Computational Statistics - Lab 3

Yusur Al-Mter (yusal621), Lakshidaa Saigiridharan (laksa656)

2/8/2019

Contents

QUESTION 1 : Cluster sampling **1**
TASK 1.1: 1
TASK 1.2: 1
TASK 1.3: 2
TASK 1.4 2
TASK 1.5 3
QUESTION 2: Different distributions **5**
TASK 2.1 5
TASK 2.2 7
Appendix **10**

QUESTION 1 : Cluster sampling

An opinion pool is assumed to be performed in several locations of Sweden by sending interviewers to this location. Of course, it is unreasonable from the financial point of view to visit each city. Instead, a decision was done to use random sampling without replacement with the probabilities proportional to the number of inhabitants of the city to select 20 cities. Explore the file population.xls. Note that names in bold are counties, not cities.

TASK 1.1:

Import necessary information to R.

```r
# TASK 1.1

# Importing the file
data <- read.csv2("population.csv")
```

TASK 1.2:

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).

```r
# TASK 1.2

# Function which uses a uniform random number generator to select a city from a list
city_selection <- function(data){
data$cumsum <- cumsum(data$Population)/sum(data$Population)
selected <- which(runif(1) <= data$cumsum)[1]
city_selected <- data$Municipality[selected]
as.numeric(rownames(data[selected,]))
}
```

TASK 1.3:

Use the function you have created in step 2 as follows:

(a) Apply it to the list of all cities and select one city

(b) Remove this city from the list

(c) Apply this function again to the updated list of the cities

(d) Remove this city from the list

(e) . . . and so on until you get exactly 20 cities.

```
# TASK 1.3

select <- data.frame()

# List of all cities stored in data1
data1 <- data

# Selecting 20 cities
for(i in 1:20){
  selected <- which(city_selection(data1) == as.numeric(rownames(data)))
  select <- rbind(select, data1[selected,])
  # Removing the selected city from the list
  data1 <- data1[-c(selected),]
}
```

TASK 1.4

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

```
# TASK 1.4

# Dataframe of 20 selected cities
cat("Selected cities : \n")
```

```
## Selected cities :
```

```
select$Municipality
```

```
##  [1] Borgholm         Sollefte\345     Osby             Torsby
##  [5] Sigtuna          Bjurholm         Hjo              Gnesta
##  [9] Trosa            Ludvika          Mullsj\366       Tran\345s
## [13] Haninge          S\366dert\344lje Arvidsjaur       Enk\366ping
## [17] Oskarshamn       Flen             Gullsp\345ng     Sval\366v
## 290 Levels: Ale Alings\345s Alvesta Aneby Arboga Arjeplog ... \326vertorne\345
```

The selected cities along with their respective size is shown below :

```
# Selected cities along with their respective sizes
city_size = data.frame(City = select$Municipality, Size = select$Population)

kable(city_size, booktabs = T) %>%
kable_styling(latex_option = "striped")
```

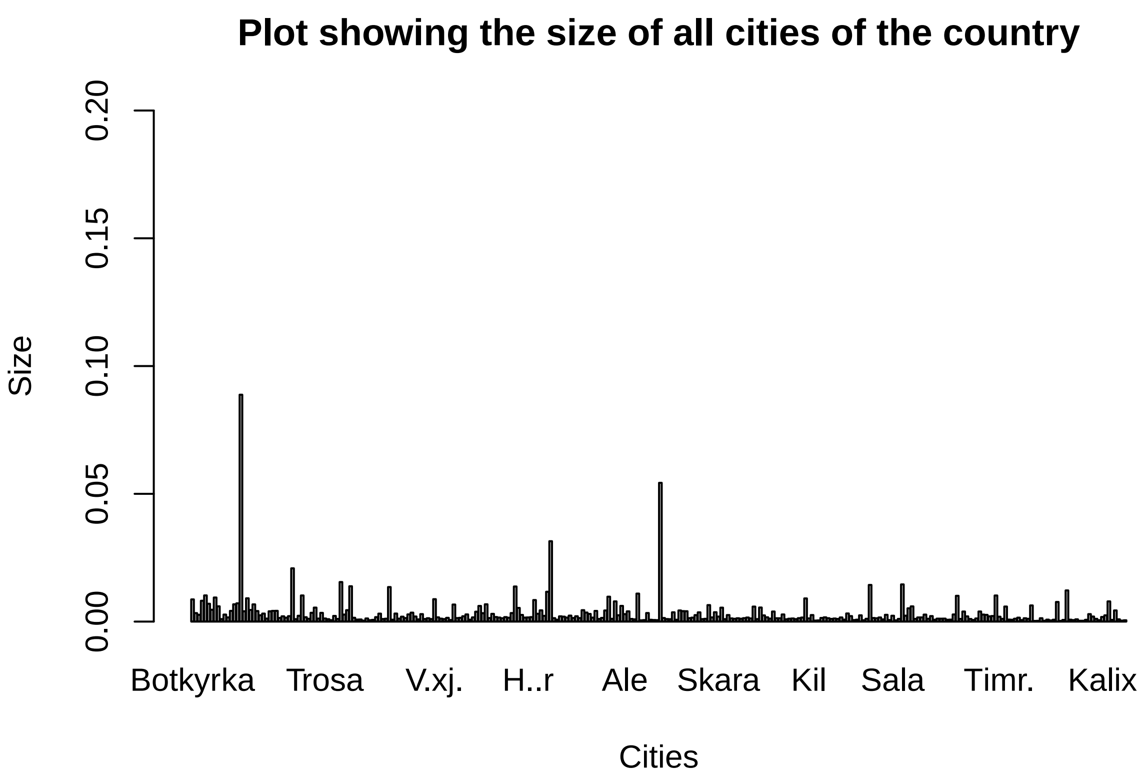
City	Size
Borgholm	10806
Sollefte<e5>	20442
Osby	12656
Torsby	12508
Sigtuna	39219
Bjurholm	2500
Hjo	8859
Gnesta	10318
Trosa	11446
Ludvika	25650
Mullsj<f6>	7027
Tran<e5>s	18043
Haninge	76237
S<f6>dert<e4>lje	85270
Arvidsjaur	6622
Enk<f6>ping	39360
Oskarshamn	26232
Flen	16139
Gullsp<e5>ng	5335
Sval<f6>v	13290

On observing the sizes of the cities that have been selected by the function, it can be seen that the function picked 20 cities which are of relatively large sizes. This is because the cities with huge populations have a much higher probability of being selected than the ones with small population.

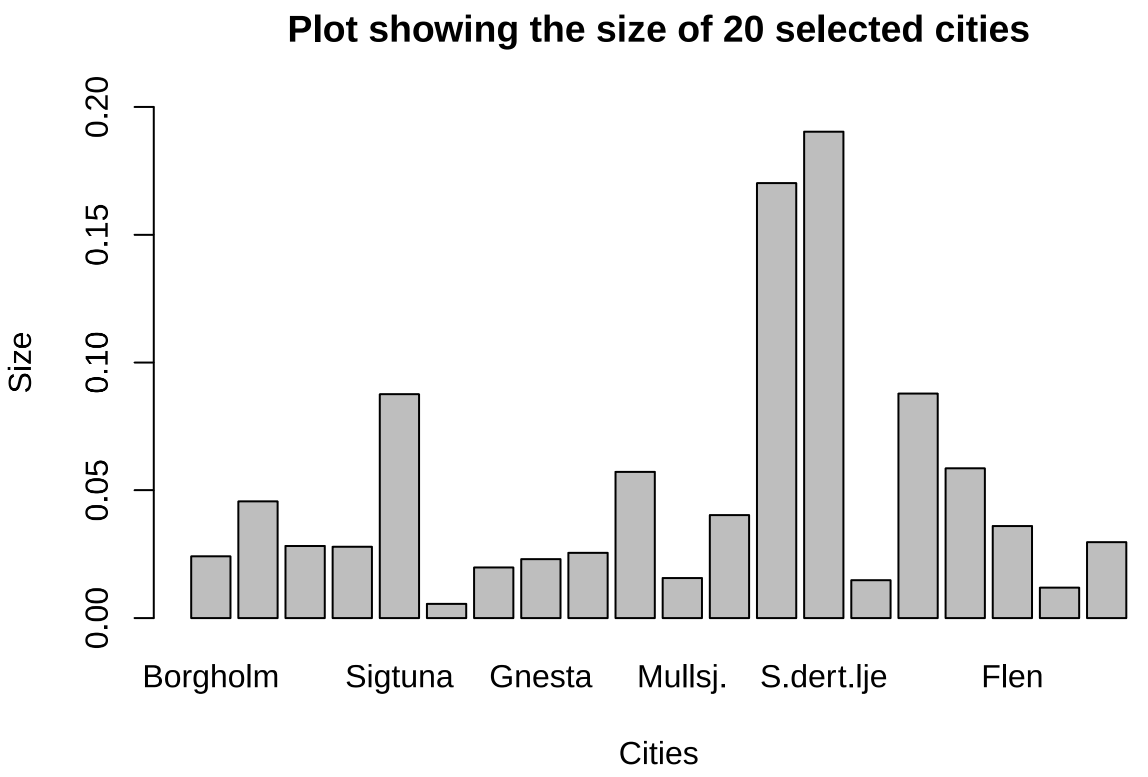
TASK 1.5

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

```
# TASK 1.5

# Plot showing the size of all cities of the country
percentage<-data$Population/sum(data$Population)
barplot(percentage,names.arg = data$Municipality, xlab="Cities", ylab = "Size", ylim = c(0, 0.20))
title("Plot showing the size of all cities of the country")
```



```
# Plot showing the size of 20 selected cities
percentage_select<-select$Population/sum(select$Population)
barplot(percentage_select, names.arg = select$Municipality, xlab="Cities", ylab = "Size", ylim = c(0, 0
title("Plot showing the size of 20 selected cities")
```



It can be seen that the two histograms follow a similar distribution. This shows that the function implemented to select 20 cities based on the probabilities proportional to the number of inhabitants of the city performs a good job in the city selection.

QUESTION 2: Different distributions

TASK 2.1

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.

The probability density for the double exponential (Laplace) function (PDF) is given by the formula:

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

where the variable x is a real number as is the location parameter μ while the parameter α is a real positive number.

The cumulative density function (CDF) of a continuous random variable X is defined as:

$$F(x) = \int_{-\infty}^{x} f(x)dx, \qquad -\infty < x < \infty$$

Thus, the cumulative distribution for the double exponential distribution is given by:

For $x > \mu$:

$$F(x) = \int_{-\infty}^{x} \frac{\alpha}{2} e^{-\alpha(x-\mu)}dx, \qquad x > \mu$$

$$F(x) = 1 - \int_{x}^{\infty} \frac{\alpha}{2} e^{-\alpha(x-\mu)}dx$$

Integrating with respect to x, we have:

$$F(x) = 1 - \frac{1}{2} e^{-\alpha(x-\mu)}$$

Now, for $x \leq \mu$:

$$F(x) = \int_{-\infty}^{x} \frac{\alpha}{2} e^{\alpha(x-\mu)}dx, \qquad x \leq \mu$$

Integrating with respect to x, we have:

$$F(x) = \frac{1}{2} e^{\alpha(x-\mu)}$$

The inverse CDF technique:

Given a uniform random number between zero and one in U (i.e $U \sim U(0,1)$), a random number from a Double-exponential distribution is given by solving the equation F(x)=U for x, as shown below:

For $U > 1/2$, we have:

$$U = 1 - \frac{1}{2} e^{-\alpha(x-\mu)}$$

Solving for x, we will get:

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

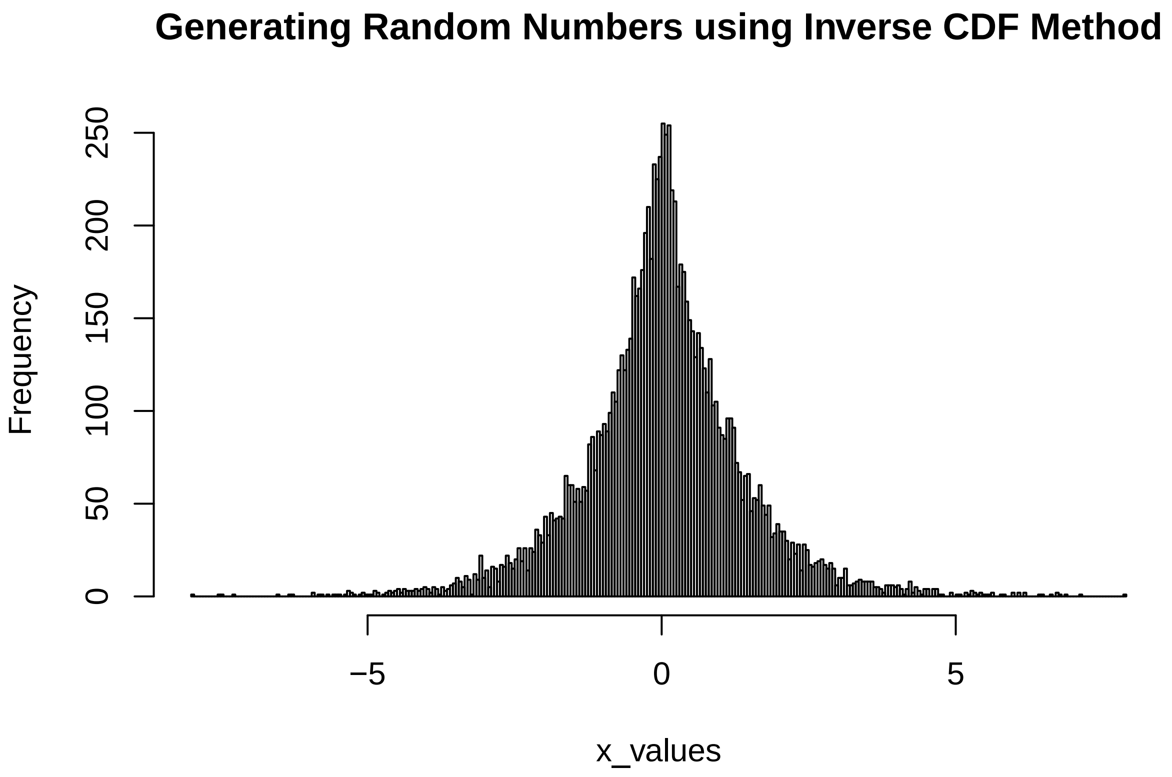
For $U \leq 1/2$, we have:

$$U = \frac{1}{2}e^{\alpha(x-\mu)}$$

Solving for x, we will get:

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

```
# TASK 2.1

inverse_cdf = function(mu,alpha){
  U = runif(1,0,1)
  if (U <= 0.5){
    x=mu+(log(2*U))/alpha
  }else{
    x=mu-(log(2-2*U))/alpha
  }
  return(x)
}

x_values <- c()
for (i in 1:10000){
  x_values[i] <-inverse_cdf(0,1)
  x_values
}

hist(x_values, breaks = 300, main = "Generating Random Numbers using Inverse CDF Method")
```



We can conclude from the histogram, that the implementation of our Non Uniform Random Number generators using inverse CDF method appears to be correct, at least when compared empirically against standard histogram for the laplace distribution

TASK 2.2

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 theacceptance / rejection procedure. What is the expected rejection rate ER and how close is it to R? Generate 2000 numbers from N(0,1) using standard rnorm() procedure, plot the histogram and compare the obtained two histograms.

Acceptance/rejection methods for generating realizations of a random variable X make use of realizations of another random variable Y whose probability density $g_y(x)$ is similar to the probability density of X, $f_x(x)$. The random variable Y is chosen so that we can easily generate realizations of it and so that its density $g_y(x)$ can be scaled to majorize $f_x(x)$ using some constant c; that is,

$$c * g_y(x) \geq f_x(x) \qquad for \quad all \quad x$$

Where, the density $g_y(x)$ is called the *majorizing* density or the *proposal* density, and $c * g_y(x)$ is called the majorizing function. The density of interest, $f_x(x)$, is called the *target* density.

Thus, our majorizing density $f_y(x) \sim DE(0, 1)$ is,

$$g_y(x) = \frac{1}{2} e^{-|x|}$$

And the target density $f_y(x) \sim N(0, 1)$ is,

$$f_y(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

To evaluate the majorizing constant c, we have,

$$c \geq \frac{f_x(x)}{g_y(x)}$$

Which is,

$$c \geq \sqrt{\frac{2}{\pi}} e^{-\frac{x^2}{2}+|x|}$$

By differentiating the above formula and setting to zero we got the maximum at x=1, Hence the c value will be,

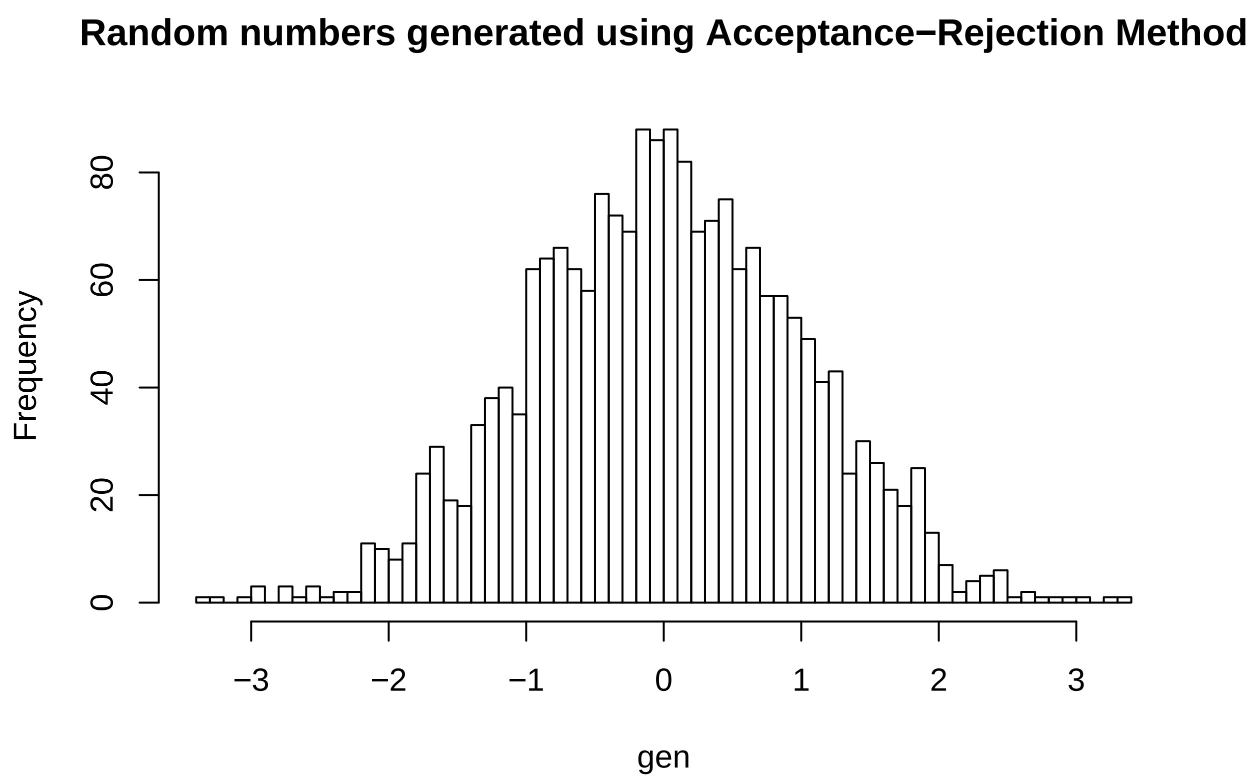
$$\sqrt{\frac{2}{\pi}} e^{\frac{1}{2}} = \sqrt{\frac{2e}{\pi}}$$

```
# TASK 2.2

accept_reject <- function(){

  X<-c()
```

```r
  counter= 0  # counter for how many numbers been rejected

   while (length(X)==0) { # as long as our vector is empty >> do
    counter = counter+1

  Y<-inverse_cdf(0,1)
  fy <- (1/sqrt(2*pi))*exp(-(1/2)*Y^2)

  Gy <- 0.5*exp(-abs(Y))

  c <- sqrt(2*exp(1)/pi)

  U = runif(1,0,1)

  if (U <= fy/(c*Gy)){
    X=Y
  }
  }
  gen_itr <- cbind("gen" = X, "itr" = counter)
  return(gen_itr)
}
```

```r
# generating 2000 random numbers using Acceptance\Rejection method
res <- list()
for (i in 1:2000){
  res[[i]]<- accept_reject()
}
```

```r
# excluding the random numbers from the list for plot
gen <- c()
for (i in 1:2000){
  gen[i]<- res[[i]][1]
}

# plotting the generated random numbers using N(0,1)
hist(gen , breaks = 50, main = "Random numbers generated using Acceptance-Rejection Method")
```



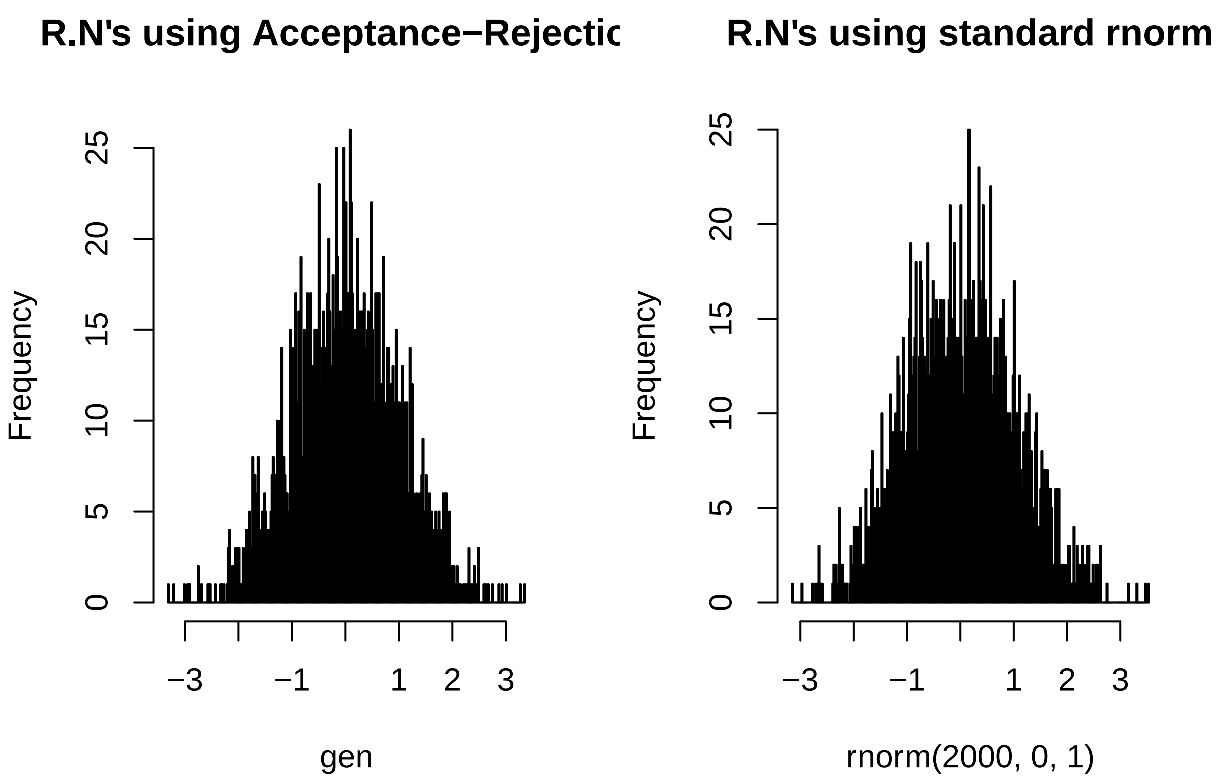
```r
# getting the total number of iterations (accepted+rejected) to commpute the average
itr <- c()
for (i in 1:2000){
  itr[i]<- res[[i]][2]
}

# average rejection rate
average_rej <-1-(2000/sum(itr))
average_rej
```

```
## [1] 0.2351816
```

```r
# expected rejection rate
c <- sqrt(2*exp(1)/pi)
expected_rej_rate<- 1-(1/c)
expected_rej_rate
```

```
## [1] 0.2398265
```

```r
# Comparing Two plots for the generated random numbers

# plotting the generated random numbers using Acceptance-Rejection Method
par(mfrow=c(1,2))
hist(gen , breaks = 300, main = "R.N's using Acceptance-Rejection")
# histogram for 2000 R.N's generated by standard rnorm()
hist(rnorm(2000,0,1), breaks = 300,main = "R.N's using standard rnorm")
```



The above two histograms for the Random Numbers using Acceptance/Rejection method and the exact normal distribution show similar pattern, that means our method is quite acceptable.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
# Loading required R packages
library(ggplot2)
library(kableExtra)
library(gridExtra)


# QUESTION 1 : Cluster sampling


# TASK 1.1

# Importing the file
data <- read.csv2("population.csv")


# TASK 1.2

# Function which uses a uniform random number generator to select a city from a list
city_selection <- function(data){
data$cumsum <- cumsum(data$Population)/sum(data$Population)
selected <- which(runif(1) <= data$cumsum)[1]
```

```
city_selected <- data$Municipality[selected]
as.numeric(rownames(data[selected,]))
}


# TASK 1.3

select <- data.frame()

# List of all cities stored in data1
data1 <- data

# Selecting 20 cities
for(i in 1:20){
  selected <- which(city_selection(data1) == as.numeric(rownames(data)))
  select <- rbind(select, data1[selected,])
  # Removing the selected city from the list
  data1 <- data1[-c(selected),]
}


# TASK 1.4

# Dataframe of 20 selected cities
cat("Selected cities : \n")
select$Municipality


# Selected cities along with their respective sizes
city_size = data.frame(City = select$Municipality, Size = select$Population)

kable(city_size, booktabs = T) %>%
kable_styling(latex_option = "striped")


# TASK 1.5

# Plot showing the size of all cities of the country
percentage<-data$Population/sum(data$Population)
barplot(percentage,names.arg = data$Municipality, xlab="Cities", ylab = "Size", ylim = c(0, 0.20))
title("Plot showing the size of all cities of the country")

# Plot showing the size of 20 selected cities
percentage_select<-select$Population/sum(select$Population)
barplot(percentage_select, names.arg = select$Municipality, xlab="Cities", ylab = "Size", ylim = c(0, 0
title("Plot showing the size of 20 selected cities")


# QUESTION 2 : Different distributions


# TASK 2.1
```

```r
inverse_cdf = function(mu,alpha){
  U = runif(1,0,1)
  if (U <= 0.5){
    x=mu+(log(2*U))/alpha
  }else{
    x=mu-(log(2-2*U))/alpha
  }
  return(x)
}

x_values <- c()
for (i in 1:10000){
  x_values[i] <-inverse_cdf(0,1)
  x_values
}

hist(x_values, breaks = 300, main = "Generating Random Numbers using Inverse CDF Method")


# TASK 2.2

accept_reject <- function(){

  X<-c()
  counter= 0  # counter for how many numbers been rejected

   while (length(X)==0) { # as long as our vector is empty >> do
    counter = counter+1

  Y<-inverse_cdf(0,1)
  fy <- (1/sqrt(2*pi))*exp(-(1/2)*Y^2)

  Gy <- 0.5*exp(-abs(Y))

  c <- sqrt(2*exp(1)/pi)

  U = runif(1,0,1)

  if (U <= fy/(c*Gy)){
    X=Y
  }
  }
  gen_itr <- cbind("gen" = X, "itr" = counter)
  return(gen_itr)
}

# generating 2000 random numbers using Acceptance\Rejection method
res <- list()
for (i in 1:2000){
  res[[i]]<- accept_reject()
}
# excluding the random numbers from the list for plot
gen <- c()
```

```r
for (i in 1:2000){
  gen[i]<- res[[i]][1]
}

# plotting the generated random numbers using N(0,1)
hist(gen , breaks = 50, main = "Random numbers generated using Acceptance-Rejection Method")

# getting the total number of iterations (accepted+rejected) to commpute the average
itr <- c()
for (i in 1:2000){
  itr[i]<- res[[i]][2]
}

# average rejection rate
average_rej <-1-(2000/sum(itr))
average_rej
# expected rejection rate
c <- sqrt(2*exp(1)/pi)
expected_rej_rate<- 1-(1/c)
expected_rej_rate
# Comparing Two plots for the generated random numbers

# plotting the generated random numbers using Acceptance-Rejection Method
par(mfrow=c(1,2))
hist(gen , breaks = 300, main = "R.N's using Acceptance-Rejection")
# histogram for 2000 R.N's generated by standard rnorm()
hist(rnorm(2000,0,1), breaks = 300,main = "R.N's using standard rnorm")
```