R U T C O R
R ESEARCH
R E P O R T

YASAI: YET ANOTHER ADD-IN FOR TEACHING ELEMENTARY MONTE CARLO SIMULATION IN EXCEL

Jonathan Eckstein[a] Steven T. Riedmueller[b]

RRR 27-2001, APRIL, 2001

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/~rrr

[a] MSIS Department, Faculty of Management, and RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854. E-mail: jeckstei@rutcor.rutgers.edu
[b] MSIS Department, School of Business – New Brunswick, and Livingston College Honors Program, Rutgers University. Permanent address: 15 Grandview Circle, Flemington, NJ 08822. E-mail: stever321@yahoo.com

RUTCOR RESEARCH REPORT
RRR 27-2001, APRIL, 2001

YASAI: YET ANOTHER ADD-IN FOR TEACHING ELEMENTARY MONTE CARLO SIMULATION IN EXCEL

Jonathan Eckstein Steven T. Riedmueller

Abstract. This report describes YASAI (Yet Another Simulation Add-In), a tool for performing Monte Carlo simulations with Excel spreadsheet models. Although numerous similar tools do already exist, we needed one that was simple to use for mathematically unsophisticated beginning students, and could be used and installed easily without system administrator privileges. We describe the design philosophy of YASAI and its features for generating random variables, identify simulation outputs, and specifying scenarios. We conclude with a simple example.

Acknowledgements: Implementing YASAI was Steven Riedmueller's Livingston college senior honors project, supervised by Jonathan Eckstein. We would like to thank Ben Melamed for his assistance in directing us to [1]. Steven would also like to thank the Excel experts group at AllExperts.com for their priceless advice.

1 Introduction

Spreadsheet-based tools provide a straightforward environment for teaching the elements of probability modeling and Monte Carlo simulation. For beginning students, spreadsheets — and in particular Microsoft Excel (see *e.g.* [2]) — provide a familiar modeling environment removing the obstacle of having to master wholly new software tools at the same time as learning the basic concepts of probability and simulation.

While elementary simulations can be performed by combining Excel's built-in RAND() function with its data table generator, a more convenient approach is a *simulation add-in*. The add-in supplements Excel's basic functionality by supplying the ability to generate random variables from standard probability distributions. Every time the spreadsheet is recalculated, new random values are drawn for these variables, and the remainder of the cells recomputed accordingly. The user designates certain cells as outputs, and directs the add-in to recalculate the model a large number of times (for example, 1000), while tabulating statistics on the output cells. Finally, the add-in displays the output statistics.

Numerous commercial, semi-commercial, and non-commercial add-ins presently exist, including *@Risk* [6] and *Crystal Ball* [3]. However, none of them proved ideal for our purposes, which involved teaching elementary Monte Carlo simulation to large groups of undergraduate business majors. This paper describes a new simulation add-in, YASAI (*Y*et *A*nother *S*imulation *A*dd-*I*n), which we created to meet our needs. (Note: "yasai" means "vegetables" in Japanese.)

2 Why yet another simulation add-in?

For several years, we used one of the commercial spreadsheet add-in products to teach a four-week simulation and probability module in our Operations Management course. However, students encountered numerous difficulties and frustrations:

Licensing and Installation: First, while we had a license for the software in our school of business computer lab, the operating hours and location of the lab were inconvenient for many students. In other modules of the class, which required only standard components of Excel, many students used general-purpose computer clusters located widely around the university. Unfortunately, a site license covering all these clusters would have been prohibitively costly. Even if the add-in were included, for example, in their textbooks, or a limited trial version downloaded from the internet, students would still have been unable to use it at these general-use clusters because the software requires administrator privileges to install. Students with their own computers were able to download a trial version, but the download was slow for those lacking a high-speed internet connection, the software would only work for a limited time, and some students still had installation difficulties.

Reliability and Configuration: While the commercial add-in would usually work reliably on faculty computers, it was often slow and unreliable in the student lab. Apparently, it made assumptions about the configuration of Excel that did not hold in the lab. It was based on

advanced inter-application communication features in Windows, and these features often worked very slowly in the lab environment. Delays on the order of 20 minutes in launching the add-in were common.

Complexity: In order to create a model, run the simulation, examine the results, and convert the results to printable form, students had to navigate numerous toolbars and dialog boxes, some of which included idiosyncratic terminology. Valuable class time, including extra classes in the computer lab, had to be spent to teach students how to navigate the user interface.

While the reliability and configuration issues may have been unique to the particular product we were using, the remaining issues seem to be common to the dominant commercial products. To provide the full functionality and efficiency demanded by the commercial marketplace, they must be large applications with extensive functionality, and a resultingly complex user interface.

Some simpler commercial and freeware products were also available (for example, [4] and [5]), but none exactly matched our needs. Some of the simpler products, for instance, require users to generate random variables values by explicitly applying the variable's inverse cumulative distribution to a uniform unit-interval argument generated by RAND(). While trivial for the mathematically adept, such techniques would be an unnecessary obstacle for our student population. Also, they may lead to inefficient generation of random numbers.

3 The design of YASAI

Because of the difficulties just described, we decided to create YASAI, our own simulation add-in software, in accordance with the following principles:

- The add-in is implemented as a single file in the Visual Basic for Applications (VBA) language. Any user on any computer equipped with a reasonably recent version of Excel should be able to use it, provided they are allowed to run Excel macros.

- The number of dialog boxes and steps in the user interface is kept to an absolute minimum. As many features as possible are controlled directly with functions that can be invoked from spreadsheet cell formulas, which minimizes the number of interface elements needed, and tends to afford users the maximum flexibility in combining the functions provided by the software. Running a simulation should produce full, printable output report in a single step.

- Beginning students should be able to easily construct and run simple models, without mathematical contortions such as inverting cumulative distributions.

- The code is open-source and freely available for any purpose (although, of course, there is no warranty). Users who want additional features should be able to add them and share them with the rest of the user community.

Because of these design principles, our entire student population should be able to use YASAI on whatever computer they prefer, without encountering excessive technical barriers. Instructors should not need to spend large amounts of time training students to navigate complicated toolbars and dialog boxes.

Naturally, there are some drawbacks to our approach:

- Simulations may run significantly more slowly than with a commercial package, because VBA is an interpreted language, and many parts of the commercial packages consist of compiled machine code. However, for simple classroom-oriented models, speed should not be an overriding concern, especially considering constant advances in microprocessor clock rates.

- YASAI lacks the extensive functionality and options of some of the commercial packages.

Apropos of this last point, the current version of YASAI produces a single, fairly simple output report, and has no graphics capabilities. Although too much complicated functionality would violate our basic design philosophy, some simple graphics enhancements, for example, would be straightforward to add to the software. Another disadvantage, relative to commercial packages, is that we have only six kinds of probability distributions. Again, more distribution generators could easily be added.

One further noteworthy aspect of YASAI's design is that it uses efficient random number generation algorithms derived from [1], whose average running time is bounded by a constant independent of their parameters. We have noticed that some of the commercial packages' random variable generators for discrete distributions, such as binomial and Poisson, can fail for large values of certain arguments, and we wished to correct this problem. Furthermore, using a relatively slow compiled language suggested that it would be best to use asymptotically efficient methods in which loops have a low average number of iterations.

4 Using YASAI

4.1 Specifying random variables

YASAI provides six random variable generators that we have found to be sufficient for our introductory class:

GENUNIFORM(*a*, *b*): Returns a value uniformly distributed over the interval [*a*, *b*).

GENNORMAL(*m, s*): Returns a value with a normal distribution with mean *m* and standard deviation *s*.

GENBINOMIAL(*n, p*): Returns an integer drawn at random from a binomial distribution with *n* trials and probability *p* of success at each trial. If *n* = 0, then the return value is 0.

GENPOISSON(*m*): Returns a an integer chosen from a Poisson distribution with mean value *m*. If *m* is zero, so is the returned value.

GENTABLE(*V, P*): Allows a discrete distribution to be specified by an arbitrary table. The arguments *V* and *P* are blocks of cells or lists of values (for example, "{1,3,7}") having the same shape. The probability of choosing any particular value from *V* is equal to the corresponding value in *P*.

GENEXPON(*a*): Follows an exponential distribution with mean value 1/*a*.

These functions may appear in cell formulas in arbitrarily nested and complicated ways. For example, the formula

=GENNORMAL(GENBINOMIAL(175,0.63),GENUNIFORM(1,4)) + GENPOISSON(5)

generates a normal random variable whose mean is in turn a random binomial variable of 175 trials with success probability 0.63, and whose standard deviation is in turn a uniform random variable between 1 and 4; to this value is added a Poisson variable with mean 5. Some commercial add-ins provide this level of flexibility, but others require the navigation of complicated dialog boxes to create random variables. We feel that the function-based design is simpler to teach, easier to use, and more flexible.

4.2 Specifying outputs

YASAI's method for specifying model outputs is very simple, and appears to be unique. To specify that output statistics should be tabulated for a model quantity, say *x*, one includes somewhere in the spreadsheet the formula SIMOUTPUT(*x*, *name*). Here, *name* is an argument that specifies what the output should be called. The SIMOUTPUT function always returns the value *x*. When working interactively with a spreadsheet, it does nothing more. However, when a simulation is running, SIMOUTPUT records each value it sees for later analysis.

For example, suppose that a model contains total revenue in cell B5, total cost in cell B6, and cell B7 computes profit via the formula =B5 – B6. If we wish the simulation to record total profit, we would replace the =B5 – B6 formula in B7 with

=SIMOUTPUT(B5 – B6,"Profit") .

SIMOUTPUT functions can be embedded into formulas in arbitrary ways. For example, in this same context, the formula

=SIMOUTPUT(SIMOUTPUT(B5,"Revenue") – SIMOUTPUT(B6,"Cost"),"Profit")

Simultaneously records the value of B5 – B6 as "Profit", the value of B5 as "Revenue", and the value of B6 as "Cost".

In a simpler example, suppose cell C6 contains a Poisson random variable whose mean value is specified by cell C3. While C6's main purpose is to serve as an input to other formulas in the model, we would like the simulation output to contain statistics about it. To ensure the output contains this information, we just change the formula in C6 from =GENPOISSON(C3) to, for example

=SIMOUTPUT(GENPOISSON(C3),"Poisson Input Variable") .

Creative use of conditional formulas can afford even greater flexibility. Suppose that cell D12 contains net cash flow. The formula

=SIMOUTPUT(ABS(D12),IF(D12>=0,"Profit","Loss"))

will create two outputs. The first, "Profit", will contain statistics on the net cash flow in D12, but tabulated only over those instances where it was nonnegative. The second, "Loss", will tabulate the absolute value of net cash flow, but only over those instances where net cash flow was negative.

In a further measure of flexibility, if multiple SIMOUTPUT expressions in a model have the same *name* argument, their values are pooled together in the simulation output.

We are not aware of another spreadsheet add-in that provides such broad flexibility in specifying outputs. Nevertheless, our design is simple to use and implement, and the added flexibility it affords is not a distraction to beginning students, who may just learn the simplest cases. Specifying an output does not require any special user interface elements. Furthermore, since the output specification is part of the ordinary formula mechanism, the spreadsheet cannot "forget" which cells are outputs. Similarly, examining the spreadsheet formulas reveals at a glance which cells are outputs — an advantage when grading assignments or trying to understand other people's models.

4.3 Specifying scenarios

In most of our class assignments, as in many applications of simulation, the actual objective is to evaluate a number of similar models, which vary only in some selected decision parameters. One may then wish to select the parameter combination that gives the highest expected profit or lowest expected cost, perhaps subject to some acceptable level of risk quantified by other some other set of outputs. We call each tested set of parameter values a *scenario.*

Like some commercial add-ins, YASAI provides a mechanism for automating the comparison of scenarios. In keeping with our design philosophy, the mechanism is based on a function that can be embedded arbitrarily into cell formulas: PARAMETER(*L*, *k, name*). The first argument, *L*, is a block of cells or a list specifying the possible return values. The *name* argument is a character string describing the parameter, and is used only in the output reports. The *k* argument specifies the number of scenarios between changes in the parameter. For example, a cell containing =PARAMETER({1,2,3},1,"fred") specifies a parameter called "fred" that will change every scenario, first taking the value 1, then 2, and then 3. A cell containing the formula =PARAMETER({1,8,2.3,–2},2,"daisy") defines a parameter called "daisy" that takes the value 1 for the first two scenarios, then 8 for the next two, 2.3, for the next two, and –2 for the last two. The argument *k* makes it possible to try combinations of values of various parameters without having to write every combination explicitly in a list (although that approach is possible). For instance, suppose that the cells E6:E8 contain the formulas

=PARAMETER({0,1},1,"A")
=PARAMETER({100,150,200},2,"B")
=PARAMETER({20,40},6,"C") ,

signifying parameters called *A*, *B*, and *C*, respectively. Then, if we run 12 scenarios, the parameters would assume the following values:

Scenario	***A***	***B***	***C***
1	0	100	20
2	1	100	20
3	0	150	20
4	1	150	20
5	0	200	20
6	1	200	20
7	0	100	40
8	1	100	40
9	0	150	40
10	1	150	40
11	0	200	40
12	1	200	40

Thus, all 12 possible parameter combinations would be tested, without having to code an exhaustive list of combinations into the spreadsheet.

When a model is being used interactively, the values of PARAMETER values change every *k* recalculations, so that the user can easily observe the model recalculating with all the conceivable parameter combinations.

4.4 Running the simulation

To run a simulation, one selects "YASAI Simulation" from Excel's *Tools* menu, which brings up YASAI's single dialog box, shown in Figure 1.

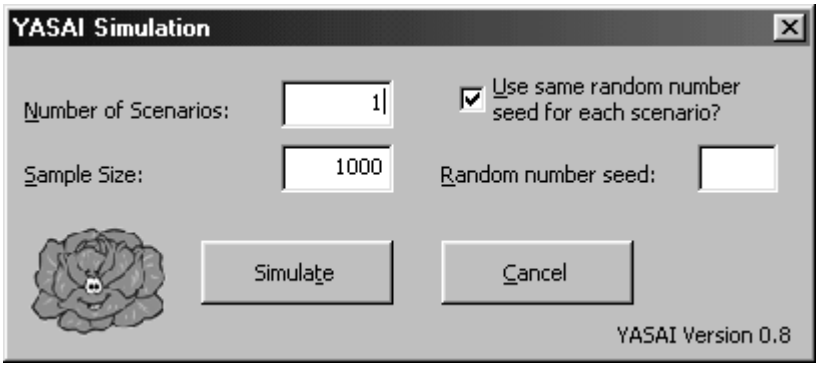

Figure 1. YASAI's only dialog box.

The dialog box allows the user to specify the number of scenarios and the number of sample recalculations per scenario. Ordinarily, nothing else needs to be entered. There is an option, however, to specify a fixed random number seed value (important when trying to exactly reproduce simulation behavior, and useful for model debugging). If the random seed field is not filled, YASAI constructs a seed from the system clock. A final option, turned on by default in accordance with standard practice, causes the same random seed value to be used for all scenarios.

Clicking the "Simulate" button on the dialog box starts the simulation. YASAI recalculates the spreadsheet *NS* times, where *N* is the sample size and *S* is the number of scenarios, storing the *x* arguments of all SIMOUTPUT functions, organized by output name and scenario number. During this process, YASAI shows a moving progress bar, and the user has may abort the simulation by clicking an "Abort" button.

When all *NS* recalculations are complete, YASAI produces an output report. The output report becomes a new ply of the current workbook, and contains information for each output name/scenario number pair. For each such pair, the report contains the mean, standard deviation, minimum, maximum, and percentiles in 5% intervals. No histograms or other graphics are currently provided. For a partial sample output report, see the example below. The reports can be immediately printed via Excel's standard printing mechanism. To delete them, one just deletes the corresponding spreadsheet ply.

5 A simple example

To give a simple illustration of using YASAI, consider the following problem, drawn from our standard set of teaching examples, and originally due to D. Dentcheva:

Lingua Translations Company wants to determine how many part-time Hungarian translators it should hire. The number of orders per day for Hungarian translations follows a Poisson distribution with an expected value of 8. Each translator can take one order a day. Lingua pays a fixed amount of $10 per day to each translator, whether or not the translator has to fill an order. Furthermore, a translator gets $50 for translating an order. If the company receives more orders than it has translators, it asks the translators whether they are available for overtime. For each translator, the chance of being available for overtime is 40%, in which case the translator can process one additional order. Lingua pays $75 for overtime translation of an order. The company charges its customers $90 per translation order.

In order to maximize its average daily profits, how many translators should Lingua keep on staff? Consider all possible staffing levels from 5 to 10 translators, inclusive. At the optimum staffing level, how many orders per day, on average, will Lingua turn away for lack of sufficient staff?

Figure 2 shows a YASAI model for this problem, and Figure 3 displays the corresponding output report. Each of the six possible staffing levels constitutes a scenario, and the sample size for each scenario was 1000 recalculations. The best scenario appears to be scenario 5, with a staffing level of 9. At this staffing level, Lingua refuses an average of 0.148 orders per day.

6 Conclusion

YASAI provides a simple, freely-available, open-source tool for teaching elementary Monte Carlo simulation. As the example above shows, it is simple to use, and a formula-mode printout of the model, together with the output report, fully documents the analysis. It can be installed without administrator privileges on any computer that can execute Excel Visual Basic macros. While it lacks the full speed and functionality of some commercial packages, its simplicity should more than compensate for these deficiencies in an introductory-level classroom setting. Its method for specifying outputs is novel, powerful, and flexible, without intimidating beginning users. The method for specifying scenarios provides a simple way to try combinations of multiple parameters without having to write out an exhaustive list of all possible combinations. While only six distribution generators are currently provided, and there are no graphics, additional random number generators and simple graphics capabilities could be added to YASAI without great difficulty.

The YASAI software and documentation may be freely downloaded from the website http://business.rutgers.edu/~yasai.

	A	B	C	D	E
1	**Translator Hiring Problem**				
2					
3	Expected Demand	8		Regular Orders Filled	=MIN(B10,B12)
4	Chance of Overtime Availability	0.4		Overtime Orders Filled	=MIN(B12-E3,B13)
5	Fixed Cost per Translator	10		Orders Refused	=simoutput(B12-E3-E4,"Orders Refused")
6	Regular Order Cost	50			
7	Overtime Order Cost	75		Fixed Cost	=B10*B5
8	Revenue per Filled Order	95		Regular Cost	=E3*B6
9				Overtime Cost	=B7*E4
10	Translators Hired	=parameter({5,6,7,8,9,10},1,A10)		Total Cost	=SUM(E7:E9)
11					
12	Actual Demand	=genpoisson(B3)		Revenue	=B8*(E3+E4)
13	Overtime Translators Available	=genbinomial(B10,B4)			
14				Profit	=simoutput(E12-E10, "Profit")

Figure 2. A YASAI Model for the translator example problem.

YASAI Simulation Output

Workbook	translator-example.xls
Sheet	Model
Start Date	3/26/01
Start Time	11:42:55 AM
Run Time (h:mm:ss)	0:00:27
Scenarios:	6
Sample Size:	1000

	Parameter
Scenario	Translators Hired
1	5
2	6
3	7
4	8
5	9
6	10

Output Name	**Scenario**	**Observations**	**Mean**	**Standard Deviation**	**Minimum**	**5th Percentile**	**95th Percentile**	**Maximum**
Orders Refused	1	1000	1.825	2.269	0.000	0.000	6.000	13.000
Orders Refused	2	1000	1.082	1.755	0.000	0.000	5.000	11.000
Orders Refused	3	1000	0.611	1.338	0.000	0.000	4.000	11.000
Orders Refused	4	1000	0.318	0.984	0.000	0.000	2.000	8.000
Orders Refused	5	1000	0.148	0.696	0.000	0.000	1.000	9.000
Orders Refused	6	1000	0.070	0.501	0.000	0.000	0.000	8.000
Profit	1	1000	195.120	39.857	-5.000	130.000	235.000	275.000
Profit	2	1000	222.010	53.956	-15.000	120.000	290.000	310.000
Profit	3	1000	238.910	70.540	-25.000	110.000	325.000	365.000
Profit	4	1000	248.615	86.327	-80.000	100.000	340.000	400.000
Profit	5	1000	253.845	97.322	-90.000	90.000	375.000	435.000
Profit	6	1000	251.300	110.026	-100.000	80.000	410.000	450.000

Figure 3. YASAI output report for the example model. The 10^{th} through 90^{th} percentiles are also present in the report, but are not shown here for space reasons.

References

[1] L. Devroye, *Non-Uniform Random Variate Generation*, Springer-Verlag, 1986

[2] M. Dodge and C. Stinson, *Running Microsoft Excel 2000*, Microsoft Press, 1999.

[3] J. R. Evans and D. L. Olson, *Introduction to Simulation and Risk Analysis*, Prentice-Hall, 1998.

[4] R. Myerson, http://www.kellogg.nwu.edu/faculty/myerson/ftp/addins.htm

[5] S. L. Savage, *INSIGHT.XLA: Business Analysis Software for Microsoft Excel*, Brooks/Cole, 1998

[6] W. L. Winston, *Simulation Modeling Using @RISK: Updated for Version 4*, Brooks/Cole, 2000.