`random.org`: Introduction to Randomness and Random Numbers

Mads Haahr

June 1999*

Background

Oh, many a shaft at random sent
Finds mark the archer little meant!
And many a word at random spoken
May soothe, or wound, a heart that's broken!"
— Sir Walter Scott

Randomness and random numbers have traditionally been used for a variety of purposes, for example games such as dice games. With the advent of computers, people recognized the need for a means of introducing randomness into a computer program. Surprising as it may seem, however, it is difficult to get a computer to do something by chance. A computer running a program follows its instructions blindly and is therefore completely predictable.

Computer engineers chose to introduce randomness into computers in the form of pseudo-random number generators. As the name suggests, pseudo-random numbers are not truly random. Rather, they are computed from a mathematical formula or simply taken from a precalculated list. A lot of research has gone into pseudo-random number theory and modern algorithms for generating them are so good that the numbers look exactly like they were really random. Pseudo-random numbers have the characteristic that they are predictable, meaning they can be predicted if you know where in the sequence the first number is taken from. For some purposes, predictability is a good characteristic, for others it is not.

Random numbers are used for computer games but they are also used on a more serious scale for the generation of cryptographic keys and for some classes of scientific experiments. For scientific experiments, it is convenient that a series of random numbers can be replayed for use in several experiments, and pseudo-random numbers are well suited for this purpose. For cryptographic use, however, it is important that the numbers used to generate keys are not just seemingly random; they must be truly unpredictable.

*This vignette for the R package `random` is a transcribed version of the original essay by Mads Haahr that was published in 1999 and is still available at `http://random.org/essay.html`. Hyperlinks from the original html document are also present in the pdf version. All errors introduced in the transformation from html to LaTeX are solely the responsibility of the R package maintainer, Dirk Eddelbuettel.

True Random Numbers

"It is impossible to predict the unpredictable."
— Don Cherry

Cryptographic algorithms come in a variety of flavors. Some are strong (meaning difficult to crack) but make substantial demands to processing power and key management. Others are weak (meaning easier to crack) but generally less demanding and therefore better suited for some applications. All strong cryptography requires true random numbers to generate keys, but how many depends on the encryption scheme. The strongest possible method, One Time Pad (OTP for short) encryption, is the most demanding of all; it requires as many random bits as there are bits of information to be encrypted.

True random numbers are typically generated by sampling and processing a source of entropy outside the computer. A source of entropy can be very simple, like the little variations in somebody's mouse movements or in the amount of time between keystrokes. In practice, however, it can be tricky to use user input as a source of entropy. Keystrokes, for example, are often buffered by the computer's operating system, meaning that several keystrokes are collected before they are sent to the program waiting for them. To the program, it will seem as though the keys were pressed almost simultaneously.

A really good source of entropy is a radioactive source. The points in time at which a radioactive source decays are completely unpredictable, and can be sampled and fed into a computer, avoiding any buffering mechanisms in the operating system. In fact, this is what the HotBits people at Fourmilab in Switzerland are doing. Another source of entropy could be atmospheric noise from a radio, like that used here at random.org, or even just background noise from an office or laboratory. The lavarand people at Silicon Graphics have been clever enough to use lava lamps to generate random numbers, so their entropy source not only gives them entropy, it also looks good! The latest random number generator to come online (both lavarand and HotBits precede random.org) is Damon Hart-Davis' Java EntropyPool which gathers random bits from a variety of sources including HotBits and random.org, but also from web page hits received by the EntropyPool's web server.

Random.org

"This is the third time; I hope good luck lies in odd numbers....
There is divinity in odd numbers, either in nativity, chance, or death."
— William Shakespeare

The idea of using atmospheric noise to generate random numbers came up when some friends and I were building a prototype of an online gambling system. Using noise in this way isn't a particularly original idea, though. Other people thought of it before us, and as mentioned above, there are already several public random number services out there, some of which use more advanced methods. So, why yet another? The most important reason is because it was fun to make. The second most important reason is that existing services are mostly for educative purposes — and for fun! I hope random.org will prove itself informative and fun but also useful for certain (non-critical) applications that need random numbers. Random.org is the only service I know of which offers a large (16K) block of numbers at once and which has a CORBA interface.

The way the random.org random number generator works is quite simple. A radio is tuned into a frequency where nobody is broadcasting. The atmospheric noise picked up by the receiver is fed into a Sun SPARC workstation through the microphone port where it is sampled by a program as an eight bit mono signal at a frequency of 8KHz. The upper seven bits of each sample are discarded immediately and the remaining bits are gathered and turned into a stream of bits with a high content of entropy. Skew correction is performed on the bit stream, in order to ensure that there is an approximately even distribution of 0s and 1s.

The skew correction algorithm used is based on transition mapping. Bits are read two at a time, and if there is a transition between values (the bits are 01 or 10) one of them - say the first - is passed on as random. If there is no transition (the bits are 00 or 11), the bits are discarded and the next two are read. This simple algorithm was originally due to Von Neumann and completely eliminates any bias towards 0 or 1 in the data. It is only one of several ways of performing skew correction, though, and has a number of drawbacks. First, it takes an indeterminate number of input bits. Second, it is quite inefficient, resulting in the loss of 75% of the data, even when the bit stream is already unbiased. RFC1750 discusses skew correction in general and lists this method as well as three others. Paul Crowley discusses skew correction for use with Geiger counters and also maintains a page with implementations of a number of skew correction algorithms.

Statistics

"There are three kinds of lies: lies, damn lies, and statistics."
– -Benjamin Disraeli

The quality of random numbers can be measured in a variety of ways. One common method is to compute the information density, or entropy, in a series of numbers. The higher the entropy in a series of numbers is, the more difficult it is to predict a given number on the basis of the preceding numbers in the series. A sequence of good random numbers will have a high level of entropy, although a high level of entropy does not guarantee randomness. (As an example, a file compressed with a software compressor such as gzip or winzip has a high level of entropy, but the data is highly structured and therefore not random.) Hence, for a thorough test of a random number generator, computing the level of entropy in the numbers alone is not enough.

A number of basic tests, such as the entropy test described above, were implemented by John Walker from Fourmilab in form of his ent program. Here is what John's program said about one megabyte of numbers from random.org:

```
Entropy = 7.999805 bits per character.

Optimum compression would reduce the size
of this 1048576 character file by 0 percent.

Chi square distribution for 1048576 samples is 283.61, and randomly
would exceed this value 25.00 percent of the times.

Arithmetic mean value of data bytes is 127.46 (127.5 = random).
Monte Carlo value for PI is 3.138961792 (error 0.08 percent).
Serial correlation coefficient is 0.000417 (totally uncorrelated = 0.0).
```

John's documentation for the ent program contains detailed explanations of each test, but they generally show the random.org numbers to be quite good. The Chi Square test is probably the most commonly used test for randomness, and numbers pass this test if the percentage figure falls between 10% and 90%. Every single number generated at random.org is subjected to John Walker's tests and the results are logged. The last twenty log entries at any time can be seen on the online statistics page.

There are also other test suites for testing the quality of random numbers. A popular one is the Diehard program by George Marsaglia. Unfortunately, this test suite seems not to have been maintained for the last few years.

In April 2001, Louise Foley, a student of statistics at Trinity College, Dublin submitted a comprehensive analysis of random.org's numbers as her final year project. As part of the project, Louise identified and implemented a series of five tests particularly suitable for true random numbers and compared the quality of the numbers generated at random.org with those generated by Silicon Graphics' lavarand generator. Both generators passed the tests. Louise's report is available online.

Suggested Online Reading

- The Cryptography FAQ is a good introduction to cryptography.
- RFC1750 discusses randomness for computer security purposes.
- The Snake Oil FAQ warns that crypto software varies a great deal in quality and gives guidelines as how to judge a given product.
- True Random Numbers contains a review of several commercial hardware devices for true random number generation.
- Randomness for Crypto contains a large selection of links to online articles on randomness.