Name: Seung Woo Choi
Project Topic: 17, Tausworthe, 1 member

Implementation and Analysis of the Tausworthe Pseudo-Random Number Generator

Abstract:

In this project, I built a successful implementation of the Tausworthe pseudo-random number generator using reasonably large values of the parameters r, q, and l. The maximum values for the parameters were set as: 15 for q and 15 for l. These values were carefully selected to illustrate the mechanics of the Tausworthe generator while ensuring that R would be able to perform the computations in a timely manner.

I also performed multiple statistical tests on the Tausworthe generator to verify that it produces PRNs that are approximately i.i.d. Uniform(0,1), which it did. The statistical tests include the chi-square test to check the goodness of fit, the runs test for independence, and the Kolmogorov-Smirnov test to see what type of distribution the data has. I also plotted a histogram to visualize the distribution of the data.

Adjacent PRNs were then plotted on the unit square to investigate patterns. Based on the results, the adjacent PRNs did not have a pattern because they are random.

Lastly, I applied the inverse transform theorem and Box-Muller method to transform Uniform(0,1) variables to Normal(0,1) variables to showcase the Tausworthe generator in action.

Background and Description of Problem:

The Tausworthe generator competes with variations of the linear congruential generator (LCG) and is regarded as a good PRN generator. This generator is more commonly used in the field of computer science and utilizes binary digits in its computations. Important equations for the Tausworthe generator include the below:

Figure 1. Tausworthe generator equations

(1) $B_i = (\sum_{j=1}^{q} c_j B_{i-j}) \; mod \; 2$ where $c_j = 0 \; or \; 1$

(2) $B_i = (B_{i-r} + B_{i-q}) \; mod \; 2 = B_{i-r} XOR \; B_{i-q}$ for $0 < r < q$

(3) $(l \; bits \; in \; base \; 2) \; / \; 2^l$

Equation (1) is used to define a sequence of binary digits (e.g. B_1, B_2, ...). Equation (2) shows a common implementation to compute B_i values for the Tausworthe generator. Once the B_i's are computed, equation (3) can be applied to get the Uniform(0,1) variables.

In the sections below, I discuss the major findings, what the results of the code demonstrate, and the conclusion.

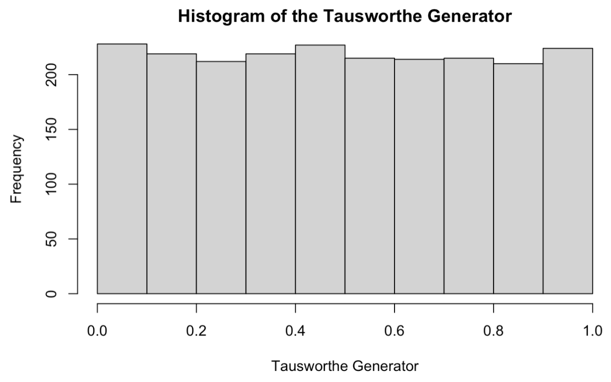
Main Findings:

The problems at hand involve building the Tausworthe generator algorithm from scratch and implementing the generator for reasonably large values of r, q, and l, performing multiple statistical tests to confirm that the generator produces i.i.d. Uniform(0,1) values, plotting adjacent PRNs on the unit square to check for any patterns, and transforming Uniform(0,1) random variables into Normal(0,1) random variables.

The first task in this project required building a Tausworthe generator from scratch using reasonably large values for r, q, and l. I utilized the three equations outlined in the **Background and Description of Problem** section and designed a function that checks if proper r, q, and l values are provided as input, initializes and elongates the B_i sequence, and then converts bits from base 2 to base 10 to produce a final Uniform(0,1) output. I experimented with various values for r, q, and l but decided to limit the values for q and l to 15 and 15, respectively. I chose 15 for q because the period when q = 15 is 32767, which is relatively a long cycle length. I selected 15 for l because I specified a maximum period of $2^{15} - 1 = 32767$ for all function calls, so at a minimum, when l = 15, the length of the Tausworthe generator output would be 2184 Uniform(0,1) variables. The limit on r is not too important as long as it is greater than 0 and less than q. Under these constraints, the number of Uniform(0,1) random variables is approximately equal to $ceiling((period - l) / l)$, which can range from 2,184 values (when l = 15) to 16,383 values (when l = 2).

The second task involved performing various statistical tests to verify that the generator produces i.i.d. Uniform(0,1) variables. For this part, I plotted a histogram and performed 3 statistical tests. Figure 2 below shows the distribution of the data created by the generator using r = 9, q = 10, and l = 15.

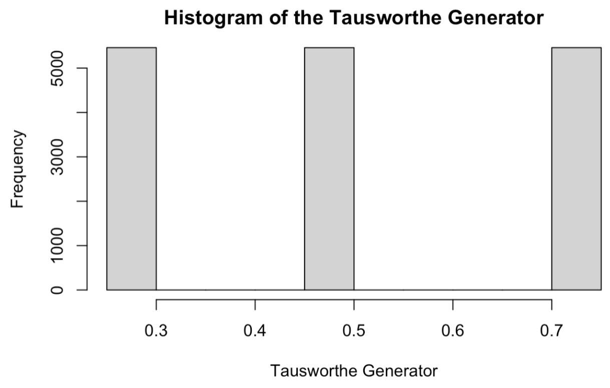
Figure 2. Histogram of the Tausworthe generator using good input values



Once the histogram was plotted, I then performed a chi-squared test for goodness-of-fit, a runs test to check if the data are independent, and a Kolmogorov-Smirnov test to check that the distribution of the data is uniform. For the chi-squared test, the null hypothesis is that the data all have equal probabilities of occurring, and the alternative hypothesis is that the data do not have equal probabilities of occurring. Since the p-value of 1 is greater than 0.05, we fail to reject the null hypothesis that the data all have equal probabilities of occurring. This means that the data have equal probabilities of occurring. For the runs test, the null hypothesis is that the data was produced randomly, and the alternative hypothesis is that the data was not produced randomly. Since the p-value of 0.3915 is greater than 0.05, we fail to reject the null hypothesis that the data was produced randomly, which means that the data was produced randomly. For the Kolmogorov-Smirnov test, the null hypothesis is that the data comes from a Uniform(0,1) distribution, and the alternative hypothesis is that the data does not come from a Uniform(0,1) distribution. Since the p-value of 0.9998 is greater than 0.05, we fail to reject the null hypothesis that the data comes from a Uniform(0,1) distribution. Thus, this means that the data does come from a Uniform(0,1) distribution. Through the three statistical tests, there is statistically significant evidence that supports that the Tausworthe generator produces i.i.d. Uniform(0,1) PRNs when $r = 9$, $q = 10$, and $l = 15$.

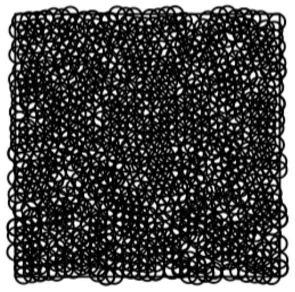
While the Tausworthe generator produces i.i.d. Uniform(0,1) PRNs with the aforementioned r, q, and l values, the generator does not always generate good PRNs. By experimenting with different values of r, q, and l, I noticed that the Tausworthe generator fails different statistical tests depending on the values of r, q, and l. The combinations of r, q, and l values that produce less-than-ideal results do not appear to have a pattern, but they do exist. For example, the Tausworthe generator outputs when $r = 1$, $q = 2$, and $l = 2$ fail each of the three aforementioned statistical tests. The histogram created by these inputs are shown in Figure 3. The main reason that this occurs is that while the Tausworthe generator produces PRNs that appear to be i.i.d. Uniform(0,1), the generator is not truly creating i.i.d. Uniform(0,1) outputs because the three equations utilized in the computations create values that inform the calculations of later values ("Generating Uniform Random Numbers"). Thus, the Tausworthe generator built in this project is a good PRN generator but only for certain value combinations for r, q, and l.

Figure 3. Histogram of the Tausworthe generator using bad input values



The third task required plotting adjacent PRNs on the unit square to see if any patterns exist. Through visual inspection of Figure 3, it is apparent that no patterns exist in the unit square. This is expected since the goal of the Tausworthe generator is to create random observations without any patterns.

Figure 3. Plot of adjacent PRNs on the unit square



The final task involved generating Normal(0,1) variables from Uniform(0,1) variables. I completed the transformation utilizing the inverse transform theorem and the Box-Muller method. Figure 4 shows a histogram of the inverse of the CDF. The inverse transform theorem was applied to obtain the normal distribution by getting the inverse of the CDF of the Tausworthe generator output.

Figure 4. Inverse transform theorem

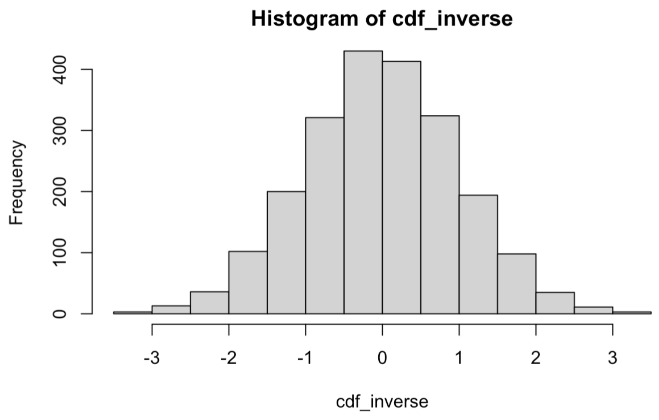
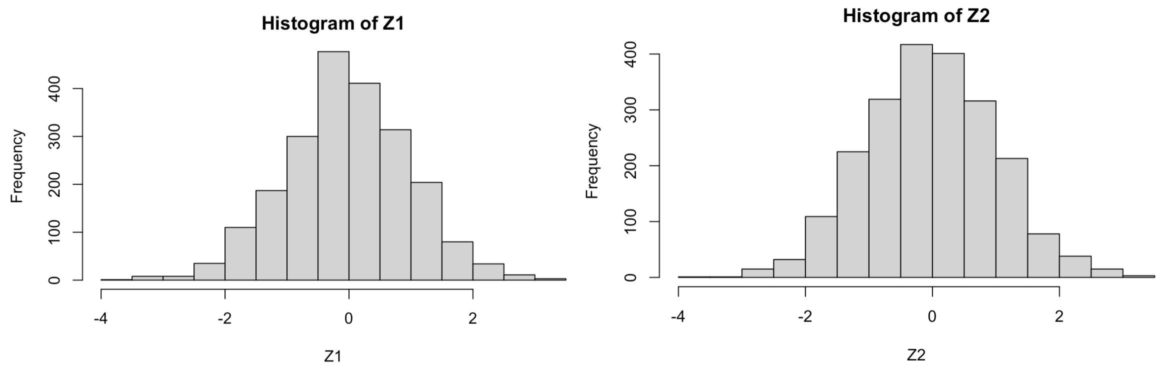


Figure 5 shows the Box-Muller method in action. Two Uniform(0,1) variables were transformed into Normal(0,1) variables utilizing the Box-Muller equation. Both histograms appear to be normally distributed.

Figure 5. Box-Muller method



Two code files have been submitted along with my written report. The first code file is an HTML file that illustrates all the outputs once the code is run. The second code file is an R markdown file that contains all the commented code for the project. The code blocks can be run individually, and the primary tunable parameters include r, q, and l.

Conclusions:

In this project, I learned much about the Tausworthe generator, how it works, and what its limitations are. Building the Tausworthe generator algorithm from scratch required a lot of logic and reasoning to understand how the code needs to flow in order to work properly. I had to add constraints to ensure that the inputs are appropriate and reasoned through every step to check that the logic was correct. By experimenting with various combinations of r, q, and l and performing statistical tests on the generator output, I realized that the generator produces variables that appear to be i.i.d. Uniform(0,1), but there are instances where the generator fails because the PRNs are not truly i.i.d. Uniform(0,1). I plotted adjacent PRNs on a unit square to observe firsthand that the PRNs do, in fact, appear random. Lastly, I learned that the output of the Tausworthe generator can easily be transformed into Normal(0,1) numbers using the inverse transform theorem or the Box-Muller method.

Appendix:

Two supplementary code files for this project have been attached with this report.

References:

“A Guide to dnorm, pnorm, rnorm, and qnorm in R”. *Geeks for Geeks*, 21 Apr. 2022, https://www.geeksforgeeks.org/a-guide-to-dnorm-pnorm-rnorm-and-qnorm-in-r/.

“Base Conversions: Converts Numbers From Binmode, Octmode or Hexmode to Decimal and Vice Versa”. *R Documentation*, n.d., https://www.rdocumentation.org/packages/DescTools/versions/0.99.40/topics/Base%20Conversions.

“Checking the inputs of your R functions”. *R-bloggers*, 9 Mar. 2022, https://www.r-bloggers.com/2022/03/checking-the-inputs-of-your-r-functions/.

“chisq.test: Pearson's Chi-squared Test for Count Data”. *R Documentation*, n.d., https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/chisq.test.

“Concatenate Vector of Character Strings in R (2 Examples) | How to Combine Text Cases”. *Statistics Globe*, n.d., https://statisticsglobe.com/concatenate-vector-of-character-strings-in-r.

“Generating Uniform Random Numbers”. *OMSCS Notes*, n.d., https://www.omscs-notes.com/simulation/generating-uniform-random-numbers/.

“Goodness-of-Fit Test”. *R Documentation*, n.d., https://search.r-project.org/CRAN/refmans/EnvStats/html/gofTest.html.

“How to do modulus operation in R?”. *Project Pro*, 16 Dec. 2022, https://www.projectpro.io/recipes/do-modulus-operation-r.

“ks.test: Kolmogorov-Smirnov Tests”. *R Documentation*, n.d., https://www.rdocumentation.org/packages/dgof/versions/1.4/topics/ks.test.

“ln: Logarithms.”. *R Documentation*, n.d., https://www.rdocumentation.org/packages/SciViews/versions/0.9-13.1/topics/ln.

“Logical Operators”. *R Documentation*, n.d., https://stat.ethz.ch/R-manual/R-devel/library/base/html/Logic.html.

Patel, Chiragkumar. “Uniform Distribution using Chi-Squared Test For Goodness of Fit in R Programming and Hand Calculation”. *Medium*, 5 Jan. 2022, https://medium.com/@chirag1162/uniform-distribution-using-chi-squared-test-for-goodness-of-fit-in-r-and-hand-calculation-6628f6a12e84.

“plotting a circle inside a square in R”. *Stack Overflow*, n.d., https://stackoverflow.com/questions/5411796/plotting-a-circle-inside-a-square-in-r.

“R - Functions”. *Tutorials Point*, n.d. https://www.tutorialspoint.com/r/r_functions.htm.

“R if…else Statement”. *Data Mentor*, n.d., https://www.datamentor.io/r-programming/if-else-statement/.

“R Return Value from Function”. *Data Mentor*, n.d., https://www.datamentor.io/r-programming/return-function/.

“Runs Test for Randomness”. *R Documentation*, n.d., https://search.r-project.org/CRAN/refmans/DescTools/html/RunsTest.html.

“Subsetting R Objects”. *Bookdown*, n.d., https://bookdown.org/rdpeng/rprogdatascience/subsetting-r-objects.html.

“The reverse/inverse of the normal distribution function in R”. *Stack Overflow*, n.d., https://stackoverflow.com/questions/19589191/the-reverse-inverse-of-the-normal-distribution-function-in-r.

Zach. “How to Perform Runs Test in R”. *Statology*, 3 Oct. 2020, https://www.statology.org/runs-test-in-r/.

Zach. “Kolmogorov-Smirnov Test in R (With Examples)”. *Statology*, 20 Oct. 2020, https://www.statology.org/kolmogorov-smirnov-test-r/.