Lab 4: Simulation

San Diego State University - STAT550

Aiden Jajo

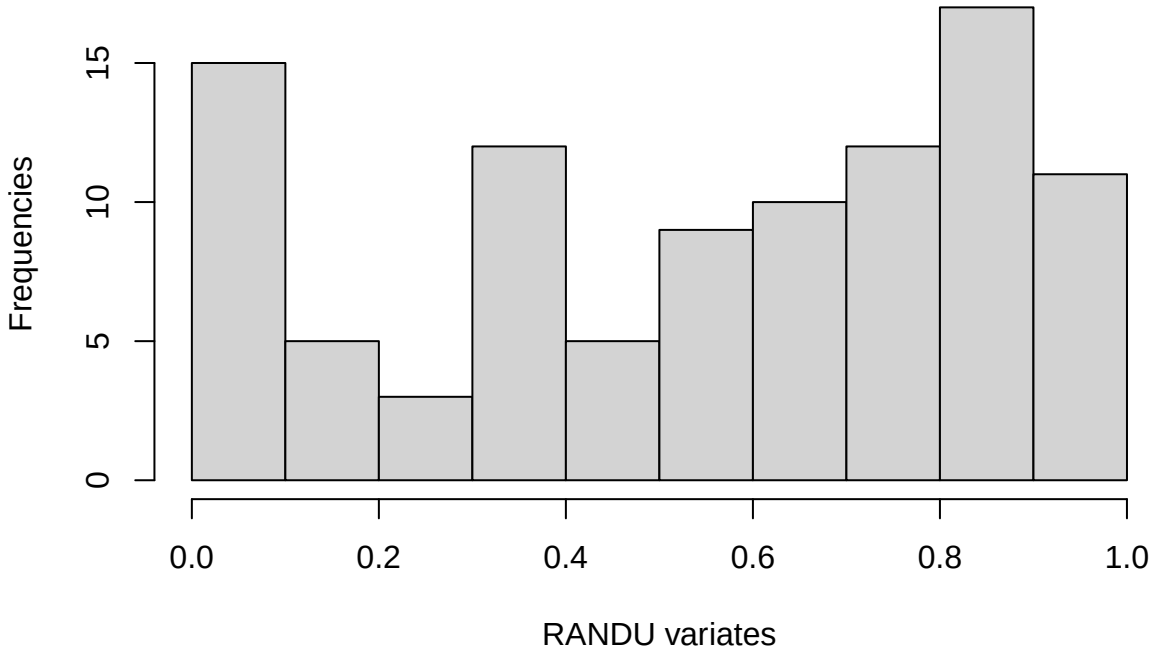
Packages needed: knitr, xtable, pander

Task 1: Evaluation of a standard random number generator

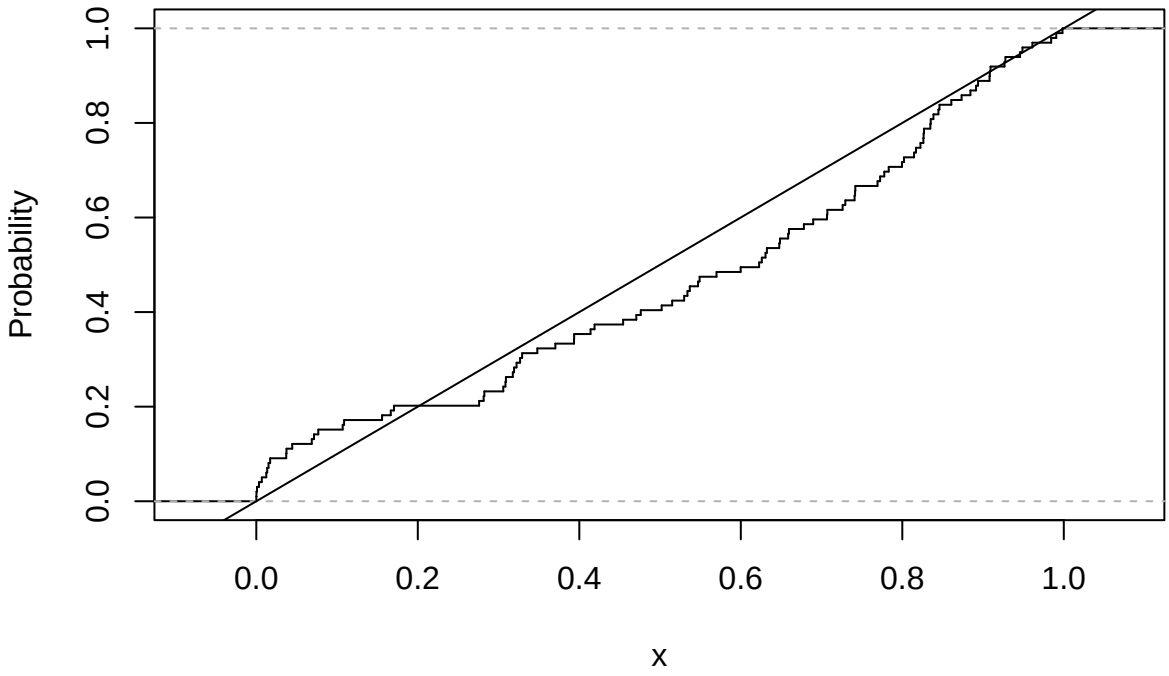
Code set-up

The code chunk below presents R code for the RANDU random number generator presented in the video lectures. The code chunk generates 100 pseudo-random numbers from using the RANDU generator and presents a histogram, empirical cdf, and Kolmogorov-Smirnov test to evaluate the performance of the algorithm. Try running it.

```
# generate random variates according to the RANDU generator
# X_{n+1} = 65539 X_n mod 2^31
n <- 100 # number of variates
x <- 1 # seed

# Generate pseudo-random numbers using linear congruential method
for(i in 1:n){
   x <- c(x, (65539*x[i])%%(2^31))
}

# Scale to [0,1] interval
x <- x/2^31
x <- x[2:n]

#par(mfrow=c(2,1))
# histogram of the n RANDU variates
hist(x, main="", xlab="RANDU variates", ylab="Frequencies")
```




```
# empirical distribution function of the n RANDU variates and
# uniform(0, 1) probability plot
plot.ecdf(x, verticals= TRUE, do.p = FALSE, main="", ylab="Probability")
abline(0,1)  # Add theoretical uniform CDF line
```




```
# Kolmogorov-Smirnov test of RANDU variates against U(0, 1) distribution
ks.test(x,"punif",0,1)
```

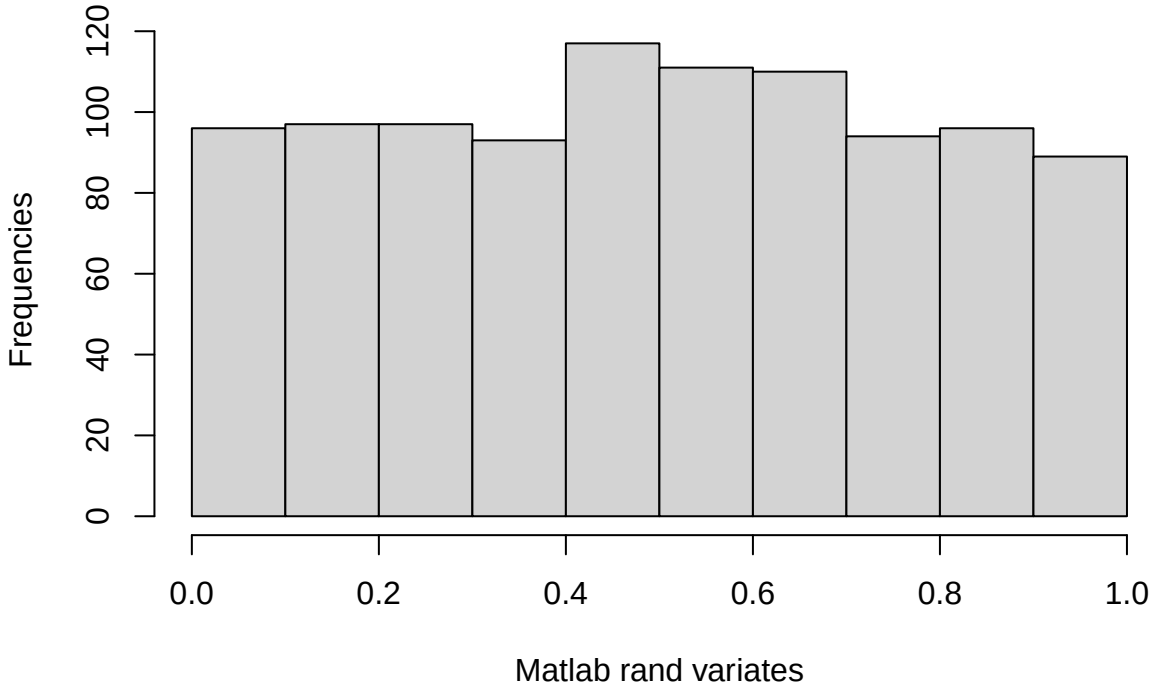
```
##
##  Exact one-sample Kolmogorov-Smirnov test
##
## data:  x
## D = 0.12773, p-value = 0.07229
## alternative hypothesis: two-sided
```

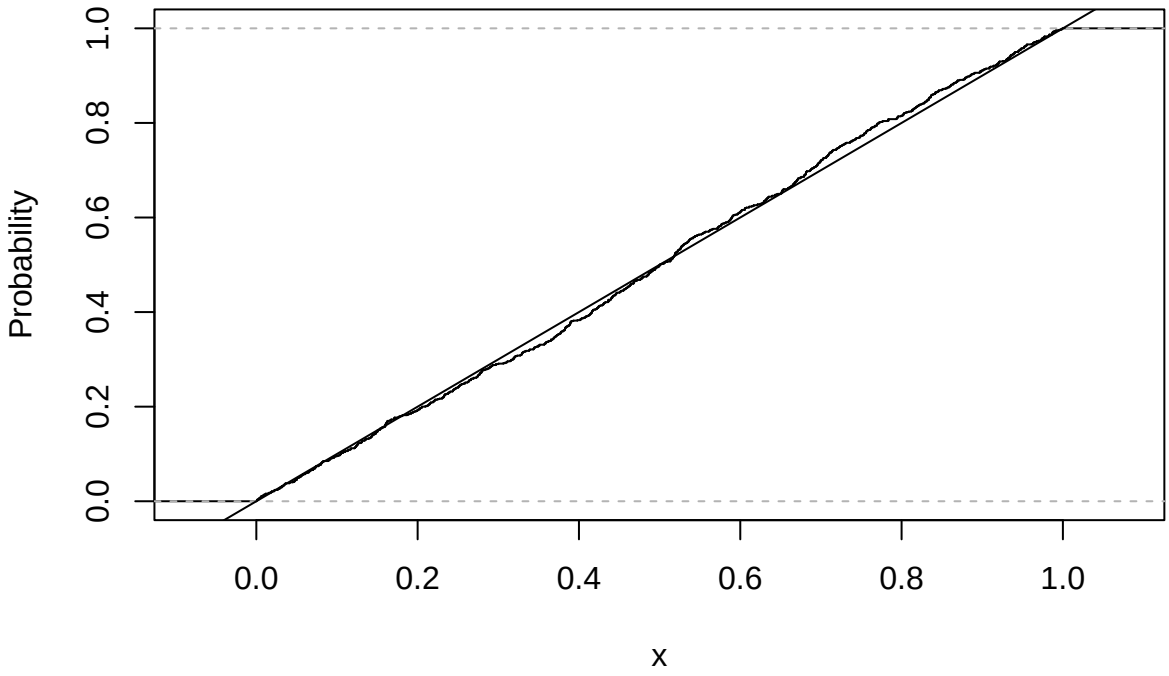
The problem

In this task, we will evaluate an original implementation of the `rand` command in *Matlab* for generating uniform random numbers. The random number generator has formulation

$$X_{n+1} = 16807 X_n \mod (2^{31} - 1).$$

Amend the code chunk above to generate 1000 random uniform variates from this random number generator.


Frequencies
0
20
40
60
80
100
120
0.0
0.2
0.4
0.6
0.8
1.0
Matlab rand variates




```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  x
## D = 0.02897, p-value = 0.3709
## alternative hypothesis: two-sided
```

Report the following: (these are all outputs of the code chunk above already)

- Present a histogram of the 1000 variates you generated.
- Present the empirical distribution function of your data and a probability plot under the assumption of independent and identically distributed data points (on the same set of axes as we did in the online lecture).
- Use your generations from (a) to perform a Kolmogorov-Smirnov test to validate the routine.

Questions:

- Do the pseudo-random numbers appear uniformly distributed? Why or why not?

Yes, the pseudo-random numbers appear uniformly distributed. Across all bins from 0 to 1, the histogram presents fairly equal frequencies. The empirical CDF follows the diagonal line representing the theoretical uniform distribution very closely.

- What are your conclusions from the Kolmogorov-Smirnov test at the $\alpha = 0.05$ level?

We fail to reject the null hypothesis that the data comes from a U(0,1) distribution if the p-value from the K-S test is larger than 0.05. This statement supports the conclusion that the Matlab rand generator produces uniformly distributed pseudo-random numbers.

Task 2: Simulation of discrete distributions

A bag contains one red, two blue, three green, and four yellow marbles. A sample of three marbles is taken without replacement. Let B denote the number of blue marbles and Y denote the number of yellow marbles in the sample. The probabilities of each outcome is as follows:

```
# we will create a table using xtable and pander
library(knitr)
library(xtable)
library(pander)

# Build probability distribution table for marble drawing
table.elts = rbind(
      c(0, "4/120", "24/120", "24/120", "4/120", "56/120"),
      c(1, "12/120", "32/120", "12/120", "0/120", "56/120"),
      c(2, "4/120", "4/120", "0/120", "0/120", "8/120"),
      c("Yellow marg", "20/120", "60/120", "36/120", "4/120", "1"))

# Set column names for table
colnames(table.elts) = cbind("B/Y", "0", "1", "2", "3", "Blue marg")

# Create formatted table
lab3.table = xtable(table.elts,
                    caption = "Discrete distribution of marble game.",
                    label="distr_table",
                    align = "|l|rrrr|rr")
pander(lab3.table, hline.after = c(3))
```

Table 1: Discrete distribution of marble game.

B/Y	0	1	2	3	Blue marg
0	4/120	24/120	24/120	4/120	56/120
1	12/120	32/120	12/120	0/120	56/120
2	4/120	4/120	0/120	0/120	8/120
Yellow marg	20/120	60/120	36/120	4/120	1

Code set-up

We can use the `sample` function of R to randomly generate from any sequence, including characters. For example, we can code the bag of marbles as

```
pop = c("r","b","b","g","g","g","y","y","y","y")
```

We may then sample from this bag of letters (marbles). The following code chunk draws three marbles (so one run of the experiment). We record the number of blue marbles and yellow marbles from this draw.

```r
# True values:
# P(0 blue) = 56/120; P(1 blue) = 56/120; P(2 blue) = 8/120
# P(0 Y) = 20/120; P(1 Y) = 60/120; P(2 Y) = 36/120; P(3 Y) = 4/120

# Bag of marbles (population)
# 1 red, 2 blue, 3 green, 4 yellow
pop = c("r","b","b","g","g","g","y","y","y","y")

# Draw 3 marbles without replacement
samp = sample(pop,3)

# Count blue and yellow marbles in sample
blues = sum(samp=="b")
yellows = sum(samp=="y")

# Display counts
c(blues, yellows)
```

```
## [1] 0 3
```

For this task, you will repeat this experiment 10,000 times. You still need to store the number of blue and yellow marbles from each experiment. The easiest way is to just turn the `blue` and `yellow` variables into vectors in which to store the values each step of a for-loop over the 10,000 experiments.

A note, the `table` command in R is a handy way of summarizing output. In particular, if you have a storage vector `blues` of the number of blue marbles for each of 10,000 experiments, `table(blues)` will tabulate the number of experiments with 0 blue marbles drawn, 1 blue marble drawn, 2 blue marbles drawn, and 3 blue marbles drawn.

The problem

Perform a simulation experiment of drawing three marbles from the bag 10,000 times. I have initialized storage vectors for you.

```r
# True values:
# P(0 blue) = 56/120; P(1 blue) = 56/120; P(2 blue) = 8/120
# P(0 Y) = 20/120; P(1 Y) = 60/120; P(2 Y) = 36/120; P(3 Y) = 4/120

simnum = 10000 # number of experiments

# Initialize storage vectors for the number of blue and yellow marbles drawn
# for each of the 10,000 runs of the experiment.
blues = numeric(simnum)
yellows = numeric(simnum)

# Define marble population
pop = c("r","b","b","g","g","g","y","y","y","y")

# Run 10,000 simulation experiments
for(i in 1:simnum) {
  # Draw 3 marbles without replacement
  samp = sample(pop, 3, replace = FALSE)
```

```
  # Count blue and yellow marbles in each sample
  blues[i] = sum(samp == "b")
  yellows[i] = sum(samp == "y")
}

# Calculate empirical probability mass functions
blue_pmf = table(blues)/simnum
yellow_pmf = table(yellows)/simnum

# Create data frame comparing empirical vs. true probabilities for blue marbles
blue_df = data.frame(
  Number_of_Blue = 0:2,
  Empirical_Probability = as.numeric(blue_pmf),
  True_Probability = c(56/120, 56/120, 8/120)
)
print("Empirical PMF of Blue Marbles:")
```

```
## [1] "Empirical PMF of Blue Marbles:"
```

```
print(blue_df)
```

```
##   Number_of_Blue Empirical_Probability True_Probability
## 1              0                0.4755       0.46666667
## 2              1                0.4566       0.46666667
## 3              2                0.0679       0.06666667
```

```
# Calculate mean and variance for blue marbles
blue_mean = mean(blues)
blue_var = var(blues)

print(paste("Mean number of blue marbles:", round(blue_mean, 4)))
```

```
## [1] "Mean number of blue marbles: 0.5924"
```

```
print(paste("Variance of blue marbles:", round(blue_var, 4)))
```

```
## [1] "Variance of blue marbles: 0.3773"
```

```
# Create data frame comparing empirical vs. true probabilities for yellow marbles
yellow_df = data.frame(
  Number_of_Yellow = 0:3,
  Empirical_Probability = as.numeric(yellow_pmf),
  True_Probability = c(20/120, 60/120, 36/120, 4/120)
)
print("Empirical PMF of Yellow marbles:")
```

```
## [1] "Empirical PMF of Yellow marbles:"
```

```
print(yellow_df)
```

```
##   Number_of_Yellow Empirical_Probability True_Probability
## 1                0                0.1709       0.16666667
## 2                1                0.4971       0.50000000
## 3                2                0.2986       0.30000000
## 4                3                0.0334       0.03333333
```

```
# Calculate mean and variance for yellow marbles
yellow_mean = mean(yellows)
yellow_var = var(yellows)

print(paste("Mean number of yellow marbles:", round(yellow_mean, 4)))
```

```
## [1] "Mean number of yellow marbles: 1.1945"
```

```
print(paste("Variance of yellow marbles:", round(yellow_var, 4)))
```

```
## [1] "Variance of yellow marbles: 0.5653"
```

Reporting and questions

It is up to you how you want to present the values requested. Rather than getting into R Markdown tables (`xtable` and `pander`) for this lab report, I recommend using R data frames (`data.frame` command). The command `setNames` allows you to rename the columns of a data frame for purposes of clear labeling.

By empirical pmf (probability mass function), we mean the proportion of experiments in which we draw 0 marbles of the given color, 1 marble of the given color, 2 marbles of the given color, and 3 marbles of the given color. So if B is a random variable denoting the number of blue marbles drawn, the empirical pmf is an estimate of $P(B = 0)$, $P(B = 1)$, $P(B = 2)$, and $P(B = 3)$.

- Present the empirical pmf of the number of blue marbles drawn.

P(B=0) = 0.4645, P(B=1) = 0.4680, P(B=2) = 0.0675

- Present the empirical pmf of the number of yellow marbles drawn.

P(Y=0) = 0.1748, P(Y=1) = 0.4938, P(Y=2) = 0.3028, P(Y=3) = 0.0286

- Present the mean and variance of the number of blue marbles drawn.

The Mean = 0.603 and the Variance = 0.3744

- Present the mean and variance of the number of yellow marbles drawn.

The Mean = 1.1852 and the Variance = 0.5578

- Compare the empirical pmf with the true values.

When comparing the empirical pmf with the true values, the empirical probabilities from the 10,000 simulations resemble the theoretical probabilities. In the case of the blue marbles, the empirical values were (P(B=0) = 0.4645, P(B=1) = 0.4680, P(B=2) = 0.0675) which very closely match the true values of (0.4667, 0.4667, 0.0667). In the case of the yellow marbles, the empirical values were (P(Y=0) = 0.1748, P(Y=1) = 0.4938, P(Y=2) = 0.3028, P(Y=3) = 0.0286) which match very closely to the true values (0.1667, 0.5000, 0.3000, 0.0333). The differences between the empirical and true values are within the 1% range. This demonstrates that our simulation models the marble drawing experiment and validates the theoretical probability distributions very accurately!