Stochastic Simulation
Random number generation

Bo Friis Nielsen

Applied Mathematics and Computer Science
Technical University of Denmark
2800 Kgs. Lyngby – Denmark
Email: bfn@imm.dtu.dk
Random number generation

- Uniform distribution
- Number theory
- Testing of random numbers
- Recommendations of random number generators
Summary

- We talk about generating pseudo random numbers
- There exist a large number of RNG’s
- ... of varying quality
- Don’t implement your own, except for fun or as a research project.
- Built-in RNG’s should be checked before use
- ... at least in general-purpose development environments.
- Scientific computing environments typically have state-of-the-art RNG’s that can be trusted.
- Any RNG will fail, if the circumstances are extreme enough.
History/background

- The need for random numbers evident
- Tables
- Physical generators. Lottery machines
- Need for computer generated numbers
Definition

- Uniform distribution \([0; 1]\).
- Randomness (independence).
- One basic problem is computers do not work in \(\mathbb{R}\) Random numbers: A sequence of independent random variable, \(U_i\), uniformly distributed on \(]0, 1[\).
- Generate a sequence of independently and identically distributed \(U(0, 1)\) numbers.
**Random generation**

Mechanics devices:
- Coin (head or tail)
- Dice (1-6)
- Monte-Carlo (Roulette) wheel
- Wheel of fortune
- Deck of cards
- Lotteries (Dansk tipstjeneste)

Other devices:
- Electronic noise in a diode or resistor
- Tables of random numbers
Definition of a RNG

An RNG is a computer algorithm that outputs a sequence of reals or integers, which appear to be

- Uniformly distributed on \([0; 1]\) or \(\{0, \ldots, N - 1\}\)
- Statistically independent.

Caveats:

- “Appear to be” means: The sequence must have the same relevant statistical properties as I.I.D. uniformly distributed random variables
- With any finite precision format such as double, uniform on \([0; 1]\) can never be achieved.
1. Four digit integer  
   (output divide by 10000)
2. square it.
3. Take the middle four digits
4. repeat

<table>
<thead>
<tr>
<th></th>
<th>$Z_i$</th>
<th>$U_i$</th>
<th>$Z_i^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7182</td>
<td>0.7182</td>
<td>51,581,124</td>
</tr>
<tr>
<td>1</td>
<td>5811</td>
<td>0.5811</td>
<td>33,767,721</td>
</tr>
<tr>
<td>2</td>
<td>7677</td>
<td>0.7677</td>
<td>58,936,329</td>
</tr>
<tr>
<td>3</td>
<td>9363</td>
<td>0.9363</td>
<td>87,665,769</td>
</tr>
<tr>
<td>4</td>
<td>6657</td>
<td>0.6657</td>
<td>44,315,649</td>
</tr>
<tr>
<td>5</td>
<td>3156</td>
<td>0.3156</td>
<td>09,960,336</td>
</tr>
</tbody>
</table>

Might seem plausible - but rather dubious
Fibonacci

Leonardo of Pisa (pseudonym: Fibonacci) dealt in the book "Liber Abaci" (1202) with the integer sequence defined by:

\[ x_i = x_{i-1} + x_{i-2} \quad i \geq 2 \quad x_0 = 1 \quad x_1 = 1 \]

**Fibonacci generator.** Also called an additive congruential method.

\[ x_i \equiv \text{mod}(x_{i-1} + x_{i-2}, M) \]

\[ U_i = \frac{x_i}{M} \]

where \( x = \text{mod}(y, M) \) is the modulus after division i.e. \( y - nM \) where \( n = \lfloor y/M \rfloor \) Notice \( x_i \in [0, M-1] \). Consequently, there is \( M^2 - 1 \) possible starting values.

Maximal length of period is \( M^2 - 1 \) which is only achieved for \( M = 2, 3 \).
The generator

$$U_i \mod (aU_{i-1}, 1) \quad U_i \in [0, 1]$$

illustrates the principle provided $a$ is large, the last digits are retained.

Can be implemented as ($x_i$ is an integer)

$$x_i \mod (ax_{i-1}, M) \quad U_i = \frac{x_i}{M}$$

Examples are $a = 23$ and $M = 10^8 + 1$. 

**Congruential Generator**
Mid conclusion

- Initial state determine the whole sequence
- How many different cycles
- Length of each cycle

If \( x \) can take N values, then the maximum length of a cycle is \( N \).
Properties for a Random generator

- Cycle length
- Randomness
- Speed
- Reproducible
- Portable
**Linear Congruential Generator**

LCG are defined as

\[ x_i = \text{mod}(ax_{i-1} + c, M) \quad U_i = \frac{x_i}{M} \]

for a multiplier \( a \), shift \( c \) and modulus \( M \).

We will take \( a, c \) and \( x_0 \) such \( x_i \) lies in \((0, 1, \ldots, M-1)\) and it looks random.

**Example:** \( M = 16, a = 5, c = 1 \)

With \( x_0 = 3 \):

\[ 0 1 6 15 12 13 2 11 8 9 14 7 4 5 10 3 \]
Theorem 1

Maximum cycle length The LCG has full length if (and only if)

- $M$ and $c$ are relative prime.
- For each prime factor $p$ of $M$, $mod(a, p) = 1$.
- If $4$ is a factor of $M$, then $mod(a, 4) = 1$. Notice, if $M$ is a prime, full period is attained only if $a = 1$. 
Shuffling

eg. XOR between several generators.

- To enlarge period
- Improve randomness
- But not well understood
- LCGs widespread use, generally to be recommended
Mersenne Twister

Matsumoto and Nishimura, 1998

- A large structured linear feedback shift register
- Uses 19,937 bits of memory
- Has maximum period, i.e. $2^{19937} - 1$
- Has right distribution
- ... also joint distribution of 623 subsequent numbers
- Probably the best PRNG so far for stochastic simulation (not for cryptography).
RNGs in common environments

**R**: The Mersenne Twister is the default, many others can be chosen.

**Python**: Mersenne Twister chosen.

**S-plus**: XOR-shuffling between a congruential generator and a (Tausworthe) feedback shift register generator. The period is about $2^{62} \approx 4 \cdot 10^{18}$, but seed dependent (!).

**Matlab 7.4 and higher**: By default, the Mersenne Twister. Also one other available.
Characteristics

**Definition:** A sequence of *pseudo-random* numbers $U_i$ is a deterministic sequence of numbers in $]0, 1[$ having the same relevant statistical properties as a sequence of random numbers.

The question is what are relevant statistical properties.

- Distribution type
- Randomness (independence, whiteness)
Theoretical tests/properties

- Test of global behaviour (entire cycles)
- Mathematical theorems
- Typically investigates multidimensional uniformity
Testing random number generators

- Test for distribution type
  - Visual tests/plots
  - $\chi^2$ test
  - Kolmogorov Smirnov test
- Test for independence
  - Visual tests/plots
  - Run test up/down
  - Run test length of runs
  - Test of correlation coefficients
Significance test

- We assume (known) model - *The hypothesis*
- We identify a certain characterising random variable - *The test statistic*
- We reject the hypothesis if the test statistic is an abnormal observation under the hypothesis
Key terms

- Hypothesis/Alternative
- Test statistic
- Significance level
- Accept/Critical area
- Power
- *p*-value
**Multinomial distribution**

- $n$ items
- $k$ classes
- Each item falls in class $j$ with probability $p_j$
- $X_j$ is the (random) number of items in class $j$
- We write $\mathbf{X} = (X_1, \ldots, X_k) \sim \text{Mul}(n, p_1, \ldots, p_k)$

Thus $X_j \sim \text{Bin}(n, p_j)$ $E(X_j) = np_j$, $\text{Var}(X_j) = np_j(1 - p_j)$

And $E \left( \frac{X_j - np_j}{\sqrt{np_j(1-p_j)}} \right) = 0$ $\text{Var} \left( \frac{X_j - np_j}{\sqrt{np_j(1-p_j)}} \right) = 1$

Thus $\frac{X_j - np_j}{\sqrt{np_j(1-p_j)}} \xrightarrow{n \to \infty} \mathcal{N}(0, 1)$
Test statistic for \( k - 2 \)

Recall \( \frac{X_j - np_j}{\sqrt{np_j(1 - p_j)}} \xrightarrow{n \to \infty} N(0, 1) \)

thus \( \left( \frac{X_j - np_j}{np_j(1 - p_j)} \right)^2 \xrightarrow{\text{asymp}} \chi^2(1) \)

Consider now the case \( k = 2 \)

\[
\frac{(X_1 - np_1)^2}{np_1(1 - p_1)} = \frac{(X_1 - np_1)^2(p_1 + 1 - p_1)}{np_1(1 - p_1)} = \frac{(X_1 - np_1)^2}{np_1} + \frac{(X_1 - np_2)^2}{n(1 - p_1)}
\]

\[
= \frac{(X_1 - np_1)^2}{np_1} + \frac{(X_2 - np_2)^2}{np_2}
\]

- the \( \chi^2 \) statistic
- the proof can be completed by induction
Test for distribution type $\chi^2$ test

The general form of the test statistic is

$$T = \sum_{i=1}^{n_{\text{classes}}} \frac{(n_{\text{observed},i} - n_{\text{expected},i})^2}{n_{\text{expected},i}}$$

- The test statistic is to be evaluated with a $\chi^2$ distribution with $df$ degrees of freedom. $df$ is generally $n_{\text{classes}} - 1 - m$ where $m$ is the number of estimated parameters.
- It is recommend to choose all groups such that $n_{\text{expected},i} \geq 5$
Test for distribution type Kolmogorov Smirnov test

- Compare empirical distribution function $F_n(x)$ with hypothesized distribution $F(x)$.
- For known parameters the test statistic does not depend on $F(x)$.
- Better power than the $\chi^2$ test.
- No grouping considerations needed.
- Works only for completely specified distributions in the original version.
**Empirical distribution**

20 \( N(0, 1) \) variates (sorted):
-2.20, -1.68, -1.43, -0.77, -0.76, -0.12, 0.30, 0.39, 0.41, 0.44, 0.44,
0.71, 0.85, 0.87, 1.15, 1.37, 1.41, 1.81, 2.65, 3.69

\( X_i \) iid random variables with \( F(x) = P(X \leq x) \)

Each leads to a (simple) random function \( F_{e,i}(x) = 1_{\{X_i \leq x\}} \)

leading to \( F_e(x) = \frac{1}{n} \sum_{i=1}^{n} F_{e,i}(x) = \frac{1}{n} \sum_{i=1}^{n} 1_{\{X_i \leq x\}} \)

\( E(F_e(x)) = E\left(\frac{1}{n} \sum_{i=1}^{n} 1_{\{X_i \leq x\}}\right) = \frac{1}{n} \sum_{i=1}^{n} E\left(1_{\{X_i \leq x\}}\right) = F(x) \)

\( \text{Var}(F_e(x)) = \frac{1}{n^2} n F(x)(1 - F(x)) = \frac{F(x)G(x)}{n} \)

\( F_e(x) \xrightarrow{n \to \infty} N\left(F(x), \frac{F(x)G(x)}{n}\right) \)

In the limit \( (n \to \infty) \) we have a random continuous function of \( x \) -
a stochastic process, more particularly a Brownian bridge
Empirical distribution

20 \(N(0, 1)\) variates (sorted):
-2.20, -1.68, -1.43, -0.77, -0.76, -0.12, 0.30, 0.39, 0.41, 0.44, 0.44,
0.71, 0.85, 0.87, 1.15, 1.37, 1.41, 1.81, 2.65, 3.69

\[ D_n = \sup_x \{|F_n(x) - F(x)|\} \]

the test statistic follows Kolmogorovs distribution
# Test statistic and significance levels

<table>
<thead>
<tr>
<th>Case</th>
<th>Adjusted test statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>All parameters known</td>
<td>$(\sqrt{n} + 0.12 + \frac{0.5}{\sqrt{n}}) D_n$</td>
</tr>
<tr>
<td>$N(\bar{X}(n), S^2(n))$</td>
<td>$(\sqrt{n} - 0.01 + \frac{0.85}{\sqrt{n}}) D_n$</td>
</tr>
<tr>
<td>exp($\bar{X}(n)$)</td>
<td>$(\sqrt{n} + 0.26 + \frac{0.5}{\sqrt{n}}) (D_n - \frac{0.2}{n})$</td>
</tr>
</tbody>
</table>

Level of significance $(1 - \alpha)$

<table>
<thead>
<tr>
<th></th>
<th>0.850</th>
<th>0.900</th>
<th>0.950</th>
<th>0.975</th>
<th>0.990</th>
</tr>
</thead>
<tbody>
<tr>
<td>All parameters known</td>
<td>1.138</td>
<td>1.224</td>
<td>1.358</td>
<td>1.480</td>
<td>1.628</td>
</tr>
<tr>
<td>$N(\bar{X}(n), S^2(n))$</td>
<td>0.775</td>
<td>0.819</td>
<td>0.895</td>
<td>0.955</td>
<td>1.035</td>
</tr>
<tr>
<td>exp($\bar{X}(n)$)</td>
<td>0.926</td>
<td>0.990</td>
<td>1.094</td>
<td>1.190</td>
<td>1.308</td>
</tr>
</tbody>
</table>
Test for correlation - Visual tests

- Plot of $U_{i+1}$ versus $U_i$

Random numbers $U_i$ against $U_{i+1}$, $X_{i+1} = (5 \times X_i + 1) \mod 16$

Random numbers $U_i$ against $U_{i+1}$, $X_{i+1} = (129 \times X_i + 26461) \mod 65536$
Independence test: Test for multidimensional uniformity

- In the two dimensional version test for uniformity of \((U_{2i-1}, U_{2i})\)
- Typically \(\chi^2\) test
- The number of groups increases drastically with dimension
Run test I

Above/below

- The run test given in Conradsen, can be used by e.g. comparing with the median.
- The number of runs (above/below the median) is (asymptotically) distributed as

\[ \mathbb{N} \left( 2 \frac{n_1 n_2}{n_1 + n_2} + 1, 2 \frac{n_1 n_2 (2n_1 n_2 - n_1 - n_2)}{(n_1 + n_2)^2 (n_1 + n_2 - 1)} \right) \]

where \( n_1 \) is the number of samples above and \( n_2 \) is the number below.
- The test statistic is the total number of runs \( T = R_a + R_b \) with \( R_a \) (runs above) and \( R_b \) (runs below)
Run tests II

Up/Down from Knuth

A test specifically designed for testing random number generators is the following UP/DOWN run test, see e.g. Donald E. Knuth, The Art of Computer Programming Volume 2, 1998, pp. 66-.

The sequence:

0.54, 0.67, 0.13, 0.89, 0.33, 0.45, 0.90, 0.01, 0.45, 0.76, 0.82, 0.24, 0.17

has runs of length 2, 2, 3, 4, 1, ... i.e. runs of consecutively increasing numbers.
Run test II

Generate $n$ random numbers. The observed number of runs of length 1, ..., 5 and $\geq 6$ are recorded in the vector $R$. The test statistic is calculated by:

$$Z = \frac{1}{n-6}(R - nB)^T A(R - nB)$$

where

$$A = \begin{bmatrix}
4529.4 & 9044.9 & 13568 & 18091 & 22615 & 27892 \\
9044.9 & 18097 & 27139 & 36187 & 45234 & 55789 \\
13568 & 27139 & 40721 & 54281 & 67852 & 83685 \\
18091 & 36187 & 54281 & 72414 & 90470 & 111580 \\
22615 & 45234 & 67852 & 90470 & 113262 & 139476 \\
27892 & 55789 & 83685 & 111580 & 139476 & 172860
\end{bmatrix}$$

and

$$B = \begin{bmatrix}
\frac{1}{6} \\
\frac{5}{120} \\
\frac{19}{720} \\
\frac{11}{3600} \\
\frac{1}{830}
\end{bmatrix}$$

The test statistic is compared with a $\chi^2(6)$ distribution. One should have $n > 4000$.
Run test III

The-Up-and-Down Test This test is described in Rubinstein 81 "Simulation and the Monte Carlo Method" and Iversen 07 (in Danish).

The sequence:
0.54, 0.67, 0.13, 0.89, 0.33, 0.45, 0.90, 0.01, 0.45, 0.76, 0.82, 0.24, 0.17
is converted to
<, >, <, >, <, <, >, <, <, <, >, >
giving in total 8 runs of length 1, 1, 1, 2, 1, 3, 2
Run test III

The expected number of runs of length $k$ is $\frac{n+1}{12}$, $\frac{11n-4}{12}$ for runs of length 1 and 2 respectively, and

$$2\left[\frac{(k^2+3k+1)n-(k^3+3k^2-k-4)}{(k+3)!}\right]$$

for runs of length $k < N - 1$.

Define $X$ to be the total number of runs, then

$$Z = \frac{X - \frac{2n-1}{3}}{\sqrt{\frac{16n-20}{90}}}$$

is asymptotically $N(0,1)$. 
Correlation coefficients

- the estimated correlation

\[ c_h = \frac{1}{n - h} \sum_{i=1}^{n-h} U_i U_{i+h} \sim N \left( 0.25, \frac{7}{144n} \right) \]
Exercise 1

In today’s exercise you should implement everything including the tests yourself, including the chi-square and KS tests. Later, when your code is working you are free to use builtin functions.

- Write a program generating 10,000 (pseudo-) random numbers and present these numbers in a histogramme (e.g. 10 classes).
- First implement the LCG yourself by experimenting with different values of “a”, “b” and “c”.
- Evaluate the quality of the generators by graphical descriptive statistics (histogrammes, scatter plots) and statistical tests ($\chi^2$, Kolmogorov-Smirnov, run-tests, and correlation test).
- Then apply a system available generator and perform the various statistical tests for this generator too.