An Interactive Tool to Teach Monte-Carlo Simulation and VBA

Roger B. Grinde, University of New Hampshire, Durham, New Hampshire (USA)

ABSTRACT

This paper presents an author-developed interactive tool that can be used to help students develop and run spreadsheet-based Monte-Carlo Simulation models, with output statistics automatically calculated. The software also provides the instructor the ability to help students learn some aspects of Visual Basic for Applications (VBA). Unlike commercial alternatives, it requires no special installation, and runs on both Windows and OS X versions of Excel. Students can quickly use it productively. The simple results screen and its interactive nature give it some advantages over commercial and native Excel simulation approaches for the classroom environment. Download links for the tool and an example file are provided.

Keywords: Monte-Carlo Simulation, VBA, Analytics, Spreadsheet Modeling

INTRODUCTION

Business Analytics has become recognized in industry as providing value, and interest in academic programs has increased as well. At the author's institution, the number of undergraduate business students specializing in information systems and/or business analytics has increased significantly over the past several years. Coupled with this is an increase in the number of specialized master's programs in analytics and related areas. The increased demand creates opportunities as well as challenges. With curriculum space not always as large as the need for courses in analytics-related areas, some courses are called upon to cover additional topics, or to combine topics in order make more efficient use of time in courses.

This paper presents an interactive tool developed by the author that can be used to help students develop and run spreadsheet-based Monte-Carlo Simulation models, with output statistics automatically calculated. No special installation is needed, and the tool runs on both Windows and OS X versions of Excel. Students can quickly use the tool productively. The software also provides the instructor the ability to help students learn some aspects of Visual Basic for Applications (VBA). For courses where exposure to some concepts of programming is desired, as opposed to a full course in programming, the ability to illustrate how VBA can extend the capabilities of spreadsheet models is beneficial.

There are alternative tools for Monte-Carlo Simulation. The two extremes in a spreadsheet context are commercial add-ins (e.g., Crystal Ball, Analytic Solver Platform, @Risk) and native Excel. The decision of approach often takes into consideration overall course goals, other topics in the course, and the time allocated to the topics. The commercial tools are quite powerful, yet require separate installation and a license code, often at a cost. Native Excel can be cumbersome as the user needs to set up a Data Table to run the simulation, followed by manually writing the summary functions and developing desired charts.

The tool presented here lies between these two boundaries, providing an easy to use way to quickly run simple simulations. When not much course time can be devoted to simulation, and the instructor does not want students to need to purchase additional software, this can be an appealing option. The course may also have broader goals, including exposing students to programming or enhancement of spreadsheet models with VBA. It is such a course for which the tool discussed in the paper was developed.

The paper briefly reviews relevant literature. An overview of the tool and illustration of its use is presented. Discussion of implementation follows, along with teaching applications, comparisons to other tools, and conclusions.

LITERATURE REVIEW

Spreadsheet-based Monte-Carlo simulation is a core topic in most business school quantitative methods courses (e.g., Business Analytics, Management Sciences, Quantitative Decision Making). Grossman (2015) aims at practitioners of Operations Research with an emphasis on spreadsheet modeling. Similarly, LeBlanc & Grossman (2008) discuss the use of spreadsheets for Management Science and Operations Research practitioners. Leong & Cheong (2004) discuss a business modeling course, and how spreadsheets are used in it. Complementing commercial add-ins for simulation, Eckstein (2002) presents an Excel add-in for Monte-Carlo simulation.

Simulation has also been found to be an effective methodology for teaching other topics. For example, Meilczarek & Zabawa (2011) explain how they use Monte-Carlo simulation to help teach a variety of other Management Science topics. Tsai & Wardell (2006) use a VBA-based tool to teach statistics concepts. Doane (2004) makes an argument for using simulation to teach about probability distributions. Similarly, Haney (2015) and Weltman (2015) teach students sampling distribution concepts using Monte-Carlo simulation. Valle & Nordell (2013) teach about forecast uncertainty using Monte-Carlo simulation. In the quality control arena, Balakrishnan (2005) presents a VBA-based tool for teaching statistical process control and process management. And Pappas et al. (1982) discuss a tool designed to teach statistical quality control that turns out to be effective in teaching simulation.

Turning to the teaching of VBA, there is support for it in the literature. Ragsdale (2001) discusses Management Science education, and the role VBA can play in teaching courses in it. Bauer (2006) makes the case for teaching VBA to Finance students. Martin (2000) discusses a stepwise progression approach to teaching VBA in a masters-level program. Palocsay & Markham (2002) discuss teaching a decision support systems (DSS) course using VBA as the platform. And Botchkarev (2015) evaluates VBA suitability for performing Monte-Carlo simulation.

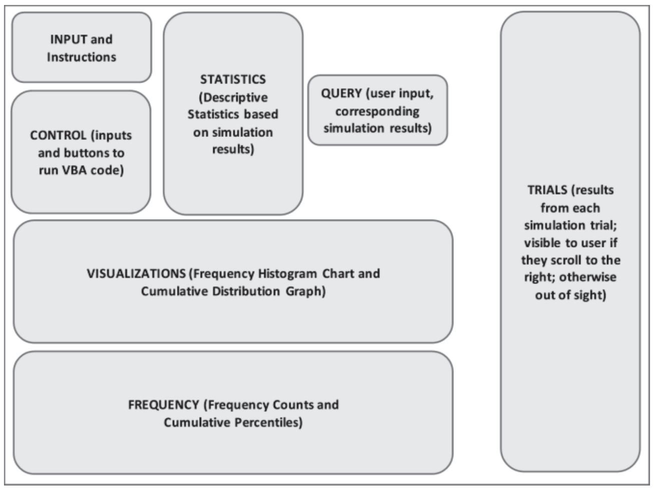
The contribution discussed in this paper is that the interactive simulation tool, while primarily designed to help students learn Monte-Carlo simulation, is also a suitable platform from which to begin, or complement, teaching VBA as well. The tool is highly visual, promotes understanding of what is happening as a simulation runs, and provides the opportunity to delve deeper into how the tool works.

MODEL OVERVIEW AND EXAMPLE

This section presents the overall design, provides an overview of the logic, and presents an example of the tool's use. Ease of use is a design goal, to a) allow the instructor and student to focus on model building and analysis, and b) provide the user with visualizations and statistics automatically and interactively. It is contained in single Excel workbook, with several VBA macros, saved with an XLSM extension. Upon opening, the user needs to enable macros. No add-in installation or license code is required.

Figure 1 shows the overall structure of the main worksheet. In the INPUT section, the user enters a random variable function (e.g., =RAND(), or any formula directly or indirectly involving randomness). The user can also get detailed instructions (Instructions button); the input cell also has a detailed comment. The CONTROL section allows the user to run one or more trials, user-specified. It also allows resetting the simulation (deleting all trials information), and toggling screen updating on or off while the trials run. The STATISTICS section calculates descriptive statistics. The QUERY section lets the user query a specific cumulative probability value, by entering either the cumulative probability, or the value of the output measure. The VISUALIZATIONS section shows a frequency histogram and a cumulative distribution. The number of bins of the frequency histogram can be changed using a slider control. Below the visualizations is the FREQUENCY section, showing the details of the frequencies. To the right, usually out of the user's sight, available if desired, is the result from each trial (TRIALS section).

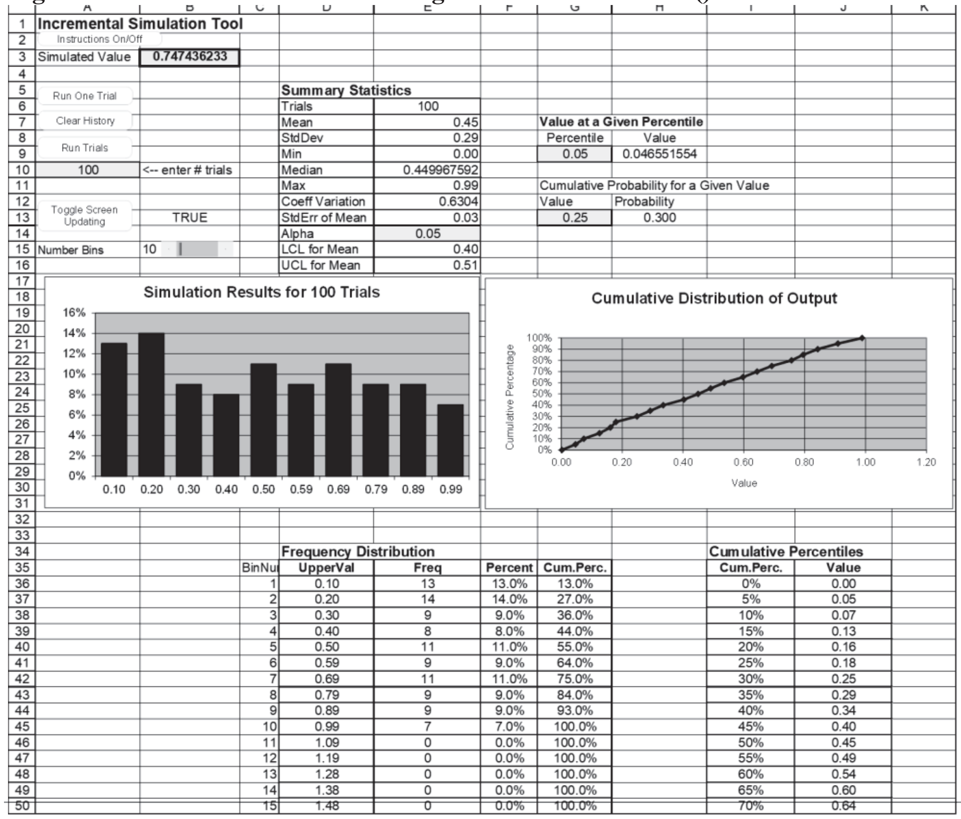
Figure 1. Overall Layout of Interactive Simulation Tool


INPUT and Instructions
CONTROL (inputs and buttons to run VBA code)
STATISTICS (Descriptive Statistics based on simulation results)
QUERY (user input, corresponding simulation results)
TRIALS (results from each simulation trial; visible to user if they scroll to the right; otherwise out of sight)
VISUALIZATIONS (Frequency Histogram Chart and Cumulative Distribution Graph)
FREQUENCY (Frequency Counts and Cumulative Percentiles)

Basic Usage

Figure 2 is a screen shot showing results of 100 trials from the U(0,1) distribution, with cell B3 containing the formula =RAND(). This is the most basic usage of the tool. While the trials are being generated, the results update as long as Screen Updating is set to TRUE. The user can request an additional 100 (or any number) trials, without deleting the original 100, effectively pausing the simulation after running each set of trials. Students can observe how sample size affects results (statistics, output distribution, confidence intervals, etc.). Naturally, as the number of trials increases, confidence intervals become narrower and charts more defined. The number of bins in the histogram is easily changed with a slider control, beneficial in the display of results.

Figure 2. Screen Shot of Tool Showing 100 Trials of =RAND()



	A	B	C	D	E	F	G	H
1	Incremental Simulation Tool							
2	Instructions On/Off							
3	Simulated Value	0.747436233						
4								
5	Run One Trial			Summary Statistics				
6				Trials	100			
7	Clear History			Mean	0.45		Value at a Given Percentile	
8				StdDev	0.29		Percentile	Value
9	Run Trials			Min	0.00		0.05	0.046551554
10	100	<-- enter # trials		Median	0.449967592			
11				Max	0.99		Cumulative Probability for a Given Value	
12	Toggle Screen Updating			Coeff Variation	0.6304		Value	Probability
13		TRUE		StdErr of Mean	0.03		0.25	0.300
14				Alpha	0.05			
15	Number Bins	10		LCL for Mean	0.40			
16				UCL for Mean	0.51			

	C	D	E	F	G	H	I	J
34		Frequency Distribution					Cumulative Percentiles	
35	BinNu	UpperVal	Freq	Percent	Cum.Perc.		Cum.Perc.	Value
36	1	0.10	13	13.0%	13.0%		0%	0.00
37	2	0.20	14	14.0%	27.0%		5%	0.05
38	3	0.30	9	9.0%	36.0%		10%	0.07
39	4	0.40	8	8.0%	44.0%		15%	0.13
40	5	0.50	11	11.0%	55.0%		20%	0.16
41	6	0.59	9	9.0%	64.0%		25%	0.18
42	7	0.69	11	11.0%	75.0%		30%	0.25
43	8	0.79	9	9.0%	84.0%		35%	0.29
44	9	0.89	9	9.0%	93.0%		40%	0.34
45	10	0.99	7	7.0%	100.0%		45%	0.40
46	11	1.09	0	0.0%	100.0%		50%	0.45
47	12	1.19	0	0.0%	100.0%		55%	0.49
48	13	1.28	0	0.0%	100.0%		60%	0.54
49	14	1.38	0	0.0%	100.0%		65%	0.60
50	15	1.48	0	0.0%	100.0%		70%	0.64

Most cells are protected from accidental change (if desired, the user can un-protect the sheet and make modifications). The following cells and buttons are the ones the user can modify without removing protection:

- Button: Run One Trial. Runs a single trial and updates results.
- Button: Clear History. Clears any existing trials and resets results.
- Button: Run Trials. Runs the number of trials specified in cell A10, and updates results.
- Button: Toggle Screen Updating. Toggles screen updating on/off during the running of multiple trials.
- Cell B3: formula directly or indirectly dependent on random values, including reference to a cell on another sheet or workbook.
- Cell A10: number of trials to run in the next batch.
- Cell B15 (change with slider): Number of bins used in histogram.
- Cell E14: Significance level α for confidence interval calculations.
- Cell G9: Cumulative percentile; corresponding value from cumulative distribution returned in H9.
- Cell G13: Cumulative distribution value; corresponding percentile returned in H13.

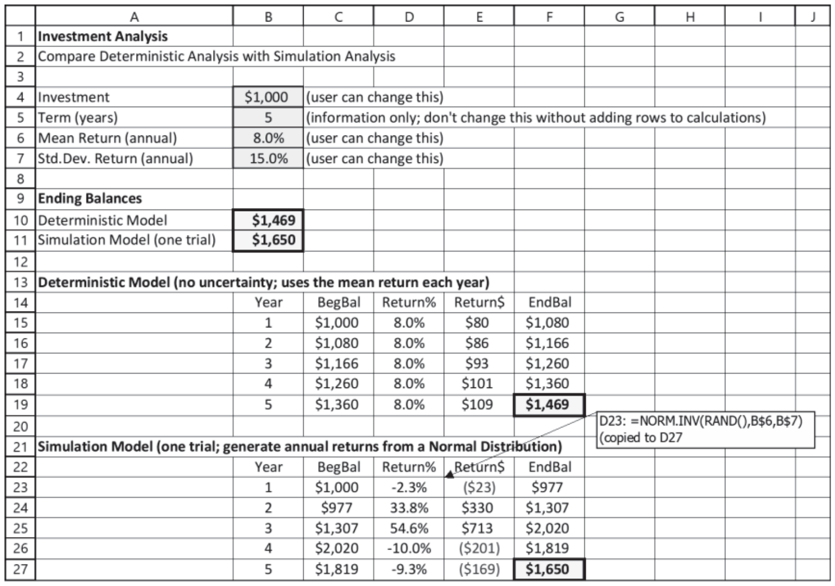
Besides this simple demonstration, the tool can help students gain more understanding of the effects of sample size, different probability distributions, and estimation of statistical parameters. Example formulas for Cell B3 are many, but include:

- =RAND()+RAND(). Asking students what they expect the shape of this distribution challenges them to think about how independent random variables combine together (i.e., even though each component distribution is uniform, the resulting combined distribution is not). Additional RAND() functions can be added, and students can gain intuition for one of the results of the Central Limit Theorem as the distribution approaches a Normal Distribution.
- =NORM.INV(mean, std.dev). This is the most recognized distribution by students. In terms of simulation, it can be used to use to illustrate that we may need many trials from a simulation to get a representative picture of what may happen in real life.
- =RANDBETWEEN(1,6). Simulating a die roll. This use can emphasize that the average of the trial values, which converges to 3.5, may be a value that has *zero* probability of occurring on any single trial. Students may have a tendency to put more emphasis on an *average* result than on the *distribution* of outcomes. Relating this to business situations emphasizes the importance of the distribution of potential outcomes and estimating probabilities of specific outcomes, rather than relying only on an estimate of the average.

Example

An example illustrating the initial motivation for the tool is discussed here. The formula in cell B3 (Figure 2) is set to the 1-trial output value of a simulation model. Figure 3 shows a simple 5-year investment model with a starting balance of $1000, stored in a separate workbook. First, the 5-year ending balance from a deterministic model ($1469) is calculated (rows 13-19). For the simulation model (rows 21-27), a normally-distributed return is generated for each year using the NORM.INV function. Simulation output for one trial is in Cell F27.

Figure 3. Investment Simulation Example

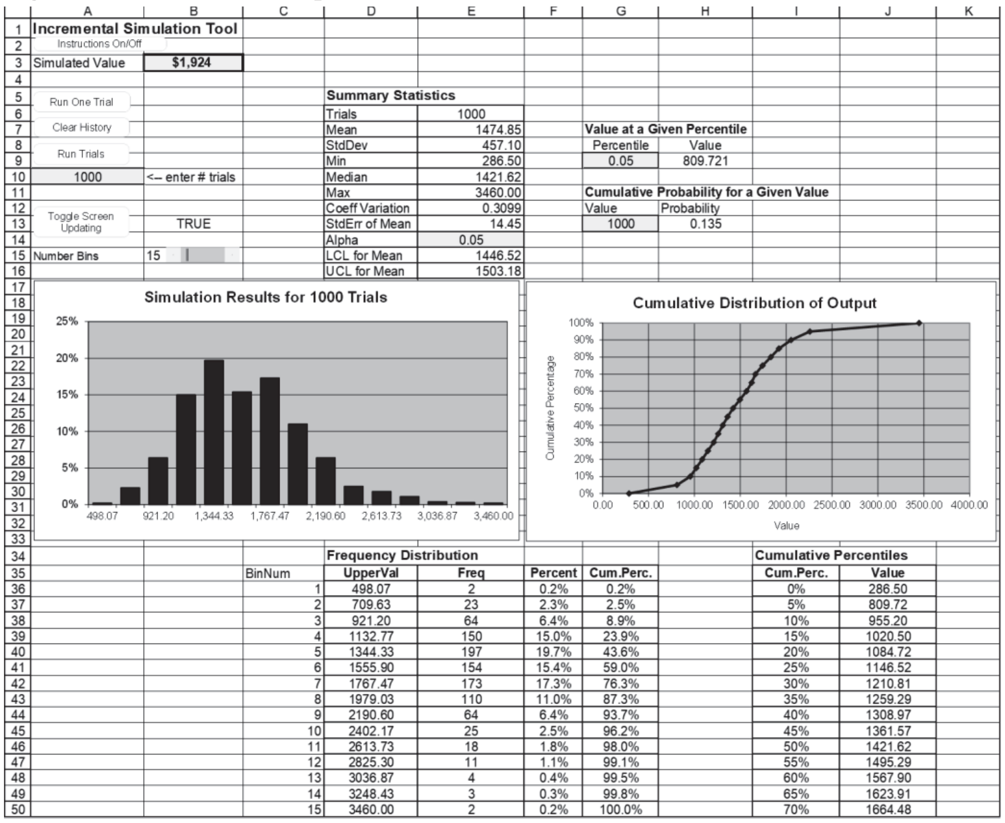


	A	B	C	D	E	F	G	H	I	J
1	**Investment Analysis**									
2	Compare Deterministic Analysis with Simulation Analysis									
3										
4	Investment	$1,000	(user can change this)							
5	Term (years)	5	(information only; don't change this without adding rows to calculations)							
6	Mean Return (annual)	8.0%	(user can change this)							
7	Std.Dev. Return (annual)	15.0%	(user can change this)							
8										
9	**Ending Balances**									
10	Deterministic Model	**$1,469**								
11	Simulation Model (one trial)	**$1,650**								
12										
13	**Deterministic Model (no uncertainty; uses the mean return each year)**									
14		Year	BegBal	Return%	Return$	EndBal				
15		1	$1,000	8.0%	$80	$1,080				
16		2	$1,080	8.0%	$86	$1,166				
17		3	$1,166	8.0%	$93	$1,260				
18		4	$1,260	8.0%	$101	$1,360				
19		5	$1,360	8.0%	$109	**$1,469**				
20										
21	**Simulation Model (one trial; generate annual returns from a Normal Distribution)**									
22		Year	BegBal	Return%	Return$	EndBal				
23		1	$1,000	-2.3%	($23)	$977				
24		2	$977	33.8%	$330	$1,307				
25		3	$1,307	54.6%	$713	$2,020				
26		4	$2,020	-10.0%	($201)	$1,819				
27		5	$1,819	-9.3%	($169)	**$1,650**				

D23: =NORM.INV(RAND(),B$6,B$7)
(copied to D27

In the simulation tool (Figure 4), cell B3 refers to Cell F27 in the investment example file (Figure 3), using the formula ='[Investment-Example-5year.xlsx] Model'!$F$27. First, use the "Clear History" button; otherwise new trials will be appended to any existing trials, which may have been from a different model. Then set the number of trials (cell A10), and use the "Run Trials" button. The figure shows a screen shot after 1000 trials. As expected given the mathematics of the example, the mean of the simulation results, $1475, is essentially equal to the calculation from the deterministic model. But now the user has access to information relating to the risk profile of the investment, insight about the shape of the output distribution, and a straightforward way to learn about the cumulative distribution function. Perhaps most importantly, probabilities of specific outcomes can be estimated.

Figure 4. Investment Example Simulation Results



	A	B	C	D	E	F	G	H
1	Incremental Simulation Tool							
3	Simulated Value	$1,924						
5				Summary Statistics				
6				Trials	1000			
7				Mean	1474.85		Value at a Given Percentile	
8				StdDev	457.10		Percentile	Value
9				Min	286.50		0.05	809.721
10	1000	<-- enter # trials		Median	1421.62			
11				Max	3460.00		Cumulative Probability for a Given Value	
12				Coeff Variation	0.3099		Value	Probability
13		TRUE		StdErr of Mean	14.45		1000	0.135
14				Alpha	0.05			
15	Number Bins	15		LCL for Mean	1446.52			
16				UCL for Mean	1503.18			

	C	D	E	F	G	I	J
34		Frequency Distribution				Cumulative Percentiles	
35	BinNum	UpperVal	Freq	Percent	Cum.Perc.	Cum.Perc.	Value
36	1	498.07	2	0.2%	0.2%	0%	286.50
37	2	709.63	23	2.3%	2.5%	5%	809.72
38	3	921.20	64	6.4%	8.9%	10%	955.20
39	4	1132.77	150	15.0%	23.9%	15%	1020.50
40	5	1344.33	197	19.7%	43.6%	20%	1084.72
41	6	1555.90	154	15.4%	59.0%	25%	1146.52
42	7	1767.47	173	17.3%	76.3%	30%	1210.81
43	8	1979.03	110	11.0%	87.3%	35%	1259.29
44	9	2190.60	64	6.4%	93.7%	40%	1308.97
45	10	2402.17	25	2.5%	96.2%	45%	1361.57
46	11	2613.73	18	1.8%	98.0%	50%	1421.62
47	12	2825.30	11	1.1%	99.1%	55%	1495.29
48	13	3036.87	4	0.4%	99.5%	60%	1567.90
49	14	3248.43	3	0.3%	99.8%	65%	1623.91
50	15	3460.00	2	0.2%	100.0%	70%	1664.48

Simulation results include descriptive statistics, calculation of the confidence interval for the mean for a user-specified significance level, and cumulative distribution lookup values. In Figure 4, the user entered 0.05 for the percentile (cell G9), corresponding to $809.72, indicating a 0.05 probability of ending the 5-year period with this amount or less. The user also entered a value of $1000 (cell G13), corresponding to a cumulative probability of 0.135, indicating the probability of ending the 5-year period with $1000 or less (i.e., incurring a net loss after 5 years of investing with this strategy). By running a few batches of trials in succession without clearing the history, students see what a simulation is doing in terms of refining estimates of output statistics.

IMPLEMENTATION

As an Excel/VBA model, one decides the relative roles native Excel and VBA have in the look, feel, and operation. Here, a design relatively heavy on native Excel was used, with VBA's role to automate specific aspects. With a design goal for a tool that was easy to use and helped the user learn about the details of simulation, this approach seemed appropriate. The tool needs no installation, and works with Excel on both Windows and OS X platforms. Commercial tools tend to be Windows-only, and students sometimes struggle with installation. For students using company-owned computers, policies sometimes restrict installation of new software. Most of the commercial tools also require a license fee. This section explains the native Excel design, and provides an overview of the VBA usage to make the tool operational.

Native Excel Design

Referring to Figure 1 and Figure 2, the native Excel portion is essentially all the sections except the CONTROL section. The TRIALS section will contain the results from each of the trials. The range of values from the trials is named Results_1. The STATISTICS and QUERY sections are built using standard Excel formulas, using Results_1 as the data range. In the FREQUENCY section, the Cumulative Percentiles are computed using the PERCENTILE function. The Frequency Distribution is somewhat more complex, as the user can enter the number of bins desired, using a slider control. Based on the number of bins, the bin size is computed based on the maximum and minimum values from the simulation, and the number of bins. The bin size is used in the frequency distribution calculations for the upper bound on each bin, and the FREQUENCY array function is used to find the actual frequencies.

The Cumulative Distribution chart is a standard scatter chart of the cumulative distribution calculations. The histogram is a column chart showing the frequency distribution values based on the user-specified number of bins. The chart data range is dynamic, using named ranges BinRange and PercentRange. These are dynamic ranges defined using the OFFSET function. The chart title is dynamic to show the number of trials.

The worksheet is protected, but not locked with a password. The VBA code changes the protection setting when needed to make changes to the worksheet. In addition, the user can make changes to the worksheet, recommended only on a copy of the file.

VBA Functionality

VBA is needed to provide control over the calculation of trials, resetting trials, and the screen updating toggle. The amount of VBA code is quite small. Below is a list of the procedures, and a short description of their functions:

- Sub Workbook_Open(). Event-based; runs when the workbook opens. Calls the Initialization routine.
- Sub Initialization(). Initializes settings so that each time the tool is opened, the user has a similar interface.
- Sub ToggleInstructions(). Toggles whether the instructions are shown. The instructions are stored as a comment in Cell A1. Mapped to a button on the interface.
- Sub ClearHistory(). Clears any trials data; that is, any results in the Results_1 range. Mapped to a button on the interface.
- Sub ToggleScreeenUpdate(). Turns on/off screen updating. Mapped to a button on the interface.
- Sub OneTrial(). Runs one trial of the simulation, and appends the trial result to any previously-run trials. Specifically, it forces a calculation of Cell B3, and stores the result of the trial in the range defined by the Result_1 array. Mapped to a button on the interface.
- Sub ManyTrials(). Runs the number of trials specified by the user in cell A10. Mapped to a button on the interface.

The OneTrial and ManyTrials routines provide the main numerical functionality from the VBA standpoint. OneTrial tells Cell B3 (with the range name “Output”) to calculate using Range("Output").Calculate. As B3 can contain any formula directly or indirectly involving randomness, the calculation cascades to any of B3’s predecessors, and any of B3’s dependents (e.g., statistics, frequencies, etc.). ManyTrials is similar.

Random Number Generators

Besides the VBA routines listed above, several random number generators (RNG’s) are included and listed here (each have applicable calling parameters):

- RndUniform: Continuous uniform
- RndDUniform: Discrete (integer) uniform
- RndNormal: Normal
- RndBinomial: Binomial
- RndExponential: Exponential
- RndTriangular: Triangular
- RndPoisson: Poisson

- RndDiscrete: Discrete; input is range of values and range of frequencies or probabilities
- RndReSample: Input is range of values; one is chosen at random; works on numeric and non-numeric ranges.
- RndContEmp: Input is value range and associated cumulative probabilities. Converted to piecewise continuous distribution.
- RndGeom: Geometric
- RndNegBinom: Negative Binomial

This functionality puts the tool between native Excel simulation, and simulation using a powerful commercial add-in. Use of the RNG's is optional; one can still write native Excel formulas to generate the random values, as in the example discussed earlier. But some distributions are more difficult to write in native Excel, so use of RNG functions can be a convenience. If a user chooses to use these functions in the model, the simulation tool file and the model file are more tightly linked than otherwise, as the model file will be referring to VBA user-defined functions in a different file. Alternately, the simulation tool file can be copied and the model built in a worksheet of the copied file.

TEACHING SCENARIOS

The initial motivation was to develop a tool to make implementing simple Monte-Carlo simulations easily, without add-ins. Once the tool was developed, it became apparent that it could also be used to assist in VBA instruction. The tool can be used in different ways, from using simulation to illustrate concepts in an Operations Management, Marketing, or Finance class, to usage in a Business Analytics class that covers simulation methodology in more depth, with or without VBA. It can the only demonstration of how to do Monte-Carlo simulation, or combined with coverage of native Excel (e.g., Data Tables) and/or a full-scale commercial tool (e.g., Analytic Solver Platform, Crystal Ball, @Risk). The author's primary use is in a course that covers simulation and a brief introduction to some capabilities of VBA, along with numerous other topics (e.g., optimization, data mining). By also covering some foundations of programming, students can enhance capabilities of their models through VBA. The tool here shows how just a little bit of VBA code can add functionality to a model. A course having a primary objective to teach programming usually uses a more structured approach. Nevertheless, introducing VBA and some programming concepts can be done through examples like this, in a course where teaching programming is not the primary objective.

It can also be used to help teach about random variables and probability distributions, and descriptive statistics. With the ability to quickly generate values from any distribution one can code into a workbook cell (using the user-defined function the workbook itself, this list is larger), students can learn more intuitively and visually about various probability distributions.

From a simulation perspective, students can interactively see the results being generated one trial, or a batch of trials, at a time. This tends to improve understanding as students derive insights from many trials. The incremental use of the tool seems to be beneficial. One can set the number of trials (cell A10) to something like 100, and repeatedly use the "Run Trials" button. In this way, the effect of sample size on parameter estimates can be reinforced.

Students can take this tool with them after the course is over, without any licensing restrictions. This is typically not the case for commercial tools. Working professionals sometimes want to continue using simulation after the course is over, and they may not be able to convince their firms to purchase a commercial license. With this tool, they can at least demonstrate what simulation can do for their firm, to bolster their case for a more powerful tool.

VBA Instruction Possibilities

The rest of this section addresses how some aspects of VBA can be illustrated. VBA is a programming language, with additional capability through the Microsoft Office object library. In a course where *exposure* to programming is the objective (as opposed to a full-scale course in programming), an example-first approach followed by illustration of some programming concepts, can be time-efficient while also providing students with a sense of some of the capabilities of programming. Interested students can then be directed to more structured treatments of programming. As a language, VBA has mechanisms for defining subs/functions, variables, data types, input/output, loops, and conditional logic. Programmers typically write code in the Visual Basic Editor. VBA also has access to much of the functionality of native Excel through the object library. One does not need to learn everything about any aspect initially, but rather some basics, and then incrementally as needed.

Recording a macro is often a user's first exposure to VBA. In the tool, displaying the instructions (cell A1 comment) is one feature. A macro can be recorded to show the comment, and another to hide it. By studying the VBA code, students can begin to learn the structure of VBA code (subs, statements, how VBA references Excel objects, etc.). This is also an opportunity to point out that the recorded code often needs to be cleaned up.

For example, a number of programming concepts and language features are illustrated by studying the code for running simulation trials, using the ManyTrials sub. This routine also references Excel's object model, especially the Range object, to direct calculation and to put the trial value into the output section of the spreadsheet. Figure 5 shows the commented code for the ManyTrials sub.

Figure 5. ManyTrials VBA Code (comments shaded)

```
01 Sub ManyTrials()
02    'This sub runs one or more trials of the simulation, increments the trials counter,
03    'and stores the trial values.
04    'Comments precede each line of code.
05    'Counter variables
06    Dim i, n As Long
07    'Number of trials completed already.
08    Dim vartrials As Long
09    'Unprotect sheet to allow changes
10    Sheets("Sim Inc").Unprotect
11    'Set screen updating to user preference, stored in screenupdatevalue.
12    Application.ScreenUpdating = screenupdatevalue
13    'Current number of trials (already run)
14    vartrials = Range("Trials").Value
15    'Number of new trials to run
16    n = Range("num_trials").Value
17    'Loop over number of new trials
18    For i = 1 To n
19        'Calculate the Output cell (implies calculation of predecessor and successor cells)
20        Range("Output").Calculate
21        'Increment trials counter
22        Range("HistoryHeader").Cells(vartrials + i + 1, 1).Value = vartrials + i
23        'Store new trial value
24        Range("HistoryHeader").Cells(vartrials + i + 1, 2).Value = Range("Output").Value
25    'Loop for next new trial; when n new trials have been done, exit loop.
26    Next i
27    'Re-protect the sheet to avoid inadvertent changes.
28    Sheets("Sim Inc").Protect
29 End Sub
```

In the ManyTrials routine, the following programming and VBA concepts are used:

- Line 1/29. Declaration and ending of a subroutine.
- Lines 2-5, and others (shaded). Comments.
- Lines 6, 8. Variable declarations.
- Lines 10, 12. Using Excel's object model to control Excel. Specifically, unprotect the sheet and set screen updating to the user preference.
- Lines 14, 16. Assigning values to variables, using values stored on the worksheet.
- Lines 17-26. Illustration of a loop control structure.
- Line 18. Top of the loop.
- Line 20. Forcing cell B3 (named "Output") to calculate.
- Lines 22, 24. Adding the new trial to the list of trials.
- Line 26. End of the loop.
- Line 28. Cleaning up; re-protect the sheet, as it was when entering the subroutine.

In the other subs, some other concepts are illustrated. For example, the Workbook_Open sub runs automatically whenever the workbook is opened. Showing this can be a way to introduce the concept of event-based procedures. If the course spends more time on programming, this can be a convenient first example for an event-based procedure.

In a course utilizing spreadsheet modeling for business problems, and not an emphasis on programming per se, it does not take long to illustrate the VBA code and show students how to view it. One can also use this simulation tool as the basis for either lab-based exercises or homework to develop improvements and/or new functionality to the code.

COMPARISON TO OTHER TOOLS

The tool discussed here provides a way for students to learn about Monte-Carlo simulation, experiment and play with it, and generate useful results. It can also be used as a launching point for learning about VBA and programming in general. It is not a complete Monte-Carlo simulation solution. Commercial products are very powerful; e.g., Analytic Solver Platform, @Risk, and Crystal Ball. These products provide more functionality with respect to tracking multiple output cells, built-in probability distributions, fitting data to distributions, correlated random variables, and stochastic optimization. The intent of this tool is not to replace any of these, but to complement. By providing a tool that interactively shows users the results of the simulation as they are generated (raw results, summary statistics, frequency distributions, and cumulative percentiles), the user can develop a better understanding (and trust) of how simulation can be used in a decision-making situation. The tool also provides an easy to use simulation platform to enable demonstration of key concepts in other courses.

CONCLUSION

This paper presented an interactive Monte-Carlo simulation tool that allows users to see the results of their simulation as they are generated. It requires no installation; it is a standard macro-enabled Excel file (XLSM). Unlike commercial tools, it contains no license expiration. In addition, it provides an avenue in which students can begin to learn programming using VBA. While most business majors will not become programmers, students in business disciplines can benefit from at least an introduction to logic of programming in some language, even if they never take a course dedicated to programming. This tool provides a flexible way to provide that introduction at a level of the instructor's choosing.

LINKS TO SIMULATION TOOL AND EXAMPLE

These two links give access to download the XLSM tool file, and the investment model example used in the paper: Simulation Tool, https://goo.gl/q6fMaq; Investment Example, https://goo.gl/BXWr8o.

REFERENCES

Balakrishnan, Jaydeep; Sherry L. Oh (2005). An Interactive VBA Tool for Teaching Statistical Process Control (SPC) and Process Management Issues. *INFORMS Transactions on Education*, 5:3:19-32.

Bauer, Richard J. Jr. (2006). Teaching Excel VBA to Finance Students. *Journal of Financial Education*, 32: 43-63.

Botchkarev, A., (2015). Assessing Excel VBA Suitability for Monte Carlo Simulation. *Spreadsheets in Education*, 8:2, Article 3.

Doane, D.P. (2004). Using Simulation to Teach Distributions. *Journal of Statistics Education*, 12:1.

Eckstein, J.; S.T. Riedmueller, (2002). YASAI: Yet Another Add-in for Teaching Elementary Monte Carlo Simulation in Excel. *INFORMS Transactions on Education*, 2(2):12-26.

Grossman, T. (2015). Spreadsheet Modeling for Operations Research Practice. *Wiley Encyclopedia of Operations Research and Management Science*. 1–8.

Haney, Mark H. (2015) A Spreadsheet Simulation to Teach Concepts of Sampling Distributions and the Central Limit Theorem. *Spreadsheets in Education*, 8:3, Article 3.

LeBlanc, Larry J.; Thomas A. Grossman (2008). Introduction: The Use of Spreadsheet Software in the Application of Management Science and Operations Research. *Interfaces*, 38:4:225-227.

Leong, Thin-Yin; Michelle L. F. Cheong, (2008). Teaching Business Modeling Using Spreadsheets. *INFORMS Transactions on Education*, 9(1):20-34.

Martin, A (2000). An integrated introduction to spreadsheet and programming skills for operational research students. *Journal of the Operational Research Society*, 51:12:1399-1408.

Mielczarek, Bożena; Jacek Zabawa (2011). Integrating Monte Carlo Simulation with other Spreadsheet-Based Modelling Methods to Teach Management Science. *Proceedings 25th European Conference on Modelling and Simulation*. Krakow, Poland, June 7-10.

Palocsay, Susan W; Markham, Ina S. (2002). Teaching Spreadsheet-Based Decision Support Systems with Visual Basic for Applications. *Information Technology, Learning, and Performance Journal*, 20:1:27-35.

Pappas, I.A.; Maniatopoulos, K.; Protosiegelos, S.; Vakalopoulos, A. (1982). A tool for teaching Monte-Carlo simulation without really meaning it. *European Journal of Operational Research*, 11:3:217-221.

Ragsdale, C.T. (2001). Teaching Management Science with Spreadsheets: From Decision Models to Decision Support. *INFORMS Transactions on Education*, 1:2:68-74.

Saltzman, Robert M.; Theresa M. Roeder (2013) Perspectives on teaching simulation in a college of business. *Proceedings of the 2013 Winter Simulation Conference*, Washington, D.C., December 08-11.
Tsai, Weiyu; Wardell, D.G. (2006). An Interactive Excel VBA Example for Teaching Statistics Concepts. *INFORMS Transactions on Education*, 7:1:125-135.
Valle, M.; Norvell, T. (2013). Using Monte Carlo Simulation to Teach Students about Forecast Uncertainty. *Business Education Innovation Journal*, 5:2:35-41.
Weltman, David (2015). Using Monte-Carlo Simulation with Oracle Crystal Ball to Teach Business Students Sampling Distribution Concepts. *Business Education Innovation Journal*, 7:2:59-63.

Roger B. Grinde, Ph.D. is an associate professor of management science in the Peter T. Paul College of Business and Economics at the University of New Hampshire. His Ph.D. in Industrial Engineering and Operations Research is from The Pennsylvania State University. Earlier degrees are from Carroll College (MT) and Oregon State University. His research interests are in applied optimization, business analytics, end-user modeling & analysis, and pedagogy.