Assignment2: Quality Evaluation of Different Linear Congruential Generators

yifan

Background

We have already learned in class the linear congruential generator (LCG), specified by (a, c, m), where we choose a seed $X_0 \in \{0, 1, ..., m-1\}$ as the seed and generate a sequence of $\{X_1, X_2, ..., X_N\}$ using the recursive formula: $X_i = (aX_{i-1} + c) \bmod m$ (for $i = 1, 2, ..., N$). When selecting the parameters (a, c, m), the textbook suggested three criteria, to assure that the sequence has a period length of m:

(a) m and c are relatively prime;

(b) a − 1 is divisible by every prime factor of m;

(c) if m is a multiple of 4, then a − 1 is a multiple of 4.

Also, since the sequence will be eventually periodic, we must assure that the period is longer than the amount of random numbers we use, so it would be wise to select an m that is as large as possible. Intuitively, the larger the m, the more "random" the sequence would seem to be (if we select a and c cautiously). We want to evaluate the quality (randomness) of the sequence generated by different LCGs, one that we would assume a better quality, and one that we would not. The randomness can be evaluated in the following two ways:

(1) Hypothesis testing of whether $\{X_1, X_2, ..., X_N\}$ can be taken as i.i.d. samples from $U\{0, 1, ..., m-1\}$;

(2) Visualizing the dependence between pairs of consecutive samples.

Preparation

For hypothesis testing, we want to use K-S test, as well as Chi-Squared test.

For K-S test, the test statistic is defined as $sup_{x \in \mathcal{R}} |F_n(x) - F_0(x)|$, where $F_0(x)$ is the distribution function of the distribution that we assume the data follows, and $F_n(x)$ is the empirical distribution function of the sample. When both distributions are discrete, the value is simply the maximum absolute difference between the two distribution functions at n points:

```
ks_test_statistic <- function(observed_values, expected_values) {
  # Calculate the cumulative distribution function (CDF) for the observed values
  observed_cdf <- cumsum(observed_values) / sum(observed_values)

  # Calculate the cumulative distribution function (CDF) for the expected values
  expected_cdf <- cumsum(expected_values) / sum(expected_values)

  # Calculate the maximum difference between the two CDFs (K-S test statistic)
  ks_statistic <- max(abs(observed_cdf - expected_cdf))

  return(ks_statistic)
}
```

For Chi-Squared test, the test statistic is defined as $\sum_{j=0}^{m-1}(O_j - E_j)^2/E_j$, where O_j stands for the observed numbers of samples in the jth block, and E_j stands for the expected numbers of samples in the jth block:

```r
chi_square_statistic <- function(observed_values, expected_values) {
  # Calculate the Chi-square statistic
  chi_square_stat <- sum((observed_values - expected_values)^2 / expected_values)

  return(chi_square_stat)
}
```

Hypothesis testing of the sequences generated by the LCGs

First, we choose $m = 64$, which is really small, so we wouldn't expect this LCG to perform so well. Still, we want to ensure that the period length of the generated sequence is m, so we set $a = 9$ and $c = 5$, which satisfies the aforementioned 3 criteria. Since the sequence length should be shorter than 64 (the period length), we choose $N = 60$. Furthermore, we separate $0, 1, ..., 63$ into 16 bars, each with a length of 4, and we set X_0 to 1:

```r
# Set parameters
m <- 64
a <- 9
c <- 5
X_0 <- 1
num_values <- 60

# Initialize the vector to store the random numbers
random_numbers_1 <- numeric(num_values)

# Set the first value
random_numbers_1[1] <- X_0

# Generate the random numbers using the LCG formula
for (i in 2:num_values) {
  random_numbers_1[i] <- (a * random_numbers_1[i-1] + c) %% m
}

# Create an empty vector to count the frequencies in each bin (16 bins)
bin_counts_1 <- numeric(16)

# Count the numbers in each bin
for (num in random_numbers_1) {
  bin_index <- floor(num / 4) + 1  # Divide by 4 to get the bin index (0-3, 4-7, ..., 60-63)
  bin_counts_1[bin_index] <- bin_counts_1[bin_index] + 1
}
```

Then, we perform hypothesis testing using the functions we wrote.

The null hypothesis is: $H_0 : X_1, ..., X_{60} \text{i.i.d} \sim U(0, 1, ..., 63)$

For K-S test:

```r
expected_values <- rep(1/16 * 60, 16)

ks_test_statistic(bin_counts_1, expected_values)
```

```
## [1] 0.03333333
```

As suggested by the textbook, we want to perform a two-sided test (we choose p=0.05), which is very hard for K-S test, since this type of test is usually one-sided, thus the lower bound of the acceptance region cannot be obtained through table lookup. Plus, the distribution of the statistic is complicated, when H_0 is true. Therefore, we obtain the rejection region by random simulation:

```r
# Set the random seed to ensure reproducibility
set.seed(0)

# Number of simulations and sample size
num_simulations <- 10000
sample_size <- 50

# Distribution used for simulation, assuming the sample comes from a standard normal distribution
distribution <- rnorm

ks_statistics <- numeric(num_simulations)

# Perform the random simulation
for (i in 1:num_simulations) {
  sample <- distribution(sample_size)
  ks_test <- ks.test(sample, "pnorm", mean = 0, sd = 1)
  ks_statistics[i] <- ks_test$statistic
}

# Obtain the 0.025 quantile
ks_0025th_percentile <- quantile(ks_statistics, 0.025)

print(ks_0025th_percentile)
```

```
##       2.5%
## 0.06500464
```

Through table lookup, we can obtain that the upper bound of the acceptance region is approximately 0.21. Thus, the rejection region is approximately $R_{rej} = \{F < 0.064\} \cup \{F > 0.21\}$, here F represents the value of the test statistic.

Since $F_1 = 0.033$ falls into the rejection region, we reject $H_0 : X_1, ..., X_{60}$i.i.d$\sim U(0, 1, ..., 63)$. Intuitively, the generated sequence is too regular, which is the reason why we rejected H_0.

Moreover, we can perform hypothesis testing using Chi-Square test:

```r
chi_square_statistic(bin_counts_1, expected_values)
```

```
## [1] 2.4
```

When H_0 is true and N is large, the test statistic Q follows χ^2 distribution with 15 degrees of freedom. Through table lookup, we can obtain the rejection region: $ R_{rej} = {Q < 6.26} {Q>27.49}$. Since Q_1 falls into that region, we reject $H_0 : X_1, ..., X_{60}$i.i.d$\sim U(0, 1, ..., 63)$. Since two hypothesis testing methods both reject H_0, we can conclude that the randomness of the generated sequence is quite poor.

To compare with this LCG with $m = 64$, we want to establish an LCG that would perform better, with a larger m. Therefore, we choose $(a, c, m) = (314159269, 453806245, 2^{31})$, suggested by Kobayashi. We generate 1000 random numbers in this way, and we choose $X_0 = 1$. We still separate the random numbers into 16 columns:

```r
# Set parameters
m <- 2^{31}
a <- 314159269
```

```r
c <- 453806245
X_0 <- 1
num_values <- 1000

# Initialize the vector to store the random numbers
random_numbers_2 <- numeric(num_values)

# Set the first value
random_numbers_2[1] <- X_0

# Generate the random numbers using the LCG formula
for (i in 2:num_values) {
  random_numbers_2[i] <- (a * random_numbers_2[i-1] + c) %% m
}

# Create an empty vector to count the frequencies in each bin (16 bins)
bin_counts_2 <- numeric(16)

# Count the numbers in each bin
for (num in random_numbers_2) {
  bin_index <- floor(num / 134217728) + 1  # Divide by 4 to get the bin index (0-3, 4-7, ..., 60-63)
  bin_counts_2[bin_index] <- bin_counts_2[bin_index] + 1
}
```

Then we perform hypothesis testing. The null hypothesis is: $H_0 : X_1, ..., X_{100}$i.i.d$\sim U(0, 1, ..., 2^{31} - 1)$.

First, we use K-S test:

```r
expected_values <- rep(1/16 * 1000, 16)

ks_test_statistic(bin_counts_2, expected_values)
```

```
## [1] 0.021
```

Then, similarly we obtain the bounds of the acceptance region by random simulation (since $N = 1000$ is too large for table lookup):

```r
# Set the random seed to ensure reproducibility
set.seed(0)

# Number of simulations and sample size
num_simulations <- 10000
sample_size <- 1000

# Distribution used for simulation, assuming the sample comes from a standard normal distribution
distribution <- rnorm

ks_statistics <- numeric(num_simulations)

# Perform the random simulation
for (i in 1:num_simulations) {
  sample <- distribution(sample_size)
  ks_test <- ks.test(sample, "pnorm", mean = 0, sd = 1)
  ks_statistics[i] <- ks_test$statistic
}
```

```r
# Obtain the 0.025 quantile
ks_0025th_percentile <- quantile(ks_statistics, 0.025)
ks_0975th_percentile <- quantile(ks_statistics, 0.975)

print(ks_0025th_percentile)
```

```
##       2.5%
## 0.01494952
```

```r
print(ks_0975th_percentile)
```

```
##      97.5%
## 0.04640426
```

Therefore, the rejection region is $R_{rej} = \{F < 0.015\} \cup \{F < 0.046\}$. Since $F_{Kob} = 0.021$ does not fall into R_{rej}, we accept H_0 to be true.

Furthermore, we can perform Chi-Square test:

```r
chi_square_statistic(bin_counts_2, expected_values)
```

```
## [1] 7.648
```

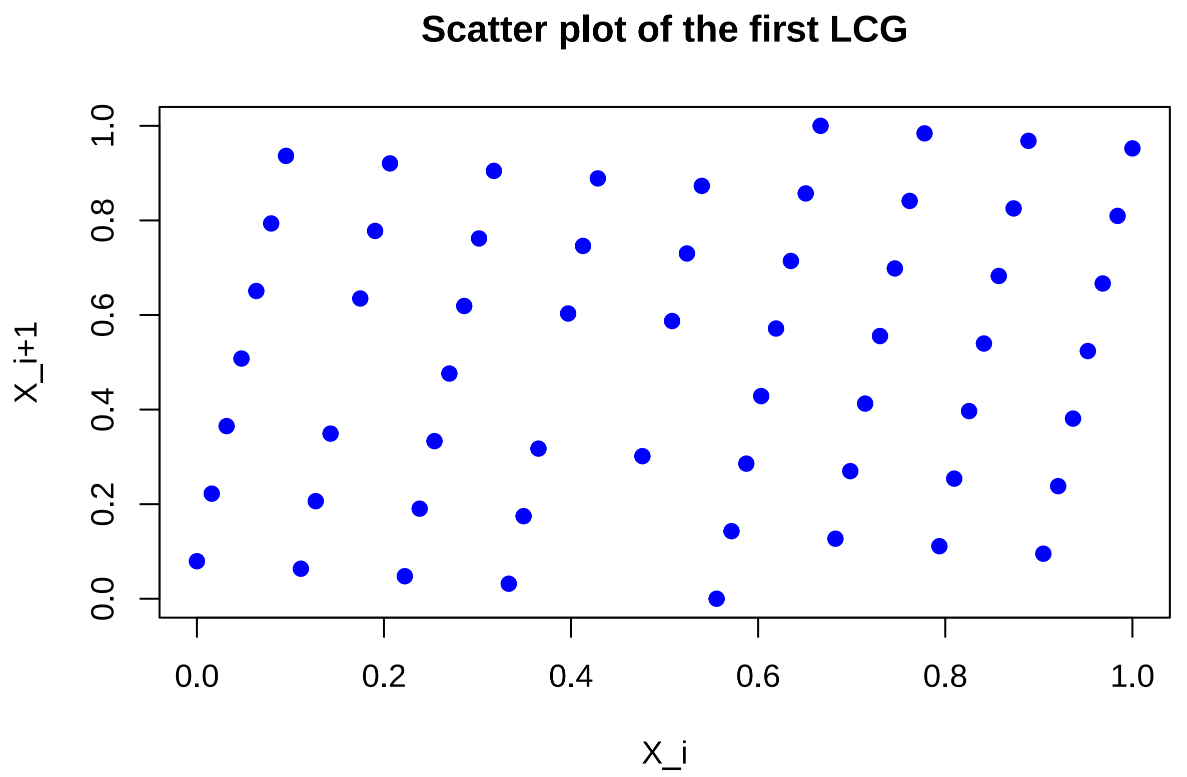
When H_0 is true and N is large, the test statistic Q still follows χ^2 distribution with 15 degrees of freedom, and the rejection region remains the same: $ R_{rej} = {Q < 6.26} {Q>27.49}$. Since $Q_{Kob} = 7.648$ does not fall into that region, H_0 is accepted. Since both hypothesis testing methods accept H_0, we can conclude that the generated sequence is indeed random enough.

Visualization of the generated sequences

We want to evaluate the independence of the sequences generated by the LCGs visually. More specifically, we want to access the dependence between pairs of consecutive samples (X_{i-1}, X_i) for $i = 2, 3, ...N - 1$ through a scatter plot.

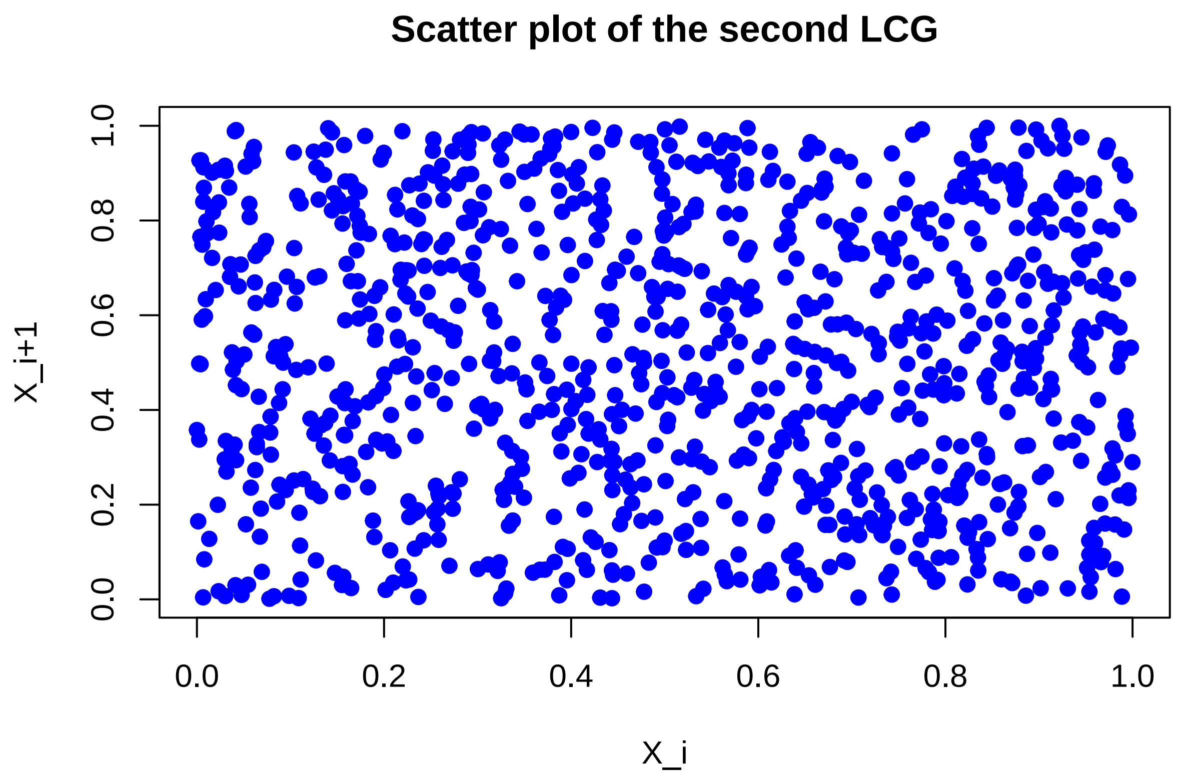
First, we plot the scatter plot of the pairs of consecutive samples generated by the first LCG (compressed into $[0, 1]$:

```r
x <- random_numbers_1[1:(length(random_numbers_1)-1)] / 63
y <- random_numbers_1[2:length(random_numbers_1)] / 63

plot(x, y, main="Scatter plot of the first LCG", xlab="X_i", ylab="X_i+1", pch=19, col="blue")
```



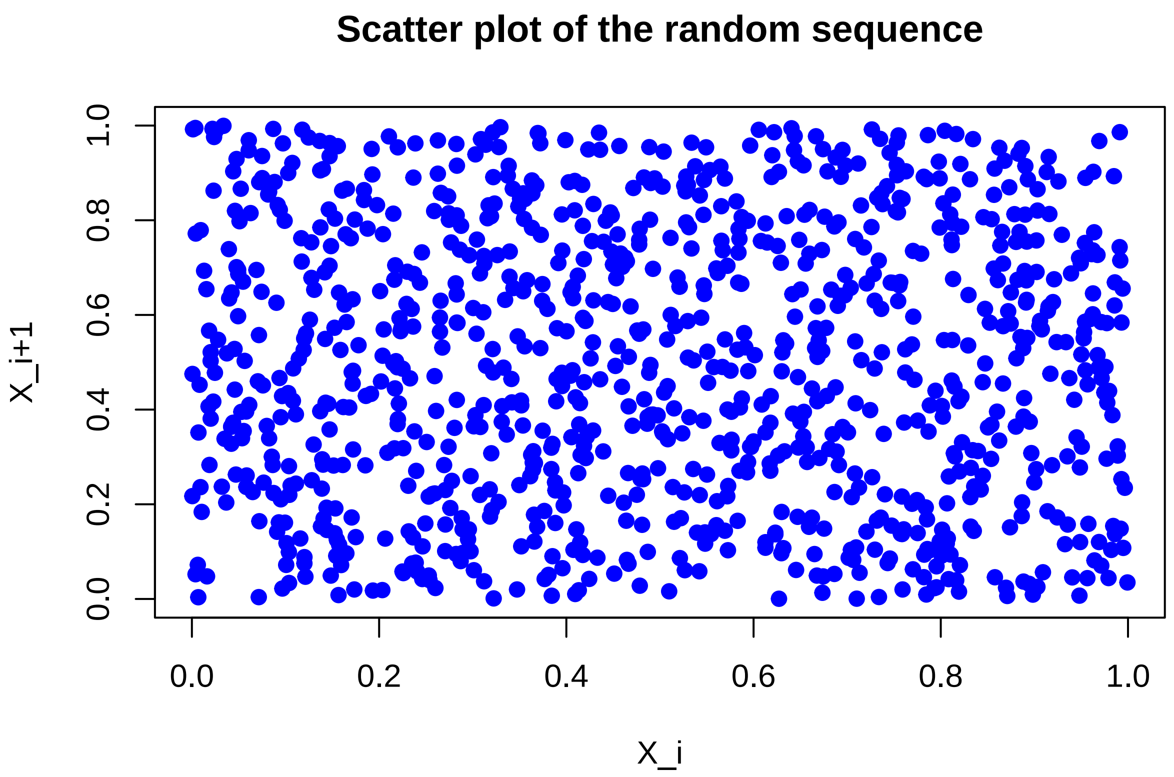
Apparently, the points in this scatter plot are all lined up, showing great dependence between the pairs of consecutive samples. To contrast this scatter plot, we plot the scatter plot of the pairs of consecutive samples generated by the second LCG (compressed into $[0, 1]$):

```r
x <- random_numbers_2[1:(length(random_numbers_2)-1)] / (2^31 - 1)
y <- random_numbers_2[2:length(random_numbers_2)] / (2^31 - 1)

plot(x, y, main="Scatter plot of the second LCG", xlab="X_i", ylab="X_i+1", pch=19, col="blue")
```



Finally, we generate a sequence of 1000 i.i.d. samples that follows the $U(0, 1)$ distribution, and plot the scatter plot similarly, to compare with our LCGs.

```
#Set the random seed to ensure reproducibility
set.seed(123)

random_numbers <- runif(1000, min = 0, max = 1)

x <- random_numbers[1:(length(random_numbers_2)-1)]
y <- random_numbers[2:length(random_numbers_2)]

plot(x, y, main="Scatter plot of the random sequence", xlab="X_i", ylab="X_i+1", pch=19, col="blue")
```



We can see that the scatter plot is very similar to the scatter plot of the second LCG, which means that the pairs of consecutive samples generated by our second LCG seem to be independent (though they are actually not).