Random Thoughts [Updated]

Posted on **November 26, 2014**

A long time ago, I conceived of an algorithm for generating random numbers, inspired by the famous Fibonacci sequence of numbers. The one that goes like this:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765…

The basic idea behind the sequence was that each number is the sum of the previous two. My idea was what if an array of counters were maintained in a ring, where each value was the sum of the previous two? The numbers would cascade and overflow and become very chaotic. Still I was never sure if such an algorithm would actually work. I implemented it and it did'nt. The sums sped off until they were all zero. I fixed the algorithm by shifting one of the terms to the right by one bit and now it seemed to work. Still, I was haunted by that line from the essential text: "The Art of Computer Programming" that says (after expounding the flaws in many fine algorithms thought to be suitable):

The moral of this story is that random numbers should not be generated with a method chosen at random.

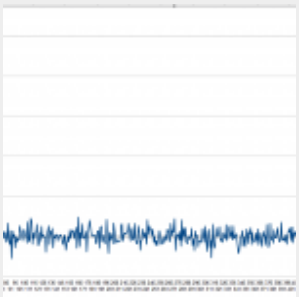
Donald Knuth

I had no way of determining the quality of the numbers generated, so my code languished on the back burner. I realize now that what I was lacking was an understanding of statistics and statistical analysis. I went to university, but I am sad to say that I never took a stats course. I really wish I had.

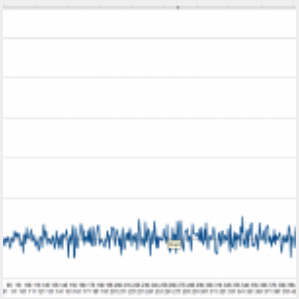
Normally, I can figure things out from Wikipedia, but maybe my cognitive facilities are fading because it was mostly just greek to me. A few things stood out. The Chi² measure is often used in grading pseudo random number generators. I created a test bed and ran 10,000,000 iterations of a six way test (like a cubic dice) using both my creation and the one built into the Ruby language (ruby 1.9.3p484 (2013-11-22) [i386-mingw32]). Lower values of Chi² indicate closer conformance to the expected distribution of results. The values obtained were:

Generator	Chi Squared Value
Home brewed	6.040861333333967e-08
Built In	1.7266209333333707e-07

Now, I still don't know how to interpret these results fully, but my home brew generator had lower scores which means the values were more evenly distributed. Which is good… I think. I was still very uncertain. The second test I tried was an Auto-correlation test. The nature of this test is too complex to explain here except to say that it is excellent in finding hidden patterns in seemingly random data. Here are the results:



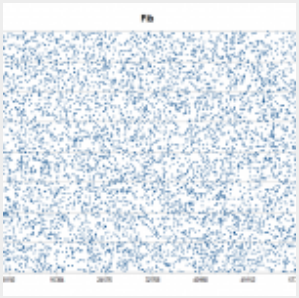
— Fibonacci Generator



— Internal Generator

The correlation is high when the data is compared literally to itself. This is expected. Note that as soon as the comparison starts to shift, correlation collapses and becomes essentially random. This is also expected. So far my homebrew code is holding out.

The final test is a simple scatter graph. The random number generator is used to generate random coordinates that are plotted on a graph. Patterns in the data will show up as patterns in the graph that may be visible. Here is the result of this test.



— Fibonacci Scatter

And this is where things stand so far. All the tests I have run give me the confidence that my little algorithm produces random numbers at least as well as commercial generators, while using only addition and divide by 2 and no higher math operators. The code for this project may be found at: https://github.com/PeterCamilleri/fibonacci_rng. The code is also available as a Ruby Gem at: https://rubygems.org/gems/fibonacci_rng. Take a look at the code if you like, use for any purpose, just don't expect any guarantees! After all, 9,9,9,9,9,9,9,9… *could* be a random sequence!

As always, comments and suggestions are most welcome.

Best regards;

Peter Camilleri (aka Squidly Jones)

[Update] November 29, 2014

I recently watched the most fascinating video about the cracking of the Enigma cipher in World War 2. Here is a link to Turing's Enigma Problem – Computerphile. The most fascinating thing for me was the analysis of the enemy's efforts to make the code harder to crack. These changes resulted in inconsistencies that actually made it ***easier*** to crack! It re-affirms Knuth's assertion that random things done at random are NOT assured to increase security!

This entry was posted in **Announcements** by **Peter Camilleri**. Bookmark the **permalink [http://teuthida-technologies.com/?p=1583]** .