Intro to Monte Carlo Simulation Lab

Rolling Dice using @RISK

v1.0 by Andrew Lucchesi, Spring 2013

Section 1: Intro to Monte Carlo Simulation and @RISK

Summary – The core idea of Monte Carlo methods is to use random samples of parameters (or inputs) to explore the behavior of a complex system or process. These simulations are ideal for situations involving uncertainty in many dimensions that are too complex for an analytical solution.

> "Monte Carlo simulation performs risk analysis by building models of possible results by substituting a range of values—a *probability distribution*—for any factor that has inherent uncertainty. It then calculates results over and over, each time using a different set of random values from the probability functions. Depending upon the number of uncertainties and the ranges specified for them, a Monte Carlo simulation could involve thousands or tens of thousands of recalculations before it is complete. Monte Carlo simulation produces distributions of possible outcome values. " -Palisade

For a better idea on how the Monte Carlo Method compares to using a Point Estimate, watch the video below:

> http://www.youtube.com/watch?v=rWoarUe7shs

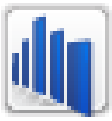
@RISK is an Excel plug-in that allows you to create your model in a spreadsheet, running the simulation through the plug-in. There are other Monte Carlo Excel plug-ins, which are similar to @RISK as well as separate software all together for Monte Carlo simulation, however, you can download @RISK for a free 2 week trial, and get a student version for only $50:

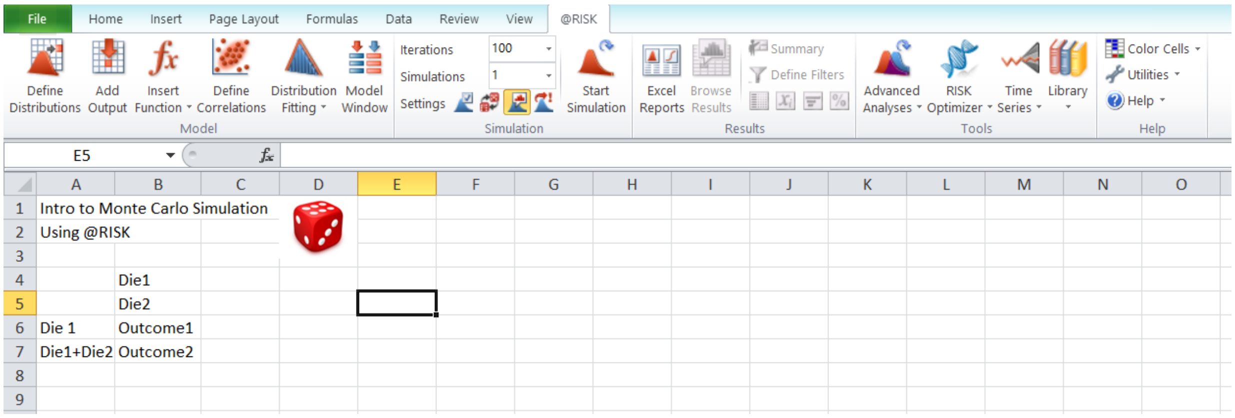
> http://www.palisade.com/academic/students.asp

Section 2: Rolling a die with @RISK

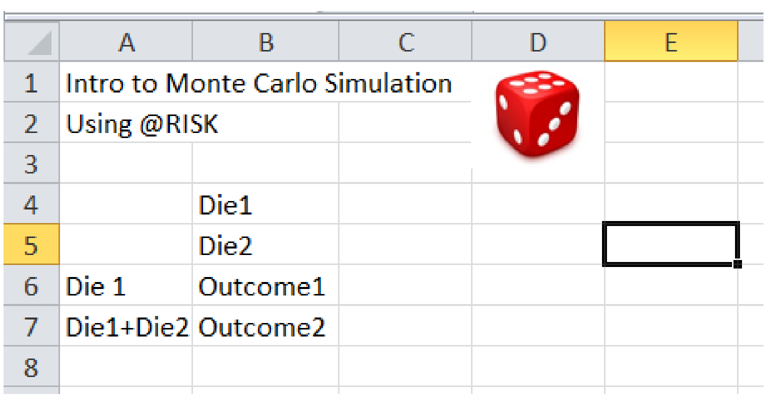
In this beginner lab we will be using a tool for Monte Carlo called @RISK. We're going to roll a die multiple times and look at the distribution of the outcomes from rolling that die. There are 6 sides, all equally likely to occur. That makes this an integer with a uniform distribution. This will hopefully make you feel a little more comfortable managing the @RISK ribbon and its functionalities.

<u>Step 1</u>: Launch @RISK . Clicking on the icon (shown below) will bring up a spreadsheet in Excel with the ribbon loaded (also below).

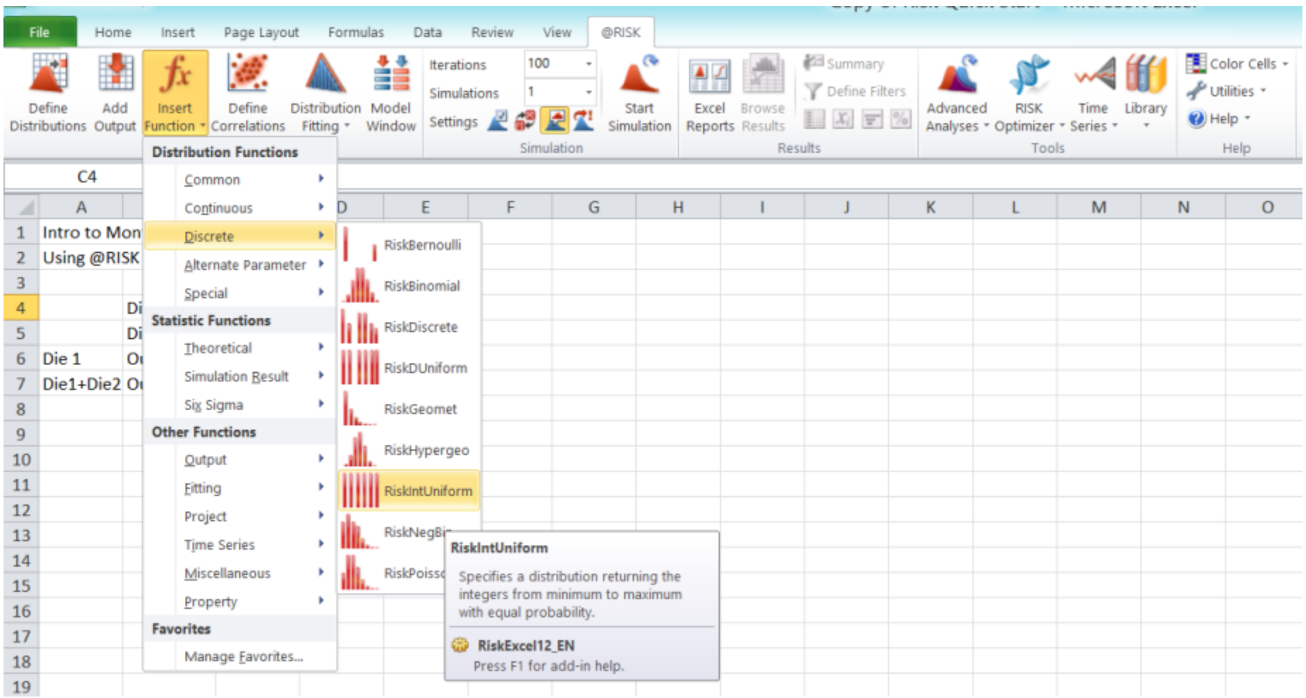




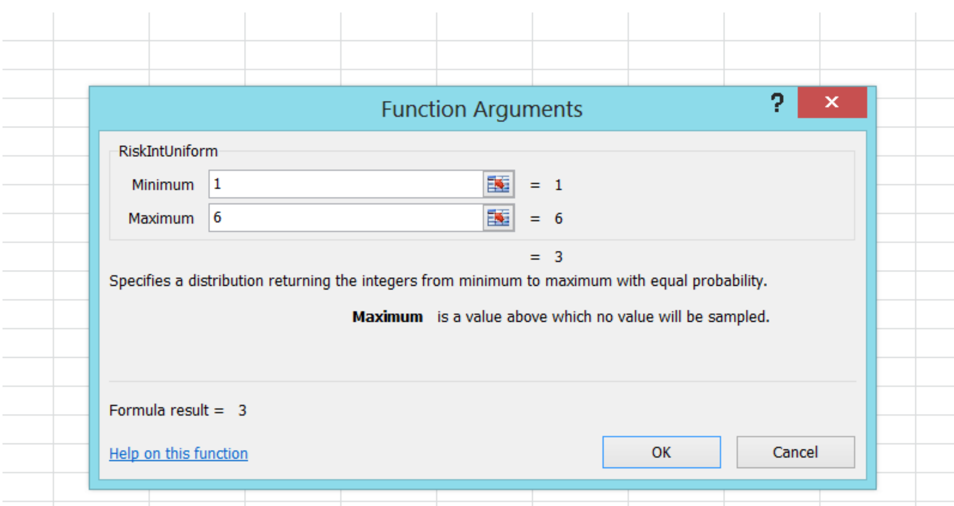
<u>Step 2</u>: Construct your spreadsheet. Fill out the spreadsheet like mine (below). This is the shell of our model.



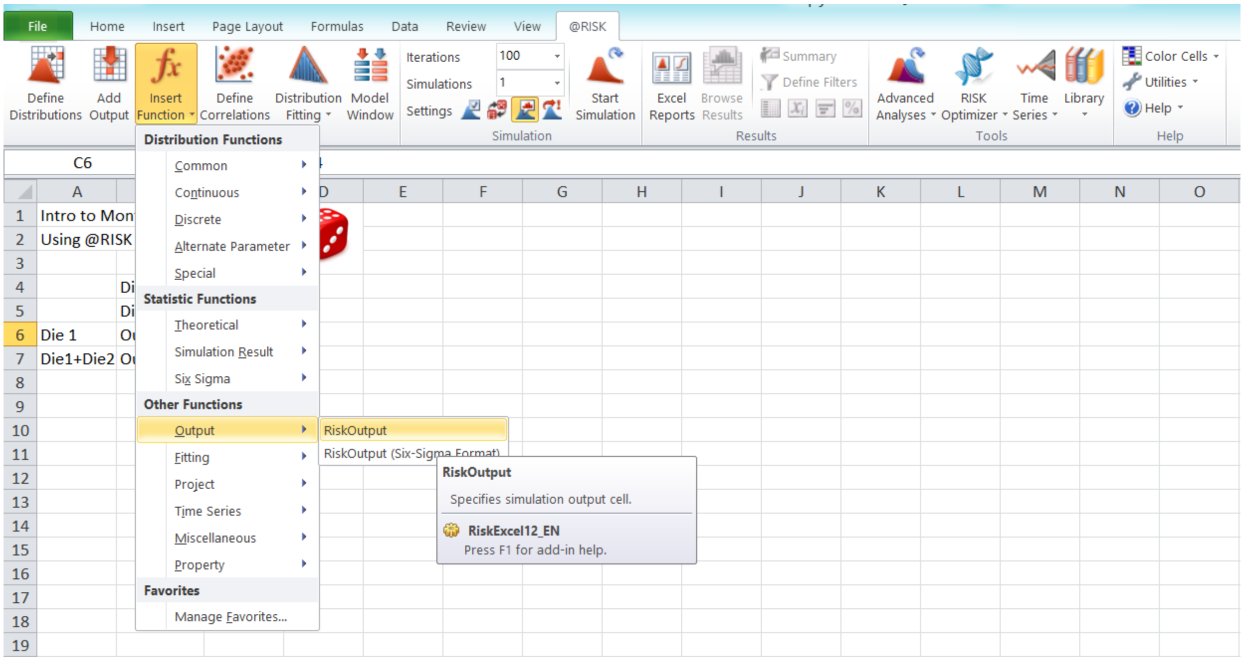
Step 3: Create the first distribution. Go to the "Insert Function" dropdown. We want a discrete distribution. For this (while the box next to Die1 is selected) select "Discrete" and then "RiskInitUniform" on the next menu down. If you hover over the different options it will give you a short summary on what they're used for. (Information about a few of these distributions and their uses can be found at http://www.palisade.com/risk/monte_carlo_simulation.asp.)



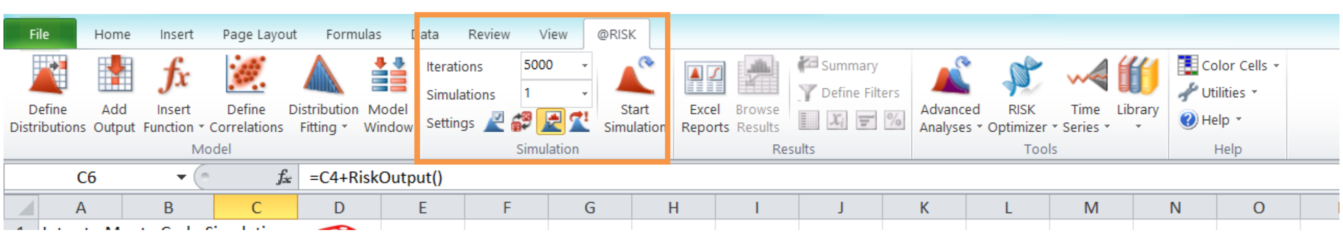
Step 4: Choose parameters.The minimum will be 1 and the maximum will be 6 (remember it's a die we're modeling). You can name the box if you choose. Notice the formula that is now in the cell is "=RiskIntUniform(1, 6)".



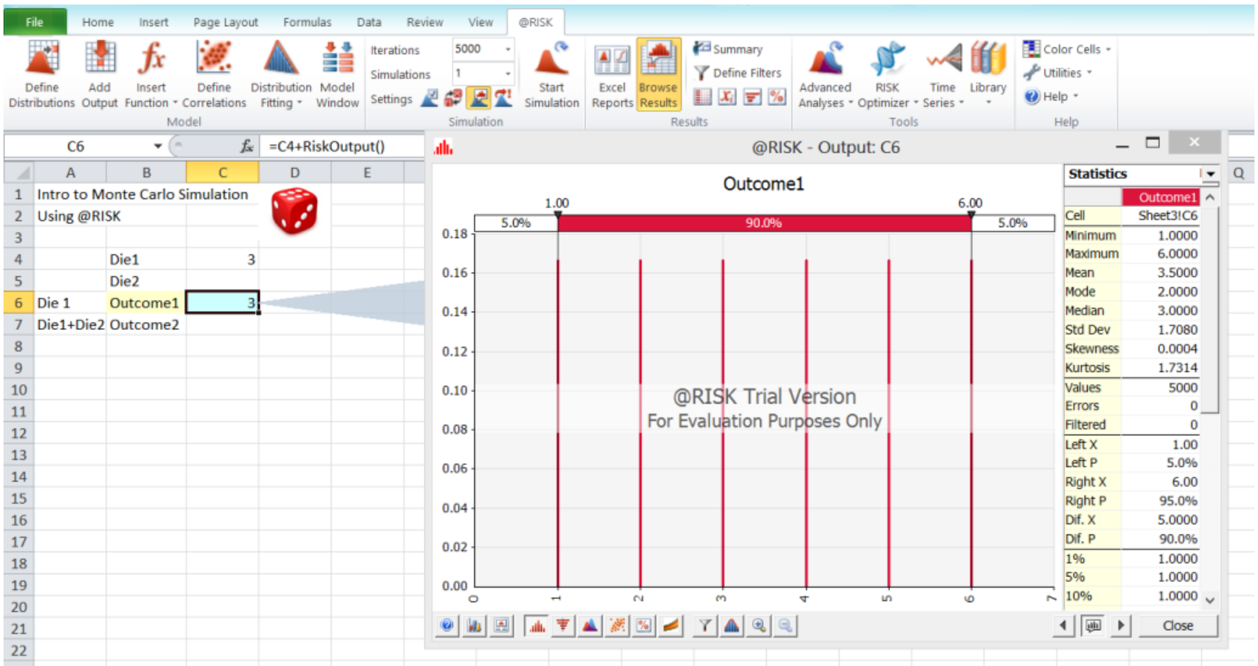
Step 5: Identify the outcome cell. For this first example it will just be the result from rolling that one die. Select the cell next to “Outcome1” and enter (in my case) “=C4” *or whatever cell that holds the distribution you made earlier for Die1*. With the cell selected go to the “Insert Function” dropdown and select "Output" -> "RiskOutput." A Functions Arguments box will come up that asks if youwould like to enter in a Name, Range name, and position in range. (These are optional, so I’m leaving them blank and selecting OK.)



Step 6: Choose your iterations. Your box should now have “=yourcell(for me C4)+RiskOutput()” Finally, let's change the iterations (or number of times we will simulate rolling the die) to 5000 from the dropdown box .



Step 7: Simulate! Select “Start Simulation” and it will run the simulation once (with 5000 rolls, or *replications*). A window should pop up next to the outcome cell displaying them uniformly between 1 and 6. Next to the graph, on the side bar, all sorts of good data will be there, including the mean, standard deviation, variance, and so forth.



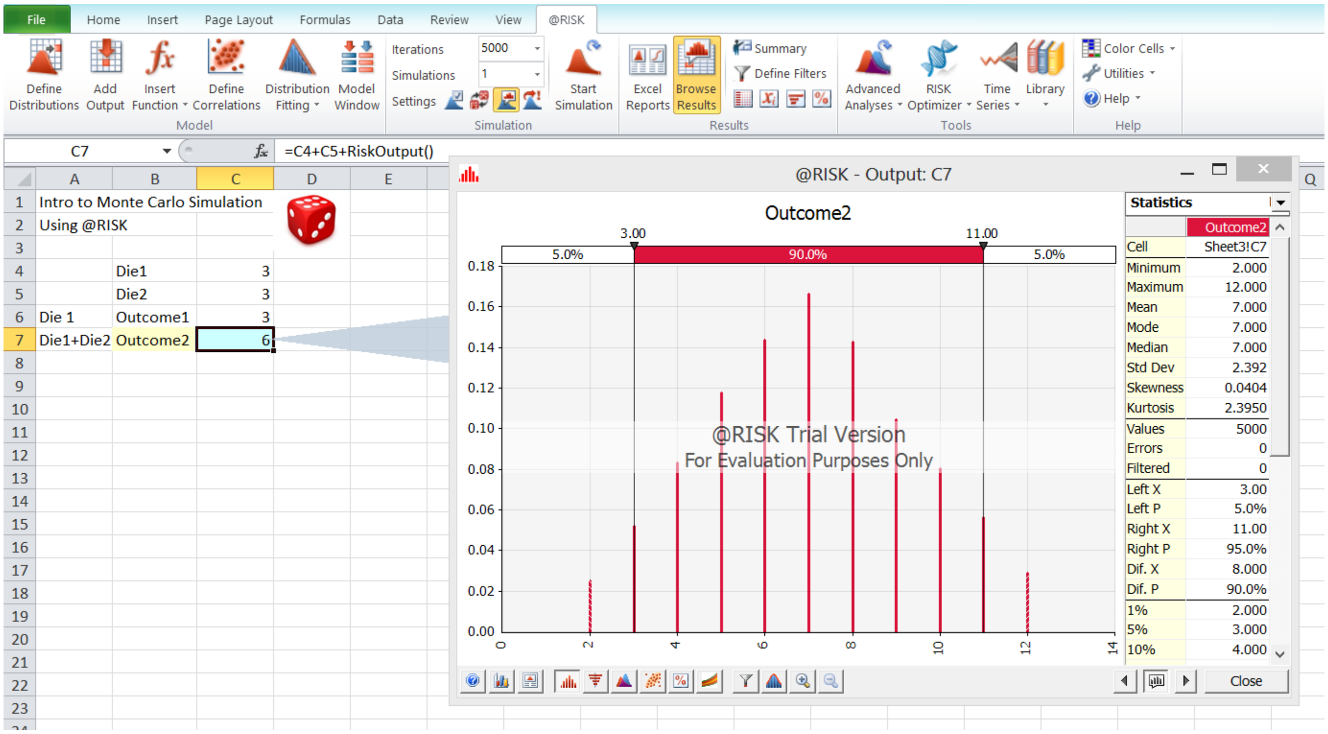
Section 3: Rolling 2 die

Now what if we want to roll 2 die, add the results, and see what that outcome would look like?

Step 1: Create the same properties for Die 2 as we did previously for die 1. To do that let's just drag Die 1 to Die 2 to copy it (it will automatically rename it for that cell).

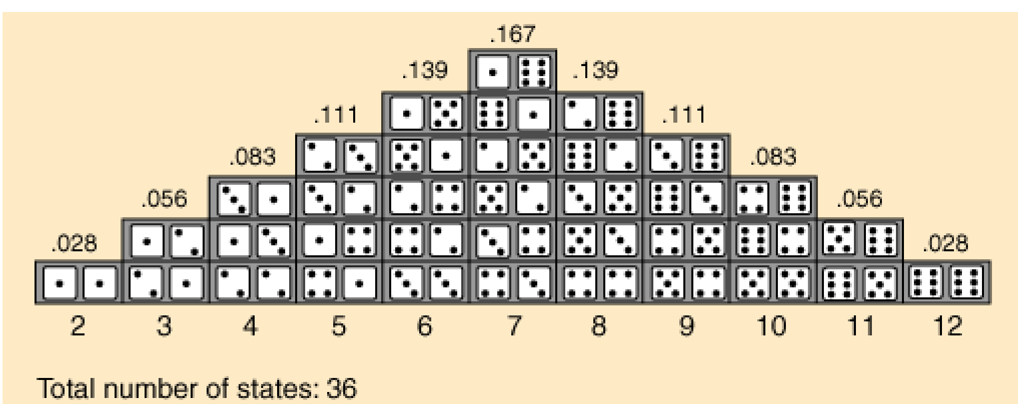
Step 2: Create your outcome cell. In the Outcome2 cell add the two cells next to Die1 and Die2 “=yourcellforDie1(for me C4) + yourcellforDie2(for me C5).” Now we will need to indicate the output cell the same way we did earlier but in the cell next to Outcome2 (Input Function, Output, RiskOutput).

Step 3: Run the simulation! You should now have an output chart depicting a normal distribution from adding these 2 random variables. This again shows the shape, mean (in this case 7), standard variation, and so on.



Section 4: Conclusion

Rolling a die 5000 times through the simulation proved that it's just as likely to land on 1 as it is 2, 3, 4, 5, or 6 (assuming there's nothing wrong with the die of course). However for rolling 2 die 5000 times, we see the probability of every number that could be equaled in the 36 different combinations of dice rolls.



This picture shows the probability of a particular outcome. For example, there are six different ways that the dice could sum to seven, so the probability of rolling seven is equal to 6/36=.167. Instead of doing it that way manually, the Monte Carlo simulated throwing dice 5000 times. With more complex models you could see how this could be helpful! In dealing with uncertainty in many directions, this can help judge different possible scenarios and risks, optimizing risk management in decision making.

References:

Dr. Jim Grayson

http://www.goldsim.com/Web/Introduction/Probabilistic/MonteCarlo/

http://www.palisade.com/risk/monte_carlo_simulation.asp

http://www.solver.com/monte-carlo-simulation-overview