Assignment 3

Ariz Kazani

2024-07-28

Assignment 3

Name: Ariz Kazani

Student ID: 101311311

Notes

```
# Libraries

# NOTE: if you do not have any of the below libraries installed,
# un-comment the line and run it

# install.packages("rmarkdown")
library(rmarkdown)

# install.packages("ggplot2")
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.1
```

```
# useful libraries not required for this assignment
# install.packages("patchwork")
# library(patchwork)

# install.packages("dplyr")
# library(dplyr)

# install.packages("tidyr")
# library(tidyr)

# install.packages("gapminder")
# library(gapminder)
```

Solutions

1.

A. Simulate rolling a fair twelve-sided die 500 times in R.

```
simulateDiceRole <- function(n, seed) {
  set.seed(seed)
  outcome <- sample(1:12, n, replace = TRUE)
  return(outcome)
}

outcome <- simulateDiceRole(500, 123)

outcome
```

```
##   [1]  3  3 10  2  6 11  5  4  6  9 10 11  5  3 11  9 12  9  9  3  8 10  7 10  9
##  [26]  3  4  1 11  7  5 12 10  7  9  9 10  7 11 12  5  7  5 11  6  9  2  5  8 12
##  [51]  2  1  9 11  9  6  5  9 10 12  4  6 11  8  6  6  7  1  6  2  1  2  4  5  6
##  [76]  3  9  4  6  9  9  7  3  8 12  9  3  7  3  7  6 10  5  5  8  3 10  2 10 12
## [101]  2 10  6 12  4  1  6  3  8  3  8  1  7 11  7  7 10  6  7 11 10  5  6  8  5
## [126]  7 11  4 12 12  3  9  7  6 10  9  7  2  3  8  4  7  4  1  8  4  9  8  6 11
## [151]  4  8  3  4  4 12  6  1 10 11  4 11  9  7  8  5  2 11  6  9  8 12 10  4  5
## [176]  7  1  8  8 10  9  8  2  5  9  7  7 10 12 10  8  6  7 10 11  1  9  3 10  5
## [201]  6 11 11  9  4 10  7  9  7 10  4  8  9  9  9  5  7  6 11  1 10 10  1 10  1
## [226] 10 11  5  7 12  5 10  9  4  6  2  1  5  9  4  3  9  1  2  4 10  1  5  5  9
## [251]  8  7  9  5  2 10  6  7  9  1  5 12  5  8  5  7  4  2  1  1  2  1  2  8  1
## [276]  3  2  2  5 12  9 10  6 10 10  6 12  3  4 11 11  3  7  3  2  7  3  9  7  9
## [301]  4  2  6 10 12  9 12 11  7  5 11  1  3 12  2  5  1 12  3  9  2  6  3  3  9
## [326]  2  1  6 10  8 10 10  4  6  4  4  9  8  7 10  1  2  8  3 10  7 11  1 12  2
## [351]  3  3  6 10  5  4  2  1 12  1 12  5 12  2  2 12  3  5  6 10  3 11  3  1  8
## [376]  1 11  4  8 12 10 10  8 12  7  2  1  4 11  9  5 11 12  8  2  5  2  6 10 10
## [401]  9 11  7  6 12  3 11  8 11  7  4  8 12  3  8  6  4  7 10 11  8  4 10  9  6
## [426]  3  4  9  9  4  4  6  5  1  8  7 12  5  6  3  8 10  5  7  4 12  9 11 11  3
## [451] 10  7  2  9  6  4  6  2  2  8  8 11  5  9  7  3  7  8 11 10  5  1  7  6  8
## [476]  4  6 12  1  3 12  4  1  5 11  4  2  7  3 10  3 11  1  7  6 12  4  5  7  3
```

B. Calculate the empirical probability of rolling a prime number or a multiple of 3 based on the simulation results. Is this consistent with what you would expect?

```
allOutcome <- table(outcome)
allOutcome
```

```
## outcome
##  1  2  3  4  5  6  7  8  9 10 11 12
## 36 35 42 41 41 43 48 39 49 51 39 36
```

```r
POM3 <- c(2, 3, 5, 6, 7, 9, 11, 12)

sumPrimeORm3 <- sum(allOutcome[POM3])
sumPrimeORm3
```

```
## [1] 333
```

```r
EP <- sumPrimeORm3 / 500

cat("Expected Probability", 8 / 12, "\n")
```

```
## Expected Probability 0.6666667
```

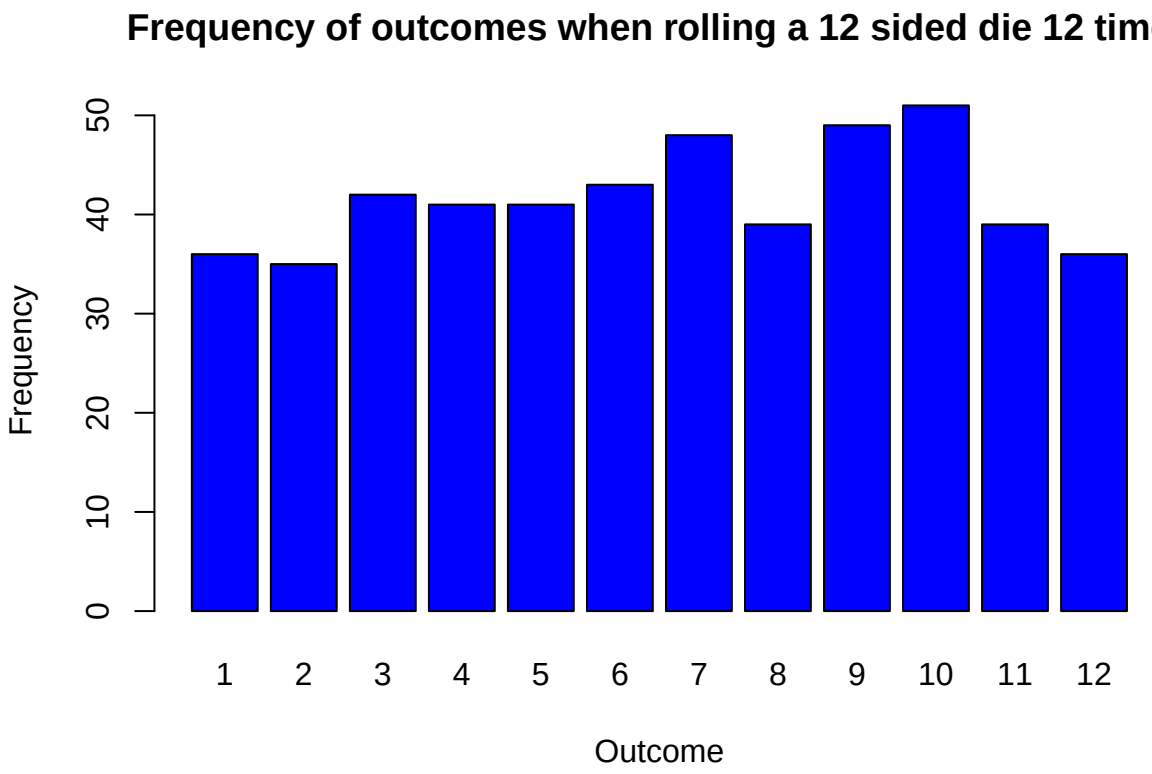
```r
cat("Simulated Probability", EP, "\n")
```

```
## Simulated Probability 0.666
```

When I initially ran the simulation, the probability I ended up with was 0.67 which is very very close to the expected probability of 0.666

C. Plot a bar chart showing the frequencies of each outcome (1 through 12) obtained from the simulation.

```r
barplot(
  allOutcome,
  main = "Frequency of outcomes when rolling a 12 sided die 12 times",
  xlab = "Outcome",
  ylab = "Frequency",
  col = "blue",
)
```



2.

A. Write R code to calculate the probability of getting at least 4 heads in 8 coin tosses, assuming p = 0.6.

```
pminFourHead <- 1 - pbinom(3, size = 8, prob = 0.6)
pminFourHead
```

```
## [1] 0.8263296
```

B. Compute the cumulative probability of getting 4 or more heads.

```
# aren't a and b asking the same thing?
# this question did not specify the p value
#so I'm going to assume it's 0.5 otherwise its the same as q1

pminFourHead <- 1 - pbinom(3, size = 8, prob = 0.5)
pminFourHead
```

```
## [1] 0.6367187
```

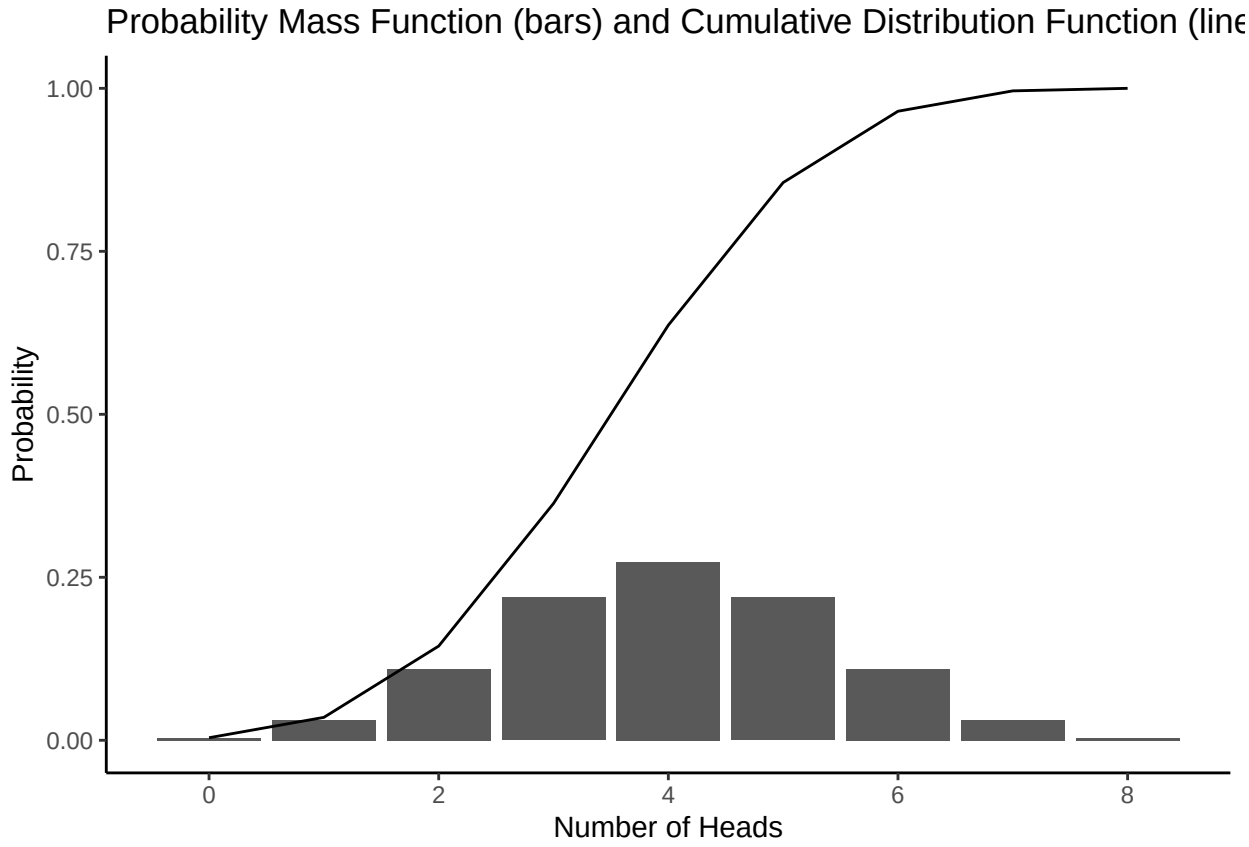
C. Plot the probability function (PMF) and cumulative distribution function (CDF) on the same graph.

```r
index <- 0:8

pmf <- dbinom(index, size = 8, prob = 0.5)
cdf <- pbinom(index, size = 8, prob = 0.5)

df <- data.frame(index, pmf, cdf)

ggplot(df, aes(x = index, y = pmf)) +
  geom_bar(stat = "identity") +
  geom_line(aes(y = cdf)) +
  labs(
    title = "Probability Mass Function (bars) and Cumulative Distribution Function (line)",
    x = "Number of Heads",
    y = "Probability") +
  theme_classic()
```



3.

A. Use R to compute the probability of observing exactly 3 events in a given time interval, assuming lambda = 2.5 events per hour.

```
POE3E <- dpois(3, 2.5)
POE3E
```

```
## [1] 0.213763
```

B. Compute the cumulative probability of observing 3 or fewer events.

```
POE3EOF <- ppois(3, 2.5)
POE3EOF
```

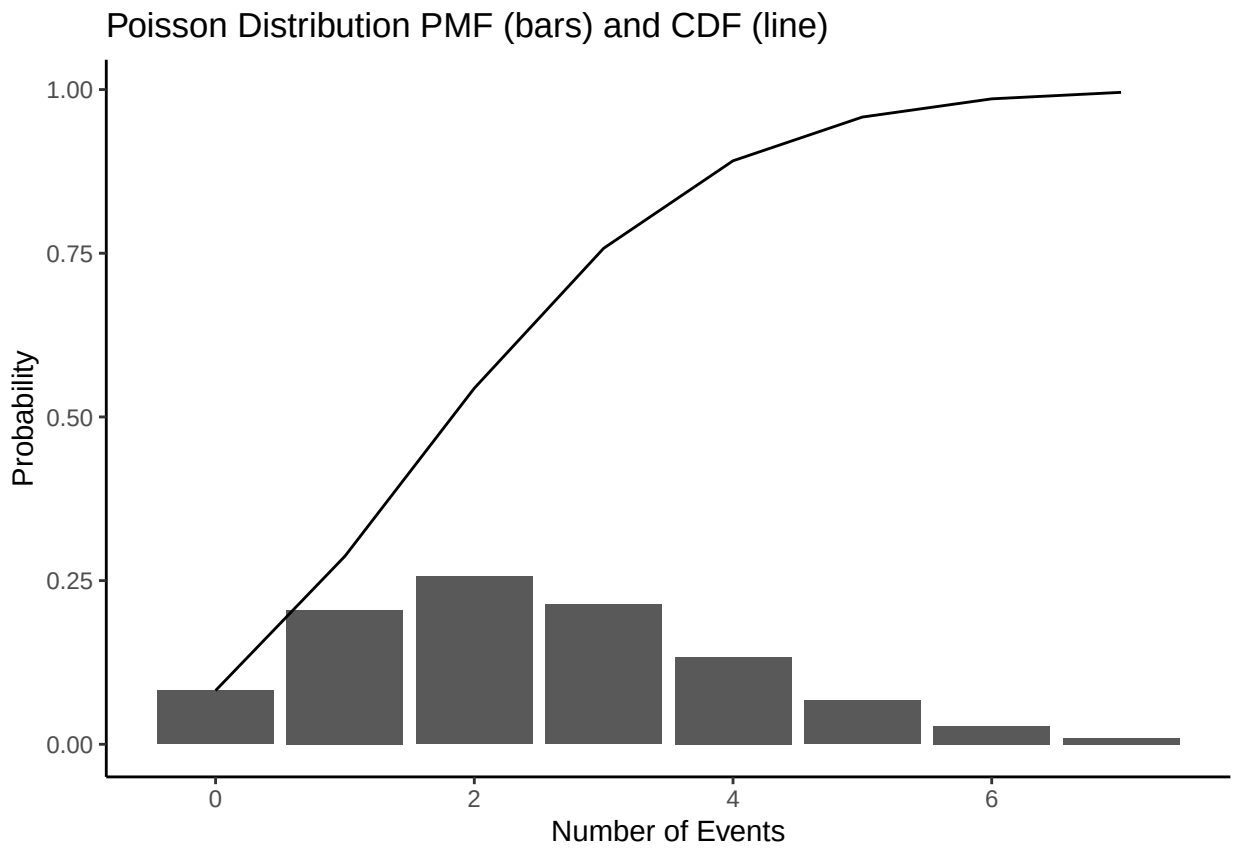
```
## [1] 0.7575761
```

C. Plot the probability function (PMF) and cumulative distribution function (CDF) on the same graph.

```
events <- 0:qpois(0.99, 2.5)

pmf <- dpois(events, 2.5)
cdf <- ppois(events, 2.5)
data <- data.frame(events, pmf, cdf)

ggplot(data, aes(x = events, y = pmf)) +
  geom_bar(stat = "identity") +
  geom_line(aes(y = cdf)) +
  labs(
    title = "Poisson Distribution PMF (bars) and CDF (line)",
    x = "Number of Events",
    y = "Probability") +
  theme_classic()
```



4.

Write a function to implement the Linear Congruence Method for pseudo random numbers (PRN). The function should take a length n, the parameters m, a, and c as well as the seed X0 as input and should return a vector X = (X1, X2, . . . , Xn). Test your function by calling it with the parameters m = 8, a = 5, c = 1 and by comparing the output with the results from class.

```
lc <- function(n, seed, m, a, c) {
  arr <- numeric(n + 1)

  arr[1] <- seed
  for (i in 2:(n + 1)) {
    arr[i] <- (m * arr[i - 1] + a) %% c
  }
  return(arr[-1])
}
print("output based on the description and a random seed: ")
```

```
## [1] "output based on the description and a random seed: "
```

```r
lc(10, 7, 8, 5, 1)
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

```r
print("Using m = 1, a = 7 and c = 10 and a seed of 7 (based on lecture video #14")
```

```
## [1] "Using m = 1, a = 7 and c = 10 and a seed of 7 (based on lecture video #14"
```

```r
print("Expected output (based on lecture 14) 4, 1, 8, 5, 2, 9, 6, 3, 0, 7")
```

```
## [1] "Expected output (based on lecture 14) 4, 1, 8, 5, 2, 9, 6, 3, 0, 7"
```

```r
print("my my function")
```

```
## [1] "my my function"
```

```r
lc(10, 7, 1, 7, 10)
```

```
##  [1] 4 1 8 5 2 9 6 3 0 7
```

This question was kind of ambiguous in the sense that the letters were different form the one in lecture 14 so I assumed m, a and c were mapped to a, b, c respectively. For the first part we can see that we are modding by 1, so the answer will always be 0. When trying the example from the lecture we can see that it is correctly implemented and returns the expected output.

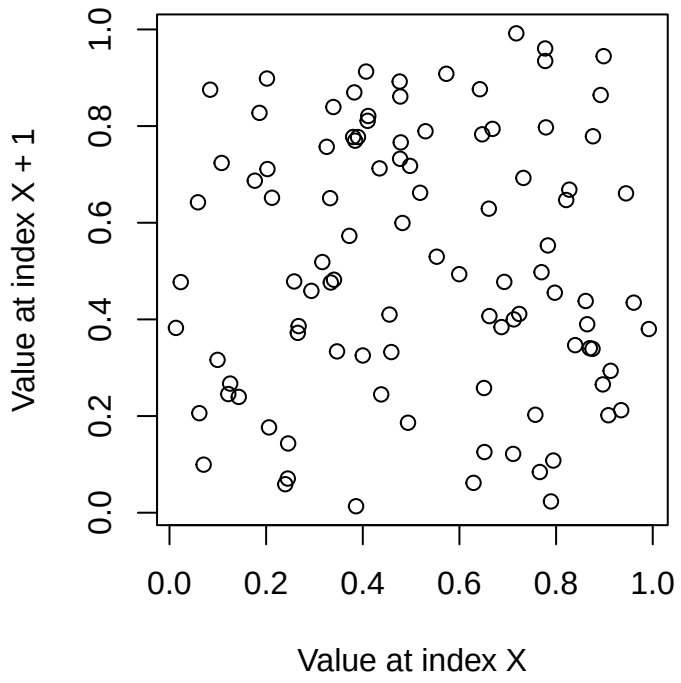
5.

Given a sequence X1, X2, ... , Xn of U[0,1]- distributed PRN, we can use a scatter plot of (Xi ,Xi+1) for i = `1 , ... , n -` in order to try to assess whether the Xi are independent.

A. Create such a plot using the built in random number generator in R. Can you explain the resulting plot.

```r
set.seed(0)
arr <- runif(100)

par(pty = "s")
plot(arr[-100],
     arr[-1],
     main = "Scatter plot of X and X + 1 when using R's random function",
     xlab = "Value at index X",
     ylab = "Value at index X + 1")
```

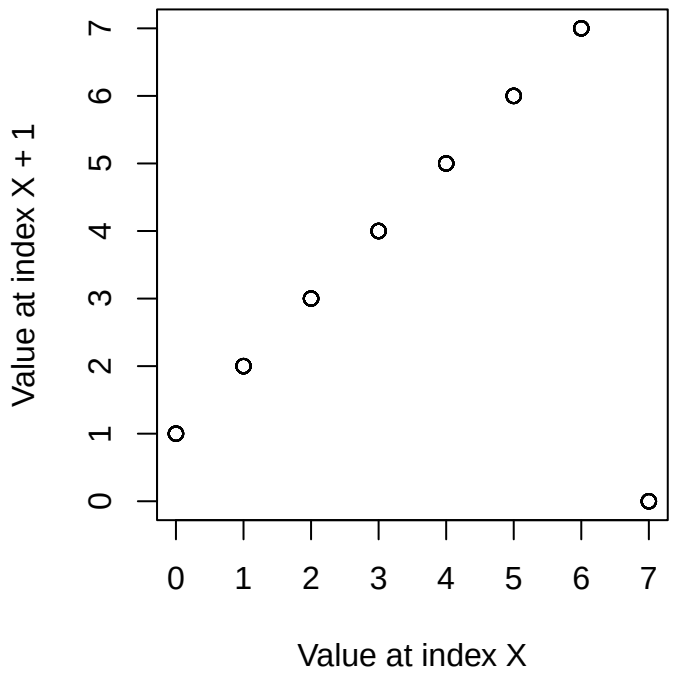


I cannot explain the contents of the plot as they look completely random to me and I see no correlation or pattern.

B. Create a similar plot using your function from exercise 1 with m = 81, a = 1, c = 8 and seed 0. Discuss the resulting plot.

```
arr <- lc(100, 0, 81, 1, 8)

par(pty = "s")
plot(arr[-100],
     arr[-1],
     main = "Scatter plot of X and X + 1 when using a custom random function",
     xlab = "Value at index X",
     ylab = "Value at index X + 1")
```

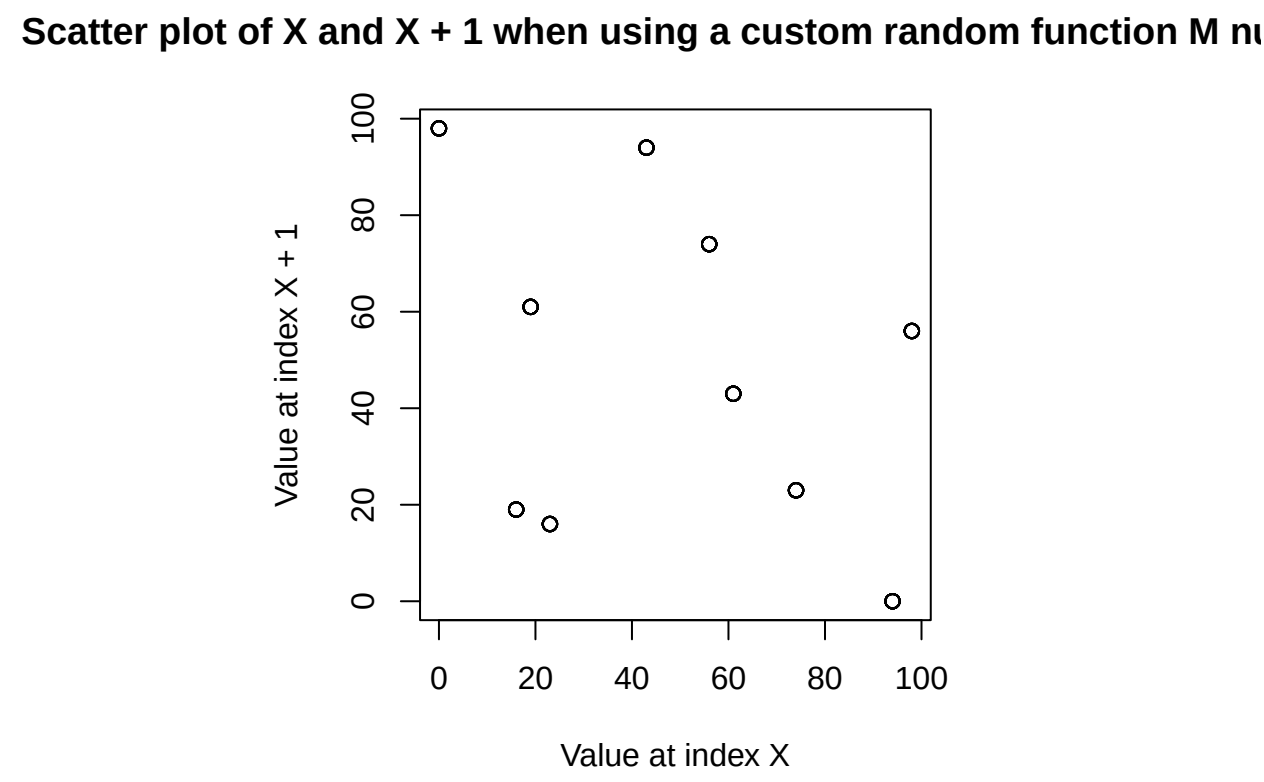


We can see that the length of the output is 100 but the graph only shows 8 points, this shows us that the parameters m = 81, a = 1, c = 8, are in fact bad for creating random numbers as the numbers generated are cyclical.

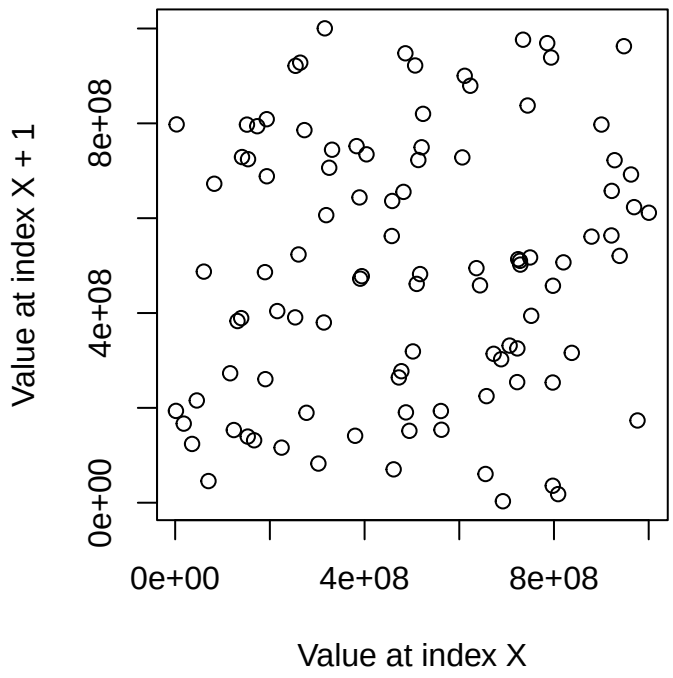
C. Repeat the experiment from (b) using the parameters m = 1024, a = 401, c = 101 and m = 2^32, a = 1664525, c = 1013904223. Discuss the results.

```
arr1 <- lc(100, 0, 1024, 401, 101)
arr2 <- lc(100, 0, 2 ** 32, 1664525, 1013904223)

par(pty = "s")
plot(arr1[-100],
     arr1[-1],
     main = "Scatter plot of X and X + 1 when using a custom random function M nums",
     xlab = "Value at index X",
     ylab = "Value at index X + 1")
```




```
plot(arr2[-100],
     arr2[-1],
     main = "Scatter plot of X and X + 1 when using a custom random function L nums",
     xlab = "Value at index X",
     ylab = "Value at index X + 1")
```



When we used the medium numbers we got a few more data points, and when we used the large numbers we got points comparable (in terms of perceived randomness) to R's built in function. This shows us that with the linear congruent method of generating PRM, the greater the input numbers are, the more randomness we can produce. This held true for all of the random seeds I tested too.

6.

Simulate a Drunkard's walk in two dimensions. The Drunkard's walk is a random process where at each step, the walker randomly chooses one of eight possible directions (North, South, East, West, Northeast, Southeast, Southwest, Northwest) with varying probabilities. Your task is to implement the simulation according to the following specifications:

A. Define a vector named probabilities representing the probabilities of moving in each direction:

- North: 10%
- South: 15%
- East: 15%
- West: 20%
- Northeast: 10%
- Southeast: 10%
- Southwest: 10%
- Northwest: 10%

```r
probabilities <- c(0.1, 0.15, 0.15, 0.2, 0.1, 0.1, 0.1, 0.1)
probabilities
```

```
## [1] 0.10 0.15 0.15 0.20 0.10 0.10 0.10 0.10
```

B. Write a function named simulate_step that simulates one step of the drunkard's walk:

The function should randomly select one of the eight directions based on the specified probabilities. It should return a vector representing the step in the chosen direction, where the first element indicates the change in the x-coordinate (East/West) and the second element indicates the change in the y-coordinate (North/South).

```r
simulate_step <- function(probabilities) {
  directions <- list(c(0, 1),
                     c(0, -1),
                     c(1, 0),
                     c(-1, 0),
                     c(1, 1),
                     c(1, -1),
                     c(-1, -1),
                     c(-1, 1))

  probI <- sample(1:8, size = 1, prob = probabilities)
  return(directions[[probI]])
}

set.seed(0)
testStep <- simulate_step(probabilities)
testStep
```

```
## [1] -1  1
```

C. Simulate the drunkard's walk for a total of 1000 steps. Initialize a matrix named positions with dimensions (1001, 2) to store the positions of the drunkard at each step. Use a for loop to iteratively call the simulate_step function and update the drunkard's position accordingly.

```r
set.seed(0)
positions <- matrix(0, nrow = 1001, ncol = 2)

for (i in 1:1000) {
  positions[i + 1, ] <- positions[i, ] + simulate_step(probabilities)
}

head(positions)
```

```
##      [,1] [,2]
## [1,]    0    0
## [2,]   -1    1
## [3,]    0    1
```

```
## [4,]    0    0
## [5,]    1   -1
## [6,]    1    0
```

D. Plot the path of the drunkard's walk in two dimensions. Use the plot function to create a line plot of the positions. Mark the starting point in red and the ending point in blue.

```
plot(
  positions[, 1],
  positions[, 2],
  type = "l",
  col = "black",
  main = "Drunkard's Walk",
  xlab = "X",
  ylab = "Y"
)
points(positions[1, 1], positions[1, 2], col = "red", pch = 10)
points(positions[1001, 1], positions[1001, 2], col = "blue", pch = 10)
```