UM–SJTU Joint Institute

Probabilistic Methods in Engineering
(VE401)

Project Report

Group 13

Tests for Pseudorandom Number

Name: Hu Bingcheng	ID:516021910219
Name: Liao Xinhao	ID:516370910037
Name: Fu Tianchi	ID:517370910007
Name: He Yifei	ID:517370910196
Name: Mao Xingyun	ID:517370910199

Date: 4 July 2019

Contents

1 Introduction

There are many situations in which random numbers are needed. Among many of the applications of random numbers, a really important one is encryption. Common cryptosystems employ keys that have to be generated in a random fashion. For example, random inputs are needed for generating digital signatures and for generating challenges in authentication protocols. There are numbers that are truly random and also the so-called pseudorandom numbers that are not truly random but still widely used in practice. Accordingly, there are two basic types of generators used to produce random sequences: random number generators (RNGs - see Section 1.1) and pseudorandom number generators (PRNGs - see Section 1.2). In this article, we will only discuss PRNGs and focus on how to determine if a PRNG is sufficiently random looking. We will introduce three useful statistical tests to help us better determine the quality of a PRNG, namely, the Frequency Test, Wald-Wolfowitz Runs Test and the Frequency Test within Blocks.

1.1 Random Number Generator (RNG)

The first type of sequence generator is a random number generator (RNG). This type of generators relies on a physical source of randomness, and measures a quantity that is either *theoretically* unpredictable, or *practically* unpredictable, i.e, it is so hard to predict that it appears random. Sources of true randomness include thermal noise, atmospheric noise, cosmic noise, nuclear decay, etc.

1.2 Pseudorandom Number Generator (PRNG)

The second generator type is a pseudorandom number generator (PRNG). A PRNG uses one or more inputs and generates multiple pseudorandom numbers. Inputs to PRNGs are called **seeds**. In most cases, the seed itself must be random and unpredictable. Therefore, a PRNG often obtains its seeds from the outputs of an RNG.

The outputs of a PRNG are obtained by some mathematical functions of the seed. This means although the seed generation is truly random, there is obvious deterministic nature in the process, which leads to the term we use "pseudorandom".

1.3 Randomness Testing

To test a randomly generated sequence, there are many different statistical tests available. We can characterize and describe certain properties of random numbers in terms of probability. One thing to note is that theoretically speaking, no specific set of tests is considered ¡°complete¡±. Another important thing is that it is often up to the tester to make decisions from the testing results and one must interpret the results with care and caution.

Again, in this article, we will only introduce three of the basic statistical tests, namely, the Frequency Test, Wald-Wolfowitz Runs Test and the Frequency Test within Blocks.

2 Frequency Test

2.1 Purpose

The purpose of this test is to determine whether the numbers of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence. This test is usually the first one to run because almost all other tests depend on the passing of this test. Failing this test would lead to a high probability of failing other tests.

2.2 Description

Suppose the PRNG gives us a sequence of number (b_n) where $b_n = 0$ or $b_n = 1$. To make it more convenient to deal with, we use the following transformation $x_n = 2b_n - 1$, so that $x_n = 0$ or $x_n = 1$. Now let us denote X_n as the random variable having probability $\frac{1}{2}$ of either 1 or -1. In addition, let $S_n = X_1 + X_2 + ... + X_n$.

According to the Central Limit Theorem (i.e. **Theorem** 2.2.22 on p292 of [3], the lecture slides), we know given that $X_1 ... X_n$ are independent random variables with arbitrary distributions with means $E[X_j] = \mu_j$ and variance $VarX_j = \sigma_j^2$ (all finite). Let $Y = X_1 + X_2 + ... + X_n$. Then under some general conditions, when n is large,

$$Z_n = \frac{Y - \sum \mu_j}{\sqrt{\sum \sigma_j^2}}$$

is approximately standard-normally distributed.

In our case, if the sequence is indeed random, then $X_1 ... X_n$ are independent and identical random variables with discrete uniform distribution at points -1 and 1. It is very easy to show that $E[X] = 0$ and $\sigma^2 = 1$. Therefore, using Central Limit Theorem, we have $\frac{S_n}{\sqrt{n}}$ follows a standard normal distribution Z for large n. It follows that the absolute value of $\frac{S_n}{\sqrt{n}}$, that is $\frac{|S_n|}{\sqrt{n}}$ should follow the half-normal distribution $|Z|$. Now let us derive $f(z)$, the density function of $|Z|$.

For standard normal distribution Z, we have

$$P[Z \leq z] = \phi(z).$$

Due to the symmetry of Z, we then have for $z \geq 0$,

$$\begin{aligned} P[|Z| \leq z] &= P[-z \leq Z \leq z] \\ &= 1 - 2(1 - \phi(z)) \\ &= 2\phi(z) - 1. \end{aligned}$$

Taking derivative on both sides, we get for $z \geq 0$,

$$\begin{aligned} f(z) &= 2f_Z(z) \\ &= \frac{2}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}. \end{aligned}$$

For $z < 0$,

$$\begin{aligned} P[|Z| \leq z] = 0 \\ f(z) = 0 \end{aligned}$$

We can now use MATLAB to plot the density function of the half-normal distribution. For comparison, the standard normal distribution Z is also plotted. The following figure shows the plotting result.

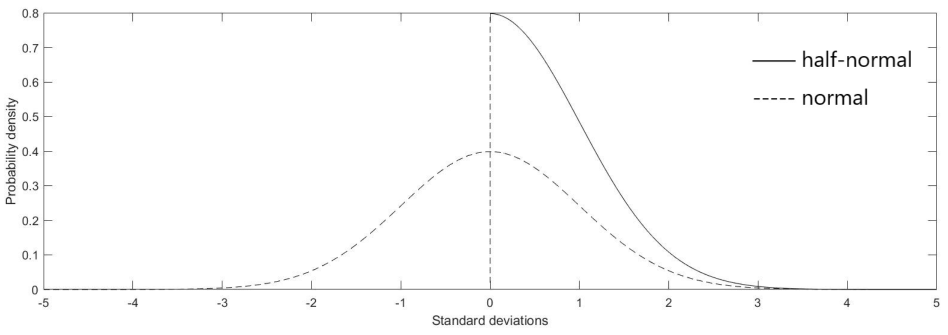

Figure 1: A plot of PDFs of standard normal distribution (dashed line) and half-normal distribution (straight line) generated by MATLAB.

Now that we have the $|Z|$ distribution, given a sample, we can calculate its P-value, and then determine whether or not it is random enough.

2.3 Testing Example

For the testing purpose, we have generated 500 pseudorandom numbers using Wolfram Mathematica (shown in Appendix A).

We can find out that there are 247 0s and 253 1s in the sequence. The value of the sum S_n that we previously defined is, therefore, $253 - 247 = 6$. Then we have

$$\frac{|s_n|}{\sqrt{n}} = \frac{6}{\sqrt{500}} = 0.268.$$

We now want to calculate the P-value. (Wolfram Mathematica has been used for the final step)

$$\begin{aligned} P - value &= P[|Z| \geq \frac{|s_n|}{\sqrt{n}}] \\ &= P[|Z| \geq 0.268] \\ &= \int_{0.268}^{+\infty} f(z)\mathrm{d}z \\ &= \int_{0.268}^{+\infty} \frac{2}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \mathrm{d}z \\ &= 0.789. \end{aligned}$$

This P-value of 0.789 is large. This means if the numbers are indeed generated randomly, it is very likely they show a pattern like this in terms of the proportions of 0s and 1s. Therefore, we fail to reject the null hypothesis that the sequence is truly random and we consider the sequence to have passed the frequency test. However, this does not promise that the sequence is indeed truly randomly generated. Again, this test is the most basic one and it cannot say too much about the sequence.

3 Wald-Wolfowitz Runs Test

3.1 Introduction

The Wald–Wolfowitz runs test, also known as the runs test, is named after Abraham Wald and Jacob Wolfowitz, two famous statisticians. A runs-test is a statistical procedure for examining whether a list of numbers of 0 and 1 is occurring randomly.

In a sequence of bits, 0 or 1, what can be always found is that some zeros or ones will show up together, as the sequence is shown below.

$$0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0$$

We call a *run* is a subsequence containing only a single type of bit and there are no other bits with the same type next to them, which means two runs next to each other must have a different type of bits, 0 or 1. For example, the sequence above contains 7 runs.

Though the probability of finding a 1 or 0 in a list of numbers could be the same, about 50%, this sequence may be not random enough because there may be a long run of 1 or 0, such that you may find a lot of same a number around one specific position you choose, which is not acceptable by the generation of pseudorandom numbers, because when the hacker knows that there could be a long list of zeros or ones in the pseudorandom numbers, the possibility of random numbers being guessed by a computer is greatly increased. The runs test can assure that a random sequence have neither too few runs nor too many.

3.2 Range, Expected Number, and Variance

Denote R the random variable representing the number of runs in a sequence of n_1 0-bits and n_2 1-bits, $n = n_1 + n_2$. Assume the hypothesis of randomness.

3.2.1 Range

For a random sequence of bits which has neither too few nor too many runs, there practically are at least 2 runs, i.e. one with all 0-bits and the other with all 1-bits. So $R \geq 2$.

For the maximum value of R, consider the $\min(n_1, n_2)$, which is n_1 if there are fewer 0-bits, or n_2 if there are fewer 1-bits.

Then if $n_1 < n_2$, the original sequence can be seen as to insert the runs with 1-bits into the sequence of runs of 0-bits. There are at most $n_1 + 1$ places to insert (i.e. $n_1 - 1$ places between any two neighbouring 0-bits, 1 place in the front, and 1 place at the end). And there can be $n_1 + 1$ runs of 1-bits since $n_2 > n_1$. So there are at most $2n_1 + 1$ runs (i.e. n_1 runs of 1 0-bit and $n_1 + 1$ runs of 1-bits).

Similarly, when $n_1 > n_2$, the original sequence can be seen as to insert the runs with 0-bits into the sequence of runs of 1-bits. There are at most $n_2 + 1$ places to insert (i.e. $n_2 - 1$ places between any two neighbouring 1-bits, 1 place in the front, and 1 place at the end). And there can be $n_2 + 1$ runs of 1-bits since $n_1 > n_2$. So there are at most $2n_2 + 1$ runs (i.e. n_2 runs of 1 0-bit and $n_2 + 1$ runs of 1-bits).

This shows that $R \leq 2\min(n_1, n_2) + 1$.

The range of R is $2 \leq R \leq 2\min(n_1, n_2) + 1$.

3.2.2 Expected Number

Label the bits in the sequence b_i where $b_i \in \{0, 1\}$ for any integer i such that $0 \leq i \leq n$.

To calculate the expected number of runs $\mathrm{E}[R] = \mu_R$, indicator random variables Z_i are used. For $1 \leq i \leq n-1$, let Z_i take the value of 1 if the ith bit is different from the $i+1$th bit, and it takes the value of 0 otherwise. Initially, there is one run, and then for every $Z_i = 1$, there is one more run. So

$$\begin{aligned} R &= 1 + \sum_{i=1}^{n-1} Z_i \\ E[R] &= E[1 + \sum_{i=1}^{n-1} Z_i] \\ &= 1 + \sum_{i=1}^{n-1} E[Z_i] \\ &= 1 + \sum_{i=1}^{n-1} P[Z_i = 1]. \end{aligned}$$

To be specific, $Z_i = 1$ is equivalent to $b_i = 0, b_{i+1} = 1$ or $b_i = 0, b_{i+1} = 1$, and so

$$\begin{aligned} P[Z_i = 1] &= P[b_i = 0, b_{i+1} = 1] + P[b_i = 1, b_{i+1} = 0] \\ &= P[b_i = 0] P[b_{i+1} = 1 \mid b_i = 0] + P[b_i = 1] P[b_{i+1} = 0 \mid b_i = 1]. \end{aligned}$$

To calculate the probabilities in the equation above, we should notice that the choosing of the first i bits is actually sampling without replacement i bits from a pool of $n = n_1 + n_2$ bits. This reminds us of the hypergeometric distribution.

Consider the random variable X_i as the number of cumulative 0-bits in the first i bits of the sequence. X_i follows the hypergeometric distribution since it can be seen as the number of bits equal to 0 that are found when sampling i bits without replacement from a pool of $n = n_1 + n_2$ bits. According to **Theorem** 1.2.23 on p104 of the lecture slides (i.e. [3]), we know that

$$\begin{aligned} E[X_{i-1}] &= (i-1)\frac{n_1}{n} \\ E[X_i] &= i\frac{n_1}{n}. \end{aligned}$$

Notice that

$$E[X_{i-1}] = \sum_{j=0}^{i-1} P[b_j = 0]$$
$$E[X_i] = \sum_{j=0}^{i} P[b_j = 0].$$

Then we have

$$P[b_i = 0] = E[X_i] - E[X_{i-1}] = \frac{n_1}{n}.$$

And so $P[b_i = 1] = 1 - \frac{n_1}{n} = \frac{n_2}{n}$. When considering $P[b_{i+1} = 0 \mid b_i = 1]$, it corresponds to sampling without replacement from a pool of $n-1$ bits (i.e. ignoring the ith bit which is already set as 1). Then as shown above, we know that

$$P[b_{i+1} = 0 \mid b_i = 1] = \frac{n_1}{n-1}.$$

Similarly, we have

$$P[b_{i+1} = 1 \mid b_i = 0] = \frac{n_2}{n-1}.$$

Finally, we have

$$\begin{aligned} P[Z_i = 1] &= P[b_i = 0]\mathrm{P}[b_{i+1} = 1 \mid b_i = 0] + P[b_i = 1]\mathrm{P}[b_{i+1} = 0 \mid b_i = 1] \\ &= \frac{n_1}{n}\frac{n_2}{n-1} + \frac{n_2}{n}\frac{n_1}{n-1} \\ &= \frac{2n_1n_2}{n(n-1)}. \end{aligned}$$

And so

$$\begin{aligned} E[R] = \mu_R &= 1 + \sum_{i=1}^{n-1} P[Z_i = 1] \\ &= 1 + \sum_{i=1}^{n-1} \frac{2n_1n_2}{n(n-1)} \\ &= 1 + \frac{2n_1n_2}{n}. \end{aligned}$$

3.2.3 Variance

We know that $\operatorname{Var} R = \mathrm{E}\left[R^2\right] - (\mathrm{E}[R])^2$, and $\mathrm{E}[R]$ is calculated, so we just need to derive $\mathrm{E}\left[R^2\right]$.

Using the expressions in 3.2.2, we have

$$\begin{aligned} R &= 1 + \sum_{i=1}^{n-1} Z_i \\ R^2 &= \left(1 + \sum_{i=1}^{n-1} Z_i\right)^2 \\ &= 1 + 2\sum_{i=1}^{n-1} Z_i + \left(\sum_{i=1}^{n-1} Z_i\right)^2 \\ &= 2R - 1 + \left(\sum_{i=1}^{n-1} Z_i\right)^2 . \end{aligned}$$

Notice that

$$\begin{aligned} \left(\sum_{i=1}^{n-1} Z_i\right)^2 &= (\sum_{i=1}^{n-1} Z_i)(\sum_{j=1}^{n-1} Z_j) \\ &= \sum_{i=1}^{n-1} {Z_i}^2 + \sum_{i\neq j, 1\leq i,j\leq n-1} Z_i Z_j \\ &= \sum_{i=1}^{n-1} {Z_i}^2 + 2\sum_{1\leq i<j\leq n-1} Z_i Z_j. \end{aligned}$$

And for $Z_i \in \{0,1\}$, clearly ${Z_i}^2 = Z_i$. So

$$\begin{aligned} \sum_{i=1}^{n-1} {Z_i}^2 = \sum_{i=1}^{n-1} Z_i &= R - 1 \\ R^2 &= 2R - 1 + \left(\sum_{i=1}^{n-1} Z_i\right)^2 \\ &= 2R - 1 + \sum_{i=1}^{n-1} {Z_i}^2 + 2\sum_{1\leq i<j\leq n-1} Z_i Z_j \\ &= 3R - 2 + 2\sum_{1\leq i<j\leq n-1} Z_i Z_j \\ E[R^2] &= 3E[R] - 2 + 2\sum_{1\leq i<j\leq n-1} E[Z_i Z_j]. \end{aligned}$$

Notice that $\mathrm{E}[Z_i Z_j] = P[Z_i = 1, Z_j = 1]$.

If $j = i + 1$, then considering the successive bits b_i, b_{i+1}, b_{i+2}, they are either $0, 1, 0$ or $1, 0, 1$. This means that

$$\begin{aligned}
P[Z_i = 1, Z_{i+1} = 1] &= P[b_i = 0, b_{i+1} = 1, b_{i+2} = 0] + P[b_i = 1, b_{i+1} = 0, b_{i+2} = 1] \\
&= P[b_i = 0]P[b_{i+1} = 1 \mid b_i = 0]P[b_{i+2} = 0 \mid b_i = 0, b_{i+1} = 1] \\
&\quad + P[b_i = 1]P[b_{i+1} = 0 \mid b_i = 1]P[b_{i+2} = 1 \mid b_i = 1, b_{i+1} = 0]
\end{aligned}$$

As shown in 3.2.2,

$$\begin{aligned}
P[b_i = 0] &= \frac{n_1}{n} \\
P[b_i = 1] &= \frac{n_2}{n} \\
P[b_{i+1} = 1 \mid b_i = 0] &= \frac{n_2}{n-1} \\
P[b_{i+1} = 0 \mid b_i = 1] &= \frac{n_1}{n-1}.
\end{aligned}$$

And for $P[b_{i+2} = 0 \mid b_{i+1} = 1, b_i = 0]$ and $P[b_{i+2} = 1 \mid b_{i+1} = 0, b_i = 1]$, they can similarly be seen as corresponding to sampling without replacement from a pool of $n - 2$ bits (i.e. ignoring the ith and $i + 1$th bits which are already set). Then as shown in 3.2.2, we know that

$$\begin{aligned}
P[b_{i+2} = 0 \mid b_{i+1} = 1, b_i = 0] &= \frac{n_1 - 1}{n-2} \\
P[b_{i+2} = 1 \mid b_{i+1} = 0, b_i = 1] &= \frac{n_2 - 1}{n-2}.
\end{aligned}$$

So

$$\begin{aligned}
P[Z_i = 1, Z_{i+1} = 1] &= \frac{n_1}{n}\frac{n_2}{n-1}\frac{n_1 - 1}{n-2} + \frac{n_2}{n}\frac{n_1}{n-1}\frac{n_2 - 1}{n-2} \\
&= \frac{n_1 n_2 (n_1 + n_2 - 2)}{n(n-1)(n-2)}.
\end{aligned}$$

If $j > i + 1$, then considering the bits $b_i, b_{i+1}, b_j, b_{j+1}$, there are 4 possible cases, which are $0, 1, 0, 1$, $0, 1, 1, 0$, $1, 0, 0, 1$, and $1, 0, 1, 0$. This means that

$$\begin{aligned}
&P[Z_i = 1, Z_j = 1] \\
=&P[b_i = 0, b_{i+1} = 1, b_j = 0, b_{j+1} = 1] + P[b_i = 0, b_{i+1} = 1, b_j = 1, b_{j+1} = 0] \\
&\quad + P[b_i = 1, b_{i+1} = 0, b_j = 0, b_{j+1} = 1] + P[b_i = 1, b_{i+1} = 0, b_j = 1, b_{j+1} = 0] \\
=&P[b_i = 0]P[b_{i+1} = 1|b_i = 0]P[b_j = 0|b_i = 0, b_{i+1} = 1]P[b_{j+1} = 1|b_i = 0, b_{i+1} = 1, b_j = 0] \\
&+ P[b_i = 0]P[b_{i+1} = 1|b_i = 0]P[b_j = 1|b_i = 0, b_{i+1} = 1]P[b_{j+1} = 0|b_i = 0, b_{i+1} = 1, b_j = 1] \\
&+ P[b_i = 1]P[b_{i+1} = 0|b_i = 1]P[b_j = 0|b_i = 1, b_{i+1} = 0]P[b_{j+1} = 1|b_i = 1, b_{i+1} = 0, b_j = 0] \\
&+ P[b_i = 1]P[b_{i+1} = 0|b_i = 1]P[b_j = 1|b_i = 1, b_{i+1} = 0]P[b_{j+1} = 0|b_i = 1, b_{i+1} = 0, b_j = 1].
\end{aligned}$$

As shown in 3.2.2,

$$\begin{aligned}
P[b_i = 0] &= \frac{n_1}{n} \\
P[b_i = 1] &= \frac{n_2}{n} \\
P[b_{i+1} = 1 \mid b_i = 0] &= \frac{n_2}{n-1} \\
P[b_{i+1} = 0 \mid b_i = 1] &= \frac{n_1}{n-1}.
\end{aligned}$$

And for P$[b_j = 0 \mid b_{i+1} = 1, b_i = 0]$, P$[b_j = 1 \mid b_{i+1} = 1, b_i = 0]$, P$[b_j = 0 \mid b_{i+1} = 0, b_i = 1]$, and P$[b_j = 1 \mid b_{i+1} = 1, b_i = 0]$, they can similarly be seen as corresponding to sampling without replacement from a pool of $n-2$ bits (i.e. ignoring the ith and $i+1$th bits which are already set). Then as shown in 3.2.2, we know that

$$\begin{aligned}
P[b_j = 0 \mid b_{i+1} = 1, b_i = 0] &= \frac{n_1 - 1}{n-2} \\
P[b_j = 1 \mid b_{i+1} = 1, b_i = 0] &= \frac{n_2 - 1}{n-2} \\
P[b_j = 0 \mid b_{i+1} = 0, b_i = 1] &= \frac{n_1 - 1}{n-2} \\
P[b_j = 1 \mid b_{i+1} = 0, b_i = 1] &= \frac{n_2 - 1}{n-2}.
\end{aligned}$$

And also for P$[b_{j+1} = 1 \mid b_j = 0, b_{i+1} = 1, b_i = 0]$, P$[b_{j+1} = 0 \mid b_j = 1, b_{i+1} = 1, b_i = 0]$, P$[b_{j+1} = 1 \mid [b_j = 0, b_{i+1} = 0, b_i = 1]$, and P$[b_{j+1} = 0 \mid b_j = 1, b_{i+1} = 1, b_i = 0]$, they can similarly be seen as corresponding to sampling without replacement from a pool of $n-3$ bits (i.e. ignoring the ith, $i+1$th, and jth bits which are already set). Then as shown in 3.2.2, we know that

$$\begin{aligned}
P[b_{j+1} = 1 \mid b_j = 0, b_{i+1} = 1, b_i = 0] &= \frac{n_2 - 1}{n-3} \\
P[b_{j+1} = 0 \mid b_j = 1, b_{i+1} = 1, b_i = 0] &= \frac{n_1 - 1}{n-3} \\
P[b_{j+1} = 1 \mid [b_j = 0, b_{i+1} = 0, b_i = 1] &= \frac{n_2 - 1}{n-3} \\
P[b_{j+1} = 0 \mid b_j = 1, b_{i+1} = 1, b_i = 0] &= \frac{n_1 - 1}{n-3}.
\end{aligned}$$

So

$$\begin{aligned}
P[Z_i = 1, Z_j = 1] &= \frac{n_1}{n}\frac{n_2}{n-1}\frac{n_1-1}{n-2}\frac{n_2-1}{n-3} + \frac{n_2}{n}\frac{n_1}{n-1}\frac{n_2-1}{n-2}\frac{n_1-1}{n-3} \\
&\quad + \frac{n_1}{n}\frac{n_2}{n-1}\frac{n_2-1}{n-2}\frac{n_1-1}{n-3} + \frac{n_2}{n}\frac{n_1}{n-1}\frac{n_1-1}{n-2}\frac{n_2-1}{n-3} \\
&= \frac{4n_1n_2(n_1-1)(n_2-1)}{n(n-1)(n-2)(n-3)}.
\end{aligned}$$

Finally, we have

$$\begin{aligned}
\sum_{1\le i<j\le n-1} E[Z_iZ_j] &= \sum_{1\le i\le n-2} P[Z_i = 1, Z_{i+1} = 1] + \sum_{1\le i<i+1<j\le n-1} P[Z_i = 1, Z_j = 1] \\
&= \sum_{1\le i\le n-2} \frac{n_1n_2(n_1+n_2-2)}{n(n-1)(n-2)} + \sum_{1\le i<i+1<j\le n-1} \frac{4n_1n_2(n_1-1)(n_2-1)}{n(n-1)(n-2)(n-3)} \\
&= (n-2)\frac{n_1n_2(n_1+n_2-2)}{n(n-1)(n-2)} + \sum_{i=1}^{n-3}(n-i-2)\frac{4n_1n_2(n_1-1)(n_2-1)}{n(n-1)(n-2)(n-3)} \\
&= \frac{n_1n_2(n_1+n_2-2)}{n(n-1)} + \frac{(n-2)(n-3)}{2}\frac{4n_1n_2(n_1-1)(n_2-1)}{n(n-1)(n-2)(n-3)} \\
&= \frac{n_1n_2(n_1+n_2-2)}{n(n-1)} + \frac{2n_1n_2(n_1-1)(n_2-1)}{n(n-1)} \\
&= \frac{n_1n_2(2n_1n_2-n)}{n(n-1)} \\
E[R^2] &= 3E[R] - 2 + 2\sum_{1\le i<j\le n-1} E[Z_iZ_j] \\
&= 1 + \frac{6n_1n_2}{n} + \frac{2n_1n_2(2n_1n_2-n)}{n(n-1)} \\
&= 1 + \frac{2n_1n_2(2n_1n_2+2n-3)}{n(n-1)}.
\end{aligned}$$

And so

$$\begin{aligned}
\operatorname{Var} R &= E\left[R^2\right] - (E[R])^2 \\
&= 1 + \frac{2n_1n_2(2n_1n_2+2n-3)}{n(n-1)} - (1+\frac{2n_1n_2}{n})^2 \\
&= \frac{2n_1n_2(2n_1n_2-n)}{n^2(n-1)}.
\end{aligned}$$

3.3 Probabilities

3.3.1 $P[R = 2k]$ (R is even)

As proved above, $2 \leq R \leq 2\min(n_1, n_2) + 1$, when $R = 2k$ (R is even), the problem can be transformed to a distribution problem. If the first run is for 1-bits, we need to place n_1 0-bits to $\frac{R}{2} = k$ places, and n_2 1-bits should be placed to $\frac{R}{2} = k$ places, too. The schematic diagram is shown in Fig.2.

is even.png is even.bb

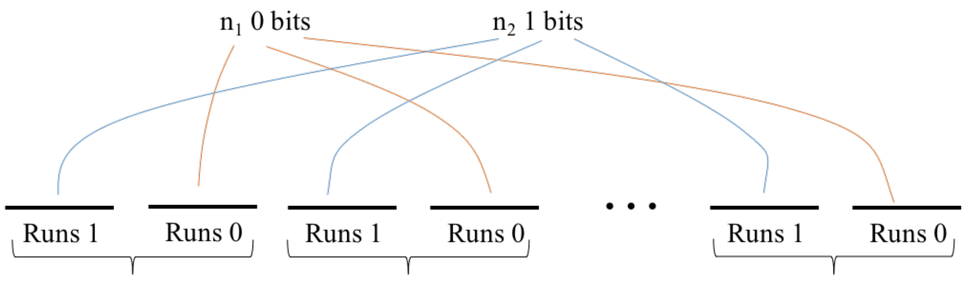

Figure 2: Schematic diagram when $R = 2k$.

When the first *run* consists of 1-bits, the second *run* should consist of 0-bits, and this constitutes a cyclical cycle. To place n_1 0-bits to $\frac{R}{2} = k$ places, it's the same as place $k - 1$ separators to the $n_1 - 1$ position between characters (as shown in Fig.3). This way, we can get k runs.

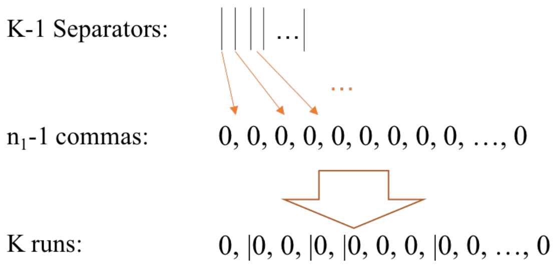

Figure 3: Separators placing.

So this problem becomes a combination problem. And there are

$$\binom{n_1 - 1}{k - 1}$$

ways of placing $k - 1$ separators into the $n_1 - 1$ spaces between the zeros, with no more than one separator per space.

So for ones, it can be exact same process to get the ways of separating n_2 1-bits with $k-1$ separators. With the same process, we can calculate that there are

$$\binom{n_2-1}{k-1}$$

ways of placing $k-1$ separator into the n_2-1 spaces between the 1-bits, with no more than one separator per space. According to the multiplication rule, the number of ways to put these two sets of runs together to form $R=2k$ runs beginning with a run with 1-bits is

$$\binom{n_2-1}{k-1}\binom{n_1-1}{k-1}.$$

In the case that this system started with runs of 0-bits, the number of ways is

$$\binom{n_2-1}{k-1}\binom{n_1-1}{k-1},$$

which is the same as the one that begins with the run of 1-bits. To find the ways of all possibilities, it can be transformed to the question: how many ways are there to separate a group with n_1+n_2 numbers into two parts? Because in this condition, all the possibilities to place runs are considered. And the number of ways is

$$\binom{n_1+n_2}{n_1}.$$

In conclusion, the probability for $R=2k$ is

$$\begin{aligned} P[R=2k] &= \frac{\binom{n_1-1}{k-1}\binom{n_2-1}{k-1}+\binom{n_2-1}{k-1}\binom{n_1-1}{k-1}}{\binom{n_1+n_2}{n_1}} \\ &= \frac{2\binom{n_1-1}{k-1}\binom{n_2-1}{k-1}}{\binom{n_1+n}{n_1}}. \end{aligned}$$

3.3.2 $P[R=2k+1]$ (R is odd)

As proved above, $2 \leq R \leq 2\min(n_1,n_2)+1$, when $R=2k+1$ and R is odd, the problem can be transformed to a distribution problem. If the first run is for 1-bits, we need to place n_1 0-bits to $\frac{R-1}{2} = k$ places, and n_2 1-bits should be placed to $\frac{R+1}{2} = k + 1$ places. The schematic diagram is shown in Fig.4.

is odd.png is odd.bb

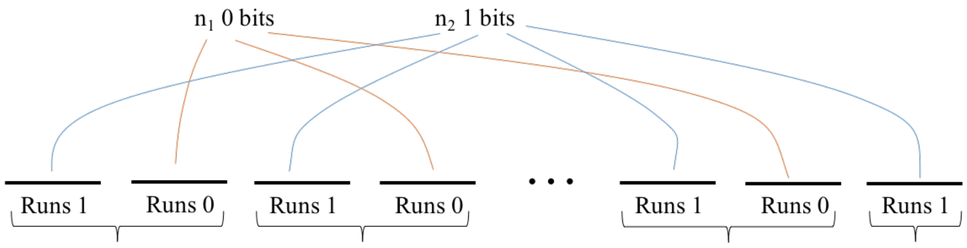

Figure 4: Schematic diagram when $R = 2k + 1$

In this case, there are $k + 1$ runs for n_2 1-bits and k runs for n_1 0-bits. So there are

$$\binom{n_2 - 1}{k}$$

ways of placing $k + 1 - 1 = k$ separators into the $n_2 - 1$ spaces between the 1 bits, with no more than one separator per space. In this case, for 0-bits, it can be exact same process to get the ways of separating n_1 0-bits with $k - 1$ separator. With the same process, we can calculate that there are

$$\binom{n_1 - 1}{k - 1}$$

ways of placing $k-1$ separator into the n_1-1 spaces between the 0-bits, with no more than one separator per space. According to the multiplication rule, the number of ways to put these two sets of runs together to form $R = 2k+1$ runs beginning with a run 1 bits is

$$\binom{n_2 - 1}{k}\binom{n_1 - 1}{k - 1}.$$

In the case that this system started with runs of 0-bits, the number of ways is

$$\binom{n_1 - 1}{k}\binom{n_2 - 1}{k - 1}.$$

In 3.3.1, It has been calculated that the number of all ways of placing runs is

$$\binom{n_1+n_2}{n_1}.$$

In conclusion, the probability for $R = 2k+1$ is

$$P[R = 2k+1] = \frac{\binom{n_1-1}{k}\binom{n_2-1}{k-1} + \binom{n_2-1}{k}\binom{n_1-1}{k-1}}{\binom{n_1+n_2}{n_1}}.$$

3.4 Exact Test

For the exact test, the exact probability expressions given in 3.3 are applied.

3.4.1 Description

The run test is defined with the null hypothesis

$$H_0 : \text{the sequence was produced in a random manner,}$$

and the alternative hypothesis

$$H_a : \text{the sequence was not produced in a random manner.}$$

It should be a two-tailed test since too few runs mean that there is likely to be not enough variation for a random process, and too many runs mean too frequent oscillation for a random process.

So we have the p-value

$$P = P\left[|R - \mu_R| \geq |r - \mu_R|\right]$$

where $\mu_R = \mathrm{E}[R] = 1 + \frac{2n_1n_2}{n}$ is the expected number of runs under the hypothesis of randomness, and r is the number of runs in the tested sequence.

We can only reject the null hypothesis of randomness when P-value is small enough.

3.4.2 Application

Using the pseudorandom sequence of bits in Appendix A, we can find the value of R, denoted by r, to be 261, $n_1 = 247$, and $n_2 = 253$ (with the help of Wolfram Mathematica as shown in Appendix B).

Then μ_R can be calculated as

$$\mu_R = \frac{2n_1n_2}{n} + 1 = \frac{62741}{250} \approx 250.964$$

.

Thus P value can be calculated as

$$\begin{aligned} P &= P\left[|R - \mu_R| \geq |r - \mu_R|\right] \\ &= P\left[|R - 250.964| \geq |261 - 250.964|\right] \\ &= 1 - P[240.982 < R < 261] \\ &= 1 - \sum_{k=241}^{260} P[R = k] \\ &= 0.371. \end{aligned}$$

Using the equations for P$[R = k]$ shown in 3.3, the probabilities can be calculated to be 0.376 (as shown in Appendix B).

There is not enough evidence to reject the hypothesis of randomness since P-value is not small enough.

3.5 Normal Approximation

3.5.1 Description

According to the Central Limit Theorem, for a large n

$$Z = \frac{R - \mu}{\sqrt{\operatorname{Var} R}}$$

is approximately standard normal.

Notice that R can only be integers, so continuity-corrected standard normal statistic

$$Z_{cc} = \begin{cases} \frac{R - \mu_R - 0.5}{\sqrt{\operatorname{Var} R}} & \text{if } R \geq \mu_R \\ \frac{R - \mu_R + 0.5}{\sqrt{\operatorname{Var} R}} & \text{if } R < \mu_R \end{cases}$$

should be considered.

And then the corresponding P-value should be

$$P = \begin{cases} P[Z_{cc} > \frac{r - \mu_R - 0.5}{\sqrt{\operatorname{Var} R}}] + P[Z_{cc} < \frac{\mu_R - r + 0.5}{\sqrt{\operatorname{Var} R}}] & \text{if } r \geq \mu_R \\ P[Z_{cc} > \frac{r - \mu_R + 0.5}{\sqrt{\operatorname{Var} R}}] + P[Z_{cc} < \frac{\mu_R - r - 0.5}{\sqrt{\operatorname{Var} R}}] & \text{if } r < \mu_R \end{cases}$$

3.5.2 Application

Using the pseudorandom sequence of bits in Appendix A, we can find the value of R, denoted by r, to be 261, $n_1 = 247$, and $n_2 = 253$.

Then μ_R can be calculated as

$$\mu_R = \frac{2n_1n_2}{n} + 1 = \frac{62741}{250} \approx 250.964.$$

And $\operatorname{Var} R$ can be calculated as

$$\operatorname{Var} R = \frac{2n_1n_2(2n_1n_2 - n)}{n^2(n-1)} \approx 124.714$$

So the P-value is

$$\begin{aligned}
P &= P[Z_{cc} > \frac{r - \mu_R - 0.5}{\sqrt{\operatorname{Var} R}}] + P[Z_{cc} < \frac{\mu_R - r + 0.5}{\sqrt{\operatorname{Var} R}}] \\
&= P[Z_{cc} > \frac{261 - 250.964 - 0.5}{\sqrt{124.714}}] + P[Z_{cc} < \frac{250.964 - 261 + 0.5}{\sqrt{124.714}}] \\
&= P[Z_{cc} > 0.854] + P[Z_{cc} < -0.854] \\
&= 0.393.
\end{aligned}$$

The result is close to that of the exact test. And since P is not small enough, there is not enough evidence to reject the null hypothesis of randomness.

4 Frequency Test Within a Block

After doing research, we find that *frequency test within a block* is an appropriate method to test the randomness of a sequence. Before applying the method to the sequence randomly generated in the previous part, we will briefly introduce the working principle of the test.

4.1 Introduction

The basic idea of the test is to divide the n-bit sequence ε into N non-overlapping blocks with length M, where $N = \lfloor \frac{n}{M} \rfloor$. Discard any unused bits.

For example, if $n = 13$, $M = 4$ and $\varepsilon = 0100110110101$, 3 blocks would be created, consisting of 0100, 1101 and 1010. The final 1 will be discarded.

Then, we determine the proportion of **ones** in each block.To be specific, the n^{th} bit of the sequence (ε) is called b_n, which can be either 0 or 1. Thus, the proportion of ones in the i^{th} block, denoted as π_i, can be defined as

$$\pi_i = \frac{\sum_{j=1}^{M} b_{(i-1)M+j}}{M},$$

where $1 \leq i \leq N$.

For the example above, $\pi_1 = \frac{1}{4}$, $\pi_2 = \frac{3}{4}$ and $\pi_3 = \frac{1}{2}$.

Then, in the following section, we will examine the extent to which π_i in each block is approximately $\frac{1}{2}$, namely whether there are approximately $\frac{M}{2}$ ones in each block.

To make the test more accurate, we require that $n \geq 100$ and $M \geq Max\{20, 0.01n\}$.

4.2 Normal Approximation of π

If the tested sequence is composed of random numbers, the probability of the occurrence of 0 and 1 in the sequence should be identical and independent, following the binomial distribution. Therefore, the π_i are also independent and can be approximated as a normal distribution. The procedure of determining the mean and variance is shown below.

Since ε is composed of random numbers, obviously we can conclude that $E[\pi_i] = \frac{1}{2}$ for any i.

Now we focus on the variance of π_i. In terms of a sequence with M values, according to the Cardano's principle, the probability that the proportion π equals to $\frac{k}{M}$ is that

$$\begin{aligned} P[\pi_i = \frac{k}{M}] &= \frac{\text{number of M-bit sequences with k ones}}{\text{number of all possible M-bit sequences}} \\ &= \frac{\binom{M}{k}}{\sum_{i=1}^{M}\binom{M}{i}} = \frac{\binom{M}{k}}{2^M}. \end{aligned}$$

Then we can express the 2^{nd} *moment* of π_i as follows

$$\begin{aligned} E[\pi_i^2] &= \sum_{k=0}^{M}\{(\frac{k}{M})^2 \cdot P[\pi = \frac{k}{M}]\} \\ &= \frac{1}{2^M}\sum_{k=0}^{M}[(\frac{k}{M})^2 \cdot \binom{M}{k}]. \end{aligned}$$

Since

$$(\frac{k}{M})^2\binom{M}{k} = \frac{k^2}{M^2}\cdot\frac{M!}{(M-k)!k!} = \frac{(M-1)!k}{M(k-1)!(M-k)!} = \frac{k}{M}\binom{M-1}{k-1}.$$

We can get that

$$\begin{aligned}
E[\pi_i^2] &= \frac{1}{2^M}\sum_{k=1}^{M}[\frac{k}{M}\binom{M-1}{k-1}] \\
&= \frac{1}{M\cdot 2^M}\sum_{k=1}^{M}[(M-k+1)\binom{M-1}{M-k}](\text{Substitute } k \text{ with } M+1-k) \\
&= \frac{1}{M\cdot 2^M}\sum_{k=1}^{M}[(M-k+1)\binom{M-1}{k-1}] \\
&= \frac{1}{M\cdot 2^M}\{[\sum_{k=1}^{M-1}(M-k+1)\binom{M-1}{k-1}]+1\} \\
&= \frac{1}{M\cdot 2^M}\{[\sum_{k=1}^{M-1}(M-1)\binom{M-2}{k-1}+\binom{M-1}{k-1}]+1\} \\
&= \frac{1}{M\cdot 2^M}[(M-1)\sum_{k=0}^{M-2}\binom{M-2}{k}+\sum_{k=0}^{M-1}\binom{M-1}{k}] \\
&= \frac{(M-1)\cdot 2^{M-2}+2^{M-1}}{M\cdot 2^M} = \frac{M+1}{4M}.
\end{aligned}$$

Therefore, the variance of π_i is that

$$\sigma^2 = E[\pi_i^2]-(E[\pi_i])^2 = \frac{1+M}{4M}-\frac{1}{4} = \frac{1}{4M}.$$

Given that the mean value of π_i is $\frac{1}{2}$ and the variance is $\frac{1}{4M}$, we can conclude that $4M(\pi_i-\frac{1}{2})$ follows the standard normal distribution.

4.3 Chi-squared distribution of $\chi^2_{(obs)}$

To combine all the π_i together, we introduce a new parameter and use it as a test statistic, namely $\chi^2_{(obs)}$. It is defined as

$$\chi^2_{(obs)} = 4M\sum_{i=1}^{N}(\pi_i-1/2)^2.$$

We notice that if the sequence is random, $\chi^2_{(obs)}$ will be the sum of squares of N independent standard normal variables, so it follows the chi-squared distribution with N degrees of freedom.

To test the randomness of the sequence, we need to derive a P-value. First, we set the null hypothesis

$$H_0 : \varepsilon \ is \ random.$$

We assume that H_0 is true, namely $\chi^2_{(obs)}$ follows the chi-squared distribution. Then, we use the two-tailed test to calculate the P-value

$$\begin{aligned} P - value &= 2 \times Min\{P[\chi^2_N \leq \chi^2_{(obs)}|H_0], P[\chi^2_N \geq \chi^2_{(obs)}|H_0]\} \\ &= \frac{2}{\Gamma(N/2)2^{N/2}} Min\{\int_{\chi^2_{(obs)}}^{\infty} e^{-u/2}u^{N/2-1}\mathrm{d}u, \int_0^{\chi^2_{(obs)}} e^{-u/2}u^{N/2-1}\mathrm{d}u\}. \end{aligned}$$

After calculating the P-value, there are 2 possibilities.

- $P - value < 0.01$

  We reject H_0 at the 1% level of significance, which means that we accept that the sequence is not random.

- $P - value \geq 0.01$

  We fail to reject H_0 at the 1% level of significance, which means that we accept that the sequence is random.

After we do the comparison, the test is finished.

4.4 Application with Data

Here we test the 500 pseudo-random data generated in Appendix A with the method of *frequency test within a block*.

First of all, to make the result more accurate, we should satisfy $M \geq Max\{20, 0.01n\}$. Since $n = 500$, we can determine $M = 25$ here and, therefore, $N = 20$.Then we divide the 500-bit sequence(20 $\times$ 25 matrix) by the row, i.e., each row is a block of this sequence. We can get the proportion of **ones** for each block in Table 1.

Table 1: The value of π_i

i	**The value of π_i**	i	**The value of π_i**
1	13/25	11	13/25
2	12/25	12	11/25
3	13/25	13	12/25
4	15/25	14	11/25
5	11/25	15	12/25
6	13/25	16	15/25
7	15/25	17	12/25
8	13/25	18	12/25
9	15/25	19	8/25
10	13/25	20	14/25

According to the π_i collected, we can draw the histogram of π_i in figure 5. We can find that the shape of the histogram seems like a normal distribution, which is in accordance with our deduction in section 4.2.

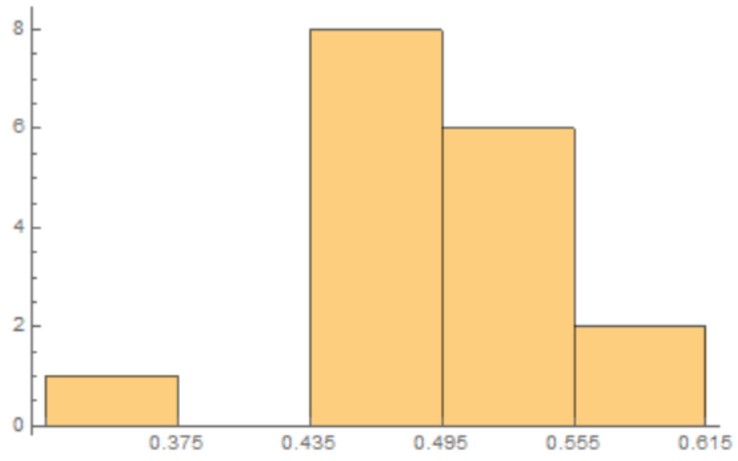

Figure 5: The histogram of π_i

Next, we calculate the test statistic $\chi^2_{(obs)}$ with the equation $\chi^2_{(obs)} = 4M\sum_{i=1}^{N}(\pi_i - 1/2)^2$. We can acquire that

$$\begin{aligned}\chi^2_{(obs)} &= 4 \times 25 \times [(\frac{8}{25} - \frac{1}{2})^2 + 3(\frac{11}{25} - \frac{1}{2})^2 + 5(\frac{12}{25} - \frac{1}{2})^2 + 6(\frac{13}{25} - \frac{1}{2})^2 + (\frac{14}{25} - \frac{1}{2})^2 + 4(\frac{15}{25} - \frac{1}{2})^2] \\ &= 9.12.\end{aligned}$$

Since $\chi^2_{(obs)}$ here is a chi-squared distribution with 20 degrees of freedom, we can determine the P-value of the hypothesis (H_0) that the sequence is random.

With the help of Matlab, we can calculate that

$$\int_0^{9.12} f_{\chi^2_{20}}(x)\mathrm{d}x = 1.85\%.$$

Due to the fact that H_0 is two-tailed, P-value is the twice of the integral, which is 3.7%, i.e., we reject H_0 at the 3.7% level of significance. Compared with the standard in section 4.3, P-value $> 1\%$, we can conclude that the sequence is random.

Moreover, we can deduce the randomness of the sequence based on the critical region. Based on the standard 1% level of significance, with a probability of $1-1\%$, $\chi^2_{1-0.01/2,20} \leq \chi^2_{(obs)} \leq \chi^2_{0.01/2,20}$. If H_0 is true, the probability that $\chi^2_{(obs)} > \chi^2_{0.01/2,20}$ or $\chi^2_{(obs)} < \chi^2_{1-0.01/2,20}$ is equal to 1%. Therefore, the critical region is determined by

$$0 < \chi^2_{(obs)} < \chi^2_{1-0.01/2,20} = 7.44 \text{ or } \chi^2_{(obs)} > \chi^2_{0.01/2,20} = 40,$$

which is shown as the shading area in Figure 6.

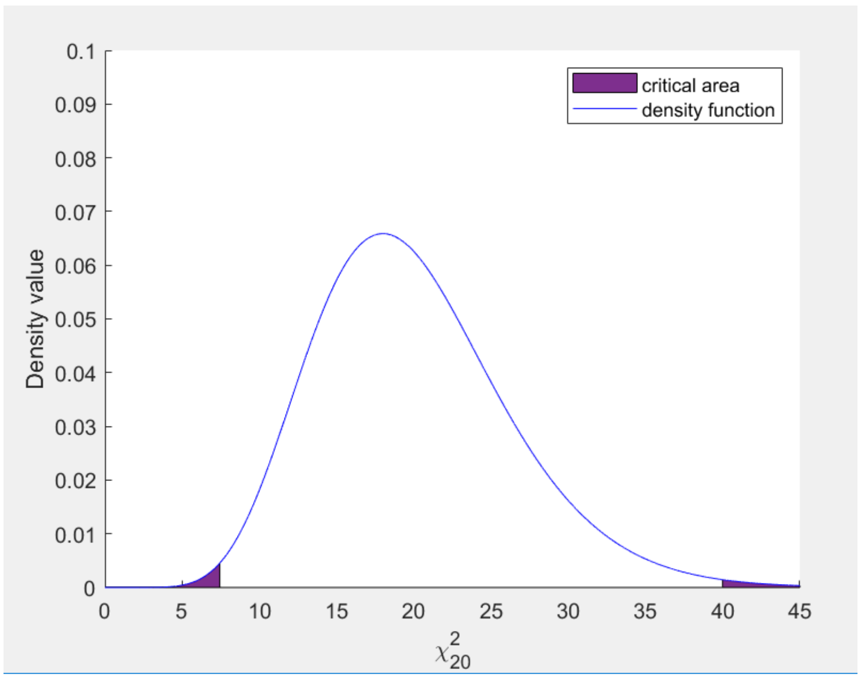

Figure 6: The critical region of $\chi^2_{(obs)}$

Since $\chi^2_{(obs)}$ of the tested sequence is 9.12, which is not located in the critical region, it is indicated that the sequence is random, which corresponds to the results before.

5 Conclusion

In this article, we talk about random and pseudorandom numbers and focus on how to determine if a generated number sequence is sufficiently random-looking. We have introduced three useful statistical tests to help us investigate its randomness, namely, the Frequency Test, Wald-Wolfowitz Runs Test and the Frequency Test within Blocks.

For Frequency Test, we test how close the proportion of 0s and 1s is to 1/2 as it would be for random numbers. For Wald-Wolfowitz Runs Test, we focus on the number of runs in the sequence and test if the sequence has an acceptable number of runs, which should be neither too few nor too many. For the Frequency Test within Blocks, which serves as an improvement of the Frequency Test, we apply the Frequency Test in each block and put them together through chi-squared distribution. By doing so, both the whole frequency and the distribution uniformity in each block can be investigated.

A pseudorandom sequence of 500 bits is generated, and the 3 tests are applied to it. We find out that the P-value in each case is not small enough for us to reject the null hypothesis. The sequence is considered to have passed the three tests. Therefore, we fail to reject the null hypothesis that it is random.

References

[1] L. Bassham, III, A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. *SP 800-22 Rev. 1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.* National Institute of Standards & Technology, Gaithersburg, MD, United States, 2010. https://csrc.nist.gov/publications/detail/sp/ 800-22/rev-1a/final.

[2] Wikipedia contributors. Wald-Wolfowitz runs test – Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Wald-Wolfowitz_runs_test, 2019. [Online; accessed 4-July-2019].

[3] H. Hohberger. *Probabilistic Methods in Engineering.* University of Michigan - Shanghai Jiaotong University Joint Institute, Shanghai Jiao Tong University, Shanghai, China, 2019.

Appendix A Pseudo-random Data Applied

The pseudo-random data applied are generated with the command

$$\textbf{RandomInteger}[1, \{20, 25\}]$$

in Wolfram Mathematica. The generated data are shown as follows.

$$\begin{pmatrix}
1&0&1&0&0&1&0&1&1&0&0&1&1&0&1&1&1&0&0&0&1&0&1&0&1\\
0&0&1&0&1&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0&1&0&1&0&1\\
1&0&1&1&1&0&1&1&1&0&1&0&0&0&1&0&1&0&1&1&0&1&0&0&0\\
0&0&0&1&1&1&0&0&1&1&0&0&1&1&0&1&1&0&1&0&1&1&1&1&1\\
1&0&1&1&0&1&0&0&0&1&1&1&1&0&0&0&1&0&0&0&0&0&1&0&1\\
1&0&1&0&1&0&1&1&0&1&1&1&0&0&1&0&0&0&1&0&1&0&0&1&1\\
0&1&0&1&1&1&1&0&1&1&0&0&0&1&1&1&1&1&0&0&0&1&1&1&0\\
1&0&1&1&1&1&1&0&1&0&0&1&0&0&0&0&0&0&1&1&0&1&1&1&0\\
0&1&0&1&1&0&1&0&0&1&0&1&1&1&1&1&1&1&0&0&1&0&1&0&1\\
1&0&1&0&1&1&1&1&0&0&0&0&0&1&1&0&0&0&0&1&1&1&1&0&1\\
0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&0&0&1&0&1&1&1&1&1&0\\
0&0&1&0&1&1&1&0&0&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&0\\
0&1&1&1&0&1&0&1&0&1&1&0&1&0&0&0&1&1&1&0&0&0&1&0&0\\
0&0&0&0&0&1&1&0&0&0&0&0&0&1&1&1&1&0&0&1&1&1&1&1&0\\
1&1&0&0&0&0&0&0&1&1&1&1&0&1&1&0&0&1&1&1&0&0&0&1&0\\
1&1&0&0&1&1&1&1&0&1&1&0&1&1&1&1&1&0&1&0&0&0&0&0&1\\
0&1&0&1&0&1&1&0&1&0&0&1&0&0&1&1&0&0&0&1&1&0&1&0&1\\
0&0&1&1&0&1&0&0&0&1&1&0&0&1&1&0&1&0&1&1&0&1&0&0&1\\
0&0&1&1&0&0&1&0&1&0&0&0&0&0&0&0&1&0&0&1&1&0&1&0&0\\
0&0&1&0&0&1&1&0&1&1&1&1&0&0&0&0&1&0&1&0&1&1&1&1&1
\end{pmatrix}$$

Appendix B Wolfram Mathematica Codes for 3.4.2

```
list = {{1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1},
{0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1},
{1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0},
{0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1},
{1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1},
{1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1},
{0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0},
{1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0},
{0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1},
{1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1},
{0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0},
{0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0},
{0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0},
{0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0},
{1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0},
{1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1},
{0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1},
{0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1},
{0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0},
{0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1}}

nlist = Flatten[list]
```

```
n1 = Length@Select[nlist, # == 0&]
```

```
247
```

```
n2 = Length@Select[nlist, # == 1&]
```

```
253
```

```
ER = (2 * n1 * n2)/(n1 + n2) + 1
```

$\frac{62741}{250}$

```
r = Length[Split[nlist]]
```

```
261
```

```
P[R_]:=If[Mod[R, 2]==0,
2 * Binomial[n1 − 1, R/2 − 1] * Binomial[n2 − 1, R/2 − 1]/Binomial[n1 + n2, n1],
1/Binomial[n1 + n2, n1]
(Binomial[n1 − 1, (R − 1)/2] * Binomial[n2 − 1, (R − 1)/2 − 1]+
Binomial[n2 − 1, (R − 1)/2] * Binomial[n1 − 1, (R − 1)/2 − 1])]
```

$1 - \sum_{R=241}^{260} P[R]//N$

```
0.370972
```