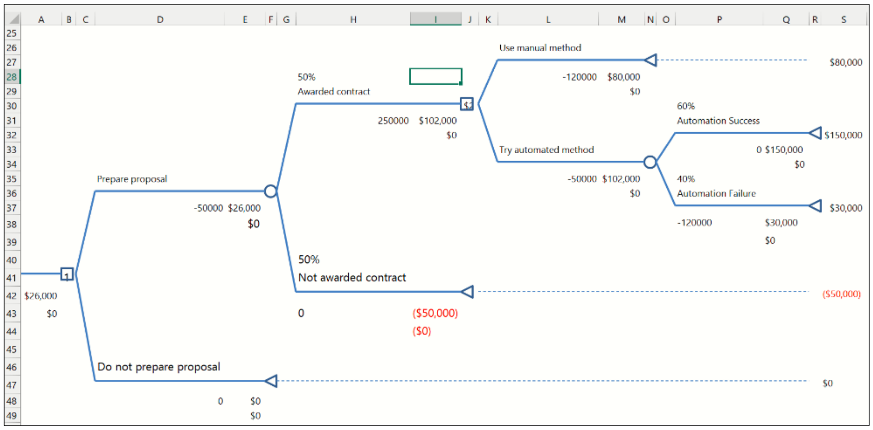
Decision Tree

Certainty Equivalents	Expected Values
Decision Node EV/CE	Maximize
Risk Tolerance	999999999999
Scalar A	1
Scalar B	1

Certainty Equivalents
Determines how the Decision Tree will compute the Certainty Equivalents.

For the **Maximize** option with exponential utility, the rollback formulas are U = A–B*EXP(-X/RT) and CE = -LN((A-EU)/B)*RT, where X and EU are cell references. For the **Minimize** option with exponential utility, the formulas are U = A-B*EXP(-X/RT) and CE = LN((A-EU)/B)*RT. RT is the value you specify for the **Risk Tolerance** option on the Platform tab, and A and B are the values of the **Scalar A** and **Scalar B** options on the Platform tab.

When you use an Exponential Utility Function, the utility (U =) and certainty equivalent (CE =) formulas are computed in extra cells on the worksheet, as shown on the next page. For example, B33 computes the **utility** of the decision at B32; B32 contains =MAX(F25,F39) as before, to choose the decision with the maximum **certainty equivalent** value; F25 contains =-LN((A-$F$26)/B)*RT.



Getting Results: Optimization

Introduction

This chapter explains how to obtain and interpret results from optimization in Analytic Solver Comprehensive and its subset optimization product, Analytic Solver Optimization, when using either Analytic Solver Desktop or Cloud. We'll also discuss **what can go wrong**, and what to do about it, and also **how to get more** than a single optimal solution from your model.

Analytic Solver Optimization has many powerful optimization algorithms, and will fully exploit the power of your PC. But the *model that you create* may be relatively easy to optimize (if you use linear functions like SUM) or extremely difficult to optimize (if you use non-convex and non-smooth functions like LOOKUP). The results you get depend on the model you create.

In the following sections, we'll focus on immediate actions you can take when you get an unexpected result – but if you read the chapter "Mastering Conventional Optimization Concepts," you'll learn more about optimization models and solution methods, and better understand *why* the unexpected result appeared, and how to design your model to get the solutions you want.

What Can Go Wrong, and What to Do About It

When you click the **Optimize** button on the Ribbon, or the **green arrow** on the Task Pane to solve, you'll normally get one of these outcomes:

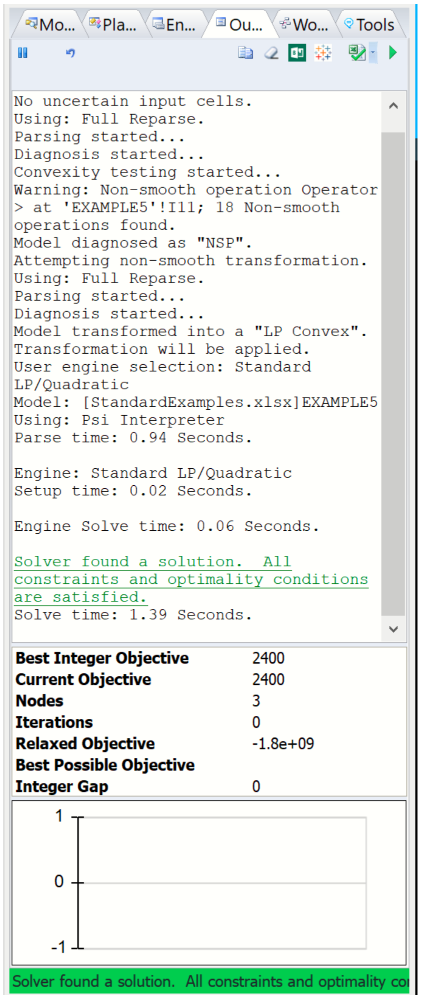
1. A solution that makes sense to you. This is normally accompanied by a Solver Result message in **green** at the bottom of the Task Pane. You can proceed to "When Things Go Right: Getting Further Results."
2. A Solver Result error message that you understand and can correct, in **red** at the bottom of the Task Pane. You can take corrective action.
3. A Solver Result error message that you *don't* understand, in **red** at the bottom of the Task Pane. You should **read the solution log** in the Output tab, **click the error message** and read Help about the message.
4. A *solution* that you *don't* understand, or that seems wrong. Again before doing anything else, you should **read the solution log** in the Output tab, **click the error message** to display Help, **run available reports** as described below, and **read the section** below "When the Solution Seems Wrong."
5. Solving runs for a very long time, and you don't get a solution or a Solver Result message until you press ESC or click Pause/Stop. You should **read the section** below "When Solving Takes a Long Time."

In rare cases, you might find Excel shutting down or "locking up" (so nothing happens when you click Pause on the Solver Task Pane). In this case please contact Frontline Systems Technical Support at (775) 831-0300 x4 or support@solver.com. Some Solver Result error messages ask you to contact Technical Support. If you can send us your model, this will be very helpful.

But experience shows that 99% of all technical support cases involve “pilot error” by the user, and that 90% of all such cases could be easily resolved by reading online Help or the User Guide. So we hope you’ll keep reading, and that you’ll take these steps before calling technical support!

Review Messages in the Output Tab

If using Analytic Solver Desktop, your first step should be to **review the messages** in the **solution log** in the Task Pane Output tab. Below is an example of the Output tab at the solution of EXAMPLE5 in **StandardExamples.xlsx** (described more fully in the chapter “Examples: Conventional Optimization”):



```
No uncertain input cells.
Using: Full Reparse.
Parsing started...
Diagnosis started...
Convexity testing started...
Warning: Non-smooth operation Operator
> at 'EXAMPLE5'!I11; 18 Non-smooth
operations found.
Model diagnosed as "NSP".
Attempting non-smooth transformation.
Using: Full Reparse.
Parsing started...
Diagnosis started...
Model transformed into a "LP Convex".
Transformation will be applied.
User engine selection: Standard
LP/Quadratic
Model: [StandardExamples.xlsx]EXAMPLE5
Using: Psi Interpreter
Parse time: 0.94 Seconds.

Engine: Standard LP/Quadratic
Setup time: 0.02 Seconds.

Engine Solve time: 0.06 Seconds.

Solver found a solution.  All
constraints and optimality conditions
are satisfied.
Solve time: 1.39 Seconds.
```

Best Integer Objective	2400
Current Objective	2400
Nodes	3
Iterations	0
Relaxed Objective	-1.8e+09
Best Possible Objective	
Integer Gap	0

Solver found a solution. All constraints and optimality co

If you click the Copy icon, the contents of the solution log will be copied to the Windows Clipboard, where you can paste it into Microsoft Word, NotePad, or an **email message to Frontline Systems Technical Support**. Below is the complete solution log from the above example:

```
---- Start Solve ----
No uncertain input cells.
Using: Full Reparse.
Parsing started...
Diagnosis started...
Convexity testing started...
```

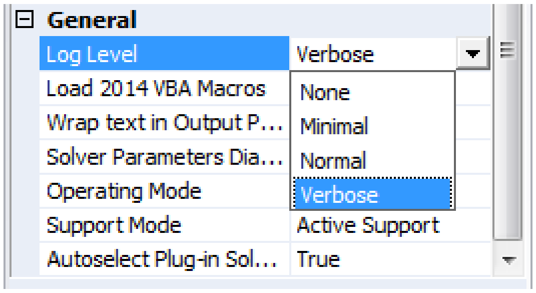
```
Warning: Non-smooth operation Operator > at 'EXAMPLE5'!I11; 18 Non-
smooth operations found.
Model diagnosed as "NSP".
Attempting non-smooth transformation.
Using: Full Reparse.
Parsing started...
Diagnosis started...
Model transformed into a "LP Convex".
Transformation will be applied.
User engine selection: Standard LP/Quadratic
Model: [StandardExamples.xlsx]EXAMPLE5
Using: Psi Interpreter
Parse time: 0.94 Seconds.
Engine: Standard LP/Quadratic
Setup time: 0.02 Seconds.
Engine Solve time: 0.06 Seconds.
Solver found a solution. All constraints and optimality conditions
are satisfied.
Solve time: 1.39 Seconds.
```

This was a *successful* solution – but in cases where you have a Solver Result *error message* that you don't understand, or a *solution* that you don't understand, the solution log can be quite helpful.

Note: Information related to the solving process in Analytic Solver Cloud is not available during parsing of the model. As a result, feedback is only available once the Solver Engine starts to solve the model.

More Detail in the Solution Log

You can obtain *more detailed* output in the solution log by setting the Task Pane Platform tab General group **Log Level** option to **Verbose** before you solve:



General

Log Level	Verbose
Load 2014 VBA Macros	
Wrap text in Output P...	
Solver Parameters Dia...	
Operating Mode	
Support Mode	Active Support
Autoselect Plug-in Sol...	True

Below is a portion of the solution log (from the LP/Quadratic Solver) for the EXAMPLE5 model when Log Level is set to Verbose.

```
---- Start Solve ----
No uncertain input cells.
Using: Full Reparse.
Releasing Memory...
Releasing Memory...
Analyzing Solver Model...
Parsing started...
Diagnosis started...
Convexity testing started...
Warning: Non-smooth operation Operator > at 'EXAMPLE5'!I11; 18 Non-
smooth operations found.
Model diagnosed as "NSP".
Attempting non-smooth transformation.
Using: Full Reparse.
Releasing Memory...
Releasing Memory...
Analyzing Solver Model...
Parsing started...
Diagnosis started...
Model transformed into a "LP Convex".
Transformation will be applied.
```

```
User engine selection: Standard LP/Quadratic
Model: [StandardExamples.xlsx]EXAMPLE5
Using: Psi Interpreter
Parse time: 1.02 Seconds.
Engine: Standard LP/Quadratic
Setup time: 0.01 Seconds.
13 1
13 2
Presolve 53 -13 27 0 112 -23
Presolve 0.01 0.01
Bounds were tightened 36
Initial range of elements is 1 1000000
Range of elements is 0.001 1000
Range of elements is 0.42180184 2.3707815
Final range of elements is 0.42201258 1
Absolute values of scaled rhs range from 2.6471969e-09 421.80226
1e+100
Absolute values of scaled objective range from 56.262229 1185390.8
Absolute values of scaled bounds range from 8.8869569 421.80189
17.791688
Primal error 0 0
Primal error 0 0
0 -3.5800379e+09 3856.921 23 0 0 0
Primal error 7.1054274e-15 1e-18
24 -1.799998e+09 0 0 0 0 0
Looking optimal checking bounds with 1e+10
Going back to original objective
Primal error 7.1054274e-15 0
Looking optimal checking bounds with 1e+10
Optimal - objective value -1.799998e+09
Dual 0.058 0.068
After Postsolve, objective -1.799998e+09 0 0 0 0
Optimal -1.799998e+09 24 0.074 0.016 0
13 1
13 2
Presolve 53 -13 27 0 112 -23
Presolve 0.012 0.012
Bounds were tightened 36
Initial range of elements is 1 1000000
Range of elements is 0.001 1000
Range of elements is 0.42180184 2.3707815
Final range of elements is 0.42201258 1
Absolute values of scaled rhs range from 2.6471969e-09 421.80226
1e+100
Absolute values of scaled objective range from 56.262229 1185390.8
Absolute values of scaled bounds range from 8.8869569 421.80189
17.791688
Primal error 0 0
Primal error 0 0
0 -3.5800379e+09 3856.921 23 0 0 0
Primal error 7.1054274e-15 1e-18
24 -1.799998e+09 0 0 0 0 0
Looking optimal checking bounds with 1e+10
Going back to original objective
Primal error 7.1054274e-15 0
Looking optimal checking bounds with 1e+10
Optimal - objective value -1.799998e+09
Dual 0.46 0.472
After Postsolve, objective -1.799998e+09 0 0 0 0
Optimal -1.799998e+09 24 0.483 0.023 0
Initial range of elements is 1 1000000
Range of elements is 0.001 1000
Range of elements is 0.31622777 3.1622777
Final range of elements is 0.17782794 1
Absolute values of scaled rhs range from 2.6471969e-09 421.80226
316.22745
Absolute values of scaled objective range from 21.779386 1581138.8
Absolute values of scaled bounds range from 1 400.75427 23.707815
Primal error 7.1054274e-15 3.8687961e-15
5
Primal error 7.1054274e-15 3.8687961e-15
```

```
0 -3.5800361e+09 3162.37 18 0 0 0
Primal error 5.6843419e-14 5.0709084e-10
20 -1.5999968e+09 0 0 0 0 0
Looking optimal checking bounds with 1e+10
Going back to original objective
Primal error 5.6843419e-14 5.0709084e-10
Looking optimal checking bounds with 1e+10
Optimal - objective value -1.5999968e+09
Initial range of elements is 1 1000000
Range of elements is 0.001 1000
Range of elements is 0.31622777 3.1622777
Final range of elements is 0.17782794 1
Absolute values of scaled rhs range from 6.8418647e-09 332.66862
316.22745
Absolute values of scaled objective range from 18.707313 1581138.8
Absolute values of scaled bounds range from 28.117066 332.6683
56.234133
Primal error 5.6843419e-14 5.0709084e-10
Primal error 5.6843419e-14 5.0709084e-10
0 -1.5999968e+09 0 0 0 0 0
Optimal - objective value -1.5999968e+09
Initial range of elements is 1 1000000
Range of elements is 0.001 1000
Range of elements is 0.31622777 3.1622777
Final range of elements is 0.17782794 1
Absolute values of scaled rhs range from 6.8418647e-09 332.66862
316.22745
Absolute values of scaled objective range from 18.707313 1581138.8
Absolute values of scaled bounds range from 0.00031622777 332.66829
0.00031622777
Primal error 5.6843419e-14 5.0709084e-10
Primal error 5.6843419e-14 5.0709084e-10
0 -1.5999968e+09 1314.2309 8 0 0 0
Primal error 5.6843419e-14 3.8687961e-15
14 600 0 0 0 0 0
Looking optimal checking bounds with 1e+10
Going back to original objective
Primal error 5.6843419e-14 3.8687961e-15
Looking optimal checking bounds with 1e+10
Optimal - objective value 600
Absolute values of scaled rhs range from 6.8418647e-09 332.66862
316.22745
Absolute values of scaled objective range from 18.707313 1581138.8
Absolute values of scaled bounds range from 0.00031622777 332.66829
0.00031622777
Primal error 5.6843419e-14 3.8687961e-15
Primal error 5.6843419e-14 3.8687961e-15
Optimal - objective value 600
Initial range of elements is 1 1000000
Range of elements is 0.001 1000
Range of elements is 0.31622777 3.1622777
Final range of elements is 0.17782794 1
Absolute values of scaled rhs range from 6.8418647e-09 332.66862
316.22745
Absolute values of scaled objective range from 18.707313 1581138.8
Absolute values of scaled bounds range from 0.00031622777 332.66829
0.00031622777
Primal error 5.6843419e-14 3.8687961e-15
Primal error 5.6843419e-14 3.8687961e-15
0 600 0 0 0 0 0
Optimal - objective value 600
Initial range of elements is 1 1000000
Range of elements is 0.081649658 12.247449
Range of elements is 0.28574404 3.4996355
Final range of elements is 1e+50 1
Absolute values of scaled rhs range from 1.4901161e-08 1000001
999999
Absolute values of scaled objective range from 10 500
Absolute values of scaled bounds range from 1 100 1e+100
Primal error 0 0
Primal error 0 0
```

```
Optimal - objective value 2400
Initial range of elements is 1 1000000
Range of elements is 0.031591201 31.654384
Range of elements is 0.17773914 5.6262229
Final range of elements is 3.6531096 3.6531096
Absolute values of scaled rhs range from 1.4901161e-08 1000001
999999
Absolute values of scaled objective range from 10 1461.2438
Absolute values of scaled bounds range from 1 100 1e+100
Primal error 1.1641532e-10 0
Primal error 1.1641532e-10 0
0 4.0001132e+08 1002715 4 410 2 2
Primal error 1.4901161e-08 0
Primal error 0 0
1 2400 0 0 0 0 0
Looking optimal checking bounds with 1e+10
Going back to original objective
Primal error 0 0
Looking optimal checking bounds with 1e+10
Optimal - objective value 2400
66 1
66 2
Presolve 0 -66 0 -27 0 -135
Bounds were tightened 0
Crash put 1 0
Optimal - objective value 2400
After Postsolve, objective 2400 0 0 0 0
Engine Solve time: 0.98 Seconds.
Solver found a solution. All constraints and optimality conditions
are satisfied.
Solve time: 2.34 Seconds.
```

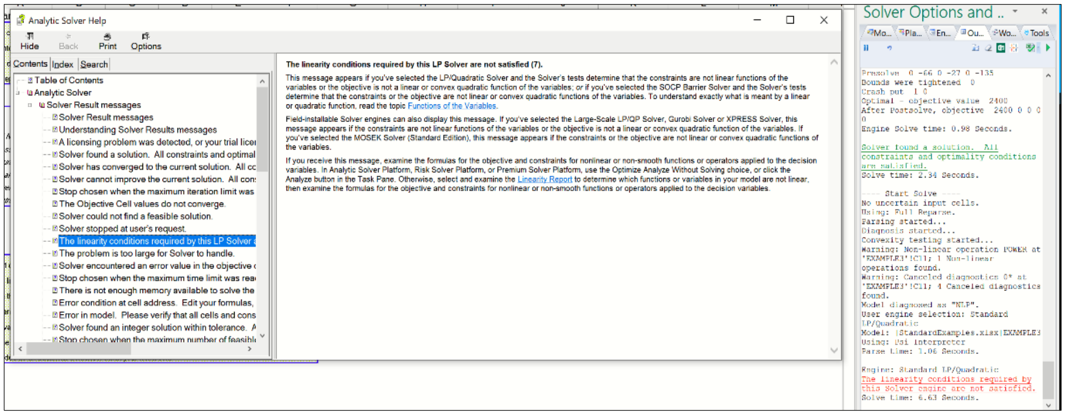
If you are having problems finding the solution you want or expect, this detailed log can sometimes be helpful.

Note: This functionality is not supported in Analytic Solver Cloud.

Click the Solver Result Message for Help

The Solver Result message is always **underlined** – it is a **hyperlink** to Help. If you aren't sure that you fully understand it, **click the link** to open online Help to a detailed discussion of the message.

Below is an example of Help that appears in Analytic Solver Desktop when you solve EXAMPLE3 in **StandardExamples.xlsx**, and click on the Solver Result error message in red, "The linearity conditions required by this Solver engine are not satisfied." Notice the two hyperlinks in the Help text, to Functions of the Variables and the Linearity Report. If you click the latter link, the Help text below appears.


Analytic Solver Help
Hide
Back
Print
Options
Contents
Index
Search
Table of Contents
Analytic Solver
Solver Result messages
Solver Result messages
Understanding Solver Results messages
A licensing problem was detected, or your trial licen
Solver found a solution. All constraints and optimal
Solver has converged to the current solution. All cc
Solver cannot improve the current solution. All con:
Stop chosen when the maximum iteration limit was
The Objective Cell values do not converge.
Solver could not find a feasible solution.
Solver stopped at user's request.
The linearity conditions required by this LP Solver a
The problem is too large for Solver to handle.
Solver encountered an error value in the objective c
Stop chosen when the maximum time limit was rea
There is not enough memory available to solve the
Error condition at cell address. Edit your formulas,
Error in model. Please verify that all cells and cons
Solver found an integer solution within tolerance. A
Stop chosen when the maximum number of feasibl
Solver Options and ..
Mo...
Pla...
En...
Ou...
Wo...
Tools

The linearity conditions required by this LP Solver are not satisfied (7).

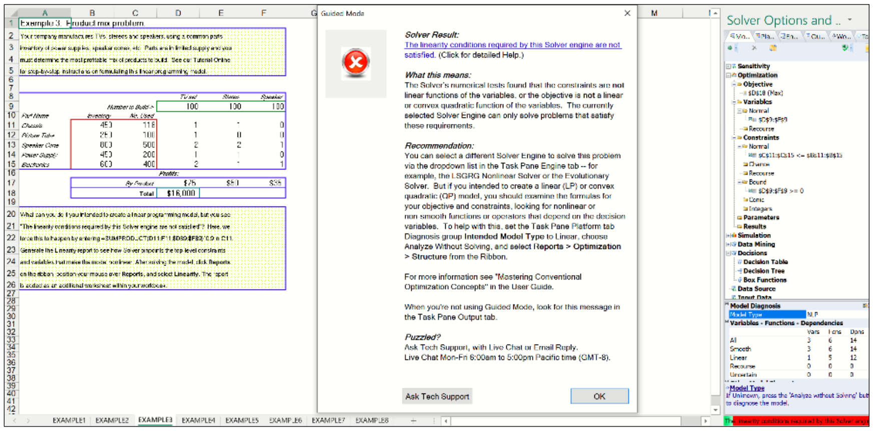
This message appears if you've selected the LP/Quadratic Solver and the Solver's tests determine that the constraints are not linear functions of the variables or the objective is not a linear or convex quadratic function of the variables; or if you've selected the SOCP Barrier Solver and the Solver's tests determine that the constraints or the objective are not linear or convex quadratic functions of the variables. To understand exactly what is meant by a linear or quadratic function, read the topic Functions of the Variables.

Field-installable Solver engines can also display this message. If you've selected the Large-Scale LP/QP Solver, Gurobi Solver or XPRESS Solver, this message appears if the constraints are not linear functions of the variables or the objective is not a linear or convex quadratic function of the variables. If you've selected the MOSEK Solver (Standard Edition), this message appears if the constraints or the objective are not linear or convex quadratic functions of the variables.

If you receive this message, examine the formulas for the objective and constraints for nonlinear or non-smooth functions or operators applied to the decision variables. In Analytic Solver Platform, Risk Solver Platform, or Premium Solver Platform, use the Optimize Analyze Without Solving choice, or click the Analyze button in the Task Pane. Otherwise, select and examine the Linearity Report to determine which functions or variables in your model are not linear, then examine the formulas for the objective and constraints for nonlinear or non-smooth functions or operators applied to the decision variables.

```
Presolve  0 -66 0 -27 0 -135
Bounds were tightened  0
Crash put  1 0
Optimal - objective value  2400
After Postsolve, objective  2400 0 0 0
0
Engine Solve time: 0.98 Seconds.

Solver found a solution.  All
constraints and optimality conditions
are satisfied.
Solve time: 2.34 Seconds.

---- Start Solve ----
No uncertain input cells.
Using: Full Reparse.
Parsing started...
Diagnosis started...
Convexity testing started...
Warning: Non-linear operation POWER at
'EXAMPLE3'!C11; 1 Non-linear
operations found.
Warning: Canceled diagnostics 0* at
'EXAMPLE3'!C11; 4 Canceled diagnostics
found.
Model diagnosed as "NLP".
User engine selection: Standard
LP/Quadratic
Model: |StandardExamples.xlsx|EXAMPLE3
Using: Psi Interpreter
Parse time: 1.06 Seconds.

Engine: Standard LP/Quadratic
The linearity conditions required by
this Solver engine are not satisfied.
Solve time: 6.63 Seconds.
```

Below is an example of Help that appears in Analytic Solver Cloud when you solve the same example and then click the red hyperlink, "The linearity conditions required by this Solver engine are not satisfied."



Solver Result:
The linearity conditions required by this Solver engine are not satisfied. (Click for detailed Help.)

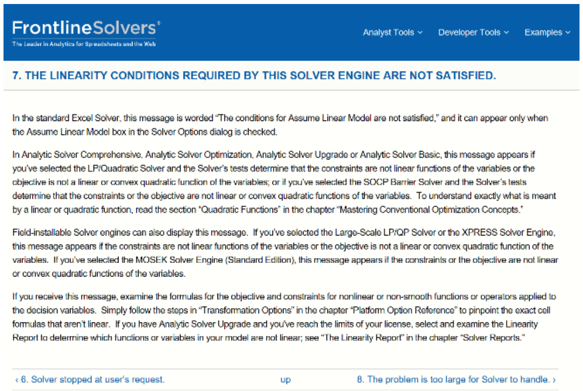
What this means:
The Solver's numerical tests found that the constraints are not linear functions of the variables, or the objective is not a linear or convex quadratic function of the variables. The currently selected Solver Engine can only solve problems that satisfy these requirements.

Recommendation:
You can select a different Solver Engine to solve this problem via the dropdown list in the Task Pane Engine tab -- for example, the LSGRG Nonlinear Solver or the Evolutionary Solver. But if you intended to create a linear (LP) or convex quadratic (QP) model, you should examine the formulas for your objective and constraints, looking for nonlinear or non-smooth functions or operators that depend on the decision variables. To help with this, set the Task Pane Platform tab Diagnosis group **Intended Model Type** to Linear, choose Analyze Without Solving, and select **Reports > Optimization > Structure** from the Ribbon.

For more information see "Mastering Conventional Optimization Concepts" in the User Guide.

When you're not using Guided Mode, look for this message in the Task Pane Output tab.

Puzzled?
Ask Tech Support, with Live Chat or Email Reply.
Live Chat Mon-Fri 6:00am to 5:00pm Pacific time (GMT-8).



7. THE LINEARITY CONDITIONS REQUIRED BY THIS SOLVER ENGINE ARE NOT SATISFIED.

In the standard Excel Solver, this message is worded "The conditions for Assume Linear Model are not satisfied," and it can appear only when the Assume Linear Model box in the Solver Options dialog is checked.

In Analytic Solver Comprehensive, Analytic Solver Optimization, Analytic Solver Upgrade or Analytic Solver Basic, this message appears if you've selected the LP/Quadratic Solver and the Solver's tests determine that the constraints are not linear functions of the variables or the objective is not a linear or convex quadratic function of the variables; or if you've selected the SOCP Barrier Solver and the Solver's tests determine that the constraints or the objective are not linear or convex quadratic functions of the variables. To understand exactly what is meant by a linear or quadratic function, read the section "Quadratic Functions" in the chapter "Mastering Conventional Optimization Concepts."

Field-installable Solver engines can also display this message. If you've selected the Large-Scale LP/QP Solver or the XPRESS Solver Engine, this message appears if the constraints are not linear functions of the variables or the objective is not a linear or convex quadratic function of the variables. If you've selected the MOSEK Solver Engine (Standard Edition), this message appears if the constraints or the objective are not linear or convex quadratic functions of the variables.

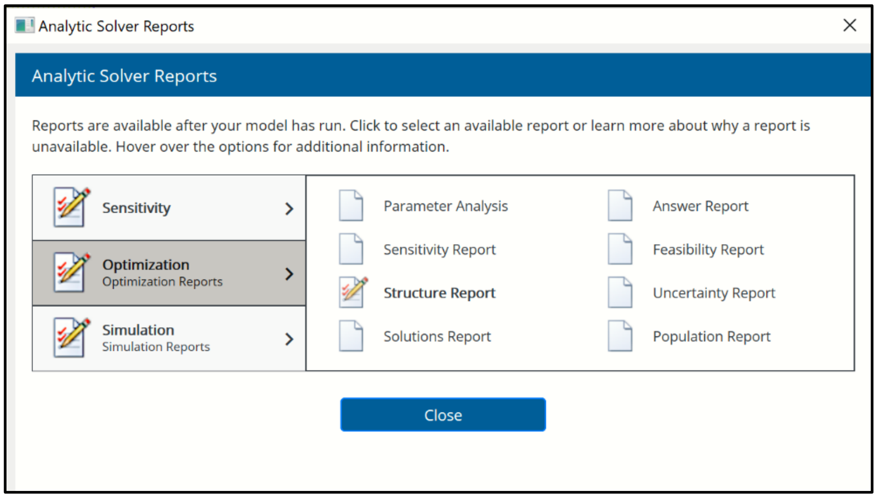
If you receive this message, examine the formulas for the objective and constraints for nonlinear or non-smooth functions or operators applied to the decision variables. Simply follow the steps in "Transformation Options" in the chapter "Platform Option Reference" to pinpoint the exact cell formulas that aren't linear. If you have Analytic Solver Upgrade and you've reach the limits of your license, select and examine the Linearity Report to determine which functions or variables in your model are not linear; see "The Linearity Report" in the chapter "Solver Reports."

‹ 6. Solver stopped at user's request. up 8. The problem is too large for Solver to handle. ›

Given our experience that 90% of technical support cases could be easily resolved by **reading online Help** or the User Guide, you can save yourself, as well as Frontline Systems, time if you *do* read online Help first.

Choose Available Optimization Reports

The Help above suggests that we select the Linearity Report when we receive this Solver Result error message. As a general rule, it's good idea to **examine the available reports** and produce the reports that may help you understand what's wrong with your model. Just select **Reports – Optimization Reports** on the Ribbon – the reports in the gallery are updated each time you solve.



Reports are available after your model has run. Click to select an available report or learn more about why a report is unavailable. Hover over the options for additional information.

- The **Linearity Report** and the **Structure Report** can help when you encounter "The linearity conditions required by this Solver engine are not satisfied."
- The **Feasibility Report** and **Feasibility-Bound**s Report can help when you encounter "Solver could not find a feasible solution."
- The **Scaling Report** can help when you encounter either of these messages, or other unexpected messages or solution values. See the section below "Problems with Poorly Scaled Models."

See the section "An 'Accidentally' Nonlinear Model" in the chapter "Examples: Conventional Optimization" to see how the **Linearity Report**, and the **Structure Report** produced by the PSI Interpreter can help us pinpoint the nonlinear formulas in this model.

When Solving Takes a Long Time

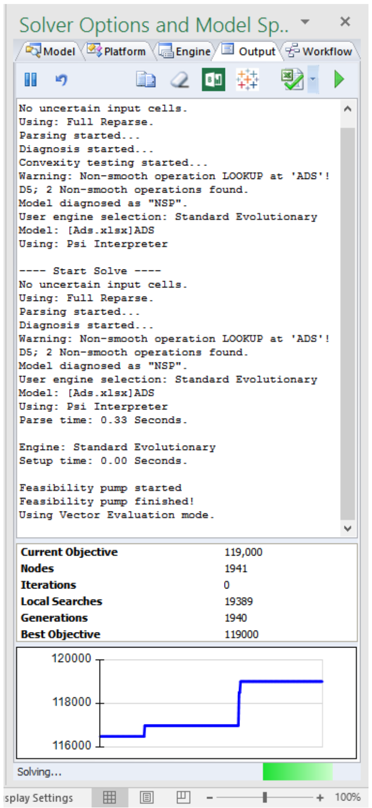
When your model takes a long time to solve, the Task Pane Output tab can be helpful *during* the solution process – at a minimum, to reassure you that the Solver is still making progress, and has not "hung up." If it is not already visible when you first start solving in Analytic Solver Desktop, the Output tab will **appear automatically** after a few seconds of solution time, as long as the Task Pane itself is visible.

The Output tab shows the objective of the best solution found so far, and for problems with integer constraints, the Best Integer Objective ("incumbent"), the Best Possible Objective ("best bound"), and the Integer Gap or percentage difference between these two objectives.

In Analytic Solver Cloud, the Task Pane Output tab will display a moving task pane to indicate the status of the model during the solution process.

Running Chart of the Objective

The bottom part of the Output tab shows a running chart of the objective of the best solution found so far. On the next page is an example of the Output tab on a problem called **Ads.xlsx** that takes about 1 minutes to solve, with the Evolutionary Solver, on a modern 8 - core PC:

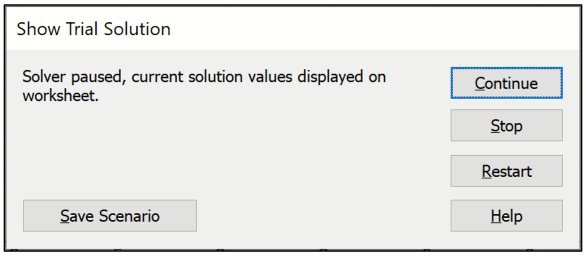


```
No uncertain input cells.
Using: Full Reparse.
Parsing started...
Diagnosis started...
Convexity testing started...
Warning: Non-smooth operation LOOKUP at 'ADS'!
D5; 2 Non-smooth operations found.
Model diagnosed as "NSP".
User engine selection: Standard Evolutionary
Model: [Ads.xlsx]ADS
Using: Psi Interpreter

---- Start Solve ----
No uncertain input cells.
Using: Full Reparse.
Parsing started...
Diagnosis started...
Warning: Non-smooth operation LOOKUP at 'ADS'!
D5; 2 Non-smooth operations found.
Model diagnosed as "NSP".
User engine selection: Standard Evolutionary
Model: [Ads.xlsx]ADS
Using: Psi Interpreter
Parse time: 0.33 Seconds.

Engine: Standard Evolutionary
Setup time: 0.00 Seconds.

Feasibility pump started
Feasibility pump finished!
Using Vector Evaluation mode.
```

Current Objective	119,000
Nodes	1941
Iterations	0
Local Searches	19389
Generations	1940
Best Objective	119000

Interrupting the Solution Process

You can interrupt the solution process at any time, by **pressing the ESC key** in Analytic Solver Desktop, or by **clicking the ⏸ button** in either Desktop or Cloud, in the Task Pane Output tab. If your model is very large, you might have to hold down the ESC key for a second or two. Since the input focus may not be on the Task Pane when you click with the mouse, you might have to click the ⏸ button twice. A dialog like the one below will appear:



Solver paused, current solution values displayed on worksheet.

Click **Continue** if you want to continue solving; click **Stop** to cause the Solver to stop the solution process (this may take a few seconds) and display the message “Solver stopped at user’s request.” Clicking **Restart** may be useful with the Evolutionary Solver – see the Frontline Solvers Reference Guide.

Reasons Why Solving Takes a Long Time

There are several reasons why solving might take a long time:

1. Most often, you’ve used Excel formulas or functions that make your model non-smooth or non-convex, and much more difficult to solve.

You might gain a lot from consulting assistance, or by reading the chapter “Mastering Conventional Optimization Concepts.”

2. If **model diagnosis** or **problem setup** is slow (see timing messages in the Task Pane Output tab), you might be using array formulas, LOOKUP functions with large ranges as arguments, or long “chains” of formulas where each formula depends on an earlier formula.
3. You might be using a Solver Engine, or Engine option settings, that aren’t appropriate for your problem. You can **check the box** “Automatically Select Engine” on the Task Pane Engine tab. **Click Engine options** to display Help on each option. You can gain further insight by reading “Optimization Problems and Solution Methods” in the chapter “Mastering Conventional Optimization Concepts.”
4. Your model might be well formulated, but very large, or it might require non-smooth functions, or many integer variables. A more powerful plug-in Solver Engine may help considerably, and a faster processor, multi-core processor, or more memory can often help.

Adjusting Automatic Mode Options to Gain Speed

Be sure to read “Automatic Mode and Solution Time” in the chapter “Analytic Solver Overview.” If you’re using the **Automatic** setting (the ***default*** for new models) for any of the seven options described in this section, you can save time by pre-setting these options to the same values chosen automatically.

Model Design for Maximum Optimization Speed

Paying attention to your Excel worksheet layout and design, and avoiding certain practices, will help you get the most from the PSI Interpreter. Below is a brief list of Do’s and Don’ts that will help you realize the best possible speed:

- **Do** build worksheets starting from the “upper left corner.” **Don’t** place cells or formulas at extreme row and column addresses all over the worksheet.
- **Do** build the model on a small number of worksheets. **Don’t** include references on these worksheets to other worksheets that aren’t required for the simulation model.
- **Do** use numeric and logical formulas and functions. **Don’t** create string results in the middle of numeric calculations (see example below).
- **Do** use built-in Excel functions (including Analysis ToolPak functions) freely. **Don’t** use third-party or user-written functions unless truly needed. Such functions must not have “side-effects” other than the value returned.
- **Do** use operators like + **-** * / ^ and functions like SUM, AVERAGE, MAX, MIN, AND, OR, NOT, IF, CHOOSE. **Don’t** use INDIRECT, OFFSET, or TEXT, SEARCH, REPLACE, FIXED, DOLLAR, or ROMAN.
- If possible, **Don’t** use array formulas, LOOKUP functions with large ranges as arguments, or long “chains” of formulas where each formula depends on an earlier formula (and hence on *all* earlier formulas). Each of these things can slow down analysis of your model considerably (by 10x in some cases).

A common practice that slows down the PSI Interpreter, and is easy to avoid, is illustrated by the following:

A1: =IF(B1>100,"Yes","No") and later A10: =IF(A1="Yes",A2,50)

You don't need "Yes" and "No" when Excel provides built-in values TRUE and FALSE. You could write =IF(B1>100,TRUE,FALSE), but this is simpler:

A1: =B1>100 and later A10: =IF(A1,A2,50)

The formula =B1>100 evaluates to either TRUE or FALSE. A1 holds this value and you can test it later in another formula.

When the Solution Seems Wrong

When the solution on the worksheet seems wrong, before doing anything else, you should **read the solution log** in the Output tab, **click the error message** to display Help, and **run available reports** as described above. We emphasize this, because very often when we are contacted in technical support, the user has not taken these basic steps.

Although software bugs are always possible, consider carefully the possibility that the solution found by the Solver is **correct for the model you've defined**, and that your expectation is wrong. This may mean that what your model actually says is different from what you intended. In the majority of cases we see in technical support, the user has an error in a formula or in the expression of a constraint that leads to the unexpected solution.

Problems with Poorly Scaled Models

Many unexpected Solver Result messages are due to a poorly scaled model. A *poorly scaled* model is one that computes values of the objective, constraints, or intermediate results that differ by several orders of magnitude. A classic example is a financial model that computes a dollar amount in millions or billions and a return or risk measure in fractions of a percent. Because of the finite precision of computer arithmetic, when these values of very different magnitudes (or others derived from them) are added, subtracted, or compared – in the user's model or in the Solver's own calculations – the result will be accurate to only a few significant digits. After many such steps, the Solver may detect or suffer from "numerical instability."

The effects of poor scaling in a large, complex optimization model can be among the most difficult problems to identify and resolve. It can cause Solver engines to return messages such as "Solver could not find a feasible solution," "Solver could not improve the current solution," or even "The linearity conditions required by this Solver engine are not satisfied," with results that are suboptimal or otherwise very different from your expectations. The effects may not be apparent to you, given the initial values of the variables, but when the Solver explores Trial Solutions with very large or small values for the variables, the effects will be greatly magnified.

Dealing with Poor Scaling

Most Solver engines include a **Use Automatic Scaling** option on the Task Pane Engine tab. When this option is set to True, the Solver rescales the values of the objective and constraint functions internally in order to minimize the effects of poor scaling. But this can only help with the Solver's own calculations – it can't help with poorly scaled results that arise *in the middle of your Excel formulas*.

The best way to avoid scaling problems is to carefully choose the "units" implicitly used in your model so that all computed results are within a few orders of magnitude of each other. For example, if you express dollar amounts in units of (say) millions, the actual numbers computed on your worksheet may range from perhaps 1 to 1,000.

If you're experiencing results that may be due to poor scaling, you can check your model for scaling problems that arise *in the middle of your Excel formulas* by selecting the Scaling Report after solving your model. If you're using Analytic Solver Upgrade, you'll have to go through each of your formulas and play "what-if" manually to identify such problems.

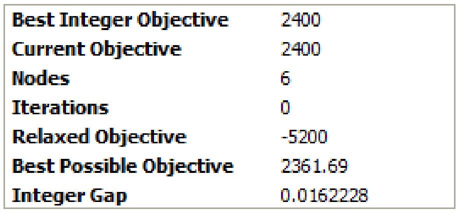
The Integer Tolerance Option

Users who solve problems with integer constraints using the standard Excel Solver occasionally report that "Solver claims it found an optimal solution, but I manually found an even better solution." What happens in such cases is that the Solver stops with the message "Solver found a solution" because it found a solution *within the range* of the true integer optimal solution *allowed by the Tolerance* option in the standard Solver's Options dialog. In similar cases, Analytic Solver displays a message "Solver found an integer solution within tolerance," to avoid confusion.

When you solve a problem with integer constraints, the solution process in almost all Solver engines is governed by the **Integer Tolerance** option on the Task Pane Engine tab. When this option value is non-zero – say 0.05, as in the standard Excel Solver – the Solver engine stops when it has found a solution satisfying the integer constraints whose objective is within 5% of the true integer optimal solution. Therefore, you may know of or be able to discover an integer solution that is better than the one found by the Solver.

To avoid this common problem, the *default* **Integer Tolerance** value in Analytic Solver Platform V9.0 and beyond is **0 or 0.001** (depending on the engine used) rather than 0.05. But this has an important consequence for solution time: The solution process for integer problems often finds a near-optimal solution (sometimes *the* optimal solution) relatively quickly, and then spends far more time exhaustively checking other possibilities to find (or verify that it has found) the very best integer solution.

To avoid this extra time, you can either set the Integer Tolerance to a non-zero value – say 0.025 or 0.05 – or you can watch the Task Pane Output tab during the solution process, which displays the current Integer Gap – the value against which the Integer Tolerance is compared. Here's an example from the solution of the EXAMPLE5 model shown earlier, where the Integer Gap was 1.6% just before the Solver "proved optimality" and reported the solution:



Best Integer Objective	2400
Current Objective	2400
Nodes	6
Iterations	0
Relaxed Objective	-5200
Best Possible Objective	2361.69
Integer Gap	0.0162228

You can make a real-time decision that the current solution is "good enough," and press ESC or click the Pause/Stop button, then click the Stop button in the Show Trial Solution dialog, to stop the Solver with the current solution.

When Things Go Right: Getting Further Results

When you *do* receive a solution that makes sense to you, and you have a Solver Result message in **green** at the bottom of the Task Pane, there are several ways you can get further results. We'll cover three possibilities:

1. **Obtaining dual values** from a linear or smooth nonlinear optimization problem.
2. **Obtaining multiple solutions** from an integer programming problem, or a global optimization problem.
3. Performing **multiple parameterized optimizations**, and capturing all the solutions in reports and charts.

Dual Values

When you formulate and solve a linear programming problem, or a smooth nonlinear optimization problem, the solution process also yields numbers, called *dual values*, for the decision variables and constraints that are "pressed to the limit" at the optimal solution. A dual value can tell you, for example, how much you could pay to acquire more units of a scarce resource that is fully utilized in the solution; it is sometimes called a *shadow price* or *marginal value*.

- The **dual value for a decision variable** is nonzero only when the variable's value is equal to its upper or lower bound at the optimal solution. This is called a *nonbasic* variable, and its value was driven to the bound during the optimization process. Moving the variable's value away from the bound will *worsen* the objective function's value; conversely, "loosening" the bound will *improve* the objective. The dual value measures the change in the objective function's value *per unit change* in the variable's value.
- The **dual value for a constraint** is nonzero only when the constraint is equal to its bound. This is called a *binding* constraint, and its value was driven to the bound during the optimization process. Moving the constraint left hand side's value away from the bound will *worsen* the objective function's value; conversely, "loosening" the bound will *improve* the objective. The dual value measures the change in the objective function's value *per unit change* in the constraint's bound.

In nonlinear optimization problems, the dual values are valid only at the *single point* of the optimal solution – if there is any curvature involved, the dual values begin to change as soon as you move away from the optimal solution. In linear programming problems, the dual values remain constant over a *range* of increases and decreases in the variables' objective coefficients and the constraints' right hand sides, respectively.

To obtain dual values in report form, simply select **Reports – Optimization – Sensitivity** from the Ribbon. An example Sensitivity Report appears below.

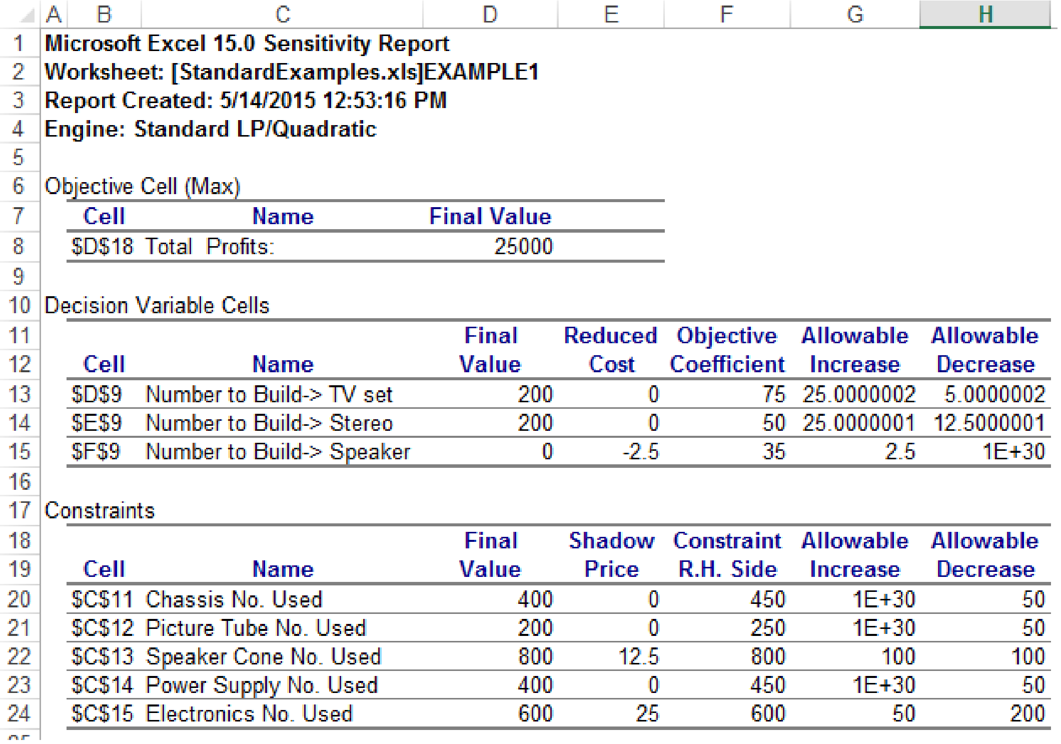
In Analytic Solver Desktop, you can also obtain dual values via the Analytic Solver Object-Oriented API, with simple references such as myProb.VarDecision.DualValue(i) or myProb.FcnConstraint.DualValue(i) in your VBA code. This is described in greater depth in the chapter "Automating Optimization in VBA."

On the next page is a Sensitivity Report for EXAMPLE1, the Product Mix model which is the first example in the chapter "Examples: Conventional Optimization." (Note: Number formatting has been expanded in cells E15 and

E22 to show decimal values.) At the optimal solution, we use all 800 Speaker Cones and 600 Electronics units, but not all of the other components. We don't produce any Speakers – cell F9 is driven to its lower bound of 0. What do the dual values tell us about the two binding constraints, and the one "nonbasic" decision variable?

The dual value of 12.5 for Speaker Cones tells us that we could increase Total Profits by $12.50 for every additional Speaker Cone we can acquire, up to 100 more. Similarly, the dual value of 25 for Electronics units tells us we could use up to 50 more units and increase Total Profits by $25.00 for each extra unit.

The dual value of -2.5 for F9 tells us that, if we were forced to produce some Speakers, we would reduce Total Profits by $2.50 for each Speaker we made (because we'd give up production of another product that is more profitable).



Microsoft Excel 15.0 Sensitivity Report
Worksheet: [StandardExamples.xls]EXAMPLE1
Report Created: 5/14/2015 12:53:16 PM
Engine: Standard LP/Quadratic

Objective Cell (Max)

Cell	Name	Final Value
$D$18	Total Profits:	25000

Decision Variable Cells

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
$D$9	Number to Build-> TV set	200	0	75	25.0000002	5.0000002
$E$9	Number to Build-> Stereo	200	0	50	25.0000001	12.5000001
$F$9	Number to Build-> Speaker	0	-2.5	35	2.5	1E+30

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
$C$11	Chassis No. Used	400	0	450	1E+30	50
$C$12	Picture Tube No. Used	200	0	250	1E+30	50
$C$13	Speaker Cone No. Used	800	12.5	800	100	100
$C$14	Power Supply No. Used	400	0	450	1E+30	50
$C$15	Electronics No. Used	600	25	600	50	200

Multiple Solutions

When you solve an integer programming problem or a global optimization problem, the final solution you see on the Excel worksheet is typically the *best* of *several* candidate solutions that were found during the solution process. You can obtain and examine these other solutions; this can be useful, especially if you have other criteria, not captured in the formulation of the optimization model, for preferring one solution over another. For each candidate solution, you can examine the final values of the decision variables and the objective.

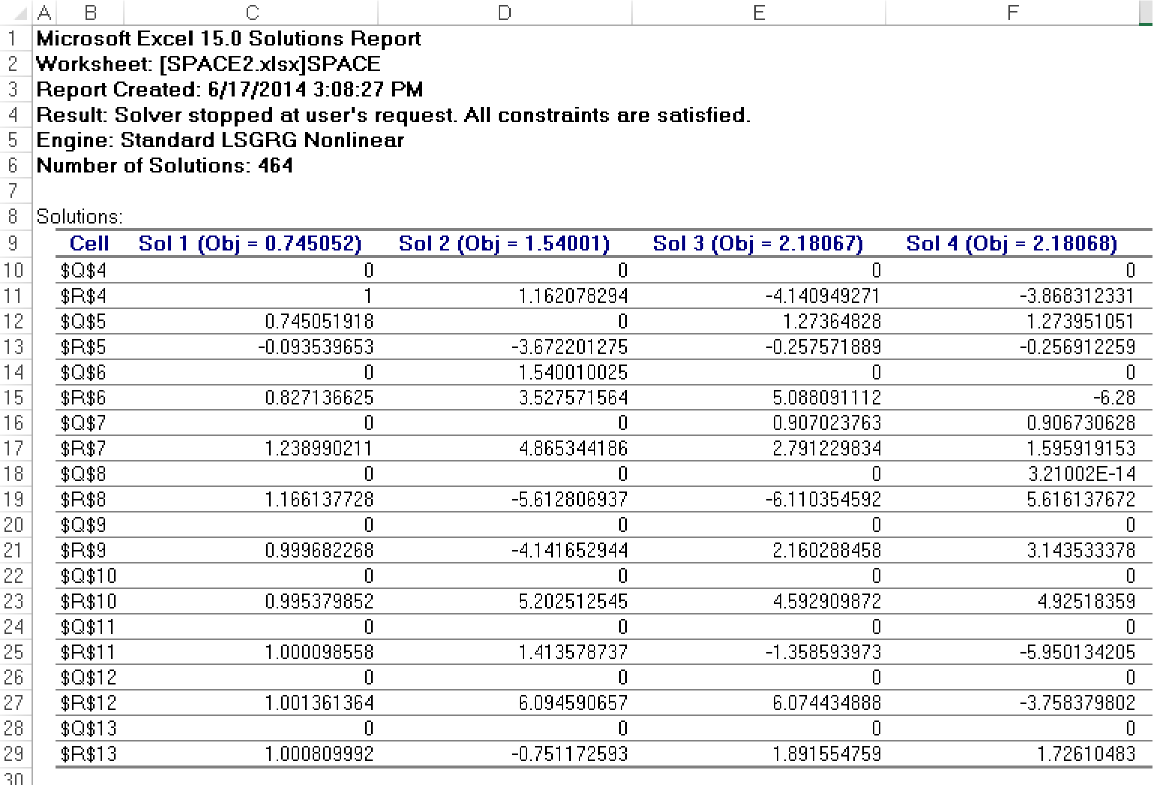
To obtain multiple solutions in report form, select **Reports – Optimization – Solutions** from the Ribbon.

- For **integer programming** problems, the report shows each 'incumbent' or feasible integer solution found by the Branch & Bound method during the solution process.

- For **global optimization** problems, the report shows each locally optimal solution found by the Multistart method.
- For **non-smooth optimization** problems solved with the Evolutionary Solver, the report shows key members of the final population of solutions.

For all types of problems, if only one candidate solution was found, the **Solutions choice will *not* appear** in the Reports – Optimization gallery. For example, advanced Solvers for integer programming problems may find optimal solutions at the root node of the Branch & Bound tree, without considering any other incumbent solutions. For integer programming problems solved with the LP/Quadratic or Large-Scale LP/QP Solver, the Engine tab Integer group PreProcessing option must be set to None in order to create a Solutions Report.

On the next page is an example of a Solutions Report created when the GRG Nonlinear Solver with Multistart was used to solve a model called SPACE2, a global optimization problem with many locally optimal solutions. (Given more time, the Solver would find an even better solution than the ones shown.)



Microsoft Excel 15.0 Solutions Report
Worksheet: [SPACE2.xlsx]SPACE
Report Created: 6/17/2014 3:08:27 PM
Result: Solver stopped at user's request. All constraints are satisfied.
Engine: Standard LSGRG Nonlinear
Number of Solutions: 464

Solutions:

Cell	Sol 1 (Obj = 0.745052)	Sol 2 (Obj = 1.54001)	Sol 3 (Obj = 2.18067)	Sol 4 (Obj = 2.18068)
$Q$4	0	0	0	0
$R$4	1	1.162078294	-4.140949271	-3.868312331
$Q$5	0.745051918	0	1.27364828	1.273951051
$R$5	-0.093539653	-3.672201275	-0.257571889	-0.256912259
$Q$6	0	1.540010025	0	0
$R$6	0.827136625	3.527571564	5.088091112	-6.28
$Q$7	0	0	0.907023763	0.906730628
$R$7	1.238990211	4.865344186	2.791229834	1.595919153
$Q$8	0	0	0	3.21002E-14
$R$8	1.166137728	-5.612806937	-6.110354592	5.616137672
$Q$9	0	0	0	0
$R$9	0.999682268	-4.141652944	2.160288458	3.143533378
$Q$10	0	0	0	0
$R$10	0.995379852	5.202512545	4.592909872	4.92518359
$Q$11	0	0	0	0
$R$11	1.000098558	1.413578737	-1.358593973	-5.950134205
$Q$12	0	0	0	0
$R$12	1.001361364	6.094590657	6.074434888	-3.758379802
$Q$13	0	0	0	0
$R$13	1.000809992	-0.751172593	1.891554759	1.72610483

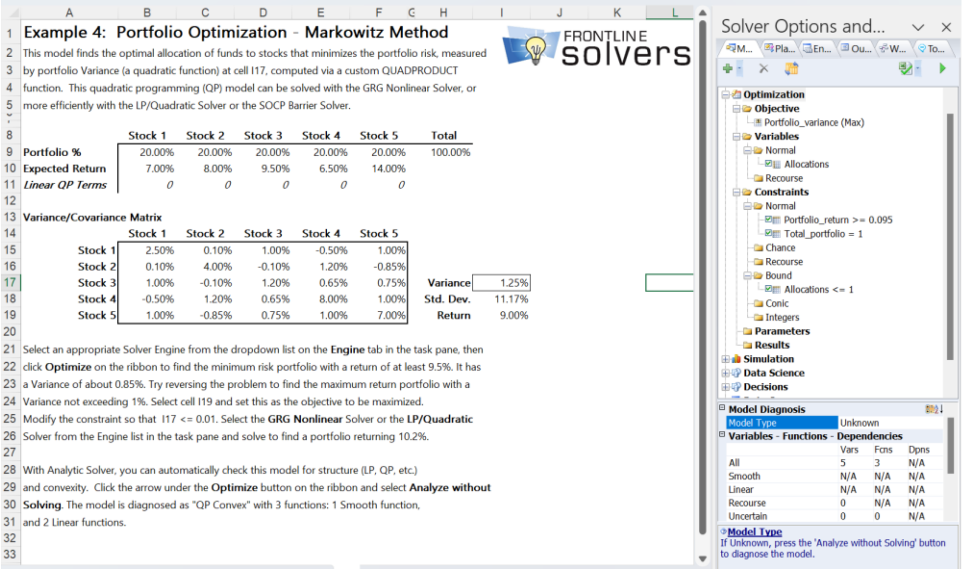
You can also obtain multiple solutions via the Analytic Solver Object-Oriented API, by accessing a property such as myProb.Solver.NumSolutions, setting the property myProb.Solver.SolutionIndex to choose a solution, then accessing the solution in the usual manner.

Multiple Parameterized Optimizations

Once you have a model where you're getting an optimal solution that makes sense for one set of inputs or parameters, it's often useful to vary one or more parameters across a range of values, and find the optimal solution for each individual parameter value. With Analytic Solver, you can easily define optimization parameters, run multiple optimizations and save the solutions to each optimization, and summarize the results in reports and charts.

The earlier chapter “Examples: Parameters and Sensitivity Analysis” gives an overview of the role of parameters for optimization, simulation, and sensitivity analysis, and the Frontline Solvers Reference Guide fully documents the features available for multiple parameterized optimization.

In this chapter, we’ll illustrate what you can do with EXAMPLE4, the Markowitz portfolio optimization model in **StandardExamples.xlsx** described in the chapter “Examples: Conventional Optimization” under “Nonlinear Programming Examples.” The model is pictured on the next page.



Example 4: Portfolio Optimization - Markowitz Method

This model finds the optimal allocation of funds to stocks that minimizes the portfolio risk, measured by portfolio Variance (a quadratic function) at cell I17, computed via a custom QUADPRODUCT function. This quadratic programming (QP) model can be solved with the GRG Nonlinear Solver, or more efficiently with the LP/Quadratic Solver or the SOCP Barrier Solver.

	Stock 1	Stock 2	Stock 3	Stock 4	Stock 5	Total
Portfolio %	20.00%	20.00%	20.00%	20.00%	20.00%	100.00%
Expected Return	7.00%	8.00%	9.50%	6.50%	14.00%	
Linear QP Terms	*0*	*0*	*0*	*0*	*0*	

Variance/Covariance Matrix

	Stock 1	Stock 2	Stock 3	Stock 4	Stock 5		
Stock 1	2.50%	0.10%	1.00%	-0.50%	1.00%		
Stock 2	0.10%	4.00%	-0.10%	1.20%	-0.85%		
Stock 3	1.00%	-0.10%	1.20%	0.65%	0.75%	**Variance**	1.25%
Stock 4	-0.50%	1.20%	0.65%	8.00%	1.00%	**Std. Dev.**	11.17%
Stock 5	1.00%	-0.85%	0.75%	1.00%	7.00%	**Return**	9.00%

Select an appropriate Solver Engine from the dropdown list on the **Engine** tab in the task pane, then click **Optimize** on the ribbon to find the minimum risk portfolio with a return of at least 9.5%. It has a Variance of about 0.85%. Try reversing the problem to find the maximum return portfolio with a Variance not exceeding 1%. Select cell I19 and set this as the objective to be maximized. Modify the constraint so that I17 <= 0.01. Select the **GRG Nonlinear** Solver or the **LP/Quadratic** Solver from the Engine list in the task pane and solve to find a portfolio returning 10.2%.

With Analytic Solver, you can automatically check this model for structure (LP, QP, etc.) and convexity. Click the arrow under the **Optimize** button on the ribbon and select **Analyze without Solving**. The model is diagnosed as "QP Convex" with 3 functions: 1 Smooth function, and 2 Linear functions.

Model Diagnosis

Model Type	Unknown

Variables - Functions - Dependencies

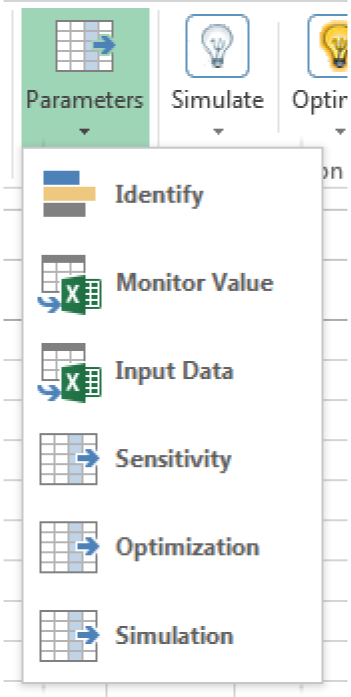
	Vars	Fcns	Dpns
All	5	3	N/A
Smooth	N/A	N/A	N/A
Linear	N/A	N/A	N/A
Recourse	0	N/A	N/A
Uncertain	0	0	N/A

Model Type
If Unknown, press the 'Analyze without Solving' button to diagnose the model.

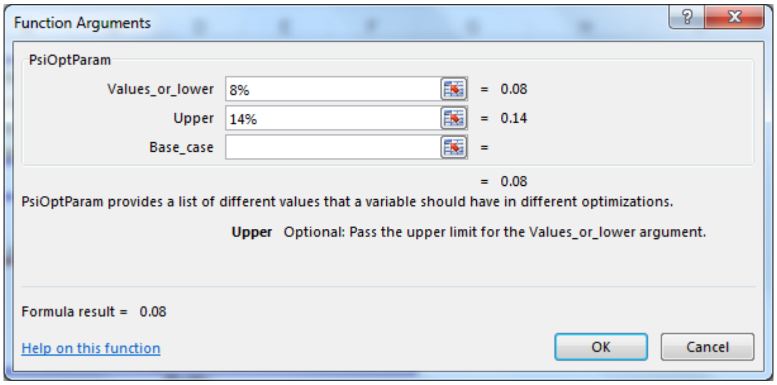
When we perform a single optimization with this model, we get an optimal allocation of funds to stocks that minimizes portfolio variance (a risk measure) while earning a portfolio rate of return of at least 9.5%, as shown in the Task Pane Model tab. This solution is a single point on the “efficient frontier;” we can find other points on this frontier by solving the problem for different levels of the required portfolio return.0

This is easy to do in Analytic Solver: We’ll define a single optimization parameter, use it in the portfolio return constraint right hand side, and then run multiple (say 10) optimizations where the parameter is automatically varied from (say) 8% to 14%. Let’s do this step by step.

To define an optimization parameter, select an empty cell – say **I20**, and choose **Parameters – Optimization** from the Ribbon, as shown below.



Analytic Solver displays a dialog where you can enter a lower and upper limit for the parameter value, or alternatively a list of values or a cell range:


Function Arguments

PsiOptParam

Values_or_lower 8% = 0.08

Upper 14% = 0.14

Base_case =

= 0.08

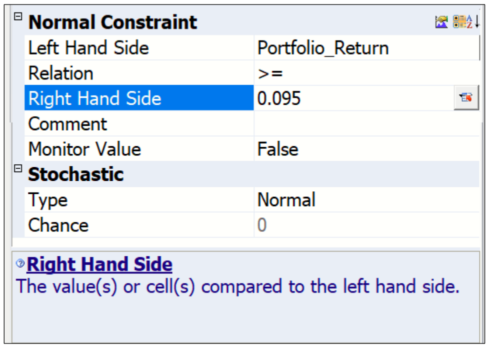
PsiOptParam provides a list of different values that a variable should have in different optimizations.

Upper Optional: Pass the upper limit for the Values_or_lower argument.

Formula result = 0.08

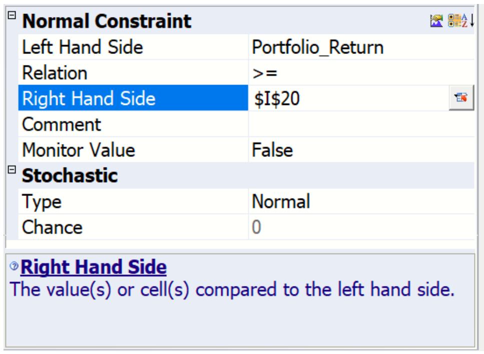
When you click OK, the formula **=PsiOptParam(0.08,0.14)** appears in cell I20.

Next, we’ll use this parameter in the portfolio return constraint. To do this, we select the constraint ‘Portfolio Return >= 0.095’ in the Task Pane Model tab, and edit its properties in the lower part of the Task Pane.


Normal Constraint	
Left Hand Side	Portfolio_Return
Relation	>=
Right Hand Side	0.095
Comment	
Monitor Value	False
Stochastic	
Type	Normal
Chance	0

Right Hand Side
The value(s) or cell(s) compared to the left hand side.

We click the cell selector icon to the right of the field containing 0.095, and then point and click to select cell I20. The properties of the constraint are updated to use cell I20 for the right hand side, as shown on the next page.

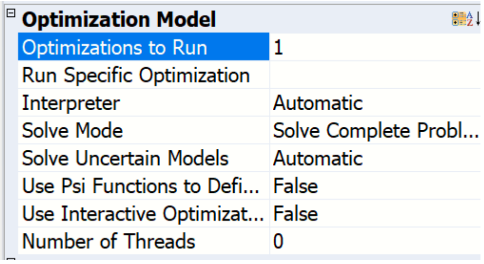


Normal Constraint	
Left Hand Side	Portfolio_Return
Relation	>=
Right Hand Side	$I$20
Comment	
Monitor Value	False
Stochastic	
Type	Normal
Chance	0

Right Hand Side
The value(s) or cell(s) compared to the left hand side.

Note: Editing optimization parameters in the task pane is not supported in Analytic Solver Cloud. To change a constraint in the Cloud, simply double click the I19 >= 0.095 constraint in the task pane to open the Add Constraint dialog, then change 0.095 to I20.

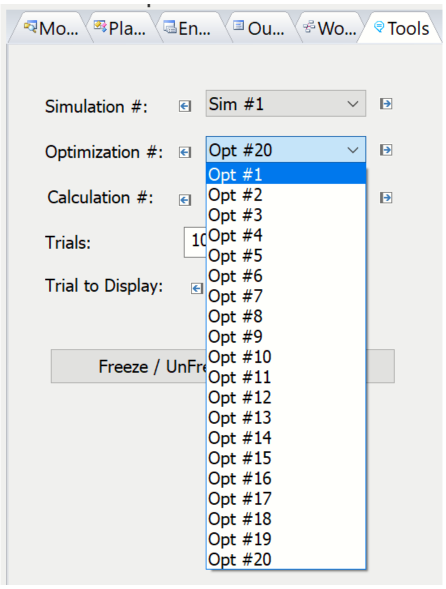
Just one more step is required: We must specify how many optimizations we want to perform. On the Task Pane Platform tab, we set the very first option **Optimizations to Run** to 10, as shown on the next page.



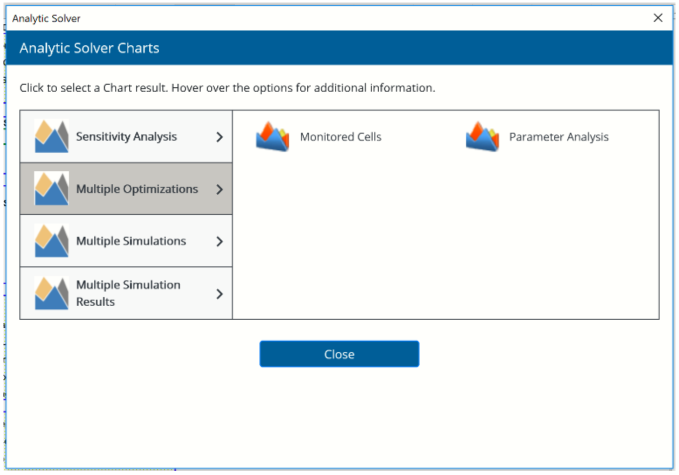
Optimization Model	
Optimizations to Run	1
Run Specific Optimization	
Interpreter	Automatic
Solve Mode	Solve Complete Probl...
Solve Uncertain Models	Automatic
Use Psi Functions to Defi...	False
Use Interactive Optimizat...	False
Number of Threads	0

Now we simply click the **Optimize** button on the Ribbon, or the green right arrow on the Task Pane, to run all 10 optimizations. By default, Analytic Solver saves the optimal objective and decision variable values from all 10 optimizations; you can save other results, such as constraint values, by setting their Monitor property in the Task Pane to True.

Just after solving, Analytic Solver Optimization displays the results of the last optimization on the worksheet. But you can display the results of any of the other 9 optimizations by selecting **Optimizations # dropdown list** on the Tools tab of the Solver Task Pane:

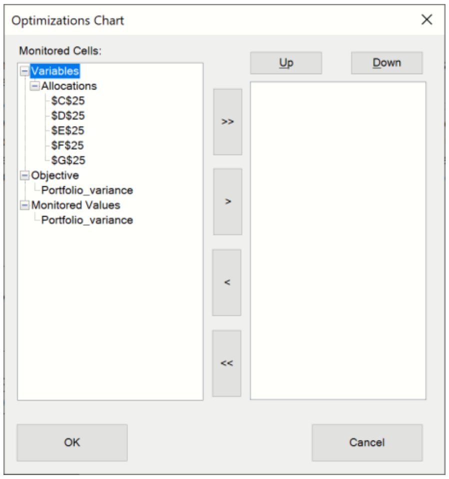


Analytic Solver has built-in facilities to create charts of multiple parameterized optimizations. Just select **Charts – Multiple Optimizations – Monitored Cells** from the Ribbon.

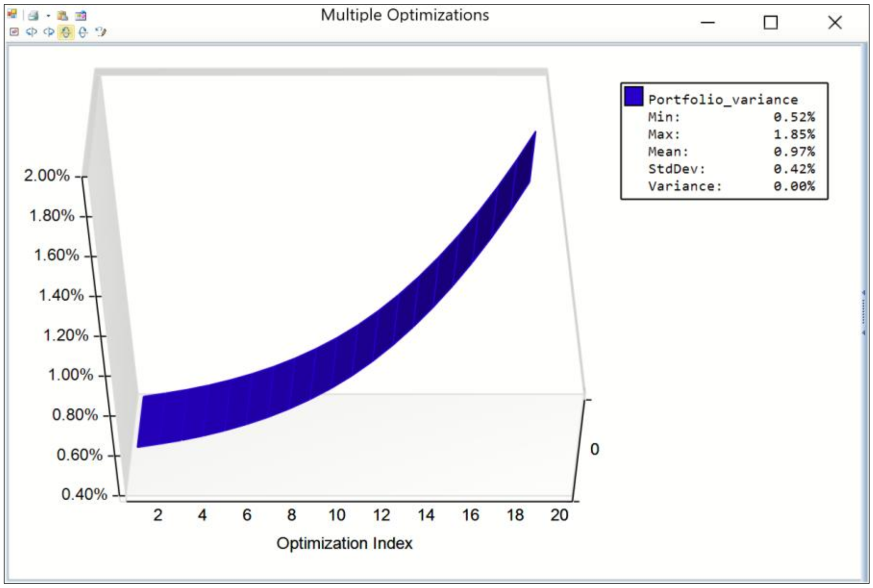


Click to select a Chart result. Hover over the options for additional information.

Select the optimization result you would like to chart: In this example, we want to plot the optimal objective (Portfolio Variance) across the 10 optimizations, where the Portfolio Return threshold is varied on each optimization.



When you click OK, Analytic Solver draws the chart, as shown below.



Getting Results: Simulation

Introduction

This chapter explains how to obtain and interpret results from simulation in Analytic Solver Comprehensive and its subset product Analytic Solver Simulation. We'll discuss **what can go wrong**, and what to do about it, and also **how to get maximum insight** from the results of a simulation.

Simulation is 'simpler' than optimization in the sense that it requires only (i) sampling of input values, (ii) calculation of your Excel model with these values, and (iii) collection of the results; no 'search for a best solution' is involved. A simulation can be performed on almost any model, using any Excel functions.

But the way you build your model does affect the *speed* of simulation, or the time required to get results. And the structure of your model will become even more important in the next chapter, when we consider optimization and models that include uncertainty.

In the following sections, we'll focus on immediate actions you can take when you get an unexpected result – but if you read the chapter "Mastering Simulation and Risk Analysis Concepts," you'll learn more about the Monte Carlo simulation process, and better understand how to design your model to get the best results in the least time.

What Can Go Wrong, and What to Do About It

When you click the **Simulate** button on the Ribbon, or the **green arrow** on the Task Pane to run a simulation, you'll normally get one of these outcomes:

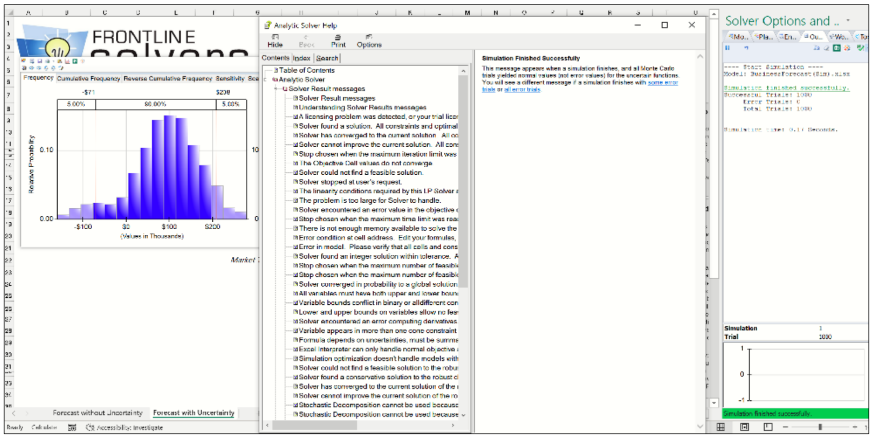
1. Results on the worksheet, and the message "Simulation finished successfully" in **green** at the bottom of the Task Pane. You can proceed to "When Things Go Right: Getting Further Results."
2. An error message that you understand and can correct, in **red** at the bottom of the Task Pane. You can take corrective action.
3. An error message that you *don't* understand, in **red** at the bottom of the Task Pane. You should **read the solution log** in the Output tab, **click the error message** and read Help about the message.
4. Results on the worksheet that you don't understand. Again, before doing anything else, you should **read the solution log** in the Output tab, **click the error message** to display Help, and **read the section** below "When Simulation Results Seem Wrong."
5. Simulation runs for a very long time, and you don't get results or a message in the Task Pane until you press ESC or click Pause/Stop. You should **read the section** "When Simulation Takes a Long Time."

In rare cases, you might find Excel shutting down or "locking up" (so nothing happens when you press and hold the ESC key for several seconds). In this case please contact Frontline Systems Technical Support at (775) 831-0300 x4 or support@solver.com. Some error messages ask you to contact Technical Support. If you can send us your model, this will be very helpful.

But experience shows that 99% of all technical support cases involve “pilot error” by the user, and that 90% of all such cases could be easily resolved by reading online Help or the User Guide. So we hope you’ll keep reading, and that you’ll take these steps before calling technical support!

Review Messages in the Output Tab

If using Analytic Solver Desktop, your first step should be to **review the messages** in the **solution log** in the Task Pane Output tab. Below is an example of the most common simulation error message, and Help that appears when you click the message:



Simulation Finished Successfully

This message appears when a simulation finishes, and all Monte Carlo trials yielded normal values (not error values) for the uncertain functions. You will see a different message if a simulation finishes with some error trials or all error trials.

```
---- Start Simulation ----
Model: BusinessForecast(Sim).xlsx

Simulation finished successfully.
Successful Trials: 1000
     Error Trials: 0
     Total Trials: 1000


Simulation time: 0.17 Seconds.
```

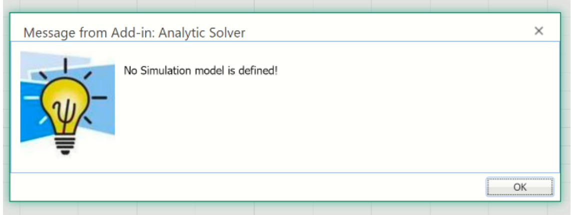
Note: Information related to the solving process in Analytic Solver Cloud is not available during parsing of the model. As a result, feedback is only available on the Output tab once the Solver Engine (for a simulation, Risk Solver Engine) starts to solve the model.

Click the Error Message for Help

Most simulation error messages are **underlined** – they are **hyperlinks** to online Help. If you aren’t sure that you fully understand it, **click the link** to open Help to a detailed discussion of the message.

As the Help text explains in the example above, if your model doesn’t define any simulation outputs, either via a PsiOutput() function that is added to the formula in an output cell or that refers to the output cell, or via a PSI Statistics function that refers to the output cell, there will be *no results* when you run a simulation. As corrective action, you simply need to define the formula cells for which you want results as simulation outputs.

Here is this error as displayed in Analytic Solver Cloud.



Message from Add-in: Analytic Solver

No Simulation model is defined!

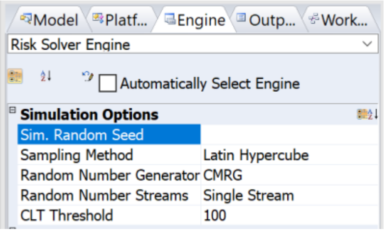
Hyperlinks are not supported in Analytic Solver Cloud.

Role of the Random Number Seed

The **seed**, or initial value, of the **random number generator** used in the simulation process determines whether your results are *exactly reproducible* when you re-run a simulation, or whether your results are *similar but not identical* because a different random sample was drawn.

By **default**, the seed for each simulation is set from the value of the system clock, which is **different** each time you run a simulation. If you set a seed value as explained below, the same seed value is used on each simulation, which means that the entire stream of random numbers drawn, and hence your simulation results, will be *exactly reproducible* on each simulation run.

To set the seed, use the Task Pane **Engine tab**. From the dropdown list at the top of the tab, select **Risk Solver Engine**, the software engine that actually performs a simulation. The options for Risk Solver Engine will be displayed:



☐ Automatically Select Engine

Simulation Options

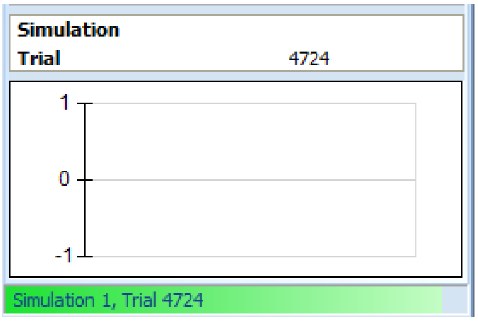
Sim. Random Seed	
Sampling Method	Latin Hypercube
Random Number Generator	CMRG
Random Number Streams	Single Stream
CLT Threshold	100

Enter a positive number for the option **Sim. Random Seed**. A value of 0 here means "use the value of the system clock as the seed on each simulation."

When Simulation Takes a Long Time

When a simulation takes a long time, or if you are running multiple parameterized simulations, the Task Pane Output tab shows you a progress indicator. If it is not already visible when you first start solving, the Output tab will **appear automatically** after a few seconds, as long as the Task Pane itself is visible.

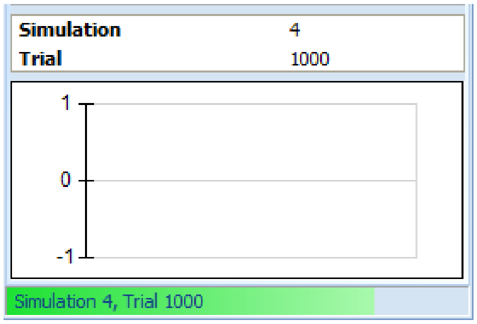
If you are using Analytic Solver Desktop and have the Task Pane Platform tab Simulation group Interpreter option set to **Excel Interpreter**, the progress indicator will be updated during the Monte Carlo trials of a single simulation, as shown in the example below.



Simulation
Trial 4724

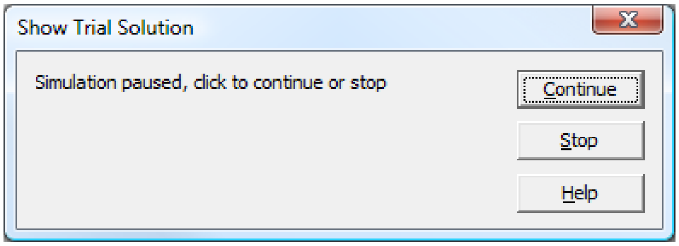
Simulation 1, Trial 4724

If you have the Interpreter option set to **Psi Interpreter**, the progress indicator usually ***won't*** be updated during the trials of a single simulation, because all of these trials are being executed in parallel. If you are running a multiple parameterized simulation, the progress indicator will be updated for each new simulation, as shown in the example below.


Simulation 4
Trial 1000

Interrupting the Simulation Process

You can interrupt the simulation process at any time, by **pressing the ESC key** in Analytic Solver Desktop, or by **clicking the ⏸ button** in the Task Pane Output tab in either app. If your model is very large, you might have to hold down the ESC key for a second or two. Since the input focus may not be on the Task Pane when you click with the mouse, you might have to click the ⏸ button twice. A dialog like the one below will appear:


Show Trial Solution

Simulation paused, click to continue or stop

Click **Stop** to stop the simulation or **Continue** if you want to continue.

Reasons Why Simulation Takes a Long Time

Analytic Solver Simulation is by far the fastest Monte Carlo simulation software for Excel on the market, and it takes full advantage of multi-core processors for even faster simulations. But it's still possible to create a model where a simulation takes a long time. There are several reasons for this:

1. If you have the Platform tab Simulation group Interpreter option set to **Excel Interpreter** in Analytic Solver Desktop, Excel is used to recalculate the worksheet on each Monte Carlo trial. This is usually an order of magnitude slower than the PSI Interpreter, though it requires significantly less memory. This functionality is not supported in Analytic Solver Cloud.
2. If you have Platform tab Simulation group Interpreter option set to **Psi Interpreter** (this is always the case in Analytic Solver Cloud), and the simulation is slow, you may be using certain Excel functions in ways that slow down the PSI Interpreter. See below for Do's and Don'ts to get the most from the PSI Interpreter.
3. Your model may simply be **very large**, with many complex formulas, references to other worksheets or workbooks, etc. that take a long time to evaluate. If your workbook includes sizable elements that aren't **essential to the simulation model** and its results, these extra elements may be slowing down the simulation process.

Time Versus Memory Tradeoffs in Analytic Solver Desktop

The PSI Interpreter is designed to take maximum advantage of available memory, and trade off memory against time. When Excel interprets =A1+A2*(A3-4), it uses memory for only four numbers – the values of A1, A2, A3 and the constant 4. But when PSI interprets this formula, it uses memory for 3,001 numbers – 1,000 each for A1, A2 and A3. If your PC has plenty of memory, PSI will help you take advantage of it. But if you are tight on memory, you may find that simulations slow down, because Windows will use 'virtual memory' and swap your data between main memory and your hard disk. You can add more memory **--** DRAM chips for PCs and notebooks are an amazing value today. However, the best solution is simply to run your simulation model using Analytic Solver Cloud! Since the Cloud app runs on Microsoft's Azure App Service, memory is not an issue in the cloud!

Model Design for Maximum Simulation Speed

Paying attention to your Excel worksheet layout and design, and avoiding certain practices, will help you get the most from the PSI Interpreter. Below is a brief list of Do's and Don'ts that will help you realize the best possible speed:

- **Do** build worksheets starting from the "upper left corner." **Don't** place cells or formulas at extreme row and column addresses all over the worksheet.
- **Do** build the model on a small number of worksheets. **Don't** include references on these worksheets to other worksheets that aren't required for the simulation model.
- **Do** use numeric and logical formulas and functions. **Don't** create string results in the middle of numeric calculations (see example below).
- **Do** use built-in Excel functions (including Analysis ToolPak functions) freely. **Don't** use third-party or user-written functions unless truly needed. Such functions must not have "side-effects" other than the value returned.
- **Do** use operators like + **-** * / ^ and functions like SUM, AVERAGE, MAX, MIN, AND, OR, NOT, IF, CHOOSE. **Don't** use INDIRECT, OFFSET, or TEXT, SEARCH, REPLACE, FIXED, DOLLAR, or ROMAN.
- If possible, **Don't** use array formulas, LOOKUP functions with large ranges as arguments, or long "chains" of formulas where each formula depends on an earlier formula (and hence on *all* earlier formulas). Each of these things can slow down analysis of your model considerably (by 10x in some cases).

A common practice that slows down the PSI Interpreter, and is easy to avoid, is illustrated by the following:

A1: =IF(B1>100,"Yes","No") and later A10: =IF(A1="Yes",A2,50)

You don't need "Yes" and "No" when Excel provides built-in values TRUE and FALSE. You could write =IF(B1>100,TRUE,FALSE), but this is simpler:

A1: =B1>100 and later A10: =IF(A1,A2,50)

The formula =B1>100 evaluates to either TRUE or FALSE. A1 holds this value and you can test it later in another formula.

When Simulation Results Seem Wrong

When the simulation results seem wrong, before doing anything else, you should **read any messages** in the Output tab, and **click the message** to display Help as described above. We emphasize this, because very often when we are contacted in technical support, the user has not taken these basic steps. As *next* steps:

1. Make sure you understand how your formulas behave in an ordinary **Excel recalculation**. If you've "raced ahead" with a simulation model before doing this and you are using Analytic Solver Desktop, you can use the Freeze/UnFreeze button on the Tools tab of the Solver Task Pane to turn your simulation model back into a "what-if" model (by moving all PSI function calls into cell comments), and later click Freeze/UnFreeze again to restore the PSI functions that make up your simulation model.
2. Bear in mind that an uncertain function cell normally displays the calculated value for the **last** Monte Carlo trial of the **last** simulation run. It's usually more meaningful to look at the results returned by PSI Statistics functions, such as PsiMean() or PsiPercentile(), across all the trials of the simulation.
3. It can be very helpful to **cycle through the Monte Carlo trials** of a simulation, and examine the values of uncertain function cells, as well as intermediate formula cells on which they depend. Just use the **left and right arrows** on the Analytic Solver Desktop Ribbon or the up and down arrows on the Tools tab of the Solver task pane in Analytic Solver Cloud to change the **trial index**. You may find that your model calculates a result you did not expect for some values of the uncertain variables.
4. If you are using multiple simulations, check that the **Sim # index** on the Ribbon in Analytic Solver Desktop or on the Tools tab of the Solver task pane in Analytic Solver Cloud, or the **Sim # dropdown** in the Uncertain Function dialog title bar, are selecting the simulation you want, and that parameters (PsiSimParam() functions) are returning the values you expect for that simulation.
5. If you haven't already done so, double-click uncertain function cells and examine the **Frequency tab** chart and **Percentiles tab** numbers to see how the values of this function were distributed across Monte Carlo trials in the simulation. This will help explain the values of PSI Statistics functions.

Although software bugs are always possible, consider carefully the possibility that the simulation results are **correct for the model you've defined**, and that your expectation is wrong. We sometimes find in technical support that what your model actually says is different from what you intended.

When Things Go Right: Getting Further Results

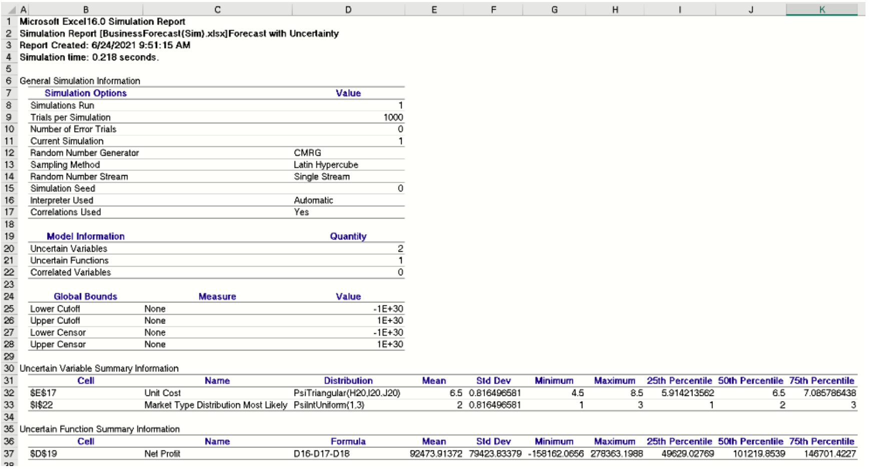
When you have simulation results on the worksheet that make sense to you, and you have the message "Simulation finished successfully" in **green** at the bottom of the Task Pane, there are several ways you can get further results. We'll cover five possibilities:

1. **Documenting your results** in a Simulation Report.
2. **Using all the features** of the Uncertain Function dialog.
3. **Fitting an analytic distribution** to uncertain function results.
4. **Charting multiple uncertain functions**, typically over time.

5. Performing **multiple parameterized simulations**, and capturing the results on the worksheet or in charts.

Using the Simulation Report

One quick step you can take, when you have simulation results that you want to recall later, is to select **Reports – Simulation – Simulation Report** to produce a report worksheet, inserted into your workbook, like the one for the example model **BusinessForecast(Sim).xlsx**, (open by clicking Help on the Analytic Solver ribbon, then Example Models – Monte Carlo Simulation Examples) shown on the next page. Click the Forecast with Uncertainty tab and run a simulation then run a Simulation Report.



Microsoft Excel16.0 Simulation Report
Simulation Report [BusinessForecast(Sim).xlsx]Forecast with Uncertainty
Report Created: 6/24/2021 9:51:15 AM
Simulation time: 0.218 seconds.

General Simulation Information

Simulation Options	Value
Simulations Run	1
Trials per Simulation	1000
Number of Error Trials	0
Current Simulation	1
Random Number Generator	CMRG
Sampling Method	Latin Hypercube
Random Number Stream	Single Stream
Simulation Seed	0
Interpreter Used	Automatic
Correlations Used	Yes

Model Information	Quantity
Uncertain Variables	2
Uncertain Functions	1
Correlated Variables	0

Global Bounds	Measure	Value
Lower Cutoff	None	-1E+30
Upper Cutoff	None	1E+30
Lower Censor	None	-1E+30
Upper Censor	None	1E+30

Uncertain Variable Summary Information

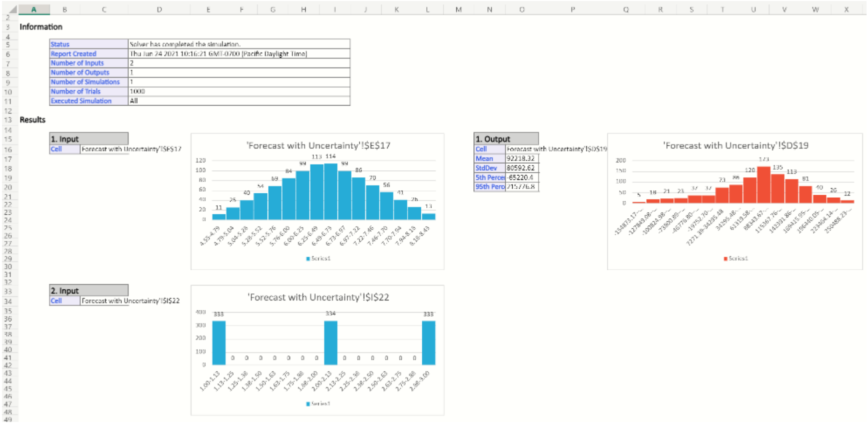
Cell	Name	Distribution	Mean	Std Dev	Minimum	Maximum	25th Percentile	50th Percentile	75th Percentile
$E$17	Unit Cost	PsiTriangular(H20,I20,J20)	6.5	0.816496581	4.5	8.5	5.914213562	6.5	7.085786438
$I$22	Market Type Distribution Most Likely	PsiIntUniform(1,3)	2	0.816496581	1	3	1	2	3

Uncertain Function Summary Information

Cell	Name	Formula	Mean	Std Dev	Minimum	Maximum	25th Percentile	50th Percentile	75th Percentile
$D$19	Net Profit	D16-D17-D18	92473.91372	79423.83379	-158162.0656	278363.1988	49629.02769	101219.8539	146701.4227

The Simulation Report in Analytic Solver Desktop documents the option settings used to perform the simulation and provides summary information about the uncertain variables and uncertain functions in your simulation model. You are much more likely to use simulation charts, or snapshots of your own worksheet layout when you are presenting reports to others, but this report can be quite useful when you want to refer later to the assumptions behind a simulation analysis.

This report has been completely redesigned in Analytic Solver Cloud. Select **Reports – Simulation – Simulation Report** to produce a report worksheet, inserted into your Excel Online or Desktop Excel workbook, like the one for the example model **BusinessForecast(Sim).xlsx**, as shown on the next page.



Information

Status	Solver has completed the simulation.
Report Created	Thu Jun 24 2021 10:16:21 GMT-0700 (Pacific Daylight Time)
Number of Inputs	2
Number of Outputs	1
Number of Simulations	1
Number of Trials	1000
Executed Simulation	All

Results

1. Input	
Cell	Forecast with Uncertainty'!$E$17

1. Output	
Cell	Forecast with Uncertainty'!$D$19
Mean	92218.32
StdDev	80592.62
5th Perce	-65220.4
95th Perc	215776.8

2. Input	
Cell	Forecast with Uncertainty'!$I$22

In Analytic Solver Cloud, this report displays the number of input and output functions in the simulation model, the number of simulations and the number of trials. The body of the report contains the distribution of values for all uncertain variables and functions along with the cell address, the mean, the standard deviation and the 5th and 95th percentiles of each.

Using the Uncertain Function Dialog

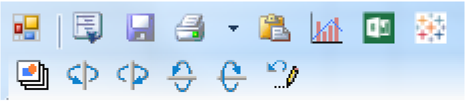
The Uncertain Function dialog has many features, described in the Frontline Solvers Reference Guide, that you can use to get enhanced results from your simulation. For an introduction to these features, refer back to “A First Simulation Example” in the chapter “Simulation and Risk Analysis,” especially the sections starting with “Viewing the Full Range of Profit Outcomes.” Here, we’ll just reiterate the results available at your fingertips in the Uncertain Function dialog:

Dialog Tabs

- The **Frequency**, **Cumulative Frequency**, and **Reverse Cumulative Frequency** tabs provide three different views of the full range of outcomes for an uncertain function. In Analytic Solver Desktop, you can click to display “crosshairs” with numerical values on these charts or add Lower and Upper bounds to see the estimated probability of a profit or loss.
- The Tornado chart on the **Sensitivity** tab quickly shows you which uncertain variables have the **greatest impact** on this uncertain function, across the full range of Monte Carlo trials. The **Scatter Plots** tab often reveals further insights about the behavior of this uncertain function versus each uncertain variable, or versus other uncertain functions.

Panels and Toolbars

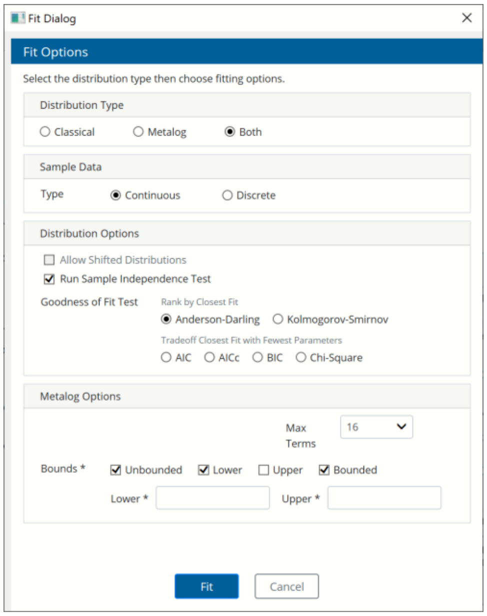
- Using the **right panel** of the dialog, you can access a drop down menu including **Statistics**, **Percentiles**, **Chart Type**, **Chart Options (Analytic Solver Desktop)**, **Axis Options (Analytic Solver Desktop)**, and **Markers (Analytic Solver Desktop)**. From the **Options dialog** (accessible from the Options button on the Analytic Solver Desktop Ribbon), on the **Charts** and **Markers** tabs, you can set default chart and marker settings for all of your charts.
- Using the **title toolbar icons** in Analytic Solver Desktop, you can save your settings, **print** the charts or numbers from any of the dialog tabs, or **copy** the charts or numbers to the Windows Clipboard, where they can be pasted into other applications. You can also **fit a distribution** (as shown below) to the simulation results or upload the simulation results to Microsoft's Power BI application or Tableau. (See the previous chapter, *Examples: Simulation and Risk Analysis*, for instructions on exporting model results to Power BI and Tableau.)
- Click the rightmost icon on the title toolbar in Analytic Solver Desktop to display the **3-D toolbar** that lets you – with a single click – shift between 2-D and 3-D, **shrink** or **magnify** the chart, or **move** or **rotate** the chart to a different perspective.



Fitting a Classical Distribution to Simulation Results

One way to analyze your simulation results is to see whether the distribution of outcomes is similar to a well-known analytic distribution. Frontline Solver's Analytic Solver supports a wide range of analytic distributions, and can fit a distribution and its parameters to sample data. To fit your sample simulation results, click the icon on the Uncertain Function dialog title toolbar in Analytic Solver Desktop or click Tools – Fit on the Analytic Solver Cloud or Analytic Solver Desktop ribbon.

Click cell D19 on the Forecast with Uncertainty tab within BusinessForecast(Sim).xlsx. When the Fit icon on the Analytic Solver Desktop Ribbon is clicked or Tools – Fit on the Analytic Solver Cloud ribbon, the **Fit Options** dialog appears.



Fit Dialog

Fit Options

Select the distribution type then choose fitting options.

Distribution Type

○ Classical ○ Metalog ● Both

Sample Data

Type ● Continuous ○ Discrete

Distribution Options

☐ Allow Shifted Distributions

☑ Run Sample Independence Test

Goodness of Fit Test

Rank by Closest Fit

● Anderson-Darling ○ Kolmogorov-Smirnov

Tradeoff Closest Fit with Fewest Parameters

○ AIC ○ AICc ○ BIC ○ Chi-Square

Metalog Options

Max Terms 16

Bounds * ☑ Unbounded ☑ Lower ☐ Upper ☑ Bounded

Lower *

Upper *

Fit Dialog Field Descriptions

- Sample Data: Type: Continuous or Discrete (Classical Distributions only.)
    - For the **Type**, select **Continuous** if the uncertain variable's values are highly divisible – such as most prices, volumes, interest rates, exchange rates, weights, and distances – or **Discrete** if the underlying physical process involves discrete, countable entities.
- Distribution Options
    - Allow Shifted Distributions **-** The checkbox **Allow Shifted Distributions** allows Analytic Solver to shift the center of analytic distribution (equivalent to using the PsiShift() property function) to better fit the sample data; sometimes this is not desirable, so the box may be unchecked.
    - Run Sample Independence Test **-** When fitting a sample to a distribution, it is important that the trial values are not in any way correlated with each other. In other words it is important

that each trial is independent. If the checkbox **Run Sample Independence Test** is checked, the Psi Interpreter will run an independence test and will report an error if the trial values are found to be dependent.

- Goodness of Fit Tests: Rank by Closest Fit

  For Continuous data, you can choose to rank the fitted distributions by **Anderson-Darling** or **Kolmogorov-Smirnoff** statistics.

  - Kolmogorov-Smirnoff –This test computes the difference (D) between the continuous distribution function (CDF) and the empirical cumulative distribution function (ECDF). The null hypothesis is rejected if, at the 90% significance level, D is larger than the critical value statistic.
  - Anderson-Darling –Ranks the fitted distribution using the Anderson Darling statistic, A^2. The null hypothesis is rejected using a 90% significance level, if A^2 is larger than the critical value statistic. This test awards more weight to the distribution tails then the Kolmogorov-Smirnoff test.

- Goodness of Fit Tests: Tradeoff Closest Fit with Fewest Parameters

  - AIC/BIC – The AIC test is a Chi Squared test corrected for the number of distribution parameters and sample size.

    AIC = Chi-Square Statistic + 2 * k + 2 * k * (k + 1) / (n – k – 1)

    where k is the number of distribution parameters and n is the sample size.
  - AICc –When the sample size is small, there is a significant chance that the AIC test will select a model with a large number of parameters. In other words, AIC will overfit the data. AICc was developed to reduce the possibility of overfitting by applying a penalty to the number of parameters. . Assuming that the model is univariate, is linear in the parameters and has normally-distributed residuals, the formula for AICc is:

    AICc = AIC + $\frac{2k^2+2k}{n-k-1}$

    where n = sample size, k = # of parameters.

    As the sample size approaches infinity, the penalty on the number of parameters converges to 0 resulting in AICc converging to AIC.
  - BIC – The Bayesian information criterion (BIC) is defined as:

    BIC = k ln(n) = 2 ln ($\hat{L}$)

    where

    $\hat{L}$ = the maximized value of the likelihood function of the model M. $\hat{L} = p(x|\theta, M)$ where θ are the parameter values that maximize the likelihood function and x is the observed data.

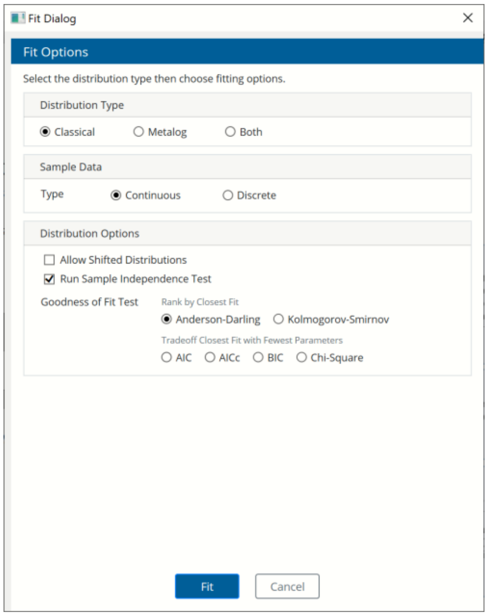
n = Sample size

k = Number of parameters

- Chi Square – Uses the chi-square statistic and distribution to rank either continuous or discrete distributions. Sample data is first divided into intervals using either equal probability, then the number of points that fall into each interval are compared with the expected number of points in each interval. The null hypotheses is rejected using a 90% significance level, if the the chi-squared test statistic is greater than the critical value statistic.

  Note: The Chi Square test is used indirectly in continuous fitting as a support in the AIC test. The AIC test must succeed in both discrete and continuous fitting as this is a necessary condition. When fitting a discrete function, the Chi Square test must also succeed. When fitting a continuous function, at least one of the tests, Chi Squared, Kolmogorov-Smirnoff, or Anderson-Darling, must succeed as well.

Select **Classical** for *Distribution Type* and click the **Fit** button.



Fit Dialog

Fit Options

Select the distribution type then choose fitting options.

Distribution Type

● Classical ○ Metalog ○ Both

Sample Data

Type ● Continuous ○ Discrete

Distribution Options

☐ Allow Shifted Distributions

☑ Run Sample Independence Test

Goodness of Fit Test

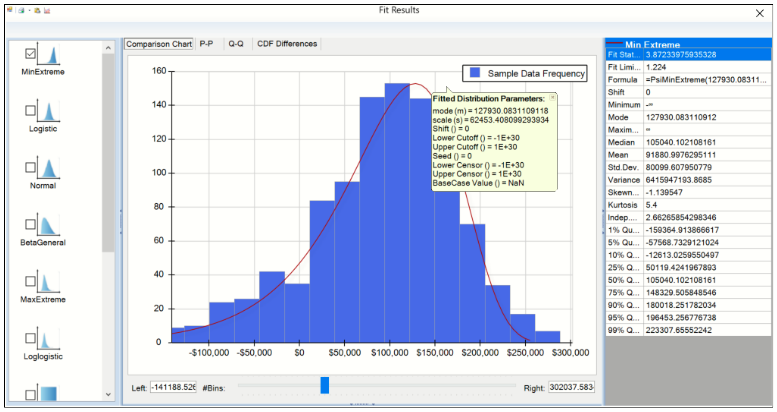
Rank by Closest Fit

● Anderson-Darling ○ Kolmogorov-Smirnov

Tradeoff Closest Fit with Fewest Parameters

○ AIC ○ AICc ○ BIC ○ Chi-Square

In a few seconds, Analytic Solver fits a wide range of distributions to the sample data, ranks them in order of the selected goodness-of-fit criterion (AIC/BIC statistics by default, for more information see the Frontline Solvers Reference Guide), and displays the best-fitting distribution.



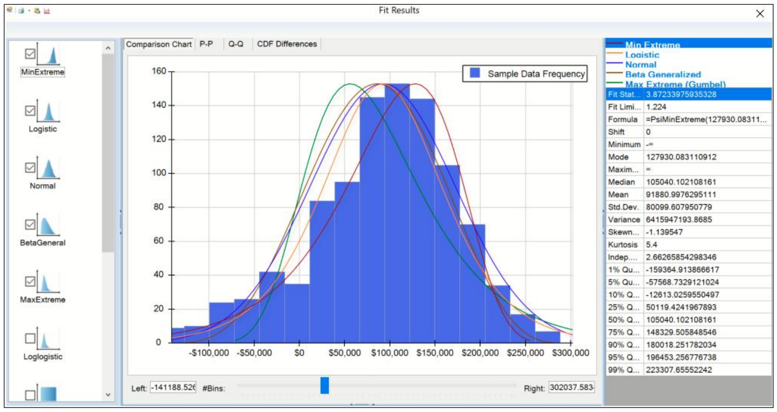
Fitted Distribution Parameters:
mode (m) = 127930.0831109118
scale (s) = 62453.408099293934
Shift () = 0
Lower Cutoff () = -1E+30
Upper Cutoff () = 1E+30
Seed () = 0
Lower Censor () = -1E+30
Upper Censor () = 1E+30
BaseCase Value () = NaN

Left: -141188.526 #Bins: Right: 302037.583

Min Extreme

Fit Stat...	3.87233975935328
Fit Limi...	1.224
Formula	=PsiMinExtreme(127930.08311...
Shift	0
Minimum	-∞
Mode	127930.083110912
Maxim...	∞
Median	105040.102108161
Mean	91880.9976295111
Std.Dev.	80099.607950779
Variance	6415947193.8685
Skewn...	-1.139547
Kurtosis	5.4
Indep....	2.66265854298346
1% Qu...	-159364.913866617
5% Qu...	-57568.7329121024
10% Q...	-12613.0259550497
25% Q...	50119.4241967893
50% Q...	105040.102108161
75% Q...	148329.505848546
90% Q...	180018.251782034
95% Q...	196453.256776738
99% Q...	223307.65552242

If you select the checkboxes adjacent to the additional distributions in the left panel, they will be superimposed on the frequency chart in the middle of the dialog.



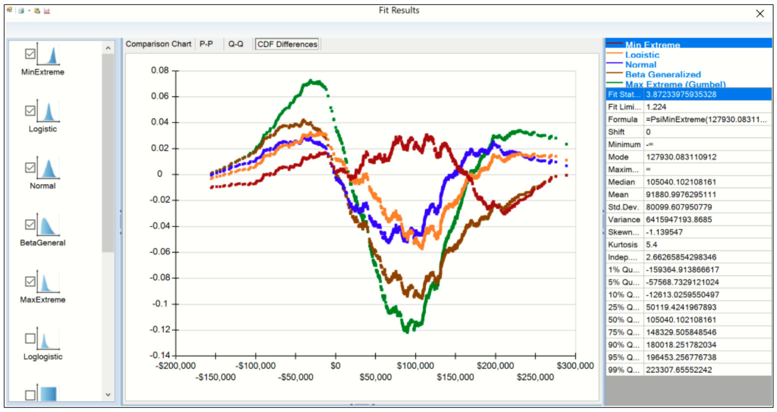
Left: -141188.526 #Bins: Right: 302037.583

Min Extreme
Logistic
Normal
Beta Generalized
Max Extreme (Gumbel)

Fit Stat...	3.87233975935328
Fit Limi...	1.224
Formula	=PsiMinExtreme(127930.08311...
Shift	0
Minimum	-∞
Mode	127930.083110912
Maxim...	∞
Median	105040.102108161
Mean	91880.9976295111
Std.Dev.	80099.607950779
Variance	6415947193.8685
Skewn...	-1.139547
Kurtosis	5.4
Indep....	2.66265854298346
1% Qu...	-159364.913866617
5% Qu...	-57568.7329121024
10% Q...	-12613.0259550497
25% Q...	50119.4241967893
50% Q...	105040.102108161
75% Q...	148329.505848546
90% Q...	180018.251782034
95% Q...	196453.256776738
99% Q...	223307.65552242

Other tabs of this dialog give you a visual picture of the fit, in the form of P-P and Q-Q charts, and a chart of CDF Differences, shown below.

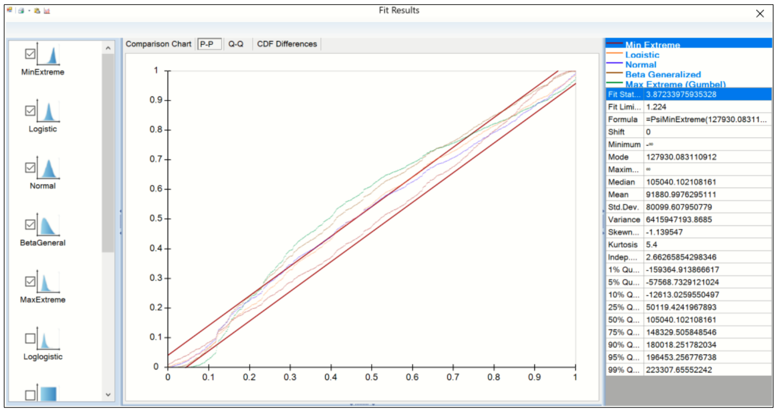
The Cumulative Distribution Function Differences chart quantifies the difference between the empirical distribution function and the cumulative distribution function of the distribution.



Min Extreme
Logistic
Normal
Beta Generalized
Max Extreme (Gumbel)

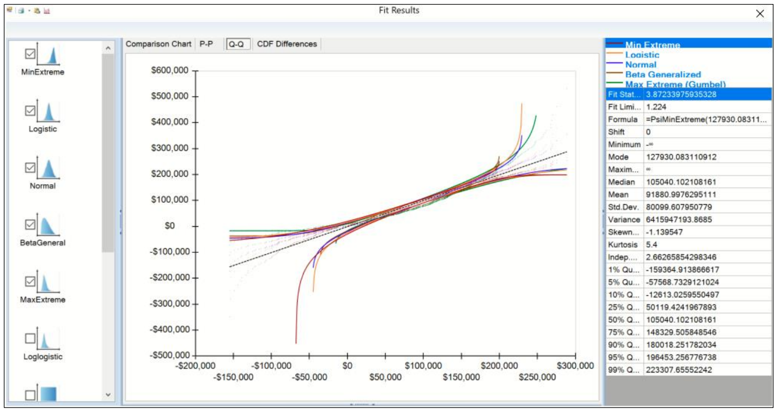
Fit Stat...	3.87233975935328
Fit Limi...	1.224
Formula	=PsiMinExtreme(127930.08311...
Shift	0
Minimum	-∞
Mode	127930.083110912
Maxim...	∞
Median	105040.102108161
Mean	91880.9976295111
Std.Dev.	80099.607950779
Variance	6415947193.8685
Skewn...	-1.139547
Kurtosis	5.4
Indep....	2.66265854298346
1% Qu...	-159364.913866617
5% Qu...	-57568.7329121024
10% Q...	-12613.0259550497
25% Q...	50119.4241967893
50% Q...	105040.102108161
75% Q...	148329.505848546
90% Q...	180018.251782034
95% Q...	196453.256776738
99% Q...	223307.65552242

The P-P tab displays the Probability-Probability chart. This graph is used to determine how well a specific distribution fits the observed data. This chart will be approximately linear (or lie on the y = x line) if the specified distribution is the correct choice.



Fit Stat...	3.87233975935328
Fit Limi...	1.224
Formula	=PsiMinExtreme(127930.08311...
Shift	0
Minimum	-∞
Mode	127930.083110912
Maxim...	∞
Median	105040.102108161
Mean	91880.9976295111
Std.Dev.	80099.607950779
Variance	6415947193.8685
Skewn...	-1.139547
Kurtosis	5.4
Indep....	2.66265854298346
1% Qu...	-159364.913866617
5% Qu...	-57568.7329121024
10% Q...	-12613.0259550497
25% Q...	50119.4241967893
50% Q...	105040.102108161
75% Q...	148329.505848546
90% Q...	180018.251782034
95% Q...	196453.256776738
99% Q...	223307.65552242

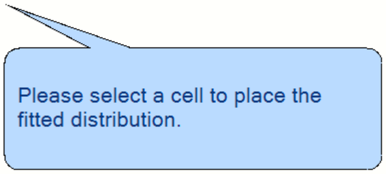
Click the Q-Q tab to see the Quantile – Quantile chart. Again, if the two distributions being compared are a good fit, the plot will lie approximately on the y = x line.



Fit Stat...	3.87233975935328
Fit Limi...	1.224
Formula	=PsiMinExtreme(127930.08311...
Shift	0
Minimum	-∞
Mode	127930.083110912
Maxim...	∞
Median	105040.102108161
Mean	91880.9976295111
Std.Dev.	80099.607950779
Variance	6415947193.8685
Skewn...	-1.139547
Kurtosis	5.4
Indep....	2.66265854298346
1% Qu...	-159364.913866617
5% Qu...	-57568.7329121024
10% Q...	-12613.0259550497
25% Q...	50119.4241967893
50% Q...	105040.102108161
75% Q...	148329.505848546
90% Q...	180018.251782034
95% Q...	196453.256776738
99% Q...	223307.65552242

Analytic Solver allows you to accept one of the distributions on the left pane. Simply unselect all but one distribution and then click the x on the upper left hand corner to close the dialog.

A small message will attach to your mouse pointer asking you to select a cell to place the fitted distribution.



Please select a cell to place the fitted distribution.

Select a blank cell to input the fitted distribution.

Use the PsiDistInfo(cell) function in a 2nd blank cell to display the name of the function. For more information on this function, see the Analytic Solver Reference Guide.

You can also fit an analytic distribution to data that you obtain from another source, such as historical observations of the process you want to model. If you have this sample data on your Excel worksheet, you can simply select the cell range containing the data and click **Tools-Fit** icon on the Analytic Solver

Ribbon. Using this tool, you can easily create an **uncertain variable** whose distribution models this data.

Fitting a Meta-Log Distribution to Simulation Results

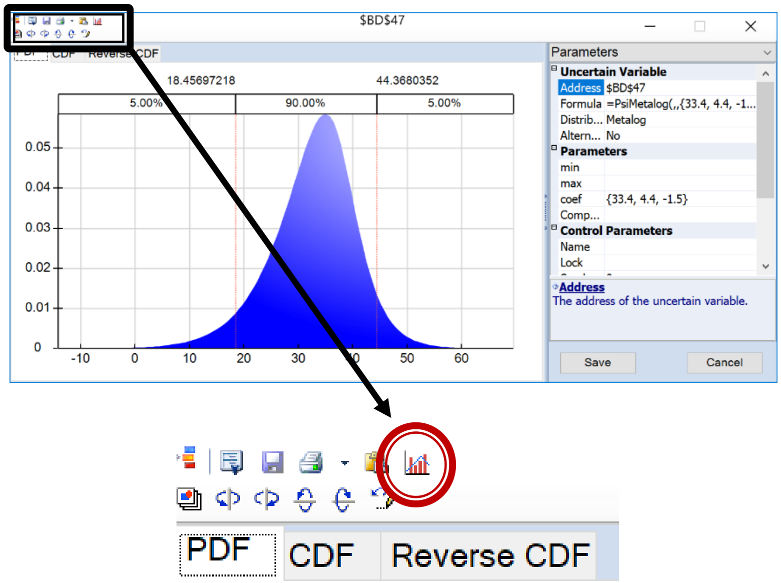
A meta-log distribution is an alternative distribution to a fitted distribution. The metalog distributions are a collection of continuous univariate probability distributions which can be used when cumulative distribution function data is available. For more information on these distributions, see the descriptions for PsiMetalog() in the Continuous Analytic Distribution subsection within the "Psi Function Reference" chapter that appears in the *Frontline Solvers Reference Guide*.

A general meta-log distribution is defined in a simulation model by using the Psi distribution, PsiMetalog(). This function has as arguments a vector of coefficients (coefficients) and optionally lower (min) and upper (max) bounds. The metalog distribution family is ***designed*** to be determined from historical data, without requiring a distribution fitting process. They are computed from a set of historical data pairs {y, x}, where y is a cumulative probability and x is the corresponding percentile. The coefficients argument is either a range or an array of 2 to 10 numerical elements.

Fitting the PsiMetalog() function in Analytic Solver may be performed in two ways, by using the Psi function PsiMetalogFit() or by using the Fit dialog. For more information on PsiMetalogFit, see the Analytic Solver Reference Guide.

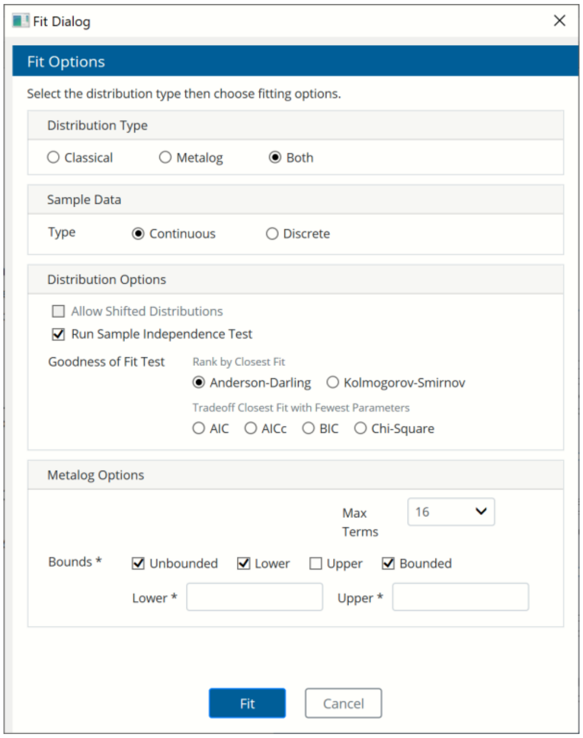
Click back to cell D19 in the BusinessForecast(Sim) workbook. To open the Fit dialog, click **Tools-Fit** on the Analytic Solver ribbon.

The Fit dialog may also be opened from the Uncertain Function dialog by clicking the Fit Distribution icon on the title bar.



Uncertain Variable
Address $BD$47
Formula =PsiMetalog(,,{33.4, 4.4, -1...
Distrib... Metalog
Altern... No
Parameters
min
max
coef {33.4, 4.4, -1.5}
Comp...
Control Parameters
Name
Lock

Address
The address of the uncertain variable.



Fit Dialog

Fit Options

Select the distribution type then choose fitting options.

Distribution Type

○ Classical ○ Metalog ● Both

Sample Data

Type ● Continuous ○ Discrete

Distribution Options

☐ Allow Shifted Distributions

☑ Run Sample Independence Test

Goodness of Fit Test

Rank by Closest Fit

● Anderson-Darling ○ Kolmogorov-Smirnov

Tradeoff Closest Fit with Fewest Parameters

○ AIC ○ AICc ○ BIC ○ Chi-Square

Metalog Options

Max Terms: 16

Bounds * ☑ Unbounded ☑ Lower ☐ Upper ☑ Bounded

Lower *

Upper *

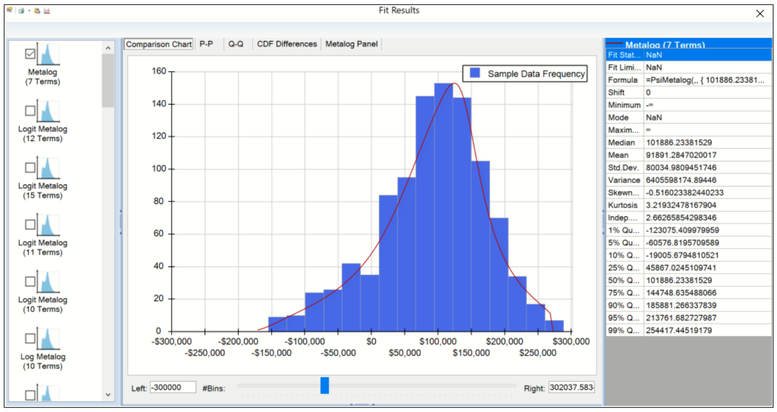
Fit Dialog Metalog Options

- Max Terms **-** Number of terms in the equation. The default is 16. The minimum number of terms is 2. Metalogs of all terms, up to Max Terms, will be calculated and displayed in the results.
- Bounds – Unbounded, Lower, Upper and/or Bounded
    - Unbounded – Select this checkbox to fit to only unbounded metalog distributions.
    - Lower – Select this checkbox to fit only to distributions with a lower bound. When this checkbox is selected, a lower bound must be entered into the Lower field. Enter a value that is less than all data points. If omitted, negative infinity is used for the lower bound.
    - Upper – Select this checkbox to fit only to distributions with an upper bound. When this checkbox is selected, an upper bound must be entered into the Upper field. Enter a value that is greater than all data points. If omitted, positive infinity is used for the upper bound.
    - Bounded – Select this checkbox to fit only to bounded distributions. When this checkbox is selected an upper and lower bound must be entered into both the Upper and Lower fields.

Note: In past versions of Analytic Solver, historical data had to be greater than (>) the entered lower bound and less than (<) the entered upper bound, otherwise fitting would fail. In the latest version of Analytic Solver, historical data may now be greater than or equal to (>=) the lower bound and less than or equal to (<=) the upper bound.

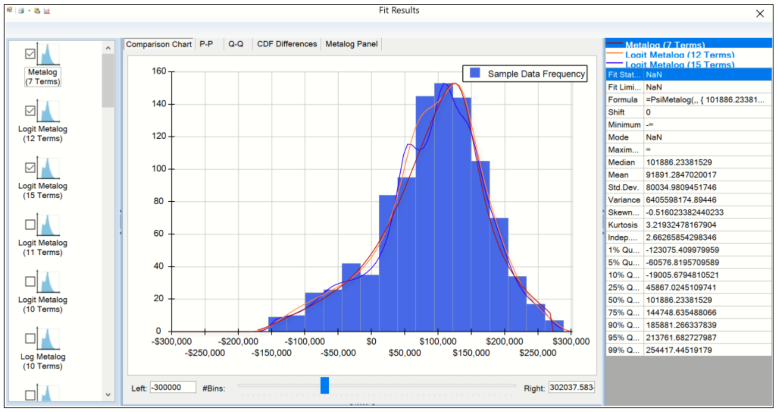
Enter -300,000 for the lower bound and 300,000 for the upper bound, then click **Fit** to fit to a Meta-log distribution. (Click Cancel to close the dialog without performing the distribution fitting.)

Analytic Solver computes and displays a ranked list of candidate fitted distributions, as shown below. Initially, the distribution with the best fit statistic (selected in the above dialog) is shown in a chart that overlays the sample data. Use the #Bins slider to add of subtract the number of bins in the histogram.



Fit Stat...	NaN
Fit Limi...	NaN
Formula	=PsiMetalog(,, { 101886.23381...
Shift	0
Minimum	-∞
Mode	NaN
Maxim...	∞
Median	101886.23381529
Mean	91891.2847020017
Std.Dev.	80034.9809451746
Variance	6405598174.89446
Skewn...	-0.516023382440233
Kurtosis	3.21932478167904
Indep....	2.66265854298346
1% Qu...	-123075.409979959
5% Qu...	-60576.8195709589
10% Q...	-19005.6794810521
25% Q...	45867.0245109741
50% Q...	101886.23381529
75% Q...	144748.635488066
90% Q...	185881.266337839
95% Q...	213761.682727987
99% Q...	254417.44519179

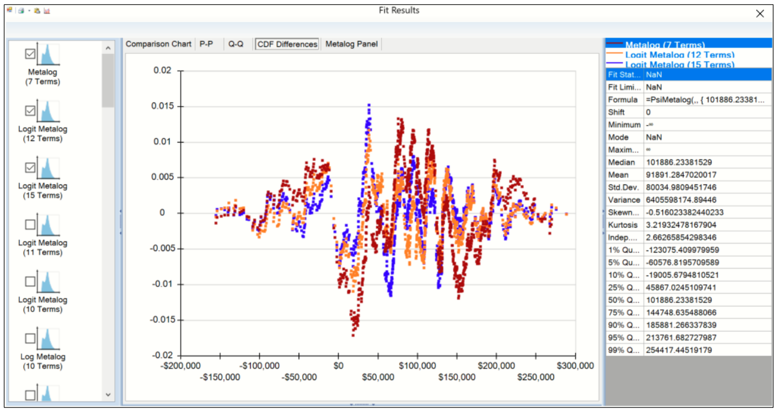
On the left of the dialog is a list of fit choices shown in order of decreasing Goodness of Fit based on the test you chose. You can select multiple choices to see how they compare and choose the one you are most happy with. See the example shown on the next page.



Fit Stat...	NaN
Fit Limi...	NaN
Formula	=PsiMetalog(,, { 101886.23381...
Shift	0
Minimum	-∞
Mode	NaN
Maxim...	∞
Median	101886.23381529
Mean	91891.2847020017
Std.Dev.	80034.9809451746
Variance	6405598174.89446
Skewn...	-0.516023382440233
Kurtosis	3.21932478167904
Indep....	2.66265854298346
1% Qu...	-123075.409979959
5% Qu...	-60576.8195709589
10% Q...	-19005.6794810521
25% Q...	45867.0245109741
50% Q...	101886.23381529
75% Q...	144748.635488066
90% Q...	185881.266337839
95% Q...	213761.682727987
99% Q...	254417.44519179

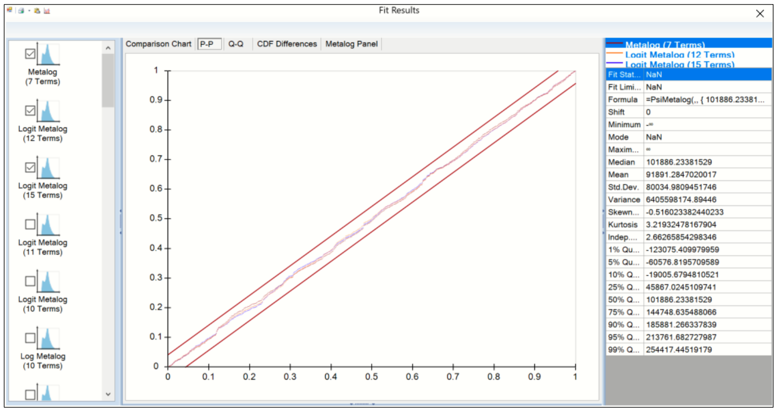
You can also click on the P-P, Q-Q, CDF Differences tabs to get additional perspective on which distribution best fits.

The Cumulative Distribution Function Differences chart quantifies the difference between the empirical distribution function and the cumulative distribution function of the distribution.



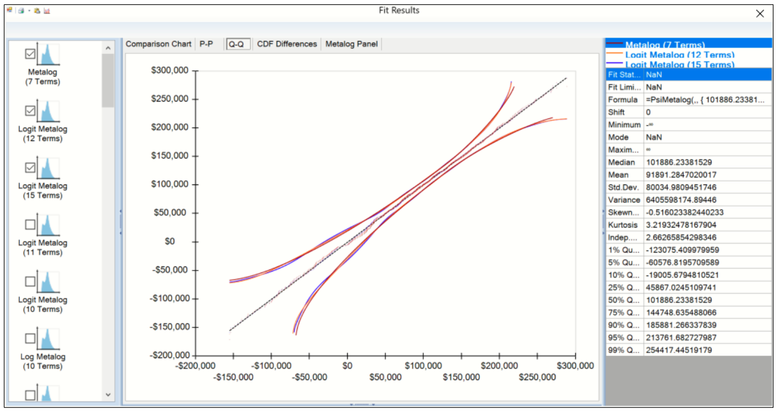
Fit Stat...	NaN
Fit Limi...	NaN
Formula	=PsiMetalog(,, { 101886.23381...
Shift	0
Minimum	-∞
Mode	NaN
Maxim...	∞
Median	101886.23381529
Mean	91891.2847020017
Std.Dev.	80034.9809451746
Variance	6405598174.89446
Skewn...	-0.516023382440233
Kurtosis	3.21932478167904
Indep....	2.66265854298346
1% Qu...	-123075.409979959
5% Qu...	-60576.8195709589
10% Q...	-19005.6794810521
25% Q...	45867.0245109741
50% Q...	101886.23381529
75% Q...	144748.635488066
90% Q...	185881.266337839
95% Q...	213761.682727987
99% Q...	254417.44519179

The P-P tab displays the Probability-Probability chart. This graph is used to determine how well a specific distribution fits the observed data. This chart will be approximately linear (or lie on the y = x line) if the specified distribution is the correct choice.



Fit Stat...	NaN
Fit Limi...	NaN
Formula	=PsiMetalog(,, { 101886.23381...
Shift	0
Minimum	-∞
Mode	NaN
Maxim...	∞
Median	101886.23381529
Mean	91891.2847020017
Std.Dev.	80034.9809451746
Variance	6405598174.89446
Skewn...	-0.516023382440233
Kurtosis	3.21932478167904
Indep....	2.66265854298346
1% Qu...	-123075.409979959
5% Qu...	-60576.8195709589
10% Q...	-19005.6794810521
25% Q...	45867.0245109741
50% Q...	101886.23381529
75% Q...	144748.635488066
90% Q...	185881.266337839
95% Q...	213761.682727987
99% Q...	254417.44519179

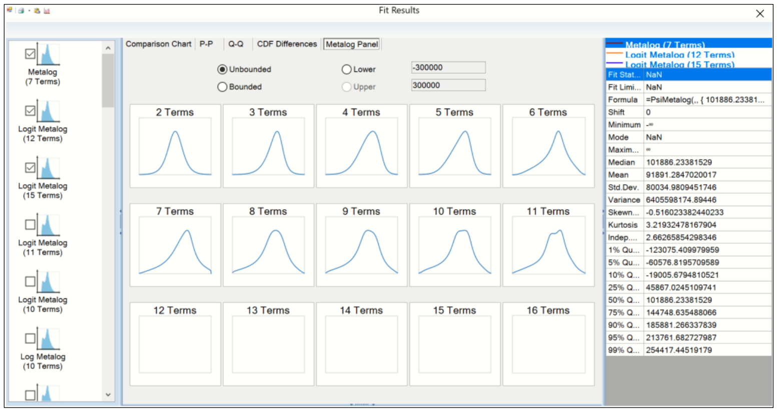
Click the Q-Q tab to see the Quantile – Quantile chart. Again, if the two distributions being compared are a good fit, the plot will lie approximately on the y = x line.



Fit Stat...	NaN
Fit Limi...	NaN
Formula	=PsiMetalog(,, { 101886.23381...
Shift	0
Minimum	-∞
Mode	NaN
Maxim...	∞
Median	101886.23381529
Mean	91891.2847020017
Std.Dev.	80034.9809451746
Variance	6405598174.89446
Skewn...	-0.516023382440233
Kurtosis	3.21932478167904
Indep....	2.66265854298346
1% Qu...	-123075.409979959
5% Qu...	-60576.8195709589
10% Q...	-19005.6794810521
25% Q...	45867.0245109741
50% Q...	101886.23381529
75% Q...	144748.635488066
90% Q...	185881.266337839
95% Q...	213761.682727987
99% Q...	254417.44519179

Tip: Once you have accepted the distribution you can open it back up and modify the lower and upper bounds, shift, etc.

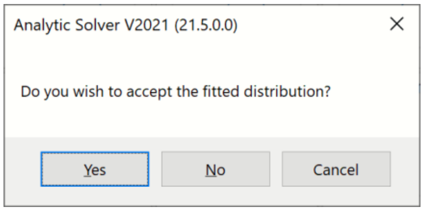
Click the Metalog Panel tab to view all feasible metalog density functions. Distributions with 7-11, and 13-15 terms are infeasible and therefore do not appear in the results. Notice how the distributions evolve as the number of terms are increased. Note that the larger the number of terms, the greater the possibility that the distribution will overfit the data.



● Unbounded
○ Bounded
○ Lower -300000
○ Upper 300000

Fit Stat...	NaN
Fit Limi...	NaN
Formula	=PsiMetalog(,, { 101886.23381...
Shift	0
Minimum	-∞
Mode	NaN
Maxim...	∞
Median	101886.23381529
Mean	91891.2847020017
Std.Dev.	80034.9809451746
Variance	6405598174.89446
Skewn...	-0.516023382440233
Kurtosis	3.21932478167904
Indep....	2.66265854298346
1% Qu...	-123075.409979959
5% Qu...	-60576.8195709589
10% Q...	-19005.6794810521
25% Q...	45867.0245109741
50% Q...	101886.23381529
75% Q...	144748.635488066
90% Q...	185881.266337839
95% Q...	213761.682727987
99% Q...	254417.44519179

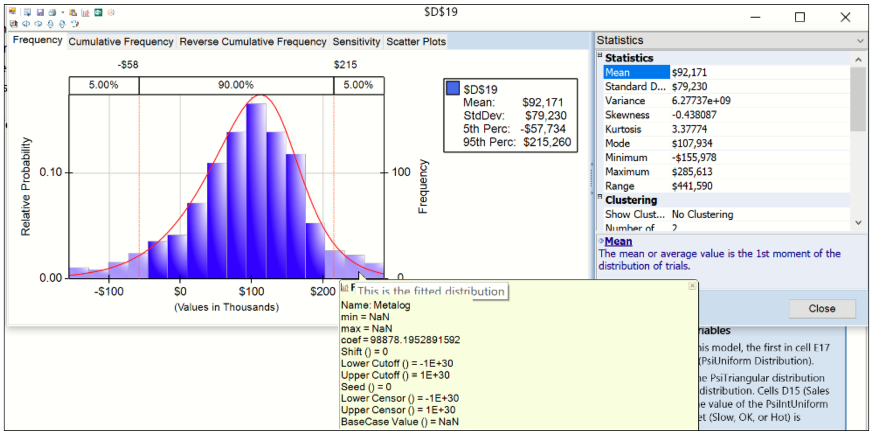
Uncheck all distributions on the left except for the top distribution, Metalog (3 Terms), then click the X in the upper right hand corner. You'll be asked if you'd like to accept the fitted distribution. Click Yes.


Analytic Solver V2021 (21.5.0.0)

Do you wish to accept the fitted distribution?

The Uncertain Function dialog opens with the updated Metalog distribution with 3 terms. Hover over the red density function to open a window containing all relevant statistics for this fit distribution.

Use the PsiDistInfo(cell) function in a 2nd blank cell to display the name of the function. For more information on this function, see the Analytic Solver Reference Guide.



$D$19
Mean: $92,171
StdDev: $79,230
5th Perc: -$57,734
95th Perc: $215,260

Statistics

Statistics

Mean	$92,171
Standard D...	$79,230
Variance	6.27737e+09
Skewness	-0.438087
Kurtosis	3.37774
Mode	$107,934
Minimum	-$155,978
Maximum	$285,613
Range	$441,590

Clustering

Show Clust...	No Clustering
Number of	2

Mean
The mean or average value is the 1st moment of the distribution of trials.

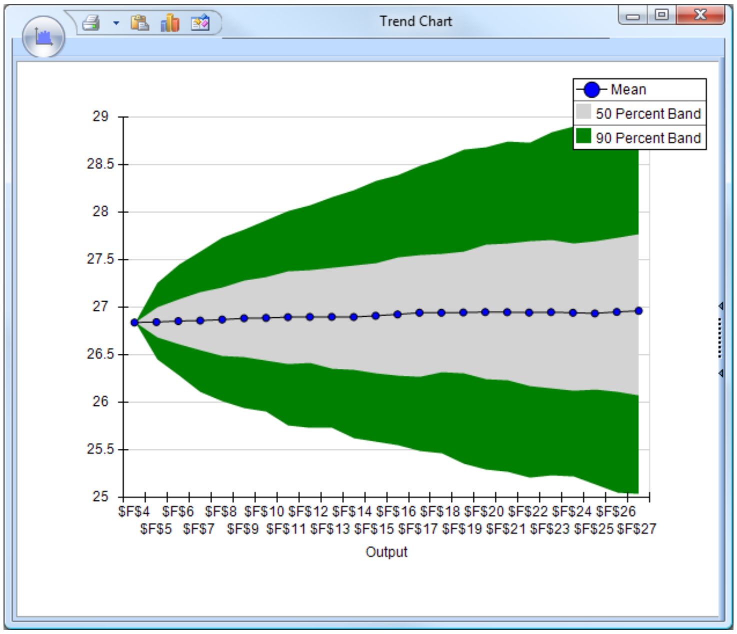
This is the fitted distribution
Name: Metalog
min = NaN
max = NaN
coef = 98878.1952891592
Shift () = 0
Lower Cutoff () = -1E+30
Upper Cutoff () = 1E+30
Seed () = 0
Lower Censor () = -1E+30
Upper Censor () = 1E+30
BaseCase Value () = NaN

It's also possible to fit a distribution to a series of historical data points. See the section *Using the Fit Feature* within the chapter **Masting Simulation and Risk Analysis Concepts** for more information.

Charting Multiple Uncertain Functions

In some simulation models, you will have a series of uncertain functions that represent the evolution of some process over time. The individual cells might represent sales or inventory levels, interest rates or exchange rates, or some other uncertain factor. You want to understand how this uncertain function changes over time, taking into account its *uncertainty*.

Analytic Solver Comprehensive has a general facility for creating charts of multiple simulation results, including Overlay, Trend, and Box-Whisker charts. Below, we've used this facility to visualize the evolution of a stock price over time, using the example model GBMSimpleModel.xlsx, which simulates stock prices using a Geometric Brownian Motion model. When we perform a simulation, select **Charts – Multiple Simulation Results – Trend**, and select the first 24 uncertain functions for inclusion in the chart, we get a Trend Chart like the one below:



Here we can see a slow upward drift of the mean value of the stock price, and also its volatility around the mean, depicted here with the 25th and 75th, and the 10th and 90th percentiles.

Multiple Parameterized Simulations

Once you have results from a single simulation, you normally want to explore "what if" scenarios, where you change some parameter that is under your control and run simulations to observe the effects of the uncertain variables that are *not* under your control.

Using Interactive Simulation in Analytic Solver Desktop

Analytic Solver Simulation offers a powerful facility to do this through Interactive Simulation: You can simply click the Simulate button on the Analytic Solver Cloud Ribbon to enter Interactive mode, then change a number on your worksheet and instantly see new simulation results. This is illustrated in "A First Simulation Example" in the earlier chapter "Simulation and Risk

Analysis." Analytic Solver is fast enough to make this sort of "what-if" analysis practical.

Using Multiple Simulations

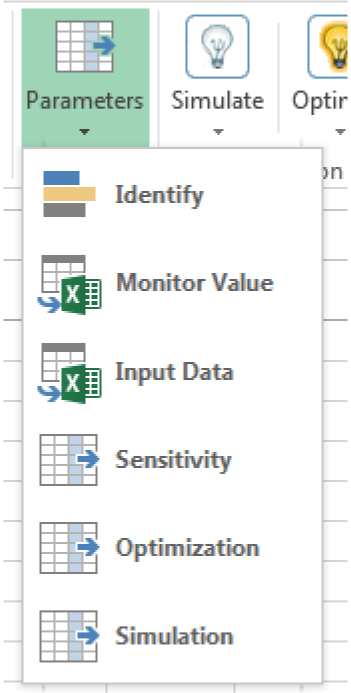
But after exploring the model and its results through Interactive Simulation, you'll likely want to automate the "what-if" process, and more systematically evaluate the effects of changing some parameter or parameters under your control on the simulation results. Analytic Solver makes this easy, with multiple parameterized simulations. This is illustrated in "An Airline Revenue Management Model" in the chapter "Simulation and Risk Analysis."

Setting Up Multiple Parameterized Simulations

Two simple steps are required to set up a multiple parameterized simulation:

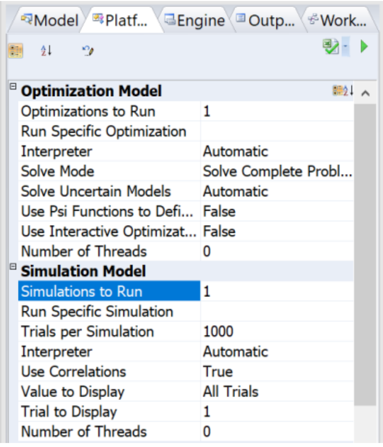
1. Define one or more **simulation parameters**, and use them in your model.
2. Set the **number of simulations** option to a value greater than 1, and run the simulation.

To define a simulation parameter, simply select **Parameters – Simulation** from the Ribbon:



Analytic Solver Simulation displays a dialog box, where you can enter a lower limit and upper limit for the parameter value, or a specific list of values that you would like the parameter to take on successive simulations. If you specify a lower and upper limit, then when you set the **number of simulations**, the parameter will take on equally-spaced values from the lower to the upper limit.

To set the number of simulations, simply edit the option in the Task Pane Platform tab Simulation group option **Simulations to Run**:



Optimization Model	
Optimizations to Run	1
Run Specific Optimization	
Interpreter	Automatic
Solve Mode	Solve Complete Probl...
Solve Uncertain Models	Automatic
Use Psi Functions to Defi...	False
Use Interactive Optimizat...	False
Number of Threads	0
Simulation Model	
Simulations to Run	1
Run Specific Simulation	
Trials per Simulation	1000
Interpreter	Automatic
Use Correlations	True
Value to Display	All Trials
Trial to Display	1
Number of Threads	0

When you next click the **green arrow** in the Task Pane, or choose **Simulate – Run Once** from the Analytic Solver Desktop Ribbon, all 10 (in this example) simulations will be run, automatically varying each simulation parameter over its range or list of values, and all the results will be collected and stored. (If you click the Simulate icon in Analytic Solver Desktop to activate *Interactive Simulation*, all 10 simulations will be run each time you make a change on the worksheet.)

Viewing Results of Multiple Simulations

After running multiple parameterized simulations, you can view the results on the worksheet, or in the Uncertain Function dialog, by selecting the simulation of interest from the **Sim #** dropdown. You can create reports and charts of results that span all or some of the simulations. We highly recommend that you read “An Airline Revenue Management Model” in the earlier chapter “Simulation and Risk Analysis,” which illustrates how this is done – creating a Trend Chart and a Box-Whisker Chart across all 10 simulations in this model.

Once you become familiar with multiple parameterized simulations, you’ll likely want to use them in nearly every simulation analysis you do. They’re a natural way to get further results from the effort you’ve put into a simulation model, and communicate those results to colleagues or clients.

Time Series Forecasting

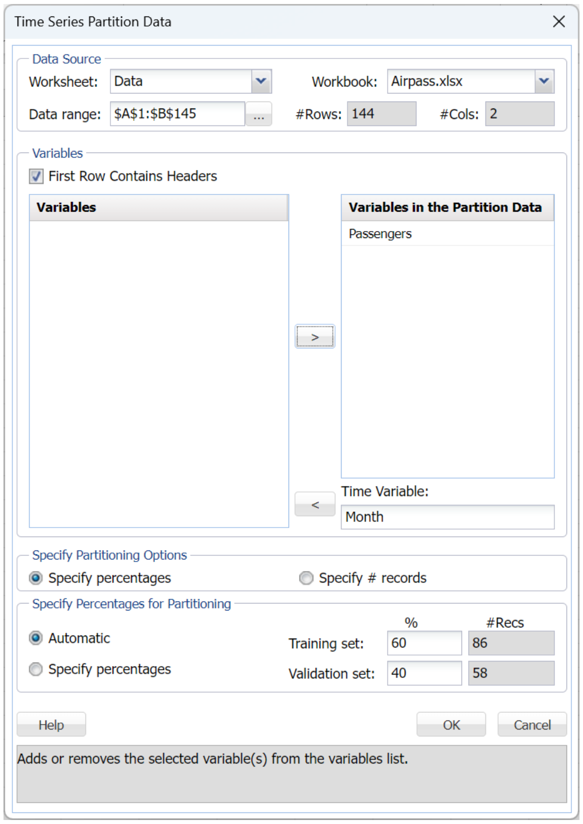
Starting with version 2014-R2, Analytic Solver Data Science includes the ability to forecast a future point in a time series in one of your spreadsheet formulas (without using the Score button on the Ribbon) using a PsiForecast() function in conjunction with a model created using ARIMA or one of our smoothing methods (Exponential, Double Exponential, Moving Average, or Holt Winters).

PsiForecast() is similar to the previous PSIForecastXXX functions supported in V2014, 2015, and 2016: it will compute future-looking forecasts based on the fitted model, using the provided new time series observations as initial points. The number of forecasts is determined by the number of selected cells in array-formula entry. The result of PsiForecast() can be deterministic, if the Simulate argument is FALSE, or non-deterministic, if the Simulate argument is TRUE– in which case the forecasts are adjusted with random normally distributed errors, defined by the forecasts’ statistics.

Open the Airpass.xlsx example dataset by clicking Help – Examples on the Data Science ribbon, then clicking Forecasting/Data Science Examples. This example dataset includes International Airline Passenger Information by month for years 1949 – 1960. Since the number of airline passengers increases during certain times of the year, for example Spring, Summer, and in the month of December, we can say that this dataset includes “seasonality”.

First, we will partition this dataset into two datasets: a training dataset and a validation dataset. We’ll use the training dataset to create the ARIMA model and then we’ll apply the model to the validation dataset to forecast six future data points, or one half year of data.

Click Partition in the Time Series section of the Data Science ribbon to open the Time Series Partition Data dialog. Select Passengers for the Variables in the Partition Data and Month for the Time Variable.



Time Series Partition Data

Data Source

Worksheet: Data

Workbook: Airpass.xlsx

Data range: $A$1:$B$145

#Rows: 144

#Cols: 2

Variables

☑ First Row Contains Headers

Variables in the Partition Data: Passengers

Time Variable: Month

Specify Partitioning Options

● Specify percentages ○ Specify # records

Specify Percentages for Partitioning

● Automatic

○ Specify percentages

	%	#Recs
Training set:	60	86
Validation set:	40	58

Adds or removes the selected variable(s) from the variables list.

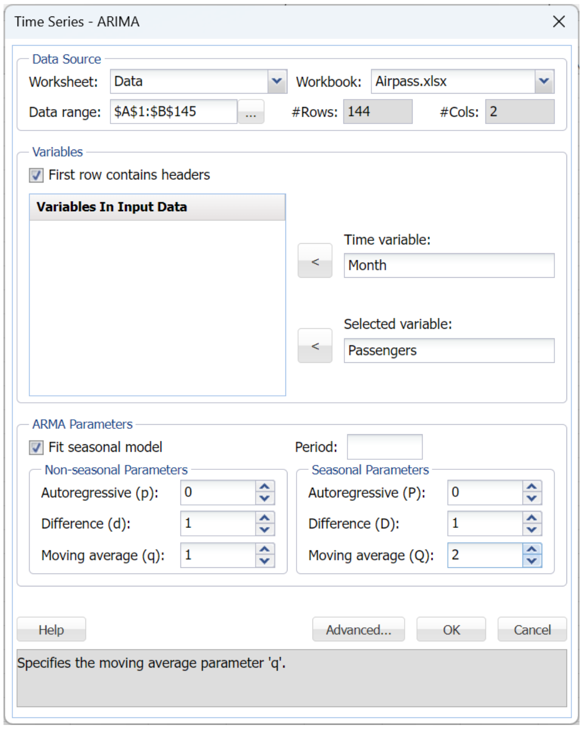
Click **OK** to accept the defaults for Specify Partitioning Options and Specify Percentages for Partitioning. Recall that when a time series dataset is partitioned, the dataset is partitioned sequentially. Therefore, 60% or the first 86 records, will be assigned to the training dataset and the remaining 40%, or 58 records, will be assigned to the validation dataset. (For more information on partitioning a time series dataset, see the previous chapter Exploring a Time Series Dataset.)

The TSPartition worksheet will be inserted into the Model tab of the Analytic Solver task pane under Transformations – Time Series Partition. Recall the steps needed to produce the forecast. Click ARIMA **--** ARIMA to open the ARIMA dialog. Month has been pre selected as the Time variable. Select Passengers as the Selected variable.

This example will use a SARIMA model, or Seasonal Autoregressive Integrated Moving Average model, to predict the next six datapoints in the dataset. (For more information on this type of time series model, please see the earlier chapter, “Exploring a Time Series Dataset.”) A seasonal ARIMA model requires 7 parameters, 3 nonseasonal (autoregressive (p), integrated (d), and moving average (q)), 3 seasonal (autoregressive (P), integrated (D), and moving average (Q)), and period. Each parameter must be a non-negative integer.

Selecting appropriate values for p, d, q, P, D, Q and period is beyond the scope of this User Guide. Consequently, this example will use a well documented SARIMA model with parameters p = 0, d = 1, q = 1, P = 0, D = 1, Q = 2 and period (P) = 12. Please refer to the classic time series analysis text **Time Series Analysis: Forecasting and Control** written by George Box and Gwilym Jenkins for more information on parameter selection.

Select Fit seasonal model and enter 12 for Period since it takes a full 12 months for the seasonal pattern to repeat. Set the Non-seasonal Parameters as Autoregressive (p) = 0, Difference (d) = 1, Moving Average (q) = 1 and the Seasonal Parameters as Autoregressive (P) = 0, Difference (D) = 1, and Moving Average (Q) = 2.


Time Series - ARIMA

Data Source
Worksheet: Data
Workbook: Airpass.xlsx
Data range: $A$1:$B$145
#Rows: 144
#Cols: 2

Variables
☑ First row contains headers
Variables In Input Data
Time variable: Month
Selected variable: Passengers

ARMA Parameters
☑ Fit seasonal model
Period:
Non-seasonal Parameters
Autoregressive (p): 0
Difference (d): 1
Moving average (q): 1
Seasonal Parameters
Autoregressive (P): 0
Difference (D): 1
Moving average (Q): 2

Specifies the moving average parameter 'q'.

Click **OK** to create the SARIMA model.

ARIMA_Output will be inserted into the Model tab of the XLMiner task pane under Reports – ARIMA. This output contains the Training Error Measures and Fitted Model Statistics. (For more information on this report, please see the chapter Exploring a Time Series Dataset within the Analytic Solver Data Science Reference Guide.) ARIMA_Stored contains the stored model parameters.

Now we’ll use this ARIMA model to predict new data points in the validation dataset using the PsiForecast() function. (*Note: The first forecasted point will be more accurate than the second, the second forecasted point more accurate than the third and so on.)* The PsiForecast() function will be interactive in the sense that if any of the input values (values passed in the 2nd argument) change, the forecast will be recomputed.

In later versions of desktop Excel, the PsiForecast() function returns a Dynamic Array. To use this function in the Cloud, you need only enter the Psi function in one cell as a normal function, i.e., not as a control array. The contents of the Dynamic Array will "spill" down the column. If a nonblank cell is "blocking" the contents of the Dynamic Array, the Psi Data Science function will return #SPILL until such time as the blockage is removed. In Excel versions that do not support Dynamic Arrays, PsiForecast() must be entered as an Excel array.

The PsiForecastARIMA function takes five arguments: Model, Input Data, Simulate, Num_forecasts, and Header. Select a blank cell on the Data worksheet and enter **=PsiForecast(**. If using a version of Excel that does not support Dynamic Arrays, select B146:B152 on the Data worksheet, then enter **=PsiForecast(.**

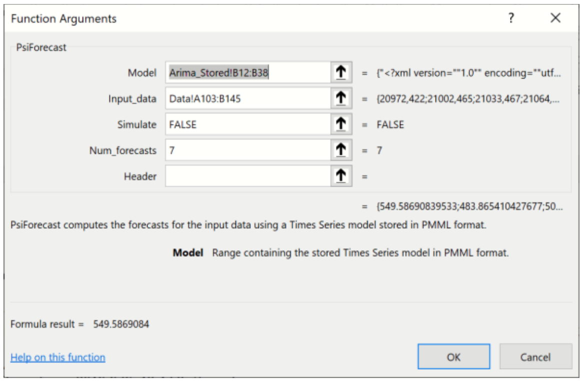
The first argument, *Model*, is the range of cells used by Analytic Solver Data Science to store the ARIMA model on the *ARIMA_Stored* worksheet. This data range will change as the forecast method changes. Select or enter **ARIMA_Stored!B12:B38,** for this argument.

The second argument, *Input_Data*, is the range containing the initial starting points from the validation data set. The minimum number of initial points that should be specified for a seasonal ARIMA model is the larger of p + d + s * (P + D) and q + s * Q. In this example, p + d + s * (P + D) is equal to 13 (0 + 1 + 12 * (0 + 1) and q + s * Q is equal to 13 (1 + 12 * 1), therefore the minimum number of initial starting points required is 13 (MAX (13, 13)). If you provide fewer than the minimum required number of starting points, PsiForecast() will return a column of zeros. (See the table below for the minimum number of initial starting points required by each Forecasting method included in Analytic Solver Data Science.) The maximum number of starting points is the number of points in the validation dataset. All points supplied in the second argument will be used in the forecast. Select or enter **Data!A1:B145,** for this argument.

Pass True or False for the third argument. Passing False will result in a static forecast that will only update if a cell passed in the 2nd argument is changed. If True is passed for this argument, a random error will be included in the forecasted points. See the Time Series Simulation example below for more information on passing True for this argument. In this case, Pass **False)** for this argument.

Your formula should now be the following: **=PsiForecast(ARIMA_Stored!B12:B38,Data!A1:B145, False**). If using a version of Excel that does not support Dynamic Arrays, press **CTRL** + **SHIFT** + **ENTER** to enter this formula as an array in all seven cells (B146:B152).

It’s also possible to enter this formula using the Insert Function dialog by clicking Formulas – Insert Function, select PSI Data Science for Category, then PsiForecastARIMA.



Model Arima_Stored!B12:B38 = {"<?xml version=""1.0"" encoding=""utf...

Input_data Data!A103:B145 = {20972,422;21002,465;21033,467;21064,...

Simulate FALSE = FALSE

Num_forecasts 7 = 7

Header =

= {549.58690839533;483.865410427677;50...

PsiForecast computes the forecasts for the input data using a Times Series model stored in PMML format.

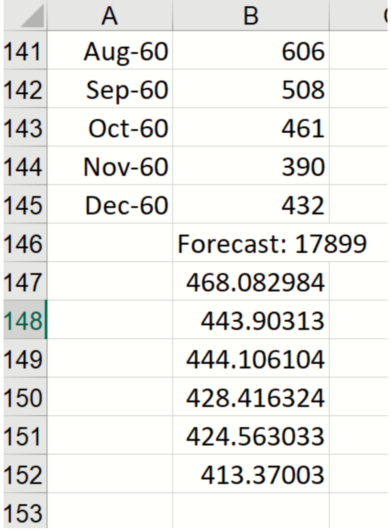
Model Range containing the stored Times Series model in PMML format.

Formula result = 549.5869084

The results from this function are displayed below.

Enter True for the Header argument to insert a heading above the forecasted values.

Notice that the formula is entered into cell B146 and the contents of the PsiForecast() Dynamic Array "spill" down into cells B147:B152.



	A	B
141	Aug-60	606
142	Sep-60	508
143	Oct-60	461
144	Nov-60	390
145	Dec-60	432
146		Forecast: 17899
147		468.082984
148		443.90313
149		444.106104
150		428.416324
151		424.563033
152		413.37003
153		

If any values change in the ranges ARIMA_Stored!B12:B38 or Data!B2:B145, the forecast will be recomputed; but if the input argument values stay the same, the PsiForecast() function will always return the same forecast values. As mentioned above, the first forecasted value in cell B146 is the most accurate predicted point. Accuracy declines as the number of forecasted points increases.

See the Frontline Solvers Reference Guide for specifications on PsiForecast().

Time Series Simulation

Analytic Solver Data Science includes the ability to perform a time series simulation, where future points in a time series are forecast on each Monte Carlo trial, using a model created via ARIMA or one of our smoothing methods (Exponential, Double Exponential, Moving Average, or Holt Winters).

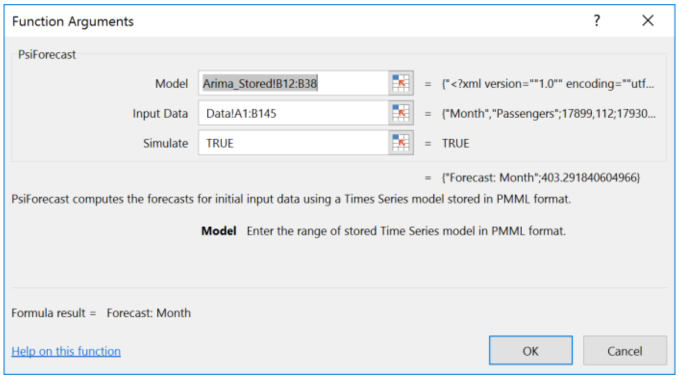
To run a time series simulation, we must pass “True” as the third argument to PsiForecast(). When the third argument is set to True, Analytic Solver will add a random (positive or negative) “epsilon” value to each forecasted point. Each time a simulation is run, 1000 trial “epsilon” values are generated using the PsiNormal distribution with parameters mean and standard deviation computed by the PsiForecast() function. You can view the output of this simulation in the same way as you would view "normal" simulation results in Analytic Solver Comprehensive, Analytic Solver Simulation or Analytic Solver Upgrade, simply by creating a PsiOutput() function and then double clicking the Output cell to view the Simulation Results dialog.

Select a blank cell, or Data!C146:C152 if using a version of Excel that does not support Dynamic Arrays, then click Formulas – Insert Function to display the Function Argument dialog.

As discussed previously, the first argument, ARIMA_Stored!B12:B38, is the range of cells used by Analytic Solver to store the ARIMA model on the *ARIMA_Stored* worksheet.

For the second argument the range containing the initial points in the series must be greater than the minimum number of initial points for a static forecast. For a seasonal ARIMA model when Simulate = True, the minimum number of initial points must be greater than Max((p + d + s * (P + D), (q + s * Q). In this example, p + d + s * (P + D) is equal to 13 (0 + 1 + 12 * (0 + 1) and q + s * Q is equal to 13 (1 + 12 * 1), therefore the minimum number of initial starting points required is 13 (Minimum #Initial Points > MAX (13, 13)).   However, when PsiForecastARIMA() is called with Simulate = True, it is recommended to add an additional number of datapoints, equal to the #Periods, to the minimum number required. In this instance the number of initial points will be 25: 13 (minimum # of points) + 12 (# of points for Period in the Time Series - ARIMA dialog).   If you provide fewer than the minimum required number of starting points (13 in this example) PsiForecastARIMA() will return #VALUE. (See the table below for the minimum number of initial starting points required by each forecast method in Analytic Solver.) All points supplied in the second argument will be used in the forecast. Select or enter **Data!A1:B145,** for this argument.

Passing TRUE for the third argument indicates to Analytic Solver Data Science that you plan to use this function call in a Monte Carlo simulation, so it should add a random epsilon value (different on each Monte Carlo trial) to each forecasted point.



Model Arima_Stored!B12:B38 = {"<?xml version=""1.0"" encoding=""utf...
Input Data Data!A1:B145 = {"Month","Passengers";17899,112;17930...
Simulate TRUE = TRUE
= {"Forecast: Month";403.291840604966}

PsiForecast computes the forecasts for initial input data using a Times Series model stored in PMML format.

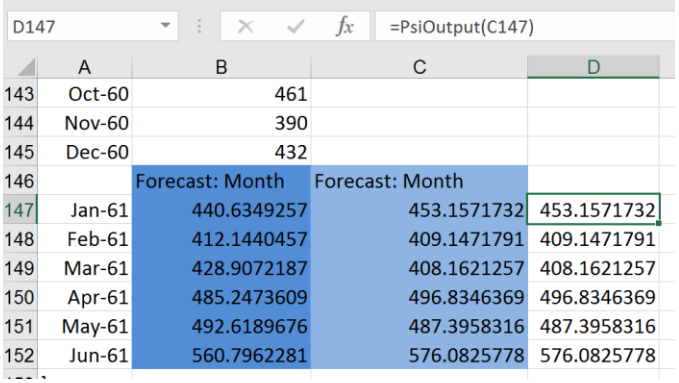
Model Enter the range of stored Time Series model in PMML format.

Formula result = Forecast: Month

In versions of Excel supporting Dynamic Arrays, this formula is entered into cell C146 and the contents of the PsiForecast() Dynamic Array "spills" down into cells C147:C152. If versions of Excel that do not support Dynamic Arrays, the formula must be entered as an Excel array.

To view the results of the simulation including frequency and sensitivity charts, statistics, and percentiles for the full range of trial values, we must first create an output cell. Select cell C147, then click Analytic Solver – Results – Referred Cell. Select cell D147 (or any blank cell on the spreadsheet) to enter the

PsiOutput formula. Copy this formula from cell D147 down to cell D152. Therefore D147 = PsiOutput(C147), D148 = PsiOutput(C148), and so on.

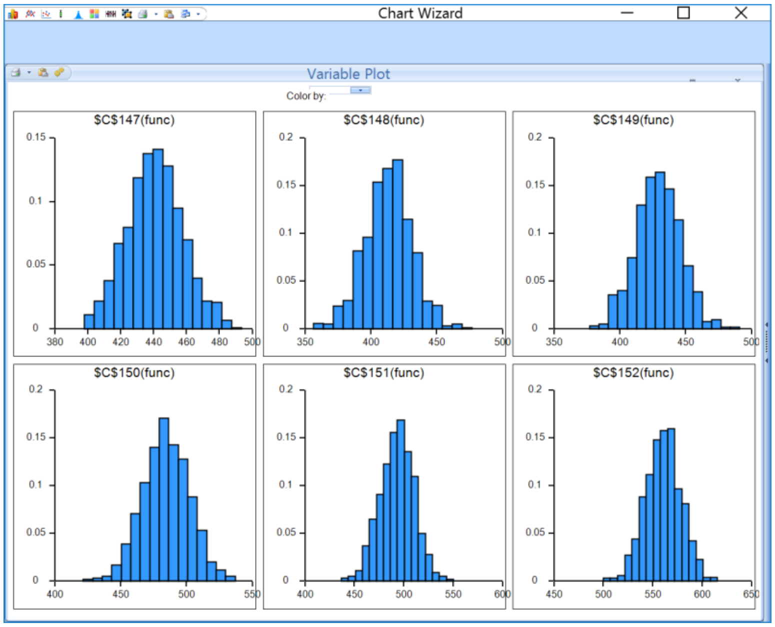


D147 =PsiOutput(C147)

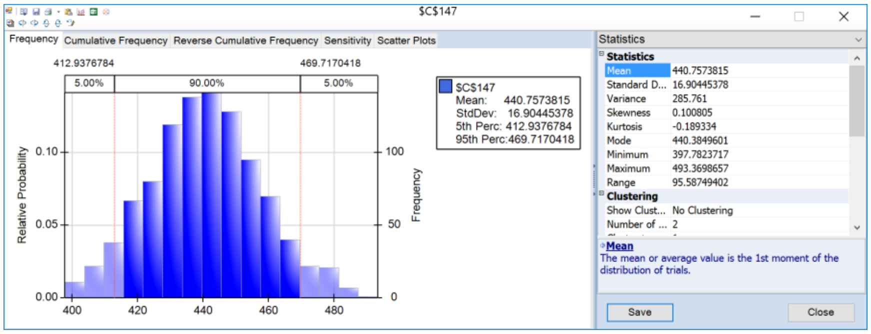
	A	B	C	D
143	Oct-60	461		
144	Nov-60	390		
145	Dec-60	432		
146		Forecast: Month	Forecast: Month	
147	Jan-61	440.6349257	453.1571732	453.1571732
148	Feb-61	412.1440457	409.1471791	409.1471791
149	Mar-61	428.9072187	408.1621257	408.1621257
150	Apr-61	485.2473609	496.8346369	496.8346369
151	May-61	492.6189676	487.3958316	487.3958316
152	Jun-61	560.7962281	576.0825778	576.0825778

Click the down arrow on the Simulate icon and select Run Once. Instantly, Analytic Solver will perform a simulation with 1,000 Monte Carlo simulation trials (the default number). Since this is the first time a simulation has been performed, the following dialog opens. Subsequent simulations will not produce this report. However, it is possible to reopen the individual frequency charts by double clicking each of the output cells (B147:B152).

Important Note: For Users who are familiar with simulation models in Analytic Solver Simulation, you'll notice that the time series simulation model that we just created now includes 6 uncertain functions, B147:B152, which are the cells containing our PsiForecast() functions. For more information on simulation with Analytic Solver, please see the Frontline Solvers User Guide chapter, "Examples: Simulation and Risk Analysis".



This dialog displays frequency charts for each of the six cells containing the forecasted data points. Double click the chart for cell C147 (top left) to open the Simulation Results dialog for the PsiForecast() function in cell C147. From here you can view frequency and sensitivity charts, statistics and percentiles for each forecasted point.

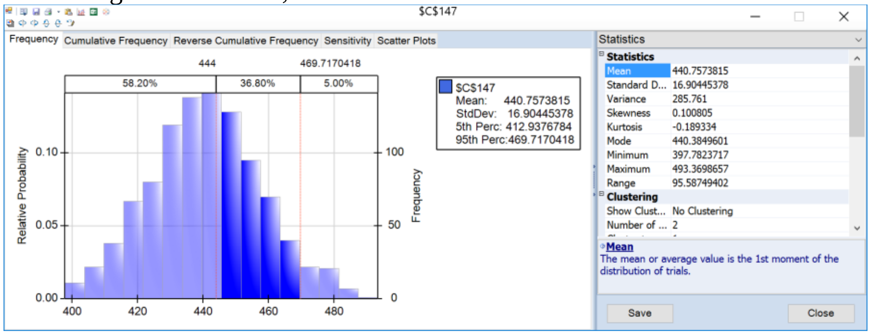


Statistics	
Mean	440.7573815
Standard D...	16.90445378
Variance	285.761
Skewness	0.100805
Kurtosis	-0.189334
Mode	440.3849601
Minimum	397.7823717
Maximum	493.3698657
Range	95.58749402
Clustering	
Show Clust...	No Clustering
Number of ...	2

Mean
The mean or average value is the 1st moment of the distribution of trials.

The frequency chart displays the distribution of all 1000 trial values for cell C147 with and observed mean 440.76 and standard deviation of 16.90 shown in the Chart Statistics. Select Simulate – Run Once a few more times (or click the green "play" button on the Solver Pane Model tab). Each time you do, another 1,000 Monte Carlo trials are run, and a slightly different mean will be displayed.

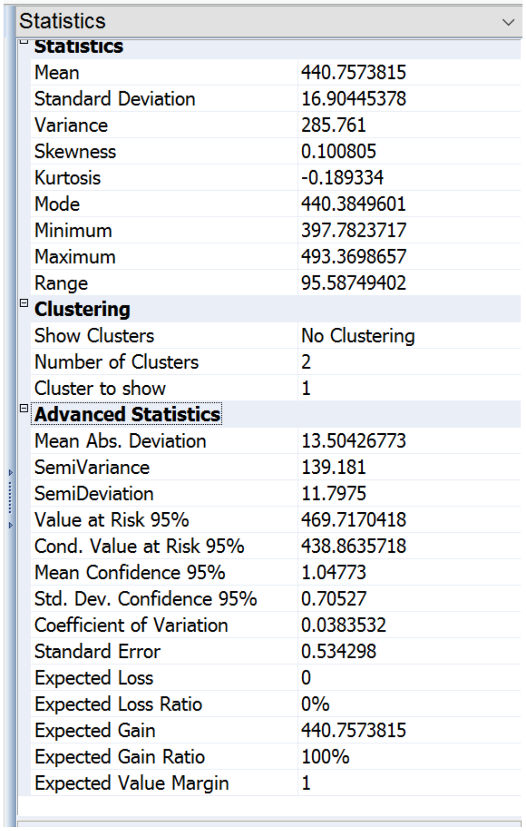
Enter 444 for the Lower Cutoff in the Chart Statistics section of the right panel, A vertical bar appears over the Frequency chart to display the frequency with which the forecasted value was greater than this value during the simulation. You can use this as an *estimate* of the probability that the actual value will be less than the forecasted value. In this case there was a 58.20% chance that the number of international airline passengers would be less than 444,000 in January 1961 and a 41.80% chance that the number of passengers would be greater than 444,000.



Statistics	
Mean	440.7573815
Standard D...	16.90445378
Variance	285.761
Skewness	0.100805
Kurtosis	-0.189334
Mode	440.3849601
Minimum	397.7823717
Maximum	493.3698657
Range	95.58749402
Clustering	
Show Clust...	No Clustering
Number of ...	2

Mean
The mean or average value is the 1st moment of the distribution of trials.

Looking to the right, you'll find the Statistics pane, which includes summary statistics for the full range of forecasted outcomes. We can see that the minimum forecasted value during this simulation was 397.78, and the maximum forecasted value was 493.37. Value at Risk 95% shows that 95% of the time, the number of international airline passengers was 469.72 or less in January 1961, in this simulation. The Conditional Value at Risk 95% value indicates that the average number of passengers we would have seen (up to the 95% percentile) was 438.86. For more information on Analytic Solver Platform's full range of features, see the Frontline Solvers User Guide chapter, "Examples: Simulation and Risk Analysis".



Statistics

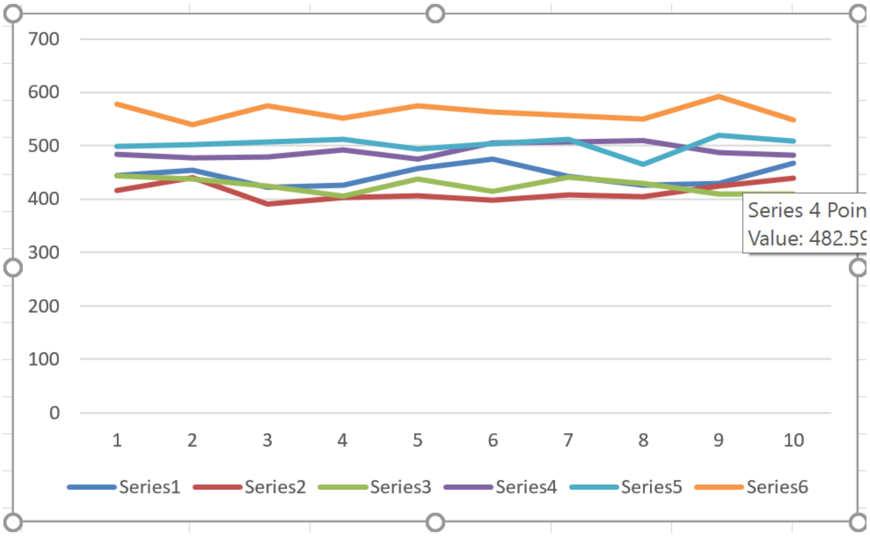
Statistics	
Mean	440.7573815
Standard Deviation	16.90445378
Variance	285.761
Skewness	0.100805
Kurtosis	-0.189334
Mode	440.3849601
Minimum	397.7823717
Maximum	493.3698657
Range	95.58749402
Clustering	
Show Clusters	No Clustering
Number of Clusters	2
Cluster to show	1
Advanced Statistics	
Mean Abs. Deviation	13.50426773
SemiVariance	139.181
SemiDeviation	11.7975
Value at Risk 95%	469.7170418
Cond. Value at Risk 95%	438.8635718
Mean Confidence 95%	1.04773
Std. Dev. Confidence 95%	0.70527
Coefficient of Variation	0.0383532
Standard Error	0.534298
Expected Loss	0
Expected Loss Ratio	0%
Expected Gain	440.7573815
Expected Gain Ratio	100%
Expected Value Margin	1

Select cells E147:E156 and then enter the formula, =PsiData(C147). The contents of this cell will "spill" down. (If using a version of Excel that does not support dynamic arrays, use CTRL + SHIFT + ENTER on the keyboard to enter this formula as an array.) Repeat the same steps to array enter "=PsiData(C148)" in cells F147:F156, "=PsiData(C149)" in cells G147:G156, "=PsiData(C150)" in cells H147:H156, "PsiData(C151)" in cells I147:I156, and "=PsiData(C152)" in cells J147:J156. Then click Simulate – Run Once to run a simulation.

The ten Excel cells in these columns will update with trial values for each of the PsiForecast() functions in column C. For example, cells E147:N147 will contain the first 10 trial values for the PsiForecast() function in cell C147, Cells E147:N147 will contain the first 10 trial values for cell C148 and so on. (For more information on the PsiData() function, please see the Excel Solvers Reference Guide chapter, "Psi Function Reference.")

Note: In Analytic Solver Cloud and in later versions of desktop Excel, the PsiData() function returns a Dynamic Array. To use this function in the Cloud, you need only enter the Psi function in one cell as a normal function, i.e., not as a control array. The contents of the Dynamic Array will "spill" down the column. If a nonblank cell is "blocking" the contents of the Dynamic Array, PsiData() will return #SPILL until such time as the blockage is removed. Use the optional *numTrials* argument to specify the number of trials returned in the Dynamic Array. If not present, all trials will be returned.

If we create an Excel chart of these values, you'll see a chart similar to the one below where each of Series1 through Series6 represents a different Monte Carlo trial. The random "epsilon" value added to each forecast value accounts for (all of) the variation among the lines. If the third argument were FALSE or omitted, all of the lines would overlap, assuming that the table or parameters and the starting values were not changing.



The remaining Forecasting methods can be used in the same way using PsiForecast() with information from their respective Stored Model sheets.

Forecasting Algorithm	**Stored Model Sheet**	**Minimum # of Initial Points when Simulate = False**	**Minimum # of Initial Points when Simulate = True**
Non- Seasonal ARIMA	ARIMA_Stored	Max(p + d, q)	Max(p + d, q)
Seasonal ARIMA	ARIMA_Stored	Max((p + d + s *(P + D), (q + s * Q)	1 + Max((p + d + s *(P + D), (q + s * Q)**
Exponential Smoothing	Expo_Stored	1	1
Double Exponential Smoothing	DoubleExpo_Stored	1	1
Moving Average Smoothing	MovingAvg_Stored	# of Intervals	# of Intervals
Holt Winters Smoothing	MulHoltWinters_Stored AddHoltWinters_Stored NoTrendHoltWinters_Stored	2 * #Periods	2 * #Periods

**Adding a number of data points equal to the Number of Periods (as shown on the Time Series – ARIMA dialog) to the Minimum # of Initial Points when Simulate = True is recommended when calling PsiForecast() with Simulate = True.

Excel 2016 Forecast Functions

Users of Analytic Solver Comprehensive, Analytic Solver Simulation and Analytic Solver Upgrade will notice two Psi Forecast functions: PsiForecastLinear() and PsiForecastETS(). These two functions were introduced to coincide with the new Excel 2016 Forecast functions: Forecast.Linear and Forecast.ETS.

PsiForecastLinear predicts future values for a time series data set (containing known or historical data) using linear regression. PsiForecastETS uses exponential smoothing to predict future values in a time series dataset with the option to either automatically detect seasonality in the data set or pass a seasonality period. Passing False as the last argument to either function will result in a static forecast. If True is passed for this argument, a random error will be included in the forecasted points.

Note: When comparing the results of the Psi functions with the Excel functions, you may see slightly different calculated forecast values depending on whether the Psi Interpreter or Excel Interpreter is selected on the Task Pane Platform tab. When the Psi Interpreter is used, a slightly different (but arguably better) methodology is used to compute these forecast functions. When the Excel Interpreter is selected, Microsoft Excel is used to calculate the worksheet, and thus Excel's methodology will be used.

To illustrate an example of how to use these two functions, we'll reuse the Airpass time series dataset. Click back to the Data tab within Airpass.xlsx. Recall that the Airpass time series dataset contains international airline passenger data from January 1949 to December 1960. Using this historical data, we will forecast the number of passengers in 1961.

PsiForecastLinear() performs linear regression of known_y values with known_x time variables to forecast the value at time, X. This function takes the following arguments.

`PsiForecastLinear(X, known_ys, known_xs[, simulate])`

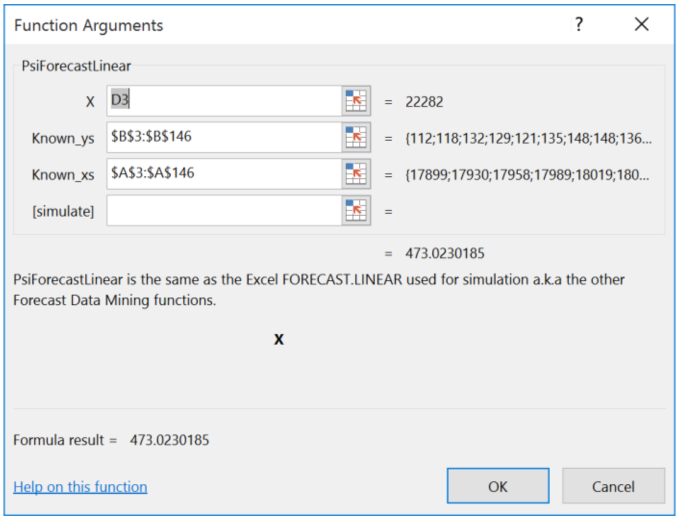
- `X`: The target date. A data point for which you want the predicted value. A data point may be date/time or numeric.
- `known_ys`: An Excel range containing the independent variables in the given dataset.
- `known_xs`: An Excel range containing the time variables in the given dataset.
- `simulate`: (Optional) Pass True or False for the third argument. Passing False (the default) will result in a static forecast that will only update if a cell passed in the known_ys argument is changed. If True is passed for this argument, a random error will be included in the forecasted points.

Click a blank cell on the worksheet, say cell E3, then in Analytic Solver Desktop click **Formulas – Insert Function** on the Ribbon to display the Function Argument dialog. In Analytic Solver Cloud, you'll need to type the formula directly into cell E3.

Select **Psi Data Science** form the *Category* drop-down menu, then scroll down to **PsiForcastLinear** and click **OK**. The *Function Arguments* dialog opens.

- `X`: Select cell Data!E3 for the `target` argument.
- `Known_ys`: Select or enter B3:B146. These are the independent y values. Be sure to anchor this range of cells (as shown in the screenshot below) so that we can easily copy the formula.
- `Known_ys`: Select or type A3:A146 for the third argument, `known_xs.` Again, anchoring this Excel range now allows easy copying and pasting later.

- `Simulate`: Pass **False** for this argument. See the Time Series Simulation example below for more information on passing True for this argument.



Function Arguments

PsiForecastLinear

X: D3 = 22282

Known_ys: $B$3:$B$146 = {112;118;132;129;121;135;148;148;136...

Known_xs: $A$3:$A$146 = {17899;17930;17958;17989;18019;180...

[simulate]: =

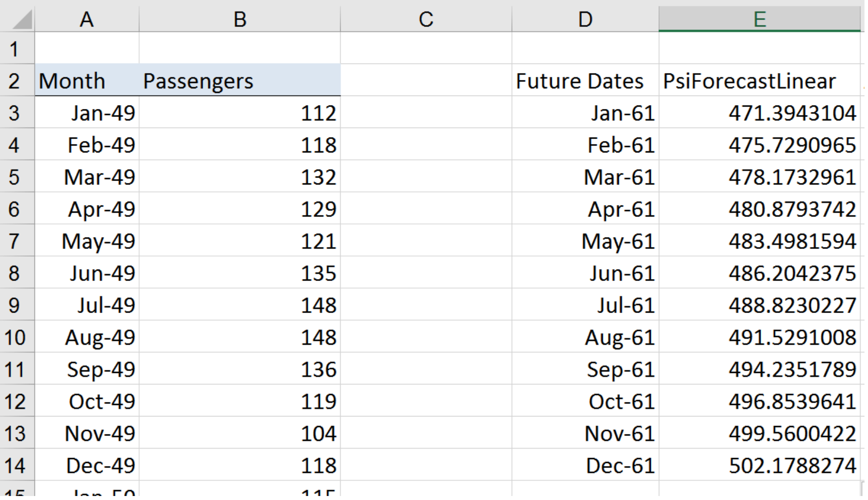
= 473.0230185

PsiForecastLinear is the same as the Excel FORECAST.LINEAR used for simulation a.k.a the other Forecast Data Mining functions.

x

Formula result = 473.0230185

Click **OK** to enter the formula into the Excel cell, then copy and paste this formula into cells E4:E14.



	A	B	C	D	E
1					
2	Month	Passengers		Future Dates	PsiForecastLinear
3	Jan-49	112		Jan-61	471.3943104
4	Feb-49	118		Feb-61	475.7290965
5	Mar-49	132		Mar-61	478.1732961
6	Apr-49	129		Apr-61	480.8793742
7	May-49	121		May-61	483.4981594
8	Jun-49	135		Jun-61	486.2042375
9	Jul-49	148		Jul-61	488.8230227
10	Aug-49	148		Aug-61	491.5291008
11	Sep-49	136		Sep-61	494.2351789
12	Oct-49	119		Oct-61	496.8539641
13	Nov-49	104		Nov-61	499.5600422
14	Dec-49	118		Dec-61	502.1788274

If any values change in the ranges Data!A3:A146 or Data!B3:B146, the forecast will be recomputed; however, if the input argument values remain the same, PsiForecastLinear() will always return the same forecast values. As mentioned above, the first forecasted value in cell E3 is the most accurate predicted point. Accuracy declines as the number of forecasted points increases.

We can create a similar forecast using PsiForecastETS(). PsiForecastETS() performs Exponential Triple Smoothing on known values to forecast future values at time, target. The fourth argument, seasonality, is optional. If not given, seasonality is automatically detected from the the data. If given, automatic seasonality detection is turned off, and the given seasonality period is enforced. This function takes the following arguments.

```
PsiForecastETS(target_date, values, timeline[,
seasonality][, data_completion][, aggregation][,
simulate])
```

- `target_date`: A data point for which you want the predicted value. A data point may be date/time or numeric. If a target date is given that appears before the start of the timeline in the dataset, PsiForecastETS will return #NUM (i.e., if 1940 is passed for X).
- `values`: An Excel range containing the historical values in the given data set.
- `timeline`: An Excel range containing the time variables in the given data set. Both functions require the historical data to be structured using a constant interval between data points. Since the Airpass data set presents monthly passenger data, the forecast must also predict the number of passengers by month.
- `seasonality`: (Optional) This argument indicates the length of the seasonal pattern. The following values are accepted as valid inputs. All other values will return a #NUM error.

  0: Signifies no seasonality exists in the data. The result is a linear prediction.

  1: (Default Value) Triggers Solver to automatically detect seasonality within the data.

  1 < N < 8,760: Positive integer values greater than 1 but less than 8,760, will can be entered as the seasonality period.
- `data_completion`: (Optional) This argument specifies how to handle missing values. The default value of 1 replaces missing values by interpolation. If 0 is passed, missing values will be replaced with 0's.
- `aggregation`: (Optional) PsiForecastETS can aggregate multiple points with the same time stamp. Pass an integer value from 0 to 6 to indicate which method should be used.

  0: (Default) Average

  1: SUM

  2: COUNT

  3: COUNTA

  4: MIN

  5: MAX

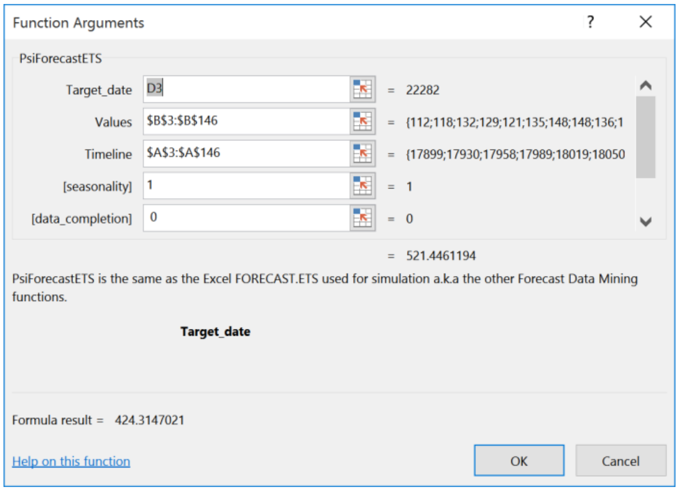
  6: MEDIAN
- `simulate`: (Optional) Pass True or False for the third argument. Passing False (the default) will result is a static forecast that will only update if a cell passed in the second argument is changed. If True is passed for this argument, a random error will be added to each forecasted point.

On the Excel ribbon, click **Formulas – Insert Function** to display the *Function Argument* dialog.

Select **Psi Data Science** form the *Category* drop-down menu, then scroll down to **PsiForcastETS** and click **OK**.

- `Target_date`: Select cell Data!E3.
- `Values`: Select or enter B3:B146. These are the independent y values. Anchoring the cell range (as shown in the screenshot below) will allow us to easily cut and paste the formula later.

- `Timeline`: Enter the dates January 1961 – December 1961 in cells D3:D11 for timeline. These are our `target` dates or the dates that we want to predict.   These are our time variables. Anchoring the cell range, as shown in the screenshot below, will allow us to easily cut and paste the formula.
- `Seasonality`: Enter 1 for this argument to automatically detect the seasonality within the data.
- `data_completion`: Enter 0 for this argument to substitue any empty value with 0.
- `aggregation`: Enter 0 for this argument to aggregate multiple points with the same date using AVERAGE.
- `Simulate`: Passing TRUE for the third argument indicates to XLMiner that you plan to use this function call in a Monte Carlo simulation. Therefore, a random epsilon value (different on each Monte Carlo trial) will be added to each forecasted point.



Target_date D3 = 22282

Values $B$3:$B$146 = {112;118;132;129;121;135;148;148;136;1

Timeline $A$3:$A$146 = {17899;17930;17958;17989;18019;18050

[seasonality] 1 = 1

[data_completion] 0 = 0

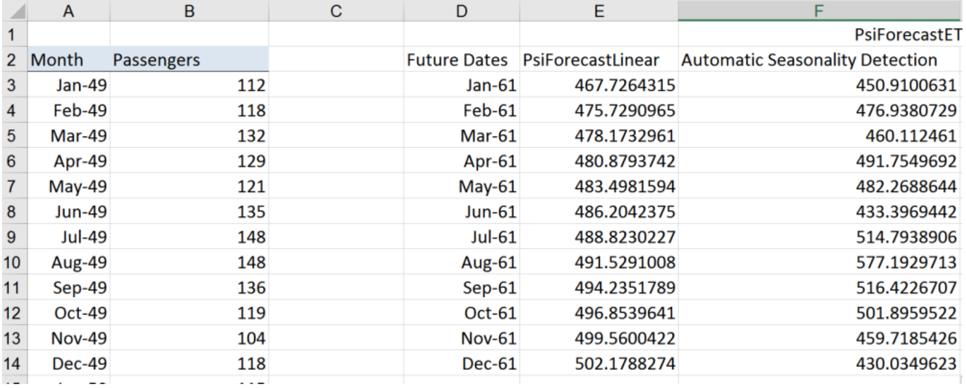
= 521.4461194

PsiForecastETS is the same as the Excel FORECAST.ETS used for simulation a.k.a the other Forecast Data Mining functions.

Target_date

Formula result = 424.3147021

Click **OK** to enter the formula into the Excel cell. This formula can easily be copied to cells E4:E14.



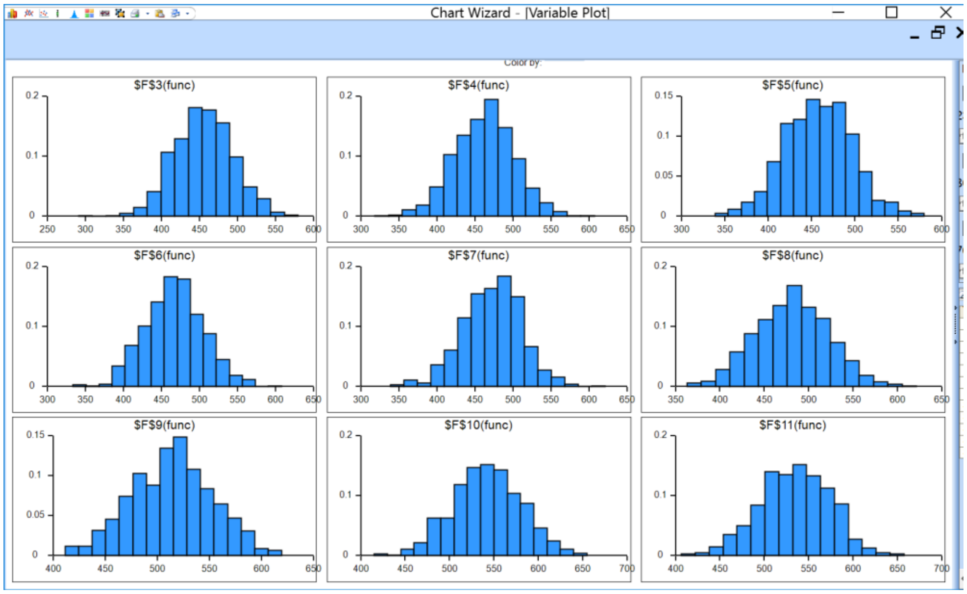
	A	B	C	D	E	F
1						PsiForecastET
2	Month	Passengers		Future Dates	PsiForecastLinear	Automatic Seasonality Detection
3	Jan-49	112		Jan-61	467.7264315	450.9100631
4	Feb-49	118		Feb-61	475.7290965	476.9380729
5	Mar-49	132		Mar-61	478.1732961	460.112461
6	Apr-49	129		Apr-61	480.8793742	491.7549692
7	May-49	121		May-61	483.4981594	482.2688644
8	Jun-49	135		Jun-61	486.2042375	433.3969442
9	Jul-49	148		Jul-61	488.8230227	514.7938906
10	Aug-49	148		Aug-61	491.5291008	577.1929713
11	Sep-49	136		Sep-61	494.2351789	516.4226707
12	Oct-49	119		Oct-61	496.8539641	501.8959522
13	Nov-49	104		Nov-61	499.5600422	459.7185426
14	Dec-49	118		Dec-61	502.1788274	430.0349623

As with the PsiForecast Data Science functions, to view the results of the simulation, including frequency and sensitivity charts, statistics, and percentiles for the full range of trial values, we must first create an output cell. Select cell F3, then click **Analytic Solver Platform – Results – Referred Cell** on the Excel ribbon. Select cell F17 (or any blank cell on the spreadsheet) to enter the

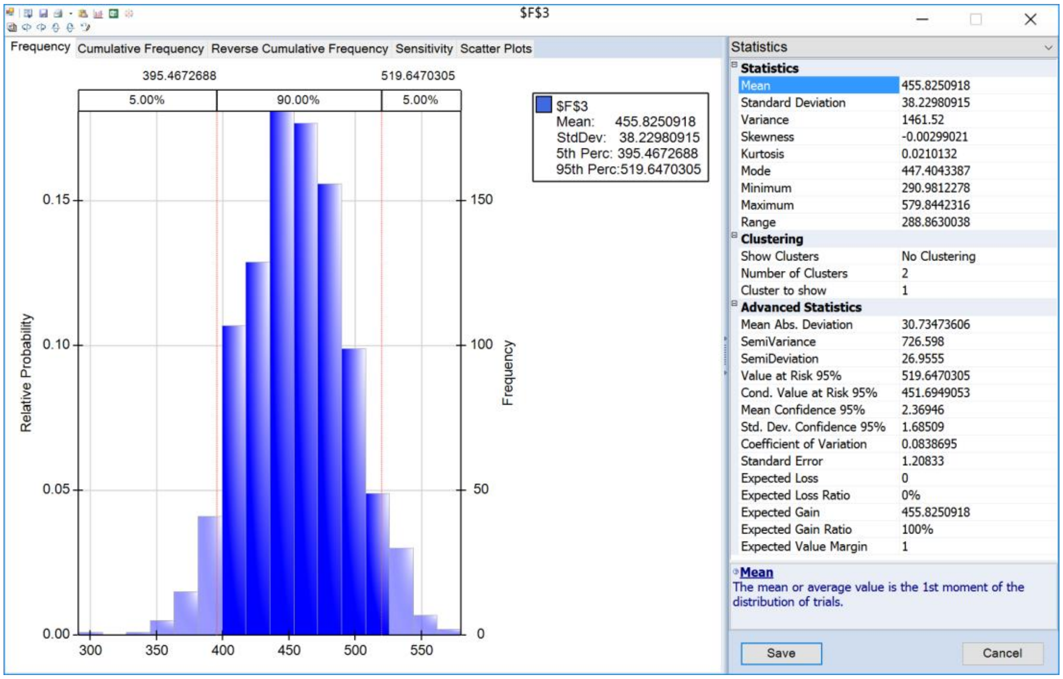
PsiOutput formula. Copy this formula from cell F17 down to cell F28. Therefore F17 = PsiOutput(F3), F18 = PsiOutput(F4), and so on.

Click the down arrow on the Simulate icon on the Analytic Solver Platform ribbon and select Run Once. Instantly, Analytic Solver Simulation performs a simulation with 1,000 Monte Carlo simulation trials (the default number). Since this is the first time a simulation has been performed, the following dialog opens. Subsequent simulations will not produce this report. However, it is possible to reopen the individual frequency charts by double-clicking each of the output cells (F17:F28).

Important Note: For users who are familiar with simulation models in Analytic Solver, you will notice that the time series simulation model that we just created now includes 12 uncertain functions (F17:F28) which are the cells containing our PsiForecastETS() functions.



The dialog above displays frequency charts for the first nine, of the total 12, uncertain function cells containing the forecasted data points. Double-click the chart for cell F3 (top-left) to open the *Simulation Results* dialog displaying PsiForecastETS() function in cell F3. From here, you can view frequency and sensitivity charts, statistics and percentiles for each forecasted point.



$F$3
Mean: 455.8250918
StdDev: 38.22980915
5th Perc: 395.4672688
95th Perc:519.6470305

Statistics	
Mean	455.8250918
Standard Deviation	38.22980915
Variance	1461.52
Skewness	-0.00299021
Kurtosis	0.0210132
Mode	447.4043387
Minimum	290.9812278
Maximum	579.8442316
Range	288.8630038
Clustering	
Show Clusters	No Clustering
Number of Clusters	2
Cluster to show	1
Advanced Statistics	
Mean Abs. Deviation	30.73473606
SemiVariance	726.598
SemiDeviation	26.9555
Value at Risk 95%	519.6470305
Cond. Value at Risk 95%	451.6949053
Mean Confidence 95%	2.36946
Std. Dev. Confidence 95%	1.68509
Coefficient of Variation	0.0838695
Standard Error	1.20833
Expected Loss	0
Expected Loss Ratio	0%
Expected Gain	455.8250918
Expected Gain Ratio	100%
Expected Value Margin	1

Mean
The mean or average value is the 1st moment of the distribution of trials.

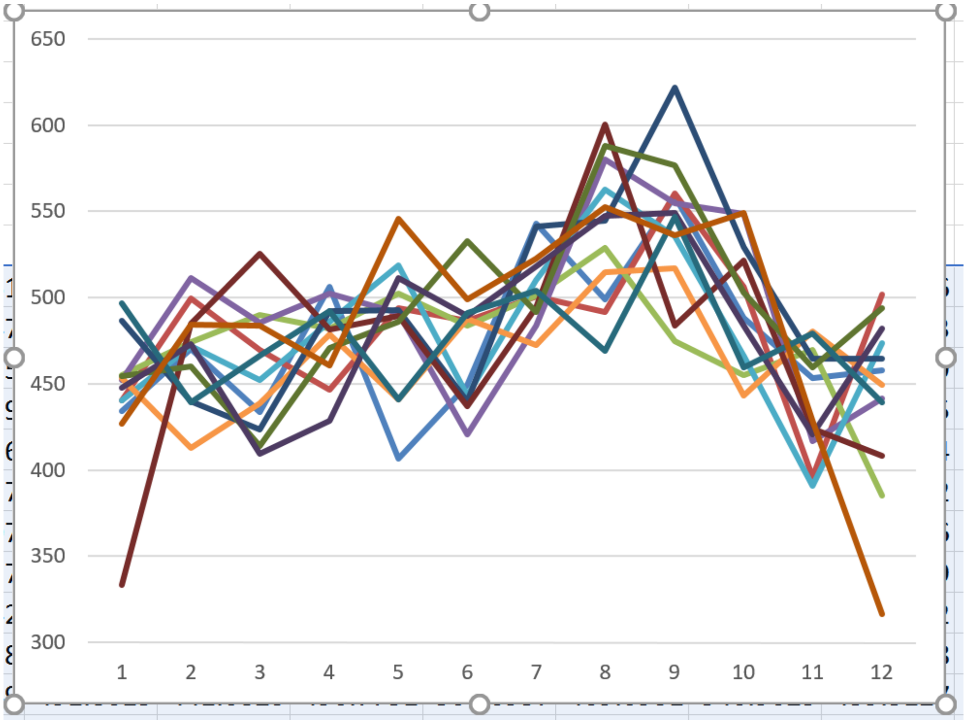
The frequency chart displays the distribution of all 1000 trial values for cell F3 with an observed mean 455.825 and standard deviation of 38.23 shown in the Chart Statistics.

On the right of the dialog, you'll find the Statistics pane, which includes summary statistics for the full range of forecasted outcomes. We can see that the minimum forecasted value during this simulation was 290.98, and the maximum forecasted value was 579.84. In this simulation, Value at Risk 95% shows that 95% of the time, the number of international airline passengers was 519.65 or less in January 1961. The Conditional Value at Risk 95% value indicates that the average number of passengers we would have seen (up to the 95% percentile) was 451.69. For more information on Analytic Solver Platform's full range of features, see the chapter, "Examples: Simulation and Risk Analysis".

As in the previous section, we will graph the first few trial values (out of a total of 1000) for each forecasted point. Select cells G17:G28, then enter the formula, =PsiData(F17). The contents of this cell will "spill" down. (If using a version of Excel that does not support dynamic arrays, use CTRL + SHIFT + ENTER on the keyboard to enter this formula as an array.) Repeat the same steps to enter "=PsiData(F18)" in cells H17:H28, "=PsiData(F19)" in cells I17:I28, and so one. On the Analytic Solver Platform ribbon, click **Simulate – Run Once** to run a simulation.

The 12 Excel cells in these columns will update with trial values for each of the PsiForecastETS() functions in cells F3:F14. Cells G17:G28 will contain the first 12 trial values for the PsiForecastETS() function in cell F3, cells G17:G28 will contain the first 12 trial values for cell F4, and so on. (For more information on the PsiData() function, please see the Excel Solvers Reference Guide chapter, "Psi Function Reference.")

If we create an Excel chart of these values, you will see a chart similar to the one below where each of Series 1 through Series 12 represents a different Monte Carlo trial. The random "epsilon" value added to each forecast value accounts for (all of) the variation among the lines. If the third argument were FALSE or omitted, all of the lines would overlap, assuming that the table or parameters and the starting values were not changing.



Getting Results: Stochastic Optimization

Introduction

This chapter explains how to obtain and interpret results from stochastic optimization (optimization of models with uncertainty) in Analytic Solver Comprehensive or Analytic Solver Simulation with Analytic Solver Optimization. We'll discuss **what can go wrong**, and what to do about it, and also **how to get more** than a single optimal solution from your model.

Analytic Solver Comprensive has powerful algorithms for automatically transforming and solving optimization models with uncertainty. But since a stochastic optimization model includes **decision variables**, **uncertain variables**, and **constraints** and an **objective** that may depend on both, there are "more things that can go wrong." Indeed, everything you've read in the previous two chapters, "Getting Results: Optimization" and "Getting Results: Simulation" will apply to stochastic optimization – plus there are a variety of special result messages and error messages that apply *only* to stochastic optimization models.

The good news is that everything you've learned so far about conventional optimization and Monte Carlo simulation *does* apply to stochastic optimization in Analytic Solver Comprehensive. Unlike most other software packages, Analytic Solver Comprehensive has one consistent user interface, set of terms, and model elements for all types of simulation and optimization models.

In the following sections, we'll focus on immediate actions you can take when you get an unexpected result – but if you read the chapter "Mastering Stochastic Optimization Concepts," you'll learn far more about stochastic optimization models and solution methods, and better understand *why* the unexpected result appeared, and how to design your model to get the solutions you want.

What Can Go Wrong, and What to Do About It

When you click the **Optimize** button on the Ribbon, or the **green arrow** on the Task Pane to solve, you'll normally get one of these outcomes:

1. A solution that makes sense to you. This is normally accompanied by a Solver Result message in **green** at the bottom of the Task Pane. You can proceed to "When Things Go Right: Getting Further Results."
2. A Solver Result error message that you understand and can correct, in **red** at the bottom of the Task Pane. You can take corrective action.
3. A Solver Result error message that you *don't* understand, in **red** at the bottom of the Task Pane. You should **read the solution log** in the Output tab (in Analytic Solver Desktop), **click the error message** and read Help about the message.
4. A *solution* that you *don't* understand, or that seems wrong. Again before doing anything else, you should **read the solution log** in the

Output tab (in Analytic Solver Desktop), **click the error message** to display Help, **run available reports** as described below, and **read the section** below "When the Solution Seems Wrong."

5. Solving runs for a very long time, and you don't get a solution or a Solver Result message until you press ESC or click Pause/Stop. You should **read the section** below "When Solving Takes a Long Time."

In rare cases, you might find Excel shutting down or "locking up" (so nothing happens when you press and hold the ESC key for several seconds in Analytic Solver Desktop or the Pause button in Analytic Solver Cloud). In this case please contact Frontline Systems Technical Support at (775) 831-0300 x4 or support@solver.com. Some Solver Result error messages ask you to contact Technical Support. If you can send us your model, this will be very helpful.

But experience shows that 99% of all technical support cases involve "pilot error" by the user, and that 90% of all such cases could be easily resolved by reading online Help or the User Guide. So we hope you'll keep reading, and that you'll take these steps before calling technical support!

Review Messages in the Output Tab

If using Analytic Solver Desktop, your first step should be to **review the messages** in the **solution log** in the Task Pane Output tab. This is **especially important** when you're solving a stochastic optimization model, to ensure that you understand the solution method that Analytic Solver Comprehensive used, and how your model was transformed and solved.

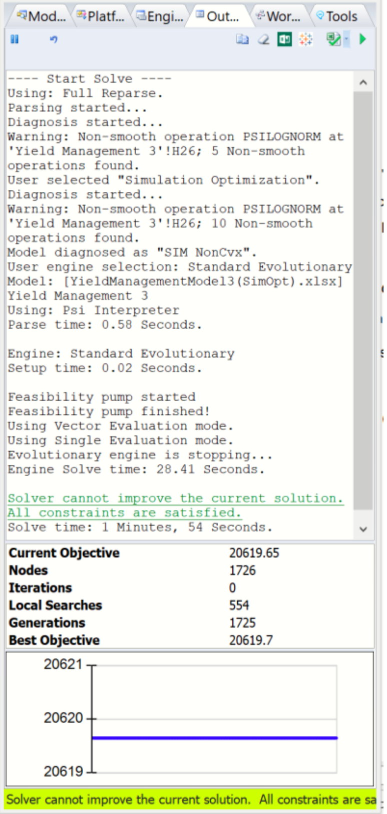
A Model Solved with Simulation Optimization

On the next page is an example of the Output tab at the solution of the example model **YieldManagementModel3.xlsx** (described more fully in the chapter "Examples: Simulation and Risk Analysis").

If you are using Analytic Solver Desktop, you can click the Copy icon to copy the contents of the solution log to the Windows Clipboard, where you can paste it into Microsoft Word, NotePad, or an **email message to Frontline Systems Technical Support**. Also, on the next page is the complete solution log from the Gas Company Chance model (before clicking the button to automatically improve the solution).

Note what happened during the solution process: Analytic Solver Comprehensive found that a Stochastic Transformation could not be applied to this model, because it has a *decision-dependent uncertainty* (the number of no-shows at cell C7 depends on the number of tickets sold, our decision variable, at C11). See the chapter "Mastering Stochastic Optimization Concepts" for more information on this important property of your model.

Analytic Solver diagnosed the model as "SIM NonCvx," and solved it using ***simulation optimization*** – the most general, but least scalable stochastic optimization method in Analytic Solver Comprehensive. It used the GRG Nonlinear Solver Engine (selected by the user) for optimization, and on every trial solution explored by the GRG Solver, it automatically performed a simulation.



Current Objective	20619.65
Nodes	1726
Iterations	0
Local Searches	554
Generations	1725
Best Objective	20619.7

Solver cannot improve the current solution. All constraints are sa

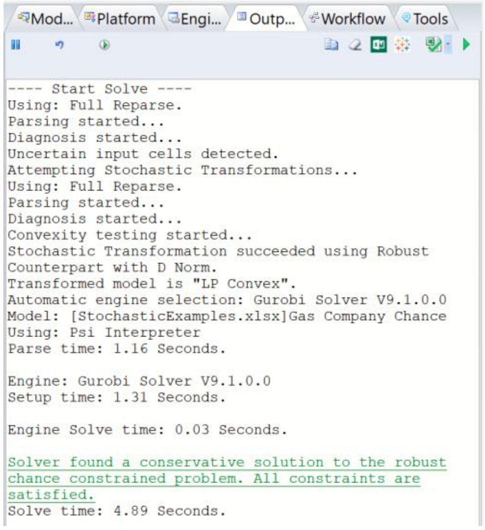
```
---- Start Solve ----
Using: Full Reparse.
Parsing started...
Diagnosis started...
Warning: Non-smooth operation PSILOGNORM at 'Yield Management
3'!H26; 5 Non-smooth operations found.
User selected "Simulation Optimization".
Diagnosis started...
Warning: Non-smooth operation PSILOGNORM at 'Yield Management
3'!H26; 10 Non-smooth operations found.
Model diagnosed as "SIM NonCvx".
User engine selection: Standard Evolutionary
Model: [YieldManagementModel3(SimOpt).xlsx]Yield Management 3
Using: Psi Interpreter
Parse time: 0.58 Seconds.
Engine: Standard Evolutionary
Setup time: 0.02 Seconds.
Feasibility pump started
Feasibility pump finished!
Using Vector Evaluation mode.
Using Single Evaluation mode.
Evolutionary engine is stopping...
Engine Solve time: 28.41 Seconds.
Solver cannot improve the current solution. All constraints are
satisfied.
Solve time: 1 Minutes, 54 Seconds.
```

Note: Information such as what is shown above is not available in the Cloud app since the RASON Server does not rely any information back to the app until the model is handed over to the Solver engine.

Let’s contrast what happened above with another example, where there are no decision-dependent uncertainties, and the underlying optimization model is a linear programming problem, but there is a *chance constraint*.

A Model Solved with Robust Optimization

Below is an example of the Output tab in Analytic Solver Desktop at the solution of the Gas Company Chance model in **StochasticExamples.xlsx** (described more fully in the chapter "Examples: Stochastic Optimization").



```
---- Start Solve ----
Using: Full Reparse.
Parsing started...
Diagnosis started...
Uncertain input cells detected.
Attempting Stochastic Transformations...
Using: Full Reparse.
Parsing started...
Diagnosis started...
Convexity testing started...
Stochastic Transformation succeeded using Robust
Counterpart with D Norm.
Transformed model is "LP Convex".
Automatic engine selection: Gurobi Solver V9.1.0.0
Model: [StochasticExamples.xlsx]Gas Company Chance
Using: Psi Interpreter
Parse time: 1.16 Seconds.

Engine: Gurobi Solver V9.1.0.0
Setup time: 1.31 Seconds.

Engine Solve time: 0.03 Seconds.

Solver found a conservative solution to the robust
chance constrained problem. All constraints are
satisfied.
Solve time: 4.89 Seconds.
```

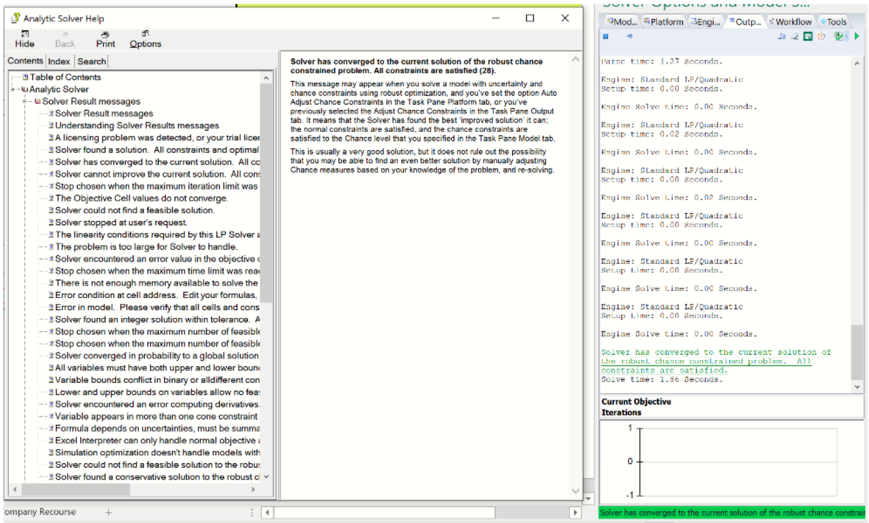
Again, if you click the Copy icon, the contents of the solution log will be copied to the Windows Clipboard, where you can paste it into Microsoft Word, NotePad, or an **email message to Frontline Systems Technical Support**. Below is the solution log from the Gas Company Chance model (before clicking the button to automatically improve the solution).

```
---- Start Solve ----
Using: Full Reparse.
Parsing started...
Diagnosis started...
Uncertain input cells detected.
Attempting Stochastic Transformations...
Using: Full Reparse.
Parsing started...
Diagnosis started...
Convexity testing started...
Stochastic Transformation succeeded using Robust Counterpart with D
Norm.
Transformed model is "LP Convex".
Automatic engine selection: Gurobi Solver V9.1.0.0
Model: [StochasticExamples.xlsx]Gas Company Chance
Using: Psi Interpreter
Parse time: 1.16 Seconds.
Engine: Gurobi Solver V9.1.0.0
Setup time: 1.31 Seconds.
Engine Solve time: 0.03 Seconds.
Solver found a conservative solution to the robust chance
constrained problem. All constraints are satisfied.
Solve time: 4.89 Seconds.
```

Click the Solver Result Message for Help

The Solver Result message in Analytic Solver Desktop is always **underlined** – it is a **hyperlink** to Help. If you aren't sure that you fully understand it, **click the link** to open online Help to a detailed discussion of the message.

Below is an example of Help that appears when you solve the Gas Company Chance model, and click on the Solver Result error message in green, "Solver found a conservative solution to the robust chance constrained problem. All constraints are satisfied.”


Analytic Solver Help
Hide
Back
Print
Options
Contents
Index
Search
Table of Contents
Analytic Solver
Solver Result messages
Solver Result messages
Understanding Solver Results messages
A licensing problem was detected, or your trial licer
Solver found a solution. All constraints and optimal
Solver has converged to the current solution. All cc
Solver cannot improve the current solution. All cons
Stop chosen when the maximum iteration limit was
The Objective Cell values do not converge.
Solver could not find a feasible solution.
Solver stopped at user's request.
The linearity conditions required by this LP Solver a
The problem is too large for Solver to handle.
Solver encountered an error value in the objective c
Stop chosen when the maximum time limit was reac
There is not enough memory available to solve the
Error condition at cell address. Edit your formulas,
Error in model. Please verify that all cells and cons
Solver found an integer solution within tolerance. A
Stop chosen when the maximum number of feasible
Stop chosen when the maximum number of feasible
Solver converged in probability to a global solution
All variables must have both upper and lower bounc
Variable bounds conflict in binary or alldifferent con
Lower and upper bounds on variables allow no fea:
Solver encountered an error computing derivatives.
Variable appears in more than one cone constraint
Formula depends on uncertainties, must be summa
Excel Interpreter can only handle normal objective a
Simulation optimization doesn't handle models with
Solver could not find a feasible solution to the robus
Solver found a conservative solution to the robust cl
ompany Recourse
Mod...
Platform
Engi...
Outp...
Workflow
Tools
Current Objective
Iterations
1
0
-1
Solver has converged to the current solution of the robust chance constrair

Solver has converged to the current solution of the robust chance constrained problem. All constraints are satisfied (28).

This message may appear when you solve a model with uncertainly and chance constraints using robust optimization, and you've set the option Auto Adjust Chance Constraints in the Task Pane Platform tab, or you've previously selected the Adjust Chance Constraints in the Task Pane Output tab. It means that the Solver has found the best 'improved solution' it can; the normal constraints are satisfied, and the chance constraints are satisfied to the Chance level that you specified in the Task Pane Model tab.

This is usually a very good solution, but it does not rule out the possibility that you may be able to find an even better solution by manually adjusting Chance measures based on your knowledge of the problem, and re-solving.

```
Parse time: 1.27 Seconds.

Engine: Standard LP/Quadratic
Setup time: 0.00 Seconds.

Engine Solve time: 0.00 Seconds.

Engine: Standard LP/Quadratic
Setup time: 0.02 Seconds.

Engine Solve time: 0.00 Seconds.

Engine: Standard LP/Quadratic
Setup time: 0.00 Seconds.

Engine Solve time: 0.02 Seconds.

Engine: Standard LP/Quadratic
Setup time: 0.00 Seconds.

Engine Solve time: 0.00 Seconds.

Engine: Standard LP/Quadratic
Setup time: 0.00 Seconds.

Engine Solve time: 0.00 Seconds.

Engine: Standard LP/Quadratic
Setup time: 0.00 Seconds.

Engine Solve time: 0.00 Seconds.

Solver has converged to the current solution of
the robust chance constrained problem.  All
constraints are satisfied.
Solve time: 1.86 Seconds.
```

Notice that the Help topic points out the option to **Auto Adjust Chance Constraints**. This is one key to better solutions when robust optimization is used, as in this example. Clicking the ▶ icon at the top of the Task Pane yields

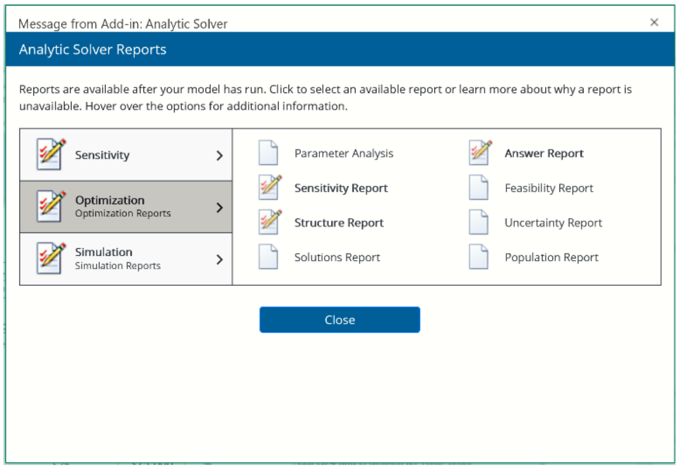
```
---- Start Solve ----
Model: [StochasticExamples.xlsx]Gas Company Chance
Using: Psi Interpreter
Using: Full Reparse.
Parsing started...
Parse time: 1.27 Seconds.
Engine: Standard LP/Quadratic
Setup time: 0.00 Seconds.
Engine Solve time: 0.00 Seconds.
Engine: Standard LP/Quadratic
Setup time: 0.02 Seconds.
Engine Solve time: 0.00 Seconds.
Engine: Standard LP/Quadratic
Setup time: 0.00 Seconds.
Engine Solve time: 0.02 Seconds.
Engine: Standard LP/Quadratic
Setup time: 0.00 Seconds.
Engine Solve time: 0.00 Seconds.
Engine: Standard LP/Quadratic
Setup time: 0.00 Seconds.
Engine Solve time: 0.00 Seconds.
Engine: Standard LP/Quadratic
Setup time: 0.00 Seconds.
Engine Solve time: 0.00 Seconds.
Solver has converged to the current solution of the robust chance
constrained problem. All constraints are satisfied.
Solve time: 1.86 Seconds.
```

This was a *successful* solution – but in cases where you have a Solver Result *error message* that you don’t understand, or a *solution* that you don’t understand, the solution log can be even more helpful.

Note: If you are using Analytic Solver Cloud, set Operating Mode (Help – Operating Mode) to either Guided Mode or Auto Help Mode to find detailed information on the Solver result messages.

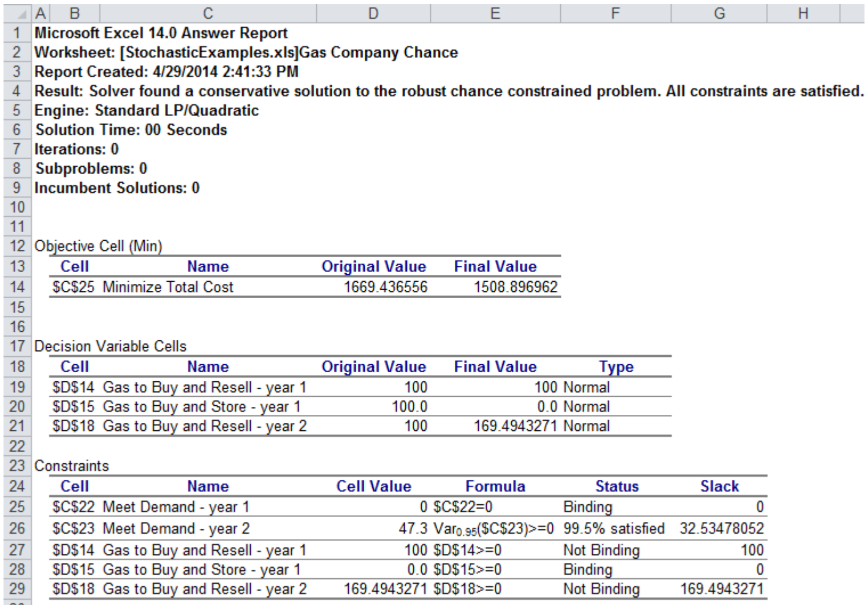
Choose Available Optimization Reports

As for conventional optimization, the available reports can help you understand the properties of a solution found by Analytic Solver. Just select **Reports – Optimization Reports** on the Ribbon – the available reports in the gallery are updated each time you solve.



Reports are available after your model has run. Click to select an available report or learn more about why a report is unavailable. Hover over the options for additional information.

Below is an Answer Report obtained for the Gas Company Chance model when we first received the Solver Result message "Solver found a conservative solution to the robust chance constrained problem. All constraints are satisfied." Note: This report is only generated in Analytic Solver Cloud when uncertain exceptions exist.



Microsoft Excel 14.0 Answer Report
Worksheet: [StochasticExamples.xls]Gas Company Chance
Report Created: 4/29/2014 2:41:33 PM
Result: Solver found a conservative solution to the robust chance constrained problem. All constraints are satisfied.
Engine: Standard LP/Quadratic
Solution Time: 00 Seconds
Iterations: 0
Subproblems: 0
Incumbent Solutions: 0

Objective Cell (Min)

Cell	Name	Original Value	Final Value
$C$25	Minimize Total Cost	1669.436556	1508.896962

Decision Variable Cells

Cell	Name	Original Value	Final Value	Type
$D$14	Gas to Buy and Resell - year 1	100	100	Normal
$D$15	Gas to Buy and Store - year 1	100.0	0.0	Normal
$D$18	Gas to Buy and Resell - year 2	100	169.4943271	Normal

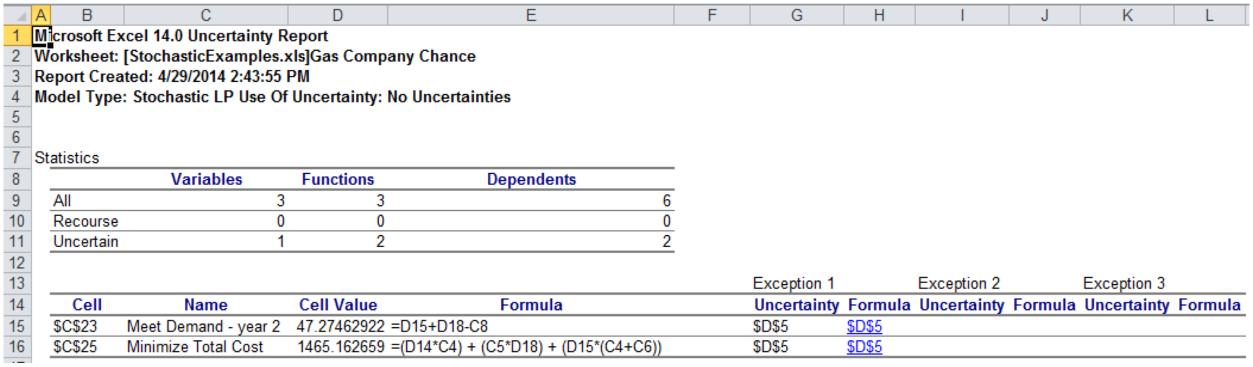
Constraints

Cell	Name	Cell Value	Formula	Status	Slack
$C$22	Meet Demand - year 1	0	$C$22=0	Binding	0
$C$23	Meet Demand - year 2	47.3	$\text{Var}_{0.95}$($C$23)>=0	99.5% satisfied	32.53478052
$D$14	Gas to Buy and Resell - year 1	100	$D$14>=0	Not Binding	100
$D$15	Gas to Buy and Store - year 1	0.0	$D$15>=0	Binding	0
$D$18	Gas to Buy and Resell - year 2	169.4943271	$D$18>=0	Not Binding	169.4943271

Note especially row 26, which reports the status of the chance constraint $\text{VaR}_{0.95}$($C$23) >=0: We asked for satisfaction of this constraint **95%** of the time, but at this solution the constraint is being satisfied **99.5%** of the time. This is a strong hint that the solution is conservative, and can be improved.

The Uncertainty Report for this model shows us the subset of the constraints and objective that depend on uncertainty. (*Note*: Since this model is automatically *transformed* to its robust counterpart, which has *no* uncertainty, you must set the Task Pane Platform tab Optimization Model group **Solve Uncertain Models** option to Simulation Optimization, then choose **Analyze Original Problem** to produce this report.)

Note: This report is not available in Analytic Solver Cloud.



Microsoft Excel 14.0 Uncertainty Report
Worksheet: [StochasticExamples.xls]Gas Company Chance
Report Created: 4/29/2014 2:43:55 PM
Model Type: Stochastic LP Use Of Uncertainty: No Uncertainties

Statistics

	Variables	Functions	Dependents
All	3	3	6
Recourse	0	0	0
Uncertain	1	2	2

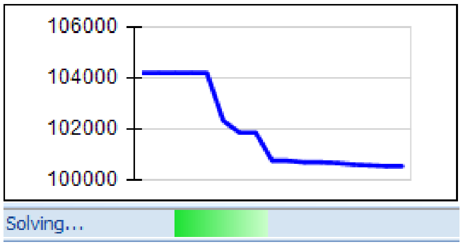
				Exception 1		Exception 2		Exception 3	
Cell	Name	Cell Value	Formula	Uncertainty	Formula	Uncertainty	Formula	Uncertainty	Formula
$C$23	Meet Demand - year 2	47.27462922	=D15+D18-C8	$D$5	$D$5				
$C$25	Minimize Total Cost	1465.162659	=(D14*C4) + (C5*D18) + (D15*(C4+C6))	$D$5	$D$5				

When Solving Takes a Long Time

When your model takes a long time to solve, the Task Pane Output tab can be helpful *during* the solution process – at a minimum, to reassure you that the Solver is still making progress, and has not “hung up.” If it is not already visible when you first start solving, the Output tab will **appear automatically** after a few seconds of solution time, as long as the Task Pane itself is visible.

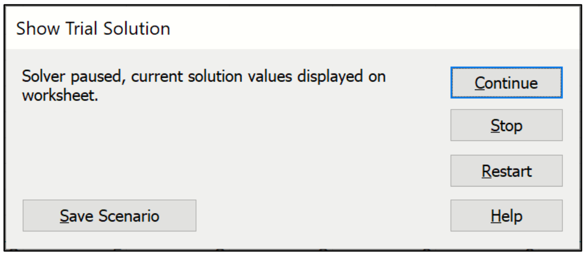
Running Chart of the Objective

The bottom part of the Output tab shows a running chart of the objective of the best solution found so far. This is most often useful for simulation optimization models, which may take some time to solve. Below is an example (staffing for a call center, not included among the installed example files):



Interrupting the Solution Process

You can interrupt the solution process at any time, by **pressing the ESC key** in Analytic Solver Desktop, or by **clicking the ⏸ button** in either Analytic Solver Desktop or the Cloud app, in the Task Pane Output tab. If your model is very large and you are using Analytic Solver Desktop, you might have to hold down the ESC key for a second or two. Since the input focus may not be on the Task Pane when you click with the mouse, you might have to click the ⏸ button twice. A dialog like the one on the next page will appear.



Solver paused, current solution values displayed on worksheet.

Click **Continue** if you want to continue solving; click **Stop** to cause the Solver to stop the solution process (this may take a few seconds) and display the message "Solver stopped at user's request." Clicking **Restart** may be useful with the Evolutionary Solver – see the Frontline Solvers Reference Guide.

Reasons Why Solving Takes a Long Time

There are several reasons why solving a stochastic optimization model might take a long time. To better understand why, read the chapter "Mastering Stochastic Optimization Concepts."

1. If your model doesn't meet the requirements for the stochastic programming or robust optimization solution methods, then the slowest method – **simulation optimization** – must be used. Since this requires an optimizer (such as the Evolutionary Solver) that will explore many Trial Solutions, and a *new simulation* must be performed for *every* Trial Solution, this method can take a lot of computing time.
2. If you are using Analytic Solver Desktop and you have the Platform tab Simulation group Interpreter option set to **Excel Interpreter**, Excel is used to recalculate the worksheet on each Monte Carlo trial. This is usually an **order of magnitude slower** than the PSI Interpreter, though it requires significantly less memory.
3. If **model diagnosis** or **problem setup** is slow (see timing messages in the Task Pane Output tab), you might be using array formulas, LOOKUP functions with large ranges as arguments, or long "chains" of formulas where each formula depends on an earlier formula, which can slow down the **PSI Interpreter**. See the analysis in the chapter "Getting Results: Optimization."
4. Your model might be well formulated, but very large, or it might require non-smooth functions, or many integer variables. A faster processor, multi-core processor, or more memory may help.

When the Solution Seems Wrong

When your optimization results seem wrong, before doing anything else, you should **read any messages** in the Output tab, and **click the message** to display Help as described above. We emphasize this, because very often when we are contacted in technical support, the user has not taken these basic steps.

Bear in mind that the uncertain function cells in your optimization model normally display the calculated value for the **last** Monte Carlo trial of the **last** simulation run. (All of Analytic Solver's solution methods – stochastic programming, robust optimization, and simulation optimization – perform at least one simulation). It can be very helpful to **cycle through the Monte Carlo trials**, and examine the values of uncertain function cells, as well as intermediate formula cells on which they depend. Just use the **left and right arrows** on the Ribbon to change the trial index. You may find that your model calculates a result you did not expect for some values of the uncertain variables.

Although software bugs are always possible, consider carefully the possibility that the simulation results are **correct for the model you've defined**, and that your expectation is wrong. We sometimes find in technical support that what your model actually says is different from what you intended.

When Things Go Right: Getting Further Results

When you *do* receive a solution that makes sense to you, and you have a Solver Result message in **green** at the bottom of the Task Pane for your stochastic optimization model, you have available **all the optimization and simulation reports and charts** that apply to your model and solution. Since a simulation is performed for every stochastic optimization model, you can **double-click** the uncertain function cells in your model (in Analytic Solver Desktop) or on the Model tab of the Solver Task Pane (in Desktop or the Cloud) to display the Uncertain Function dialog, and obtain many different views of the uncertainty in your model:

Dialog Tabs

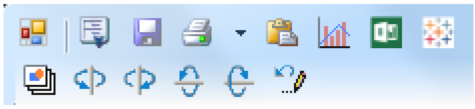
- The **Frequency**, **Cumulative Frequency**, and **Reverse Cumulative Frequency** tabs provide three different views of the full range of outcomes for an uncertain function. In Analytic Solver Desktop, you can click to display "crosshairs" with numerical values on these charts, or add Lower and Upper bounds to see the estimated probability of a profit or loss, for example.
- The Tornado chart on the **Sensitivity** tab quickly shows you which uncertain variables have the **greatest impact** on this uncertain function, across the full range of Monte Carlo trials. The **Scatter Plots** tab often reveals further insights about the behavior of this uncertain function versus each uncertain variable, or versus other uncertain functions.
- The **Statistics** and **Percentiles** tabs give you a full range of numerical results for this uncertain function, across the full range of Monte Carlo trials. Using PSI Statistics functions, you can place any of these values into a worksheet cell.

Panels and Toolbars

Analytic Solver Desktop includes rich charting abilities that are currently not available in Anlaytic Solver Cloud, but are coming soon. The suggestions below all pertain to Analytic Solver Desktop.

- Using the **right panel** of the dialog, you can change the chart type and color in both Analytic Solver Desktop and Cloud apps. In Analytic Solver Desktop you can additionally add titles, legends, gridlines, or chart markers. From the **Options dialog** (accessible from the Options button on the Analytic Solver Desktop Ribbon), on the **Charts** and **Markers** tabs, you can set default chart and marker settings for all of your charts.
- Using the **title toolbar icons** in Analytic Solver Desktop, you can save your settings, **print** the charts or numbers from any of the dialog tabs, or **copy** the charts or numbers to the Windows Clipboard, where they can be pasted into other applications. You can also **fit a distribution** to the simulation results or export your results to either Power BI or Tableau.

- In Analytic Solver Desktop, click the leftmost icon on the second row to display the **3-D toolbar** that lets you – with a single click – shift between 2-D and 3-D, **shrink** or **magnify** the chart, or **move** or **rotate** the chart to a different perspective.



Dimensional Modeling

Introduction

Dimensional Modeling is a fundamental new capability in Analytic Solver introducing new concepts such as **dimensions** and **cubes**, and provides all the tools you need to build and solve models using these concepts. If you're used to spreadsheet modeling, Dimensional Modeling will require some learning and practice. But when you want to construct *larger* models for common business situations where there are multiple products, projects or investments, multiple customer types, geographic regions, or time periods, multiple sources and destinations, dimensional modeling can turn a very complex, hard-to-maintain spreadsheet problem into a simple, well-structured, easily maintained and expandable model.

Note: Dimensional Modeling is not supported in RASON Decision Services.

If you've used **Pivot Tables** or **PowerPivot** in Excel, or if you've used a multidimensional database, data warehouse or business intelligence tool, the concepts behind Dimensional Modeling will probably be familiar. But these tools primarily deal with multidimensional *data*, generally reflecting the past. Dimensional Modeling makes it easy to construct multidimensional *models*, mostly composed of formulas, that you can use in planning for the future.

Unlike other tools, in Analytic Solver you can **use everything you know about Excel formulas** in Dimensional Modeling. For example, if cell A1 contains a number and cells B2 and C3 contains =PsiCube() functions, a formula like =A1+B2*C3 in a cell will yield a "cube result," multiplying and adding the appropriate values along several dimensions. You can also define B2 and C3 based on Pivot Tables containing input data, and easily obtain a Pivot Table that displays the results of like =A1+B2*C3.

Very often, a successful model is **expanded** to include more products, regions, time periods, etc. This is *much easier* to do with a Dimensional Model than with a traditional spreadsheet model. Formulas like =A1+B2*C3 need not change at all when you add elements (more cell or database values) to the existing dimensions. You can even add new *dimensions* to B2 and C3, or turn A1 from a single cell to a cube, and =A1+B2*C3 will automatically calculate results across the additional dimensions.

Dimensional Modeling in Optimization and Simulation

Dimensional Modeling can be used to create multidimensional "what if" models – a very powerful capability by itself. But you can also use it to create better structured, more compact and easily readable optimization and simulation models. For example, blocks of constraints that once required large ranges of repetitive formulas can now be entered in one or a few cells, drastically reducing the number of formulas. Similarly, in a simulation model, blocks of uncertain functions that once required large numbers of repetitive formula cells can be reduced to one or a few cells. You can define as many "structural dimensions" as you like, and use up to 8 of them at one time, in any PsiCube() function or calculated "cube intermediate result" in a formula.

Dimensional Modeling and Parametric Analysis

Dimensional Modeling also offers a powerful extension of Analytic Solver's capabilities for multiple parameterized optimization and simulation (described in the earlier chapters "Examples: Parameters and Sensitivity Analysis," "Getting Results: Optimization" and "Getting Results: Simulation"). It enables you to perform sensitivity analysis, or run many optimizations or simulations, varying parameters *across multiple dimensions* on each run, and capture all the results – optionally in Pivot Tables and Pivot Charts.

Without Dimensional Modeling, you define parameters for sensitivity analysis, optimization and simulation, using PsiSenParam(), PsiOptParam() and PsiSimParam() functions, respectively. These parameters function like simple "cubes" defined over just *one* dimension – the "run index" 1, 2, 3, etc. You run multiple calculations, optimizations or simulations, where all the parameters of a given type "move together" through their arrays of values.

With Dimensional Modeling, you can define as many "parametric dimensions" as you like, and create cubes of parameter values that are defined over *one or more* dimensions. You can then perform sensitivity analysis or run multiple optimizations or simulations, where any subset of your parametric dimensions are "active" (the others are "locked"), and get results for *all combinations* of the parameter values you've defined.

Just as you can **expand** a model by adding a structural dimension and generalize one or more cubes over the new dimension, without changing your formulas, you can add a parametric dimension, use it in one or more cubes, and expand your sensitivity analysis, optimization or simulation runs to include more scenarios, without changing any formulas.

Visualizing Dimensional Modeling Results

As mentioned earlier, you can define cubes based on Pivot Tables containing input data, and easily create new Pivot Tables that displays results of formulas like =A1+B2*C3. But creating a Pivot Table does require several steps, as well as "real estate" on your spreadsheet. If you want to inspect the results of intermediate calculations, you might need a separate table area for each "cube formula" cell in your model.

Analytic Solver makes this much easier. Just *double-click* on any cube formula to see a pop-up window that behaves like an "instant Pivot Table," allowing you to 'drag and drop' both structural and parametric dimensions across the columns and rows, and reorganize and display the data or calculated results. You'll see a number of examples of these pop-up windows in this chapter. For final results and reporting, you can use Pivot Tables and Charts.

Dimensions and Cubes

Let's define the concepts in Dimensional Modeling more formally. You can define two types of ***dimensions***, Structural and Parametric, and use them in calculation/sensitivity analysis, optimization and simulation models. You give a name (such as "Region") for the dimension, and you provide names/labels or numbers for the elements of the dimension – for example "North," "South," "East" and "West." Note that a dimension doesn't define the data itself – it defines a *structure*, relevant for your business situation, for the data.

Cubes are multi-dimensional arrays that hold the data. Other names used in data warehouse and business intelligence systems, are *measures* and/or *fact tables.*

A cube has an element (data value) for each combination of elements of the dimensions you specify when defining the cube. For example, you might create a cube named "Sales," defined over the "Regions" dimension, with four numeric values representing sales in the North, South, East and West regions. The structure of a cube is very much like a Pivot Table, and you can create a cube by placing a PsiPivotCube() function in a cell, referencing an existing Pivot Table.

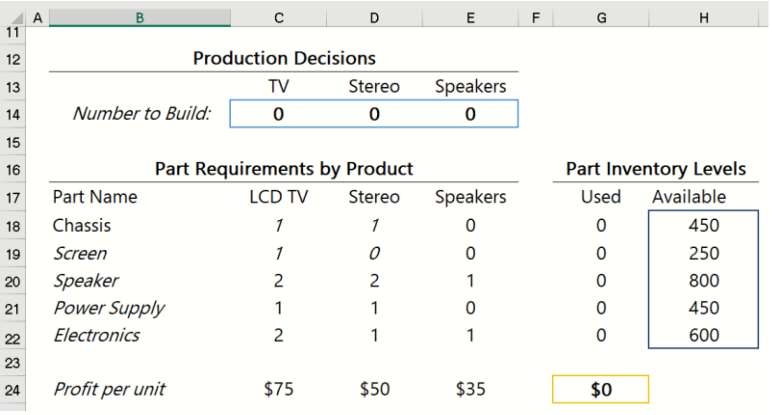
A single cube holds a single "measure" or type of value. For example, a person has many attributes such as height, weight, hair color, and eye color. You might define a dimension "Person" with elements such as "John", "Mary" and "Susan." But since a cube holds a single attribute or measure, you'd need four different cubes, each defined over the "Person" dimension, to hold the values of height, weight, hair color, and eye color.

There are three main steps involved in creating a Dimensional Model. Step 1 is to define at least one **dimension**, either parametric or structural. Step 2 is to define at least one (usually several) **cubes**. Step 3 is to define at least one **output** cell – for example, the objective cell in an optimization model – with a formula that depends on a cube. Often, you'll have outputs that *are* cubes and contain many – even thousands of – result values.

An Example Dimensional Model

This chapter will use the Product Mix example, introduced at the beginning of the chapter "Examples: Conventional Optimization," to illustrate Dimensional Modeling concepts, and the menu choices and Psi functions you can use to build a Dimensional Model step by step. The example is very simple, with just two structural dimensions – Products and Parts. This will make it easier to fully grasp the concepts and step-by-step operations, but since you can usually "lay out" two-dimensional data in regular spreadsheet columns and rows, Dimensional Modeling might seem like "overkill" for this simple example. The advantages of Dimensional Modeling will be apparent when you are modeling a more realistic business situation with many dimensions.

Click **Help – Example Models** on the Analytic Solver ribbon, to open the Frontline Example Models Overview dialog. Click the Optimization Examples button and scroll down to find and click **Structural Dimensions** from the list. When this workbook opens, click the *Product Mix Original* tab, which displays a conventional version of the Product Mix model.



	A	B	C	D	E	F	G	H
11								
12			Production Decisions					
13			TV	Stereo	Speakers			
14		*Number to Build:*	0	0	0			
15								
16			Part Requirements by Product				Part Inventory Levels	
17		Part Name	LCD TV	Stereo	Speakers		Used	Available
18		Chassis	*1*	*1*	0		0	450
19		*Screen*	*1*	*0*	0		0	250
20		*Speaker*	2	2	1		0	800
21		*Power Supply*	1	1	0		0	450
22		*Electronics*	2	1	1		0	600
23								
24		*Profit per unit*	$75	$50	$35		$0	

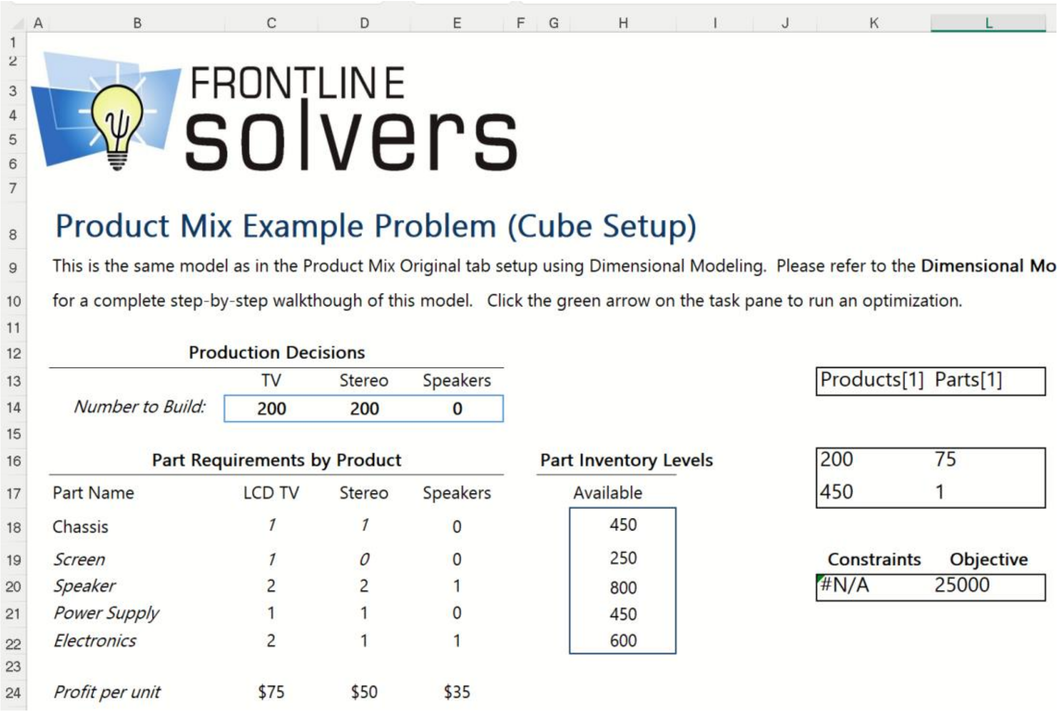
This example solves the problem of how many TVs, Stereos and Speakers a company should manufacture. It has three **decision variables** (C14:E14) that correspond to the number of TVs, Stereos and Speakers that the company will manufacture. The profit margins for each product are known. The gross margin (cell C24) for a TV is $75. The gross margin for stereos (cell D24) is $50 and the gross margin for speakers (cell E24) is $35.

Five different parts are used in the manufacturing process of these three products. The parts needed to build a TV are: 1 chassis, 1 picture tube, 2 speaker cones, 1 power supply, and 2 electronic kits. To build a stereo requires 1 chassis, 2 speaker cones, 1 power supply, and 1 electronic kit. Speakers require 1 speaker cone and 1 electronic kit.

The **constraints** are simply that the number of parts used to build each product (calculated in cells G18:G22) cannot exceed the inventory (found in cells H18:H22). A non-negativity bound has also been added to the variables (C14:E14 >= 0) as it would be impossible to manufacture a negative amount of products. Since the company wants to maximize profit, we ask Analytic Solver to maximize the value of the **objective** (cell G24), the total gross profit.

You can view the complete model in the Task Pane Model tab. **Solve the model** by clicking the green arrow in the Model tab. Analytic Solver will run an optimization to find a solution of 200 TVs (cell C14), 200 Stereos (cell E14) and 0 speakers (cell E14). This solution gives a final objective of $25,000 in cell G24.

Now let's inspect a model of this same problem setup using Dimensional Modeling: Click the *Dimensional Modeling* tab.



Product Mix Example Problem (Cube Setup)

This is the same model as in the Product Mix Original tab setup using Dimensional Modeling. Please refer to the **Dimensional Mo**

for a complete step-by-step walkthough of this model. Click the green arrow on the task pane to run an optimization.

Production Decisions

	TV	Stereo	Speakers
Number to Build:	200	200	0

Part Requirements by Product / **Part Inventory Levels**

Part Name	LCD TV	Stereo	Speakers	Available
Chassis	*1*	*1*	0	450
Screen	*1*	*0*	0	250
Speaker	2	2	1	800
Power Supply	1	1	0	450
Electronics	2	1	1	600
Profit per unit	$75	$50	$35	

Products[1]	Parts[1]

200	75
450	1

Constraints	Objective
#N/A	25000

Cells K13 and L13 define the Products and Parts dimensions. Cells K16:L17 define four cubes, using the PsiCube() function. Cells K20 and L20 contain cube formulas that calculate the constraints and objective, using the PsiReduce() function. Cells K25:K29 contain an Excel array formula{ =PsiOptData(K20)} to display the values of the left hand sides of the constraints on the spreadsheet;

but if you simply **double-click cell K17**, you'll see the same values in a pop-up window that functions like an "instant Pivot Table."

Notice that the **Task Pane Model tab** outline (not pictured here) lists the Dimensions and Cubes in the model, as well as the Variables, Constraints, Objective and other model elements. To keep things simple for users who aren't using Dimensional Modeling, these outline elements appear *only* when you have functions such as PsiDim() and PsiCube() in cells on your worksheet.

The **decision variables** are still in cells C14:E14. Try changing one of these cells to a new value such as 100. You'll see that the worksheet **recalculates** just like a normal Excel model, even though it is composed of dimensions and cubes.

Dimensional optimization models continue to use a separate cell for each decision variable – whereas they might have thousands of constraints in a single cell. Keeping decision variables in individual cells makes it easy to capture final solution values, save them in your workbook, and use Excel to recalculate the spreadsheet later. Cube formulas appear to Excel as ordinary formulas (with single values) using user-defined functions such as PsiCube() and PsiReduce(); the actual multidimensional calculations are performed "behind the scenes" by the PSI Interpreter.

Solve the model by clicking the green arrow in the Model tab, just as you did on the Original sheet. You'll see the same optimal solution appear in cells C14:E14.

The following sections will take you step by step through the process of building a Dimensional Model like the one above. You can use the Analytic Solver menu choices, grouped under the Model button, to define dimensions, cubes and results, or you can type functions such as PsiCube() and PsiReduce() directly into cells, without using the GUI at all.

Using Dimensional Modeling in Optimization

Keep in mind the following three steps when using Dimensional Modeling:

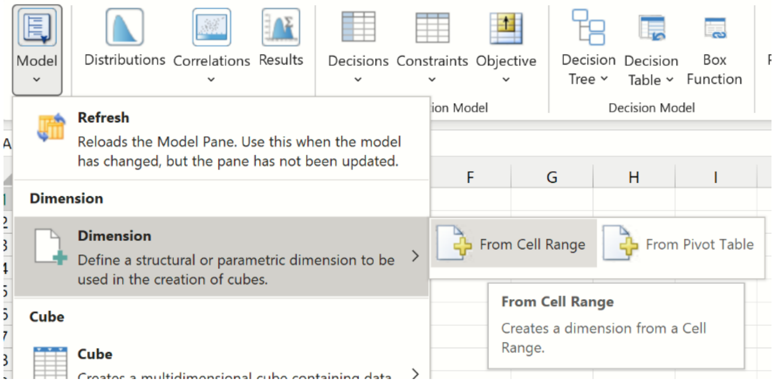
1. Define a Dimension.
2. Define a Cube.
3. Define a Cube Output cell. (This step is optional).

Defining a Dimension

Two dimensions have been created in cells K13 and L13, Products and Parts, respectively. The Products dimension (cell K13) contains three items or elements: TVs, Stereos and Speakers. The Parts dimension (cell L13) contains five items or elements: Chassis, LCD Screens, Speakers, Power Supply, and Electronics.

To create a dimension, you may either type the PsiDim() function call directly into a cell, or use the Model menu, shown below for both Analytic Solver Desktop and Analytic Solver Cloud, to insert the PsiDim() formula into the active (selected) cell.

Analytic Solver Desktop



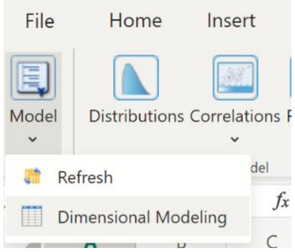
Reloads the Model Pane. Use this when the model has changed, but the pane has not been updated.

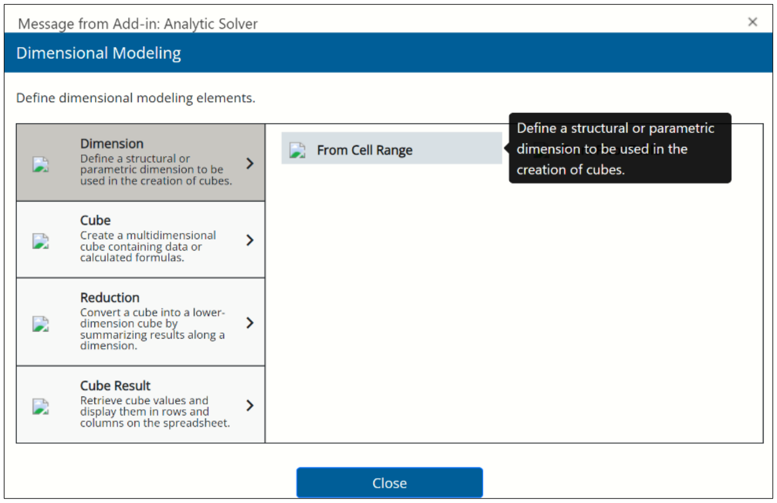
Define a structural or parametric dimension to be used in the creation of cubes.

Creates a multidimensional cube containing data

Creates a dimension from a Cell Range.

Analytic Solver Cloud





Define dimensional modeling elements.

Dimension: Define a structural or parametric dimension to be used in the creation of cubes.

Cube: Create a multidimensional cube containing data or calculated formulas.

Reduction: Convert a cube into a lower-dimension cube by summarizing results along a dimension.

Cube Result: Retrieve cube values and display them in rows and columns on the spreadsheet.

Define a structural or parametric dimension to be used in the creation of cubes.

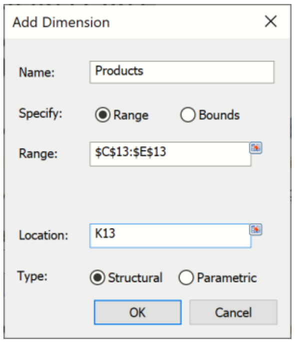
- To use the Model menu in Analytic Solver Desktop, click **Model – Dimension – From Cell Range**.
- To use the Model menu in Analytic Solver Cloud, click **Model – Dimensional Modeling – Dimension – From Cell Range**.

The Add Dimension dialog appears.

Enter a name of your choice for the *Name* field. In this case, we have used **Products**. Select *Range* and enter the Excel cell range that contains the product names. One could also enter an integer here, say “11”, which would result in the creation of a dimension with 11 elements. Select *Bounds* to enter a lower and upper bound. These bounds can be entered directly into the lower and upper bound fields or cell addresses can be used. For example, if bounds of 1 and 10 are entered directly or cell addresses of cells containing 1 and 10 are entered into the *Lower* and *Upper* fields, a dimension with 10 (upper - lower + 1) elements will be created. (In this case, there will be no names associated with the elements other than indices from 1 to n.)

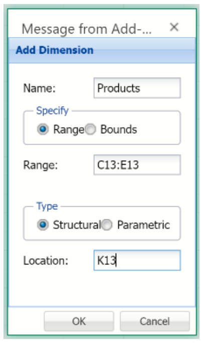
Enter the range address of **C13:E13** which hold the product names: TVs, Stereos and Speakers. Specify the cell address or defined cell name for *Location*, this example uses **K13**. Finally, select **Structural** to create a Structural dimension. (See below for instructions on how to create a Parametric Dimension.) Click **OK** to add the formula "=PsiDim("Products",C13:E13)" to cell K13, Repeat these steps in cell L13 using the *Name* **Parts** and *Range* **B18:B22**.

Analytic Solver Desktop



Name: Products
Specify: ● Range ○ Bounds
Range: $C$13:$E$13
Location: K13
Type: ● Structural ○ Parametric

Analytic Solver Cloud

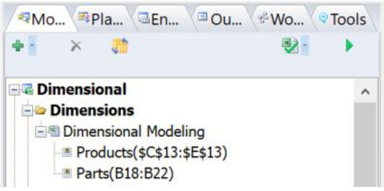


Name: Products
Specify ● Range ○ Bounds
Range: C13:E13
Type ● Structural ○ Parametric
Location: K13

For more information on the *PsiDim()* function, please see the Frontline Solvers Reference Guide.

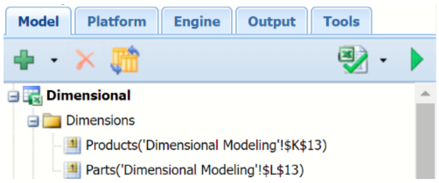
Let's take a look at the Model tab on the Solver Task Pane. You should now have two dimensions listed under the *Dimensional Modeling* worksheet, Products and Parts. The Products dimension has three elements: TVs, Stereos and Speakers. The Parts dimension has five elements: Chassis, Screen, Speaker, Power Supply and Electronics.

Analytic Solver Desktop



Dimensional
Dimensions
Dimensional Modeling
Products($C$13:$E$13)
Parts(B18:B22)

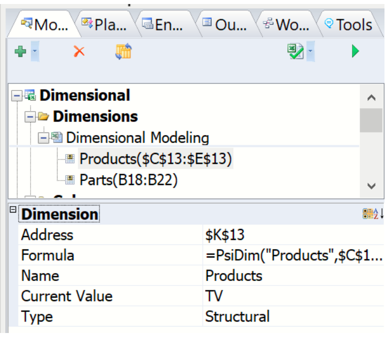
Analytic Solver Cloud



Dimensional
Dimensions
Products('Dimensional Modeling'!$K$13)
Parts('Dimensional Modeling'!$L$13)

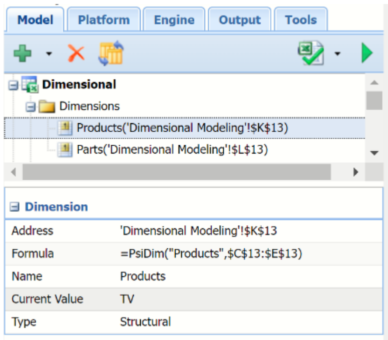
Select the Products dimension.

Analytic Solver Desktop Taskpane



Dimension	
Address	$K$13
Formula	=PsiDim("Products",$C$1...
Name	Products
Current Value	TV
Type	Structural

Analytic Solver Cloud Taskpane

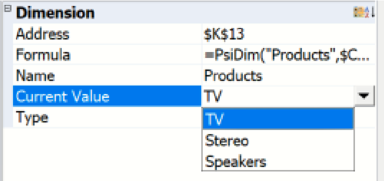

Model
Platform
Engine
Output
Tools
Dimensional
Dimensions
Products('Dimensional Modeling'!$K$13)
Parts('Dimensional Modeling'!$L$13)

Dimension	
Address	'Dimensional Modeling'!$K$13
Formula	=PsiDim("Products",$C$13:$E$13)
Name	Products
Current Value	TV
Type	Structural

Several options will appear at the bottom of the task pane under Dimension (as shown in the screenshot above). Here you can view the cell where the dimension is located (cell K13), the formula, the name of the dimension and the type of dimension (structural or parametric). These fields are all read-only.

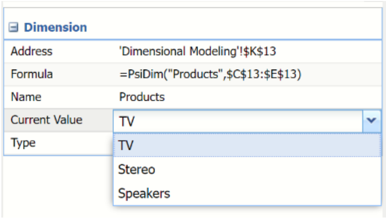
Current Value, however, controls the values displayed in the cell containing the cube. See below for more information on this field and it's affect on what is displayed in the spreadsheet.

Analytic Solver Desktop Taskpane



Dimension	
Address	$K$13
Formula	=PsiDim("Products",$C...
Name	Products
Current Value	TV
Type	

Analytic Solver Cloud Taskpane

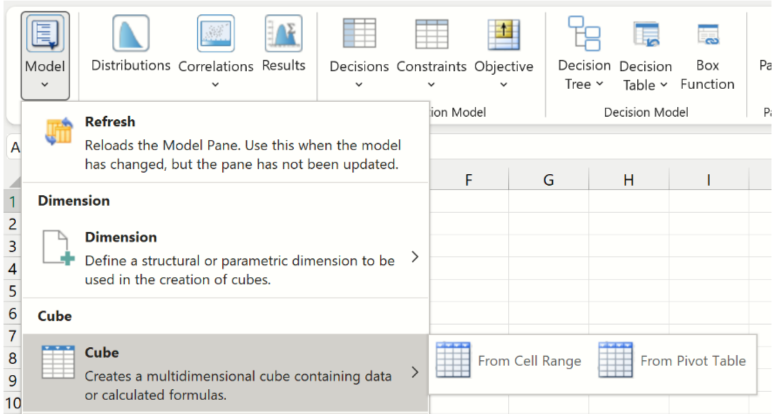


Dimension	
Address	'Dimensional Modeling'!$K$13
Formula	=PsiDim("Products",$C$13:$E$13)
Name	Products
Current Value	TV
Type	

Defining a Cube

Four Cubes have been created in cells K16 (which holds the variable cells), L16 (which holds the Profit values for each product), K17 (which holds the constraint right hand sides), and L17 (which holds the amount of parts used per product). To create a cube, you may either type the *PsiCube*() formula into a cell or use the Model menu, shown below, to insert the formula into the selected cell.

To use the Model menu in Analytic Solver Desktop, click **Model – Cube – From Cell Range**.



Refresh
Reloads the Model Pane. Use this when the model has changed, but the pane has not been updated.

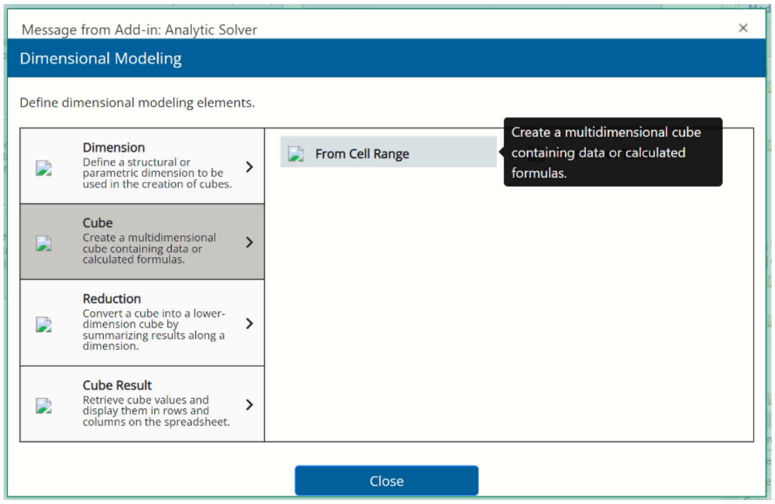
Dimension

Dimension
Define a structural or parametric dimension to be used in the creation of cubes.

Cube

Cube
Creates a multidimensional cube containing data or calculated formulas.

To use the Model menu in Analytic Solver Cloud; click Model – Dimensional Modeling, then Cube – From Cell Range.



Define dimensional modeling elements.

Dimension
Define a structural or parametric dimension to be used in the creation of cubes.

Cube
Create a multidimensional cube containing data or calculated formulas.

Reduction
Convert a cube into a lower-dimension cube by summarizing results along a dimension.

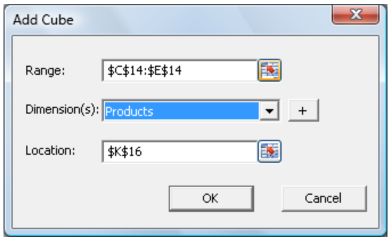
Cube Result
Retrieve cube values and display them in rows and columns on the spreadsheet.

Create a multidimensional cube containing data or calculated formulas.

In either Analytic Solver Cloud or Desktop, the *Add Cube* dialog appears.

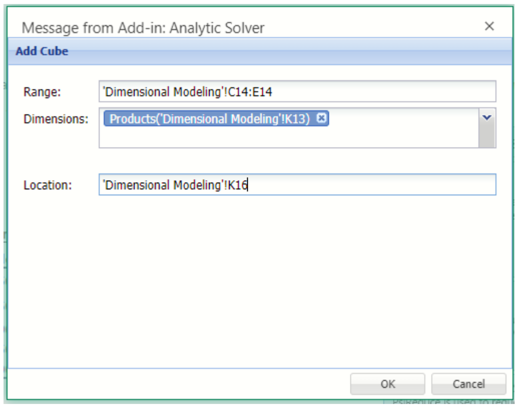
Enter an Excel cell range for *Range*, in this instance, select cells **C14:E14**, the variable values. Select **Products** from the *Dimension* drop down menu. To add more than one dimension to the cube, click "+" button. Select a blank cell for *Location* – this example uses cell K16. Click **OK** to create the cube. The following formula will be entered into cell K16, =PsiCube($C$14:$E$14,"Products"). (This formula may also be entered directly.)

Analytic Solver Desktop



Range: $C$14:$E$14
Dimension(s): Products
Location: $K$16

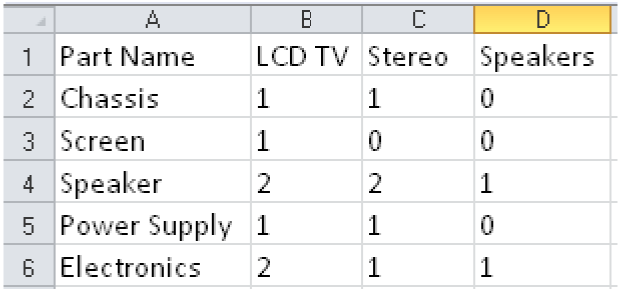
Analytic Solver Cloud



Range: 'Dimensional Modeling'!C14:E14
Dimensions: Products('Dimensional Modeling'!K13)
Location: 'Dimensional Modeling'!K16

The first argument of any cube is the cell address of the data. In the cube located in cell K16, the first argument is C14:E14 (the variable cells). The second thru ninth arguments are the dimensions for which the data will be calculated. (At least one dimension is required.) This 1-dimensional cube, calculated over the Products dimension, holds the data in cells C14:E14 which are the number of products to build - the decision variables. The data for the cube located in cell L16 can be found in cells C24:E24. This 1 dimensional cube contains the profit values for each of the products. This cube is also calculated over the Products dimension. In the 1 – dimensional cube located in cell K17 (the cube holding the constraint right hand sides), the data is contained in cells H18:H22. In cell L17, a 2-dimensional cube has been created using the *Products* and *Parts* dimensions. This cube holds the fact table containing the parts required per product (cells C18:E22).

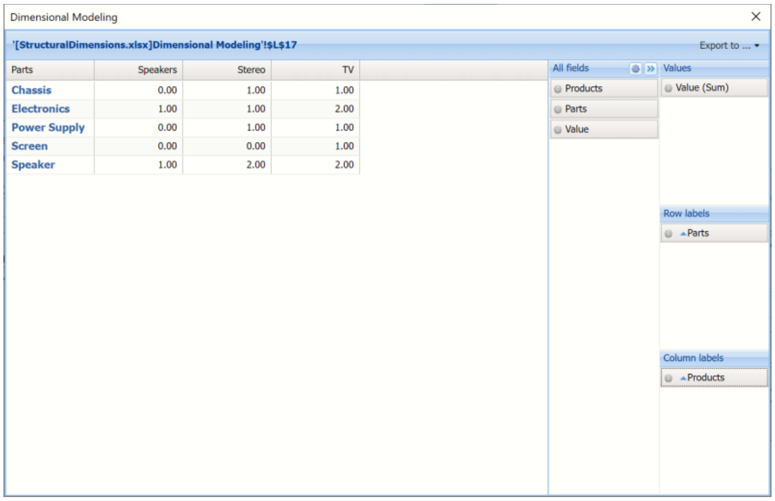
> **Note:** When calculating a cube over multiple dimensions, the "most rapidly changing" dimension *must* be first, while the "most slowly changing" dimension *must* appear last. Take the example table below.



	A	B	C	D
1	Part Name	LCD TV	Stereo	Speakers
2	Chassis	1	1	0
3	Screen	1	0	0
4	Speaker	2	2	1
5	Power Supply	1	1	0
6	Electronics	2	1	1

> Starting from cell B2 (the start of the fact table) and reading from left to right, the 1st element of the cube will be 1 (Chassis & **LCD TV**), the 2nd element of the cube will be 1 (Chassis & **Stereo**), the 3rd element of the cube will be 0 (Chassis & **Speakers**). The 4th element of the cube will be 1 (Screen & **LCD TV**), the 5th element of the cube will be 0 (Screen & **Stereo**) and the 6th element of the cube will be 0 (Screen & **Speakers**). Between Products and Parts, which dimension is changing more frequently as we read from left to right? Correct! Products is changing more frequently which means this dimension should come first in the AddCube dialog or *PsiCube()* function.

Double click cell L17 to open the fact table for the cube located in this cell. The following window appears. The data contained in this fact table is the same data contained in cells C18:E22. You can control the display formatting of the cube by dragging Products and Parts to Row Labels or Column Labels depending on personal preference.

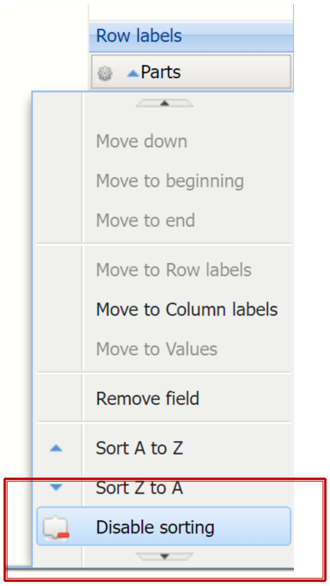

Dimensional Modeling

'[StructuralDimensions.xlsx]Dimensional Modeling'!$L$17

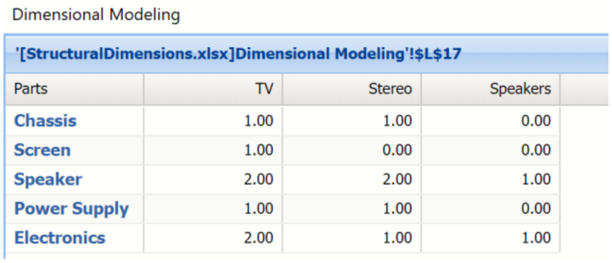
Parts	Speakers	Stereo	TV
Chassis	0.00	1.00	1.00
Electronics	1.00	1.00	2.00
Power Supply	0.00	1.00	1.00
Screen	0.00	0.00	1.00
Speaker	1.00	2.00	2.00

Notes on Grid Functionality

Parts and Products have been sorted alphabetically. To disable the sort and receive results in the same order as what is shown in the spreadsheet, click the small green arrow next to both Parts (under Row Labels) and Products (under Column Labels) and select "Disable sorting" for both.



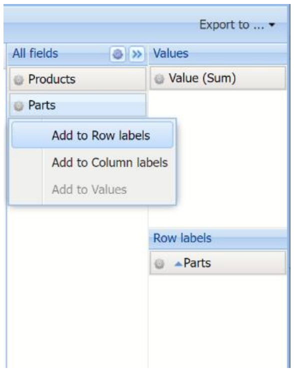
Unsorted Results of Cube in L17


Dimensional Modeling

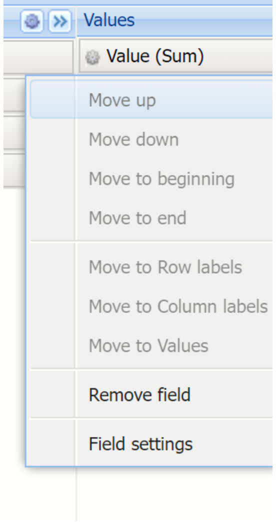
'[StructuralDimensions.xlsx]Dimensional Modeling'!$L$17

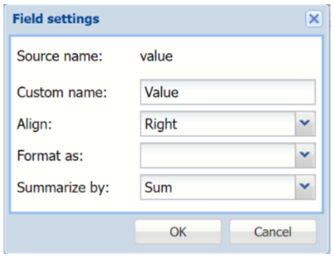
Parts	TV	Stereo	Speakers
Chassis	1.00	1.00	0.00
Screen	1.00	0.00	0.00
Speaker	2.00	2.00	1.00
Power Supply	1.00	1.00	0.00
Electronics	2.00	1.00	1.00

The organization of the grid can be easily changed to meet your needs by simply dragging the dimensions Products and Parts into Rows or Columns. You can also simply click Parts, Products or Value under "All fields" to add each directly as a Row or Column.



Click Value under "Values" to remove the field or view the Field settings.

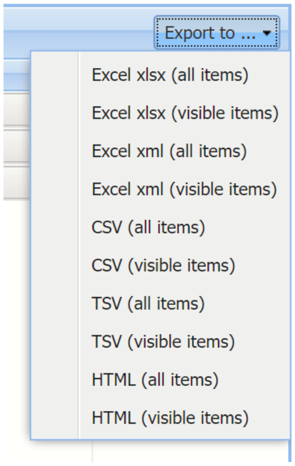



Field settings

Source name: value
Custom name: Value
Align: Right
Format as:
Summarize by: Sum

Exporting Cube Data

Click “Export to…” in the upper right hand corner of the dialog to export the cube data to Excel or a CSV, TSV or HTML file.



Cube Formulas: Operations and Dimensions

When executing an operation over cubes, if all participating cubes have the same dimensions, then the result cube will have the same dimensions too. For example, if a cube with 4 dimensions is multiplied (or added, subtracted, divided, etc.) with another cube also containing the same 4 dimensions, the result cube will contain the same 4 dimensions.

Otherwise, the result cube will be a *union* of the participating dimension sets. For example, if a cube, B2, with two dimensions is multiplied by a cube, C3, with one dimension *that also appears in the first cube*, the result B2*C3 will be two-dimensional. If C3 is defined over a dimension *different from* either of the two dimensions in B2, the result cube will be three-dimensional.

You can use almost any Excel arithmetic or intrinsic operations, as well as almost all Excel built-in functions in cube formulas. There are a few Excel functions that are not supported by the PSI Interpreter and are thus not supported in Dimensional Modeling. For a complete list of functions not supported by Analytic Solver, see the section, “Unsupported Excel Functions”, within the Analytic Solver Reference Guide.

Now let’s calculate the constraints. In the original model, the constraints are the number of parts used (cells G18:G22) must be less than the inventory (found in cells H18:H22).

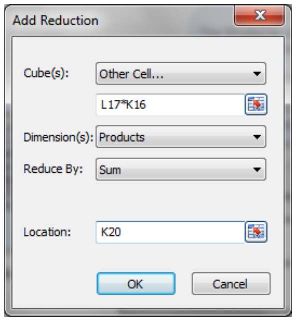
To calculate the constraint in this example using Dimensional Modeling, we must use the *PsiReduce()* function. This function reduces or aggregates a multi-dimensional cube along one dimension. A cube may be aggregated using the following supported functions: average, sum, max, min, stdev, or var. (For example, if PsiReduce() is used to aggregate a 3 –dimensional cube, the resultant cube would contain 2 dimensions.) A cube reduced along all its dimensions is called a Degenerated Cube and will be a single value, or scalar.

Select cell K20, then click **Model – Reduction** in Analytic Solver Desktop or click **Model – Dimensional Modeling – Reduction – Cube Reduction** in Analytic Solver Cloud.

The *Add Reduction* dialog appears.

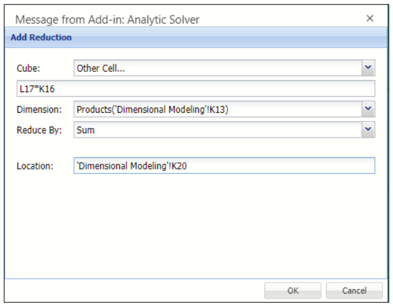
Select **Other Cell** from the *Cube* drop down menu, then enter **L17*K16** into the empty field. Since we want to reduce the cube along the Products dimension, select **Products** from the *Dimension* drop down menu. Next select **Sum** from the *Reduce By* drop down menu. This is how the cube will be aggregated. Confirm that *Location* is cell K20, then click **OK**. The formula, =PsiReduce($L$17*$K$16, "sum","Products"), is entered into cell K20. In general, the equation for this cube is: $R = \sum_j Xj * Yij$, for each i= 1 to 5, where X equals the products (TVs, stereos and speakers), Y equals the parts (chassis, screen, speaker, power supply, & electronics) and R is a 1-dimensional vector containing 5 elements.

Analytic Solver Desktop



Cube(s): Other Cell...
L17*K16
Dimension(s): Products
Reduce By: Sum
Location: K20

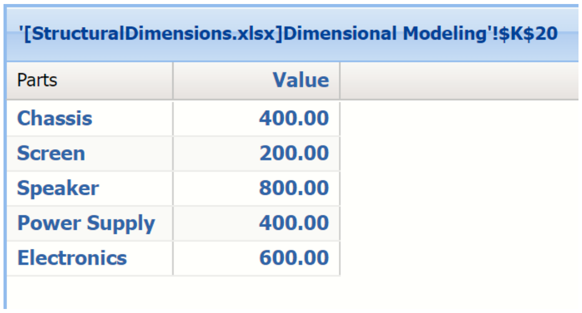
Analytic Solver Cloud



Cube: Other Cell...
L17*K16
Dimension: Products('Dimensional Modeling'!K13)
Reduce By: Sum
Location: 'Dimensional Modeling'!K20

The multiplication of the two cubes in cells L17 and K16 in the first argument of PsiReduce will result in a third cube containing the product of the *number of parts per product* and the *variable values* in cells C14:E14. The cube in cell L17 contains 15 elements and is calculated along two dimensions, Products and Parts. The cube in cell K16 has 3 elements and holds the fact table for only 1 dimension, Products. Remember, when multiplying two cubes with different dimension sets, the resultant cube will be structured along the Union of both dimension sets. In this case, the resultant cube will be structured along both the Products and Parts dimensions and will also contain 15 elements. PsiReduce() reduces or eliminates the specified dimension in the third argument, in this case the Products dimension. The cube multiplication and reduction occurring in this single cell (cell K20) perform the same operations as cells G18:G22 in the original problem! Since Cubes are single cells, they can display only the first element of their cube result. In order to see all cube elements, simply double click cell K20. A popup window appears containing the following fact table.

Note that these results are displayed as “unsorted”.



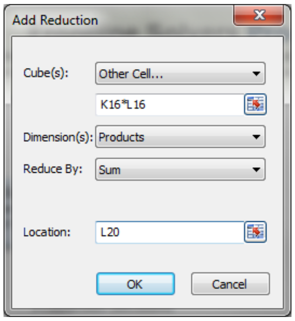
'[StructuralDimensions.xlsx]Dimensional Modeling'!$K$20

Parts	Value
Chassis	400.00
Screen	200.00
Speaker	800.00
Power Supply	400.00
Electronics	600.00

It’s also possible to check the contents of a cube on the worksheet using the PsiCubeData() function. Please see the Frontline Solvers Reference Guide for more information on this function.

PsiReduce() is used to calculate the objective function. To compute the objective, we must multiply the cube containing the decision variables (cell K16) by the cube containing the Profit values (cell L16). Select cell L20 and click **Model – Reduction**. The *Add Reduction* dialog appears.

Analytic Solver Desktop



Cube(s): Other Cell...

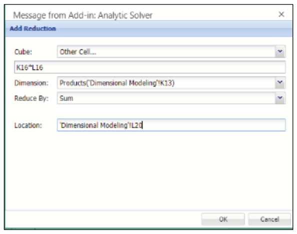
K16*L16

Dimension(s): Products

Reduce By: Sum

Location: L20

Analytic Solver Cloud



Cube: Other Cell...

K16*L16

Dimension: Products('Dimensional Modeling'!K13)

Reduce By: Sum

Location: 'Dimensional Modeling'!L20

Select **Other Cell** from the *Cube* drop down menu and enter **K16*L16** in the blank field. Since we are looking to sum over the Products dimension (sum the products of Products * Profits), select **Products** from the *Dimension* menu and **Sum** from the *Reduce By* menu. Confirm the *Location* is cell **L20** and then click **OK**. The formula, =PsiReduce($K$16*$L$16, "sum","Products"), is entered into cell L20. The result of this cube is a scalar or a single value.

Defining Outputs in Dimensional Models

It’s also possible to use an output function such as PsiOptData() to observe the contents of a fact table on the spreadsheet, rather than in a pop up fact table. Enter the formula “=PsiOptData(K20)” into cell K25. The contents of this cell will "spill" down. (If using a version of Excel that does not support dynamic arrays, highlight cells K25:K29, then use CTRL + SHIFT + ENTER on the keyboard to enter this formula as an array.) Once Solver finds a solution, these cells will be populated with the left hand side of each constraint. For example, cell K25 will display the left hand side of the first constraint, cell K26 will display the left hand side of the 2nd constraint, cell K27 will display the left hand side of the 3rd constraint, and so on. For more information on the PsiOptData function, including how to limit the results returned by dimension or "slice", see the Analytic Solver Reference Guide.

It’s also possible to create “what if” scenarios using Dimensional Modeling without ever having to run an optimization (or simulation) using a 2nd type of output function, PsiCubeOutput(). For more information, please see below under **Additional Functions**.

Now let’s create our Solver model by adding the variables and constraints to the Platform Task Pane.

Building the Model

At this point we can now start building our model in the Platform Task Pane.

1. Add the Variables: Select cells C14:E14, then click **Decisions** on the Ribbon. (Alternatively, click Decisions – Normal on the Ribbon.) If you inspect the Task Pane, you will find that cells C14:E14 now appear under the Normal Variables section.

2. Add the Constraints: Select cell K20, then click **Constraints** on the Ribbon. The Add Constraint dialog will appear. (Alternatively, click the down arrow under Constraints, then select Normal **--** <= from the menu.) Enter cell K20 as the left hand side of the constraint and cell K17 as the right hand side of the constraint, then click OK. If you inspect the Task Pane, you will find K20 <= K17 now appears under Normal Constraints.

3. Add the Objective: Select cell L20, then click **Objective**, then **OK**. (Alternatively click the down arrow under **Objective**, then select **Max – Normal** from the menu.) If you inspect the Task Pane, you will find that cell L20 now appears under Objective with “(Max)” appearing to the right.

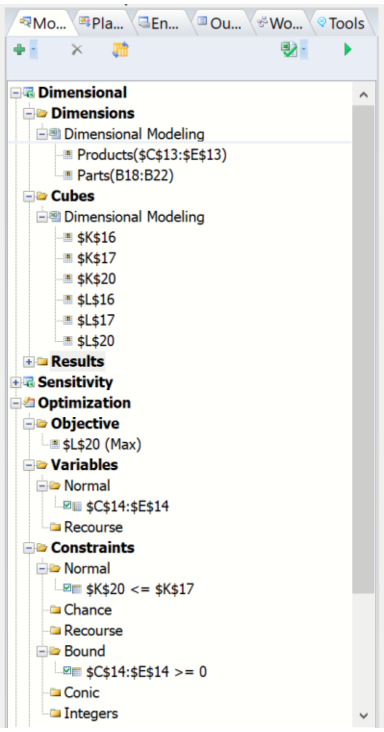
4. Add Variable Bounds: Since it would be impossible to make a negative amount of products, we must add a non-negativity bound. To do so, select cells C14:E14, then click Constraints. (Alternatively, click the down arrow under **Constraints**, and select **Variable Type/Bound --** >= from the menu**.)** The Add Constraint dialog will open. Select >= for the “sense” of the constraint (drop down menu in the middle). Enter “0” as the constraint left hand side, then click OK.

    i. In Analytic Solver Desktop, the variable bounds are added to the taskpane under Bounds.

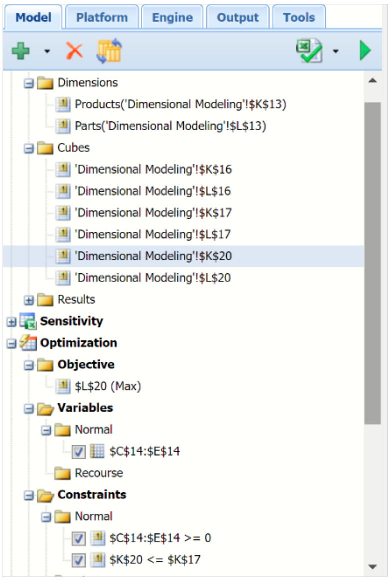
    ii. In Analytic Solver cloud, the variable bounds are added to the taskpane under Constraints.

The Task Pane will look like the following.

Analytic Solver Desktop



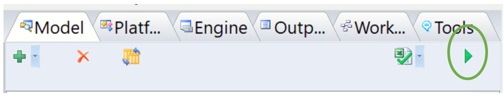
Analytic Solver Cloud


Model
Platform
Engine
Output
Tools
Dimensions
Products('Dimensional Modeling'!$K$13)
Parts('Dimensional Modeling'!$L$13)
Cubes
'Dimensional Modeling'!$K$16
'Dimensional Modeling'!$L$16
'Dimensional Modeling'!$K$17
'Dimensional Modeling'!$L$17
'Dimensional Modeling'!$K$20
'Dimensional Modeling'!$L$20
Results
Sensitivity
Optimization
Objective
$L$20 (Max)
Variables
Normal
$C$14:$E$14
Recourse
Constraints
Normal
$C$14:$E$14 >= 0
$K$20 <= $K$17

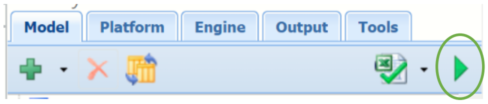
Solving the Model

Enter 0's into cells C14:E14 then solve the model by clicking the green arrow on the Model Task Pane.

Analytic Solver Desktop



Analytic Solver Cloud

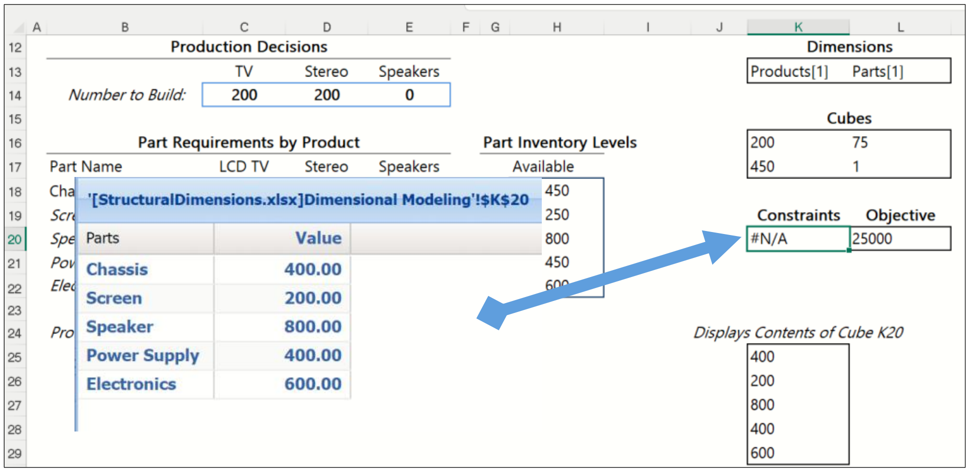


Solver will solve the model and display the final variable values in cells C14:E14: 200, 200, and 0, respectively. The status bar (found beneath the Task Pane) will display the message *Solver found a solution* with a green background. Analytic Solver will populate the objective formula located in L20 when the optimization finishes successfully.

Notes:

1. All Solver engines *except* Interval Global Solver support Dimensional Modeling.
2. Dimensional Modeling is not supported in RASON Decision Services. As a result, it is not possible to deploy a model containing cubes and dimensions.

3. Analytic Solver Cloud will display the value “400” in cell K20 in the final solution, this is the first value in the cube array. Alternately, Analytic Solver Cloud will display #N/A. This is because Analytic Solver Desktop only displays a value for PsiReduce if the formula reduces to a scalar while Analytic Solver Cloud displays the first value in the cube.



Production Decisions

	TV	Stereo	Speakers
Number to Build:	200	200	0

Part Requirements by Product

Part Inventory Levels

Part Name LCD TV Stereo Speakers Available

Available: 450, 250, 800, 450, 600

'[StructuralDimensions.xlsx]Dimensional Modeling'!$K$20

Parts	Value
Chassis	400.00
Screen	200.00
Speaker	800.00
Power Supply	400.00
Electronics	600.00

Dimensions

Products[1]	Parts[1]

Cubes

200	75
450	1

Constraints	Objective
#N/A	25000

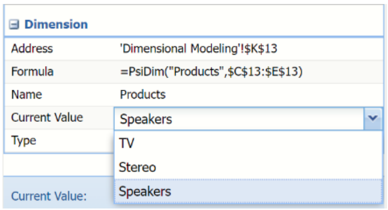
Displays Contents of Cube K20

400
200
800
400
600

Current Value

As discussed earlier in this section, the Current Value field in the lower portion of the taskpane has an effect on what is displayed in the spreadsheet.

Analytic Solver Desktop Taskpane Analytic Solver Cloud Taskpane

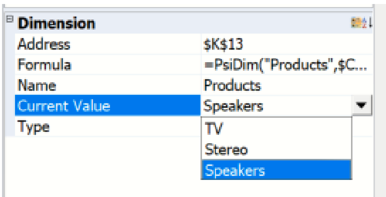


Dimension

Address	'Dimensional Modeling'!$K$13
Formula	=PsiDim("Products",$C$13:$E$13)
Name	Products
Current Value	Speakers
Type	

TV
Stereo
Speakers

Current Value:

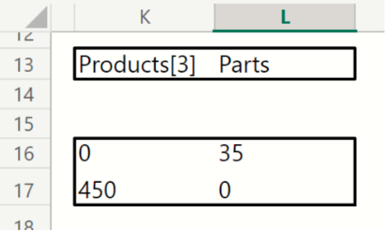


Dimension

Address	$K$13
Formula	=PsiDim("Products",$C...
Name	Products
Current Value	Speakers
Type	

TV
Stereo
Speakers

For example, after the model is solved select Speakers for Current Value. Values in cells K16, K17, L16, and L17 will change to reflect values for Speakers. For example, K13 will change to Products[3] to indicate that the third element of the Products dimension has been selected. K16 will change to 0 (the value of cell C13), K17 will change to 450 (the value of cell H18), L16 will change to 35 (the value of cell E24), and cell L17 will change to 0 to reflect that 0 chassis are used in the production of speakers.



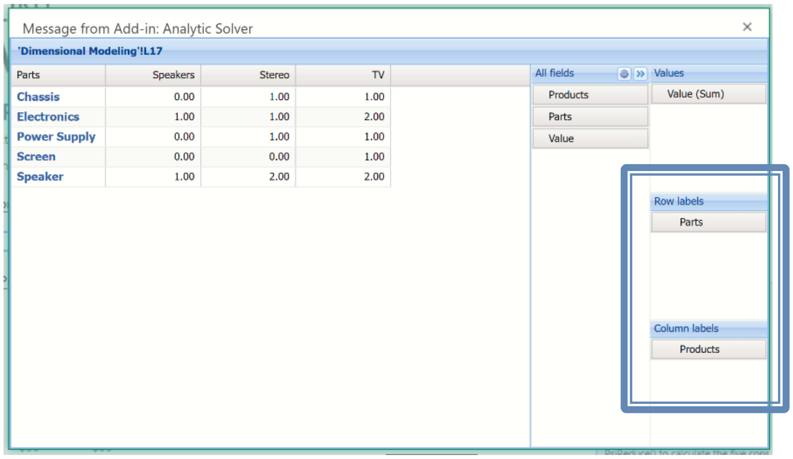
	K	L
13	Products[3]	Parts
14		
15		
16	0	35
17	450	0

Opening the Grids

In Analytic Solver Desktop, a Dimensional Modeling grid can be opened either by double clicking the cell containing the cube or dimension or by clicking the cell address of the cube or dimension in the Solver Taskpane.

In Analytic Solver Cloud, a Dimensional Modeling grid can be opened by double clicking the cell address of the cube or dimension in the Solver Taskpane.

In either product, the opened grid displays the contents of the dimension or cube. For example, double clicking cell L17 opens a grid displaying the number of parts required to manufacture each of the three products.



Parts	Speakers	Stereo	TV
Chassis	0.00	1.00	1.00
Electronics	1.00	1.00	2.00
Power Supply	0.00	1.00	1.00
Screen	0.00	0.00	1.00
Speaker	1.00	2.00	2.00

Note that the dimensions of the cube can be changed by dragging Products to Row labels and Parts to Column labels on the right of the dialog.

As you can see by the comparison of these two models, using a cube to model a problem will result in a more compact and easier to read model. Continue reading the next section to learn about the 2nd goal of dimensional modeling, the parametric dimension.

Now close this example, and continue reading to discover how to use a parametric dimension within an optimization model.

Notice that cell K20, which contains a PsiReduce() function, in the Structural Dimensions example displays #NA.

- In Analytic Solver Desktop, PsiReduce() functions are also displayed as #NA in the cell unless the cube reduces to a scalar value, such as the objective function cell in L20.
- In Analytic Solver Cloud, PsiReduce() functions always resolve.

Additional Functions

Using Dimensional Modeling, it's possible to create what if scenarios without ever having to run an optimization (or simulation). The function PsiCubeData() will list the contents of any cube on the spreadsheet much like PsiOptData() listed the left hand sides of the constraints in the example above.

If using a version of Desktop Excel that does not support dynamic arrays, use CTRL + SHIFT + ENTER on the keyboard to enter the PsiCubeData() formula as an array. For more information on array entering the PsiCubeData() function, see the Analytic Solver Reference Guide.

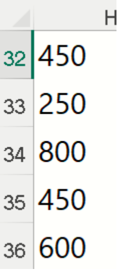
- In Analytic Solver Desktop, PsiOptData() will only be populated after an optimization is run, while PsiCubeData can be populated at any time using **Model – Cube Result – Calculate**.
- In Analytic Solver Cloud, PsiOptData() and PsiCubeData() behave the same. Both simply display the data in the cube. If the data has been calculated using Cube Result - Calculate, then both functions display the computed data. If data has been optimized, then both functions display the optimized data.

Use *PsiCubeData()* to list the contents of the Cube located in cell K17

The cube in cell K17 holds the constraint right hand sides. Enter "*=PsiCubeData(K17)*" into cell H32. The contents of this cell will "spill" down. (If using a version of Excel that does not support dynamic arrays, use CTRL + SHIFT + ENTER on the keyboard to enter this formula as an array.) Click **Model – Cube Result – Calculate** to recalculate all cubes on the worksheet.

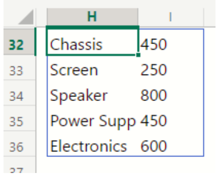
Note: In either Analytic Solver Desktop or Cloud, it's not required to recalculate all cubes (Model – Dimensional Modeling – Cube Result – Output) to display the contents of the cube using PsiCubeData. In this instance, you'll notice that the cube contents are "spilled" immediately after entering the formula. However, if any data has been changed on the worksheet, all cubes should be re-calculated in order for the cube to display the most up-to-date information.

- In Analytic Solver Desktop, this formula displays the numerical contents of the cube, as shown in the screenshot below.



	H
32	450
33	250
34	800
35	450
36	600

- In Analytic Solver Cloud, this formula displays the entire contents of the cube, including the dimension names and numerical values, as shown in the screenshot below.

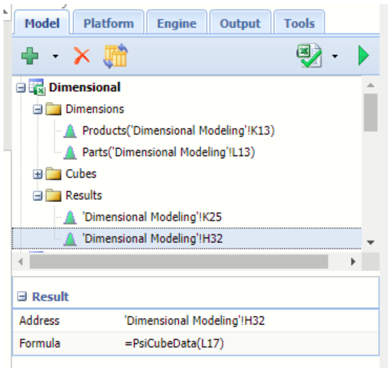


	H	I
32	Chassis	450
33	Screen	250
34	Speaker	800
35	Power Supp	450
36	Electronics	600

These values correspond to the values in cells H18:H22.

Notice that this function appears under Cube Results in the Solver Task Pane.

Analytic Solver Desktop

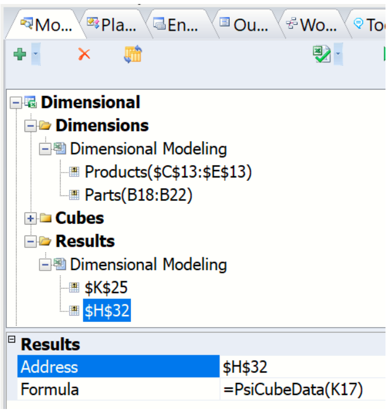


Dimensional
  Dimensions
    Products('Dimensional Modeling'!K13)
    Parts('Dimensional Modeling'!L13)
  Cubes
  Results
    'Dimensional Modeling'!K25
    'Dimensional Modeling'!H32

Result

Address	'Dimensional Modeling'!H32
Formula	=PsiCubeData(L17)

Analytic Solver Cloud



Dimensional
  Dimensions
    Dimensional Modeling
      Products($C$13:$E$13)
      Parts(B18:B22)
  Cubes
  Results
    Dimensional Modeling
      $K$25
      $H$32

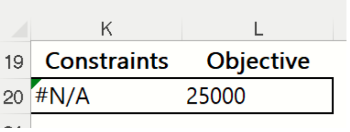
Results

Address	$H$32
Formula	=PsiCubeData(K17)

Use PsiCubeOutput() to display the value of a degenerated cube.

To calculate and display the value of a degenerated cube (a cube that has been reduced to a scalar value), such as the cube in cell L20, click **Model – Cube Result – Output** to append "+PsiCubeOutput()" to the existing formula to designate this cell as a cube output. Afterwards, click **Model – Cube Result – Calculate** to recalculate all cubes on the worksheet.

Note: PsiCubeOutput is not required in cell L20 when running an optimization (in either Analytic Solver Desktop or Cloud) as Solver will calculate and display the contents of the objective function after a solution is found.



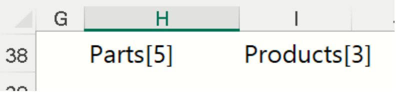
	K	L
19	Constraints	Objective
20	#N/A	25000

Use PsiCubeData(cell, "dims") to display the dimensions used in a cube.

Now let's use PsiCubeData() to display the multidimensional cube in cell L17. First, we will determine whether the result in L17 is a cube and which dimensions constitute that cube.

Enter the formula =PsiCubeData(L17, "dims") into cell H38.

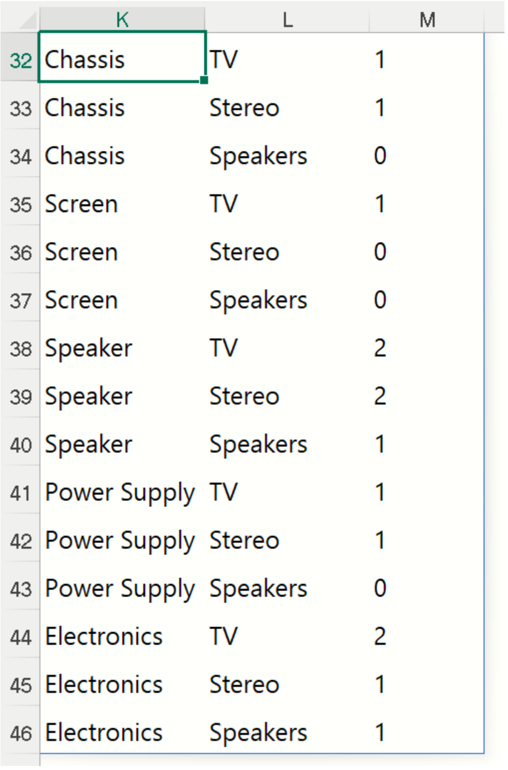
When "dims" is passed for the *struc_format* argument (2nd argument), the name and size of each dimension will be returned. The contents of this cell will "spill" across. Afterwards, I38 will display "Products[3]" indicating that there are 3 elements in the Products dimension and H38 will display "Parts[5]" indicating that there are 5 elements in the Parts dimension.



	G	H	I
38		Parts[5]	Products[3]

Use PsiCubeData(cell, "vals") to display cube results in a relational table

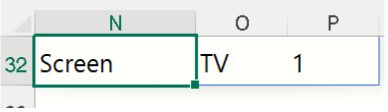
We can also display the entire fact table in Analytic Solver Desktop, by entering *PsiCubeData()* again, but this time we'll use "vals" for the *struc_format* argument. When passing "vals" for the 2nd argument, *PsiCubeData()* will display the data as a relational table. Since there are 3 elements in the Products table and 5 elements in the Parts table, the fact table for this cube will contain 15 (3 x 5) elements. Enter the formula, "=PsiCubeData(L17, "vals") into cell K32. The contents of this cell will "spill" down 15 cells and across 3 columns.



	K	L	M
32	Chassis	TV	1
33	Chassis	Stereo	1
34	Chassis	Speakers	0
35	Screen	TV	1
36	Screen	Stereo	0
37	Screen	Speakers	0
38	Speaker	TV	2
39	Speaker	Stereo	2
40	Speaker	Speakers	1
41	Power Supply	TV	1
42	Power Supply	Stereo	1
43	Power Supply	Speakers	0
44	Electronics	TV	2
45	Electronics	Stereo	1
46	Electronics	Speakers	1

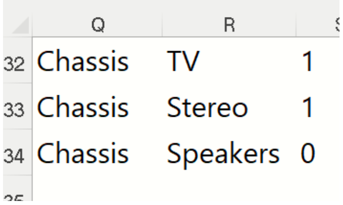
***Use PsiCubeData()* to display a slice or a single element of the fact table.**

- To display the cube data only for Part: Screen and Product: TV, enter the following formula: =PsiCubeData(L17, “[Parts].[Screen], [Products].[TV]”) into cell N32. The contents of this cell will "spill" across, displaying the following.



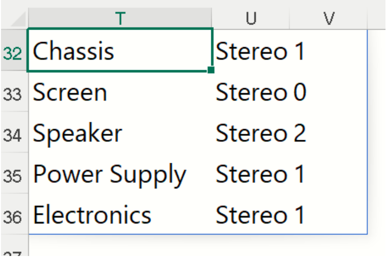
	N	O	P
32	Screen	TV	1

- To display the cube data only for Part: Chassis, across all products use =PsiCubeData(L17, "[Parts].[Chassis]") in cell Q32.



	Q	R	
32	Chassis	TV	1
33	Chassis	Stereo	1
34	Chassis	Speakers	0

- To display cube data for only one product type, Stereos, across all parts, use =PsiCubeData(L17, "[Products].[Stereo]") in cell T32.



	T	U	V
32	Chassis	Stereo 1	
33	Screen	Stereo 0	
34	Speaker	Stereo 2	
35	Power Supply	Stereo 1	
36	Electronics	Stereo 1	

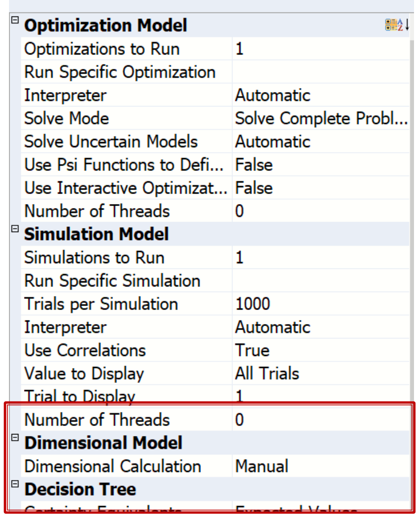
Important Notes about PsiCubeData and PsiOptData

In Analytic Solver Desktop, PsiCubeData() is supported only through the calculation API, which is useful during model development or whenever the result of a cube is desired. These formulas will *not* be populated during an optimization. To calculate a PsiCubeData function before or after an optimization, use **Cube Result** – **Calculate** on the Dimensional Modeling menu. In Analytic Solver Cloud, PsiCubeData() is populated both after an optimization and a calculation.

Setting Dimensional Calculation (on the Platform tab of the Model task pane in Analytic Solver Desktop) allows the PsiCubeData() functions to be calculated after an optimization simply by clicking F9. If Dimensional Calculation is set to Manual, use Cube Result – Calculate on the Dimensional Modeling menu to calculate the function.

The PsiOptData() function can be used in place of PsiCubeData() but only with cubes holding an optimization model component, such as a cube holding decision variables (i.e. K16), or calculating constraints (i.e. K20). PsiOptData() will be populated after an optimization regardless of the setting of Dimensional Calculation.

Dimensional Calculation option
(Supported only in Analytic Solver Desktop)



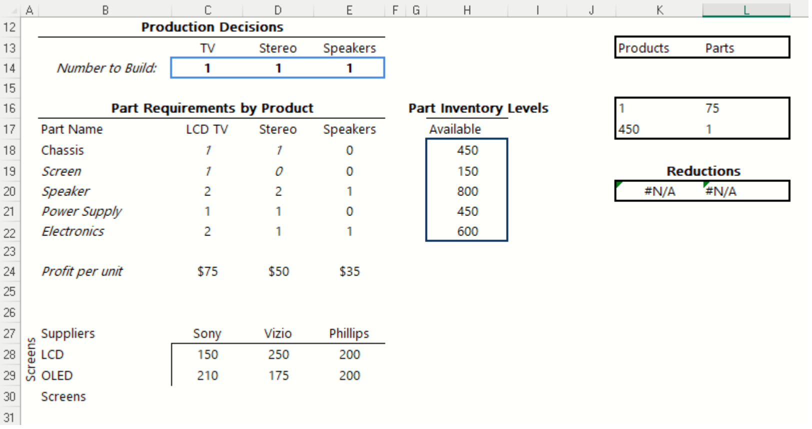
Optimization Model	
Optimizations to Run	1
Run Specific Optimization	
Interpreter	Automatic
Solve Mode	Solve Complete Probl...
Solve Uncertain Models	Automatic
Use Psi Functions to Defi...	False
Use Interactive Optimizat...	False
Number of Threads	0
Simulation Model	
Simulations to Run	1
Run Specific Simulation	
Trials per Simulation	1000
Interpreter	Automatic
Use Correlations	True
Value to Display	All Trials
Trial to Display	1
Number of Threads	0
Dimensional Model	
Dimensional Calculation	Manual
Decision Tree	
Certainty Equivalents	Expected Values

Parametric Dimensions with Optimization

You can define and use a Parametric Dimension to run multiple optimizations, simulations, and/or calculations. Analytic Solver treats all data included in a parametric cube as parameter values, such as ones defined by PsiOptParam(), PsiSimParam() and PsiSenParam(). Recall that these parameters are automatically varied by Analytic Solver in order to create a type of "what-if" analysis. Please see the chapter "Examples: Parameters and Sensitivity Analysis" in this Guide for more information on the PsiOptParam(), PsiSimParam() and PsiSenParam() functions. *If a model only includes Parametric Dimensions, each value in the Output cube will be associated with a different simulation or optimization parameter, rather than a set of model functions.*

Open the example, Parametric Dimensions.xlsx by clicking **Help – Example Models** on the Analytic Solver ribbon. The Frontline Example Models Overview opens. Select the *Optimization* tab and choose the **Parametric Dimensions** workbook from the list.

In this example, we will illustrate the use of a Parametric Dimension by adding an additional element to the model. Imagine that you have a choice to purchase two different types of TV screens: LCD and OLED, from three different suppliers, Sony, Vizio, and Phillips, for about the same price. Each supplier can deliver a specific number of screens depending on the type of screen chosen. For example, Sony can only deliver 150 LCD screens, but 210 Plasma screens while Vizio can deliver 250 LCD screens but only 175 OLED. Your job is to now not only decide how many TVs, stereos and speakers to make, but also to decide what type of screen will go into the TVs: OLED or LCD.

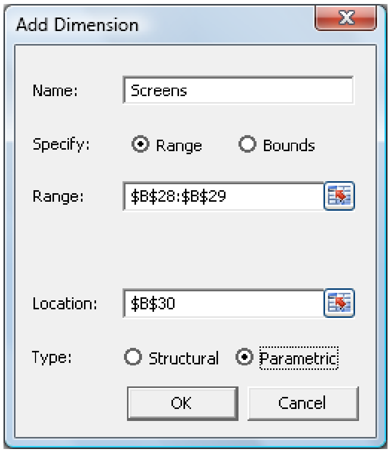


	A	B	C	D	E	H	K	L
12		**Production Decisions**						
13			TV	Stereo	Speakers		Products	Parts
14		*Number to Build:*	**1**	**1**	**1**			
15								
16		**Part Requirements by Product**				**Part Inventory Levels**	1	75
17		Part Name	LCD TV	Stereo	Speakers	Available	450	1
18		Chassis	*1*	*1*	0	450		
19		*Screen*	*1*	*0*	0	150	**Reductions**	
20		*Speaker*	2	2	1	800	#N/A	#N/A
21		*Power Supply*	1	1	0	450		
22		*Electronics*	2	1	1	600		
23								
24		*Profit per unit*	$75	$50	$35			
25								
26								
27		Suppliers	Sony	Vizio	Phillips			
28	Screens	LCD	150	250	200			
29		OLED	210	175	200			
30		Screens						
31								

Cell B30 contains a parametric dimension. To add this dimension, click **Model – Dimension – From Cell Range** in Analytic Solver Desktop or **Model – Dimensional Modeling – Dimension – From Cell Range** in Analytic Solver Cloud, to open the Add Dimension dialog. Enter **Screens** for *Name*, Select the *Range* radio button, select or enter cells **B28:B29** in the *Range* field, confirm cell B30 is selected for *Location*, then select **Parametric**. Then click **OK**.

Analytic Solver Cloud

Analytic Solver Desktop



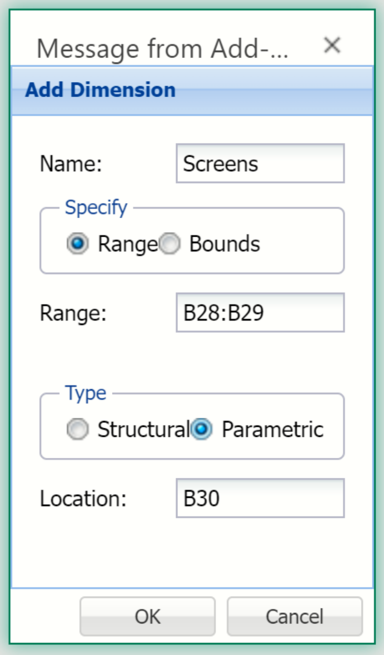
Name: Screens

Specify: ● Range ○ Bounds

Range: $B$28:$B$29

Location: $B$30

Type: ○ Structural ● Parametric



Name: Screens

Specify: ● Range ○ Bounds

Range: B28:B29

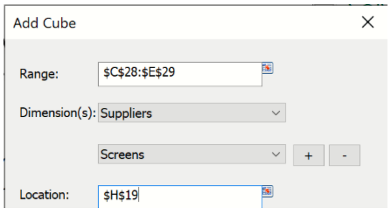
Type: ○ Structural ● Parametric

Location: B30

The following formula is entered into cell B30, = *PsiParamDim("Screens", B28:B29*). As you can see, the arguments for *PsiParamDim()* are identical to *PsiDim()*. The first argument is the name of the dimension and the 2nd argument is an Excel range. It's also possible to enter upper and lower bounds for the 2nd argument (select Bounds on the Add Dimension dialog) or to simply enter an integer n which will create a parametric dimension with n elements. (In this case, there will be no names associated with the elements other than indices from 1 to n.) Repeat the steps above to create a second Parametric Dimension in cell H19 using the name **Suppliers** and the Excel range **C27:E27**.

Now select cell H19 and click **Model – Cube – From Cell Range** in Analytic Solver Desktop or **Model – Dimensional Modeling – Cube – From Cell Range** in Analytic Solver Cloud, to bring up the *Add Cube* dialog. This cube will contain the values from cells C28:E29. Therefore, select cells **C28:E29** for *Range*. Select **Suppliers** for *Dimensions*, then click the "+" command button to add a 2nd dimension, **Screens**. Note: As mentioned above, when calculating over multiple dimensions, the "columns" dimension, or the dimension that is most often changing, must always come first. Confirm that H19 is listed for Location and click OK.

Analytic Solver Desktop



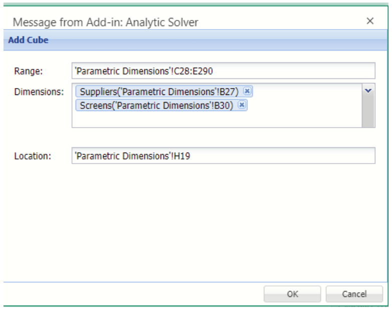
Range: $C$28:$E$29

Dimension(s): Suppliers

Screens

Location: $H$19

Analytic Solver Cloud



Range: 'Parametric Dimensions'!C28:E290

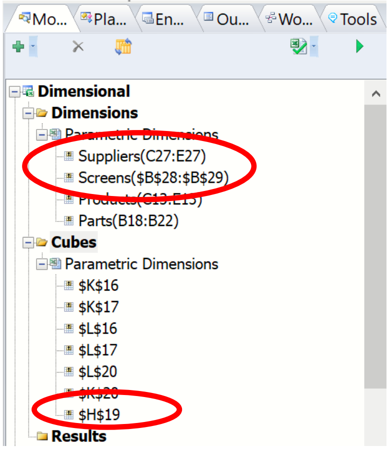
Dimensions: Suppliers('Parametric Dimensions'!B27)
Screens('Parametric Dimensions'!B30)

Location: 'Parametric Dimensions'!H19

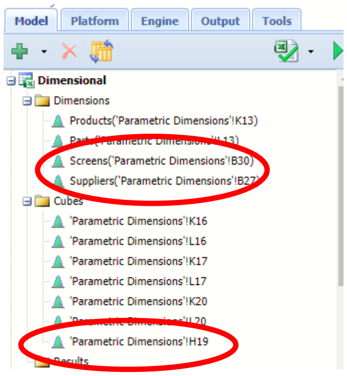
The formula =*PsiCube($C$28:$E$29,"Suppliers","Screens")* will be entered into cell H19.

If you inspect the Model tab on the Task Pane, you'll see that two more dimensions (Suppliers and Screens) and one more cube ($H$19) appear under the appropriate headings.

Analytic Solver Desktop



Analytic Solver Desktop

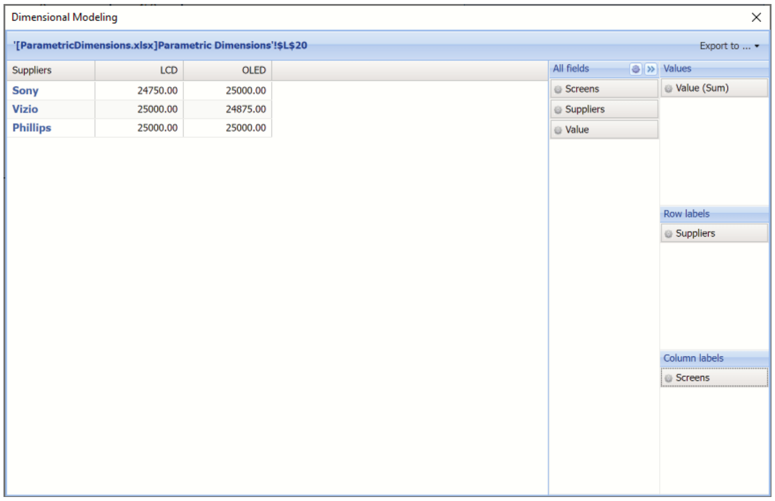

Model
Platform
Engine
Output
Tools
Dimensional
Dimensions
Products('Parametric Dimensions'!K13)
Parts('Parametric Dimensions'!L13)
Screens('Parametric Dimensions'!B30)
Suppliers('Parametric Dimensions'!B27)
Cubes
'Parametric Dimensions'!K16
'Parametric Dimensions'!L16
'Parametric Dimensions'!K17
'Parametric Dimensions'!L17
'Parametric Dimensions'!K20
'Parametric Dimensions'!L20
'Parametric Dimensions'!H19
Results

Solving with Parametric Dimensions

Now click the green arrow on the Model tab to run Solver. Solver will run six different optimizations each time using a different value from the cell range C28:E29 for cell H19. For example, for the first optimization, Solver will use 150 for the value of H19, for the 2nd optimization, Solver will use 210 for the value in cell H19, for the 3rd optimization, Solver will use 250 for the value in cell H19, etc.

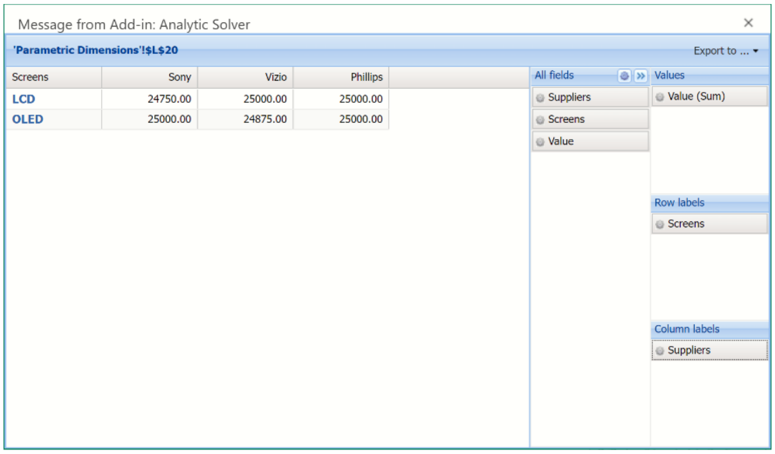
To easily view the results of the six different optimizations, simply double click cell L20 in Analytic Solver Desktop or L20 in the Model tab of the Solver Task Pane if using Analytic Solver Cloud, to display the grid shown below. (Note the results shown below are unsorted.)

Analytic Solver Desktop



Suppliers	LCD	OLED
Sony	24750.00	25000.00
Vizio	25000.00	24875.00
Phillips	25000.00	25000.00

Analytic Solver Cloud

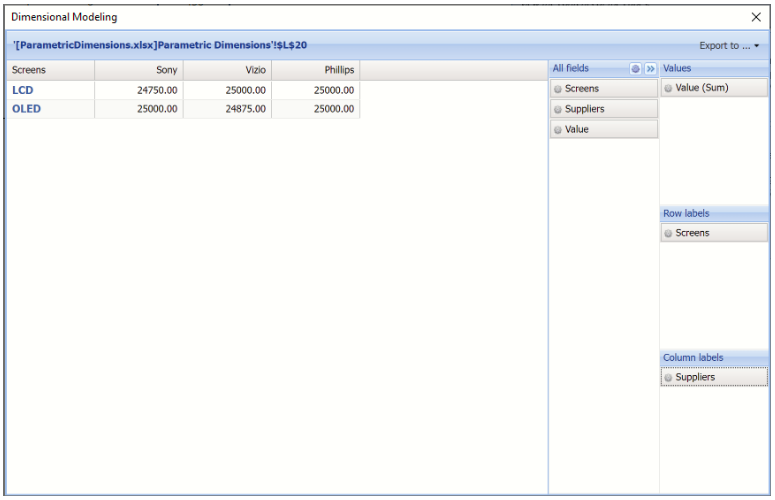


Screens	Sony	Vizio	Phillips
LCD	24750.00	25000.00	25000.00
OLED	25000.00	24875.00	25000.00

Instantly six different “what if” scenarios are displayed organized by supplier and screen. As you can easily see there are four different options that result in the largest profit. Purchasing LCD screens from either Vizio or Phillips or OLED screens from Sony or Phillips will result in a Gross Profit of $25,000. Purchasing OLED screens from Vizio will result in a decreased profit of $24,875 and purchasing LCD screens from Sony results in an even smaller profit of $24,750. To view the products across the top and the manufacturers down the left of the grid, simply drag “Screens” to Row Labels (on the right of the dialog) and “Suppliers” to "Column Labels".

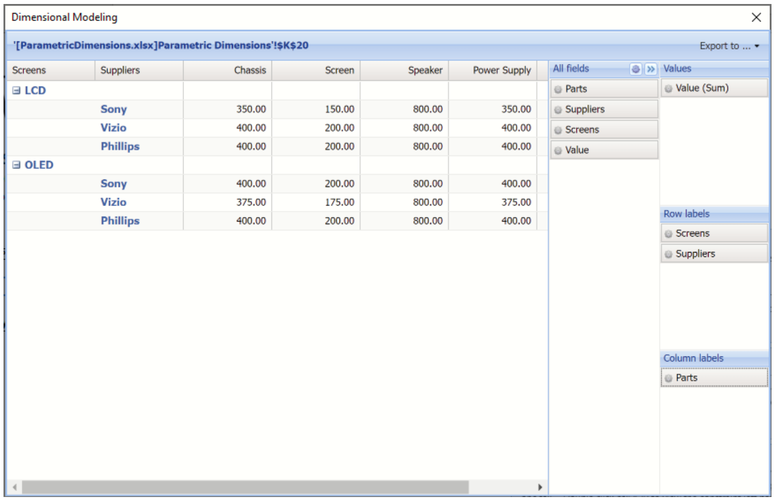
Notice that Analytic Solver Cloud delivers results in a slightly different orientation as Analytic Solver Desktop. To obtain the same orientation in Analytic Solver Desktop, simply drag Supplies to Row Labels and Screens to Column Labels, as shown in the screenshot below.

Analytic Solver Desktop



Screens	Sony	Vizio	Phillips
LCD	24750.00	25000.00	25000.00
OLED	25000.00	24875.00	25000.00

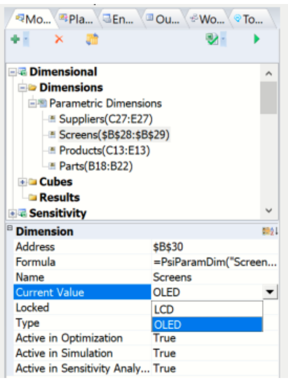
Double click cell K20 in Analytic Solver Desktop or K20 in the Model tab of the Solver Task Pane if using Analytic Solver Cloud, to view the number of parts required in the construction of the TVs for each of the six different screen options. (Note the results shown below are unsorted.)



Screens	Suppliers	Chassis	Screen	Speaker	Power Supply
LCD					
	Sony	350.00	150.00	800.00	350.00
	Vizio	400.00	200.00	800.00	400.00
	Phillips	400.00	200.00	800.00	400.00
OLED					
	Sony	400.00	200.00	800.00	400.00
	Vizio	375.00	175.00	800.00	375.00
	Phillips	400.00	200.00	800.00	400.00

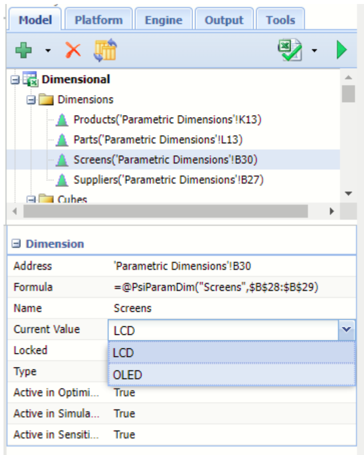
If you wanted to view the final objective value and variables in the spreadsheet when Sony OLEDs as used in the construction of the TVs, first select *Screens* on the Model tab, then select *OLED* for Current Value.

Analytic Solver Desktop



Dimension	
Address	$B$30
Formula	=PsiParamDim("Screen...
Name	Screens
Current Value	OLED
Locked	
Type	
Active in Optimization	True
Active in Simulation	True
Active in Sensitivity Analy...	True

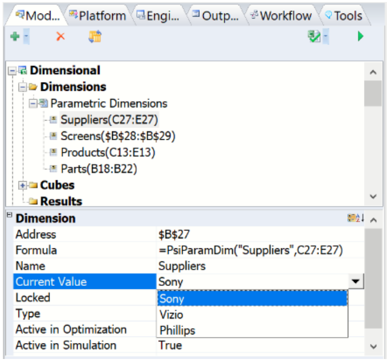
Analytic Solver Cloud


Model
Platform
Engine
Output
Tools
Dimensional
Dimensions
Products('Parametric Dimensions'!K13)
Parts('Parametric Dimensions'!L13)
Screens('Parametric Dimensions'!B30)
Suppliers('Parametric Dimensions'!B27)
Cubes
LCD
OLED

Dimension	
Address	'Parametric Dimensions'!B30
Formula	=@PsiParamDim("Screens",$B$28:$B$29)
Name	Screens
Current Value	LCD
Locked	
Type	
Active in Optimi...	True
Active in Simula...	True
Active in Sensiti...	True

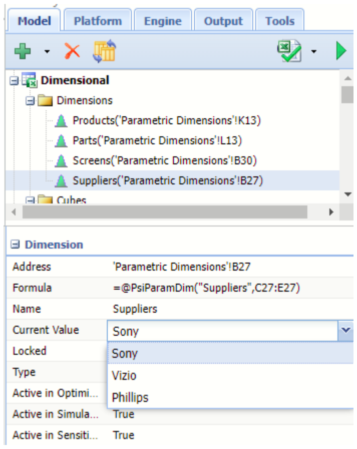
Now select *Suppliers* on the Model tab, then select *Sony* for *Current Value*.

Analytic Solver Desktop



Dimension	
Address	$B$27
Formula	=PsiParamDim("Suppliers",C27:E27)
Name	Suppliers
Current Value	Sony
Locked	
Type	
Active in Optimization	True
Active in Simulation	True

Analytic Solver Cloud


Model
Platform
Engine
Output
Tools
Dimensional
Dimensions
Products('Parametric Dimensions'!K13)
Parts('Parametric Dimensions'!L13)
Screens('Parametric Dimensions'!B30)
Suppliers('Parametric Dimensions'!B27)
Cubes
Sony
Vizio
Phillips

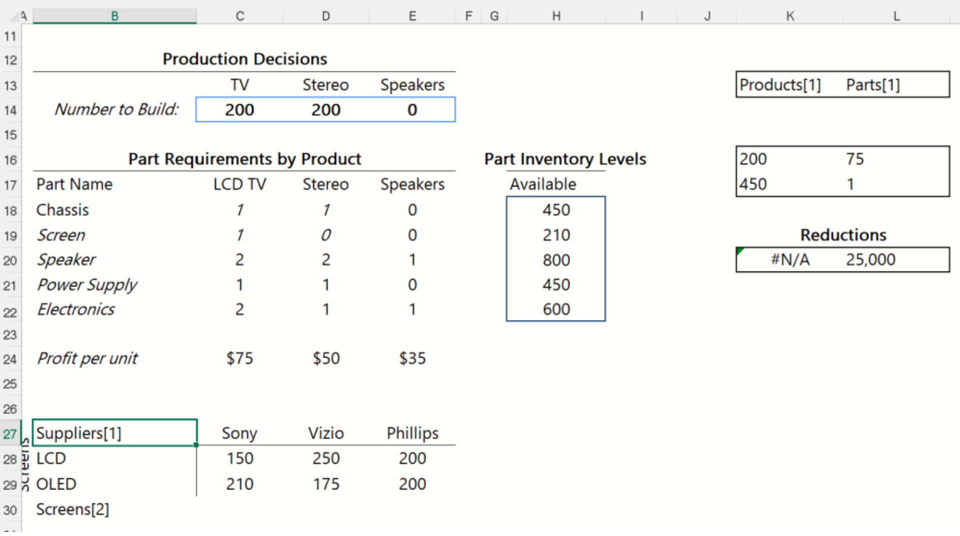
Dimension	
Address	'Parametric Dimensions'!B27
Formula	=@PsiParamDim("Suppliers",C27:E27)
Name	Suppliers
Current Value	Sony
Locked	
Type	
Active in Optimi...	True
Active in Simula...	True
Active in Sensiti...	True

Analytic Solver Cloud: Recalculating the Worksheet

If using Analytic Solver Cloud, you'll need to refresh the worksheet in order to recalculate the PsiCube() functions. (This is not required in Analytic Solver Desktop.) You can refresh the worksheet in one of two ways:

1. By clicking the refresh icon at the top of the Tsk Pane.
2. By triggering a manual recalculation by clicking Analytic Solver – Model – Dimensional Modeling – Cube Result – Calculate.

Cell H19 will change to 210 to reflect the value in cell C29 which is the number of OLED screens Sony can supply. In addition, the final variable values in cells C14:E14 will change to 200, 200, 0, the objective shown in cell L20 will change to $25,000, the parametric dimensions in cells B27 and B30 will change to Suppliers[1] (to reflect that the first element of the Suppliers dimension is selected as the current value) and Screens[2] (to reflect that the 2nd element of the Screens dimension is selected as the current value).



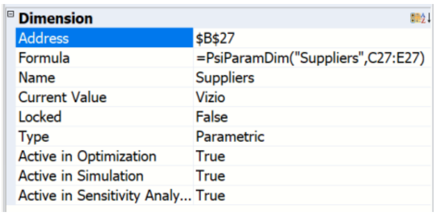
	A	B	C	D	E	H	K	L
11								
12			Production Decisions					
13			TV	Stereo	Speakers		Products[1]	Parts[1]
14		*Number to Build:*	**200**	**200**	**0**			
15								
16			**Part Requirements by Product**			**Part Inventory Levels**	200	75
17		Part Name	LCD TV	Stereo	Speakers	Available	450	1
18		Chassis	*1*	*1*	0	450		
19		*Screen*	*1*	*0*	0	210	**Reductions**	
20		*Speaker*	2	2	1	800	#N/A	25,000
21		*Power Supply*	1	1	0	450		
22		*Electronics*	2	1	1	600		
23								
24		*Profit per unit*	$75	$50	$35			
25								
26								
27	Screens	Suppliers[1]	Sony	Vizio	Phillips			
28		LCD	150	250	200			
29		OLED	210	175	200			
30		Screens[2]						

Dimension Options

Several options will appear at the bottom of the task pane under Dimension (as shown in the screenshot above) when a Parametric Dimension is selected. Here you can view the cell where the dimension is located (B27), the formula, the name of the dimension (Suppliers) and the type of dimension (structural or parametric). These fields are all read-only. However, Current Value, Locked, Active in Optimization, Active in Simulation, and Active in Sensitivity Analysis can all be edited. Current Value controls the values displayed in the cube cells. For example, select Vizio for Current Value. The value in cell H19 will change to reflect the Vizio value for the Current Value of the Screens dimension (175).

Note: If using Analytic Solver Cloud you must first recalculate the worksheet before these changes will take effect. See the *Analytic Solver Cloud: Recalculating the Worksheet* text box above, for more information.

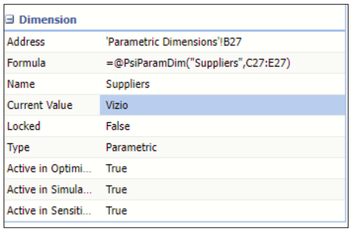
Analytic Solver Desktop



Dimension

Address	$B$27
Formula	=PsiParamDim("Suppliers",C27:E27)
Name	Suppliers
Current Value	Vizio
Locked	False
Type	Parametric
Active in Optimization	True
Active in Simulation	True
Active in Sensitivity Analy...	True

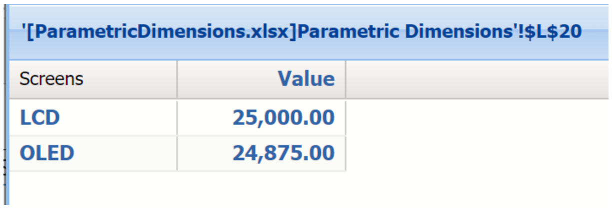
Analytic Solver Cloud



Dimension

Address	'Parametric Dimensions'!B27
Formula	=@PsiParamDim("Suppliers",C27:E27)
Name	Suppliers
Current Value	Vizio
Locked	False
Type	Parametric
Active in Optimi...	True
Active in Simula...	True
Active in Sensiti...	True

Setting Locked to True, will result in the dimension being “locked” or “frozen” to the Current Value element during optimization. For example, if Current Value is set to Sony for the Suppliers Parametric Dimension and Locked is set to True only two optimizations will be ran when the green arrow is clicked, one optimization for the value 150 (cell C28) and one optimization for the value 210 (cell C29). Double click the objective cell (L20) to display the results for the two different screens supplied by Sony.



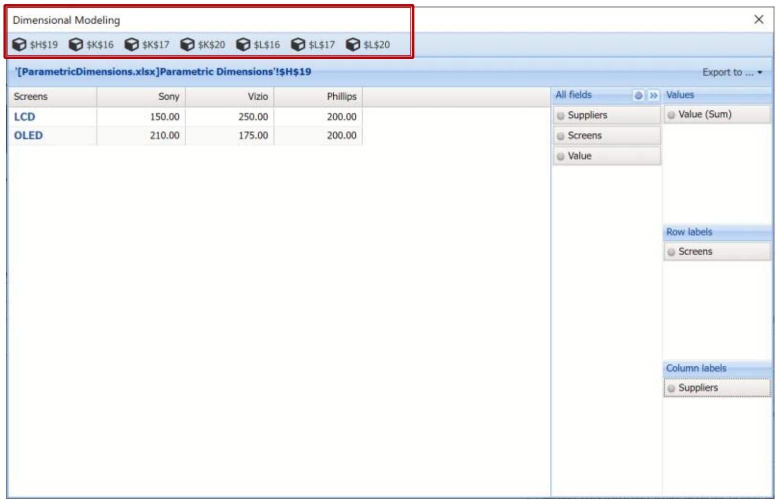
'[ParametricDimensions.xlsx]Parametric Dimensions'!$L$20

Screens	Value
LCD	25,000.00
OLED	24,875.00

Parameter Analysis

You can also reduce the dimensions displayed in the grid by performing a Parametric Analysis. Set Locked back to False and resolve the model. Then click **Reports – Optimization – Parameter Analysis** to open the Dimensional Modeling grid.

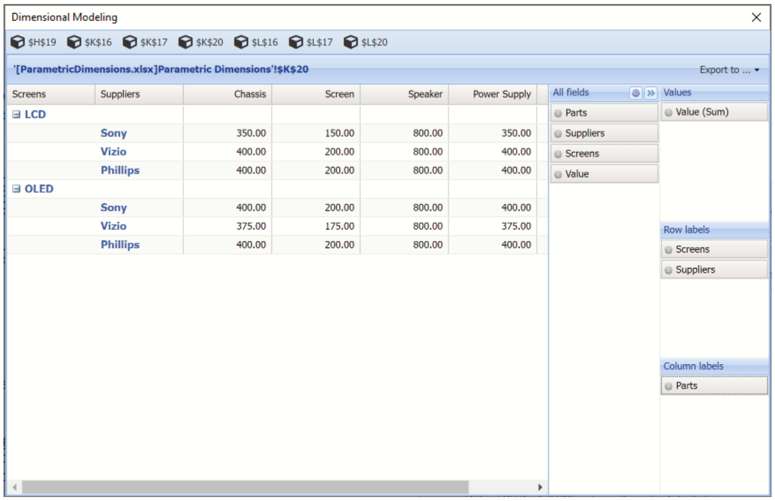
All seven cubes appear at the top of the Dimensional Modeling grid dialog. Results below are shown unsorted.



'[ParametricDimensions.xlsx]Parametric Dimensions'!$H$19

Screens	Sony	Vizio	Phillips
LCD	150.00	250.00	200.00
OLED	210.00	175.00	200.00

Here the “slices” of the structural dimensions can be viewed for each cube. Results are unsorted.



'[ParametricDimensions.xlsx]Parametric Dimensions'!$K$20

Screens	Suppliers	Chassis	Screen	Speaker	Power Supply
LCD					
	Sony	350.00	150.00	800.00	350.00
	Vizio	400.00	200.00	800.00	400.00
	Phillips	400.00	200.00	800.00	400.00
OLED					
	Sony	400.00	200.00	800.00	400.00
	Vizio	375.00	175.00	800.00	375.00
	Phillips	400.00	200.00	800.00	400.00

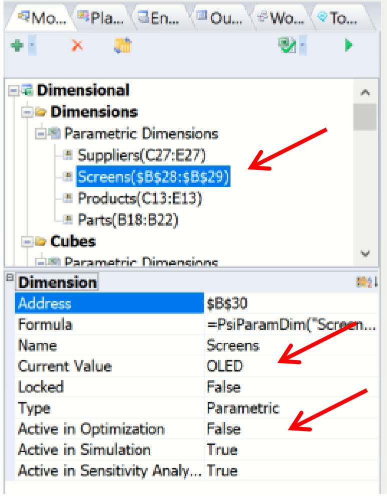
This grid displays the number of parts required when Screens are purchased from each Supplier. (Results shown are unsorted.) For example, if LCD screens are purchased from Sony, only 350 chassis will be needed. If LCD screens are purchased from Vizio or Phillips, 400 chassis will be required.

Active in Optimization, Simulation or Sensitivity

Currently both the Screens and Suppliers Parametric Dimensions are active in all three types of analyses: optimization, simulation and sensitivity. As stated earlier, this means that if you wanted to use these dimensions for a parameter analysis, you could do so as an optimization parameter, sensitivity parameter, and/or simulation parameter. To remove this Parametric dimension from any of these analyses, simply change True to False. If Active in Optimization is set to False, this dimension will be ignored in the next optimization. Likewise, if Active in Simulation or Active in Sensitivity Analysis is set to False, then this dimension will be ignored in the next simulation or sensitivity analysis.

For example, select **Screens** in the Model Task Pane, select **OLED** for *Current Value*, then change *Active in Optimization* to **False**.

Analytic Solver Desktop

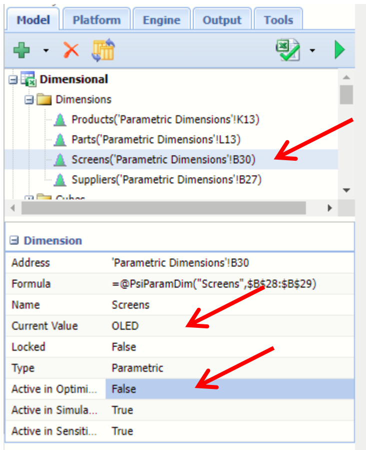


Dimensional
- **Dimensions**
  - Parametric Dimensions
    - Suppliers(C27:E27)
    - Screens($B$28:$B$29)
    - Products(C13:E13)
    - Parts(B18:B22)
- **Cubes**
  - Parametric Dimensions

Dimension

Address	$B$30
Formula	=PsiParamDim("Screen...
Name	Screens
Current Value	OLED
Locked	False
Type	Parametric
Active in Optimization	False
Active in Simulation	True
Active in Sensitivity Analy...	True

Add Cloud screenshot



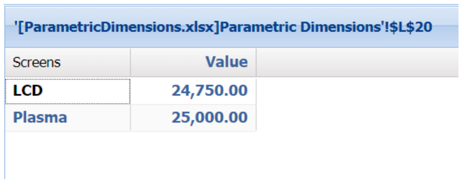
Dimensional
- Dimensions
  - Products('Parametric Dimensions'!K13)
  - Parts('Parametric Dimensions'!L13)
  - Screens('Parametric Dimensions'!B30)
  - Suppliers('Parametric Dimensions'!B27)
- Cubes

Dimension

Address	'Parametric Dimensions'!B30
Formula	=@PsiParamDim("Screens",$B$28:$B$29)
Name	Screens
Current Value	OLED
Locked	False
Type	Parametric
Active in Optimi...	False
Active in Simula...	True
Active in Sensiti...	True

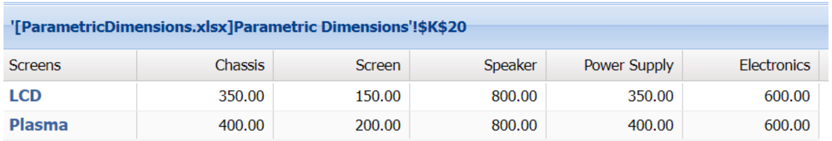
Now run an optimization either by clicking the green arrow on the Task Pane or **Optimize – Solve Complete Problem** on the ribbon.

Solver will run three optimizations – one each for values 150, 250, and 200 or the number of LCD screens (the first element in the Screens dimension) that can be supplied by Sony, Vizio and Phillips, respectively (cells C28:E28). To view the results of each value, simply select Suppliers on the Model Task Pane and select Sony, Vizio, or Phillips or double click any of the cubes to display the values for these three optimizations. For example, click cell L20 and K20 to display the following grids. (Note that sorting has been disabled.) Both grids only reveal the optimization results for the three suppliers calculated using the first element from the Screens dimension. (Results shown below are unsorted.)



'[ParametricDimensions.xlsx]Parametric Dimensions'!$L$20

Screens	Value
LCD	24,750.00
Plasma	25,000.00

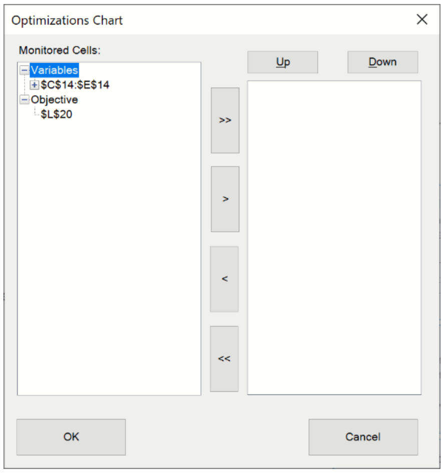


'[ParametricDimensions.xlsx]Parametric Dimensions'!$K$20

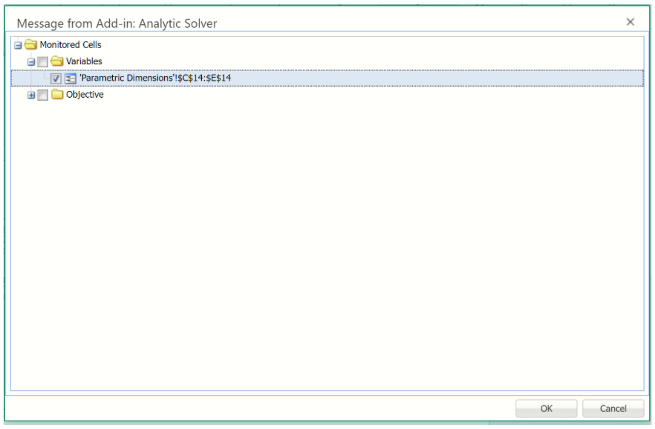
Screens	Chassis	Screen	Speaker	Power Supply	Electronics
LCD	350.00	150.00	800.00	350.00	600.00
Plasma	400.00	200.00	800.00	400.00	600.00

After an optimization has finished, click **Charts – Multiple Optimizations – Monitored Cells** to open the Optimizations Chart dialog.

Analytic Solver Desktop



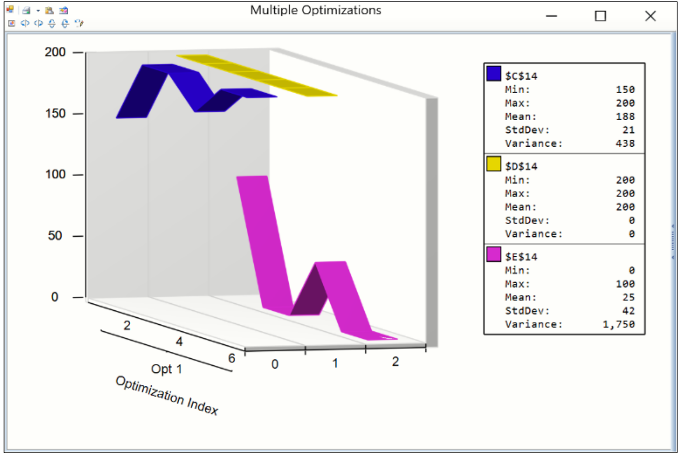
Analytic Solver Cloud



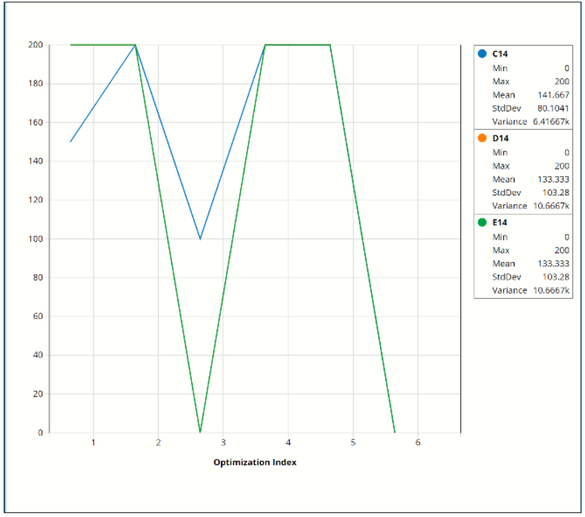
Message from Add-in: Analytic Solver
Monitored Cells
☐ Variables
☑ 'Parametric Dimensions'!$C$14:$E$14
☐ Objective
OK
Cancel

Expand Variables (by clicking the "+"), then expand C14:E14. Select **C14**, then click > to add the decision variable to the chart. Repeat this step for cells D14 and E14, then click OK to create the chart below, which tracks the decision variable values for each of the three optimizations.

Analytic Solver Desktop



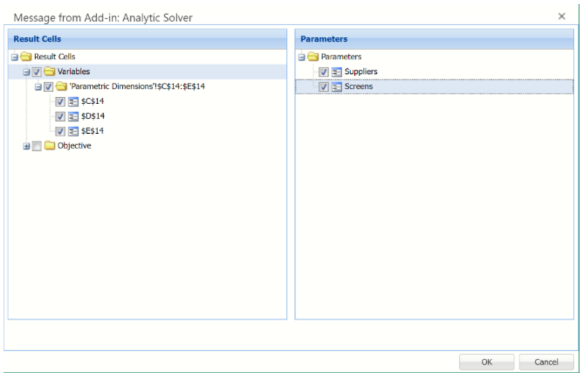
Analytic Solver Cloud



Run a parameterized analysis by clicking **Charts – Multiple Optimizations – Parameter Analysis**.

In Analytic Solver Cloud, select the variable cells under Results Cells and both parameters, Supplies and Screens. Then click OK.

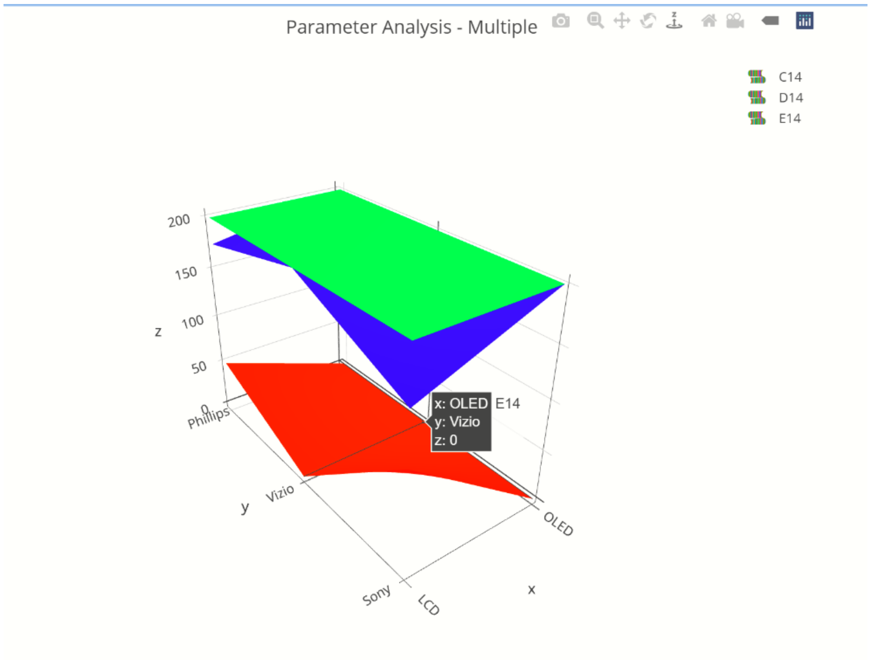
Analytic Solver Cloud



Result Cells
- Variables ☑
  - 'Parametric Dimensions'!$C$14:$E$14 ☑
    - ☑ $C$14
    - ☑ $D$14
    - ☑ $E$14
- ☐ Objective

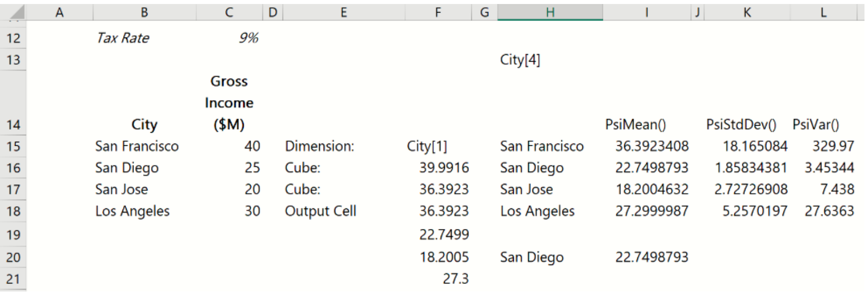
Parameters
- ☑ Suppliers
- ☑ Screens

The chart may be “grabbed” and moved in any direction. Hover over any point in the chart to view the x, y, and z coordinates.



Using Dimensional Modeling in Simulation

Dimensional modeling can also be used with simulation models as well as with simulation optimization models. (Note: Dimensional Modeling is not available with Stochastic Transformation.) Open the example, CityCube.xlsx by clicking **Help – Example Models** on the Analytic Solver ribbon. The Frontline Example Models Overview opens. Select Monte Carlo Simulation Examples and choose City Cube from the list.



	A	B	C	D	E	F	G	H	I	J	K	L
12		*Tax Rate*	*9%*									
13								City[4]				
14		**City**	**Gross Income ($M)**						PsiMean()		PsiStdDev()	PsiVar()
15		San Francisco	40		Dimension:	City[1]		San Francisco	36.3923408		18.165084	329.97
16		San Diego	25		Cube:	39.9916		San Diego	22.7498793		1.85834381	3.45344
17		San Jose	20		Cube:	36.3923		San Jose	18.2004632		2.72726908	7.438
18		Los Angeles	30		Output Cell	36.3923		Los Angeles	27.2999987		5.2570197	27.6363
19						22.7499						
20						18.2005		San Diego	22.7498793			
21						27.3						

Defining a Dimension

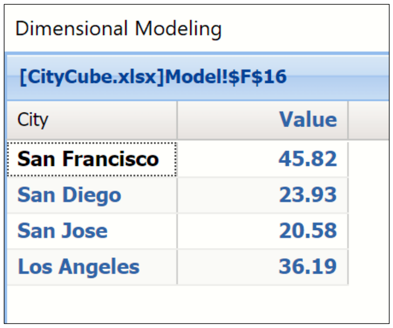
In this simple example, a dimension, City, has been created in cell F15 which is comprised of the names of major cities in California (cells B15:B18). Cells C15:C18 contain the gross income per city. Each Gross Income is uncertain. The Gross Income for San Francisco follows the Gamma distribution, Gross Income for San Diego follows a Triangular distribution, San Jose follows a Normal distribution and Los Angeles follows a Uniform distribution.

Defining a Cube

One key benefit of this model is that it defines a simulation output in one cell which would require four cells without dimensional modeling. In cell F16, a cube has been created along the City dimension which is comprised of each city's gross income in $Millions.

- In Analytic Solver Desktop, double click this cell to display the grid.
- In Analytic Solver Cloud, double click the F16 entry on the Model tab of the Solver Task Pane.

(Sorting has been disabled in this grid view.)



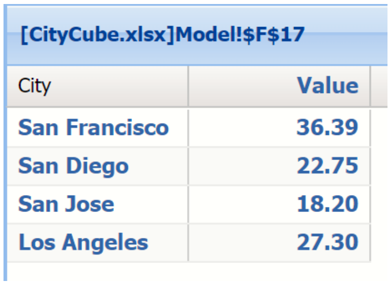
Dimensional Modeling

[CityCube.xlsx]Model!$F$16

City	Value
San Francisco	45.82
San Diego	23.93
San Jose	20.58
Los Angeles	36.19

In cell F17, a 2nd cube has been created by using the cube in cell F16 in the formula, =F16*(1-C12), which multiplies the elements of the cube in cell F16 by the after tax rate or 91% (1-.09). If dimensional modeling is not used, four output cells would be required to capture the simulation statistics, percentiles, etc, for the after-tax income of each city. However, in this example, the cube in cell F17 captures all relevant information. Results shown are unsorted.

- In Analytic Solver Desktop, click cell F17 to open the grid.
- In Analytic Solver Cloud, click cell F17 under Cubes in the Solver Task Pane, to open the grid.



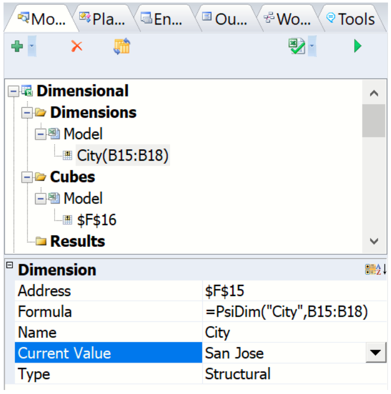
[CityCube.xlsx]Model!$F$17

City	Value
San Francisco	36.39
San Diego	22.75
San Jose	18.20
Los Angeles	27.30

Defining an Output

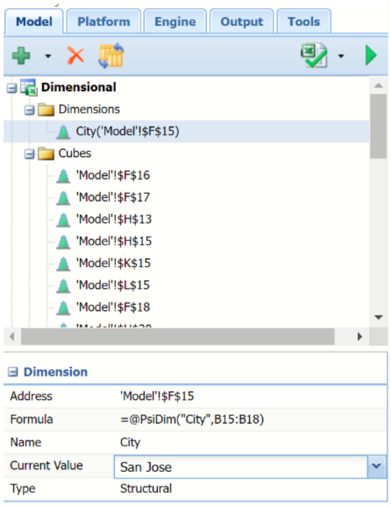
Cell F18 contains the PsiMean() function which creates both a simulation output as well as a dimensional modeling output. To view the statistics and percentiles for each city's Gross Income, simply set the Current Value of the City dimension to the desired city. For example, to view the simulation results for San Diego, select City under Dimensions in the Model tab of the Task Pane, then set Current Value to San Jose.

Analytic Solver Desktop



Address	$F$15
Formula	=PsiDim("City",B15:B18)
Name	City
Current Value	San Jose
Type	Structural

Analytic Solver Cloud



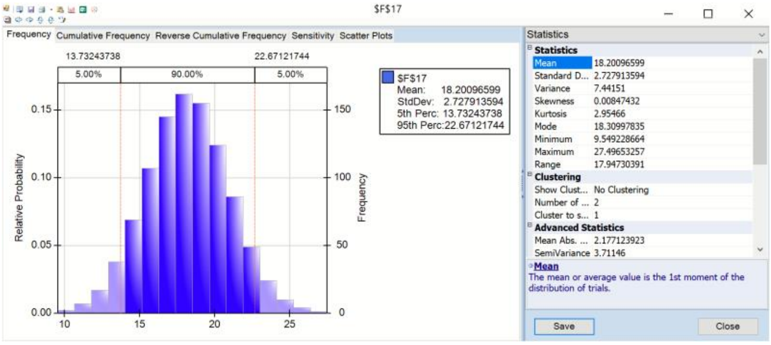
Address	'Model'!$F$15
Formula	=@PsiDim("City",B15:B18)
Name	City
Current Value	San Jose
Type	Structural

Notice that Analytic Solver Cloud lists all intermediary cubes under "Cubes" in the task pane while Analytic Solver Desktop lists only the cell containing the actual PsiCube() formula in cell F16.

Running a Simulation

Now click the green arrow on the Output tab to run a simulation. The Simulation results window that appears contains simulation results for San Jose's after-tax income.

Note: The uncertain function dialog will contain simulation results based on the Current Value of the dimension(s) contained in the cube. Cell F17 is a cube with one dimension, City. Therefore, the simulation results displayed in the uncertain function dialog are controlled by the Current Value selected for the City dimension.



$F$17
Mean: 18.20096599
StdDev: 2.727913594
5th Perc: 13.73243738
95th Perc:22.67121744

Statistics

Mean	18.20096599
Standard D...	2.727913594
Variance	7.44151
Skewness	0.00847432
Kurtosis	2.95466
Mode	18.30997835
Minimum	9.549228664
Maximum	27.49653257
Range	17.94730391
Clustering	
Show Clust...	No Clustering
Number of ...	2
Cluster to s...	1
Advanced Statistics	
Mean Abs. ...	2.177123923
SemiVariance	3.71146

Mean
The mean or average value is the 1st moment of the distribution of trials.

lists

Note: You can also bring up this dialog by clicking F17 on the Model tab of the Solver Task Pane (beneath Uncertain Functions).

Displaying Simulation Results

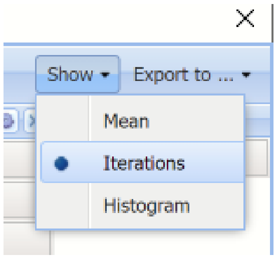
Close the uncertain function dialog, then …

- Analytic Solver Desktop: double click cell F17

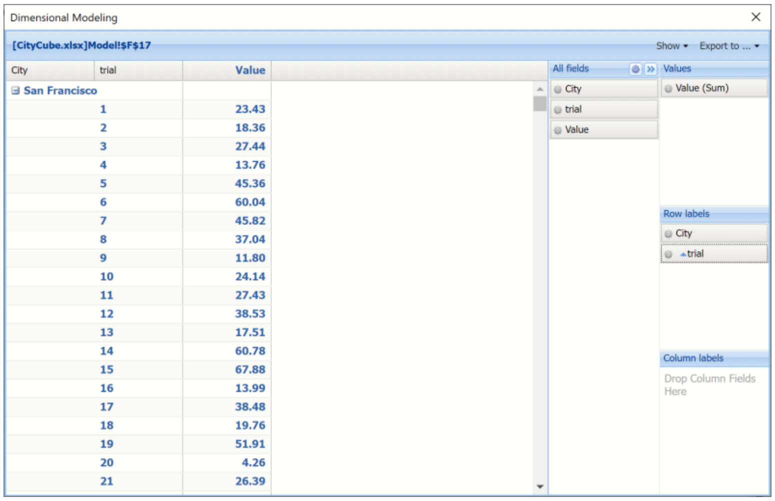
- Analytic Solver Cloud: double click F17 on the Model tab of the Solver Task Pane (beneath Cubes)

... to bring up the dimensional modeling grid, shown below.

Click the down arrow next to "Show" on the top of the dialog and select Iterations.



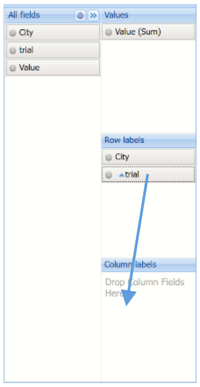
… to view each of the 1,000 simulation trials for each of the cities.

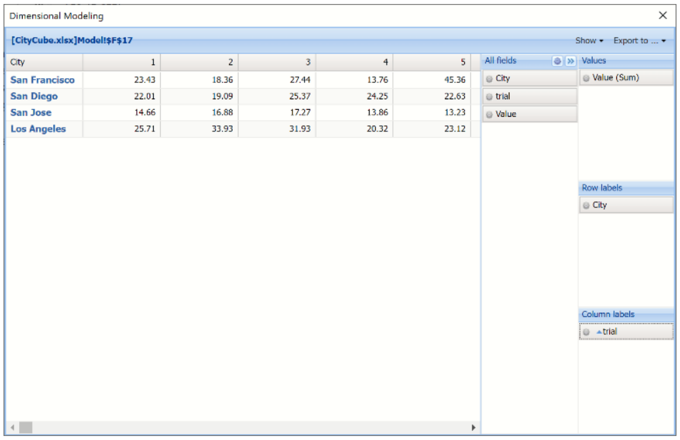


[CityCube.xlsx]Model!$F$17

City	trial	Value
San Francisco		
	1	23.43
	2	18.36
	3	27.44
	4	13.76
	5	45.36
	6	60.04
	7	45.82
	8	37.04
	9	11.80
	10	24.14
	11	27.43
	12	38.53
	13	17.51
	14	60.78
	15	67.88
	16	13.99
	17	38.48
	18	19.76
	19	51.91
	20	4.26
	21	26.39

Drag "trial" to "Column labels" to display the trial values in a different layout.

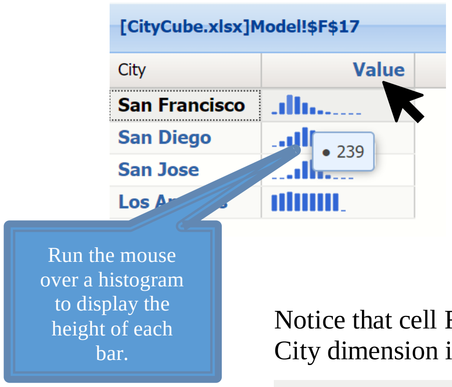




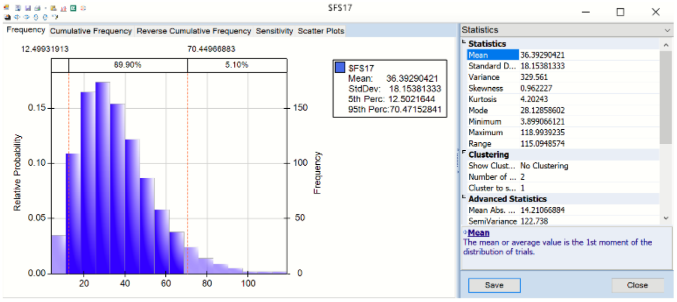
[CityCube.xlsx]Model!$F$17

City	1	2	3	4	5
San Francisco	23.43	18.36	27.44	13.76	45.36
San Diego	22.01	19.09	25.37	24.25	22.63
San Jose	14.66	16.88	17.27	13.86	13.23
Los Angeles	25.71	33.93	31.93	20.32	23.12

Click the down arrow next to Show again but this time select "Histogram" to bring up the simulation result histograms by city. Double click the histogram for any of the listed cities, say San Francisco, to bring up the simulation results for the city of San Francisco.



Run the mouse over a histogram to display the height of each bar.



$F$17
Mean: 36.39290421
StdDev: 18.15381333
5th Perc: 12.5021644
95th Perc:70.47152841

Statistics

Mean	36.39290421
Standard D...	18.15381333
Variance	329.561
Skewness	0.962227
Kurtosis	4.20243
Mode	28.12858602
Minimum	3.899066121
Maximum	118.9939235
Range	115.0948574

Clustering

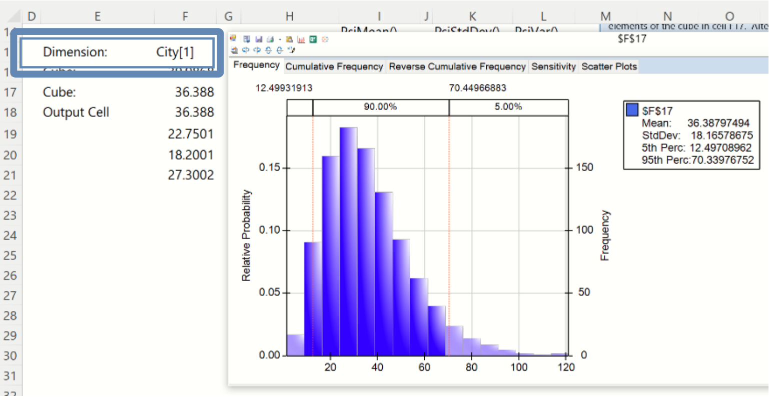
Show Clust...	No Clustering
Number of ...	2
Cluster to s...	1

Advanced Statistics

Mean Abs. ...	14.21066884
SemiVariance	122.738

Mean
The mean or average value is the 1st moment of the distribution of trials.

Notice that cell F15 displays "City[1]" to indicate that the first dimension of the City dimension is being displayed.



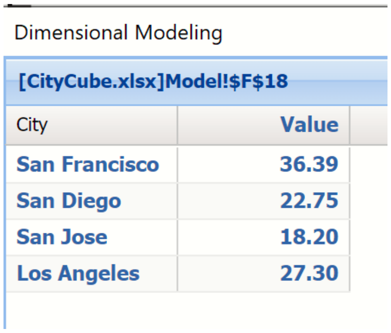
	Dimension:	City[1]
17	Cube:	36.388
18	Output Cell	36.388
19		22.7501
20		18.2001
21		27.3002

$F$17
Mean: 36.38797494
StdDev: 18.16578675
5th Perc: 12.49708962
95th Perc:70.33976752

To view the mean results for each city,

- Analytic Solver Desktop: Double click cell F18 to bring up the following grid.
- Analytic Solver Cloud: Double click F18 in the task pane (beneath Cubes) to bring up the following grid.

(Sorting has been disabled in the screenshot below.)



Dimensional Modeling

[CityCube.xlsx]Model!$F$18

City	Value
San Francisco	36.39
San Diego	22.75
San Jose	18.20
Los Angeles	27.30

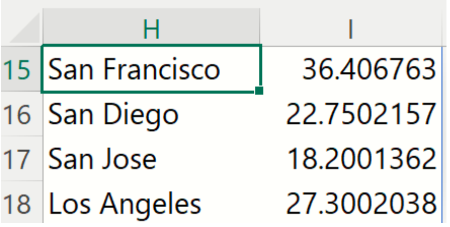
Note: PsiMean() can be substituted with any PsiXxx statistic function except PsiCorrelation and PsiFrequency.

Cells F18:F21 contain the PsiMean() simulation statistic entered as an array formula. The results of this function are similar to the results of PsiCubeData() and PsiOptData() (for objective and constraint cubes only) in that when entered as an array, the function will display the fact table for the desired cube.

Displaying Statistic Values

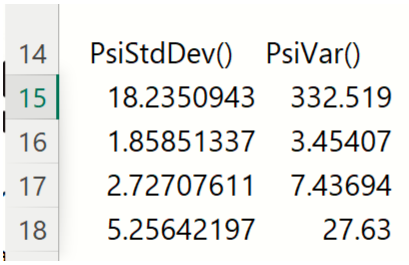
When a Psi Statistic function (such as PsiMean(), PsiStdDev(), PsiVariance(), etc.) is used with Dimensional Modeling in Analytic Solver , the statistic function will display all elements of the cube fact table formatted according to the optional 3rd argument, *struc_format*, in much the same way as PsiOptData(). Enter "=PsiMean(F17, 1, "dims")" in cell H13. When "dims" is passed for the *struc_format* argument (3rd argument), the size of each dimension contained in the cube in cell F17 will be returned. Afterwards, this cell will display "City[4]" indicating that there are 4 elements in the City dimension.

Now we'll use "vals" for the *struc_format* argument. When passing "vals" for this argument, PsiMean(), when entered as an array, will display the data as a relational table. Since there are 4 elements in the City dimension, the fact table for this cube will contain 4 elements. Enter "=PsiMean(F17, 1, "vals") into cell H15. The contents of this cell will "spill" down as shown in the screenshot below. *Note: PsiMean() can be substituted with any PsiXxx statistic function except PsiCorrelation and PsiFrequency.*



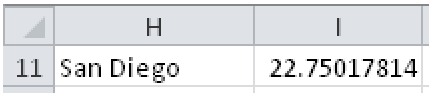
	H	I
15	San Francisco	36.406763
16	San Diego	22.7502157
17	San Jose	18.2001362
18	Los Angeles	27.3002038

Now we'll use a different statistic, PsiStdDev() but this time we'll list *only* the values in the fact table To do so, Enter "=PsiStdDev(F17)" into cell J15. The contents of this cell will "spill" down. Notice we are *only supplying the first argument*. Afterwards, these cells will display only the *values* of the fact table as shown in cells K15:K18 in the screenshot below. Repeat the same steps in cell L15 but this time using the statistic, PsiVariance() ("=PsiVariance(F17)").



14	PsiStdDev()	PsiVar()
15	18.2350943	332.519
16	1.85851337	3.45407
17	2.72707611	7.43694
18	5.25642197	27.63

A statistics function can also be used to display a slice or even a single element of the fact table. For example, to display the mean gross margin for the City of San Diego, select two blank cells, say H20:I20, and enter the following formula: =PsiMean(F17,1, "[City].[San Diego]"). The contents of this cell will "spill" down. Afterwards, the following table will be displayed.



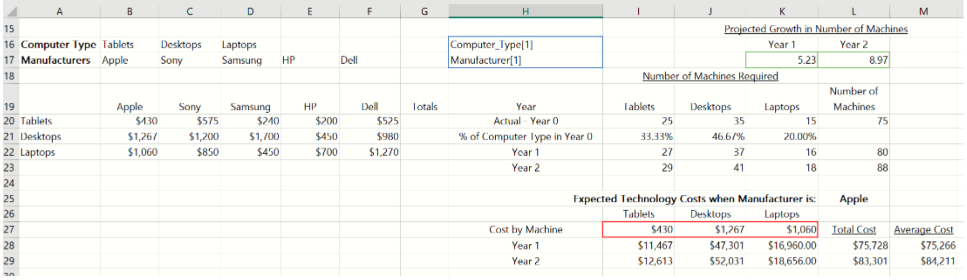
	H	I
11	San Diego	22.75017814

Now let's close this example and continue on to the next section to learn about using a parametric dimension in a simulation model.

Simulation Model with Parametric Dimensions

Now let's see how Parametric Dimensions can be used with simulation. Open the example, ParametricCube_SimExample.xlsx by clicking **Help – Example**

Models on the Analytic Solver ribbon. The Frontline Example Models Overview.xlsx file will open. Select the *Simulation* tab and choose **Parametric Cube Simulation Example.** ParametricCube_SimExample.xlsx opens.

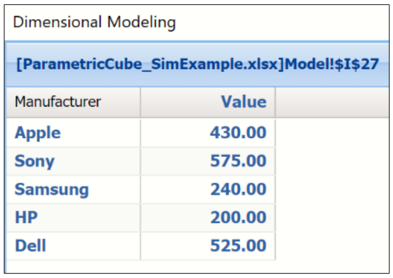


	A	B	C	D	E	F	G	H	I	J	K	L	M
15											Projected Growth in Number of Machines		
16	**Computer Type**	Tablets	Desktops	Laptops				Computer_Type[1]			Year 1	Year 2	
17	**Manufacturers**	Apple	Sony	Samsung	HP	Dell		Manufacturer[1]			5.23	8.97	
18									Number of Machines Required				
19		Apple	Sony	Samsung	HP	Dell	Totals	Year	Tablets	Desktops	Laptops	Number of Machines	
20	Tablets	$430	$575	$240	$200	$525		Actual - Year 0	25	35	15	75	
21	Desktops	$1,267	$1,200	$1,700	$450	$980		% of Computer Type in Year 0	33.33%	46.67%	20.00%		
22	Laptops	$1,060	$850	$450	$700	$1,270		Year 1	27	37	16	80	
23								Year 2	29	41	18	88	
24													
25											**Expected Technology Costs when Manufacturer is:**	**Apple**	
26									Tablets	Desktops	Laptops		
27								Cost by Machine	$430	$1,267	$1,060	Total Cost	Average Cost
28								Year 1	$11,467	$47,301	$16,960.00	$75,728	$75,266
29								Year 2	$12,613	$52,031	$18,656.00	$83,301	$84,211

This example models the Estimated Technology costs for two years in the future (year 1 and year 2) for JoeBen Gaming, based on which computer manufacturer the company selects. This model includes a structural dimension and a parametric dimension. The structural dimension is held in cell H16. This dimension contains the type of technology that JoeBen gaming will purchase for their employees: Tablets, Desktops, and Laptops. The parametric dimension is contained in cell H17. This dimension contains the manufacturers of the technology: Apple, Sony, Samsung, HP, and Dell.

Cells L22 and L23 contain the projected number of machines needed for years 1 and 2, respectively, based on the projected growth distributions in cells K17 and L17. The ratio of tablets, desktops, and laptops, to the total number of machines will remain constant. Therefore, the percentage of tablets will remain at 33%, the percentage of desktops will remain at 46.7%, and the percentage of laptops will remain at 20% for the next two years. As a result, the values in cells I28:K29 are calculated by multiplying the number of machines (cells L22 and L23) by the percentages in cells I21:K21.

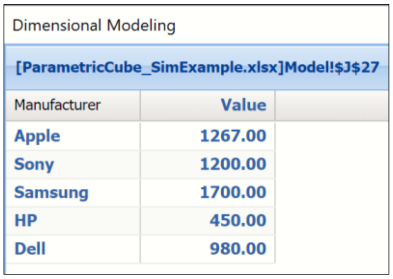
Three cubes are located in cells I27:K27. These cubes contain the data for the Manufacturer dimension. The values in these cells will determine the total technology costs for each year. Cell I27 contains pricing data for each tablet by manufacturer, cell J27 contains pricing data for each desktop by manufacturer, and cell K27 contains pricing data for each laptop by manufacturer. Double click each of these cells in Analytic Solver Desktop or the cell entries in the Model tab of the Solver Task Pane in Analytic Solver Cloud, to display their data values. Results shown are unsorted.



Dimensional Modeling

[ParametricCube_SimExample.xlsx]Model!$I$27

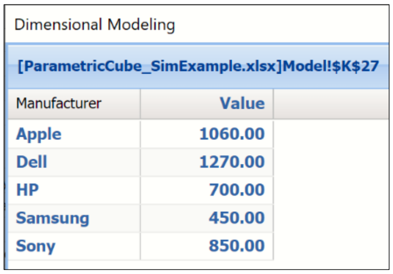
Manufacturer	Value
Apple	**430.00**
Sony	**575.00**
Samsung	**240.00**
HP	**200.00**
Dell	**525.00**



Dimensional Modeling

[ParametricCube_SimExample.xlsx]Model!$J$27

Manufacturer	Value
Apple	**1267.00**
Sony	**1200.00**
Samsung	**1700.00**
HP	**450.00**
Dell	**980.00**



Dimensional Modeling

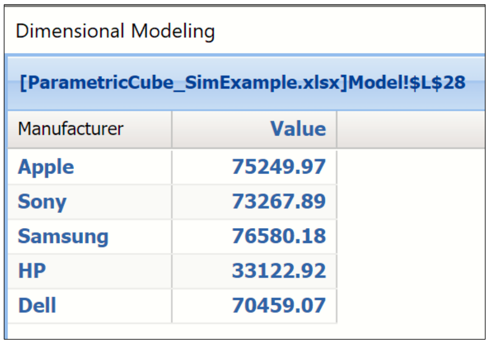
[ParametricCube_SimExample.xlsx]Model!$K$27

Manufacturer	Value
Apple	**1060.00**
Dell	**1270.00**
HP	**700.00**
Samsung	**450.00**
Sony	**850.00**

Once the green arrow has been clicked in the Model tab of the Solver task pane, or Simulate – Run Once is clicked on the ASP Ribbon, **five** different simulations will be run simultaneously in *less than one second* – one simulation for each manufacturer.

- Open the grid for cell L28 in Analytic Solver Desktop, by double clicking cell L28 to display a grid revealing the total technology costs for Year 1 *for each manufacturer*.
- To open the same grid in Analytic Solver Cloud, click Reports – Parameter Analysis, and click the L28 tab.

With Dimensional Modeling, you can perform a What If analysis with the double click of one button!

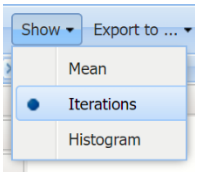


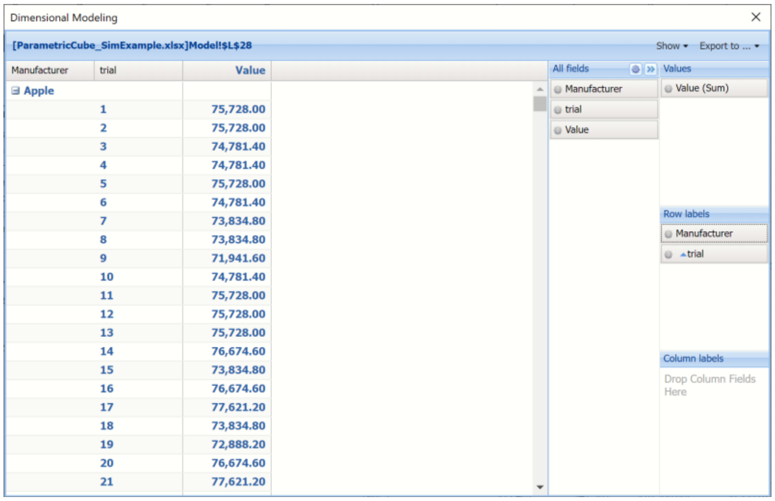
Dimensional Modeling

[ParametricCube_SimExample.xlsx]Model!$L$28

Manufacturer	Value
Apple	**75249.97**
Sony	**73267.89**
Samsung	**76580.18**
HP	**33122.92**
Dell	**70459.07**

Click the down arrow next to "Show" and select Trials from the drop-down menu.





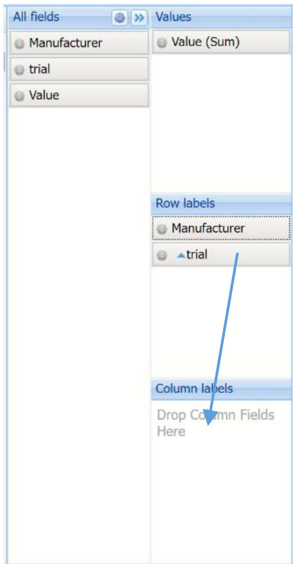
Dimensional Modeling

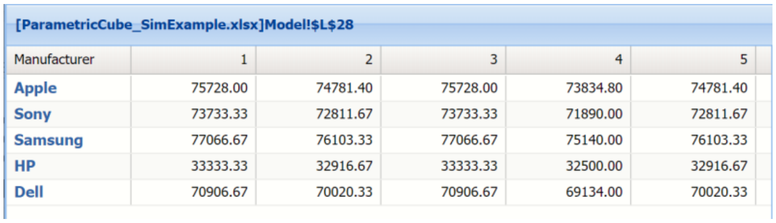
[ParametricCube_SimExample.xlsx]Model!$L$28

Manufacturer	trial	Value
Apple		
	1	75,728.00
	2	75,728.00
	3	74,781.40
	4	74,781.40
	5	75,728.00
	6	74,781.40
	7	73,834.80
	8	73,834.80
	9	71,941.60
	10	74,781.40
	11	75,728.00
	12	75,728.00
	13	75,728.00
	14	76,674.60
	15	73,834.80
	16	76,674.60
	17	77,621.20
	18	73,834.80
	19	72,888.20
	20	76,674.60
	21	77,621.20

1000 trial values are displayed for each manufacturer. Row labels are unsorted.

Drag "trial" to "Column labels" to display iterations from left to right.

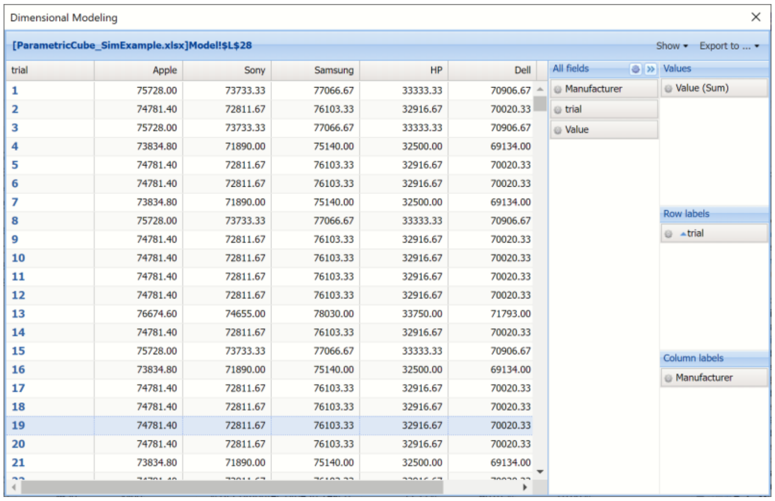




[ParametricCube_SimExample.xlsx]Model!$L$28

Manufacturer	1	2	3	4	5
Apple	75728.00	74781.40	75728.00	73834.80	74781.40
Sony	73733.33	72811.67	73733.33	71890.00	72811.67
Samsung	77066.67	76103.33	77066.67	75140.00	76103.33
HP	33333.33	32916.67	33333.33	32500.00	32916.67
Dell	70906.67	70020.33	70906.67	69134.00	70020.33

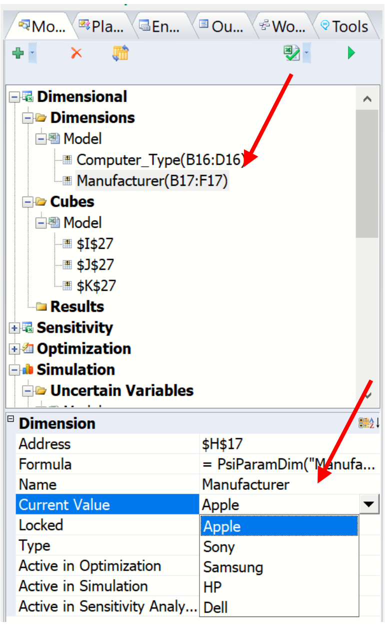
To view the trial values vertically (rather than horizontally), simply drag **Trials** to **Row labels** and **Manufacturer** to **Column labels**.



trial	Apple	Sony	Samsung	HP	Dell
1	75728.00	73733.33	77066.67	33333.33	70906.67
2	74781.40	72811.67	76103.33	32916.67	70020.33
3	75728.00	73733.33	77066.67	33333.33	70906.67
4	73834.80	71890.00	75140.00	32500.00	69134.00
5	74781.40	72811.67	76103.33	32916.67	70020.33
6	74781.40	72811.67	76103.33	32916.67	70020.33
7	73834.80	71890.00	75140.00	32500.00	69134.00
8	75728.00	73733.33	77066.67	33333.33	70906.67
9	74781.40	72811.67	76103.33	32916.67	70020.33
10	74781.40	72811.67	76103.33	32916.67	70020.33
11	74781.40	72811.67	76103.33	32916.67	70020.33
12	74781.40	72811.67	76103.33	32916.67	70020.33
13	76674.60	74655.00	78030.00	33750.00	71793.00
14	74781.40	72811.67	76103.33	32916.67	70020.33
15	75728.00	73733.33	77066.67	33333.33	70906.67
16	73834.80	71890.00	75140.00	32500.00	69134.00
17	74781.40	72811.67	76103.33	32916.67	70020.33
18	74781.40	72811.67	76103.33	32916.67	70020.33
19	74781.40	72811.67	76103.33	32916.67	70020.33
20	74781.40	72811.67	76103.33	32916.67	70020.33
21	73834.80	71890.00	75140.00	32500.00	69134.00

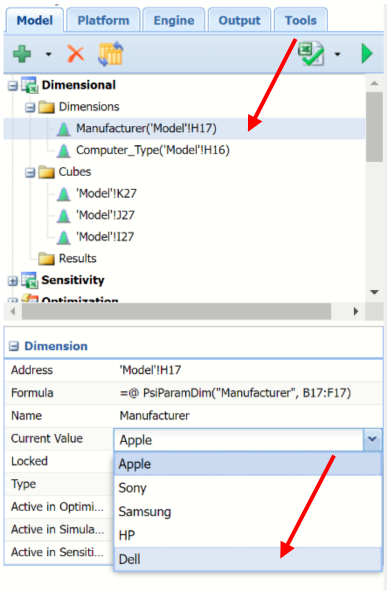
Let’s view the results for Dell only. First, select the **Manufacturer** parametric dimension in the Model tab of the Solver task pane. Then select **Dell** for *Current Value*.

Analytic Solver Desktop



Address	$H$17
Formula	= PsiParamDim("Manufa...
Name	Manufacturer
Current Value	Apple
Locked	
Type	
Active in Optimization	
Active in Simulation	
Active in Sensitivity Analy...	

Analytic Solver Cloud

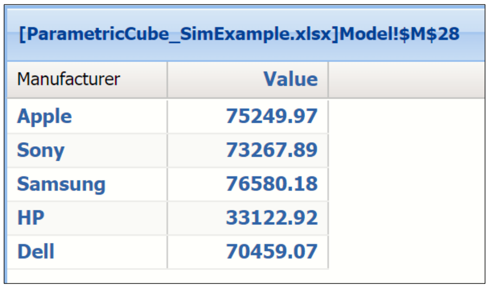


Address	'Model'!H17
Formula	=@ PsiParamDim("Manufacturer", B17:F17)
Name	Manufacturer
Current Value	Apple
Locked	
Type	
Active in Optimi...	
Active in Simula...	
Active in Sensiti...	

The values in cells I27:K27 display $525, $980 and $1,270, respectively, which equals the value for Dell in cells F20:F22.

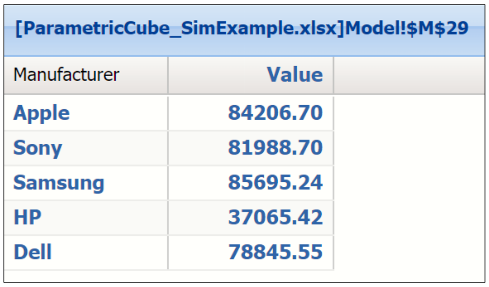
The PsiMean() functions in cells M28 and M29 display the projected mean technology costs for years 1 and 2, respectively, when computers manufactured by Dell are purchased.

- To open in Analytic Solver Desktop, double click cells M28 and M29 respectively.
- This functionality is not supported in Analytic Solver Cloud.



[ParametricCube_SimExample.xlsx]Model!$M$28

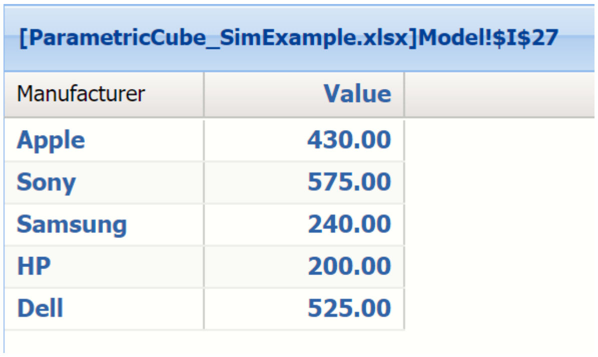
Manufacturer	Value
Apple	75249.97
Sony	73267.89
Samsung	76580.18
HP	33122.92
Dell	70459.07



[ParametricCube_SimExample.xlsx]Model!$M$29

Manufacturer	Value
Apple	84206.70
Sony	81988.70
Samsung	85695.24
HP	37065.42
Dell	78845.55

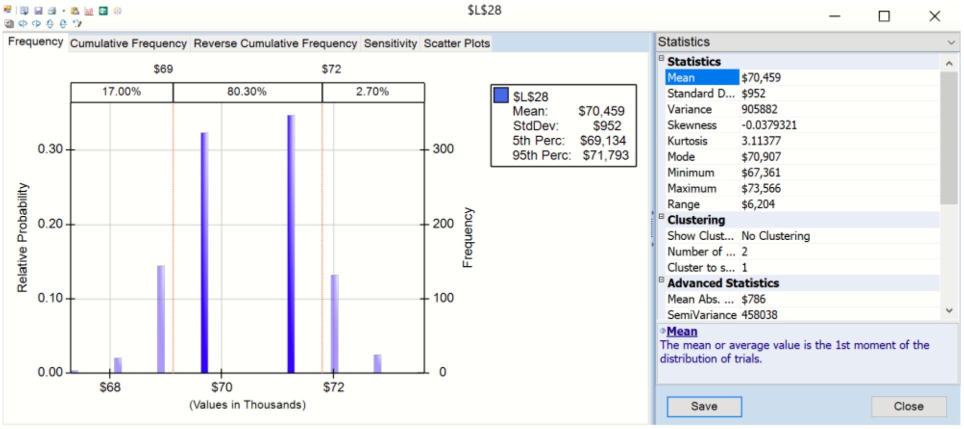
Note: The cubes in cells I27:K27 will display the tablet, desktop, or laptop values by Manufacturer. For example, open the cube in cell I27 to open the following grid which displays the tablet costs for Year 1 for each manufacturer.



[ParametricCube_SimExample.xlsx]Model!$I$27

Manufacturer	Value
Apple	430.00
Sony	575.00
Samsung	240.00
HP	200.00
Dell	525.00

Double click L28 in the Model tab of the Solver Task Pane to display the simulation results for technology costs for Year 1, when Dell is selected as the manufacturer.

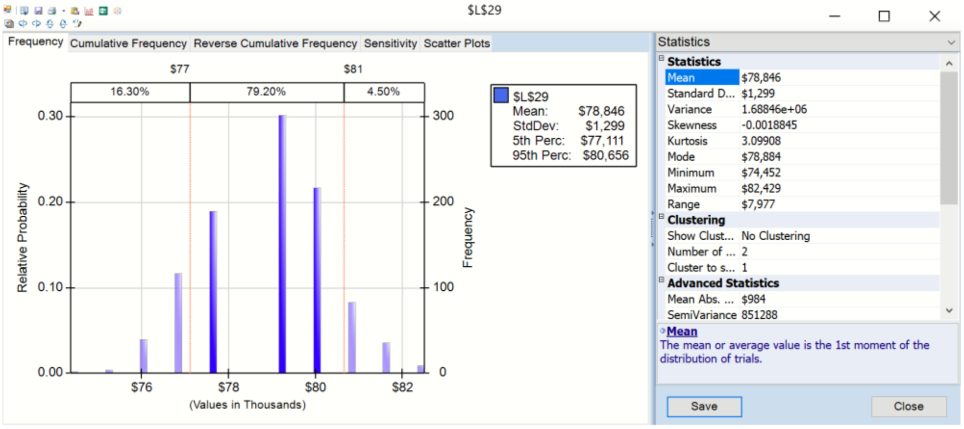


Statistics

Mean	$70,459
Standard D...	$952
Variance	905882
Skewness	-0.0379321
Kurtosis	3.11377
Mode	$70,907
Minimum	$67,361
Maximum	$73,566
Range	$6,204
Clustering	
Show Clust...	No Clustering
Number of ...	2
Cluster to s...	1
Advanced Statistics	
Mean Abs. ...	$786
SemiVariance	458038

Mean
The mean or average value is the 1st moment of the distribution of trials.

Then double click cell L29 to display the simulation results for Year 2, when Dell is selected as Current Value for the Manufacturer dimension.

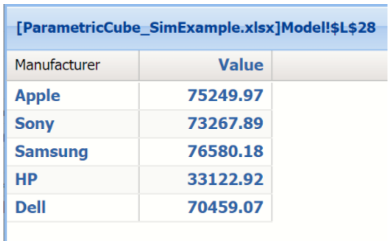


Statistics	
Mean	$78,846
Standard D...	$1,299
Variance	1.68846e+06
Skewness	-0.0018845
Kurtosis	3.09908
Mode	$78,884
Minimum	$74,452
Maximum	$82,429
Range	$7,977
Clustering	
Show Clust...	No Clustering
Number of ...	2
Cluster to s...	1
Advanced Statistics	
Mean Abs. ...	$984
SemiVariance	851288

Mean
The mean or average value is the 1st moment of the distribution of trials.

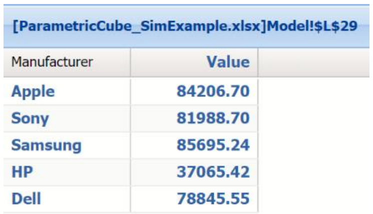
Note: To obtain simulation results for a different manufacturer, select the desired Manufacturer dimension in the Model tab of the Solver task pane, select the desired manufacturer for the Current Value field, then double click L28 or L29 on the Model tab of the Solver task pane. The uncertain function dialog will update to display the current element of the Manufacturer dimension.

- If using Analytic Solver Desktop, double click cells L28 and L29 to open the dimensional modeling grid. (Results are unsorted.)
- If using Analytic Solver Cloud, click Reports – Parameter Analysis and open the L28 and L29 tabs.



[ParametricCube_SimExample.xlsx]Model!$L$28

Manufacturer	Value
Apple	75249.97
Sony	73267.89
Samsung	76580.18
HP	33122.92
Dell	70459.07



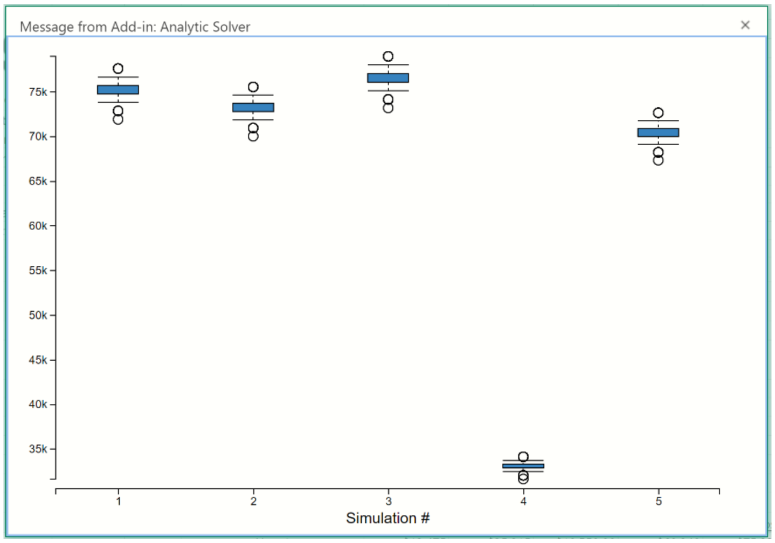
[ParametricCube_SimExample.xlsx]Model!$L$29

Manufacturer	Value
Apple	84206.70
Sony	81988.70
Samsung	85695.24
HP	37065.42
Dell	78845.55

As you can easily see, purchasing machines from HP will yield the least amount of technology costs for both Years 1 and 2.

All charts under Multiple Simulations now also support Dimensional Modeling. (“Locked” or non-active dimensions will not appear in the charts.) Click **Charts – Multiple Simulations – Box-Whisker** to open the Multiple Simulations Chart dialog. Highlight Simulation 1 and click >, then click OK. Analytic Solver draws the following chart.

Screenshot taken from Analytic Solver Cloud

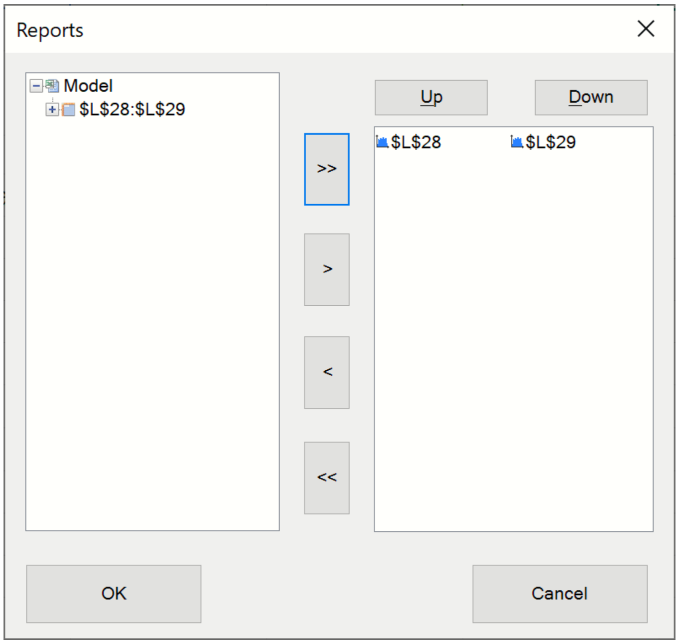


This shows a Box-Whisker diagram for each of the 15 simulations, with the mean, median, 25th and 75th percentiles, and the minimum and maximum value for net revenue on each one.

In Analytic Solver Desktop, click the right edge of the Trend and Box-Whisker charts to open a right panel with options for customizing these charts. You can use the icons on the title toolbar to print the chart, or copy it to the Clipboard, where it can be pasted into another application such as PowerPoint.

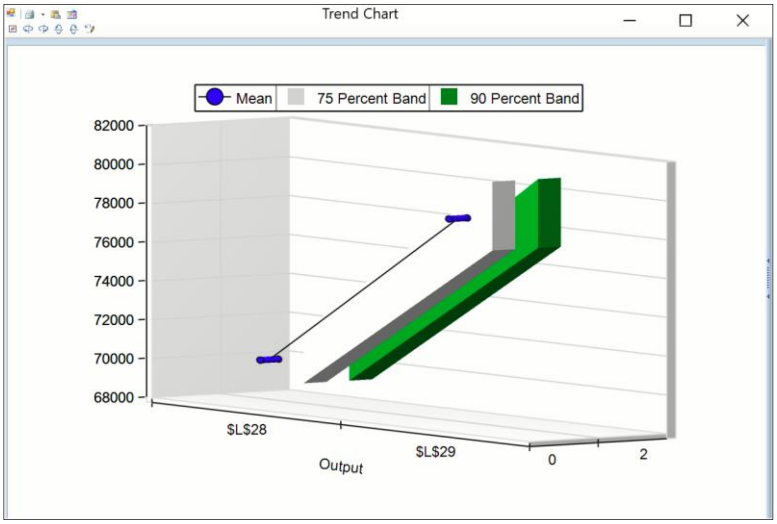
Perform the same steps for each of the remaining charts: Trend, Overlay, and Statistics. For more information on these charts, please see the Frontline Solvers Reference Guide.

In addition, the three reports under Charts – Multiple Simulation Results are also supported in Dimensional Modeling. Click **Charts – Multiple Simulation Results -- Trend** to open the Multiple Simulation Results dialog.



Click >> to select both output cells L28 and L29, then click **OK**. A trend chart of both uncertain functions is produced.

Screenshot taken from Analytic Solver Desktop



Here we can see a steep upward climb of the mean value of the total technology costs, and also the volatility around the mean, depicted here with the 25th and 75th percentiles.

Repeat the same steps to produce the remaining charts: Overlay and Box-Whisker. For an explanation of these reports, please see the Frontline Solvers Reference Guide.

Now close this example and continue onto the next section to learn how to use a pivot table with Dimensional Modeling.

Using Dimensional Modeling with Pivot Tables

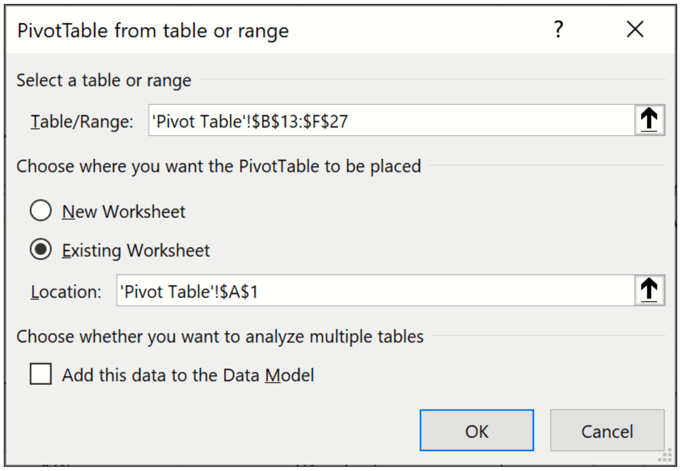
In general, cubes are defined as multi-dimensional, not just two – dimensional as shown in the examples above. Unfortunately, a normal Excel worksheet is not the easiest medium for working with multidimensional cubes. An Excel pivot table, however, lends itself to this very practice. An Excel Pivot Table is a tool within Excel that can be used to summarize data located in an Excel worksheet. Once data is inserted into a pivot table, it can easily be sorted, counted, totaled, etc. by dragging and dropping the fields into the desired layout.

Click **Help – Example Models** on the Analytic Solver ribbon. The Frontline Example Models Overview opens. Click Monte Carlo Simulation Examples and then **Pivot Table Simulation Example** from the list. (This example uses parametric dimensions but structural dimensions with Pivot Tables.)

An Excel Pivot table is located in cells B13:F27. This table is comprised of the Product Line for Mark's Breakfast Delights. This company has two product lines: Waffle Mix and Pancake Mix. Each of these product lines contain two flavors: blueberry and plain. The Sales team is comprised of 4 individuals who are responsible for a specific region of the Las Vegas area where the company is based: Henderson, Summerlin, North Las Vegas, and Fremont Street. Sales figures for each product and region are located in the Sales column. *Note: Since values in a pivot table cannot be evaluated, pivot table values must be constants.*

How to create the Pivot Table

Highlight the data on the worksheet, then click **Insert – Pivot Table – From Table/Range** on the Excel ribbon. The Create PivotTable dialog opens. *Select a table or range* is prefilled, leave this selection as is. Select Existing **Worksheet**, then select a blank cell for Location. Then click OK.



PivotTable from table or range

Select a table or range

Table/Range: 'Pivot Table'!$B$13:$F$27

Choose where you want the PivotTable to be placed

○ New Worksheet

● Existing Worksheet

Location: 'Pivot Table'!$A$1

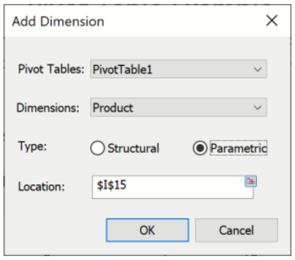
Choose whether you want to analyze multiple tables

☐ Add this data to the Data Model

The pivot table will be inserted into the existing worksheet. Manipulate the table as desired. Afterwards, click **Design – Report Layout – Show in Tabular Form**.

Next, we'll create three dimensions: *Product*, *Product_Line*, and *Region*. Select a blank cell, in this case cell I15 was used, then click **Model – Dimension – From Pivot Table** if using Analytic Solver Desktop or **Model – Dimensional Modeling – Dimension – From Pivot Table** if using Analytic Solver Cloud, to open the *Add Dimension* dialog. Select the name of the pivot table, in this case **PivotTable1**, select **Product** for *Dimensions*, select **Parametric** for *Type* and cell **I15** for *Location*. Then click **OK**. The formula =PsiPivotDim(B13,"Product", TRUE) will be inserted into cell I15. Repeat these steps in cells I16 and I17 to create parametric dimensions for Product_Line and Region, respectively. Alternatively, these formulas can also be typed directly into the cell.

Analytic Solver Desktop



Add Dimension

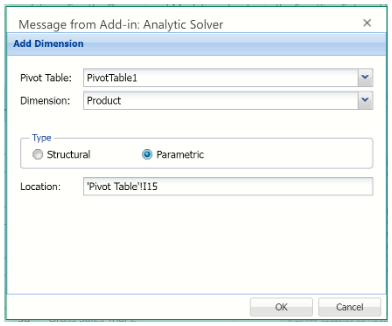
Pivot Tables: PivotTable1

Dimensions: Product

Type: ○ Structural ● Parametric

Location: $I$15

Analytic Solver Cloud



Message from Add-in: Analytic Solver

Add Dimension

Pivot Table: PivotTable1

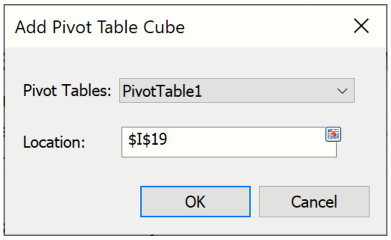
Dimension: Product

Type: ○ Structural ● Parametric

Location: 'Pivot Table'!I15

Now we will create a cube from the pivot table. Select a blank cell, in this example cell I19, then click **Model – Cube – From Pivot** Table to open the *Add Pivot Table* Cube dialog.

Analytic Solver Desktop

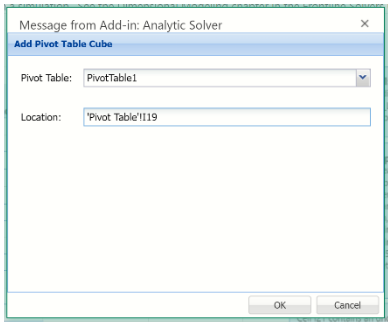


Add Pivot Table Cube

Pivot Tables: PivotTable1

Location: $I$19

Analytic Solver Cloud



Message from Add-in: Analytic Solver

Add Pivot Table Cube

Pivot Table: PivotTable1

Location: 'Pivot Table'!I19

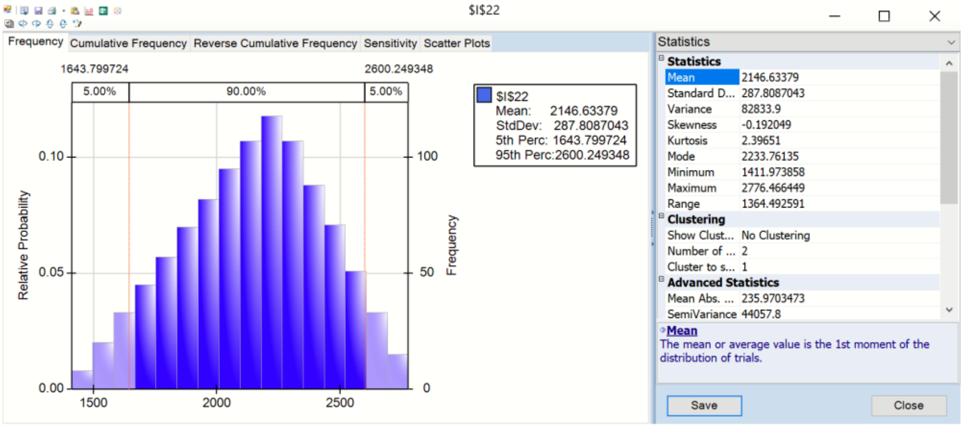
Leave PivotTable1 and Location at their defaults, and click OK. The following formula will be inserted into cell I19 =PsiPivotCube($B$13:$F27). Alternatively, this formula could have been entered into the cell directly.

From here we can create a simulation or optimization model. In this example, we've created a simulation model. Cell I21 holds an uncertain variable which calculates a selling price using a triangular distribution with a minimum parameter of 100, a maximum parameter of 200, and a likely parameter of 160. This sales price is multiplied by the cube result in cell I19.

To simulate gross sales for the Western region, select the **Product** Dimension on the Model tab in the Task Pane, then change *Current Value* to "Blueberry". Afterwards, perform the same steps for the remaining dimensions (Product_Line and Region) changing the current value of the **Product_Line** dimension to "Pancake Mix" and the current value of the **Region** dimension to "Henderson". Cell I19 will display the value in cell E15, 14. Now click the green arrow on the Model tab to run a simulation. The simulation runs using the value of 14 as the number of products sold. The simulation results for the uncertain function in cell I22 are below.

Important Note: Pivot tables cannot be evaluated by the PSI Interpreter. This means that decision variables, uncertain variables, uncertain functions, and constraint or objective functions must not be used within an Excel Pivot Table. All values inside of a pivot table must be constant, or computed from constants.

Analytic Solver Desktop



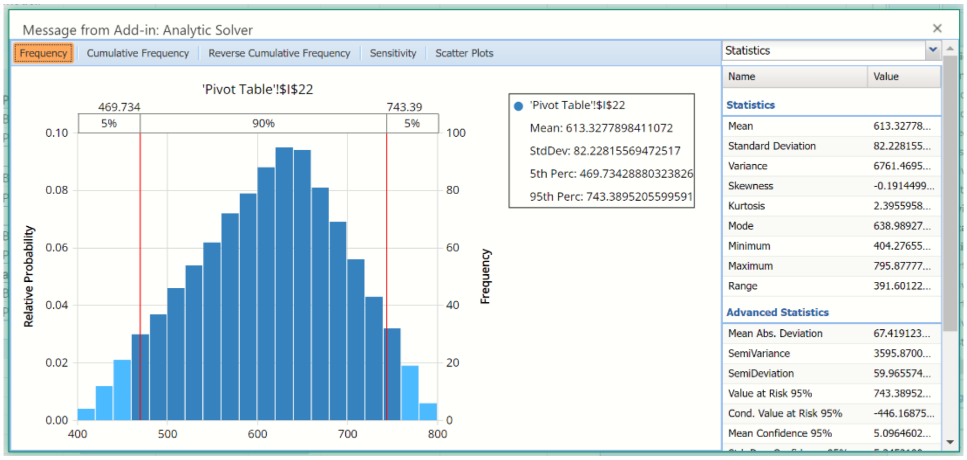
$I$22

Mean: 2146.63379
StdDev: 287.8087043
5th Perc: 1643.799724
95th Perc: 2600.249348

Statistics	
Statistics	
Mean	2146.63379
Standard D...	287.8087043
Variance	82833.9
Skewness	-0.192049
Kurtosis	2.39651
Mode	2233.76135
Minimum	1411.973858
Maximum	2776.466449
Range	1364.492591
Clustering	
Show Clust...	No Clustering
Number of ...	2
Cluster to s...	1
Advanced Statistics	
Mean Abs. ...	235.9703473
SemiVariance	44057.8

Mean
The mean or average value is the 1st moment of the distribution of trials.

Analytic Solver Cloud

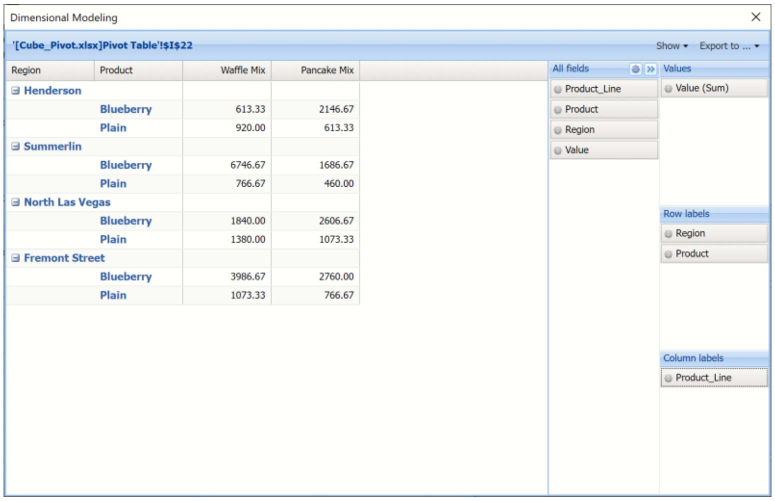


'Pivot Table'!$I$22
Mean: 613.3277898411072
StdDev: 82.22815569472517
5th Perc: 469.73428880323826
95th Perc: 743.3895205599591

Name	Value
Statistics	
Mean	613.32778...
Standard Deviation	82.228155...
Variance	6761.4695...
Skewness	-0.1914499...
Kurtosis	2.3955958...
Mode	638.98927...
Minimum	404.27655...
Maximum	795.87777...
Range	391.60122...
Advanced Statistics	
Mean Abs. Deviation	67.419123...
SemiVariance	3595.8700...
SemiDeviation	59.965574...
Value at Risk 95%	743.38952...
Cond. Value at Risk 95%	-446.16875...
Mean Confidence 95%	5.0964602...

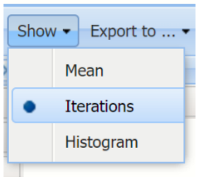
Double click the cube in cell I22 to display a grid containing the Gross Sales by product and region. Results are unsorted.

Analytic Solver Desktop

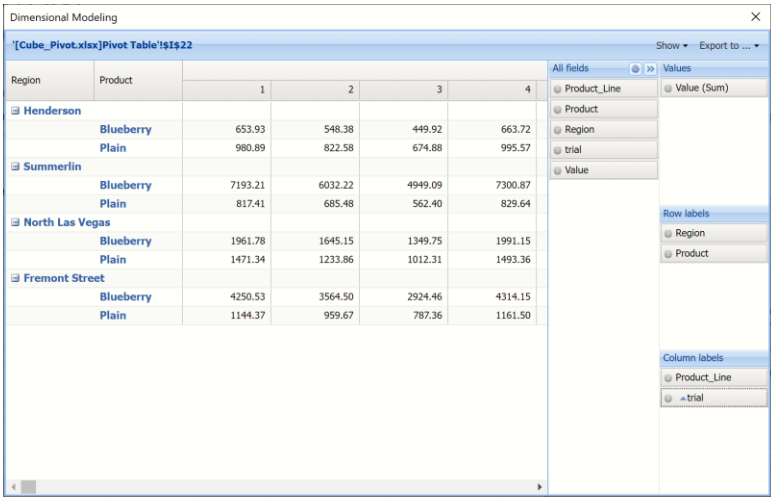


Region	Product	Waffle Mix	Pancake Mix
Henderson			
	Blueberry	613.33	2146.67
	Plain	920.00	613.33
Summerlin			
	Blueberry	6746.67	1686.67
	Plain	766.67	460.00
North Las Vegas			
	Blueberry	1840.00	2606.67
	Plain	1380.00	1073.33
Fremont Street			
	Blueberry	3986.67	2760.00
	Plain	1073.33	766.67

Click the down arrow next to *Show* and select **Iterations** to display the 1000 trial values used in the simulation.



If using Analytic Solver Desktop, double click cell I23 to open the grid containing the Mean Gross Sales by product and region. (Results are unsorted and "trial" has been moved to "Column labels".) This functionality is not supported in Analytic Solver Cloud.

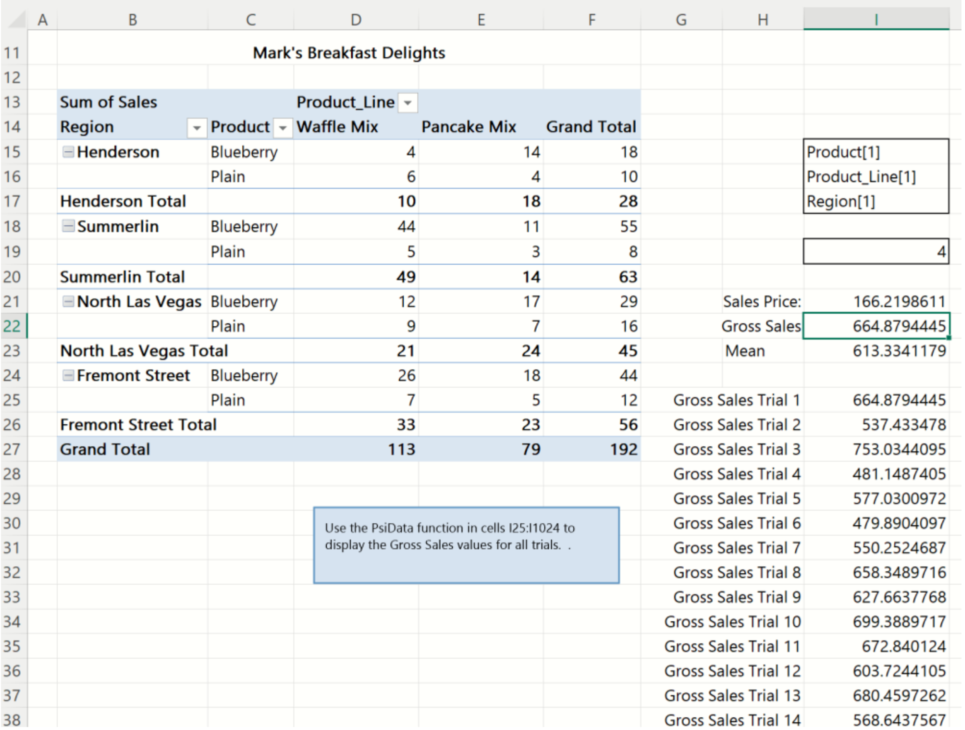


'[Cube_Pivot.xlsx]Pivot Table'!$I$22

Region	Product	1	2	3	4
Henderson					
	Blueberry	653.93	548.38	449.92	663.72
	Plain	980.89	822.58	674.88	995.57
Summerlin					
	Blueberry	7193.21	6032.22	4949.09	7300.87
	Plain	817.41	685.48	562.40	829.64
North Las Vegas					
	Blueberry	1961.78	1645.15	1349.75	1991.15
	Plain	1471.34	1233.86	1012.31	1493.36
Fremont Street					
	Blueberry	4250.53	3564.50	2924.46	4314.15
	Plain	1144.37	959.67	787.36	1161.50

It's also possible to list the trial values in cells in the spreadsheet. Cell I25 contains the function PsiData(I22). This function "spills" down to cell I1024 and displays all 1,000 trial values for cell I22, or Gross Sales.

In Analtyic Solver Cloud, PsiMean() in cell I23 will spill horizontally. Insert "@" before PsiMean() in order to prohibit the spill.



	A	B	C	D	E	F	G	H	I
11				Mark's Breakfast Delights					
12									
13		Sum of Sales		Product_Line					
14		Region	Product	Waffle Mix	Pancake Mix	Grand Total			
15		Henderson	Blueberry	4	14	18			Product[1]
16			Plain	6	4	10			Product_Line[1]
17		Henderson Total		10	18	28			Region[1]
18		Summerlin	Blueberry	44	11	55			
19			Plain	5	3	8			4
20		Summerlin Total		49	14	63			
21		North Las Vegas	Blueberry	12	17	29		Sales Price:	166.2198611
22			Plain	9	7	16		Gross Sales	664.8794445
23		North Las Vegas Total		21	24	45		Mean	613.3341179
24		Fremont Street	Blueberry	26	18	44			
25			Plain	7	5	12		Gross Sales Trial 1	664.8794445
26		Fremont Street Total		33	23	56		Gross Sales Trial 2	537.433478
27		Grand Total		113	79	192		Gross Sales Trial 3	753.0344095
28								Gross Sales Trial 4	481.1487405
29								Gross Sales Trial 5	577.0300972
30								Gross Sales Trial 6	479.8904097
31								Gross Sales Trial 7	550.2524687
32								Gross Sales Trial 8	658.3489716
33								Gross Sales Trial 9	627.6637768
34								Gross Sales Trial 10	699.3889717
35								Gross Sales Trial 11	672.840124
36								Gross Sales Trial 12	603.7244105
37								Gross Sales Trial 13	680.4597262
38								Gross Sales Trial 14	568.6437567

Use the PsiData function in cells I25:I1024 to display the Gross Sales values for all trials. .

Using Sparse Cubes

Analytic Solver Desktop allows sparsity in a cube to be exploited thereby saving memory and gaining execution speed. *Note: This feature is not supported in Analytic Solver Cloud.*

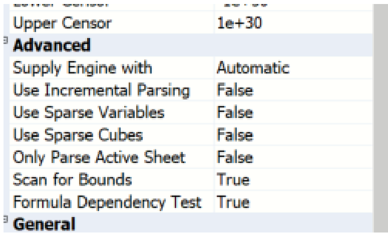
Most large cubes are *sparse* in nature. While they may contain thousands of elements, in practice, not all combinations of dimension elements are possible;

therefore, not all will define a model function during the Psi Interpreter's evaluation of the problem. This means that most cubes will provoke output results as sparse cubes (with missing constraints). Such sparsity in a cube (also known as structural sparsity) can be exploited to save memory and gain speed.

A sparse cube is defined by missing *values* in cells for PsiCube() and by missing *records* for PsiTableCube().

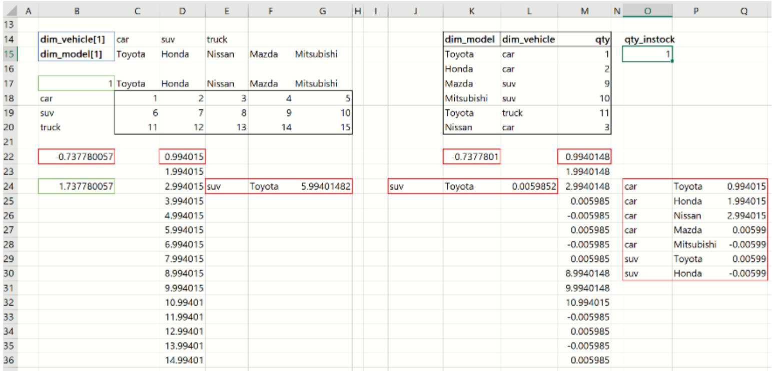
The Platform option *Use Sparse Cubes*, listed under Advanced, is used to determine if missing elements in a cube will be replaced by zero values. If this option is set to True, you have defined a cube using PsiCube() with missing values or PsiTableCube() with missing records, *and* the percentage of elements missing or empty is more than 30% of the total possible cube elements, those missing elements or records will not be included in the model. If this option is set to False, and you have defined a cube using PsiCube() or PsiTableCube(), elements missing from the cube will be considered equal to 0.

Analytic Solver Desktop



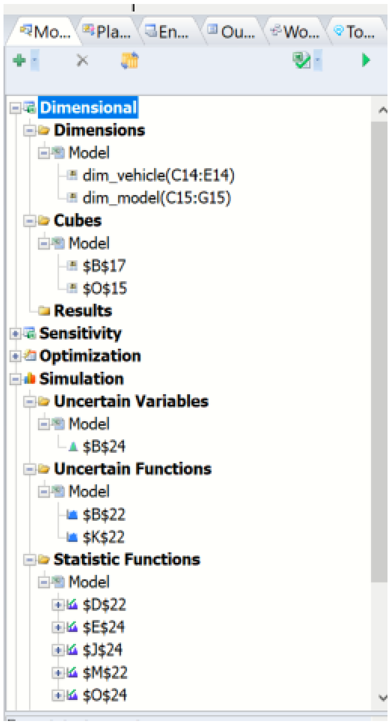
Upper Censor	1e+30
Advanced	
Supply Engine with	Automatic
Use Incremental Parsing	False
Use Sparse Variables	False
Use Sparse Cubes	False
Only Parse Active Sheet	False
Scan for Bounds	True
Formula Dependency Test	True
General	

To illustrate this new feature, we will use the newly added example, **Sparse_Cubes.xlsx**. To open, click **Help – Examples** on the Analytic Solver Desktop ribbon, click the *Simulation* tab, then the link to **Sparse_Cubes.xlsx**.

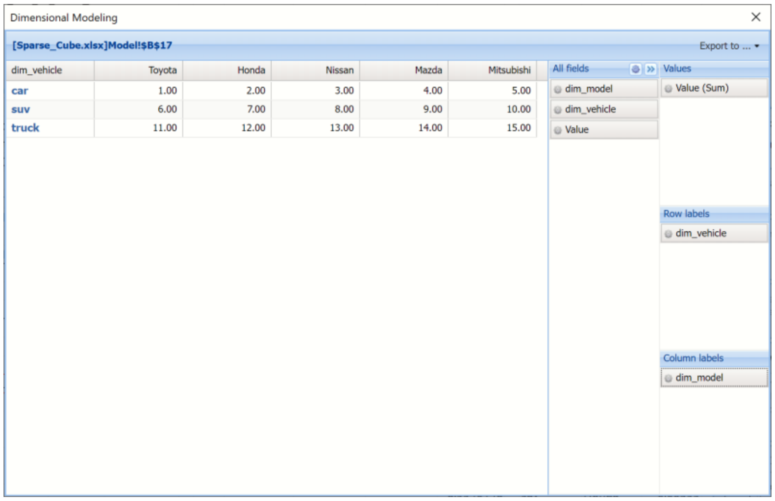


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
13																	
14		**dim_vehicle[1]**	car	suv	truck						**dim_model**	**dim_vehicle**	**qty**		**qty_instock**		
15		**dim_model[1]**	Toyota	Honda	Nissan	Mazda	Mitsubishi				Toyota	car	1		1		
16											Honda	car	2				
17		1	Toyota	Honda	Nissan	Mazda	Mitsubishi				Mazda	suv	9				
18		car	1	2	3	4	5				Mitsubishi	suv	10				
19		suv	6	7	8	9	10				Toyota	truck	11				
20		truck	11	12	13	14	15				Nissan	car	3				
21																	
22		-0.737780057		0.994015							-0.7377801		0.9940148				
23				1.994015									1.9940148				
24		1.737780057		2.994015	suv	Toyota	5.99401482			suv	Toyota	0.0059852	2.9940148		car	Toyota	0.994015
25				3.994015									0.005985		car	Honda	1.994015
26				4.994015									-0.005985		car	Nissan	2.994015
27				5.994015									0.005985		car	Mazda	0.00599
28				6.994015									-0.005985		car	Mitsubishi	-0.00599
29				7.994015									0.005985		suv	Toyota	0.00599
30				8.994015									8.9940148		suv	Honda	-0.00599
31				9.994015									9.9940148				
32				10.99401									10.994015				
33				11.99401									-0.005985				
34				12.99401									0.005985				
35				13.99401									-0.005985				
36				14.99401									0.005985				

Two dimensions have been created in cells B14 and B15, dim_vehicle and dim_model, respectively. The dim_vehicle dimension (cell B14) contains three items or elements: car, suv, and truck. The dim_model dimension (cell B15) contains five items or elements: Toyota, Honda, Nissan, Mazda, and Mitsubishi. Both dimensions together specify the number of cars, SUVs and trucks available on a car lot, organized by manufacturer.



One cube is defined using PsiCube() in cell B17 = PsiCube(C18:G20, "dim_model", "dim_vehicle"). The first argument is the cell address of the data, "C18:G20". The second and third arguments are the dimensions for which the data will be calculated. To display the contents of the cube, double-click cell B17. Note that the results in the screenshot below are unsorted.



dim_vehicle	Toyota	Honda	Nissan	Mazda	Mitsubishi
car	1.00	2.00	3.00	4.00	5.00
suv	6.00	7.00	8.00	9.00	10.00
truck	11.00	12.00	13.00	14.00	15.00

Cell B24 contains the PsiNormal distribution with mean equal to 0, and standard deviation equal to 1. This is a simple error term that will be added to each cube element.

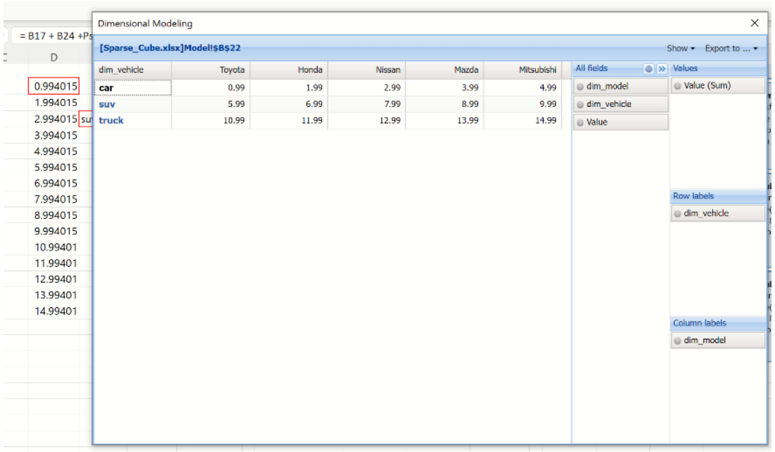
The formula in cell B22 (B22 = B17 + B24 + PsiOutput()) adds the error term (that follows the PsiNormal distribution) to each element in the cube in cell B17 (i.e., the first element of the cube = 1 + PsiNormal(0,1), the second element of the cube = 2 + PsiNormal(0, 1) and so on). PsiOutput is used to designate cell

B22 as an output function. For details on the PsiOuput() function see the previous chapter, "Examples: Stochastic Optimization."

PsiMean() has been entered into cell D22. After a simulation has been run, this formula will "spill" down to cell B36, displaying the PsiMean() value as calculated for each of the 15 elements of the cube (i.e., cell D22 will contain the mean for the first cube element, D23 will contain the mean for the second cube element, and so on).

The formula, = PsiMean(B22, 1, "[dim_vehicle].[suv], [dim_model].[maz]"), has been entered into cell E24 to illustrate how to display just one element of the cube (i.e., the number of Toyota SUVs available). This formula "spills" across cells E24:G24.

Run a simulation by clicking the down arrow beneath Simulate and selecting **Run Once**. Instantly, Analytic Solver Desktop performs a new simulation of 1,000 Monte Carlo trials, and updates each of the 15 cube elements and the worksheet with the results. The mean of each element can be found in cells D22:D36 or by double clicking cell B22, as shown in the screenshot below. (Note that the results in the screenshot below are unsorted.)



[Sparse_Cube.xlsx]Model!$B$22

dim_vehicle	Toyota	Honda	Nissan	Mazda	Mitsubishi
car	0.99	1.99	2.99	3.99	4.99
suv	5.99	6.99	7.99	8.99	9.99
truck	10.99	11.99	12.99	13.99	14.99

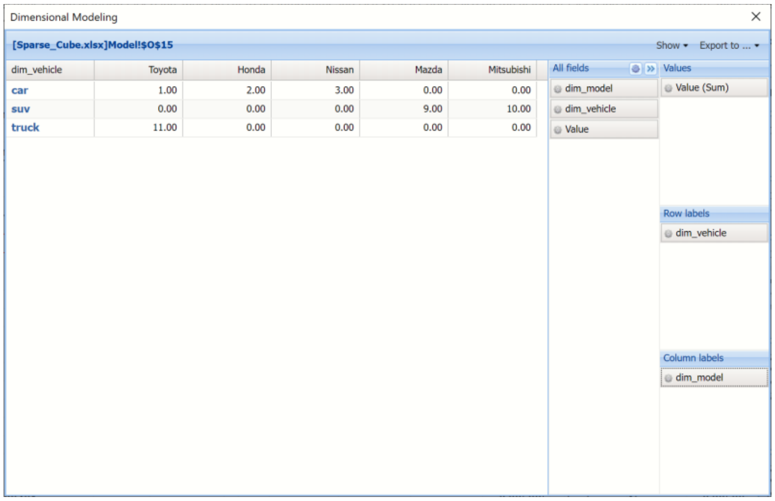
0.994015
1.994015
2.994015
3.994015
4.994015
5.994015
6.994015
7.994015
8.994015
9.994015
10.99401
11.99401
12.99401
13.99401
14.99401

Notice that the value of the first cube element (Toyota/Car) corresponds to the value in cell D22, the value of the second cube element (Honda/Car) corresponds to the value in cell D23, and so on. In addition, the value for the number of available Toyota SUVs in cell G24 corresponds to the number of Toyota SUVs in the cube display.

Next, let's move to the right of the spreadsheet. Cell O15 contains PsiTableCube(). PsiTableCube() defines a *sparse* table. The first argument passes the dimension elements, while the second argument passes the values. (For more information on PsiTableCube, see PsiTableCube() in the Frontline Solvers Reference Guide.)

Notice that only *6 out of a possible 15* elements are used in the table. When **Use Sparse Cubes** is set to **False** (the default), the missing 9 elements will be set to 0 during the Psi Interpreter's evaluation of the problem. When **Use Sparse Cubes** is set to **True**, these missing 9 elements *will not be included* in the model.

To display the cube, double-click cell O15. (Note: Results below shown unsorted.)



Dimensional Modeling

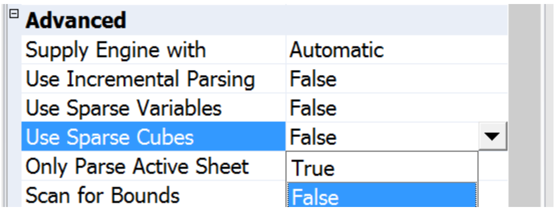
[Sparse_Cube.xlsx]Model!$O$15

dim_vehicle	Toyota	Honda	Nissan	Mazda	Mitsubishi
car	1.00	2.00	3.00	0.00	0.00
suv	0.00	0.00	0.00	9.00	10.00
truck	11.00	0.00	0.00	0.00	0.00

PsiMean has been entered into cell M22. Cell K22 contains the formula "= B24 + O15 + PsiOutput()", which adds an error term to each element in the cube table. (This formula re-uses the PsiNormal() distribution in cell B24 as the error term).

Use Sparse Cubes = False

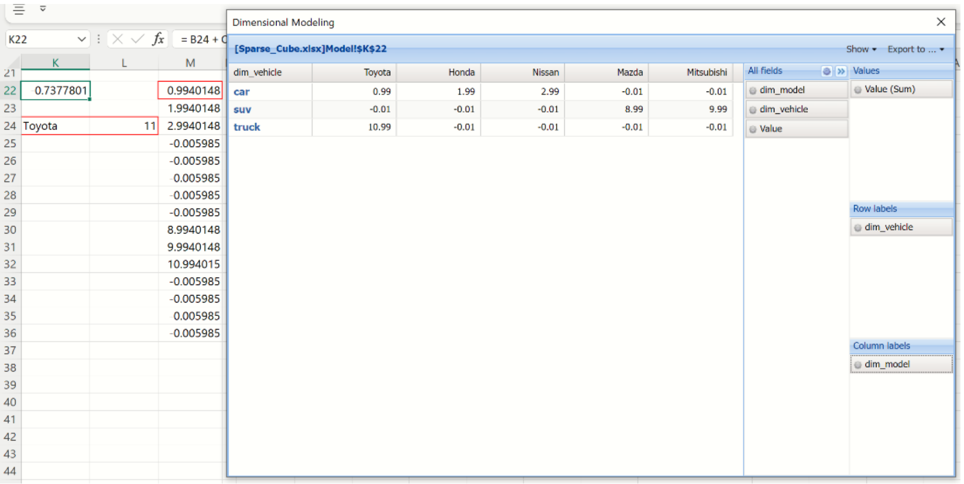
With **Use Sparse Cubes = False** on the Platform tab on the Solver Task Pane, run a simulation by clicking the down arrow beneath Simulate and selecting Run Once.



Advanced

Supply Engine with	Automatic
Use Incremental Parsing	False
Use Sparse Variables	False
Use Sparse Cubes	False
Only Parse Active Sheet	
Scan for Bounds	

Instantly, Analytic Solver Desktop performs a new simulation of 1,000 Monte Carlo trials, and updates each of the 15 cube elements and the worksheet with the results. The mean of each element can be found in cells M22:M36 or by double-clicking cell K22, as shown in the screenshot below.



K22 = B24 + C

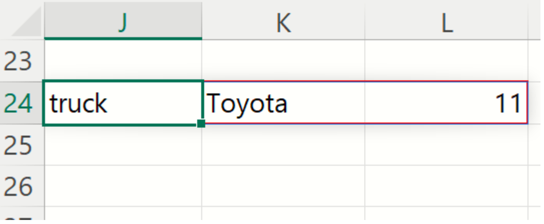
	K	L	M
21			
22	-0.7377801		0.9940148
23			1.9940148
24	Toyota	11	2.9940148
25			-0.005985
26			-0.005985
27			-0.005985
28			-0.005985
29			-0.005985
30			8.9940148
31			9.9940148
32			10.994015
33			-0.005985
34			-0.005985
35			-0.005985
36			-0.005985
37			
38			
39			
40			
41			
42			
43			
44			

Dimensional Modeling

[Sparse_Cube.xlsx]Model!$K$22

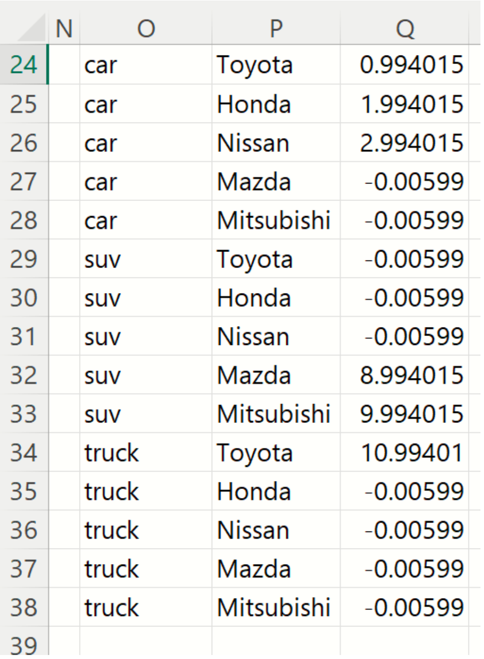
dim_vehicle	Toyota	Honda	Nissan	Mazda	Mitsubishi
car	0.99	1.99	2.99	-0.01	-0.01
suv	-0.01	-0.01	-0.01	8.99	9.99
truck	10.99	-0.01	-0.01	-0.01	-0.01

The formula, = PsiMean(K22, 1, "[dim_vehicle].[truck], [dim_model].[Toyota]"), entered into cell J24, illustrates how to display a single element of the cube, in this case the number of available Toyota trucks.



	J	K	L
23			
24	truck	Toyota	11
25			
26			

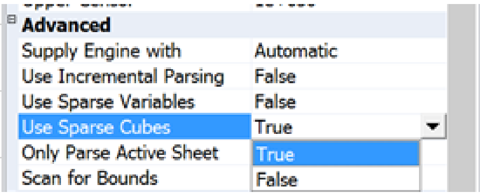
The formula, =PsiMean(K22, "vals") has been entered into cell O24 to illustrate how to display the cube dimensions and values in the worksheet. This formula spills across three columns and down 15 rows.



	N	O	P	Q
24		car	Toyota	0.994015
25		car	Honda	1.994015
26		car	Nissan	2.994015
27		car	Mazda	-0.00599
28		car	Mitsubishi	-0.00599
29		suv	Toyota	-0.00599
30		suv	Honda	-0.00599
31		suv	Nissan	-0.00599
32		suv	Mazda	8.994015
33		suv	Mitsubishi	9.994015
34		truck	Toyota	10.99401
35		truck	Honda	-0.00599
36		truck	Nissan	-0.00599
37		truck	Mazda	-0.00599
38		truck	Mitsubishi	-0.00599
39				

Use Sparse Cubes = True

Click the **Platform** tab on the Solver Task Pane, scroll down to *Advanced* and enter **True** for **Use Sparse Cubes**.

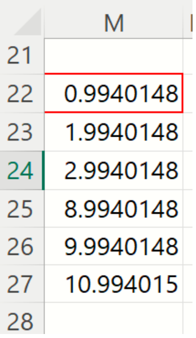


Advanced	
Supply Engine with	Automatic
Use Incremental Parsing	False
Use Sparse Variables	False
Use Sparse Cubes	True
Only Parse Active Sheet	
Scan for Bounds	

Since the ratio of the number of missing elements to the total number of possible elements is greater than 30% (6 out of a possible 15), Solver will only consider those six elements included in PsiTableCube() in the model formulation. In other words, while **Use Sparse Cubes** is set to **True**, only 6 elements are contained in the cube.

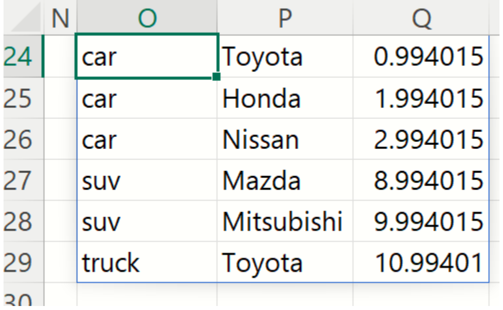
Run a simulation by clicking the down arrow beneath Simulate and selecting **Run Once**.

Notice that only the first *six* cells within the range M22:M36 hold values. This is because the cube only contains six elements! The remaining nine elements have not been included in the model. The value in cell M22 corresponds to the first cube element, Toyota cars, the value in cell M23 corresponds to the second cube element, Honda cars, the value in cell M24 corresponds to the third cube element Nissan cars, the value in cell M25 corresponds to the fourth cube element, Mazda SUVs, the value in cell M26 corresponds to the fifth cube element, Mitsubishi SUVs, and the value in cell M27 corresponds to the sixth cube element Toyota trucks.



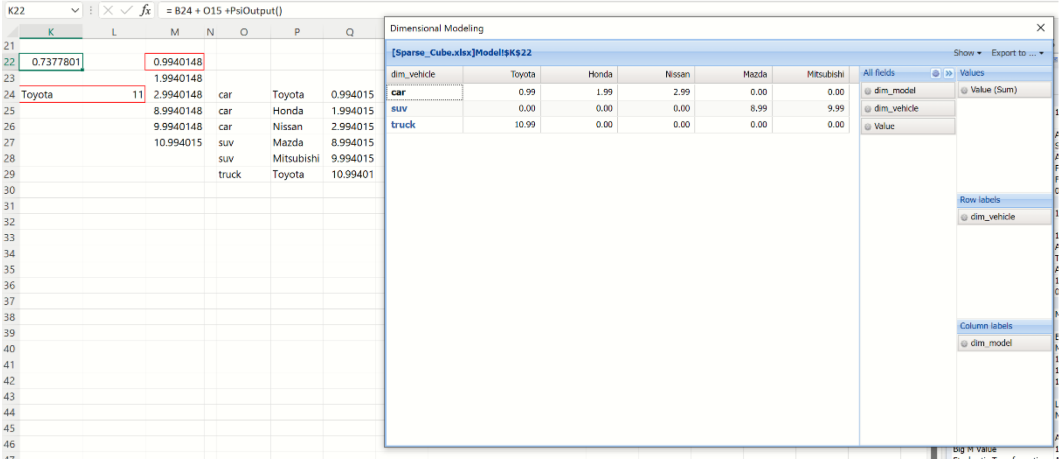
	M
21	
22	0.9940148
23	1.9940148
24	2.9940148
25	8.9940148
26	9.9940148
27	10.994015
28	

The same is true for the elements in cells O24:Q29. Only 6 elements appear because there are only 6 elements in the cube since Sparse Cube = True.



	N	O	P	Q
24		car	Toyota	0.994015
25		car	Honda	1.994015
26		car	Nissan	2.994015
27		suv	Mazda	8.994015
28		suv	Mitsubishi	9.994015
29		truck	Toyota	10.99401

To display the cube output, double-click cell K22. While the missing elements are set to 0's, these elements do not exist in the simulation model. Our 15 element cube has been reduced to 6.



K22 = B24 + O15 +PsiOutput()

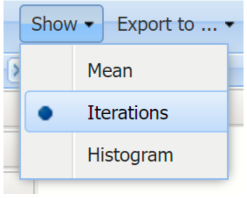
	K	L	M	N	O	P	Q
21							
22	0.7377801		0.9940148				
23			1.9940148				
24	Toyota	11	2.9940148		car	Toyota	0.994015
25			8.9940148		car	Honda	1.994015
26			9.9940148		car	Nissan	2.994015
27			10.994015		suv	Mazda	8.994015
28					suv	Mitsubishi	9.994015
29					truck	Toyota	10.99401

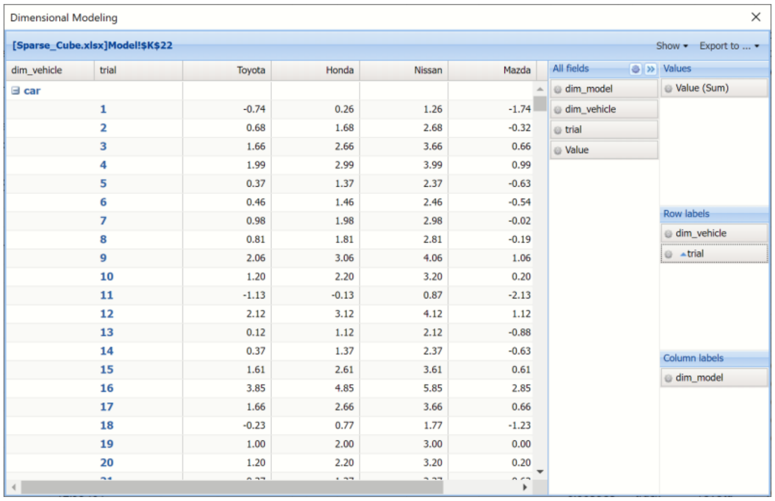
Dimensional Modeling

[Sparse_Cube.xlsx]Model!$K$22

dim_vehicle	Toyota	Honda	Nissan	Mazda	Mitsubishi
car	0.99	1.99	2.99	0.00	0.00
suv	0.00	0.00	0.00	8.99	9.99
truck	10.99	0.00	0.00	0.00	0.00

To view all 1,000 iterations, click Show (upper right hand corner), then click Iterations. The results shown below are unsorted.

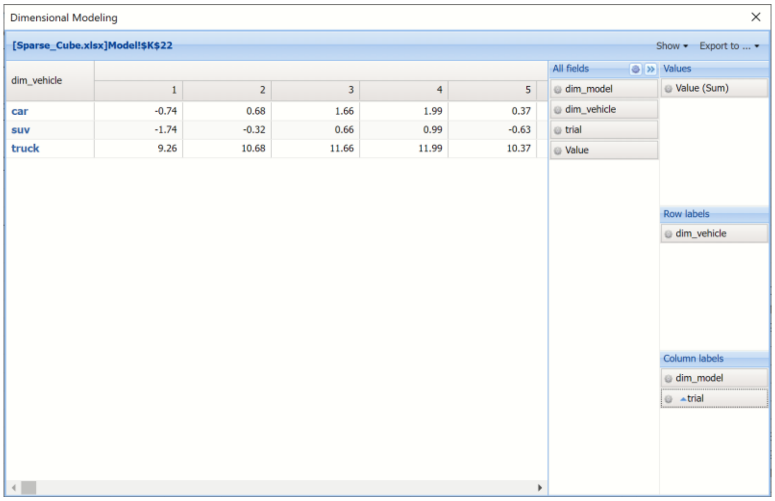



Dimensional Modeling

[Sparse_Cube.xlsx]Model!$K$22

dim_vehicle	trial	Toyota	Honda	Nissan	Mazda
car					
	1	-0.74	0.26	1.26	-1.74
	2	0.68	1.68	2.68	-0.32
	3	1.66	2.66	3.66	0.66
	4	1.99	2.99	3.99	0.99
	5	0.37	1.37	2.37	-0.63
	6	0.46	1.46	2.46	-0.54
	7	0.98	1.98	2.98	-0.02
	8	0.81	1.81	2.81	-0.19
	9	2.06	3.06	4.06	1.06
	10	1.20	2.20	3.20	0.20
	11	-1.13	-0.13	0.87	-2.13
	12	2.12	3.12	4.12	1.12
	13	0.12	1.12	2.12	-0.88
	14	0.37	1.37	2.37	-0.63
	15	1.61	2.61	3.61	0.61
	16	3.85	4.85	5.85	2.85
	17	1.66	2.66	3.66	0.66
	18	-0.23	0.77	1.77	-1.23
	19	1.00	2.00	3.00	0.00
	20	1.20	2.20	3.20	0.20

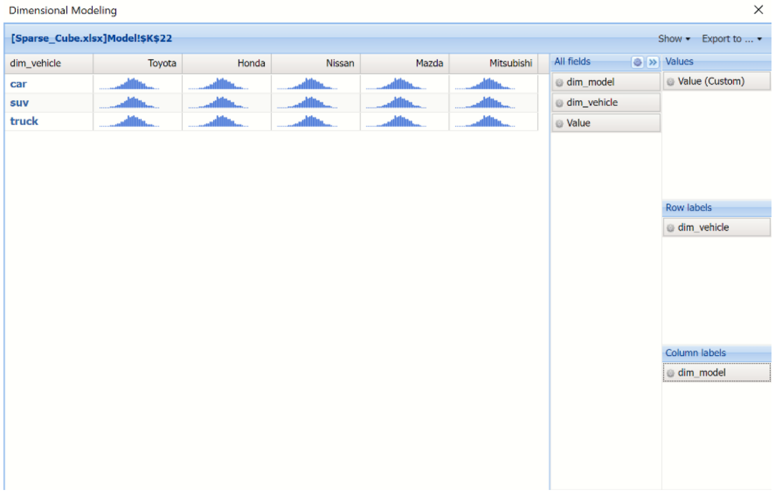
Drag "trials" beneath "Row labels" to "Column labels" to change the orientation of the grid.


Dimensional Modeling

[Sparse_Cube.xlsx]Model!$K$22

dim_vehicle	1	2	3	4	5
car	-0.74	0.68	1.66	1.99	0.37
suv	-1.74	-0.32	0.66	0.99	-0.63
truck	9.26	10.68	11.66	11.99	10.37

Click the down arrow next to Show and select Histogram to display a histogram for each element of the cube. Fields on the right of the grid can be arranged as desired.


Dimensional Modeling
[Sparse_Cube.xlsx]Model!$K$22
Show
Export to ...
dim_vehicle
Toyota
Honda
Nissan
Mazda
Mitsubishi
car
suv
truck
All fields
dim_model
dim_vehicle
Value
Values
Value (Custom)
Row labels
dim_vehicle
Column labels
dim_model

Automating Optimization in VBA

Introduction

This chapter explains how to use the Object-Oriented API in Analytic Solver Desktop to create, modify and solve optimization models under the control of your custom application written in VBA.

Note: Analytic Solver Cloud does not currently support this functionality.

This API is compatible with the object-oriented API offered by Frontline's Solver SDK, used to build custom applications of optimization and simulation using C++, C#, VB.NET, Java, MATLAB and other languages.

Analytic Solver also supports "traditional" VBA functions, which are upward compatible from the VBA functions supported by the standard Excel Solver. This API is described in the Frontline Solvers Reference Guide.

Why Use the Object-Oriented API?

The new object-oriented API is more powerful and much more convenient for programming than the "traditional" VBA functions.

With the "traditional" VBA functions:

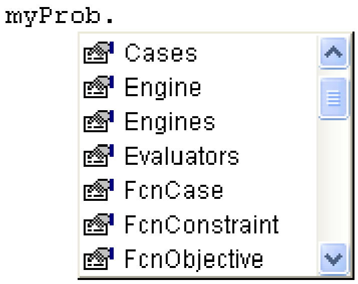
- You work with procedural functions that correspond to operations – such as **SolverOK** and **SolverSolve** – you can perform interactively in the Solver dialogs. To access the model and its variables and constraints, you must process the arrays of text and numbers returned by the **SolverGet** function.
- To obtain solution values, you must use the Excel object model (usually the **Range** object) to access the decision variable cells on the worksheet. You must take care to access the correct cells for specific decision variables.
- To obtain dual values and ranges, you must call the **SolverFinish** function to insert a report worksheet into the workbook, then use the Excel object model to access cells in the report. You must take *extra* care to access the correct report cells containing dual values and ranges.

With the new Object-Oriented API:

- You work with objects that correspond to the **Problem**, **Model**, **Solver**, **Engine**, **Variables**, and **Functions**. You can access sets of variables and constraints in the current model directly with expressions such as myProb.VarDecision and myProb.FcnConstraint.
- You can obtain solution values directly, with expressions such as myProb.VarDecision.FinalValue(i). If you need the cell address for a set of decision variable cells, you can write myProb.VarDecision.Name.
- You can access dual values and ranges for variables and constraints directly, with expressions such as myProb.VarDecision.DualValue(i) or myProb.FcnConstraint.DualValue(i).

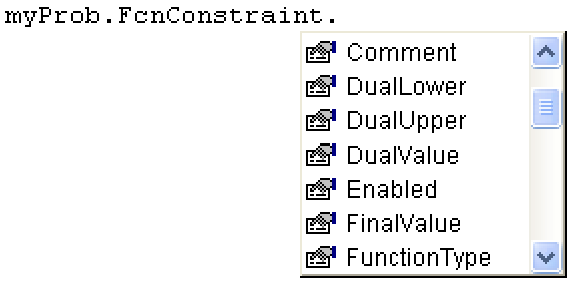
The result is VBA code that's easier to read, and easier to write in the first place. Since the VBA Editor recognizes the object model exposed by Analytic Solver – just as it recognizes the object model exposed by Excel – you'll receive

IntelliSense prompts as you write code. For example, if you type a line **Dim myProb as New Problem**, then start a new line with **myProb.**, you'll be prompted with the properties and methods available for Problems:



```
myProb.
```

If you select FcnConstraint and then type a period, you'll be prompted with the properties and methods available for Functions:



```
myProb.FcnConstraint.
```

This makes it much easier to write correct code, without consulting the manual. What's more, you can use this object-oriented API when programming Excel and Analytic Solver from new languages such as **VB.NET** and **C#**, working in Visual Studio, and receive IntelliSense prompts in the syntax of these languages!

If you're using new functionality in Analytic Solver, the object-oriented API is your best bet. And if you're planning to move your application outside of Excel in the future – so it will run as a standalone program – you'll find that Frontline's **Solver Platform SDK** offers an object-oriented API that closely resembles the new APIs in Analytic Solver.

Running Predefined Solver Models

Controlling the Solver can be as simple as *adding two lines* to your VBA code! Each worksheet in a workbook may have a Solver problem defined, which is saved automatically with the workbook. You can create this Solver model interactively if you wish. If you distribute such a workbook, with a worksheet containing a Solver model and a VBA module, you can simply add a reference to the Analytic Solver COM server (see below), activate the worksheet, and add these two lines of code:

```
Dim prob As New Problem
prob.Solver.Optimize
```

Using the Macro Recorder

If you want to set up a Solver model "from scratch" programmatically, one easy way to see how to use the object-oriented API is to turn on the Excel Macro Recorder and then set up a Solver model interactively. Microsoft Excel will record a macro in VBA that calls the object-oriented API to mimic the actions you perform. You can then edit and customize this macro, and incorporate it into your application.

Note: You must use the classic Solver dialog to record a macro, rather than the Solver Task Pane. To open, click Add-ins – Premium Solver. By default, the Excel Macro Recorder will record calls to the object-oriented API.

Adding a Reference in the VBA Editor

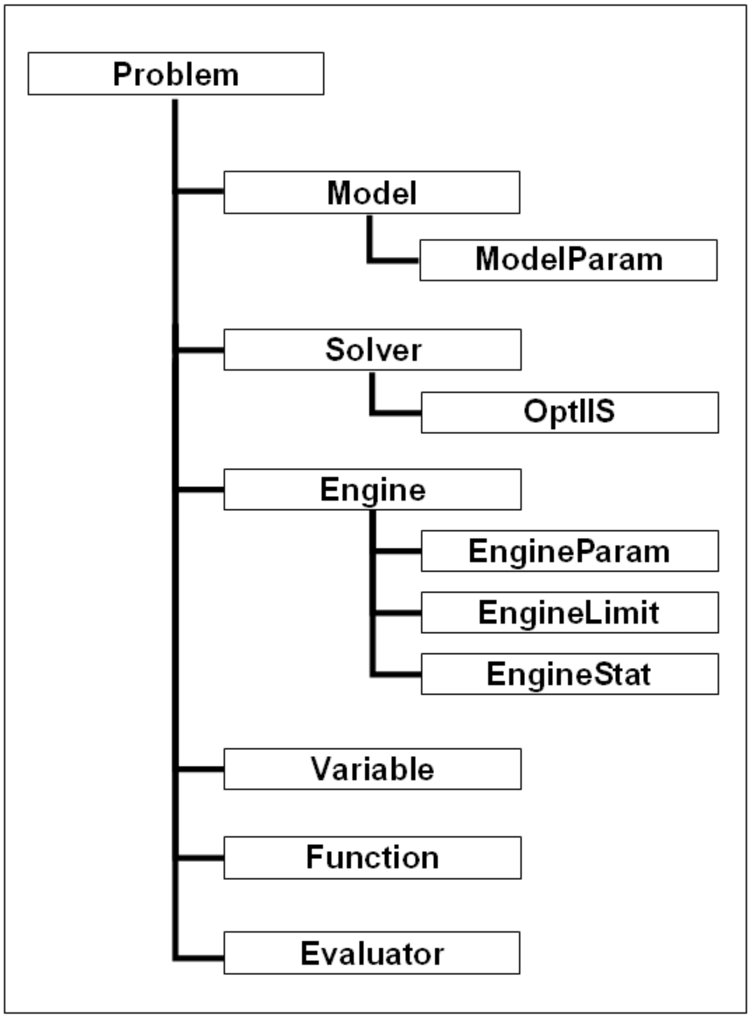
To use the new object-oriented API in VBA, you must first add a reference to the type library for the Analytic Solver COM server. To do this:

1. Press Alt-F11 to open the VBA Editor.
2. Select menu choice **Tools References**.
3. Scroll down until you find **Analytic Solver V2025 Q1 Type Library**.
4. Check the box next to this entry, and click OK to close the dialog.
5. Use File Save to save your workbook.

Note that this is a **different** reference from **Solver**, which is the reference you add in order to use the "traditional" VBA functions.

Analytic Solver Object Model

Analytic Solver makes available a hierarchy of objects for describing optimization problems, as pictured on the next page. Note that the same objects are used for both optimization and simulation problems; see the next chapter, "Automating Simulation in VBA," for more information.

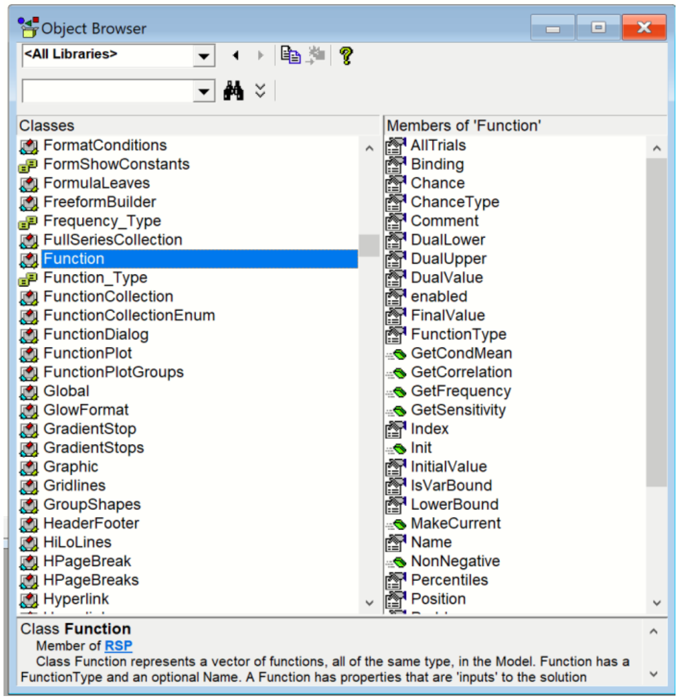


The **Problem** object represents the whole problem, and the **Model** object represents the internal structure of the model, which in Analytic Solver is defined by your formulas on the spreadsheet. The **Solver** object represents the optimization process – you call its Optimize method to find an optimal solution. The **Engine** object represents either a built-in or plug-in Solver engine. A **Variable** object represents a range of one or more contiguous cells that contains decision variables, while a **Function** object represents a range of cells that contains either constraint left hand sides or the objective. Each Problem has a collection of Variable objects, and a collection of Function objects. An **Evaluator** represents a function you write that the Solver will call on each iteration (Trial Solution), or on each subproblem in a larger problem.

The Model has a collection of **ModelParam** objects, each representing a single option or parameter of the PSI Interpreter (appearing in the Task Pane Platform tab). An Engine has a collection of **EngineParam** objects, each representing a single option or parameter of a Solver engine (appearing in the Task Pane Engine tab). It also has an **EngineLimit** object, holding problem size limits for this Solver engine, and an **EngineStat** object, holding performance statistics for the last optimization problem solved by this engine. An **OptIIS** object holds results of an infeasibility analysis of the problem.

Using the VBA Object Browser

You can examine Analytic Solver objects, properties and methods in the VBA Object Browser. To do this, press **Alt-F11** to open the VBA Editor, and select menu choice **View Object Browser**. This displays a child window like the one pictured below.



Class Function
Member of RSP
Class Function represents a vector of functions, all of the same type, in the Model. Function has a FunctionType and an optional Name. A Function has properties that are 'inputs' to the solution

The dropdown list at the top left corner of the Object Browser initially displays <All Libraries> – change this to select **RSP**. In the object browser pictured, we've highlighted the properties of the Function object

Programming the Object Model

You use the Analytic Solver object-oriented API by first creating an instance of a **Problem**, and initializing it with the Solver model defined on a worksheet in an open workbook. When you do this, a collection of **Variable** objects and a collection of **Function** objects are created automatically. Each Variable object corresponds to a cell range of decision variables that appears in the outlined list of variables, and each Function object corresponds to a cell range of constraints that appears in the list of constraints in the Task Pane.

Once you have an initialized Problem object, you can do several things:

- Set Solver and Engine parameters such as the maximum time or number of iterations, the method used to compute gradients, and other options and tolerances.
- Perform an optimization, and check the final status of the solution process.
- Get results of the optimization, by accessing properties of the Variable and Function objects, and performance statistics, by accessing properties of the EngineStat object.

Example VBA Code Using the Object Model

Below is an example of VBA code that could be linked to a command button on the worksheet:

```
Private Sub CommandButton1_Click()
  Dim prob As New RSP.Problem

  prob.Engine = prob.Engines("Standard LP/Quadratic")
  prob.Engine.Params("MaxTime") = 600

  prob.Solver.Optimize
  MsgBox "Status = " & prob.Solver.OptimizeStatus
  MsgBox "Obj = " & prob.FcnObjective.FinalValue(0)

  For i = 0 To prob.Variables.Count - 1
    For j = 0 To prob.Variables(i).Size - 1
      MsgBox prob.Variables(i).FinalValue(j)
    Next j
  Next i

  Set prob = Nothing
End Sub
```

The first line creates an instance of a Problem, which by default is associated with the Solver model defined on the active worksheet. You could associate the Problem object with a different worksheet by calling the prob.Init method:

```
Private Sub CommandButton1_Click()
  'create new problem
  Dim prob As New RSP.Problem
'initialize model on Invent2 worksheet
  prob.Init Worksheets("Invent2")
```

The second line selects the Standard LP/Quadratic Solver engine, and the third line sets the maximum solution time to 600 seconds. The **string names** of parameters such as "MaxTime" are documented in the Frontline Solvers Reference Guide.

The next set of three lines performs the optimization, displays the Solver Result status code (for example 0), and displays the final value of the objective.

The double for-loop in the next five lines steps through the Variable objects – each one representing a range of contiguous cells – and displays the final value for each variable cell in each range.

Evaluators Called During the Solution Process

You can write a VBA function that Analytic Solver will call at certain points during the solution process. In this “callback function,” you can access information about the problem and solution so far, to monitor or report progress and decide whether to stop or continue the solution process.

The object-oriented API defines an Evaluator object that is associated with your “callback function” and specifies when Analytic Solver should call it. You can define Evaluators to be called on each iteration or Trial Solution, or on each subproblem or each new solution (“incumbent”) when the solution process involves multiple subproblems (global optimization problems, and problems with integer variables).

The VBA function you write to serve as an Evaluator must be contained in a **class module** – not a ‘regular’ VBA module – and it must be declared to have the **With Events** property.

On the next page is an example of code for an Evaluator, in a class module named Class1.

```
Private WithEvents EvalIterator As RSP.Evaluator

Private Function EvalIterator_Evaluate _
  (ByVal Evaluator As RSP.IEvaluator) As _
  RSP.Engine_Action

  MsgBox "Iteration = " _
    & Evaluator.Problem.Engine.Stat.Iterations _
    & Chr(13) & Chr(10) & "Objective = " _
    & Evaluator.Problem.FcnObjective.Value(0) _
    & Chr(13) & Chr(10)

  EvalIterator_Evaluate = Engine_Action_Continue
End Function

Public Sub MySolve()
  Set EvalIterator = New RSP.Evaluator
  Dim prob As New RSP.Problem
  prob.Evaluators(Eval_Type_Iteration) = EvalIterator
  prob.Solver.Optimize
  Set EvalIterator = Nothing
End Sub
```

Having created the class module Class1, in a ‘regular’ VBA module you can create an instance of Class1, and then call the MySolve method in Class1:

```
Private Sub CommandButton1_Click()
  Dim c As New Class1
  c.MySolve
End Sub
```

Refinery.xlsx: Multiple Blocks of Variables and Functions

A further example of programming the object-oriented API is shown in the model **Refinery.xlsx**, which is installed in the Examples folder, normally at the path C:\ProgramData\Frontline Systems\Examples. (You can also open this example by clicking Help – Examples on the Ribbon, then click Optimization Examples.)

The Refinery.xlsx model, which is based on Problem 12.6 in the 3rd edition of *Model Building in Mathematical Programming* by H.P. Williams (see the Recommended Books on www.solver.com for details), has ten blocks of decision variables and eleven blocks of constraints, plus bounds on certain variables.

The VBA code for this model illustrates some of the many ways you can use the object-oriented API. For example, the following line displays the time that was required to solve the problem:

```
MsgBox prob.Engine.Stat.Milliseconds & " msec"
```

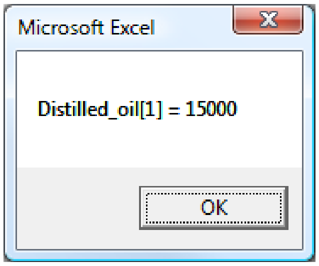
The following line inserts an Answer Report into the active workbook:

```
prob.Solver.Report "", "Answer"
```

The VBA code on the next page illustrates one way to display final solution values for each of the ten blocks of decision variables in this problem:

```
For i = 0 To prob.Variables.Count - 1
  prob.Variables(i).MakeCurrent
  For j = 0 To prob.VarDecision.Size - 1
    MsgBox prob.VarDecision.Name _
      & "[" & j+1 & "] = " & _
      prob.VarDecision.FinalValue(j)
  Next j
Next i
```

When executed, this code displays MessageBoxes such as:



Distilled_oil[1] = 15000

The line `prob.Variables(i).MakeCurrent` associates the Problem property VarDecision (a single block of decision variables) with each of the ten blocks of variables in turn, allowing you to refer to solution values and dual values of this block in more compact notation. A similar line of code can be used to make the Problem property FcnConstraint represent one of the eleven blocks of constraints. The .Name property of the block, which usually returns a string such as “$A$1:$A$10”, returns “Distilled_oil” in this case, since the Excel model has a defined name for this block of cells.

Adding New Variables and Constraints to a Model

So far, we’ve seen how to access variables and constraints that were defined interactively through the Ribbon and Task Pane (or the Solver Parameters dialog), using the Problem object and its VarDecision and FcnConstraint properties, as well as the Variables and Functions collections. But you can also add new variables and constraints to a model through the Object-Oriented API, and have them appear in the Task Pane Model tab. To illustrate some additional properties used in full Analytic Solver Comprehensive, let’s add a *recourse* decision variable and a *chance* constraint.

To add a new decision variable, we create a Variable object, associate it with a worksheet cell, set its properties, and add it to the Variables collection. To make this a recourse decision variable, we simply set its VariableType property:

```
Dim v As New RSP.Variable
  v.Init "$D$18"
  v.VariableType = Variable_Type_Recourse
  v.NonNegative
  prob.Variables.Add v
```

Similarly, to add a new constraint, we create a Function object, associate it with a formula cell on the worksheet, set its properties, and add it to the Functions collection. To make this a chance constraint, we simply set its FunctionType, ChanceType and Chance properties:

```
Dim f As New RSP.Function
  f.Init "$C$23"
  f.FunctionType = Function_Type_Chance
  f.ChanceType = Chance_Type_VaR
  f.Chance = 0.95
  f.LowerBound.Array = 0 '>= 0
  prob.Functions.Add f
```

Use `f.UpperBound.Array` to add an upper bound to the constraint.

To add an equality constraint, set `f.UpperBound.Array` and `f.LowerBound.array` to the same right hand side.

CuttingStock.xlsx: Multiple Problems and Dynamically Generated Variables

A more ambitious example of programming the object-oriented API is shown in the model **CuttingStockVBA(Opt).xlsx**, which is installed in the Examples folder, normally at the path C:\ProgramData\Frontline Systems\Examples. (You can also open this example by clicking Help – Examples on the Ribbon, then click Optimization Examples.)

This application uses the object-oriented API to define and repeatedly solve two optimization problems, passing information back and forth between the two problems. One problem is instantiated from a worksheet with the prob.Init method as mentioned earlier; the other problem is created "from scratch," with new dynamically generated decision variables added each time the problem is solved.

Cutting Stock Problem

CuttingStockVBA(Opt).xlsx solves a classical "cutting stock" problem, which arises for example in lumber and paper mills. Imagine that you have a number of sheets of wood or rolls of paper of a fixed width, waiting to be cut; you have customer orders for sheets or rolls of various different widths. Your task is to cut the larger, fixed-width sheets or rolls into different sizes in a way that minimizes the total stock used while meeting customer demand.

You might for example cut a 100 inch sheet into two sheets of 45 inches (leaving 10 inches wasted), three sheets of 31 inches (leaving 7 inches wasted), one 45-inch sheet and one 31-inch sheet (leaving 24 inches wasted), etc. Each of these is called a *pattern*, and the main problem will have a decision variable representing the number of copies of that pattern to cut. In a 'real-life" application, the number of possible patterns is exponentially large, yielding a model that's too large to solve.

Column Generation Method

We can instead use the technique of *column generation* ('columns' here refers to variables in the main problem). We choose a small initial set of patterns to include in the model, and solve the main problem (an LP). Since it is unlikely that we chose the perfect set of patterns initially, we use the dual variable information from the main problem to generate a new pattern. We generate this new pattern by solving a second optimization problem, called a ‘knapsack’ problem. A decision variable for the new pattern is dynamically created and added to the main problem, which is solved again. These two problems, the main problem and the knapsack problem, are solved in turn until no more patterns can be generated that will reduce the total stock used.

Worksheets and VBA Code

In CuttingStockVBA(Opt).xlsx, sheet Input contains the ‘knapsack’ problem, which is solved to generate new patterns, and sheet Patterns contains the main problem. Our VBA code executes a loop, alternately solving the main problem and the knapsack problem. When the solution to the main problem can no longer be improved, we solve a final problem where we add integer constraints on the variables, so the final solution yields an exact count of the patterns that should be cut.

Open CuttingStockVBA(Opt).xlsx and press Alt-F11 to view its VBA code in the VBA Editor. The first block of code clears the Patterns sheet and sets up the initial patterns. This code simply sets values and formulas into cells, using the Excel object model.

```
Dim nPat As Integer, i As Integer, _
nNumDemands As Integer
nPat = Range("Demands").Count

nNumDemands = nPat
Worksheets("Patterns").Activate
Range("$A$1:$Z$100").Clear

' create initial patterns
For i = 1 To nPat
  Cells(2 + i, i) = Int(Range("RollSize").Value2 _
      / Range("Widths").Cells(i).Value2)
  Cells(2 + i, nPat + 1).Formula = "=sumproduct(" _
      & Range(Cells(1, 1), Cells(1, nPat)).Address
      & "," & Range(Cells(2 + i, 1), _
        Cells(2 + i, nPat)).Address & ")"
  Cells(3 + nPat, i) = Range("RollSize").Value2 - _
        Cells(2 + i, i) *
Range("Widths").Cells(i).Value2
Next i
```

The code then enters the main loop in which we add a new pattern to the main problem, and set up and solve that problem using Analytic Solver’s object model:

```
  'generate patterns
  Cells(2, 1).Formula = "=sum(" _
    & Range(Cells(1, 1), Cells(1, nPat)).Address &
")"
  For i = 1 To nNumDemands
    Cells(2 + i, nPat + 1).Formula = "=sumproduct(" _
      & Range(Cells(1, 1), Cells(1, nPat)).Address _
      & "," & Range(Cells(2 + i, 1), Cells(2 + i, _
        nPat)).Address & ")"
  Next i

  Dim prob As New RSP.Problem

  ' variables
  prob.Variables.Clear
  prob.Functions.Clear
  Dim vars As New RSP.Variable
  vars.Init Range(Cells(1, 1), Cells(1, nPat))
  vars.NonNegative
  prob.Variables.Add vars
  Set vars = Nothing
```

```
  ' objective
  Dim objective As New RSP.Function
  objective.Init Range(Cells(2, 1), Cells(2, 1))
  objective.FunctionType = Function_Type_Objective
  prob.Functions.Add objective
  Set objective = Nothing
```

```
  ' constraints
  ReDim constraints(1 To nNumDemands) As _
    New RSP.Function
  For i = 1 To nNumDemands
    constraints(i).Init Range(Cells(2 + i, nPat + 1),
_
      Cells(2 + i, nPat + 1))
    constraints(i).LowerBound(0) = _
      Range("Demands").Cells(i).Value2
    prob.Functions.Add constraints(i)
  Next i
```

```
  prob.Solver.SolverType = Solver_Type_Minimize
  prob.Engine = prob.Engines("Standard LP/Quadratic")
  prob.Solver.Optimize
```

Next, the code obtains the dual values from the solution to the main problem, via the Analytic Solver object model, and stores these values as parameters of the knapsack problem on the Input worksheet, via the Excel object model:

```
  ' capture shadow prices
  For i = 1 To nNumDemands
    Worksheets("Input").Cells(2 + i, 5) = _
      prob.Functions(Range(Cells(2 + i, nPat + 1), _
      Cells(2 + i, nPat + 1))).DualValue(0)
  Next
  Worksheets("Input").Activate
  Dim j As Integer
  j = 1
  For i = 1 To prob.Functions.Count
    If prob.Functions(i - 1).FunctionType = _
       Function_Type_Constraint Then
         Cells(2 + j, 5) = _
           prob.Functions(i - 1).DualValue(0)
         j = j + 1
    End If
  Next i
```

The code then sets up and solves the knapsack problem, again using the Analytic Solver object model:

```
  Dim prob1 As New RSP.Problem
  prob1.Init Worksheets("Input")
  prob1.Engine = prob1.Engines("Standard
  LP/Quadratic")
  prob1.Engine.Params("IntTolerance").Value = 0
  prob1.Solver.Optimize
```

If the objective value of the knapsack problem is less than 1 (allowing for rounding error) – meaning that there are no more patterns that will improve the solution – we can exit the loop.

```
If 1 - prob1.FcnObjective.FinalValue(0) _
   >= -0.00001 Then
   Exit Do
End If
```

Otherwise, the code writes the new pattern to the Patterns (main problem) worksheet, using the Excel object model:

```
Worksheets("Patterns").Activate
Cells(nNumDemands + 3, nPat + 1) = _
  Range("RollSize").Value2
' write out new pattern, and associated waste
For i = 1 To nNumDemands
  Cells(2 + i, nPat + 1) = _
    prob1.VarDecision.FinalValue(i - 1)
  Cells(nNumDemands + 3, nPat + 1) = _
    Cells(nNumDemands + 3, nPat + 1).Value2 - _
    prob1.VarDecision.FinalValue(i - 1) * _
    Range("widths").Cells(i).Value
Next i

nPat = nPat + 1
Set prob1 = Nothing
```

When the Do … Loop is exited, all patterns have been generated. Finally, the code solves one more problem with integer constraints on the variables, to ensure that we produce the exact count needed to meet customer demand:

```
Worksheets("Patterns").Activate
prob.Init Worksheets("Patterns")
prob.Functions.Clear
prob.Variables.Clear

' variables
Dim finalvars As New RSP.Variable
finalvars.Init Range(Cells(1, 1), Cells(1, nPat))
For i = 1 To nPat
  finalvars.IntegerType(i - 1) =
Integer_Type_Integer
Next i
finalvars.NonNegative
prob.Variables.Add finalvars

' objective
Cells(3 + nNumDemands, nPat + 1).Formula = _
  "=sumproduct(" & Range(Cells(1, 1), _
  Cells(1, nPat)).Address & "," _
  & Range(Cells(3 + nNumDemands, 1), _
  Cells(3 + nNumDemands, nPat)).Address & ")"
Dim TotalWaste As New RSP.Function
TotalWaste.Init Range(Cells(3 + nNumDemands, _
  nPat + 1), Cells(3 + nNumDemands, nPat + 1))
  TotalWaste.FunctionType = Function_Type_Objective
  prob.Functions.Add TotalWaste

  ' constraints
  ReDim constraints(1 To nNumDemands) As _
    New RSP.Function
  For i = 1 To nNumDemands
    Cells(2 + i, nPat + 1).Formula = "=sumproduct(" _
      & Range(Cells(1, 1), Cells(1, nPat)).Address _
      & "," & Range(Cells(2 + i, 1), Cells(2 + i, _
      nPat)).Address & ")"
    constraints(i).Init Range(Cells(2 + i, nPat + 1),
_
      Cells(2 + i, nPat + 1))
    constraints(i).LowerBound(0) = _
      Range("Demands").Cells(i).Value2
    prob.Functions.Add constraints(i)
  Next i

  prob.Engine = prob.Engines("Standard LP/Quadratic")
  prob.Engine.Params("IntTolerance").Value = 0
  prob.Solver.Optimize
  Set prob = Nothing
```

This example illustrates some of the power of the object-oriented API. Although you could use the “traditional” VBA functions to obtain similar results, it would require quite a bit more code to do so, especially at the step of obtaining the dual values from the solution of the main problem and using them to solve the next knapsack problem.

If you wanted to move this application from Excel to a standalone program, you’d find that nearly all the code in CuttingStockVBA(Opt).xlsx that references the Analytic Solver object model could be re-used, with few or no changes, in building an application for the Solver Platform SDK. You’d have to rewrite the code that references cells via the Excel object model to use arrays in a programming language instead, but this would not be difficult.

Automating Simulation in VBA

Introduction

Analytic Solver Platform
Risk Solver Platform
Premium Solver Platform
Analytic Solver Pro
Premium Solver Pro
Risk Solver Pro

This chapter explains how to use the Object-Oriented API in Analytic Solver Desktop to create, modify and solve simulation models under the control of your custom application written in VBA.

Note: Analytic Solver Cloud does not currently support this functionality.

In the simplest case, you can use a few standard lines of VBA code to enable and disable Interactive Simulation, as described below. But you can do much more in VBA, to create custom risk analysis applications.

You can define a **Problem** and instantiate it from the spreadsheet with two lines of code, then access the uncertain elements of your model via **Variable** and **Function** objects. You can perform simulations, access trials and summary statistics, and present them the way you want to your end user. All the power of the Excel object model is available, including database access, charts and graphs, and custom dialogs and controls.

Analytic Solver's VBA object model closely resembles the object-oriented API of Frontline's **Solver SDK** – which includes a complete toolkit for Monte Carlo simulation. This makes it easier to move an application from Excel to a custom program written in C/C++, Visual Basic, VB.NET, Java or MATLAB.

Adding a Reference in the VBA Editor

To use the new object-oriented API in VBA, you must first add a reference to the type library for the Analytic Solver COM server. To do this:

1. Press Alt-F11 to open the VBA Editor.
2. Select menu choice **Tools References**.
3. Scroll down until you find **Analytic Solver V2025 Q1 Type Library**.
4. Check the box next to this entry, and click OK to close the dialog.
5. Use File Save to save your workbook.

Note that this is a **different** reference from **Solver**, which is the reference you add in order to use the "traditional" VBA functions.

You need this reference if your VBA code uses the Event Listener as described in the next section, or uses other elements of the Analytic Solver VBA object model, described in later sections of this chapter (or both).

Activating Interactive Simulation

If you simply want to activate or deactivate Interactive Simulation under program control, you need a few standard lines of VBA code. You can place this code in any VBA procedure you write, and cause it to be run in any manner that is convenient for your application. For example, you could use:

```
Public Sub ISActivate
    Dim ev As New EventListener
    ev.AttachEvent Application
    Set ev = Nothing
End Sub

Public Sub ISDeactivate
    Dim ev As New EventListener
    ev.DetachEvent
    Set ev = Nothing
End Sub
```

You could use the Tools Macro Run menu choice in Excel to run these VBA procedures, or you could associate a Ctrl-Key combination with each procedure, so it is run when that key combination is pressed.

Another approach appears in the example workbook **BusinessPlanPsiChart.xlsx** (normally installed at C:\ProgramData\Frontline Systems\Examples): Two buttons named CommandButton1 and CommandButton2 are placed on the spreadsheet, along with two images named Picture Bulb On and Picture Bulb Off, that are stacked on top of each other. The VBA code is:

```
Private Sub CommandButton1_Click()
    Dim ev As New EventListener
    ev.AttachEvent Application
    Set ev = Nothing
    Me.Shapes("Picture Bulb On").Visible = True
    Me.Shapes("Picture Bulb Off").Visible = False
End Sub

Private Sub CommandButton2_Click()
    Dim ev As New EventListener
    ev.DetachEvent
    Set ev = Nothing
    Me.Shapes("Picture Bulb On").Visible = False
    Me.Shapes("Picture Bulb Off").Visible = True
End Sub
```

When the “On” button (CommandButton1) is clicked, Interactive Simulation is activated, and the light bulb is “turned on.” When the “Off” button is clicked, Interactive Simulation is deactivated, and the light bulb is “turned off.”

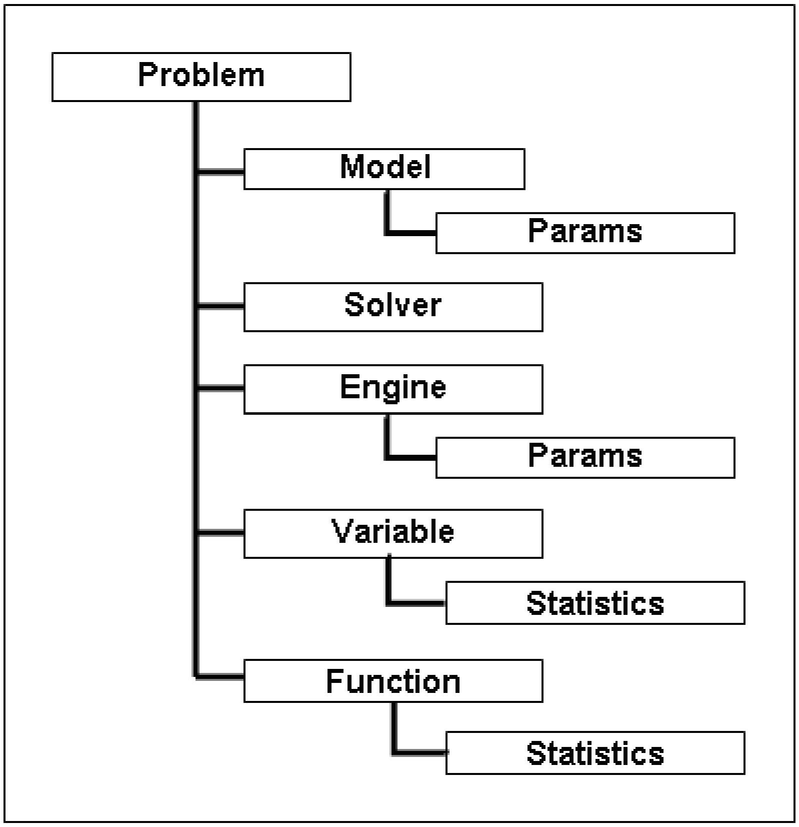
Using VBA to Control Analytic Solver

You can do much more with VBA: You can perform simulations under your control, rather than on every worksheet recalculation. You can obtain results of the simulation – statistics, percentiles, or even raw trial data – that you’d otherwise obtain through worksheet functions such as PsiMean(), PsiPercentile(), or PsiData(). You can even define certain worksheet cells as uncertain variables and supply trial data for them in a simulation, without using functions such as PsiNormal() or PsiUniform() in cell formulas.

You also have access in VBA to the Excel object model, which provides a very rich source of high-level functions for manipulating cells and ranges, creating charts and graphs, accessing external databases, and much more. This gives you a powerful set of tools for developing custom risk analysis applications.

Analytic Solver Object Model

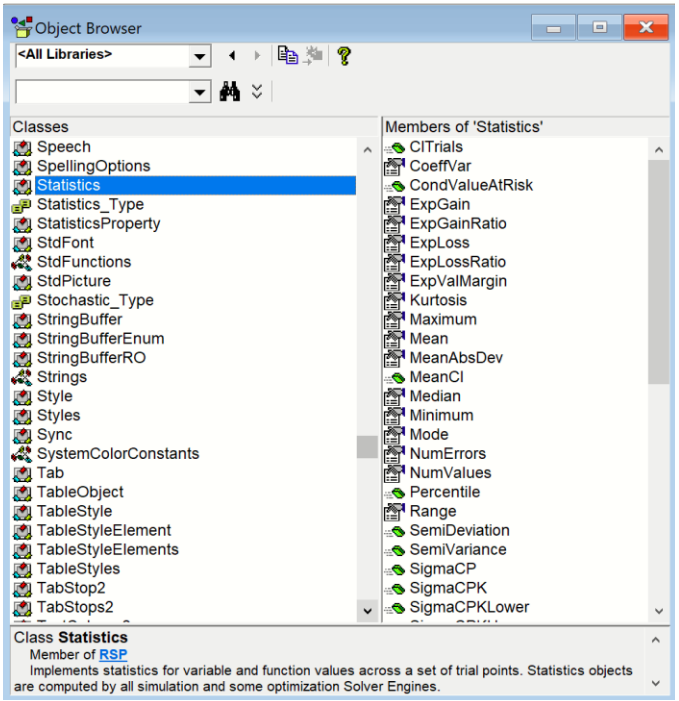
Analytic Solver makes available a hierarchy of objects for describing Monte Carlo simulation problems, pictured below. This object model is a simplified subset of the object hierarchy offered by Frontline's Solver Platform SDK product, which is used to build custom applications in C/C++, Visual Basic, VB.NET, Java or MATLAB. Note that the same objects are used for both optimization and simulation problems.



The **Problem** object represents the whole problem, and the **Model** object represents the internal structure of the model, which in Analytic Solver is defined by your formulas on the spreadsheet. The **Solver** object represents the Monte Carlo process – you call its Simulate method to perform a simulation. The **Engine** object represents the PSI Technology "engine" – its parameters include the sampling method, for example. A **Variable** object represents a range of one or more contiguous cells that contains uncertain variables, while a **Function** object represents a range of cells that contains uncertain functions. You may have a collection of Variable objects, and a collection of Function objects in one Problem.

Using the VBA Object Browser

You can examine the Analytic Solver objects, properties and methods in the VBA Object Browser. To do this, press Alt-F11 to open the VBA Editor, and select menu choice View Object Browser. This displays a child window like the one pictured below. The dropdown list at the top left corner of the Object Browser initially displays <All Libraries> – change this to select RSP. Below, we’ve highlighted the properties of the Statistics object, which is a child of the Variable and Function objects.



Class **Statistics**
Member of RSP
Implements statistics for variable and function values across a set of trial points. Statistics objects are computed by all simulation and some optimization Solver Engines.

Using Analytic Solver Objects

You use the Analytic Solver object model by first creating an instance of a **Problem**, and initializing it with the simulation model defined in your Excel workbook. Once you have an initialized Problem object, you can do several things:

- Add one or more new Variable objects to the problem. (This is an advanced step, covered near the end of this chapter.)
- Set Solver and Engine parameters such as the number of simulations, number of trials per simulation, the sampling method, and the random seed.
- Perform a simulation.
- Get results of the simulation, by accessing properties of the Variable and Function objects, and of their “child” Statistics objects.

The simplest action you might want to take is to create a Problem representing the workbook, set parameters, perform a simulation, and display results. Below is an example that could be linked to a command button on the worksheet:

```
Private Sub CommandButton3_Click()
   Dim prob As New RSP.Problem
   prob.Init ActiveWorkbook

   prob.Solver.NumTrials = 5000
   prob.Engine.Params("SamplingMethod") = 2
   prob.Solver.Simulate

   For i = 0 To prob.Functions.Count - 1
      MsgBox prob.Functions(i).Statistics.Mean(0)
   Next i
   Set prob = Nothing
End Sub
```

The first two lines create an instance of a Problem, and initialize it with the simulation model defined in your Excel workbook. The next two lines set the number of trials in the simulation to 5000, and the sampling method to Latin Hypercube. The fifth line performs a simulation.

The for-loop in the next three lines will step through the Function objects – assuming, for simplicity here, that each Function object represents just one cell – and display the Mean property of the child Statistics object (the mean or average value of the function across all trials) for each one.

Using Variable and Function Objects

When you create and initialize a **Problem**, a collection of **Variable** objects and a collection of **Function** objects are created automatically. Each Variable object corresponds to a range of one or more cells that contain PSI Distribution functions. Each Function object corresponds to a formula cell referenced as the first argument of a PSI Statistics function, or a range of one or more formula cells referenced by PsiOutput(). For more information on the grouping of cells into Function objects, please see the Frontline Solvers Reference Guide.

Indexing Variable and Function Objects

The VBA code example above assumes that any cells containing =PsiOutput() are separated from each other on the worksheet. When uncertain variable and uncertain function cells lie in a contiguous range – for example A1:A5 or A1:E1 – one Variable or Function object is created to represent all the cells in the range. The Size property of this object tells you the number of cells in the range, and its other properties may be indexed to access statistics – for example – of individual cells. The for-loop above could be written in more general form as:

```
 For i = 0 To prob.Functions.Count - 1
      For j = 0 To prob.Functions(i).Size - 1
         MsgBox prob.Functions(i).Statistics.Mean(j)
      Next j
   Next i
```

Percentiles and AllTrials Properties

In addition to the Statistics child object used above, Variable and Function objects have properties named Percentiles and AllTrials (the raw trial data).

The Percentiles property plays the same role as the PsiPercentile() function on the spreadsheet. It yields a DoubleMatrix object that takes two subscripts: the index (starting from 0) of the cell in the range represented by the Variable or Function object, and the percentile index, which runs from 0 to 98 for the 1st through 99th percentile.

The AllTrials property plays the same role as the PsiData() function on the spreadsheet. It yields a DoubleMatrix object that takes two subscripts: the index (starting from 0) of the cell in the range represented by the Variable or Function object, and the index of the Monte Carlo trial (starting from 0).

The example code below illustrates how to obtain and display the values of the uncertain functions for each simulation trial.

```
For i = 0 To prob.Functions.Count - 1
      For j = 0 To prob.Functions(i).Size - 1
      MsgBox prob.Functions(i).Percentiles(j,94)
        For k = 0 To prob.Solver.NumTrials -1
            MsgBox prob.Functions(i).AllTrials(j, k)
        Next k

    Next j

Next i
```

GetFrequency Method

Variable and Function objects also have a GetFrequency method, that plays the same role as the PsiFrequency() function on the spreadsheet. It takes an argument specifying the type of frequency distribution (density, cumulative, etc.) and an array argument specifying the upper limits of the "bins" for which you want to obtain frequency data. It yields a DoubleMatrix object that takes two subscripts: the index (starting from 0) of the cell in the range represented by the Variable or Function object, and the index (starting from 0) of the frequency bin.

The example code below illustrates the use of the Get Frequency method to create 14 different "bins" for categorizing the values of the uncertain functions for each simulation trial.

```
      For i = 0 To prob.Functions.Count - 1
           For j = 0 To prob.Functions(i).Size - 1
                 Dim binlimits(14) As Double
                 binlimits(0) = -125000
           For Count = 1 To 13
                         binlimits(Count) =
                         binlimits(Count - 1) + 25000
                 Next
                    Dim mymat As New RSP.DoubleMatrix
                    Set mymat =
prob.Functions(i).GetFrequency(Frequency_Type_Density
,                     binlimits)
                    For Count = 0 To 13
                            MsgBox "Upper Bin Limit"
& ": " & binlimits(Count) & " = " &
mymat(0, Count)
                 Next Count
           Next j
      Next i
```

Controlling Simulation Parameters in VBA

Analytic Solver provides a number of parameters you can use to control the simulation process, such as the number of Monte Carlo trials to perform, or the random number seed. You can set these parameters interactively, or by setting certain Solver properties or Engine.Params properties in VBA.

For example, the first line below sets the number of Monte Carlo trials to 5000. The second line set the random number seed to a fixed value – so the same random number sequence is used on each run. The third line sets the sampling method to Latin Hypercube:

```
prob.Solver.NumTrials = 5000
prob.Engine.Params("RandomSeed") = 12345
prob.Engine.Params("SamplingMethod") = 2
```

For a list of simulation parameters that you can set in VBA, see the descriptions of the "EngineParam Object" and the "ModelParam Object" in the Frontline Solvers Reference Guide.

Evaluators Called During the Simulation Process

You can write a VBA function that Analytic Solver will call at certain points during the simulation process. In this "callback function," you can access information about the problem and simulation so far, to monitor or report progress and decide whether to stop or continue the simulation process.

The object-oriented API defines an Evaluator object that is associated with your "callback function" and specifies when Analytic Solver should call it. The VBA function you write to serve as an Evaluator must be contained in a **class module** – not a 'regular' VBA module – and it must be declared to have the **WithEvents** property.

Here is an example of code for an Evaluator, in a class module named Class1:

```
Private WithEvents EvalSim As RSP.Evaluator

Private Function EvalSim_Evaluate _
  (ByVal Evaluator As RSP.IEvaluator) As _
  RSP.Engine_Action

  MsgBox "Current Simulation = " _
    & Evaluator.Problem.Engine.Stat.Simulations

  EvalSim_Evaluate = Engine_Action_Continue
End Function

Public Sub MySim()
  Set EvalSim = New RSP.Evaluator
  Dim prob As New RSP.Problem
  prob.Init ActiveWorkbook
  prob.Evaluators(Eval_Type_Simulation) = EvalSim
  prob.Solver.Simulate
  Set EvalSim = Nothing
End Sub
```

Having created the class module Class1, in a 'regular' VBA module you can create an instance of Class1, and then call the MySim method in Class1:

```
Private Sub CommandButton1_Click()
  Dim c As New Class1
  c.MySim
End Sub
```

You can define Evaluators to be called when the random sample for all uncertain variables is generated (Eval_Type_Sample), after each simulation is completed (Eval_Type_Simulation), or after each trial is completed (Eval_Type_Trial). However, because PSI Technology *evaluates all trials in parallel*, an Evaluator to be called on each trial is meaningful only if you use the Excel Interpreter rather than the PSI Interpreter to calculate the worksheet on each trial.

Working with Trials and Simulations in VBA

Analytic Solver can perform multiple simulations in one run, where each simulation consists of a number of Monte Carlo trials that you specify. Multiple simulations can be run whenever you choose **Simulate – Run Once**, whenever you change a number with Interactive Simulation, or whenever you call the Problem.Solver.Simulate method in VBA.

In VBA, you can set the property **Problem.Solver.NumSimulations** = *n* to perform *n* different simulations on a single call to Problem.Solver.Simulate.

After a simulation is run, Analytic Solver Simulation can display each Monte Carlo trial from the simulation on the Excel worksheet, if desired. In VBA, you can set the property **Problem.Solver.TrialIndex** = *n* to display the *n*th trial, or you can call the method **Problem.Solver.TrialStep** to cycle through the trials.

Displaying Normal or Error Trials

Analytic Solver **filters out** 'error trials' from your simulation results when computing statistics and displaying charts. An 'error trial' is a Monte Carlo trial where *any* uncertain function returns an Excel error value. However, all trials are saved in memory, and you can access the values of uncertain variables and uncertain functions for each trial, or display each trial on the Excel worksheet.

The Variable and Function objects contain an embedded Statistics object (described later in this chapter). The property **Statistics.NumValues** gives the number of 'normal' trials, over which statistics were calculated, and property **Statistics.NumErrors** gives the number of error trials that occurred.

You can cause a specific Monte Carlo trial to be displayed on the Excel worksheet by setting the property **Problem.Solver.TrialIndex** = *n* to the index (starting from 1) of the trial you want. This causes the PSI Distribution function for each uncertain variable to return the sample value it had on the *n*th trial. The worksheet is then recalculated, so that each uncertain function will have the value it had on the *n*th trial.

To "step through" all trials, normal trials only, or error trials only, and display them on the Excel worksheet, call the method **Problem.Solver.TrialStep** *stepsize, trialtype*. Trialtype is 0 for all trials, 1 for normal trials, and 2 for error trials. The *stepsize* may be positive (1, 2, etc.) or negative (-1, -2, etc.). After each call to the TrialStep method, the TrialIndex property is set to the index of the trial just displayed. If stepsize is so large that it would go beyond the last trial (if positive, or the first trial if negative), the TrialIndex will be set to the last (or first) trial, and that trial will be displayed.

Using Multiple Simulations

For multiple simulations to be useful, some parameter of the model – normally something you can control – must have a different value in each individual simulation. Also, you must access statistics, percentiles, and trial data (if used) for each simulation, so you can compare the results.

If you set **Problem.Solver.NumSimulations** = *n* to perform several simulations on each run, you can use the function **PsiSimParam()**. This function takes either two arguments *lower* and *upper*, as in PsiSimParam(1,3), or one argument that's a cell range or array of numbers, such as PsiSimParam({6.0, 7.5, 9.0}). On the *n*th simulation, PsiSimParam() returns the *n*th value from its argument list, held constant for all the trials in that simulation. In calls to the **PSI**

Statistics functions, you can select the simulation for which you want results. For example, if you have an uncertain function in cell F1, you can write =PsiMean(F1,1), =PsiMean(F1,2), and =PsiMean(F1,3) to access the mean value of F1 across all the trials in the first simulation, second simulation, and third simulation, respectively.

If you leave the NumSimulations property at its default value of 1, and call **Problem.Solver.Simulate** inside a for-loop, you can set parameter values that you compute on-the-fly in cells before each simulation using the Excel Range object. You can either get the values of PSI Statistics functions through the Excel object model, or (often better) you can access the Statistics, Percentiles and AllTrials properties, or call the GetFrequency method of your Variables and Functions, to retrieve the results of that simulation.

Creating Uncertain Variables and SLURPs in VBA

The chapter "Mastering Simulation and Risk Analysis Concepts" describes Stochastic Library Units, Relationships Preserved or SLURPs, and shows how to create a SLURP on the spreadsheet. You can also create SLURPs in VBA, and use the SLURP trial data for uncertain variables in your spreadsheet model.

To do this, you first create a Problem object, and initialize it with the simulation model defined in your Excel workbook. At this point, the Problem's collection of Variables will contain Variable objects for each contiguous range of cells containing PSI Distribution functions in the workbook (if any). You can then:

1. Create a new Variable object in VBA.
2. Set its Name property to an unused cell range (say "Sheet1:A1:A5").
3. Set its AllTrials property to the SLURP trial data.
4. Add the Variable object to the Problem's collection of Variables.

The simulation model then behaves just as if these cells contained =PsiSlurp() function calls that referred to the SLURP trial data you supplied via VBA. The SLURP data could be generated by your VBA program, read from a database, or otherwise obtained – it never appears on the Excel spreadsheet. For example:

```
Dim prob As New RSP.Problem
  prob.Init ActiveWorkbook

  Dim trials As RSP.DoubleMatrix
  Set trials = New RSP.DoubleMatrix
  trials.InitDense 1, prob.Solver.NumTrials

  Randomize
  For i = 0 To prob.Solver.NumTrials - 1
      trials(0, i) = Rnd()
  Next i

  Dim var As New RSP.Variable
  var.VariableType = Variable_Type_Uncertain
  var.Init Range("Sheet1$A$1")
  var.AllTrials = trials

  prob.Variables.Add var
  prob.Solver.Simulate
```

The first two lines create an instance of a Problem, and initialize it with the simulation model defined in your Excel workbook. The next three lines create a

DoubleMatrix object named *trials* to hold SLURP data, and set its dimensions – just 1 x NumTrials in this case, since we're adding just one uncertain variable.

The next four lines show how the trials matrix can be initialized with newly generated values. Of course, you'd want to use a better, application-specific method – not the Rnd() function – to generate this trial data, or perhaps read the trial data from a database or external file.

In the next three lines, we create a Variable object, set its Name property to the cell address Sheet1!A1 (this must be an empty cell on the spreadsheet), and set its AllTrials property to the trial data we just generated.

In the last two lines, we add the new Variable object to the Problem's collection of Variables, and then perform a simulation.

The newly added Variable object, and the SLURP data represented by its AllTrials property, participates in the simulation model only for so long as the VBA Problem object exists – it is "transient" and is not saved in the workbook. Of course, your VBA program code *is* saved with the workbook, and it can be run at a later time to re-create the Variable object and re-generate or retrieve the SLURP data.

Mastering Conventional Optimization Concepts

Introduction

This chapter explains basic and advanced concepts of optimization, such as the types of problems you can solve, types of constraints (regular, integer, conic, alldifferent) you can specify, the nature of linear, quadratic and nonlinear functions, convex and non-convex functions, smooth and non-smooth functions, and the algorithms and methods used by Analytic Solver (or its optimization sub-set products) and plug-in Solver engines.

This chapter focuses on conventional or *deterministic* optimization models, which do not include any uncertainty. The next chapter "Mastering Simulation and Risk Analysis Concepts" explains the concepts of *stochastic* optimization models – those that do include uncertainty – and solution methods such as robust optimization, stochastic programming, and simulation optimization.

If you are using Analytic Solver for the first time, we recommend that you try out the examples described in the chapter "Examples: Conventional Optimization" before tackling this material. If you are relatively new to optimization, you may find it useful to read the first section below, "Elements of Solver Models," and then proceed to the Examples chapters. If you've been using the Solver for a while, and you'd like a more in-depth review of the mathematical relationships found in Solver models, and the optimization methods and algorithms used by the Solver, read the more advanced sections of this chapter, and the next chapter.

Elements of Solver Models

The basic purpose of the Solver is to find a *solution* – that is, values for the *decision variables* in your model – that satisfies all of the *constraints* and maximizes or minimizes the *objective function* value (if there is one). Let's examine this framework more closely.

The model you create for use with the Solver is no different from any other spreadsheet model. It consists of input values; formulas that calculate values based on the input values or on other formulas; and other elements such as formatting. You can play "what if" with a Solver model just as easily as with any other spreadsheet model. This familiar concept can be very useful when you wish to present your results to managers or clients, who are usually "spreadsheet literate" even if they are unfamiliar with Solvers or optimization.

Decision Variables and Parameters

Some of the input values may be numbers that you use, but you cannot change on your own – for example, prevailing interest rates or supplier's prices. We'll call these values *parameters* of the model. You may have several cases, scenarios, or variations of the same problem to solve, and the parameter values will change in each problem variation; you can define these parameters via the Parameters button on the Ribbon. In this chapter, we'll assume that parameter values are *certain*, but the chapter "Mastering Stochastic Optimization Concepts" will cover situations where the parameter values are *uncertain*.

Other input values may be quantities that are variable, or under your control in the course of finding a solution. We'll refer to these as the variables or *decision variables*; the Excel Solver refers to them as Changing Cells. The Solver will find optimal values for these variables or cells. Often, some of the same cell values you use to play "what if" are the ones for which you'll want the Solver to find solution values. These cells are listed in the Variables group in the Task Pane Model tab, or the Solver Parameters dialog.

The Objective Function

The quantity you want to maximize or minimize is called the *objective* or *objective function*; the Excel Solver often uses the term Set Cell for the objective. For example, this could be a calculated value for projected profits (to be maximized), or costs, risk, or error values (to be minimized). It appears under 'Objective' in the Task Pane Model tab, or the Solver Parameters dialog.

You may have a Solver model that has nothing to maximize or minimize, in which case no cell will be listed for the objective. In this situation the Solver will simply find a solution that satisfies the constraints. Typically this will be only one of many such solutions, located close to the starting values of the decision variables.

The Solver also permits you to enter a specific value that you want the objective function to achieve. This feature was originally included in the Excel Solver to match the Excel Goal Seek... command, which allows you to seek a specific value for a cell by adjusting the value of one other cell on which it depends. Using the 'Value Of' option for the objective cell has the same effect as adding an = constraint in the outlined list, with the objective cell on the left hand side and the constant value on the right hand side; again there is nothing to maximize or minimize.

There is rarely a good reason to use the Value of option. If your problem requires only a single objective cell and a single variable cell with no constraints, you can just use the Goal Seek... command. If you have nothing to maximize or minimize, we recommend that you omit the objective and enter all of your constraints in the outlined list under "Constraints."

Constraints

Constraints are relations such as A1 >= 0. A constraint is *satisfied* if the condition it specifies is true *within a small tolerance.* This is a little different from a logical formula such as =A1>=0 evaluating to TRUE or FALSE which you might enter in a cell. In this example, if A1 were -0.0000001, the logical formula would evaluate to FALSE, but with the default Solver Precision setting, the constraint would be satisfied. Because of the numerical methods used to find solutions to Solver models and the finite precision of computer arithmetic, it would be unrealistic to require that constraints like A1 >= 0 be satisfied exactly – such solutions would rarely be found.

In the Excel Solver, constraints are specified by giving a cell reference such as A1 or A1:A5 (the "left hand side"), a relation (<=, = or >=), and an expression for the "right hand side." Although Excel allows you to enter any numeric expression on the right hand side, for reasons that will be explained in the chapter "Best Practices for Building Large-Scale Models," we strongly encourage you to use only *constants*, or references to cells that contain *constant values* on the right hand side. (A constant value to the Solver is any value that does not depend on any of the decision variables.)

A constraint such as A1:A5 <= 10 is shorthand for A1 <= 10, A2 <= 10, A3 <= 10, A4 <= 10, A5 <=10. A constraint such as A1:A5 <= B1:B5 is shorthand for A1 <= B1, A2 <= B2, A3 <= B3, A4 <= B4, A5 <= B5.

Another type of constraint is of the form A1:A5 = integer, where A1:A5 are decision variables. This specifies that the solution values for A1 through A5 must be integers or whole numbers, such as -1, 0 or 2, *to within a small tolerance*. This form of constraint, and related forms such as A1:A5 = binary, A1:A5 = semicontinuous, and A1:A5 = alldifferent, are explored in the section "More About Constraints."

A new type of constraint supported by Analytic Solver is of the form A1:A5 = conic, where A1:A5 are decision variables. This is called a *second order cone* constraint and is further described in "More About Constraints."

Solutions: Feasible, "Good" and Optimal

A solution (set of values for the decision variables) for which all of the constraints in the Solver model are satisfied is called a *feasible solution.* In some problems, a feasible solution is already known; in others, finding a feasible solution may be the hardest part of the problem.

An *optimal solution* is a feasible solution where the objective function reaches its maximum (or minimum) value – for example, the most profit or the least cost. A *globally optimal solution* is one where there are no other feasible solutions with better objective function values. A *locally optimal solution* is one where there are no other feasible solutions "in the vicinity" with better objective function values – you can picture this as a point at the top of a "peak" or at the bottom of a "valley" which may be formed by the objective function and/or the constraints.

The Solver is designed to find feasible and optimal solutions. In the best case, it will find the globally optimal solution – but this is not always possible. In other cases, it will find a locally optimal solution, and in still others, it will stop after a certain amount of time with the best solution it has found so far. But like many users, you may decide that it's most important to find a *good solution* – one that is better than the solution, or set of choices, you are using now.

The kind of solution the Solver can find depends on the nature of the mathematical relationships between the variables and the objective function and constraints (and the solution algorithm used). As explained below, if your model is ***smooth convex***, you can expect to find a globally optimal solution; if it is smooth but ***non-convex***, you will usually be able to find a locally optimal solution; if it is ***non-smooth***, you may have to settle for a "good" solution that may or may not be optimal.

Below, we summarize the capabilities of the five Solver engines bundled with Analytic Solver Comprehensive and Analytic Solver Optimization within both Analytic Solver Desktop and the Cloud app: the LP/Quadratic Solver, SOCP Barrier Solver, nonlinear GRG Solver, Interval Global Solver, and Evolutionary Solver. (Analytic Solve Upgrade uses the LP/Quadratic Solver, the nonlinear GRG Solver, and the Evolutionary Solver.) Later sections of this chapter provide an overview of the optimization methods and algorithms employed by each of these Solver engines.

Linear Simplex and LP/Quadratic Solver

The Simplex LP Solver with the Excel Solver finds optimal solutions to problems where the objective and constraints are all linear functions of the variables. (The term *linear function* is explained below, but you can imagine its graph as a straight line.) Since all linear functions are ***convex***, the Solver normally can find the *globally optimal solution*, if one exists. Because a linear function (a straight line) can always be increased or decreased without limit, the optimal solution is always determined by the constraints; there is no natural "peak" or "valley" for the objective function itself.

In Analytic Solver Comprehensive, Analytic Solver Optimization and Analytic Solver Upgrade, the linear Simplex Solver is extended to the LP/Quadratic Solver. This Solver handles problems where the constraints are all linear, and the objective may be linear or quadratic (explained further below). If the quadratic objective function is ***convex*** (if minimizing, or ***concave*** if maximizing) the Solver will normally find a globally optimal solution. If the objective is ***non-convex*** (further explained below), the Solver will find only a locally optimal solution.

SOCP Barrier Solver

The SOCP Barrier Solver, included in all optimization products, finds optimal solutions to problems where the objective and constraints are all linear or convex quadratic functions of the variables. (This is in contrast to the LP/Quadratic Solver, which permits only the objective function to be quadratic.) It also finds optimal solutions to problems with a linear objective, linear constraints, and *second order cone* (SOC) constraints; this is called a second order cone programming (SOCP) problem, as explained further below. Since all linear functions and SOC constraints are ***convex***, the SOCP Barrier Solver normally finds a *globally optimal solution*, if one exists.

Nonlinear GRG Solver

The nonlinear GRG Solver (in all Analytic Solver products except Analytic Solver Simulation) finds optimal solutions to problems where the objective and constraints are all smooth (***convex*** or ***non-convex***) functions of the variables. (The term *smooth function* is explained below, but you can imagine a graph – whether straight or curved – that contains no "breaks.") For non-convex problems, the Solver normally can find a *locally optimal solution*, if one exists – but this may or may not be the globally optimal solution. A nonlinear objective function can have a natural "peak" or "valley," but in most problems the optimal solution is partly or wholly determined by the constraints. The nonlinear GRG Solver can be used on problems with all-linear functions, but it is much less effective and efficient than the LP/Quadratic Solver or the SOCP Barrier Solver on such problems.

If you use multistart methods for global optimization with the nonlinear GRG Solver, you will have a better chance (but not a guarantee) of finding the globally optimal solution. The idea behind multistart methods is to automatically start the Solver from a variety of starting points, to find the best of the locally optimal solutions – ideally the globally optimal solution. These methods are more fully described (and contrasted with other methods for global search) below under "Global Optimization" and in the chapter "Solver Engine Option Reference" in the Frontline Solvers Reference Guide.

Interval Global Solver

The Interval Global Solver, included in all optimization products, finds *globally optimal solutions* to problems where the objective and constraints are all smooth (***convex*** or ***non-convex***) functions of the variables. Unlike the Evolutionary Solver or the GRG Solverswith multistart methods, the Interval Global Solver is normally able to determine *for certain* that the solution is globally optimal. The tradeoff is that the Interval Global Solver usually takes much more time to solve a given problem than the GRG Solver, and this time rises steeply as the number of variables and constraints in the problem increases. Hence, the Interval Global Solver is practically able to solve only smaller problems, compared to the GRG Solver.

Evolutionary Solver

The Evolutionary Solver in Analytic Solver, and all subsets except Analytic Solver Simulation, usually finds *good solutions* to problems where the objective and constraints include non-smooth or discontinuous functions of the variables – in other words, where there are no restrictions on the formulas that are used to compute the objective and constraints. For example, if your model uses IF, LOOKUP or similar functions of the variables, it's likely that the graphs of these functions will contain "jumps" or "breaks." For this class of problems, the Solver will return the best feasible solution (if any) that it can find in the time allowed.

The Evolutionary Solver can be used on problems with all-smooth functions that may have multiple locally optimal solutions, in order to seek a globally optimal solution, or simply a better solution than the one found by the nonlinear GRG Solver alone; however, the Interval Global Solver or the combination of multistart methods and the GRG Solver are likely to do as well or better than the Evolutionary Solver on such problems. It can be used on problems with smooth convex functions, but it is usually less effective and efficient than the nonlinear GRG oSolver on such problems. Similarly, it can be used on problems with all-linear functions, but there is little point in doing so when the Simplex (Excel Solver), LP/Quadratic, or SOCP Barrier Solver is available.

More About Constraints

This section explains in greater depth the role of certain types of constraints, including bounds on the decision variables, equality and inequality constraints, second order cone constraints, and different forms of integer constraints.

Bounds on the Variables

Constraints of the form A1 >= -5 or A1 <= 10 (for example), where A1 is a decision variable, are called *bounds on the variables* and are treated specially by the Solver. These constraints affect only one variable, whereas general constraints have an indirect effect on several variables that have been used in a formula such as A1+A2. Each of the Solver engines takes advantage of this fact to handle bounds on the variables more efficiently than general constraints.

The most common type of bound on a variable is a lower bound of zero (A1 >= 0), which makes the variable non-negative. Many variables represent physical quantities of some sort, which cannot be negative. As a convenience, most Solver Engines offer an option "Assume Non-Negative," which automatically places a lower bound of zero on every variable which has not been given an explicit lower bound in the model outline. If you need bounds other than zero, the Task Pane Platform tab Default Bounds section lets you set default *lower* and *upper* bounds on every decision variable.

Regardless of the Solver engine chosen, bounds on the variables always help speed up the solution process, because they limit the range of values that the Solver must explore. In many problems, you will be aware of realistic lower and upper bounds on the variables, but they won't be of any help to the Solver unless you include them in the Constraints list box! Bounds on the variables are especially important to the performance of the Evolutionary Solver, the Interval Global Solver, and multistart methods for global optimization. They are also very important if you want the Solver to automatically transform your model, replacing non-smooth functions (such as IF) with additional variables and linear constraints.

Equations and Inequalities

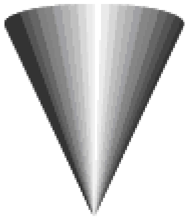
Constraints such as A1 = 0 are called *equality constraints* or *equations*; constraints such as A1 <= 0 are called *inequality constraints* or simply *inequalities*. An equality is much more restrictive than an inequality. For example, if A1 contains the formula =C1+C2, where C1 and C2 are decision variables, then A1 <= 0 restricts the possible solutions to a *half plane*, whereas A1 = 0 restricts the solutions to a *line* where all possible values of C1and C2 must sum to 0 (C1 = **-**C2 within a small tolerance, as explained above). Since there is only a tiny chance that two randomly chosen values for C1 and C2 will satisfy C1+C2 = 0, solution methods that rely on random choices, such as genetic algorithms, may have a hard time finding any feasible solutions to problems with equality constraints. To satisfy equality constraints, the Solver generally must exploit properties of the constraint formula – such as *linearity* or *smoothness*, discussed below – to solve for one variable in terms of another.

A linear equality constraint (like C1+C2 = 0 above) maintains the convexity of the overall problem, but a nonlinear equality constraint is ***non-convex***, and makes the overall problem non-convex. Interior point methods may have difficulty solving problems with nonlinear equality constraints, since they restrict the ability of the Solver to follow the "central path" inside the feasible region.

A problem with *only* equality constraints (and no objective) is sometimes called a *system of equations*. The Solver can be used to find solutions to systems of both linear and nonlinear equations. If there are several different solutions (sets of values for the decision variables) that satisfy the equations, most Solver engines will find just one solution that is "close" to the starting values of the variables; but the Interval Global Solver can be used to find *all* real solutions to a system of smooth nonlinear equations – a capability that was once felt to be beyond the limits of any known algorithm.

Second Order Cone Constraints

Analytic Solver supports constraints of the form A1:A5 = conic. This is called a *second order cone* (SOC) constraint; it specifies that the vector formed by the decision variables A1:A5 must lie within the second-order cone (also called the Lorentz cone, or "ice cream cone") of dimension 5 – a ***convex*** set that looks like the figure below in three dimensions.



Algebraically, a second-order cone constraint specifies that, given a value for one variable, the *L2*-norm of the vector formed by the remaining variables must not exceed this value: In linear algebra notation, $a_1 \geq \|a_2{:}a_5\|_2$. In Excel, this could be written as A1 >= SQRT(SUMSQ(A2:A5)). You can also use a variant called a “rotated second order cone” constraint. A problem with a linear objective and linear or SOC constraints is called a *second order cone programming* (SOCP) problem; it is always a ***convex*** optimization problem.

Decision variables that are constrained to be non-negative also belong to a cone, called the *non-negative orthant*. A problem with all linear functions – a linear programming problem – is a special case of an SOCP problem, where the only cone constraint is non-negativity.

A convex quadratic objective or constraint can be transformed into an equivalent second order cone constraint. Hence, a problem with a quadratic objective – a quadratic programming or QP problem – or a problem with quadratic constraints – called a QCP problem – is also a special case of an SOCP problem. The SOCP Barrier Solver and the MOSEK Solver will automatically transform quadratics into SOC form internally; you can simply define your quadratic objective and/or constraints using ordinary Excel formulas and <= or >= relations, and use these Solver engines to obtain fast, reliable, globally optimal solutions to your problem.

Integer and Binary Constraints

As explained in the last section, integer constraints are of the form A1:A5 = integer, where A1:A5 are decision variables. This specifies that the solution values for A1 through A5 must be integers or whole numbers, such as -1, 0 or 2, to within a small tolerance. A common special case that can be entered directly in the Add Constraint dialog is A1 = binary, which is equivalent to specifying A1 = integer, A1 >= 0 and A1 <= 1. This implies that A1 must be *either 0 or 1* at the solution; hence A1 can be used to represent a “yes/no” decision. Integer constraints have many important applications, but the presence of even one such constraint in a Solver model makes the problem an integer programming problem (discussed below), which may be much more difficult to solve than a similar problem without the integer constraint.

Semi-Continuous Constraints

It is often useful to place a *semi-continuous* constraint on a decision variable. This specifies that, at the solution, the variable must be either 0, or else a continuous value within a range, determined by the bounds on the variable. For example, if a machine is either “off” or running at a speed between 5 and 50, you can model the machine’s speed with A1 = semicontinuous, A1 >= 5 and A1 <= 50. In situations where you might need a binary integer variable *and* a regular (continuous) variable, you can sometimes use a single semi-continuous variable instead – and such a variable can be handled very efficiently by most Solver engines.

Alldifferent Constraints

A special type of integer constraint supported by Analytic Solver and its subset products is called an "alldifferent" constraint. Such a constraint is of the form (for example) A1:A5 = alldifferent, where A1:A5 is a group of decision variables, and it specifies that these variables must be integers in the range 1 to N (N = 5 in this example), with each variable different from all the others at the solution. Hence, A1:A5 will contain a *permutation* of integers, such as 1,2,3,4,5 or 1,3,5,2,4. The alldifferent constraint can be used to model problems involving ordering of choices, such as the Traveling Salesman Problem.

Functions of the Variables

Since there are large differences in the time it takes to find a solution and the *kinds* of solutions – globally optimal, locally optimal, or simply "good" – that you can expect for different types of problems, it pays to understand the differences between linear, quadratic, smooth nonlinear, and non-smooth functions, and especially **convex** and **non-convex** functions. To begin, let's clarify what it means to say that the spreadsheet cells you select for the objective and constraints are "functions of the decision variables."

The objective function in a Solver problem is a cell calculating a value that depends on the decision variable cells; the job of the Solver is to find some combination of values for the decision variables that maximizes or minimizes this cell's value. During the optimization process, *only the decision variable cells are changed*; all other "input" cells are held constant. If you analyze the chain of formulas that calculates the objective function value, you will find that parts of those formulas (those which refer to non-decision variable cells) are unchanging in value and could be replaced by a numeric constant for the purposes of the optimization.

If you have constant values on the right hand sides of constraints, then the same observation applies to the left hand sides of constraints: Parts of the constraint formulas (those which refer to non-decision variable cells) are unchanging in value, and only the parts that are dependent on the decision variables "count" during the optimization.

When you consider whether your objective and constraints are linear, quadratic, smooth nonlinear, or non-smooth, or **convex** or **non-convex** functions of the variables, always bear in mind that only the parts of formulas that are *dependent on the decision variables* "count." Below, we explain that linear functions are most desirable, and non-smooth and non-convex functions are least desirable in a Solver model (if you want the fastest and most reliable solutions). A formula such as =IF(C1>=10,D1,2*D1) is non-smooth if C1 depends on the decision variables; but if C1 *doesn't* depend on the variables, then only D1 or 2*D1 – not both – can be selected during the solution process. Hence if D1 is a linear function of the variables, then the IF expression is also a linear function of the variables.

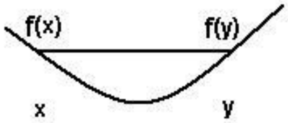
You may also find that a function that is "bad" (non-smooth or non-convex) over its full domain (any possible values for the decision variables) can still be "good" (smooth and/or convex) over the domain of interest to you, determined by other constraints including bounds on the variables. For example, if C1 depends on the variables, then =IF(C1>=10,D1,2*D1) is non-smooth over its full domain, but smooth – in fact linear – if C1 is constrained to be 10 or more. =SIN(C1) is non-convex over its full domain, but is convex from –pi to 0, or from pi to 2*pi.

Convex Functions

The key property of functions of the variables that makes a problem "easy" or "hard" to solve is *convexity*. If *all* constraints in a problem are convex functions of the variables, and if the objective is convex if minimizing, or concave if maximizing, then you can be confident of finding a globally optimal solution (or determining that there is no feasible solution), even if the problem is very large – thousands to hundreds of thousands of variables and constraints.

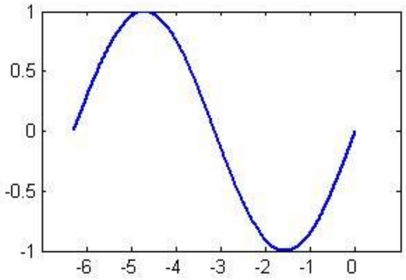
In contrast, if *any* of the constraints are non-convex, or if the objective is either non-convex, concave if minimizing, or convex if maximizing, then the problem is far more difficult: You cannot be certain of finding a feasible solution even if one exists; you must either "settle for" a locally optimal solution, or else be prepared for very long solution times and rather severe limits on the size of problems you can solve to global optimality (a few hundred to perhaps one thousand variables and constraints), even on the fastest computers. So it pays to understand convexity!

Geometrically, a function is *convex* if, at any two points x and y, the line drawn from x to y (called the *chord* from x to y) lies *on or above* the function – as shown in the diagram below, for a function of one variable. A function is *concave* if the chord from x to y lies *on or below* the function. This property extends to any number of 'dimensions' or variables, where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$.



Algebraically, a function *f* is *convex* if, for any points x and y, and any t between 0 and 1, $f(tx + (1-t)y) <= tf(x) + (1-t)f(y)$. A function *f* is *concave* if $-f$ is convex, i.e. if $f(tx + (1-t)y) >= tf(x) + (1-t)f(y)$. A *linear* function – described below – is both convex and concave: The chord from x to y lies on the line, and $f(tx + (1-t)y) = tf(x) + (1-t)f(y)$. As we'll see, a problem with all linear functions is the simplest example of a convex optimization problem that can be solved efficiently and reliably to very large size.

A non-convex function "curves up and down." A familiar example is the sine function (SIN(C1) in Excel), which is pictured below.



The feasible region of an optimization problem is formed by the intersections of the constraints. The intersection of several convex constraints is always a convex region, but even one non-convex function can make the whole region non-convex – and hence make the optimization problem far more difficult to solve.

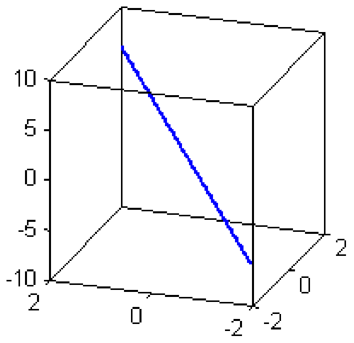
Linear Functions

In many common cases, the objective and/or constraints are *linear* functions of the variables. This means that the function can be written as a sum of terms, where each term consists of one decision variable multiplied by a (positive or negative) constant. Algebraically, we can write:

$a_1x_1 + a_2x_2 + ... + a_nx_n$

where the a_is, which are called the *coefficients*, stand for constant values and the x_is stand for the decision variables. A common example is =SUM(C1:C5), where C1:C5 are decision variables and the a_is are all 1. Note that a linear function does not have to be written in exactly the form shown above on the spreadsheet. For example, if cells C1 and C2 are decision variables, B1 = C1+C2, and B2 = A1*B1 where A1 is constant in the problem, then B2 is a linear function (=A1*C1+ A1*C2).

Geometrically, a linear function is always a straight line, in *n*-dimensional space where *n* is the number of decision variables. On the next page is a perspective plot of $2x_1 + 1x_2$. As noted above, a linear function is always convex.



Remember that the a_is need only be *constant in the optimization problem*, i.e. not dependent on any of the decision variables. For example, suppose that the function is =B1/B2*C1 + (D1*2+E1)*C2, where only C1 and C2 are decision variables, and the other cells contain constants (or formulas that don't depend on the variables). This would still be a linear function, where a_1 = B1/B2 and a_2 = (D1*2+E1) are the coefficients, and x_1 = C1 and x_2 = C2 are the variables.

Note that the SUMPRODUCT and DOTPRODUCT functions compute exactly the algebraic expression shown above. If we were to place the formula =B1/B2 in cell A1, and the formula =(D1*2+E1) in cell A2, then we could write the example function above as:

=SUMPRODUCT(A1:A2,C1:C2)

This is simple and clear, and is also useful for *fast problem setup* as described in the chapter "Best Practices for Building Large-Scale Models." If the decision variable cells that should participate in the expression are not all contiguous on the spreadsheet, the DOTPRODUCT function can be used instead of SUMPRODUCT.

As explained below in the section "Derivatives, Gradients, Jacobians and Hessians," each coefficient a_i in the linear expression $a_1x_1 + a_2x_2 + ... + a_nx_n$ is the first partial derivative of the expression with respect to variable x_i. These partial derivatives are always *constant* in a linear function – and all higher-order derivatives are zero.

A *nonlinear* function (explained further below), as its name implies, is any function of the decision variables which is not linear, i.e. which cannot be written in the algebraic form shown above – and its partial derivatives are *not* constant. Examples would be = 1/C1, =LOG(C1), =C1^2 or =C1*C2 where both C1 and C2 are decision variables. If the objective function or any of the constraints are nonlinear functions of the variables, then the problem cannot be solved with an LP Solver.

Testing for a Linear Model

What if you have already created a complex spreadsheet model without using functions like SUMPRODUCT, and you aren't sure whether your objective function and constraints are linear or nonlinear functions of the variables? If you have Analytic Solver, you can easily find out by pressing the Analyze button in the Task Pane. Moreover, you can easily obtain a report showing exactly which cells contain formulas that are nonlinear.

You can try solving the model with the standard LP/Quadratic Solver. If the problem contains nonlinear functions of the variables, you will (in virtually all cases) receive the message "The linearity conditions required by this Solver engine are not satisfied." You can then ask the Solver to produce a Linearity Report, which shows whether the objective and each of the constraints is a linear or nonlinear function of the variables. This report also shows which variables occur linearly, and which occur nonlinearly in your model – another way of summarizing the same information. You should next look closely at the objective or constraint formulas that the Linearity Report indicates are nonlinear, and decide whether (or not) the formula can be written in linear form.

Quadratic Functions

The last two examples of nonlinear functions above, =C1^2 or =C1*C2, are simple instances of *quadratic* functions of the variables. A more complex example is:

=2*C1^2+3*C2^2+4*C1*C2+5*C1

A quadratic function is a sum of terms, where each term is a (positive or negative) constant (again called a *coefficient*) multiplied by a single variable or the product of two variables. In linear algebra notation, we can write $x^TQx + cx$ where x is a vector of *n* decision variables, Q is an *n* x *n* matrix of coefficients, and c is an *n* vector of linear coefficients. The QUADPRODUCT function computes values of exactly this form. If we put the constant 5 in A1, 0 in B1, 2 in A2, 4 in B2, 0 in A3 and 3 in B3, then we could write the above example as:

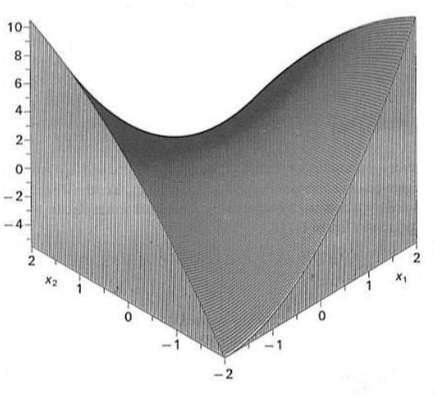
=QUADPRODUCT(C1:C2,A1:B1,A2:B3)

Common uses for quadratic functions are to compute the mean squared error in a curve-fitting application, or the variance or standard deviation of security returns in a portfolio optimization application.

As explained below in the section "Derivatives, Gradients, Jacobians and Hessians," the coefficients that multiply single variables in a quadratic function are the first partial derivatives of the function with respect to those variables; the coefficients that multiply the *products* of two variables are the *second* partial derivatives of the function, with respect to those two variables. In a quadratic function, these first and second order derivatives are always *constant*, and higher order derivatives are zero. The matrix Q of second partial derivatives is called the *Hessian* of the function.

Convex, Concave and Non-Convex Quadratics

A quadratic function of at least two variables may be convex, concave, or non-convex. The matrix Q in the general form x^TQx has a closely related algebraic property of *definiteness*. If the Q matrix is *positive definite*, the function is convex; if the Q matrix is *negative definite*, the function is concave. You can picture the graph of these functions as having a "round bowl" shape with a single bottom (or top). If the Q matrix is *semi-definite,* the function has a bowl shape with a "trough" where many points may have the same objective value, but it is still convex or concave. If the Q matrix is *indefinite*, the function is **non-convex**: It has a "saddle" shape, but its true minimum or maximum is not found in the "interior" of the function but on its boundaries with the constraints, where there may be many locally optimal points. Below is a plot of an example non-convex quadratic $x_1^2 + 2x_1x_2 - \frac{1}{2}(x_2^2 - 1)$:



A problem with **convex** quadratic functions is easily solved to global optimality up to very large size, but a problem with **non-convex** quadratic functions is a difficult global optimization problem that, in general, will require solution time that grows *exponentially* with the number of variables. The way that the Solver handles such functions is explained further below under "Quadratic Programming."

Nonlinear and Smooth Functions

A *nonlinear* function is any function of the variables that is not linear, i.e. which cannot be written in the algebraic form:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n$$

Examples, as before, are =1/C1, =LOG(C1), and =C1^2, where C1 is a decision variable. All of these are called *continuous* functions, because their graphs are curved but contain no "breaks." =IF(C1>10,D1,2*D1) is also a nonlinear function, but it is "worse" (from the Solver's viewpoint) because it is *discontinuous*: Its graph contains a "break" at C1=10 where the function value jumps from D1 to 2*D1. At this break, the rate of change (i.e. the derivative) of the function is undefined. As explained below in the section "Derivatives, Gradients, Jacobians and Hessians," most Solver algorithms rely on derivatives to seek improved solutions, so they may have trouble with a Solver model containing functions like =IF(C1>10,D1,2*D1). The Interval Global Solver does not accept discontinuous functions at all.

If the graph of the function's *derivative* also contains no breaks, then the original function is called a *smooth* function. If it does contain breaks, then the original function is *non-smooth*. Every discontinuous function is also non-smooth. An example of a continuous function that is non-smooth is =ABS(C1) – its graph is an unbroken “V” shape, but the graph of its derivative contains a break, jumping from –1 to +1 at C1=0. Many nonlinear Solver algorithms rely on second order derivatives of at least the objective function to make faster progress, and to test whether the optimal solution has been found; they may have trouble with functions such as =ABS(C1). The Interval Global Solver does not accept any non-smooth functions.

As explained below in the section “Derivatives, Gradients, Jacobians and Hessians,” general nonlinear functions have first, second, and sometimes higher order derivatives that *change* depending on the point (i.e. values of the decision variables) at which the function is evaluated.

Convex, Concave and Non-Convex Smooth Functions

A general nonlinear function of even one variable may be convex, concave or non-convex. A function can be convex but non-smooth: =ABS(C1) with its V shape is an example. A function can also be smooth but non-convex: = SIN(C1) is an example. But the “best” nonlinear functions, from the Solver’s point of view, are both *smooth* and *convex* (concave for the objective if you are maximizing).

If a smooth function’s second derivative is always nonnegative, it is a *convex* function; if its second derivative is always nonpositive, it is a *concave* function. This property extends to any number of ‘dimensions’ or variables, where the second derivative becomes the Hessian and “nonnegative” becomes “positive semidefinite.”

Discontinuous and Non-Smooth Functions

Microsoft Excel provides a very rich formula language, including many functions that are discontinuous or non-smooth. As noted above, discontinuous functions cause considerable difficulty, and non-smooth functions cause some difficulty for most nonlinear Solvers; such functions are not accepted by the Interval Global Solver. Some models can only be expressed with the aid of these functions; in other cases, you have a degree of choice in how you model the real-world problem, and which functions you use. Even when you have a “full arsenal” of Solver engines available, as you do with Analytic Solver Comprehensive and its subset products, you’ll get better results if you try to use the most “Solver-friendly” functions in your model.

By far the most common discontinuous function in Excel is the IF function where the conditional test depends on the decision variables, as in the example =IF(C1>10,D1,2*D1). Here is a short list of common *discontinuous* Excel functions:

IF, CHOOSE
LOOKUP, HLOOKUP, VLOOKUP
COUNT
INT, ROUND
CEILING, FLOOR

Here is a short list of common *non-smooth* Excel functions:

ABS
MIN, MAX

Formulas involving relations such as <=, = and >= (on the worksheet, not in constraints) and logical functions such as AND, OR and NOT are discontinuous at their points of transition from FALSE to TRUE values. Functions such as

SUMIF and the database functions are discontinuous if the criterion or conditional argument depends on the decision variables.

If you aren't sure about a particular function, try graphing it (by hand or in Microsoft Excel) over the expected range of the variables; this will usually reveal whether the function is discontinuous or non-smooth. If using Analytic Solver, just create a model using the function, and use the Analyze button to automatically diagnose the model type.

Analytic Solver Comprehensive, Analytic Solver Optimization and Analytic Solver Upgrade can **automatically transform** a model that uses IF, AND, OR, NOT, ABS, MIN and MAX, and relations <, <=, >= and > to an equivalent model where these functions and relations are replaced by additional binary integer and continuous variables and additional constraints, that have the same effect – for the purpose of optimization – as the replaced functions. This powerful facility may be able to transform your non-smooth model into a smooth or even linear model with integer variables. An example is shown in the EXAMPLE5 worksheet of the **StandardExamples.xlsx** workbook, which is normally installed into C:\ProgramData\Frontline Systems\Examples.

Derivatives, Gradients, Jacobians, and Hessians

To find feasible and optimal solutions, most optimization algorithms rely heavily on derivatives of the problem functions (the objective and constraints) with respect to the decision variables. First derivatives indicate the direction in which the function is increasing or decreasing, while second derivatives provide curvature information.

The partial derivatives of a function $f(x_1,x_2,...,x_n)$ with respect to each variable are denoted $\partial f/\partial x_1$, $\partial f/\partial x_2$, …, $\partial f/\partial x_n$. They give the rate of change of the function in each dimension. For a linear function $a_1x_1 + a_2x_2 + ... + a_nx_n$, the partial derivatives are the coefficients: $\partial f/\partial x_1 = a_1$, $\partial f/\partial x_2 = a_2$, and so on.

To recap the comments about derivatives made in the sections above:

- Linear functions have *constant* first derivatives – the coefficients a_i – and all higher order derivatives (second, third, etc.) are zero.
- Quadratic functions have *constant* first and second derivatives, and all higher order (third, etc.) derivatives are zero.
- Smooth nonlinear functions have first and second derivatives that are *defined*, but not constant – they change with the point at which the function is evaluated.
- Non-smooth functions have second derivatives that are *undefined* at some points; discontinuous functions have first derivatives that are undefined at some points.

The *gradient* of a function $f(x_1,x_2,...,x_n)$ is the vector of its partial derivatives:

$$[\ \partial f/\partial x_1,\ \partial f/\partial x_2,\ \ldots,\ \partial f/\partial x_n\ ]$$

This vector points in the direction (in n-dimensional space) along which the function increases most rapidly. Since a Solver model consists of an objective and constraints, all of which are functions of the variables $x_1,x_2,...,x_n$, it is often useful to collect these gradients into a matrix, where each row is the gradient vector for one function:

$$\left| \begin{array}{l} \partial f_1/\partial x_1,\ \ \partial f_1/\partial x_2,\ \ \ldots,\ \ \partial f_1/\partial x_n \\ \partial f_2/\partial x_1,\ \ \partial f_2/\partial x_2,\ \ \ldots,\ \ \partial f_2/\partial x_n \\ \ldots \\ \partial f_m/\partial x_1,\ \ \partial f_m/\partial x_2,\ \ \ldots,\ \ \partial f_m/\partial x_n \end{array} \right|$$

This matrix is called the *Jacobian* matrix. In a linear programming problem, this is the LP coefficient matrix, and all of its elements (the a_{ij}s) are constant.

The second partial derivatives of a function $f\,(x_1,x_2,...,x_n)$ with respect to each pair of variables x_i and x_j are denoted $\partial^2 f\,/\,\partial x_i \partial x_j$. There are n^2 second partial derivatives, and they can be collected into an n x n matrix:

$$\left| \begin{array}{l} \partial^2 f/\partial x_1\partial x_1,\ \ \partial^2 f/\partial x_1\partial x_2,\ \ \ldots,\ \ \partial^2 f/\partial x_1\partial x_n \\ \partial^2 f/\partial x_2\partial x_1,\ \ \partial^2 f/\partial x_2\partial x_2,\ \ \ldots,\ \ \partial^2 f/\partial x_2\partial x_n \\ \ldots \\ \partial^2 f/\partial x_n\partial x_1,\ \ \partial^2 f/\partial x_n\partial x_2,\ \ \ldots,\ \ \partial^2 f/\partial x_n\partial x_n \end{array} \right|$$

This matrix is called the *Hessian* matrix. It provides second order (curvature) information for a single problem function, such as the objective. The Hessian of a linear function would have all zero elements; the Hessian of a quadratic function has all *constant* elements; and the Hessian of a general nonlinear function may change depending on the point (values of the decision variables) where it is evaluated.

When reading the next section, “Optimization Problems and Solution Methods,” bear in mind that the different classes of Solver problems, and the computing time required to solve these problems, is directly related to the nature of the derivatives (constant, changing, or undefined) of their problem functions, as outlined above.

For example, because the first derivatives of linear functions are *constant*, they need be computed only once – and second derivatives (which are zero) need not be computed at all. For quadratic functions, the first and second derivatives can be computed only once, whereas for general nonlinear functions, these derivatives may have to be computed many times.

A major difference between the Analytic Solver products versus Excel Solver, is the method used to compute derivatives. The Polymorphic Spreadsheet Interpreter in Analytic Solver can supply fast, accurate derivatives to Solver engines via a process called *automatic differentiation*.

What if your optimization problem requires the use of non-smooth or discontinuous functions? With Analytic Solver, you have several choices. First, for common non-smooth functions such as ABS, MAX and MIN, and even for some IF functions, the nonlinear GRG, Large-Scale GRG and Large-Scale SQP Solvers often yield acceptable results, though you may need to use multistart methods to improve the chances of finding the optimal solution. Second, you can use the Evolutionary Solver (which does not require any derivative values) to find a “good” solution, though you’ll have to give up guarantees of finding an optimal solution, and it’s likely to take considerably more computing time to find a solution. Third, you can use the automatic transformation feature to replace many of these functions with additional variables and linear constraints; if all discontinuous or non-smooth functions in the model are automatically replaced, the problem should be solvable with the nonlinear Solvers, or even with the linear Solvers in some cases. Fourth, you can manually reformulate your model with binary integer variables and associated constraints. You can then use the nonlinear GRG Solver, or even the LP/Quadratic Solver, in combination with the Branch & Bound method, to find the *true optimal solution* to your problem. These ideas are explored further in the chapter "Best Practices for Building Large-Scale Models."

Optimization Problems and Solution Methods

A model in which the objective function and all of the constraints (other than integer constraints) are linear functions of the decision variables is called a *linear programming* (LP) problem. (The term "programming" dates from the 1940s and the discipline of "planning and programming" where these solution methods were first used; it has nothing to do with computer programming.) As noted earlier, a linear programming problem is always **convex**.

If the problem includes integer constraints, it is called an *integer linear* programming problem. A linear programming problem with some "regular" (continuous) decision variables, and some variables that are constrained to integer values, is called a *mixed-integer programming* (MIP) problem. Integer constraints are **non-convex**, and they make the problem far more difficult to solve; see below for details.

A *quadratic programming* (QP) problem is a generalization of a linear programming problem. Its objective is a **convex** *quadratic* function of the decision variables, and all of its constraints must be *linear* functions of the variables. A problem with linear and convex quadratic *constraints*, and a linear or convex quadratic objective, is called a *quadratically constrained* (QCP) problem.

A model in which the objective function and all of the constraints (other than integer constraints) are smooth nonlinear functions of the decision variables is called a *nonlinear programming* (NLP) or *nonlinear optimization* problem. If the problem includes integer constraints, it is called an *integer nonlinear* programming problem. A model in which the objective or any of the constraints are non-smooth functions of the variables is called a *non-smooth optimization* (NSP) problem.

Linear Programming

Linear programming (LP) problems are intrinsically easier to solve than nonlinear (NLP) problems. First, they are convex, where a general nonlinear problem is often non-convex. Second, since all constraints are linear, the globally optimal solution always lies at an "extreme point" or "corner point" where two or more constraints intersect. (In some problems there may be multiple solutions with the same objective value, all lying on a line between two corner points.) This means that an LP Solver needs to consider many fewer points than an NLP Solver, and it is always possible to determine (subject to the limitations of finite precision computer arithmetic) that an LP problem (i) has no feasible solution, (ii) has an unbounded objective, or (iii) has a globally optimal solution.

Problem Size and Numerical Stability

Because of their structural simplicity, the main limitations on the size of LP problems that can be solved are time, memory, and the possibility of numerical "instabilities" which are the cumulative result of the small errors intrinsic to finite precision computer arithmetic. The larger the model, the more likely it is that numerical instabilities will be encountered in solving it.

Most large LP models are *sparse* in nature: While they may include thousands of decision variables and constraints, the typical constraint will depend upon only a few of the variables. This means that the Jacobian matrix of partial derivatives of the problem functions, described earlier, will have many elements that are *zero*. Such sparsity can be exploited to save memory and gain speed in solving the problem.

The Simplex Method

LP problems are most often solved via the Simplex method. The standard Microsoft Excel Solver uses a straightforward implementation of the Simplex method to solve LP problems, when the Assume Linear Model box is checked in the Solver Options dialog. Analytic Solver Comprehensive, Analytic Solver Optimization and Analytic Solver Upgrade use a far more sophisticated implementation of the Simplex method which exploits sparsity in the LP model and uses techniques such as presolving, matrix factorization using the LU decomposition , a fast, stable LU update, and dynamic Markowitz refactorization.

The Large-Scale LP/QP Solver and MOSEK Solver engines use even more powerful implementations of the methods mentioned above. They have been used to solve LP problems with millions of variables and constraints.

The Large-Scale SQP Solver engine includes a powerful linear programming Solver that uses "active set" methods (closely related to the Simplex method). It is practical for problems up to 100,000 variables and constraints. This same Solver engine also handles large-scale QP and NLP problems very efficiently.

The Gurobi Solver and XPRESS Solver engine are Frontline's fastest and most powerful Solvers for linear programming and especially *mixed-integer* linear programming problems. Their advanced primal and dual Simplex and Barrier methods, combined with state-of-the-art Branch and Cut methods for integer problems, yield solutions in record time.

Quadratic Programming

Quadratic programming problems are more complex than LP problems, but simpler than general NLP problems. They have only one feasible region with "flat faces" on its surface (due to their linear constraints), but the optimal solution may be found anywhere within the region or on its surface. Since a QP problem is a special case of an NLP problem, it *can* be solved with the standard nonlinear GRG Solver, but this may take considerably more time than solving an LP of the same size. The LP/Quadratic Solver in Analytic Solver Comprehensive and Analytic Solver Optimization solves QP problems using a variant of the Simplex method to determine the feasible region, and special methods based on the properties of quadratic functions to find the optimal solution.

Most quadratic programming algorithms are specialized to handle only positive definite (or negative definite) quadratics. The LP/Quadratic Solver, however, can also handle semi-definite quadratics; it will find one of the equivalent (globally) optimal solutions – which one depends on the starting values of the decision variables. When applied to an indefinite quadratic objective function, the LP/Quadratic Solver provides only the guarantees of a general nonlinear Solver: It will converge to a locally optimal solution (either a saddle point in the interior, or a locally optimal solution on the constraint surface).

The Large-Scale LP/QP Solver, Large-Scale GRG Solver, Large-Scale SQP Solver, Knitro Solver, Gurobi Solver, MOSEK Solver, and XPRESS Solver engines can all be used to efficiently solve large QP problems.

Quadratically Constrained Programming

A problem with linear and convex quadratic *constraints*, and a linear or convex quadratic objective, is called a *quadratically constrained* (QCP) problem. Such a problem is more general than a QP or LP problem, but less general than a convex nonlinear problem. The Simplex-based methods used in the LP/Quadratic Solver, the Large-Scale LP/QP Solver, and the XPRESS Solver Engine handle only quadratic objectives, not quadratic constraints. But QCP problems – since they are **convex** – can be solved efficiently to global optimality with Barrier methods, also called Interior Point methods.

The SOCP Barrier Solver uses a Barrier method to solve LP, QP, and QCP problems. The MOSEK Solver Engine uses an even more powerful Barrier method to solve very large scale LP, QP, and QCP problems, as well as smooth convex nonlinear problems. Both of these Solvers form a logarithmic "barrier function" of the constraints, combine this with the objective, and take a step towards a better point on each major iteration. Unlike the Simplex method, which moves from one corner point to another on the *boundary* of the feasible region, a Barrier method follows a path – called the *central path* – that lies strictly *within* the feasible region.

A Barrier method relies heavily on second derivative information, specifically the Hessian of the Lagrangian (combination of the constraints and objective) to determine its search direction on each major iteration. The ability of the Polymorphic Spreadsheet Interpreter to efficiently compute this second derivative information is key to the performance of this method.

Second Order Cone Programming

Second order cone programming (SOCP) problems are a further generalization of LP, QP, and QCP problems. An SOCP has a linear objective and one or more linear or *second order cone* (SOC) constraints. As explained earlier, a second order cone constraint such as "A1:A5 = conic" specifies that the vector formed by the decision variables A1:A5 must lie within the second-order cone (also called the Lorentz cone) of dimension 5. Algebraically, the constraint specifies that $a_1 \geq \|a_2{:}a_5\|_2$. SOCPs are always **convex**; the SOCP Barrier Solver and the MOSEK Solver Engine are both designed to solve SOCP problems, efficiently to global optimality.

Any convex quadratic constraint can be converted into an SOC constraint, with several steps of linear algebra. A convex quadratic objective $x^TQx + cx$ can be handled by introducing a new variable *t*, making the objective minimize *t*, adding a constraint $x^TQx + cx <= t$, and converting this constraint to SOC form. The SOCP Barrier Solver and the MOSEK Solver Engine both make these transformations automatically; in effect they solve all LP, QP, QCP and SOCP problems in the same way. Second order cone programming can be viewed as the *natural generalization of linear programming*, and is bound to become more popular in the future.

You can also solve an SOCP with the GRG Nonlinear Solver or the Large-Scale GRG, Large-Scale SQP, or Knitro Solver engines. Although these Solvers do not recognize SOC constraints directly, Analytic Solver products will compute values and derivatives for SOC constraints, based on their algebraic form shown above. Hence, these general nonlinear Solvers handle SOC constraints like other general nonlinear constraints. Using these Solvers, you can find optimal solutions for problems containing a mix of linear, general nonlinear, and SOC constraints – bearing in mind that such problems may be non-convex.

Nonlinear Optimization

As outlined above, nonlinear programming (NLP) problems are intrinsically more difficult to solve than LP, QP, QCP or SOCP problems. They may be **convex** or **non-convex**, and since their second derivatives are not constant, an NLP Solver must compute or approximate the Hessians of the problem functions many times during the course of the optimization. Since a non-convex NLP may have multiple feasible regions and multiple locally optimal points within such regions, there is no simple or fast way to determine with certainty that the problem is infeasible, that the objective function is unbounded, or that an optimal solution is the "global optimum" across all feasible regions. But some NLP problems *are* convex, and many problems include linear or convex quadratic constraints in addition to general nonlinear constraints. Frontline's field-installable nonlinear Solver engines are each designed to take advantage of NLP problem structure in different ways, to improve performance.

If you use the GRG Nonlinear Solver – the only choice for NLPs in the standard Excel Solver and Analytic Solver Upgrade – bear in mind that it applies the *same* method to *all* problems, even those that are really LPs or QPs. If you don't select another Solver engine from the Task Pane Engine tab dropdown list box (or, in the standard Microsoft Excel Solver, if you don't check the Assume Linear Model box in the Solver Options dialog), this Solver will be used – and it may have difficulty with LP or QP problems that could have been solved easily with one of the other Solvers. Analytic Solver Comprehensive can automatically determine the type of problem, and select only the "good" or "best" Solver engine(s) for that problem.

The GRG Method

Analytic Solver includes the enhanced GRG Solver while the Excel Solver uses the Generalized Reduced Gradient method as implemented in Lasdon and Waren's GRG2 code. The GRG method can be viewed as a nonlinear extension of the Simplex method, which selects a basis, determines a search direction, and performs a line search on each major iteration – solving systems of nonlinear equations at each step to maintain feasibility. This method and specific implementation have been proven in use over many years as one of the most robust and reliable approaches to solving difficult NLP problems.

As with the Simplex method, the GRG method in the standard Excel Solver uses a "dense" problem representation, and its memory and solution time increases with the number of variables *times* the number of constraints. It is also subject to problems of numerical instability, which may be even more severe than for LP and QP problems. The GRG Solver engine uses sparse storage methods and better numerical methods for nonlinear models, such as matrix condition testing and degeneracy handling, to solve much larger NLP problems.

The SQP Method

The Large-Scale SQP Solver engine for Analytic Solver Comprehensive and Analytic Solver Optimization uses a Sequential Quadratic Programming (SQP) method to solve nonlinear optimization problems. This method forms and solves a QP subproblem, with a quadratic merit function and linearized constraints, on each major iteration. Because it includes a highly efficient QP solver, a powerful linear programming solver using "active set" methods, and sparsity-exploiting matrix factorization, updating and refactorization methods, the Large-Scale SQP Solver engine is very fast in solving all types of LP, QP and NLP problems. However, the SQP method typically follows a path of *infeasible* trial points until it finds the solution that is both feasible and optimal. Hence, if you stop the Solver before it reports an optimal solution, the GRG method is far more likely than the SQP method to return a feasible solution as its "best point so far."

Interior Point and SLQP Methods

The Knitro Solver engine for Analytic Solver Comprehensive and Analytic Solver Optimization uses a Barrier or Interior Point method, specialized for **non-convex** problems, to solve general nonlinear optimization problems. As with the SOCP Barrier and MOSEK Solvers, this method forms a logarithmic "barrier function" of the constraints, combines this with the objective, and takes a step towards a better point on each major iteration. (The actual process of taking a step and the path followed are more complex, because Knitro assumes that the problem may be non-convex.) The Knitro Solver uses the PSI Interpreter to efficiently compute second derivative information, but it also has options to work with only first derivative information.

The Knitro Solver engine also includes a new, high performance Sequential Linear-Quadratic (SLQP) method, which is an "active set" method similar to the SQP method. On highly constrained problems, notably those with equality constraints, this method typically outperforms the Interior Point method. On loosely constrained or unconstrained problems, the Interior Point method can greatly outperform SQP and GRG methods, solving problems much larger than either of these methods. Benchmark studies in the academic literature have demonstrated exceptionally good performance for the Knitro Solver, on a wide range of test problems.

The GRG, SQP and Interior Point methods are all subject to the intrinsic limitations cited above for nonlinear optimization problems: For smooth **convex** nonlinear problems, they will (subject to the limitations of finite precision computer arithmetic) find the globally optimal solution; but for **non-convex** problems, they can only guarantee a locally optimal solution. To have a reasonable chance – let alone a guarantee – that you'll find the globally optimal solution to a non-convex problem, you must use special methods for global optimization.

Global Optimization

Analytic Solver Comprehensive and its subset products include powerful tools to help you find the globally optimal solution for a smooth nonlinear **non-convex** problem. These tools include *multistart methods*, which can be used with the nonlinear GRG Solver, the Large-Scale GRG Solver, the Large-Scale SQP Solver, and the Knitro Solver; the Interval Global Solver offers powerful interval methods for global optimization in a commercial software product; and the Evolutionary Solver, for global solutions of smooth and non-smooth problems.

The Multistart Method

The basic idea of the multistart method is to automatically run a nonlinear Solver from different starting points, reaching different locally optimal solutions, then select the best of these as the proposed globally optimal solution. Both *clustering* and *topographic search* multistart methods are included in Analytic Solver and its subset products.

The multistart method operates by generating candidate starting points for the nonlinear Solver (with randomly selected values between the bounds you specify for the variables). These points are then grouped into "clusters" – through a method called *multi-level single linkage* – that are likely to lead to the same locally optimal solution, if used as starting points for the Solver. The nonlinear Solver is then run repeatedly, once from (a representative starting point in) each cluster. The process continues with successively smaller clusters that are increasingly likely to capture each possible locally optimal solution. A Bayesian test is used to determine whether the process should continue or stop.

For many smooth nonlinear problems, the multistart method has a limited guarantee that it will "converge in probability" to a globally optimal solution. This means that as the number of runs of the nonlinear Solver increases, the probability that the globally optimal solution has been found also increases towards 100%. (To attain convergence for constrained problems, an exact penalty function is used in the process of "clustering" the starting points.) For most nonlinear problems, this method will at least yield very good solutions. As discussed below, the multistart method, like the Evolutionary Solver, is a *nondeterministic* method, which by default may yield different solutions on different runs. (To obtain the *same* solution on each run, you can set a Random Seed option for either of these solution algorithms, as discussed in the chapter "Solver Engine Option Reference" in the Frontline Solvers Reference Guide.)

As discussed below, the Evolutionary Solver has been enhanced with "filtered local search" methods that offer many of the benefits of multistart methods – making the Evolutionary Solver even more effective for global optimization problems.

The multistart method can be used on smooth nonlinear problems that also contain integer variables and/or "alldifferent" constraints. But this can take a great deal of solution time, since the multistart method is used for each subproblem generated by the Branch & Bound method for integer problems, and it can also impact the Solver's ability to find feasible integer solutions. If you have many integer variables, or alldifferent constraints, try the Evolutionary Solver as an alternative to the multistart method.

The Interval Branch & Bound Method

In contrast to the multistart methods and the Evolutionary Solver's methods, which are *nondeterministic* methods for global optimization that offer no firm guarantees of finding the globally optimal solution, the Interval Global Solver in uses a *deterministic* method: An *Interval Branch & Bound* algorithm that will find the globally optimal solution – given enough time, and subject to some limitations related to roundoff error, as discussed in the Frontline Solvers Reference Guide.

The Interval Branch & Bound algorithm processes a list of "boxes" that consist of bounded intervals for each decision variable, starting with a single box determined by the bounds that you specify. On each iteration, it seeks lower and upper bounds for the objective and the constraints in a given box that will allow it to discard all or a portion of the box (narrowing the intervals for some of the variables), by proving that the box can contain no feasible solutions, or that it can contain no objective function values better than a known best bound on the globally optimal objective. Boxes that cannot be discarded are subdivided into smaller boxes, and the process is repeated. Eventually, the boxes that remain each enclose a locally optimal solution, and the best of these is chosen as the globally optimal solution.

Several methods are used to obtain good bounds on the values of the objective and constraints within a box or region. *Classic interval methods* rely on the ability of the PSI Interpreter to evaluate Excel functions over intervals and interval gradients. Local constraint propagation methods (also known as *hull consistency* methods) are used to narrow intervals at each stage of evaluation of the problem functions. *Second-order methods* rely on the Interpreter's ability to compute interval Hessians of Excel functions, and use a variant of the *Interval Newton* method to rapidly minimize function values within a region. Innovative *linear enclosure* methods – implemented for the first time in the Interval Global Solver – bound each problem function with a linear approximation that can be used in a Simplex method-based test for feasibility and local optimality.

The Interval Global Solver also has a unique ability to find *all real solutions* for a system of nonlinear equations – which can be listed in the Solutions Report. It can also find an "inner solution" for a system of nonlinear inequalities – a region or "box" (bounds on the variables) within which *all* points satisfy the inequalities. These capabilities are summarized in the chapter "Getting Results: Optimization."

Non-Smooth Optimization

The most difficult type of optimization problem to solve is a non-smooth problem (NSP). Such a problem may not only have multiple feasible regions and multiple locally optimal points within each region – because some of the functions are non-smooth or even discontinuous, derivative or gradient information generally cannot be used to determine the direction in which the function is increasing (or decreasing). In other words, the situation at one possible solution gives very little information about where to look for a better solution.

In all but the simplest problems, it is impractical to exhaustively enumerate all of the possible solutions and pick the best one, even on a fast computer. Hence, most methods rely on some sort of controlled random search, or sampling of possible solutions – combined with deterministic (non-random) methods for exploring the search space. The Evolutionary Solver, based on genetic algorithms, relies fairly heavily on controlled random search, whereas the OptQuest Solver Engine, based on tabu search and scatter search, relies more heavily on deterministic search methods.

A drawback of these methods is that a solution is "better" only in comparison to other, presently known solutions; both the Evolutionary and OptQuest Solvers normally *have no way to test whether a solution is optimal*. This also means that these methods must use heuristic rules to decide *when to stop*, or else stop after a length of time, or number of iterations or candidate solutions, that you specify.

Genetic and Evolutionary Algorithms

A non-smooth optimization problem can be attacked – though not often solved to optimality – using a genetic or evolutionary algorithm. (In a genetic algorithm the problem is encoded in a series of bit strings that are manipulated by the algorithm; in an "evolutionary algorithm," the decision variables and problem functions are used directly. Most commercial Solver products are based on evolutionary algorithms.)

An evolutionary algorithm for optimization is different from "classical" optimization methods in several ways. First, it relies in part on random sampling. This makes it a *nondeterministic* method, which may yield different solutions on different runs. (To obtain the *same* solution on each run, you can set a Random Seed option for the Evolutionary Solver.)

Second, where most classical optimization methods maintain a single best solution found so far, an evolutionary algorithm maintains a *population* of candidate solutions. Only one (or a few, with equivalent objectives) of these is "best," but the other members of the population are "sample points" in other regions of the search space, where a better solution may later be found. The use of a population of solutions helps the evolutionary algorithm avoid becoming "trapped" at a local optimum, when an even better optimum may be found outside the vicinity of the current solution.

Third – inspired by the role of mutation of an organism's DNA in natural evolution – an evolutionary algorithm periodically makes random changes or *mutations* in one or more members of the current population, yielding a new candidate solution (which may be better or worse than existing population members). There are many possible ways to perform a "mutation," and the Evolutionary Solver actually employs five different mutation strategies. The result of a mutation may be an infeasible solution, and the Evolutionary Solver attempts to "repair" such a solution to make it feasible; this is sometimes, but not always, successful.

Fourth – inspired by the role of sexual reproduction in the evolution of living things – an evolutionary algorithm attempts to combine elements of existing solutions in order to create a new solution, with some of the features of each "parent." The elements (e.g. decision variable values) of existing solutions are combined in a *crossover* operation, inspired by the crossover of DNA strands that occurs in reproduction of biological organisms. As with mutation, there are many possible ways to perform a "crossover" operation – some much better than others – and the Evolutionary Solver actually employs multiple variations of four different crossover strategies.

Fifth – inspired by the role of natural selection in evolution – an evolutionary algorithm performs a *selection* process in which the "most fit" members of the population survive, and the "least fit" members are eliminated. In a constrained optimization problem, the notion of "fitness" depends partly on whether a solution is feasible (i.e. whether it satisfies all of the constraints), and partly on its objective function value. The selection process is the step that guides the evolutionary algorithm towards ever-better solutions.

Hybrid Evolutionary and Other Algorithms

You might imagine that better results could be obtained by combining the strategies used by an evolutionary algorithm with the "classical" optimization methods used by the nonlinear GRG and LP/Quadratic Solver. Frontline Systems has done just that.

The Evolutionary Solver operates as described above, but it also employs classical methods in two situations: First, when the evolutionary algorithm generates a new best point, a local search is conducted to try to improve that point. This step can use a "random local search" method, a gradient-free, deterministic direct search method, a gradient-based quasi-Newton method, or a "linearized local gradient" method. Second, when the evolutionary algorithm generates an infeasible point, the Solver can use "repair methods", a quasi-Newton method, or even a specialized Simplex method (for subsets of the constraints that are linear) to transform the infeasible point into a feasible one.

The Evolutionary Solver takes advantage of the diagnostic information available from the PSI Interpreter: It automatically applies genetic algorithm methods to non-smooth variable occurrences (where classical methods cannot be used) and classical methods to smooth and linear variable occurrences. In the local search phase, it can either fix non-smooth variables, or allow them to vary. And it can automatically select the most appropriate local search method, based on linearity and smoothness of the problem functions.

The Evolutionary Solver uses a "distance filter" and a "merit filter" to determine whether to carry out a local search when the genetic algorithm methods find an improved starting point. The "distance filter" plays a role similar to "clustering" in the multistart methods described earlier; both filters contribute to the excellent performance of the Evolutionary Solver on global optimization problems.

The "Achilles' heel" of most evolutionary algorithms is their handling of constraints – they are typically unable to handle more than a few inequalities, or any equality constraints at all. In contrast, the hybrid Evolutionary Solver has been able to find good solutions to non-smooth problems with many – even hundreds – of constraints.

Tabu Search and Scatter Search

The OptQuest Solver Engine for Analytic Solver Comprehensive and Analytic Solver Optimization is based on the principles of tabu search and scatter search. These methods have strong analogies with – and actually predate – genetic algorithm methods, but they rely less heavily on random choice. They work with a population of solutions, which are modified and combined in different ways, then subjected to a selection process. Scatter search methods can sample the space of possible solutions, avoid becoming "trapped" in regions close to local optima, and adaptively diversify or intensify the search. Tabu search uses memory of past search steps to avoid repeated steps and improve future searches. Use of the OptQuest Solver is described in more depth in the **Frontline Solvers Engine User Guide**.

Integer Programming

When a Solver model includes integer constraints (for example A1:A10 = integer, A1:A10 = binary, A1:A10 = semicontinuous, or A1:A10 = alldifferent), it is called an *integer programming* problem. Integer constraints make a model **non-convex**, and finding the optimal solution to an integer programming problem is equivalent to solving a global optimization problem. Such problems may require *far* more computing time than the same problem without the integer constraints.

The standard Microsoft Excel Solver uses a basic Branch & Bound method, in conjunction with the linear LP/Quadratic or nonlinear GRG Solver, to find optimal solutions to problems involving general integer or binary integer variables. Analytic Solver Upgrade uses a much more sophisticated Branch & Bound method that is extended to handle alldifferent constraints, and that often greatly speeds up the solution process for problems with integer variables. In Analytic Solver Comprehensive and Analytic Solver Optimization, the LP/Quadratic Solver uses improved pseudocost-based branch and variable selection, reduced cost fixing, primal heuristics, cut generation, Dual Simplex and preprocessing and probing methods to greatly speed up the solution of *integer linear* programming problems.

The Evolutionary Solver handles integer constraints, in the same form as the other Solver engines (including alldifferent constraints), but it does not make use of the Branch & Bound method; instead, it generates many trial points and uses "constraint repair" methods to satisfy the integer constraints. (The constraint repair methods include classical methods, genetic algorithm methods, and integer heuristics from the local search literature.) The Evolutionary Solver can often find good solutions to problems with integer constraints, but where the Branch & Bound algorithm can *guarantee* that a solution is optimal or is within a given percentage of the optimal solution, the Evolutionary Solver cannot offer such guarantees.

The Branch & Bound Method

The Branch & Bound method begins by finding the optimal solution to the "relaxation" of the integer problem, ignoring the integer constraints. If it happens that in this solution, the decision variables with integer constraints already have integer values, then no further work is required. If one or more integer variables have non-integral solutions, the Branch & Bound method chooses one such variable and "branches," creating two new subproblems where the value of that variable is more tightly constrained. For example, if integer variable A1 has the value 3.45 at the solution, then one subproblem will have the additional constraint A1 <= 3 and the other subproblem will add the constraint A1 >= 4. These subproblems are solved and the process is repeated, "branching" as needed on each of the integer decision variables, until a solution is found where all of the integer variables have integer values (to within a small tolerance).

Hence, the Branch & Bound method may solve many subproblems, *each one* a "regular" Solver problem. The number of subproblems may grow *exponentially*. The "bounding" part of the Branch & Bound method is designed to eliminate sets of subproblems that do not need to be explored because the resulting solutions cannot be better than the solutions already obtained.

Cut Generation

The LP/Quadratic Solver, the Large-Scale LP/QP Solver, Large-Scale SQP Solver, MOSEK Solver, Gurobi Solver, and XPRESS Solver all make use of "cut generation" methods to improve performance on integer linear programming problems. Cut generation derives from so-called "cutting plane" methods that were among the earliest methods applied to integer programming problems, but they combine the advantages of these methods with the Branch & Bound method to yield a highly effective approach, often referred to as a "Branch & Cut" algorithm.

A *cut* is an automatically generated linear constraint for the problem, in addition to the constraints that you specify. This constraint is constructed so that it "cuts off" some portion of the feasible region of an LP subproblem, without eliminating any possible integer solutions. Many cuts may be added to a given LP subproblem, and there are many different methods for generating cuts. For example, *Gomory cuts* are generated by examining the reduced costs at an LP solution, while *knapsack cuts*, also known as *lifted cover inequalities*, are generated from constraints involving subsets of the 0-1 integer variables. Cuts add to the work that the LP solver must perform on each subproblem (and hence they do not always improve solution time), but on many problems, cut generation enables the overall Branch & Cut algorithm to more quickly discover integer solutions, and eliminate branches that cannot lead to better solutions than the best one already known.

The Alldifferent Constraint

In Analytic Solver Comprehensive and its subset products, a constraint such as A1:A5 = alldifferent specifies that the variables A1:A5 must be integers in the range 1 to 5, with each variable different from all the others at the solution. Hence, A1:A5 will contain a *permutation* of the integers from 1 to 5, such as 1,2,3,4,5 or 1,3,5,2,4.

To solve problems involving alldifferent constraints, Analytic Solver employs an extended Branch & Bound method that handles these constraints as a native type. Whenever variables in an "alldifferent group" have non-integral solution values, or integral values that are not all different, the Branch & Bound method chooses one such variable and "branches," creating two new subproblems where the value of that variable is more tightly constrained.

The nonlinear GRG Solver, Large-Scale GRG Solver, Large-Scale SQP Solver, and Knitro Solver engines use this extended Branch & Bound method to solve problems with integer and alldifferent constraints.

The Large-Scale LP/QP Solver, MOSEK Solver, Gurobi Solver, and XPRESS Solver use their own Branch & Cut methods. They transform alldifferent constraints into equivalent sets of binary integer variables and additional linear constraints, then apply their preprocessing, probing and cut generation methods to these variables and constraints.

The Evolutionary Solver uses methods from the genetic algorithm literature to handle alldifferent constraints as permutations, including several mutation operators that preserve the "alldifferent property," and several crossover operators that generate a "child" permutation from "parents" that are also permutations.

Since Solver engines use quite different methods to handle the alldifferent constraint, you'll want to try a variety of Solver engines to see which one performs best on your model. This is especially true if your model uses smooth nonlinear or – even better – linear functions aside from the alldifferent constraint.

Looking Ahead to Models with Uncertainty

If you've read through this chapter, **congratulations** – you've learned a *great deal* about the nature of optimization problems, and how they are solved! But as we said in "Elements of Solver Models" at this chapter's beginning, we've assumed that all parameters of the model are *certain*. The chapter "Mastering Stochastic Optimization Concepts" will cover situations where the parameter values are *uncertain* – and the problems are called *stochastic optimization* problems. In these problems, it's ***more important than ever*** to know how the objective and constraints depend on the decision variables.

We'll see that **stochastic linear programming** problems can be transformed and solved to optimality far more easily and quickly than **stochastic nonlinear, non-convex** problems. We'll also see that certain ways of summarizing the **uncertainty** involved in the model, in chance constraints and expected-value and risk-measure objectives, allow us to form a **convex** optimization model, that can be solved quickly and 'scaled up' to large size. We'll see that even one **decision-dependent uncertainty** usually makes a model non-convex and non-smooth, and far more difficult to solve – but we can still find *good* solutions for such a model, using high-speed simulation optimization in Analytic Solver.

We explained in this chapter that it's usually a mistake to apply a *general-purpose* Solver to a problem that has a simpler structure – for example, applying the GRG Solver or Evolutionary Solver to a linear programming problem: Solution times are much longer than necessary, solutions are less reliable, and the problem typically cannot be scaled up to large size and still solved by these methods.

In a similar way, we'll see that it's usually a mistake to apply a general-purpose *stochastic optimization method* to a problem that has a simpler structure – for example, applying **simulation optimization** (which requires a general-purpose Solver) to a stochastic linear programming problem. Again solution times will be *much* longer than necessary, solutions are *much* less reliable, and the problem typically cannot be scaled up to large size and still solved by these methods.

Mastering Simulation and Risk Analysis Concepts

Introduction

To build a Monte Carlo simulation model in Excel, you begin with a conventional spreadsheet model, designed for 'what-if' analysis. Next, you identify the *inputs* to your model that are uncertain, and use PSI Distribution functions to describe the uncertainty. Then, you identify the *outputs* of special interest (such as Net Profit), and use PSI Statistics functions to examine or summarize how they behave in light of the uncertainty.

This User Guide assumes you have some experience building conventional spreadsheet models, and you almost certainly have experience using such models to ask 'what if' questions, by manually changing input values on the spreadsheet. **The key step in risk analysis is to think in terms, not of just one 'what if' scenario at a time, but of a *range of scenarios, considered at once*.** Think of your spreadsheet cells as representing not just single numbers, that you change once in a while to explore alternatives, but *arrays* of numbers that cover the range of possibilities.

Although you can usually visualize the range of values for one *input* value at a time, it is very difficult to foresee – without computer assistance – the range of outcomes for an *output* value that depends on *several* interacting inputs, each one subject to uncertainty. But Analytic Solver Comprehensive, Analytic Solver Simulation and Analytic Solver Upgrade (in both the Desktop and Cloud apps) *automatically* compute the full range of outcomes for every cell in your spreadsheet model, and enables you to quickly see statistics such as the mean, standard deviation, or 10th and 90th percentiles of the range of outcomes.

What Happens During Monte Carlo Simulation

You'll find it easier to understand simulation results if you have a good grasp of the Monte Carlo simulation process. At its heart, this process is very simple. It consists of the following steps:

1. Generate a **random sample** for the **uncertain variables** in your model. If you specify (say) 1,000 Monte Carlo trials per simulation, then 1,000 randomly chosen values will be generated for *each* uncertain variable.
2. For each of 1,000 Monte Carlo trials, **recalculate** your model, with the right sample values in each uncertain variable cell. When Excel is used to recalculate, the PSI Distribution function in each uncertain variable cell returns the correct sample value for that trial. When the PSI Interpreter is used (as is always the case in Analytic Solver Cloud), this is done internally (much faster) by Analytic Solver.
3. On each Monte Carlo trial, monitor and save the calculated value of each **uncertain function** in your model. (Recall that any formula cell containing a call to PsiOutput(), or referenced in the first argument of PsiOutput() or a

PSI Statistics function, is monitored as an uncertain function cell.) For 1,000 trials, there will be 1,000 saved values for *each* uncertain function.

When the simulation process is complete, Analytic Solver Simulation uses the 1,000 saved values of each uncertain function to calculate statistics and percentiles, draw frequency distributions, scatter plots and other charts, and compute values for each PSI Statistic function call in your model.

Random Number Generation and Sampling

On each Monte Carlo trial, *sample values* are drawn from the probability distributions represented by the PSI Distribution functions in your model. Sample values are computed by first *drawing* a "random number" between 0 and 1, then *transforming* this uniform random sample value into a sample value that:

- Constrains the samples drawn to obtain better coverage of the sample space, where each PSI Distribution function is a 'dimension' of that space
- Ensures that the frequency distribution of samples drawn properly reflects the shape and parameters of the PSI Distribution function
- Ensures that the samples drawn for multiple PSI Distribution functions properly reflect the correlation of distributions with each other

Random Number Generator

Analytic Solver Simulation includes an advanced set of random number generation capabilities. In common applications, any good random number generator is sufficient – but for challenging applications (for example in financial engineering) that involve many uncertain variables and many thousands of trials, the advanced features of Analytic Solver Simulation can make a real difference.

Computer-generated numbers are never truly "random," since they are always computed by an algorithm – they are called *pseudorandom* numbers. A random number generator is designed to quickly generate sequences of numbers that are as close to statistically independent as possible. Eventually, an algorithm will generate the same number seen sometime earlier in the sequence, and at this point the sequence will begin to repeat. The *period* of the random number generator is the number of values it can generate before repeating.

A long period is desirable, but there is a tradeoff between the length of the period and the degree of statistical independence achieved within the period. Hence Analytic Solver Simulation offers a choice of four random number generators:

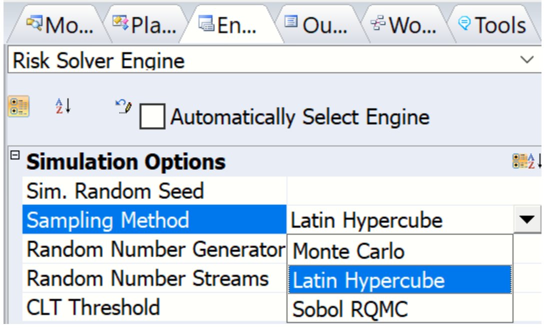
- **Park-Miller** "Minimal" Generator with Bayes-Durham shuffle and safeguards. This generator has a period of 2^{31}-2. Its properties are good, but the following choices are usually better.
- Combined Multiple Recursive Generator (**CMRG**) of L'Ecuyer. This generator has a period of 2^{191}, and excellent statistical independence of samples within the period.
- Well Equidistributed Long-period Linear (**WELL1024**) generator of Panneton, L'Ecuyer and Matsumoto. This very new generator combines a long period of 2^{1024} with very good statistical independence.

- **Mersenne Twister** generator of Matsumoto and Nishimura. This generator has the longest period of 2^{19937}-1, but the samples are not as “equidistributed” as for the WELL1024 and CMRG generators.

Random Number Seeds

As explained in the chapter “Getting Results: Simulation,” the **seed**, or initial value, of the random number generator determines whether your results are *exactly reproducible* when you re-run a simulation, or whether your results are *similar but not identical* because a different random sample was drawn.

You can set a seed for the entire simulation run using the **Sim. Random Seed** option – the first option on the Task Pane Engine tab when you select **Risk Solver Engine** (the simulation engine) from the dropdown list. Any positive integer sets a specific seed; 0 means the seed will be *different* on every run.



You can also set a seed for any specific uncertain variable, using the Seed option in the Uncertain Variable dialog, or by supplying a **PsiSeed()** property function as an argument to the PSI Distribution function call. This means the uncertain variable will have its own independent stream of random numbers starting from the given seed, whether or not you’ve set a seed for the whole simulation run.

Sampling Method

In standard Monte Carlo sampling, numbers generated by the chosen random number generator are used directly to obtain sample values for the uncertain variables (PSI Distribution functions) in the model. With this method, the variance or *estimation error* in computed samples for uncertain functions is inversely proportional to the square root of the number of trials; hence to cut the error in half, four times as many trials are required.

Analytic Solver Simulation provides two other sampling methods than can significantly improve the ‘coverage’ of the sample space, and thus reduce the variance in computed samples for output functions. This means that you can achieve a given level of accuracy (low variance or error) with fewer trials. You choose this via the **Sampling Method** option on the **Engine tab**, shown above.

Latin Hypercube Sampling. Latin Hypercube sampling begins with a stratified sample in each dimension (one for each uncertain variable), which constrains the random numbers drawn to lie in a set of subintervals from 0 to 1. Then these one-dimensional samples are combined and randomly permuted so that they ‘cover’ a unit hypercube in a stratified manner. This often reduces the variance of uncertain functions.

Sobol numbers (Randomized QMC). Sobol numbers are an example of so-called “Quasi Monte Carlo” or “low-discrepancy numbers,” which are generated with a goal of coverage of the sample space rather than “randomness” and statistical independence. Analytic Solver adds a “random shift” to Sobol numbers, which improves their statistical independence. Sobol numbers are frequently used in quantitative finance applications, where they are often effective at reducing variance.

Random Number Streams

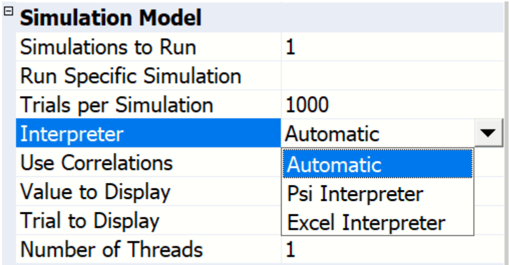
Most Monte Carlo simulation tools generate a *single* sequence of random numbers, taking values consecutively from this sequence to obtain samples for each of the distributions in a model. This introduces a subtle dependence between the samples for all distributions in one trial. Analytic Solver Comprehensive and Analytic Solver Simulation allow you to specify that an independent random number sequence (stream) should be used for each distribution in the model – using the **Random Number Streams** option on the **Engine tab**, as shown on the previous page. This capability works for Monte Carlo sampling and Latin Hypercube sampling; it does not apply to Sobol numbers.

In many applications, the difference between a single stream and multiple strams is very small – but in some cases, found in financial engineering and other demanding applications, better results are obtained if independent random number sequences (streams) are used.

The PSI Interpreter and Simulation

Analytic Solver Desktop can use either Excel, or its own *Polymorphic Spreadsheet Interpreter* (PSI Technology) to perform Monte Carlo simulation trials at high speed – often 100 times faster or more than performing the trials by allowing Microsoft Excel to recalculate the spreadsheet. (Analytic Solver Cloud *always* uses PSI Technology.) Normally, you’ll want to use the PSI Interpreter for simulation trials, since it is designed to compute the same values as Excel does, but much faster than the Excel Interpreter.

However, there are a few features of Excel formulas and functions that the PSI Interpreter does not handle; if you use these features in your model, you’ll see an error message when you try to run a simulation. If your model requires the use of Excel features that are not supported by the PSI Interpreter, you may have to use the Excel Interpreter instead. To do this, change the Task Pane Platform tab Simulation Model group **Interpreter** option, as shown below.



Simulation Model	
Simulations to Run	1
Run Specific Simulation	
Trials per Simulation	1000
Interpreter	Automatic
Use Correlations	
Value to Display	
Trial to Display	
Number of Threads	1

Note: Functions not supported by the PSI Interpreter are: Call(), Cell(), CubeX(), EuroConvert(), GetPivotData(), HyperLink(), Info(), RegisterID(), and SqlRequest().

More About the PSI Interpreter

So, how does the PSI Interpreter work, and how can it be so much faster than Excel alone for simulation? To answer this, we must look at how Excel itself computes values for your formulas.

Microsoft Excel is an *interpreter* for spreadsheet formulas. When you enter a formula such as =A1+A2*(A3-4) in cell B1, Excel first scans and *parses* the formula, storing the results in a coded internal form. It also maintains storage for A1, A2, A3 and thousands of other cells. When you change a number and recalculate, Excel refers to the coded internal form, looks up the values of A1, A2 and A3 and fetches the constant 4, performs the arithmetic, and saves the result in storage reserved for B1. *Only a small part of the time is spent on the actual arithmetic* – most of the time goes into processing the encoded formula, and looking up and saving cell values.

PSI stands for ***Polymorphic Spreadsheet Interpreter***. PSI is also an *interpreter* for spreadsheet formulas, that accepts the same formula syntax and built-in functions as Microsoft Excel. But where Excel evaluates formulas only for one datatype – single numbers, PSI can evaluate formulas for many different datatypes. (The word "polymorphic" comes from object-oriented programming, where it has essentially the same meaning.)

One of the special PSI datatypes is Monte Carlo trials, where each number is replaced by a vector of 1,000 or more trial values. The heart of PSI speed comes from the fact that it processes the encoded formula *once*, fetches 1,000 data values *at once*, and performs the arithmetic for all 1,000 values *at once*. Hence, the "overhead" of the interpreter is incurred *once* rather than 1,000 times; *most of the time is spent doing the actual arithmetic*. PSI is so fast because it is *specialized* for simulation (and optimization).

Uncertain Functions, Statistics, and Risk Measures

Once a Monte Carlo simulation is complete, what statistics should you use to evaluate the outputs of special interest in your model? The answer depends on your application, but some general guidelines can be given.

Descriptive statistics are usually classified into measures of central tendency, and measures of variation or dispersion. You should look at both kinds of measures, and at some kind of *quantile* measure, to assess almost any output in your model.

Measures of Central Tendency

Analytic Solver provides several measures of central tendency:

- **PsiMean**, the average of all the values
- **PsiPercentile** for the median or 50th percentile
- **PsiMode**, the most frequently occurring single value

Measures of Variation

Analytic Solver provides several standard measures of variation:

- **PsiVariance**, which describes the *spread* of the distribution of values
- **PsiStdDev** for standard deviation, the square root of variance
- **PsiSkewness**, which describes the *asymmetry* of the distribution of values
- **PsiKurtosis**, which describes the *peakedness* of the distribution of values

- **PsiMin**, **PsiMax**, and **PsiRange** for the minimum and maximum values, and the difference between them

Risk Measures

Analytic Solver also provides several risk measure functions that are most often used in quantitative finance applications, but may be used in any model.

- **PsiAbsDev** for 'MAD', which measures *absolute* deviations from the mean
- **PsiSemiVar** for semivariance or lower partial moment, which measures and weights *negative* deviations from the mean
- **PsiSemiDev** for semideviation, the square root of semivariance (*q*th root for the lower partial moment)

PsiSemiVar and PsiSemiDev are useful in situations where 'upward' variation – for example, higher stock prices or increased profits – is desirable, but 'downward' variation – lower prices or losses – is undesirable.

Quantile Measures

To get a complete grasp of the range of outcomes, it's essential to look at *quantile* measures, such as percentiles and Value at Risk, in addition to measures of central tendency and variation. Quantile measures allow you to answer questions such as 'How much money might we lose, with 5% or 10% probability?" or "What is the probability that we'll make at least $100,000?" based on your simulation model. Analytic Solver provides:

- **PsiPercentile**, which provides percentile values from 1% to 99%
- **PsiTarget**, which returns the proportion of values less than or equal to a target value
- **PsiBVaR**, which measures standard ('Basel') Value at Risk
- **PsiCVaR**, which measures Conditional Value at Risk

Confidence Intervals

Every Monte Carlo simulation uses a *sample* of the possible values of your uncertain variables; hence any statistic resulting from the simulation involves some degree of sampling error. For the mean and standard deviation of an output value, Analytic Solver Simulation provides functions that help you assess this error, and estimate the range or interval in which you can be *confident* that the true mean or standard deviation lies, at a confidence level that you specify:

- **PsiMeanCI**, which returns a confidence interval for the mean
- **PsiStdDevCI**, which returns a confidence interval for the standard deviation
- **PsiCITrials**, which returns the number of simulation trials needed to obtain a confidence interval of a given size, at a given confidence level

Uncertain Variables and Probability Distributions

For experienced spreadsheet modelers, the most challenging task in creating a Monte Carlo simulation model is usually not identifying the key inputs and outputs, but selecting an appropriate probability distribution and parameters to model the uncertainty of each input variable.

Analytic Solver provides over 40 analytic probability distributions – which one should you use? Again the answer depends on your application, but some general guidelines can be given.

If a **Certified Distribution** (see below under "Probability Management Concepts") is available for an uncertain variable – for example, your company's estimate of the range and probabilities for prices of certain chemical feedstocks, or a service provider's estimate of the range and probabilities for stock or bond prices – you can simply use the Certified Distribution.

Discrete Vs. Continuous Distributions

If you must choose or create your own distribution, the first step is to determine whether to use a **discrete** or **continuous** form. If there are a small number of possible values for the uncertain variable, you may be able to use a discrete analytic distribution, or construct a discrete custom distribution. If the underlying physical process involves discrete, countable entities – such as the number of customers arriving at a service window – you can use a discrete distribution. If the possible values are highly divisible – such as most prices, volumes, interest rates, exchange rates, weights, distances, etc. – you will likely use a continuous distribution. In some cases, you may use a continuous distribution to *approximate* a discrete distribution.

Metalog Distributions

Since mid-2017, Analytic Solver has also supported the increasingly popular Metalog family of distributions, created by Dr. Tom Keelin, and recently popularized by the nonprofit Probability Management group. Metalog distributions can closely approximate virtually any classical continuous distribution, and often they can better fit user data than classical distributions. Analytic Solver includes a powerful newfacility to automatically fit user data to the full range of possible (bounded and unbounded, multi-term) Metalog distributions. See **Using the Fit Feature** below to learn how to fit a series of historical data to a Metalog distribution.

Bounded Vs. Unbounded Distributions

Another characteristic that distinguishes input distributions is the range of sample values they can generate. Some distributions are intrinsically **bounded** – samples are guaranteed to lie between a known minimum and maximum value. Examples are the Uniform, Triangular, Beta, and Binomial distributions. Other analytic distributions are **unbounded** – sample values may cluster around the distribution's mean, but may sometimes have extreme negative or positive values. Examples are the Normal, Logistic, and Extreme Value distributions. Still other distributions are **partially bounded**, with a known minimum such as zero, but no maximum value. Examples are the Exponential, Poisson, and Weibull distributions.

At times, you may find that the most appropriate distribution (say the Normal) is unbounded, but you know that the realistic values of the physical process are bounded, or your model is designed to handle values only up to some realistic limit. You can impose bounds on any distribution using Analytic Solver Simulation's Uncertain Variable dialog. For information about "cutoff bounds" and “censor bounds” (both may be used), consult the Frontline Solvers Reference Guide.

Analytic Vs. Custom Distributions

A third characteristic of input distributions is whether they are analytic (also called parametric) or custom (sometimes called non-parametric) distributions. An **analytic** distribution has a form derived from certain theoretical assumptions about the problem. For example, a Poisson distribution is derived from an assumption that events are *independent* and occur at a known *average rate*, and an Exponential distribution is derived from an assumption of a *constant rate* of decay in some process. A **custom** distribution has a form dictated by either past data or expert opinion about the range and frequency of sample values. Analytic Solver offers five general-purpose functions – PsiCumul, PsiDiscrete, PsiDisUniform, PsiGeneral and PsiHistogram – to help you model custom distributions. Generally speaking, you should choose an analytic distribution if – and *only* if – the theoretical assumptions truly apply in your situation.

Creating your Own Distributions When Past Data is Available

If you have, or you can collect **data on the past performance** of the uncertain variable – and if you believe that ‘past performance’ is likely to be representative of future performance – you have three options:

- If you can **fit the data** (past observations) to a specific type of analytic distribution and its parameters, and if there is reason to believe that the underlying process that the uncertain variable is measuring is *consistent* with the *assumptions* from which the analytic distribution is derived, you can use this distribution (for example PsiNormal, PsiWeibull, etc.) for the variable. See “Using the Fit Feature” section below. *Note: Data used in fitting a distribution should be independent.*

- If you have a reasonably **large number** of observations of past performance of the variable, compared to the number of simulation trials you want to run, you can use the *past data itself* for simulation trials, in the form of a SIP (Stochastic Information Packet) or a DIST (Distribution String), and use the PsiSip() or PsiSlurp() distribution function for the uncertain variable and PsiTSSip for a time series. See “Probability Management Concepts” and, for DISTs, “Stochastic Libraries: SIPs and SLURPs” below.

- If – as is often the case – you have a relatively **small number** of observations of past performance compared to the number of trials you want to run, you may be better off *resampling* the past performance data. To do this, store the past data in a cell range or SIP, and use the PsiDisUniform() function (single values) or the PsiResample() function (multiple values). The difference is that, instead of using all of the past observations (one per simulation trial), you *randomly sample* the past observations on each trial.

You can use the Analytic Solver GUI to find the best-fitting analytic distribution. Both discrete and continuous distributions can be fitted to data; 29 common distributions (out of the 40+ available) are used for fitting. You can also call the Distribution object Fit and AutoFit methods in your VBA code, as described in the chapter “VBA Object Model Reference” within the Frontline Solvers Reference Guide.

When Past Data is Not Available

If you *don't* have, and you cannot easily collect data on the past performance of the uncertain variable – or if past performance is not likely to be representative of future performance – you must tackle the problem in a different way:

- Consult the **literature for your industry**, if available, to find examples of applications like yours where simulation models were built. Find out – by contacting the authors if necessary – what kinds of distributions were used for the uncertain variables, and the rationale for choosing them.
- If you cannot find reports on industry-specific applications like yours, consult the **publications of professional societies** like INFORMS, where simulation applications are reported. One rich source is the past proceedings of the Winter Simulation Conference (www.wintersim.org).
- In the chapter **PSI Function Reference**, read the descriptions of the different PSI Distribution functions, which include brief comments on the types of applications where each distribution has been used in the past. See the books listed at the beginning of that chapter for further information.

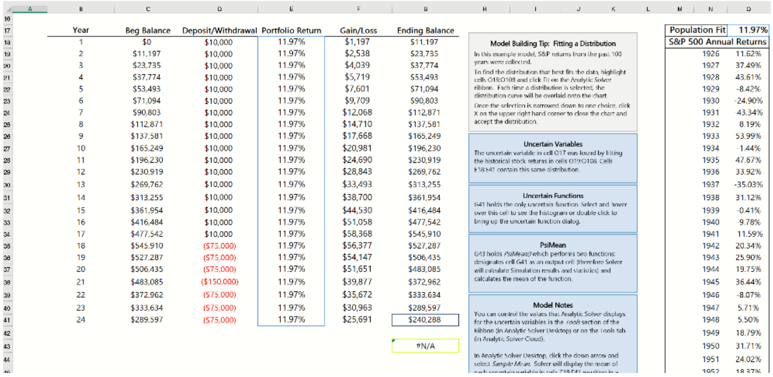
You are well-advised to *keep it simple!* Many physical, social and biological phenomena are well described by the Normal distribution, or – *if* the possible values are *equally likely* to occur, as in a coin flip or single die – the Uniform distribution. Bear in mind that when any set of distributions are summed, the result (quickly) tends towards the Normal distribution.

Applications that involve *queuing* – customers arriving or departing, parts awaiting assembly, etc. – have been well studied, so you can often find appropriate distributions in the literature. Applications that use the Project Evaluation and Review Technique (PERT) can often use the PsiPert() function to model uncertainty.

Using the Fit Feature

To fit a series of data using the Fit tool in Analytic Solver follow the following steps. To open the example model, click **Help – Example Models** on the Analytic Solver ribbon, then click **Monte Carlo Simulation Examples – College Fund Growth 1**.

1. Select the range of data you want to fit. For this example, select cells O19:O112.



Year	Beg Balance	Deposit/Withdrawal	Portfolio Return	Gain/Loss	Ending Balance
1	$0	$10,000	11.97%	$1,197	$11,197
2	$11,197	$10,000	11.97%	$2,538	$23,735
3	$23,735	$10,000	11.97%	$4,039	$37,774
4	$37,774	$10,000	11.97%	$5,719	$53,493
5	$53,493	$10,000	11.97%	$7,601	$71,094
6	$71,094	$10,000	11.97%	$9,709	$90,803
7	$90,803	$10,000	11.97%	$12,068	$112,871
8	$112,871	$10,000	11.97%	$14,710	$137,581
9	$137,581	$10,000	11.97%	$17,668	$165,249
10	$165,249	$10,000	11.97%	$20,981	$196,230
11	$196,230	$10,000	11.97%	$24,690	$230,919
12	$230,919	$10,000	11.97%	$28,843	$269,762
13	$269,762	$10,000	11.97%	$33,493	$313,255
14	$313,255	$10,000	11.97%	$38,700	$361,954
15	$361,954	$10,000	11.97%	$44,530	$416,484
16	$416,484	$10,000	11.97%	$51,058	$477,542
17	$477,542	$10,000	11.97%	$58,368	$545,910
18	$545,910	($75,000)	11.97%	$56,377	$527,287
19	$527,287	($75,000)	11.97%	$54,147	$506,435
20	$506,435	($75,000)	11.97%	$51,651	$483,085
21	$483,085	($150,000)	11.97%	$39,877	$372,962
22	$372,962	($75,000)	11.97%	$35,672	$333,634
23	$333,634	($75,000)	11.97%	$30,963	$289,597
24	$289,597	($75,000)	11.97%	$25,691	$240,288
					#N/A

Model Building Tip: Fitting a Distribution

In this example model, S&P returns from the past 100 years were collected.

To find the distribution that best fits the data, highlight cells O19:O108 and click Fit on the Analytic Solver ribbon. Each time a distribution is selected, the distribution curve will be overlaid onto the chart.

Once the selection is narrowed down to one choice, click X on the upper right hand corner to close the chart and accept the distribution.

Uncertain Variables

The uncertain variable in cell O17 was found by fitting the historical stock returns in cells O19:O108. Cells E18:E41 contain this same distribution.

Uncertain Functions

G41 holds the only uncertain function. Select and hover over this cell to see the histogram or double click to bring up the uncertain function dialog.

PsiMean

G43 holds *PsiMean()* which performs two functions: designates cell G41 as an output cell (therefore Solver will calculate Simulation results and statistics) and calculates the mean of the function.

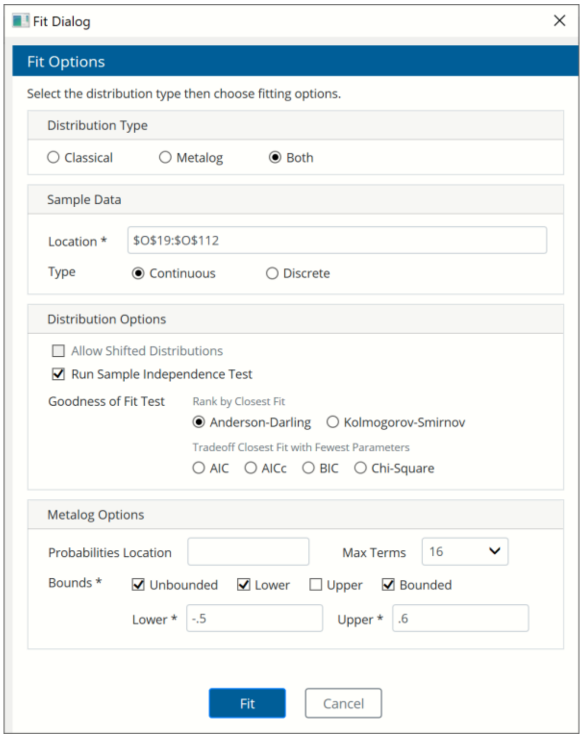
Model Notes

You can control the values that Analytic Solver displays for the uncertain variables in the *Tools* section of the Ribbon (in Analytic Solver Desktop) or on the Tools tab (in Analytic Solver Cloud).

In Analytic Solver Desktop, click the down arrow and select *Sample Mean.* Solver will display the mean of

Population Fit	11.97%
S&P 500 Annual Returns	
1926	11.62%
1927	37.49%
1928	43.61%
1929	-8.42%
1930	-24.90%
1931	-43.34%
1932	8.19%
1933	53.99%
1934	-1.44%
1935	47.67%
1936	33.92%
1937	-35.03%
1938	31.12%
1939	-0.41%
1940	-9.78%
1941	11.59%
1942	20.34%
1943	25.90%
1944	19.75%
1945	36.44%
1946	-8.07%
1947	5.71%
1948	5.50%
1949	18.79%
1950	31.71%
1951	24.02%

2. Select **Tools -- Fit** on the Analytic Solver Ribbon to open the Fit dialog.
3. Make any changes to the default settings on the Fit Options dialog.



Select the distribution type then choose fitting options.

○ Classical ○ Metalog ● Both

Location * $O$19:$O$112

Type ● Continuous ○ Discrete

☐ Allow Shifted Distributions

☑ Run Sample Independence Test

Goodness of Fit Test

Rank by Closest Fit

● Anderson-Darling ○ Kolmogorov-Smirnov

Tradeoff Closest Fit with Fewest Parameters

○ AIC ○ AICc ○ BIC ○ Chi-Square

Probabilities Location

Max Terms 16

Bounds * ☑ Unbounded ☑ Lower ☐ Upper ☑ Bounded

Lower * -.5 Upper * .6

Leave *Both* selected for *Distribution Type* to fit the distribution to both the Classical and Metalog distributions. For more information on Metalog distributions, see the topic, **Fitting a Meta-Log Distribution to Simulation Results**, that appears previously in this guide.

For **Sample Data Type**, select **Continuous** if the uncertain variable's values are highly divisible – such as most prices, volumes, interest rates, exchange rates, weights, and distances – or **Discrete** if the underlying physical process involves discrete, countable entities. Since the data in this example are annual rates of return, keep *Continuous* selected.

The checkbox **Allow Shifted Distributions** allows Analytic Solver to shift the center of analytic distribution (equivalent to using the PsiShift() property function) to better fit the sample data; sometimes this is not desirable, so this option will remain unchecked.

When fitting a sample to a distribution, it is important that the trial values are not in any way correlated with each other. In other words it is important that each trial is independent. If the checkbox **Run Sample Independence Test** is checked, the Psi Interpreter will run an independence test and will report an error if the trial values are found to be dependent. Use the default setting (checked) for this example.

For Continuous data, you can choose to rank the fitted distributions by Anderson-Darling, Kolmogorov-Smirnoff, AIC, AICc, BIC or Chi-Square statistics. For Discrete data, only AIC, AICc, BIC and Chi Square statistics are meaningful (and may be selected). For this example, leave the default Anderson-Darling selected.

- Kolmogorov-Smirnoff –This test computes the difference (D) between the continuous distribution function (CDF) and the empirical cumulative distribution function (ECDF). The null hypothesis is rejected if, at the 90% significance level, D is larger than the critical value statistic.

- Anderson-Darling –Ranks the fitted distribution using the Anderson Darling statistic, A^2. The null hypothesis is rejected using a 90% significance level, if A^2 is larger than the critical value statistic. This test awards more weight to the distribution tails then the Kolmogorov-Smirnoff test.
- AIC/BIC – The AIC test is a Chi Squared test corrected for the number of distribution parameters and sample size.

  AIC = Chi-Square Statistic + 2 * k + 2 * k * (k + 1) / (n – k – 1)

  where k is the number of distribution parameters and n is the sample size.
- AICc –When the sample size is small, there is a significant chance that the AIC test will select a model with a large number of parameters. In other words, AIC will overfit the data. AICc was developed to reduce the possibility of overfitting by applying a penalty to the number of parameters. . Assuming that the model is univariate, is linear in the parameters and has normally-distributed residuals, the formula for AICc is:

  $\text{AICc} = \text{AIC} + \frac{2k^2+2k}{n-k-1}$

  where n = sample size, k = # of parameters.

  As the sample size approaches infinity, the penalty on the number of parameters converges to 0 resulting in AICc converging to AIC.
- BIC – The Bayesian information criterion (BIC) is defined as:

  $\text{BIC} = k \ln(n) = 2 \ln(\hat{L})$

  where

  $\hat{L}$ = the maximized value of the likelikhood function of the model M. $\hat{L} = p(x|\theta, M)$ where θ are the parameter values that maximize the likelihood function and x is the observed data.

  n = Sample size

  k = Number of parameters
- Chi Square – Uses the chi-square statistic and distribution to rank either continuous or discrete distributions. Sample data is first divided into intervals using either equal probability, then the number of points that fall into each interval are compared with the expected number of points in each interval. The null hypotheses is rejected using a 90% significance level, if the the chi-squared test statistic is greater than the critical value statistic.

  Note: The Chi Square test is used indirectly in continuous fitting as a support in the AIC test. The AIC test must succeed in both discrete and continuous fitting as this is a necessary condition. When fitting a discrete function, the Chi Square test must also succeed. When fitting a continuous function, at least one of the tests, Chi Squared, Kolmogorov-Smirnoff, or Anderson-Darling, must succeed as well.

Leave *Probabilities* blank. This is an Excel range containing historical cumulative probabilities. If omitted, each data point is assumed to be equally likely.

Leave *Max Terms* at the default of 16. This is the number of terms in the Metalog function. The minimum number of terms is 2. Metalogs of all terms, up to Max Terms, will be calculated and displayed in the results.

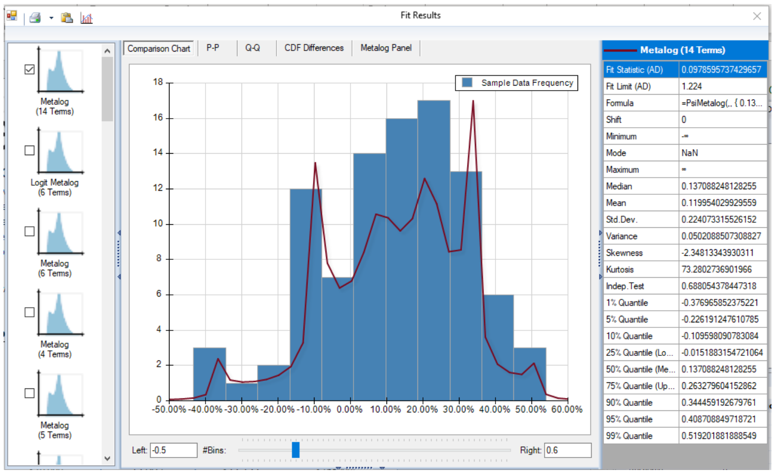
The lower bound must be a value lower than any data point and the upper bound must be a value greater than any data point. Enter **-.50** for *Lower* and **.60** for *Upper*. There are no data points less than -45% and 55%.

Click **Fit** to fit the historical data to a distribution. (Click Cancel to close the dialog without performing the distribution fitting.)

Analytic Solver computes and displays a ranked list of candidate fitted distributions, as shown below. Initially, the distribution with the best fit statistic (selected in the above dialog) is shown in a chart that overlays the sample data. Use the #Bins slider to add of subtract the number of bins in the histogram.

Tip: Once you have accepted the distribution you can change the fit options by clicking the Fit Options icon on the title bar of the Fit Results dialog.

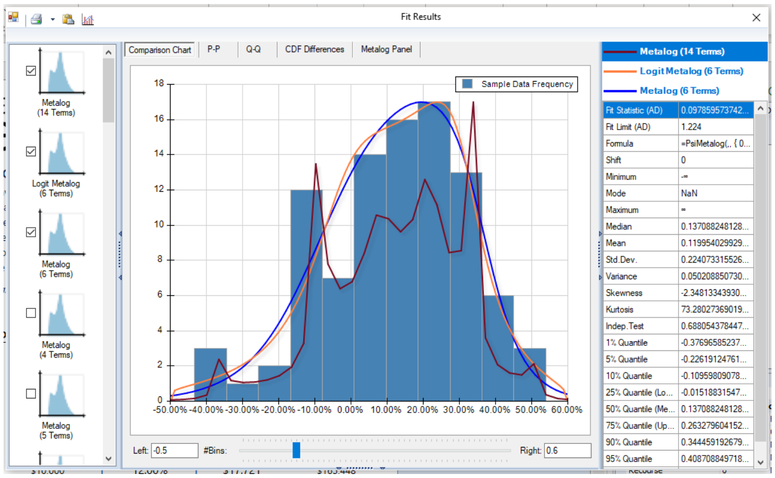
4. Review and accept or modify the results. Relevant distribution statistics are contained in the pane on the right of the dialog.



Left: -0.5 #Bins: Right: 0.6

Metalog (14 Terms)	
Fit Statistic (AD)	0.0978595737429657
Fit Limit (AD)	1.224
Formula	=PsiMetalog(,, { 0.13...
Shift	0
Minimum	-∞
Mode	NaN
Maximum	∞
Median	0.137088248128255
Mean	0.119954029929559
Std.Dev.	0.224073315526152
Variance	0.0502088507308827
Skewness	-2.34813343930311
Kurtosis	73.2802736901966
Indep.Test	0.688054378447318
1% Quantile	-0.376965852375221
5% Quantile	-0.226191247610785
10% Quantile	-0.109598090783084
25% Quantile (Lo...	-0.0151883154721064
50% Quantile (Me...	0.137088248128255
75% Quantile (Up...	0.263279604152862
90% Quantile	0.344459192679761
95% Quantile	0.408708849718721
99% Quantile	0.519201881888549

On the left of the dialog is a list of fit choices shown in order of decreasing Goodness of Fit based on the test you chose. You can select multiple choices to see how they compare and choose the one you are most happy with.



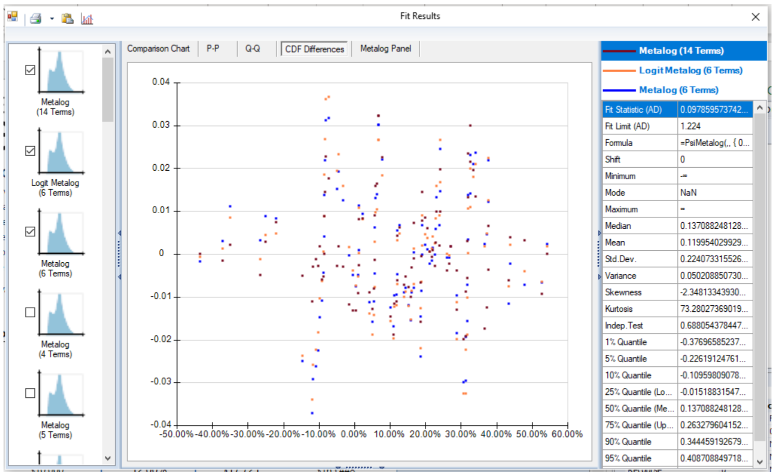
Left: -0.5

Right: 0.6

Fit Statistic (AD)	0.097859573742...
Fit Limit (AD)	1.224
Formula	=PsiMetalog(,, { 0...
Shift	0
Minimum	-∞
Mode	NaN
Maximum	∞
Median	0.137088248128...
Mean	0.119954029929...
Std.Dev.	0.224073315526...
Variance	0.050208850730...
Skewness	-2.34813343930...
Kurtosis	73.28027369019...
Indep.Test	0.688054378447...
1% Quantile	-0.37696585237...
5% Quantile	-0.22619124761...
10% Quantile	-0.10959809078...
25% Quantile (Lo...	-0.01518831547...
50% Quantile (Me...	0.137088248128...
75% Quantile (Up...	0.263279604152...
90% Quantile	0.344459192679...
95% Quantile	0.408708849718...

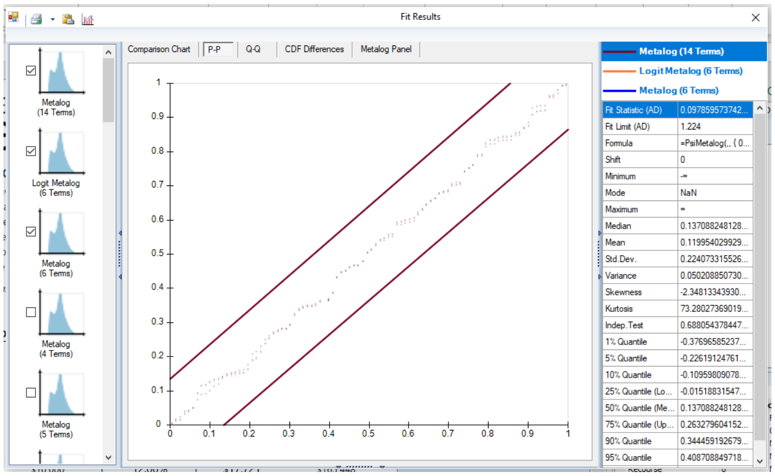
You can also click on the P-P, Q-Q, CDF Differences tabs to get additional perspective on which distribution best fits.

The Cumulative Distribution Function Differences chart quantifies the difference between the empirical distribution function and the cumulative distribution function of the distribution.



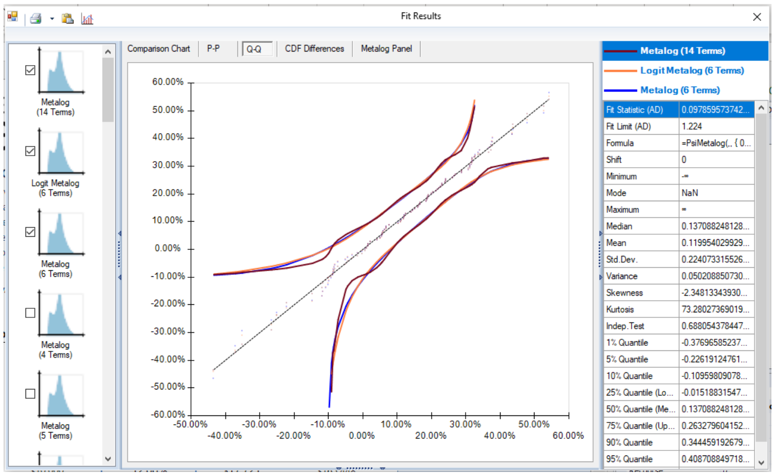
Fit Statistic (AD)	0.097859573742...
Fit Limit (AD)	1.224
Formula	=PsiMetalog(,, { 0...
Shift	0
Minimum	-∞
Mode	NaN
Maximum	∞
Median	0.137088248128...
Mean	0.119954029929...
Std.Dev.	0.224073315526...
Variance	0.050208850730...
Skewness	-2.34813343930...
Kurtosis	73.28027369019...
Indep.Test	0.688054378447...
1% Quantile	-0.37696585237...
5% Quantile	-0.22619124761...
10% Quantile	-0.10959809078...
25% Quantile (Lo...	-0.01518831547...
50% Quantile (Me...	0.137088248128...
75% Quantile (Up...	0.263279604152...
90% Quantile	0.344459192679...
95% Quantile	0.408708849718...

The P-P tab displays the Probability-Probability chart. This graph is used to determine how well a specific distribution fits the observed data. This chart will be approximately linear (or lie on the y = x line) if the specified distribution is the correct choice.



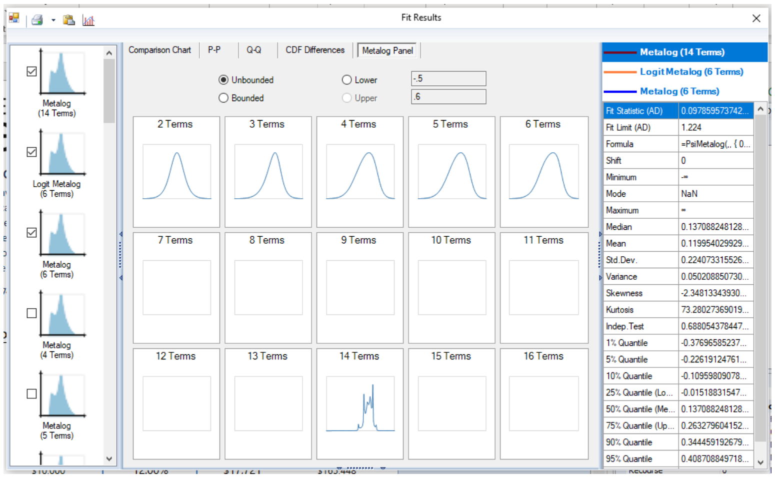
Fit Statistic (AD)	0.097859573742...
Fit Limit (AD)	1.224
Formula	=PsiMetalog(,, { 0...
Shift	0
Minimum	-∞
Mode	NaN
Maximum	∞
Median	0.137088248128...
Mean	0.119954029929...
Std.Dev.	0.224073315526...
Variance	0.050208850730...
Skewness	-2.34813343930...
Kurtosis	73.28027369019...
Indep.Test	0.688054378447...
1% Quantile	-0.37696585237...
5% Quantile	-0.22619124761...
10% Quantile	-0.10959809078...
25% Quantile (Lo...	-0.01518831547...
50% Quantile (Me...	0.137088248128...
75% Quantile (Up...	0.263279604152...
90% Quantile	0.344459192679...
95% Quantile	0.408708849718...

Click the Q-Q tab to see the Quantile – Quantile chart. Again, if the two distributions being compared are a good fit, the plot will lie approximately on the y = x line.



Fit Statistic (AD)	0.097859573742...
Fit Limit (AD)	1.224
Formula	=PsiMetalog(,, { 0...
Shift	0
Minimum	-∞
Mode	NaN
Maximum	∞
Median	0.137088248128...
Mean	0.119954029929...
Std.Dev.	0.224073315526...
Variance	0.050208850730...
Skewness	-2.34813343930...
Kurtosis	73.28027369019...
Indep.Test	0.688054378447...
1% Quantile	-0.37696585237...
5% Quantile	-0.22619124761...
10% Quantile	-0.10959809078...
25% Quantile (Lo...	-0.01518831547...
50% Quantile (Me...	0.137088248128...
75% Quantile (Up...	0.263279604152...
90% Quantile	0.344459192679...
95% Quantile	0.408708849718...

Click the Metalog Panel tab to view all feasible metalog density functions. Distributions with 7-13, and 15-16 terms are infeasible and therefore do not appear in the results. Notice how the distributions evolve as the number of terms are increased. Note that the larger the number of terms, the greater the possibility that the distribution will overfit the data.

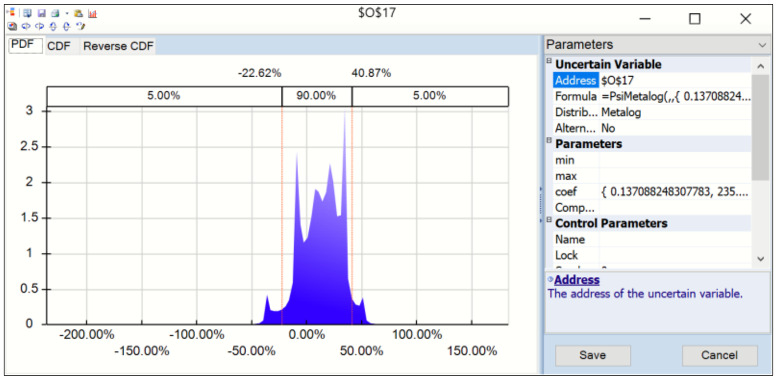


● Unbounded
○ Bounded
○ Lower -.5
○ Upper .6

Fit Statistic (AD)	0.097859573742...
Fit Limit (AD)	1.224
Formula	=PsiMetalog(,, { 0...
Shift	0
Minimum	-∞
Mode	NaN
Maximum	∞
Median	0.137088248128...
Mean	0.119954029929...
Std.Dev.	0.224073315526...
Variance	0.050208850730...
Skewness	-2.34813343930...
Kurtosis	73.28027369019...
Indep.Test	0.688054378447...
1% Quantile	-0.37696585237...
5% Quantile	-0.22619124761...
10% Quantile	-0.10959809078...
25% Quantile (Lo...	-0.01518831547...
50% Quantile (Me...	0.137088248128...
75% Quantile (Up...	0.263279604152...
90% Quantile	0.344459192679...
95% Quantile	0.408708849718...

5. Save the resulting distribution on the worksheet by clicking “X” at the top right of the menu, clicking on “Yes” in the resulting dialog to accept, and choosing where you want to place it in Excel. For this example, select cell O17. In Analytic Solver Cloud, you'll be asked for the cell address. Simply type the cell address and then click OK.

   The Uncertain Variable dialog appears. Click Save to save the distribution to cell O17.



Uncertain Variable

Address	$O$17
Formula	=PsiMetalog(,,{ 0.13708824...
Distrib...	Metalog
Altern...	No

Parameters

min	
max	
coef	{ 0.137088248307783, 235....
Comp...	

Control Parameters

Name	
Lock	

Address
The address of the uncertain variable.

   You can see the new distribution where you placed it by clicking the cell. It will show up in the formula bar and if you leave the mouse over that cell a pop-up will show your distribution.

More Hints and Warnings

Using a Triangular Distribution

If you have only estimates of the minimum, maximum, and most likely values of an uncertain variable – and no other past data or literature references – you can create a **PsiTriangular** distribution from these three numbers. This is *unlikely* to be a highly accurate representation of the uncertainty, but it will allow you to get started, and it is *far* better than a single average value. If your ‘minimum’ and ‘maximum’ values are really low- and high-percentile *estimates* rather than the absolute lowest and highest values that can occur, consider using the **PsiTriangGen** distribution instead.

Define Each Uncertain Variable Only Once

Often, you’ll need to use the same uncertain variable in several different formulas in your model. A very common error is to enter the same distribution function, with the same parameters (say PsiNormal(100, 10)), *several times* in a model – in a belief that these instances will yield the same results on each trial. This is *incorrect* – by doing this, you’ve actually defined *several independent* uncertain variables that may well sample *different values* on each trial. You should instead enter =PsiNormal(100, 10) in a cell such as A1, and use A1 in each cell formula where the variable is needed.

Compute Statistics Only on Final Outputs

Another common error is to use formulas in your model to compute a statistic across two or more uncertain variables, and then *use this statistic* in further calculations that are part of the same simulation model.

For example, suppose you have two uncertain variables, =PsiUniform(0,10) in cell A1 and =PsiUniform(0,20) in B1. You want to know the mean or average value of these two variables, across the trials of the simulation. The *wrong* way to do this is to put = (A1+B1)/2 or =AVERAGE(A1,B1) in cell C1. This takes the average of A1 and B1 *on each trial* – the result will be a distribution that tends to the Normal, with a different (larger) mean than you probably intended. The right way to do this is to put =PsiDisUniform(A1,B1) in cell C1. Now the distribution of cell C1 is “flat:” On each trial only one of A1 or B1 is selected.

David Vose, principal of Vose Consulting, offers his ***cardinal rule of risk analysis modeling***: “Every trial of a risk analysis model must be a scenario that could actually occur.” To ensure that this is true, you may need to specify *correlations* between your uncertain variables, as described in the next section.

Dependence and Correlation

Unless you specify otherwise, Monte Carlo simulation assumes that each of your uncertain variables is *independent* of all the other uncertain variables. When two variables are independent, the value of one variable on a given trial conveys no information about the value of the other variable. When two variables are *dependent*, there is a statistical relationship between them: On average, the value of one variable on a given trial *does* convey information about the value of the other variable on that trial.

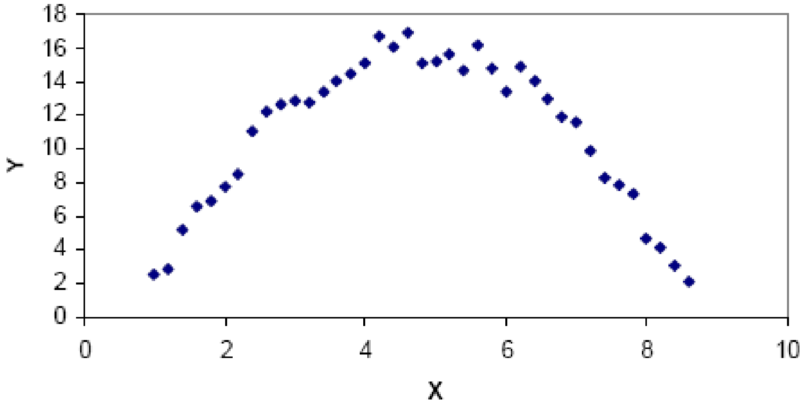
Measuring Observed Correlation

Correlation is a statistical measure of the degree to which one variable is related to another. When we observe that two variables are correlated, it may be that one variable is dependent on the other, or that both variables are dependent on a third variable, *or* that the correlation appeared by chance and there is no real dependence of one variable on the other.

The most common parametric measure of correlation is the **Pearson product moment correlation coefficient**. You can use the **PsiCorrelation** function to compute this correlation coefficient, across the trials of a simulation, for any pair of uncertain functions or variables in your model. The value of a product moment correlation coefficient can range from -1 to +1.

A correlation coefficient captures a simple *linear* relationship between two variables; it cannot capture all of the ways one variable may depend on another. A low or zero correlation coefficient between two uncertain variables or functions does *not* necessarily mean that the variables or functions are *independent* – the two variables or functions might have a strong relationship, but one that cannot be captured by a single correlation coefficient.

The figure below is a scatter plot of two variables X and Y. There is an obvious relationship between these two variables – when X is 1, Y is about 2, when X is 5, Y is about 15, when X is 9, Y is about 2, and so on – yet the Pearson product moment correlation coefficient between X and Y is zero.



More complex relationships among variables can be expressed in several ways including *newly supported copulas* or a very general way using SLURPs (Stochastic Library Units, Relationships Preserved). For more information on copulas, see below. For more information SLURPs, see below under "Probability Management Concepts."

Inducing Correlation Among Uncertain Variables

If you can compute a value for A1 with a formula based on an uncertain variable B1, then certainly A1 is dependent on B1. But in many modeling situations, both A1 and B1 are *uncertain* variables, you have no way to directly compute one from the other, yet A1 may *statistically depend* on B1. For example, mortgage interest rates depend on bond market interest rates (since mortgages are pooled and sold as securities in the bond market), and both rates depend on inflation expectations, but it is quite difficult to specify a formula that relates these variables.

In cases like these, you'll need to specify the statistical dependence between uncertain variables using PSI Property functions, supplied as arguments to PSI Distribution functions. This is called *inducing correlation* between uncertain variables, that would otherwise be considered independent. For example, you can write =PsiNormal (100, 10, **PsiCorrIndep("MyCorr")**) in cell B1 and write =PsiUniform(0, 100, **PsiCorrDepen("MyCorr", 0.9)**) in cell A1 to specify that A1 depends heavily on B1. "MyCorr" is an arbitrary string name.

The number 0.9 in the example above is a **Spearman rank order correlation coefficient**. This is a *nonparametric* measure of correlation that is computed from a rank ordering of the trial values drawn for both variables. It can be used to induce correlations between *any* two uncertain variables, whether they have the same or different analytic distributions, or even custom distributions. Like the product moment correlation coefficient, its value can range from -1 to +1.

Meaning of Rank Correlation Coefficients

- A correlation coefficient of +1 forces the sampled values of the uncertain variables to be exactly positively correlated, meaning that the pth percentile value from one distribution will be sampled on the same trial as the pth percentile value from the other distribution. Coefficients from 0 to +1 will produce varying degrees of positive correlation.
- A correlation coefficient of -1 forces the sampled values of the uncertain variables to be exactly negatively correlated, meaning that the pth percentile

value from one distribution will be sampled on the same trial as the (100-p^{th}) percentile value from the other distribution. Coefficients from 0 to -1 will produce varying degrees of negative correlation.

- A correlation coefficient of 0 means there is no induced relationship between the variables. In practice, one usually uses coefficients less than +1 or -1, and uses 0 only in a correlation matrix that defines relationships among several variables (see below).

Computing Rank Correlations from Sample Data

If you have sufficient data on the past performance of two uncertain variables where the observations occurred at the same time for each variable, you can compute the Spearman rank correlation coefficient r as follows, where ΔR is the difference between the ranks of corresponding observations of the two variables:

$$r = 1 - \frac{6\sum_{i=1}^{n} \Delta R_i^{\,2}}{n\left(n^2 - 1\right)}$$

$$\Delta R_i = \Delta R_{i1} - \Delta R_{i2}$$

ΔR_i is the difference between the ranks of observation i for the two variables

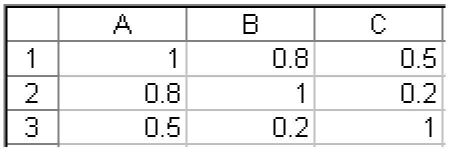
ΔR_{ij} is the rank of observation i for variable j, $j = 1, 2$

You can easily compute this value on a spreadsheet, using Excel's RANK function, which returns the rank of a given cell value within a range of values. Suppose that A1:A100 contains trial values for uncertain variable A, and B1:B100 contains trial values for uncertain variable B. In C1 enter the formula =RANK(A1, $A$1:$A$100,1), in D1 enter =RANK(B1,$B$1:$B$100,1), and in E1 enter the formula =(C1-D1)^2. Then copy C1:E1 down to row 100. Then the rank order correlation coefficient is =1-(6*SUM(E1:E100)/(100*(100^2-1)).

If you don't have data on past performance of the uncertain variables, you will have to use judgment to estimate rank correlation coefficients.

Using a Rank Correlation Matrix for Several Variables

What if you have three, four, or more uncertain variables that should all be correlated with each other? You can create a small table or matrix of rank correlation coefficients in a cell range on the worksheet, and use this cell range in the PSI Property function **PsiCorrMatrix**. Below is an example of a 3x3 correlation matrix:



	A	B	C
1	1	0.8	0.5
2	0.8	1	0.2
3	0.5	0.2	1

You pass PsiCorrMatrix (*matrix cell range, position*) as an argument to the PSI Distribution function, for example =PsiNormal (10,5,PsiCorrMatrix(A1:C3,1)) for the first uncertain variable covered by the correlation matrix. You'd pass PsiCorrMatrix(A1:C3,2) to the PSI Distribution function for the second variable, and PsiCorrMatrix(A1:C3,3) for the third. This specifies that the first variable has a rank correlation coefficient of 0.8 with the second variable, and 0.5 with the third variable. The second and third variables are correlated with each other, with a rank correlation coefficient of 0.2.

Note that a correlation matrix must always have 1's on the diagonal, because an uncertain variable is always perfectly correlated with itself. Also, the matrix must be symmetric: If row 2, column 1 contains 0.8, then row 1, column 2 must also contain 0.8. Finally, the correlation coefficients must be consistent with each other: For example, if uncertain variable 1 is strongly positively correlated with variable 2, and variable 2 is strongly positively correlated with variable 3, then variable 1 cannot be negatively correlated with variable 3. Formally, the matrix must be *positive semidefinite* – it cannot have any negative eigenvalues. Analytic Solver tests for this condition, and displays #N/A in the PSI Distribution cells for the uncertain variables that refer to an inconsistent correlation matrix.

Note: PsiCorrMatrix accepts an optional third string argument, *Instance,* where the name of the correlation matrix may be passed. For more information, see the PsiCorrMatrix entry in the Psi Function Reference chapter within the Analytic Solver Reference Guide.

To learn more about rank correlation and its uses in Monte Carlo simulation, consult the book *Risk Analysis: A Quantitative Guide* by David Vose, mentioned in the Introduction.

Modeling Correlation Using Copulas

Analytic Solver includes copulas to improve the method of defining the correlation or dependence between two or more uncertain variables. Copulas offer more flexibility over the rank order correlation method, and are able to capture complex correlations between multiple uncertain variables. In recent years, the popularity of copulas in finance and insurance risk models has skyrocketed.

An n – dimensional copula C is a multi-variate probability distribution where the marginal probability distribution of each variable follows the Uniform(0,1) distribution. A major benefit of copulas is that they allow two or more uncertain variables to be correlated without changing the shape of the original uncertain variable distributions.

For some copula C, a multi-variate distribution F with distributions of F_1, F_2, … F_n can be written as:

$$\mathbf{F(x_1, \ldots, x_n) = C(F_1 (x_1), F_2 (x_2), \ldots, F_n (x_n))}$$

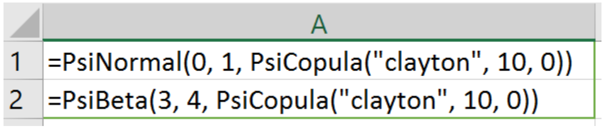
Analytic Solver Desktop supports five types of copulas: three Archimedean copulas (clayton, frank, and gumbel) and two elliptical copulas (Gauss and Student). Analytic Solver Cloud supports one type: Gaussian.

Each of these copulas may be bi-variate (correlating only two uncertain variables), or multi-variate (correlating more than two uncertain variables). Each copula type has its own parameter domain and method of calculation. The signatures for each copula are the same, except that multi-variate Archimedean copulas lack the "reflection" argument. For details regarding Archimedian copulas, see the following sections. For a complete discussion of the theory behind all five copula types, see the section *Psi Property Functions* in the *Frontline Solvers Reference Guide*.

Archimedean Copulas

Archimedean copulas can be bi-variate (correlating two uncertain variables) or multi-variate (correlating more than two uncertain variables). A bi-variate copula has a density function associated with it, which is similar to the probability density of a bi-variate distribution. Archimedean Copulas are not supported in Analytic Solver Cloud.

To create a bi-variate Archimedean copula, enter two uncertain variable distributions into two Excel cells, and then use the PsiCopula property as an argument to both. See the screenshot below for an example.



	A
1	=PsiNormal(0, 1, PsiCopula("clayton", 10, 0))
2	=PsiBeta(3, 4, PsiCopula("clayton", 10, 0))

Cell A1 contains an uncertain variable that follows the PsiNormal distribution with mean = 0 and standard deviation = 1. Cell A2 contains an uncertain variable following the PsiBeta distribution with shape parameters equal to 3 and 4. The PsiCopula property is passed as the third argument to each uncertain variable. All three Archimedean copulas use the same signature.

```
PsiCopula(type, param, [reflection], [instance])
```

`Type`: As previously mentioned, Analytic Solver supports three types of Archimedean copulas: clayton, frank, and gumbel. The copula type should be passed in quotes as, "clayton", "frank", or "gumbel". In this example, "clayton" is passed. The two elliptical copulas have their own Psi property names: PsiCopulaGauss and PsiCopulaStudent. See the example below for their signatures.

`Param`: The parameter of a copula determines the strength of the correlation. In this example, a parameter equal to 10 is passed. See the following parameter restrictions for each Archimedean copula type.

Clayton:

- Bi-variate: param >= -1, param ≠ 0.
- Muilti-variate: param > 0

Gumbrel

- Bi-variate or Multi-variate: param >= 1

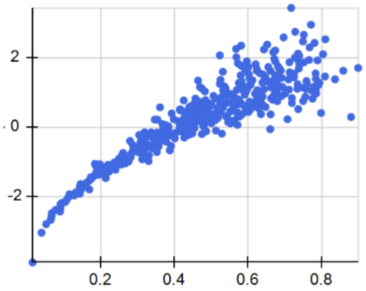
Frank

- Bi-variate: param ≠ 0.
- Muilti-variate: param > 0

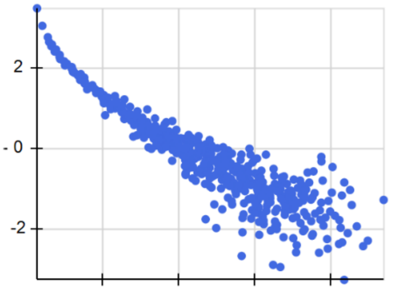
Note: Param = 0 for a multi-variate Archimedean copula is supported, but indicates no correlation between the uncertain variables.

`Reflection`: (Optional) Analytic Solver allows you to control the direction of a bi-variate Archimedean copula using an optional reflection parameter. This argument may take on values of 0, 1, 2, or 3, which control the reflection of no variables using the value of 0, the first variable using the value of 1, the second variable using the value of 2 or both variables using the value of 3. By default, the reflection option is set to 0, which indicates no reflection. In this example, a 0 is passed for the reflection argument to illustrate how this parameter should be passed. In practice, a value of 0 need not be present.

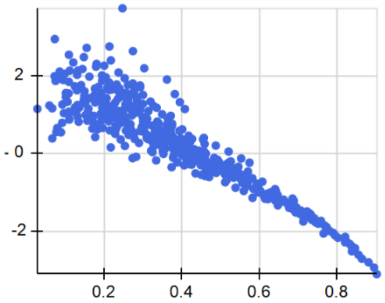
The scatter plots below illustrate the four different positions of a clayton copula correlating two uncertain variables with distributions PsiNormal(0,1) and PsiBeta(3,4).



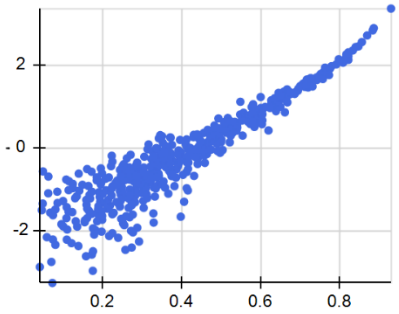
Reflection = 0



Reflection = 1

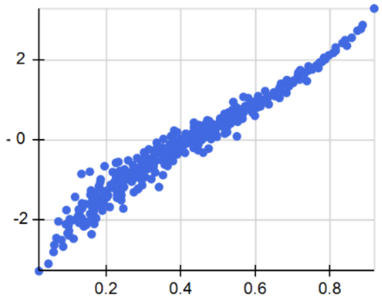


Reflection = 2

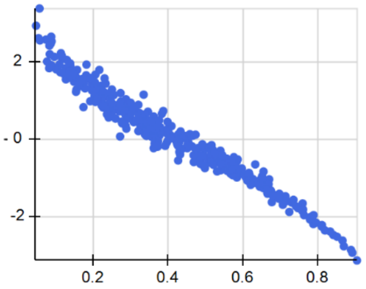


Reflection = 3

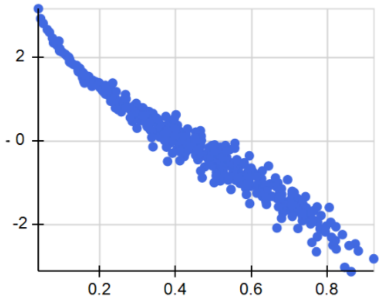
The scatter plots below illustrate the four different positions of a type gumbel copula correlating two uncertain variables with distributions PsiNormal(0,1) and PsiBeta(3,4).



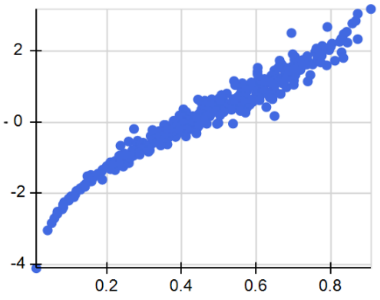
Reflection = 0



Reflection = 1

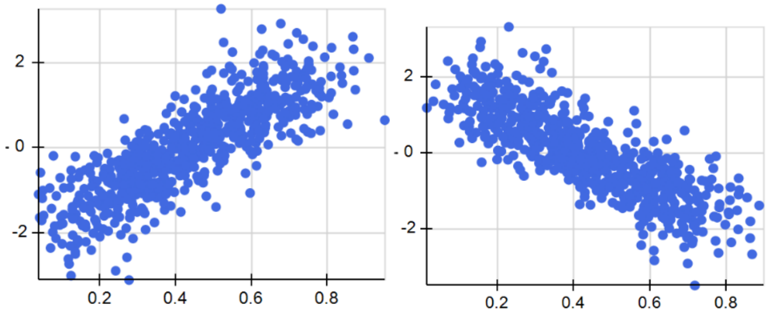


Reflection = 2

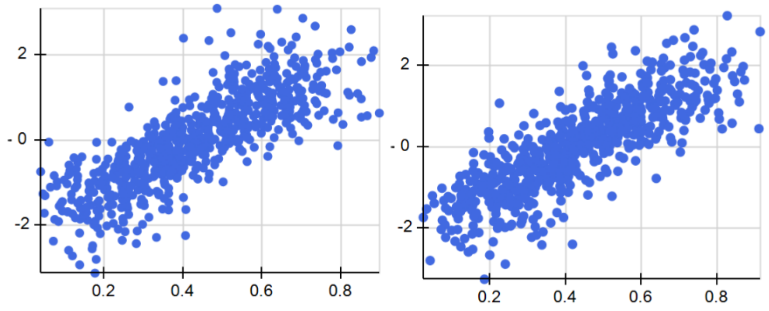


Reflection = 3

The scatter plots below illustrate the four different positions of a type frank copula correlating two uncertain variables with distributions PsiNormal(0,1) and PsiBeta(3,4). Note: Since the Frank copula is symmetric, this type of copula has only one reflection.



Reflection = 0 Reflection = 1



Reflection = 2 Reflection = 3

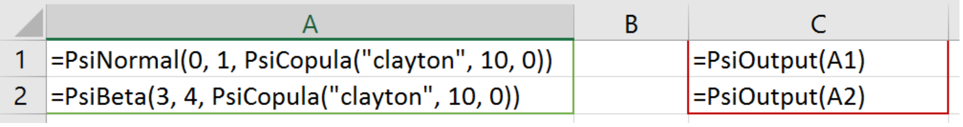
`Instance`: (Optional) Instance is the string name given to the copula. An Archimedean copula is explicitly identified by the Instance argument, and implicitly identified by its argument values. When multiple copulas are present in the same workbook, it is considered "best practice" to use this argument.

In this example, we correlate the two uncertain variables using the PsiCopula property, =PsiCopula("clayton", 10, 0). Since the Instance property is missing, the copula is identified *implicitly* by its unique set of arguments, in this case "clayton" and "10". *When not passing the Instance property, the PsiCopula property within each uncertain variable signature, MUST use the same arguments.* Otherwise, the correlation between the uncertain variables will not be invoked.

Note: PsiCopula("clayton", 10) and PsiCopula("clayton", 10, 0) is considered as the same copula since 0 is the default for Reflection argument.

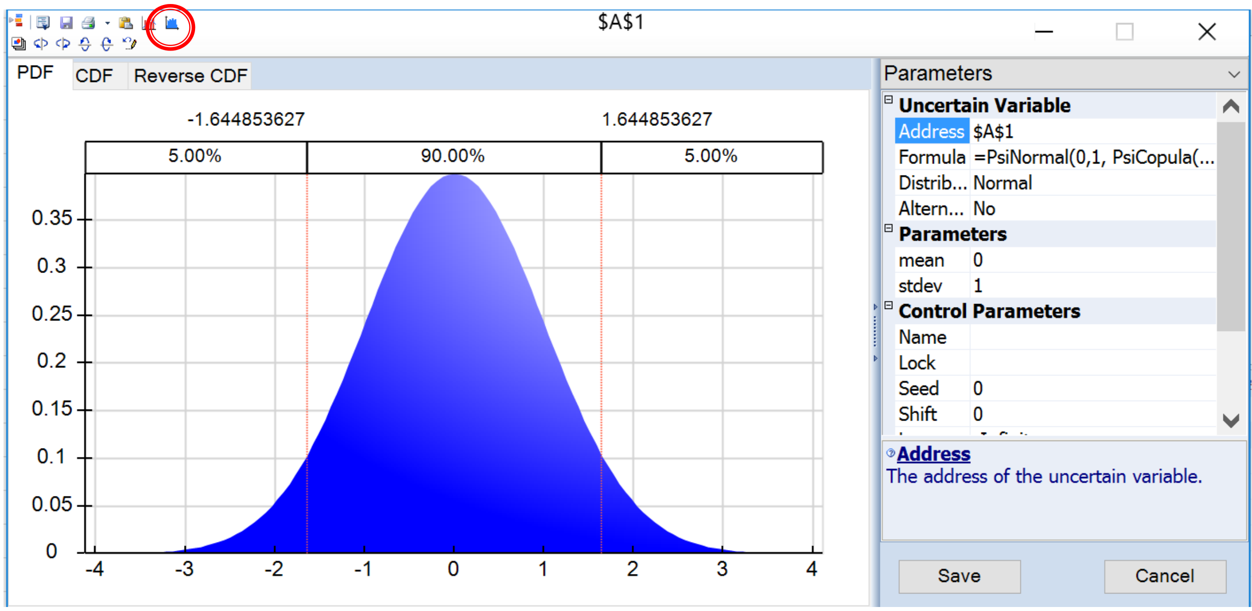
To correlate a new group of uncertain variables using a second copula, such as PsiCopula("clayton", 12) or PsiCopula("frank", 10), the Instance argument is still not required since either copula is identified by its unique parameters of "clayton" and "12" or "frank" and "10". However, if correlating this same new group of uncertain variables with a copula using the same arguments of "clayton" and "10", we MUST pass a unique name to the Instance argument, for example, "copula2".

Finally, to view the output of the copula, we must make each uncertain variable an uncertain function. We can easily do so by clicking cell A1, then **Results – Output -- Referred** on the Analytic Solver ribbon. Next, select an empty Excel cell, say C1. The formula, =PsiOutput(A1), is inserted into this cell. Perform similar steps to make cell A2 an output function. (See the screenshot below for an example.)



	A	B	C
1	=PsiNormal(0, 1, PsiCopula("clayton", 10, 0))		=PsiOutput(A1)
2	=PsiBeta(3, 4, PsiCopula("clayton", 10, 0))		=PsiOutput(A2)

Double-click cell A1 to open the Uncertain Variable dialog, then click the Show Results icon.



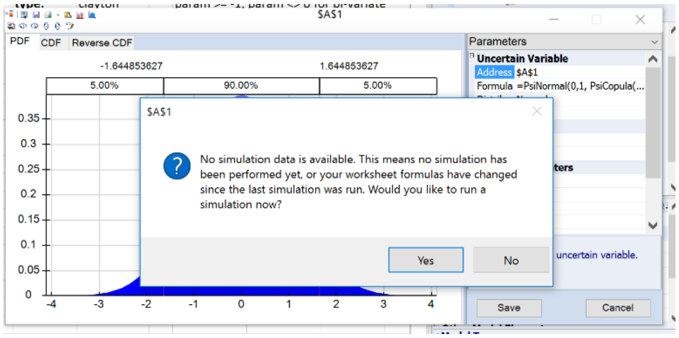
Uncertain Variable
Address $A$1
Formula =PsiNormal(0,1, PsiCopula(...
Distrib... Normal
Altern... No

Parameters
mean 0
stdev 1

Control Parameters
Name
Lock
Seed 0
Shift 0

Address
The address of the uncertain variable.

When Analytic Solver Simulation prompts you to perform a simulation, click **Yes**.



$A$1

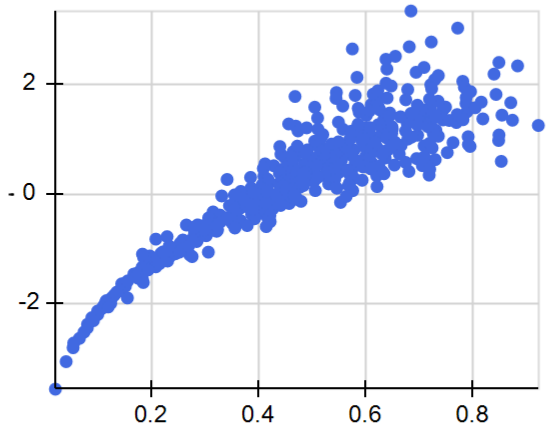
No simulation data is available. This means no simulation has been performed yet, or your worksheet formulas have changed since the last simulation was run. Would you like to run a simulation now?

Yes   No

Instantly, a simulation with 1,000 trial values is executed. Excel cells A1 and A2 return 1,000 trial values from the sampled distribution, generating the pattern of correlation defined by the copula.

Click the Scatter Plots tab to view the results of the copula. In this example, a copula of type "clayton" is created correlating two uncertain variables with distributions of PsiNormal(0,1) and PsiBeta(3, 4). The distribution generated for the first uncertain variable will still follow the Normal(0,1) distribution, and the distribution generated for the second variable will still follow the Beta(3,4) distribution. The scatter plot below displays the copula correlating the two uncertain variables using a parameter of 10.

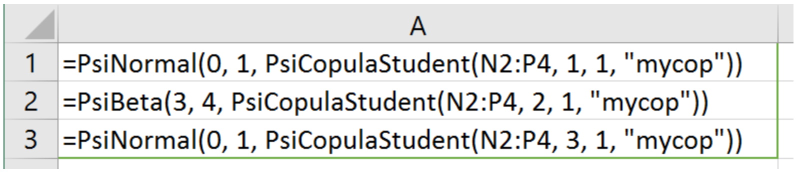
Note: If the copula is multi-variate, the associated scatter plots will be 2-D projections.



Elliptical Copulas

Analytic Solver Comprehensive supports two elliptical copulas that can be either bi-variate or multi-variate: Gaussian and Student. Each elliptical copula has its own signature. Student copulas are not supported in Analytic Solver Coud.

To create a multi-variate Student copula, enter three or more uncertain variable distributions into three (or more) different Excel cells, and then use the PsiCopulaStudent() property as an argument to each. See the screenshot below for an example.

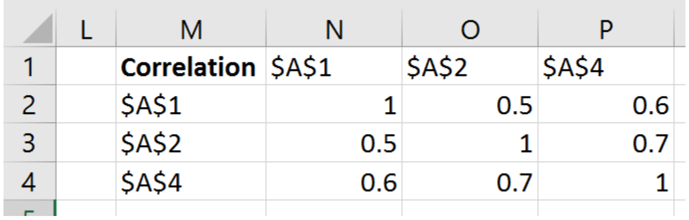


	A
1	=PsiNormal(0, 1, PsiCopulaStudent(N2:P4, 1, 1, "mycop"))
2	=PsiBeta(3, 4, PsiCopulaStudent(N2:P4, 2, 1, "mycop"))
3	=PsiNormal(0, 1, PsiCopulaStudent(N2:P4, 3, 1, "mycop"))

Cells A1 and A3 contain an uncertain variable that follows the PsiNormal distribution with mean equal 0 and standard deviation = 1. Cell A2 contains an uncertain variable following the PsiBeta distribution with shape parameters equal to 3 and 4. The PsiStudentCopula property is passed as the third argument to each uncertain variable. The PsiCopulaStudent property uses the following signature. Note: The signature for PsiCopulaGauss() is the same minus the "df" argument.

```
PsiCopulaStudent(number_range, position, df,
[instance])
```

`Number range`: If bi-variate, a number between -1 and 1; if multi-variate, the Excel cell range where the correlation matrix is located in the Excel spreadsheet. In this instance, the copula is multi-variate, so the correlation matrix in cells N2:P4 is passed.



	L	M	N	O	P
1		**Correlation**	$A$1	$A$2	$A$4
2		$A$1	1	0.5	0.6
3		$A$2	0.5	1	0.7
4		$A$4	0.6	0.7	1

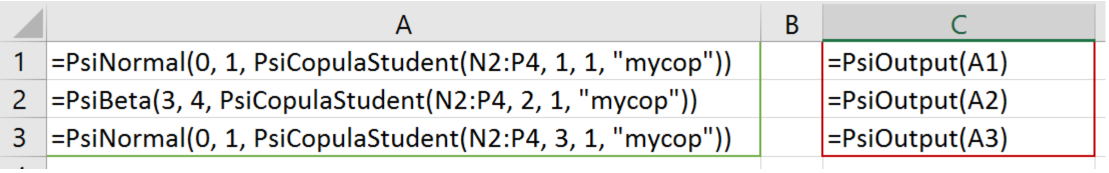
`Position`: Specifies the uncertain variable index in the correlation matrix. For example, the "2" passed in the formula, =PsiBeta(3, 4, PsiCopulaStudent(N2:P4, **2**, 1, "mycop") specifies that the correlation coefficients in the second column of the correlation matrix in cells N2:P4 will be applied to the PsiBeta() uncertain variable.

`df`: Enter an integer greater than 1. This parameter specifies the degrees of freedom for the PsiCopulaStudent function. In this example, 1 degree of freedom is used. (Note: The formula for PsiCopulaGauss() does not include this argument.)

`Instance`: (Optional) Instance is the string name given to the copula. An elliptical copula is explicitly identified by the Instance argument and implicitly identified by the location of the correlation matrix. PsiCopulaStudent and PsiCopulaGauss support multiple copulas using the same correlation matrix. If passing the same type of elliptical copula using the same correlation matrix within the same workbook, this argument **must** be present. If the workbook contains multiple copulas of different types, then this argument may be omitted. When multiple copulas are present in the same workbook, it is considered "best practice" to use this argument.

In this example, we correlate the three uncertain variables using the PsiCopulaStudent property, =PsiCopulaStudent(N2:P4, 1, 1, "mycop"). Since the Instance property is present, the copula is identified *explicitly* by the name given in the last argument, "mycop". *When not passing the Instance property, the PsiCopulaStudent property within each uncertain variable signature, MUST use the same correlation matrix.* Otherwise, the correlation between the uncertain variables will not be invoked.

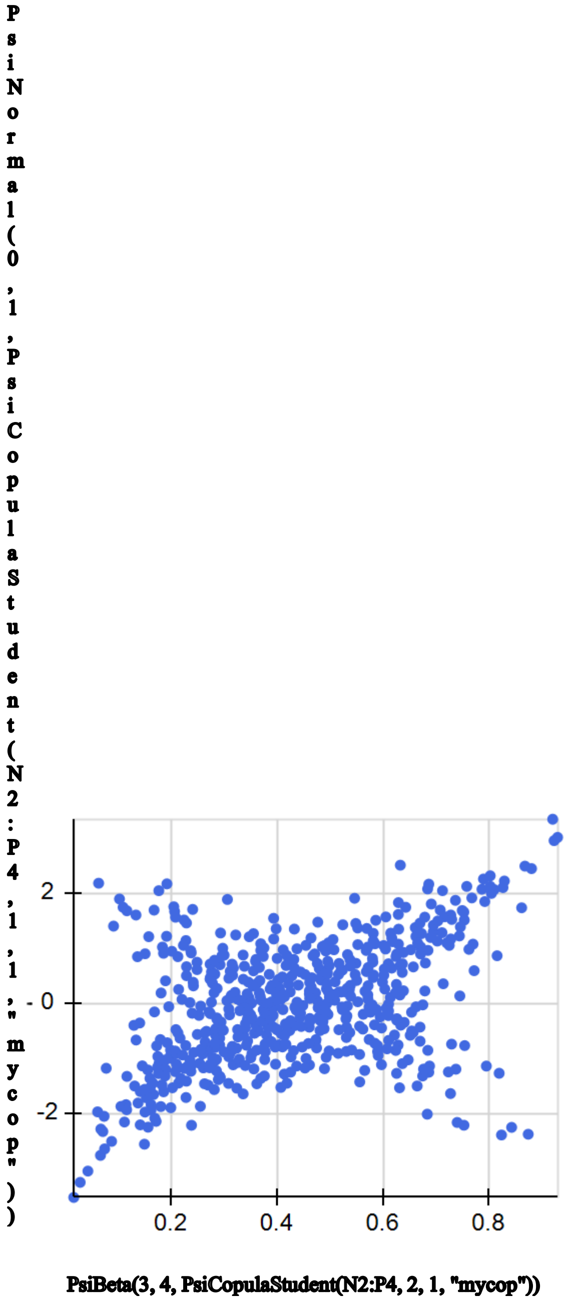
To view the output of the copula, we must make each uncertain variable an uncertain function. We can easily do so by clicking cell A1, and then **Results – Output -- Referred** on the Analytic Solver ribbon. Next, select an empty Excel cell, say C1. The formula, =PsiOutput(A1), is inserted into this cell. Perform similar steps to make cells A2 and A3 output functions. (See the screenshot below for an example.)

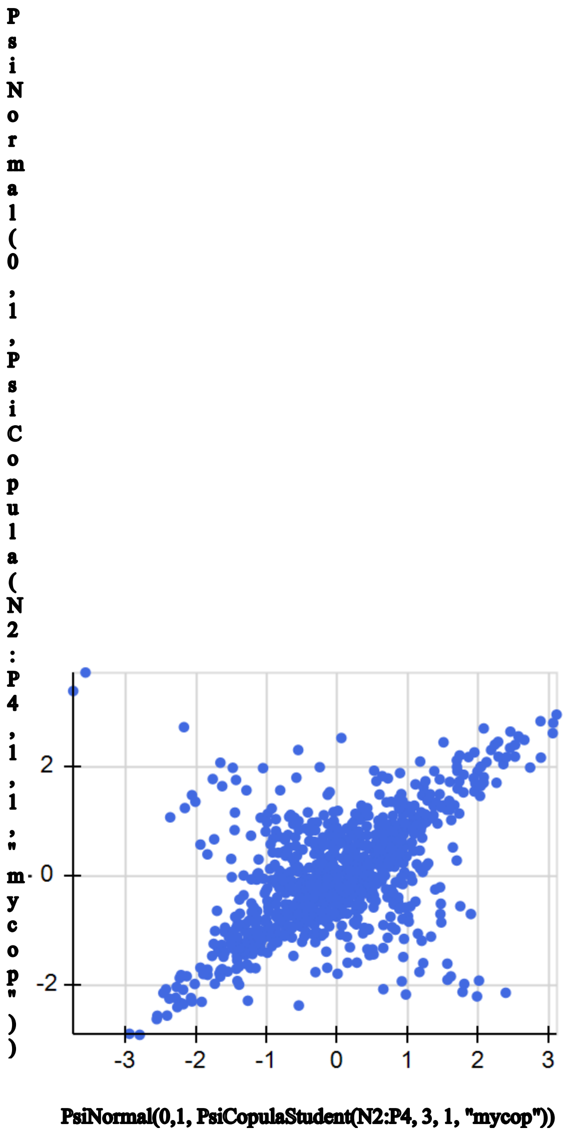


	A	B	C
1	=PsiNormal(0, 1, PsiCopulaStudent(N2:P4, 1, 1, "mycop"))		=PsiOutput(A1)
2	=PsiBeta(3, 4, PsiCopulaStudent(N2:P4, 2, 1, "mycop"))		=PsiOutput(A2)
3	=PsiNormal(0, 1, PsiCopulaStudent(N2:P4, 3, 1, "mycop"))		=PsiOutput(A3)

On the Analytic Solver ribbon, click the down arrow under Simulate, then select Run Once from the menu. Instantly, a simulation with 1000 trial values is executed. The result is a correlated uncertain function following the Uniform(0,1) distribution. Excel cells A1, A2 and A3 return 1000 trial values from the sampled distribution generating the pattern of correlation defined by the copula.

In this example, a copula of type "student" is created correlating three uncertain variables with distributions of PsiNormal(0,1) and PsiBeta(3, 4). The distributions generated for the first and third uncertain variables will still follow the Normal(0,1) distribution and the distribution generated for the second variable will still follow the Beta(3,4) distribution. The scatter plots below show plots from the PsiNormal(0,1) distribution, the PsiBeta(3,4) distribution and the copula correlating the three uncertain variables.


PsiNormal(0,1,PsiCopulaStudent(N2:P4,1,1,"mycop"))
2
0
-2
0.2
0.4
0.6
0.8
PsiBeta(3, 4, PsiCopulaStudent(N2:P4, 2, 1, "mycop"))



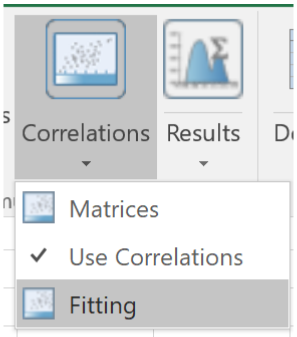
Correlation Fitting

The previous chapter, "Getting Results: Simulation", discussed how to fit a distribution to simulation results. Recall that, in this chapter, the simulation results from the BusinessForecast(Sim).xlsx model were fit to the Min-Extreme uncertain variable distribution. Analytic Solver calculated the parameters of several distributions from the data within the example model and listed the distributions with the best fit, with Min-Extreme at the top of the list.

Similarly, the goal of correlation fitting is to find the parameters of the correlation function that match the existing (but unknown) correlation between two or more samples of historical data. Each sample is assumed to come from an independent uncertain variable. Analytic Solver offers correlation fitting using 6 correlation types: rank, clayton, gumbel, frank, gauss and student. It is up to the user to determine which correlation type results in the best fit to the data. Note: Rank is not a copula correlation, but rather a stand-alone type of correlation.

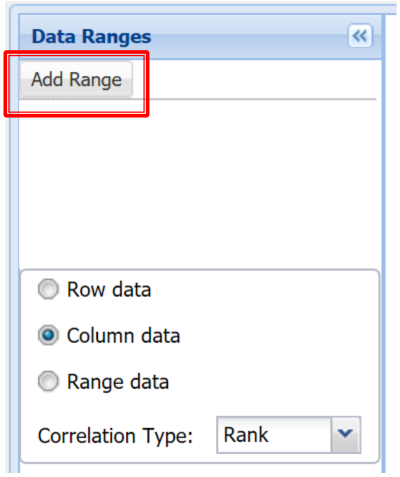
For example, if a scatter plot is drawn from two independent samples resembling a tilted ellipse, one can assume that the correlation will be a “Gauss” correlation. If the correlation is fit, using the Gauss fitting algorithm, one rank-correlation coefficient will be returned. If more than 2 samples are drawn, the fitting algorithm will return a correlation matrix of size n.

Analytic Solver users can fit a correlation using the Copula Fit dialog which can be opened from the Analytic Solver ribbon by clicking the down arrow beneath the Correlations icon and selecting Fitting.



Two or more Excel cell ranges containing independent samples can be added to the Copula Fit dialog by clicking Add Range. Data may exist in row form, column form, or in a matrix. For example, if you had two independent samples located in cells A1:A100 and B1:B100 you would select Column Data, then click Add Range on the Copula Fit dialog.

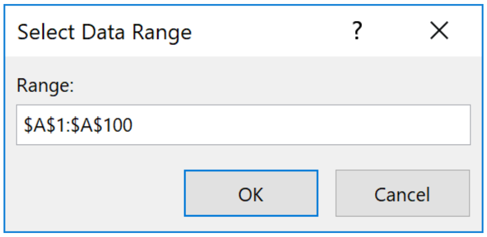
If the data were contained by row, then you would select Row data and if data were contained in, say, a 10 x 10 matrix in cells A1:J10, then you would select Range data.



○ Row data
● Column data
○ Range data

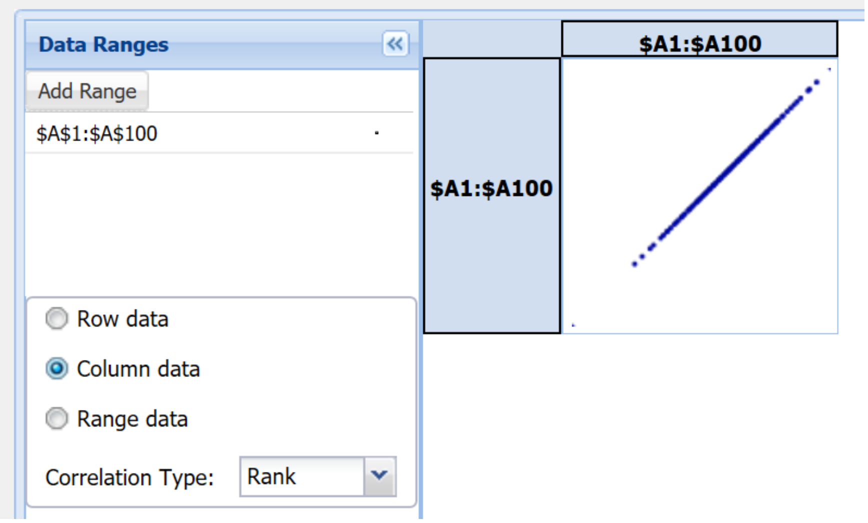
Correlation Type: Rank

Then either select or enter A1:100 on the Select Data Range dialog.



Range: $A$1:$A$100

After you click OK, the data range will appear under Data Ranges and a scatter plot will appear on the right of the dialog.

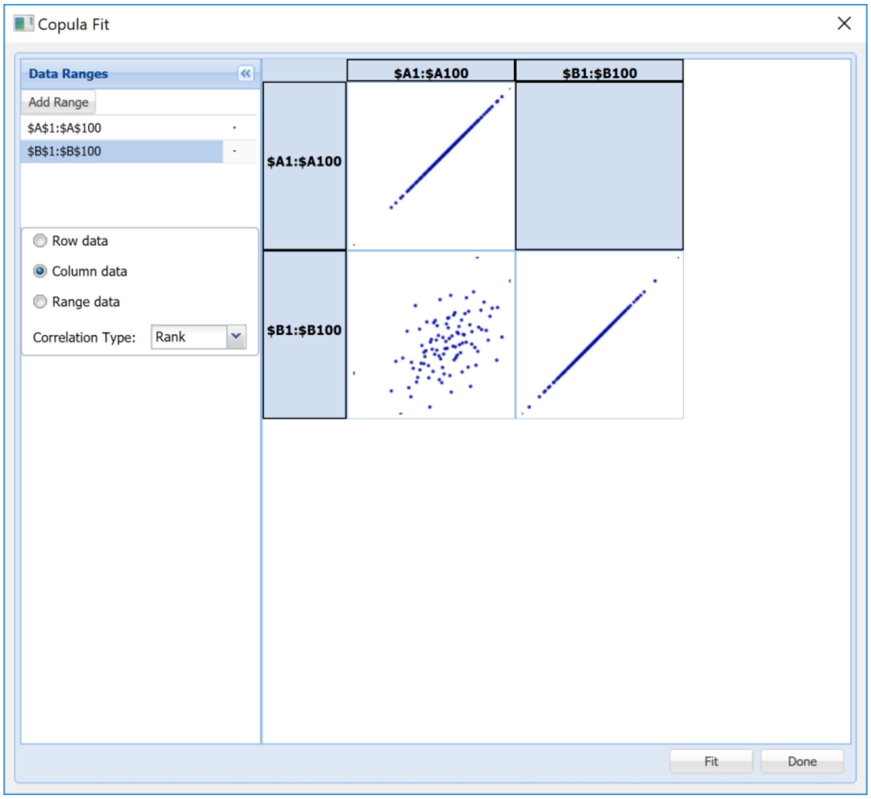


$A$1:$A$100

○ Row data
● Column data
○ Range data

Correlation Type: Rank

To enter the 2nd range, keep Column Data selected, then click Add Range again, enter the Excel range, B1:B100, then click OK to add the data range to the Copula Fit dialog.



$A$1:$A$100
$B$1:$B$100

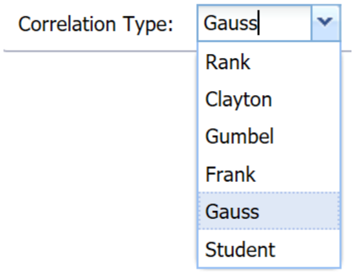
○ Row data
● Column data
○ Range data

Correlation Type: Rank

Notice that the scatter plot that has been drawn for the two ranges resembles an ellipse. From this scatter plot, we can assume that the copula correlation type for fitting is “Gauss”. A shape resembling a star points to a Student copula for Correlation Type and a shape resembling a funnel would indicate a Clayton copula. Although a complete discussion of the definition of copulas is beyond the scope of this guide, see the previous section, “Modeling Correlation Using

Copulas” for a brief introduction of each type of copula defined in the Correlation Type drop down menu.

Select Gauss as Correlation Type…

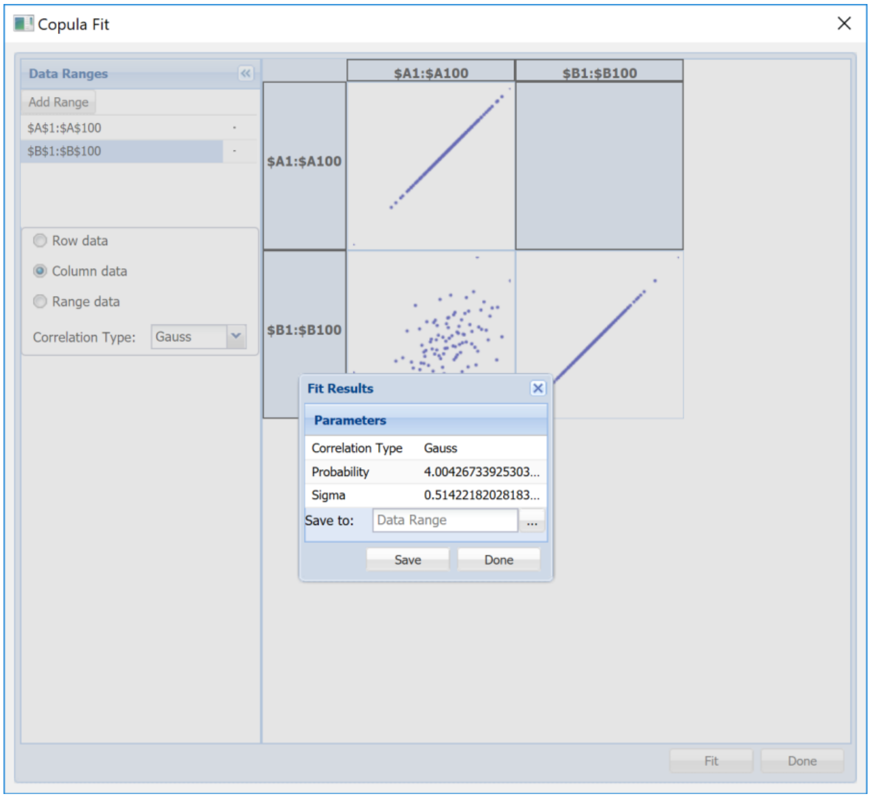


Correlation Type: Gauss

…then click Fit at the bottom of the Copula Fit dialog to fit the correlation.

The Fit Results Parameters dialog opens to display the Correlation Type (Gauss) and the Probability and Sigma fitting parameters. Select a blank cell, say D1, to save the fitting parameters to the worksheet, then click Save.

Note: The cell entered for “Save to” on the Fit Results dialog will become the top left corner of the data range on the Excel sheet. Any value or text contained in this cell or in any cell used to save the fit Results Parameters, will be overwritten.



$A$1:$A$100
$B$1:$B$100

○ Row data
● Column data
○ Range data

Correlation Type: Gauss

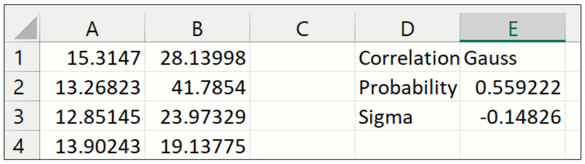
Fit Results

Parameters

Correlation Type	Gauss
Probability	4.00426733925303...
Sigma	0.51422182028183...

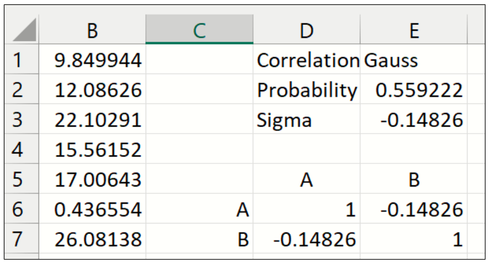
Save to: Data Range

Then click Done to close the Copula Fit dialog and return to the worksheet. Notice that the information from the Fit Results dialog has been saved to the Excel range, D1:E3. The size of this range will vary depending on the Correlation Type and number of Data Ranges present in the Copula Fit dialog.



	A	B	C	D	E
1	15.3147	28.13998		Correlation	Gauss
2	13.26823	41.7854		Probability	0.559222
3	12.85145	23.97329		Sigma	-0.14826
4	13.90243	19.13775			

From here, create a 2 x 2 correlation matrix with the Sigma value as the upper and lower coefficients and 1's on the diagonal.



	B	C	D	E
1	9.849944		Correlation	Gauss
2	12.08626		Probability	0.559222
3	22.10291		Sigma	-0.14826
4	15.56152			
5	17.00643		A	B
6	0.436554	A	1	-0.14826
7	26.08138	B	-0.14826	1

Now that the copula has been fit, the Sigma value will be used in the PsiCopulaGauss signature, for example…

=PsiNormal(5,2,**PsiCopulaGuass(E3, 2)**) where

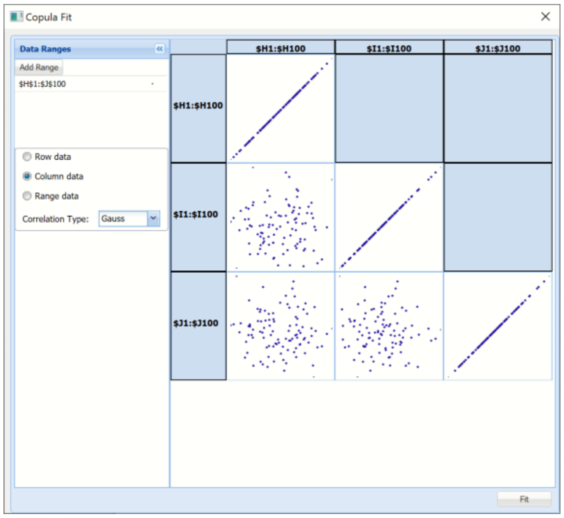
- E3 is the sigma value from the fitted copula and
- 2 specifies the correlation coefficients in the 2nd column of the correlation matrix which will be applied to the PsiNormal() uncertain variable.

If more than 2 ranges are added, Copula Fit returns a correlation matrix of n x n where n is the number of ranges. Use this correlation matrix as the 2nd argument to the copula.

Assume the data is included in cells H1:J100 (3 ranges of 100 cells each).

- Click Correlations – Fitting,
- click Add Range -- H1:J100,
- select Column Data,
- select Gauss for correlation type

to produce the chart below.



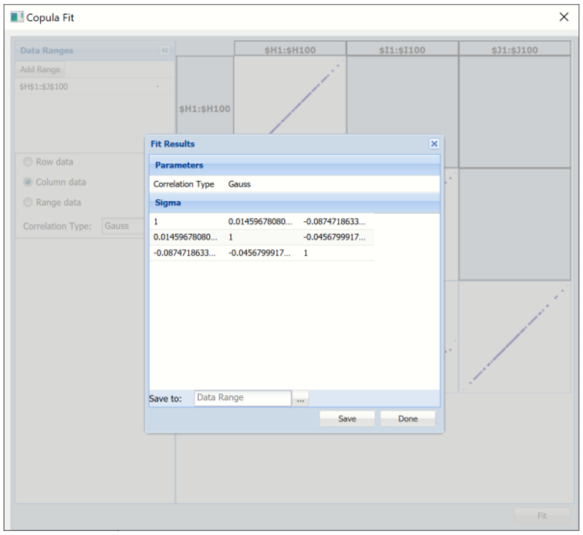
$H$1:$J$100

● Column data

Correlation Type: Gauss

Notice that the selection is divided into columns because "Column Data" was selected.

Click Fit to fit the copula. Then enter a blank cells for Save to and click Save.



● Column data

Correlation Type: Gauss

Parameters

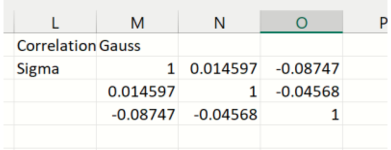
Correlation Type Gauss

Sigma

1	0.01459678080...	-0.0874718633...
0.01459678080...	1	-0.0456799917...
-0.0874718633...	-0.0456799917...	1

Save to: Data Range

A correlation matrix with three columns is saved to the worksheet.



L	M	N	O	P
Correlation	Gauss			
Sigma	1	0.014597	-0.08747	
	0.014597	1	-0.04568	
	-0.08747	-0.04568	1	

When using the copula in an uncertain variable signature, use just the correlation matrix cells, for example:

=PsiNormal(5,2,**PsiCopulaGauss(M2:O4,2)**) where

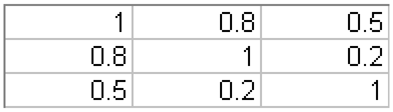
- M2:O4 is the location of the correlation matrix returned from Copula Fit.
- 2 specifies the correlation coefficients in the 2nd column of the correlation matrix which will be applied to the PsiNormal() uncertain variable.

Using the Correlations Dialog

The Correlation dialog offers an easy way to create, edit, and remove correlation matrices in your model. It appears when you click the Correlations button on the Ribbon.

You use a correlation matrix to *induce* a statistical correlation among two or more uncertain variables. When you do this, the trial values of these variables in a Monte Carlo simulation will tend to be drawn from related percentiles of their distributions on each trial. Starting with V2016-R2, the numbers in a correlation matrix may be **Spearman rank order correlation coefficients** or specifying an **Archimedean** or **Elliptical copula.** For more information on correlation matrices, see “Inducing Correlation Among Uncertain Variables” in the Dependence and Correlation section above.

A correlation matrix is stored in a contiguous cell range, with an equal number of columns and rows. To correlate *N* variables together, you need a matrix of *N* columns and *N* rows. Here's an example of a 3x3 correlation matrix:

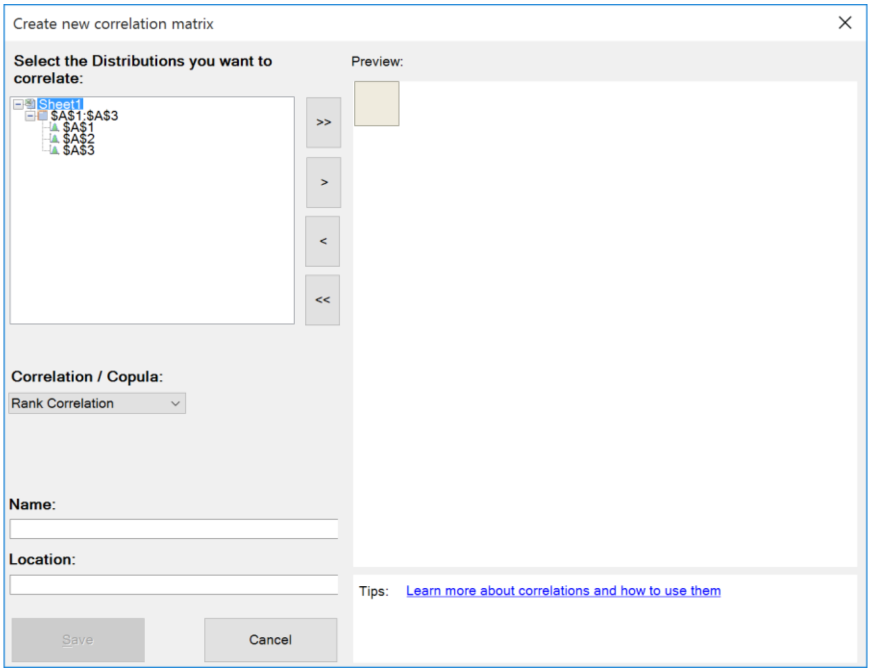


1	0.8	0.5
0.8	1	0.2
0.5	0.2	1

You can type in values for a correlation matrix directly on the Excel worksheet, and then reference this matrix in the PsiCorrMatrix() property function, passed as an argument to PSI Distribution functions for each of the correlated variables. But it is easier to use the Correlation Dialog to create the matrix for you.

Creating a Correlation Matrix

When you first click the Correlations button on the Ribbon, the Correlation dialog appears with an empty matrix area, and a left pane (similar to the Task Pane displayed by the Model button) that lists the uncertain variables in the model:

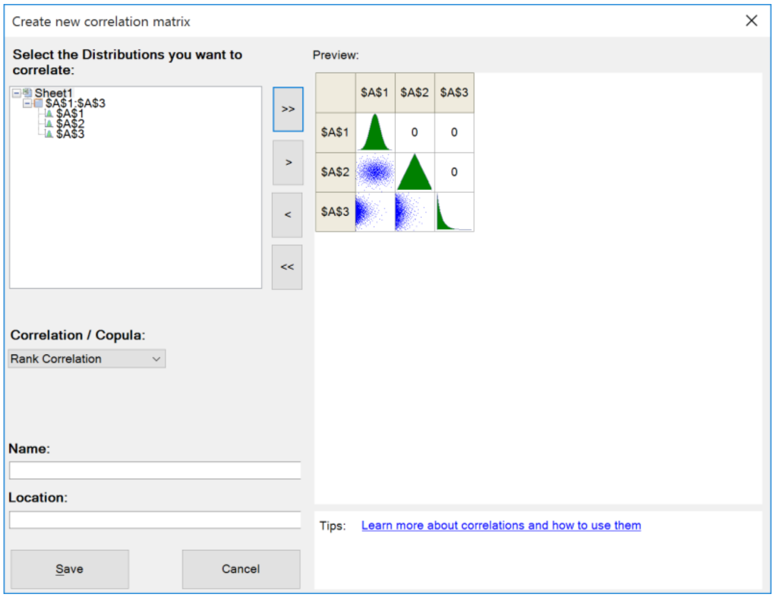


Create new correlation matrix

Select the Distributions you want to correlate:

Correlation / Copula: Rank Correlation

You can either choose the specific cells for the uncertain variables you want to correlate (press Ctrl or Shift and click for more than one) and then click the ">" button to include them in the matrix, or you can click the ">>" button to include all of them. The initial matrix with correlation values set to zero will display similar to the example below:


Create new correlation matrix
Select the Distributions you want to correlate:
Sheet1
$A$1:$A$3
$A$1
$A$2
$A$3
>>
>
<
<<
Preview:
$A$1
$A$2
$A$3
0
0
0
Name:
Location:
Tips:
Learn more about correlations and how to use them
Save
Cancel

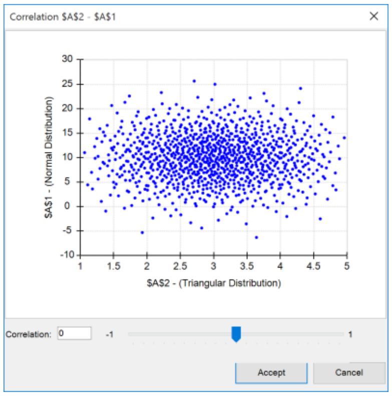
Correlation / Copula: Rank Correlation

The preview for the correlation Matrix has three parts:

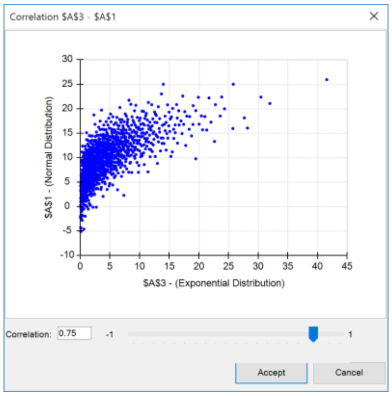
1. Black numerical values for each correlation in the top right of the matrix. In this case we have three values initially set with a zero correlation which you can edit. Since correlation matrices have to be symmetrical (the correlation between A1 and A2 must be the same as the correlation between A2 and A1) we only show one value. Note: Both values are shown in Analytic Solver Cloud.

2. Green graphical representations of each distribution diagonally from top left to bottom right. In this example, we used a Normal distribution in our worksheet in cell A1 (show here in the top left part of the matrix), a Triangular distribution in cell A2, and a LogNormal distribution in cell A3. Note: Graphical representations are not currently available in Analytic Solver Cloud.

3. Blue scatterplots reflecting the current correlations between the distributions. When you edit the values of each correlation, these will automatically update giving you a visual representation of the updated correlation. Note, you can click on any scatterplot here and a dialog will come up where you can more clearly see the scatterplot and how the scatterplot changes as you adjust the correlation (see below). In the example on the left the correlation is set to zero while on the right we have adjusted it (either directly typing in 0.75 into the Correlation field or adjusting the slider. Note: Scatterplots are not currently available in Analytic Solver Cloud.


Correlation $A$2 - $A$1
$A$1 - (Normal Distribution)
$A$2 - (Triangular Distribution)
-1
1
Accept
Cancel

Correlation: 0

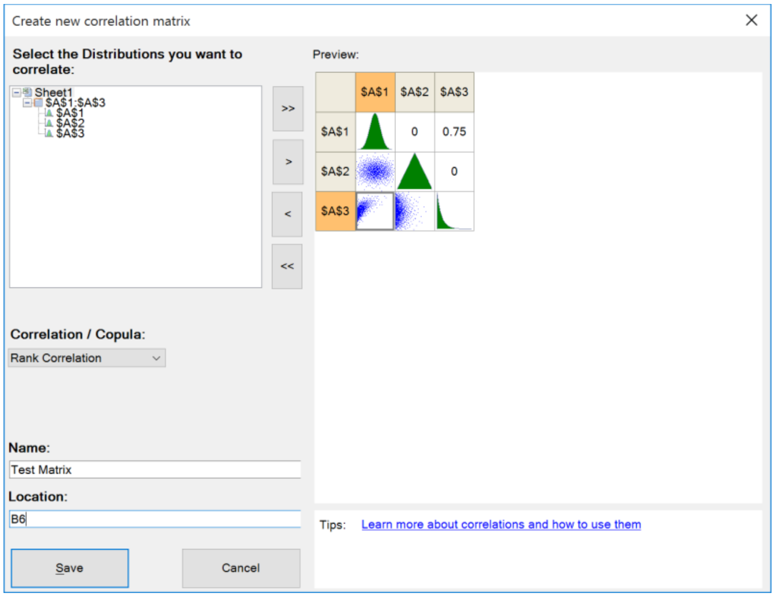

Correlation $A$3 - $A$1
$A$1 - (Normal Distribution)
$A$3 - (Exponential Distribution)
-1
1
Accept
Cancel

Correlation: 0.75

Once you click “Accept” you are taken back to the initial dialog and it shows the updated correlation matrix.

At this point you can name your correlation matrix and then choose the location you want to place it in your worksheet. While optional, naming your matrix is recommended, and is important if you want to use the same matrix several times. To place your matrix, simply click your cursor in the location field and click where you want to place the matrix on your worksheet.

Analytic Solver Desktop


Create new correlation matrix
Select the Distributions you want to correlate:
Sheet1
$A$1:$A$3
$A$1
$A$2
$A$3
>>
>
<
<<
Preview:
Tips:
Learn more about correlations and how to use them
Save
Cancel

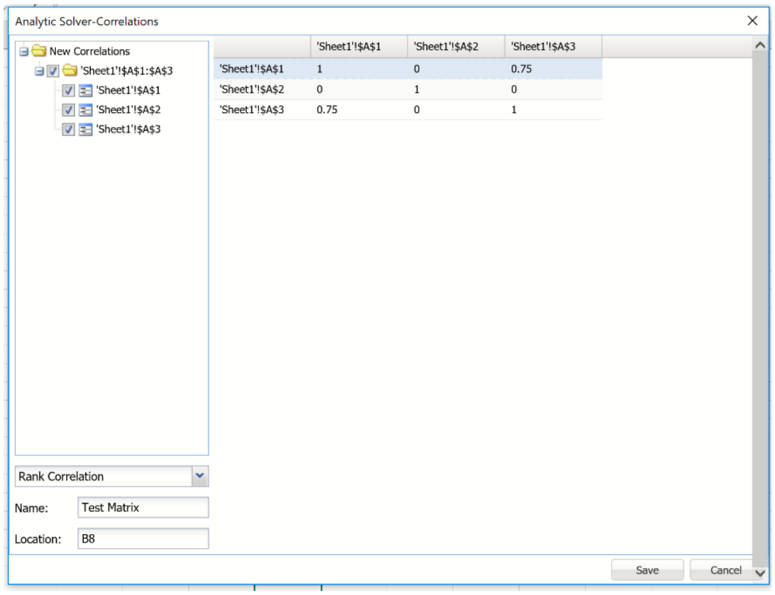
	$A$1	$A$2	$A$3
$A$1		0	0.75
$A$2			0
$A$3			

Correlation / Copula: Rank Correlation

Name: Test Matrix

Location: B6

Analytic Solver Cloud



	'Sheet1'!$A$1	'Sheet1'!$A$2	'Sheet1'!$A$3
'Sheet1'!$A$1	1	0	0.75
'Sheet1'!$A$2	0	1	0
'Sheet1'!$A$3	0.75	0	1

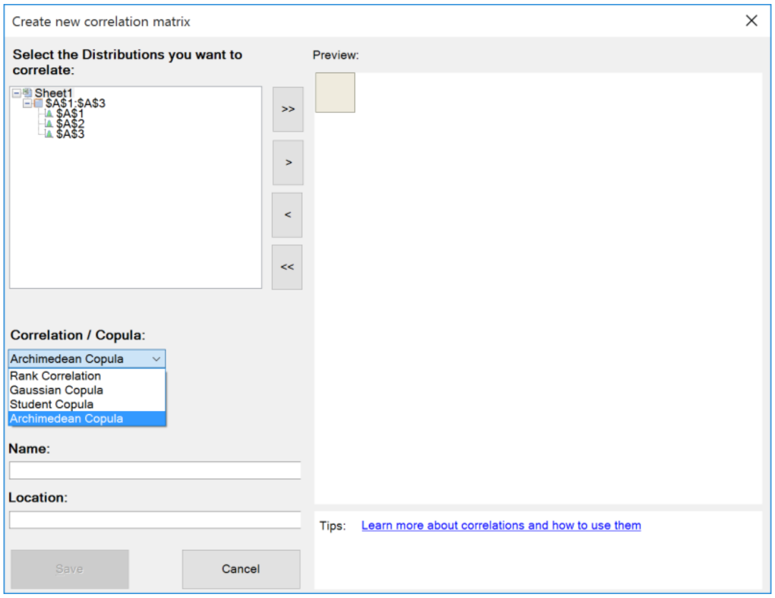
Rank Correlation

Name: Test Matrix

Location: B8

Managing Copulas

Analytic Solver includes the ability to correlate uncertain variables using Gaussian, Student or Archimedean Copulas. To correlate the three uncertain variables in cells A1, A2, and A3 using an Archimedian Copula, select Archimedian Copula from the Correlation/Copula drop down menu. Note: Only Rank Order and Gaussian Copulas are supported in Analytic Solver Cloud.



Correlation / Copula: Archimedean Copula

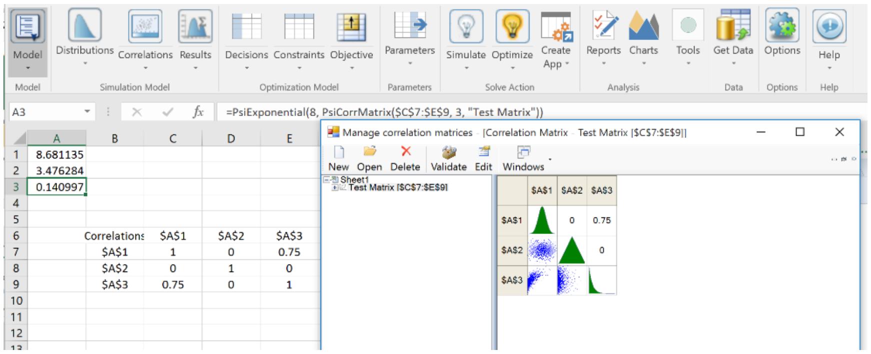
Name:

Location:

Tips: Learn more about correlations and how to use them

Once the Copula type is selected, three more fields will be enabled, Type, Parameter, and Reflection. Using these three parameters you may choose the type of Archimedean copula (Clayton, Frank, or Gumbel), specify the copula parameter, and choose a Reflection type. If Gaussian Copula is selected, no additional parameters are required. If Student Copula is selected, one additional parameter is required, Degrees of Freedom. For more information on copulas and copula parameters, see “Inducing Correlation Among Uncertain Variables” in the Dependence and Correlation section above.

Lastly, click “Save” to complete the process. Your new correlation matrix will then be saved and placed on your worksheet (in this case in cell B6) and the Manage Correlation Matrices Dialog will show in place of the Create New Correlation Matrix dialog. This dialog lists all the matrices you have set up on the left side of the dialog, and when you double-click on a particular matrix it will show on the right side.



A3

=PsiExponential(8, PsiCorrMatrix($C$7:$E$9, 3, "Test Matrix"))

	A	B	C	D	E
1	8.681135				
2	3.476284				
3	0.140997				
4					
5					
6		Correlations	$A$1	$A$2	$A$3
7		$A$1	1	0	0.75
8		$A$2	0	1	0
9		$A$3	0.75	0	1
10					
11					
12					

As the matrix is created, Analytic Solver Simulation also edits the PSI Distribution function formulas for the uncertain variables (at A1, A2, & A3 in the example above) to include the property function **PsiCorrMatrix(C7:E9, *n*)**, where *n* is 1 for A1, 2 for A2, and 3 for A3. During a Monte Carlo simulation, when trial values are drawn for the uncertain variables A1, A2 and A3, Analytic Solver will use the correlation matrix at C7:E9 to adjust the ‘draws’ so that the values are properly correlated across all the simulation trials.

The matrix C7:E9 now appears in the left pane of the Manage Correlation Matrices dialog. If you close and later reopen the Correlation dialog, this matrix will reappear in the lower left pane. If you create a correlation matrix ‘by hand’ on the Excel worksheet, and insert the PsiCorrMatrix() calls yourself into the PSI Distribution function calls for the variables, the matrix you create will also appear in this left pane the next time you open the Correlation dialog. You can double-click on each matrix to display and edit that matrix.

Removing a Correlation Matrix

To remove a correlation matrix from your model, in the Manage Correlation Matrices dialog click on the matrix you want to remove and then click on the “Delete” icon in the dialog’s ribbon menu. The matrix will disappear from the Correlations dialog, and the PsiCorrMatrix() property function formula referencing this matrix will be removed from the PSI Distribution function calls for the corresponding uncertain variables. However, the cell range on the worksheet where the matrix elements were written is left undisturbed – you can either delete them in Excel worksheet mode, or re-use them later.

To remove a correlation matrix in Analytic Solver Cloud, simply select the matrix in the Model task pane, then click the red X to delete.

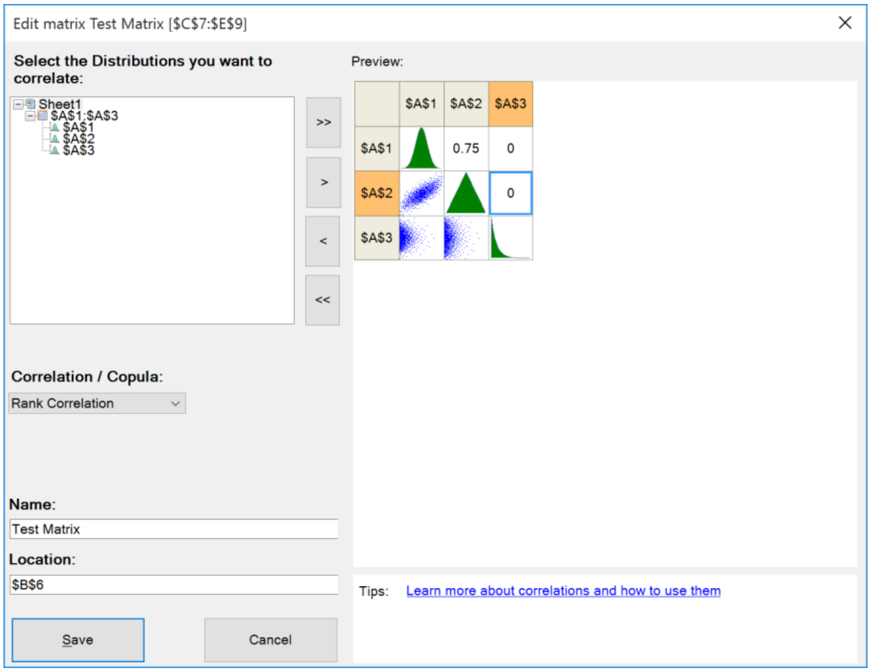
Editing a Correlation Matrix

As shown in the both the Create Correlations and Manage Correlations dialogs, the value of each correlation is numerically shown in the top right portion of the matrix. To edit the value of a correlation you can simply click on the correlation you wish to edit and enter a new value from -1 (exactly negatively correlated) to +1 (exactly positively correlated). The default value of 0 means the two variables have zero correlation.

When you are finished, click the “X” button in the top right of the dialog to close it and your change(s) will be saved. Importantly, we will automatically check to make sure your matrices are valid (Positive Semi-Definite; see “Making a Matrix Consistent” below) and if they are not will offer help to correct them.

Alternatively, you can click on the Edit button on the Manage Correlations Ribbon to go to the Edit Matrix Dialog. The Edit Matrix dialog (see below) will appear, which looks very similar to the Create Matrix dialog. At this point you can change the name and location of the matrix, add or remove uncertain variables, and change correlation values. When done, simply hit **Save**.

In Analytic Solver Cloud, double click the correlation matrix in the Model task pane to edit. Click save when finished.


Select the Distributions you want to correlate:
Sheet1
$A$1:$A$3
$A$1
$A$2
$A$3
>>
>
<
<<
Preview:
Save
Cancel
Tips:
Learn more about correlations and how to use them

Edit matrix Test Matrix [$C$7:$E$9]

	$A$1	$A$2	$A$3
$A$1		0.75	0
$A$2			0
$A$3			

Correlation / Copula: Rank Correlation

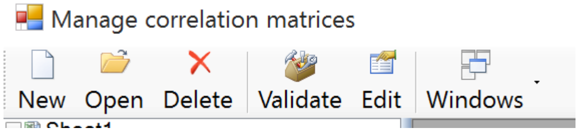
Name: Test Matrix

Location: $B$6

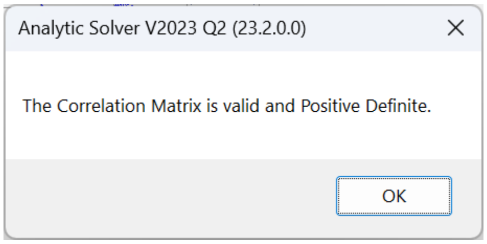
Making a Matrix Consistent

A correlation matrix must be not just symmetrical, but ***consistent***. This means for example that if variable A has a high positive correlation with variable B, and B has a positive correlation with C, then variable A cannot have a high negative correlation with C. In mathematical terms, the correlation matrix must be *positive semidefinite*. This property depends on all of the elements of the matrix. If the matrix you create by hand does not satisfy this property (the example matrix above does not), Analytic Solver can automatically adjust the matrix elements so that the matrix is positive semidefinite.

To check whether the matrix you’ve entered is positive semidefinite, in the Manage Correlations dialog simply click the Validate icon in the dialog’s ribbon menu:

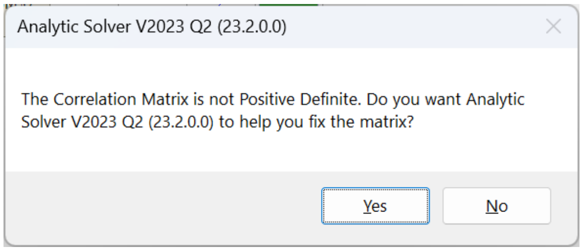


In our example, the matrix is valid so the following message is displayed and no action is needed:


Analytic Solver V2023 Q2 (23.2.0.0)

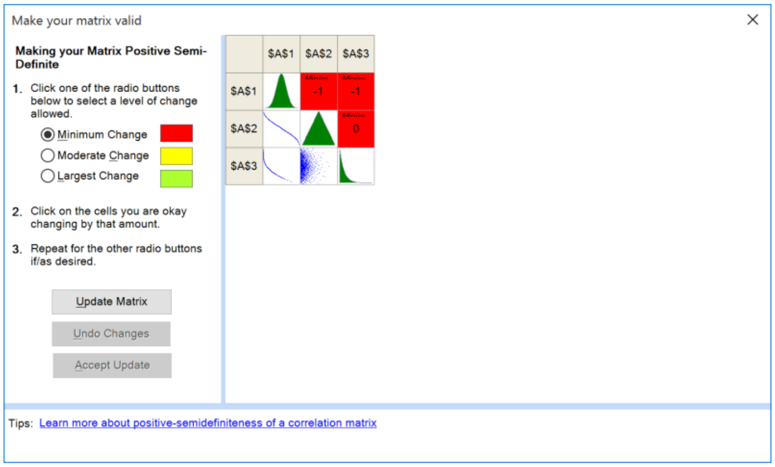
The Correlation Matrix is valid and Positive Definite.

If, for example, we set all our correlations to be -1, we would have a matrix which could not be valid and you will see the following message instead:


Analytic Solver V2023 Q2 (23.2.0.0)

The Correlation Matrix is not Positive Definite. Do you want Analytic Solver V2023 Q2 (23.2.0.0) to help you fix the matrix?

If you click No, the matrix remains as-is. If you click Yes, a new dialog appears:


Make your matrix valid

Making your Matrix Positive Semi-Definite

1. Click one of the radio buttons below to select a level of change allowed.
   - ● Minimum Change 🟥
   - ○ Moderate Change 🟨
   - ○ Largest Change 🟩
2. Click on the cells you are okay changing by that amount.
3. Repeat for the other radio buttons if/as desired.

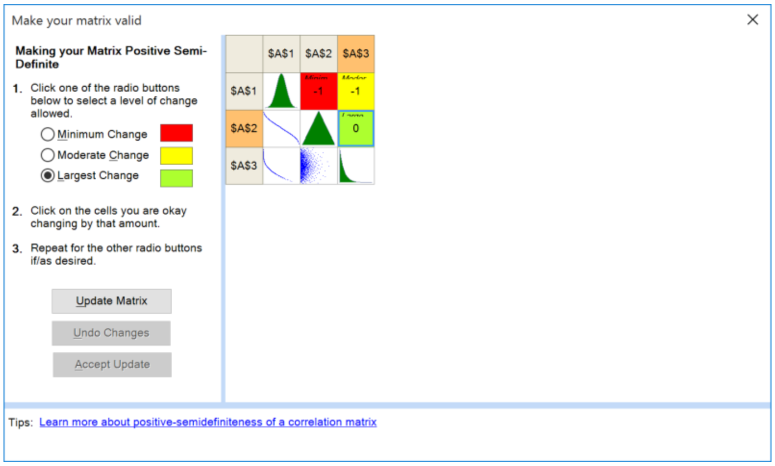
Tips: Learn more about positive-semidefiniteness of a correlation matrix

In this dialog, we can simply click the Update Matrix button to allow Analytic Solver to adjust the matrix to be consistent (positive semidefinite). This may involve changing all of the elements of the matrix, by various amounts. But Analytic Solver can do more than this.

In many applications, you can determine 'good' values for correlations between certain pairs of variables, but for other pairs you have no special information. Analytic Solver Simulation can adjust the matrix to make it positive semidefinite while minimizing changes to the matrix elements you care about, and making greater changes in the elements you don't care about.

In the example above, let’s say we feel very confident there is a strong negative correlation between A1 and A2 but are less sure about A1 and A3 and very uncertain about any correlation between A2 and A3. We can click first on the radio button on the left relating to the amount of change we are comfortable with and then click on the cell(s) we want changed by that general amount.

Using the information above we would click on the “Moderate Change” radio button and then click on the correlation for A1 and A3 and then click on the “Largest Change” radio button followed by clicking on the correlation for A2 and A3. Our Matrix will now look like the following:



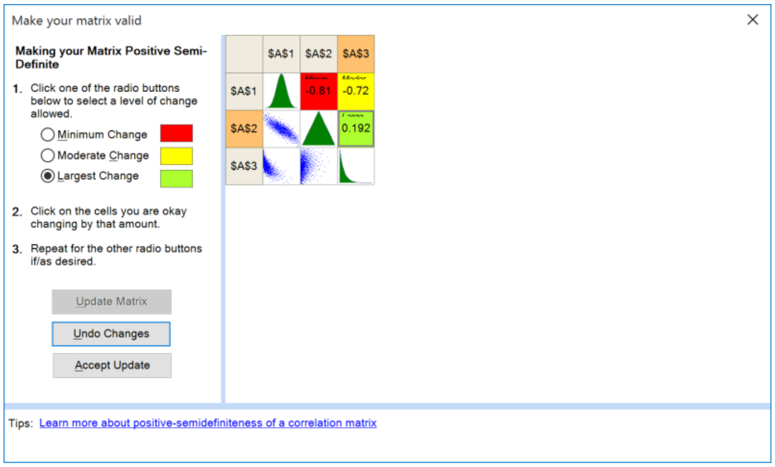
Making your Matrix Positive Semi-Definite

1. Click one of the radio buttons below to select a level of change allowed.
   - ○ Minimum Change 🟥
   - ○ Moderate Change 🟨
   - ● Largest Change 🟩
2. Click on the cells you are okay changing by that amount.
3. Repeat for the other radio buttons if/as desired.

	$A$1	$A$2	$A$3
$A$1		-1	-1
$A$2			0
$A$3			

Update Matrix
Undo Changes
Accept Update
Tips: Learn more about positive-semidefiniteness of a correlation matrix

At this point we just click the “Update Matrix” button and our matrix is updated to be consistent (Positive Semi-Definite) with new correlation values as shown below:



Making your Matrix Positive Semi-Definite

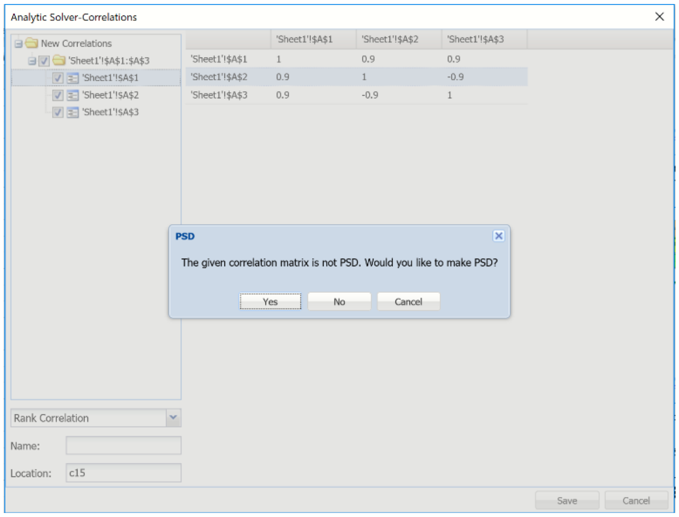
1. Click one of the radio buttons below to select a level of change allowed.
   - ○ Minimum Change 🟥
   - ○ Moderate Change 🟨
   - ● Largest Change 🟩
2. Click on the cells you are okay changing by that amount.
3. Repeat for the other radio buttons if/as desired.

	$A$1	$A$2	$A$3
$A$1		-0.81	-0.72
$A$2			0.192
$A$3			

Update Matrix
Undo Changes
Accept Update
Tips: Learn more about positive-semidefiniteness of a correlation matrix

We can now accept the update or undo the changes. We can of course further refine our correlation values if/as we wish.

In Analytic Solver Cloud, each time the Save button is clicked on the Correlations dialog, the matrix is checked for consistency (positive semi-definite). If the matrix is not consistent, you will be asked if you would like Analytic Solver to modify the matrix. Click yes, to accept Analytic Solver's modifications. Click No to save the matrix without any modifications. Click Cancel to go back to the matrix.



	'Sheet1'!$A$1	'Sheet1'!$A$2	'Sheet1'!$A$3
'Sheet1'!$A$1	1	0.9	0.9
'Sheet1'!$A$2	0.9	1	-0.9
'Sheet1'!$A$3	0.9	-0.9	1

The given correlation matrix is not PSD. Would you like to make PSD?

Rank Correlation

Name:

Location: c15

Probability Management Concepts

In the February and April 2006 issues of the publication *OR/MS Today*, Dr. Sam Savage and coauthors Stefan Scholtes (University of Cambridge) and Daniel Zweidler (Shell Global Exploration) advance a series of ideas called "Probability Management." They argue for much broader use of risk analysis methods in planning and management, especially in large enterprises, based on three ideas:

- Interactive Simulation
- Stochastic Libraries
- Certification Authority

Probability Management calls for an up-front investment in the creation of standardized Certified Distributions, created or reviewed by an expert authority, and distributed to simulation modelers, often in the form of Stochastic Libraries. The payoffs of Probability Management include:

- Much easier creation of new simulation models
- Ability to capture complex dependencies – beyond what's possible with correlation coefficients
- 'Apples-to-apples' comparison of simulation model results
- Valid 'roll-ups' of simulation models from different groups

Analytic Solver Comprehensive is the first product designed to support the concepts of Probability Management. In addition to best-in-class support for *Interactive Simulation*, it provides direct support for creating, 'publishing' and using *Certified Distributions* and *Stochastic Libraries*, as discussed in this chapter.

Analytic Distributions

As described above, and documented in the Reference Guide, Analytic Solver provides a wide range of analytic probability distributions – more than 40 distributions are available, from **PsiBernoulli()** to **PsiWeibull()**. Given knowledge and experience, you can choose and use these distributions in your simulation models. But as discussed above, this is often the most challenging part of creating a Monte Carlo simulation model, especially for users with spreadsheet modeling expertise, but limited math and statistics background.

Moreover, when different end user modelers choose their own analytic distributions and parameters, they will often create models that cannot be compared or combined (‘rolled up’), because the relationships between uncertain variables and functions would not be preserved. In large firms especially, where different business units must take into account common uncertainties, a better approach is needed.

Certified Distributions and Stochastic Libraries

A Certified Distribution is a custom probability distribution, created and/or reviewed by an expert, that is made available to end user modelers as a ‘prepackaged unit.’ These modelers need only the *name* of the Certified Distribution; they need not choose, or even be aware of, its analytic form, parameters or correlations.

An analytic Certified Distribution can be given its own random number seed, which overrides a user-specified seed for the model. This ensures that end user modelers employing the Certified Distribution will use the same *samples* (sequence of Monte Carlo trials) each time they run a simulation – as long as they use the same simulation software package and version.

Given the capacity of modern computers and networks, an even better idea is to create and distribute Certified Distributions in the form of **Stochastic Libraries**. Such libraries contain *pre-generated trial data* for a group of (potentially statistically dependent) distributions. The sequence of trials is predetermined, so that if two or more end users develop and run simulation models using the same Stochastic Library, their model results can be compared and combined, on a trial-by-trial basis if necessary.

Analytic Solver Desktop supports the use of both analytic distributions and Stochastic Libraries as Certified Distributions. But since they contain pre-generated trial data, the use of Stochastic Libraries can *guarantee* that models will be run with exactly the trial data intended, regardless of the software (or version thereof) installed on end user modelers’ PCs – as long as it accepts Stochastic Libraries. *This functionality is not currently supported in Analytic Solver Cloud.*

Publishing and Using Certified Distributions

Certified Distributions can be prepared and tested using a variety of tools. One of these tools is Analytic Solver Desktop itself, as illustrated below. When creating a Stochastic Library, any good Monte Carlo simulation software package can be used, inside or outside of Excel, as long as it offers a way to save all the Monte Carlo trials. Once prepared, Certified Distributions should be approved and ‘published’ in a form where they can be made available to end user modelers.

Analytic Solver Simulation provides a property function **PsiCertify()** that you can use to name and “certify” a distribution as ready for publication. It also provides a distribution function **PsiCertified()** that end user modelers can use to access the Certified Distribution using only its name.

Certified Distributions can be physically distributed or made accessible in a variety of ways: On CDs or DVDs, via a network file share, or via email or

Web download. The data for Monte Carlo trials may be easily stored in a relational database or a multidimensional data warehouse.

Analytic Solver Simulation, with its *Interactive Simulation* facilities, is the ideal 'client' tool for end user modelers using Certified Distributions. It provides an easy way to access Certified Distributions by *name*, using Microsoft Excel as a "universal data access client" to access distributions and Stochastic Libraries.

Analytic Solver Simulation makes it easy for an expert or CPO to create an Excel workbook, separate from any end user's simulation model workbook, where Certified Distributions are defined. This may be a **standard workbook** or an Excel **add-in workbook**; the latter provides certain advantages, since it is normally hidden from display and loaded automatically when Excel starts.

For example, it is straightforward to create an Excel add-in workbook that defines a Certified Distribution that draws its trial data from a Stochastic Library (see below) that is actually stored in a relational database or data warehouse. When the add-in workbook is (automatically) opened, either Excel data access functions or VBA code can be used to query the database for the trial data.

Stochastic Libraries: SIPs and SLURPs

As described by Dr. Savage in *OR/MS Today*, the simplest element of a coherent Stochastic Library is a *Stochastic Information Packet* or SIP, which is just a list of trial values for a single uncertain variable. An example might be 1,000 sample values, in a specific sequence, for the future price of oil at some point or interval of time. In a Monte Carlo simulation, trials will be drawn from the SIP ***in the order in which they were generated***.

If the SIPs for several different uncertain variables are generated as a group, in a manner that preserves the statistical dependence between them, they may be combined into a Stochastic Library Unit, Relationships Preserved, or SLURP. A SLURP is a table of trial values, where each column represents a specific uncertain variable, and each row represents a distinct trial. The SLURP's uncertain variables may be dependent in ways not measured by traditional correlation. But if a Monte Carlo simulation draws trials in the specific order given by the SLURP, this dependence will be reflected in the simulation model results.

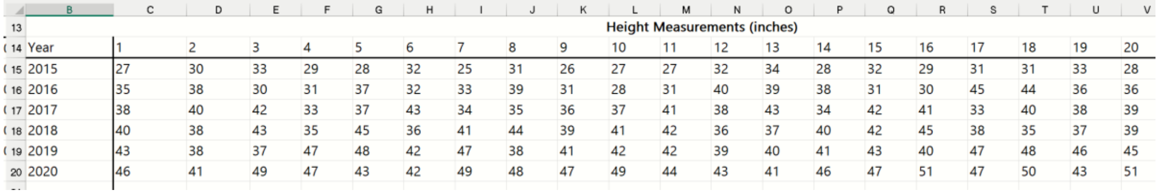
On an Excel spreadsheet, a SIP is naturally represented by a column of cell values, and a SLURP is most easily represented by a two-dimensional table of cell values. As noted above, the data for this table of cell values may be easily drawn from a relational database or multidimensional data warehouse, or it may be created and maintained directly in Excel.

Analytic Solver provides three PSI Distribution functions, **PsiSip(), PsiTSSip()** and **PsiSlurp()**, that make it easy to define distributions using SIPs and SLURPs. PsiSip() takes one argument: a cell range (normally a column) containing the trial values for one uncertain variable. PsiTSSip also takes one argument: a cell range containing values from a *time series*. PsiSlurp() takes two arguments: a two-dimensional cell range containing the SLURP data, and a column index (starting from 1) for the uncertain variable whose trials should be returned by the PsiSlurp() function. You'll want to ensure that the number of rows in the first argument range is at least as large as the number of simulation trials to be run.

PsiSip(), PsiTSSip() and PsiSlurp(), like other PSI Distribution functions, can be used directly in simulation models to return sample values from a SIP, a time series SIP or SLURP. But they are especially powerful when you name and publish a Certified Distribution based on them. Doing this is as simple as including a **PsiCertify()** function call as an argument to PsiSip(), PsiTSSip or PsiSlurp() – as described below in the section “Creating and Using Certified Distributions.”

Using the PsiTSSip Function

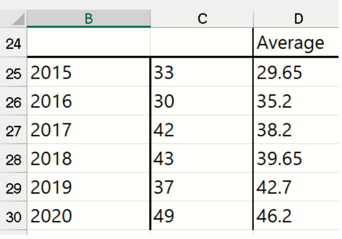
The ChildHeightsPsiTSSip(Sim).xlsx example illustrates how a PsiTSSip can be used to utilize past time series data in a simulation model. In this example, a pediatrician has collected height measurements from a sample of 20 children at various times in their development.



Height Measurements (inches)

Year	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2015	27	30	33	29	28	32	25	31	26	27	27	32	34	28	32	29	31	31	33	28
2016	35	38	30	31	37	32	33	39	31	28	31	40	39	38	31	30	45	44	36	36
2017	38	40	42	33	37	43	34	35	36	37	41	38	43	34	42	41	33	40	38	39
2018	40	38	43	35	45	36	41	44	39	41	42	36	37	40	42	45	38	35	37	39
2019	43	38	37	47	48	42	47	38	41	42	42	39	40	41	43	40	47	48	46	45
2020	46	41	49	47	43	42	49	48	47	49	44	43	41	46	47	51	47	50	43	51

Cells C15:V20 contain the height measurements of each of the 20 children at their 1, 2, 3, 4, and 5 year appointments.

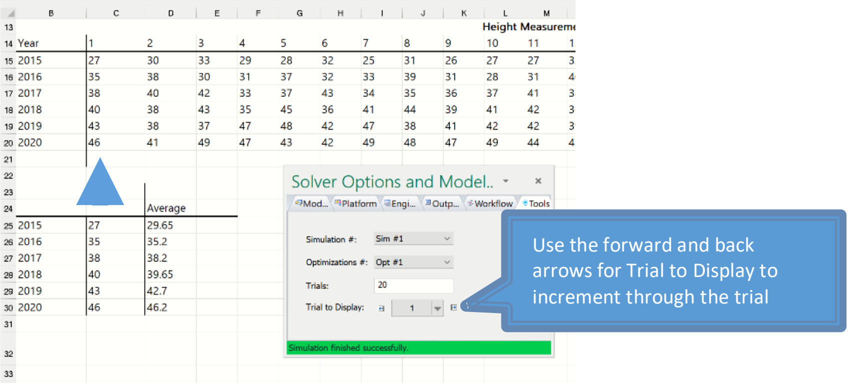


		Average
2015	33	29.65
2016	30	35.2
2017	42	38.2
2018	43	39.65
2019	37	42.7
2020	49	46.2

Cells C25:C30 contain the array-entered function, PsiTSSip(C15:V20).

To run a simulation, click the green arrow on the Solver Task Pane. After the simulation has completed, click the Tools tab and use the forward Trial to Display arrow to increment the trial number and display the TSSip values, trial by trial, in cells C25:C30.

- For the first trial, cells C25:C30 will display TSSIP values from the 1st column



Height Measureme

Year	1	2	3	4	5	6	7	8	9	10	11
2015	27	30	33	29	28	32	25	31	26	27	27
2016	35	38	30	31	37	32	33	39	31	28	31
2017	38	40	42	33	37	43	34	35	36	37	41
2018	40	38	43	35	45	36	41	44	39	41	42
2019	43	38	37	47	48	42	47	38	41	42	42
2020	46	41	49	47	43	42	49	48	47	49	44

		Average
2015	27	29.65
2016	35	35.2
2017	38	38.2
2018	40	39.65
2019	43	42.7
2020	46	46.2

Simulation #: Sim #1

Optimizations #: Opt #1

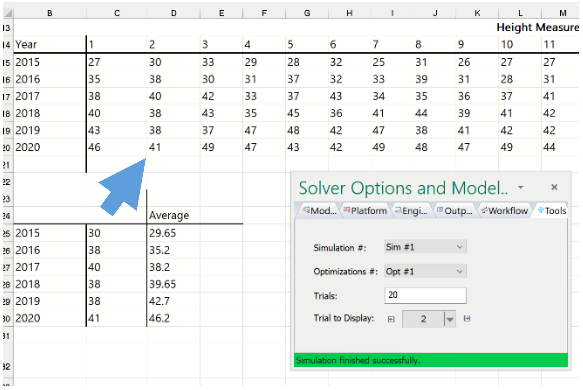
Trials: 20

Trial to Display: 1

Simulation finished successfully.

Use the forward and back arrows for Trial to Display to increment through the trial

- For the second trial, cells C25:C30 will display TSSIP values from the 2nd trial, etc.



Height Measure

Year	1	2	3	4	5	6	7	8	9	10	11
2015	27	30	33	29	28	32	25	31	26	27	27
2016	35	38	30	31	37	32	33	39	31	28	31
2017	38	40	42	33	37	43	34	35	36	37	41
2018	40	38	43	35	45	36	41	44	39	41	42
2019	43	38	37	47	48	42	47	38	41	42	42
2020	46	41	49	47	43	42	49	48	47	49	44

		Average
2015	30	29.65
2016	38	35.2
2017	40	38.2
2018	38	39.65
2019	38	42.7
2020	41	46.2

Solver Options and Model..

Simulation #: Sim #1

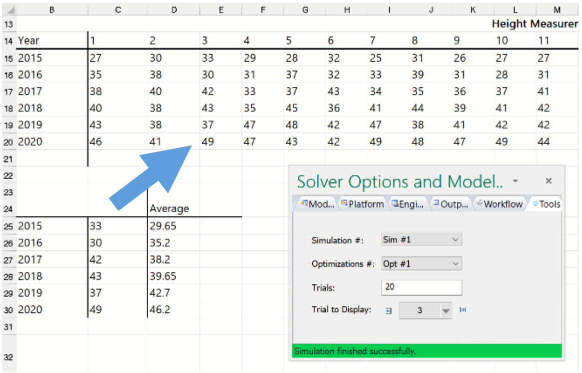
Optimizations #: Opt #1

Trials: 20

Trial to Display: 2

Simulation finished successfully.

- For the third trial, cells C25:C30 will display TSSIP values from the 3rd trial, etc.



Height Measurer

Year	1	2	3	4	5	6	7	8	9	10	11
2015	27	30	33	29	28	32	25	31	26	27	27
2016	35	38	30	31	37	32	33	39	31	28	31
2017	38	40	42	33	37	43	34	35	36	37	41
2018	40	38	43	35	45	36	41	44	39	41	42
2019	43	38	37	47	48	42	47	38	41	42	42
2020	46	41	49	47	43	42	49	48	47	49	44

		Average
2015	33	29.65
2016	30	35.2
2017	42	38.2
2018	43	39.65
2019	37	42.7
2020	49	46.2

Solver Options and Model..

Simulation #: Sim #1

Optimizations #: Opt #1

Trials: 20

Trial to Display: 3

Simulation finished successfully.

A PsiTSSip may have unlimited dimensions and or time periods (up to Excel's row/column limits).

Calculation of Psi Statistics

The PsiMean() functions in cells D25:D30 calculate the mean for each time moment (or year) across all trials.

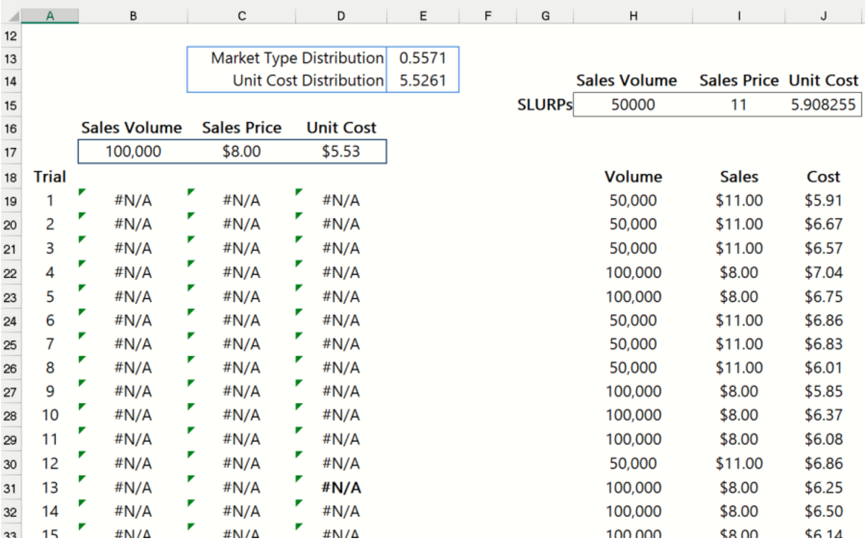
Since 20 children were in the sample, the number of trials is set to 20. If the number of trials is greater than the number of time dimensions, the TSSIP values wrap back to the 1st column. For example, if the number of trials is set to 30, the values for the first trial (cells C15:C20) would be used for trial number 21, D15:D20 for trial number 22, and so on.

The number of simulations also affects any Psi Statistics computations. For example, PsiMean(), when the number of simulations is set to 20, is simply the average of all 20 height measurements by year, i.e. PsiMean(C25) is calculated as =SUM(C15:V15)/20. However, if the number of simulations is equal to 21, Solver will wrap back to the first trial values in cells C25:C30 for the 21st trial, i.e. PsiMean(C25) is calculated as =(SUM(C15:V15) + C15)/21. For more information on PsiTSSip() see the Analytic Solver Reference Guide.

Creating Stochastic Libraries

Of course, a Stochastic Library must first be created before it can be used. In some cases, you may have a data source from which you can draw SLURP data directly. For example, if you are working with demographic data such as age, family size, income, and taxes paid, you might be able to use a representative sample directly as a SLURP. But in many cases, an expert (perhaps like you) will have to select appropriate analytic probability distributions and their parameters, determine whether and how they should be correlated, and then generate the trial data through a Monte Carlo process.

Analytic Solver is a great tool for *creating* Stochastic Libraries, as well as using them. It supports a wide range of analytic distributions, shifting and truncation of distributions, and rank order correlation of different distributions. When you run a simulation, the trial data is generated, and with the PsiData() function, you can easily save the trial data in a column on the spreadsheet – this will create a SIP. Several PsiData() functions in adjacent spreadsheet columns will create a SLURP. To see an example, open the simulation example workbook **BusinessPlanSlurp.xlsx** (normally installed into C:\ProgramData\Frontline Systems\Examples), pictured below.



	A	B	C	D	E	F	G	H	I	J
12										
13				Market Type Distribution	0.5571					
14				Unit Cost Distribution	5.5261			Sales Volume	Sales Price	Unit Cost
15							SLURPs	50000	11	5.908255
16		Sales Volume	Sales Price	Unit Cost						
17		100,000	$8.00	$5.53						
18	Trial							Volume	Sales	Cost
19	1	#N/A	#N/A	#N/A				50,000	$11.00	$5.91
20	2	#N/A	#N/A	#N/A				50,000	$11.00	$6.67
21	3	#N/A	#N/A	#N/A				50,000	$11.00	$6.57
22	4	#N/A	#N/A	#N/A				100,000	$8.00	$7.04
23	5	#N/A	#N/A	#N/A				100,000	$8.00	$6.75
24	6	#N/A	#N/A	#N/A				50,000	$11.00	$6.86
25	7	#N/A	#N/A	#N/A				50,000	$11.00	$6.83
26	8	#N/A	#N/A	#N/A				50,000	$11.00	$6.01
27	9	#N/A	#N/A	#N/A				100,000	$8.00	$5.85
28	10	#N/A	#N/A	#N/A				100,000	$8.00	$6.37
29	11	#N/A	#N/A	#N/A				100,000	$8.00	$6.08
30	12	#N/A	#N/A	#N/A				50,000	$11.00	$6.86
31	13	#N/A	#N/A	**#N/A**				100,000	$8.00	$6.25
32	14	#N/A	#N/A	#N/A				100,000	$8.00	$6.50
33	15	#N/A	#N/A	#N/A				100,000	$8.00	$6.14

In this workbook, we generate SLURP data for the BusinessPlanPsi.xlsx model (also included as an example workbook). This model has three uncertain variables: Sales in units, Price per unit, and Unit cost. In cells E13 and E14 we have **PsiUniform()** and **PsiTriangular()** functions, and in cells B17, C17 and D17 we generate trial data as we did in the BusinessPlanPsi.xlsx model. For example, B17 contains the formula **=IF(E13>0.5,100000,50000)**.

Cell B19 contains the formula, =PsiData(B17). This function will "spill" down the number of trials performed. In this case 1000 trials were performed in the simulation, so the function will "spill" down 1000 cells. Each cell will display a trial value for cell B17. The first trial value in cell B19, 2nd in B20, 3rd in B21, etc. The 1000 trial values becomes a SIP for Sales in units.

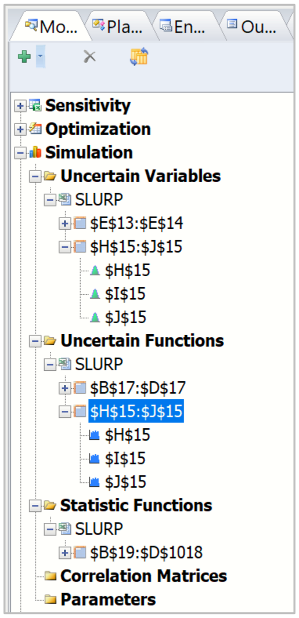
Likewise cell C19 contains the formula, PsiData(C17). This function will also "spill" down the number of trials in the simulation, creating a SIP for Sales Price.

Finally cell D19 contains the formula, =PsiData(D17).   After a simulation is performed, this function will "spill" down 1000 cells creating a SIP for Unit Cost.

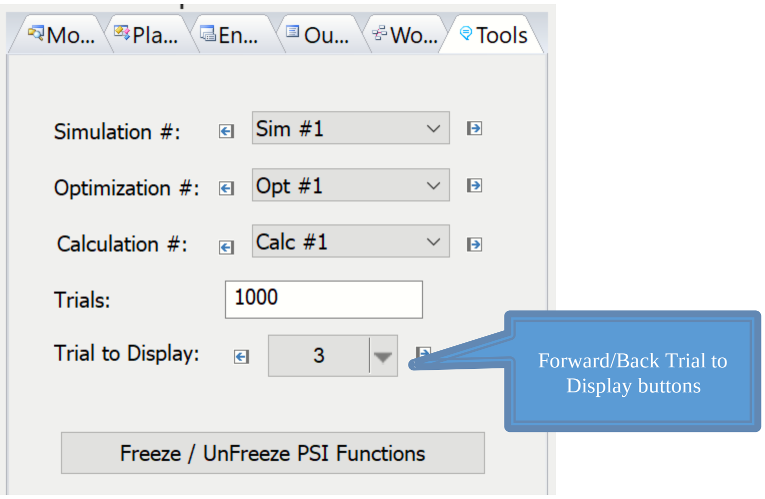
To create and save the SLURP, simply run a simulation, by pressing the Simulate button on the Analytic Solver ribbon. The SLURP data will appear starting at cells B19, C19 and D19. Save the workbook. We've generated 1,000 trials for each uncertain variable, enough to match the number of trials in our simulation.

Then select cells B19:D1018, press CTRL + C to copy, then select cell H19 and press CTRL + V to paste the contents of cells B19:D1018 to the cell range H19:J1018.

Cells H15:J15 contain three PsiSlurp functions. Click the Model tab of the Solver Task Pane and notice that these three cells are now treated as uncertain variables and, since they contain the PsiOutput() function, as uncertain functions as well.



Click the Tools tab on the Solver Task Pane. Click the forward and back buttons for Trial to Display to increment through the trial values.



Simulation #: Sim #1

Optimization #: Opt #1

Calculation #: Calc #1

Trials: 1000

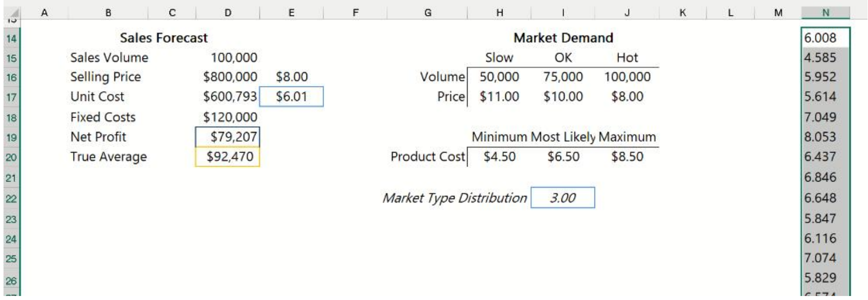
Trial to Display: 3

Forward/Back Trial to Display buttons

Using the DIST Feature

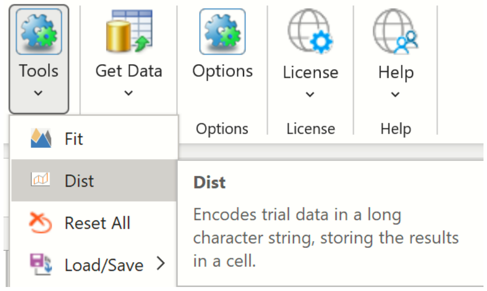
Another way to use historical data is to use the DIST feature. This tool allows you to encode your specific historical data as a distribution and then use the PsiSip function to sample that data during the simulation trials. Open the BusinessForecast(Sim).xlsx example by clicking Help – Example Models – Monte Carlo Simulation. Select cells N14:N40, then enter PsiData(E17) in array form (CTRL + ALT + Enter). Click the green arrow on the task pane to run a simulation. Immediately, cells N14:N40 populate with the values for cell E17 for the first 26 trial values, i.e. N14 = value for cell E17 for the first trial, N15 = value for cell E17 for the 2nd trial, etc.

1. Select your data (cells N14:N40) just as you did for the Fit process above.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
14		Sales Forecast							Market Demand					6.008
15		Sales Volume		100,000				Slow	OK	Hot				4.585
16		Selling Price		$800,000	$8.00		Volume	50,000	75,000	100,000				5.952
17		Unit Cost		$600,793	$6.01		Price	$11.00	$10.00	$8.00				5.614
18		Fixed Costs		$120,000										7.049
19		Net Profit		$79,207				Minimum	Most Likely	Maximum				8.053
20		True Average		$92,470			Product Cost	$4.50	$6.50	$8.50				6.437
21														6.846
22							*Market Type Distribution*		*3.00*					6.648
23														5.847
24														6.116
25														7.074
26														5.829

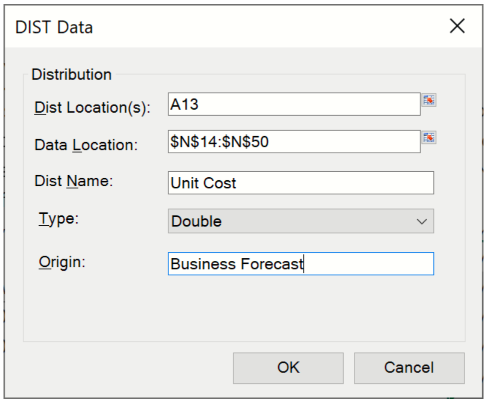
2. Click Tools -- Dist button on the Analytic Solver Desktop ribbon.



Dist

Encodes trial data in a long character string, storing the results in a cell.

3. The next dialog will allow you to choose where to place the DIST after it is generated as well as add information to it such as a name, comments on where the data came from, as well as the type of encoding.



DIST Data

Distribution

Dist Location(s): A13

Data Location: $N$14:$N$50

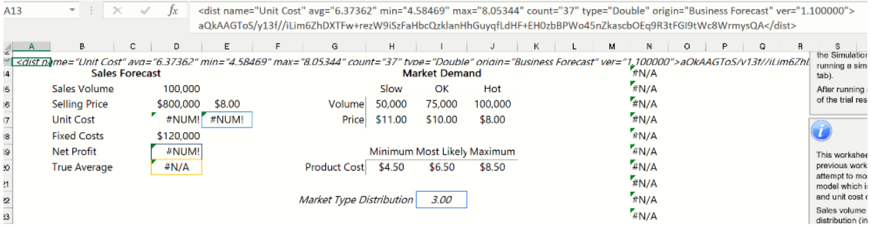
Dist Name: Unit Cost

Type: Double

Origin: Business Forecast

Tip" The default encoding choice is "Double". If you have 0-10,000 data points you can use this setting. If you have 10,001 to 20,000 data points you should use "Single". If you have more than 20,000 data points you should use the Fit feature.

4. You can see the results after you hit "OK" below.



A13

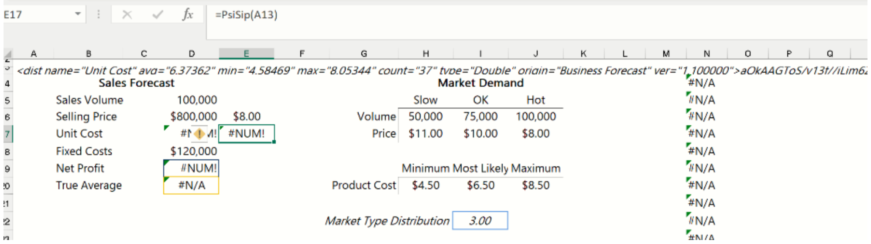
```
<dist name="Unit Cost" avg="6.37362" min="4.58469" max="8.05344" count="37" type="Double" origin="Business Forecast" ver="1.100000">
aQkAAGToS/y13f//iLim6ZhDXTFw+rezW9iSzFaHbcQzklanHhGuyqfLdHF+EH0zbBPWo45nZkascbOEq9R3tFGI9tWc8WrmysQA</dist>
```

Sales Forecast				Market Demand			#N/A
Sales Volume	100,000			Slow	OK	Hot	#N/A
Selling Price	$800,000	$8.00	Volume	50,000	75,000	100,000	#N/A
Unit Cost	#NUM!	#NUM!	Price	$11.00	$10.00	$8.00	#N/A
Fixed Costs	$120,000						#N/A
Net Profit	#NUM!			Minimum	Most Likely	Maximum	#N/A
True Average	#N/A		Product Cost	$4.50	$6.50	$8.50	#N/A
							#N/A
			Market Type Distribution	3.00			#N/A

Note: In this case we chose cell A13 to place the data. You will see the DIST in the formula bar (only part of the DIST is visible above). The more data points you have the longer the DIST string will be.

Tip: The DIST you've created can now be copied and used in any spreadsheet. The data that created it no longer needs to be kept for it to work.

5. Use the PsiSip function to add the DIST to your model.



E17

```
=PsiSip(A13)
```

Sales Forecast				Market Demand			#N/A
Sales Volume	100,000			Slow	OK	Hot	#N/A
Selling Price	$800,000	$8.00	Volume	50,000	75,000	100,000	#N/A
Unit Cost	#NUM!	#NUM!	Price	$11.00	$10.00	$8.00	#N/A
Fixed Costs	$120,000						#N/A
Net Profit	#NUM!			Minimum	Most Likely	Maximum	#N/A
True Average	#N/A		Product Cost	$4.50	$6.50	$8.50	#N/A
							#N/A
			Market Type Distribution	3.00			#N/A

To use this DIST we enter "=PsiSip(A13)" in cell E17. Now our simulation model will use the DIST we just created.

Tip: When you run a simulation using a DIST you won't see the value in the cell containing it change as it will always show the first data point in your distribution and is working correctly.

Creating and Using Certified Distributions

As described above, Analytic Solver supports the use of both analytic distributions and Stochastic Libraries as Certified Distributions. It provides a property function **PsiCertify()** that you can use to name and certify a distribution as ready for publication. It also provides a distribution function **PsiCertified()** that end user modelers can use to access the Certified Distribution via its name.

Certified Distributions may be defined in an Excel workbook different from the end user modeler's workbook. This may be a **standard workbook** or an Excel **add-in workbook**; the latter provides certain advantages, since it is normally hidden from display and may be loaded automatically when Excel starts up.

***Note*:** The workbook that defines Certified Distributions must be a 'complete model' on its own: It must **define at least one output** by using a PSI Statistics function (PsiMean(), PsiOutput(), etc.) in a formula.

Publishing Distributions with PsiCertify

To name and certify an analytic distribution, you simply include the PsiCertify() property function in the formula that defines the distribution. For example:

=PsiNormal(0.1, 1, PsiTruncate(-2,2), PsiSeed(3), PsiCertify("MyDist",0.1))

will define a Certified Distribution named “MyDist” based on a Normal distribution with mean 0.1 and standard deviation 1, truncated so that its samples lie in the interval -2 to 2, and generated using a random number seed of 3 (this overrides any random seed specified in the end user’s model). To use this distribution, the end user modeler enters **=PsiCertified(“MyDist”)** in his or her own workbook. When no simulation has been performed, the cell containing the PsiCertified() call will display the default value 0.1.

To name and certify a distribution based on trial data in a Stochastic Library, you also use the PsiCertify() property function in the formula defining the distribution. For example:

=PsiSip(A1:A1000, PsiCertify("MyDist",100))

will define a Certified Distribution named “MyDist” whose trials are drawn sequentially from the range A1:A1000, with default value 100 to be displayed when no simulation has been performed.

When (as is often the case) the SIPs for several different uncertain variables are generated as a group, in a manner that preserves the statistical dependence between them, they should be combined into a SLURP, i.e. a table of trial values where each column represents an uncertain variable, and each row represents a Monte Carlo trial. Analytic Solver allows you to place a character string name for each column (uncertain variable) in the first row of the SLURP data. You can then name and certify the entire SLURP, for example:

=PsiSlurp(A10:C1010, PsiCertify("MyPlan"))

where range A10:C10 contains labels such as “Sales”, “Price” and “UnitCost”, and A11:C1010 contains the SLURP trial data.

Now the end user modeler can write **=PsiCertified(“MyPlan”, “Sales”)** to define an uncertain variable in a model whose samples will be sales figures.

The PsiCertify() function allows you to define more properties of the Certified Distribution than just its name. It is good practice to document the name, description, author, history of creation, and other relevant information about a Certified Distribution using PsiCertify(). The full set of arguments is:

=PsiCertify(***name, default value, short description, full description, version, author, copyright, trademark, history*****)**

Using Distributions with PsiCertified

To use a Certified Distribution, the end user modeler simply enters:

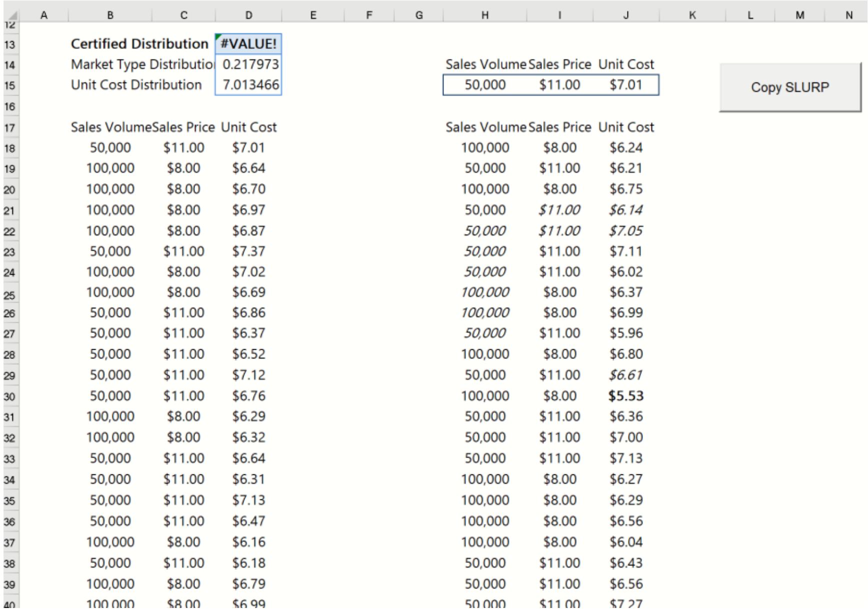
=PsiCertified(*name*) or **=PsiCertified(*slurp,sip*)**

in his or her own workbook, where *name* or *slurp* and *sip* are quoted strings. The user needs no knowledge of the form or parameters of the distribution, or of the name or folder path of the workbook that defines the distribution.

Analytic Solver scans the open workbooks (including regular workbooks and add-in workbooks) for distributions that include the **PsiCertify()** function, and adds the names found in PsiCertify() calls to the Certified dropdown gallery.

Certified Distributions may also be used with Risk Solver Engine, when the Analytic Solver GUI is not available. The PSI Interpreter matches calls to **=PsiCertified()** to the known Certified Distributions. If a named distribution is not found, the PsiCertified() call will return the error value #NAME?

To see an example of publishing and using Certified Distributions, open the simulation example workbook **BusinessForecastingCertified(Sim).xlsx** (normally installed into C:\ProgramData\Frontline Systems\Examples), shown on the next page. This example uses a regular workbook, rather than an add-in. Look for "Creating and Using a Certified Distribution" in the Cloud app. Click the **Certified Distributions** tab.

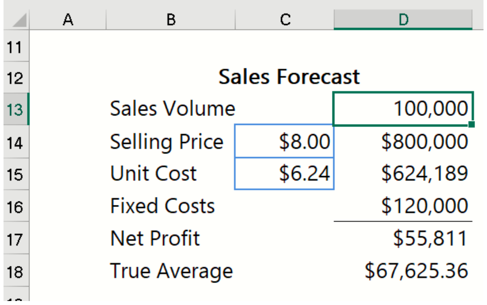


	A	B	C	D	E	F	G	H	I	J
12										
13		Certified Distribution		#VALUE!						
14		Market Type Distributio		0.217973				Sales Volume	Sales Price	Unit Cost
15		Unit Cost Distribution		7.013466				50,000	$11.00	$7.01
16										
17		Sales Volume	Sales Price	Unit Cost				Sales Volume	Sales Price	Unit Cost
18		50,000	$11.00	$7.01				100,000	$8.00	$6.24
19		100,000	$8.00	$6.64				50,000	$11.00	$6.21
20		100,000	$8.00	$6.70				100,000	$8.00	$6.75
21		100,000	$8.00	$6.97				50,000	*$11.00*	*$6.14*
22		100,000	$8.00	$6.87				*50,000*	*$11.00*	*$7.05*
23		50,000	$11.00	$7.37				*50,000*	$11.00	$7.11
24		100,000	$8.00	$7.02				*50,000*	$11.00	$6.02
25		100,000	$8.00	$6.69				*100,000*	$8.00	$6.37
26		50,000	$11.00	$6.86				*100,000*	$8.00	$6.99
27		50,000	$11.00	$6.37				*50,000*	$11.00	$5.96
28		50,000	$11.00	$6.52				100,000	$8.00	$6.80
29		50,000	$11.00	$7.12				50,000	$11.00	*$6.61*
30		50,000	$11.00	$6.76				100,000	$8.00	**$5.53**
31		100,000	$8.00	$6.29				50,000	$11.00	$6.36
32		100,000	$8.00	$6.32				50,000	$11.00	$7.00
33		50,000	$11.00	$6.64				50,000	$11.00	$7.13
34		50,000	$11.00	$6.31				100,000	$8.00	$6.27
35		50,000	$11.00	$7.13				100,000	$8.00	$6.29
36		50,000	$11.00	$6.47				100,000	$8.00	$6.56
37		100,000	$8.00	$6.16				100,000	$8.00	$6.04
38		50,000	$11.00	$6.18				50,000	$11.00	$6.43
39		100,000	$8.00	$6.79				50,000	$11.00	$6.56
40		100,000	$8.00	$6.99				50,000	$11.00	$7.27

This worksheet contains the function call =PsiSlurp(H17:J1017,PsiCertify("ForecastDist")) in cell D13. This cell (D13) displays #VALUE! since there are labels rather than numbers in the first row of the SLURP range, but this doesn't affect use of the Certified Distribution.

To use the certified distribution, simply insert a function call =PsiCertified("ForecastDist") into the currently selected cell. That's all there is to it.

To see how this set of Certified Distributions can be used, click the Forecast using Certified Dist tab, shown on the next page.



	A	B	C	D
11				
12			Sales Forecast	
13		Sales Volume		100,000
14		Selling Price	$8.00	$800,000
15		Unit Cost	$6.24	$624,189
16		Fixed Costs		$120,000
17		Net Profit		$55,811
18		True Average		$67,625.36

This model uses **=PsiCertified ("ForecastDist", "Sales Volume")** in cell D13, **=PsiCertified ("ForecastDist", "Sales Price")** in C14, and **=PsiCertified ("ForecastDist", "Unit Cost")** in C15. It no longer requires an Excel 'external reference' to a specific folder path, workbook filename, or cell range. In fact, there might not *exist* a single cell range containing fixed trial data – the trials might be loaded from an external database, or generated dynamically from one or more analytic distributions – as illustrated below.

Packaging Analytic Distributions

As described above, you can use PsiCertify() to name and publish an analytic distribution, with a specific form and parameters and a pre-specified random number seed, as a Certified Distribution. Further, you can “package” several analytic distributions that are correlated through a rank correlation matrix in the form of Certified Distributions. You can place these definitions in a standard workbook, or in an add-in workbook that is automatically loaded when Excel starts, hiding a good deal of complexity from end users who simply need to use these Certified Distributions.

Mastering Stochastic Optimization Concepts

Introduction

Analytic Solver offers you one unified framework for modeling and solving optimization problems, with or without uncertainty. Models may include ‘normal’ decision variables, constraints and an objective, plus uncertainties, recourse decision variables, chance constraints, and recourse constraints. This chapter seeks to explain this framework, starting from the basics of Solver models. It assumes only a limited knowledge of optimization – but if you’ve first read the chapter “Mastering Conventional Optimization Concepts,” you’ll gain a much deeper understanding from reading this chapter.

Our goal is to find a ***solution*** – values for the ***decision variables*** in our model – that satisfies the ***constraints*** and that maximizes or minimizes the ***objective***. In computing the objective and constraints, we use ***parameters*** of the model – for example quantities, distances, shipping costs, interest rates, etc.

Certain and Uncertain Parameters

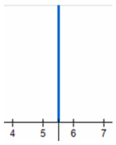
In conventional optimization, all the parameters are *assumed* to be numbers known with certainty. If estimation of a parameter involves uncertainty or noise, an average or other ‘nominal’ value is used. This is a simplification and *idealization* of the real world situation: In almost all business and engineering problems, there is at least some uncertainty in the values of the parameters.

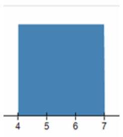
In some models, the uncertainty or variation in the parameters may be small enough to be ignored. But in many models, it is large enough to be of concern. And in some models, the uncertainty is so large that it is a major feature of the problem, and much effort is (or should be) invested in modeling the uncertainty.

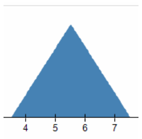
In the discussion to follow, by **uncertain parameter** we mean a parameter of your optimization model, such as an objective coefficient or constraint right hand side, that is subject to uncertainty. Such parameters might be *computed* from a “primitive uncertainty,” such as a future interest rate or rainfall amount; we will call this an **uncertain variable**, and in Analytic Solver, we’ll use the Uncertain Variable dialog to define and quantify this primitive uncertainty.

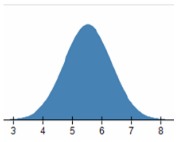
Realizations of Uncertain Parameters

Where a *certain* parameter has a *single*, known value, we can think of an *uncertain* parameter as having *many possible values.* We call each such value a *realization* of the uncertainty. Below, we’ve pictured one certain and three uncertain parameters.









An uncertain parameter might have an infinite number of possible realizations, but – as discussed near the end of this chapter – normally we can work with a finite ***sample*** of possible realizations. We just don't know *which one* of the sample values is the 'true' value, or closest to this true value.

Decision-Dependent Uncertainties

In many real-world problems, the uncertainties being modeled are ***independent* of the decision variables**. For example, in a crop planning model where the acres to plant of different crops are decision variables, and the rainfall in a future period is an uncertainty, the uncertainty is almost surely independent of the decisions.

In other real-world problems, the uncertain parameters being modeled are ***dependent* on the decision variables** – they change if the decisions change. For example, in a market response model that includes competitor actions in a future period, where your product prices are decision variables, and your competitors' product prices are uncertainties, it is quite likely that the uncertainties will depend on the decisions.

In portfolio models where one is investing in publicly traded, large capitalization stocks, a decision to purchase some number of shares is unlikely to affect the uncertain future returns on those stocks. On the other hand, a large institution investing in thinly traded small-capitalization stocks may find that decisions to purchase a large number of shares may move the market price and impact the uncertain future returns. So both types of problems arise in practice. As we'll see later, the presence of ***decision-dependent uncertainties*** has a **huge impact** on how, and even whether, our model can be solved.

Resolving Uncertainty and Recourse Decisions

In some real-world problems, we are uncertain about the value of a parameter today, and **we will remain uncertain about its value in the future**; we have to make decisions in the presence of this uncertainty. This often arises in engineering design models, where there is uncertainty about the exact strength of a steel girder or the exact resistance or capacitance of an electronic component. The uncertainty may be *irreducible* – perhaps inherent in nature – or it may be *reducible* through better data collection, measurement or calibration – but these steps may be impractical for us.

In other real-world problems, we are uncertain about the value of a parameter today, but **we will know its value with certainty at some point in the future**. This often arises in marketing and finance models, where a competitor's price or next year's commodity prices or interest rates are uncertain today, but will become known at some future date. We say that the uncertainty will be ***resolved*** at the future date; its *many possible values* will be reduced to a *single* value, and the uncertain parameter will become certain.

If we are dealing with uncertainty that will not be resolved, we must find values of the decision variables that yield feasible (or *nearly* feasible) solutions for many possible realizations of the uncertainty. On the other hand, if we are dealing with uncertainty that *will* be resolved at a future date, we can ask what decisions we must make 'here and now,' *before* the uncertainty becomes known, and what decisions we can make on a 'wait and see' basis, *after* the uncertainty is known. This leads to the concept of ***recourse decisions***, described below – and it has a **huge impact** on how our model should be formulated and solved.

Uncertainty and Conventional Optimization

The impact of uncertainty is *magnified* by conventional optimization. By its nature, optimization drives the decision variables to values that *just satisfy* certain constraints, while maximizing or minimizing the objective function. In doing so, optimization exploits any 'noise' or error, as well as the 'true value,' captured in the fixed numbers used in the model for uncertain parameters.

This effect is very evident in investment portfolio optimization, where the parameters of the model are estimated returns, variances, and covariances between securities. Practitioners often describe portfolio allocations from an optimizer as "unrealistic," and such allocations often do not perform well in practice. When the actual security returns, variances and covariances in future periods vary from the parameters used in the optimization, even by small amounts, the 'optimized' portfolio return often suffers by large amounts.

As shown by Ben-Tal and Nemirovski in a 2000 study of the NETLIB linear programming problems, "In real-world applications of linear programming, one cannot ignore the possibility that a small uncertainty in the data can make the usual optimal solution completely meaningless from practical viewpoint." Summarizing the effects of conventional investment portfolio optimization, Michaud wrote in a 1998 book "The result is that optimized portfolios are 'error maximized' and often have little, if any, investment value." ***Conventional optimization is not enough!***

Elements of Solver Models

We'll now describe all the elements of a Solver model with uncertainty. Again our goal is to find a ***solution*** – values for the ***decision variables*** in our model – that satisfies the ***constraints*** and that maximizes or minimizes the ***objective***. In computing the objective and constraints, we may use many ***parameters*** – some of which are computed from primitive ***uncertain variables***. In Excel, the decision variables and parameters are represented by worksheet cells; the objective and constraints are computed by formulas in other worksheet cells.

Uncertain Variables

A *certain* parameter is easy to model, in a worksheet cell containing a number. A primitive uncertainty requires a way to specify the range and 'shape' of the values it can assume. In Analytic Solver, we model primitive uncertainties as **uncertain variables** in worksheet cells that contain special add-in functions. For example:

=PsiUniform(0,1) – a uniform distribution ranging from 0 to 1

=PsiNormal(2,1.5) – a Normal distribution with mean 2, standard deviation 1.5

=PsiBeta(A1,B1) – a Beta distribution with shape parameters A1 and B1

=PsiSip(A1:A1000) – a 'Stochastic Information Packet' containing a range of user-supplied sample values, or realizations, for this uncertainty

Any of the 40-odd PSI Distribution functions that you can use for Monte Carlo simulation may also be used as uncertain variables for stochastic optimization.

You can think of the cell for an uncertain variable, containing a formula such as =PsiUniform(0,1), as holding an *array* of sample values, each one selected at random from the range 0 to 1 inclusive. Each value is one possible *realization* of the uncertainty. Analytic Solver can generate such an array of sample values automatically, using Monte Carlo simulation 'behind the scenes'.

Looking ahead to the section "Functions of the Variables and Uncertainties," note that any formula cell that *depends* on an uncertain variable cell must also compute an *array* of sample values. Each value corresponds to one possible *realization* of all the uncertain variables on which the formula depends.

In the example **PsiBeta(A1,B1)**, it is important to know whether A1 or B1 is dependent on the decision variables. A model that includes such a ***decision-dependent uncertainty*** is *much* harder to solve, and will require the methods of simulation optimization. Analytic Solver can automatically diagnose your model and determine whether any uncertain variables depend – possibly through a chain of formula cells – on the decision variables.

Decision Variables

Conventional optimization deals with only one type of decision variable, which represents a decision that must be made 'here and now,' irrespective of any uncertainty in the model. We call this a normal or first-stage variable.

If we are dealing with uncertainty that will be resolved in the future, then at some point the *array* of sample values for the uncertainty is effectively replaced by a *single* value, the *realization* of the uncertainty as it actually occurs.

If the situation we are modeling allows us to make certain decisions *after* the uncertainty becomes known, on a 'wait and see' basis, we can model these decisions with recourse variables, also called second-stage variables. (At the 'second stage,' the uncertainty has become known.)

Normal or First-Stage Variables

A **normal variable** for a 'here and now' decision is represented by a worksheet cell – say A1 – containing a number; the Solver will replace this number with an optimal value, when the problem is solved. (The term 'normal' here is unrelated to a Normal distribution, which is one way to define an uncertain variable.)

Recourse or Second-Stage Variables

A **recourse variable** for a 'wait and see' decision is also represented by a worksheet cell – say A2 – containing a number. But at the time the optimization is performed, a single recourse variable effectively represents an *array* of solution values, one for each *realization* of the uncertainties in the model.

Functions of the Variables and Uncertainties

Any formula that participates (in the objective and constraints, see below) in your model will normally depend on the decision variables and the parameters.

As a simple example, the objective of the Product Mix model EXAMPLE1in the **Examples.xlsx** workbook, is SUMPRODUCT(D17:F17,D9:F9) which can also be written as D17*D9 + E17*E9 + F17*F9. Here, D17:F17 are parameters (the selling prices of the three products), and D9:F9 are decision variables (the amounts of each product to build). In this model, the parameters are assumed to be *certain*.

An objective or constraint might be computed, from the parameters and the decision variables, through a chain of several formula cells. For example, in the Gas Company Chance and Recourse models in **StochasticExamples.xlsx**, illustrated in the chapter “Examples: Stochastic Optimization,” the constraint at cell C23 is D15+D18-C8 >= 0, where D15 and D18 are decision variables (amounts of gas to purchase), C8 = 100+80*D5 (estimated demand next year), and D5 is an uncertain variable in the model.

Model Uncertainty and Optimization Methods

Recall that an uncertain variable effectively holds an *array* of sample values. For example, D5 =PsiUniform(0,1) represents an array of *realizations* of the uncertainty, each one selected at random from the range 0 to 1. If your objective or constraint (say C8 = 100+80*D5) depends on D5, it must also compute an *array* of sample values. Each value corresponds to one possible *realization* of all the uncertainties on which the formula depends.

Traditional Solver Engines cannot handle an objective or constraints that depend on uncertainty, and hence represent an array of sample values. To solve a problem with such an objective and constraints, we must choose one of three possible approaches:

1. Use the ‘Stochastic Decomposition Engine’ that *can* handle constraints that depend on uncertainty. *Note: This engine is not supported in Analytic Solver Cloud.*
2. *Transform* a model whose objective and/or constraints depend on uncertainty into a ‘deterministic equivalent’ or ‘robust counterpart’ model that may be much larger, but whose objective and constraints no longer depend on uncertainty. Then we can solve the transformed model using a traditional Solver Engine, and translate the results into a solution for the original model. To be transformed successfully, the original model must have only uncertainties that are *independent* of the decisions, and have an objective and constraints that are *linear* (or certain other) functions of the decision variables.
3. Ensure that all ‘top-level’ formula cells computing the objective and constraints represent only *single* values – statistical summaries of the arrays of sample values. We can do this manually, with formulas in top-level cells that call PSI Statistics functions such as PsiMean(), PsiVariance() or PsiPercentile() – or Analytic Solver can do this automatically and implicitly, when we define our objective and constraints as described in the next two sections. This choice is flexible, but it requires that we use simulation optimization – the most general, but also the slowest and least scalable optimization method.

In all these approaches, we must specify what it *means* to maximize or minimize an objective, or to satisfy a constraint that depends on uncertainty – implicitly or explicitly reducing an array of sample values to a single value. The next two sections explain how we can do this.

The Objective Function

The quantity you want to maximize or minimize is called the *objective function*. In Excel, the objective is calculated by a formula cell – say A1 – listed in the Task Pane Model tab or the Solver Parameters dialog, in the outline list box under **Objective**. This could be a calculated value for projected profits (to be maximized), or costs, risk, or error values (to be minimized).

If the objective function depends on uncertainties, we must specify how we want to ‘optimize’ this function. The most common practice is to maximize or minimize the *expected value* (informally, the mean value) of the objective, over all realizations of the uncertainties. In Analytic Solver, you do this by selecting the objective in the Task Pane Model tab, and in the lower Properties area, selecting **Expected** from the Type dropdown list in the Stochastic group.

Instead of maximizing or minimizing the *expected value* of a function of the decision variables and uncertainties, you can maximize or minimize a *measure of the uncertainty* in a function. Analytic Solver converts the objective to the form max t or min t, where t is a new variable, and the model includes a *chance constraint* A1 >= t or A1 <= t with the measure of uncertainty that you specify.

Implicit and Explicit Forms for the Objective

When you select Expected from the Type dropdown list in the Stochastic group, Analytic Solver treats the objective cell as implicitly containing **E**[*objective*], or using sample realizations of the uncertainty PsiMean(*objective*). To make this *explicit* on the worksheet, you can direct Analytic Solver to automatically create a **PsiObj()** function call with an "Expected" argument – using either the “Use Psi Functions to Define Model” option, or the Save Model button. You can also manually enter a **PsiMean()** function call in the objective cell. Similar options apply to the constraints, as discussed below.

Constraints: Normal, Chance, Recourse

Constraints are relations such as $f(x_1 , x_2 \ldots, ,x_n) <= b$, where $x_1 , x_2 \ldots, ,x_n$ are decision variables. In Excel, each x_i corresponds to a cell, $f(x_1 , x_2 \ldots, ,x_n)$ is computed by a formula in another cell (say A1), the constant b is in another cell (say B1), and we enter **A1 <= B1** via the Ribbon or Task Pane. A constraint is *satisfied* when the relation it specifies is true within a small tolerance.

When your model includes uncertainty, we must examine how each constraint depends on the uncertainties and the decision variables:

- If a constraint depends only on *certain* parameters and *normal* decision variables, it is ‘deterministic’ and is handled in the usual way by the Solver. We call this a ***normal*** **constraint**.
- If a constraint depends on *uncertain variable*s and *normal* decision variables, we must specify what it *means* for the constraint to be satisfied. There are many possible *realizations* for the uncertain variables, but only single values for the decision variables. The Solver must find values for the decision variables that cause the constraint to be satisfied for *all*, or perhaps *most* but not all, realizations of the uncertainties. We call this a ***chance*** **constraint**. For example, we might specify that the constraint must be satisfied 95% or 99% of the time; it can be violated 5% or 1% of the time. For 95%, we denote such a constraint as **VaR** $_{0.95}$ **A1 <= B1**. But this form may not be your best choice – alternatives called CVaR and USet are discussed in the section “More on Chance Constraints.”
- If a constraint depends on *uncertain variable*s and *recourse* decision variables, then the Solver will find an *array* of values for each of the recourse variables, corresponding to the *realizations* of the uncertain variables. Recourse decisions give the Solver flexibility to satisfy constraints that involve uncertainty; but in effect, each such constraint has many *realizations* – one for each realization of the uncertainties. We call this a ***recourse*** **constraint**.
- A constraint may also depend on *recourse* decision variables, and possibly normal decision variables, but not depend on any uncertain variables. This is also a ***recourse*** **constraint**, with many *realizations*. The Solver must find

values for the recourse variables that satisfy all the constraints where they appear – some with uncertainties, and some without.

Implicit and Explicit Forms for Constraints

When you select VaR from the dropdown list in the Constraint dialog and enter a Chance of 0.95, Analytic Solver treats the constraint cell as implicitly containing $\textbf{VaR}_{0.95}$ [*constraint*], or using sample realizations of the uncertainty PsiPercentile(*constraint*, 0.95) where *constraint* is the left hand side. To make this *explicit* on the worksheet, you can direct Analytic Solver to automatically create a **PsiCon()** function call with a "VaR" argument – using either the "Use Psi Functions to Define Model" option, or the Save Model button. You can also manually enter a **PsiPercentile()** function in the constraint left hand side cell. For more information on the PsiObj() and PsiCon() functions, see the chapter "Psi Function Reference" within Frontline Solvers Reference Guide.

Multiple Uncertainties May Offset Each Other

What happens when a constraint depends on *several* different uncertainties? Is such a constraint harder or easier to satisfy than a constraint that depends on just one uncertainty?

In the simplest case, suppose we have a *linear* constraint, with coefficients a_i and decision variables x_i:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq b$$

Suppose that each coefficient a_i is uncertain (and independent of all the others), with sample values drawn randomly from PsiUniform (a_i – 0.5, a_i + 0.5).

The average or *nominal* value of each coefficient is a_i. The 'worst' that can happen is that a sample is drawn where *every* coefficient is a_i + 0.5 – this makes the left hand side (LHS) as large as possible, so it is very likely to violate the condition LHS $\leq b$. *But this case is very unlikely to occur*.

In *most* realizations of the uncertainties, some coefficients (randomly drawn from the range a_i – 0.5 to a_i + 0.5) will be less than a_i, and some will be greater than a_i. The more uncertainties are involved, the greater the chance that some of them will draw samples less than a_i.

If we use a *chance constraint* to specify that the relation must be satisfied (say) 95% or 99% of the time, we actually have a *better* chance of satisfying this constraint when it depends on *many* uncertainties than when it depends on just one – as long as the uncertainties are independent, or at least not highly correlated with each other.

Solutions: Feasible, Optimal, Well-Hedged

A solution (set of values for the normal and recourse variables) for which all of the constraints in the Solver model are satisfied is called a *feasible solution.* For constraints that depend on uncertainties, this means that chance and recourse constraints meet their criteria for satisfaction over all the *realizations* of the uncertainties considered in the optimization.

An *optimal solution* is a feasible solution where the objective function reaches its maximum (or minimum) value – for example, the most profit or the least cost. When the objective function depends on uncertainties, this normally means that the *expected value* of the objective reaches a maximum (minimum) – but you can also use max t or min t with a chance constraint A1 >= t or A1 <= t, as described earlier.

The optimal solution to a problem with uncertainty (the ‘stochastic solution’) will never have an objective value that is better than one could obtain ‘with 20/20 hindsight,’ by solving a conventional optimization problem where the uncertainties are known. The difference between the objective value obtained ‘in hindsight’ and the optimal value obtained ‘in foresight’ is called the *expected value of perfect information* (EVPI).

The stochastic solution will often be (quite) different from the optimal solution to a problem where all uncertain variables are replaced with their nominal or average values. The latter optimization will exploit the ‘point values’ of the parameters, with no consideration for their variability; this will often yield a better objective value for the nominal problem, but the solution may not be feasible, let alone optimal, for many different realizations of the uncertain variables. Compared to the ‘nominal’ solution, the stochastic solution is more robust or ‘well-hedged’ against possible variations in the uncertain variables.

Solving for Recourse Variables

If we are dealing with uncertainty that *will* be resolved in the future, then at that future date, each uncertain variable will be replaced by a single value, the *realization* of the uncertainty that actually occurs. In Analytic Solver, you can enter these single values for uncertain variables in the Lock value field in the Uncertain Variable dialog, then choose to *solve for* single values of the *recourse* variables, given known single values for both the normal variables (determined by an earlier optimization) and for the uncertainties.

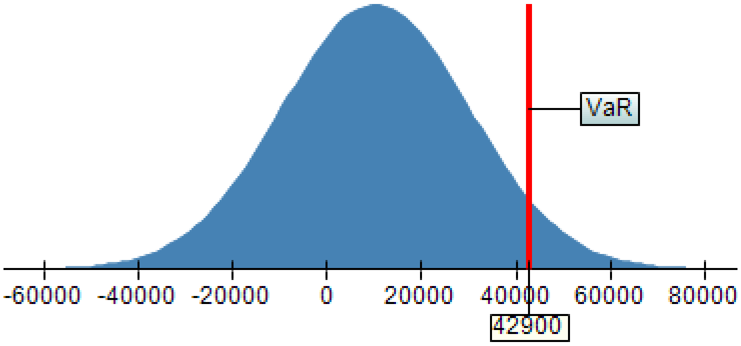
When Analytic Solver solves a problem by transforming the model, using the Deterministic Equivalent or the Robust Counterpart transformation, it computes an *array* of values for the recourse variables, one for each *realization* of the uncertainties considered during the optimization. Using the left and right Trial # arrows on the Ribbon, you can display on the worksheet the different sample values of the uncertain variable cells, and the corresponding values of the recourse variable cells – the ‘wait and see’ decisions in many possible scenarios. Now, the *actual* values of the uncertainties, once they become known, might be different from any one of these scenarios – so it is still useful to solve for single values of the recourse variables as described above.

More on Chance Constraints

As explained above, if a constraint depends on *uncertain variable*s and *normal* decision variables, we can seek solution values for the variables that cause the constraints to be satisfied for *all*, or perhaps *most* but not all, realizations of the uncertainties. If we insist that the constraints are satisfied for *all* realizations, we may not be able to *find* values for the decision variables that meet this requirement – and if we do, we will very likely ‘pay for this’ via worse values for the objective function.

Instead, we can seek solution values for the variables that cause the constraints to be satisfied for *most*, but not necessarily all, realizations of the uncertainties. We might specify that the constraint must be satisfied (it must not exceed a given limit) 95% or 99% of the time; it can be violated 5% or 1% of the time.

This is depicted in the chart below, where 95% of the area under the curve is to the left of the bar (i.e. the constraint right hand side value), and 5% is to its right.



This is one form of a *chance constraint*; the criterion that it must be satisfied for all realizations of the uncertainties up to a given percentile (say 95%) makes it a *VaR* (Value at Risk) constraint. We write this constraint as **VaR $_{0.95}$ A1 <= B1**. Analytic Solver supports two other criteria besides *VaR* that may be better choices for many models.

A chance constraint includes:

- A *left hand side* that depends on decision variables and uncertainties.
- A *relation* that must be either <= or >=. (A *chance* constraint can't be an equality. Note however that a *recourse* constraint *can* be an equality.)
- A *right hand side* that should be constant in the problem (if the RHS depends on decision variables or uncertainties, Analytic Solver converts LHS <= RHS into the form LHS – RHS <= 0).
- A *criterion* that may be VaR (Value at Risk), CVaR (Conditional Value at Risk), or USet (Uncertainty Set). These criteria are discussed below.
- A *measure* that may be a percentile 0.01 – 0.99 for VaR or CVaR, or a 'budget of uncertainty' (any positive value) for USet.

Value at Risk Measure

Chance constraints defined by a percentile or VaR (Value at Risk) measure have been used since the early 1960s. Such constraints offer a good deal of modeling flexibility, and they are easy to understand in terms of the probability that the constraint will be satisfied. Value at Risk is used in the banking and securities industries, and its use is mandated by the international Basel II accords. But chance constraints in this form have several drawbacks:

- A VaR constraint with probability 95% requires only that the constraint be satisfied – not violated – 95% of the time; it **says nothing about the magnitude of the violation** that may occur the other 5% of the time. For example, a portfolio of securities that is VaR-constrained to lose no more than $100,000 95% of the time could still lose $1 million+ at other times.
- As a measure of risk, the VaR criterion is **not subadditive**, a property expected of any 'coherent risk measure.' For example, if two portfolios A and B are VaR-constrained to not lose money 95% of the time, it is reasonable to expect that a combined portfolio A+B should have a *95% or better* chance of not losing money – but this is *not* guaranteed by the two portfolio VaR constraints.
- A VaR constraint is **not necessarily convex**; hence, using such a constraint in an otherwise convex model (for example, any linear programming or convex quadratic model) will radically affect its 'solvability' – it means that

a globally optimal solution cannot be guaranteed, and solution time may rise exponentially with model size.

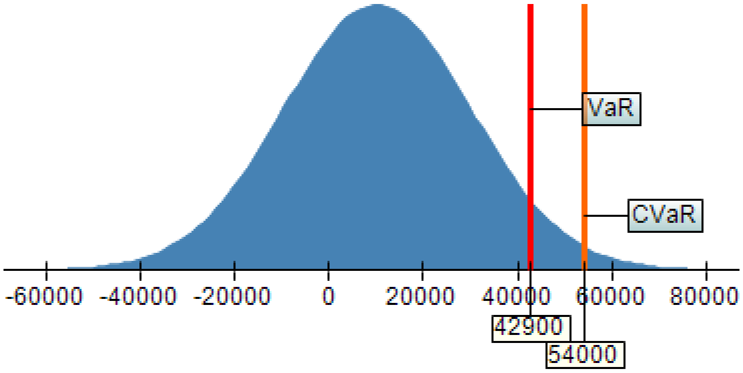
Further, when Analytic Solver uses robust optimization methods to automatically transform a model with VaR constraints into a larger but deterministic 'robust counterpart' model, it first approximates the non-convex VaR constraint with a convex CVaR constraint, and then transforms the CVaR constraint. Since CVaR is always **more conservative** than VaR as a risk measure, the robust counterpart solution will 'pay a price' in conservativeness, with a worse objective value. Users are often better off using CVaR directly.

Note: **$\text{VaR}_{0.95}$ A1 <= B1** specifies that the 95th percentile of the realizations of A1 must be less than or equal to B1. Since VaR only describes the left side of the graph, not the right or "tail", **$\text{VaR}_{0.95}$ A1 >= B1** specifies that the *5th* percentile of the realizations of A1 must be less than or equal to B1.

Conditional Value at Risk Measure

To deal with the problems of Value at Risk cited above, an alternative risk measure called *Conditional Value at Risk* or CVaR (also called *Expected Tail Loss* or ETL) was developed in the late 1990s. **$\text{VaR}_{0.95}$ A1 <= B1** specifies that the 95th percentile of the realizations of A1 must be less than or equal to B1; realizations beyond the 95th percentile may be greater than B1 by any amount. In contrast, **$\text{CVaR}_{0.95}$ A1 <= B1** specifies that the *expected value* of ***all*** the realizations of A1 up to the 95th percentile must be less than or equal to B1.

Below is a chart that compares VaR and CVaR. VaR is the value (42,900) that lies at the 95th percentile of the realizations of the constraint left hand side; 5% of the realizations are greater than 42,900 and lie in the graph to the right of this point. CVaR (54,000) is the *expected* value (i.e. the mean or average value) of ***all*** the realizations that lie in the 'tail' to the right of the VaR (Note that, if **$\text{CVaR}_{0.05}$ A1 <= B1** is satisfied for some B1, then **$\text{VaR}_{0.05}$ A1 <= B1** is also (more than) satisfied.



As a risk measure, Conditional Value at Risk has several advantages over VaR:

- Unlike VaR, a CVaR constraint at 95% places a **bound on the average magnitude of the violations** that may occur 95% of the time.
- CVaR is a 'coherent risk measure.' It is **subadditive**, so if two portfolios A and B are CVaR-constrained to not lose money 95% of the time, then a combined portfolio A+B has the *same or better* chance of not losing money.
- A CVaR constraint is **always convex**. Models consisting of all convex functions can be solved to global optimality, and solved to very large size using modern interior point optimization methods.

Uncertainty Set Measure

Analytic Solver supports a third criterion for uncertainty in a chance constraint, which reflects the approach taken in most of the literature on *robust optimization* methods. This criterion, called *USet* (for uncertainty set), is applicable only to *linear* constraints, with coefficients a_i and variables x_i:

$a_1x_1 + a_2x_2 + ... + a_nx_n \leq b$

where some or all of the coefficients a_i may depend on the uncertainties. It is useful to think of the vector $[a_1\ a_2\ ...\ a_n]$ as having a *nominal* or expected value, and a *variation* from this value for each realization of the uncertainties.

A constraint of the form **USet$_\Omega$ A1 <= B1**, where $a_1x_1 + a_2x_2 + ... + a_nx_n$ is in A1, and *b* is in B1, specifies that A1 <= B1 must be satisfied for ***all** variations* from the *nominal* value of $[a_1\ a_2\ ...\ a_n]$ that do not exceed a bound Ω, measured by a norm. The bound Ω is often called the *budget of uncertainty* for the constraint. A very large Ω says that the constraint must be satisfied for practically all variations of the coefficients from nominal; a Ω of 0 effectively ignores uncertainty, requiring only that A1 <= B1 for the nominal value of $[a_1\ a_2\ ...\ a_n]$, and saying nothing about departures from this value.

Analytic Solver allows you to choose among four different norms to measure variation from the nominal value: The L1, L2, L-Infinity and D norms – as described in “Uncertainty Sets and Norms” later in this chapter.

Diagnosing Your Model’s Use of Uncertainty

When you create a large model, or modify a large conventional optimization model to include many uncertainties and many constraints, you may or may not realize exactly how the model depends on the uncertainties. Just as you might accidentally use a formula that creates a nonlinear dependence on a decision variable, when you intended to create a linear programming model, you might introduce a dependence on uncertainty in a constraint that you intended to be deterministic. Analytic Solver can *diagnose* the use of uncertainty in your model, comparing it to an ‘intended’ use of uncertainty that you specify in the Task Pane Platform tab Diagnosis group **Intended Use of Uncertainty** option. After clicking the Analyze button, you can select **Reports – Optimization – Uncertainty** for a report of *exceptions* to your intended use of uncertainty.

Problems and Solution Methods

Previous sections of this chapter have described how you can **create models that involve uncertainty** in Analytic Solver, using certain and uncertain parameters, normal and recourse decision variables, normal and chance constraints, and your objective function. You create your models in the same way, regardless of the model transformation or solution method that will be used to solve the problem.

As outlined in the Introduction chapter, Analytic Solver is designed to find *good* solutions – given enough time – to models involving uncertainty in their **most general form**. But Analytic Solver is also designed to *automatically* recognize common special cases such as stochastic linear programming models, and find proven *optimal* solutions at much *faster* speeds. As with conventional LPs, the modeling assumptions of stochastic LPs are somewhat restrictive, but in a wide range of real-world applications, these assumptions can be satisfied. The great advantage of a stochastic LP is its potential *scalability* to tens of thousands of variables and constraints or more.

Before we can discuss classes of problems and how they are solved using Analytic Solver, we must explain how two key *concepts* – decision-dependent uncertainties and recourse decisions – affect the use of two key *solution methods* – stochastic programming and simulation optimization.

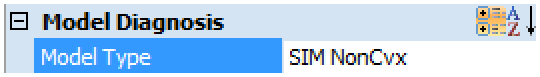
Decision-Dependent Uncertainties

As described in the Introduction section, in many real-world problems, the uncertain variables being modeled are *independent* of the decision variables. In other real-world problems, the uncertainties are *dependent* on the decision variables – they change when the decision variables change.

The methods of ***stochastic programming*** (SP) were created, more than fifty years ago, to deal with problems where the uncertainties are *independent* of the decisions. Much more recently, methods of *robust optimization* (RO) were created, also to deal with problems where uncertainties are *independent* of the decisions. Most methods in the SP and RO literature **cannot be applied** to problems where the uncertainties are dependent on decision variables. But where they can be applied, these methods are scalable to large models, and they generally yield optimal solutions.

The methods of ***simulation optimization*** were created, also more than fifty years ago, to deal with problems where the uncertainties may be *dependent* on the decisions. *This dependence makes the problem much harder to solve*. Where SP and RO methods often are based on linear programming and conic optimization, simulation optimization is usually based on methods such as genetic and evolutionary algorithms, and tabu and scatter search. These methods are more flexible and general – but they require far more solution time, are generally not scalable to large models, and yield only 'improved' solutions, not optimal solutions.

In Microsoft Excel, you can create models where the uncertainties are either independent of, or dependent on the (normal or 'first stage') decision variables. Analytic Solver can detect decision-dependent uncertainties and report this when you choose **Optimize – Analyze Original Problem** or click the Analyze button in the Task Pane:


⊟ Model Diagnosis

Model Type SIM NonCvx

To solve *SIM NonCvx* (non-convex) problems using simulation optimization, you can use the built-in Evolutionary Solver (based on genetic algorithms) or the plug-in OptQuest Solver (which uses tabu and scatter search). You can also use built-in or plug-in nonlinear Solver Engines, in conjunction with Analytic Solver's *multistart* methods, to solve many such problems.

Resolving Uncertainty and Recourse Decisions

The methods of ***simulation optimization*** were developed to solve problems where the uncertainty will not be resolved over the horizon of interest. Hence, these solution methods find optimal values only for normal or 'here and now' decision variables; virtually all simulation optimization methods in the literature **have no concept of recourse decisions**.

Unfortunately, simulation optimization has often been misapplied to problems where the uncertainty *will* be resolved over the horizon of interest. Users may believe that their simulation optimization model will find the best possible decision in the presence of uncertainty. But this is very unlikely to be true, if it's possible to make certain decisions on a 'wait and see' basis. Recourse decisions create *flexibility* for the optimizer, making it easier to satisfy the constraints and attain a better objective.

The methods of ***stochastic programming*** were developed to solve problems where the uncertainty will be resolved over the horizon of interest. Recourse decisions – the ability to make certain decisions on a 'wait and see' basis – have been extensively studied in the SP literature, and many high-performance solution methods for two-stage SP problems with recourse have been developed.

Classes of Problems Involving Uncertainty

An optimization problem that includes uncertain variables is usually called a *stochastic optimization* or *stochastic programming* problem.

Below, we'll classify such problems based on the following factors:

- How the objective and constraints depend on the decision variables
- Whether the uncertain variables depend on the decision variables
- Whether the problem includes recourse decisions (i.e. whether it is one-stage with only normal variables, or two-stage with recourse variables)
- Whether the problem includes chance constraints

We'll also briefly describe how the three solution methods currently supported by Analytic Solver – robust optimization, stochastic programming, and simulation optimization – can be applied to these problems.

One-Stage Problems

One-Stage Linear Programming Problems

The simplest case is a one-stage stochastic linear programming problem:

- The objective and constraints **depend linearly** on the decision variables.
- The uncertain variables are **independent** of the decision variables.
- The problem includes **no recourse decisions**.
- The problem includes **chance constraints** to deal with the uncertainty.

This kind of problem arises if you start with a conventional, deterministic linear programming model (where all coefficients are certain), you add uncertain variables using functions such as **=PsiUniform(0,1)** or **=PsiNormal(2,1.5)** in cells, and you add formulas to compute the objective or certain constraints based on these cells, so that the LP coefficients are now uncertain.

Since no recourse decisions are available, the Solver must find values for the (normal or first-stage) decision variables that satisfy the constraints (normal and chance constraints) and maximize or minimize the objective.

We could solve this problem with **simulation optimization** methods. But since it is linear, this problem is especially well suited for **robust optimization** methods. With RO methods we can solve *much larger* models, hundreds of times faster than with simulation optimization. In the Task Pane Platform tab, we set the Optimization Model group option **Solve Uncertain Models** to Stochastic Transformation, and we set the Transformation group option **Stochastic Transformation** to Robust Counterpart. When we click Optimize, the transformed model is created and solved.

One-Stage Quadratic Programming Problems

A slightly more complex case is a one-stage stochastic problem with a quadratic objective and all linear constraints:

- The objective is a **deterministic, convex quadratic** function of the decision variables – it doesn't depend on the uncertainties.
- The uncertain variables are **independent** of the decision variables.
- The problem includes **no recourse decisions**.
- The problem includes **chance constraints** to deal with the uncertainty.

The situation here is basically the same as for one-stage stochastic linear programming problems. We can apply **simulation optimization** methods, but we can also apply more scalable **robust optimization** methods, as long as the objective doesn't depend on the uncertainties.

One-Stage Nonlinear Optimization Problems

The situation changes if we have a one-stage stochastic *nonlinear* problem:

- The objective or the constraints are **general nonlinear**, convex or non-convex functions of the decision variables.
- The uncertain variables may be **dependent** on the decision variables.
- The problem includes **no recourse decisions**.
- The problem includes **chance constraints** to deal with the uncertainty.

We cannot use Analytic Solver's robust optimization methods or stochastic programming methods on problems with general nonlinear, possibly non-convex functions. Our only option is to use **simulation optimization**; this is feasible since the problem does not include any recourse decisions. Given that we are using simulation optimization, we can allow the uncertainties to be dependent on the decisions, although some speed gains are possible if they are not.

Two-Stage Problems

Two-Stage Linear Programming Problems

A very common case is a two-stage stochastic linear programming problem:

- The objective and constraints **depend linearly** on the decision variables.
- The uncertain variables are **independent** of the decision variables.
- The problem includes both normal decisions and **recourse decisions**.
- The problem may or may not include **chance constraints**.

The presence of recourse decisions in the problem implies that at least some (and usually all) of the uncertainty *will* be resolved over the horizon of interest. Hence the problem is called 'two-stage,' where the normal decisions are made at the first stage – before the uncertainties are resolved – and the recourse decisions are made at the second stage – after they are resolved. The Solver must find single values for the normal variables, and multiple values for the recourse variables (for each realization of the uncertainties) that satisfy the constraints and optimize the objective.

We cannot solve this problem with conventional **simulation optimization** methods, because these methods have no concept of recourse decisions. But since the problem is linear, we can use either **stochastic programming** or **robust optimization** methods – and with these methods we can solve *much larger* models, hundreds of times faster than with simulation optimization. In the Task Pane Platform tab, we set the Optimization Model group option **Solve Uncertain Models** to Stochastic Transformation, and we set the Transformation group option **Stochastic Transformation** to either Deterministic Equivalent or Robust Counterpart. When we click Optimize, the transformed model is created and solved.

If we choose the 'Determ Equivalent,' we can place integer constraints on the normal and recourse variables (though this may make the model more difficult to solve). But the model cannot include chance constraints, and the transformed model may become quite large (its size depends on the *product* of the number of recourse variables and constraints, and the number of simulation trials).

If we choose the 'Robust Counterpart', the model can include chance constraints, and the transformed model will be significantly smaller and solve faster than if we choose 'Determ Equivalent.' But we cannot place integer constraints on the recourse variables, and the solution may be more conservative than with 'Determ Equivalent.'

Two-Stage Quadratic Programming Problems

A slightly more complex case than the one above is a two-stage stochastic problem with a quadratic objective and all linear constraints:

- The objective is a **deterministic, convex quadratic** function of the decision variables – it doesn't depend on the uncertainties or the recourse variables.
- The uncertain variables are **independent** of the decision variables.
- The problem includes both normal decisions and **recourse decisions**.
- The problem includes **no chance constraints**.

Again the presence of recourse decisions in the problem implies that at least some (and usually all) of the uncertainty *will* be resolved over the horizon of interest. We cannot solve this problem with conventional **simulation optimization** methods, because these methods have no concept of recourse decisions. But we can use **stochastic programming** methods – and these methods are far more scalable than simulation optimization methods would be. In the Task Pane Platform tab, we set the Optimization Model group option **Solve Uncertain Models** to Stochastic Transformation, and we set the Transformation group option **Stochastic Transformation** to either Deterministic Equivalent. When we click Optimize, the transformed model is created and solved.

Two-Stage Nonlinear Optimization Problems

If we have a two-stage stochastic nonlinear problem:

- The objective or the constraints are **general nonlinear**, convex or non-convex functions of the decision variables.

- The uncertain variables may be **dependent** on the decision variables.
- The problem includes **recourse decisions**.
- The problem may or may not include **chance constraints**.

We cannot use Analytic Solver's **robust optimization** methods or **stochastic programming** methods on problems with general nonlinear functions, and we cannot use conventional **simulation optimization** methods either, because these methods have no concept of recourse decisions. But future Frontline Systems products may be able to solve such problems; contact us for more information.

Advanced Topics

Bounds, Discretization, and Correlation

An uncertainty specified via a probability distribution such as PsiUniform(0,1) has finite bounds of 0 and 1, but it has a practically infinite set of possible values between 0 and 1; we can only consider a *sample* of the possible values during an optimization. An uncertainty specified via PsiNormal(0,1) has infinite bounds, as well as an infinite set of possible values. In practice, however, most of the probability mass of the Normal distribution lies within three standard deviations on either side of the mean, and a reasonable *sample* of values from the Uniform and Normal distributions can be used for computations.

Analytic Solver uses Monte Carlo simulation to generate samples of uncertainties, based on the probability distributions specified by the user. It uses the minimum and maximum values for each distribution obtained during the simulation as 'effective' lower and upper bounds, which are used in constructing the robust counterpart to an uncertain model.

When Analytic Solver solves a stochastic LP (as explained above) by forming the 'deterministic equivalent' problem, the Monte Carlo trials are used as 'scenarios' in constructing this problem; hence they serve as a discretization of any continuous probability distributions specified by the user.

The impacts of uncertainties on each coefficient of the objective and the constraints may be *correlated* – either because multiple coefficients depend on the same primitive uncertainty, or because correlation was deliberately induced among several uncertain variables. When Analytic Solver solves a stochastic LP by forming the 'deterministic equivalent' problem, the correlations are reflected in the Monte Carlo trials used to construct the problem. When it solves a stochastic LP using robust optimization methods, the robust transformation takes into account the observed correlations in the sample.

Uncertainty Sets and Norms

As described in the earlier section "More on Chance Constraints," if a chance constraint is linear in the decision variables, you can use the *USet* (uncertainty set) criterion, in lieu of the VaR or CVaR criterion. The advantage of using this criterion is that the robust counterpart model more accurately reflects the degree to which you want the chance constraint to be satisfied, which can lead to less conservative solutions, with better objective values. Consider a constraint:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq b$$

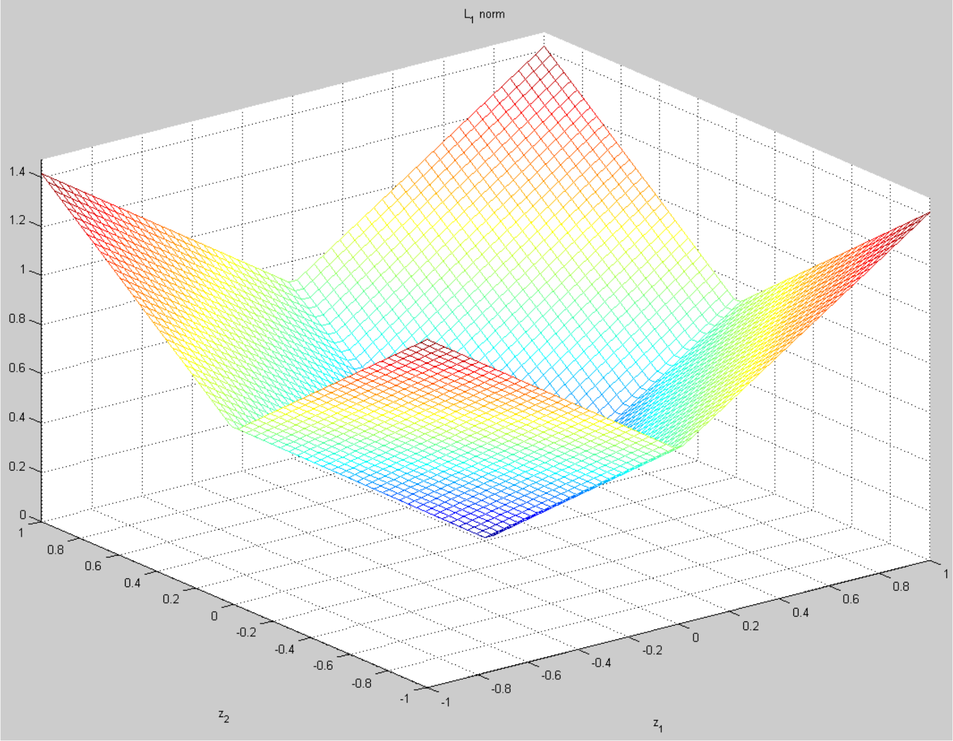
where $=a_1x_1 + a_2x_2 + ... + a_nx_n$ is in A1, *b* is in B, and some or all of the coefficients a_i may depend on uncertain variables z_1, z_2, ... It is useful to think of the vector $[a_1\ a_2\ ...\ a_n]$ as having a *nominal* or expected value, and a *variation* from this value for each realization of the uncertain variables.

A constraint of the form **USet$_\Omega$ A1 <= B1** specifies that A1 <= B1 must be satisfied for ***all** variations* from the *nominal* value of $[a_1\ a_2\ ...\ a_n]$ that do not exceed a bound Ω, measured by a norm. The *uncertainty set* includes all the points formed by adding a vector of allowed variations to the vector of nominal values; the bound Ω is often called the *budget of uncertainty* for the constraint.

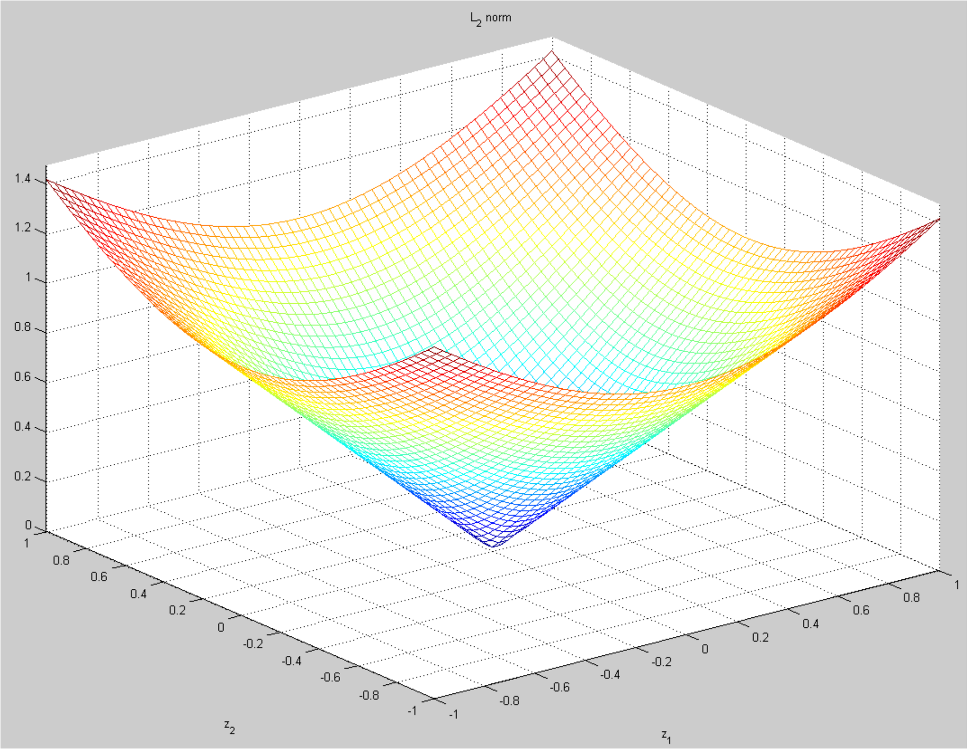
Analytic Solver allows you to choose among four different norms to measure variation from the nominal value: The L1, L2, L Inf (Infinity) and D norms. (One choice of norm applies to all chance constraints.) The graphs shown on the following pages may help you visualize the shape of the uncertainty set (based on two uncertain variables z_1, z_2) for each of the norms.

The D norm represents the intersection of the L1 norm and the L-Inf norm; thus it can define a 'tighter' uncertainty set than either of these norms alone. For the D norm, Ω can be interpreted as a bound on the *number* of coefficients $[a_1\ a_2\ ...\ a_n]$ that depart from nominal values. When the D norm is used, the robust counterpart of a stochastic LP problem is a (larger, conventional) LP problem; when the L2 norm is used, the robust counterpart is an SOCP problem.

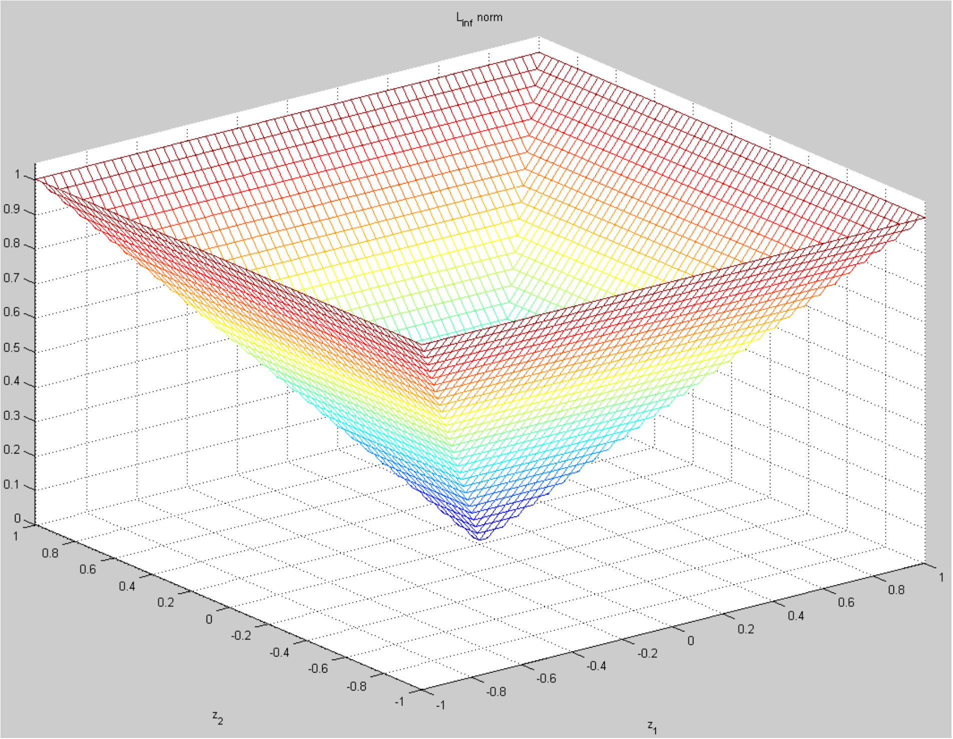
L1 Norm



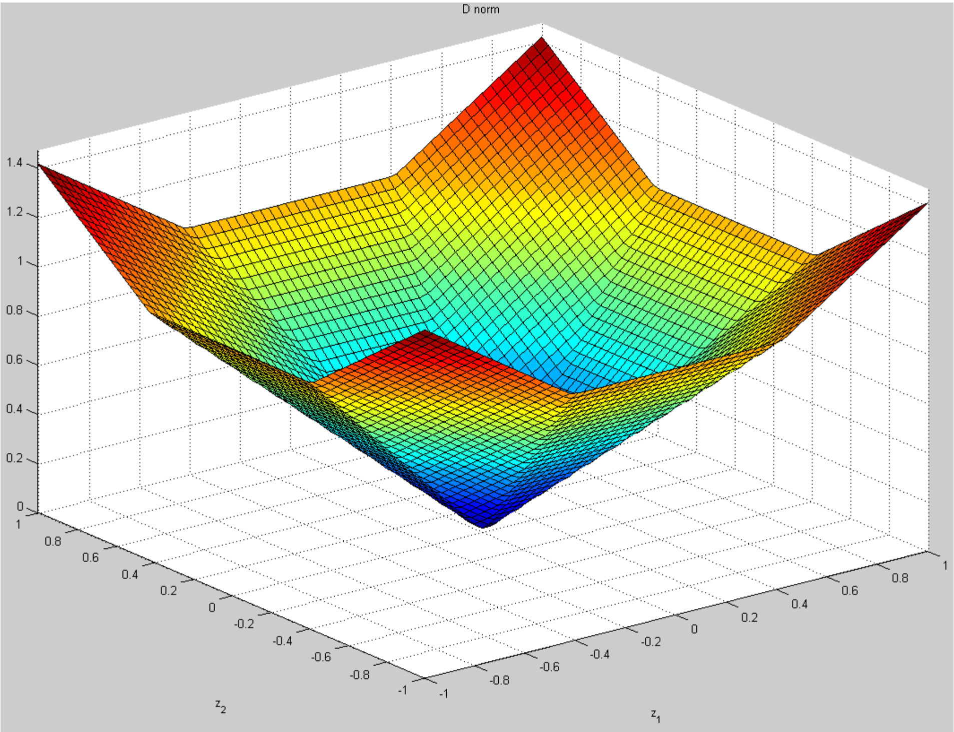
L2 Norm



L Inf Norm



D Norm


D norm
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
z_2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
z_1

Best Practices for Building Large-Scale Models

Introduction

It's a maxim that a successful Solver model will grow in size over time. When the initial results from an optimization model demonstrate ways to achieve significant cost savings, improved schedules, higher quality or increased profits, management is naturally interested in applying these methods to bigger problems. This might involve extending the model to include more plants, warehouses, assembly lines, or personnel; to bring in other divisions or geographic regions; or to cover more time periods, more detailed process steps, or more specific parts or products. The result is an increase in the number of decision variables, constraints, and cells in your model.

When your model grows in size, it becomes more challenging to design and maintain, and also more challenging to solve. Good modeling practices – touched upon in the chapter "Examples: Conventional Optimization" – become *far* more important, so your model remains comprehensible to other Excel users, auditable for errors, and easy to modify. Issues such as your model type (LP, QP, QCP, SOCP, NLP or NSP), sparsity, and scaling also become *far* more important, since they strongly influence the time it takes to solve your model, and the reliability of the solutions you obtain.

This chapter can only briefly survey good modeling practices – entire books have been devoted to this subject (we will recommend some). It focuses on steps you can take to obtain faster and more reliable solutions for large models using the Analytic Solver including:

- Steps towards better performance that are easy to apply in most situations
- Steps you can take – with more design and modeling effort – to improve the *formulation* of your model, by replacing non-smooth or nonlinear constraints with linear (or integer linear) constraints
- Steps you can take to enable Analytic Solver to analyze your model more efficiently

Designing Large Solver Models

A large Solver model in Microsoft Excel is both a large spreadsheet workbook and a large optimization model. If you plan to build such a model, you'll be well advised to learn about good spreadsheet modeling practices, and about good optimization modeling techniques.

We highly recommend the textbook *Management Science: The Art of Modeling with Spreadsheets* by Stephen G. Powell and Kenneth R. Baker, published by John Wiley & Sons, listed in the following chapter "References and Further Reading." Unlike other management science textbooks, this book teaches you "best practices" in modeling and spreadsheet engineering, as well as techniques of linear and nonlinear optimization using Excel.

Other books on good spreadsheet design are hard to find, but through resources like the Amazon.com Marketplace, you may be able to locate a copy of John Nevison's book *Microsoft Excel Spreadsheet Design* (Prentice-Hall, 1990), or his earlier works *1-2-3 Spreadsheet Design* (1989) or *The Elements of Spreadsheet Style* (1987), both of which are still useful in designing modern spreadsheets. A relatively new (2003) book, *Excel Best Practices for Business* by Loren Abdulezer, includes chapters on spreadsheet construction techniques, "makeovers" of spreadsheets developed by others, and spreadsheet auditing.

Training courses in Microsoft Excel often cover at least some elements of good spreadsheet design. They are offered in many venues, from universities and community colleges to public seminars, in-house corporate training, and classes sponsored by computer dealers. Check the course outline or syllabus to see if it features spreadsheet design and good modeling practices, and other topics most relevant to you.

A readily available book on optimization modeling techniques is H. Paul William's *Model Building in Mathematical Programming, 4th Edition* (John Wiley, 1999), listed at the end of the chapter "Introduction." Focusing on modeling for linear and integer programming problems, it includes a treatment of large-scale model structure and decomposition methods that is hard to find elsewhere.

Spreadsheet Modeling Hints

Below is a brief set of suggestions for planning, designing and constructing large Solver models:

Start with a Plan. Plunging in and entering numbers and formulas immediately will quickly lead to problems when constructing a large spreadsheet. Write down your objectives and sketch out a design before you begin working on the real spreadsheet.

Build a Prototype. Plan in advance to build a prototype, *throw it away*, and then build the real spreadsheet model. What you learn from building and solving the prototype will probably save you time in the long run.

Create a Table of Contents. In the upper left corner of your first worksheet, include comments that point readers to the major areas or sections of the spreadsheet.

Separate Data and Formulas. Avoid using constants in formulas, unless they are intrinsic to the mathematical definition of the function you are using. Instead, place constants in cells, and refer to those cells in formulas. Create separate areas on the spreadsheet for input data and for calculations, and identify these with distinct colors, borders or shading.

Document Assumptions, Parameters and Methods. As John Nevison suggested, seek to "surface and label every assumption" in your model. Use labels or cell comments to document key formulas and complex calculations.

Use defined names. Use Excel's Insert Name Define and Insert Name Create commands to assign meaningful names to individual cells and cell ranges. This will help make your formulas clearer and more flexible.

Use and Separate Two-Dimensional Tables. Many elements of your model will lend themselves to a row-column table representation. Create separate table areas, with distinct colors, borders and shading. Collect non-table data (such as individual parameters) into a separate area.

Use Excel Tools to View and Audit Your Spreadsheet. Use the slider to zoom in Excel 2013/2016 to get a high-level view of your spreadsheet's structure. Select 'Show Formulas' on the Formulas tab in Excel 2013/2016 to display formulas instead of values, and scan them for consistency. Learn to use Excel's auditing functions to trace precedents and dependents of your formulas.

Use a Spreadsheet Auditing Tool. Several auditing tools are available, including *SpACE* from the UK Customs and Excise Audit unit, *OAK* from Operis Ltd. in the UK, and the *Spreadsheet Detective* from Southern Cross Software in Australia.

Optimization Modeling Hints

Identify Your Model's Index Sets. Your decision variables, constraints, and many intermediate calculations will fall into groups that are *indexed* by elements such as products (A, B, …), regions (North, South, …), time periods (January, February, …) and similar factors. Identify and write down these index sets and their members. Then organize the columns and rows of your table areas using these index sets. Use the top row and left column of each table area for index set member names as labels.

Identify Your Decision Variables. Once you've identified the quantities that will be decision variables, and how they are indexed (for example, units made by product A, B,… or shipments by region North, South,…), it's usually easier to determine the constraints and their indexing.

Determine the Data You'll Need. In building large optimization models, you will frequently spend a good part of your time figuring out what data you need, how you will get it (and keep it up to date), and how you'll have to summarize or transform it for the purposes of the model. This may involve getting help from your IT department or from other groups that create or maintain the data.

Define Balance Constraints. It is easy to overlook "balance" or "continuity" constraints that arise from the physical or logical structure of your model. For example, in a multi-period inventory model, the ending inventory at time t must equal the beginning inventory at time $t+1$. At each node of a network model (such as a warehouse), the beginning item quantity plus incoming deliveries minus outgoing shipments must equal the ending item quantity ("what goes in must come out").

Learn to Use Binary Integer Variables. Many relationships that you might find difficult to model at all, and many where you might otherwise use IF, CHOOSE or other non-smooth or discontinuous functions, can be effectively modeled with binary integer variables. The section below "Improving the Formulation of Your Model" describes many situations where you can use such variables to organize your model.

Using Multiple Worksheets and Data Sources

Large Solver models and their data are often organized into multiple worksheets of a single workbook. Some large models reference data found in other workbooks. Given the large number of data elements, the sources from which you are getting the data, and the procedures you use to keep the data up to date, multiple worksheets are often necessary or at least useful for organizing your data.

Analytic Solver Pro, Risk/Premium Solver Pro and Premium Solver Pro require that cells containing decision variables and constraint left hand sides are on the active worksheet. But Analytic Solver allows you to define decision variables and constraint left hand sides on any worksheet of a workbook. For this and many other reasons, you are well advised to upgrade to one of these products if your model grows in size. With either product, the formulas in your objective and constraint cells can refer to cells on other worksheets, and those cells on other worksheets can contain formulas that depend, directly or indirectly, on decision variable cells. For more information, see the Frontline Solvers Reference Guide.

Several commentators on good spreadsheet modeling practice feel that models defined on a single worksheet are easier to understand and maintain. In Excel 2013 and beyond, a single worksheet can have up to 16,384 columns and 1,048,576 rows. So you may want to keep the core of your Solver model – the formulas (i) that are used to compute your objective and constraints and (ii) that depend on the decision variables – on a *single worksheet*. If you find that you can better structure your model by placing decision variables and constraints on different worksheets, it’s highly recommended that you adopt a consistent scheme for choosing blocks of variable and constraint (and other formula) cells, and referencing these cells across worksheets.

Some of the data you need may be available in relational databases, OLAP databases or data warehouses. Microsoft Excel provides rich facilities, such as external data ranges and PivotTables, to bring such data into an Excel worksheet. The raw data, even if partially summarized from database records or transactional data, often needs to be further transformed and summarized on your worksheet(s). This is usually easy to do with Excel formulas. But for clarity in your model, we recommend that you use separate worksheet areas, with distinct colors, borders or shading, for formulas that simply massage the data and do not participate in the solution process (i.e. do not depend on the variables). The Solver can determine which formulas depend on the variables, but *you* or your colleagues may find it difficult to do so if the formulas are intermixed.

Quick Steps Towards Better Performance

The rest of this chapter focuses on steps you can take to obtain *faster and more reliable solutions* for large models from Analytic Solver.   This section describes steps that are easy to apply in most situations.

For users of Analytic Solver Upgrade, the best recommendation we can make to improve performance is to *upgrade to Analytic Solver Comprehensive, Analytic Solver Optimization or Analytic Solver Simulation.* This is more than just a “sales pitch” – every step you take costs something, either in terms of money or your effort. For most professionals, the cost of upgrading will be repaid if it saves just a few hours of time. And you can find out *at no cost* whether the upgrade will be worthwhile.

For users of Analytic Solver Comprehensive or Analytic Solver Optimization, we highly recommend that you *try solving your model with our field-installable Solver Engines* – especially the Large-Scale SQP Solver, Knitro Solver, MOSEK Solver Engine, and XPRESS Solver Engine. While the difference in cost may be greater, the same rationale applies: If you can solve your model more quickly or more reliably by upgrading the software, this is almost always cheaper (and yields results sooner) than spending many hours or days of valuable professional time.

Ensure that You Have Enough Memory

If the Solver seems unusually slow and you are using Analytic Solver Desktop, check whether the hard disk activity LED (present on most PCs) is flickering during the solution process. If it is, memory demands may be causing Windows to swap data between main memory and disk, which greatly slows down the Solver. If you're investing money and, especially, hours of your time to develop an optimization model, consider that RAM is very cheap, and relatively easy to install. We recommend *at least* 1 GB RAM if you are working with large Solver models – 2 GB or more is certainly desirable. However, the best solution is simply to run your simulation model using Analytic Solver Cloud. Since the Cloud app runs on Microsoft's Azure App Service, memory is not an issue in the cloud!

Analyze Your Model for Scaling Problems

Poorly scaled calculations are a frequent cause of long solution times and unreliable solution results, for both linear and nonlinear problems. For a further discussion, see "Problems with Poorly Scaled Models" in the chapter "Getting Results: Conventional Optimization."

Add Constraints to Your Model

Frequently, you can improve solution time by adding constraints to your model which may not be *essential* in defining the problem, but which do *further constrain* the search space that the Solver must explore. It's true that the Solver must do more work to handle the additional constraints, but this extra work usually has an excellent payoff if the constraints are "binding" (i.e. satisfied with equality) at some point during the solution process.

The greatest payoff often comes from additional constraints that are simple bounds on the decision variables. This is because (i) it's usually easier for you to determine realistic lower and upper bounds on the variables than to formulate new general constraints, (ii) it's easy to enter bounds on the variables via the Ribbon Constraints choice, and (iii) each of the Solver engines is able to handle bounds on the variables more efficiently than general constraints.

Users often omit upper bounds on their decision variables, and sometimes omit lower bounds as well. A first step towards improving performance is to enter the tightest bounds on the variables that you can, without eliminating possible good solutions.

Since bounds on the variables are especially important for the performance of the Evolutionary Solver and for multistart methods for global optimization used with the nonlinear Solver engines, the Engine tab options for these Solver engines include an option **Require Bounds on Variables**, which is *True* by default. When this box is checked, the Solver will stop with an error message if some variables do not have lower or upper bounds at the time you click Solve. If you are using the Interval Global Solver or the OptQuest Solver, bounds on all variables are *required* – the Solver will always stop with an error message if bounds on the variables are missing.

Improving the Formulation of Your Model

The type of problem you are trying to solve, and the solution method or Solver engine that must be used, has a major impact on solution time:

- Linear programming problems can be solved most quickly.

- Quadratic programming problems take somewhat more time.
- Nonlinear optimization problems take considerably more time.
- Non-smooth problems take by far the greatest amount of time.

This section discusses techniques you can use to replace nonlinear functions, and even non-smooth functions, with equivalent (or nearly equivalent) linear or quadratic functions, or with linear functions and binary integer variables. As explained in the chapter "Mastering Conventional Optimization Concepts," a problem with integer variables can take much longer to solve than a problem without such variables. However, an integer linear problem formulated using the techniques described in this section may still take significantly *less* time to solve than the equivalent nonlinear or non-smooth problem. Moreover, if your problem is integer linear, you can find a *guaranteed* optimal solution, or a solution that is guaranteed to be within at least x% of optimal, whereas with a nonlinear or non-smooth problem you will have no such guarantees. As a rough guide, non-smooth models with more than 1,000 variables may be difficult or impossible to solve in a reasonable amount of time – but equivalent models formulated with linear functions and binary integer variables can often be solved efficiently with the LP/Quadratic Solver. And with the Large-Scale LP/QP Solver, MOSEK Solver or XPRESS Solver, you can often solve linear integer problems of 10,000, 100,000 or more variables in a reasonable amount of time.

A caveat: If you currently have a model with many nonlinear or non-smooth functions, and you decide to implement some of these techniques to speed up solution of your model, bear in mind that you can use the LP/Quadratic Solver, Large-Scale LP/QP Solver, or XPRESS Solver only for models where *all of the problem functions* are linear (except for the objective function, which may be quadratic). If you create a model with a *mix* of nonlinear or non-smooth functions and linear functions using binary integer variables, it may still take a long time to solve.

Convert MMULT Functions to DOTPRODUCT Functions

If you're experiencing prolonged Solver runtimes, particularly with numerous MMULT functions in your model, consider enhancing efficiency by converting these functions to DOTPRODUCT functions.

The MMULT function computes the matrix product of two arrays using the syntax: =MMULT(Array1, Array2). For detailed guidance on the MMULT function, refer to Excel's function Help via Formulas – Insert Function in Excel.

DOTPRODUCT serves as a broader implementation of the Excel function SUMPRODUCT, structured as: =DOTPRODUCT(cell reference, cell reference), where one cell reference encompasses all decision variables and the other comprises constant values in the Solver problem.

For comprehensive insights into the DOTPRODUCT function, see the Solver Add-in Math Functions section within the Analytic Solver Reference Guide.

Techniques Using Linear and Quadratic Functions

Below are three common situations where you might at first expect that a nonlinear function is required to express the desired relationship – but with a simple transformation or approximation, you can use a linear or quadratic function instead.

Ratio Constraints

You may want to express a relationship that seems to require dividing one or more variables by other variables. Suppose that you have a portfolio of 1-month, 3-month and 6-month CDs, with the amounts of each CD in cells C1, D1 and E1, and you wish to limit the average maturity to 3 months. You might write a constraint such as:

(1*C1 + 3*D1 + 6*E1) / (C1 + D1 + E1) <= 3

This constraint left hand side is a nonlinear function of the variables, so you would have to use the GRG Solver to find a solution. However, the same constraint can be rewritten (multiplying both sides by the denominator, then collecting terms) as:

(1*C1 + 3*D1 + 6*E1) <= 3*(C1 + D1 + E1), i.e. -2*C1 + 3*E1 <= 0

This constraint is a linear function of the variables, so you would be able to use the much faster LP/Quadratic Solver to find a solution. (This transformation above relies on the fact that C1 + D1 + E1 >= 0.)

Mini-Max and Maxi-Min

You may want to minimize the maximum of a group of cells such as C1:C5 (or maximize the minimum of a group of cells). It is tempting to use an objective function such as MAX(C1:C5)– but as explained in the chapter "Mastering Conventional Optimization Concepts," MAX (and MIN) are non-smooth functions, so you'd need to use at least the GRG Solver, and perhaps the Evolutionary Solver to find a solution. Instead, you can introduce another variable D1, make D1 the objective to be minimized, and add the constraint:

C1:C5 <= D1

The effect of this constraint is to make D1 equal to the maximum of C1:C5 at the optimal solution. And if the rest of your model is linear, you can use the much faster LP/Quadratic Solver to find a solution.

Linear Reformulation of an Absolute Value Function

An absolute value function, $z = | f(x) |$ may be reformulated into two linear expressions if the function $f(x)$ is a linear function. For example, the function, $z = |x|$, is equal to two piecewise linear functions: $z = x$ if $x > 0$ and $-z = x$ if $x < 0$.

If the absolute value appears in the objective in the form, MIN: $|f(x)| + g(y)$, or is a maximization of the form, MAX: $-|f(x)| + g(y)$, then the model may be reformulated by introducing a new continuous variable, C, and two new constraints $f(x) <= C$ and $-f(x) <= C$.

Take the example below, MIN $|x + y|$ where x and y are variables. To reformulate:

i. Introduce a new variable, C. This very must be non-negative.

ii. Add two constraints: $x + y <= C$ and $–(x + y) <= C$.

iii. Replace $|x + y|$ in the objective with C.

As Solver minimizes the objective, the value of C will be forced to be as small as possible while satisfying the two constraints in ii.

For example, if $x = -4$ and $y =3$,

- The first constraint, $x + y = -4 + 3 = -1$.

- The second constraint $-(x + y) = -(-4+3)= 1$.
- Since C is being minimized (in the objective), C is forced to 1.

Quadratic Approximations

If you cannot represent the entire problem using linear functions of the variables, try to formulate it as a quadratic (QP) or quadratically constrained (QCP) problem, with a quadratic objective and/or constraints. You may be able to use a local, quadratic approximation to a smooth nonlinear function f near a point a:

$$f(x) \cong f(a) + f'(a)(x - a) + \tfrac{1}{2} f''(a)(x - a)^2$$

where $f'(a)$ denotes the first derivative, and $f''(a)$ denotes the second derivative of the function f at the point a. Several Solver engines offer excellent performance on QP problems, and the SOCP Barrier Solver and the MOSEK Solver offer good to excellent performance on QCP problems.

Even if you cannot eliminate nonlinear functions from your problem altogether, you can improve solution time by making an effort to ensure that as many variables as possible occur linearly in the objective and all of the constraints. If you're using the GRG Solver, you can set its Engine tab option **Recognize Linear Variables** to True, to save time during the solution process. And the Large-Scale GRG Solver and Large-Scale SQP Solver engines also recognize both *linearly occurring variables* and *linear constraints* automatically, for still faster solutions. The Large-Scale SQP Solver is particularly effective when used with Analytic Solver Comprehensive or Analytic Solver Optimization because it uses the Interpreter's Structure analysis to *break down each function into linear and nonlinear terms*, which it handles as efficiently as possible.

You can use the Ribbon choice **Optimize – Analyze Original Problem** or the Task Pane Analyze button to easily determine the number of linear variables, functions, and occurrences of variables in functions, as illustrated in the chapter "Examples: Conventional Optimization."

Techniques Using Linear Functions and Binary Integer Variables

Below are three common situations where you might at first expect that a non-smooth function such as IF is required to express the desired relationship – but you can instead use a binary integer variable and one or two linear functions to define an equivalent relationship. The techniques described here are similar to those used when Analytic Solver *automatically transforms* your model, but you can apply these techniques yourself to handle situations where the automatic transformation is not available.

Fixed-Charge Constraints

You may have a quantity x in your model that must "jump" from zero to some (fixed or variable) non-zero value, under certain conditions. For example, a machine on a production line may have a fixed setup time or cost if it is used at all, plus a time or cost per unit produced. You can avoid creating a non-smooth function for x by introducing a binary integer variable y (which is 1 if x is used and 0 if it isn't), and adding a constraint $x <= My$, where M is a constant that is larger than any possible value for x.

For example, suppose you have a machine that has a setup time of 10 minutes, but once set up will process a widget every 30 seconds. Let cell C1 hold the number of widgets you are producing on this machine, and use cell E1 for a binary integer variable *y* that is 1 if you produce *any* widgets on this machine. Then the total production time can be computed as =0.5*C1+10*E1. Assuming that C1 can be at most 10,000, let M1 = 10000 and add a constraint:

C1 <= M1*E1   (or C1 – M1*E1 <= 0)

If variable C1 is nonnegative (C1 >= 0) and variable E1 is binary integer (E1 = binary), then C1 is forced to be 0 whenever E1 is 0, or equivalently E1 is forced to be 1 whenever C1 is greater than 0. Since the production time calculation and the constraint are both linear functions, you can solve the problem with the LP/Quadratic Solver and the Branch & Bound method. This is called a *fixed-charge* constraint.

You can sometimes use a **semi-continuous** variable to model this kind of situation, instead of a binary variable, continuous variable, and “Big M” constraint. At the optimal solution, a semi-continuous variable must either be zero, or must lie within a specified continuous range. This is usually even more efficient than using a fixed-charge constraint as outlined above.

Either-Or Constraints

Constraints in an optimization problem are implicitly connected by the logical operator AND – all of them must be satisfied. Sometimes, however, your model may call for either one constraint (say $f(x) <= F$) or another constraint (say $g(x) <= G$) to be satisfied. You might consider using the OR function in Excel, but as noted in the chapter “Mastering Conventional Optimization Concepts,” this function is non-smooth. Instead, you can introduce a binary integer variable y and a constant M, where M is greater than any possible value for $f(x)$ or $g(x)$, and add the constraints $f(x) – F <= My$ and $g(x) – G <= M(1-y)$. Now, when y=0, $g(x)$ is unrestricted and $f(x) <= F$; when y=1, $f(x)$ is unrestricted and $g(x) <= G$.

For example, imagine you want to allocate your purchases among several suppliers in different geographic regions, each of whom has imposed certain conditions on their price bids. Suppose that one supplier’s bid requires that you either purchase at least 400 units from their Chicago warehouse or else purchase at least 600 units from their Phoenix warehouse, in order to obtain their most favorable pricing. Let cell C1 hold the number of units you would purchase from Chicago, and cell D1 hold the number of units you would purchase from Phoenix. Assume that cell M1 contains 10,000 which is more than the maximum number of units you intend to purchase. You can model the supplier’s either-or requirement with a binary integer variable in cell E1 and the following constraints:

400 – C1 <= M1*E1
600 – D1 <= M1*(1-E1)

Notice that we have reversed the sense of the constraint left hand sides to reflect the “at least” (>=) requirement. If E1=0, then C1 (units purchased from Chicago) must be at least 400, and the second constraint has no effect. If E1=1, then D1 (units purchased from Phoenix) must be at least 600, and the first constraint has no effect.

IF Functions

In the chapter “Mastering Conventional Optimization Concepts,” we used =IF(C1>10,D1,2*D1), where C1 depends on the decision variables, as an example of a non-smooth function: Its value “jumps” from D1 to 2*D1 at C1=10. If you use this IF function directly in your model, you’ll either have to try the Task Pane Platform tab **Nonsmooth Model Transformation** option, or solve the model with the Evolutionary Solver. Instead, you can avoid the IF function and solve the problem with the nonlinear GRG Solver – or even the LP/Quadratic Solver – by introducing a binary integer variable (say E1) that is 1 if the conditional argument of the IF is TRUE, and 0 otherwise. Add the constraints:

C1 – 10 <= M1*E1
10 – C1 <= M1*(1–E1)

When E1 is 0, the first constraint forces C1 <= 10, and the second constraint has no effect. When E1 is 1, the first constraint has no effect, and the second constraint forces C1 >= 10. (If C1=10 exactly, this formulation allows either E1=0 or E1=1, whichever one yields the better objective.) The value of the IF function can then be calculated as D1*E1 + 2*D1*(1-E1), which simplifies to D1*(2-E1) in this example. If D1 is constant in the problem, this is a linear function; if D1 depends linearly on the variables, it is a quadratic; otherwise, it is smooth nonlinear. In all cases, the non-smooth behavior has been eliminated.

Depending on how you use the result of the IF function in the rest of your model, you may be able to take this strategy further. Suppose, for example, that if *f(x) >= F* then you want to impose the constraint *g(x) <= G*; if *f(x) < F* then you don't need this constraint. You can then use a binary variable *y* (cell E1 in the example above), and impose constraints like the pair above plus an additional constraint on *g(x)*:

f(x) – F <= My
F – f(x) <= M(1-y)
g(x) – G <= M(1-y)

If y is 0, *f(x) <= F* is enforced, and the second and third constraints have no effect. If y is 1, *f(x) >= F* and *g(x) <= G* are enforced, and the first constraint has no effect. If *f(x)* and *g(x)* are linear functions of the variables, the constraints involving *y* remain linear, and the problem can be solved with Branch & Bound and the LP/Quadratic Solver.

IF Functions Alternative Method

An alternative method for modeling an IF function in a linear way would be to replace the entire IF formula with a continuous variable.

For example, consider the following function: =IF (f(x) > 0, g(x), h(x)).

b) Introduce a new continuous variable r. In this formulation, the entire IF statement (IF(f(x) > 0, g(x), h(x)) will be replaced by r.

c) Introduce a new binary variable, y.

d) Introduce a large number, M, that is large enough to satisfy f(x) < M, g(x) < M, h(x) < M for all x.

e) Add the following constraints:

   i. f(x) <= M * y and f(x) >= M * (y-1)

   ii. g(x) – M * y <= r <= g(x) + M * y

   iii. h(x) – M (1-y) <= r <= h(x) + M * (1-y)

Alternative Method in Practice

Let's say that the original IF statement was entered into the spreadsheet as IF(C1>0, 2*D1, D1) where C1 is a continuous variable, so f(x) = C1, g(x) = 2*D1 and h(x) = D1. C1 will represent the r variable, introduced in step b.

If C1 (the r variable) is a positive value (say C1 = 0 or 4), then:

1. The first constraint, f(x) <= M * y, forces y to 1 to satisfy the constraint, 4 <= M * 1. The second constraint, f1(x) >= M * (y-1), evaluates to 4 >= M * 0 which simplifies to 4 >= 0. This constraint is satisfied when C1 = 0 or 4.
2. Since y has already been forced to 1, the ii constraints evaluate to: 2*D1 – M * 1 <= 4 and 4 <= 2*D1 + M *1. Both of these constraints are satisfied when C1 = 0 or 4.
3. Since y has already been forced to 1, the iii constraints evaluate to: D1 – M * (1-1) <= 4 or D1 <= 4 and 4 <= D1. *These two constraints force r (the value in cell C1), to equal D1.*

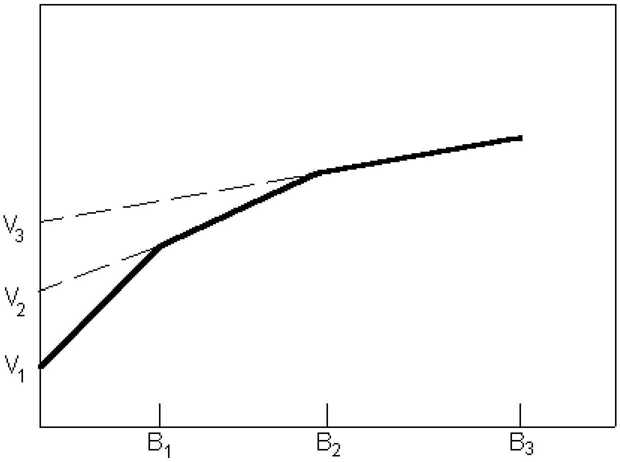
If C1 is a negative value (say C1 = -4), then:

1. The first constraint, f(x) <= M * y, evaluates to -4 <= M * y. At this point, y can be either 0 or 1 to satisfy the constraint. The second constraint, f(x) >= M * (y-1) evaluates to -4 >= M * (y-1). This constraint forces the y variable (the binary variable) to 0 in order to satisfy the constraint.
2. Since y has been forced to 0, the next set of constraints evaluate to: 2*D1 <= -4 <= 2*D1. *These two constraints force r (the value in cell C1) to equal 2 * D1.*
3. The next set of constraints evaluate to: D1 – M * (1 – 0) <= -4 <= D1 + M * (1- 0) which simplifies to D1 – M <= -4 <= D1 + M.

Using Piecewise-Linear Functions

Many problems involve “stepped” price schedules or quantity discounts, where you might at first expect that a non-smooth function such as CHOOSE or LOOKUP is required to express the relationship. You might be surprised to learn that you can instead use linear functions and binary integer variables to express the relationship.

For example, you might be purchasing parts from a vendor who offers discounts at various quantity levels. The graph on the next page represents such a discount schedule, with three prices and three “breakpoints.” You have a decision variable *x* representing the quantity to order.



The three prices (slopes of the line segments) are c_1, c_2 and c_3. V_1 represents a fixed initial cost; V_2 and V_3 are also constant in the problem and can be computed from:

$V_2 = V_1 + c_1*B_1 - c_2*B_1$
$V_3 = V_2 + c_2*B_2 - c_3*B_2$

In the model, the variable x is replaced by three variables x_1, x_2 and x_3, representing the quantity ordered or shipped at each possible price. Also included are three 0-1 or binary integer variables y_1, y_2 and y_3. Since you want to minimize costs, the objective and constraints are:

Minimize $V_1*y_1 + V_2*y_2 + V_3*y_3 + c_1*x_1 + c_2*x_2 + c_3*x_3$
Subject to $x_1 \leq B_1*y_1$, $x_2 \leq B_2*y_2$, $x_3 \leq B_3*y_3$

If the cost curve is concave as shown above, this is sufficient; but if the function is non-concave (it may vary up and down), additional “fill constraints” are needed:

$y_1 + y_2 + y_3 \leq 1$
$x_1 \leq B_1*y_2$
$x_2 \leq B_2*y_3$

This approach is called a “piecewise-linear” function. It can be used in place of a CHOOSE or LOOKUP function, and it results in a linear integer model instead of a difficult-to-solve non-smooth model. Piecewise-linear functions can also be used to approximate a smooth nonlinear function, by using line segments with slopes matching the gradient of the nonlinear function at various intermediate points.

Organizing Your Model for Fast Solution

This section describes ways you can organize your model so that Analytic Solver can analyze it more efficiently. Most of this section is devoted to an in-depth discussion of “fast problem setup” for *linear* and *quadratic* models (possibly with integer variables); it is not applicable to *nonlinear* and *non-smooth* models. Because the Polymorphic Spreadsheet Interpreter in Analytic Solver has largely superseded this form of fast problem setup, this section is relevant primarily for very large LP models (100,000 variables or more), where memory required for the Interpreter may be greater than available RAM causing swapping to disk. In order to use Fast Problem Setup, the Psi Interpreter setting must be set to Excel Interpreter on the Platform tab of the Solver Task Pane.

Fast Problem Setup

If your model is linear or quadratic, you may find that the Solver spends most of its time with “Setting Up Problem...” on the Excel status bar, then speeds through the “Trial Solutions” very quickly. The setup time is required to extract the LP coefficients of the problem functions by recalculating the worksheet. (The LP coefficients are the first partial derivatives of the problem functions with respect to the variables, and they are obtained by the method of *finite differencing*, which requires $n + 1$ recalculations if there are n decision variables.)

Although the time taken to extract the LP coefficients was drastically decreased in V2017-R2, even more of this setup time can be avoided if you write the formulas for the objective function and *all* of the constraint left hand sides using the functions recognized for *fast problem setup*: SUM, SUMPRODUCT, DOTPRODUCT, QUADPRODUCT andmay require some work on your part to revise a model you have already constructed, but you'll be rewarded with a *5- to 100-fold speed improvement* in setup time, compared to the time taken by the Excel Solver or the Excel Interpreter.

You can always express a linear or quadratic programming problem using these functions for the objective and all of the constraints, although you may need to introduce new sets of cells to hold the calculated coefficients so that these cells can be referenced by one of the fast problem setup functions.

The PSI Interpreter in Analytic Solver uses the techniques of *automatic differentiation* to obtain the LP coefficients faster and more accurately than they can be obtained via finite differencing. Because the Interpreter handles almost every kind of Excel formula and built-in function, you don't have to do the work of designing your model – or revising an existing model – to use only the small set of functions recognized for fast problem setup.

Fast problem setup is still available in Analytic Solver as a specialized method of extracting the constant Jacobian matrix (the LP coefficients) of a linear or quadratic problem, and the constant Hessian matrix (the QP coefficients) of a quadratic objective function – but it is used only if you set the Task Pane Platform tab Optimization group Interpreter option to **Excel Interpreter**. If your LP or QP model is very large, defining it in fast problem setup format may still save time compared to use of the PSI Interpreter – but the advantage is not nearly as great as the 5- to 100-fold speed improvement mentioned above.

The following subsections describe the functions supported for fast problem setup, and the form of the function arguments that you must use to ensure that they are recognized for fast setup purposes.

The SUM Function

The simplest case of a function recognized for fast problem setup is a formula such as =SUM(C1:C10) where C1 through C10 are decision variables. An example of the use of SUM to define constraints can be found in the "Shipping Routes" sheet in the SOLVSAMP.XLSX workbook included with Microsoft Excel. Note that a SUM of decision variables is a linear function where all of the coefficients are 1. To be recognized in fast problem setup, your formula must consist *only* of =SUM(*cells*) (with no constants) where every cell referenced is a decision variable. You may use absolute or relative references or defined names in the arguments to SUM.

The SUMPRODUCT Function

The SUMPRODUCT function is documented in Microsoft Excel online Help. It returns the sum of the products of corresponding elements in its two arguments, which is exactly the form of a linear function:

$a_1x_1 + a_2x_2 + ... + a_nx_n$

SUMPRODUCT requires that its two arguments refer to the same *number* of cells in the same *orientation* (either a row, a column or a rectangular area of cells). Only single selections, not multiple selections, are permitted in the arguments. If SUMPRODUCT is used in an array formula (see below), it will return the same value in every resulting array element, which is usually not the result you want. (The DOTPRODUCT function, described below, has more flexible arguments and is far more useful in array formulas.) To be recognized in fast problem setup, your formula must consist *only* of =SUMPRODUCT(*cell range*, *cell range*) where one of the cell ranges consists *entirely* of decision variables, and the other cell range consists *entirely* of cells that are constant in the Solver problem. You may list the arguments in either order, using absolute or relative references or defined names.

Other Functions for Fast Problem Setup

Use of the MMULT function is illustrated below under “Using Array Formulas.” To be recognized in fast problem setup, your formula must follow the same rules as for SUMPRODUCT: It must consist *only* of =MMULT(*cell range*, *cell range*) where one cell range specifies the decision variables, and the other cell range specifies the corresponding coefficients.

The DOTPRODUCT and QUADPRODUCT functions are described in their own sections below. To be recognized in fast problem setup, your usage of these two functions must follow the same rules as for SUMPRODUCT.

To qualify as a quadratic programming (QP) problem – which can be solved efficiently by the LP/Quadratic Solver engine – only the *objective function* (not any of the constraints) may use QUADPRODUCT, or any other quadratic form.

Using Array Formulas

Optimization models in algebraic form can often be expressed more compactly using indexing and summation notation. For example, the five constraints in EXAMPLE1 could be written as:

$$\sum_{j=1}^{3} a_{ij}\, x_j,\ i=1,...,5$$

The SUMPRODUCT function corresponds to the summation expression above for one constraint. The *entire set* of five constraint formulas could be defined with the array form of the DOTPRODUCT function (described in detail later in this section). In EXAMPLE1, you would select cells C11 to C15 and “array-enter” the following formula:

{=DOTPRODUCT(D9:F9,D11:F15)}

The braces above are not typed, but they appear when you “array-enter” the formula by pressing CTRL-SHIFT-ENTER instead of just ENTER. If you aren’t familiar with array formulas in Microsoft Excel, you can read about them in Excel’s online Help. They are one of the most useful features of Microsoft Excel.

If your LP model is “dense” and regular in form rather than “sparse,” you may wish to consider use of Microsoft Excel’s matrix built-in functions, such as MMULT which (when array-entered into a group of cells) yields the matrix product of its two operands. For example, the five constraints in EXAMPLE1 (normally installed into C:\ProgramData\Frontline Systems\Examples\StandardExamples.xlsx could be written in vector-matrix form as:

$$\mathbf{Ax} <= \mathbf{b}$$

where **A** is the matrix of coefficients, **x** is the vector of decision variables and **b** is the vector of constraint right hand sides. In the Microsoft Excel formula language, the left hand side of this expression could be written as:

{=MMULT(_A,TRANSPOSE(_X))}

(The TRANSPOSE function is needed only to “match up” the orientation of the matrix _A with the row vector _X.) In worksheet EXAMPLE1, if you insert defined names _A for the coefficients D11:F15 and _X for the variables D9:F9, then select cells C11:C15 and array-enter the above formula, it will compute the values of all five constraints.

The PSI Interpreter in Analytic Solver recognizes most kinds of array formulas supported by Microsoft Excel. But (for rather technical reasons) the use of array formulas actually involves a speed *disadvantage* in the PSI Interpreter when the coefficients are extracted via automatic differentiation.

If you're using Analytic Solver, we recommend that you use array formulas where they make sense, and focus on making your model simple and easy to maintain. In a large model, you'll probably find that you want or need to use multiple tabular areas for the formulas that define your constraints, and it may be inconvenient or impractical to define entire constraint left hand sides with functions like MMULT and TRANSPOSE.

Using the Add-in Functions

Analytic Solver products define two special Excel functions: DOTPRODUCT and QUADPRODUCT. These functions behave just like Excel built-in functions: You can use them in formulas in any spreadsheet (not only in Solver models). When you use the Insert Function… menu option, these functions will appear in the "Select a Function" or "Paste Function" list (classified as Math & Trig functions), and you'll be prompted with named edit fields for their arguments.

In addition, DOTPRODUCT and QUADPRODUCT are recognized for purposes of fast problem setup as described earlier in this chapter. They are also recognized by the PSI Interpreter.

Using DOTPRODUCT

DOTPRODUCT is a generalized version of the Excel function SUMPRODUCT, and it is very useful for defining the objective function and constraints of linear programming problems. DOTPRODUCT is also recognized for fast problem setup as described above, provided that you follow the rules outlined earlier: Your formula must consist *only* of =DOTPRODUCT(*cell reference*, *cell reference*) where all of the cells in one of the cell references are decision variables, and all of the cells in the other cell reference are constant in the Solver problem. Each *cell reference* must be either an *individual selection* or a *defined name*, but the cell ranges specified by the two arguments need not have the same "shape" (row, column, or rectangular area).

For use in Excel and for purposes of fast problem setup, DOTPRODUCT will accept defined names that specify *multiple selections* for either of its arguments. For example, if you had designed a model where the decision variables consisted of *several* rectangular cell selections, you could still calculate the objective function for your model with *one* call to DOTPRODUCT.

DOTPRODUCT always processes its arguments in *column, row, area* order – in an individual selection it reads cells across columns, wrapping around to subsequent rows, and in a multiple selection it reads the individual cell selections in the order in which they are listed. For example, the formula:

=DOTPRODUCT(A1:C2,D1:D6)

will calculate as =A1*D1+B1*D2+C1*D3+A2*D4+B2*D5+C2*D6.

The Array Form of DOTPRODUCT

If SUMPRODUCT is used in an array formula, it returns a scalar (single number) result, which is returned in every cell of the array. However, if DOTPRODUCT is used (with the proper arguments) in an array formula, it returns an *array result*. You can use this capability to calculate the left hand sides of several constraints with a single array formula. In a sparse model where you'd like to use the built-in function MMULT to compute the constraint values, but the variables and constraints aren't laid out in a single matrix, you can use the array form of DOTPRODUCT instead.

Further, when you use the array form of DOTPRODUCT, Analytic Solver will recognize this form and use it to process many constraints at once in problem setup. (The array form is recognized for fast problem setup, and it's also recognized by the PSI Interpreter.) If you can't use the array form, even the simple form of DOTPRODUCT will save time in problem setup.

DOTPRODUCT will return an array value when the number of cells in one of its arguments is an *even multiple* of the number of cells in its other argument. As an example, consider the calculation of parts used in the LP model EXAMPLE1. The decision variables are in cells D9 to F9 (3 cells), and the coefficients of the constraint left hand sides – the number of parts used for each product – are in cells D11 to F15 (15= 3*5 cells). We want to calculate the left hand sides of the constraints in cells C11 to C15. To do this, we would first select the group of five cells C11:C15 with the mouse. Then we would type:

=DOTPRODUCT(D9:F9,D11:F15)

completing the entry with CTRL+SHIFT+ENTER instead of just ENTER. The formula will display as {=DOTPRODUCT(D9:F9,D11:F15)} – the braces are added by Microsoft Excel when the formula is array-entered. With the cell values shown in EXAMPLE1 prior to solution (e.g. 100 for each of the decision variables), this array formula will calculate 200 in C11, 100 in C12, 500 in C13, 200 in C14 and 400 in C15. Hence, it will compute the same set of values as the array expression shown earlier: {=MMULT(_A, TRANSPOSE(_X))}.

Whether it is used in the simple form or the array form, DOTPRODUCT always processes its arguments in *column, row, area* order. In the array form, when the cells in the "shorter" argument have all been processed and cells remain to be processed in the "longer" argument, DOTPRODUCT "wraps around" to the beginning of the "shorter" argument. In the example above, cell C11 calculates the value =D9*D11+E9*E11+F9*F11; cell C12 computes =D9*D12+E9*E12+F9*F12; and so on. Keep this rule in mind when you use the array form of DOTPRODUCT, and keep your spreadsheet layouts as simple as possible!

Using QUADPRODUCT

The QUADPRODUCT function can be used to define the objective for quadratic programming problems in a single function call, as required for fast problem setup.

As explained in "Quadratic Functions" in the chapter "Mastering Conventional Optimization Concepts," a quadratic function is a sum of terms, where each term is a (positive or negative) constant (called a *coefficient*) multiplied by a *single variable* or the *product of two variables*. This means that in order to represent the most general quadratic function, we might have a coefficient for each instance of a single variable, and a coefficient for each possible *pair* of two variables. The QUADPRODUCT function is designed to supply coefficients for each single variable and each pair of variables, in a manner similar to SUMPRODUCT and DOTPRODUCT.

You supply the arguments of QUADPRODUCT as shown below:

=QUADPRODUCT(*variable cells, single coefficients, pair coefficients*)

The first argument must consist entirely of decision variable cells. The second and third arguments must consist entirely of cells whose values are *constant* in the optimization problem; if these cells contain formulas, those formulas must not refer to any of the decision variables. The second argument supplies the coefficients to be multiplied by each single variable in the first argument, using an element-by-element correspondence. The third argument supplies the coefficients to be multiplied by each *pair* of variables drawn from the first argument. Hence, if there are n cells in the first argument, there must be n^2 cells in the third argument. If the variables are represented by $x_1,x_2,...,x_n$, the single coefficients by $a_1,a_2,...,a_n$, and the pair coefficients by $c_1,c_2,...,c_N$ where $N = n^2$, QUADPRODUCT computes the function:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{n(i-1)+j}\, x_i\, x_j + \sum_{j=1}^{n} a_j\, x_j$$

The pairs are enumerated starting with the first cell paired with itself, then the first cell paired with the second cell, and so on. For example, if the first argument consisted of the cells A1:A3, there should be nine cells in the third argument, and the values in those cells will be multiplied by the following pairs in order: A1*A1, A1*A2, A1*A3, A2*A1, A2*A2, A2*A3, A3*A1, A3*A2, and A3*A3. The value returned by QUADPRODUCT is the sum of all of the coefficients multiplied by their corresponding single variables or pairs of variables.

Multiple selections can be used for each argument of QUADPRODUCT, subject to the same considerations outlined above for DOTPRODUCT: You can use the general syntax for multiple selections in Microsoft Excel, but defined names are needed for purposes of fast problem setup, and multiple selections are not accepted by the PSI Interpreter.

A common application of quadratic programming is to find an “efficient portfolio” of securities – often called *portfolio optimization*. Worksheet EXAMPLE4 in the **StandardExamples.xlsx** workbook, used in the chapter “Examples: Conventional Optimization,” illustrates portfolio optimization using the Markowitz method. This model uses the QUADPRODUCT function to compute the variance of a portfolio of five securities, and uses this quantity as the objective to be minimized, subject to a constraint that gives a lower limit on the portfolio return. Because the *objective* is a quadratic function, and the *constraints* (including the bound on return) are all linear functions, this Solver model is a quadratic programming (QP) problem which can be handled by the LP/Quadratic Solver in Analytic Solver Comprehensive and Analytic Solver Optimization.

References and Further Reading

Introduction

Although this Guide provides many valuable hints for making effective use of Analytic Solver, it does not attempt to teach you everything about optimization, Monte Carlo simulation, and stochastic optimization. We strongly recommend that you consult one of the books cited below, or discuss your problem with a **knowledgeable consultant** at Frontline Systems, at a nearby university, or at a firm specializing in optimization and simulation methods. There's a vast literature on problems of various types and for various industries and business situations that have been solved successfully with the methods available in the Solver. Don't reinvent the wheel – find out how others have solved problems similar to yours!

You may also want to attend a **training seminar** on risk analysis and simulation, optimization, and other advanced techniques in Excel, presented by Frontline Systems instructors or consultants who work closely with us. For current information on training seminars, visit www.solver.com or contact us at info@solver.com.

Textbooks Using Analytic Solver

***Management Science: The Art of Modeling with Spreadsheets*, 4th** Edition by Stephen G. Powell and Kenneth R. Baker, published by John Wiley & Sons, ISBN 978-1118582695.

Management Science, written by faculty members at Dartmouth's Tuck School of Business, helps business professionals gain the essential skills needed to develop real expertise in business modeling. The 4th Edition uses Analytic Solver Platform throughout the text, and describes how to use most of its features. Overall, this book teaches you "best practices" in modeling and spreadsheet engineering, methods for data visualization, analysis and data science, and techniques of linear and nonlinear optimization, Monte Carlo simulation, and stochastic optimization using Excel. Additional open-ended case studies that are less structured are included, along with new exercises.

Spreadsheet Modeling and Decision Analysis: A Practical Introduction to Management Science, 7th Edition by Cliff T. Ragsdale, published by Cengage Publishing, ISBN 978-1285418681.

Cliff Ragsdale's book was first to base all of its optimization examples on the Microsoft Excel Solver, and it has become the best-selling MBA textbook for management science. You'll find a discussion of linear, nonlinear and integer programming; an explanation of sensitivity analysis and how to use the Solver's reports; topics like goal programming and multiobjective optimization; and additional coverage of regression, time series analysis, queuing, project management, decision analysis, and other topics. The 7th Edition uses Analytic Solver Platform for its examples.

Essentials of Business Analytics, 1st Edition by Jeffrey D. Camm, James J. Cochran, Michael J. Fry, Jeffrey W. Ohlmann and David R. Anderson, published by Cengage Publishing, ISBN 978-1285187273.

This book provides coverage over the full range of descriptive analytics (classical statistics), predictive analytics (data science), and prescriptive analytics (optimization and simulation), integrating several important analytics topics not covered by any other single book. It includes step-by-step instructions to help students learn how to use Excel and Analytic Solver Platform. Practical, relevant problems at a variety of difficulty levels help students learn the material. Applications are drawn from all functional business areas: finance, marketing, operations, etc. Data sets are available for most exercises and cases.

Business Analytics, 1st Edition by James Evans, published by Pearson Publishing, ISBN 978-0132950619.

This book provides readers with the fundamental concepts and tools needed to understand the emerging role of business analytics in organizations, how to apply basic business analytics tools in Excel, and how to communicate with analytics professionals to effectively use and interpret analytic models and results for making better business decisions. The 1st Edition (which appeared before Analytic Solver Platform was released) teaches both forecasting and data science methods using XLMiner, and optimization, simulation and other topics using Risk Solver Platform.

Textbooks Using Solver or Premium Solver

VBA for Modelers - Developing Decision Support Systems Using Microsoft Excel, 4th Edition by S. Christian Albright, published by Cengage Publishing.

This unique book will prove invaluable to anyone seeking to use VBA (Visual Basic Application Edition) to programmatically control Microsoft Excel and build custom applications. It includes a basic introduction to VBA and the Excel object model, and 16 example applications developed in depth with VBA source code, many of them calling the Solver's VBA functions. The applications include blending, product mix, production scheduling and similar models, plus capital budgeting, stock trading, option pricing and portfolio optimization.

Other Textbooks

Model Building in Mathematical Programming, 5th Edition by H.P. Williams, published by John Wiley, ISBN 978-1118443330. Though it doesn't cover spreadsheet optimization, this book is still valuable for its explanation of model-building approaches, especially if you are building large-scale optimization models. It provides an in-depth treatment of modeling for linear and integer programming problems. It mentions nonlinear models only briefly, but it offers a unique treatment of large-scale model structure and decomposition methods. It also includes a complete discussion of 24 models drawn from various industries.

Academic References

The following academic journal articles, written by the developers of the Excel Solver, Premium Solver Pro, Analytic Solver Pro, Premium Solver Platform, Risk Solver Platform and Analytic Solver Platform, describe many of the algorithms and technical methods used in these products. The first article describes the design of the original Excel Solver. You can download PDF versions of the first three articles at **http://www.solver.com/academic.htm**:

D. Fylstra, L. Lasdon, J. Watson and A. Waren. Design and Use of the Microsoft Excel Solver. *INFORMS Interfaces* 28:5 (Sept-Oct 1998), pp. 29-55.

I. Nenov and D. Fylstra. Interval Methods for Accelerated Global Search in the Microsoft Excel Solver. *Reliable Computing* 9 (2003): pp. 143–159.

D. Fylstra, "Introducing Convex and Conic Optimization for the Quantitative Finance Professional," *Wilmott Magazine* (March 2005), pp. 18-22.

For a technical description of the nonlinear GRG solver included with the standard Microsoft Excel Solver and the Premium Solver, please consult the following:

L.S. Lasdon, A. Waren, A. Jain and M. Ratner. Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming. *ACM Transactions on Mathematical Software* 4:1 (1978), pp. 34-50.

L.S. Lasdon and S. Smith. Solving Sparse Nonlinear Programs Using GRG. *INFORMS Journal on Computing* 4:1 (1992), pp. 2-15.