ISSUE 36 | JULY 2010

SOCIETY OF ACTUARIES Technology Section

CompAct

ELECTRONIC NEWSLETTER

QUICK LINKS

Technology Section Web site

Share Print Article Search Back issues

RAND-DUMB OR RANDOM?

by Mary Pat Campbell

First: you have no business doing Monte Carlo modeling in Excel.

Clear?

Fine.

Now that that's out of the way, we're using Excel all the time for inappropriate things, so what's a few random trials between friends?

Even if you're using a tool that isn't fully appropriate for Monte Carlo modeling in terms of efficiency, there's no reason to make the situation worse: to wit, using Excel's built-in pseudorandom number generating function RAND().

Below I will give a few reasons why using this function is a bad idea, as well as a few options to use in place of RAND(). Along the way, we will see some principles in choosing a pseudorandom number generator (PRNG) for various purposes, as well as pointing out some resources within this realm.

1. RAND() is a volatile function.

Volatile functions are ones that recalculate every time Excel recalculates. For mundane volatile functions such as TODAY(), this is not such a big deal, as they rarely change their results upon recalculation (and generally, you're only calling them a few times

Council

Links of Interest

Fiction Contest

Howard Callif, Editor

SOA Staff
Meg Weber, Staff Partner

Sue Martz,
Section Specialist

Sam Phillips, Staff Editor

within a spreadsheet–not a million times).

But every time Excel recalculates, you get a different number resulting from RAND(). And the slightest thing will make Excel recalculate. Change a font? Excel recalculates. Hit enter in an empty cell? Excel recalculates. Sneeze violently? Well, if you accidentally hit a key Excel will recalculate.

This can be a big pain if you have a lot of calls to RAND() in your spreadsheet. Every single one of the cells that contains that function will recalculate.

To be sure, there are a few ways to get around this problem. First, you can set your calculation mode to Manual, in which case the spreadsheet is recalculating only when you request it to.

Second, you can copy your "random number cells" and paste as values over those very same cells. Of course, if you need more "random" numbers, you're going to need to put the formulas back in.

2. RAND() is not seeded.

What do I mean by a seeded PRNG? I mean that you give the PRNG a specific number, a "seed," which initializes the algorithm underlying the routine and will give you the exact same sequence of pseudo-random numbers in a particular order each time you use that particular seed.

This makes it easy to reproduce a particular Monte Carlo run without needing to store all the pseudorandom variates that go into producing your Monte Carlo estimate.

Now you may wonder why it's necessary to be able to reproduce your random variates that go into your estimate:

a. Auditability–as Monte Carlo calculations become used for regulatory purposes, an auditor will need to be able to follow your work from author inputs to final numbers. This will be especially important as many of these processes will be new, requiring more detailed investigation on the part of auditors. Of course, many Monte Carlo estimates are purely for exploration or quick estimate purposes, and do not need to be audited.

b. Sensitivity Analysis/Analysis of Change–many times, even for exploratory purposes, you would like to determine the impact of model assumption changes. Say that you use a different lapse assumption for a variable annuity with a GMDB. If you don't use the same PRN (pseudo-random number) sequences, how do you know if variance of results are from the new assumption or a new pseudorandom sequence?

c. Testing goodness-of-PRNG–it's difficult to test how "good" a PRNG is if you can't even rely on what the starting sequence will be.

What should you look for in a PRNG for Monte Carlo purposes?

1. Reproducibility (i.e., seeded)
2. Good statistical behavior
3. Ease of use with your system

I already explained why reproducibility is useful, and of course ease of use is just common sense. So what do I mean by "good statistical behavior"?

In Monte Carlo modeling, you are trying to estimate something–an expected value, a standard deviation, Value at Risk or Conditional Tail Expectation, histogram of the distribution–all those sorts of things. If there is bad behavior on the part of your PRNG, your results can be biased and your estimates off in particular ways. The most obvious is when the PRNG does give a uniform distribution of numbers (which can be tested using a chi-square goodness-of-fit test). But it goes well beyond just checking uniform distribution.

Another issue is independence between consecutive PRNs, or triplets, or quadruplets, and so forth. If there are correlations between the numbers produced, then if you are using the PRNs for a sequential purpose (say, the random walk of stock returns), you can end up with biased results.

There is a suite of statistical tests that one can apply, called Diehard, that checks for issues like those above and so much more. I provide a link to a particular version, called the Dieharder suite, that is a very rigorous battery of tests. I also link to a paper about even stronger tests.

The problem is that most PRNGs will fail at least one of these tests. Is that a problem? Not necessarily. The problem would be if you're using these algorithms for cryptographic purposes–failing one of those tests means that a "randomized" crypto algorithm may be easier for someone to crack. With regards to Monte Carlo modeling, we don't need such strength–just something that gives us confidence our results will not be biased.

So let's look at some alternatives to RAND():

A. Excel's VBA function Rnd(), which is seeded. Of course, this requires coding some in VBA, but it's not too bad to implement. Problem: it's a black box (I've seen claims of knowing what code goes into this, but I've not pinned it down), but you can run the results through the randomness test suite Dieharder and see how it does for what you want.

B. The free Excel add-in NtRand. This is based on the popular-for-MC-PRNG [enough jargon yet?] Mersenne Twister, and is fairly easy to use with Excel. What's more, it will produce correlated normal variates, which is pretty handy if you don't want to be doing a Cholesky decomposition yourself. NtRand can be found here: http://www.ntrand.com/

C. Use your own code in VBA. You can find standard code for PRNGs at Numerical Recipes (yes, they're not in VBA, but it's relatively easy to take the FORTRAN code and do a transfer). As well, sometimes one finds PRNGs explicitly coded in spreadsheets online (such as the Academy's last release of the interest rate generator). Reusing other people's code is a long-time practice of mature programmers; just be sure you test it.

D. Random.org–now, interestingly, this does not give you pseudorandom sequences–it's based on observed noise in the atmosphere. This is not easily used with Excel, though.

By the way, I do use RAND() all the time–to create simple illustrative calculations to show how various formulas work in a stochastic cash-flow model. It's fine for a quick simulation or two, but not serious Monte Carlo estimation.

I also found a RAND() substitute that was, if anything, worse than the original. I leave it as an exercise for the reader of this article to see what's bad about the following:

```
=(NOW()*100000 - INT(NOW()*100000))
```

Information about why the author of this formula thought this was a good idea resides at this link: http://support.microsoft.com/kb/44738/en-us

If you'd like to argue in favor of this approach, buy me a drink, and I'll listen to your defense of that monstrosity. But I'm going to need more than just one drink to sit still for it.

Thanks to the community at the Actuarial Outpost for discussion of PRNG issues in Excel.

Links:

Diehard(er) tests

More on the Diehard tests

NtRand

Numerical Recipes

Random.org: come on, you can guess this one

On some difficult-to-pass randomness tests.

Mary Pat Campbell can be contacted at marypat.campbell@gmail.com.