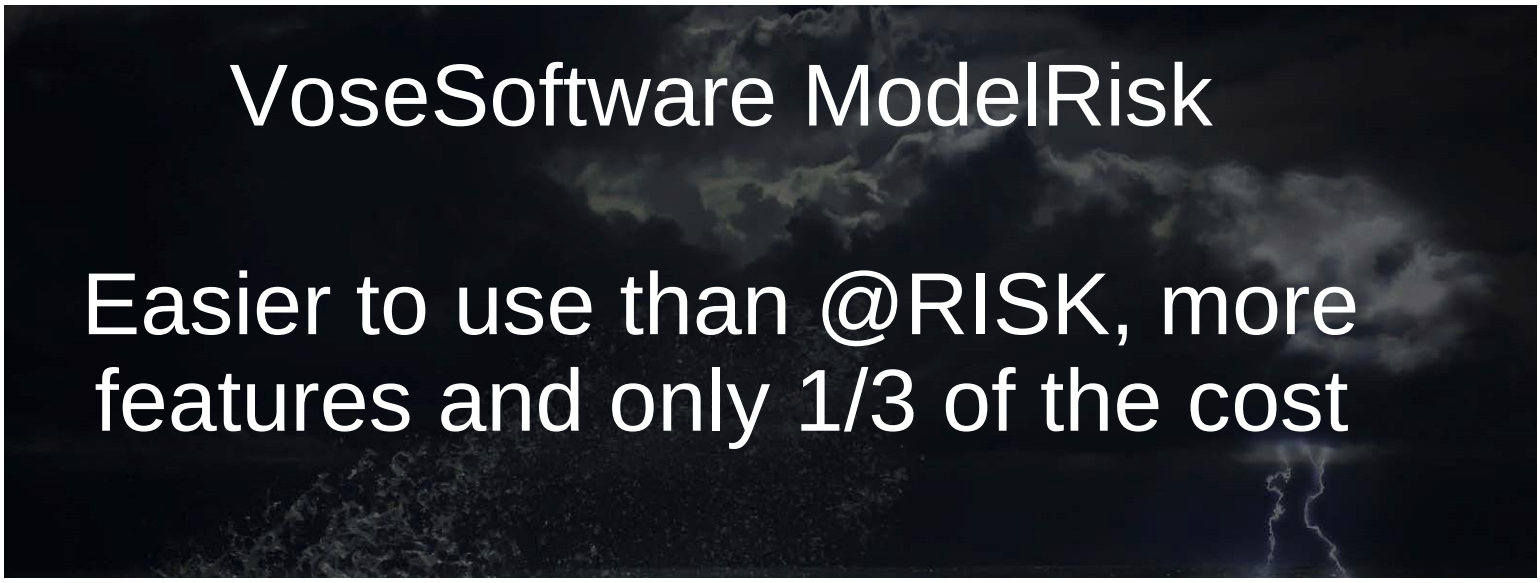


A wide range of problems are typically evaluated using Microsoft Excel. These include financial models, sales forecasts and cost estimates, but there are many other types of problems for which a spreadsheet modeling environment is also suited. An Excel model will only provide a single estimate of the outcome being modeled so it is common practice to use a risk analysis Excel add-in that performs a **Monte Carlo simulation** to assess the uncertainty around the results.

There are a number of risk analysis Excel add-ins available, but only two commercial products have a large and comparable set of capabilities, namely:

- **@RISK from Palisade Corporation**
- **ModelRisk from Vose Software**

Palisade Corporation develops add-ins for Excel, including BigPicture - a diagramming software add-in, Evolver – an optimizer add-in, PrecisionTree – a decision tree drawing add-in, StatTools – a basic statistical analysis add-in, and NeuralTools – a basic neural network add-in.

Vose Software develops a range of integrated risk analysis software products that operate in different environments. They include **Pelican** – a web-based enterprise risk management system, **Tamara** – a stand-alone project risk analysis tool, **StopRisk** – an operation risk analysis tool for banks, and **ModelRisk Cloud** – a web-based system for deploying risk analysis models.

SIMILARITIES

Both @RISK and ModelRisk come in three editions - Standard, Professional and Industrial. The Professional

editions are the most similar, and are therefore compared here.

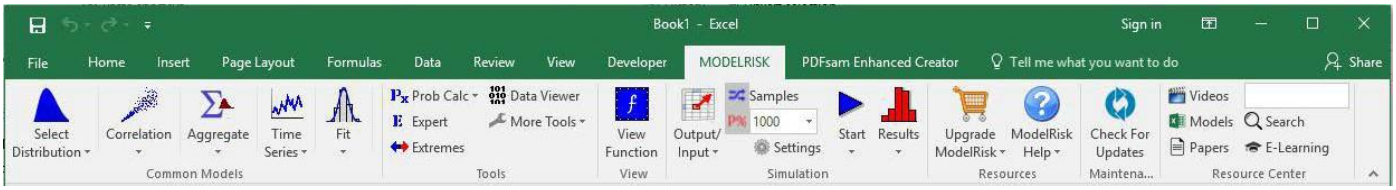
Both products use Excel-style functions to generate random values to describe the uncertainty about values in the mode. For example:

@RISK: =RiskTriang(1,2,5)

ModelRisk: =VoseTriangle(1,2,5)

perform the same operation of randomly sampling from Triangle distribution.

Both products add a ribbon to Excel from which most functionality can be accessed. The ModelRisk Pro ribbon looks like this:



During basic simulation runs, the products will run at about the same speed unless you have more than 2 CPUs on your computer (@RISK only supports 3+ CPUs in their Industrial version). They both allow an unlimited number of samples, include the ability to set seed values, offer live updating during simulation and running multiple simulations together, and use the same default random number generator.

After recent @RISK releases, both products now use the exact same methods for correlation, distribution fitting, distribution splicing, and offer the same range of graphical reports.

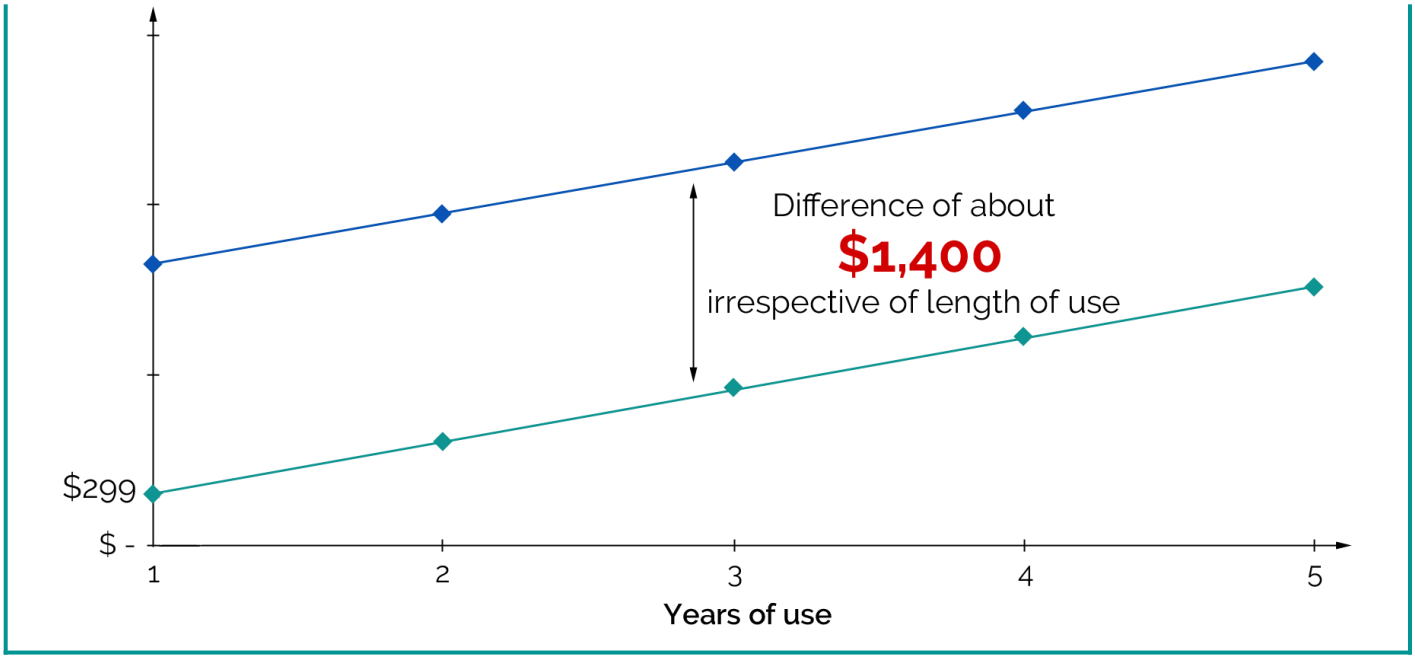
Both products have a converter for Crystal Ball models. ModelRisk also provides a converter for @RISK models.

Differences

Did you know?

ModelRisk costs about 1/3 the price of @RISK,

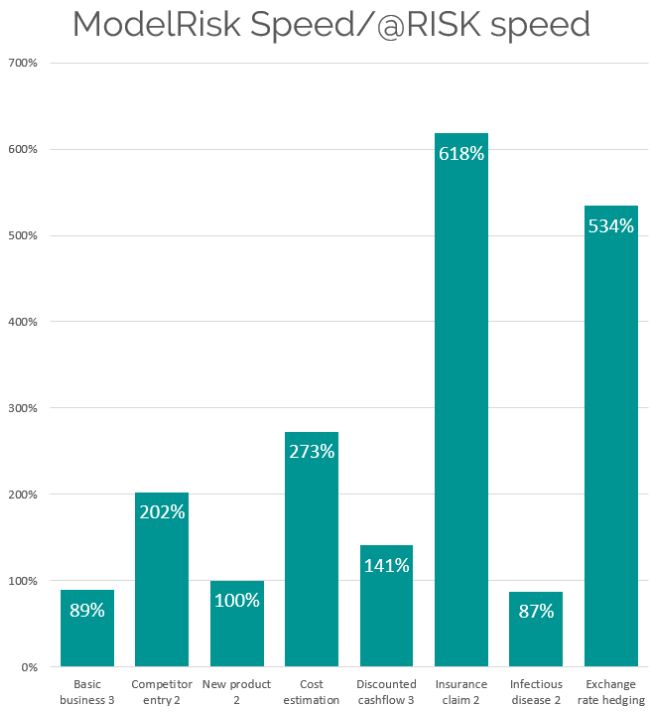
but there are other, even more interesting differences than the cost saving!



Aside from the dramatically larger cost of @RISK, there are also a number of other important differences.

Speed

Simulation experiments were conducted using @RISK example models and equivalent ModelRisk models with no change in logic. At worst, ModelRisk achieved 87% of @RISK's speed and at best 618% (i.e. six times faster). HOWEVER, please be aware that this experiment was conducted using the trial version of @RISK which uses all 8 of the CPUs in the test computer, whereas the Professional edition is limited to using just 2 CPUs, so would presumably run at a quarter of the speed produced in these tests.



Distributions

ModelRisk provides about three times as many distributions as @RISK.

Tools

ModelRisk offers a wide range of additional tools that simplify model building and allow the creation of vastly smaller and faster models - for example,

StopSum, SumProduct, the Extremes toolset, and Combined.

Languages

@RISK is available in several languages. ModelRisk is currently only available in English.

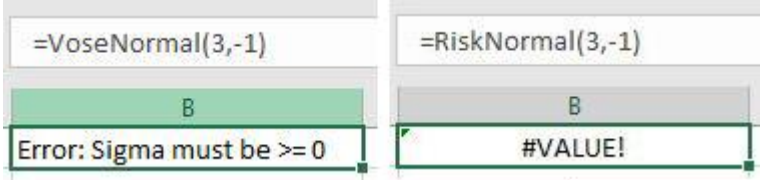
Ease of use

ModelRisk has a much simpler method for combining functions. For example, the following formulae perform exactly the same tasks:

	A	B	C	J
1				
2		Input distribution		
3		ModelRisk	=VoseInput("Cost","$")+VosePERT(5,10,20)	
4		@RISK	=RiskMakeInput(RiskPert(5,10,20,RiskName("Cost"),RiskUnits("$")))	
5				
6		Fitted distribution		
7		ModelRisk	=VoseLognormalFit(I3:I231)	
8		@RISK	=RiskFitDistribution(I3:I231,1,"Lognorm")	
9				
10		Correlated distribution		
11		ModelRisk	=VosePERT(1,2,4,E2)	
12		@RISK	=RiskPert(1,2,4,RiskCopula(Copula1,1))	
13				
14		GBM time series		
15		ModelRisk	=VoseTimeGBM(0.002,0.025,100,)	
16		@RISK	=RiskGBM(0.0002,0.025,RiskTsTransform(1,0),RiskTsIntegrate(1,0),RiskTsSync(2,100))	
17				

Checking for errors

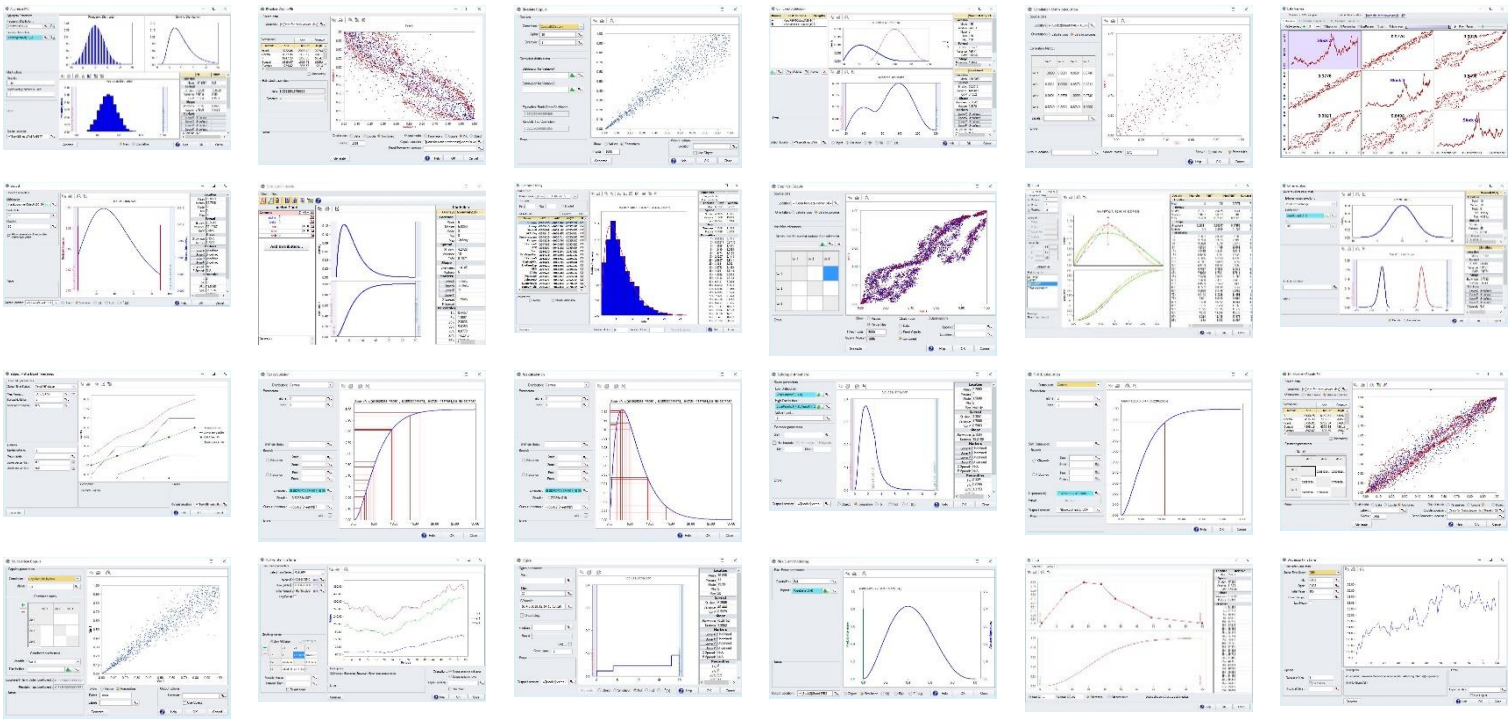
Whenever an invalid value is entered for a ModelRisk function, it returns an informative error message, whilst @RISK doesn't. For example:



Visual interfaces

ModelRisk Pro also has a vast number of visual interfaces to help you (ModelRisk Industrial has even more). Select a cell with a ModelRisk function, click the View Function icon, and the interface for that function will pop

up, showing you precisely what the function is doing.



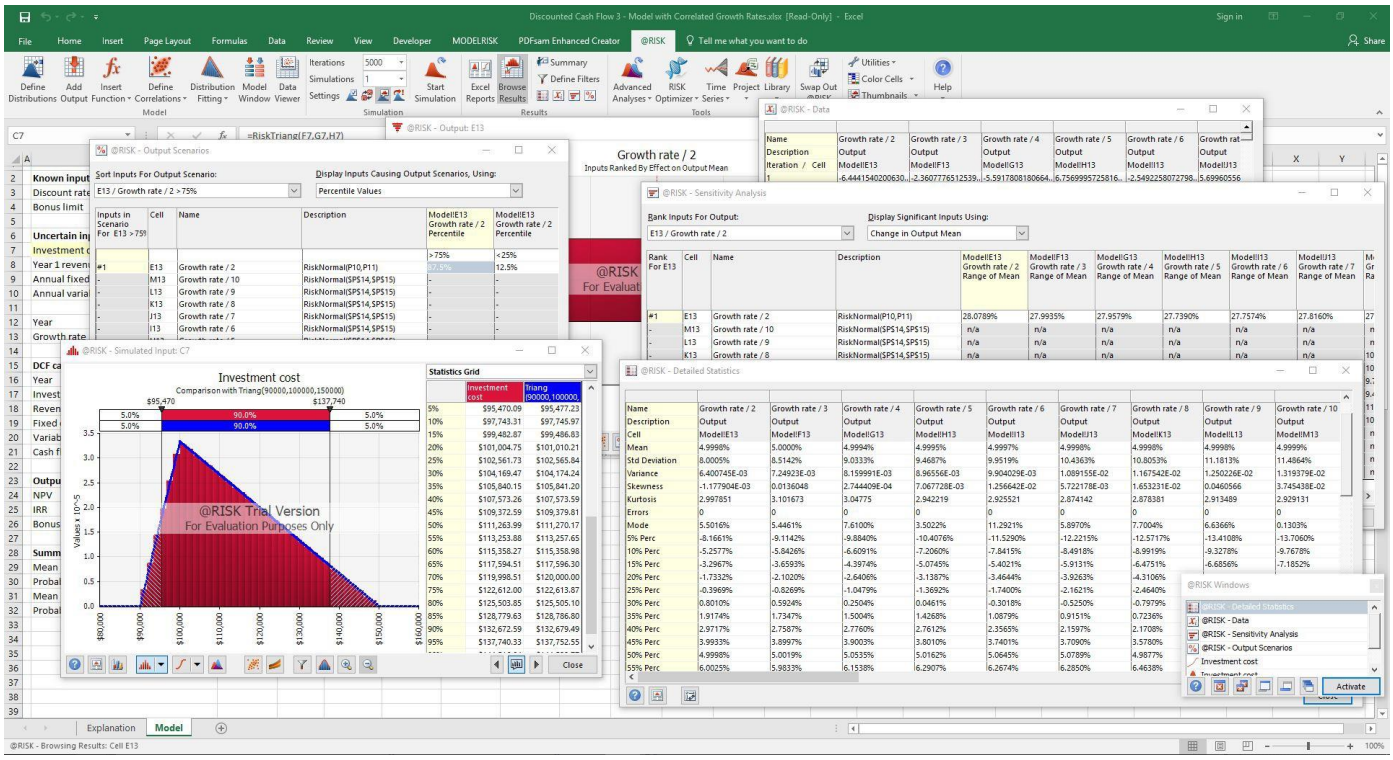
Project schedule risk

@RISK incorporates the ability to simulate uncertainty in a schedule built in Microsoft Project, by building a duplicate of the Gantt chart in Excel and swapping simulation data between Excel and Project. Vose Software, in contrast, offers a separate product called Tamara that imports MS project or Primavera files, and which share simulation results with ModelRisk. Tamara can handle schedules of almost any size and runs several hundred times faster than @RISK.

Reporting

The method of displaying and sharing reports is completely different. ModelRisk uses the Results Viewer, which contains all the simulation results within one location, and where a new tab is created for each new graph or table required. ModelRisk's Results Viewer remembers all graphs and tables that have been created, and will reproduce the same report the next time the model is run, even if that model is run on another computer.

With @RISK, the same reporting is divided between many different interfaces, and plotting graphs rapidly becomes a confusion of separate windows:



Sharing results

The Results Viewer can also be used independently of ModelRisk, so a modeler can send a colleague a ModelRisk results file instead of the colleague needing a copy of ModelRisk and running it themself. The Results Viewer also allows the user to export the entire report in one go to PDF, PowerPoint, Word or Excel. With @RISK, one either has to create a report in Excel or create graphs one at a time, copy each graph as a bitmap, and then paste into another document.

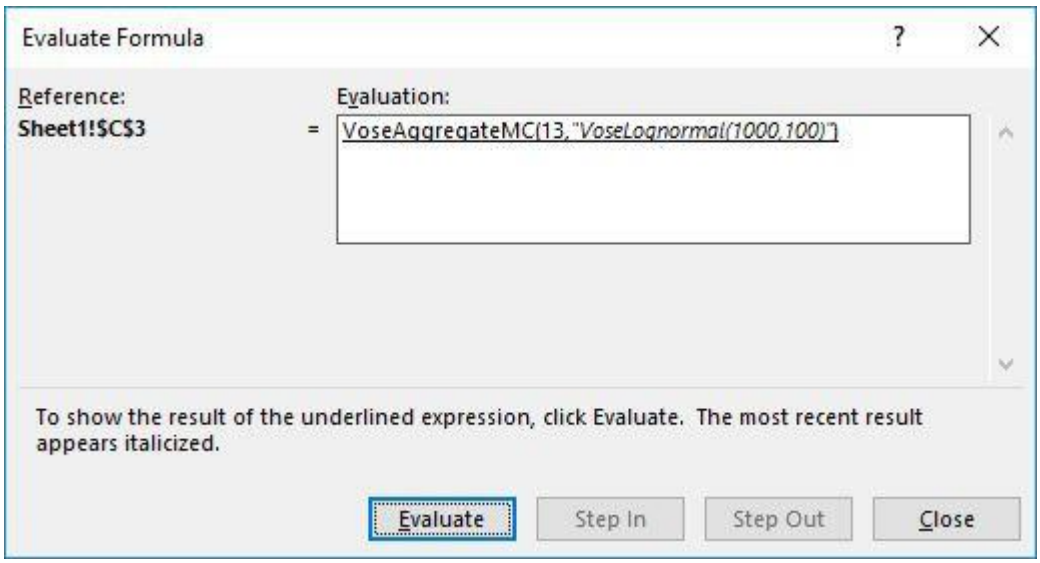
Compatibility with Excel's rules

ModelRisk offers a unique ability we call 'Objects', giving the user the ability to define the random variable that is to be used. Objects can be used with many more advanced ModelRisk functions, often greatly simplify a model, and ensure that we can offer powerful functions that are consistent with Excel's evaluation process.

For example, the VoseAggregateMC(x,y) function allows one to add up x independent samples drawn from the distribution y. To specify y, you use a distribution object function. @RISK made a version of the AggregateMC function called RiskCompound(x,y) but where y is the same type of function ussually used for sampling from a distribution. Side-by-side, they look like this:

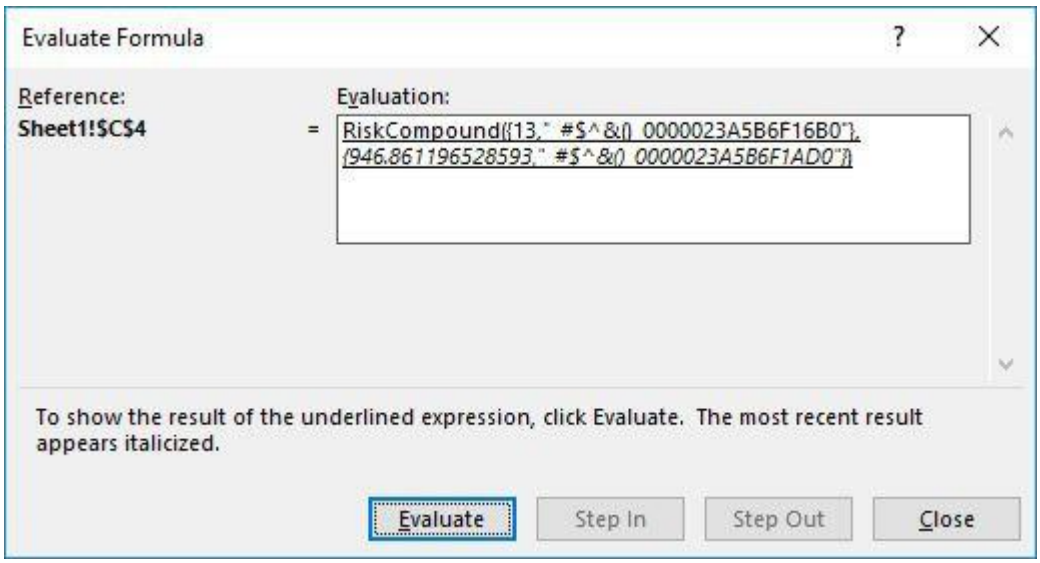
	A	B	C	D
1				
2		Compound (Aggregate) distribution		
3		ModelRisk	=VoseAggregateMC(VosePoisson(10),VoseLognormalObject(1000,100))	
4		@RISK	=RiskCompound(RiskPoisson(10),RiskLognorm(1000,100))	
5				

When Excel evaluates the ModelRisk function, its internal calculations see this before the last step of returning a value:



The VoseLognormalObject function has returned a text result describing the distribution which the VoseAggregateMC will then take (in this case 7, the result of evaluating the VosePoisson function) samples from.

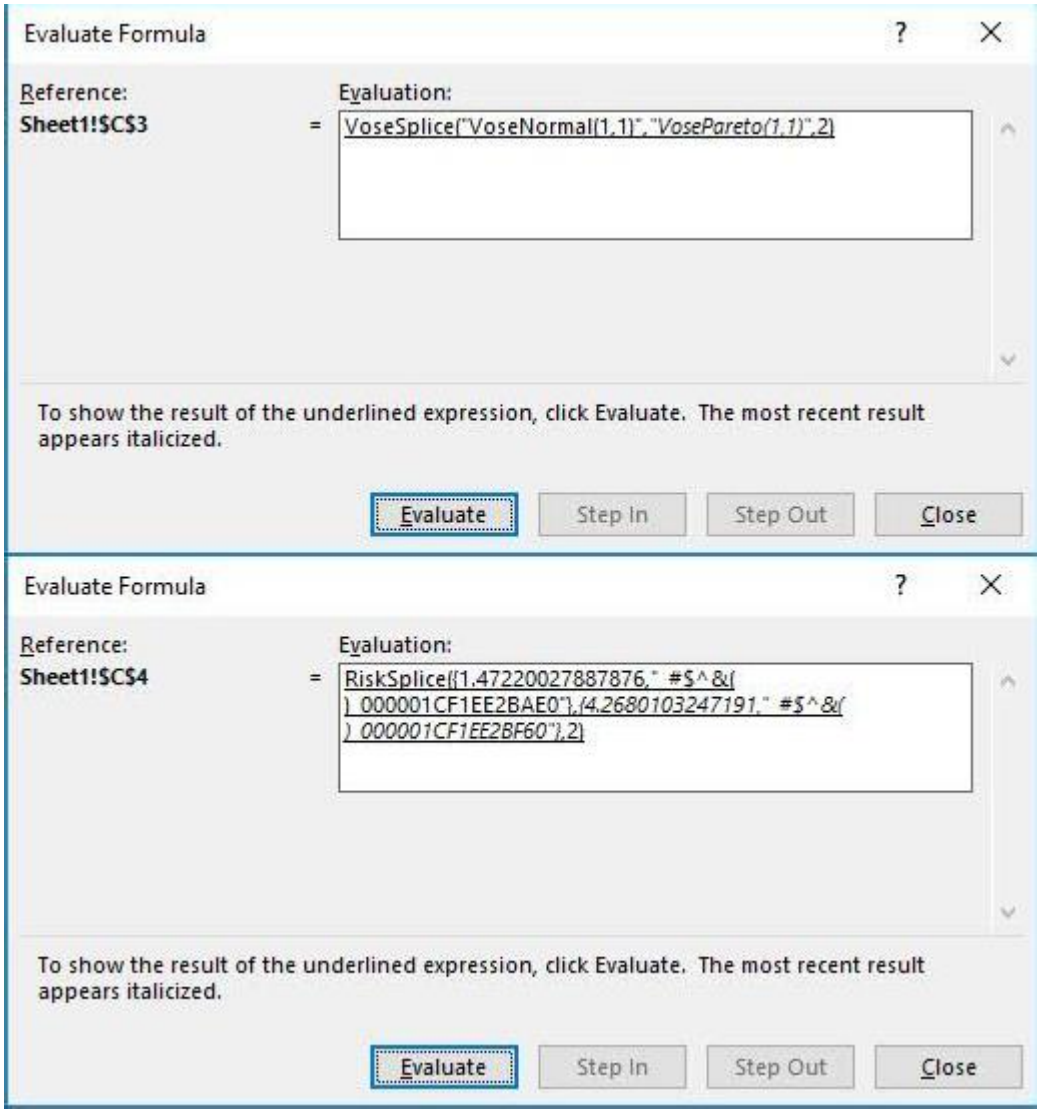
Now look what happens with the @RISK function:



The problem is not unique to one function, it occurs everywhere @RISK needs to define a distribution. For example, splicing two distributions, the formulae look like this:

	Splicing distributions	
	ModelRisk	=VoseSplice(VoseNormalObject(1,1),VoseParetoObject(1,1),2)
	@RISK	=RiskSplice(RiskNormal(1,1),RiskPareto(1,1),2)

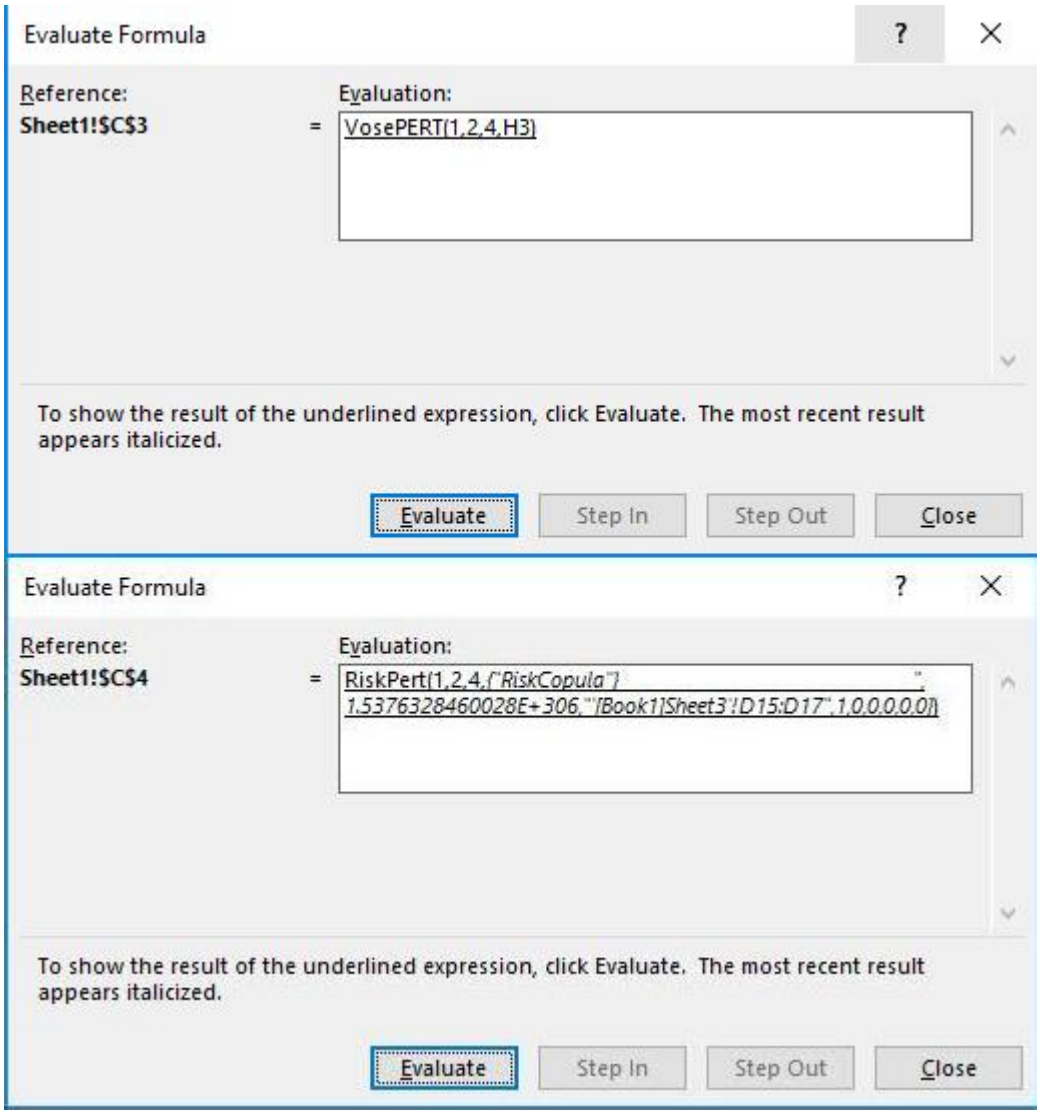
and just before returning a value, Excel has evaluated them like this:



Every time @RISK correlates variables it must do something similar. For example, these two functions sample from the same correlated distribution:

		Correlated distribution	
		ModelRisk	=VosePERT(1,2,4,H3)
		@RISK	=RiskPert(1,2,4,RiskCopula(Copula1,1))

The last step in Excel's evaluation before returning the random sample looks like this:



Note that the ModelRisk evaluation occurs in a single step, which apart from being transparent also has the benefit of being much faster.

@RISK has had to force Excel to behave in a completely different way - the two parameters are not evaluated in the usual way as required by Excel, but instead each is converted into a complex array of indecipherable code. Since it is no longer obeying Excel's rules for function evaluation, the user can no longer be sure of what the function is doing.

출처: https://www.vosesoftware.com/products/CompareToAtRisk/