# Random Numbers

Based on D. E. Knuth, *The Art of Computer Programming*, Vol. 2: *Seminumerical Algorithms* (3rd ed.), Addison-Wesley

Donald E. Knuth

Stanford University

Random Number Generation – Graduate Course

# Chapter 3 Overview

Introduction (§3.1)

The Linear Congruential Method (§3.2.1)

Other Generation Methods (§3.2.2)

Statistical Tests (§3.3)

Generating Other Distributions (§3.4)

What Is a Random Sequence? (§3.5)

Summary & Key Takeaways

# What Are Random Numbers? (§3.1)

## Knuth's opening observation

*"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."* — John von Neumann (1951)

**The fundamental tension:**

- Computers are **deterministic** — they cannot produce truly random output
- Yet simulation, cryptography, numerical methods, and games all **need** random numbers
- Solution: **pseudorandom sequences** — deterministic sequences that *behave* randomly

**A cautionary tale (§3.1):**

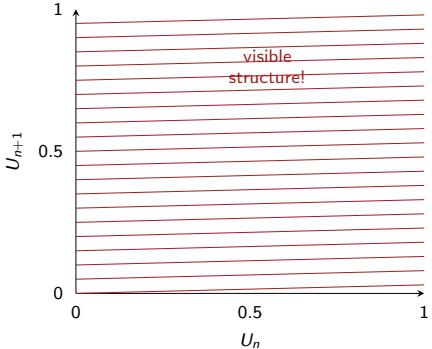
Consider the sequence produced by:

$$X_{n+1} = (aX_n + c) \bmod m$$

with poorly chosen $a$, $c$, $m$. Knuth famously showed that many early generators were *catastrophically* non-random, e.g. IBM's RANDU.

# Bad LCG: Visible Lattice Structure

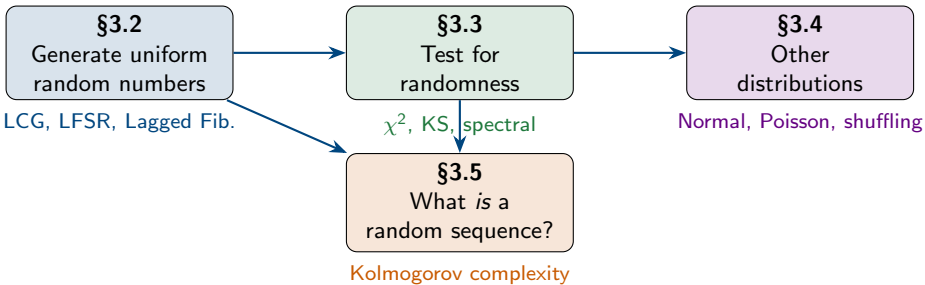
Bad LCG ($a = 65$, $m = 2048$): visible lattice structure




## Marsaglia's Theorem (1968)

All $s$-tuples $(U_n, U_{n+1}, \ldots, U_{n+s-1})$ from an LCG lie on at most

# The Goals of Chapter 3




## Knuth's Philosophy

Do not use a generator you do not understand. **Test** your generator before trusting it, choose parameters **theoretically**, and understand the **limitations** of every method.

# The Linear Congruential Generator (LCG)

**The recurrence (Lehmer, 1949):**

$$X_{n+1} = (aX_n + c) \bmod m$$

| Parameter | Role |
|---|---|
| $m > 0$ | **modulus** |
| $0 < a < m$ | **multiplier** |
| $0 \le c < m$ | **increment** |
| $0 \le X_0 < m$ | **seed** |

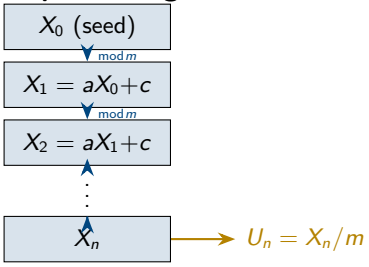
Output: $U_n = X_n/m \in [0, 1)$

## Hull-Dobell Theorem (full period)

An LCG has full period $m$ if and only if:

1. $\gcd(c, m) = 1$
2. $a \equiv 1 \pmod{p}$ for every prime $p \mid m$
3. If $4 \mid m$ then $a \equiv 1 \pmod 4$

**Sequence diagram:**




## Key property

Maximum period $= m$.
For $m = 2^{32}$: period $\approx 4 \times 10^9$.

# Choosing Good LCG Parameters

**Choice of modulus $m$:**

- $m = 2^e$ (word size): fast mod via truncation
- $m$ = large Mersenne prime: better theoretical properties
- $m = 2^{31} - 1 = 2{,}147{,}483{,}647$ is popular

**Choice of multiplier $a$:**

- Must pass the **spectral test** (§3.3.4)
- Knuth recommends $a \approx 0.01m$ with good spectral values
- Avoid obvious patterns in binary representation

## Potency

The **potency** $s$ of an LCG is the smallest integer such that $(a-1)^s \equiv 0 \pmod{m}$. Higher potency $\Rightarrow$ better statistical mixing.

**Knuth's recommended parameters** (§3.2.1):

| $m$ | $a$ | Source |
|---|---|---|
| $2^{32}$ | 1,664,525 | Knuth |
| $2^{32}$ | 69,069 | Marsaglia |
| $2^{31}-1$ | $7^5 = 16{,}807$ | Lewis et al. |
| $2^{31}-1$ | 630,360,016 | Fishman & Moore |
| $2^{48}$ | 25,214,903,917 | POSIX `drand48` |

## Multiplicative LCG ($c = 0$)

Simpler, but $X_0 \neq 0$ required; full period is impossible. Maximum period $= m/4$ for $m = 2^e$. For prime $m$: period up to $m - 1$.

# Other Uniform Generation Methods (§3.2.2)

**1. Lagged Fibonacci Generators:**

$$X_n = X_{n-j} \star X_{n-k}, \quad j < k$$

where $\star \in \{+, -, \times, \oplus\}$.

- Very long periods (up to $\sim 2^k$)
- Excellent equidistribution in high dimensions
- Require $k$ words of state

**2. Linear Feedback Shift Registers:**

$$X_n = \bigoplus_{i=1}^{k} c_i X_{n-i}, \quad c_i \in \{0, 1\}$$

- Period up to $2^k - 1$ (primitive polynomial)
- Extremely fast in hardware
- Low linear complexity — cryptographically weak

# Other Uniform Generation Methods (§3.2.2)

**3. Combined Generators:**

Combine two or more generators for longer periods and better statistical properties.

**L'Ecuyer (1988):**

$$Z_n = (X_n - Y_n) \bmod (m_1 - 1)$$

where $X_n$, $Y_n$ are independent LCGs with different moduli.

**4. Mersenne Twister (MT19937):**

- Period: $2^{19937} - 1$
- 623-dimensionally equidistributed
- Default in Python, R, MATLAB
- State: 624 words ($\approx 2.5$ KB)

## Knuth's warning

*"Never use a random number generator that you have not subjected to extensive empirical tests."* (§3.2.2)

# Generator Comparison

| Generator | Recurrence | Period | State | Speed | Notes |
|---|---|---|---|---|---|
| LCG | $aX + c \bmod m$ | $m$ | 1 word | Fast | Simple; weak in high dims |
| Mult. LCG | $aX \bmod m$ | $m/4$ | 1 word | Fast | No even seeds |
| Lagged Fib. | $X_{n-j} \star X_{n-k}$ | $\sim 2^k$ | $k$ words | Fast | Excellent equidist. |
| LFSR | XOR feedback | $2^k - 1$ | $k$ bits | Very fast | Cryptographically weak |
| MT19937 | Twist + tempering | $2^{19937} - 1$ | 624 w. | Med. | Industry standard |
| PCG | LCG + permutation | $2^{128}$ | 2 words | Fast | Modern; excellent quality |
| xoshiro256 | Shift + rotate | $2^{256} - 1$ | 4 words | Fast | Current best practice |

## Choosing a generator

For most simulation work: **MT19937** or **PCG**.
For cryptography: use a **CSPRNG** (e.g., AES-CTR, ChaCha20).
For parallel simulation: choose generators with **multiple streams**.

## Period is necessary but not sufficient

A long period does *not* guarantee good statistical quality. Always verify with the spectral test and empirical test batteries (TestU01, PractRand).

# Statistical Testing: The Framework (§3.3.1)

**The testing problem:**

Given a sequence $U_1, U_2, \ldots, U_n$ supposedly $\mathrm{Uniform}(0,1)$, does it *behave* randomly?

**Chi-Square Test:**

$$V = \sum_{s=1}^{k} \frac{(Y_s - n/k)^2}{n/k} \sim \chi^2_{k-1}$$

$Y_s$ = observed count in cell $s$. Tests frequency uniformity.

**Kolmogorov-Smirnov (KS) Test:**

$$K_n^+ = \sqrt{n} \sup_x \big(F_n(x) - F(x)\big)$$

$F_n$ = empirical CDF. Sensitive to deviations anywhere in $[0,1]$.

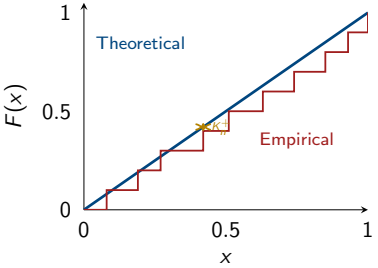
## p-values

Report the p-value under $H_0$.
**Suspicious:** $p < 0.01$ *or* $p > 0.99$.
Both tails matter for PRNGs!

# Statistical Testing: The Framework (§3.3.1)

**KS Test illustration:**

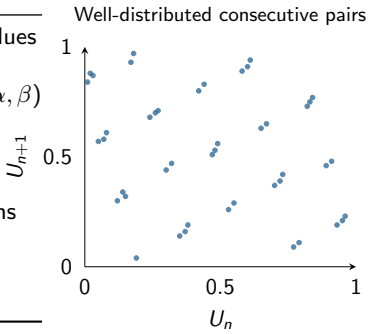
KS Test: Empirical vs. Theoretical CDF



# Knuth's Battery of Empirical Tests (§3.3.2)

| Test | What it checks |
|---|---|
| **Frequency** | Uniform distribution of individual values |
| **Serial** | Joint distribution of pairs $(U_n, U_{n+1})$ |
| **Gap** | Lengths of gaps between values in $[\alpha, \beta)$ |
| **Poker** | Pattern frequencies in groups of 5 |
| **Coupon collector** | Waiting time to collect all $d$ values |
| **Permutation** | Ordering patterns in groups of $t$ |
| **Run** | Lengths of ascending/descending runs |
| **Maximum-of-$t$** | Distribution of $\max(U_1, \ldots, U_t)$ |
| **Collision** | Birthday paradox collision counts |
| **Serial corr.** | Correlation $\mathrm{corr}(U_n, U_{n+k})$ |

**Serial test in 2D (good generator):**



# The Spectral Test (§3.3.4)

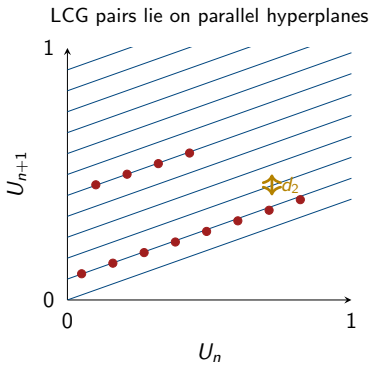
**The most powerful theoretical test for LCGs.**
**Marsaglia's theorem (1968):**
The $t$-tuples $(X_n, X_{n+1}, \ldots, X_{n+t-1})/m$ from an LCG fall on at most $(t!\, m)^{1/t}$ parallel hyperplanes in $[0,1)^t$.
**Figure of merit:** max inter-hyperplane gap $d_t$.

$$\nu_t = \frac{1}{d_t\sqrt{t}} \geq 0.5 \text{ required}$$

## Why this matters in practice

For $t = 3$ and a bad LCG with $m = 2^{32}$, all 3-tuples may lie on as few as $\sim$1600 planes. Any simulation using 3-tuples will have severe structural bias.

**Hyperplane structure in 2D:**




*The gap $d_2$ between adjacent lines measures how "flat" the 2D distribution is.*

# Theoretical vs. Empirical Tests

**Theoretical tests:**

- Analyze the *mathematical structure* of the generator
- Give **guarantees** for the full period
- Examples: Hull-Dobell, spectral test, serial correlation formula

**Empirical tests:**

- Test *actual output* on a finite sample
- Can detect failures that theory misses
- Examples: $\chi^2$, KS, gap, run, serial

## Knuth's verdict

Pass all tests $\not\Rightarrow$ truly random.
Fail any test $\Rightarrow$ suspect the generator.
**Use both** theoretical and empirical tests.

**Recommended test priority:**

| Test | Type | Priority |
|---|---|---|
| Spectral test | Theoretical | **Must** |
| $\chi^2$ frequency | Empirical | **Must** |
| Serial test | Empirical | High |
| Gap test | Empirical | High |
| Serial correlation | Both | High |
| Run test | Empirical | Medium |
| Poker test | Empirical | Medium |
| Coupon collector | Empirical | Medium |
| Permutation test | Empirical | Lower |

## Modern suites

**TestU01** (L'Ecuyer & Simard, 2007)
**PractRand** (Doty-Humphrey, 2011)
**NIST STS** (cryptographic focus)

# Generating Non-Uniform Distributions (§3.4.1)

**Method 1 — Inversion (CDF method):**

If $F$ is invertible, generate $U \sim \mathrm{Uniform}(0,1)$ and set

$$X = F^{-1}(U).$$

- Exponential: $X = -\ln(U)/\lambda$
- Geometric: $X = \lfloor \ln U / \ln(1-p) \rfloor + 1$
- Exact for any distribution with tractable $F^{-1}$

**Method 2 — Rejection (von Neumann):**

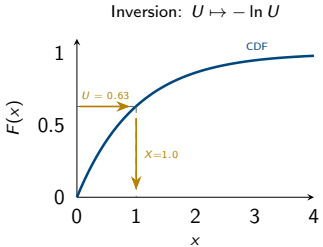
To sample from $f(x) \le c\,g(x)$:

1. Generate $X \sim g$, $U \sim \mathrm{Uniform}(0,1)$
2. **Accept** if $U \le f(X)/(c\,g(X))$
3. Else reject and repeat

Expected acceptance rate: $1/c$.

# Generating Non-Uniform Distributions (§3.4.1)

**Inversion for the Exponential:**




**Method 3 — Box-Muller (Normal):**

$$Z_1 = \sqrt{-2 \ln U_1} \cos(2\pi U_2) \sim N(0,1)$$

$$Z_2 = \sqrt{-2 \ln U_1} \sin(2\pi U_2) \sim N(0,1)$$

Both $Z_1, Z_2$ independent.

# Random Sampling and Shuffling (§3.4.2)

**Problem 1 — Random Sampling without Replacement:**

Select $n$ items from $N$ (Algorithm S, Knuth §3.4.2):

1. $t \leftarrow 0$, $m \leftarrow 0$
2. Generate $U \sim \mathrm{Uniform}(0, 1)$
3. If $(N - t)U \geq n - m$: skip item $t + 1$
4. Else: select item $t + 1$, $m \leftarrow m + 1$
5. $t \leftarrow t + 1$; stop if $m = n$, else go to 2

Exactly uniform over all $\binom{N}{n}$ subsets.

**Problem 2 — Random Permutation (Shuffle):**

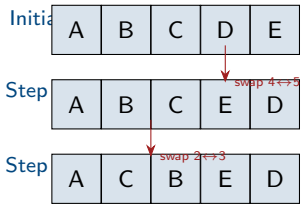
## Fisher-Yates / Knuth Shuffle

For $j = n$ down to 2:
$k \leftarrow \lfloor U \cdot j \rfloor + 1$
swap $A[k] \leftrightarrow A[j]$

Each of the $n!$ permutations is equally likely.

# Random Sampling and Shuffling (§3.4.2) (Cont.)

**Shuffle example ($n = 5$):**




Continue until $j = 2\ldots$

## Common mistake

Using $k \leftarrow \lfloor U \cdot n \rfloor + 1$ (range $1..n$) instead of $1..j$ gives a **biased** shuffle!

# Generating Common Distributions — Summary Table

| Distribution | Method | Formula / Key Step | Notes |
|---|---|---|---|
| Uniform$(a, b)$ | Scale | $a + (b - a)U$ | Exact |
| Exponential$(\lambda)$ | Inversion | $-\ln(U)/\lambda$ | Exact |
| Normal$(0, 1)$ | Box-Muller | $\sqrt{-2\ln U_1}\cos(2\pi U_2)$ | Exact, needs trig |
| Normal$(0, 1)$ | Polar | $\sqrt{-2\ln S/S}\,(V_1, V_2)$ | No trig; faster |
| Gamma$(\alpha, 1)$ | Marsaglia-Tsang | $d(1 + cZ)^3$, $Z \sim N(0, 1)$ | $\alpha \geq 1$ |
| Poisson$(\mu)$ | Waiting time | Count until $\prod U_i < e^{-\mu}$ | Small $\mu$ |
| Poisson$(\mu)$ | Normal approx. | $\lfloor \mu + \sqrt{\mu}Z + 0.5 \rfloor$ | Large $\mu$ |
| Binomial$(n, p)$ | Sum | $\sum_{i=1}^{n} \mathbf{1}[U_i < p]$ | Small $n$ |
| Geometric$(p)$ | Inversion | $\lfloor \ln U / \ln(1 - p) \rfloor + 1$ | Exact |
| Discrete (general) | Alias method | Precompute table, $O(1)$ sample | Walker (1977) |

## The alias method

Generates any discrete distribution over $k$ outcomes in $O(1)$ per sample after $O(k)$ preprocessing. Uses a **probability table** and an **alias table**.

## Knuth's rule

Always generate from the **exact** distribution when possible. Approximations save time but may invalidate simulation results.

# The Philosophy of Randomness (§3.5)

**Three conceptions of randomness:**

**1. Frequency / Statistical (von Mises, 1919):**

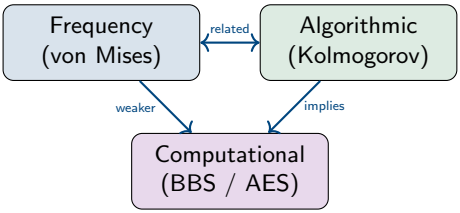
- A sequence is random if it passes all statistical tests restricted to *admissible* selection rules (Church, 1940)
- Problem: infinitely many possible tests

**2. Algorithmic / Kolmogorov (1965):**
A string $s$ of length $n$ is random if

$$K(s) \approx n \text{ bits}$$

where $K(s)$ = length of its shortest description.
**Problem:** $K$ is uncomputable (Turing, 1936).

**3. Computational / Cryptographic:**
A sequence is pseudorandom if no **polynomial-time** algorithm can distinguish it from truly random output.



# The Philosophy of Randomness (§3.5) (Cont.)

## Knuth's conclusion (§3.5)

*"A random sequence is a vague notion embodying the idea of a sequence in which each term is unpredictable to the observer."*

## Practical implication

No finite sequence can be *proven* random. We can only accumulate **evidence** of randomness through testing.

# Kolmogorov Complexity and Linear Complexity

**Kolmogorov complexity examples:**

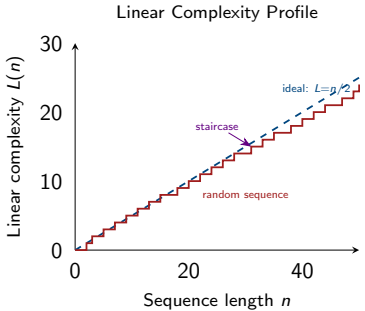
- $s = \underbrace{00\ldots0}_{n}$: $K(s) \approx \log_2 n$ (very compressible)
- $s =$ digits of $\pi$: $K(s) = O(1)$ (short program)
- $s =$ truly random bits: $K(s) \approx n$ (incompressible)

## Martin-Löf's theorem (1966)

A sequence is random (in the strongest sense) iff it passes *all* computable statistical tests, which holds iff $K(s_1 \cdots s_n) \geq n - c$ for all $n$ (some constant $c$).

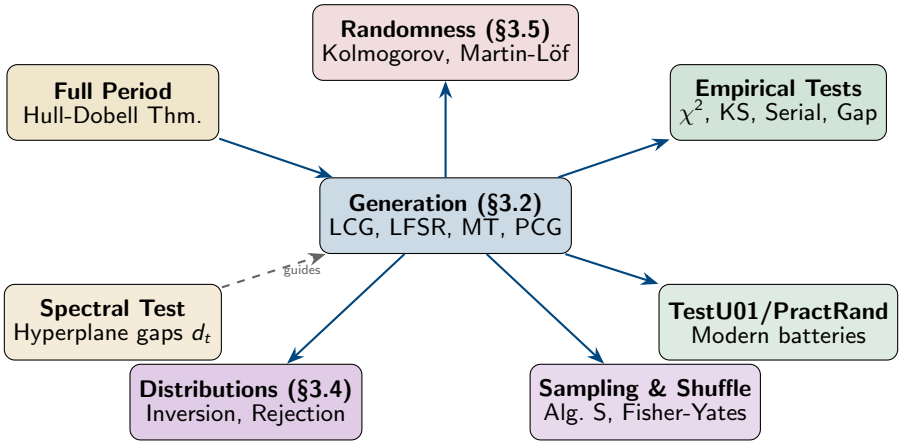
# Kolmogorov Complexity and Linear Complexity (Cont.)

**Linear complexity:**
$L(n)$ = length of shortest LFSR producing $s_1 \cdots s_n$.
A good PRNG should have $L(n) \approx n/2$ (Berlekamp-Massey).
**Linear complexity profile:**




*Steps of size $\approx 1$ indicate good randomness. Large jumps indicate exploitable structure.*

# Chapter 3 Concept Map



# Key Results and Formulas

**LCG recurrence:**

$$X_{n+1} = (aX_n + c) \bmod m$$

Full period $\Leftrightarrow$ Hull-Dobell conditions.

**Marsaglia's theorem:**
$t$-tuples lie on $\leq (t!\, m)^{1/t}$ hyperplanes.

**Spectral figure of merit:**

$$\nu_t = \frac{1}{d_t \sqrt{t}} \geq 0.5 \text{ required}$$

**Chi-square statistic:**

$$V = \sum_{s=1}^{k} \frac{(Y_s - n/k)^2}{n/k} \sim \chi^2_{k-1}$$

**Box-Muller transform:**

$$Z_{1,2} = \sqrt{-2 \ln U_1} \begin{pmatrix} \cos(2\pi U_2) \\ \sin(2\pi U_2) \end{pmatrix} \sim N(0,1)$$

**Fisher-Yates shuffle:**
For $j = n$ down to 2:
  swap $A[j] \leftrightarrow A[\lfloor jU \rfloor + 1]$

**Kolmogorov complexity:**

$$K(s) \approx n \;\Leftrightarrow\; s \text{ is "random"}$$

## Knuth's golden rule

*"Random numbers should not be generated with a method chosen at random."*

— Donald Knuth (§3.1)

# Discussion Questions

1. The Hull-Dobell theorem guarantees full period for an LCG. Does full period imply good statistical quality? Give an example of a full-period LCG that fails the spectral test.

2. The spectral test measures the maximum gap between parallel hyperplanes containing $t$-tuples from an LCG. Why does this matter practically? What goes wrong in a simulation if $d_t$ is large?

3. Knuth recommends testing for both *suspiciously small* ($p < 0.01$) and *suspiciously large* ($p > 0.99$) p-values. Why would a p-value too close to 1 be a red flag for a PRNG?

4. Compare the Box-Muller transform and the polar method for generating Normal random variables. What are the trade-offs in cost, accuracy, and implementation complexity?

5. Kolmogorov complexity gives a clean definition of a random sequence, yet it is uncomputable. What does this tell us about the fundamental limits of statistical testing?

# References


- D. E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, 3rd ed., Addison-Wesley, 1997.
- P. L'Ecuyer, "Tables of linear congruential generators of different sizes and good lattice structure," *Math. Comp.*, 68(225):249–260, 1999.
- G. Marsaglia, "Random numbers fall mainly in the planes," *Proc. Natl. Acad. Sci.*, 61(1):25–28, 1968.
- M. Matsumoto & T. Nishimura, "Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator," *ACM TOMACS*, 8(1):3–30, 1998.
- A. J. Walker, "An efficient method for generating discrete random variables with general distributions," *ACM TOMS*, 3(3):253–256, 1977.
- G. E. P. Box & M. E. Muller, "A note on the generation of random normal deviates," *Ann. Math. Statist.*, 29(2):610–611, 1958.

# Questions?