Generating Uniform Random Numbers

Christos Alexopoulos and Dave Goldsman

Georgia Institute of Technology, Atlanta, GA, USA

3/22/20

Outline

1. Introduction
2. Some Lousy Generators We Won't Use
3. Linear Congruential Generators
4. Tausworthe Generator
5. Generalizations of LCGs
6. Choosing a Good Generator — Some Theory
7. Choosing a Good Generator — Statistical Tests
   - χ^2 Goodness-of-Fit Test
   - Tests for Independence

Introduction

Uniform(0,1) random numbers are the key to random variate generation in simulation — you transform uniforms to get other RVs.

Goal: Give an algorithm that produces a sequence of *pseudo-random numbers (PRNs)* $R_1, R_2, \ldots$ that "appear" to be iid Unif(0,1).

Desired properties of algorithm

- output appears to be iid Unif(0,1)
- very fast
- ability to reproduce any sequence it generates

References: Banks, Carson, Nelson, and Nicol (2010); Bratley, Fox, and Schrage (1987); Knuth (2) (1981); Law (2015).

Classes of Unif(0,1) Generators

- Some lousy generators
  - output of random device
  - table of random numbers
  - midsquare
  - Fibonacci
- Linear congruential (most commonly used in practice)
- Tausworthe (linear recursion mod 2)
- Hybrid

Outline

1 Introduction

2 Some Lousy Generators We Won't Use

3 Linear Congruential Generators

4 Tausworthe Generator

5 Generalizations of LCGs

6 Choosing a Good Generator — Some Theory

7 Choosing a Good Generator — Statistical Tests
- χ^2 Goodness-of-Fit Test
- Tests for Independence

Some Generators We Won't Use

a. Random Devices

Nice randomness properties. However, Unif(0,1) sequence storage difficult, so it's tough to repeat experiment.

Examples:

- flip a coin
- particle count by Geiger counter
- least significant digits of atomic clock

b. Random Number Tables

List of digits supplied in tables.

- *A Million Random Digits with 100,000 Normal Deviates*
  www.rand.org/content/dam/rand/pubs/monograph_reports/MR1418/MR1418.digits.pdf

Cumbersome, slow, tables too small — not very useful.

Once tabled no longer random.

c. Mid-Square Method (J. von Neumann)

Idea: Take the middle part of the square of the previous random number. John von Neumann was a brilliant and fun-loving guy, but method is terrible!

Example: Take $R_i = X_i/10000$, $\forall i$, where the X_i's are positive integers < 10000.

Set *seed* $X_0 = 6632$; then $6632^2 \rightarrow 43\mathbf{983}424$;
So $X_1 = 9834$; then $9834^2 \rightarrow 96\mathbf{707}56$;
So $X_2 = 7075$, etc,...

Unfortunately, positive serial correlation in R_i's.

Also, occasionally degenerates; e.g., consider $X_i = 0003$.

d. Fibonacci and Additive Congruential Generators

These methods are also no good!!

Take

$$X_i = (X_{i-1} + X_{i-2}) \bmod m, \quad i = 1, 2, \ldots,$$

where $R_i = X_i/m$, m is the *modulus*, X_{-1}, X_0 are *seeds*, and $a = b \bmod m$ iff a is the remainder of b/m, e.g., $6 = 13 \bmod 7$.

Problem: Small numbers follow small numbers.

Also, it's not possible to get $X_{i-1} < X_{i+1} < X_i$ or $X_i < X_{i+1} < X_{i-1}$ (which should occur w.p. 1/3).

Outline

1 Introduction

2 Some Lousy Generators We Won't Use

3 Linear Congruential Generators

4 Tausworthe Generator

5 Generalizations of LCGs

6 Choosing a Good Generator — Some Theory

7 Choosing a Good Generator — Statistical Tests

- χ^2 Goodness-of-Fit Test
- Tests for Independence

Linear Congruential Generators

LCGs are the most widely used generators. These are pretty good when implemented properly.

$X_i = (aX_{i-1} + c) \bmod m$, where X_0 is the seed.

$R_i = X_i/m,\ i = 1, 2, \ldots$

Choose a, c, m carefully to get good statistical quality and long *period* or *cycle length*, i.e., time until LCG starts to repeat itself.

If $c = 0$, LCG is called a *multiplicative* generator.

Trivial Example: For purposes of illustration, consider the LCG

$$X_i = (5X_{i-1} + 3) \bmod 8$$

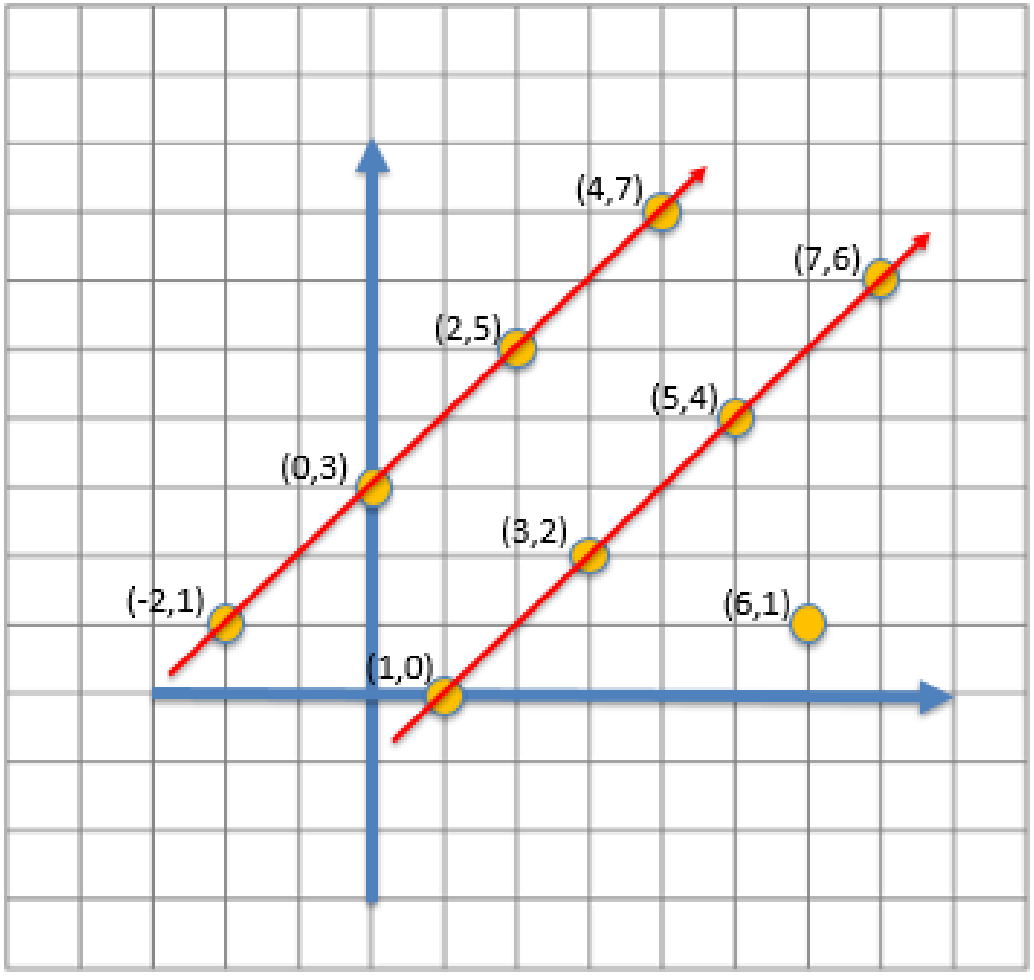
If $X_0 = 0$, we have $X_1 = (5X_0 + 3) \bmod 8 = 3$; continuing,

i	0	1	2	3	4	5	6	7	8	9
X_i	0	3	2	5	4	7	6	1	0	3
R_i	0	$\frac{3}{8}$	$\frac{2}{8}$	$\frac{5}{8}$	$\frac{4}{8}$	$\frac{7}{8}$	$\frac{6}{8}$	$\frac{1}{8}$	0	$\frac{3}{8}$

so that the sequence starts repeating with $X_8 = 0$.

This is a *full-period generator*, since it has cycle length $m = 8$.
Generally speaking, full-period is a good thing. □

Plot of (X_{i-1}, X_i)



The random numbers fall mainly on the planes

Easy Exercises:

1. For the above generator, plot X_i vs. X_{i-1} and see what happens.

2. Consider the generator

$$X_i = (5X_{i-1} + 2) \bmod 8$$

(which is very similar to the previous one). Does it achieve full cycle? Explain.

Better Example (desert island generator): Here's our old 16807 implementation (BFS 1987), which I've translated from FORTRAN. It works fine, is fast, and is full-period with cycle length > 2 billion,

$$X_i \;=\; 16807\, X_{i-1} \bmod (2^{31} - 1).$$

Algorithm: Let X_0 be an integer seed between 1 and $2^{31} - 1$.

For $i = 1, 2, \ldots,$

$K \leftarrow \lfloor X_{i-1} \,/\, 127773 \rfloor$ (integer division leaves no remainder)
$X_i \leftarrow 16807(X_{i-1} - 127773K) - 2836K$
if $X_i < 0$, then set $X_i \leftarrow X_i + 2147483647$
$R_i \leftarrow X_i * 4.656612875\text{E-}10$

Example: If $X_0 = 12345678$, then $K = 96$ and

$$X_1 \leftarrow 16807\big[12345678 - 127773(96)\big] - 2836(96) = 1335380034,$$

so that

$$R_1 \;=\; 1335380034 * 4.656612875\text{E-}10 \;=\; 0.621835.$$

Similarly, you get $R_2, R_3, \ldots$. $\square$

Stay tuned for various ways to assess the quality of PRN generators.

First of all, what can go wrong with LCGs?

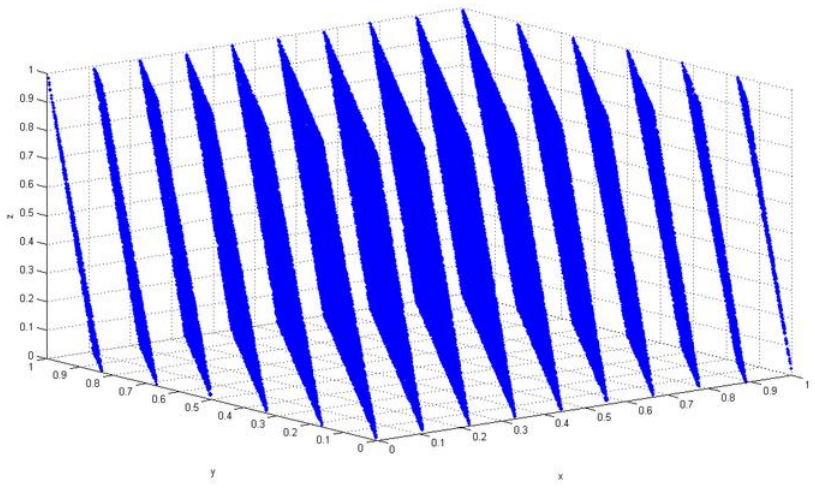
- Something like $X_i = (4X_{i-1} + 2) \bmod 8$ is *not* full-period, since it only produces even integers.
- Something like $X_i = (X_{i-1} + 1) \bmod 8$ is full-period, but it produces very non-random output: $X_1 = 1$, $X_2 = 2$, $X_3 = 3$, etc.
- In any case, if m is small, you'll get quick cycling whether or not the generator is full period. "Small" could mean anything less than 2 billion or so!
- And just because m is big, you still have to be careful. Some subtle problems can arise. Take a look at RANDU....

Example: The infamous RANDU generator,

$$X_i \;=\; 65539\, X_{i-1} \bmod 2^{31},$$

was popular during the 1960's.

Here's what (R_{i-2}, R_{i-1}, R_i) look like if you plot them in 3-D (stolen from Wikipedia). If they were truly iid Unif(0,1), you'd see dots randomly dispersed in the unit cube. But instead, the random numbers fall entirely on 15 hyperplanes (not good).



Exercises:

1. Implement RANDU and see how it does. That is, plot R_{i-1} vs. R_i for $i = 1, 2, \ldots, 1000$ and see what happens. Try a few different seeds and maybe you'll see some hyperplanes. You'll certainly see them if you plot (R_{i-2}, R_{i-1}, R_i) in 3-D.

2. Now do the same thing with the 16807 generator. You probably won't be able to see any hyperplanes here.

Outline

1 Introduction

2 Some Lousy Generators We Won't Use

3 Linear Congruential Generators

4 Tausworthe Generator

5 Generalizations of LCGs

6 Choosing a Good Generator — Some Theory

7 Choosing a Good Generator — Statistical Tests
- χ^2 Goodness-of-Fit Test
- Tests for Independence

Tausworthe Generator

Define a sequence of binary digits $B_1, B_2, \ldots$, by

$$B_i \;=\; \left(\sum_{j=1}^{q} c_j B_{i-j}\right) \bmod 2,$$

where $c_j = 0$ or 1. Looks a bit like a generalization of LCGs.

Usual implementation (saves computational effort):

$$B_i \;=\; (B_{i-r} + B_{i-q}) \bmod 2 \;=\; B_{i-r} \text{ XOR } B_{i-q} \quad (0 < r < q).$$

Obtain

$$B_i = 0, \text{ if } B_{i-r} = B_{i-q} \quad \text{or} \quad B_i = 1, \text{ if } B_{i-r} \neq B_{i-q}.$$

To initialize the B_i sequence, specify $B_1, B_2, \ldots, B_q$.

Example (Law 2015): $r = 3, q = 5; B_1 = \cdots = B_5 = 1$. Obtain

$$B_i = (B_{i-3} + B_{i-5}) \bmod 2 = B_{i-3} \text{ XOR } B_{i-5}, \quad i > 5$$

$$B_6 = (B_3 \text{ XOR } B_1) = 0, B_7 = (B_4 \text{ XOR } B_2) = 0, \text{ etc.}$$

$$1111\ 1000\ 1101\ 1101\ 0100\ 0010\ 0101\ 1001\ 1111 \quad \square$$

The period of 0-1 bits is always $2^q - 1 = 31$.

How do we go from B_i's to Unif(0,1)'s?

Easy way: Use (ℓ-bits in base 2)/2^ℓ and convert to base 10.

Example: Set $\ell = 4$ in previous example and get:

$$1111_2, 1000_2, 1101_2, 1101_2, \ldots \to \frac{15}{16}, \frac{8}{16}, \frac{13}{16}, \frac{13}{16}, \ldots. \quad \square$$

Lots of potential for Tausworthe generators. Nice properties, including long periods, fast calculation.

Outline

1 Introduction

2 Some Lousy Generators We Won't Use

3 Linear Congruential Generators

4 Tausworthe Generator

5 Generalizations of LCGs

6 Choosing a Good Generator — Some Theory

7 Choosing a Good Generator — Statistical Tests

- χ^2 Goodness-of-Fit Test
- Tests for Independence

Generalizations of LCGs

A Simple Generalization:

$X_i = \left(\sum_{j=1}^{q} a_j X_{i-j}\right) \bmod m$, where the a_j's are constants.

Extremely large periods possible (up to $m^q - 1$ if parameters are chosen properly). But watch out! — Fibonacci is a special case.

Combinations of Generators:

Can combine two generators $X_1, X_2, \ldots$ and $Y_1, Y_2, \ldots$ to construct $Z_1, Z_2, \ldots$. Some suggestions:

- Set $Z_i = (X_i + Y_i) \bmod m$
- Shuffling
- Set $Z_i = X_i$ or $Z_i = Y_i$

Sometimes desired properties are improved (difficult to prove).

A Really Good Combined Generator due to L'Ecuyer (1999) (and discussed in Law 2015).

Initialize $X_{1,0}, X_{1,1}, X_{1,2}, X_{2,0}, X_{2,1}, X_{2,2}$. For $i \geq 3$, set

$$\begin{aligned} X_{1,i} &= (1{,}403{,}580\, X_{1,i-2} - 810{,}728\, X_{1,i-3}) \bmod (2^{32} - 209) \\ X_{2,i} &= (527{,}612\, X_{2,i-1} - 1{,}370{,}589\, X_{2,i-3}) \bmod (2^{32} - 22{,}853) \\ Y_i &= (X_{1,i} - X_{2,i}) \bmod (2^{32} - 209) \\ R_i &= Y_i / (2^{32} - 209) \end{aligned}$$

As crazy as this generator looks, it's actually pretty simple, works well, and has an amazing cycle length of about 2^{191}!

Remarks:

- It is of interest to note that Matsumoto and Nishimura have developed the "Mersenne Twister" generator, which has a period of $2^{19937} - 1$ (yes, that's a prime number).
- You'll often need several billion PRNs in any practical application, but 2^{191} or 2^{19937} is a bit of overkill, eh? ☺
- These days, all standard stats and simulation packages use a good generator.
- Nevertheless, it's a good idea to discuss how one would choose a good generator, just in case....

Outline

1 Introduction

2 Some Lousy Generators We Won't Use

3 Linear Congruential Generators

4 Tausworthe Generator

5 Generalizations of LCGs

6 Choosing a Good Generator — Some Theory

7 Choosing a Good Generator — Statistical Tests
- χ^2 Goodness-of-Fit Test
- Tests for Independence

Choosing a Good Generator — Some Theory

Here are some miscellaneous results due to Knuth and others that are helpful in determining the quality of a PRN generator.

Theorem: The generator $X_i = aX_{i-1} \bmod 2^n$ $(n > 3)$ can have cycle length of at most 2^{n-2}. This is achieved when X_0 is odd and $a = 8k + 3$ or $a = 8k + 5$ for some k.

Example (BCNN): $X_i = 13X_{i-1} \bmod(64)$.

X_0	X_1	X_2	X_3	X_4	$\cdots$	X_8	$\cdots$	X_{16}
1	13	41	21	17	$\cdots$	33	$\cdots$	**1**
2	26	18	42	34	$\cdots$	**2**		
3	39	56	63	51	$\cdots$	35	$\cdots$	**3**
4	52	36	20	**4**				

The minimum period = 4... terrible random numbers! □

Lots of these types of cycle length results.

Theorem: $X_i = (aX_{i-1} + c) \bmod m$ $(c > 0)$ has full cycle if (i) c and m are relatively prime; (ii) $a - 1$ is a multiple of every prime which divides m; and (iii) $a - 1$ is a multiple of 4 if 4 divides m.

Corollary: $X_i = (aX_{i-1} + c) \bmod 2^n$ $(c, n > 1)$ has full cycle if c is odd and $a = 4k + 1$ for some k.

Theorem: The *multiplicative* generator $X_i = aX_{i-1} \bmod m$, with prime m has full period $(m - 1)$ if and only if (i) m divides $a^{m-1} - 1$; and (ii) for all integers $i < m - 1$, m does not divide $a^i - 1$.

Remark: For $m = 2^{31} - 1$, it can be shown that 534,600,000 multipliers yield full period, the "best" of which is $a = 950{,}706{,}376$ (Fishman and Moore 1986).

Geometric Considerations

Theorem: The k-tuples $(R_i, \ldots, R_{i+k-1})$, $i \geq 1$, from multiplicative generators lie on parallel hyperplanes in $[0, 1]^k$.

The following geometric quantities are of interest.

- Minimum number of hyperplanes (in all directions). Find the multiplier that maximizes this number.
- Maximum distance between parallel hyperplanes. Find the multiplier that minimizes this number.
- Minimum Euclidean distance between adjacent k-tuples. Find the multiplier that maximizes this number.

Remark: The RANDU generator is particularly bad since it lies on only 15 hyperplanes.

Can also look at one-step serial correlation.

Serial Correlation of LCGs (Greenberger 1961):

$$\text{Corr}(R_1, R_2) \leq \frac{1}{a}\left(1 - \frac{6c}{m} + 6\left(\frac{c}{m}\right)^2\right) + \frac{a+6}{m}$$

This upper bound is very small for m in the range of 2 billion and, say, $a = 16807$.

Lots of other theory considerations that can be used to evaluate the performance of a particular PRN generator.

Outline

1 Introduction

2 Some Lousy Generators We Won't Use

3 Linear Congruential Generators

4 Tausworthe Generator

5 Generalizations of LCGs

6 Choosing a Good Generator — Some Theory

7 Choosing a Good Generator — Statistical Tests
- χ^2 Goodness-of-Fit Test
- Tests for Independence

Choosing a Good Generator — Statistical Tests

We'll look at two classes of tests:

Goodness-of-fit tests — are the PRNs approximately Unif(0,1)?

Independence tests — are the PRNs approximately independent?

If a particular generator passes both types of tests (in addition to other tests that we won't tell you about), we'll be happy to use the PRNs it generates.

All tests are of the form H_0 (our null hypothesis) vs. H_1 (the alternative hypothesis).

We regard H_0 as the status quo, so we'll only reject H_0 if we have "ample" evidence against it. (Innocent until proven guilty.)

In fact, we want to avoid incorrect rejections of the null hypothesis. Thus, when we design the test, we'll set the *level of significance*

$$\alpha \equiv P(\text{Reject } H_0 | H_0 \text{ true}) = P(\text{Type I error})$$

(typically, $\alpha = 0.05$ or 0.1).

We can also define

$$\beta \equiv P(\text{Fail to Reject } H_0 | H_0 \text{ false}) = P(\text{Type II error}),$$

but we won't worry about Type II error at this point.

χ^2 Goodness-of-Fit Test

Test $H_0 : R_1, R_2, \ldots R_n \sim$ Unif(0,1).

Divide the unit interval into k cells (subintervals). If you choose equi-probable cells $[0, \frac{1}{k}), [\frac{1}{k}, \frac{2}{k}), \ldots, [\frac{k-1}{k}, 1]$, then a particular observation R_j will fall in a particular cell with prob $1/k$.

Tally how many of the n observations fall into the k cells. If $O_i \equiv$ # of R_j's in cell i, then (since the R_j's are i.i.d.), we can easily see that $O_i \sim \text{Bin}(n, \frac{1}{k})$, $i = 1, 2, \ldots, k$.

Thus, the expected number of R_j's to fall in cell i will be $E_i \equiv \text{E}[O_i] = n/k$, $i = 1, 2, \ldots, k$.

We reject the null hypothesis H_0 if the O_i's don't match the E_i's well.

The χ^2 goodness-of-fit statistic is

$$\chi_0^2 \equiv \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}.$$

A large value of this statistic indicates a bad fit.

In fact, we *reject* the null hypothesis H_0 (that the observations are uniform) if $\chi_0^2 > \chi_{\alpha,k-1}^2$, where $\chi_{\alpha,k-1}^2$ is the appropriate $(1-\alpha)$ quantile from a χ^2 table, i.e., $P(\chi_{k-1}^2 < \chi_{\alpha,k-1}^2) = 1-\alpha$.

If $\chi_0^2 \leq \chi_{\alpha,k-1}^2$, we *fail to reject* H_0.

Usual recommendation from baby stats class: For the χ^2 g-o-f test to work, pick k, n such that $E_i \geq 5$ and n at least 30. But...

Unlike what you learned in baby stats class, when we test PRN generators, we usually have a *huge* number of observations n (at least millions) with a large number of cells k. When k is large, we can use the approximation

$$\chi^2_{\alpha,k-1} \approx (k-1)\left[1 - \frac{2}{9(k-1)} + z_\alpha\sqrt{\frac{2}{9(k-1)}}\right]^3,$$

where z_α is the appropriate standard normal quantile.

Remarks: (1) 16807 PRN generator usually passes the g-o-f test just fine. (2) We'll show how to do g-o-f tests for other distributions later on — just doing uniform PRNs for now. (3) Other g-o-f tests: Kolmogorov–Smirnov test, Anderson–Darling test, etc.

Illustrative Example: $n = 1000$ observations, $k = 5$ intervals.

interval	[0,0.2]	(0.2,0.4]	(0.4,0.6]	(0.6,0.8]	(0.8,1.0]
E_i	200	200	200	200	200
O_i	179	208	222	199	192

Turns out that

$$\chi_0^2 \equiv \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} = 5.27.$$

Let's take $\alpha = 0.05$. Then from χ^2 tables, we have

$$\chi^2_{\alpha,k-1} = \chi^2_{0.05,4} = 9.49.$$

Since $\chi_0^2 < \chi^2_{\alpha,k-1}$, we fail to reject H_0, and so we'll assume that the observations are approximately uniform. □

Runs Tests for Independence

Now consider the hypothesis $H_0 : R_1, R_2, \ldots, R_n$ are independent.

Let's look at Three Little Bears examples of coin tossing:

A: H, T, H, T, H, T, H, T, H, T,... (negative correlation)
B: H, H, H, H, H, T, T, T, T, T,... (positive correlation)
C: H, H, H, T, T, H, T, T, H, T,... ("just right")

Definition: A *run* is a series of similar observations.

In A above, the runs are: "H", "T", "H", "T",.... (many runs)

In B, the runs are: "HHHHH", "TTTTT", ... (very few runs)

In C: "HHH", "TT", "H", "TT",.... (medium number of runs)

A *runs test* will reject the null hypothesis of independence if there are "too many" or "too few" runs, whatever that means. There are various types of runs tests; we'll discuss two of them.

Runs Test "Up and Down". Consider the following sequence of uniforms.

$$.41 \quad .68 \quad .89 \quad .84 \quad .74 \quad .91 \quad .55 \quad .71 \quad .36 \quad .30 \quad .09\ldots$$

If the uniform increases, put a $+$; if it decreases, put a $-$ (like H's and T's). Get the sequence

$$+ + - - + - + - - - \ldots$$

Here are the associated runs:

$$++, --, +, -, +, - - -, \ldots$$

So do we have too many or two few runs?

Let A denote the total number of runs "up and down" out of n observations. ($A = 6$ in the above example.)

Amazing Fact: If n is large (say, ≥ 20) and the R_j's are actually independent, then

$$A \;\approx\; \text{Nor}\left(\frac{2n-1}{3},\, \frac{16n-29}{90}\right).$$

We'll reject H_0 if A is too big or small. The test statistic is

$$Z_0 \;=\; \frac{A - \text{E}[A]}{\sqrt{\text{Var}(A)}},$$

and we reject H_0 if $|Z_0| > z_{\alpha/2}$.

Illustrative Example: Suppose that $n = 100$ and $A = 55$. Then $A \approx \text{Nor}(66.33, 17.46)$, and so $Z_0 = -2.71$. If $\alpha = 0.05$, then $z_{\alpha/2} = 1.96$, and we reject independence. $\square$

Runs Test "Above and Below the Mean". Again consider

$$.41 \quad .68 \quad .89 \quad .84 \quad .74 \quad .91 \quad .55 \quad .71 \quad .36 \quad .30 \quad .09\ldots$$

If $R_i \geq 0.5$, put a $+$; if $R_i < 0.5$, put a $-$. Get the sequence

$$- + + + + + + + + - - -\ldots$$

Here are the associated runs, of which there are $B = 3$:

$$-, \quad + + + + + + + +, \quad - - -$$

Fact: If n is large and the R_j's are actually independent, then

$$B \;\approx\; \text{Nor}\left(\frac{2n_1n_2}{n} + \frac{1}{2},\; \frac{2n_1n_2(2n_1n_2 - n)}{n^2(n-1)}\right),$$

where n_1 is the number of observations ≥ 0.5 and $n_2 = n - n_1$.

The test statistic is $Z_0 = (B - \mathrm{E}[B])/\sqrt{\mathrm{Var}(B)}$, and we reject H_0 if $|Z_0| > z_{\alpha/2}$.

Illustrative Example (from BCNN): Suppose that $n = 40$, with the following $+/-$ sequence.

$$- + + + + + + + + - - - + + - + - - - - -$$
$$- - + + - - - - + + - - + - + - - + + -$$

Then $n_1 = 18$, $n_2 = 22$, and $B = 17$. This implies that $\mathrm{E}[B] \doteq 20.3$ and $\mathrm{Var}(B) \doteq 9.54$. And this yields $Z_0 = -1.07$.

Since $|Z_0| < z_{\alpha/2} = 1.96$, we *fail* to reject the test; so we can treat the observations as independent. $\square$

Lots of other tests available for independence: Other runs tests, correlation tests, gap test, poker test, birthday test, etc.

Correlation Test. Assuming that the R_i's are all Unif(0,1), let's conduct a correlation test for H_0: R_i's independent.

We define the *lag-1 correlation* of the R_i's by $\rho \equiv \text{Corr}(R_i, R_{i+1})$. Ideally, ρ should equal zero. A good estimator for ρ is given by

$$\hat{\rho} \equiv \frac{12}{n-1} \sum_{k=1}^{n-1} R_k R_{1+k} - 3.$$

Under H_0, it turns out that

$$\hat{\rho} \approx \text{Nor}\left(0, \frac{13n-19}{(n-1)^2}\right).$$

The test statistic $Z_0 = \hat{\rho}/\sqrt{\text{Var}(\hat{\rho})}$, and we reject if $|Z_0| > z_{\alpha/2}$.

Illustrative Example: Consider the following $n = 30$ PRNs.

0.29	0.38	0.46	0.29	0.69	0.73	0.80	0.74	0.99	0.75
0.88	0.66	0.56	0.41	0.35	0.22	0.18	0.05	0.25	0.36
0.39	0.45	0.50	0.62	0.76	0.81	0.97	0.72	0.11	0.55

After a little algebra, we get

$$\hat{\rho} = 0.950 \quad \text{and} \quad \text{Var}(\hat{\rho}) = \frac{13n - 19}{(n-1)^2} = 0.441.$$

So $Z_0 = 0.950/\sqrt{0.441} = 1.43$.

Since $|Z_0| < z_{\alpha/2} = 1.96$, we *fail* to reject the test, meaning that we can treat the PRNs as independent. (Of course, $n = 30$ is sort of small, and perhaps this decision will change if we increase n.) □