Randomness: Fact or Fiction?

S-STEM Research Group

Dept of Mathematical Sciences

Appalachian State University
Boone, North Carolina

1. Overview

S-STEM Research Group

Aneisy Cardo, Nadine Lambert, Lauren Murray, and Tyrel Winebarger.

Special Thanks

Dr. Rahman Tashakkori, Dr. Vicky Klima, NSF, the S-STEM Program, the McKinney Scholars Program, the Department of Computer Science, and the Department of Mathematics.

Abstract

Everyday our society is moving towards a more face-paced lifestyle, structured by schedules. Given such a large population of off-campus students, transportation systems are critical for timely arrivals to campus. We will utilize tropical linear algebra in a max-plus system to analyze the efficiency of Appalachian States AppalCart system. When analyzing, we will assume constant travel times, turnover of passengers, and prompt departures following the turnovers.

2. Linear Congruential Generator

The linear congruential generator is a type of pseudo-random number generator, which is easily implemented and fast. It is calculated using the algorithm:

$$X_{n+1} = (aX_n + c) mod \text{ m, where}$$

- X is an initial value
- Xn+1 is the value that you determine
- Xn is the previous value
- m is the modulus
- a is the multiplier
- c is the increment

3. Java Parameters

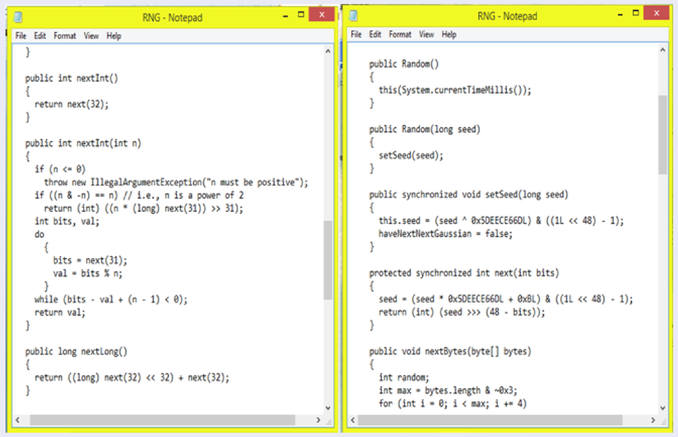

Figure: Java Code for the Random Class

$a = 25214903917, c = 11,$ and $m = 2^{48}$

These are the parameters that Java uses. We assume the program would choose parameters that result in good randomness; thus we will suppose that numbers generated with these coefficients will be sufficiently random. Next, we generate two arrays of random numbers, import them into R® statistical software, and create histograms to visually represent the frequency distribution of each array.

4. Frequency Distributions

The first histogram models the distribution of 1,000 numbers between 0 and 1, while the second histogram models the distribution of 100,000 numbers. Notice that the larger sample appears more uniform than the smaller sample, as we would expect given a good quality of randomness.

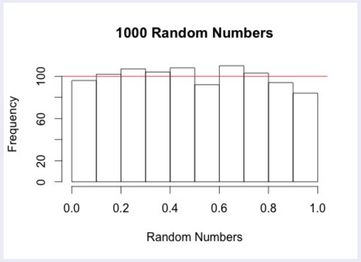

Figure: 1,000 Random Numbers

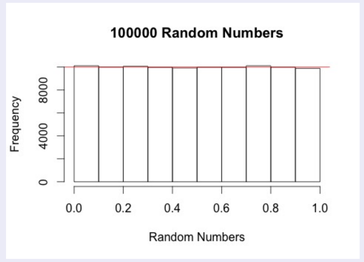

Figure: 100,000 Random Numbers

5. Changing Parameters

To see what happens when we use parameters other than those specified in the Java code, we changed the previous a value to a+1 and then generated an array of 2,000 numbers.This histogram shows a loop around 0.3; we deduce that the numbers are non-random based on the histogram.

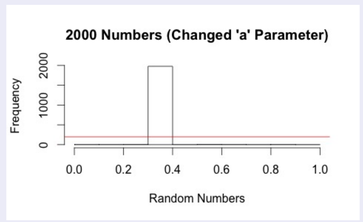

Figure: 2,000 Numbers Caught in a Loop

6. Does Distribution Determine Randomness?

Below is an example of a uniform distribution. Notice that the list of numbers is a repetitive cycle, implying poor randomness. This shows that a uniform distribution does not necessarily guarantee randomness.

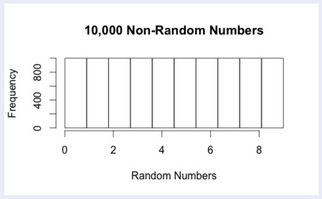

Figure: 10,000 Numbers

```
1.02.03.04.05.06.07.08.09.00.0
1.02.03.04.05.06.07.08.09.00.0
1.02.03.04.05.06.07.08.09.00.0
1.02.03.04.05.06.07.08.09.00.0
```

Figure: Clearly non-random numbers

7. Conclusion

We have shown that

Sufficient Randomness	$\Longrightarrow$	Uniform Distribution
Modal Distribution	$\Longrightarrow$	Poor Randomness
Uniform Distribution	$\not\Longrightarrow$	Sufficient Randomness.

It is clear that distribution alone is an insufficient test for randomness. There exist sophisticated and complex tests, and each one uses distribution as an indicator of sufficient randomness; however distribution alone can lead to incorrect assumptions, and must be paired with several other methods of detection. Even still, randomness is difficult to quantify; there is still much debate about the validity of the methods used for testing randomness, which begs the question: without efficient detection, will we ever know for sure if perfect randomness can be achieved, or even exist?