Homework 3

Zachary Lazerick

14 February 2023

1) From Simulation by Sheldon Ross, Chapter 4. [Question 5 in the book] Another method of generating a random permutation, different from the one presented in Example 4b, is to successively generate a random permutation of the elements $1, 2, \ldots, n$ starting with n = 1, then n = 2, and so on. (Of course, the random permutation when n = 1 is 1.) Once one has a random permutation of the first $n \breve{} 1$ elements – call it $P_1, P_2, \ldots, P_{n-1}$ – the random permutation of the n elements $1, \ldots, n$ is obtained by putting n in the final position – to obtain the permutation $P_1, P_2, \ldots, P_{n-1}$, n – and then interchanging the element in position n (namely, n) with the element in a randomly chosen position which is equally likely to be either position 1, position 2, …, or position n.

a) Write an algorithm that accomplishes the above. Let n = 10 and display 5 of your random permutations.

```
RandomPermutations <- function(n) {
  X <- sample(x = c(1:(n-1)), size = n - 1, replace = F); X <- append(X, n)
  U <- runif(1); temp <- X[floor(n*U + 1)];
  X[floor(n*U + 1)] <- X[n]; X[n] <- temp
  X
}

RandomPermutations(10)
```

```
## [1] 6 2 9 5 8 3 4 10 1 7
```

```
RandomPermutations(10)
```

```
## [1] 7 8 5 1 6 10 4 9 3 2
```

```
RandomPermutations(10)
```

```
## [1] 9 4 1 2 7 6 5 10 8 3
```

```
RandomPermutations(10)
```

```
## [1] 5 7 2 6 3 4 8 9 10 1
```

```
RandomPermutations(10)
```

```
## [1] 8 4 7 1 2 3 9 5 10 6
```

b) [Question #4 in the book] A deck of 100 cards – numbered $1, 2, \ldots, 100$ – is shuffled and then turned over one card at a time (use your function from part (a) for this). Say that a "hit" occurs whenever card i is the ith card to be turned over, $i = 1, 2, \ldots, 100$. Write a simulation program to estimate the expectation (i.e. mean) and variance of the total number of hits.

```r
total_hits <- rep(0, 1000)
for (i in 1:1000) {
  counter <- 0;  X <- RandomPermutations(100)
  for (j in 1:length(X)) {
    if (j == X[j]) {
      counter <- counter + 1
    }
  }
  total_hits[i] <- counter
}

## Mean
mean(total_hits)
```

```
## [1] 1.037
```

```r
## Variance
var(total_hits)
```

```
## [1] 0.9986296
```

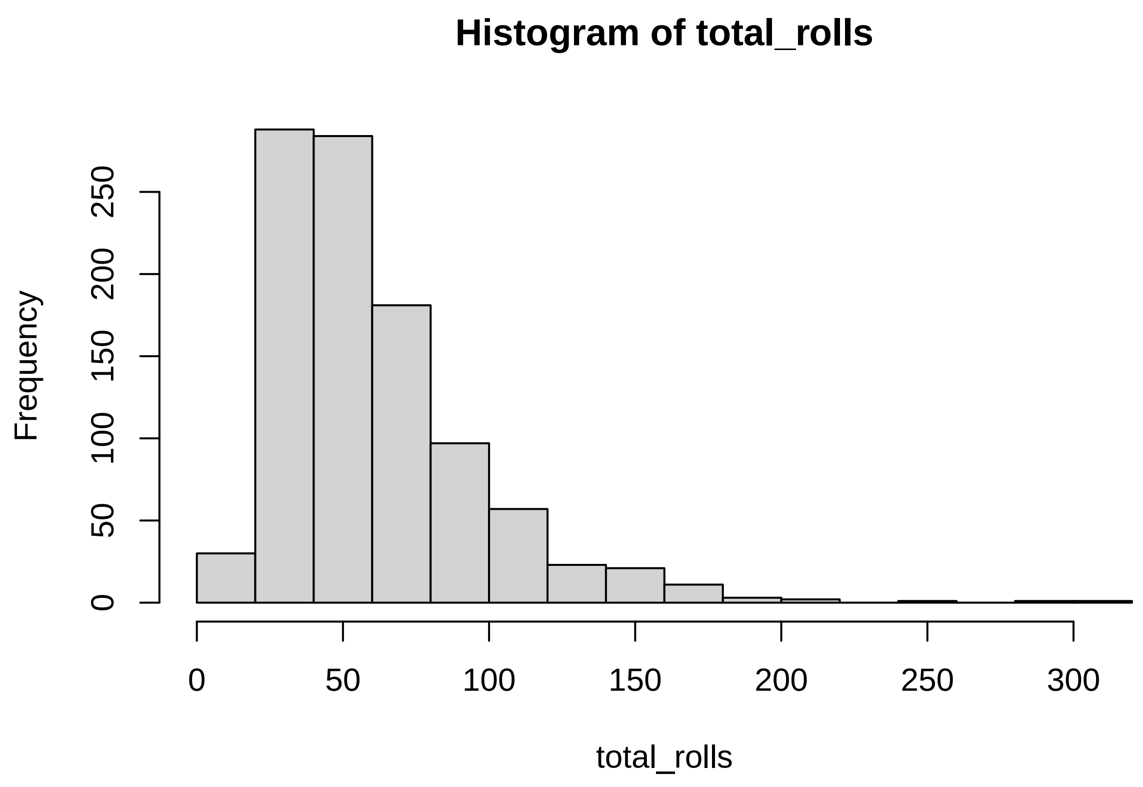
2) [Question 7 in book] A pair of fair dice are to be continually rolled until all the possible outcomes $2, 3, \ldots, 12$ have occurred at least once. Develop a simulation study to estimate the expected number of dice rolls that are needed. Print out the mean number of rolls needed and make a histogram of your results (the histogram is a visual representation of the sampling distribution of the random variable).

```
total_rolls <- rep(0, 1000)
for (i in 1:1000) {
  counter <- 0; Table_Roll_Sums <- rep(0, 11);
  while (0 %in% Table_Roll_Sums) {
    Dice <- sample(x = c(1, 2, 3, 4, 5, 6), size = 2, replace = T)
    Table_Roll_Sums[sum(Dice) - 1] <- Table_Roll_Sums[sum(Dice) - 1] + 1
    counter <- counter + 1
  }
  total_rolls[i] <- counter
}

## Mean
mean(total_rolls)
```

```
## [1] 61.294
```

```
## Histogram of Results
hist(total_rolls)
```

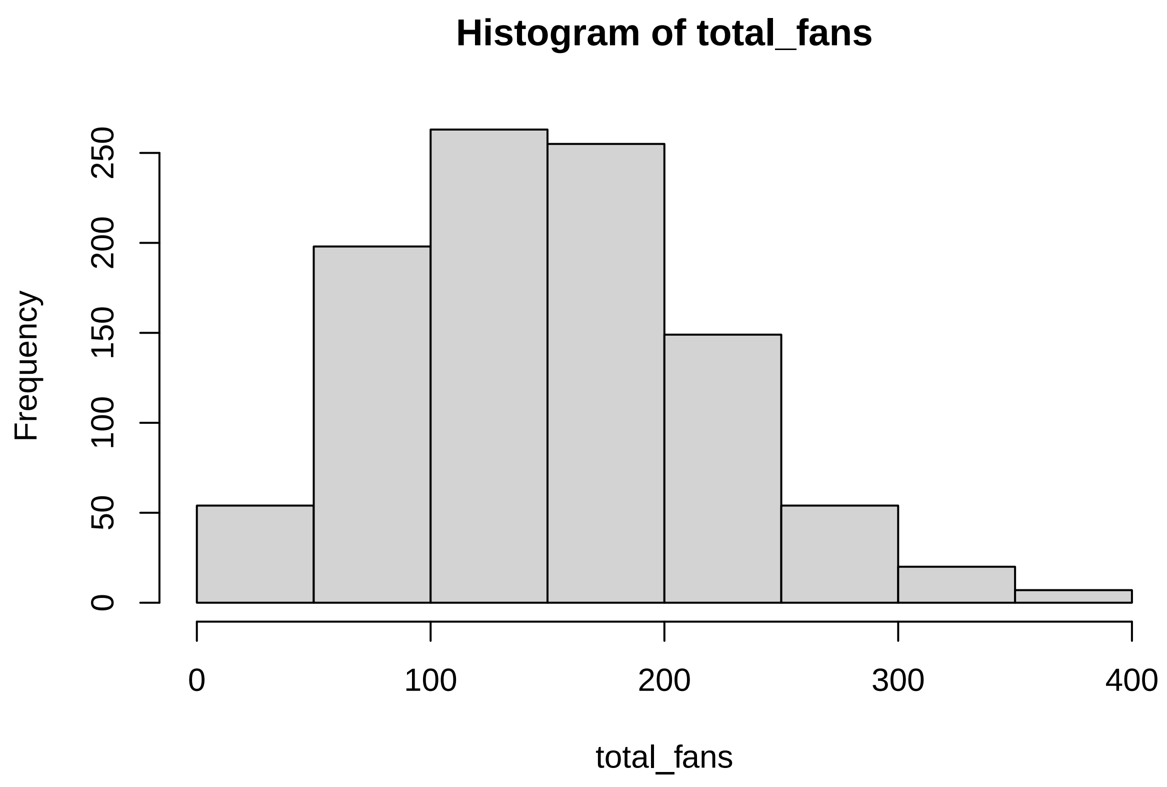


3) From Simulation by Sheldon Ross, Chapter 5. [Question 29 in textbook] Buses arrive at a sporting event according to a Poisson process with a rate of 5 per hour. Each bus is equally likely to contain either $20, 21, \ldots, 40$ fans, with the numbers in the different buses being independent. Write an algorithm to simulate the arrival of fans to the event by time t $= 1$.

a) Simulate 1000 values for the total number of fans that arrive by time t $= 1$ and make a histogram of this distribution.

```r
## Buses 5 per hour --> \lambda = 5 --> \beta = 1/5
total_fans <- rep(0, 1000)
for (i in 1:1000) {
  t = 0; counter <- 0;
  while (t < 1) {
    U <- runif(1); X <- (-1/5)*log(1-U)
    t <- t + X; counter <- counter + 1
  }
  fans <- sample(x = 20:40, size = (counter - 1), replace = T)
  total_fans[i] <- sum(fans)
}

## Histogram
hist(total_fans)
```



b) What are the mean and variance for the total number of fans that arrive? Note: The true mean is 150 fans.

```
## Mean
mean(total_fans)
```

```
## [1] 151.643
```

```
## Variance
var(total_fans)
```

```
## [1] 4854.36
```

c) What is P(X = 150 fans arrive)?

```
counter <- 0
for (i in 1:length(total_fans)) {
  if (total_fans[i] == 150) {
    counter <- counter + 1
  }
}

## Report P(toal_fans = 150)
counter/length(total_fans)
```

```
## [1] 0.005
```

4) Simulating from a discrete distribution. Write a program that uses the built-in uniform random number generator to generate 7500 independent realizations from each of the following distributions:

a) Geometric, with parameter $p = 0.83$.

```r
GeometricGenerator <- function(n, p) {
  total_until_success <- rep(0, n)
  for (i in 1:n) {
    U <- runif(1); X <- ceiling(log(1-U)/log(1- p))
    total_until_success[i] <- X
  }
  cat("The Mean is ", mean(total_until_success), "and the Variance is",
      var(total_until_success))
}
```

b) Binomial, with parameters $n = 200, p = 0.11$.

```r
BinomialGenerator <- function(m, n, p) {
  total_success <- rep(0, m)
  for (i in 1:m) {
    U <- runif(n); X <- rep(0, length(U))
    for (j in 1:length(U)) {
      if (U[j] < p) {
        X[j] = 1
      }
    }
    total_success[i] <- sum(X)
  }
  cat("The Mean is ", mean(total_success), "and the Variance is",
      var(total_success))
}
```

c) Poisson, with parameter $\lambda = 2.1$. (Simulate a Poisson by generating $U = Unif(0, 1)$ and setting the exponential random variable $X = \frac{-1}{\lambda} * \ln(U)$. We'll talk about why this works next week)

```r
PoissonGenerator <- function(n, l) {
  total_Poisson <- rep(0, n)
  for (i in 1:n) {
    t = 0; counter <- 0;
    while (t < 1) {
      U <- runif(1); X <- (-1/l)*log(1-U)
      t <- t + X; counter <- counter + 1
    }
    total_Poisson[i] <- counter - 1
  }
  cat("The Mean is ", mean(total_Poisson), "and the Variance is",
      var(total_Poisson))
}
```

d) Find the mean and variance of each set of 7500 realizations, and compare with the corresponding theoretical mean and variance.

```
## Geometric Mean and Variance

##Theoretical Quantities
## Mean = (1-p)/p --> (.17)/(.83) = .2048, ## Variance = (1-p)/p^2 --> (.17)/(.83)^2 = .2468

set.seed(0719)
GeometricGenerator(7500, .83) ## 7500 Realizations
```

```
## The Mean is  1.207333 and the Variance is 0.2441121
```

```
## Binomial Mean and Variance

## Theoretical Quantities
## Mean = np --> 200*.11 = 22, ## Variance = npq --> 200*.11*.89 = 19.58

set.seed(1923)
BinomialGenerator(7500, 200, .11) ## 7500 Realizations
```

```
## The Mean is  22.02 and the Variance is 19.84545
```

```
## Poisson Mean and Variance

## Theoretical Quantities
## Mean = \lambda --> 2.1, Variance = \lambda --> 2.1

set.seed(724)
PoissonGenerator(7500, 2.1) ## 7500 Realizations
```

```
## The Mean is  2.093467 and the Variance is 2.049804
```

e) Finally, report the theoretical probabilities that the integer k is observed in combination with your simulated probability of observing k for $k = \{0, 1, 2, 3\}$. (Notes for part (e): (i) As you did in HMWK #2, you can use the table() command to tabulate the simulated probability of observing each outcome. Theoretical probabilities can be obtained using the dgeom(), dbinom(), and dpois() functions or doing the calculations out by hand. See the help files for the syntax of these functions. (ii) It's not easy to create a table in R markdown, do the best you can displaying the essential information. Try creating a matrix or data frame and then printing to the screen.

Probability P(K = k) (Simulated/Theoretical)	P(k = 0)	P(k = 1)	P(k = 2)	P(k = 3)
Geometric (p = .83)	0/0	0.8269/0.83	0.1437/0.1411	0.0249/0.0240
Binomial (n = 200, p = .11)	$0/(7.55 * 10^{-11})$	$0/(1.86 * 10^{-9})$	$0/(2.29 * 10^{-8})$	$0/(1.87 * 10^{-7})$
Poisson ($\lambda = 2.1$)	0.1191/0.1225	0.2623/0.2572	0.2685/0.2700	0.1899/0.1890

5) Rolling Dice. Two friends independently roll a die until they obtain a 6. Use the geometric random variable simulator written in the previous question to calculate the probability that the two friends stop on the same roll. In other words, simulate a large number (10,000) experiments and count the number of times the two friends roll their die an equal number of times.

```r
## Rewritten Geometric Distribution Generator
GeometricGenerator <- function(p) {
    U <- runif(1); X <- ceiling(log(1-U)/log(1- p))
}

## Dice Role
RollingDice <- function(n) {
  counter <- 0
  for (i in 1:n) {
    X <- GeometricGenerator(1/6); Y <- GeometricGenerator(1/6)
    if (X == Y) {
      counter <- counter + 1
    }
  }
  counter
}

RollingDice(10000)
```

```
## [1] 894
```

6) Chance Variation? Two pollsters each decide to call 20 people to answer their particular survey question. The first pollster calls 20 women and of the 20, 15 answer the question ‘yes’. The second pollster calls 20 men and receives only 9 ‘yes’ answers. If the overall proportion of men and women who would answer this question ‘yes’ is the same, what is the probability of observing a difference of 6 or larger by chance alone? This is an example of using simulation to determine a p-value for a hypothesis test.

Notes:

- Use $\frac{24}{40}$ as the common proportion of ‘yes’ answers
- Simulate 20 responses for men and an independent 20 responses for women at least 10,000 times and count the number of times the difference in ‘yes’ responses is at least 6.

```r
VotingYes <- function(n, men, women, p) {
  counter <- 0
  for (k in 1:n) {
    success_men <- rep(0, men); success_women <- rep(0, women)
    for (i in 1:men) {
      U <- runif(1)
      if (U < p) {
        success_men[i] <- 1
      }
    }
    for (j in 1:women) {
      U <- runif(1)
      if (U < p) {
        success_women[j] <- 1
      }
    }
    total_success_men <- sum(success_men); total_success_women <- sum(success_women)
    if (abs(total_success_men - total_success_women) >= 6) {
      counter <- counter + 1
    }
  }
  counter
}

VotingYes(10000, 20, 20, 24/40)
```

```
## [1] 744
```

7) From An Introduction to Statistical Computing by Voss. [Question E1.4 in textbook] Write a program which uses the inverse transform method to generate random numbers with the following density:

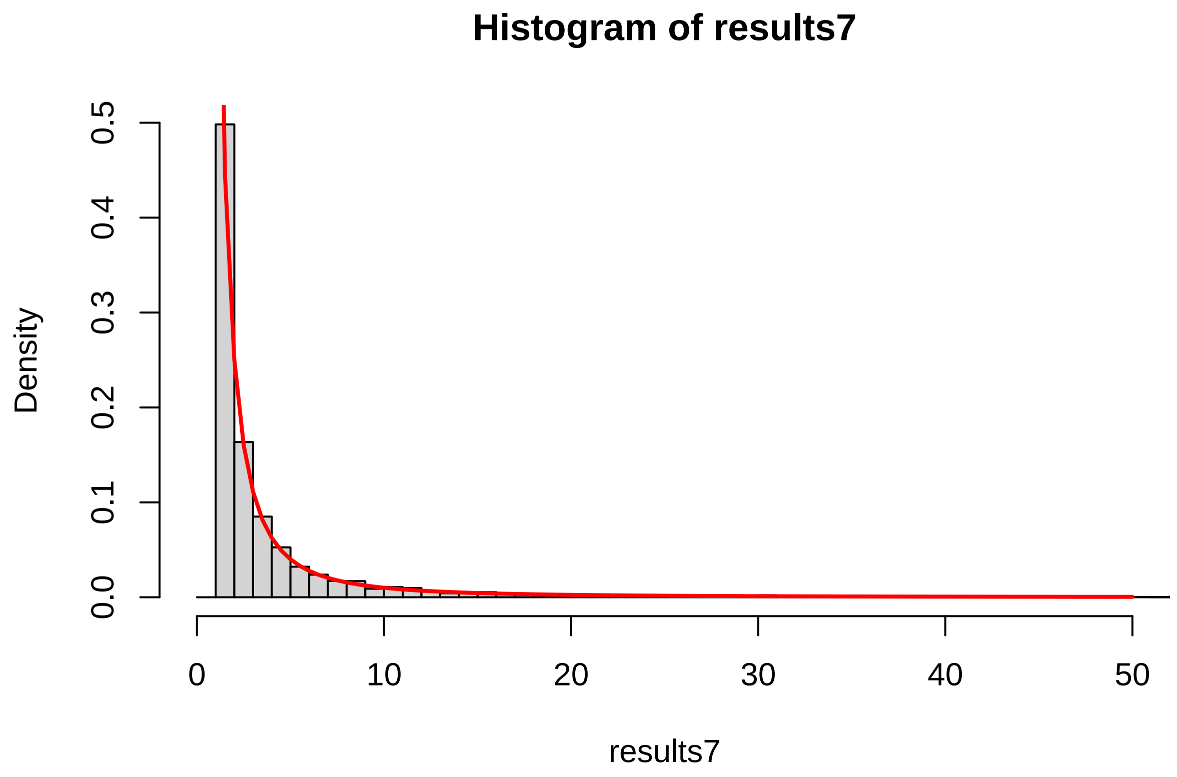
$$f(x) = \begin{cases} \frac{1}{x^2} & x \geq 1 \\ 0 & \text{elsewhere} \end{cases}$$

Generate 10,000 values of X according to this density function, print out the largest value of X, and then make a relative frequency histogram of your values together with the density function f. (Hint: You'll need to use the commands: hist(X, breaks = seq(0, max(X)+binsize, binsize), xlim = c(0,50), freq = FALSE); lines(x,Y, col = 'red') The first scales the graph, while the second adds on the density function, with x = seq(1,50,0.5) [or similar] and $Y = \frac{1}{x^2}$)

```
## f(x) = 1/x^2 --> F(x) = -1/t ]_1^x = -1/x - -1/1 --> U = -1/x + 1 --> X = -1/(U-1)
GenerateX <- function(n) {
  X <- rep(0, n)
  for (i in 1:n) {
    U <- runif(1)
    X[i] <- -1/(U-1)
  }
  return(X)
}

results7 <- GenerateX(10000)
```

```
hist(results7, breaks = seq(0, max(results7)+1, 1), xlim = c(0,50), freq = FALSE)
X <- seq(1, 50, 0.5); Y <- 1/(X^2)
lines(X, Y, col = 'red', lwd = 2)
```



8) From Simulation by Ross, Chapter 5. [Question 1 in textbook] Use the inverse CDF method to generate a random variable having density function.

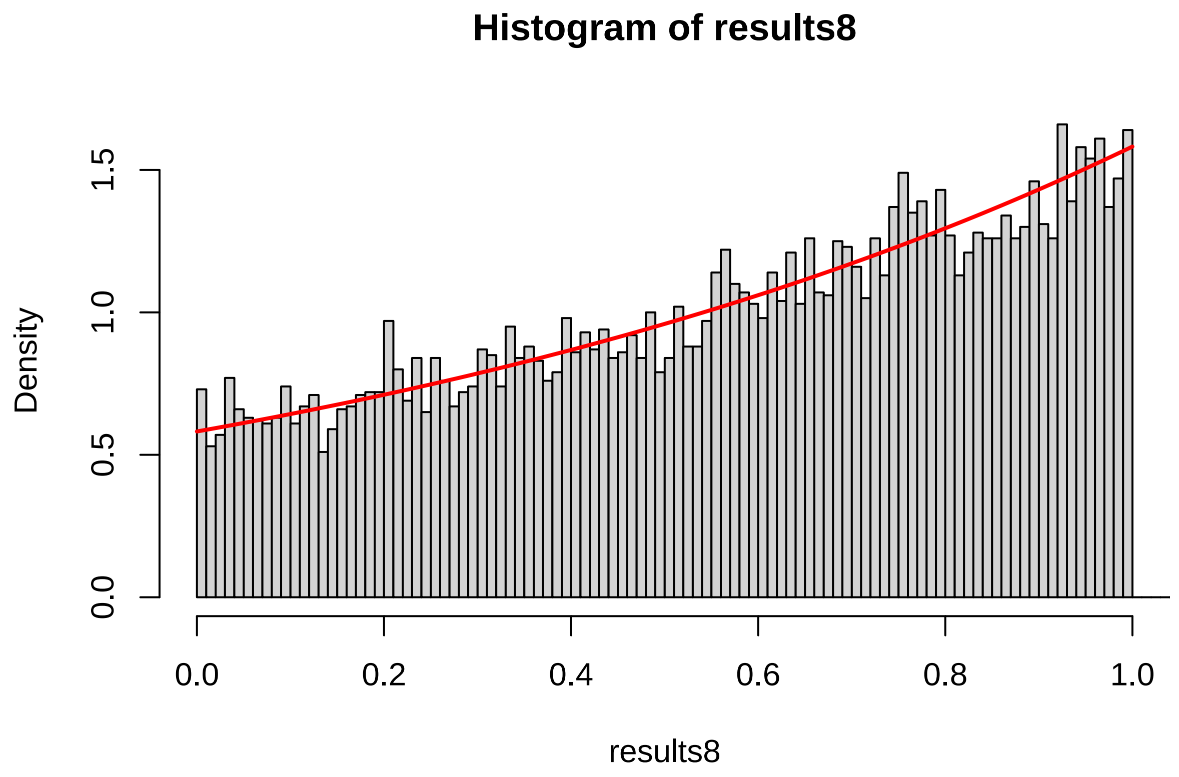
$$f(x) = \frac{e^x}{e-1} 0 \leq x \leq 1\}$$

Generate 10,000 values of X according to this density function and then make a relative frequency histogram of your values together with the density function f. (Hint: You'll need to use the commands: hist(X, breaks = 40, freq = FALSE); lines(x,y, col = 'red'). The first scales the graph, while the second adds on the density function, with x = seq(0,1,0.01) [or similar] and y = exp(x)/(exp(1)-1).)

```r
## f(x) = e^x/(e-1) --> F(x) =  e^t/e-1]_0^x =  --> e^x/e-1 - 1/e-1
## U = e^x/e-1 - 1/e-1 --> (e-1)(U + 1/e-1) = e^x --> X = ln((e-1)(U + 1/e-1))
GenerateX <- function(n) {
  X <- rep(0, n)
  for (i in 1:n) {
    U <- runif(1)
    X[i] <- log(x = (exp(1) - 1)*(U + (1/(exp(1)-1))), base = exp(1))
  }
  return(X)
}

results8 <- GenerateX(10000)
```

```r
hist(results8, breaks = seq(0, max(results8)+.1, .01), xlim = c(0,1), freq = FALSE)
X <- seq(0, 1, 0.01); Y <- exp(X)/(exp(1)-1)
lines(X, Y, col = 'red', lwd = 2)
```



9) [Question 4 in textbook] A random variable with distribution function

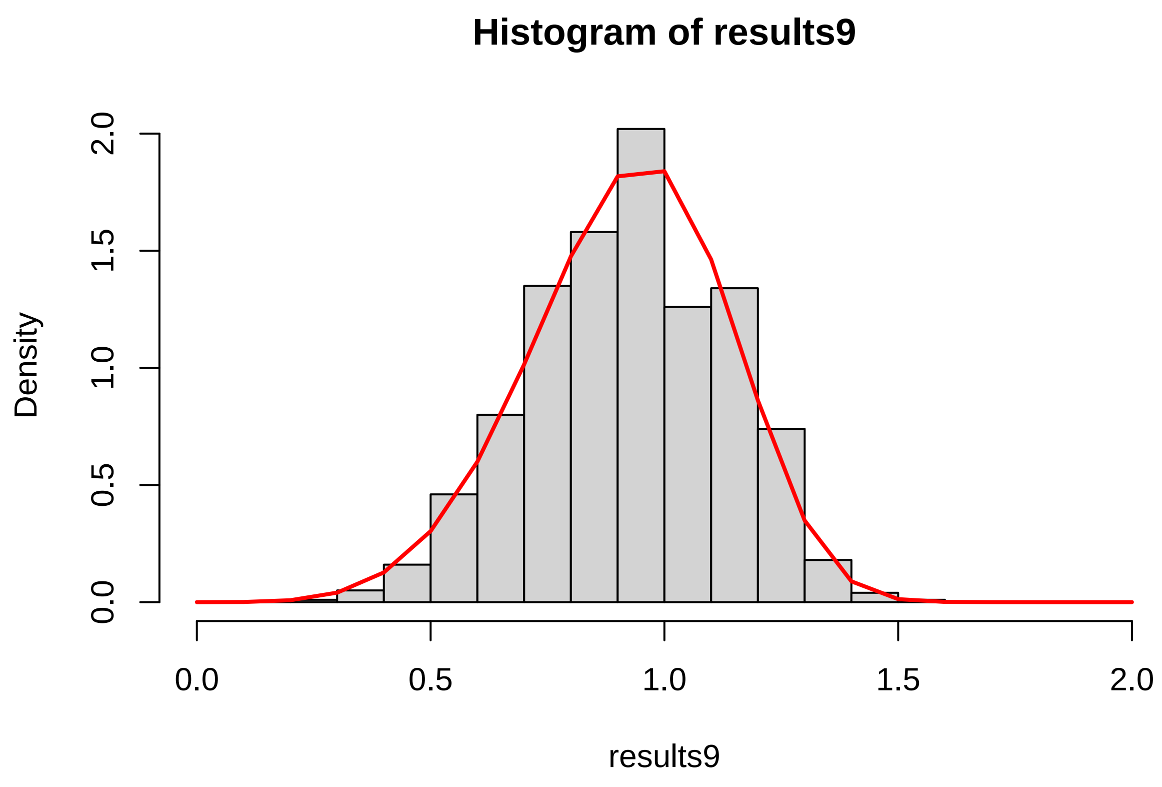
$$F(x) = 1 - exp(-\alpha x^{\beta}), 0 \leq x \leq \infty$$

is said to be a Weibull random variable with scale parameter α and shape parameter β. Use the inverse CDF method to generate 1000 realizations of a Weibull(1,5) distribution and then make a histogram of your simulated values with the density function superimposed. Note: Use dweibull(x, shape, scale) with an appropriate range of values for x to create the density function.

```r
## F(x) = 1 - e^(-ax^b)
## U =  1 - e^(-ax^b) --> 1 - U = e^(-ax^b) --> ln(1 - U) = -ax^b --> (ln(1 - U)/-a)^1/b = x
GenerateWeibull <- function(n, alpha, beta) {
  X <- rep(0, n)
  for (i in 1:n) {
    U <- runif(1)
    X[i] <- (log(1 - U)/-alpha)^(1/beta)
  }
  return(X)
}

results9 <- GenerateWeibull(1000, 1, 5)
```

```r
hist(results9, breaks = seq(0, max(results9)+.5, .1), xlim = c(0,2), freq = FALSE)
X <- seq(0, 2, .1); Weibull <- dweibull(X, 5, 1)
lines(X, Weibull, col = 'red', lwd = 2)
```



10) From Statistical Computing with R by Rizzo, Chapter 3. [Question 2 in textbook] The standard Laplace distribution has density $f(x) = \frac{1}{2}e^{-|x|}$, for $x \in \mathbb{R}$. Use the inverse transform method to generate a random sample of size 1000 from this distribution and then make a histogram of your simulated values with the density function superimposed. (Note: If you install the "VGAM" package, you can use dlaplace(x, location = 0, scale = 1, log = FALSE) with an appropriate range for x to create the density function. Otherwise, follow the approach from Q8.) Hint:

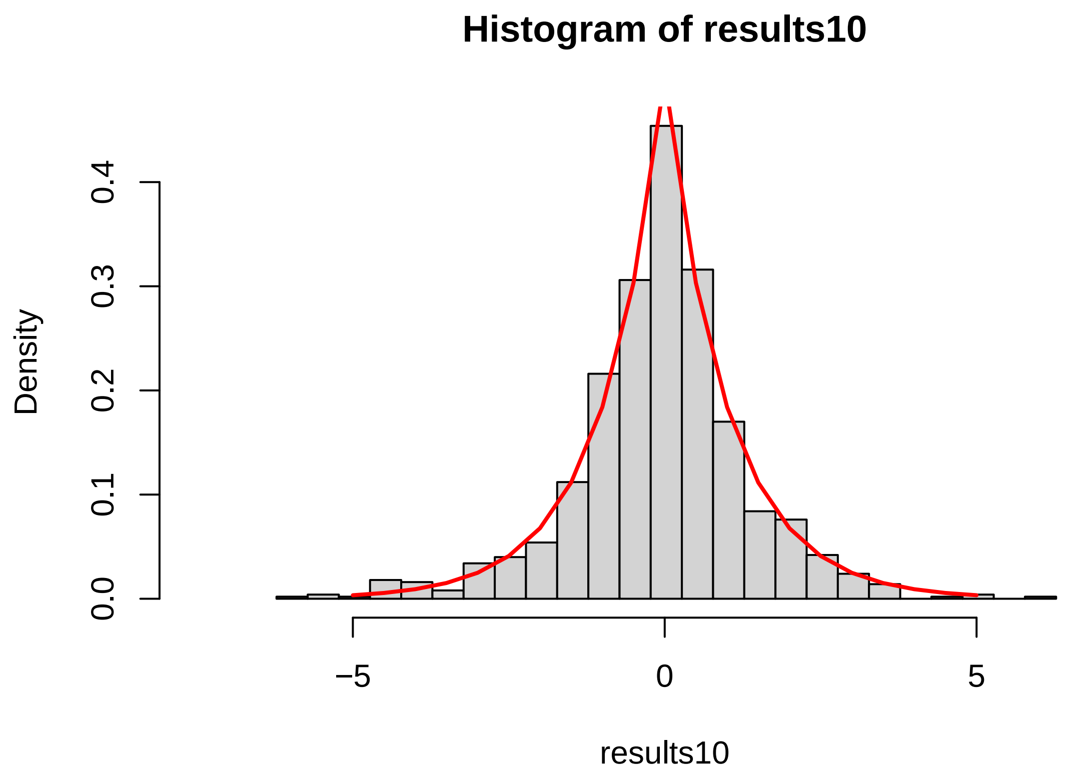
$$f(x) = \begin{cases} \frac{1}{2}e^{x} & x < 0 \\ \frac{1}{2}e^{-}x & x \geq 0 \end{cases}$$

So create the sampler in two parts, one where $U < \frac{1}{2}$ and another for $U \geq \frac{1}{2}$. For the latter, the CDF is $\frac{1}{2} + \int_0^x \frac{1}{2}e^{-x}dx$

```r
GenerateLaplace <- function(n) {
  X <- rep(0, n)
  for (i in 1:n) {
    U <- runif(1)
    if (U < (1/2)) {
      X[i] <- log(2*U, base = exp(1))
    }
    else {
      X[i] <- -1*log(2-2*U, base = exp(1))
    }
  }
  return(X)
}

set.seed(942)
results10 <- GenerateLaplace(1000)
```

```r
hist(results10, breaks = seq(min(results10) - .5, max(results10) + .5, .5),
     xlim = c(-7.5, 7.5), freq = FALSE)
X <- seq(-5, 5, .5); Laplace <- dlaplace(X, location = 0, scale = 1, log = FALSE)
lines(X, Laplace, col = 'red', lwd = 2)
```


Histogram of results10
Density
0.0
0.1
0.2
0.3
0.4
−5
0
5
results10