Computational Statistics Lab 3

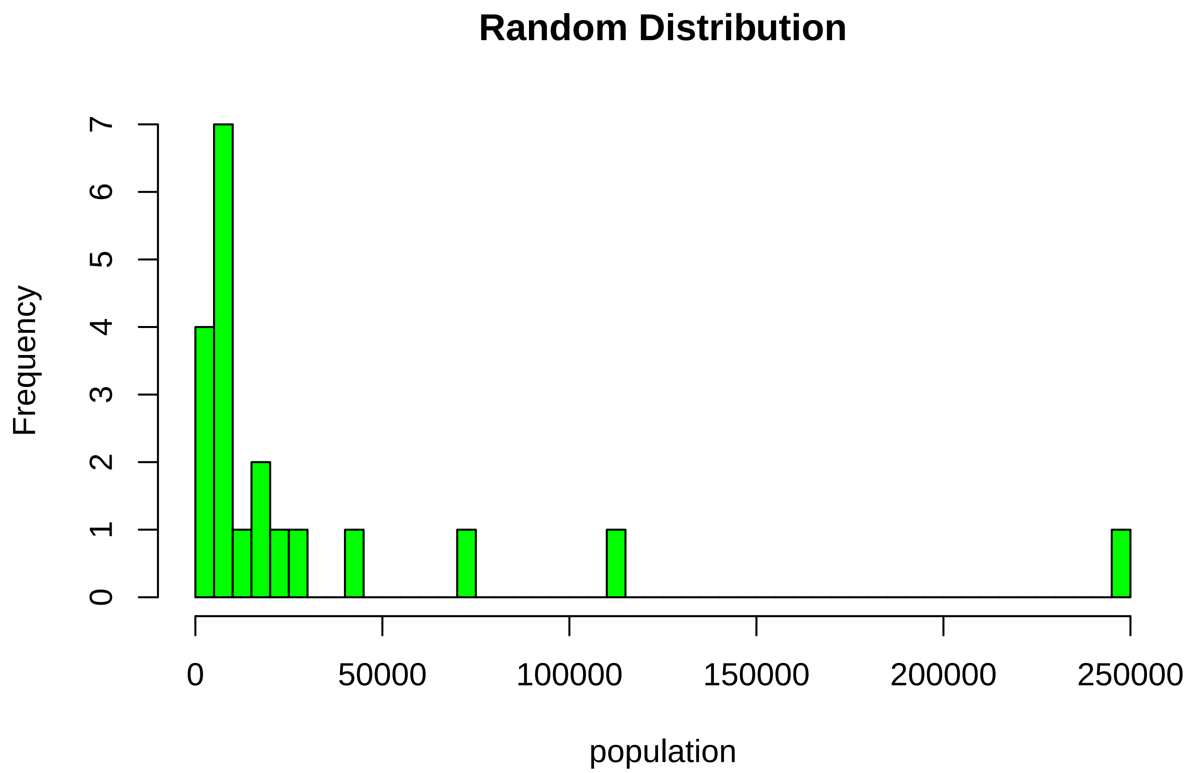
Gowtham & Biswas

#1 cluster sampling

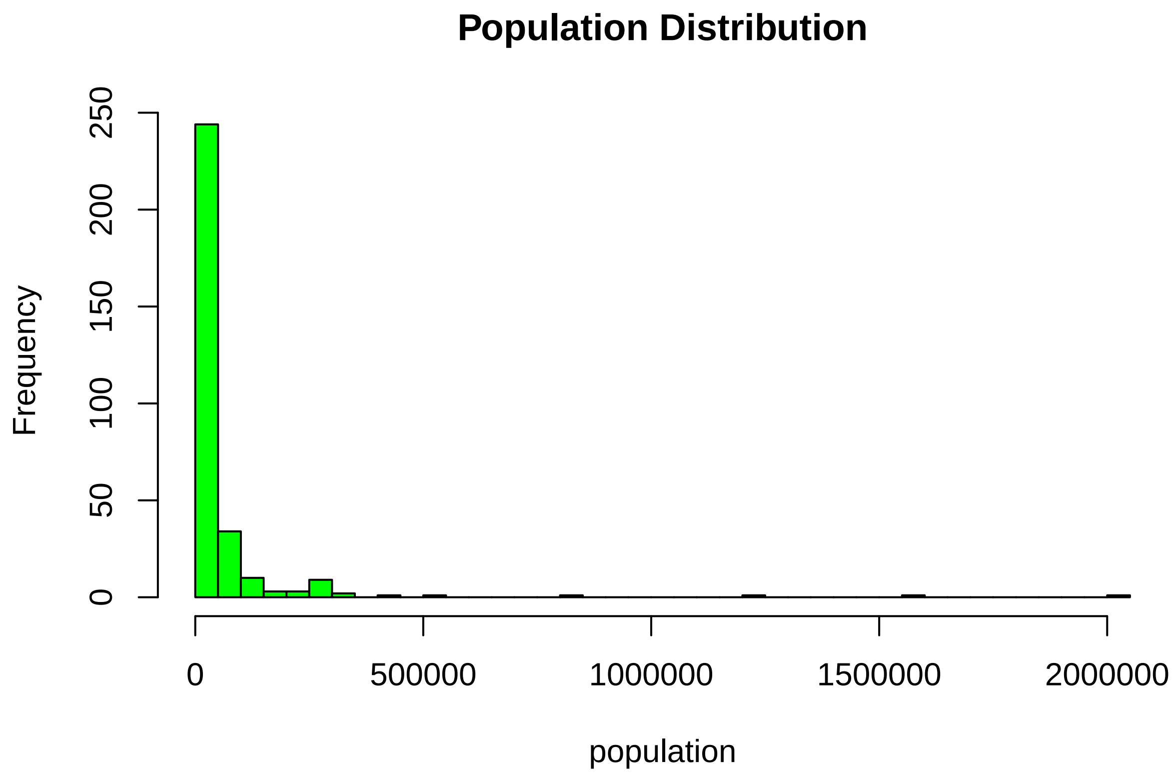
```
## New names:
## * `` -> ...2
## * `` -> ...3
## * `` -> ...4
## * `` -> ...5
## * `` -> ...6
## * ... and 14 more problems
```

```
## [1] "The selected cities and its population are :"
```

```
## # A tibble: 20 x 2
##    city       population
##    <chr>      <chr>
##  1 Umeå       114075
##  2 Vilhelmina 7156
##  3 Vindeln    5519
##  4 Vännäs     8357
##  5 Åsele      3133
##  6 Norrbotten 249019
##  7 Arjeplog   3143
##  8 Arvidsjaur 6622
##  9 Boden      27408
## 10 Gällivare  18533
## 11 Haparanda  10112
## 12 Jokkmokk   5210
## 13 Kalix      16926
## 14 Kiruna     22969
## 15 Luleå      73950
## 16 Pajala     6309
## 17 Piteå      40860
## 18 Älvsbyn    8387
## 19 Överkalix  3670
## 20 Övertorneå 4920
```



Random Distribution
Frequency
0
1
2
3
4
5
6
7
0
50000
100000
150000
200000
250000
population



The probabilitities of the cities is evaluated by : $Probability = \frac{Population_{city}}{TotalPopulation}$

We have selecetd and saved population and cities in a dataframe for the specific use amongst all data set.Then, as per instruction, we have created a function to randomly generate a city from the list of all cities based on the probability of the size of population. Higher the population, higher the weightage and porbabilty to be choosen.Coding wise,we are choosing the city that has lower probabilty than the generated random number.We then remove the selected city from th list of the cities, and run the function again till only 20 cities are retained.

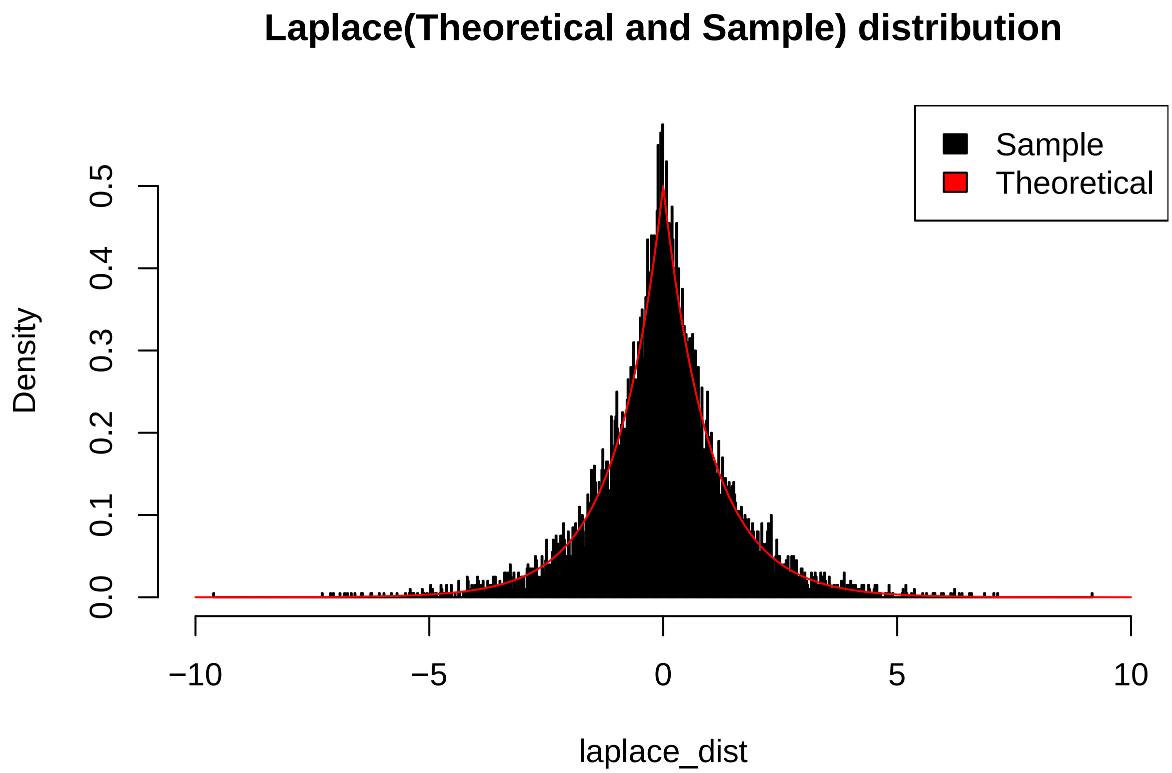
The list of the selected 20 cities are provided above. These selected cities have mixed size of population but more on the lower side.

Looking at the histogram(random distribution), we see that the frequency of cities being selected are weighted higher towards left side but cities with high population (more than 200k) has also been selecetd.

This indicates that the cities has been selected randomly. Also looking at the "population distribution" gives us the understanding that there are alot of cities with smaller population and hence the majority of random sample of 20 cities can be from such small cities.

It is therefore, we feel that cities with higher population should have been given a better weightage,to further justify the population based selection.

#2.1 Inverse CDF method



Looking at the above theoretical PDF of the laplace distribution and the sample laplace distribution for 10,000 sample, we find that sample is similar to the theoretical value, which is a good indication.

Laplace distribution is given by : $DE(\mu, \sigma) = \frac{\alpha}{2} exp(-\alpha \mid x - \mu \mid)$

Taking $\mu = 0, \alpha = 1$

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

$FX(x) = \int_{-\infty}^{\infty} f(x)\, dx$

$$= \begin{cases} 1 - \frac{1}{2}\int_{x}^{\infty} e^{-x} dt \\ \\ 1 - \frac{1}{2}\int_{-\infty}^{x} e^{x} \end{cases}$$

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

On solving F(x), $y = \frac{e^x}{2} \implies 2y = e^x \implies x = \ln 2y$

$y = 1 - \frac{e^{-x}}{2} \implies 2(1-y) = e^{-x} \implies x = -\ln(2-2y)$

Inverse CDF : When $x > \mu$ $\ln(2-2y) = x$ For $U \sim U(0,1), \ln(2-2y) = X$

when $x \leq \mu$,

For $U \sim U(0,1), \ln(2y) = X$

#2.2 Acceptance/Rejection method

Acceptance-Rejection method with DE(0,1) as majoring density to generate N(0,1) variables. Calculating constant C from given Laplace distribution ,

$Cf_y(x) \geq f_X(x)$ $c \geq \frac{fX(x)}{f_y(x)}$

where, f_y is majoring density, proposal density f_x is target density c is majoring constant

Generate $Y \sim f_y$, which is $F_x(u)$

Generate $U \sim unif(0,1), Uvalues$ are used in $F_x(U)$ which are f_y and f_x , check if X is accepted, end if and while loops.

This algorithm is tested by generating 2000 random numbers from N(0,1) using standard rnorm.

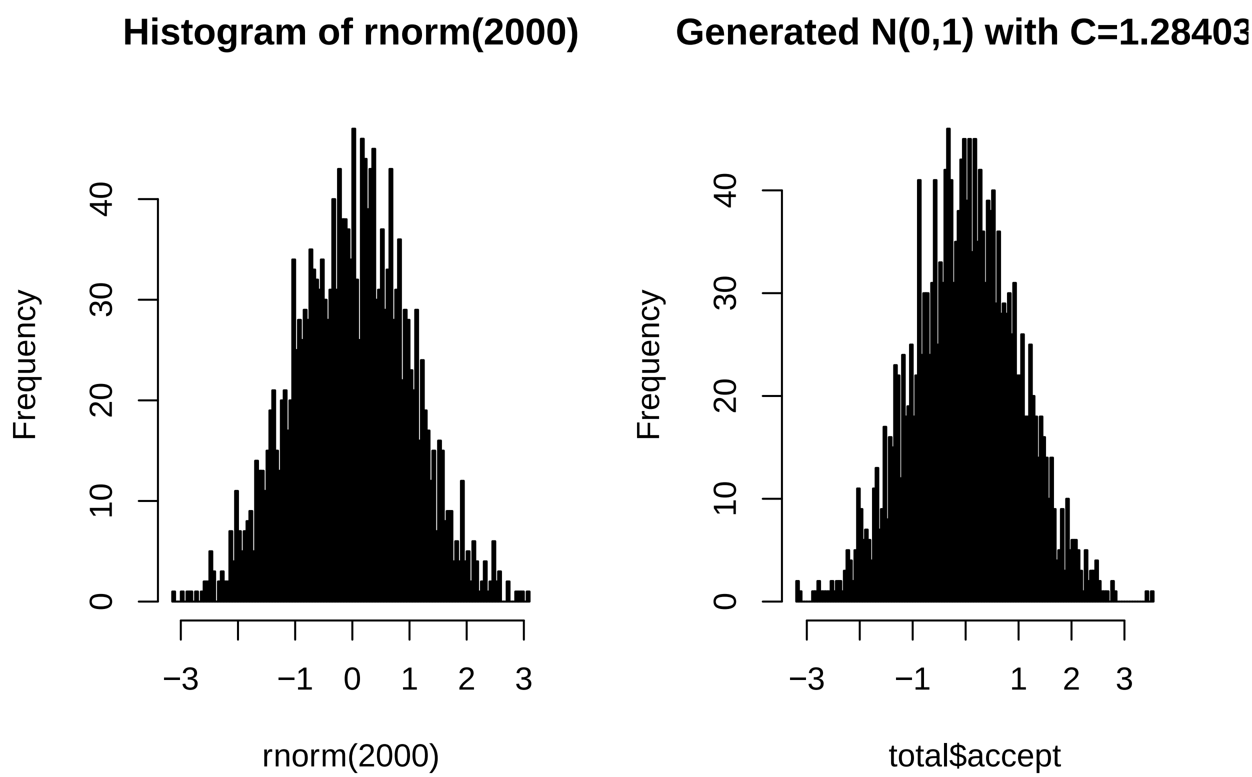
Number of draws $\sim \frac{1}{C}$.

$Rejection \sim Acceptance - \frac{1}{C}$

which gives $1 - \frac{1}{C} = 0.221$ Rejection rate)

```
## The value of C is 1.284037
```

```
## The theoretical Rejection rate is  0.221206
```




```
## Rejection Ratio 0.2208804
```

The expected Rejection ration (0.22088) is close to theoretical rejection ration(0.221206). # Code Appendix

```
knitr::opts_chunk$set(echo = FALSE)
library(readxl)
library(smoothmest)


#step1
RNGversion("3.6.2")
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")

data = read_excel("population.xls")
df = data[9:319,2:3] # extract counties and population since probability= population of each city/total
colnames(df) <- c("city","population")
population <- as.numeric(df$population)
probability <-population/sum(population)
df_original<-data[9:319,2:3]

#cumsum(probability)

#step2 & 3
find_city<-function(df){
    list_cities<-c()
    list_cities<-df$city[which(probability<runif(1))][1]
    return(list_cities)
}
#find_city(df)

while(length(df$city)!=20){
  temp_city<-find_city(df)
  if(!is.na(temp_city)){
  df=df[-which(df$city==temp_city),]}
  }
#step 4
print("The selected cities and its population are :")
print(df)

#step 5
hist(as.numeric(df$population),xlab="population",main="Random Distribution",col="green",breaks=50)
# original population distrution
#df_original
hist(as.numeric(df_original$...3),xlab="population",main="Population Distribution",col="green",breaks=5




y <- runif(10000,0,1)
inv_cdf <-function(y){
  laplace <-c()
  for (i in 1:length(y)) {
    if (y[i]>0.5){
      laplace[i] <- (log(2-2*y[i]))
    }
    else{
```

```
      laplace[i] <- -(log(2*y[i]))
    }
  }
  return(laplace)
}

#laplace_dist <-inv_cdf(runif(10000,0,1))
laplace_dist <- inv_cdf(y)
normal<-seq(from=-10,to=10,by=0.01)
fn<-exp(-abs(normal))/2
hist(laplace_dist, breaks = 1000,xlim=c(-10,10),probability = TRUE,main="Laplace(Theoretical and Sample)
points(fn~normal, type="l",col="red")
legend(x = "topright", legend = c("Sample","Theoretical"), fill = c("black","red"))

RNGversion("3.6.2")
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
# C value

normal_dist<-dnorm(normal,0,1)
# finding the difference between Laplace distribution and normal distribution
difference_dist <-normal_dist -fn
max_diff <- which.max(difference_dist)
C <- normal_dist[max_diff]/fn[max_diff]
cat("The value of C is",C,"\n")
# Rejection rate
RR<-1-(1/C)
cat("The theoretical Rejection rate is ",RR)

# Generating 2000 Random numbers
RNGversion("3.6.2")
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
# creating the function
ran<-function(C){
# initializing acceptance and rejection , and count
accept<-c()
count<-0
reject<-c()
while(length(accept)<2000){
y<-rdoublex(1)
run<-runif(1)
X<-dnorm(y)
Y<-ddoublex(y)
if(run<=X/(C*Y)){
       accept<-c(accept,y)}else{
       reject<-c(reject,y)}
count<-count+1
}
return(list(accept=accept,reject=reject,R.R=1-2000/count))
}
total<-ran(C)

par(mfrow=c(1,2))
hist(rnorm(2000),breaks=100,col="black")
```

```r
hist(total$accept,breaks=100,col="black",main="Generated N(0,1) with C=1.284037")
# Rejection Ratio
Rejection_ratio<-total$R.R
cat("Rejection Ratio",Rejection_ratio)
```