RISKOptimizer Guide

This short guide is designed to give you a high-level overview of RISKOptimizer, the sophisticated stochastic optimizer built into @RISK. This guide assumes you are already familiar with @RISK. If you have never used @RISK before, please see the @RISK Getting Started Guide.*

Approximately the first half of this document is quite theoretical in nature, introducing you to new terminology and concepts associated with optimization, and in particular, stochastic optimization. The second half is more practical in nature, using one of the example models that is distributed with @RISK to show how to configure, run, and analyze the results of an optimization.

Please note: RISKOptimizer is only available in the Industrial Edition of @RISK.

Version 2 - Last Updated 9/2/2020

Palisade

Minimize risk. Maximize potential.

Contents

What is Optimization? 3
Deterministic Optimization in Excel 3
Stochastic Optimization in RISKOptimizer 4
Solving Methods 5
Optimization Engines 7
Efficient Frontier Analysis 8
Configuring a RISKOptimizer Model 9
Running an Optimization 13
Analyzing the Results of an Optimization 14
Conclusion 15

What is Optimization?

Optimization is the process of trying to find the "best" solution to a problem. For example, a company might have three manufacturing plants, each producing different quantities of different products. Given the cost for each plant to produce each product, the costs for each plant to ship to each store, and the limitations of each plant, the goal is to find the optimal way to adequately meet the demand of local retail stores while minimizing the transportation costs.

This is one type of problem that optimization is designed to answer, but it is certainly not the only one. Optimization models of various types appear in the fields of operations management, finance, marketing, economics, engineering, and others.

An optimization model generally consists of three things:

- A ***target*** or ***goal*** that you want to minimize or maximize.
- A set of ***adjustable values*** that may be changed in order to improve the target.
- A set of ***constraints*** that need to be satisfied.

A piece of software designed to perform optimization is called an ***optimizer.*** During the optimization process, the optimizer tries different combinations of adjustable values. Each such set is often called a ***trial***. For each trial, the model is recalculated and a new value for the target is generated. Any trial that fails one or more constraints is not among the possible solutions of the optimization problem (i.e. it is not a "valid" solution).

An optimization ends either when the optimizer decides it has reached the best possible value, some predefined ***stopping condition*** is met, or we simply decide that we are happy with the best result that has been found and stop the optimization manually. (Only for certain kinds of optimization problems will an optimizer be able to know that it has reached the absolute best possible value, so externally defined stopping conditions can often be useful.)

Deterministic Optimization in Excel

Traditional optimization problems are ***deterministic***. That simply means the optimization model doesn't contain any uncertain quantities. (The Evolver add-in produced by Palisade is designed to solve deterministic optimization problems.)

In an Excel-based deterministic optimization model, the target, adjustable values, and constraints are typically defined with cell references and formulas. For example, we might have an optimization model where:

- The target is to maximize the value of cell C5.
- We accomplish this by adjusting the values in the cell range A1:A5, each of which are constrained to be between 0 and 100.
- An additional constraint says that another cell in the model must obey E1 < 0.5 for the solution to be valid.
- Model logic combines the adjustable cells, perhaps in quite a complicated fashion (using a large number of worksheets, formulas in other cells, etc.) into the target cell at C5 and the additional constraint cell at E1.

For each trial of the optimization, the optimizer will put new set of test values in the cells A1:A5 and then recalculate the model. It then checks that the additional constraint in cell E1 is met; if it is not met, it checks by how much the solution exceeds the specified maximum value so in the future it can make corrections. (Note: The constraints on cells A1:A5 to be between 0 and 100 will automatically be met, since optimizers are usually smart enough to not generate solutions that fall outside simple constraints on their domains.) If so, it will then look at the value of the target. Based on what it finds, the optimizer uses an ***optimization algorithm*** to make modifications to come up with the next set of adjustable values it will try, trying to meet the constraint and further improve the target value. This process repeats until the optimization ends.

Stochastic Optimization in RISKOptimizer

In contrast, the RISKOptimizer feature of @RISK offers the ability to perform ***stochastic*** optimization. That means the model can contain uncertain quantities. This effectively merges the ideas of simulation and optimization together.

Stochastic optimization problems appear in all fields. Here are some examples:

- Selection of optimal production and capacity levels for new products with uncertain market conditions.
- Identification of optimal inventory levels with uncertain demand.
- Portfolio allocation for risk minimization.
- Identification of an optimal product mix when product markets are geographically distributed, and demand levels are uncertain.
- Determination of optimal levels for options purchases when hedging.
- Yield management when the same product is sold at different prices under different restrictions.
- Scheduling with uncertain task times.

In RISKOptimizer, the optimization target usually becomes a statistical measure. In addition, optimization constraints often will be as well. Here's a stochastic version of the model described in the

previous section. Please note, underlined text is used here to highlight important differences from the original deterministic model:

- The target is to maximize the 75th percentile of the uncertain value in cell C5.
- We accomplish this by adjusting the values in the cell range A1:A5, each of which are constrained to be between 0 and 100.
- An additional constraint says that the mean value of the cell E1 in the model must be less than 0.5.
- Model logic combines the adjustable cells, perhaps in quite a complicated fashion (using a large number of worksheets, formulas in other cells, and uncertain @RISK distribution functions, etc.) into the target cell at C5 and the additional constraint cell at E1.

For each trial of the optimization, RISKOptimizer runs a full @RISK simulation. In each iteration of one of these simulations, probability distribution functions in the model are sampled and a new value for the target cell is generated. At the end of a simulation, the result for the trial solution is the statistic of the target cell's distribution that you want to optimize (and the statistics of the constrained cells). These values are then used by the optimizer to generate new and better trial solutions. For each new trial solution, another simulation is run and new values for the target statistics and constrained statistics are generated.

To avoid confusion, it is helpful to distinguish carefully between the terms "trials" and "iterations" in the RISKOptimizer process. The overall process involves a sequence of "trials," where one set of adjustable cell values is tested each trial. To evaluate the target cell (and the constrained cells) for a trial, a standard @RISK simulation is run for a specified number of "iterations." This distinction helps you understand why RISKOptimizer can take quite some time to run! For example, it might require 1000 trials to converge to optimality, and each of these trials might require a 500-iteration simulation. This is a lot of computer calculation, especially for complex models.

When defining a constraint in a RISKOptimizer model, we must distinguish between probabilistic and non-probabilistic constraints. A ***non-probabilistic constraint*** is identical to a constraint in a deterministic model. A ***probabilistic constraint*** is based on a statistic of the distribution of the cell's values, such as "the mean of A1 <= 100." RISKOptimizer evaluates this type of constraints at the end of each simulation to decide whether it is satisfied.

Solving Methods

RISKOptimizer supports six different ***solving methods.*** Think of these as different ways to classify how RISKOptimizer will change a group of adjustable cells together. A single optimization model can combine multiple solving methods together.

It's worth noting that the overwhelming majority of problems are solved using the Recipe solving method. This is so common, in fact, that all adjustable cells you add in RISKOptimizer will automatically use this method, and you must go out of your way to choose a different one.

The methods are:

Recipe Method – This is the simplest, and by far the most popular, solving method. It is used when the values in an adjustable cell group can be varied independently of one another. Think of each variable as the amount of an ingredient in a cake. Using the Recipe solving method is like asking RISKOptimizer to find values that produce the best mix.

Order Method – This is the second most popular solving method, used in a very specific context: problems that involve determining the best ordering, or permutation, of items. For example, if there are 10 jobs to schedule on a machine, one after the other, and the desired result is the ordering the results in the least machine downtime.

Grouping Method – This is used for problems that involve multiple items to be grouped into sets. For example, suppose a large number of securities is to be placed into six groups, so that the variability within groups (on some measure such as 6-month return) is minimized. Simply stated, the aggregated securities in each group need to be as "similar" as possible.

Budget Method – This is the same as the Recipe Method, except that sum of the adjustable cell values is fixed at the initial value of the sum. For example, the adjustable cell values might be percentages of a total investment in various stocks. If the initial values of these percentages add to 100% and the Budget method is used, then the percentages will continue to sum to 100% in all trial solutions.

Note: An alternative method would be to use the Recipe method and add a constraint on the sum of the adjustable cell values. These approaches are equivalent if the OptQuest Engine (described later in this document) is used. However, if the Genetic Algorithm Engine is used, the Budget method is significantly more efficient.

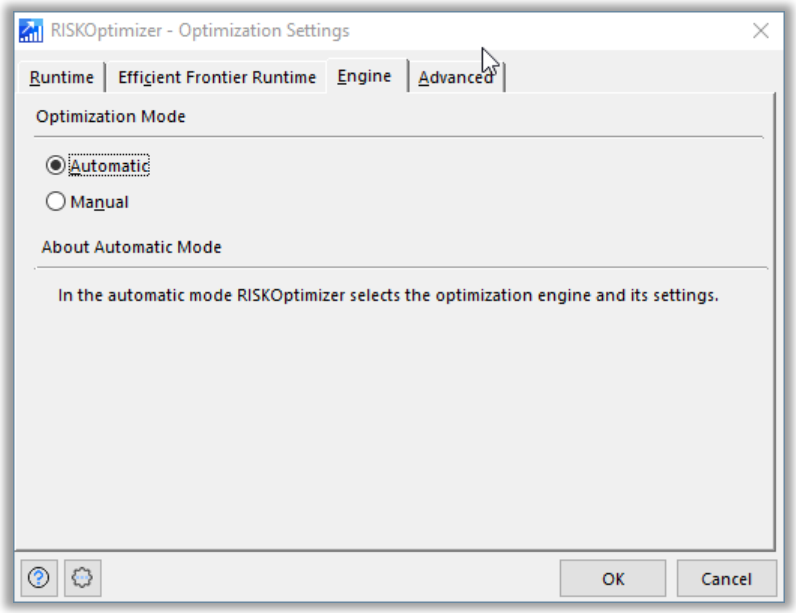
Project Method – This is similar to the Order Method except that certain items (tasks) must precede others. This solving method is especially useful in project management to order tasks with the precedence constraints.

Schedule Method – This is used to schedule tasks to time blocks. Each task is assumed to take the same amount of time, much as classes at a school are all of the same length.

Optimization Engines

The true heart of the optimization process is the step where the optimizer feeds the results of previous trials into a set of sophisticated algorithms to determine the next set of adjustable values to try. In RISKOptimizer, there are two different collections of algorithms, called ***engines***, to choose from: the OptQuest Engine and the Genetic Algorithm (GA) Engine.

For the vast majority of problems, you can let RISKOptimizer automatically choose the appropriate engine for your problem. However, if you want to experiment with the different engines or control them in more detail, you can find options on the Engine tab of the Optimization Settings dialog.



OptQuest Engine

The OptQuest Engine was developed by OptTek Systems, Inc. as a general-purpose optimizer. It uses state-of-the-art metaheuristic procedures, including Tabu Search, Neural Networks, Scatter Search, and Linear/Integer Programming, into a single composite method. To learn more about OptQuest, please visit www.opttek.com

Genetic Algorithm (GA) Engine

RISKOptimizer's GA Engine mimics Darwinian principles of natural selection by creating an environment where hundreds of possible solutions to a problem compete with one another, and only the "fittest" survive. As in biological evolution, each solution can pass along its good "genes" through "offspring" solutions so that the entire population of solutions will continue to evolve better solutions.

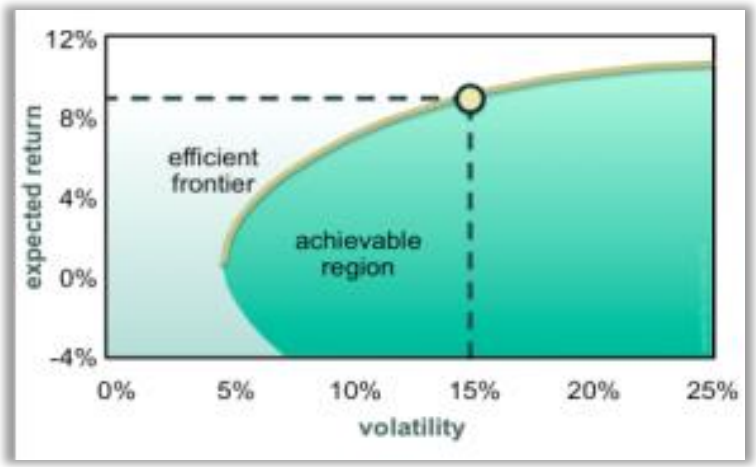
The first genetic algorithms were developed in the early 1970s by John Holland at the University of Michigan. Holland was impressed by the ease in which biological systems could perform tasks that eluded even the most powerful super-computers. For example, animals can flawlessly recognize objects, understand and translate sounds, and generally navigate through a dynamic environment almost instantaneously. Holland's idea was to try to replicate the ideas of genetic inheritance and survival pressure to solve complicated computational problems.

Efficient Frontier Analysis

Efficient frontier analysis is an advanced, special kind of optimization. It is used when there are two competing goals. One of them is chosen as the target for an optimization, while the other is constrained to be no worse than a specified limit. Then a number of optimizations is performed, each time changing the limit value for the constraint.

Although efficient frontier analysis is most commonly used for financial portfolio optimization, it can be applied to a variety of problems where there are two competing goals. Determining a waste management system that minimizes cost and minimizes the pollution level would be one example. Because these goals are pulling in opposite directions, the optimization model might be configured to minimize cost while putting various upper limits on the allowable pollution level. Alternatively, the same model could be reconfigured to minimize the pollution level, while putting various upper limits on the allowable cost.

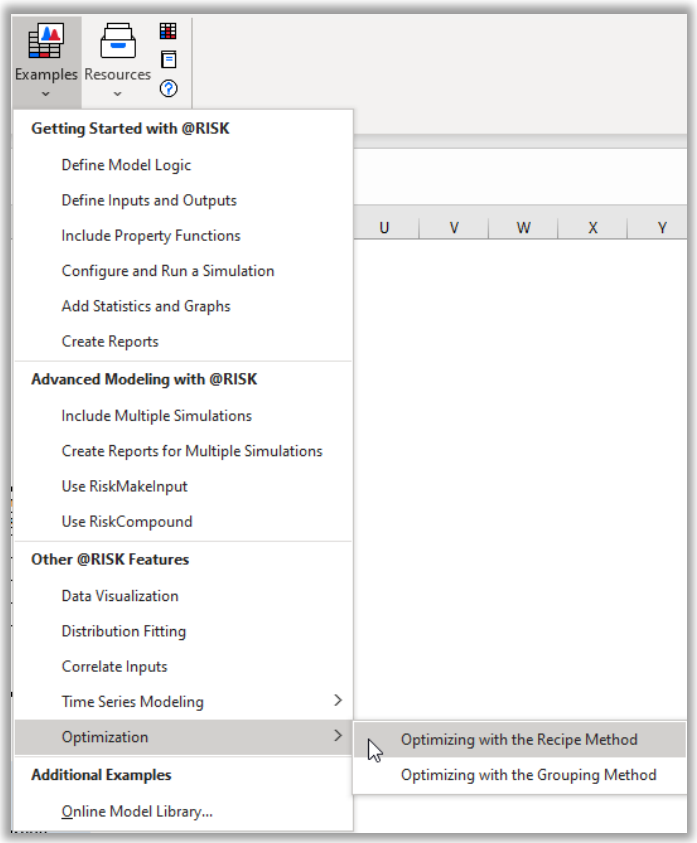
For illustration, however, this discussion will be in usual context of portfolio optimization. In this context, the goal is to find a portfolio of securities that maximizes expected portfolio return and minimizes risk, usually measured by the standard deviation of portfolio return. As in the pollution example, these two goals pull in opposite directions, so efficient frontier analysis typically minimizes the risk (standard deviation), while putting a lower bound on the expected return. (It could be performed in the opposite way, by maximizing expected return with an upper bound on risk.) By performing several optimizations, each with a different lower bound on the expected return, the efficient frontier is mapped out. It indicates, for any given required expected return, the portfolio that minimizes the risk. Then from all the portfolios on the efficient frontier, one can choose a preferred portfolio given the risk preferences.



The graph shown left illustrates an efficient frontier, the yellow curve. Optimal portfolios can be found on this curve, and these are the only portfolios an investor should consider. Points in the darker green area below and to the right of the efficient frontier correspond to portfolios that have more risk than necessary for any specified expected return, so they are suboptimal and shouldn't be considered. Points in the lighter green area above and to the left of the efficient frontier are impossible to achieve.

Configuring a RISKOptimizer Model

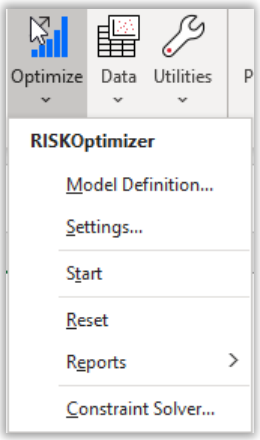
Now that we've covered some terminology and theory associated with optimization, let's see how @RISK's RISKOptimizer feature implements these ideas. The example model "Optimizing with the Recipe Method" that is included with @RISK is an excellent way to learn how to set up an optimization model. The model can be easily found by opening the Examples menu on the @RISK ribbon and selecting it from the Optimization submenu.

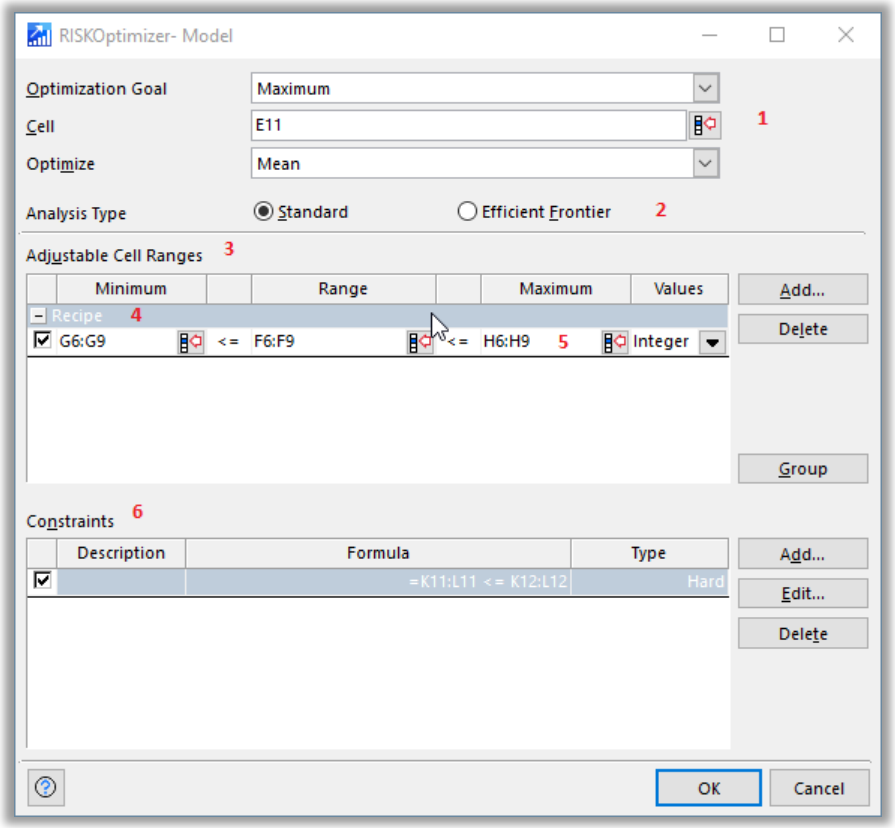


The model contains a detailed description of its design, which won't be repeated here. However, it is worth examining where the concepts described earlier in this document come into play in the software's interface. Let's start by looking at some items in the Optimize menu on the @RISK ribbon.

Model Definition

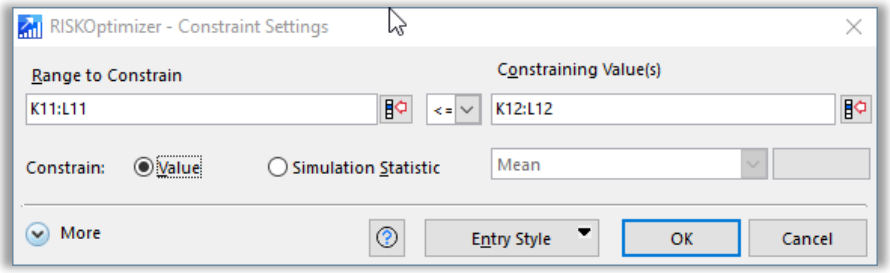
With the example model open in Excel and @RISK, drop down the Optimize menu from the @RISK ribbon and choose the Model Definition item. The image below shows that dialog for this model, with some annotations in red discussed below.





1. **Target Definition –** here is where the optimization target is defined. In this case the target is to maximize the mean of cell E11.
2. **Analysis Type –** here is where you can specify whether or not to run an efficient frontier optimization. In addition to changing this setting, to run an efficient frontier analysis you must also declare a constraint as an efficient frontier constraint and specify a range of values corresponding to different versions of the constraint.
3. **Adjustable Cell Ranges –** here is where the list of adjustable cells is declared. Many models, like this one, will have a single entry in this table although you can have as many as you need.
4. **Solving Method –** this shows you that everything listed is using the Recipe solving method. Any cells you add will always use this method unless you choose to change it via the Group menu.
5. **Adjustable Cells Row –** here there is a range of four cells in the block F6:F9 that are the adjustable values in the model. They have minimum and maximum allowed values specified in the ranges G6:G9 and H6:H9, respectively.
6. **Constraints –** there is an additional constraint (really a pair of constraints for cells K11 and L11) defined here.

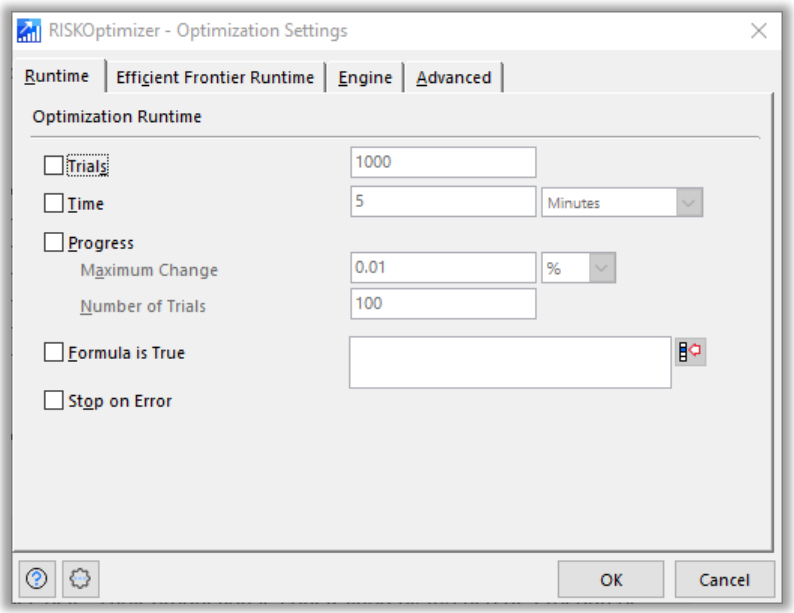
For the constraint, more information can be specified by clicking on the Edit button to bring up the Constraint Settings dialog. For this particular constraint, that dialog looks like this:



This relatively simple definition says that cell K11 must be less than on equal to K12 and cell L11 must be less than or equal to L12. Note, both of these constraints are not stochastic. That is, they do not require a simulation statistic to be calculated in order to be evaluated.

RISKOptimizer Settings

Now choose the second item on the @RISK ribbon's Optimize menu to show the RISKOptimizer settings dialog.



The first tab (for regular optimizations) or the second tab (for efficient frontier optimizations) specify conditions under which an optimization will automatically stop. If RISKOptimizer can determine *theoretically* that is has reached the absolute optimal solution, it will always stop. However, most problems are not amenable to that kind of analysis, and thus you are able to specify additional, external stopping conditions.

It is completely acceptable, as is done here, to not specify *any* additional condition, in which case it is up to you to terminate the optimization when you don't want to keep exploring more incremental improvements. However, stopping an optimization after a fixed number of trials, some fixed amount of time, or after the optimizer has entered the realm of diminishing returns is also quite common.

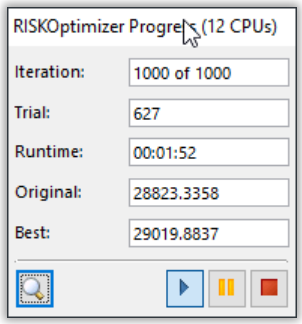
The “Engine” tab contains settings for controlling which optimization engine should be used and any relevant parameters to control it. In general, it isn’t necessary to change those settings.

Simulation Settings

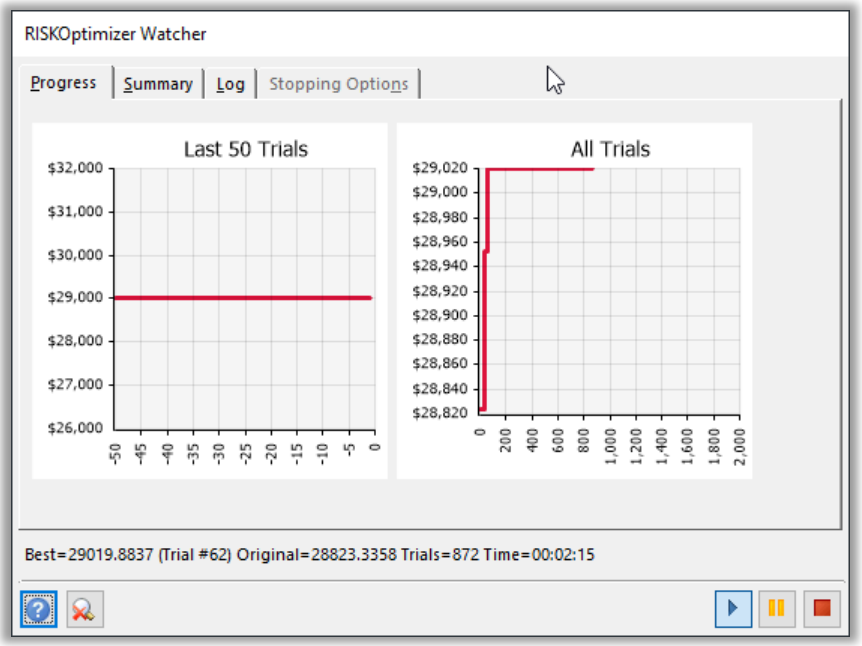
It is worth noting that since each trial solution in RISKOptimizer runs a full simulation, many of the standard @RISK simulation settings are still relevant when optimizing. Most important of these, of course, is how many iterations to run. This can be specified directly on the @RISK ribbon, or by bringing up the full simulation settings dialog.

Running an Optimization

With the model configured for RISKOptimizer, running the optimization is simple. Just click the Start command on the Optimize menu. Once the initialization period has completed, you can watch the optimization improve your solution in the small progress window that appears:



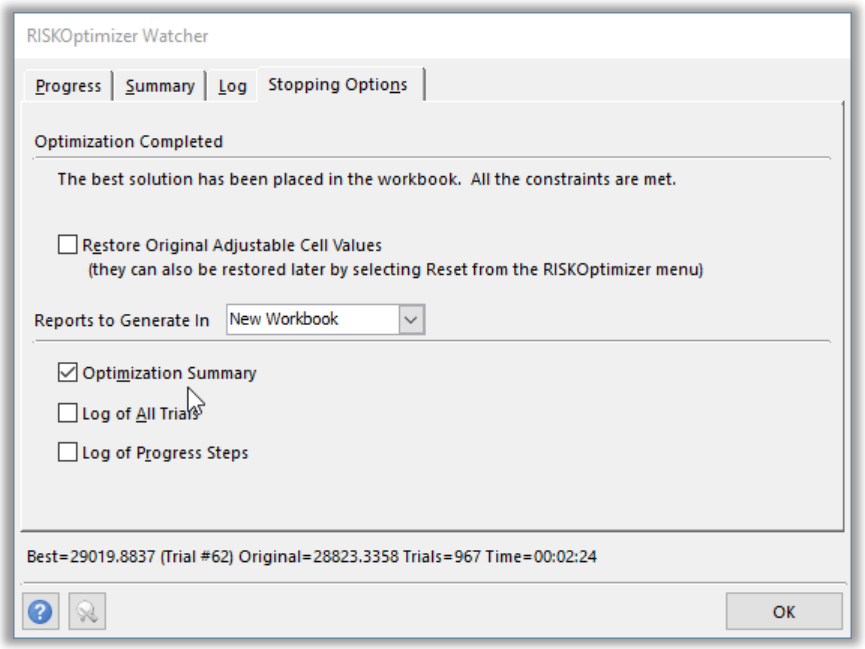
For more comprehensive details, clicking the button in the lower-left of the progress window will display the RISKOptimizer “Watcher” window where you can find, among other things, graphs showing how the best solution found has changed as the optimization has proceeded.



This particular model has no stopping conditions defined (and is not a model where RISKOptimizer can decide if it has reached the theoretical best value) thus it will only stop when you click the stop button in the lower right of the progress window (or of the Watcher window.)

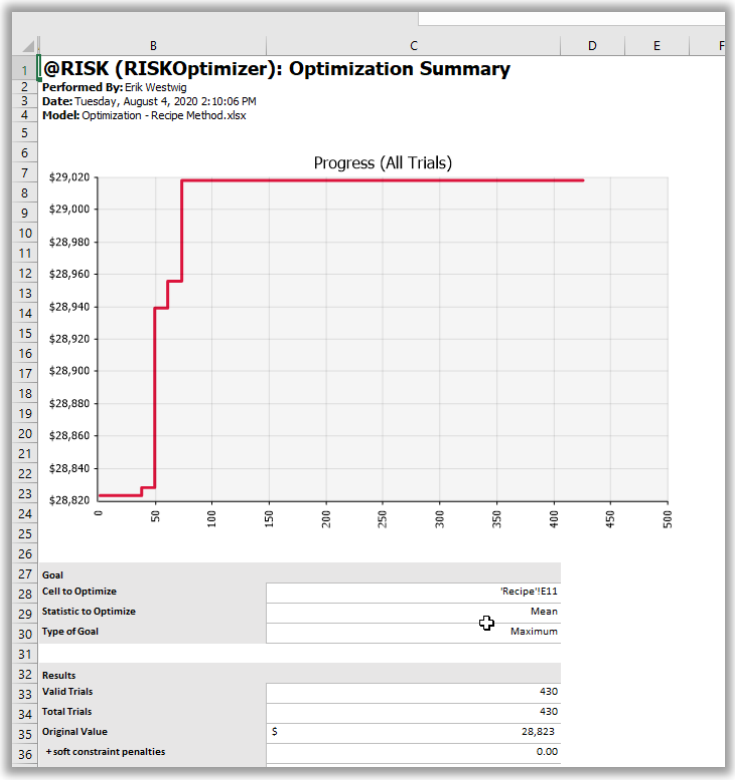
Analyzing the Results of an Optimization

When RISKOptimizer stops, it displays a dialog:



Normally, when you stop an optimization, you want RISKOptimizer to automatically put the best values it found for each of the adjustable cells directly into the spreadsheet. However, that step can by bypassed, instead restoring the original values.

Several different reports can be requested from this dialog. (These same reports can be generated after this dialog is closed directly from the Optimizer menu on the @RISK ribbon.) The most useful report is the Optimization Summary Report, which in addition to showing a graph of the optimization's progress, also includes tables showing the best value found for each adjustable cell.



Conclusion

As you can see, RISKOptimizer is a powerful feature of @RISK. Our hope is that this document has been a useful introduction to stochastic optimization and now you can get started using it with your own models!