Computational statistics Lab 03 Report

Karthikeyan Devarajan - Karde799

Question 1: Cluster sampling

1.Import necessary information to R.

2. Use a uniform random number generator to create a function that selects 1 city from the whole list by the probability scheme offered above (do not use standard sampling functions present in R).

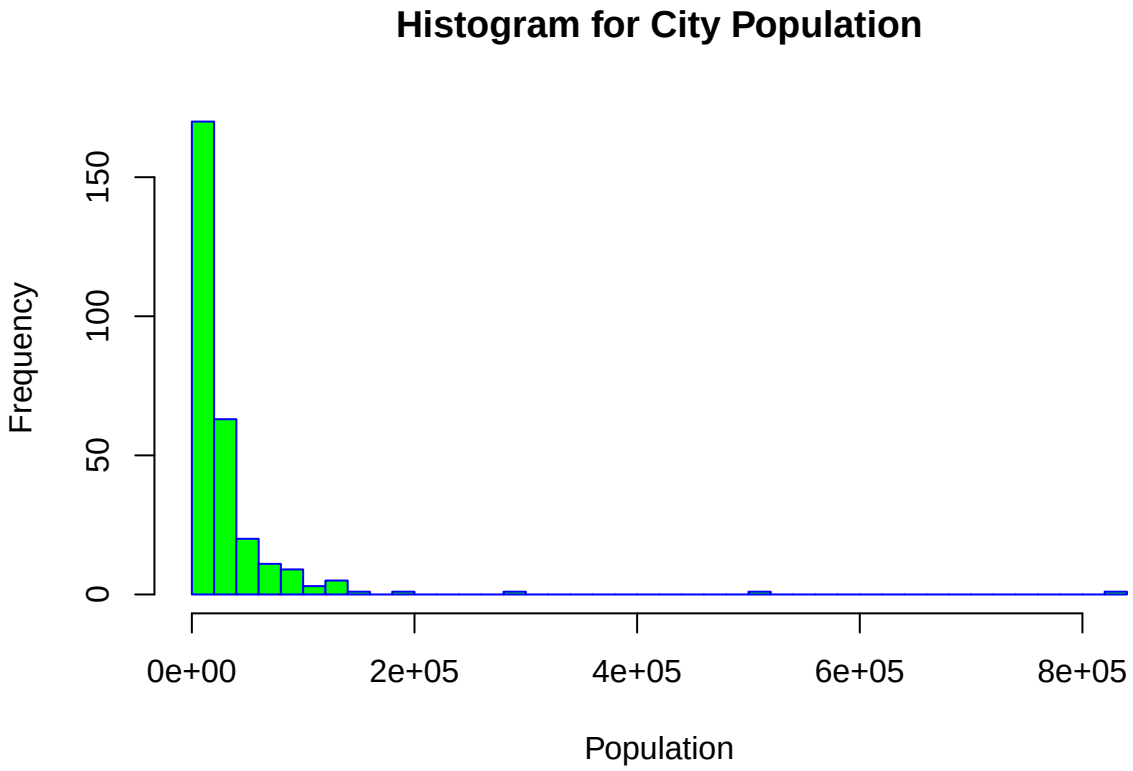
3. Use the function you have created in step 2 as follows

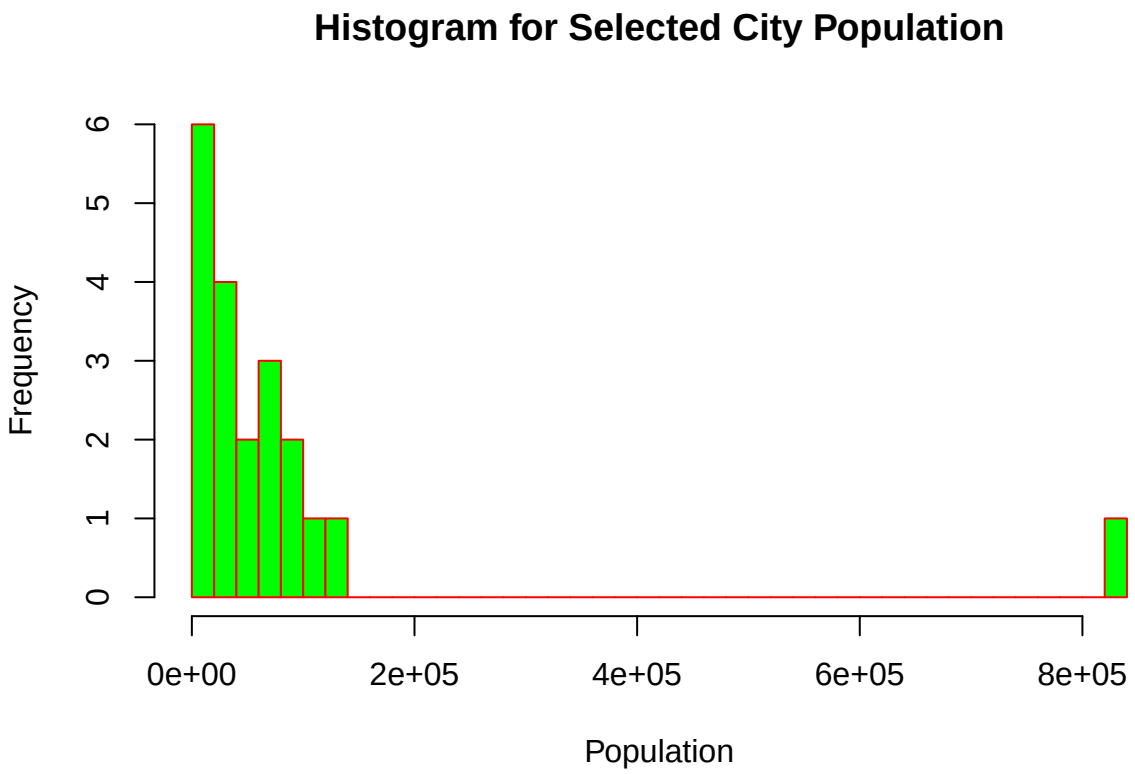
4. Run the program. Which cities were selected? What can you say about the size of the selected cities?

```
##       twenty_cities  population
##  [1,] "Huddinge"     "95798"
##  [2,] "Norrtälje"    "55927"
##  [3,] "Stockholm"    "829417"
##  [4,] "Södertälje"   "85270"
##  [5,] "Vingåker"     "8911"
##  [6,] "Jönköping"    "126331"
##  [7,] "Klippan"      "16382"
##  [8,] "Kristianstad" "78788"
##  [9,] "Kävlinge"     "28638"
## [10,] "Lund"         "109147"
## [11,] "Perstorp"     "6983"
## [12,] "Staffanstorp" "21949"
## [13,] "Stenungsund"  "23983"
## [14,] "Trollhättan"  "54873"
## [15,] "Filipstad"    "10626"
## [16,] "Grums"        "9142"
## [17,] "Bollnäs"      "26175"
## [18,] "Hofors"       "9873"
## [19,] "Skellefteå"   "71770"
## [20,] "Luleå"        "73950"
```

Cities which have large number of population have a big probability to getting selected. So Selected cities have relativly higher population. Some cities do not have high population becuase of the we use random sampling without replacement.

5. Plot one histogram showing the size of all cities of the country. Plot another histogram showing the size of the 20 selected cities. Conclusions?





The cities we choose using the propotional probabilities are close to each other, whereas in the histogram of all cities there are many outliers due to the big difference in the population for each city.

Question 2: Different distributions

1. Write a code generating double exponential distribution DE(0, 1) from Unif(0, 1) by using the inverse CDF method. Explain how you obtained that code step by step. Generate 10000 random numbers from this distribution, plot the histogram and comment whether the result looks reasonable.

Double exponential distribution

$DE(\mu, \alpha) = \frac{\alpha}{2} \exp(-\alpha|x - \mu|)$

When $x \leq \mu$ PDF;

$f(x\ |\mu, \alpha) = \frac{\alpha}{2} \exp(-\alpha|x - \mu|)$

$f(x\ |\mu, \alpha) = \frac{\alpha}{2} \exp(-\alpha(\mu - x))$

When $x > \mu$ PDF;

$f(x\ |\mu, \alpha) = \frac{\alpha}{2} \exp(-\alpha|x - \mu|)$

$f(x\ |\mu, \alpha) = \frac{\alpha}{2} \exp(-\alpha(x - \mu))$

Now let's find the CDF Function;

When $x \leq \mu$ CDF;

$F_X(x) = \int_{-\infty}^{x} \frac{\alpha}{2} \exp(-\alpha(\mu - x))dx$

$F_X(x) = \left[\frac{exp(-\alpha(\mu-x))}{2}\right]_{-\infty}^{x}$

$F_X(x) = \left[\frac{exp(-\alpha(\mu-x))}{2} - 0\right]$

$F_X(x) = \left[\frac{exp(\alpha(x-\mu))}{2}\right]; x \leq \mu$

When $x > \mu$ CDF;

$F_X(x) = \int_{\mu}^{x} \frac{\alpha}{2} \exp(-\alpha(x - \mu))dx$

$F_X(x) = \left[\frac{-exp(-\alpha(x-\mu))}{2}\right]_{\mu}^{x}$

$F_X(x) = \left[\frac{-exp(-\alpha(x-\mu))}{2}\right]_{\mu}^{x}$

$F_X(x) = \left[\frac{-exp(-\alpha(x-\mu))}{2} + 1\right]$

$F_X(x) = \left[1 - \frac{exp(-\alpha(x-\mu))}{2}\right]; x > \mu$

Now let's find the Inverse $F_X^{-1}(y)$ Function;

When $x \leq \mu$, $F_X^{-1}(y)$;

$y = \frac{1}{2} \exp(\alpha(x - \mu))$

$2y = \exp(\alpha(x - \mu))$

$\ln(2y) = \alpha(x - \mu)$

$\frac{\ln(2y)}{\alpha} = (x - \mu)$

$x = \mu + \frac{\ln(2y)}{\alpha}$

$F_X^{-1}(y) = \mu + \frac{\ln(2y)}{\alpha}; y \leq 0.5$

When $x > \mu$ $F_X^{-1}(y)$;

$y = 1 - \frac{\exp(-\alpha(x-\mu))}{2}$

$1 - y = \frac{\exp(-\alpha(x-\mu))}{2}$

$2(1 - y) = \exp(-\alpha(x - \mu))$

$\ln(2(1 - y)) = -\alpha(x - \mu)$

$\frac{-\ln(2(1-y))}{\alpha} = x - \mu$

$x = \mu - \frac{\ln(2(1-y))}{\alpha}$

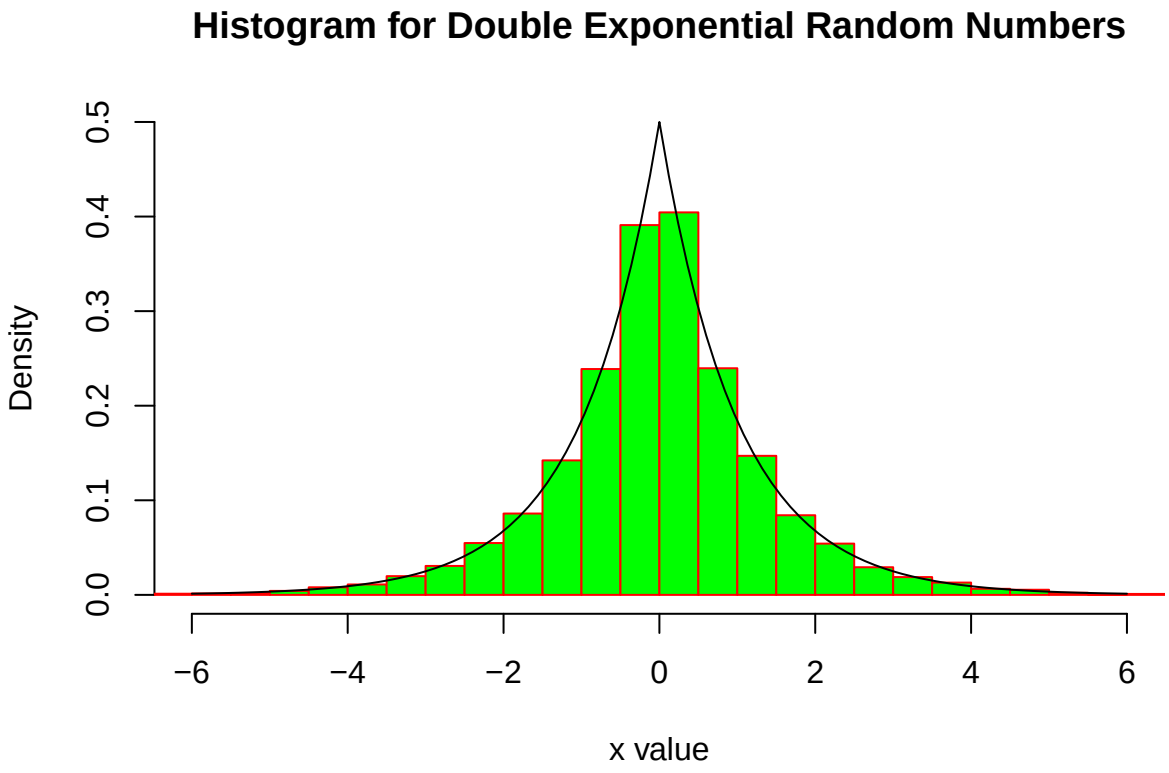
$F_X^{-1}(y) = \mu - \frac{\ln(2(1-y))}{\alpha}; y > 0.5$

Since $X \sim DE(0, 1)$

$F_X^{-1}(y) = \ln(2y); y \leq 0.5$

$F_X^{-1}(y) = -\ln(2(1-y))\,; y > 0.5$

$x \sim DE(0,1)$

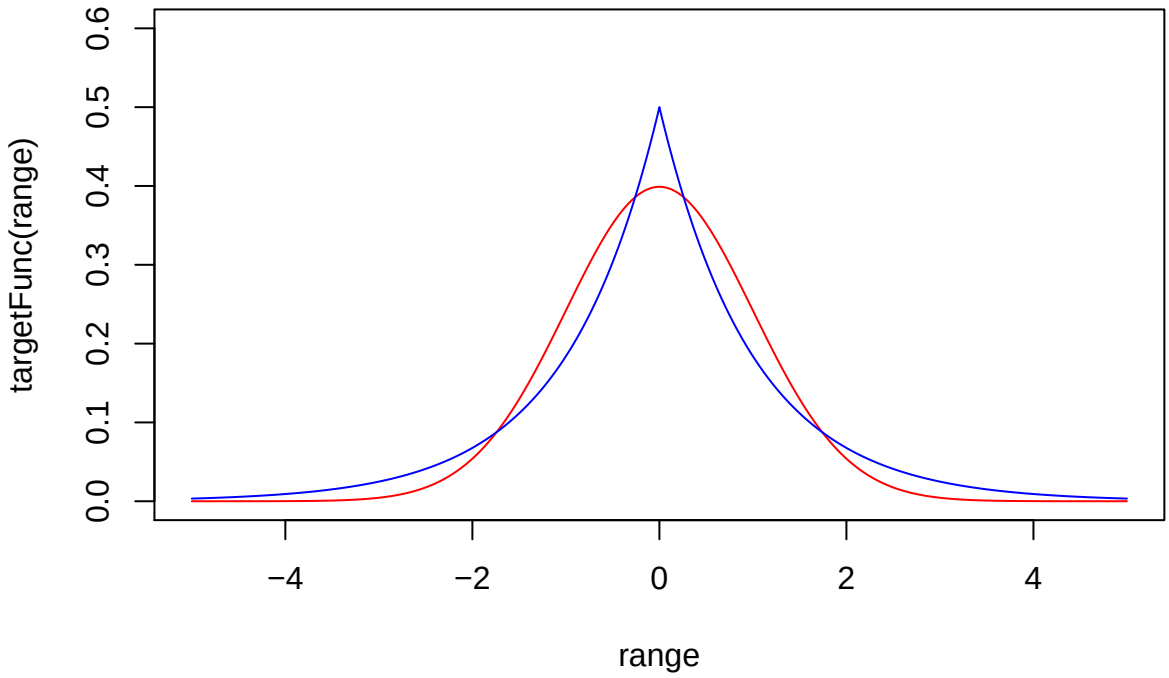


Genrated random numbers for Double Exponential Distribution is close to the R generated 'dlaplace' curve. Hence results look reasonable.

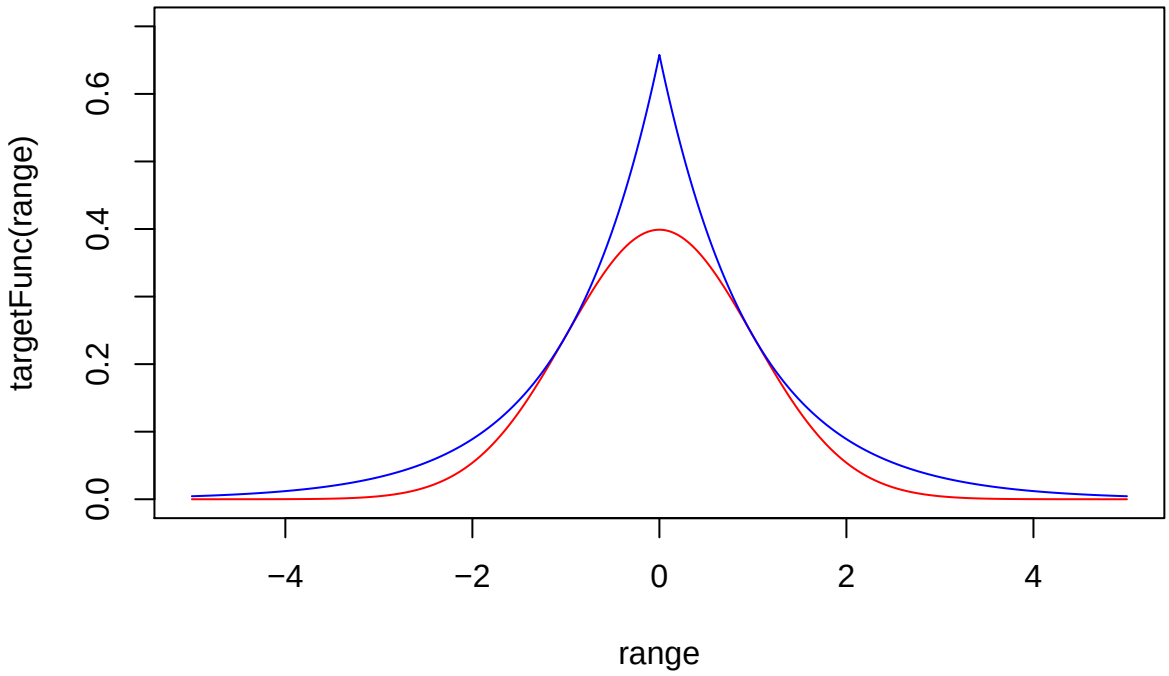
2. Use the Acceptance/rejection method with DE(0,1) as a majorizing density to generate N(0,1) variables. Explain step by step how this was done. How did you choose constant c in this method? Generate 2000 random numbers N(0,1) using your code and plot the histogram. Compute the average rejection rate R in the acceptance/rejection procedure. What is the expected rejection rate ER and how close is it to R? Generate 2000 num- bers from N (0, 1) using standard rnorm() procedure, plot the histogram and compare the obtained two histograms.

N(0,1) is the target function which we need to generate random numbers. DE(0,1) is the majorizing function.

We can calculate c value by taking the maximum of our (target function divided by majorizing function) or taking upper integral limit.Now we need to find constant 'c' value which is Normal Distribution should be fully coverd by the Double Exponential Distribution. Without 'c' paramter, Normal distrubution is not fully covered by the Double Exponential Distribution:-

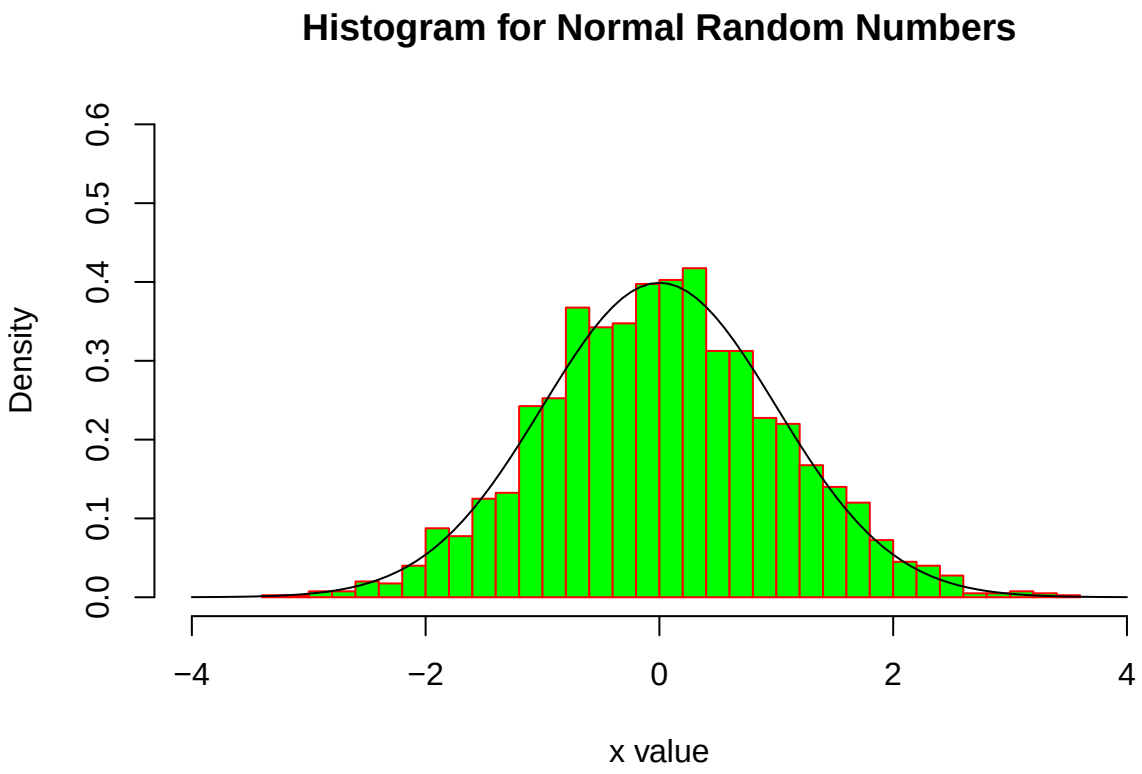


Now we need to find the minimum 'c' parameter which is $max(targetFunc(x)/knownFunc(x))$. After Applying c :-




```
## Optimal C value is :-  1.315489
```

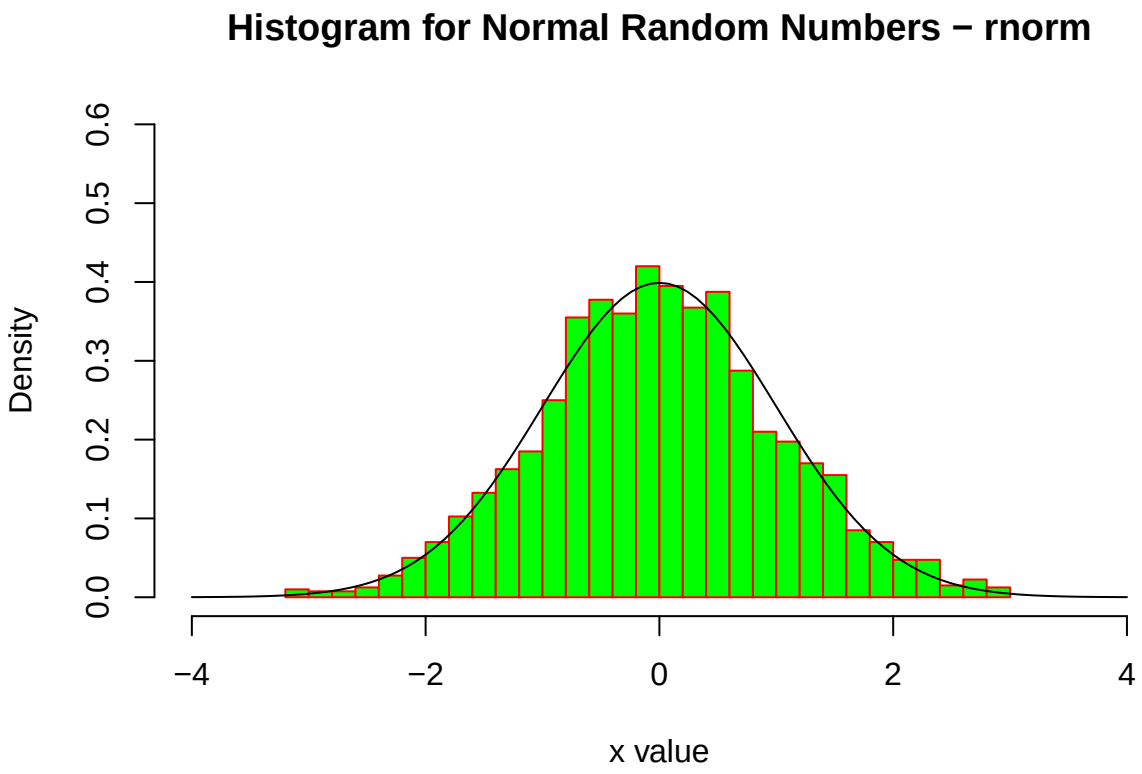
Create Randome 2000 points




```
## Rejection Rate :-  24.86852

## Expected Rejection Rate :-  23.98265
```

rnorm created Normal Random Number generator



Since set.seed makes the randomization produce similar each iteration.The Rstudio uses same mechanism for generating random numbers for rnorm and for our code. This was the cause that the two graph (i.e random number generation and normal random number) are same.

References

1. https://en.wikipedia.org/wiki/Laplace_distribution
2. https://www.youtube.com/watch?v=WFswb-zLe4Y

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
# Question 1: Cluster sampling
library(readxl)
RNGversion(min(as.character(getRversion()),"3.6.2"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
data = read_xls(file.choose())
data = data[9:nrow(data),1:3]
colnames(data) = c("id","city","population")
data$id = as.integer(data$id)
data <- data[data$id >99,]
data$population = as.integer(data$population)
# Part 2
```

```
select_city = function (data){
           sum_population = sum(data$population)
           population_precentage = data$population/sum_population
           cumulative = 0
           data=data
           data = cbind(data, cumulative)
           #data = data[order(data$population),]

           #Calculating the proportion of cities
             for (i in 1:length(data$population)){
               if (i == 1){
                 data$cumulative[i] = data$population[i]
               }
               else{
               data$cumulative[i] = (data$population[i] +data$cumulative[i-1])
               }
             }
           cumulative_percentage = data$cumulative/sum_population
           data <- cbind(data,cumulative_percentage)

           city = 0
           random_value = runif(1,0,1)
           for (i in 1:length(data$cumulative_percentage)){
             if (random_value <= data$cumulative_percentage[i]){
               city = data$city[i]
               break
             }
           }
           return(city)
         }
# Part 3
          twenty_cities = c()
          function_data = data
          for (i in 1:20){
           city_num = select_city(data = function_data)
           twenty_cities = c(twenty_cities,city_num)
#b: removing the city from the list
           function_data = function_data[!function_data$city==city_num,]
          }

          population = data[which(data$city %in% twenty_cities),]$population
          twenty_cities = data[which(data$city %in% twenty_cities),]$city
          df = cbind(twenty_cities,population)

          # plot(population, xlab = "Cities", ylab = "The Population", main = "20 Cities vs Population"

# Part 4

 print(df)

hist(data$population,
     main = "Histogram for City Population",
     xlab = "Population",
```

```
     border = "blue",
     col = "green",
     breaks = 30)

hist(population,
     main = "Histogram for Selected City Population",
     xlab = "Population",
     border = "red",
     col = "green",
     breaks = 30)

library(LaplacesDemon)

unifor_vect = runif(10000, min = 0, max = 1)

getLaplas <- function(uniformVect) {
  laplase_vect = c()
  for (item in uniformVect) {
    if (item <= 0.5) {
      data_point = log(2 * item)
      laplase_vect = c(laplase_vect, data_point)
    } else {
      data_point = -log(2 * (1 - item))
      laplase_vect = c(laplase_vect, data_point)
    }
  }
  return(laplase_vect)
}

laplase_data = getLaplas(unifor_vect)

hist(laplase_data,
     main = "Histogram for Double Exponential Random Numbers",
     xlab = "x value",
     border = "red",
     col = "green",
     prob = TRUE,
     ylim = c(0,0.5),
     xlim = c(-6,6),
     breaks = 30)
curve(dlaplace(x,0,1), add = TRUE)



targetFunc = function(x) {
  # x ~ N(0,1)
  # \mu = 0 , \sigma = 1
  return(sqrt(1/(2*pi)) * exp(-x^2 / 2))
}

knownFunc = function(x) {
  return((1/2) * exp(-abs(x)))
}
```

```r
range = seq(-5,5,0.001) # For Graphing

plot(x = range,
     y = targetFunc(range),
     type = 'l',
     ylim = c(0,0.6),
     col = 'red')

lines(x = range,
     y = knownFunc(range),
     col = 'blue')
min_M = max(targetFunc(range)/knownFunc(range))

plot(x = range,
     y = targetFunc(range),
     type = 'l',
     ylim = c(0,0.7),
     col = 'red')

lines(x = range,
      y = min_M *knownFunc(range),
      col = 'blue')
cat('Optimal C value is :- ', min_M, '\n')
getNormalPoints = function(n) {
  normal_vect = c()
  allPoints = 0
  for (i in 1:n) {
    repeat {
      allPoints = allPoints + 1
      x = getLaplas(runif(1, min = 0, max = 1))
      u = runif(1, min = 0, max = 1)
      if (u < targetFunc(x) / (min_M * knownFunc(x))) {
        break
      }
    }
    normal_vect = c(normal_vect, x)
  }
  returnData = list("Data" = normal_vect,"RR" = ((allPoints - n) / allPoints) * 100)
  return(returnData)
}

normalPoints = getNormalPoints(2000)

hist(normalPoints$Data,
     main = "Histogram for Normal Random Numbers",
     xlab = "x value",
     border = "red",
     col = "green",
     prob = TRUE,
     ylim = c(0,0.6),
     xlim = c(-4,4),
     breaks = 30)
curve(dnorm(x,0,1), add = TRUE)
```

```r
rej_rate =  normalPoints$RR
exp_rej_rate = (1 - (1 / min_M)) * 100

cat('Rejection Rate :- ', rej_rate, '\n')
cat('Expected Rejection Rate :- ', exp_rej_rate, '\n')


# --- rnorm

hist(rnorm(2000, mean = 0, sd = 1),
     main = "Histogram for Normal Random Numbers - rnorm",
     xlab = "x value",
     border = "red",
     col = "green",
     prob = TRUE,
     ylim = c(0,0.6),
     xlim = c(-4,4),
     breaks = 30)
curve(dnorm(x,0,1), add = TRUE)
```