Recap - Review of prob, Law of large numbers + CLT ⇒ Confidence Intervals

Today + Thursday - Generate random numbers - How does np.random generate U[0,1] (primitive)

- (Fundamental 'theorem' of simulation) Every random var / random process can be generated using U[0,1]

ORIE 4580/5580: Simulation Modeling and Analysis

ORIE 5581: Monte Carlo Simulation

Unit 4: Generating Random Numbers

Sid Banerjee

School of ORIE, Cornell University

Today

- Generate U[0,1] using PRNG (pseudorandom generator)
- Inversion method

generating uniformly distributed random variables

random number: a sample from $U[0,1]$

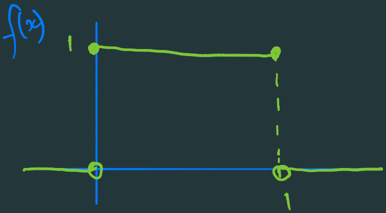
the 'fundamental theorem' of simulation

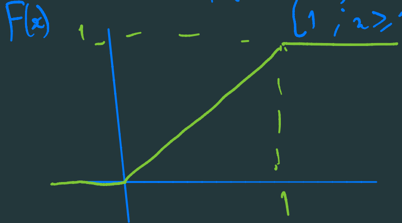
can 'transform' a stream of $U[0,1]$ to any other random variable

- arbitrary probability distribution
- arbitrary correlations
- complex processes

$U \sim U[0,1]$

$$F(x) = \begin{cases} 0 \,; & x \leq 0 \\ x \,; & x \in [0,1] \\ 1 \,; & x \geq 1 \end{cases}$$





where can we find random numbers?

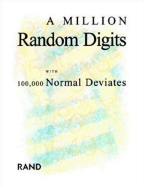
Books › Science & Math › Mathematics

A Million Random Digits with 100,000 Normal Deviates 0th Edition

by The RAND Corporation (Author)

682 customer reviews

Look inside



Best Deal

Hardcover	Paperback	Other Sellers
$174.95	$50.27 - $57.12	from $50.27

Buy used $50.27

Buy new prime **$57.12**

In Stock.

Ships from and sold by Amazon.com. Gift-wrap available.

FREE Shipping for Prime members Details

List Price: ~~$68.00~~ Save: $10.88 (16%)

8 New from $57.12

Qty: 1

Want it Wednesday, Sept. 6? Order within **21 hrs** and choose **Two-Day Shipping** at checkout. Details

Add to Cart

Turn on 1-Click ordering

Ship to:

Weirdest sudoku book ever

By John Peter O'connor on October 6, 2012

Format: Paperback

This has got to be the most useless set of sudoku puzzles ever.

In my copy of the book, all of the puzzles were already filled in which I find really annoying and what is worse, most of them have been filled in wrongly.

I have been through the whole book really carefully and only found seven puzzles that had been filled out correctly! Yes, just seven.

Well, making the best of a bad job, I am now going through the book trying to correct all of the faulty puzzles and I will then submit my corrections.

Perhaps a second edition will be more useful.

I did find last week's winning lottery numbers on page 18 though.

Comment 135 people found this helpful. Was this review helpful to you? Yes No Report abuse

Too unpredictable

By pontifex on January 24, 2011

Format: Paperback

The book is too hard to follow, the author randomly shifts from one number to another without any prior warning.

1 comment | 408 people found this helpful. Was this review helpful to you? Yes No Report abuse

Not really random

By TDB on September 26, 2012

Format: Paperback

physical Methods

manual methods: coin toss, dice throw, drawing from an urn

objects that appear random: computer clock

physical devices: circuit noise, gamma-ray detectors

advantage

"true" random numbers (critical for cryptographic applications)

- for example, check out Radiolab podcast on launching a cryptocurrency

drawbacks

- **slow** (if generated as needed)
  expensive (if precomputed and stored in memory)
- **bias** may exist in the device
  for example, see Persi Diaconis on coin-tossing
- **hard to replicate** the random input sequence

essential for simulation
hash functions
(big-data algorithms)

'pseudo-random' generators

- mid-square method (von Neumann, 1949)

$$
\begin{aligned}
8234 \times 8234 &= 67(7987)56 \\
7987 \times 7987 &= 63(7921)69 \\
7921 \times 7921 &= 62(7422)41 \\
7422 \times 7422 &= 55(0860)84 \quad \ldots .
\end{aligned}
$$

0.8234
0.7987
0.7921
⋮

- **objection**: random numbers are not random at all!
  - this criticism applies to all pseudo-random number generators
  - need tests to determine if algorithm produces "valid" outputs

linear congruential generators (LCG)

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

- (fixed) parameters: modulus m, multiplier a, increment c integer $a, c \in \{0, \dots, m-1\}$
- seed: X_0 (the first input) is typically supplied by the user (seed) $X_0 \in [m-1]$
- each X_n is an integer in the set $\{0, 1, \dots, m-1\}$.
- to get pseudorandom number $U_n \in (0, 1)$, set:

- $U_n = \frac{X_n}{m}$ (may get $U_n = 0$ - Undesirable)
- $\boxed{U_n = \frac{X_n + 1}{m + 1}} \in (0, 1)$

LCG: example

$m = 32, \quad a = 11, \quad c = 0,$ different seeds.

n	X_n	X_n	X_n
0	1	2	4
1	11	22	12
2	25	18	4
3	19	6	12
4	17	2	4
5	27	22	12
6	9	18	4
7	3	6	12
8	1	2	4
9	11	22	12
$\vdots$	$\vdots$	$\vdots$	$\vdots$

different seeds

$(1 \times 11 + 0) \bmod 32$

$(4 \times 11 + 0) \bmod 32$

LCG: properties

- an LCG produces periodic output. period $\leq m$
  1. if period $= m$ with a seed X_0, then period $= m$ *for any* seed
  2. if period $< m$, then it may depend on the seed
- full period is desirable:
  1. one should never use the whole period of a LCG, otherwise dependencies between the random numbers will occur.
  2. not have full period $\implies$ gaps in the output sequence.
- full period $\implies$ granularity $= 1/m$.
  not a problem when m is large.

LCG: theory

(only for reference for anyone interested)

the period of LCGs is well understood.

Theorem: LCGs with full period

an LCG(m, a, c) has full period if all of the following are true

1. m and c are relatively prime.
2. if q is a prime number that divides m, then q divides $a - 1$.
3. if 4 divides m, then 4 divides $a - 1$.

Corollary:

an LCG with $m = 2^b$ has full period if c is odd and 4 divides $a - 1$.

example – $m = 8, \quad a = 5, \quad c = 3$

multiplicative generators

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

even faster as only using multiplication and modulo

- $X_n = 0 \implies X_n = X_{n+1} = X_{n+2} = \ldots = 0$
- for an MG, Full period $\implies \{1, \ldots, m-1\}$.

Theorem: MGs with full period

an MG has full period if all of the following are true

1. m is prime 2. $a^{m-1} - 1$ is divisible by m.

3. there is no $j < m-1$ such that $a^j - 1$ is divisible by m.

Theorem: Sufficient conditions for good MGs

the largest possible period for a MG with $m = 2^b$ is $m/4$

this is achieved when X_0 is odd and a is of the form:

$$a = 3 + 8k \quad \text{or} \quad a = 5 + 8k,$$

for some positive integer k.

deficiencies of LCGs

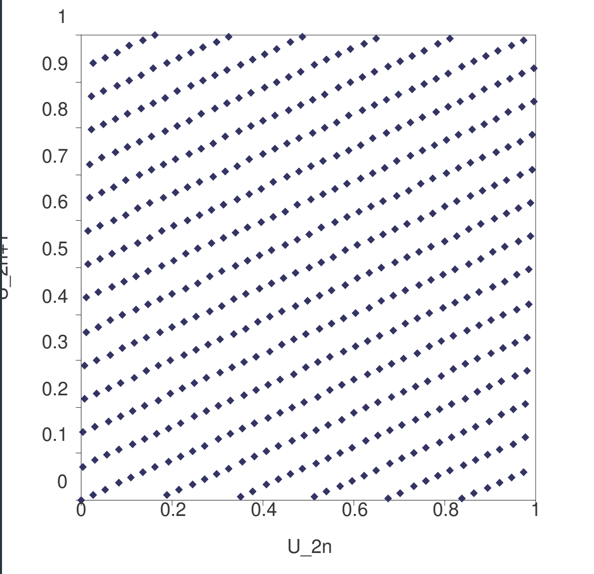
- LCG's possess **theoretical deficiencies**
  (any deterministic generator must have deficiencies.)
- if $U_0, U_1, \ldots$ are iid $U[0,1]$, then the points

$$(U_0, U_1),\ (U_2, U_3),\ (U_4, U_5), \ldots$$

  should lie uniformly over the square $[0,1] \times [0,1]$.
- suppose $U_0, U_1, \ldots$ be generated by a LCG:
  How do the points $(U_0, U_1),\ (U_2, U_3),\ (U_4, U_5), \ldots$ behave?

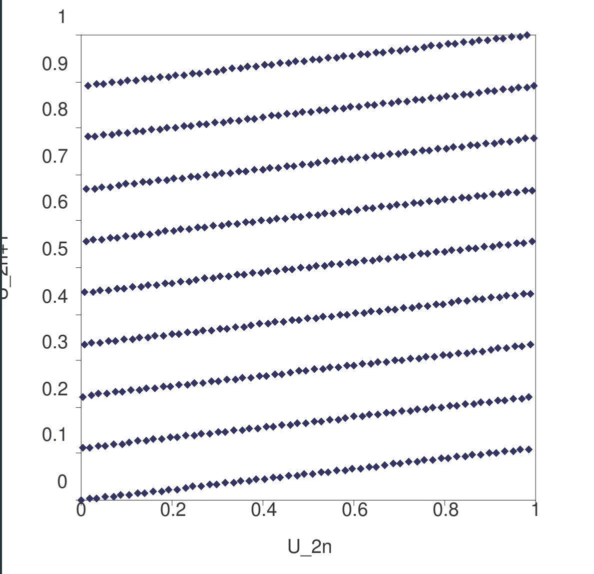
deficiencies of LCGs

$$m = 2^{10},\ a = 37,\ c = 1$$



deficiencies of LCGs

$m = 2^{10}, \ a = 57, \ c = 1$



deficiencies of LCGs

- the points

$$(U_0, U_1),\ (U_2, U_3),\ (U_4, U_5), \ldots$$

  lie on a relatively small number of parallel lines!

- in general, the points

$$(U_0, U_1, U_2, \ldots, U_{d-1}),\ (U_d, U_{d+1}, U_{d+2}, \ldots, U_{2d-1}), \ldots$$

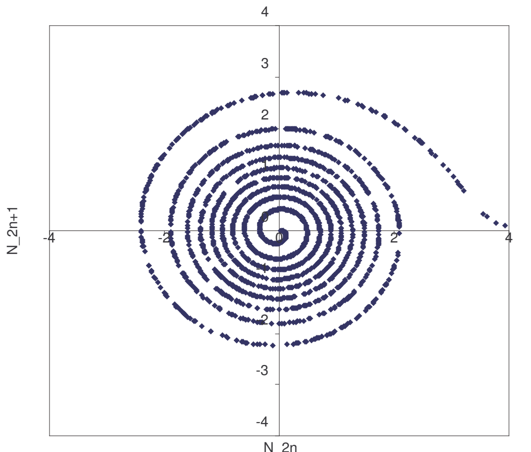
  lie on parallel $(d-1)$-dimensional hyperplanes!

- problematic in simulations of geometric phenomena.
  OK for discrete-event simulations.

deficiencies of LCGs for generating other rvs

let $N_0, N_1, N_2, \ldots$ be samples from $\mathcal{N}(0,1)$ generated using the Box-Muller method using $U_0, U_1, U_2, \ldots$ from an LCG

– then the pairs $(N_0, N_1),\ (N_2, N_3),\ (N_4, N_5), \ldots$ lie on a spiral in two-dimensional space. E.g., $a = 9, m = 2^{21}, c = 1$



combining generators

- $m = 2^{31} - 1$ is popular, but period is only about 2 billion.
- not sufficient! E.g. traffic simulators need lots of random numbers. (10s of 000s of vehicles, 1000s of random disturbances, lots of replications).
- shouldn't use anywhere near the full period - maybe $\leq 1\%$
- to generate longer period, take two MG's

$$X_{n+1} = (a_1 X_n) \bmod m_1 \quad , \quad Y_{n+1} = (a_2 Y_n) \bmod m_2$$

  and set

$$Z_n = (X_n + Y_n) \bmod m_3.$$

- period can be on the order of $m_1 m_2$. For example, set $a_1 = 40014$, $a_2 = 40692$, $m_1 = 2147483563$, $m_2 = 2147483399$ and $m_3 = m_1$.
- can combine more than two.

streams and substreams

- useful to divide the numbers produced by a PRNG into streams and substreams
- stream = simulation replication
  substream = source of randomness
- useful for debugging and for variance-reduction techniques

tl;dr

- hyperplane/spiral problems are well understood (and avoided)
- current generators have been carefully tested, pass lots of statistical tests (but must fail at least one test...)

the last word

modern PRNGs are

- random enough for your sim answer to be correct
- deterministic enough (by setting seed) for your sim to be repeatable

- IMPORTANT -

np.random.seed(0)

(set different seeds for testing/debugging)

for repeatable expts