Random Numbers Generation

Prof. Americo Cunha Jr

Rio de Janeiro State University – UERJ

americo.cunha@uerj.br

www.americocunha.org

@AmericoCunhaJr @AmericoCunhaJr

@AmericoCunhaJr @AmericoCunhaJr

Random Numbers in a Computer

Random Number Generator (RNG)

Computer is a deterministic device,
it can not generate random numbers!

The best that one can do is to use a deterministic algorithm to generate numerical values which appear to be random, i.e., generate **pseudorandom numbers**.

A sequence of such numbers must pass stringent tests designed to ensure that they can provide the same results that truly random numbers would produce for the given problem.

R. W. Shonkwiler, and F. Mendivil, *Explorations in Monte Carlo Methods*, Springer, 2009.

Random Number Generator (RNG)

A generic RNG then has the following structure:

1. **Initialize:** Draw the seed $\mathcal{S}_0$ from the distribution μ on $\mathcal{S}$. Set $n = 0$.
2. **Transition:** Set $\mathcal{S}_n = f(\mathcal{S}_{n-1})$.
3. **Output:** Set $\mathcal{U}_n = g(\mathcal{S}_n)$.
4. **Repeat:** Set $n = n + 1$ and return to Step 2.

With respect to this generic algorithm:

- $\mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3, \cdots$ is a sequence of **pseudorandom numbers**
- the starting value $\mathcal{S}_0$ is called seed
- the sequence repeat after a certain index (periodicity)

D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo Methods*, Wiley, 2011.

Properties of a Good RNG

- pass statistical tests
- theoretical support
- reproducible
- fast and efficient
- large period
- multiple streams
- cheap and easy
- not produce 0 or 1

D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo Methods*, Wiley, 2011.

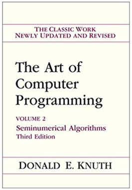
An early attempt to contruct a RNG

Knuth's "Super-random" algorithm

1. Take N to be the most significant digit of X, a 10-digit decimal number. Steps 2-13 are repeated exactly $N+1$ times.
2. Let M be the second most significant digit of X. Jump to step $3+M$.
3. If $X < 5 \times 10^9$, set $X = X + 5 \times 10^9$.
4. Replace X by $\lfloor X^2/10^5 \rfloor \mod 10^{10}$.
5. Replace X by $(1001001001 \times X) \mod 10^{10}$.
6. If $X < 10^8$ then $X = X + 9814055677$; otherwise $X = 10^{10} - X$.
7. Interchange the lower-order five digits of X with the higher-order five digits of X.
8. Replace X by $(1001001001 \times X) \mod 10^{10}$.
9. For each digit d of X, decrease d by 1 if $d > 0$.
10. If $X < 10^5$, set $X = X^2 + 99999$; otherwise $X = X - 99999$.
11. If $X < 10^9$, set $X = 10 \times X$ and repeat this step.
12. Replace X by the middle 10 digits of $X(X-1)$.
13. If $N > 0$, decrease N by one and return to step 2. If $N = 0$, the algorithm terminates with the current value of X as the next value in the sequence.

©

Donald E. Knuth



R. W. Shonkwiler, and F. Mendivil, *Explorations in Monte Carlo Methods*, Springer, 2009.

Sequence generated by Knuth's algorithm

Sometimes complexity hides simple behaviour:

- The first time Knuth ran this algorithm it almost immediately converged to 6065038420 (a fixed point for the algorithm).
- After this, when he ran it with a different starting value, it converged to a cycle having length 3178!

Lessons from this example:

- Complexity is not a substitute for randomness.
- Random numbers should not be generated with a method chosen at random!

R. W. Shonkwiler, and F. Mendivil, *Explorations in Monte Carlo Methods*, Springer, 2009.

Linear Congruential Generators (LCG)

Fix the positive integer parameters m, a, c, and X_0

- m – modulus, $0 < m$
- a – multiplier, $0 \leq a < m$
- c – increment, $0 \leq c < m$
- X_0 – seed, $0 \leq x_0 < m$

Given an integer X_n , the following iteration is computed

$$X_{n+1} = (a\, X_n + c) \mod m, \qquad n \in \mathbb{N}.$$

R. W. Shonkwiler, and F. Mendivil, *Explorations in Monte Carlo Methods*, Springer, 2009.

Sequence generated by LCG

Example: $(m = 8,\ a = 5,\ c = 1,\ x_0 = 0)$

$$
\begin{array}{lll}
X_0 = 0 & X_8 = 0 & X_{16} = 0 \\
X_1 = 1 & X_9 = 1 & X_{17} = 1 \\
X_2 = 6 & X_{10} = 6 & X_{18} = 6 \\
X_3 = 7 & X_{11} = 7 & X_{19} = 7 \\
X_4 = 4 & X_{12} = 4 & X_{20} = 4 \\
X_5 = 5 & X_{13} = 5 & X_{21} = 5 \\
X_6 = 2 & X_{14} = 2 & X_{22} = 2 \\
X_7 = 3 & X_{15} = 3 & X_{23} = 3
\end{array}
\quad \cdots
$$

- maximum number of possible different outcome is equal to m
- in real applications, a very large m is taken (e.g. $m = 2^{32}$)

R. W. Shonkwiler, and F. Mendivil, *Explorations in Monte Carlo Methods*, Springer, 2009.

Choosing a Good Random Number Generator

- Like choosing a new car: for some people speed is preferred, while for others robustness and reliability are more important.
- For Monte Carlo simulation distributional properties of RNG are paramount, whereas in coding/cryptography unpredictability is crucial.
- As with cars, there are many poorly designed and outdated models available that should be avoided. Several of standard generators that come with popular programming languages and computing packages can be appallingly poor.

D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo Methods*, Wiley, 2011.

Choosing a Good Random Number Generator

- Faster generators are not necessarily better.
  (indeed, often the contrary is true)
- A small period is in general bad.
  (but a larger period is not necessarily better)
- Good equidistribution is a necessary requirement.
  (but not a sufficient requirement)

D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo Methods*, Wiley, 2011.

Choosing a Good Random Number Generator

Two classes of RNG with good performance:

- **Combined multiple recursive generators**
  - simple
  - fast
  - large period
  - excellent statistical properties
  - multiple streams
- **Twisted general feedback shift register generators**
  - fast
  - extremely long periods
  - very good equidistributional properties
  - MT19937ar – MATLAB default RNG

D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo Methods*, Wiley, 2011.

Random Variables in a Computer

Some Methods for Random Variables Generation (RVG)

- Uniform Random Number Generator
- Box–Muller transformation
- Inverse Transform Method
- Acceptance/Rejection Method
- Markov-Chain Monte Carlo
    - Metropolis-Hastings Algorithm
    - Gibbs Sampler
    - Langevin Metropolis-Hastings Algorithm
    - Soize's ISDE Algorithm
    - ...
- many others

R. W. Shonkwiler, and F. Mendivil, *Explorations in Monte Carlo Methods*, Springer, 2009.

Uniform Random Number Generator

LCG produce positive intergers number X_n between zero and m.

In this way, the fraction

$$U_n = X_n/m$$

generates real numbers between zero and one, i.e., $0 \leq U_n \leq 1$.

R. W. Shonkwiler, and F. Mendivil, *Explorations in Monte Carlo Methods*, Springer, 2009.

Box–Muller transformation

Input: a pair of independent uniform random variables

$$U_1 \sim \mathcal{U}(0,1) \qquad U_2 \sim \mathcal{U}(0,1)$$

Output: a pair of independent Gaussian random variables

$$Z_1 = R \, cos \, \Theta \sim \mathcal{N}(0,1)$$

and

$$Z_2 = R \, sin \, \Theta \sim \mathcal{N}(0,1)$$

where

$$R = \sqrt{-2 \, \ln U_1} \quad \Theta = 2 \, \pi \, U_2$$

R. W. Shonkwiler, and F. Mendivil, *Explorations in Monte Carlo Methods*, Springer, 2009.

Inverse Transform Method

Let X be a random variable with distribution F_X.

Inverse function of F_X, denoted by F_X^{-1}, is defined as

$$F_X^{-1}(u) = \inf \{x \in \mathbb{R} \mid u \leq F_X(x)\}, \qquad 0 < u < 1.$$

Let $U \sim \mathcal{U}(0,1)$.

The distribution of random variable $Y = F_X^{-1}(U)$ is given by

$$F_Y(x) = \mathcal{P}(Y \leq x) = \mathcal{P}(F_X^{-1}(U) \leq x) = \mathcal{P}(U \leq F_X(x)) = F_X(x).$$

Algorithm to generate $X \sim F_X$:

1. draw $U \sim \mathcal{U}(0,1)$
2. set $X = F_X^{-1}(U)$

D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo Methods*, Wiley, 2011.

An example with the inverse transform method

Generate X from

$$p_X(x) = \begin{cases} 2\,x, & 0 \leq x \leq 1 \\ 0, & \text{otherwise.} \end{cases}$$

The CDF of X is given by

$$F_X(x) = x^2, \quad 0 \leq x \leq 1$$

and its inverse by

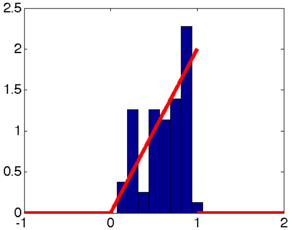
$$F_X^{-1}(u) = \sqrt{u}, \quad 0 \leq u \leq 1.$$

Algorithm :

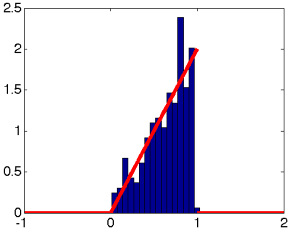
1. draw $U \sim \mathcal{U}(0,1)$
2. set $X = \sqrt{U}$

D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo Methods*, Wiley, 2011.

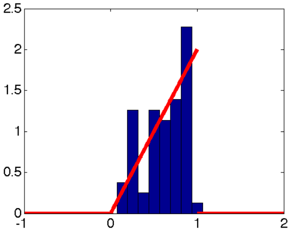
An example with the inverse transform method



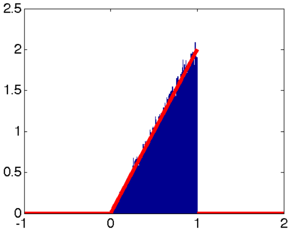
64 samples



256 samples



1024 samples



65536 samples

D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo Methods*, Wiley, 2011.

References

- S. B. Volchan, *What Is a Random Sequence?* **The American Mathematical Monthly**, 109: 46–63, 2002.
- T. Homem-de-Mello and G. Bayraksan, *Monte Carlo sampling-based methods for stochastic optimization*, **Surveys in Operations Research and Management Science**, 19: 56–85, 2014.
- A. Cunha Jr, R. Sampaio, R. Nasser, H. Lopes, and K. Breitman, *Uncertainty quantification through Monte Carlo method in a cloud computing setting*, **Computer Physics Communications**, 185: 1355-1363, 2014.
- R. W. Shonkwiler, and F. Mendivil, **Explorations in Monte Carlo Methods**, Springer, 2009.
- C. Robert, and G. Casella, **Monte Carlo Statistical Methods**, Springer, 2nd edition, 2005.
- S. Asmussen, and P. W. Glynn, **Stochastic Simulation: Algorithms and Analysis**, Springer, 2007.
- J. E. Gentle, **Random Number Generation and Monte Carlo Methods**, Springer, 2nd Edition, 2004.
- J. S. Liu, **Monte Carlo Strategies in Scientific Computing**, Springer, 2009.
- R. Y. Rubinstein, and Dirk P. Kroese, **Simulation and the Monte Carlo Method**, Wiley, 3rd Edition, 2016.
- D. P. Kroese, T. Taimre, and Z. I. Botev, **Handbook of Monte Carlo Methods**, Wiley, 2011.

How to cite this material?

A. Cunha Jr, *Random Numbers Generation*, Rio de Janeiro State University – UERJ, 2021.

@AmericoCunhaJr @AmericoCunhaJr

@AmericoCunhaJr @AmericoCunhaJr

Ⓒ copyrighted material

Content excluded from our Creative Commons license

- Donald Knuth photo:
  Wikimedia Commons, File:KnuthAtOpenContentAlliance.jpg — Wikimedia Commons, the free media repository
  https://commons.wikimedia.org/w/index.php?title=File:KnuthAtOpenContentAlliance.jpg