Tutorial: Random Number Generation

John Lau
johnlau@umail.ucsb.edu

Henry Yu
henryyu@umail.ucsb.edu

June 2018

1 Abstract

This tutorial will cover the basics of Random Number Generation. This includes properties of random numbers and pseudo-random numbers, generation of pseudo-random numbers, physical and computational techniques and methods for generating random numbers, tests for random numbers, as well as applications. We will cover computational techniques such as Linear and Multiplicative Congruential Method, Combined Linear Congruential Generators as well as Random Number Streams. For tests of uniformity and independence, we will discuss frequency test, runs test, gap test, poker test, Kolmogorov-Smirnov test, and Chi-Square test. We will compare True random numbers to that of pseudo-random numbers. Lastly, we will discuss some of the applications of Random Numbers.

2 Motivation

- Games → Lottery, Casino
- Politics → Allotment
- Religion → Divination
- Statistics → Sampling, Analysis, Simulations
- Art, Music → Stochastic, Generative Music
- Cryptography → Encryption

3 Introduction

Before we dive into random number generators, let us first understand a little bit about the background of randomness. Random numbers have traditionally been used for a variety of purposes throughout history, ranging from gambling dice games, coin flipping to being the foundation of many security protocols and

functions. But is rolling a die, picking a card out of a shuffled deck, or flipping a coin truly random? In order to be truly random, the die or coin should land on all possible values equally, and can not have any obvious patterns in a sequence.

With the rise of technology of computers, the means of introducing true randomness proves to be quite difficult as computer instructions are predictable in nature. Thus, there becomes a split in random number generation: Pseudo Random Number Generator (PRNG) and True Random Number Generator (TRNG). There are also Hybrid Random Number Generators and Quantum Random Number Generators but we will only focus on Pseudo and True in this short tutorial.

4 Properties

A sequence of random numbers R1, ... Ri must fulfill two statistical properties:

- Uniformity
  - Consequence of this property: If the interval [0,1] is divided into n classes, or sub intervals of equal length, the expected number of observations in each interval is N/n, where N is the total number of observations
- Independence
  - Consequence of this property: The probability of observing a value in a particular interval is independent of the previous value drawn

The random number Ri must be independently drawn from a uniform distribution with pdf:

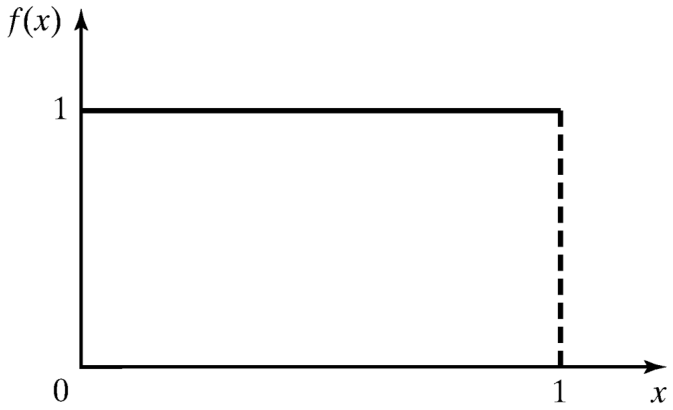
$$f(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & otherwise \end{cases}$$

Expected Value:

$$E(R) = \int_0^1 x \, \mathrm{d}x = \frac{x^2}{2}\Big|_0^1 = \frac{1}{2}$$

Variance:

$$V(R) = \int_0^1 x^2 \, \mathrm{d}x - [E(R)]^2 = \frac{x^3}{3}\Big|_0^1 - (\tfrac{1}{2})^2 = \frac{1}{12}$$



Other properties that random number generators should have:

- Efficiency
  - The generator should be fast and efficient.
- Reproducibility
  - The generator should be able to generate the same stream of random numbers repeatedly. This is mainly for testing and debugging purposes.
- Long Cycle Length
  - The generator should take a very long time before numbers start to repeat.

5 Pseudo-Random Number Generation

It is considered "Pseudo" because generating numbers using a known method disqualifies the potential of true randomness. Computers with no access to outside "true randomness" phenomena can only run deterministic algorithms, thus rendering it at most a pseudo random number generator. There are multiple algorithms for generating pseudo random numbers.

- Linear Congruential Method
  - To produce a sequence of integers, X1, X2, ... between 0 and m-1 by following a recursive relationship:

    $$X_{i+1} = (aX_i + c) \mod m, i = 0, 1, 2...$$

    X0 is the seed

The selection of the values for a, c, m, and X0 drastically affects the statistical properties and the cycle length.

The random integers are being generated in the range [0,m-1], and to convert the integers to random numbers:

$$R_i = \frac{X_i}{m}, i = 1, 2...$$

Example: Use X0 = 27, a = 17, c = 43, and m = 100.

The Xi and Ri values are:

$$X_1 = (17 * 27 + 43) \mod 100 = 502 \mod 100 = 2, R_1 = 0.02;$$

$$X_2 = (17 * 2 + 43) \mod 100 = 77 \mod 100 = 77, R_2 = 0.77;$$

$$X_3 = (17 * 77 + 43) \mod 100 = 1352 \mod 100 = 52 R_3 = 0.52;$$

Notice that the numbers generated assume values only from the set

$$I = 0, 1/m, 2/m, ..., (m-1)/m$$

because each Xi is an integer in the set 0,1,2,. . . .,m-1

Thus each Ri is discrete on I, instead of continuous on interval [0,1]

Generally speaking, do not use the lower order bits from linear congruential generators as they are not very random. The congruential PRNG has the disadvantage that it is not free of sequential correlations on successive calls. They fall into planes. If k random numbers at a time are used to plot points in k-dimensional space, then the points lie on k-1 dimensional planes rather than filling up all k-space. There are at most m1/k planes, or fewer (which is even worse) if a, c, or M are not well chosen. If you take k numbers together, the correlation can be as high as $2^{-M/k}$.

- Multiplicative Congruential Method
  - Looking at the above linear method: when c=0 it is called multiplicative congruential method
- Combined Linear Congruential Generators
  - A common trick in designing random number generators is to combine several not especially good random number generator. An example is the Wichman-Hill generator which combines three linear congruential generators.
  - Could also combine two or more multiplicative congruential generators in such a way to produce a generator with good statistical properties
- Random Number Streams

- The seed for a linear congruential random number generator is the integer value X0 that initializes the sequence
- Any value in the sequence can be used as the seed
- Random Number Streams refers to a starting Seed from the sequence X0, X1, X2...
- If the streams are b values apart, then stream i could be defined by the starting seed:

$$S_i = X_{b(i-1)} for \quad i = 1, 2... \frac{P}{b}$$

$$Older : b = 10^5$$

$$Newer : b = 10^{37}$$

- A single random-number generator with k streams can act like k distinct virtual random-number generators

6 Advantages, Disadvantages of PRNG

- Advantages
  - Compared to True Random Number Generators, Pseudo-Random Number Generators can be generated with very fast calculations. They are easier to debug and test due to its cyclic nature, it requires low memory and no external hardware is required. Lastly, this makes Pseudo-Random Number Generators cost efficient and scalable.
- Disadvantages
  - However, Pseudo-Random Number Generators are of course not truly random. Correlation among sequential random numbers may be very high and cycles will repeat. Thus, pseudo-random generators are crack-able by brute force or other methods.

7 True Random Number Generation

This is also known as Hardware Random Number Generation as external hardware is needed to extract random numbers from a physical phenomena, rather than a deterministic algorithm on a computer. Such devices use atmospheric noise such as thermal noise and other stochastic natural processes. A True random number generator uses a transducer to convert aspects of physical phenomena to a signal, then uses an amplifier to increase the amplitude of the random fluctuations to a measurable level. An analog to digital converter is used to convert the output into a binary digit 0 or 1. A true series of random numbers are attained after repeated sampling. The use of natural entropy such

as nuclear decay and clock drift is used as well in True random number generation. The true randomness property can be justified by chaos theory, but we will not go into specific details. In the generation of True random numbers outputting binary digits, bias must also be taken into consideration.

8 Pseudo Random Numbers vs. True random numbers

Now that we know some methods of generating pseudo and random numbers, we can detail some of the differences between the 2. A comparison of these 2 may provide some insight on why sometimes compromises are made and Pseudo Random Number Generators suffice, such as in casino gaming and online gambling sites.

Pseudo Random Number Generators cannot truly recreate random events such a dice rolls. Pseudo Random Number Generators are algorithms that utilize mathematical formulas to produce sequences that will appear random, or at least have the effect of randomness.

If the results of a Pseudo Random Number Generator mimicking dice rolls are listed it will appear random. However, statistical analysis will prove that the numbers are not really random, but actually predetermined. Thus the results can be measured and standardized, and overall controlled.

True Random Number Generators behave differently since the results are truly unpredictable. If a computer tries to produce a real random sequence of numbers, it must base its numbers on a naturally occurring physical phenomenon. This can include radioactive decay of isotopes, static in airwaves, or the waves of the ocean. Clearly True Random Number Generators are not cost efficient when compared to Pseudo Random Number Generators. Additionally, True Random Number Generators are subject to wear and tear since naturally occurring phenomenons are subject to entropy.

Another efficiency of Pseudo Random Number Generators is that the same sequence of numbers can be reproduced if the starting point of the sequence is known. This allows for better facilitation in real life use and applications - such as casino control boards. The probability of this happening naturally would take extremely long unless the actual algorithm is acquired.

Because of these reasons, in many experiments or games that require randomness, such as casinos, Pseudo Random Number Generators are the go to because of its cost efficiency and periodicity.

9 Testing of Random Number Generators

Clearly there are many different methods of generating random numbers. However, not all sources of random numbers behave and some are better than others in different applications. Plenty of methods have been created to test Random Number Generator's and the sequences they create. There are 2 distinct groups

of tests: *Empirical* and *Theoretical*. *Empirical* tests are conducted on the sequences generated by Random Number Generators, and do not require knowledge on the how the sequence is produced. *Theoretical* tests require knowledge of the structure of the Random Number Generator but do not necessarily require the sequence generated. Theoretical tests are better when available. This paper will focus only on empirical tests. However, before we go into empirical tests, we will go over two major tests that lay down the foundations for empirical tests: the $chi - square$ test and $Kolmogorov - Smirnov$ test.

9.1 The Chi-Square Test

The chi-square test (χ^2) was published in 1900 by Karl Pearson. The chi-square test can be used in many situations, and approximates probability as to how likely a given outcome is. Suppose there are n independent observations, each in one of k categories. Let $\mathrm{Y_s}$ be the number of observations falling into the sth category and $\mathrm{p_s}$ be the probability that an observation falls into category s. Then, for large values of n, we would expect that

$$Y_s \approx np_s$$

In order to measure "how far away" we are from these expected values, we define a reasonable statistic V_i as the following

$$V_i = (Y_1 - np_1)^2 + (Y_2 - np_2)^2 + ... + (Y_k - np_k)^2$$

This will give some way to measure how close the actual results are to the expected. V_i gives equal weight to each category. If not every p_s is the same, some discrepancies can be overemphasized or hidden. Thus, we must modify the statistic to V (which is the one that is actually used, and is called the chi-square statistic). All we need to do here is divide each ith term by np_i. After this, V can be written as

$$\sum_{1 \leq s \leq k} \frac{(Y_s - np_s)^2}{np_s}$$

or, if rearranged a bit,

$$\frac{1}{n} \sum_{1 \leq s \leq k} \frac{Y_s^2}{p_s} - n$$

Now, what is a reasonable value for V? We can utilize a table that shows the selected percentage points of the Chi-Square Distribution

Selected Percentage Points Of The Chi-Square Distribution

	$p = .01$	$p = .05$	$p = .25$	$p = .50$	$p = .75$	$p = .95$	$p = .99$
$\nu = 1$	0.00016	0.00393	0.1015	0.4549	1.323	3.841	6.635
$\nu = 2$	0.02010	0.1026	0.5753	1.386	2.773	5.991	9.210
$\nu = 3$	0.1148	0.3518	1.213	2.366	4.108	7.815	11.34
$\nu = 4$	0.2971	0.7107	1.923	3.357	5.385	9.488	13.28
$\nu = 5$	0.5543	1.1455	2.675	4.351	6.626	11.07	15.09
$\nu = 6$	0.8720	1.635	3.455	5.348	7.841	12.59	16.81
$\nu = 7$	1.239	2.167	4.255	6.346	9.037	14.07	18.48
$\nu = 8$	1.646	2.733	5.071	7.344	10.22	15.51	20.09
$\nu = 9$	2.088	3.325	5.899	8.343	11.39	16.92	21.67
$\nu = 10$	2.558	3.940	6.737	9.342	12.55	18.31	23.21
$\nu = 11$	3.053	4.575	7.584	10.34	13.70	19.68	24.73
$\nu = 12$	3.571	5.226	8.438	11.34	14.84	21.03	26.22
$\nu = 15$	5.229	7.261	11.04	14.34	18.25	25.00	30.58
$\nu = 20$	8.260	10.85	15.45	19.34	23.83	31.41	37.57
$\nu = 30$	14.95	18.49	24.48	29.34	34.80	43.77	50.89
$\nu = 50$	29.71	34.76	42.94	49.33	56.33	67.50	76.15

To use the table, look at the line with v = k - 1, then compare V to the contents of that row. Imagine if k = 9, then v would be 8. p = 0.99 entry is 20.09, meaning that V < 20.09 around 99% of the time. Any value of V too far above 20.09 would be suspiciously high in this scenario. We look at the (k-1)th row for the following reason: $Y_1, Y_2, ..., Y_k$ can be seen as independent Poisson random variable because they sum up to n. If $Y_1, Y_2, ...Y_{k-1}$ are known, Y_k can be calculated, as k - 1 of the Y values are independent. The correct choice for n to use in each situation is not clear - a large n can detect global nonrandomness, but might smooth out local nonrandomness. The should be performed multiple times on the same sequence, so a different value of n could be used for different tests to increase accuracy. One method of interpreting the results of a chi-square test:
V <1% entry or V >99% entry is considered "Nonrandom"
$1\% \leq V \leq 5\%$ entries or $5\% \leq V \leq 99\%$ entries are considered "suspect"
$5\% \leq V \leq 10\%$ entries or $90\% \leq V \leq 95\%$ entries are "almost suspect"
If two or more tests are "suspect" for a sequence, it is not sufficiently random. The chi-square test is a foundation of many empirical tests, and is probably the most-used test for Random Number Generators.

9.2 The Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (KS) test originated in 1933 by Kolmogorov, and was improved in 1939 by Smirnov, leading to the joint name of the test. This test is useful in areas where chi-square is not, but can also be used together with chi-square. First a very common function in probability ehtory must be introduced: The cumulative distribution function (cdf).
Given a random variable X, the cdf $F_x(x)$ is:

$$F_x(x) = probabilitythat(X \leq x)$$

This probability has a range of [0,1] and will always be increasing as x goes from $-\infty$ to $+\infty$ X takes on the values of the sequence generated by a Random Number Generator. For the KS test, $F_x(x)$ must be continuous. It deals with a different situation, where the numbers generated by a Random Number generator are allowed to take on any value with in a certain interval, leading to a continuous cdf. This $F_x(x)$ is the theoretical distribution we would expect the Random Number Generator to have.
If n independent observations of X are made, giving the values $X_1, X_2, ..., X_n$. The empirical distribution function $F_n(x)$ is defined as

$$F_n(x) = \frac{numberof X_1, X_2, ..., X_n thatare \leq x}{n}$$

The KS test compares $F_x(x) to F_n(x)$ by measuring the difference between the 2 distribution functions. When n is sufficiently large, we would expect the 2 functions to be similar if the sequence we're examining is truly random. Therefore the difference is measured with the following statistics:

$$K_n^+ = \sqrt{n} max(F_n(x) - F_x(x)), -\infty < x < +\infty$$

$$K_n^- = \sqrt{n} max(F_x(x) - F_n(x)), -\infty < x < +\infty$$

K_n^+ is the greatest deviation where F_n is greater than F_x, and K_n^- is the greatest deviation where F_n is less than F_x. We compare these statistics to the following table in a similar manner to the chi-square test

Selected Percentage Points Of The Distributions Of K_n^+ And K_n^-

	$p = .01$	$p = .05$	$p = .25$	$p = .50$	$p = .75$	$p = .95$	$p = .99$
$n = 1$	0.01000	0.05000	0.2500	0.5000	0.7500	0.9500	0.9900
$n = 2$	0.01400	0.06749	0.2929	0.5176	0.7071	1.0980	1.2728
$n = 3$	0.01699	0.07919	0.3112	0.5147	0.7539	1.1017	1.3589
$n = 4$	0.01943	0.08789	0.3202	0.5110	0.7642	1.1304	1.3777
$n = 5$	0.02152	0.09471	0.3249	0.5245	0.7674	1.1392	1.4024
$n = 6$	0.02336	0.1002	0.3272	0.5319	0.7703	1.1463	1.4144
$n = 7$	0.02501	0.1048	0.3280	0.5364	0.7755	1.1537	1.4246
$n = 8$	0.02650	0.1086	0.3280	0.5392	0.7797	1.1586	1.4327
$n = 9$	0.02786	0.1119	0.3274	0.5411	0.7825	1.1624	1.4388
$n = 10$	0.02912	0.1147	0.3297	0.5426	0.7845	1.1658	1.4440
$n = 11$	0.03028	0.1172	0.3330	0.5439	0.7863	1.1688	1.4484
$n = 12$	0.03137	0.1193	0.3357	0.5453	0.7880	1.1714	1.4521
$n = 15$	0.03424	0.1244	0.3412	0.5500	0.7926	1.1773	1.4606
$n = 20$	0.03807	0.1298	0.3461	0.5547	0.7975	1.1839	1.4698
$n = 30$	0.04354	0.1351	0.3509	0.5605	0.8036	1.1916	1.4801

The table is read the same way as the chi-square, and like the chi-square n needs to be selected carefully. n should be big enough so that we can detect if the distribution function $F_n(x)$ and $F_x(x)$ are significantly different. Again, n being too large will usually smooth out local nonrandomness.
KS can be used to create a better procedure of describing a sequence. We can do this by making m independent (χ^2) tests on different parts of a random

sequence and record the values $V_1, V_2, ..., V_m$. Then apply a KS test to these V_i's. F_m is described by the plotted values of each V_i, and F_v can be found from a chi-square distribution

9.3 Empirical Tests

We will now go over some empirical tests used to determine the "randomness" of a sequence. Empirical tests are performed on a sequence produced by a Random Number Generator, and do not require knowledge on how the Random Number Generator works. [U_n] denotes the sequence U_0, U_1, U_2, ... of real numbers between 0 and 1. The integers should be independently and uniformly throughout the interval. [Y_n] refers to the sequence of integers Y_0, Y_1, Y_2, .. of integers between 0 and d - 1 where d is an integer, with the rule

$$Y_i = \lfloor dU_i \rfloor$$

This preserves the same properties of being independent and uniformly distributed, between 0 and d-1. The rest of this section will go over some examples of Empirical tests, and how they are used to determine the "randomness" of a sequence.

1. *FrequencyTest* The Frequency test is the simplest of the empirical tests. Given a Sequence

   $$\lfloor U_n \rfloor$$

   require the elements to be uniformly distributed between 0 and 1. Apply the Kolmogorov-Smirnov (KS) test with

   $$F(x) = x for 0 \leq x \leq 1$$

   Now we can have categories to use in a chi-square test. For every integer r where

   $$0 \leq r < d$$

   count the number of times that Y_i = r for

   $$0 \leq i < n$$

   Each integer that fulfills this r defines a chi-square category, where we can now use the chi-square test with k = d (There are d integers between 0 and d - 1) and $p_r = 1/d$ (To make sure it is uniformly distributed). This says: for a sequence of 0s and 1s generated by a Random Number Generator, we would expect to get about the same number of 0s and 1s if we were to take a sample of the sequence.

2. *RunsTest* The Runs Test requires another definition before it can be described. A *monotonesequence* is a sequence who's elements are either

all increasing or all decreasing. We test a sequence for its "runs up" and its "runs down" - examine the lengths of the sequence's monotone sub-sequences. For example, the sequence 1,3,8,7,5,2,6,7,1,6 divided into "runs up" would look like 1,3,8—7—5—2,6,7—1,6 This would yield a run of length 3, then 2 run lengths of 1, another run length of 3, and lastly a run length of 2. A noticeable pattern is that long runs tend to be followed by short runs, which in turn tend to be followed by long runs. This means that consecutive observations are not independent from each other, invalidating the use of a chi-square test. Instead, a more complicated statistic is used to perform the test, but this is the basic idea of the runs test.

3. *GapTest* The Gap Test looks at each U_j in a certain range and examines the length of the "gap" between this element and the next element to fall in that range. is α and β are 2 real numbers such that

$$0 \leq \alpha > \beta \leq 1$$

we would look for the length of consecutive subsequences $U_j, U_{j+1}, ..., U_{j+r}, U_{j+(r+1)}$ such that U_j and $U_{j+(r+1)}$ are between α and β but the other elements in the subsequence are not. Use the different lengths of the caps as the categories for a (χ^2) test, and the probabilities as: $p_0 = p, p_1 = p(1-p), p_2 = p(1-p)^2, ..., pk = p(1-p)^k, ...$ Here p = $\beta - \alpha$, and is the probability that any element U_j is between α and β. An algorithm can be utilized to record the lengths of consecutive gaps, but will not be described here.

4. *PokerTest* The poker test examines n groups of five consecutive integers and put them each of these groups into one of the following categories: All different: abcde One pair: aabcd Two pairs: aabbc Three of a kind: aaabc Full house: aaabb Four of a kind: aaaab Five of a kind: aaaaa A (χ^2) is applied to these 7 categories. Categories that are less likely can be grouped together to meet the "five in each category" requirement. If categories are combined, their respective probabilities would be added together. The groups of integers cannot overlap in order to preserve the independence required by the (χ^2) test.

10 Applications for Random Numbers

We now know some methods of generating random numbers, and how to tell if a random number generator is "good" or not, but an unanswered question might be why we need random number generators at all. This section will list some of the applications of random number generators, and how they are useful in math, science, and everyday life.

1. *NumericalAnalysis* - Random Number Generators can be used to help solve many problems that are either too difficult to solve, or too difficult to solve within a reasonable time. There are techniques that can approximate

the problem by relying on random numbers. An example of a problem in this case are the Monte Carlo Methods.

2. *Statistics* - In statistical sampling, studies usually involve a large number of samples, and being able to add randomness to the collection can improve the accuracy of the statistical tests.

3. *Simulations* - In any some sort of simulation, randomness is always needed to make the model as real as possible. Some examples are traffic simulations, economics, game simulations, physics simulations, and many more. Pseudo Random Number Generators provide a large benefit here, as the sequence can be set so it starts at the same place in the sequence each time. This allows for more control in the simulation, and the ability to observe the effects of changing only certain parameters while keeping the "randomness".

4. *ComputerPrograms* - Similar to numerical analysis, many computer algorithms and programs require a random number or random sequence. Computer Algorithms can also be tested with by using random inputs.

5. *Recreation* - Random numbers have been utilized for a variety of recreational purposes. Many computer games include some degree of randomness. One of the most successful and widespread use of random numbers in recreation is gambling - There are card games based on randomness and probability, and slot machines at casinos utilize Random Number Generators to operate.

6. *Cryptography* - Cryptography systems use extremely large amounts of random data. RSA, for example, requires the use of large random primes for its security. Two factor authentication will often send users a one time code that is randomly generated to verify the user's identity. One time pads require a long stream of random integers to serve as a key that is the same size or longer than the message being sent.

11 Conclusion

In this tutorial we described what Random Number Generators are, what properties they hold. We went over some different methods of generating random numbers. We looked at Pseudo Random Number Generators and True random number Generators, and what some of the pros and cons of each are. Pseudo Random Number Generators have had widespread adoption in the recreation and gaming worlds, but may not be the best choice in other situations. Testing methods to test the quality of sequences and Random Number Generators were described, with their meanings explained. Lastly, we went over some applications of Random Number Generators and how they are used in the science, mathemetical, and recreational worlds today.

12 References

Dutang, Christophe, Diethelm Wuertz. A note on random number generation. September 2009. Print.

Kurlberg, Par, Carl Pomerance. On The Period of the Linear Congruential and Power Generators. May 2004. Print.

P. L'Ecuyer, Random Number Generation, in Handbook of Computational Statistics, Gentle, Haerdle, Morita (eds.), Springer, 2004

Trimbitas, Radu. Random Number Generation: Nuts and Bolts of Simulation. 2011.

Biebighauser, Dan. Testing Random Number Generators. 2000