

CDM

(Pseudo-) Randomness

Klaus Sutner
Carnegie Mellon University
www.cs.cmu.edu/~sutner

randomness

1

Battleplan

- Randomness in Physics
- Formalizing Randomness
- Practical RNGs

randomness

2

Intuitive Randomness

randomness

3

Generating Random Numbers

Random numbers (or random bits) are a crucial ingredient in many algorithms.

For example, all industrial strength primality testing algorithms rely on the availability of random bits. Modern cryptography is unimaginable without randomness.

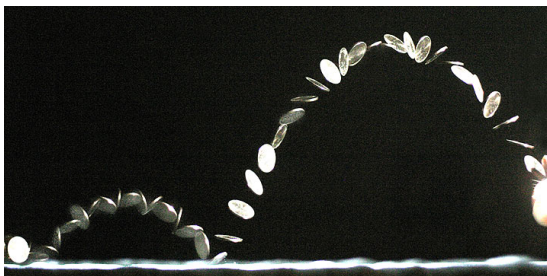
Since computers are deterministic devices (more or less), it is actually not all that easy to produce random bits using a computer: whatever program we run will produce the same bits if we run it again.

To make matters worse, it is rather difficult to even say exactly what is meant by a sequence of random bits. Of course, intuitively we all know what randomness means, right?

randomness

4

Flipping Coins



randomness

5

How Random Is It?

Is the randomness in a coin-toss real or is it actually confined to just the initial conditions?

Persi Diaconis, a Stanford mathematician and highly accomplished professional magician, supposedly can consistently produce ten consecutive heads flipping a coin – by carefully controlling the initial conditions.



Rolling Dice



Lava Lamps

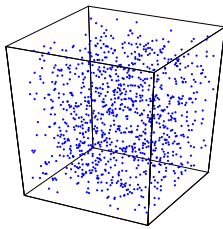


Krypton-85

Radioactivity is another great source of randomness – except that no one likes to keep a lump of radioactive material and a Geiger-Müller counter on their desk.

But if you can keep the radioactive stuff someplace else and obtain the random bits over the web it's not so bad.

Here are true random bits from www.fourmilab.ch/hotbits.



Pre-History

In the olden days, the RAND Corporation used a kind of electronic roulette wheel to generate a million random digits (rate: one per second).

In 1955 the data were published under the title:

A Million Random Digits With 100,000 Normal Deviates

"Normal deviates" simply means that the distribution of the random numbers is bell-shaped rather than uniform. But the New York Public Library shelved the book in the psychology section.

The RAND guys were surprised to find that their original sequence had several defects and required quite a bit of post-processing before it could pass muster as a random sequence. This took years to do.

Available at <http://www.rand.org/publications/classics/randomdigits>.

Fiat Lux

Incidentally, Noll and Cooper at Silicon Graphics discovered one day that the pretty lava lamps were completely irrelevant: they could get even better random bits with the lens cap on (there is enough noise in the circuits to get good randomness).

Another way to use light, very much unlike the original lava lamp system, is to exploit an elementary quantum optical process: a photon hitting a semi-transparent mirror either passes or is reflected.

The Quantis systems was developed at the University of Geneva, the first practical model was released in 1998.

Note that quantum physics is the only part of physics that claims that the outcome of certain processes is random (which is why Einstein always disliked quantum physics rather strongly).

The Magic Device



Quantis RNG

Features

- True quantum randomness
- High bit rate of 4Mbits/sec (up to 16Mbits/sec for PCI card)
- Low-cost device
- Compact and reliable
- PCI card comes with drivers for Windows (2000/XP), Linux (2.4.x, 2.6.x), FreeBSD (4.x, 5.x) and Solaris (sparc & x86, 8, 9).

Applications

- Numerical Simulations
- Statistical Research
- Lotteries and gambling
- *Cryptography*

Formalizing Randomness

Intuition

Which 30-bit sequence below is random?

1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0
 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1
 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1

Incidentally, each one has probability 2^{-30} .

Defining Randomness

What exactly does it mean for a sequence of bits to be random?

The point here is that we would like a purely mathematical definition without any recourse to physics. We could, of course, try to give a formal definition of, say, a coin flip. Alas, axiomatizing physics is an unsolved problem at this point – and has been since 1900 when Hilbert put it on his famous list (item 6: axiomatize probability and mechanics).

What we can do easily instead is to home in on a few central properties of randomness – properties that we feel intuitively to be associated with randomness rather than being able to derive them from first principles. But one has to be careful, intuition is not always reliable.

It is somewhat easier to consider infinite sequences $\vec{x} \in 2^{\omega}$ rather than finite ones. Infinity is often an excellent approximation for finiteness.

Obstructions to Randomness

Let's turn around for the moment and look at properties that would disqualify a sequence from being random in any intuitive sense of the word. Here are two obvious potential problems.

- Bias (or skew): the probability of a 0 is not $1/2$.
- Correlation: the $i + 1$ st bit is not independent from the i th bit.

Fortunately, we don't need to achieve perfection in either category: there are algorithms that can turn a slightly biased and/or correlated sequence into an unbiased and uncorrelated one.

Here is a simple though not entirely satisfactory method to eliminate bias due to von Neumann. Dealing with correlated bits is harder, we won't get involved.

Removing Bias

Suppose we have an imperfect source of random bits (real world sources typically fall into this category) that are already independent but that the probability of a 0 is $1/2 + \epsilon$. To eliminate this bias, John von Neumann suggested the following algorithm:

- Read the bits, two at a time.
- Skip 00 and 11.
- For 01 and 10 output the first bit.

The probabilities of all 2-blocks are easily computed since we assume independence:

$$\begin{array}{lll} 00 & (1/2 + \epsilon)^2 & = 1/4 + \epsilon + \epsilon^2 \\ 01 & (1/2 + \epsilon)(1/2 - \epsilon) & = 1/4 - \epsilon^2 \\ 10 & (1/2 - \epsilon)(1/2 + \epsilon) & = 1/4 - \epsilon^2 \\ 11 & (1/2 - \epsilon)^2 & = 1/4 - \epsilon + \epsilon^2 \end{array}$$

The resulting sequence has no bias, as needed.

Density and the Law of Large Numbers

Is it really clear what bias means? What is the "probability of getting a 0?"

Definition 1. *Density*

Let $\vec{x} \in 2^\omega$ and define the **density** of \vec{x} up to n to be $D(\vec{x}, n) = \frac{1}{n} \sum_{i < n} x_i$.

The **limiting density** of \vec{x} is $\lim_n D(\vec{x}, n)$, if that limit exists.

To be unbiased means to obey the Law of Large Numbers: the limiting density is $1/2$ (though one might wonder why the limit should actually exist in the strict sense of convergence in analysis).

Also, one would want a certain degree of convergence: $D(\vec{x}, n)$ should tend to $1/2$ but not too quickly: rapid convergence would be suspicious, it should take a while before we get close to $1/2$.

Counting Blocks

We can look at density not just for single bits but for arbitrary finite blocks $w \in 2^m$; the number of occurrences of a particular block w in $x_1 x_2 \dots x_n$ should approach $n/2^m$.

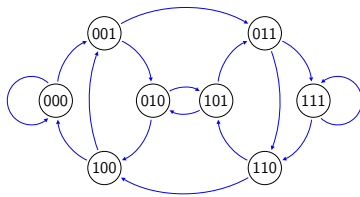
A good way of thinking about this is to have \vec{x} trace a path in a de Bruijn graph. Recall that the de Bruijn graph of order k is defined as

$$\mathcal{B}_k = \langle 2^k, E \rangle$$

$$E = \{ (au, ub) \mid a, b \in 2, u \in 2^{k-1} \}$$

If we slide a window of width k over \vec{x} we obtain a sequence of nodes in \mathcal{B}_k , and the sequence is in fact a path in the graph.

De Bruijn Graph of Order 3

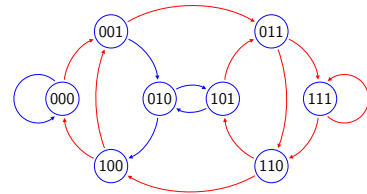


De Bruijn graphs are Hamiltonian, and the Hamiltonian cycles are called de Bruijn sequences: they contain every block of length k exactly once and have length $2^k + k - 1$.

Any finite or infinite sequence of bits (of length at least k) traces a path in the graph. If the sequence is unbiased the path hits each node equally often in the limit.

Example

The first 30 bits of the binary expansion of $\sqrt{5}$ from above produces



After 59 bits all edges lie on the path.

Exercise. Suppose we have a de Bruijn sequence X of order k and let $\vec{x} = X^\omega$. That that \vec{x} has limiting density $1/2^m$ for all blocks of length $m \leq k$ but not for larger blocks.

Decimation

How about using Roman military traditions to define randomness?

In 1919 Richard von Mises suggested a notion of randomness based on the limiting density of the sequence itself and certain derived subsequences.

The idea is that "reasonable" subsequences of the given sequence should also have limiting density $1/2$.



Definition 2. An infinite sequence $\vec{x} \in 2^\omega$ is **Mises random** if the limiting density of any subsequence (x_{i_j}) is $1/2$ where the subsequence is selected by a *Auswahlregel*.

Auswahlregeln

So what on earth is a *Auswahlregel*, a selection rule?

Intuitively, the following all should have limiting density $1/2$:

$$x_0, x_1, x_2, \dots, x_n, \dots$$

$$x_0, x_2, x_4, \dots, x_{2n}, \dots$$

$$x_1, x_4, x_7, \dots, x_{3n+1}, \dots$$

$$x_0, x_1, x_4, \dots, x_{n^2}, \dots$$

In fact, we might want for any reasonable strictly monotonic function $f : \mathbb{N} \rightarrow \mathbb{N}$ that

$$x_{f(0)}, x_{f(1)}, x_{f(2)}, \dots, x_{f(n)}, \dots$$

has limiting density $1/2$.

Mises' Definition

The actual definition used by Mises is a bit more technical. He considers functions

$$\Phi : 2^* \rightarrow \mathbb{B}$$

and selects x_n to be in the subsequence \vec{x}_{Φ} if $\Phi(x_0x_1 \dots x_{n-1})$.

However, there is one big caveat: the function Φ must be defined without any knowledge of \vec{x} : otherwise we can simply pick a subsequence of all 0's.

Now suppose we have a countable system of Auswahlregeln and our sequence passes all these tests. In other words, for all Φ we have

$$\lim D(\vec{x}_{\Phi}, n) = 1/2.$$

Then \vec{x} is Mises-random. One can show that for any countable collection of Auswahlregeln there are always uncountably many sequences that are random in this sense.

Sounds all eminently reasonable.

Ville's Counterexample

Unfortunately, in 1939 J. Ville showed that for any countable system of Auswahlregeln there is always a sequence \vec{x} that passes all the tests (i.e., the limiting density is 1/2 for all these subsequences) but that is nonetheless biased towards 1.

More precisely, it was known that a random sequence should have

$$\limsup_n \sqrt{\frac{2n}{\log \log n}} (D(\vec{x}, n) - 1/2) = 1$$

$$\liminf_n \sqrt{\frac{2n}{\log \log n}} (D(\vec{x}, n) - 1/2) = -1$$

and Ville's example violated the second condition.

Martin-Löf Randomness

Still, the idea of using some kind of test to rule out non-random sequences is most valuable – we just need to come up with the right kind of test. As we will see, computability plays a major role so we need to make sure we are dealing with finitary objects.

First off, in order to describe a collection of infinite sequences $S \subseteq 2^\omega$ we can think of these sequences as branches in the infinite complete binary tree. The collection of all initial segments $x \sqsubset \vec{x}$ of the branches forms a collection S_0 of binary strings that is closed under prefixes.

For example, if

$$S = \{0^k 1^\omega \mid k \geq 0\}$$

then

$$S_0 = \{0^k 1^l \mid k, l \geq 0\}$$

Note that S cannot be recovered from S_0 . Why?

Tests

Consider a descending chain of sets of binary words

$$K_0 \supseteq K_1 \supseteq K_2 \supseteq \dots \supseteq K_n \supseteq \dots$$

where each K_n is closed under prefixes.

Furthermore, let $\mu(K_n) \leq 2^{-n}$ so that these sets are not too fat.

Then the infinite branches that lie in all these trees are considered to be failures as far as randomness goes: they can be described by the sequence of test sets.

The choice of the K_n is obviously important here, and we have to make sure they are not too complicated (otherwise we could refute the existence of any random sequence).

Example: Density

For example, to check that the limiting density is at least $1/2 + \epsilon$ we can use the test sets

$$K_n = \bigcup_{k \geq n} \{x \sqsubset \vec{x} \mid D(\vec{x}, k) \geq 1/2 + \epsilon\}$$

If \vec{x} has limiting density at least $1/2 + \epsilon$ it traces an infinite branch in the tree $\bigcap_n K_n$ and is thus eliminated.

Of course, we need to perform countably many such tests to make sure that the limiting density is in fact equal to 1/2 (actually, not; see below).

Digression: Semi-Decidable Sets

Recall that a set $A \subseteq 2^*$ is decidable iff there is a decision algorithm for A iff there is a TM that, on input x , halts with output 1 if $x \in A$, and halts with output 0 otherwise.

Definition 3. A set is **semi-decidable** if there is a TM that, on input x , halts if x belongs to the set, and fails to halt otherwise.

Lemma 1. The Halting set K is semi-decidable.

Proof. Use a UTM to simulate the computation of M_e on input e . If M_e halts in the simulation, halt too. Otherwise just keep running forever. \square

Decidable vs. Semi-Decidable

Lemma 2. *A set is decidable if, and only if, the set and its complement are both semi-decidable.*

Proof. Let A be decidable. The TM M that decides membership in A can easily be turned into machines that semi-decide membership in A and $2^* - A$. E.g., for A run M ; if it halts on 1 also halt, but if it halts on 0 go into an infinite loop.

For the opposite direction, suppose we have machines M_1 and M_2 that semi-decide A and $2^* - A$. Given x , run both machines in parallel on x . One of them must halt, output 0 or correspondingly and halt. \square

It's crucial for this argument that we can run two TMs in parallel (time-sharing).

Enumerations

Let us say that

Definition 4. *A TM M enumerates a set A if, possibly running forever, it writes the elements of A on a special write-only output tape, one after the other.*

output tape: $\#a_0\#a_1\#a_2\#\dots\#a_n\#\dots$

The rules are:

- Every element must appear at some point, and nothing else may appear.
- Elements may be repeated.
- The elements do not have to appear in any special order, though.

Of course, for infinite sets the computation never stops.

Recursively Enumerable Sets

Definition 5. *A set recursively enumerable (r.e.) if it can be enumerated by a Turing machine.*

Lemma 3. *The r.e. sets are precisely the semi-decidable sets.*

Proof. Given a TM that enumerates A we can easily build a TM that semi-decides A : enumerate away and check if x appears. If so, halt.

If A is semi-decidable via TM M we can run M in parallel on multiple inputs. At level s we run M for s steps on $x = 0, 1, \dots, s$. If M halts on any of these inputs we write them on the output tape. Then we continue with level $s + 1$. \square

Sequential Test

Back to randomness. The key in designing a good randomness test lies in imposing a computability constraint.

Definition 6. *A sequential test has the additional property that*

$$K = \{ (n, x) \mid x \in K_n \subseteq 2^* \}$$

is semi-decidable.

Here is an amazing result that shows that in essence we only need to deal with a single test.

Theorem 1. *There is a universal sequential test U such that for any sequential test K we have $K_{n+c} \subseteq U_n$ for some constant c .*

The proof uses the existence of a universal Turing machine and is not particularly difficult.

The Definition

Definition 7. *An infinite sequence \vec{x} is Martin-Löf random if it passes a universal sequential test.*

This definition is very strongly supported by the empirical fact that any practical test of randomness in ordinary probability theory can be translated into a sequential test. So, we are just dealing with all of these tests at once (plus all conceivable others).

Furthermore, Martin-Löf randomness provides a good conceptual framework and can be used to prove real theorems about randomness. But, the definition does not yield any methods to construct a random sequence. Aux contraire:

Theorem 2. *Any Martin-Löf random sequence fails to be computable.*

Exercises

Exercise 1. *Give a detailed proof that limiting density can be checked by a suitable test set. In particular, make sure that semi-decidability holds.*

Exercise 2. *Show how to check for the frequencies of blocks of arbitrary finite length in a sequential test.*

Exercise 3. *Show that every Martin-Löf random sequence fails to be computable. Assume a random sequence is computable and show how to construct a test that rejects it.*

Exercise 4. *Show that there are uncountably many Martin-Löf random sequences (in fact, they form a set of measure 1).*

Generating PRNs

Pseudo-Randomness

In the real world, one often makes do with a *pseudo-random number generator (PRNG)* based on iteration: the pseudo-random sequence is the orbit of a some initial element under some function.

x_0 chosen somehow, at random :-)

$$x_{n+1} = f(x_n)$$

where f is easily computable, typically using arithmetic and some bit-plumbing. Of course, we are taking a huge step away from real randomness here, this sequence would perish miserably when exposed to a Martin-Löf test.

Still, there are many applications where this type of pseudo-randomness is sufficient.

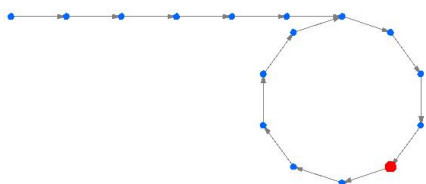
However, cryptography is usually not among them!

The Sad Truth

Anyone attempting to produce random numbers by purely arithmetic means is, of course, in a state of sin.

John von Neumann

The function f typically operates on some finite domain such as 32-bit words. But then every orbit looks like so:



So we can only hope to make f fast and guarantee long periods.

The Seed

Needless to say, running a PRNG twice with the same seed x_0 is going to produce exactly the same "random" sequence.

This can be a huge advantage, because it makes computations that are base on the random numbers reproducible (important for debugging and verification).

More generally, if we are willing to pay for a truly random seed we would hope that the iterative PRNG would amplify the randomness: we provide m truly random bits and get back n high-quality pseudo-random bits where $n \gg m$.

Hence PRNGs reduces the need for truly random bits but does not entirely eliminate them.

Linear Congruential Generator

A typical example: a simple affine map modulo m .

$$x_{n+1} = a x_n + b \pmod{m}$$

The trick here is to choose the proper values for the coefficients. Can be found in papers and on the web.

A choice that works reasonably well is

$$a = 1664525, b = 1013904223, m = 2^{32}$$

Note that a modulus of 2^{32} amounts to unsigned integer arithmetic on a 32-bit architecture, so this is implementation-friendly.

Multiplicative Congruential Generator

Omit the additive offset and use multiplicative constants only.

If need be, use a higher order recurrence.

$$x_n = a_1 x_{n-1} + a_2 x_{n-2} + \dots + a_k x_{n-k} \pmod{m}$$

For prime moduli one can achieve period length $m^k - 1$.

This is almost as fast and easy to implement as LCG (though there is of course more work involved in calculating modulo a prime).

Note, though, that there is more state: we need to store all of $x_{n-1}, x_{n-2}, \dots, x_{n-k}$.

Inverse Congruential Generator

Choose the modulus m to be a prime number and write

$$\bar{x} = \begin{cases} 0 & \text{if } x = 0, \\ x^{-1} & \text{otherwise.} \end{cases}$$

Then we can define a pseudo-random sequence by

$$x_{n+1} = a \bar{x}_n + b \pmod{m}$$

Computing the inverse can be handled by the extended Euclidean algorithm.

Again, it is crucial to choose the proper values for the coefficients.

Mersenne Twister

Fairly recent (1998) method by Matsumoto and Nishimura, seems to be the tool of choice at this point.

- Has huge period of $2^{19937} - 1$, a Mersenne prime.
- Is statistically random in all the bits of its output (after a bit of post-processing).
- Has negligible serial correlation between successive values.
- Only statistically unsound generators are much faster.

The method is very clever and not exactly obvious.

The algorithm works on bit-vectors of length w (typically 32 or 64).

Let k be the degree of the recursion, and choose $1 \leq m < k$ and $0 \leq r < w$.

The MT Recurrence

Recall that $x_i \in 2^w$.

Define the join $\text{join}(x, y)$ of $x, y \in 2^w$ to be the first $w - r$ bits of x followed by the last r bits of y .

$$x_{n+k} = x_{n+m} + \text{join}(x_n, x_{n+1}) \cdot A$$

where A is a sparse companion-type matrix that makes it easy to perform the vector-matrix multiplication. E.g.

$$A = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & & \ddots & \ddots & \\ 0 & 0 & 0 & \dots & 1 \\ a_{w-1} & a_{w-2} & a_{w-3} & \dots & a_0 \end{pmatrix}$$

Good Parameters

Here is an excellent choice for the parameters:

$$w = 32, k = 624, m = 397, r = 31$$

and the A matrix is given by $\vec{a} = 0x9908B0DF$.

Note that this requires a bit of storage for the state: we are dealing with a recurrence of order 624 (need an array of 624 words).

This choice achieves the theoretical upper bound for the period:

$$2^{wk-r} = 2^{19937} \approx 4.32 \times 10^{6001}.$$

After a little bit of post-processing this methods produces very high quality pseudo-random numbers, and is not overly costly.

Summary

- Attempts to define randomness
 - Block Density
 - Van Mises: decimation
 - Martin-Löf: effective tests
- Practical RNGs
 - Linear congruential generators
 - Multiplicative congruential generators
 - Inverse congruential generators
 - Mersenne twister