# Computer Science Extended Essay

**Topic: Comparing the entropies of integers produced by true-random generators against integers produced by pseudo-random generators**

**RQ: Under what circumstances are pseudo-random generators a viable replacement for true-random generators?**

**Personal Code: hgh332**

**Word Count: 3890 Words**

# Table of Contents

# Introduction

Computers are integral to virtually every aspect of modern society. Having grown up in an era where technology rapidly evolved, my early exposure to mobile devices, starting with an HP iPAQ and later the iPhone, sparked a deep fascination with how computers work and the systems behind them. The first 'mobile' device I encountered was my aunt's HP iPAQ. Compared to the mobile devices of today, the iPAQ, with its capacitive screen and a stylus that needed to be recalibrated every few minutes, was a nightmare to use. Even so, the idea of carrying a whole computer in your pocket was enough to blow my mind. A few years later when I saw the first iPhone, I was lost for words at the prospect of just touching things with your fingers to interact with them, no stylus necessary! This, combined with iPhone's relatively high-resolution display, sparked the flame that would become my passion for technology.

Since I was so utterly fascinated with computers, mobile computers, in particular, I naturally wanted to find out how they worked. After years of learning how different aspects of computers worked, I noticed a pattern. Almost every aspect of computer software had some sort of relation or even dependence on a randomized algorithm. What puzzled me the most, however, was when I learnt that computers are actually physically incapable of generating a random value on their own and that all 'random' numbers generated by computers are actually just the product of an extremely complicated mathematical algorithm. While this technique works reasonably well for many applications, it is a nightmare for many people, like cyber-security experts. This is because several cryptographic techniques, like encryption, rely on these random numbers to secure data. In theory, if the 'random' numbers were generated through

the use of a mathematical algorithm, an attacker can technically just generate ‘random’ numbers of their own and bypass any security intended to thwart them.

Fortunately, there is a solution to this. Instead of simply putting values into an algorithm to be ‘randomized’, true random generators use variables like atmospheric noise as their input and then put those values through some sort of complex algorithm for good measure. Since the input variables are in fact completely random, the values produced through these generators are also completely random. These techniques are absolutely essential in applications like encryption and hashing, where pseudo-random algorithms are just not a viable solution.

My chosen Research Question will attempt to explore just how much the perceived and the actual ‘randomness’ of true-random algorithms compares to the randomness of pseudo-random algorithms. I will also attempt to explore applications where some pseudo-random algorithms can actually be a viable replacement for true-random algorithms, as well as comparing the average efficiency of all the algorithms.

# Background Information

## Pseudo-Random Number Generators

A Pseudo-Random Number Generator (PRNG) generates integers through a mathematical formula that requires only a seed value, which serves as the starting point for the sequence[1]. Unlike true random generators, PRNGs rely solely on algorithms rather than external inputs, such as physical phenomena. Many implementations of PRNGs do not even require a seed value as an input and can sometimes have a predetermined seed value already set. Most implementations of PRNGs also give the user the option to specify a range or limit for which values can be generated. If a range or lower bound is explicitly specified by the user, the PRNG will use that value as the seed value. However, many modern PRNGs often have a different mathematical formula for the seed value, using the user-inputted value or predetermined value to generate a completely different value as the seed value. This increases the overall entropy and security of the algorithm even more[2].

---

[1] 76, E., 2018. *Pseudo Random Number Generator (PRNG) - Geeksforgeeks*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/#:~:text=Pseudo%20Random%20Number%20Generator(PRNG)%20refers%20to%20an%20algorithm%20that,state%20using%20a%20seed%20state> [Accessed 23 June 2020].

[2] Owen, A., 2013. *Monte Carlo Theory, Methods And Examples*. 1st ed. Stanford: Stanford University, pp.Chapter 3.1.

There are a few different algorithms that a PRNG can utilize to generate a series of random integers. One of the older, more commonly used algorithms is known as the 'Linear congruential generator' or LCG for short. The fundamental equation that all varieties of the LCG algorithm are based on is shown here;

$$X_{n+1} = (aX_n + c) \mod m$$ [3]

where $X$ is the sequence of pseudorandom values, and

$m$, $0 < m$ — the "modulus"
$a$, $0 < a < m$ — the "multiplier"
$c$, $0 \leq c < m$ — the "increment"
$X_0$, $0 \leq X_0 < m$ — the "seed" or "start value"

In essence, the algorithm works like this:

1. The algorithm starts with a seed value ($X_0$).
2. The seed value is then multiplied with the multiplier value (a).
3. This value is then added to the increment value (c).
4. The final result is then calculated using the modulus value (m).
5. This result then becomes $X_1$. This is repeated n amount of times.

[3] En.wikipedia.org. n.d. *Linear Congruential Generator*. [online] Available at: <https://en.wikipedia.org/wiki/Linear_congruential_generator> [Accessed 23 June 2020].

Worked Example:

1. $X_0 = 2, a = 3, c = 5, m = 3$
2. $(3 * 2 + 5) = 11$
3. $11 \; mod \; 3 = 2$

   *Since...*

4. $3 * 3 = 9$

   *And...*

5. $11 - 9 = \mathbf{2} \; (X_1)$

*It is important to note that the values 'a', 'c' and 'm' are constants and never change, unlike $X_n$ and $X_{n+1}$.*

More advanced and secure implementations of the LCG algorithm often use seemingly 'random' variables like the time or the CPU's current clock speed to derive a seed value, multiplier, or increment. Although this works well for many scenarios, it is not suitable for many others since many of the variables the algorithm uses to derive its constant values can easily be predicted or calculated. Generally speaking, prime numbers are a good choice for the modulus, since they usually increase the entropy and the range of numbers produced through the algorithm[4]. Larger values for the modulus, multiplier, and increment also tend to produce a much more uniformly distributed set of integers.

---

[4] Woo, E., 2014. *Random Numbers (2 Of 2: Linear Congruential Generator)*. [image] Available at: <https://www.youtube.com/watch?v=PtEivGPxwAI> [Accessed 23 June 2020].

Unfortunately, while making use of these techniques can certainly increase the entropy of integers produced, it is certainly not the most efficient way of producing a set of integers. The most efficient LCG algorithm would use a value of just 2 for the modulus. The reason this is so efficient is that there are only two possible values using this. The two values are 0 and 1. Predictably, however, since this method only produces two distinct values, it is never a viable option for any real-life applications, and any security system that incorporated this algorithm would be childishly easy to break, even by someone with very little experience.

Although this algorithm is quite commonly used, it is not the only algorithm that is used for modern-day applications. Another example of an algorithm that is quite popular is known as the 'Multiplicative congruential generator', abbreviated to MCG. The reason that MCG algorithms are not nearly as widely used as LCG algorithms is that they are almost exactly the same, with one key difference. This is the general formula for any MCG algorithm[5];

$$X_{k+1} = a \cdot X_k \mod m$$

In case it isn't immediately obvious, this formula is exactly the same as the general formula for LCG algorithms, without the inclusion of the '*+c'* variable (the increment). The main restriction that this creates is that $X_0$, the initial seed, and the modulus (*m)* must be coprime to each other. This means that the only common factor the two numbers can have is 1.

---

[5] En.wikipedia.org. n.d. *Lehmer Random Number Generator*. [online] Available at: <https://en.wikipedia.org/wiki/Lehmer_random_number_generator> [Accessed 23 June 2020].

This restrictive nature of MCG algorithms is why many software engineers and developers choose to opt for an LCG algorithm instead. However, MCG algorithms have the added benefit of being more efficient, which makes them a viable solution for many real-life applications and projects that prioritize efficiency and runtime-speed over flexibility.

## True-Random Number Generators

Unlike PRNGs, TRNGs (True-Random Number Generators) do not rely on any specific mathematical formulas or functions to generate integers. Instead, most TRNGs rely on physical phenomena that are known to be random and highly volatile, such as atmospheric noise, thermal noise, electromagnetic phenomena, and even certain quantum phenomena, to generate a series of truly random integers. This is the reason that conventional computers cannot generate truly random numbers without any external hardware. Computer CPUs are designed to process ones and zeros, the entire process is extremely logical and largely dependent on mathematical formulas and equations, there is no way for a computer to pull a number out of thin air.

Fortunately, there are certain peripherals and modifications that can be added to almost any computer to generate true-random numbers. One such device is known as the 'TrueRNG - USB Hardware Random Number Generator'[6]. This device is relatively small and compact, it interfaces through a USB CDC Serial Virtual port and can be connected to any PC running a relatively modern version of Windows, macOS, or even any Linux distribution.

---

[6] Amazon.com. 2016. *Truerng V3 - USB Hardware Random Number Generator*. [online] Available at: <https://www.amazon.com/TrueRNG-V3-Hardware-Random-Generator/dp/B01KR2JHTA> [Accessed 23 June 2020].

Two other similar products are the 'BitBabbler'[7] and the 'OneRNG', which also interface through a USB CDC Serial Virtual port and can also be connected to most versions of Windows, macOS, and several popular Linux distributions, like Ubuntu or Debian. If hardware generators are not a viable solution, there are a variety of TRNG services available online. Some demand a fee, but many of them are free to use. These services work by hosting a server, or several different servers, on a publicly accessible website. These servers are often connected to a variety of different scientific instruments that listen for physical phenomena like the ones mentioned earlier. One of the most popular and well-renowned services is hosted at a website called 'random.org'. The website states that its servers generate random numbers by listening to atmospheric noise and then translating those values into random integers using their own algorithm. The website states that the service is ideal for anyone who wants to host some sort of lottery or sweepstakes, but needs a more efficient method than simply rolling dice or picking a number out of a hat. One of two other well-known services is called the 'ANU Quantum Random Numbers Server' which generates integers by measuring quantum fluctuations in a vacuum, the website was created by researchers at the Australian National University. The other service is called 'HotBits', which generates random integers by measuring radioactive decay.

---

[7] Bitbabbler.org. 2014. *A Hardware RNG We Could Trust - Bitbabbler*. [online] Available at: <http://www.bitbabbler.org/> [Accessed 23 June 2020].

[8] Cheetam, J., 2019. *Onerng - Hardware Random Number Generator*. [online] Onerng.info. Available at: <https://onerng.info/> [Accessed 23 June 2020].

## Comparison

[9]

| **PRNGs** | **TRNGs** |
|---|---|
| Integers produced always create a periodic sequence, which means they repeat | Sequences never intentionally repeat, there is never an actual pattern, and therefore no period |
| Can neglect some integers, due to limitations of the modulus or seed number. Prime modulus values should be chosen to offset this. | Doesn't need a modulus value, any value can be generated in theory. |
| Choosing prime numbers for the modulus usually increases the overall entropy of the integers produced. | Doesn't need a modulus, so entropy is not affected. |
| It is deterministic, which means that all values depend on the previous value. | Consecutive values have no relation. |
| Generally, much more efficient than TRNGs. | Less efficient due to the inclusion of a physical variable rather than a mathematical function. |

[9] Woo, E., 2014. *Random Numbers (1 Of 2: True Vs. Pseudo Rngs)*. [video] Available at: <https://www.youtube.com/watch?v=fEWigU1dcp8> [Accessed 23 June 2020].

## Hypothesis

The theory and purpose of both PRNGs and TRNGs have been explored to a reasonable extent in the previous section. This section will aim to predict how the data will answer the primary research question. By looking at the 'Comparison' subsection of the research, it is quite clear that TRNGs appear to produce sets of integers with much higher overall entropies. In addition to this, TRNGs produce sequences that are not deterministic. This means that the values TRNGs produce have no relation to each other and do not depend on each other, this is in contrast to PRNGs which can only produce deterministic sequences. The research also implies that the modulus value (m) and the seed value ($X_0$) are vital in determining the overall entropy of the sequence of integers produced by the PRNG. Fortunately, however, PRNGs are almost always more efficient than their TRNG counterparts. This is due to the fact that PRNGs rely almost solely on mathematical algorithms and functions to produce integers, while TRNGs are forced to rely on hardware-based methods and measuring physical phenomena.

The research would seem to suggest that PRNGs are vastly inferior to TRNGs in terms of the entropy of the integers produced and the uniformity of their distribution. The only practical advantages PRNGs seem to have are efficiency and consequently, cost and convenience. This would mean that PRNGs should never be considered as a viable replacement for TRNGs, under almost any circumstance. However, it is my personal opinion that the improved efficiency, convenience, and lower cost of PRNGs far outweigh their lower entropy. I say this because I think it is possible to significantly increase the entropy of most PRNG algorithms by simply selecting more effective seed and modulus values. This could increase the entropy of the values

to a point where they can be considered random enough for most use cases. Of course, there will always be some scenarios where TRNGs are absolutely necessary, however, I think the data will show that PRNGs are suitable for the majority of real-life situations.

# Research Methodology

## Independent Variables

In any scientific investigation, the Independent variable is the variable that is being changed. In the case of this investigation, the main independent variable will be the random number generator being used. The investigation will explore a variety of different PRNGs and TRNGs, on different platforms and from different sources. Some of the sources were mentioned in the 'Background Information' section. The modulus, seed value, multiplier, and increment can also be considered as independent variables since they will be changed across many different PRNGs. However, changing all of those values would produce an exuberant amount of data that simply cannot be analyzed effectively for the purposes of this investigation. I will just be changing the seed values to a variety of different types of integers, like prime and square numbers, in order to probe a deeper understanding of what makes an effective PRNG.

## Dependent Variables

The dependent variable is usually the variable that is being measured. In the case of this investigation, the main dependent variable would be the security and entropy of the integers produced. The perceived entropy could be determined by graphing the integers produced by each generator. The level of security could be determined by attempting to obtain the seed

value and producing the same sequence. The accuracy of the predicted sequence would determine the overall level of security.

## Controlled Variables

| **Variable** | **Importance** | **Relevant Specifications** |
|---|---|---|
| Computer and Operating System | Using different hardware or software can have a substantial impact on the efficiency and/or the entropy of the integers produced. | **Model:** MacBook Pro (Retina, 13-inch, Early 2015)<br>**CPU:** Intel Core i5-5257U@2.7 GHz<br>**GPU:** Intel Iris Graphics 6100 1536 MB<br>**Memory:** 8 GB 1867 MHz DDR3<br>**OS:** macOS Catalina (10.15.6) |
| The browser used for the cloud-based services | Using a different browser is unlikely to affect the entropy of the integers produced, but will most definitely affect the perceived entropy of the generator. | **Browser:** Google Chrome<br>**Version Number:** 83.0.4103.116 |
| The IDEs used for the OOP based generators | Using a different IDE will almost certainly affect the runtime and compilation times, which can severely impact the perceived efficiency of the generator. | **IDE:** Xcode Version 11.7<br>**Build:** 11E801a |
| The maximum time given to each generator | In order to accurately measure the efficiency of each algorithm, the maximum time given must be exactly the same. | **Time:** 1 minute (60 seconds) |
| The same range of values | Requesting a different range of values can cause both the entropy and efficiency of a generator to vary wildly. | **Range:** 0-100 |
| Data analysis and graphing software | Different software can impact how the data is interpreted and lead to a vastly different conclusion. | **Software 1:** Microsoft Excel<br>**Version Number:** 16.0.6742.2048<br>**Software 2:** Logger Pro 3<br>**Version Number:** 3.16 |

## Procedure

1. Install all the necessary SDKs and libraries for all the different programming languages.
2. Generate a series of integers from different libraries across different programming languages, using the same hardware, operating system, and IDE.
3. Repeat with all of the other selected programming languages, use the same IDE, hardware, and operating system.
4. Repeat steps 2-3 with different seed values, for the applicable algorithms.
5. Generate a series of integers from a variety of TRNGs that can be downloaded, on custom hardware, or hosted online.
6. Graph the results using the appropriate graphing software, like Logger Pro or Excel.
7. Use the graphs to analyze and extrapolate the necessary data.
8. Attempt to brute-force the seed value and produce the same sequence each of the sequences. Graph these results.
9. Interpret a reasonable and well-supported conclusion from the processed data.
10. Discuss the possible implications.

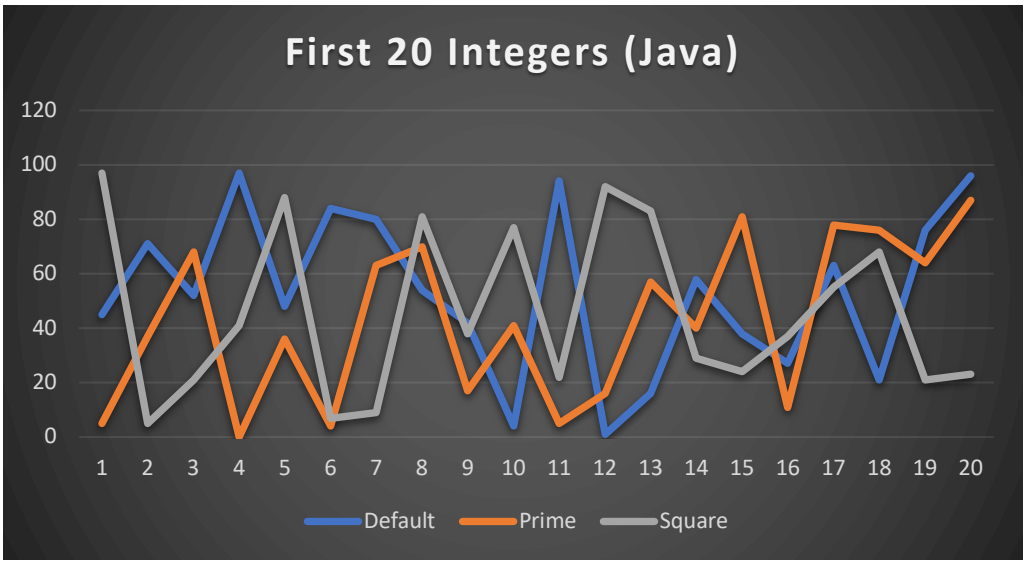
# Raw Data

## PRNGs

*(Only the first 20 Integers are shown. The complete tables and graphs are available at Appendix A)*

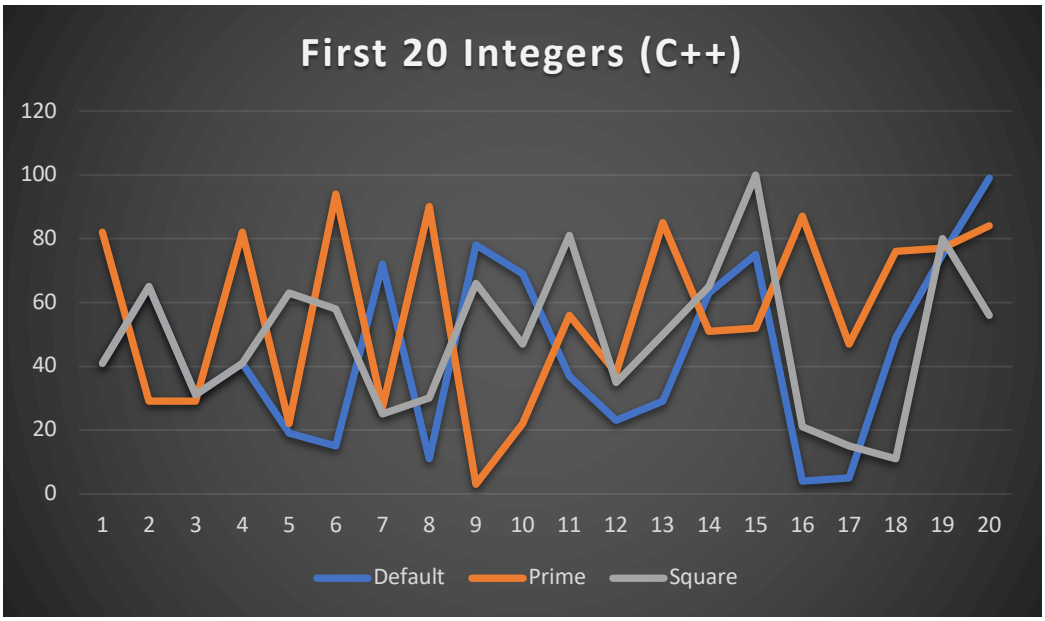
### Java (Java.util.Random)

| Default | Prime | Square |
|---|---|---|
| 45 | 5 | 97 |
| 71 | 37 | 5 |
| 52 | 68 | 21 |
| 97 | 0 | 41 |
| 48 | 36 | 88 |
| 84 | 4 | 7 |
| 80 | 63 | 9 |
| 54 | 70 | 81 |
| 42 | 17 | 38 |
| 4 | 41 | 77 |
| 94 | 5 | 22 |
| 1 | 16 | 92 |
| 16 | 57 | 83 |
| 58 | 40 | 29 |
| 38 | 81 | 24 |
| 27 | 11 | 37 |
| 63 | 78 | 55 |
| 21 | 76 | 68 |
| 76 | 64 | 21 |
| 96 | 87 | 23 |



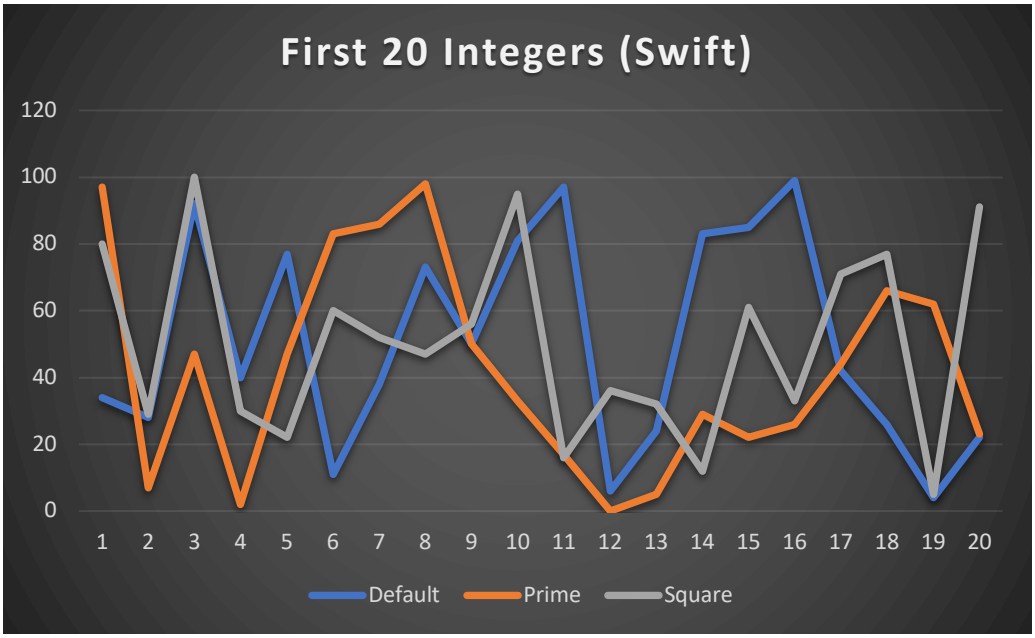
## C++ (Rand())

| Default | Prime | Square |
|---|---|---|
| 41 | 82 | 41 |
| 65 | 29 | 65 |
| 31 | 29 | 31 |
| 41 | 82 | 41 |
| 19 | 22 | 63 |
| 15 | 94 | 58 |
| 72 | 27 | 25 |
| 11 | 90 | 30 |
| 78 | 3 | 66 |
| 69 | 22 | 47 |
| 37 | 56 | 81 |
| 23 | 38 | 35 |
| 29 | 85 | 50 |
| 63 | 51 | 65 |
| 75 | 52 | 100 |
| 4 | 87 | 21 |
| 5 | 47 | 15 |
| 49 | 76 | 11 |
| 75 | 77 | 80 |
| 99 | 84 | 56 |



## Swift (GKMersenneTwister)

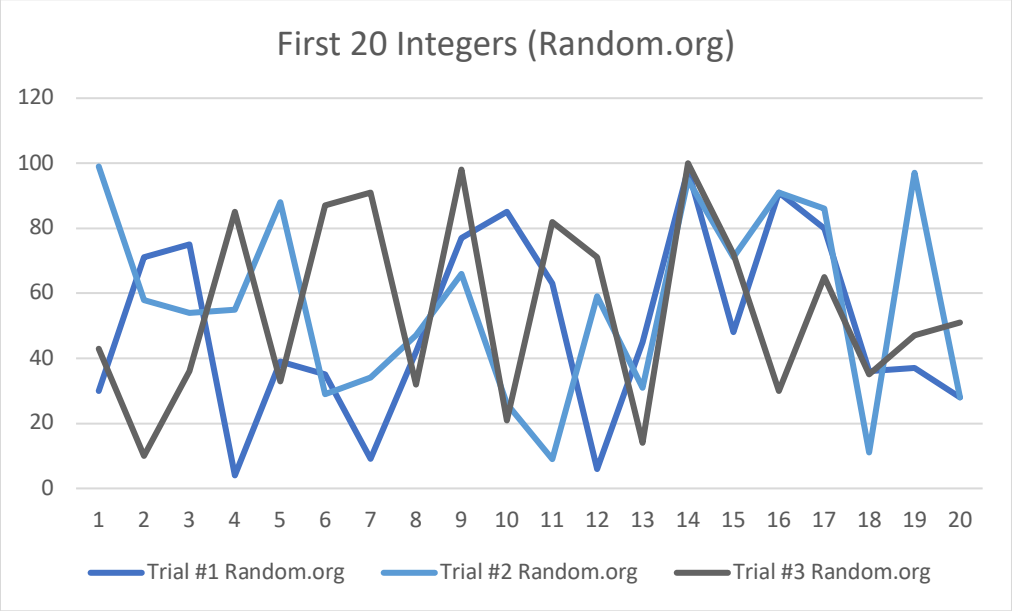
| Default | Prime | Square |
|---|---|---|
| 34 | 97 | 80 |
| 28 | 7 | 29 |
| 92 | 47 | 100 |
| 40 | 2 | 30 |
| 77 | 47 | 22 |
| 11 | 83 | 60 |
| 38 | 86 | 52 |
| 73 | 98 | 47 |
| 50 | 50 | 56 |
| 81 | 33 | 95 |
| 97 | 17 | 16 |
| 6 | 0 | 36 |
| 24 | 5 | 32 |
| 83 | 29 | 12 |
| 85 | 22 | 61 |
| 99 | 26 | 33 |
| 42 | 44 | 71 |
| 26 | 66 | 77 |
| 4 | 62 | 5 |
| 22 | 23 | 91 |



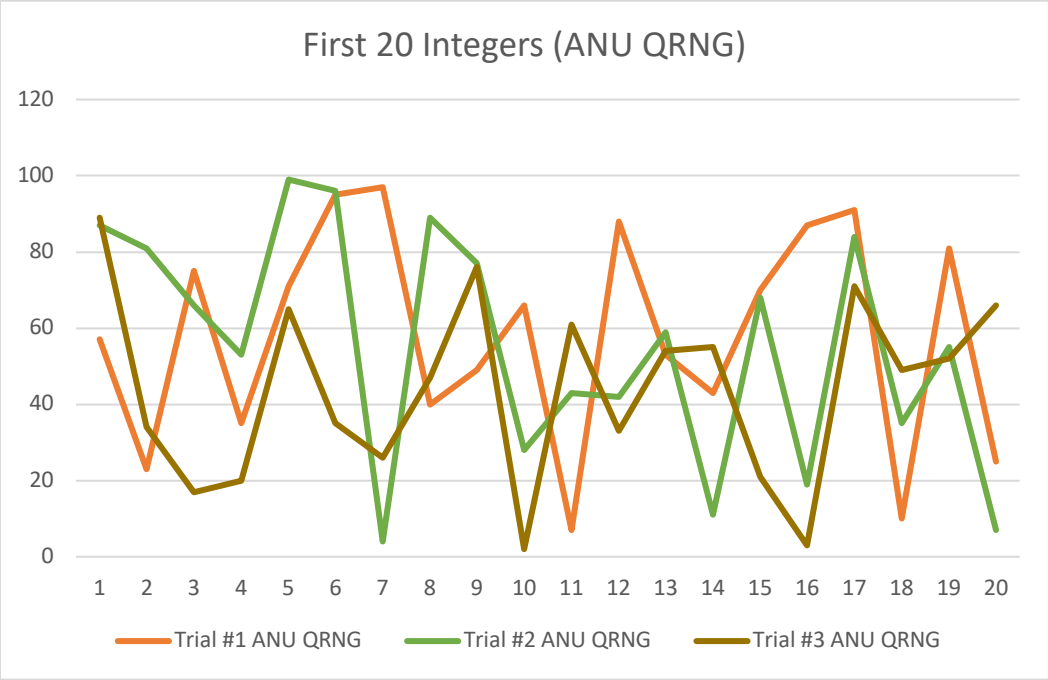
## TRNGs

*(Only the first 20 Integers are shown. The complete tables and graphs are available at Appendix B)*

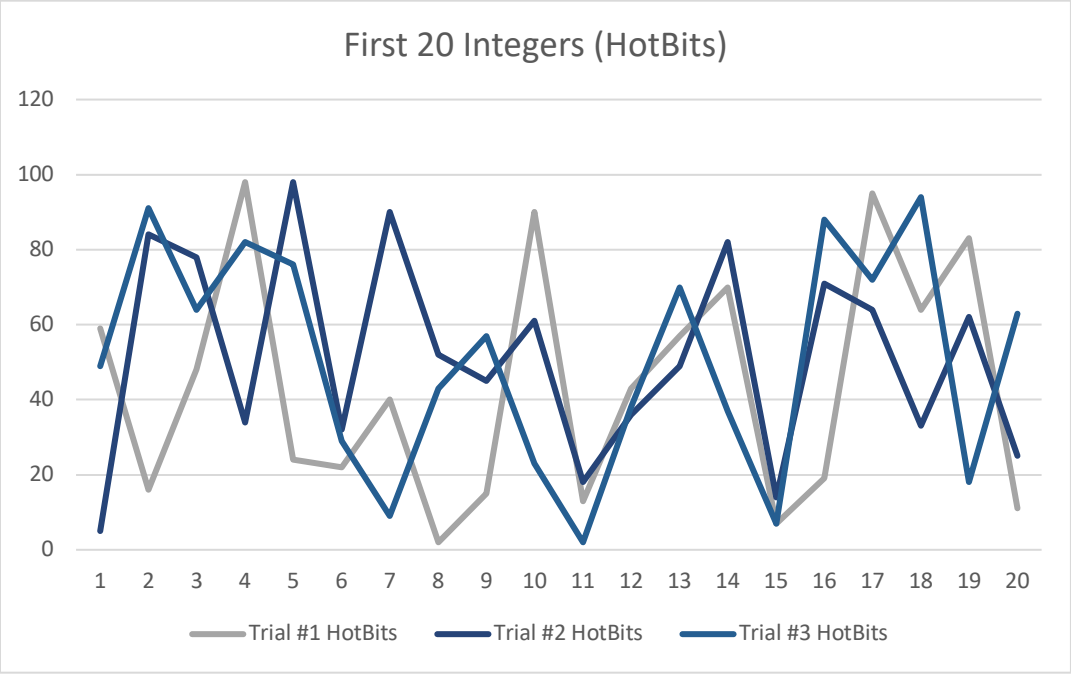
| **Random.org** | | |
|---|---|---|
| **Trial #1** | **Trial #2** | **Trial #3** |
| 30 | 99 | 43 |
| 71 | 58 | 10 |
| 75 | 54 | 36 |
| 4 | 55 | 85 |
| 39 | 88 | 33 |
| 35 | 29 | 87 |
| 9 | 34 | 91 |
| 42 | 47 | 32 |
| 77 | 66 | 98 |
| 85 | 26 | 21 |
| 63 | 9 | 82 |
| 6 | 59 | 71 |
| 45 | 31 | 14 |
| 98 | 95 | 100 |
| 48 | 71 | 72 |
| 91 | 91 | 30 |
| 80 | 86 | 65 |
| 36 | 11 | 35 |
| 37 | 97 | 47 |
| 28 | 28 | 51 |



| ANU QRNG | | |
|---|---|---|
| **Trial #1** | **Trial #2** | **Trial #3** |
| 57 | 87 | 89 |
| 23 | 81 | 34 |
| 75 | 66 | 17 |
| 35 | 53 | 20 |
| 71 | 99 | 65 |
| 95 | 96 | 35 |
| 97 | 4 | 26 |
| 40 | 89 | 47 |
| 49 | 77 | 76 |
| 66 | 28 | 2 |
| 7 | 43 | 61 |
| 88 | 42 | 33 |
| 53 | 59 | 54 |
| 43 | 11 | 55 |
| 70 | 68 | 21 |
| 87 | 19 | 3 |
| 91 | 84 | 71 |
| 10 | 35 | 49 |
| 81 | 55 | 52 |
| 25 | 7 | 66 |



| HotBits | | |
|---|---|---|
| **Trial #1** | **Trial #2** | **Trial #3** |
| 59 | 5 | 49 |
| 16 | 84 | 91 |
| 48 | 78 | 64 |
| 98 | 34 | 82 |
| 24 | 98 | 76 |
| 22 | 32 | 29 |
| 40 | 90 | 9 |
| 2 | 52 | 43 |
| 15 | 45 | 57 |
| 90 | 61 | 23 |
| 13 | 18 | 2 |
| 43 | 36 | 38 |
| 57 | 49 | 70 |
| 70 | 82 | 37 |
| 7 | 14 | 7 |
| 19 | 71 | 88 |
| 95 | 64 | 72 |
| 64 | 33 | 94 |
| 83 | 62 | 18 |
| 11 | 25 | 63 |



# Data Analysis

## PRNGs

### Java (Java.util.Random)

In order to measure the entropy and distribution of each generator, I thought it would be a good idea to see if it was possible to predict the integers produced by each generator. As I found in my background research, the Java.util.Random generator is a Linear Congruential Generator (LCG). This means that it essentially uses the same formula as the one I detailed above. After I confirmed that this generator does use this algorithm, it was just a matter of finding the missing values. For the sequences with the prime and square seeds, the only unknown variables were; the modulus, the increment, and the multiplier. For the sequence with default seeds, this would prove to be more of a challenge. Fortunately, the modulus, increment, and multiplier values of this particular generator were all available online[10]. In order to find the seed values of the default sequence, a brute-force algorithm was required.

[10] Bajaj, T., 2020. Random Setseed() Method In Java With Examples - Geeksforgeeks. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/random-setseed-method-in-java-with-examples/> [Accessed 16 September 2020].

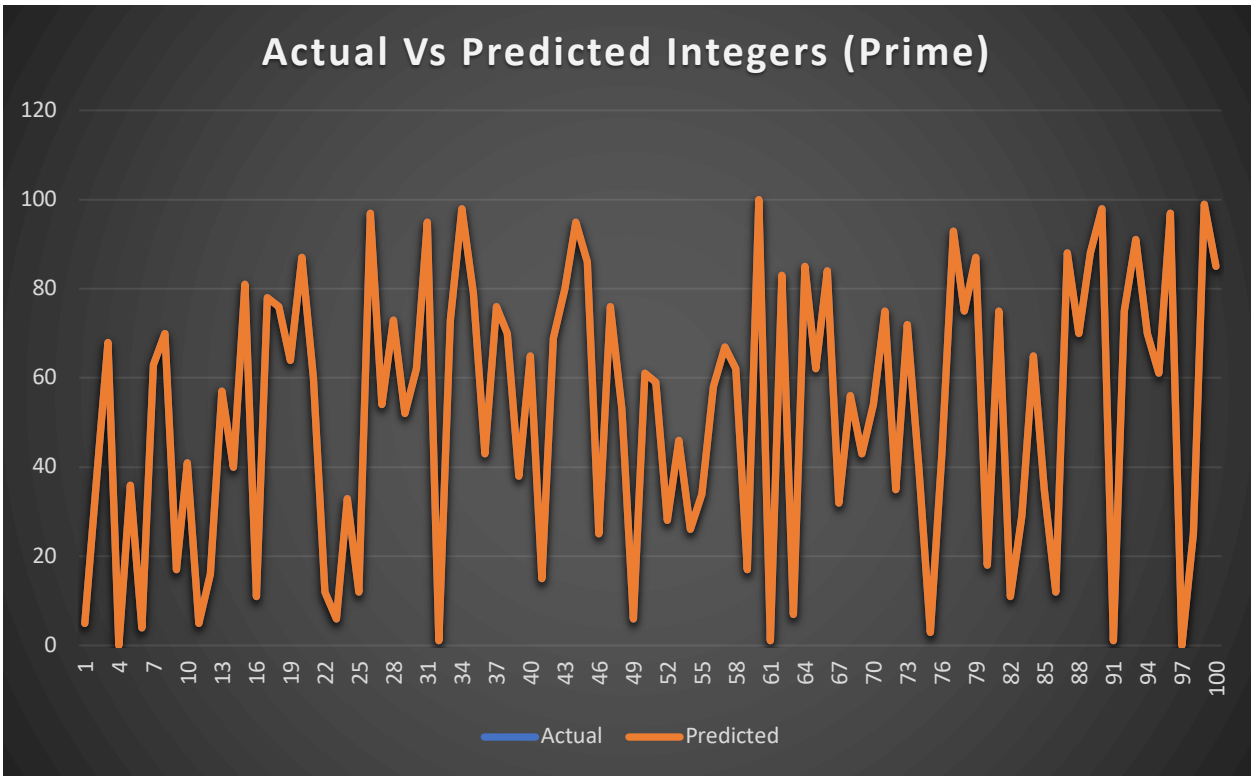
I downloaded a GitHub project that claimed to do this[11] and implemented it into my own program *(Available at Appendix C)*. The following are the results:

### Prime Seeds:

*(Only the initial 20 integers are shown in the table. The full table is available at Appendix D)*

| Actual | Predicted |
|---|---|
| 5 | 5 |
| 37 | 37 |
| 68 | 68 |
| 0 | 0 |
| 36 | 36 |
| 4 | 4 |
| 63 | 63 |
| 70 | 70 |
| 17 | 17 |
| 41 | 41 |
| 5 | 5 |
| 16 | 16 |
| 57 | 57 |
| 40 | 40 |
| 81 | 81 |
| 11 | 11 |
| 78 | 78 |
| 76 | 76 |
| 64 | 64 |
| 87 | 87 |

[11] 2014. *Replicatedrandom*. GitHub

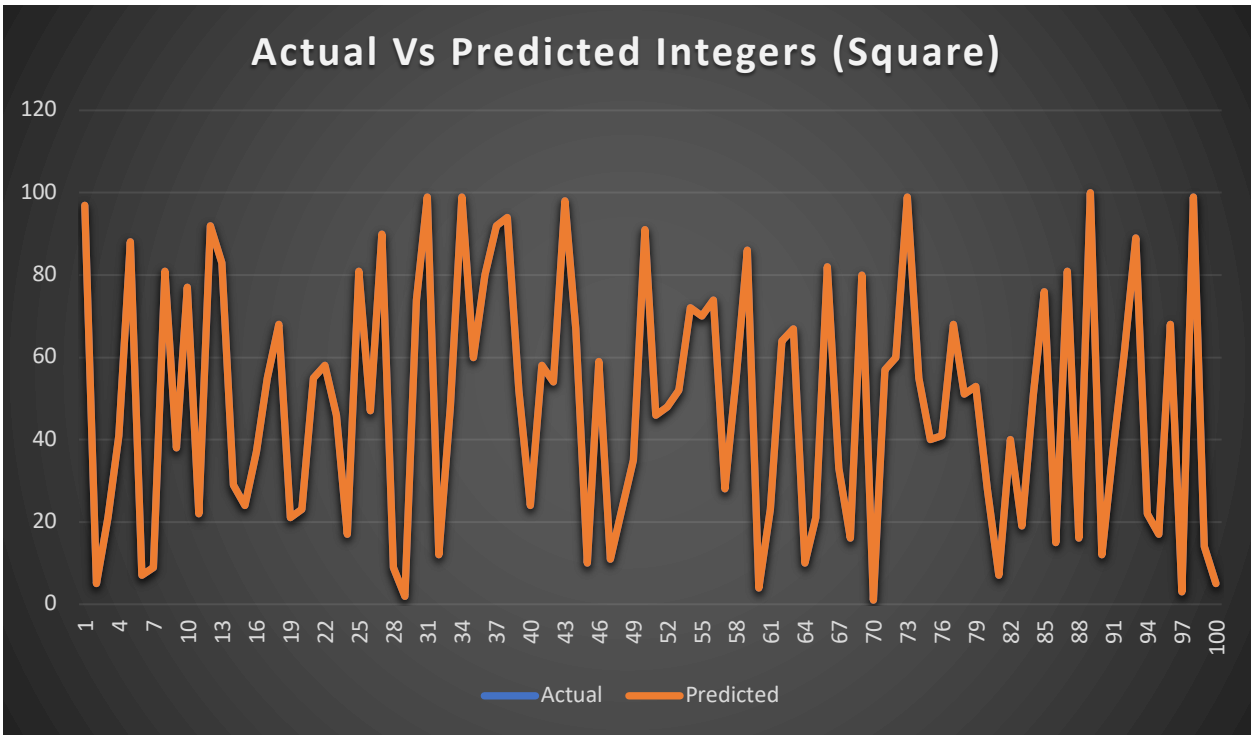



***(Blue line cannot be seen because it is exactly aligned with the orange line)***

### Square Seeds:

*(Only the initial 20 integers are shown in the table. The full table is available at Appendix D)*

| Actual | Predicted |
|---|---|
| 97 | 97 |
| 5 | 5 |
| 21 | 21 |
| 41 | 41 |
| 88 | 88 |
| 7 | 7 |
| 9 | 9 |
| 81 | 81 |
| 38 | 38 |
| 77 | 77 |
| 22 | 22 |
| 92 | 92 |
| 83 | 83 |
| 29 | 29 |
| 24 | 24 |
| 37 | 37 |
| 55 | 55 |
| 68 | 68 |
| 21 | 21 |
| 23 | 23 |

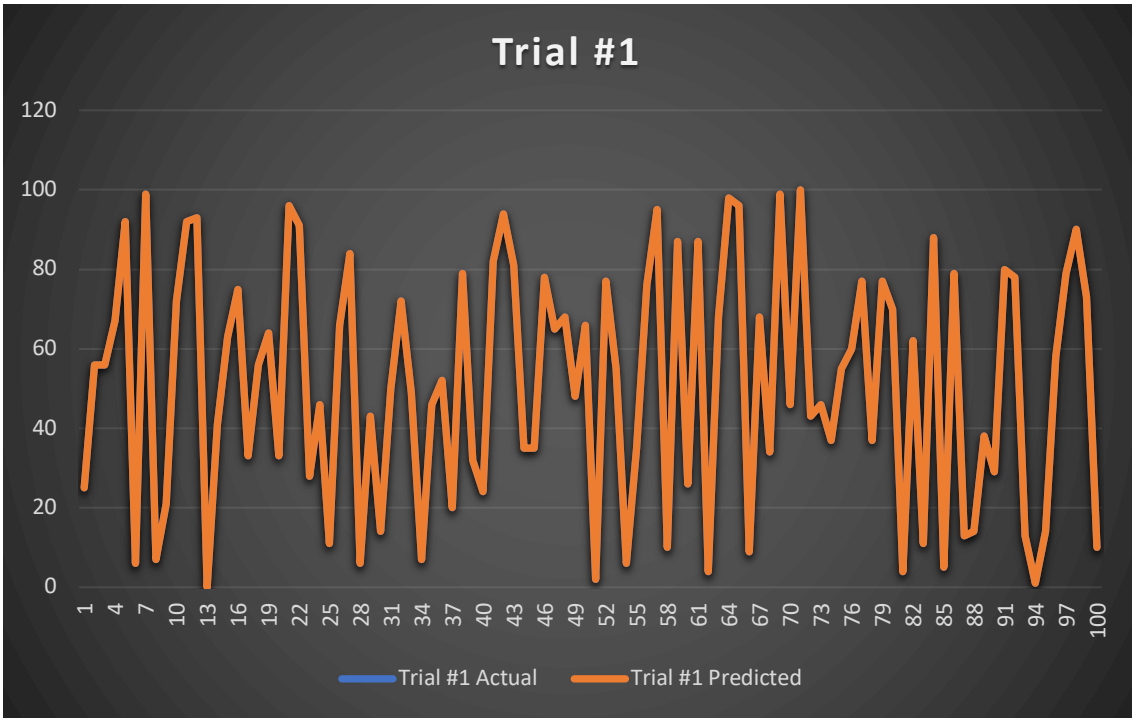


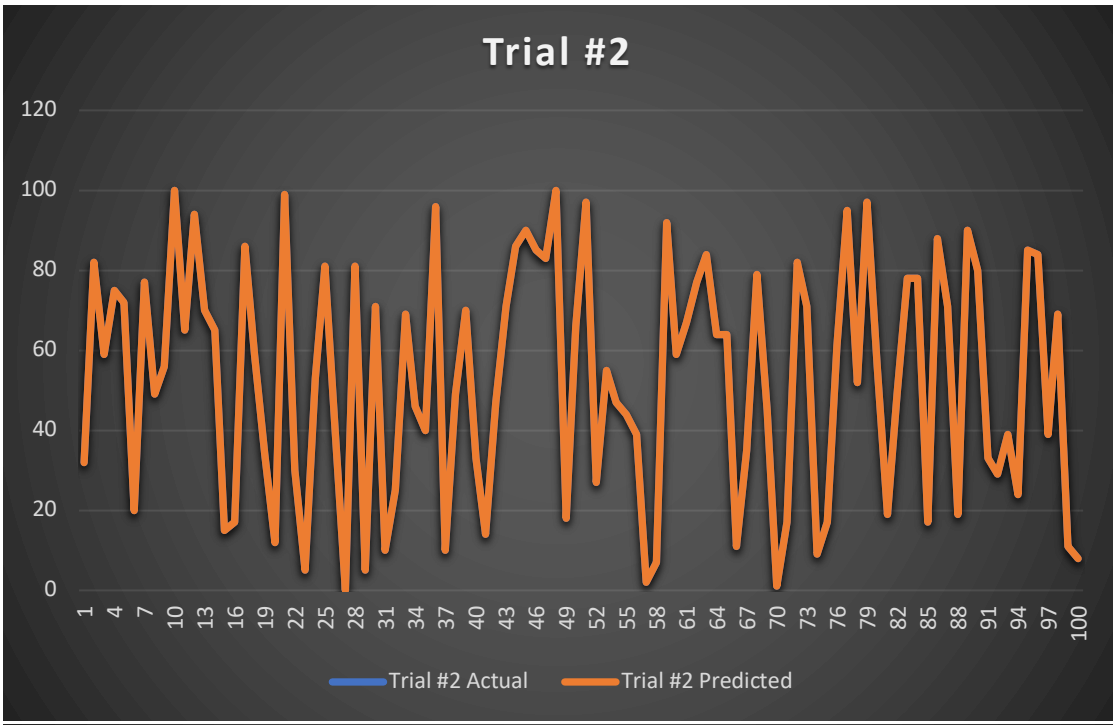

***(The blue line cannot be seen because it is exactly aligned with the orange line)***

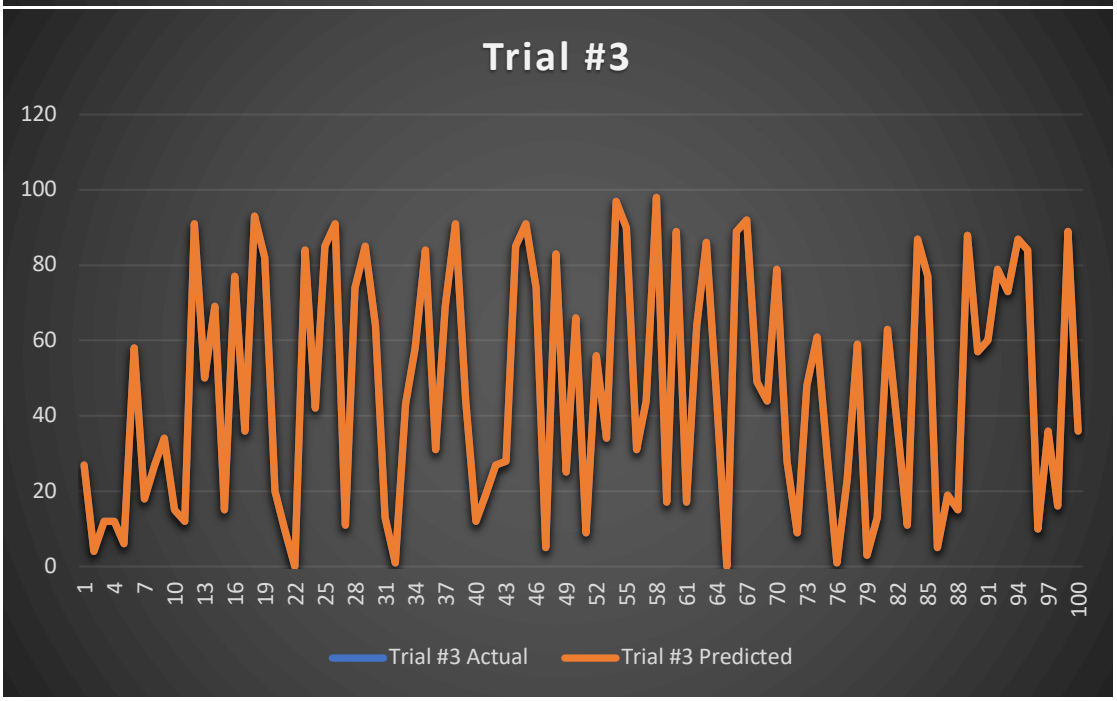
## Default Seeds:

*(Only the initial 20 integers are shown in the table. The full table is available at Appendix D)*

| Trial #1 | | | Trial #2 | | | Trial #3 | |
|---|---|---|---|---|---|---|---|
| **Actual** | **Predicted** | | **Actual** | **Predicted** | | **Actual** | **Predicted** |
| 25 | 25 | | 32 | 32 | | 27 | 27 |
| 56 | 56 | | 82 | 82 | | 4 | 4 |
| 56 | 56 | | 59 | 59 | | 12 | 12 |
| 67 | 67 | | 75 | 75 | | 12 | 12 |
| 92 | 92 | | 72 | 72 | | 6 | 6 |
| 6 | 6 | | 20 | 20 | | 58 | 58 |
| 99 | 99 | | 77 | 77 | | 18 | 18 |
| 7 | 7 | | 49 | 49 | | 27 | 27 |
| 21 | 21 | | 56 | 56 | | 34 | 34 |
| 72 | 72 | | 100 | 100 | | 15 | 15 |
| 92 | 92 | | 65 | 65 | | 12 | 12 |
| 93 | 93 | | 94 | 94 | | 91 | 91 |
| 0 | 0 | | 70 | 70 | | 50 | 50 |
| 41 | 41 | | 65 | 65 | | 69 | 69 |
| 63 | 63 | | 15 | 15 | | 15 | 15 |
| 75 | 75 | | 17 | 17 | | 77 | 77 |
| 33 | 33 | | 86 | 86 | | 36 | 36 |
| 56 | 56 | | 59 | 59 | | 93 | 93 |
| 64 | 64 | | 34 | 34 | | 82 | 82 |






***(Blue line cannot be seen because it is exactly aligned with the orange line)***

## C++ (Rand())

Unlike Java's PRNG which uses 48-bit seeds, the C++ rand function uses 32-bit seeds[12]. It is also an MCG, rather than a full LCG. This makes it much easier to crack, but it also requires a different approach. Instead of building my own program from scratch, I found another GitHub project called 'Untwister'[13]. I did not make any changes whatsoever to this code as I did for the previous project. The following are the results:

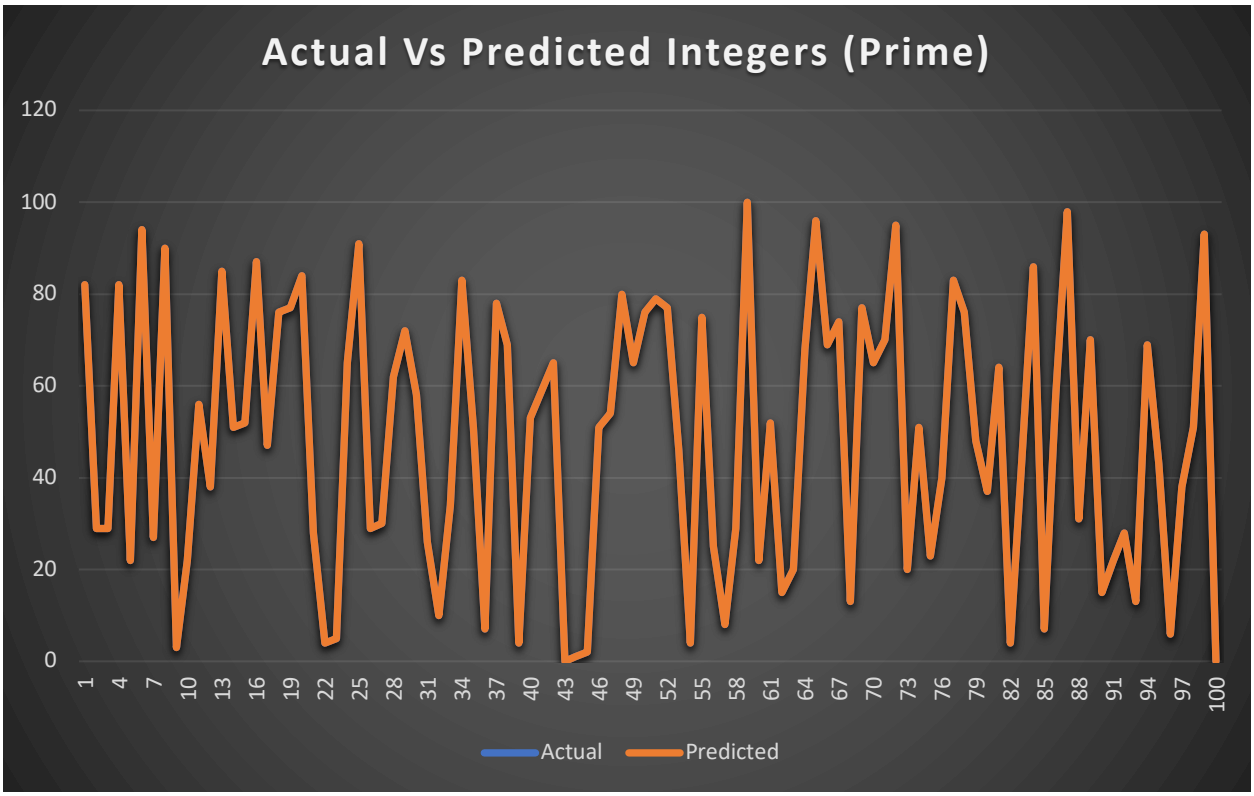
### Prime Seeds:

*(Only the initial 20 integers are shown in the table. The full table is available at Appendix D)*

| Actual | Predicted |
|---|---|
| 82 | 82 |
| 29 | 29 |
| 29 | 29 |
| 82 | 82 |
| 22 | 22 |
| 94 | 94 |
| 27 | 27 |
| 90 | 90 |
| 3 | 3 |
| 22 | 22 |
| 56 | 56 |
| 38 | 38 |
| 85 | 85 |
| 51 | 51 |
| 52 | 52 |
| 87 | 87 |
| 47 | 47 |
| 76 | 76 |
| 77 | 77 |
| 84 | 84 |

[12] GeeksforGeeks. Undated. Rand() And Srand() In C/C++ - Geeksforgeeks. [online] Available at: <https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/> [Accessed 16 September 2020].

[13] 2014. Untwister. GitHub.



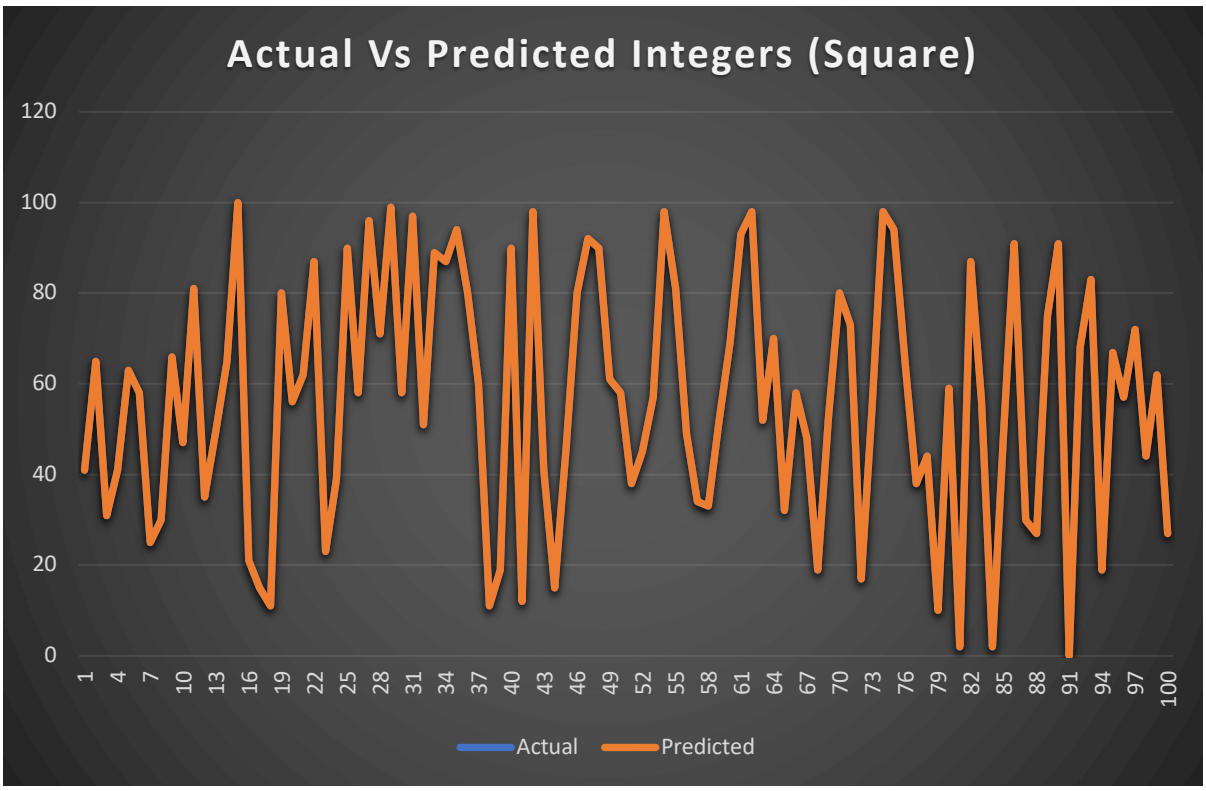

***(The blue line cannot be seen because it is exactly aligned with the orange line)***

## Square Seeds:

*(Only the initial 20 integers are shown in the table. The full table is available at Appendix D)*

| Actual | Predicted |
|---|---|
| 41 | 41 |
| 65 | 65 |
| 31 | 31 |
| 41 | 41 |
| 63 | 63 |
| 58 | 58 |
| 25 | 25 |
| 30 | 30 |
| 66 | 66 |
| 47 | 47 |
| 81 | 81 |
| 35 | 35 |
| 50 | 50 |
| 65 | 65 |
| 100 | 100 |
| 21 | 21 |

| | |
|---|---|
| 15 | 15 |
| 11 | 11 |
| 80 | 80 |
| 56 | 56 |



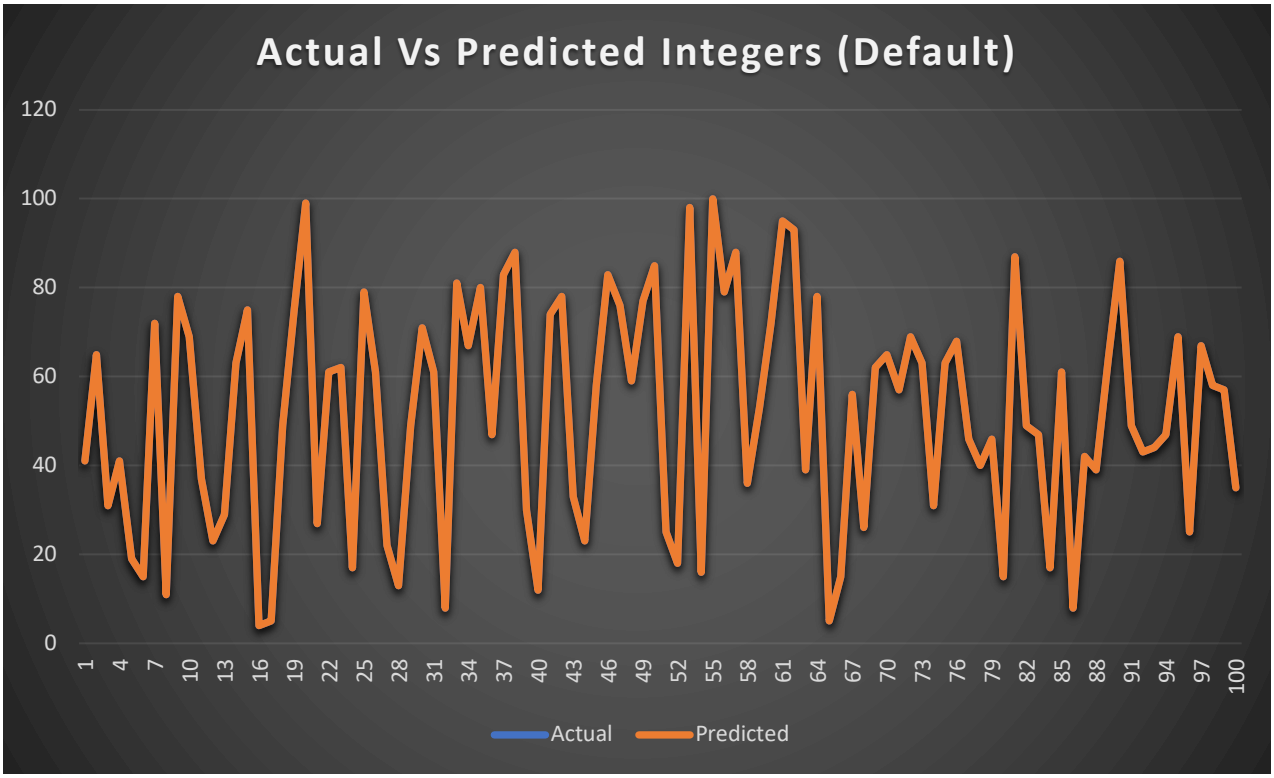

***(Blue line cannot be seen because it is exactly aligned with the orange line)***

## Default Seeds:

*(Only the initial 20 integers are shown in the table. The full table is available at Appendix D)*

| **Actual** | **Predicted** |
|---|---|
| 41 | 41 |
| 65 | 65 |
| 31 | 31 |
| 41 | 41 |
| 19 | 19 |
| 15 | 15 |
| 72 | 72 |
| 11 | 11 |
| 78 | 78 |
| 69 | 69 |
| 37 | 37 |
| 23 | 23 |
| 29 | 29 |

| | |
|---|---|
| 63 | 63 |
| 75 | 75 |
| 4 | 4 |
| 5 | 5 |
| 49 | 49 |
| 75 | 75 |
| 99 | 99 |




***(Blue line cannot be seen because it is exactly aligned with the orange line)***

## Swift (GKMersenneTwister)

The Mersenne Twister algorithm is the most secure and commonly used out of all the three. Instead of using 32-bit or 48-bit seeds, the Mersenne Twister uses 64-bit seeds[12]. The Mersenne Twister is a TGFSR algorithm, based on the GFSR algorithm[13]. Fortunately, the program used for the C++ rand function will also work for the Mersenne Twister. The following are the results:

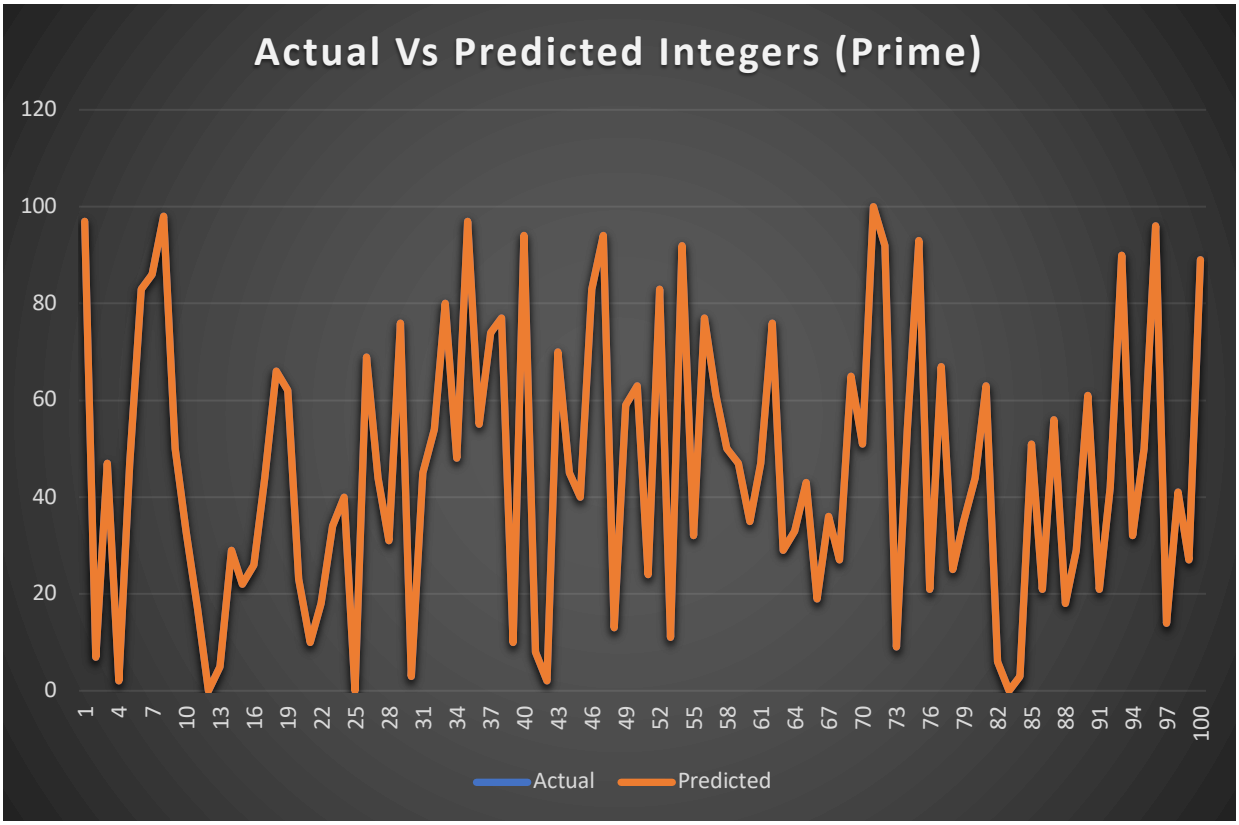
### Prime Seeds:

*(Only the initial 20 integers are shown in the table. The full table is available at Appendix D)*

| Actual | Predicted |
|---|---|
| 97 | 97 |
| 7 | 7 |
| 47 | 47 |
| 2 | 2 |
| 47 | 47 |
| 83 | 83 |
| 86 | 86 |
| 98 | 98 |
| 50 | 50 |
| 33 | 33 |
| 17 | 17 |
| 0 | 0 |
| 5 | 5 |
| 29 | 29 |
| 22 | 22 |
| 26 | 26 |
| 44 | 44 |
| 66 | 66 |
| 62 | 62 |
| 23 | 23 |

[14] En.wikipedia.org. Undated. Mersenne Twister. [online] Available at: <https://en.wikipedia.org/wiki/Mersenne_Twister> [Accessed 16 September 2020].

[15] Developer.apple.com. n.d. Apple Developer Documentation. [online] Available at: <https://developer.apple.com/documentation/gameplaykit/gkmersennetwisterrandomsource> [Accessed 16 September 2020].



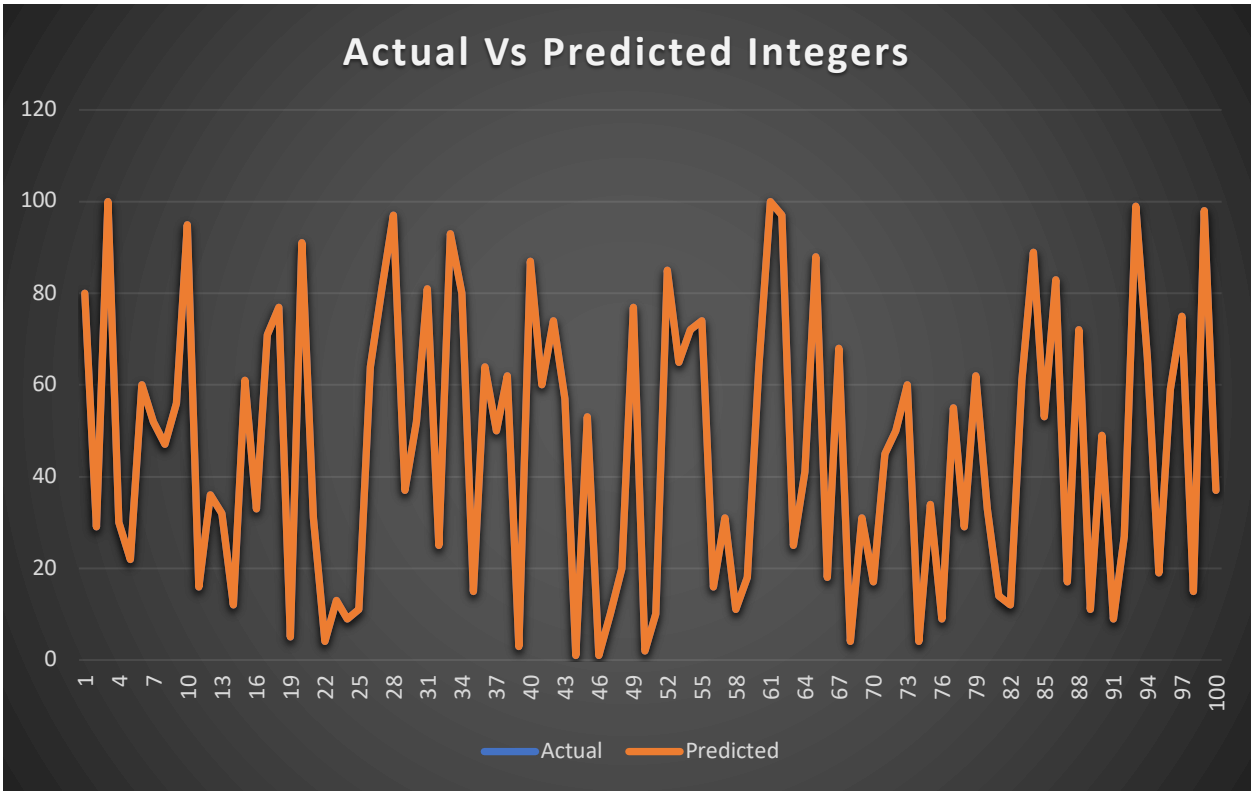

***(The blue line cannot be seen because it is exactly aligned with the orange line)***

## Square Seeds:

*(Only the initial 20 integers are shown in the table. The full table is available at Appendix D)*

| Actual | Predicted |
|---|---|
| 80 | 80 |
| 29 | 29 |
| 100 | 100 |
| 30 | 30 |
| 22 | 22 |
| 60 | 60 |
| 52 | 52 |
| 47 | 47 |
| 56 | 56 |
| 95 | 95 |
| 16 | 16 |
| 36 | 36 |
| 32 | 32 |
| 12 | 12 |
| 61 | 61 |

| | |
|---|---|
| 33 | 33 |
| 71 | 71 |
| 77 | 77 |
| 5 | 5 |
| 91 | 91 |



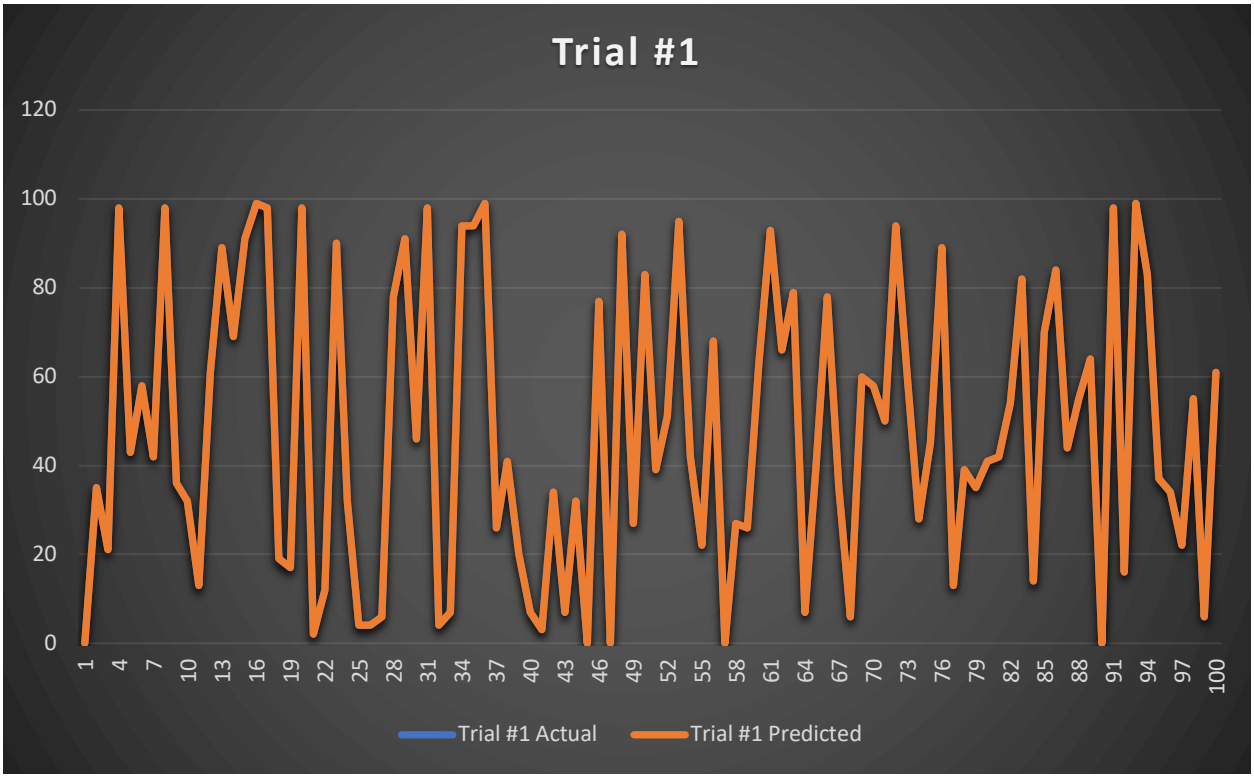

***(The blue line cannot be seen because it is exactly aligned with the orange line)***

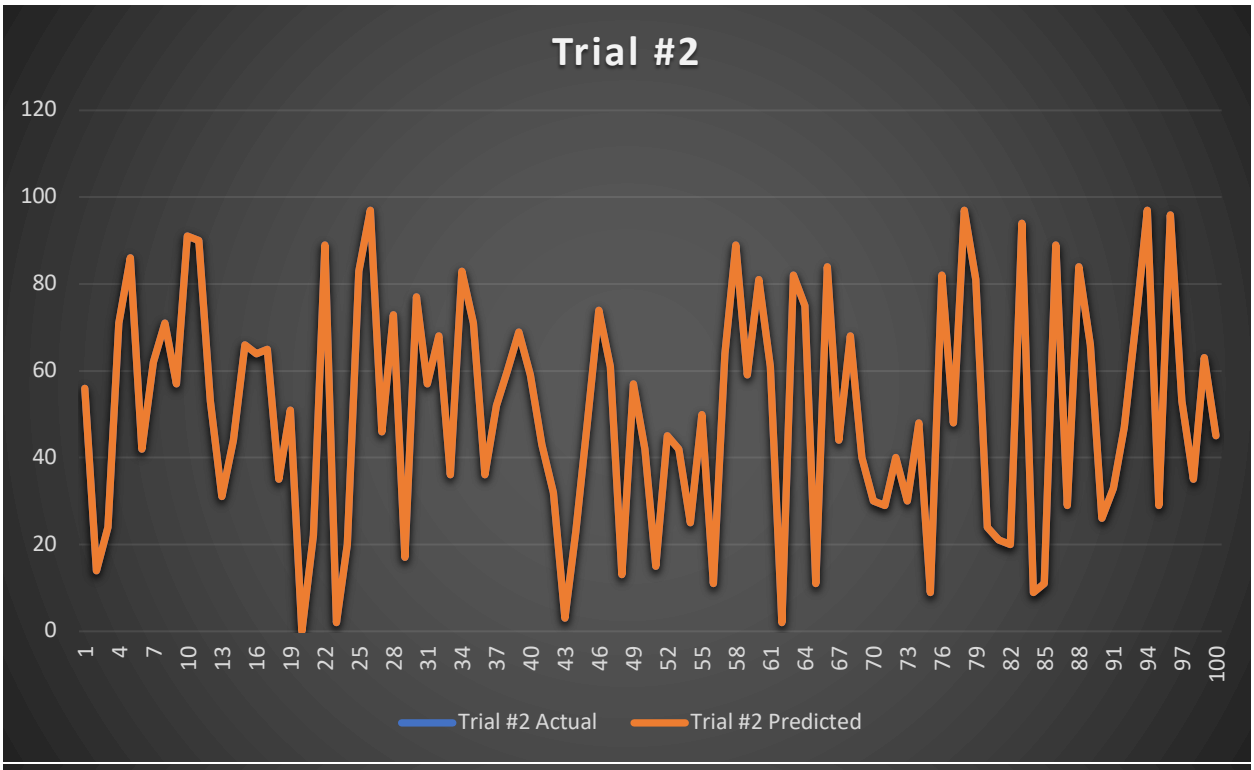
## Default Seeds:

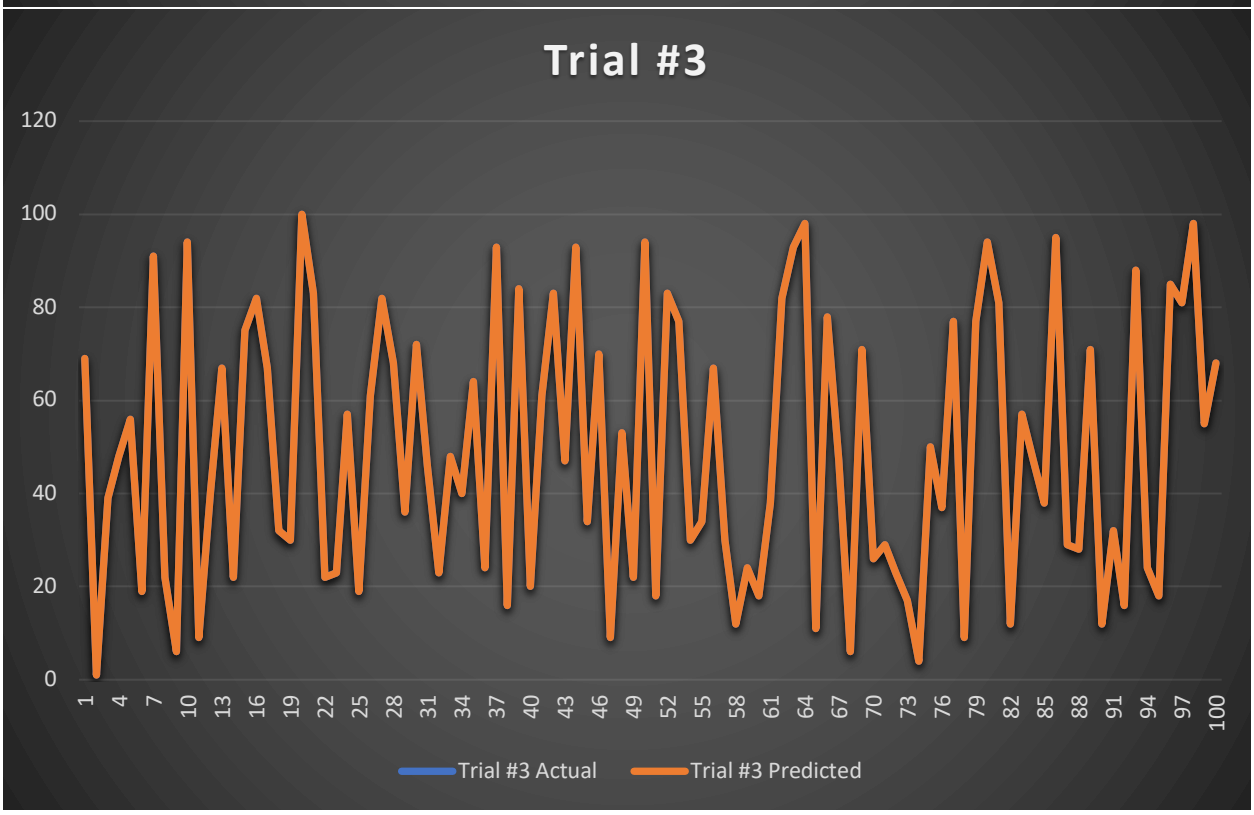
*(Only the initial 20 integers are shown in the table. The full table is available at Appendix D)*

| **Trial #1** | | | **Trial #2** | | | **Trial #3** | |
|---|---|---|---|---|---|---|---|
| **Actual** | **Predicted** | | **Actual** | **Predicted** | | **Actual** | **Predicted** |
| 0 | 0 | | 56 | 56 | | 69 | 69 |
| 35 | 35 | | 14 | 14 | | 1 | 1 |
| 21 | 21 | | 24 | 24 | | 39 | 39 |
| 98 | 98 | | 71 | 71 | | 48 | 48 |
| 43 | 43 | | 86 | 86 | | 56 | 56 |
| 58 | 58 | | 42 | 42 | | 19 | 19 |
| 42 | 42 | | 62 | 62 | | 91 | 91 |
| 98 | 98 | | 71 | 71 | | 22 | 22 |
| 36 | 36 | | 57 | 57 | | 6 | 6 |
| 32 | 32 | | 91 | 91 | | 94 | 94 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 13 | 13 | | 90 | 90 | | 9 | 9 |
| 61 | 61 | | 53 | 53 | | 40 | 40 |
| 89 | 89 | | 31 | 31 | | 67 | 67 |
| 69 | 69 | | 44 | 44 | | 22 | 22 |
| 91 | 91 | | 66 | 66 | | 75 | 75 |
| 99 | 99 | | 64 | 64 | | 82 | 82 |
| 98 | 98 | | 65 | 65 | | 67 | 67 |
| 19 | 19 | | 35 | 35 | | 32 | 32 |
| 17 | 17 | | 51 | 51 | | 30 | 30 |
| 98 | 98 | | 0 | 0 | | 100 | 100 |





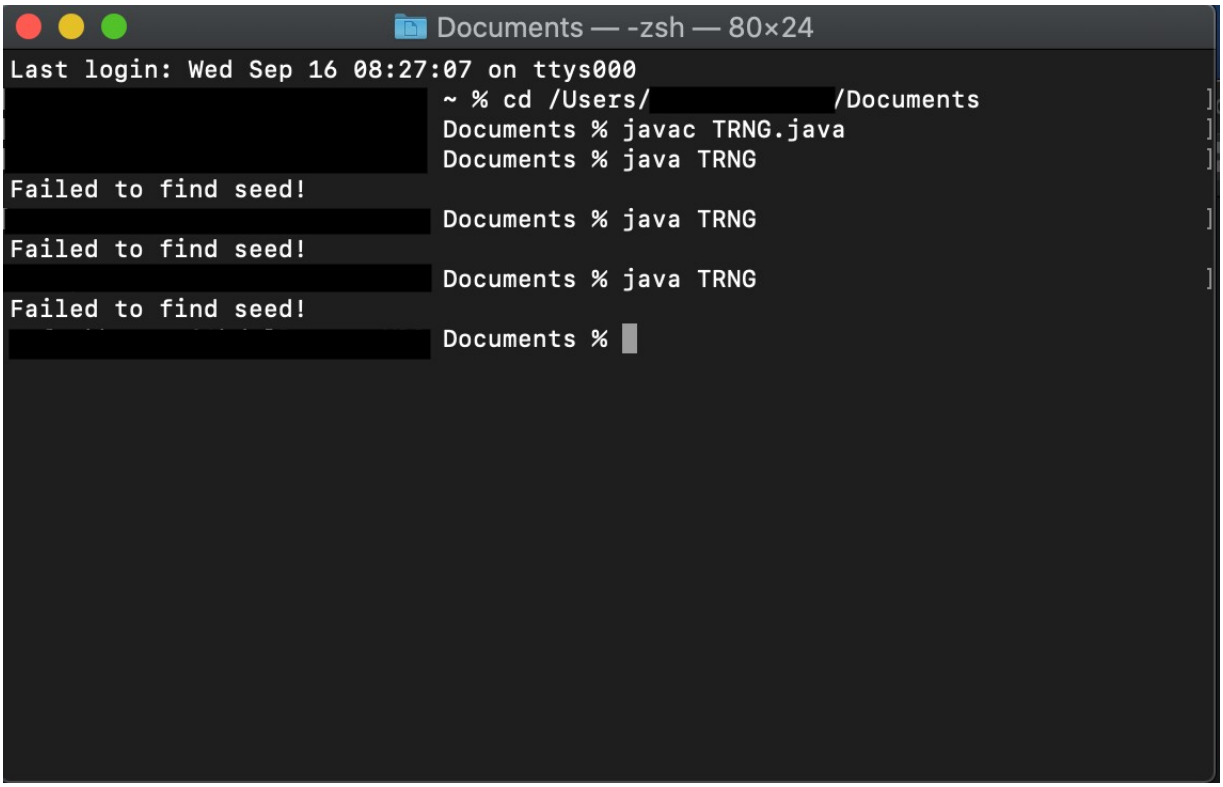

***(The blue line cannot be seen because it is exactly aligned with the orange line)***

## TRNGs

Unlike PRNGs, TRNGs do not have one seed as a default. They take seeds from extremely random sources, such as radioactive decay, and use that as their new seed every time. This means that TRNGs are not deterministic, which means every value is independent of the previous. In addition to this, TRNGs typically use proprietary algorithms that are not published, making them even more difficult to crack. So, it is expected that none of the previous programs would be able to crack it and predict the next values. To test this, I put all of the TRNG generated sequences through both of the programs, trying every possible configuration. As expected, neither of the programs were able to crack any of the three sequences. The following is the output:

### Program #1:

*(Some information is censored in order to protect my identity)*

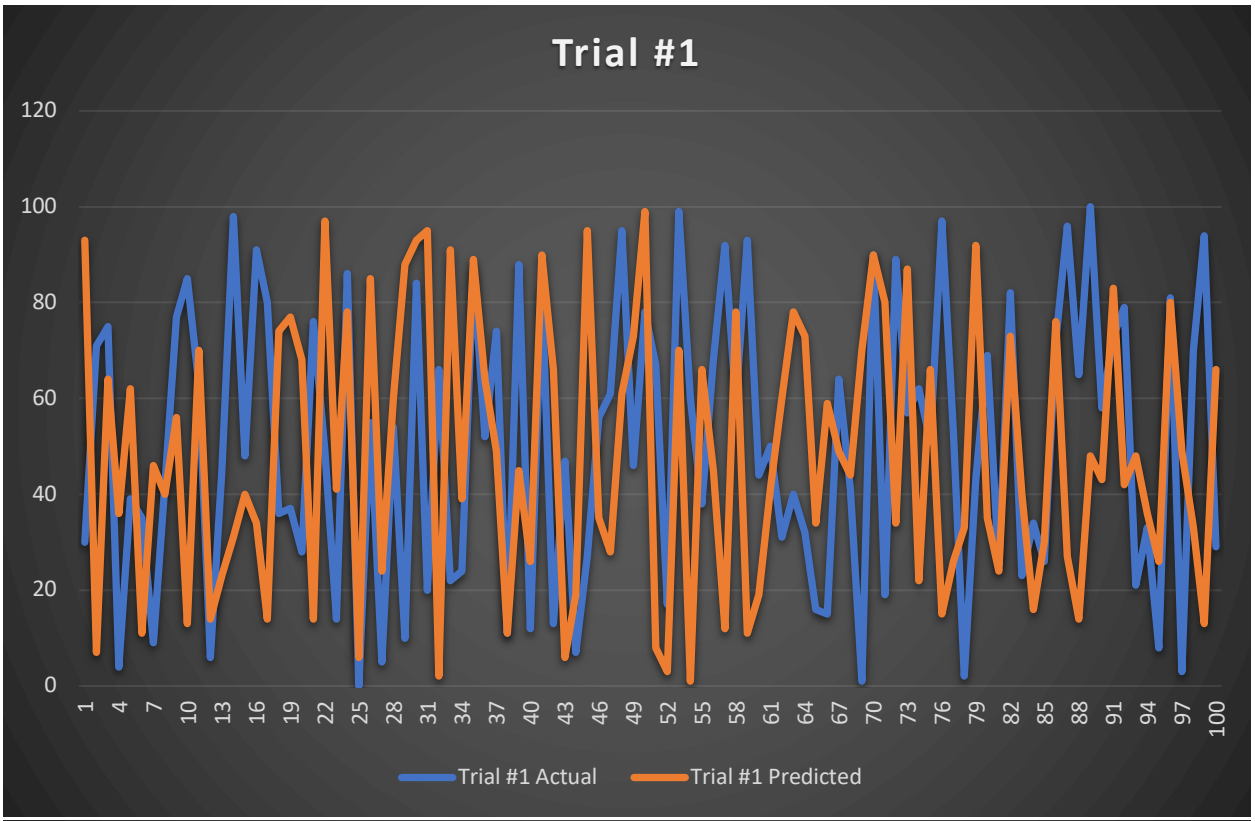


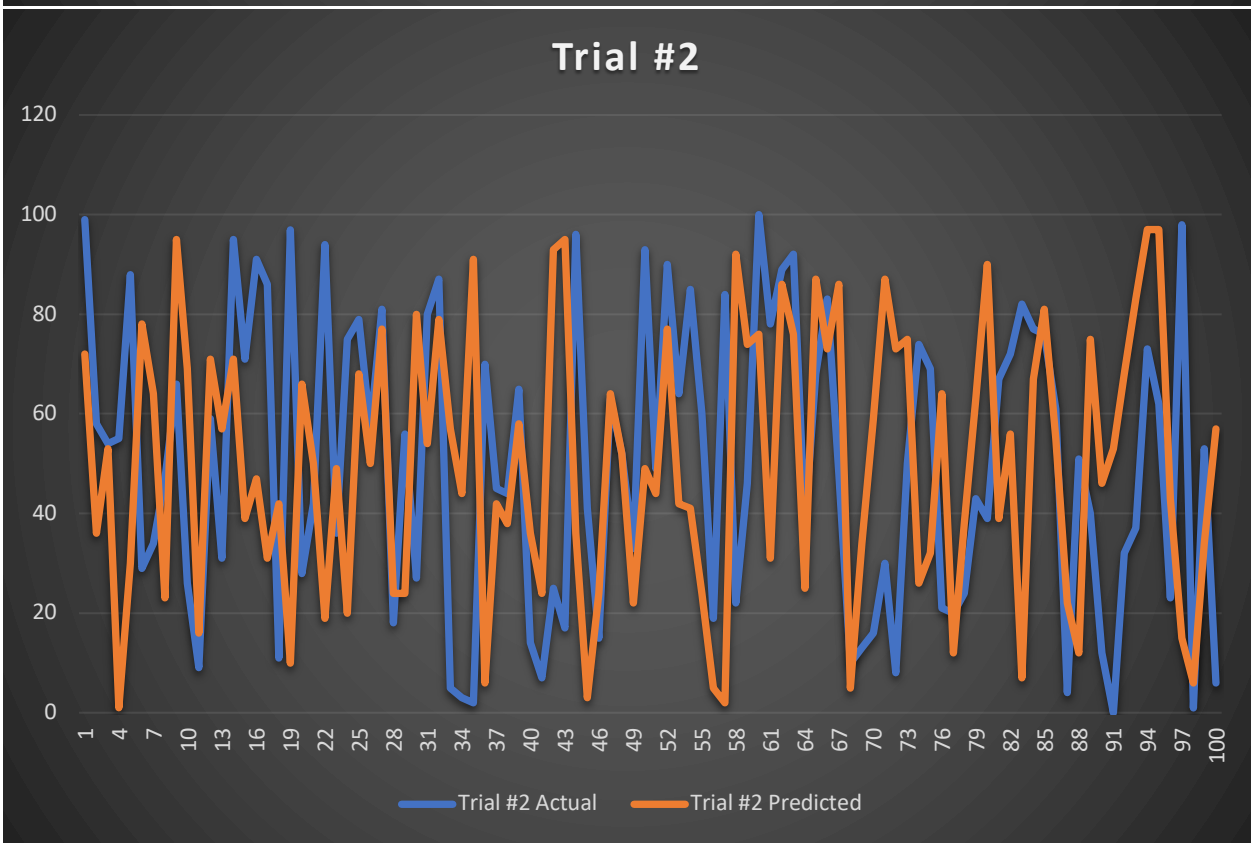
## Program #2:

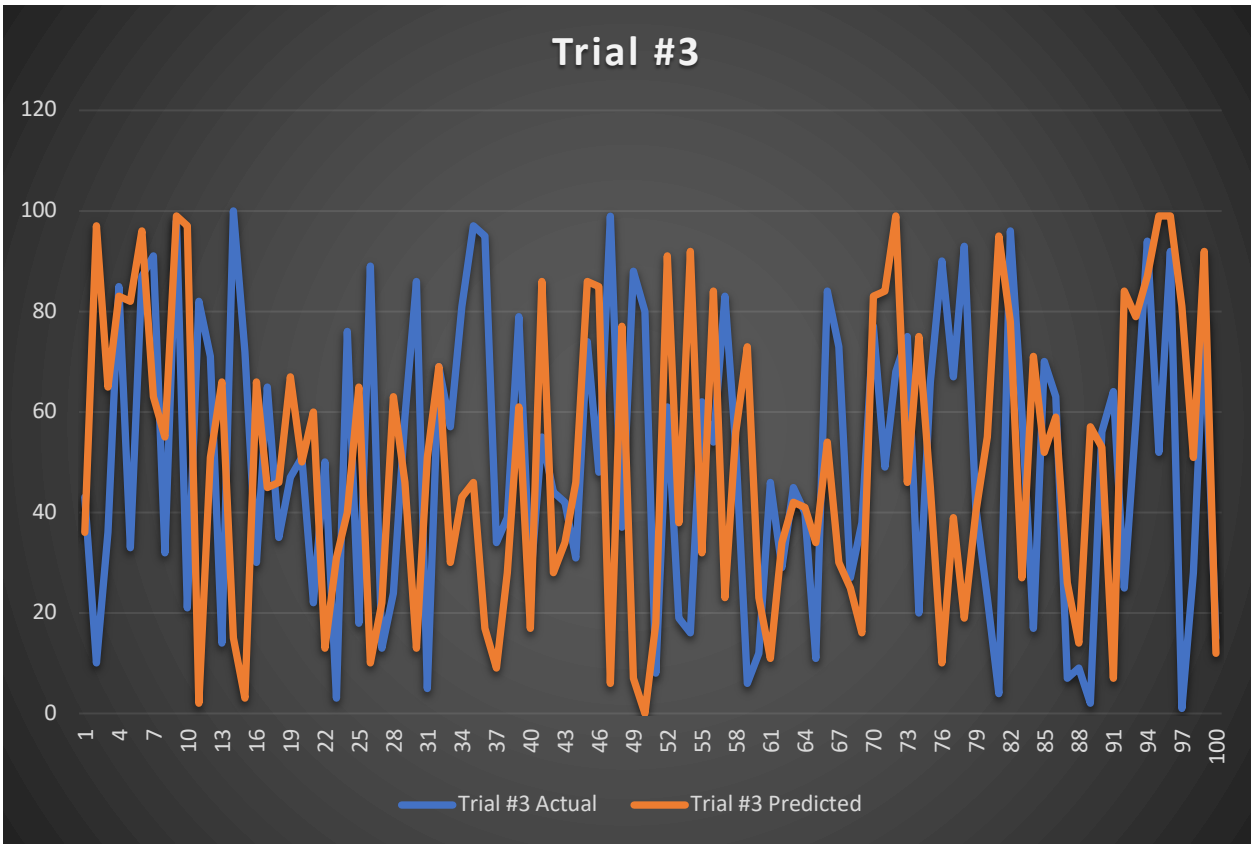
### Random.org

*(Only the initial 20 integers are shown in the table. The full table is available at Appendix E)*

| **Trial #1** | | | **Trial #2** | | | **Trial #3** | |
|---|---|---|---|---|---|---|---|
| **Actual** | **Predicted** | | **Actual** | **Predicted** | | **Actual** | **Predicted** |
| 30 | 93 | | 99 | 72 | | 43 | 36 |
| 71 | 7 | | 58 | 36 | | 10 | 97 |
| 75 | 64 | | 54 | 53 | | 36 | 65 |
| 4 | 36 | | 55 | 1 | | 85 | 83 |
| 39 | 62 | | 88 | 30 | | 33 | 82 |
| 35 | 11 | | 29 | 78 | | 87 | 96 |
| 9 | 46 | | 34 | 64 | | 91 | 63 |
| 42 | 40 | | 47 | 23 | | 32 | 55 |
| 77 | 56 | | 66 | 95 | | 98 | 99 |
| 85 | 13 | | 26 | 69 | | 21 | 97 |
| 63 | 70 | | 9 | 16 | | 82 | 2 |
| 6 | 14 | | 59 | 71 | | 71 | 51 |
| 45 | 23 | | 31 | 57 | | 14 | 66 |
| 98 | 31 | | 95 | 71 | | 100 | 15 |
| 48 | 40 | | 71 | 39 | | 72 | 3 |
| 91 | 34 | | 91 | 47 | | 30 | 66 |
| 80 | 14 | | 86 | 31 | | 65 | 45 |
| 36 | 74 | | 11 | 42 | | 35 | 46 |
| 37 | 77 | | 97 | 10 | | 47 | 67 |
| 28 | 68 | | 28 | 66 | | 51 | 50 |


Trial #1
120
100
80
60
40
20
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100
Trial #1 Actual
Trial #1 Predicted


Trial #2
120
100
80
60
40
20
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100
Trial #2 Actual
Trial #2 Predicted

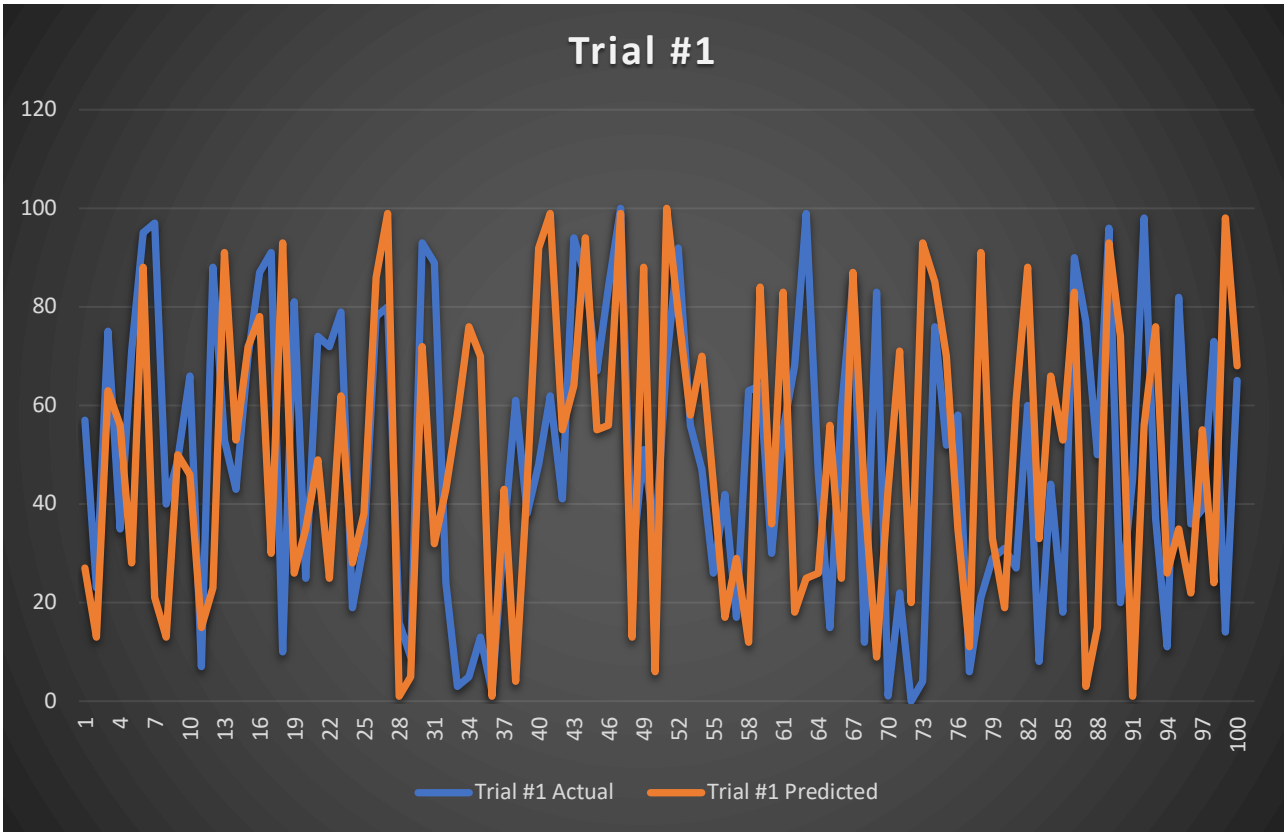


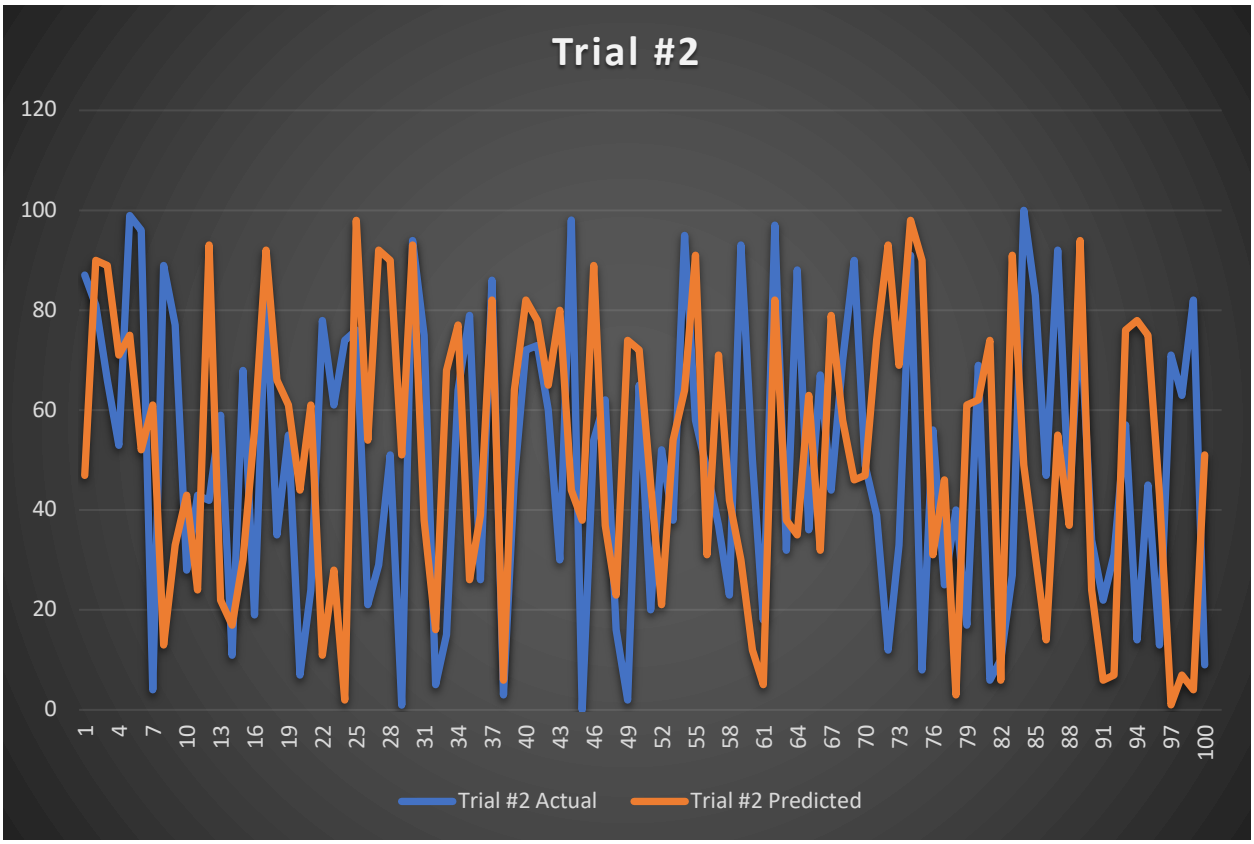

## ANU QRNG

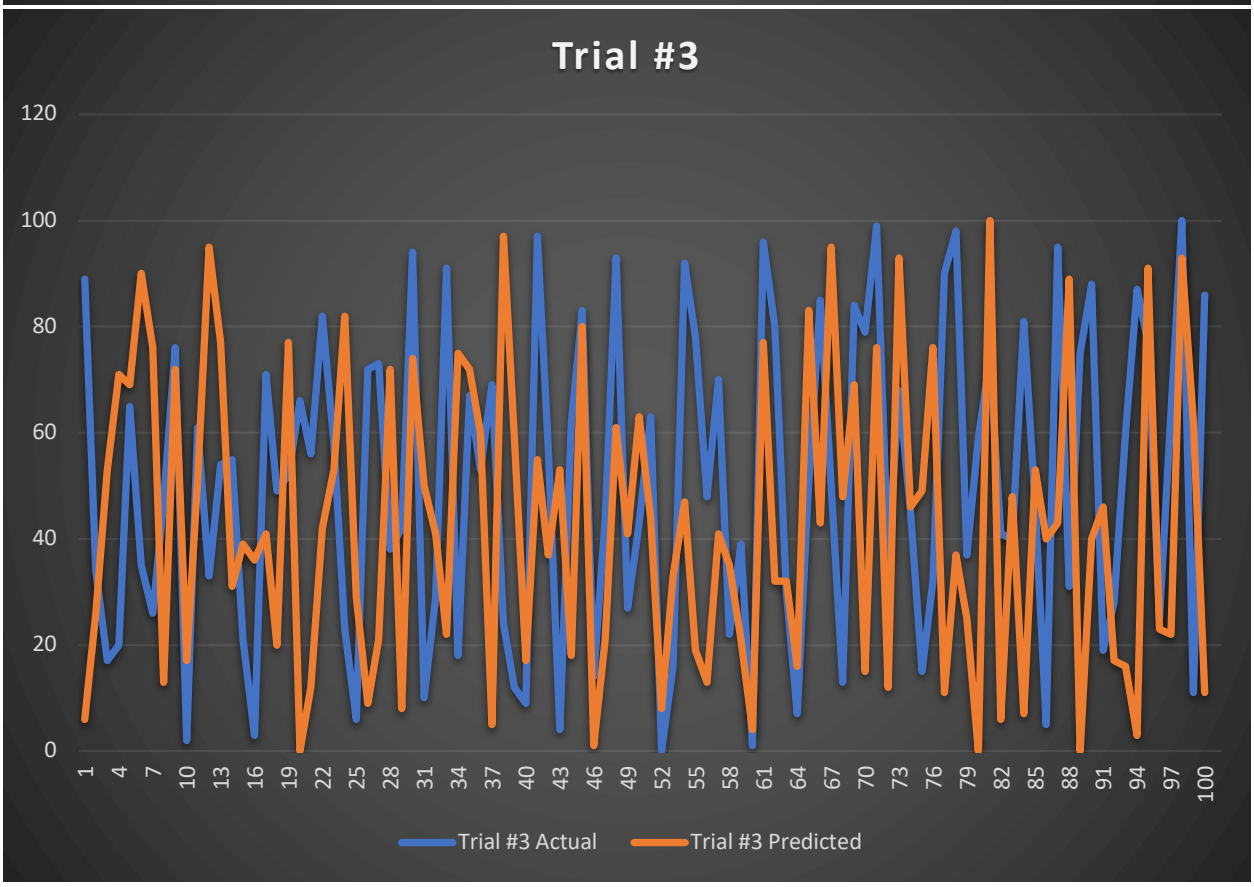
*(Only the initial 20 integers are shown in the table. The full table is available at Appendix E)*

| **Trial #1** | | | **Trial #2** | | | **Trial #3** | |
|---|---|---|---|---|---|---|---|
| **Actual** | **Predicted** | | **Actual** | **Predicted** | | **Actual** | **Predicted** |
| 57 | 27 | | 87 | 47 | | 89 | 6 |
| 23 | 13 | | 81 | 90 | | 34 | 26 |
| 75 | 63 | | 66 | 89 | | 17 | 53 |
| 35 | 56 | | 53 | 71 | | 20 | 71 |
| 71 | 28 | | 99 | 75 | | 65 | 69 |
| 95 | 88 | | 96 | 52 | | 35 | 90 |
| 97 | 21 | | 4 | 61 | | 26 | 76 |
| 40 | 13 | | 89 | 13 | | 47 | 13 |
| 49 | 50 | | 77 | 33 | | 76 | 72 |
| 66 | 46 | | 28 | 43 | | 2 | 17 |
| 7 | 15 | | 43 | 24 | | 61 | 51 |
| 88 | 23 | | 42 | 93 | | 33 | 95 |
| 53 | 91 | | 59 | 22 | | 54 | 77 |
| 43 | 53 | | 11 | 17 | | 55 | 31 |
| 70 | 72 | | 68 | 30 | | 21 | 39 |
| 87 | 78 | | 19 | 56 | | 3 | 36 |
| 91 | 30 | | 84 | 92 | | 71 | 41 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 10 | 93 | | 35 | 66 | | 49 | 20 |
| 81 | 26 | | 55 | 61 | | 52 | 77 |
| 25 | 35 | | 7 | 44 | | 66 | 0 |



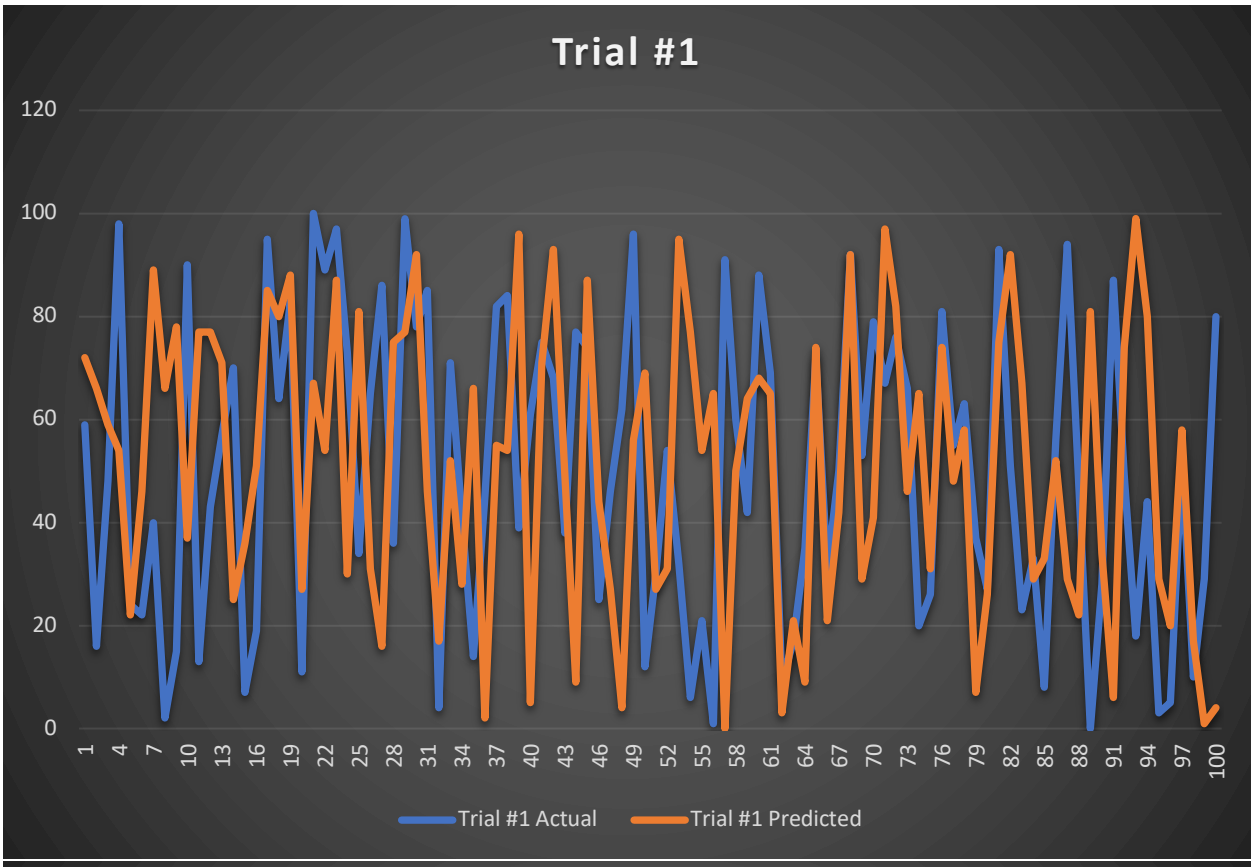

Trial #2
120
100
80
60
40
20
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100
Trial #2 Actual
Trial #2 Predicted

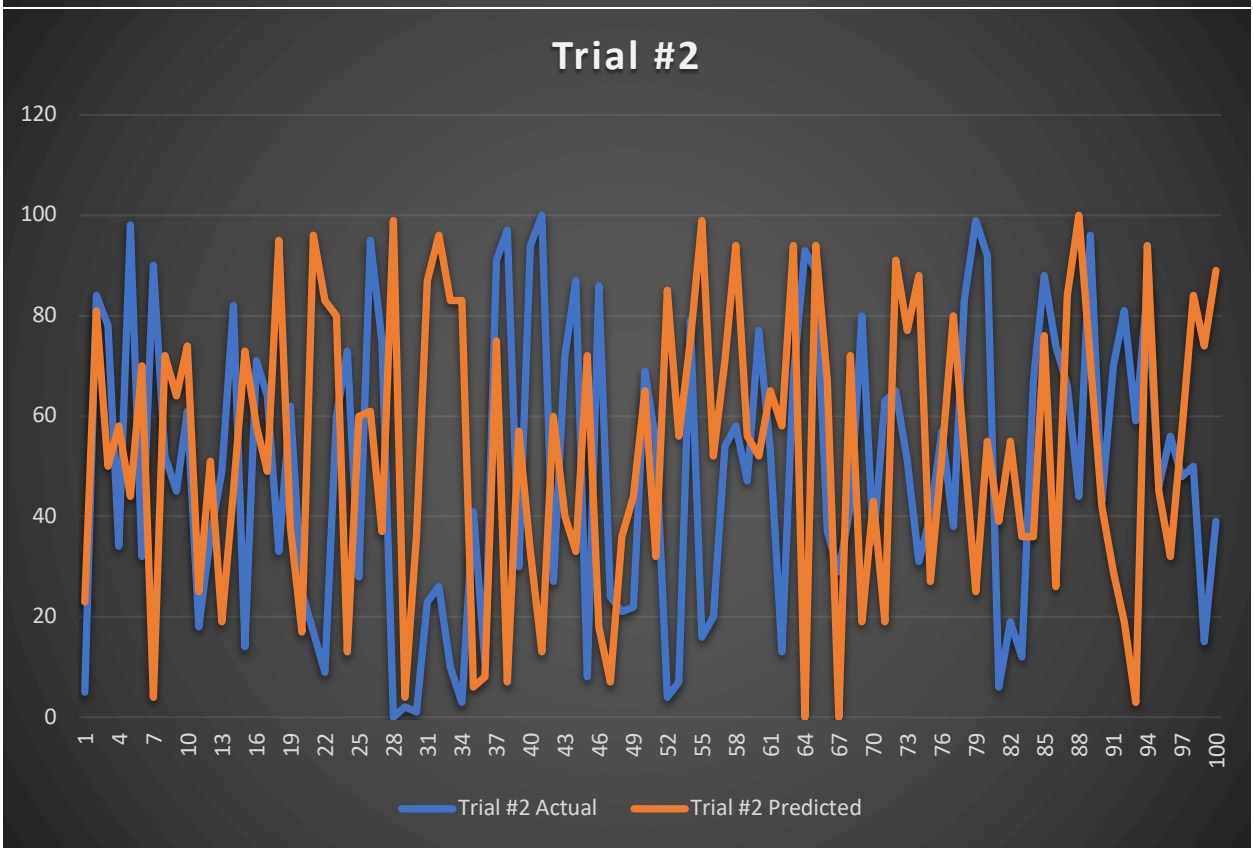

Trial #3
120
100
80
60
40
20
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100
Trial #3 Actual
Trial #3 Predicted

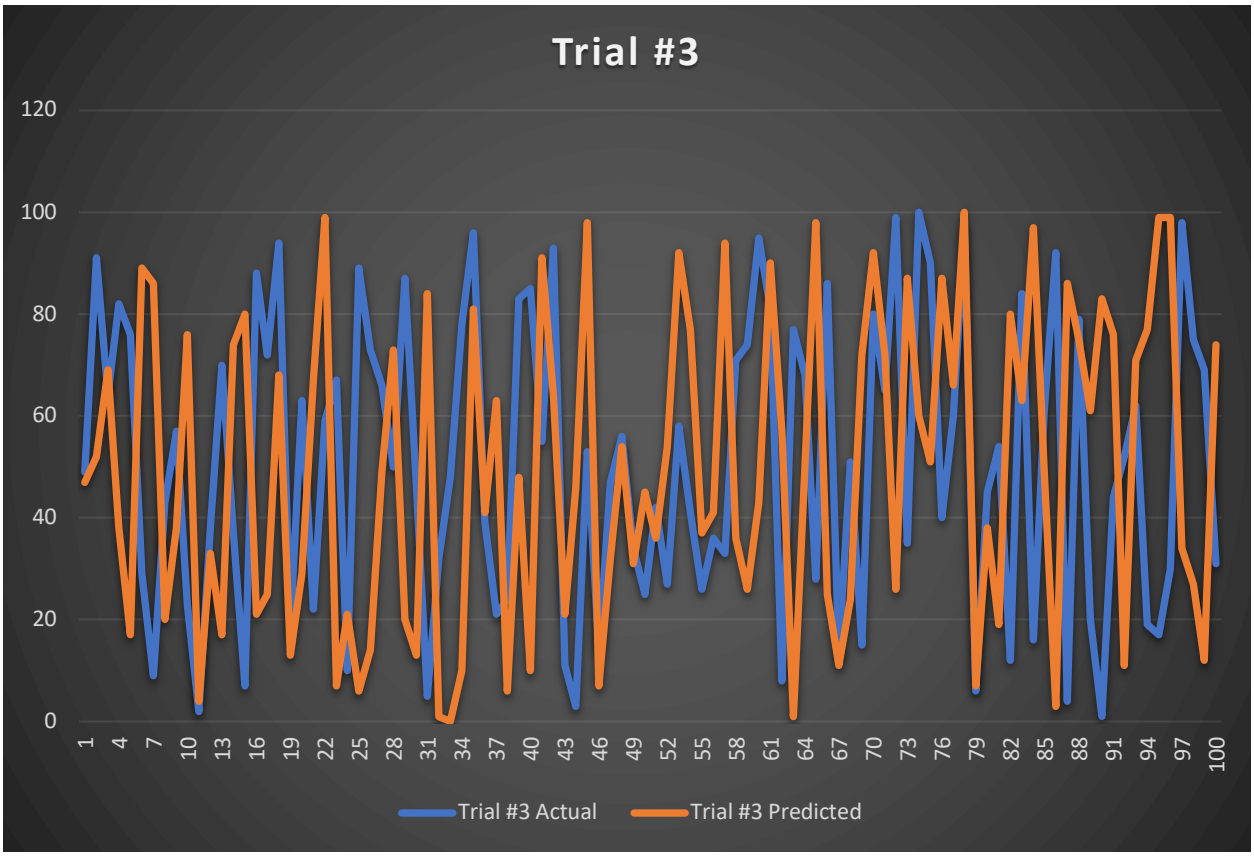
## HotBits

*(Only the initial 20 integers are shown in the table. The full table is available at Appendix E)*

| **Trial #1** | | | **Trial #2** | | | **Trial #3** | |
|---|---|---|---|---|---|---|---|
| **Actual** | **Predicted** | | **Actual** | **Predicted** | | **Actual** | **Predicted** |
| 59 | 72 | | 5 | 23 | | 49 | 47 |
| 16 | 66 | | 84 | 81 | | 91 | 52 |
| 48 | 59 | | 78 | 50 | | 64 | 69 |
| 98 | 54 | | 34 | 58 | | 82 | 38 |
| 24 | 22 | | 98 | 44 | | 76 | 17 |
| 22 | 46 | | 32 | 70 | | 29 | 89 |
| 40 | 89 | | 90 | 4 | | 9 | 86 |
| 2 | 66 | | 52 | 72 | | 43 | 20 |
| 15 | 78 | | 45 | 64 | | 57 | 38 |
| 90 | 37 | | 61 | 74 | | 23 | 76 |
| 13 | 77 | | 18 | 25 | | 2 | 4 |
| 43 | 77 | | 36 | 51 | | 38 | 33 |
| 57 | 71 | | 49 | 19 | | 70 | 17 |
| 70 | 25 | | 82 | 45 | | 37 | 74 |
| 7 | 36 | | 14 | 73 | | 7 | 80 |
| 19 | 51 | | 71 | 58 | | 88 | 21 |
| 95 | 85 | | 64 | 49 | | 72 | 25 |
| 64 | 80 | | 33 | 95 | | 94 | 68 |
| 83 | 88 | | 62 | 38 | | 18 | 13 |
| 11 | 27 | | 25 | 17 | | 63 | 29 |


Trial #1
120
100
80
60
40
20
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100
Trial #1 Actual
Trial #1 Predicted


Trial #2
120
100
80
60
40
20
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100
Trial #2 Actual
Trial #2 Predicted




## Results Discussion

The results clearly demonstrate that TRNGs consistently produce more random and, consequently, more secure sets of integers than PRNGs. Two relatively simple algorithms were able to brute-force the seed values and correctly predict all the values in the sequence. Out of all the PRNGs, not one of them had a single value that one of the algorithms wasn't able to predict. I expected the Java and C++ algorithms to be cracked relatively easily, but the Mersenne Twister is one of the most widely used PRNGs. The research seemed to imply that the Mersenne Twister would not be cracked as easily, due to the fact that it uses 64-bit seed values and the fact that it uses its own proprietary algorithm.

The TRNGs, on the other hand, proved to be significantly and noticeably more secure. Even the initial raw graphs showed a much more random distribution of integers. None of the algorithms was able to crack any of their seed values or predict any integers correctly. This was not unexpected, due to the nature of TRNG algorithms and their seed values. TRNGs typically have much more complex, proprietary, algorithms that are known only to a select group of people. In addition to this, their seed values are determined by measuring physical phenomena like quantum fluctuations in a vacuum (ANU QRNG), or radioactive decay (HotBits).

## Evaluation

Although this investigation has produced tangible, reasonable, and precise results, there were still several limitations that may impact the overall accuracy of the final conclusion. The limitations of this investigation include:

- While this investigation conclusively demonstrated that TRNGs are significantly more secure than commonly used PRNGs, it did not directly measure entropy. Future studies using tests like the Diehard battery could offer a more precise evaluation of randomness.
- The investigation only considered a relatively small range of values. Although 0-100 may initially seem like a reasonable range, in terms of data analysis, during the actual analysis, I found that this range was far too small to produce any obvious patterns. In order to comprehensively analyze the PRNGs, a minimum range of about 0-2^32, or 0-2^48 would be ideal. These ranges would prove that PRNGs produce periodic sequences. Practically speaking, it would be extremely difficult to produce and collect this much data, let alone analyze it.

- Perhaps the biggest limitation of this investigation was the fact that it didn't consider Cryptographically Secure Pseudorandom Number Generators (CSPRNGs). CSPRNGs are very similar to normal PRNGs, except for the fact that they integrate several properties that make them significantly more difficult to crack, and therefore significantly more secure and random. Although, CSPRNGs are not without their flaws. In 2017 cryptographers at The University of Pennsylvania and Johns Hopkins University revealed that it was possible to brute-force hard-coded seed keys used by the WPA2 wireless protocol. Nevertheless, the inclusion of CSPRNGs in this investigation may have pointed to an entirely different conclusion. It is worth noting though that CSPRNGs are not nearly as widespread as PRNGs.

## Conclusion

Conducting the experiment and analyzing the results allowed me to answer and evaluate the original research question. For many scenarios, where security is not a priority or a major concern, PRNGs may be considered a perfectly suitable solution. The data, especially from the graphs, has shown that even the least secure PRNGs at least appear to be random. This means that they can be implemented, without issue, in several applications such as board game apps, the shuffle feature in most music players, or even with in-game loot-boxes. PRNGs are also sometimes used in simulation and modeling applications[17]. These are all applications where security is not the highest priority and the only function that random numbers serve is to give the illusion of randomness. Although all of the generators ran extremely efficiently on my device, not all of them will run as well on older or less powerful hardware. In addition to this, all

of the TRNGs I used ran on servers, which required a stable network connection in order to allow for communication between my computer and the servers. In cases where network connectivity is weak, PRNGs may be the only solution, short of buying or creating custom hardware.

The only applications where TRNGs appear to be absolutely necessary are security-sensitive applications, like when generating lottery numbers, encryption keys, gambling, and computer simulations. However, this is where the limitations of the investigation have an effect on the final conclusion. As I mentioned earlier, the biggest limitation of this investigation has to be the fact that it didn't consider CSPRNGs[16]. These are algorithms specifically designed for these types of applications. We may have found that they are still not as secure as TRNGs, or we may have found that they are just as secure or even possibly more secure. The research seems to suggest that TRNGs would still be more secure, however, we cannot be certain because we haven't tested it. The testing methods didn't leave much room for uncertainty, so the results should be quite reliable and accurate. The data was also quite precise and not very skewed.

---

[16] En.wikipedia.org. n.d. Cryptographically Secure Pseudorandom Number Generator. [online] Available at: <https://en.wikipedia.org/wiki/Cryptographically_secure_pseudorandom_number_generator> [Accessed 16 September 2020].

[17] GeeksforGeeks. n.d. *Pseudo Random Number Generator (PRNG) - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/#:~:text=PRNGs%20are%20suitable%20for%20applications,are%20simulation%20and%20modeling%20applications.> [Accessed 5 December 2020].

PRNGs remain a suitable solution in applications where the goal is to create the appearance of randomness, such as in games or simulations. In these cases, TRNGs offer no functional advantage, whereas their use is essential in security-sensitive applications.

# Bibliography


- 76, E., 2018. *Pseudo Random Number Generator (PRNG) - Geeksforgeeks*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/#:~:text=Pseudo%20Random%20Number%20Generator(PRNG)%20refers%20to%20an%20algorithm%20that,state%20using%20a%20seed%20state> [Accessed 23 June 2020].

- Owen, A., 2013. *Monte Carlo Theory, Methods And Examples*. 1st ed. Stanford: Stanford University, pp.Chapter 3.1.

- En.wikipedia.org. n.d. *Linear Congruential Generator*. [online] Available at: <https://en.wikipedia.org/wiki/Linear_congruential_generator> [Accessed 23 June 2020].

- En.wikipedia.org. n.d. *Lehmer Random Number Generator*. [online] Available at: <https://en.wikipedia.org/wiki/Lehmer_random_number_generator> [Accessed 23 June 2020].

- Woo, E., 2014. *Random Numbers (1 Of 2: True Vs. Pseudo Rngs)*. [video] Available at: <https://www.youtube.com/watch?v=fEWigU1dcp8> [Accessed 23 June 2020].

- Woo, E., 2014. *Random Numbers (2 Of 2: Linear Congruential Generator)*. [image] Available at: <https://www.youtube.com/watch?v=PtEivGPxwAI> [Accessed 23 June 2020].

- Amazon.com. 2016. *Truerng V3 - USB Hardware Random Number Generator*. [online] Available at: <https://www.amazon.com/TrueRNG-V3-Hardware-Random-Generator/dp/B01KR2JHTA> [Accessed 23 June 2020].

- Bitbabbler.org. 2014. *A Hardware RNG We Could Trust - Bitbabbler*. [online] Available at: <http://www.bitbabbler.org/> [Accessed 23 June 2020].

- Cheetam, J., 2019. *Onerng - Hardware Random Number Generator*. [online] Onerng.info. Available at: <https://onerng.info/> [Accessed 23 June 2020].

- Bajaj, T., 2020. Random Setseed() Method In Java With Examples - Geeksforgeeks. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/random-setseed-method-in-java-with-examples/> [Accessed 16 September 2020].

- alternative?, S., Eisfeld, A. and Eisfeld, A., 2016. *Swift - Seeding Arc4random_Uniform? Or Alternative?*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/38679670/swift-seeding-arc4random-uniform-or-alternative> [Accessed 16 September 2020].

- En.wikipedia.org. Undated. Mersenne Twister. [online] Available at: <https://en.wikipedia.org/wiki/Mersenne_Twister> [Accessed 16 September 2020].

- GeeksforGeeks. Undated. Rand() And Srand() In C/C++ - Geeksforgeeks. [online] Available at: <https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/> [Accessed 16 September 2020].

- Logic, P., Løtveit, H. and Gupta, R., 2013. Prime Number Generator Logic. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/20435289/prime-number-generator-logic> [Accessed 16 September 2020].

- Swift, T., hamdy, d., Chiu, K. and garg, A., 2014. Two-Dimensional Array In Swift. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/25127700/two-dimensional-array-in-swift> [Accessed 16 September 2020].

- Developer.apple.com. n.d. Apple Developer Documentation. [online] Available at: <https://developer.apple.com/documentation/swift/array> [Accessed 16 September 2020].

- alternative?, S., Eisfeld, A. and Eisfeld, A., 2016. Swift - Seeding Arc4random_Uniform? Or Alternative?. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/38679670/swift-seeding-arc4random-uniform-or-alternative> [Accessed 16 September 2020].

- Developer.apple.com. n.d. Apple Developer Documentation. [online] Available at: <https://developer.apple.com/documentation/gameplaykit/gkmersennetwisterrandomsource> [Accessed 16 September 2020].

- Ta, F., 2014. Predicting The Next Math.Random() In Java. [online] Franklin Ta. Available at: <https://franklinta.com/2014/08/31/predicting-the-next-math-random-in-java/> [Accessed 16 September 2020].

- 2014. *Replicatedrandom*. GitHub.

- En.wikipedia.org. n.d. Diehard Tests. [online] Available at: <https://en.wikipedia.org/wiki/Diehard_tests> [Accessed 16 September 2020].

- En.wikipedia.org. n.d. Cryptographically Secure Pseudorandom Number Generator. [online] Available at: <https://en.wikipedia.org/wiki/Cryptographically_secure_pseudorandom_number_generator> [Accessed 16 September 2020].

- 2014. Untwister. GitHub.

- GeeksforGeeks. n.d. *Pseudo Random Number Generator (PRNG) - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/#:~:text=PRNGs%20are%20suitable%20for%20applications,are%20simulation%20and%20modeling%20applications.> [Accessed 5 December 2020].

# Appendices

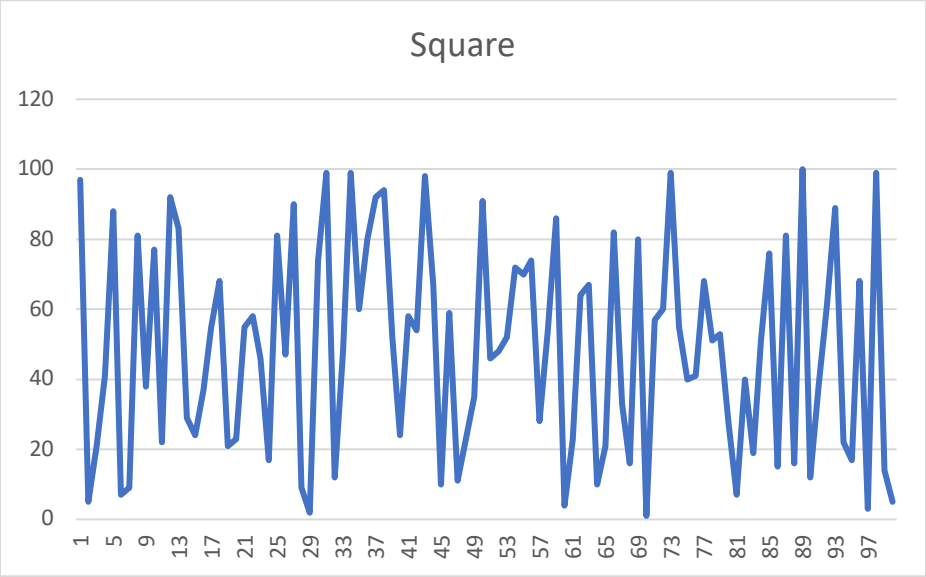
## Appendix A

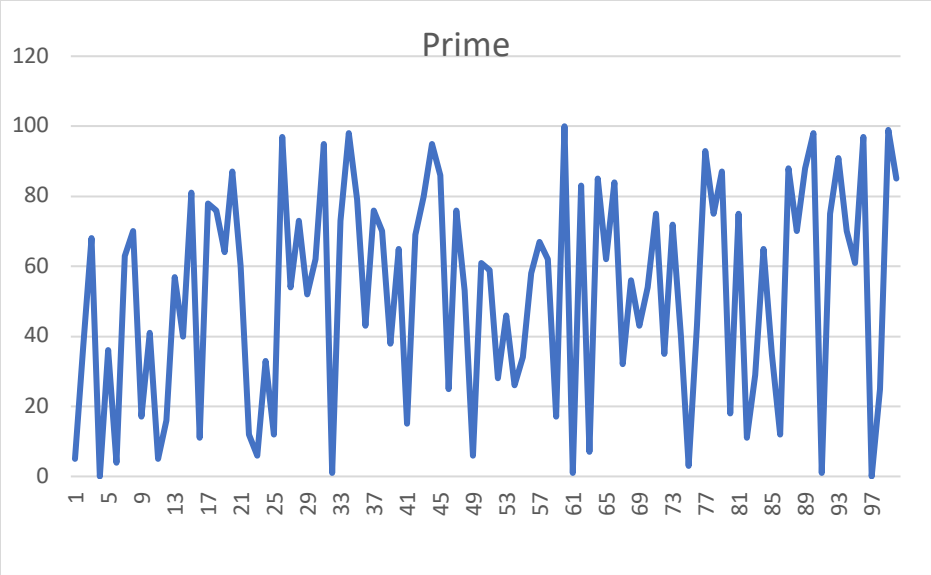
### Java

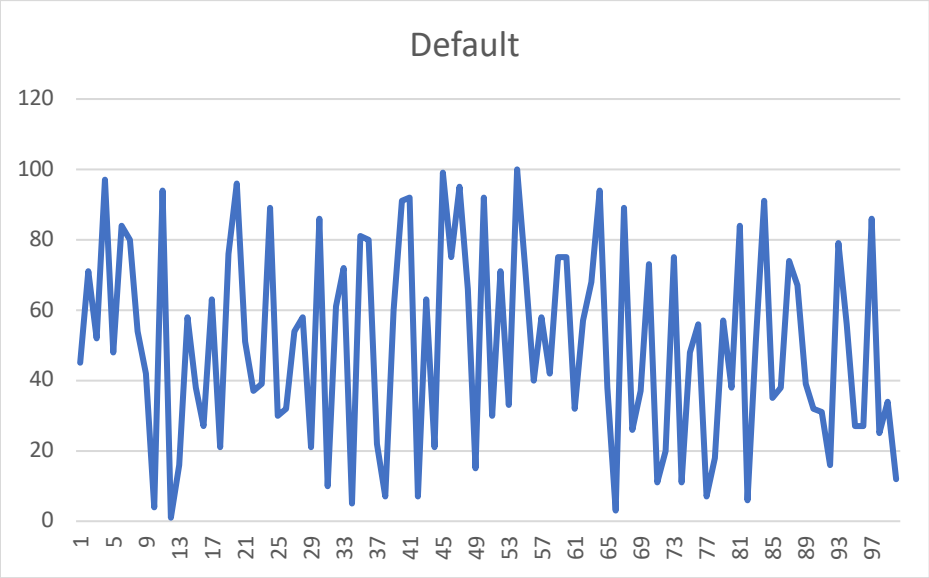
| Default | Prime | Square |
|---|---|---|
| 45 | 5 | 97 |
| 71 | 37 | 5 |
| 52 | 68 | 21 |
| 97 | 0 | 41 |
| 48 | 36 | 88 |
| 84 | 4 | 7 |
| 80 | 63 | 9 |
| 54 | 70 | 81 |
| 42 | 17 | 38 |
| 4 | 41 | 77 |
| 94 | 5 | 22 |
| 1 | 16 | 92 |
| 16 | 57 | 83 |
| 58 | 40 | 29 |
| 38 | 81 | 24 |
| 27 | 11 | 37 |
| 63 | 78 | 55 |
| 21 | 76 | 68 |
| 76 | 64 | 21 |
| 96 | 87 | 23 |
| 51 | 60 | 55 |
| 37 | 12 | 58 |
| 39 | 6 | 46 |
| 89 | 33 | 17 |
| 30 | 12 | 81 |
| 32 | 97 | 47 |
| 54 | 54 | 90 |
| 58 | 73 | 9 |
| 21 | 52 | 2 |
| 86 | 62 | 74 |
| 10 | 95 | 99 |
| 61 | 1 | 12 |
| 72 | 73 | 48 |
| 5 | 98 | 99 |
| 81 | 79 | 60 |
| 80 | 43 | 80 |

| | | |
|---|---|---|
| 22 | 76 | 92 |
| 7 | 70 | 94 |
| 60 | 38 | 52 |
| 91 | 65 | 24 |
| 92 | 15 | 58 |
| 7 | 69 | 54 |
| 63 | 80 | 98 |
| 21 | 95 | 67 |
| 99 | 86 | 10 |
| 75 | 25 | 59 |
| 95 | 76 | 11 |
| 66 | 53 | 23 |
| 15 | 6 | 35 |
| 92 | 61 | 91 |
| 30 | 59 | 46 |
| 71 | 28 | 48 |
| 33 | 46 | 52 |
| 100 | 26 | 72 |
| 71 | 34 | 70 |
| 40 | 58 | 74 |
| 58 | 67 | 28 |
| 42 | 62 | 54 |
| 75 | 17 | 86 |
| 75 | 100 | 4 |
| 32 | 1 | 23 |
| 57 | 83 | 64 |
| 68 | 7 | 67 |
| 94 | 85 | 10 |
| 38 | 62 | 21 |
| 3 | 84 | 82 |
| 89 | 32 | 33 |
| 26 | 56 | 16 |
| 37 | 43 | 80 |
| 73 | 54 | 1 |
| 11 | 75 | 57 |
| 20 | 35 | 60 |
| 75 | 72 | 99 |
| 11 | 40 | 55 |
| 48 | 3 | 40 |
| 56 | 43 | 41 |
| 7 | 93 | 68 |
| 18 | 75 | 51 |
| 57 | 87 | 53 |

| | | |
|---|---|---|
| 38 | 18 | 28 |
| 84 | 75 | 7 |
| 6 | 11 | 40 |
| 51 | 29 | 19 |
| 91 | 65 | 51 |
| 35 | 35 | 76 |
| 38 | 12 | 15 |
| 74 | 88 | 81 |
| 67 | 70 | 16 |
| 39 | 88 | 100 |
| 32 | 98 | 12 |
| 31 | 1 | 38 |
| 16 | 75 | 61 |
| 79 | 91 | 89 |
| 56 | 70 | 22 |
| 27 | 61 | 17 |
| 27 | 97 | 68 |
| 86 | 0 | 3 |
| 25 | 25 | 99 |
| 34 | 99 | 14 |
| 12 | 85 | 5 |





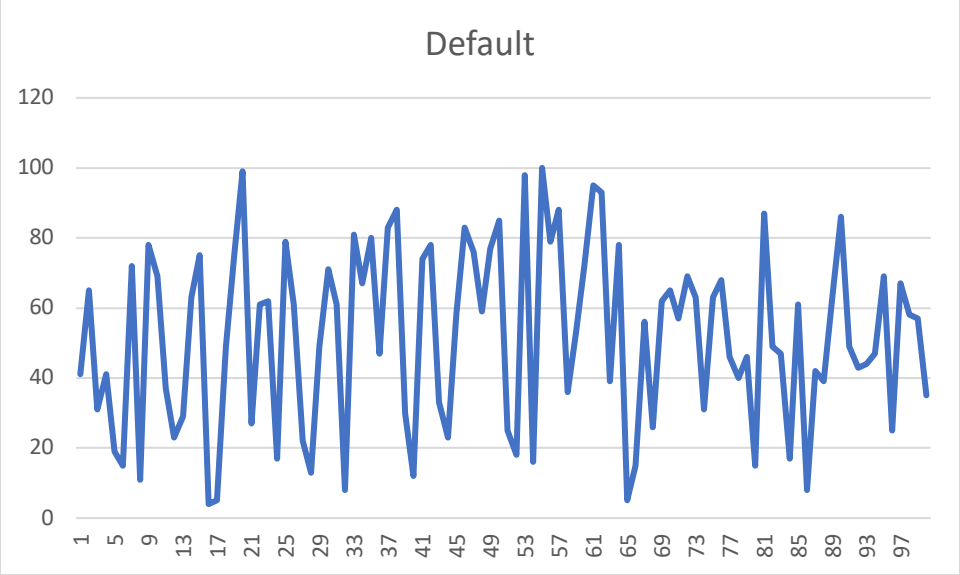

C++

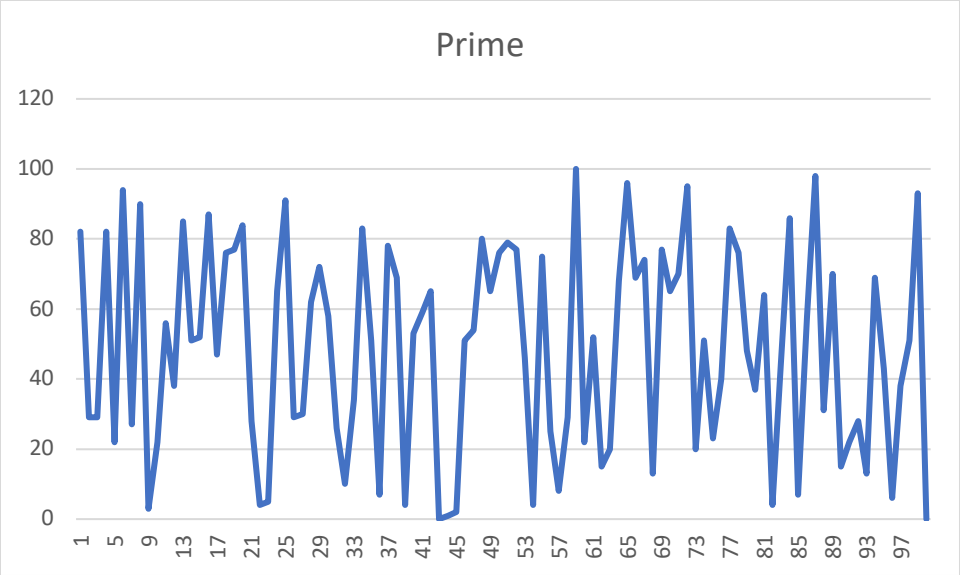
| Default | Prime | Square |
|---|---|---|
| 41 | 82 | 41 |
| 65 | 29 | 65 |
| 31 | 29 | 31 |
| 41 | 82 | 41 |
| 19 | 22 | 63 |
| 15 | 94 | 58 |
| 72 | 27 | 25 |
| 11 | 90 | 30 |

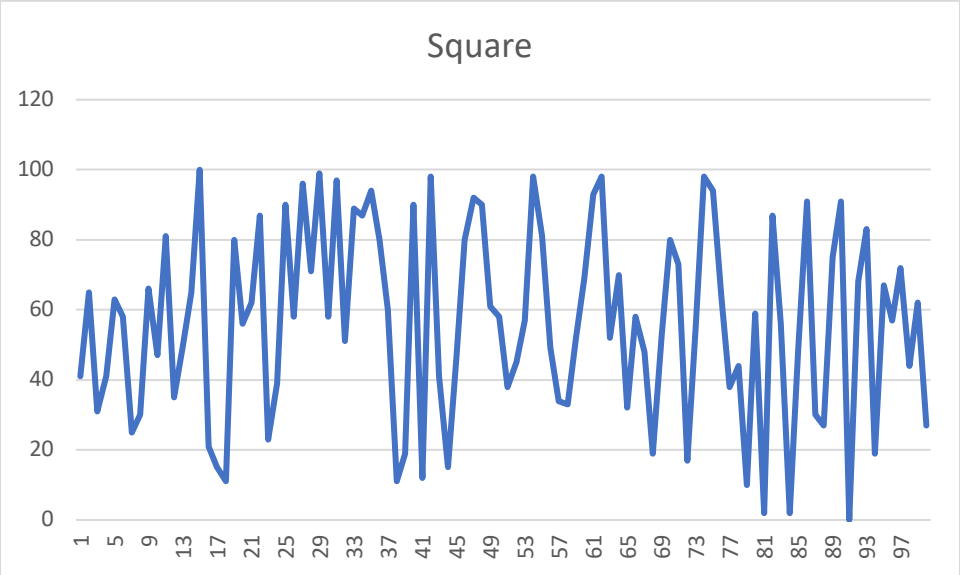
| | | |
|---|---|---|
| 78 | 3 | 66 |
| 69 | 22 | 47 |
| 37 | 56 | 81 |
| 23 | 38 | 35 |
| 29 | 85 | 50 |
| 63 | 51 | 65 |
| 75 | 52 | 100 |
| 4 | 87 | 21 |
| 5 | 47 | 15 |
| 49 | 76 | 11 |
| 75 | 77 | 80 |
| 99 | 84 | 56 |
| 27 | 28 | 62 |
| 61 | 4 | 87 |
| 62 | 5 | 23 |
| 17 | 65 | 39 |
| 79 | 91 | 90 |
| 61 | 29 | 58 |
| 22 | 30 | 96 |
| 13 | 62 | 71 |
| 49 | 72 | 99 |
| 71 | 58 | 58 |
| 61 | 26 | 97 |
| 8 | 10 | 51 |
| 81 | 34 | 89 |
| 67 | 83 | 87 |
| 80 | 51 | 94 |
| 47 | 7 | 80 |
| 83 | 78 | 60 |
| 88 | 69 | 11 |
| 30 | 4 | 19 |
| 12 | 53 | 90 |
| 74 | 59 | 12 |
| 78 | 65 | 98 |
| 33 | 0 | 41 |
| 23 | 1 | 15 |
| 58 | 2 | 46 |
| 83 | 51 | 80 |
| 76 | 54 | 92 |

| | | |
|---|---|---|
| 59 | 80 | 90 |
| 77 | 65 | 61 |
| 85 | 76 | 58 |
| 25 | 79 | 38 |
| 18 | 77 | 45 |
| 98 | 46 | 57 |
| 16 | 4 | 98 |
| 100 | 75 | 81 |
| 79 | 25 | 49 |
| 88 | 8 | 34 |
| 36 | 29 | 33 |
| 53 | 100 | 52 |
| 72 | 22 | 69 |
| 95 | 52 | 93 |
| 93 | 15 | 98 |
| 39 | 20 | 52 |
| 78 | 68 | 70 |
| 5 | 96 | 32 |
| 15 | 69 | 58 |
| 56 | 74 | 48 |
| 26 | 13 | 19 |
| 62 | 77 | 53 |
| 65 | 65 | 80 |
| 57 | 70 | 73 |
| 69 | 95 | 17 |
| 63 | 20 | 55 |
| 31 | 51 | 98 |
| 63 | 23 | 94 |
| 68 | 40 | 64 |
| 46 | 83 | 38 |
| 40 | 76 | 44 |
| 46 | 48 | 10 |
| 15 | 37 | 59 |
| 87 | 64 | 2 |
| 49 | 4 | 87 |
| 47 | 44 | 56 |
| 17 | 86 | 2 |
| 61 | 7 | 48 |
| 8 | 58 | 91 |

| | | |
|---|---|---|
| 42 | 98 | 30 |
| 39 | 31 | 27 |
| 64 | 70 | 75 |
| 86 | 15 | 91 |
| 49 | 22 | 0 |
| 43 | 28 | 68 |
| 44 | 13 | 83 |
| 47 | 69 | 19 |
| 69 | 43 | 67 |
| 25 | 6 | 57 |
| 67 | 38 | 72 |
| 58 | 51 | 44 |
| 57 | 93 | 62 |
| 35 | 0 | 27 |





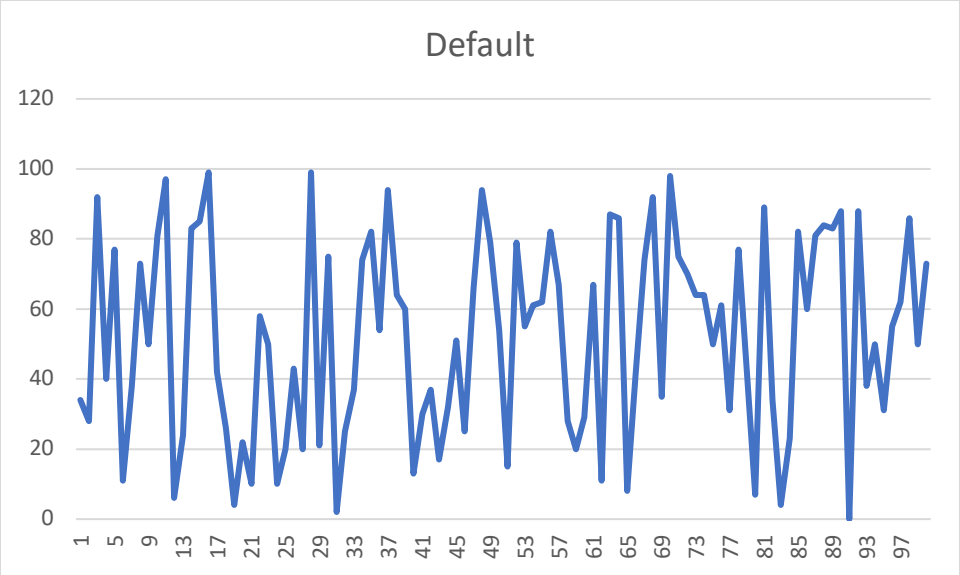

## Swift

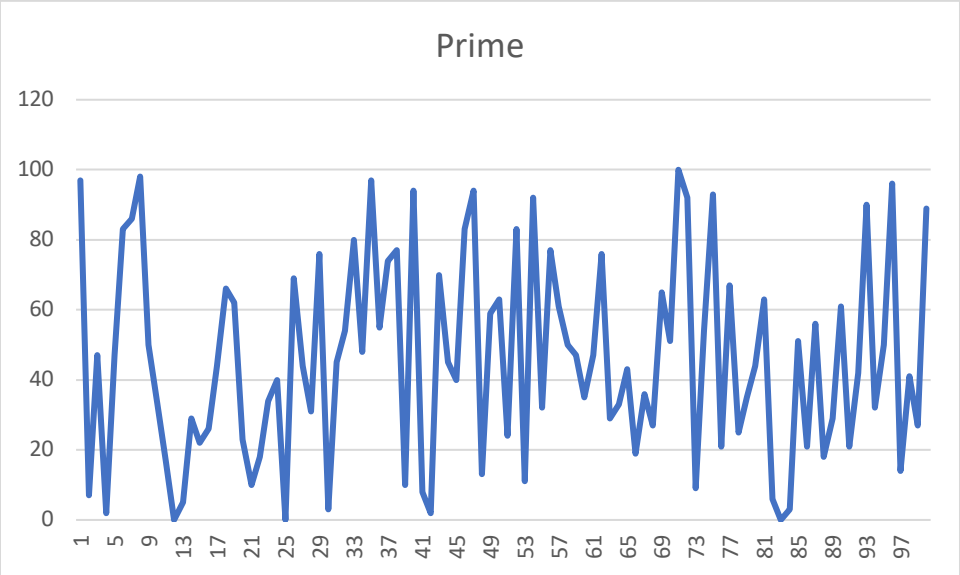
| Default | Prime | Square |
|---|---|---|
| 34 | 97 | 80 |
| 28 | 7 | 29 |
| 92 | 47 | 100 |
| 40 | 2 | 30 |
| 77 | 47 | 22 |
| 11 | 83 | 60 |
| 38 | 86 | 52 |

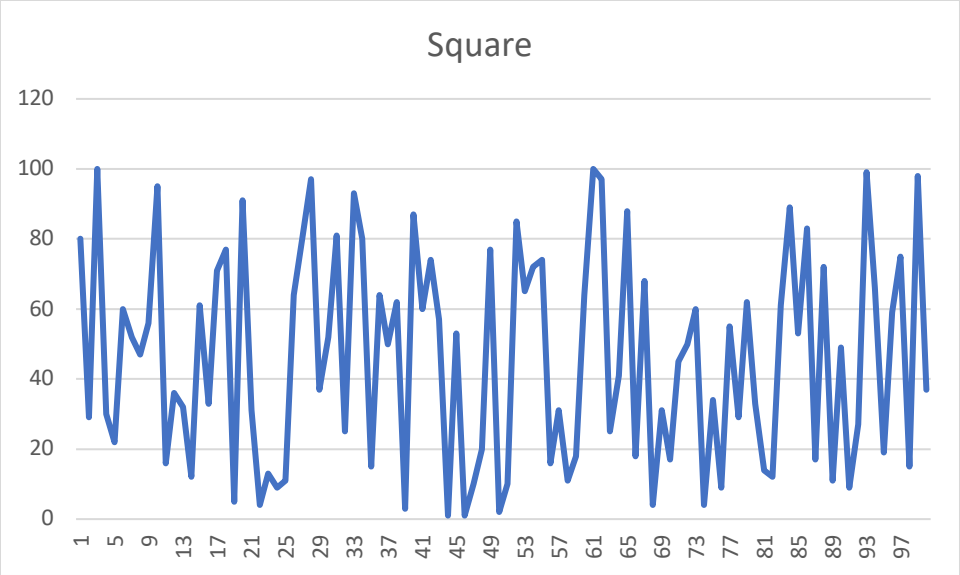
| | | |
|---|---|---|
| 73 | 98 | 47 |
| 50 | 50 | 56 |
| 81 | 33 | 95 |
| 97 | 17 | 16 |
| 6 | 0 | 36 |
| 24 | 5 | 32 |
| 83 | 29 | 12 |
| 85 | 22 | 61 |
| 99 | 26 | 33 |
| 42 | 44 | 71 |
| 26 | 66 | 77 |
| 4 | 62 | 5 |
| 22 | 23 | 91 |
| 10 | 10 | 31 |
| 58 | 18 | 4 |
| 50 | 34 | 13 |
| 10 | 40 | 9 |
| 20 | 0 | 11 |
| 43 | 69 | 64 |
| 20 | 44 | 81 |
| 99 | 31 | 97 |
| 21 | 76 | 37 |
| 75 | 3 | 52 |
| 2 | 45 | 81 |
| 25 | 54 | 25 |
| 37 | 80 | 93 |
| 74 | 48 | 80 |
| 82 | 97 | 15 |
| 54 | 55 | 64 |
| 94 | 74 | 50 |
| 64 | 77 | 62 |
| 60 | 10 | 3 |
| 13 | 94 | 87 |
| 30 | 8 | 60 |
| 37 | 2 | 74 |
| 17 | 70 | 57 |
| 32 | 45 | 1 |
| 51 | 40 | 53 |
| 25 | 83 | 1 |

| | | |
|---|---|---|
| 66 | 94 | 10 |
| 94 | 13 | 20 |
| 79 | 59 | 77 |
| 54 | 63 | 2 |
| 15 | 24 | 10 |
| 79 | 83 | 85 |
| 55 | 11 | 65 |
| 61 | 92 | 72 |
| 62 | 32 | 74 |
| 82 | 77 | 16 |
| 67 | 61 | 31 |
| 28 | 50 | 11 |
| 20 | 47 | 18 |
| 29 | 35 | 64 |
| 67 | 47 | 100 |
| 11 | 76 | 97 |
| 87 | 29 | 25 |
| 86 | 33 | 41 |
| 8 | 43 | 88 |
| 41 | 19 | 18 |
| 74 | 36 | 68 |
| 92 | 27 | 4 |
| 35 | 65 | 31 |
| 98 | 51 | 17 |
| 75 | 100 | 45 |
| 70 | 92 | 50 |
| 64 | 9 | 60 |
| 64 | 54 | 4 |
| 50 | 93 | 34 |
| 61 | 21 | 9 |
| 31 | 67 | 55 |
| 77 | 25 | 29 |
| 42 | 35 | 62 |
| 7 | 44 | 33 |
| 89 | 63 | 14 |
| 34 | 6 | 12 |
| 4 | 0 | 61 |
| 23 | 3 | 89 |
| 82 | 51 | 53 |

| | | |
|---|---|---|
| 60 | 21 | 83 |
| 81 | 56 | 17 |
| 84 | 18 | 72 |
| 83 | 29 | 11 |
| 88 | 61 | 49 |
| 0 | 21 | 9 |
| 88 | 42 | 27 |
| 38 | 90 | 99 |
| 50 | 32 | 66 |
| 31 | 50 | 19 |
| 55 | 96 | 59 |
| 62 | 14 | 75 |
| 86 | 41 | 15 |
| 50 | 27 | 98 |
| 73 | 89 | 37 |


Default
120
100
80
60
40
20
0
1
5
9
13
17
21
25
29
33
37
41
45
49
53
57
61
65
69
73
77
81
85
89
93
97


Prime
120
100
80
60
40
20
0
1
5
9
13
17
21
25
29
33
37
41
45
49
53
57
61
65
69
73
77
81
85
89
93
97




## Appendix B

| Trial #1 | | |
|---|---|---|
| **Random.org** | **ANU QRNG** | **HotBits** |
| 30 | 57 | 59 |
| 71 | 23 | 16 |
| 75 | 75 | 48 |
| 4 | 35 | 98 |
| 39 | 71 | 24 |
| 35 | 95 | 22 |
| 9 | 97 | 40 |
| 42 | 40 | 2 |
| 77 | 49 | 15 |
| 85 | 66 | 90 |
| 63 | 7 | 13 |
| 6 | 88 | 43 |
| 45 | 53 | 57 |
| 98 | 43 | 70 |
| 48 | 70 | 7 |
| 91 | 87 | 19 |
| 80 | 91 | 95 |
| 36 | 10 | 64 |
| 37 | 81 | 83 |
| 28 | 25 | 11 |

| | | |
|---|---|---|
| 76 | 74 | 100 |
| 49 | 72 | 89 |
| 14 | 79 | 97 |
| 86 | 19 | 73 |
| 0 | 32 | 34 |
| 55 | 78 | 65 |
| 5 | 80 | 86 |
| 54 | 16 | 36 |
| 10 | 9 | 99 |
| 84 | 93 | 78 |
| 20 | 89 | 85 |
| 66 | 24 | 4 |
| 22 | 3 | 71 |
| 24 | 5 | 41 |
| 87 | 13 | 14 |
| 52 | 2 | 45 |
| 74 | 34 | 82 |
| 11 | 61 | 84 |
| 88 | 38 | 39 |
| 12 | 48 | 61 |
| 83 | 62 | 75 |
| 13 | 41 | 68 |
| 47 | 94 | 38 |
| 7 | 86 | 77 |
| 27 | 67 | 74 |
| 56 | 84 | 25 |
| 61 | 100 | 46 |
| 95 | 28 | 62 |
| 46 | 51 | 96 |
| 78 | 33 | 12 |
| 67 | 69 | 30 |
| 17 | 92 | 54 |
| 99 | 56 | 32 |
| 60 | 47 | 6 |
| 38 | 26 | 21 |
| 68 | 42 | 1 |
| 92 | 17 | 91 |
| 59 | 63 | 60 |
| 93 | 64 | 42 |

| | | |
|---|---|---|
| 44 | 30 | 88 |
| 50 | 55 | 69 |
| 31 | 68 | 9 |
| 40 | 99 | 17 |
| 32 | 46 | 35 |
| 16 | 15 | 72 |
| 15 | 59 | 31 |
| 64 | 85 | 50 |
| 41 | 12 | 92 |
| 1 | 83 | 53 |
| 90 | 1 | 79 |
| 19 | 22 | 67 |
| 89 | 0 | 76 |
| 57 | 4 | 66 |
| 62 | 76 | 20 |
| 51 | 52 | 26 |
| 97 | 58 | 81 |
| 53 | 6 | 55 |
| 2 | 21 | 63 |
| 43 | 29 | 37 |
| 69 | 31 | 27 |
| 25 | 27 | 93 |
| 82 | 60 | 51 |
| 23 | 8 | 23 |
| 34 | 44 | 33 |
| 26 | 18 | 8 |
| 72 | 90 | 56 |
| 96 | 77 | 94 |
| 65 | 50 | 47 |
| 100 | 96 | 0 |
| 58 | 20 | 28 |
| 73 | 45 | 87 |
| 79 | 98 | 49 |
| 21 | 37 | 18 |
| 33 | 11 | 44 |
| 8 | 82 | 3 |
| 81 | 36 | 5 |
| 3 | 39 | 52 |
| 70 | 73 | 10 |

| | | |
|---|---|---|
| 94 | 14 | 29 |
| 29 | 65 | 80 |

| **Trial #2** | | |
|---|---|---|
| **Random.org** | **ANU QRNG** | **HotBits** |
| 99 | 87 | 5 |
| 58 | 81 | 84 |
| 54 | 66 | 78 |
| 55 | 53 | 34 |
| 88 | 99 | 98 |
| 29 | 96 | 32 |
| 34 | 4 | 90 |
| 47 | 89 | 52 |
| 66 | 77 | 45 |
| 26 | 28 | 61 |
| 9 | 43 | 18 |
| 59 | 42 | 36 |
| 31 | 59 | 49 |
| 95 | 11 | 82 |
| 71 | 68 | 14 |
| 91 | 19 | 71 |
| 86 | 84 | 64 |
| 11 | 35 | 33 |
| 97 | 55 | 62 |
| 28 | 7 | 25 |
| 42 | 24 | 17 |
| 94 | 78 | 9 |
| 36 | 61 | 60 |
| 75 | 74 | 73 |
| 79 | 76 | 28 |
| 57 | 21 | 95 |
| 81 | 29 | 75 |
| 18 | 51 | 0 |
| 56 | 1 | 2 |
| 27 | 94 | 1 |
| 80 | 75 | 23 |
| 87 | 5 | 26 |
| 5 | 15 | 10 |
| 3 | 64 | 3 |

| | | |
|---|---|---|
| 2 | 79 | 41 |
| 70 | 26 | 11 |
| 45 | 86 | 91 |
| 44 | 3 | 97 |
| 65 | 46 | 30 |
| 14 | 72 | 94 |
| 7 | 73 | 100 |
| 25 | 60 | 27 |
| 17 | 30 | 72 |
| 96 | 98 | 87 |
| 41 | 0 | 8 |
| 15 | 54 | 86 |
| 63 | 62 | 24 |
| 52 | 16 | 21 |
| 33 | 2 | 22 |
| 93 | 65 | 69 |
| 49 | 20 | 55 |
| 90 | 52 | 4 |
| 64 | 38 | 7 |
| 85 | 95 | 79 |
| 60 | 58 | 16 |
| 19 | 48 | 20 |
| 84 | 37 | 54 |
| 22 | 23 | 58 |
| 46 | 93 | 47 |
| 100 | 50 | 77 |
| 78 | 18 | 53 |
| 89 | 97 | 13 |
| 92 | 32 | 68 |
| 38 | 88 | 93 |
| 68 | 36 | 89 |
| 83 | 67 | 37 |
| 48 | 44 | 29 |
| 10 | 70 | 42 |
| 13 | 90 | 80 |
| 16 | 49 | 35 |
| 30 | 39 | 63 |
| 8 | 12 | 65 |
| 50 | 33 | 51 |

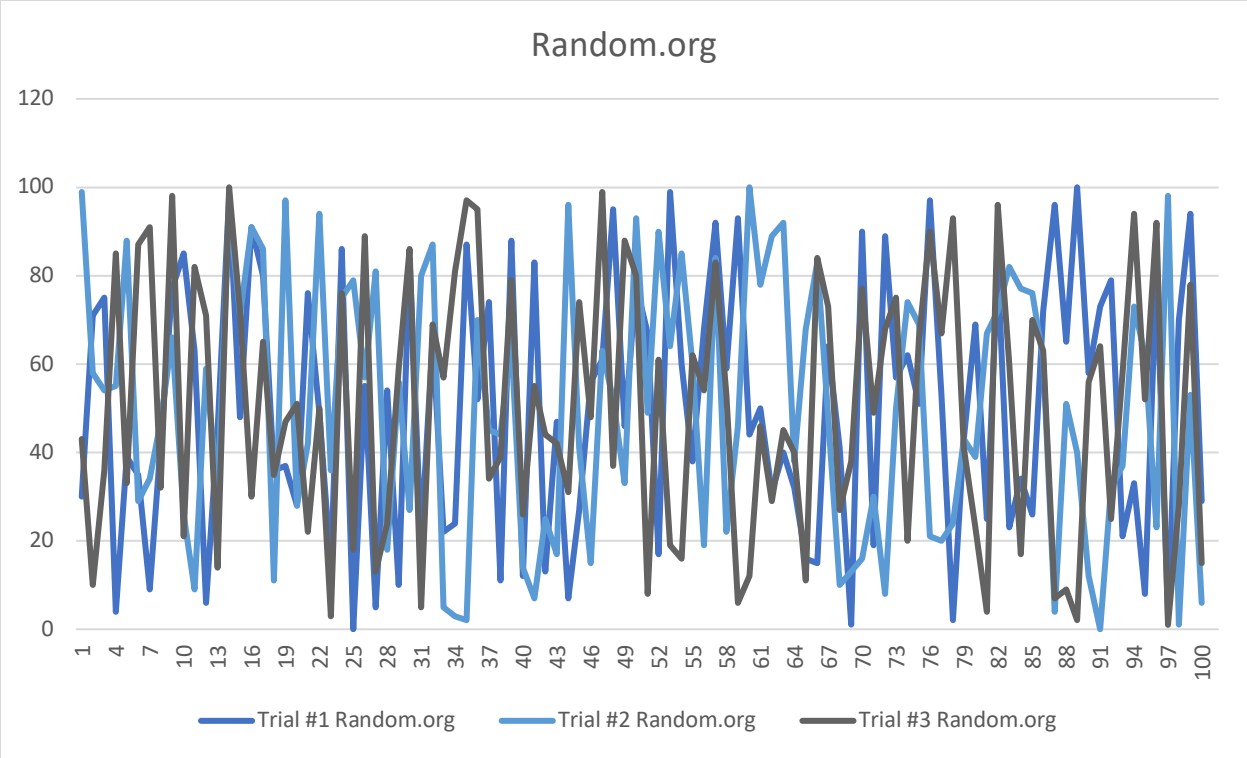
| | | |
|---|---|---|
| 74 | 91 | 31 |
| 69 | 8 | 40 |
| 21 | 56 | 57 |
| 20 | 25 | 38 |
| 24 | 40 | 83 |
| 43 | 17 | 99 |
| 39 | 69 | 92 |
| 67 | 6 | 6 |
| 72 | 10 | 19 |
| 82 | 27 | 12 |
| 77 | 100 | 67 |
| 76 | 83 | 88 |
| 61 | 47 | 74 |
| 4 | 92 | 66 |
| 51 | 41 | 44 |
| 40 | 80 | 96 |
| 12 | 34 | 43 |
| 0 | 22 | 70 |
| 32 | 31 | 81 |
| 37 | 57 | 59 |
| 73 | 14 | 85 |
| 62 | 45 | 46 |
| 23 | 13 | 56 |
| 98 | 71 | 48 |
| 1 | 63 | 50 |
| 53 | 82 | 15 |
| 6 | 9 | 39 |

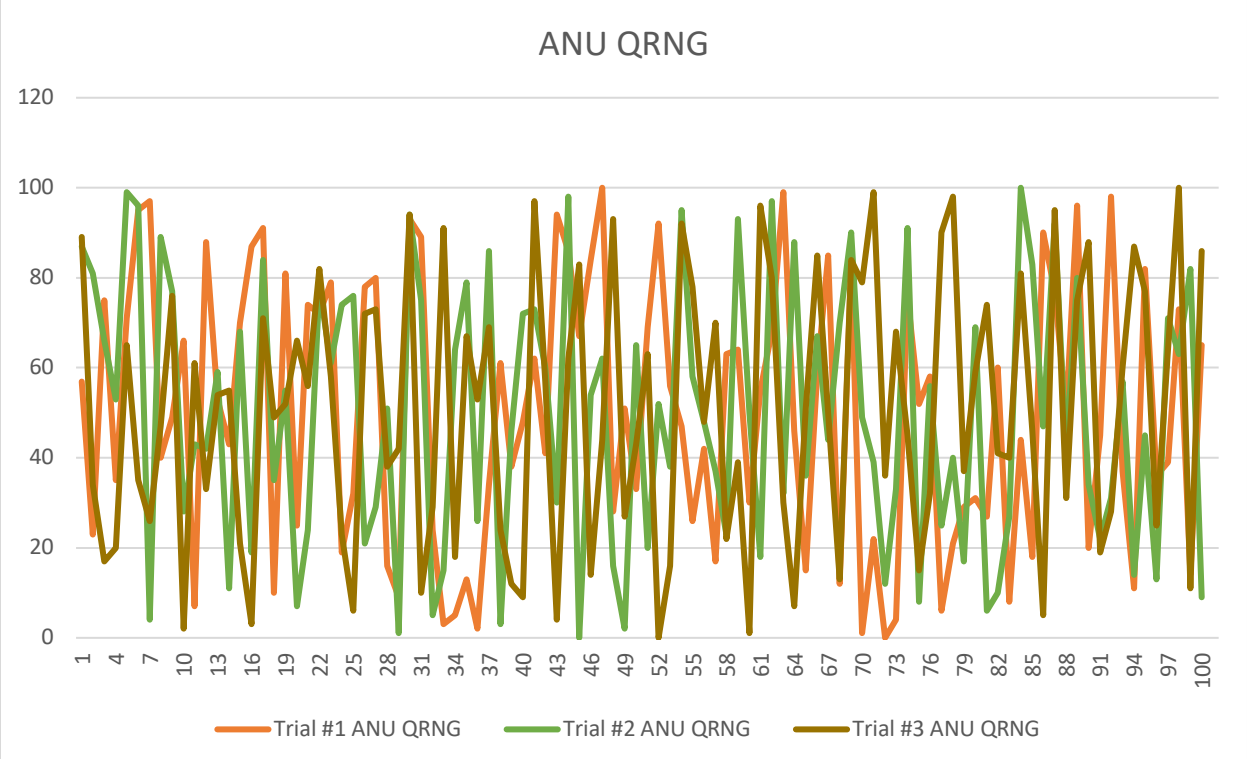
| **Trial #3** | | |
|---|---|---|
| **Random.org** | **ANU QRNG** | **HotBits** |
| 43 | 89 | 49 |
| 10 | 34 | 91 |
| 36 | 17 | 64 |
| 85 | 20 | 82 |
| 33 | 65 | 76 |
| 87 | 35 | 29 |
| 91 | 26 | 9 |
| 32 | 47 | 43 |

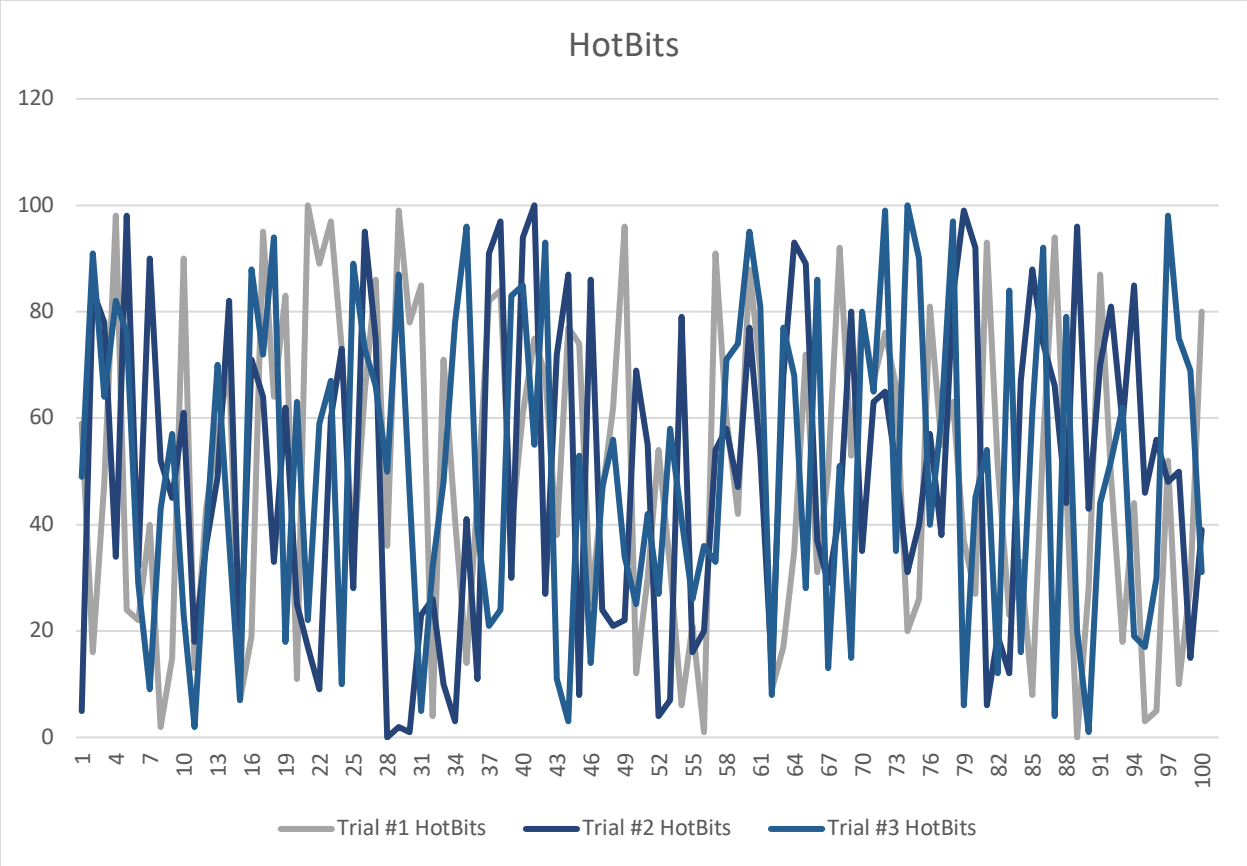
| | | |
|---|---|---|
| 98 | 76 | 57 |
| 21 | 2 | 23 |
| 82 | 61 | 2 |
| 71 | 33 | 38 |
| 14 | 54 | 70 |
| 100 | 55 | 37 |
| 72 | 21 | 7 |
| 30 | 3 | 88 |
| 65 | 71 | 72 |
| 35 | 49 | 94 |
| 47 | 52 | 18 |
| 51 | 66 | 63 |
| 22 | 56 | 22 |
| 50 | 82 | 59 |
| 3 | 58 | 67 |
| 76 | 23 | 10 |
| 18 | 6 | 89 |
| 89 | 72 | 73 |
| 13 | 73 | 66 |
| 24 | 38 | 50 |
| 59 | 42 | 87 |
| 86 | 94 | 46 |
| 5 | 10 | 5 |
| 69 | 29 | 32 |
| 57 | 91 | 48 |
| 81 | 18 | 78 |
| 97 | 67 | 96 |
| 95 | 53 | 39 |
| 34 | 69 | 21 |
| 39 | 24 | 24 |
| 79 | 12 | 83 |
| 26 | 9 | 85 |
| 55 | 97 | 55 |
| 44 | 57 | 93 |
| 42 | 4 | 11 |
| 31 | 62 | 3 |
| 74 | 83 | 53 |
| 48 | 14 | 14 |
| 99 | 44 | 47 |

| | | |
|---|---|---|
| 37 | 93 | 56 |
| 88 | 27 | 34 |
| 80 | 43 | 25 |
| 8 | 63 | 42 |
| 61 | 0 | 27 |
| 19 | 16 | 58 |
| 16 | 92 | 41 |
| 62 | 78 | 26 |
| 54 | 48 | 36 |
| 83 | 70 | 33 |
| 53 | 22 | 71 |
| 6 | 39 | 74 |
| 12 | 1 | 95 |
| 46 | 96 | 81 |
| 29 | 80 | 8 |
| 45 | 30 | 77 |
| 40 | 7 | 68 |
| 11 | 51 | 28 |
| 84 | 85 | 86 |
| 73 | 50 | 13 |
| 27 | 13 | 51 |
| 38 | 84 | 15 |
| 77 | 79 | 80 |
| 49 | 99 | 65 |
| 68 | 36 | 99 |
| 75 | 68 | 35 |
| 20 | 45 | 100 |
| 66 | 15 | 90 |
| 90 | 32 | 40 |
| 67 | 90 | 60 |
| 93 | 98 | 97 |
| 41 | 37 | 6 |
| 23 | 59 | 45 |
| 4 | 74 | 54 |
| 96 | 41 | 12 |
| 60 | 40 | 84 |
| 17 | 81 | 16 |
| 70 | 46 | 61 |
| 63 | 5 | 92 |

| | | |
|---|---|---|
| 7 | 95 | 4 |
| 9 | 31 | 79 |
| 2 | 75 | 20 |
| 56 | 88 | 1 |
| 64 | 19 | 44 |
| 25 | 28 | 52 |
| 58 | 60 | 62 |
| 94 | 87 | 19 |
| 52 | 77 | 17 |
| 92 | 25 | 30 |
| 1 | 64 | 98 |
| 28 | 100 | 75 |
| 78 | 11 | 69 |
| 15 | 86 | 31 |


Random.org
120
100
80
60
40
20
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100
Trial #1 Random.org
Trial #2 Random.org
Trial #3 Random.org


ANU QRNG
120
100
80
60
40
20
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100
Trial #1 ANU QRNG
Trial #2 ANU QRNG
Trial #3 ANU QRNG




## Appendix C

The program was too large to show here. It is available at the following link:

https://drive.google.com/drive/folders/1U3qhGfhJYWRKcnpXKeaUyEaidvK19Ox3?usp=sharing

The original program has been referenced in the bibliography.

## Appendix D

### Java

#### Prime Seeds

| Actual | Predicted |
|---|---|
| 5 | 5 |
| 37 | 37 |
| 68 | 68 |
| 0 | 0 |

| | |
|---|---|
| 36 | 36 |
| 4 | 4 |
| 63 | 63 |
| 70 | 70 |
| 17 | 17 |
| 41 | 41 |
| 5 | 5 |
| 16 | 16 |
| 57 | 57 |
| 40 | 40 |
| 81 | 81 |
| 11 | 11 |
| 78 | 78 |
| 76 | 76 |
| 64 | 64 |
| 87 | 87 |
| 60 | 60 |
| 12 | 12 |
| 6 | 6 |
| 33 | 33 |
| 12 | 12 |
| 97 | 97 |
| 54 | 54 |
| 73 | 73 |
| 52 | 52 |
| 62 | 62 |
| 95 | 95 |
| 1 | 1 |
| 73 | 73 |
| 98 | 98 |
| 79 | 79 |
| 43 | 43 |
| 76 | 76 |
| 70 | 70 |
| 38 | 38 |
| 65 | 65 |
| 15 | 15 |
| 69 | 69 |
| 80 | 80 |

| | |
|---|---|
| 95 | 95 |
| 86 | 86 |
| 25 | 25 |
| 76 | 76 |
| 53 | 53 |
| 6 | 6 |
| 61 | 61 |
| 59 | 59 |
| 28 | 28 |
| 46 | 46 |
| 26 | 26 |
| 34 | 34 |
| 58 | 58 |
| 67 | 67 |
| 62 | 62 |
| 17 | 17 |
| 100 | 100 |
| 1 | 1 |
| 83 | 83 |
| 7 | 7 |
| 85 | 85 |
| 62 | 62 |
| 84 | 84 |
| 32 | 32 |
| 56 | 56 |
| 43 | 43 |
| 54 | 54 |
| 75 | 75 |
| 35 | 35 |
| 72 | 72 |
| 40 | 40 |
| 3 | 3 |
| 43 | 43 |
| 93 | 93 |
| 75 | 75 |
| 87 | 87 |
| 18 | 18 |
| 75 | 75 |
| 11 | 11 |

| | |
|---|---|
| 29 | 29 |
| 65 | 65 |
| 35 | 35 |
| 12 | 12 |
| 88 | 88 |
| 70 | 70 |
| 88 | 88 |
| 98 | 98 |
| 1 | 1 |
| 75 | 75 |
| 91 | 91 |
| 70 | 70 |
| 61 | 61 |
| 97 | 97 |
| 0 | 0 |
| 25 | 25 |
| 99 | 99 |
| 85 | 85 |

## Square Seeds

| **Actual** | **Predicted** |
|---|---|
| 97 | 97 |
| 5 | 5 |
| 21 | 21 |
| 41 | 41 |
| 88 | 88 |
| 7 | 7 |
| 9 | 9 |
| 81 | 81 |
| 38 | 38 |
| 77 | 77 |
| 22 | 22 |
| 92 | 92 |
| 83 | 83 |
| 29 | 29 |
| 24 | 24 |
| 37 | 37 |
| 55 | 55 |
| 68 | 68 |
| 21 | 21 |
| 23 | 23 |

| | |
|---|---|
| 55 | 55 |
| 58 | 58 |
| 46 | 46 |
| 17 | 17 |
| 81 | 81 |
| 47 | 47 |
| 90 | 90 |
| 9 | 9 |
| 2 | 2 |
| 74 | 74 |
| 99 | 99 |
| 12 | 12 |
| 48 | 48 |
| 99 | 99 |
| 60 | 60 |
| 80 | 80 |
| 92 | 92 |
| 94 | 94 |
| 52 | 52 |
| 24 | 24 |
| 58 | 58 |
| 54 | 54 |
| 98 | 98 |
| 67 | 67 |
| 10 | 10 |
| 59 | 59 |
| 11 | 11 |
| 23 | 23 |
| 35 | 35 |
| 91 | 91 |
| 46 | 46 |
| 48 | 48 |
| 52 | 52 |
| 72 | 72 |
| 70 | 70 |
| 74 | 74 |
| 28 | 28 |
| 54 | 54 |
| 86 | 86 |
| 4 | 4 |
| 23 | 23 |
| 64 | 64 |
| 67 | 67 |

| | |
|---|---|
| 10 | 10 |
| 21 | 21 |
| 82 | 82 |
| 33 | 33 |
| 16 | 16 |
| 80 | 80 |
| 1 | 1 |
| 57 | 57 |
| 60 | 60 |
| 99 | 99 |
| 55 | 55 |
| 40 | 40 |
| 41 | 41 |
| 68 | 68 |
| 51 | 51 |
| 53 | 53 |
| 28 | 28 |
| 7 | 7 |
| 40 | 40 |
| 19 | 19 |
| 51 | 51 |
| 76 | 76 |
| 15 | 15 |
| 81 | 81 |
| 16 | 16 |
| 100 | 100 |
| 12 | 12 |
| 38 | 38 |
| 61 | 61 |
| 89 | 89 |
| 22 | 22 |
| 17 | 17 |
| 68 | 68 |
| 3 | 3 |
| 99 | 99 |
| 14 | 14 |
| 5 | 5 |

## Default Seeds

| **Trial #1** | | | **Trial #2** | | | **Trial #3** | |
|---|---|---|---|---|---|---|---|
| **Actual** | **Predicted** | | **Actual** | **Predicted** | | **Actual** | **Predicted** |
| 25 | 25 | | 32 | 32 | | 27 | 27 |
| 56 | 56 | | 82 | 82 | | 4 | 4 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 56 | 56 | | 59 | 59 | | 12 | 12 |
| 67 | 67 | | 75 | 75 | | 12 | 12 |
| 92 | 92 | | 72 | 72 | | 6 | 6 |
| 6 | 6 | | 20 | 20 | | 58 | 58 |
| 99 | 99 | | 77 | 77 | | 18 | 18 |
| 7 | 7 | | 49 | 49 | | 27 | 27 |
| 21 | 21 | | 56 | 56 | | 34 | 34 |
| 72 | 72 | | 100 | 100 | | 15 | 15 |
| 92 | 92 | | 65 | 65 | | 12 | 12 |
| 93 | 93 | | 94 | 94 | | 91 | 91 |
| 0 | 0 | | 70 | 70 | | 50 | 50 |
| 41 | 41 | | 65 | 65 | | 69 | 69 |
| 63 | 63 | | 15 | 15 | | 15 | 15 |
| 75 | 75 | | 17 | 17 | | 77 | 77 |
| 33 | 33 | | 86 | 86 | | 36 | 36 |
| 56 | 56 | | 59 | 59 | | 93 | 93 |
| 64 | 64 | | 34 | 34 | | 82 | 82 |
| 33 | 33 | | 12 | 12 | | 20 | 20 |
| 96 | 96 | | 99 | 99 | | 10 | 10 |
| 91 | 91 | | 30 | 30 | | 0 | 0 |
| 28 | 28 | | 5 | 5 | | 84 | 84 |
| 46 | 46 | | 53 | 53 | | 42 | 42 |
| 11 | 11 | | 81 | 81 | | 85 | 85 |
| 66 | 66 | | 40 | 40 | | 91 | 91 |
| 84 | 84 | | 0 | 0 | | 11 | 11 |
| 6 | 6 | | 81 | 81 | | 74 | 74 |
| 43 | 43 | | 5 | 5 | | 85 | 85 |
| 14 | 14 | | 71 | 71 | | 64 | 64 |
| 50 | 50 | | 10 | 10 | | 13 | 13 |
| 72 | 72 | | 25 | 25 | | 1 | 1 |
| 49 | 49 | | 69 | 69 | | 43 | 43 |
| 7 | 7 | | 46 | 46 | | 58 | 58 |
| 46 | 46 | | 40 | 40 | | 84 | 84 |
| 52 | 52 | | 96 | 96 | | 31 | 31 |
| 20 | 20 | | 10 | 10 | | 69 | 69 |
| 79 | 79 | | 49 | 49 | | 91 | 91 |
| 32 | 32 | | 70 | 70 | | 44 | 44 |
| 24 | 24 | | 33 | 33 | | 12 | 12 |
| 82 | 82 | | 14 | 14 | | 19 | 19 |
| 94 | 94 | | 47 | 47 | | 27 | 27 |
| 81 | 81 | | 71 | 71 | | 28 | 28 |
| 35 | 35 | | 86 | 86 | | 85 | 85 |
| 35 | 35 | | 90 | 90 | | 91 | 91 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 78 | 78 | | 85 | 85 | | 74 | 74 |
| 65 | 65 | | 83 | 83 | | 5 | 5 |
| 68 | 68 | | 100 | 100 | | 83 | 83 |
| 48 | 48 | | 18 | 18 | | 25 | 25 |
| 66 | 66 | | 66 | 66 | | 66 | 66 |
| 2 | 2 | | 97 | 97 | | 9 | 9 |
| 77 | 77 | | 27 | 27 | | 56 | 56 |
| 55 | 55 | | 55 | 55 | | 34 | 34 |
| 6 | 6 | | 47 | 47 | | 97 | 97 |
| 35 | 35 | | 44 | 44 | | 90 | 90 |
| 76 | 76 | | 39 | 39 | | 31 | 31 |
| 95 | 95 | | 2 | 2 | | 44 | 44 |
| 10 | 10 | | 7 | 7 | | 98 | 98 |
| 87 | 87 | | 92 | 92 | | 17 | 17 |
| 26 | 26 | | 59 | 59 | | 89 | 89 |
| 87 | 87 | | 67 | 67 | | 17 | 17 |
| 4 | 4 | | 77 | 77 | | 64 | 64 |
| 68 | 68 | | 84 | 84 | | 86 | 86 |
| 98 | 98 | | 64 | 64 | | 45 | 45 |
| 96 | 96 | | 64 | 64 | | 0 | 0 |
| 9 | 9 | | 11 | 11 | | 89 | 89 |
| 68 | 68 | | 35 | 35 | | 92 | 92 |
| 34 | 34 | | 79 | 79 | | 49 | 49 |
| 99 | 99 | | 45 | 45 | | 44 | 44 |
| 46 | 46 | | 1 | 1 | | 79 | 79 |
| 100 | 100 | | 17 | 17 | | 28 | 28 |
| 43 | 43 | | 82 | 82 | | 9 | 9 |
| 46 | 46 | | 71 | 71 | | 48 | 48 |
| 37 | 37 | | 9 | 9 | | 61 | 61 |
| 55 | 55 | | 17 | 17 | | 29 | 29 |
| 60 | 60 | | 61 | 61 | | 1 | 1 |
| 77 | 77 | | 95 | 95 | | 23 | 23 |
| 37 | 37 | | 52 | 52 | | 59 | 59 |
| 77 | 77 | | 97 | 97 | | 3 | 3 |
| 70 | 70 | | 56 | 56 | | 13 | 13 |
| 4 | 4 | | 19 | 19 | | 63 | 63 |
| 62 | 62 | | 49 | 49 | | 38 | 38 |
| 11 | 11 | | 78 | 78 | | 11 | 11 |
| 88 | 88 | | 78 | 78 | | 87 | 87 |
| 5 | 5 | | 17 | 17 | | 77 | 77 |
| 79 | 79 | | 88 | 88 | | 5 | 5 |
| 13 | 13 | | 71 | 71 | | 19 | 19 |
| 14 | 14 | | 19 | 19 | | 15 | 15 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 38 | 38 | | 90 | 90 | | 88 | 88 |
| 29 | 29 | | 80 | 80 | | 57 | 57 |
| 80 | 80 | | 33 | 33 | | 60 | 60 |
| 78 | 78 | | 29 | 29 | | 79 | 79 |
| 13 | 13 | | 39 | 39 | | 73 | 73 |
| 1 | 1 | | 24 | 24 | | 87 | 87 |
| 14 | 14 | | 85 | 85 | | 84 | 84 |
| 58 | 58 | | 84 | 84 | | 10 | 10 |
| 79 | 79 | | 39 | 39 | | 36 | 36 |
| 90 | 90 | | 69 | 69 | | 16 | 16 |
| 73 | 73 | | 11 | 11 | | 89 | 89 |
| 10 | 10 | | 8 | 8 | | 36 | 36 |

## C++

### Prime Seeds

| Actual | Predicted |
|---|---|
| 82 | 82 |
| 29 | 29 |
| 29 | 29 |
| 82 | 82 |
| 22 | 22 |
| 94 | 94 |
| 27 | 27 |
| 90 | 90 |
| 3 | 3 |
| 22 | 22 |
| 56 | 56 |
| 38 | 38 |
| 85 | 85 |
| 51 | 51 |
| 52 | 52 |
| 87 | 87 |
| 47 | 47 |
| 76 | 76 |
| 77 | 77 |
| 84 | 84 |
| 28 | 28 |
| 4 | 4 |
| 5 | 5 |

| | |
|---|---|
| 65 | 65 |
| 91 | 91 |
| 29 | 29 |
| 30 | 30 |
| 62 | 62 |
| 72 | 72 |
| 58 | 58 |
| 26 | 26 |
| 10 | 10 |
| 34 | 34 |
| 83 | 83 |
| 51 | 51 |
| 7 | 7 |
| 78 | 78 |
| 69 | 69 |
| 4 | 4 |
| 53 | 53 |
| 59 | 59 |
| 65 | 65 |
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 51 | 51 |
| 54 | 54 |
| 80 | 80 |
| 65 | 65 |
| 76 | 76 |
| 79 | 79 |
| 77 | 77 |
| 46 | 46 |
| 4 | 4 |
| 75 | 75 |
| 25 | 25 |
| 8 | 8 |
| 29 | 29 |
| 100 | 100 |
| 22 | 22 |
| 52 | 52 |
| 15 | 15 |

| | |
|---|---|
| 20 | 20 |
| 68 | 68 |
| 96 | 96 |
| 69 | 69 |
| 74 | 74 |
| 13 | 13 |
| 77 | 77 |
| 65 | 65 |
| 70 | 70 |
| 95 | 95 |
| 20 | 20 |
| 51 | 51 |
| 23 | 23 |
| 40 | 40 |
| 83 | 83 |
| 76 | 76 |
| 48 | 48 |
| 37 | 37 |
| 64 | 64 |
| 4 | 4 |
| 44 | 44 |
| 86 | 86 |
| 7 | 7 |
| 58 | 58 |
| 98 | 98 |
| 31 | 31 |
| 70 | 70 |
| 15 | 15 |
| 22 | 22 |
| 28 | 28 |
| 13 | 13 |
| 69 | 69 |
| 43 | 43 |
| 6 | 6 |
| 38 | 38 |
| 51 | 51 |
| 93 | 93 |
| 0 | 0 |

## Square Seeds

| Actual | Predicted |
|---|---|
| 41 | 41 |
| 65 | 65 |
| 31 | 31 |
| 41 | 41 |
| 63 | 63 |
| 58 | 58 |
| 25 | 25 |
| 30 | 30 |
| 66 | 66 |
| 47 | 47 |
| 81 | 81 |
| 35 | 35 |
| 50 | 50 |
| 65 | 65 |
| 100 | 100 |
| 21 | 21 |
| 15 | 15 |
| 11 | 11 |
| 80 | 80 |
| 56 | 56 |
| 62 | 62 |
| 87 | 87 |
| 23 | 23 |
| 39 | 39 |
| 90 | 90 |
| 58 | 58 |
| 96 | 96 |
| 71 | 71 |
| 99 | 99 |
| 58 | 58 |
| 97 | 97 |
| 51 | 51 |
| 89 | 89 |
| 87 | 87 |
| 94 | 94 |
| 80 | 80 |
| 60 | 60 |

| | |
|---|---|
| 11 | 11 |
| 19 | 19 |
| 90 | 90 |
| 12 | 12 |
| 98 | 98 |
| 41 | 41 |
| 15 | 15 |
| 46 | 46 |
| 80 | 80 |
| 92 | 92 |
| 90 | 90 |
| 61 | 61 |
| 58 | 58 |
| 38 | 38 |
| 45 | 45 |
| 57 | 57 |
| 98 | 98 |
| 81 | 81 |
| 49 | 49 |
| 34 | 34 |
| 33 | 33 |
| 52 | 52 |
| 69 | 69 |
| 93 | 93 |
| 98 | 98 |
| 52 | 52 |
| 70 | 70 |
| 32 | 32 |
| 58 | 58 |
| 48 | 48 |
| 19 | 19 |
| 53 | 53 |
| 80 | 80 |
| 73 | 73 |
| 17 | 17 |
| 55 | 55 |
| 98 | 98 |
| 94 | 94 |
| 64 | 64 |

| | |
|---|---|
| 38 | 38 |
| 44 | 44 |
| 10 | 10 |
| 59 | 59 |
| 2 | 2 |
| 87 | 87 |
| 56 | 56 |
| 2 | 2 |
| 48 | 48 |
| 91 | 91 |
| 30 | 30 |
| 27 | 27 |
| 75 | 75 |
| 91 | 91 |
| 0 | 0 |
| 68 | 68 |
| 83 | 83 |
| 19 | 19 |
| 67 | 67 |
| 57 | 57 |
| 72 | 72 |
| 44 | 44 |
| 62 | 62 |
| 27 | 27 |

## Default Seeds

| **Actual** | **Predicted** |
|---|---|
| 41 | 41 |
| 65 | 65 |
| 31 | 31 |
| 41 | 41 |
| 19 | 19 |
| 15 | 15 |
| 72 | 72 |
| 11 | 11 |
| 78 | 78 |
| 69 | 69 |
| 37 | 37 |
| 23 | 23 |
| 29 | 29 |

| | |
|---|---|
| 63 | 63 |
| 75 | 75 |
| 4 | 4 |
| 5 | 5 |
| 49 | 49 |
| 75 | 75 |
| 99 | 99 |
| 27 | 27 |
| 61 | 61 |
| 62 | 62 |
| 17 | 17 |
| 79 | 79 |
| 61 | 61 |
| 22 | 22 |
| 13 | 13 |
| 49 | 49 |
| 71 | 71 |
| 61 | 61 |
| 8 | 8 |
| 81 | 81 |
| 67 | 67 |
| 80 | 80 |
| 47 | 47 |
| 83 | 83 |
| 88 | 88 |
| 30 | 30 |
| 12 | 12 |
| 74 | 74 |
| 78 | 78 |
| 33 | 33 |
| 23 | 23 |
| 58 | 58 |
| 83 | 83 |
| 76 | 76 |
| 59 | 59 |
| 77 | 77 |
| 85 | 85 |
| 25 | 25 |
| 18 | 18 |
| 98 | 98 |
| 16 | 16 |
| 100 | 100 |
| 79 | 79 |

| | |
|---|---|
| 88 | 88 |
| 36 | 36 |
| 53 | 53 |
| 72 | 72 |
| 95 | 95 |
| 93 | 93 |
| 39 | 39 |
| 78 | 78 |
| 5 | 5 |
| 15 | 15 |
| 56 | 56 |
| 26 | 26 |
| 62 | 62 |
| 65 | 65 |
| 57 | 57 |
| 69 | 69 |
| 63 | 63 |
| 31 | 31 |
| 63 | 63 |
| 68 | 68 |
| 46 | 46 |
| 40 | 40 |
| 46 | 46 |
| 15 | 15 |
| 87 | 87 |
| 49 | 49 |
| 47 | 47 |
| 17 | 17 |
| 61 | 61 |
| 8 | 8 |
| 42 | 42 |
| 39 | 39 |
| 64 | 64 |
| 86 | 86 |
| 49 | 49 |
| 43 | 43 |
| 44 | 44 |
| 47 | 47 |
| 69 | 69 |
| 25 | 25 |
| 67 | 67 |
| 58 | 58 |
| 57 | 57 |

| | |
|---|---|
| 35 | 35 |

## Swift

### Prime Seeds

| Actual | Predicted |
|---|---|
| 97 | 97 |
| 7 | 7 |
| 47 | 47 |
| 2 | 2 |
| 47 | 47 |
| 83 | 83 |
| 86 | 86 |
| 98 | 98 |
| 50 | 50 |
| 33 | 33 |
| 17 | 17 |
| 0 | 0 |
| 5 | 5 |
| 29 | 29 |
| 22 | 22 |
| 26 | 26 |
| 44 | 44 |
| 66 | 66 |
| 62 | 62 |
| 23 | 23 |
| 10 | 10 |
| 18 | 18 |
| 34 | 34 |
| 40 | 40 |
| 0 | 0 |
| 69 | 69 |
| 44 | 44 |
| 31 | 31 |
| 76 | 76 |
| 3 | 3 |
| 45 | 45 |
| 54 | 54 |
| 80 | 80 |

| | |
|---|---|
| 48 | 48 |
| 97 | 97 |
| 55 | 55 |
| 74 | 74 |
| 77 | 77 |
| 10 | 10 |
| 94 | 94 |
| 8 | 8 |
| 2 | 2 |
| 70 | 70 |
| 45 | 45 |
| 40 | 40 |
| 83 | 83 |
| 94 | 94 |
| 13 | 13 |
| 59 | 59 |
| 63 | 63 |
| 24 | 24 |
| 83 | 83 |
| 11 | 11 |
| 92 | 92 |
| 32 | 32 |
| 77 | 77 |
| 61 | 61 |
| 50 | 50 |
| 47 | 47 |
| 35 | 35 |
| 47 | 47 |
| 76 | 76 |
| 29 | 29 |
| 33 | 33 |
| 43 | 43 |
| 19 | 19 |
| 36 | 36 |
| 27 | 27 |
| 65 | 65 |
| 51 | 51 |
| 100 | 100 |
| 92 | 92 |

| | |
|---|---|
| 9 | 9 |
| 54 | 54 |
| 93 | 93 |
| 21 | 21 |
| 67 | 67 |
| 25 | 25 |
| 35 | 35 |
| 44 | 44 |
| 63 | 63 |
| 6 | 6 |
| 0 | 0 |
| 3 | 3 |
| 51 | 51 |
| 21 | 21 |
| 56 | 56 |
| 18 | 18 |
| 29 | 29 |
| 61 | 61 |
| 21 | 21 |
| 42 | 42 |
| 90 | 90 |
| 32 | 32 |
| 50 | 50 |
| 96 | 96 |
| 14 | 14 |
| 41 | 41 |
| 27 | 27 |
| 89 | 89 |

## Square Seeds

| **Actual** | **Predicted** |
|---|---|
| 80 | 80 |
| 29 | 29 |
| 100 | 100 |
| 30 | 30 |
| 22 | 22 |
| 60 | 60 |
| 52 | 52 |
| 47 | 47 |

| | |
|---|---|
| 56 | 56 |
| 95 | 95 |
| 16 | 16 |
| 36 | 36 |
| 32 | 32 |
| 12 | 12 |
| 61 | 61 |
| 33 | 33 |
| 71 | 71 |
| 77 | 77 |
| 5 | 5 |
| 91 | 91 |
| 31 | 31 |
| 4 | 4 |
| 13 | 13 |
| 9 | 9 |
| 11 | 11 |
| 64 | 64 |
| 81 | 81 |
| 97 | 97 |
| 37 | 37 |
| 52 | 52 |
| 81 | 81 |
| 25 | 25 |
| 93 | 93 |
| 80 | 80 |
| 15 | 15 |
| 64 | 64 |
| 50 | 50 |
| 62 | 62 |
| 3 | 3 |
| 87 | 87 |
| 60 | 60 |
| 74 | 74 |
| 57 | 57 |
| 1 | 1 |
| 53 | 53 |
| 1 | 1 |
| 10 | 10 |

| | |
|---|---|
| 20 | 20 |
| 77 | 77 |
| 2 | 2 |
| 10 | 10 |
| 85 | 85 |
| 65 | 65 |
| 72 | 72 |
| 74 | 74 |
| 16 | 16 |
| 31 | 31 |
| 11 | 11 |
| 18 | 18 |
| 64 | 64 |
| 100 | 100 |
| 97 | 97 |
| 25 | 25 |
| 41 | 41 |
| 88 | 88 |
| 18 | 18 |
| 68 | 68 |
| 4 | 4 |
| 31 | 31 |
| 17 | 17 |
| 45 | 45 |
| 50 | 50 |
| 60 | 60 |
| 4 | 4 |
| 34 | 34 |
| 9 | 9 |
| 55 | 55 |
| 29 | 29 |
| 62 | 62 |
| 33 | 33 |
| 14 | 14 |
| 12 | 12 |
| 61 | 61 |
| 89 | 89 |
| 53 | 53 |
| 83 | 83 |

| | |
|---|---|
| 17 | 17 |
| 72 | 72 |
| 11 | 11 |
| 49 | 49 |
| 9 | 9 |
| 27 | 27 |
| 99 | 99 |
| 66 | 66 |
| 19 | 19 |
| 59 | 59 |
| 75 | 75 |
| 15 | 15 |
| 98 | 98 |
| 37 | 37 |

## Default Seeds

| **Trial #1** | | | **Trial #2** | | | **Trial #3** | |
|---|---|---|---|---|---|---|---|
| **Actual** | **Predicted** | | **Actual** | **Predicted** | | **Actual** | **Predicted** |
| 0 | 0 | | 56 | 56 | | 69 | 69 |
| 35 | 35 | | 14 | 14 | | 1 | 1 |
| 21 | 21 | | 24 | 24 | | 39 | 39 |
| 98 | 98 | | 71 | 71 | | 48 | 48 |
| 43 | 43 | | 86 | 86 | | 56 | 56 |
| 58 | 58 | | 42 | 42 | | 19 | 19 |
| 42 | 42 | | 62 | 62 | | 91 | 91 |
| 98 | 98 | | 71 | 71 | | 22 | 22 |
| 36 | 36 | | 57 | 57 | | 6 | 6 |
| 32 | 32 | | 91 | 91 | | 94 | 94 |
| 13 | 13 | | 90 | 90 | | 9 | 9 |
| 61 | 61 | | 53 | 53 | | 40 | 40 |
| 89 | 89 | | 31 | 31 | | 67 | 67 |
| 69 | 69 | | 44 | 44 | | 22 | 22 |
| 91 | 91 | | 66 | 66 | | 75 | 75 |
| 99 | 99 | | 64 | 64 | | 82 | 82 |
| 98 | 98 | | 65 | 65 | | 67 | 67 |
| 19 | 19 | | 35 | 35 | | 32 | 32 |
| 17 | 17 | | 51 | 51 | | 30 | 30 |
| 98 | 98 | | 0 | 0 | | 100 | 100 |
| 2 | 2 | | 22 | 22 | | 83 | 83 |
| 12 | 12 | | 89 | 89 | | 22 | 22 |
| 90 | 90 | | 2 | 2 | | 23 | 23 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 32 | 32 | | 20 | 20 | | 57 | 57 |
| 4 | 4 | | 83 | 83 | | 19 | 19 |
| 4 | 4 | | 97 | 97 | | 61 | 61 |
| 6 | 6 | | 46 | 46 | | 82 | 82 |
| 78 | 78 | | 73 | 73 | | 68 | 68 |
| 91 | 91 | | 17 | 17 | | 36 | 36 |
| 46 | 46 | | 77 | 77 | | 72 | 72 |
| 98 | 98 | | 57 | 57 | | 46 | 46 |
| 4 | 4 | | 68 | 68 | | 23 | 23 |
| 7 | 7 | | 36 | 36 | | 48 | 48 |
| 94 | 94 | | 83 | 83 | | 40 | 40 |
| 94 | 94 | | 71 | 71 | | 64 | 64 |
| 99 | 99 | | 36 | 36 | | 24 | 24 |
| 26 | 26 | | 52 | 52 | | 93 | 93 |
| 41 | 41 | | 60 | 60 | | 16 | 16 |
| 20 | 20 | | 69 | 69 | | 84 | 84 |
| 7 | 7 | | 59 | 59 | | 20 | 20 |
| 3 | 3 | | 43 | 43 | | 61 | 61 |
| 34 | 34 | | 32 | 32 | | 83 | 83 |
| 7 | 7 | | 3 | 3 | | 47 | 47 |
| 32 | 32 | | 23 | 23 | | 93 | 93 |
| 0 | 0 | | 48 | 48 | | 34 | 34 |
| 77 | 77 | | 74 | 74 | | 70 | 70 |
| 0 | 0 | | 61 | 61 | | 9 | 9 |
| 92 | 92 | | 13 | 13 | | 53 | 53 |
| 27 | 27 | | 57 | 57 | | 22 | 22 |
| 83 | 83 | | 42 | 42 | | 94 | 94 |
| 39 | 39 | | 15 | 15 | | 18 | 18 |
| 51 | 51 | | 45 | 45 | | 83 | 83 |
| 95 | 95 | | 42 | 42 | | 77 | 77 |
| 42 | 42 | | 25 | 25 | | 30 | 30 |
| 22 | 22 | | 50 | 50 | | 34 | 34 |
| 68 | 68 | | 11 | 11 | | 67 | 67 |
| 0 | 0 | | 64 | 64 | | 30 | 30 |
| 27 | 27 | | 89 | 89 | | 12 | 12 |
| 26 | 26 | | 59 | 59 | | 24 | 24 |
| 63 | 63 | | 81 | 81 | | 18 | 18 |
| 93 | 93 | | 61 | 61 | | 38 | 38 |
| 66 | 66 | | 2 | 2 | | 82 | 82 |
| 79 | 79 | | 82 | 82 | | 93 | 93 |
| 7 | 7 | | 75 | 75 | | 98 | 98 |
| 39 | 39 | | 11 | 11 | | 11 | 11 |
| 78 | 78 | | 84 | 84 | | 78 | 78 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 35 | 35 | | 44 | 44 | | 47 | 47 |
| 6 | 6 | | 68 | 68 | | 6 | 6 |
| 60 | 60 | | 40 | 40 | | 71 | 71 |
| 58 | 58 | | 30 | 30 | | 26 | 26 |
| 50 | 50 | | 29 | 29 | | 29 | 29 |
| 94 | 94 | | 40 | 40 | | 23 | 23 |
| 60 | 60 | | 30 | 30 | | 17 | 17 |
| 28 | 28 | | 48 | 48 | | 4 | 4 |
| 45 | 45 | | 9 | 9 | | 50 | 50 |
| 89 | 89 | | 82 | 82 | | 37 | 37 |
| 13 | 13 | | 48 | 48 | | 77 | 77 |
| 39 | 39 | | 97 | 97 | | 9 | 9 |
| 35 | 35 | | 81 | 81 | | 77 | 77 |
| 41 | 41 | | 24 | 24 | | 94 | 94 |
| 42 | 42 | | 21 | 21 | | 81 | 81 |
| 54 | 54 | | 20 | 20 | | 12 | 12 |
| 82 | 82 | | 94 | 94 | | 57 | 57 |
| 14 | 14 | | 9 | 9 | | 47 | 47 |
| 70 | 70 | | 11 | 11 | | 38 | 38 |
| 84 | 84 | | 89 | 89 | | 95 | 95 |
| 44 | 44 | | 29 | 29 | | 29 | 29 |
| 55 | 55 | | 84 | 84 | | 28 | 28 |
| 64 | 64 | | 66 | 66 | | 71 | 71 |
| 0 | 0 | | 26 | 26 | | 12 | 12 |
| 98 | 98 | | 33 | 33 | | 32 | 32 |
| 16 | 16 | | 47 | 47 | | 16 | 16 |
| 99 | 99 | | 71 | 71 | | 88 | 88 |
| 83 | 83 | | 97 | 97 | | 24 | 24 |
| 37 | 37 | | 29 | 29 | | 18 | 18 |
| 34 | 34 | | 96 | 96 | | 85 | 85 |
| 22 | 22 | | 53 | 53 | | 81 | 81 |
| 55 | 55 | | 35 | 35 | | 98 | 98 |
| 6 | 6 | | 63 | 63 | | 55 | 55 |
| 61 | 61 | | 45 | 45 | | 68 | 68 |

# Appendix E

## Random.org

| **Trial #1** | | | **Trial #2** | | | **Trial #3** | |
|---|---|---|---|---|---|---|---|
| **Actual** | **Predicted** | | **Actual** | **Predicted** | | **Actual** | **Predicted** |
| 30 | 93 | | 99 | 72 | | 43 | 36 |
| 71 | 7 | | 58 | 36 | | 10 | 97 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 75 | 64 | | 54 | 53 | | 36 | 65 |
| 4 | 36 | | 55 | 1 | | 85 | 83 |
| 39 | 62 | | 88 | 30 | | 33 | 82 |
| 35 | 11 | | 29 | 78 | | 87 | 96 |
| 9 | 46 | | 34 | 64 | | 91 | 63 |
| 42 | 40 | | 47 | 23 | | 32 | 55 |
| 77 | 56 | | 66 | 95 | | 98 | 99 |
| 85 | 13 | | 26 | 69 | | 21 | 97 |
| 63 | 70 | | 9 | 16 | | 82 | 2 |
| 6 | 14 | | 59 | 71 | | 71 | 51 |
| 45 | 23 | | 31 | 57 | | 14 | 66 |
| 98 | 31 | | 95 | 71 | | 100 | 15 |
| 48 | 40 | | 71 | 39 | | 72 | 3 |
| 91 | 34 | | 91 | 47 | | 30 | 66 |
| 80 | 14 | | 86 | 31 | | 65 | 45 |
| 36 | 74 | | 11 | 42 | | 35 | 46 |
| 37 | 77 | | 97 | 10 | | 47 | 67 |
| 28 | 68 | | 28 | 66 | | 51 | 50 |
| 76 | 14 | | 42 | 50 | | 22 | 60 |
| 49 | 97 | | 94 | 19 | | 50 | 13 |
| 14 | 41 | | 36 | 49 | | 3 | 31 |
| 86 | 78 | | 75 | 20 | | 76 | 40 |
| 0 | 6 | | 79 | 68 | | 18 | 65 |
| 55 | 85 | | 57 | 50 | | 89 | 10 |
| 5 | 24 | | 81 | 77 | | 13 | 22 |
| 54 | 60 | | 18 | 24 | | 24 | 63 |
| 10 | 88 | | 56 | 24 | | 59 | 46 |
| 84 | 93 | | 27 | 80 | | 86 | 13 |
| 20 | 95 | | 80 | 54 | | 5 | 51 |
| 66 | 2 | | 87 | 79 | | 69 | 69 |
| 22 | 91 | | 5 | 57 | | 57 | 30 |
| 24 | 39 | | 3 | 44 | | 81 | 43 |
| 87 | 89 | | 2 | 91 | | 97 | 46 |
| 52 | 64 | | 70 | 6 | | 95 | 17 |
| 74 | 49 | | 45 | 42 | | 34 | 9 |
| 11 | 11 | | 44 | 38 | | 39 | 28 |
| 88 | 45 | | 65 | 58 | | 79 | 61 |
| 12 | 26 | | 14 | 36 | | 26 | 17 |
| 83 | 90 | | 7 | 24 | | 55 | 86 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 13 | 66 | | 25 | 93 | | 44 | 28 |
| 47 | 6 | | 17 | 95 | | 42 | 34 |
| 7 | 19 | | 96 | 36 | | 31 | 46 |
| 27 | 95 | | 41 | 3 | | 74 | 86 |
| 56 | 35 | | 15 | 24 | | 48 | 85 |
| 61 | 28 | | 63 | 64 | | 99 | 6 |
| 95 | 61 | | 52 | 52 | | 37 | 77 |
| 46 | 73 | | 33 | 22 | | 88 | 7 |
| 78 | 99 | | 93 | 49 | | 80 | 0 |
| 67 | 8 | | 49 | 44 | | 8 | 18 |
| 17 | 3 | | 90 | 77 | | 61 | 91 |
| 99 | 70 | | 64 | 42 | | 19 | 38 |
| 60 | 1 | | 85 | 41 | | 16 | 92 |
| 38 | 66 | | 60 | 24 | | 62 | 32 |
| 68 | 45 | | 19 | 5 | | 54 | 84 |
| 92 | 12 | | 84 | 2 | | 83 | 23 |
| 59 | 78 | | 22 | 92 | | 53 | 56 |
| 93 | 11 | | 46 | 74 | | 6 | 73 |
| 44 | 19 | | 100 | 76 | | 12 | 23 |
| 50 | 41 | | 78 | 31 | | 46 | 11 |
| 31 | 60 | | 89 | 86 | | 29 | 34 |
| 40 | 78 | | 92 | 76 | | 45 | 42 |
| 32 | 73 | | 38 | 25 | | 40 | 41 |
| 16 | 34 | | 68 | 87 | | 11 | 34 |
| 15 | 59 | | 83 | 73 | | 84 | 54 |
| 64 | 49 | | 48 | 86 | | 73 | 30 |
| 41 | 44 | | 10 | 5 | | 27 | 25 |
| 1 | 70 | | 13 | 34 | | 38 | 16 |
| 90 | 90 | | 16 | 59 | | 77 | 83 |
| 19 | 80 | | 30 | 87 | | 49 | 84 |
| 89 | 34 | | 8 | 73 | | 68 | 99 |
| 57 | 87 | | 50 | 75 | | 75 | 46 |
| 62 | 22 | | 74 | 26 | | 20 | 75 |
| 51 | 66 | | 69 | 32 | | 66 | 45 |
| 97 | 15 | | 21 | 64 | | 90 | 10 |
| 53 | 26 | | 20 | 12 | | 67 | 39 |
| 2 | 33 | | 24 | 38 | | 93 | 19 |
| 43 | 92 | | 43 | 63 | | 41 | 40 |
| 69 | 35 | | 39 | 90 | | 23 | 55 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 25 | 24 | | 67 | 39 | | 4 | 95 |
| 82 | 73 | | 72 | 56 | | 96 | 78 |
| 23 | 40 | | 82 | 7 | | 60 | 27 |
| 34 | 16 | | 77 | 67 | | 17 | 71 |
| 26 | 30 | | 76 | 81 | | 70 | 52 |
| 72 | 76 | | 61 | 55 | | 63 | 59 |
| 96 | 27 | | 4 | 22 | | 7 | 26 |
| 65 | 14 | | 51 | 12 | | 9 | 14 |
| 100 | 48 | | 40 | 75 | | 2 | 57 |
| 58 | 43 | | 12 | 46 | | 56 | 53 |
| 73 | 83 | | 0 | 53 | | 64 | 7 |
| 79 | 42 | | 32 | 68 | | 25 | 84 |
| 21 | 48 | | 37 | 83 | | 58 | 79 |
| 33 | 36 | | 73 | 97 | | 94 | 87 |
| 8 | 26 | | 62 | 97 | | 52 | 99 |
| 81 | 80 | | 23 | 43 | | 92 | 99 |
| 3 | 49 | | 98 | 15 | | 1 | 81 |
| 70 | 33 | | 1 | 6 | | 28 | 51 |
| 94 | 13 | | 53 | 34 | | 78 | 92 |
| 29 | 66 | | 6 | 57 | | 15 | 12 |

## ANU QRNG

| **Trial #1** | | | **Trial #2** | | | **Trial #3** | |
|---|---|---|---|---|---|---|---|
| **Actual** | **Predicted** | | **Actual** | **Predicted** | | **Actual** | **Predicted** |
| 57 | 27 | | 87 | 47 | | 89 | 6 |
| 23 | 13 | | 81 | 90 | | 34 | 26 |
| 75 | 63 | | 66 | 89 | | 17 | 53 |
| 35 | 56 | | 53 | 71 | | 20 | 71 |
| 71 | 28 | | 99 | 75 | | 65 | 69 |
| 95 | 88 | | 96 | 52 | | 35 | 90 |
| 97 | 21 | | 4 | 61 | | 26 | 76 |
| 40 | 13 | | 89 | 13 | | 47 | 13 |
| 49 | 50 | | 77 | 33 | | 76 | 72 |
| 66 | 46 | | 28 | 43 | | 2 | 17 |
| 7 | 15 | | 43 | 24 | | 61 | 51 |
| 88 | 23 | | 42 | 93 | | 33 | 95 |
| 53 | 91 | | 59 | 22 | | 54 | 77 |
| 43 | 53 | | 11 | 17 | | 55 | 31 |
| 70 | 72 | | 68 | 30 | | 21 | 39 |
| 87 | 78 | | 19 | 56 | | 3 | 36 |
| 91 | 30 | | 84 | 92 | | 71 | 41 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 10 | 93 | | 35 | 66 | | 49 | 20 |
| 81 | 26 | | 55 | 61 | | 52 | 77 |
| 25 | 35 | | 7 | 44 | | 66 | 0 |
| 74 | 49 | | 24 | 61 | | 56 | 12 |
| 72 | 25 | | 78 | 11 | | 82 | 42 |
| 79 | 62 | | 61 | 28 | | 58 | 53 |
| 19 | 28 | | 74 | 2 | | 23 | 82 |
| 32 | 38 | | 76 | 98 | | 6 | 29 |
| 78 | 86 | | 21 | 54 | | 72 | 9 |
| 80 | 99 | | 29 | 92 | | 73 | 21 |
| 16 | 1 | | 51 | 90 | | 38 | 72 |
| 9 | 5 | | 1 | 51 | | 42 | 8 |
| 93 | 72 | | 94 | 93 | | 94 | 74 |
| 89 | 32 | | 75 | 38 | | 10 | 50 |
| 24 | 43 | | 5 | 16 | | 29 | 41 |
| 3 | 58 | | 15 | 68 | | 91 | 22 |
| 5 | 76 | | 64 | 77 | | 18 | 75 |
| 13 | 70 | | 79 | 26 | | 67 | 72 |
| 2 | 1 | | 26 | 39 | | 53 | 60 |
| 34 | 43 | | 86 | 82 | | 69 | 5 |
| 61 | 4 | | 3 | 6 | | 24 | 97 |
| 38 | 44 | | 46 | 64 | | 12 | 59 |
| 48 | 92 | | 72 | 82 | | 9 | 17 |
| 62 | 99 | | 73 | 78 | | 97 | 55 |
| 41 | 55 | | 60 | 65 | | 57 | 37 |
| 94 | 64 | | 30 | 80 | | 4 | 53 |
| 86 | 94 | | 98 | 44 | | 62 | 18 |
| 67 | 55 | | 0 | 38 | | 83 | 80 |
| 84 | 56 | | 54 | 89 | | 14 | 1 |
| 100 | 99 | | 62 | 37 | | 44 | 21 |
| 28 | 13 | | 16 | 23 | | 93 | 61 |
| 51 | 88 | | 2 | 74 | | 27 | 41 |
| 33 | 6 | | 65 | 72 | | 43 | 63 |
| 69 | 100 | | 20 | 45 | | 63 | 44 |
| 92 | 79 | | 52 | 21 | | 0 | 8 |
| 56 | 58 | | 38 | 54 | | 16 | 33 |
| 47 | 70 | | 95 | 64 | | 92 | 47 |
| 26 | 44 | | 58 | 91 | | 78 | 19 |
| 42 | 17 | | 48 | 31 | | 48 | 13 |
| 17 | 29 | | 37 | 71 | | 70 | 41 |
| 63 | 12 | | 23 | 42 | | 22 | 35 |
| 64 | 84 | | 93 | 30 | | 39 | 21 |
| 30 | 36 | | 50 | 12 | | 1 | 4 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 55 | 83 | | 18 | 5 | | 96 | 77 |
| 68 | 18 | | 97 | 82 | | 80 | 32 |
| 99 | 25 | | 32 | 38 | | 30 | 32 |
| 46 | 26 | | 88 | 35 | | 7 | 16 |
| 15 | 56 | | 36 | 63 | | 51 | 83 |
| 59 | 25 | | 67 | 32 | | 85 | 43 |
| 85 | 87 | | 44 | 79 | | 50 | 95 |
| 12 | 42 | | 70 | 58 | | 13 | 48 |
| 83 | 9 | | 90 | 46 | | 84 | 69 |
| 1 | 43 | | 49 | 47 | | 79 | 15 |
| 22 | 71 | | 39 | 74 | | 99 | 76 |
| 0 | 20 | | 12 | 93 | | 36 | 12 |
| 4 | 93 | | 33 | 69 | | 68 | 93 |
| 76 | 85 | | 91 | 98 | | 45 | 46 |
| 52 | 70 | | 8 | 90 | | 15 | 49 |
| 58 | 35 | | 56 | 31 | | 32 | 76 |
| 6 | 11 | | 25 | 46 | | 90 | 11 |
| 21 | 91 | | 40 | 3 | | 98 | 37 |
| 29 | 33 | | 17 | 61 | | 37 | 25 |
| 31 | 19 | | 69 | 62 | | 59 | 0 |
| 27 | 61 | | 6 | 74 | | 74 | 100 |
| 60 | 88 | | 10 | 6 | | 41 | 6 |
| 8 | 33 | | 27 | 91 | | 40 | 48 |
| 44 | 66 | | 100 | 49 | | 81 | 7 |
| 18 | 53 | | 83 | 31 | | 46 | 53 |
| 90 | 83 | | 47 | 14 | | 5 | 40 |
| 77 | 3 | | 92 | 55 | | 95 | 43 |
| 50 | 15 | | 41 | 37 | | 31 | 89 |
| 96 | 93 | | 80 | 94 | | 75 | 0 |
| 20 | 74 | | 34 | 24 | | 88 | 40 |
| 45 | 1 | | 22 | 6 | | 19 | 46 |
| 98 | 56 | | 31 | 7 | | 28 | 17 |
| 37 | 76 | | 57 | 76 | | 60 | 16 |
| 11 | 26 | | 14 | 78 | | 87 | 3 |
| 82 | 35 | | 45 | 75 | | 77 | 91 |
| 36 | 22 | | 13 | 43 | | 25 | 23 |
| 39 | 55 | | 71 | 1 | | 64 | 22 |
| 73 | 24 | | 63 | 7 | | 100 | 93 |
| 14 | 98 | | 82 | 4 | | 11 | 62 |
| 65 | 68 | | 9 | 51 | | 86 | 11 |

## HotBits

| Trial #1 | | | Trial #2 | | | Trial #3 | |
|---|---|---|---|---|---|---|---|
| **Actual** | **Predicted** | | **Actual** | **Predicted** | | **Actual** | **Predicted** |
| 59 | 72 | | 5 | 23 | | 49 | 47 |
| 16 | 66 | | 84 | 81 | | 91 | 52 |
| 48 | 59 | | 78 | 50 | | 64 | 69 |
| 98 | 54 | | 34 | 58 | | 82 | 38 |
| 24 | 22 | | 98 | 44 | | 76 | 17 |
| 22 | 46 | | 32 | 70 | | 29 | 89 |
| 40 | 89 | | 90 | 4 | | 9 | 86 |
| 2 | 66 | | 52 | 72 | | 43 | 20 |
| 15 | 78 | | 45 | 64 | | 57 | 38 |
| 90 | 37 | | 61 | 74 | | 23 | 76 |
| 13 | 77 | | 18 | 25 | | 2 | 4 |
| 43 | 77 | | 36 | 51 | | 38 | 33 |
| 57 | 71 | | 49 | 19 | | 70 | 17 |
| 70 | 25 | | 82 | 45 | | 37 | 74 |
| 7 | 36 | | 14 | 73 | | 7 | 80 |
| 19 | 51 | | 71 | 58 | | 88 | 21 |
| 95 | 85 | | 64 | 49 | | 72 | 25 |
| 64 | 80 | | 33 | 95 | | 94 | 68 |
| 83 | 88 | | 62 | 38 | | 18 | 13 |
| 11 | 27 | | 25 | 17 | | 63 | 29 |
| 100 | 67 | | 17 | 96 | | 22 | 67 |
| 89 | 54 | | 9 | 83 | | 59 | 99 |
| 97 | 87 | | 60 | 80 | | 67 | 7 |
| 73 | 30 | | 73 | 13 | | 10 | 21 |
| 34 | 81 | | 28 | 60 | | 89 | 6 |
| 65 | 31 | | 95 | 61 | | 73 | 14 |
| 86 | 16 | | 75 | 37 | | 66 | 49 |
| 36 | 75 | | 0 | 99 | | 50 | 73 |
| 99 | 77 | | 2 | 4 | | 87 | 20 |
| 78 | 92 | | 1 | 36 | | 46 | 13 |
| 85 | 46 | | 23 | 87 | | 5 | 84 |
| 4 | 17 | | 26 | 96 | | 32 | 1 |
| 71 | 52 | | 10 | 83 | | 48 | 0 |
| 41 | 28 | | 3 | 83 | | 78 | 10 |
| 14 | 66 | | 41 | 6 | | 96 | 81 |
| 45 | 2 | | 11 | 8 | | 39 | 41 |
| 82 | 55 | | 91 | 75 | | 21 | 63 |
| 84 | 54 | | 97 | 7 | | 24 | 6 |
| 39 | 96 | | 30 | 57 | | 83 | 48 |
| 61 | 5 | | 94 | 33 | | 85 | 10 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 75 | 71 | | 100 | 13 | | 55 | 91 |
| 68 | 93 | | 27 | 60 | | 93 | 64 |
| 38 | 49 | | 72 | 40 | | 11 | 21 |
| 77 | 9 | | 87 | 33 | | 3 | 46 |
| 74 | 87 | | 8 | 72 | | 53 | 98 |
| 25 | 44 | | 86 | 18 | | 14 | 7 |
| 46 | 27 | | 24 | 7 | | 47 | 32 |
| 62 | 4 | | 21 | 36 | | 56 | 54 |
| 96 | 56 | | 22 | 44 | | 34 | 31 |
| 12 | 69 | | 69 | 65 | | 25 | 45 |
| 30 | 27 | | 55 | 32 | | 42 | 36 |
| 54 | 31 | | 4 | 85 | | 27 | 54 |
| 32 | 95 | | 7 | 56 | | 58 | 92 |
| 6 | 77 | | 79 | 75 | | 41 | 77 |
| 21 | 54 | | 16 | 99 | | 26 | 37 |
| 1 | 65 | | 20 | 52 | | 36 | 41 |
| 91 | 0 | | 54 | 71 | | 33 | 94 |
| 60 | 50 | | 58 | 94 | | 71 | 36 |
| 42 | 64 | | 47 | 56 | | 74 | 26 |
| 88 | 68 | | 77 | 52 | | 95 | 43 |
| 69 | 65 | | 53 | 65 | | 81 | 90 |
| 9 | 3 | | 13 | 58 | | 8 | 57 |
| 17 | 21 | | 68 | 94 | | 77 | 1 |
| 35 | 9 | | 93 | 0 | | 68 | 50 |
| 72 | 74 | | 89 | 94 | | 28 | 98 |
| 31 | 21 | | 37 | 67 | | 86 | 25 |
| 50 | 42 | | 29 | 0 | | 13 | 11 |
| 92 | 92 | | 42 | 72 | | 51 | 24 |
| 53 | 29 | | 80 | 19 | | 15 | 72 |
| 79 | 41 | | 35 | 43 | | 80 | 92 |
| 67 | 97 | | 63 | 19 | | 65 | 72 |
| 76 | 82 | | 65 | 91 | | 99 | 26 |
| 66 | 46 | | 51 | 77 | | 35 | 87 |
| 20 | 65 | | 31 | 88 | | 100 | 60 |
| 26 | 31 | | 40 | 27 | | 90 | 51 |
| 81 | 74 | | 57 | 52 | | 40 | 87 |
| 55 | 48 | | 38 | 80 | | 60 | 66 |
| 63 | 58 | | 83 | 53 | | 97 | 100 |
| 37 | 7 | | 99 | 25 | | 6 | 7 |
| 27 | 26 | | 92 | 55 | | 45 | 38 |
| 93 | 75 | | 6 | 39 | | 54 | 19 |
| 51 | 92 | | 19 | 55 | | 12 | 80 |
| 23 | 67 | | 12 | 36 | | 84 | 63 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 33 | 29 | | 67 | 36 | | 16 | 97 |
| 8 | 33 | | 88 | 76 | | 61 | 49 |
| 56 | 52 | | 74 | 26 | | 92 | 3 |
| 94 | 29 | | 66 | 84 | | 4 | 86 |
| 47 | 22 | | 44 | 100 | | 79 | 75 |
| 0 | 81 | | 96 | 71 | | 20 | 61 |
| 28 | 34 | | 43 | 42 | | 1 | 83 |
| 87 | 6 | | 70 | 29 | | 44 | 76 |
| 49 | 74 | | 81 | 19 | | 52 | 11 |
| 18 | 99 | | 59 | 3 | | 62 | 71 |
| 44 | 80 | | 85 | 94 | | 19 | 77 |
| 3 | 29 | | 46 | 45 | | 17 | 99 |
| 5 | 20 | | 56 | 32 | | 30 | 99 |
| 52 | 58 | | 48 | 57 | | 98 | 34 |
| 10 | 17 | | 50 | 84 | | 75 | 27 |
| 29 | 1 | | 15 | 74 | | 69 | 12 |
| 80 | 4 | | 39 | 89 | | 31 | 74 |