Computational Statistics Lab3

Vasileia Kampouraki, Weng Hang Wong, Jooyoung Lee

2020 2 13

Question 1: Cluster sampling

An opinion poll 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.

1. Import necessary information to R.

```
pop_data <- readxl::read_xls("~/Computational Stat/732A90_VT2020_Materials/population.xls",
          range="A6:C320") #Choosing only population
pop_data <- pop_data[-(1:3),] #deleting empty space
pop_data <- pop_data[(pop_data$Code>26),] #deleting counties
n <- dim(pop_data)[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 functionspresent in R).

```
city_selection <- function(data){
  U <- runif(1)
  cummulative <- 0
  prob <- data$Population/sum(data$Population)
  # Generating discrete RVs(lecture note)
  for (i in (1:n)){
    cummulative <- cummulative+prob[i]
    if ( cummulative >= U ) {
      return(i)
      } #returning the row index
  }
}
```

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.

```r
selections <- data.frame(City = as.character(), Population= as.numeric(), stringsAsFactors = FALSE)
data_use <- pop_data
for (j in 1:20){
  i <- city_selection(data=data_use)
  selections <- rbind(selections, data_use[i,c(2,3)]) #code not needed
  data_use = data_use[-i,]
}
names(selections) <- c("City","Population")
```

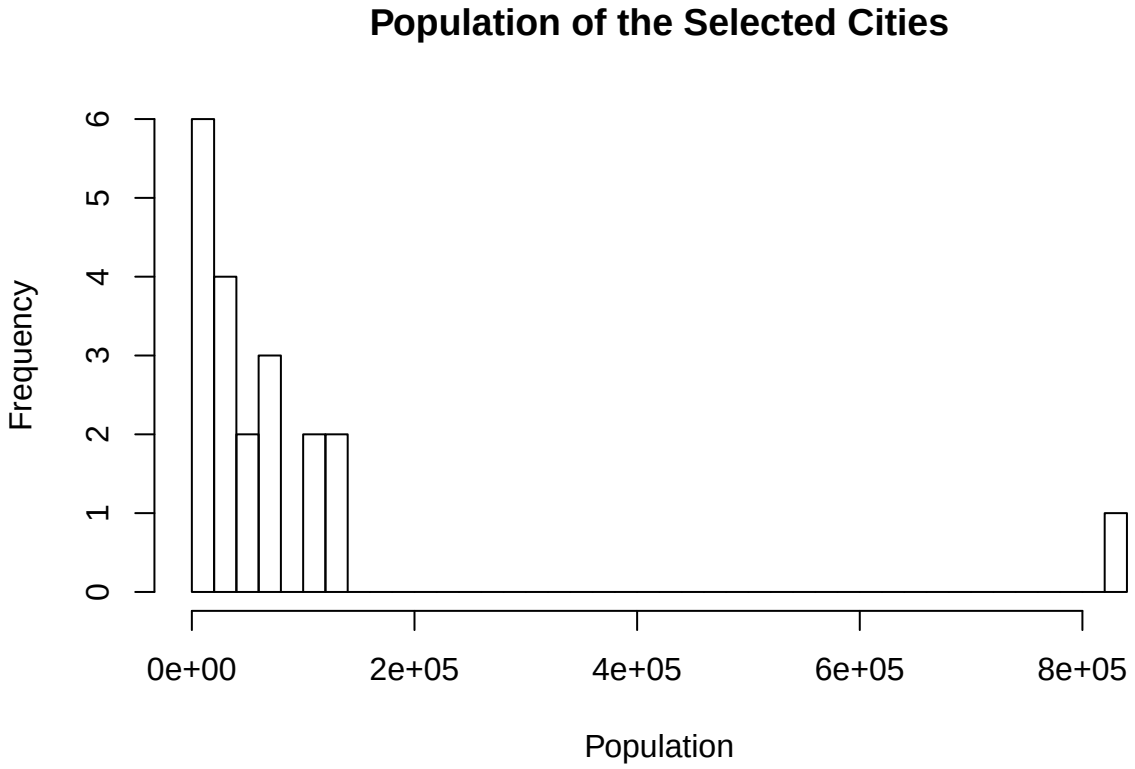
4. Run the program. Which cities were selected? What can you say about the size of the selected cities?

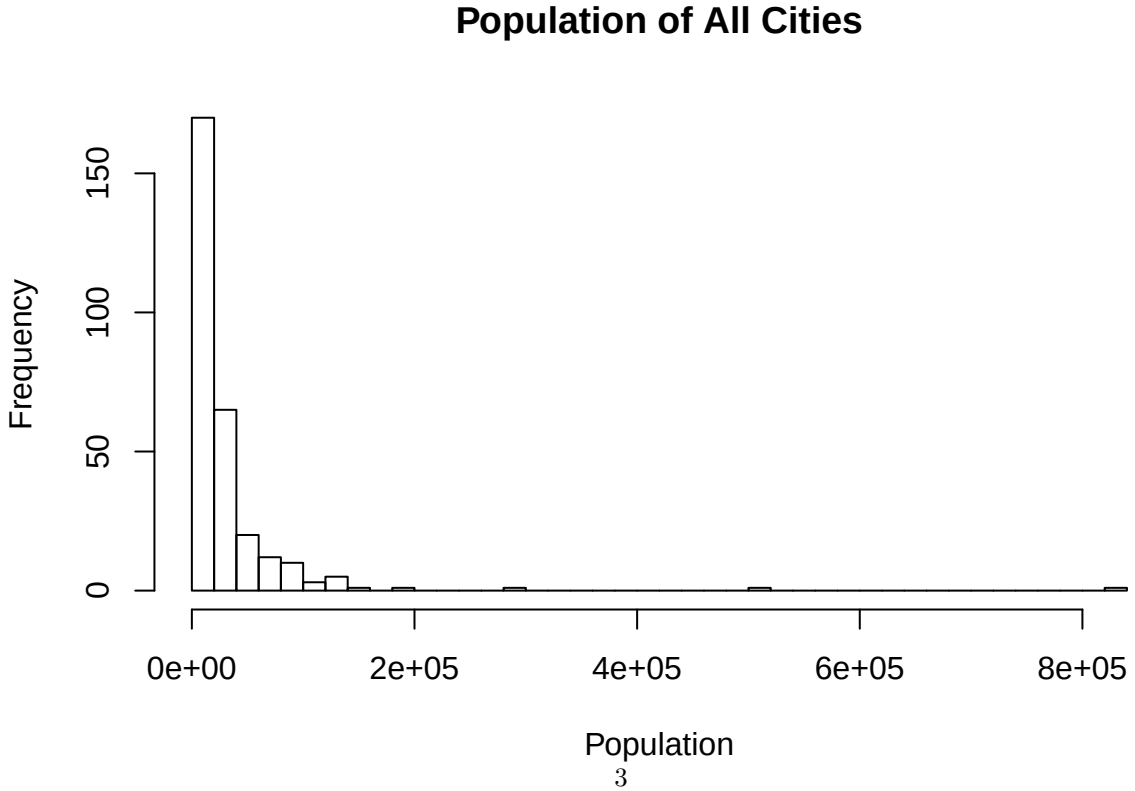
```
## List of 20 selected cities and their populations are:
```

```
## # A tibble: 20 x 2
##    City        Population
##    <chr>            <dbl>
##  1 Stockholm       829417
##  2 Vimmerby         15538
##  3 Fagersta         12249
##  4 Nykvarn           9227
##  5 Sollentuna       63347
##  6 Ale              27394
##  7 Bollebygd         8253
##  8 Umea            114075
##  9 Oskarshamn       26232
## 10 Stenungsund      23983
## 11 Borlange         48681
## 12 Tyreso           42602
## 13 Norrkoping      129254
## 14 Emmaboda          9223
## 15 Lund            109147
## 16 Flen             16139
## 17 Jonkoping       126331
## 18 Karlskrona       63342
## 19 Ekero            25095
## 20 Kalmar           62388
```

Majority of the populations in each city exceeds 20,000. Several of them has a population exceeding 200,000. Considering their population, selected cities are large cities.

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 first histogram which shows the population of the selected cities shows that majority of the population are less than 200,000 and there are only few cities that have population more than 200,000.

The second histogram which shows the populationof all cities in Sweden more clearly shows that most of the cities in Sweden have population less than 200,000 -furthermore, among these cities, majority of them have population less than 20,000- and very few cities have population more than that.

To conclude, selection of cities using uniform random number generation worked quite well in producting an opinion pool. The selection is based on the city population; the cities with large population are mostly chosen. Also, several cities having small population are also selected because such 'small-population-cities' are most of the cases.

Question 2: Different distributions

The double exponential (Laplace) distribution is given by formula:
$DE(\mu, \alpha) = \frac{a}{2} exp(-a|x - \mu|)$

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.

10000 random numbers from the double exponential distribution DE(0,1) will be generated using the inverse CDF method.

The steps of the process will be presented below.

1. The propability density function of ED(0,1) is:

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

and the Cumulative density function is:

For $\underline{x < 0:}$

$$F_x(x) = \int_{-\infty}^{x} \frac{1}{2} e^x = \frac{1}{2} e^x \Big|_{-\infty}^{x} = \frac{1}{2} e^x$$

For $\underline{x \geq 0:}$

$F_x(x) = \int_{-\infty}^{x} \frac{1}{2} e^{-x} = \int_{-\infty}^{0} \frac{1}{2} e^{-x} + \int_{0}^{x} \frac{1}{2} e^{-x} =$
$\frac{1}{2}(-e^x)\Big|_{-\infty}^{0} + \frac{1}{2}(-e^x)\Big|_{0}^{x} = \frac{1}{2} - \frac{1}{2} e^{-x} + \frac{1}{2} = 1 - \frac{1}{2} e^{-x}$

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

Now inverse CDF will be derived.

$\underline{x \geq 0:}$

$y = 1 - \frac{e^{-x}}{2} \Rightarrow e^{-x} = 2(1 - y) \Rightarrow$
$x = -ln2(1 - y),\ \frac{1}{2} < y < 1$

$\underline{x < 0:}$

$y = \frac{e^x}{2} \Rightarrow e^x = 2y \Rightarrow$
$x = ln2y,\ 0 < y < \frac{1}{2}$

Thus, the followings are obtained as a result:

$$F_x^{-1}(y) = \begin{cases} ln2y & 0 < y < \frac{1}{2} \\ -ln2(1-y) & \frac{1}{2} < y < 1 \end{cases}$$

Hence, if $U \sim U(0,1)$ then

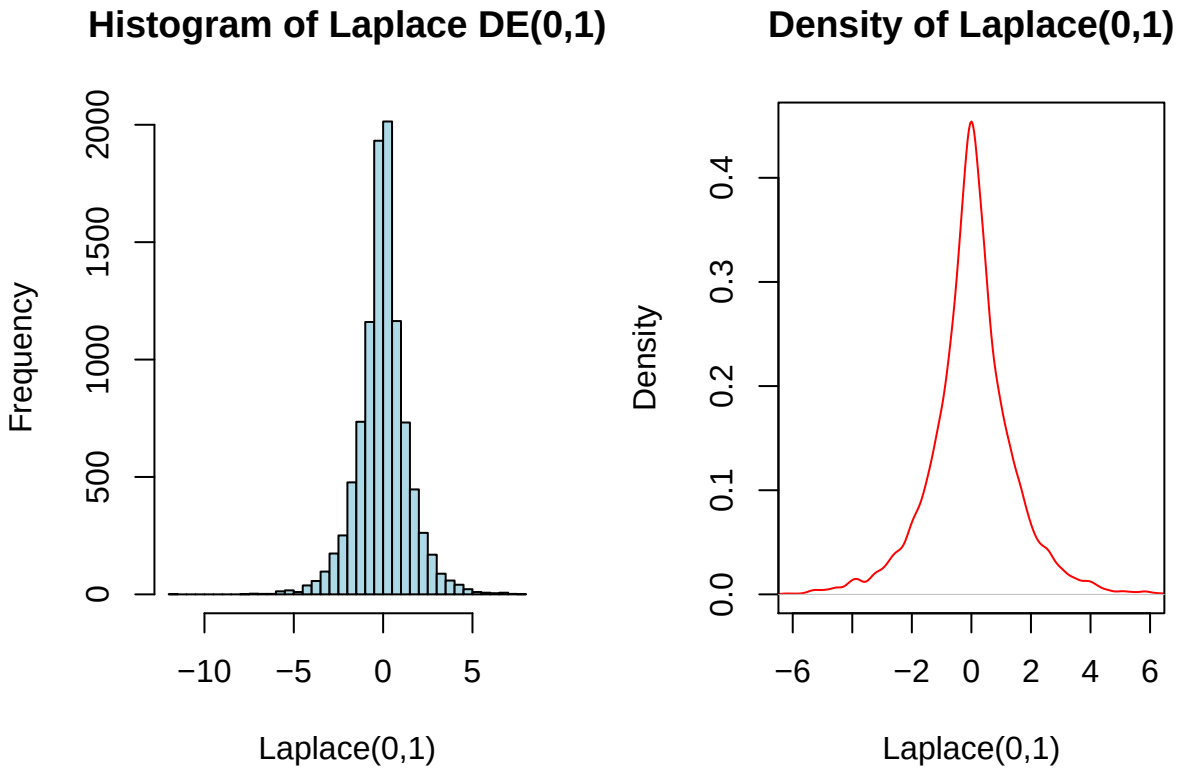
$$F_x^{-1}(U) = \begin{cases} ln2U & 0 < U < \frac{1}{2} \\ -ln2(1-U) & \frac{1}{2} < U < 1 \end{cases}$$

= X ~ E(0,1).

The algorithm for this simulation is presented as below:

Algorithm :

1) Generate U ~ U(0,1)
2) Find $F_x^{-1}(y)$
3) X = $F_x^{-1}(U) \sim E(0,1)$



To see if the histogram looks reasonable package *smoothmest* is used to plot the probability density function of the double exponential distribution DE(0,1) and compare it to the histogram. As shown on the plots above, the histogram of the 10000 random numbers generated from U(0,1) using the inverse CDF method looks distributed as DE(0,1) so the results obtained from simulation seem to be correct.

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 numbers from N(0,1) using standard rnorm() procedure, plot the histogram and compare the obtained two histograms.

Acceptance/rejection method

Let Y be a continuous random variable with pdf $f_Y(y)$ and values for another continuous random variable X with pdf $f_X(x)$ will be simulated - in this case $Y \sim DE(0,1)$ and $X \sim N(0,1)$.

So, the values of N(0,1) distribution will be simulated using DE(0,1) distribution with the Acceptance/Rejection method.

During the following simulation, values of the DE(0,1) will be simulated and then it will be decided whether to accept or not as values of X based on a criterion.

Let c be a constant, $\frac{f_X(y)}{f_Y(y)} \leq c, \forall y$

$$f_Y(y) = \frac{1}{2}e^{-|x|}, x\epsilon\Re$$

and

$$f_X(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}, x\epsilon\Re$$

Algorithm :

1) Produce a value of the random variable Y with pdf $f_Y(y)$
2) Produce a random number $U \sim U(0,1)$
3) If $U \leq \frac{f_X(y)}{cf_Y(y)}$ set X=Y else return to step 1.

2000 random numbers from N(0,1) will be generated; so the steps of the algorithm will be executed until 2000 numbers are obtained.

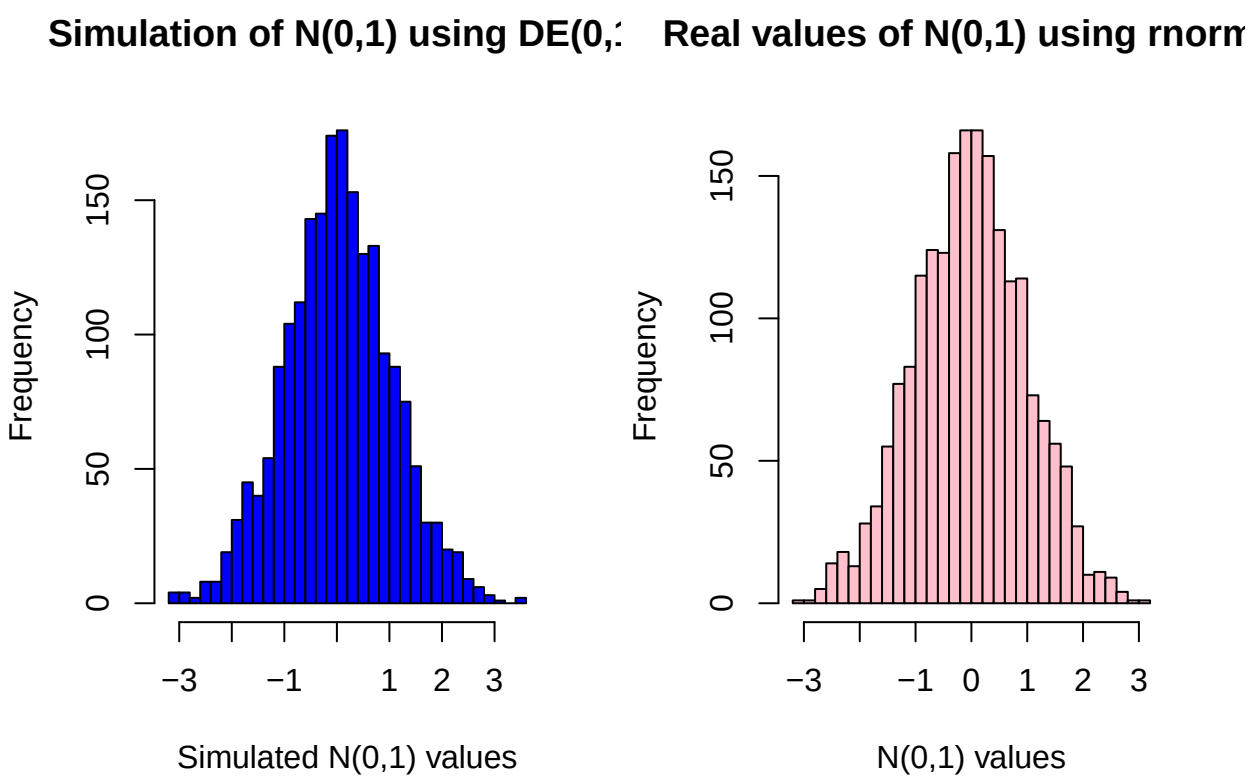
A value of X will be given for sure but it is also desirable to get it as faster as possible, i.e. have an efficient algorithm. This is achieved when c is small, because the larger the c the bigger the rejection rate.

To find c, the ratio of $\frac{f_X(y)}{f_Y(y)}$ will be derived and investigated for which value of x to get the maximum. Then the value of x which maximizes the ratio is substituted the random variable in the ratio and this will be the c.

Note: Laplace is symmetric around zero so on it will be considered as x>0 to simplify the derivation.

$$(\frac{f_X(y)}{f_Y(y)})' = (\frac{2}{\sqrt{2\pi}}e^{x-\frac{1}{2}x^2})' = \frac{2}{\sqrt{2\pi}}e^{x-\frac{1}{2}x^2}(-x+1) = 0 \Rightarrow (-x+1) = 0 \Rightarrow x = 1$$

By putting this x into the ratio, it is possible to get that $c = \frac{f_X(1)}{f_Y(1)} = \frac{2}{\sqrt{2\pi}}e^{1-\frac{1}{2}} = \frac{2}{\sqrt{2\pi}}e^{\frac{1}{2}} = 1.31$ (rounded in two decimals) which is the optimal value of c.



The histogram generated from the simulation of N(0,1) looks similar to the one obtained using the built-in function *rnorm()* which shows that the simulation worked well. The simulated N(0,1) histogram looks normally distributed as it has the characteristic bell curve.

The probability of having acceptance in one repetition of the algorithm is $\frac{1}{c}$.

Thus,the expected rejection rate will be $1 - \frac{1}{c}$.

```
## The rejection rate is
```

```
## [1] 0.2229992
```

```
## The expected rejection rate is
```

```
## [1] 0.2366412
```

The rejection rate obtained from the algorithm implemented (~ 22)% is very close to the expected rejection rate (~ 0.24)%.

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
pop_data <- readxl::read_xls("~/Computational Stat/732A90_VT2020_Materials/population.xls",
          range="A6:C320") #Choosing only population
pop_data <- pop_data[-(1:3),] #deleting empty space
pop_data <- pop_data[(pop_data$Code>26),] #deleting counties
n <- dim(pop_data)[1]
city_selection <- function(data){
  U <- runif(1)
  cummulative <- 0
  prob <- data$Population/sum(data$Population)
  # Generating discrete RVs(lecture note)
  for (i in (1:n)){
    cummulative <- cummulative+prob[i]
    if ( cummulative >= U ) {
      return(i)
      } #returning the row index
  }
}
selections <- data.frame(City = as.character(), Population= as.numeric(), stringsAsFactors = FALSE)
data_use <- pop_data
for (j in 1:20){
  i <- city_selection(data=data_use)
  selections <- rbind(selections, data_use[i,c(2,3)]) #code not needed
  data_use = data_use[-i,]
}
names(selections) <- c("City","Population")
cat("List of 20 selected cities and their populations are: ", "\n")
selections
hist(x=selections$Population, breaks = 40,
     main="Population of the Selected Cities", xlab="Population")
hist(x=pop_data$Population, breaks=40,
     main="Population of All Cities", xlab="Population")

  ## The first histogram which shows the population of the
  ## selected cities shows that majority of the population
  ## are less than 200,000 and there are only few cities that
  ## have population more than 200,000.

  ## The second histogram which shows the populationof all
  ## cities in sweden more clearly shows that most of the
  ## cities in Sweden have population less than 200,000
  ## (Furthermore, among these cities, majority of them have
  ## population less than 20,000), and very few cities have
  ## population more than that.

  ## To conclude, selection of cities using random number
  ## generation worked quite well in producting an opinion
  ## pool. The selection is based on the city population;
  ## the cities with large population are mostly chosen.
rglaplace <- function(n){

  U <- runif(n,0,1)
  X <- c()
```

```r
  for (i in 1:n){
    if (U[i]< 0.5){
     X[i] = log(2*U[i])
    } else {
      X[i] = - log(2*(1-U[i]))
    }
  }
  return(X)

}
#We will plot the density of E(0,1) to compare with the histogram
library(smoothmest)
#generate 10000 random numbers from Laplace(0,1)
rl <- rdoublex(10000,0,1)
d <- density(rl)
par(mfrow=c(1,2))
hist(rglaplace(10000),main = "Histogram of Laplace DE(0,1)",col="lightblue",breaks = 40,xlab = "Laplace
plot(d,main = "Density of Laplace(0,1)",xlim= c(-6,6),col="red",xlab = "Laplace(0,1)")
#we have x in real numbers
#range <- seq(-2,2,0.01)
#c <- max(ddoublex(range,0,1)/dnorm(range,0,1)) gives around 1.25 but ill use 1.31 from mathematical so
accept_reject <- function(c,n){
accepted <- 0
rejected <- 0
final <- c()
  while (accepted<n){   #in the end we want 2000 accepted
    Y <- rdoublex(1,0,1)
    U <- runif(1,0,1)

    if (U <= dnorm(Y,0,1)/ (c*ddoublex(Y,0,1))){
      accepted <- accepted + 1
      final[accepted] <- Y

    } else {
      rejected <- rejected + 1
    }
}

result <- list("Simulated values of X"=final,"Accepted"=accepted,"Rejected"=rejected)
}
values <- accept_reject(1.31,2000)
#histogram of simulated values
par(mfrow=c(1,2))
hist(values$`Simulated values of X`,breaks=30,main= "Simulation of N(0,1) using DE(0,1)",xlab="Simulate
hist(rnorm(2000,0,1),breaks=30,main="Real values of N(0,1) using rnorm()"
     ,xlab="N(0,1) values",col="pink")
#rejection rate
R <- values$Rejected / (values$Rejected + values$Accepted)
cat("The rejection rate is","\n")
R
#Expected rejection rate
c <- 1.31
ER <- 1- 1/c
```

```r
cat("The expected rejection rate is","\n")
ER
```