Computational Statistics (732A90) Lab3

Anubhav Dikshit(anudi287) and Thijs Quast(thiqu264)

23 January 2019

Contents

Collaborations

For this lab our classmates from group 16 explained the intuition behind question 2, and helped with the analytical derivation to solve question 2.

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.

1. Import necessary information to R.

```
data <- read.csv2("population.csv", header = TRUE)
```

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).

```
get_city <- function(data){

data$Municipality <- as.character(data$Municipality)
data$prob <- data$Population/sum(data$Population)
#sorting the dataset
data <- data %>% arrange(prob)

data$cum_prob <- cumsum(data$prob)
data$lead_cum_prob <- lead(data$cum_prob, n=1)

# filling NA
data$lag_cum_prob[1] <- 0
data$lead_cum_prob[NROW(data)] <- 1

set.seed(12345)
num <- runif(1,0,1)

X <- ifelse(((num >= data$cum_prob) & (num <=  data$lead_cum_prob)), row.names(data), NA)
X <- na.omit(X)

data_name <- data[row.names(data) %in% X,]
return(data_name$Municipality)
}
```

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.

```
data$Municipality <- as.character(data$Municipality)
# Select one city
get_city(data = data)
```

```
## [1] "Umeå"
```

```
# Remove one city
df <- data


df <- df[!df$Municipality %in% get_city(data=df),]

# apply function again
get_city(data = df)
```

```
## [1] "Lund"
```

```
# rempve this city
df <- df[!df$Municipality %in% get_city(data=df),]

# do this till 20 cities left
while(NROW(df) >20){
  df <- df[!df$Municipality %in% get_city(data=df),]
}
```

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

```
# final cities selected
df$Municipality
```

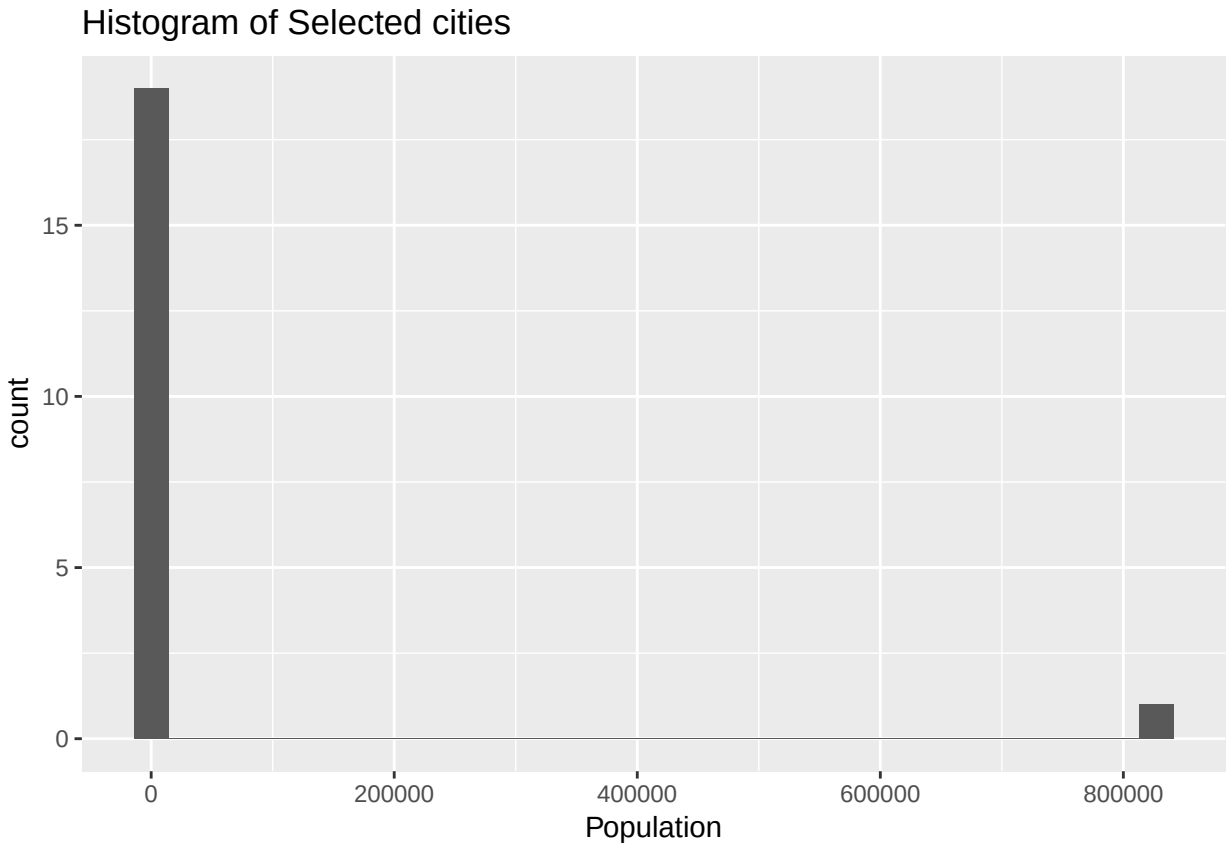
```
##  [1] "Stockholm"       "Boxholm"         "Ydre"
##  [4] "Ödeshög"         "Dals-Ed"         "Gullspång"
##  [7] "Munkfors"        "Storfors"        "Ljusnarsberg"
## [10] "Skinnskatteberg" "Bjurholm"        "Dorotea"
## [13] "Malå"            "Norsjö"          "Sorsele"
## [16] "Åsele"           "Arjeplog"        "Jokkmokk"
## [19] "Överkalix"       "Övertorneå"
```

```
data$Flag <- ifelse(data$Municipality %in% df$Municipality, "Selected", "Unselected")
```

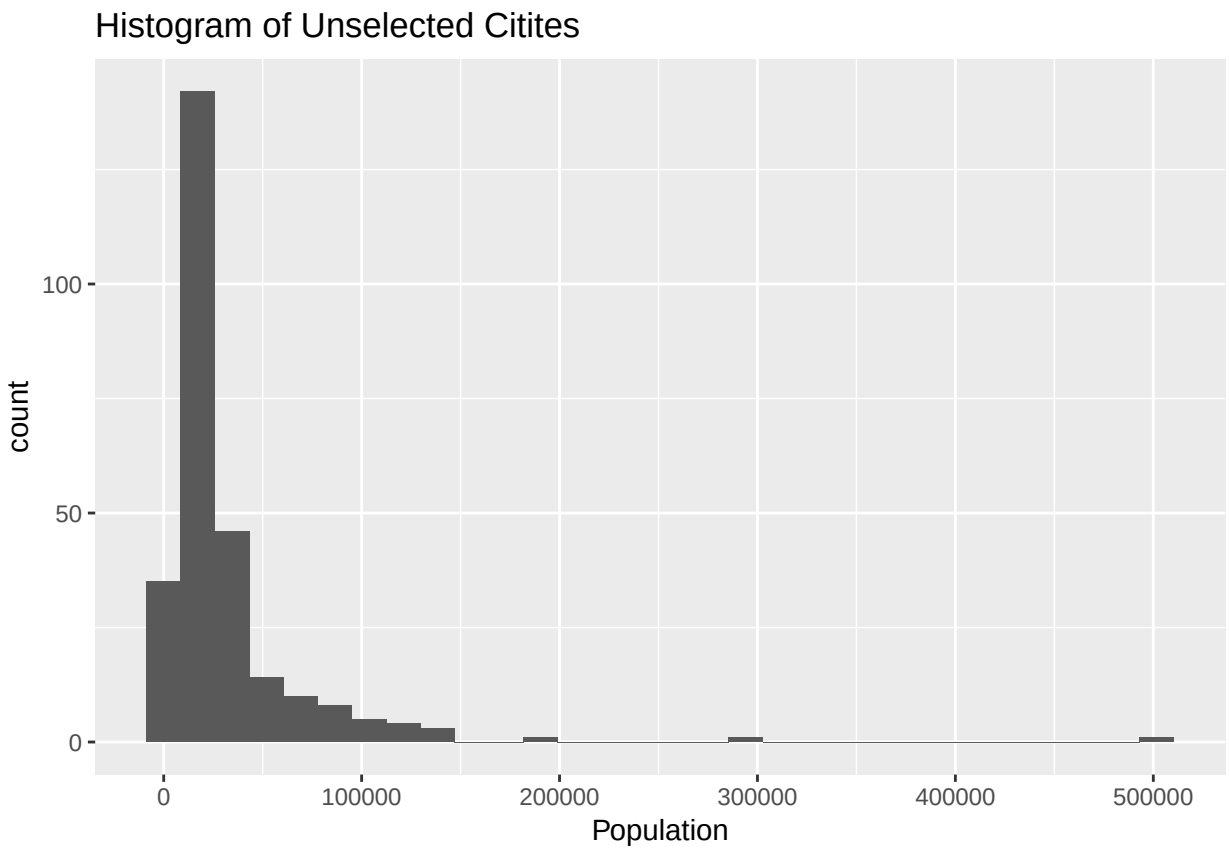
Analysis: As expected cities with large populations like Stockholm and Uppsala are selected. This is due that the probability of these cities being selected is larger because there population is larger.

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?

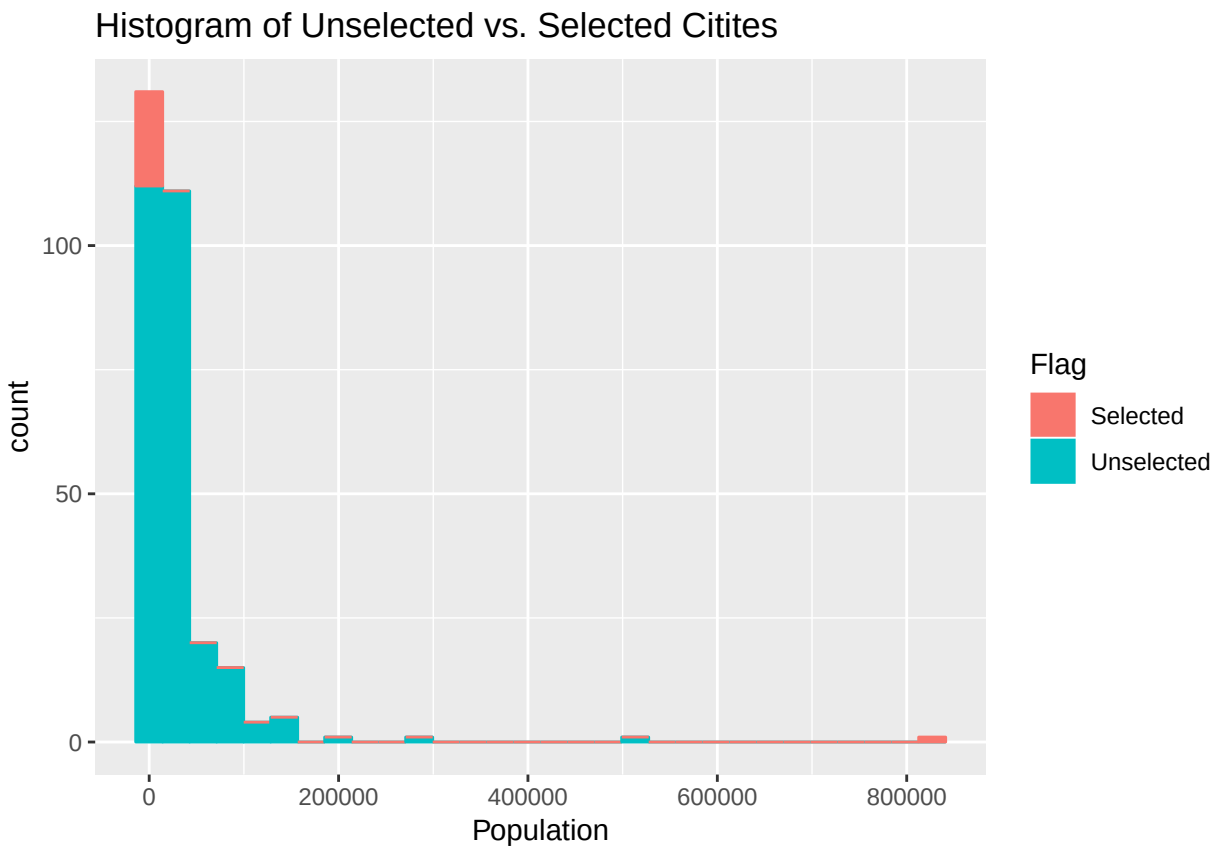
```
data %>% filter(Flag == "Selected") %>% ggplot(., aes(Population)) + geom_histogram(bins=30) + ggtitle(
```




```
data %>% filter(Flag == "Unselected") %>% ggplot(., aes(Population)) + geom_histogram(bins=30) + ggtitl
```




```
ggplot(data, aes(Population, fill=Flag, colour=Flag)) + geom_histogram(bins=30) + ggtitle("Histogram of
```



Analysis: The distribution of the populations of the sampled cities roughly seems to follow the same distribution as the full dataset.

Question 2: Different distributions

The double exponential (Laplace) distribution is given by formula:

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

The CDF is given by:

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

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

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

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

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

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

Inverse of CDF

$$For \quad x > \mu, \ we \ got \ F(x) = 1 - \frac{1}{2}e^{-\alpha(x-\mu)}$$

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

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

$$For \ U \sim U(0,1), \qquad \frac{ln(2-2U) - \alpha\mu}{-\alpha} = X$$

$$For \quad x \leq \mu, \ we \ got \ F(x) = \frac{1}{2}e^{\alpha(x-\mu)}$$

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

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

$$For \ U \sim U(0,1), \qquad \frac{ln(2U)}{\alpha} + \mu = X$$

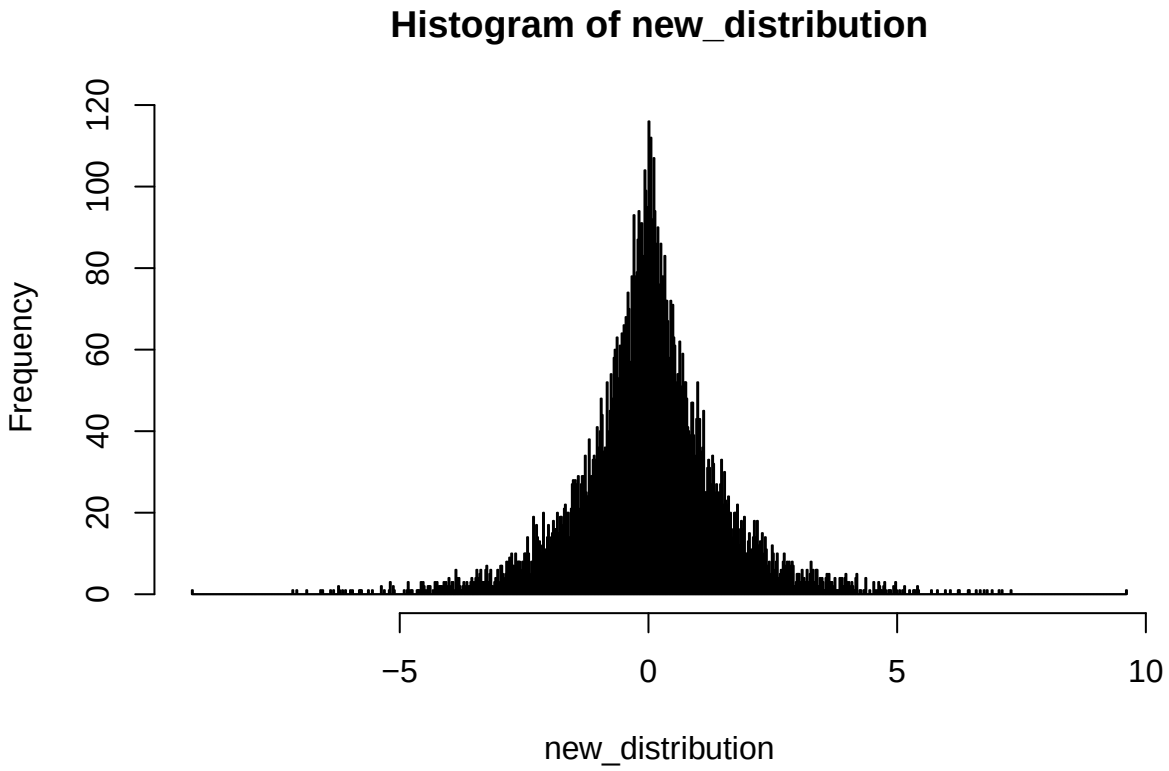
source: https://math.stackexchange.com/questions/2021342/how-is-this-inverse-function-calculated-laplace-distribution

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.

```
de_dist <- function(u, mu, a){
  de_distribution <- c()

    for (i in 1:length(u)){
      if (u[i] > 0.5){
        de_distribution[i] <- (log(2-2*u[i])-a*mu)/(-a)
      }
      else {
        de_distribution[i] <- (log(2*u[i])/a)+mu
      }
    }
  return(de_distribution)
}
```

```
new_distribution <- de_dist(runif(10000, 0, 1), mu=0, a=1)
hist(new_distribution, breaks = 1000)
```



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.

The Rejection Method: When we use the Inverse Transformation Method, we need a simple form of the cdf F(x) that allows direct computation of X= Inverse(F(U)). When F(x) doesn't have a simple form but the pdf f(x) is available, random variables with density f(x)can be generated by the rejection method. Suppose you have a method for generating a random variable having density function g(x). Now, assume you want to generate a random variable having density function f(x). Let c be a constant such that

$$\frac{f(y)}{g(y)} \leq c$$

Thus if we maximize the ratio f(y)/g(y) then that would be the value of c. This can be done by using parital derivate of the fraction.

$$Majorizing\ density\ F_Y(y) \sim DE(0,1) = \frac{1}{2}e^{-|x|}$$

$$Target\ density\ F_X(y) \sim N(0,1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

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

Differenitating with respect to x we get that the expression is maximum at x=1, thus C is:

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

$$= \frac{\sqrt{2}}{C\sqrt{\pi}} \cdot e^{-\frac{y^2}{2}+|y|}$$

$$= e^{-\frac{y^2}{2}+|y|+\frac{1}{2}}$$

Thus C is

$$C = \sqrt{(\frac{2 * e^1}{\pi})}$$

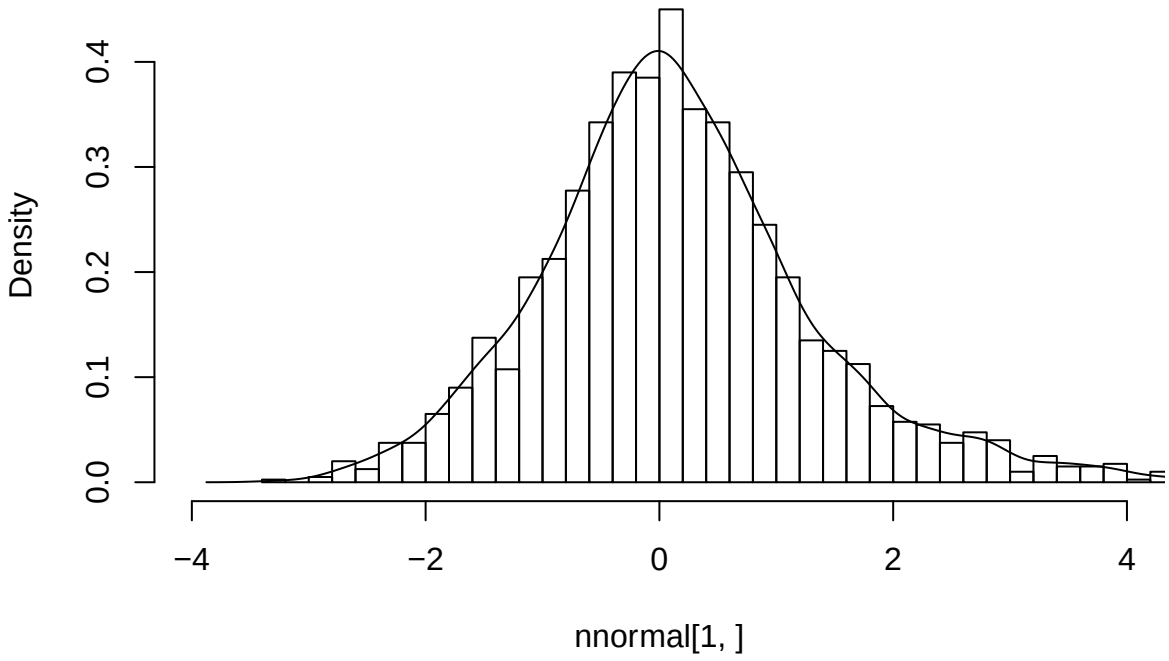
source: Introduction to Probability, Statistics, and Random Processes - Hossein Pishro-Nik

```
generate_n <- function(c){
  x <- NA
  num.reject <- 0
    while (is.na(x)){
      u <- runif(1, 0, 1)
      y <- de_dist(u, 0, 1)
      U <- runif(1)
        if (y>0 && U <= sqrt(2/pi)/c*exp(-(y^2)/2)+y){
          x <- y
        } else if (y<=0 && U<=sqrt(2/pi)/c*exp(-(y^2)/2-y)){
            x <- y
              } else {
                num.reject <- num.reject + 1
              }
    }
  c(x,num.reject)
}


c <- sqrt(2*exp(0.5)/pi)
```

```
set.seed(12345)
nnormal <- sapply(rep(c,2000), generate_n)
```

```
hist(nnormal[1,], breaks = 50, freq = FALSE, xlim = c(-4,4),
     main = "Normal distribution generated by Accept/Reject method")
lines(density(nnormal[1,]))
```




```
average_rejection <- sum(nnormal[2,])/(ncol(nnormal)+sum(nnormal[2,]))
average_rejection
```

```
## [1] 0.07278628
```

```
expected_rejection <- 1-(1/c)
expected_rejection
```

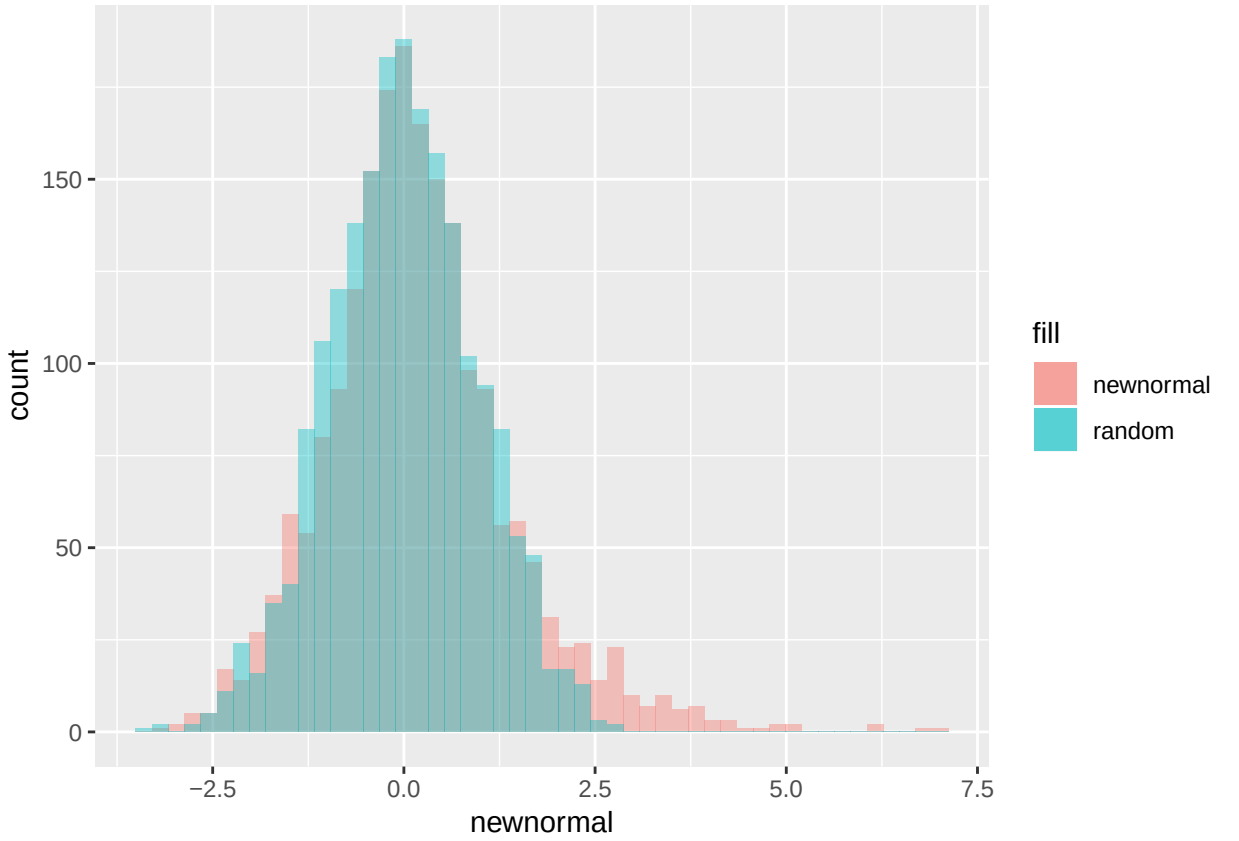
```
## [1] 0.02391797
```

```
rejection_difference <- expected_rejection - average_rejection
rejection_difference
```

```
## [1] -0.04886831
```

```
newnormal <- nnormal[1,]
random <- rnorm(2000, 0, 1)
df <- as.data.frame(cbind(newnormal, random))
```

```
ggplot(df, aes(newnormal, fill = "newnormal")) + geom_histogram(alpha = 0.4, bins = 50) +
  geom_histogram(aes(random, fill = "random"), alpha = 0.4, bins = 50)
```



Analysis: Both the samples distribution using accept/reject method and the actual distribution are close to each other. This means that the sampling method applied was decent.

The difference between the expected rejection rate and average rejection rate is around 8.5 percentage points, which is quite high.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
options(scipen=999)
library(dplyr)
library(ggplot2)
data <- read.csv2("population.csv", header = TRUE)
get_city <- function(data){

data$Municipality <- as.character(data$Municipality)
data$prob <- data$Population/sum(data$Population)
#sorting the dataset
data <- data %>% arrange(prob)

data$cum_prob <- cumsum(data$prob)
data$lead_cum_prob <- lead(data$cum_prob, n=1)

# filling NA
```

```r
data$lag_cum_prob[1] <- 0
data$lead_cum_prob[NROW(data)] <- 1

set.seed(12345)
num <- runif(1,0,1)

X <- ifelse(((num >= data$cum_prob) & (num <=  data$lead_cum_prob)), row.names(data), NA)
X <- na.omit(X)

data_name <- data[row.names(data) %in% X,]
return(data_name$Municipality)
}


data$Municipality <- as.character(data$Municipality)
# Select one city
get_city(data = data)

# Remove one city
df <- data


df <- df[!df$Municipality %in% get_city(data=df),]

# apply function again
get_city(data = df)

# rempve this city
df <- df[!df$Municipality %in% get_city(data=df),]

# do this till 20 cities left
while(NROW(df) >20){
  df <- df[!df$Municipality %in% get_city(data=df),]
}


# final cities selected
df$Municipality

data$Flag <- ifelse(data$Municipality %in% df$Municipality, "Selected", "Unselected")


data %>% filter(Flag == "Selected") %>% ggplot(., aes(Population)) + geom_histogram(bins=30) + ggtitle(

data %>% filter(Flag == "Unselected") %>% ggplot(., aes(Population)) + geom_histogram(bins=30) + ggtitl

ggplot(data, aes(Population, fill=Flag, colour=Flag)) + geom_histogram(bins=30) + ggtitle("Histogram of


de_dist <- function(u, mu, a){
  de_distribution <- c()

    for (i in 1:length(u)){
```

```
      if (u[i] > 0.5){
        de_distribution[i] <- (log(2-2*u[i])-a*mu)/(-a)
      }
      else {
        de_distribution[i] <- (log(2*u[i])/a)+mu
      }
    }
  return(de_distribution)
}
new_distribution <- de_dist(runif(10000, 0, 1), mu=0, a=1)
hist(new_distribution, breaks = 1000)

generate_n <- function(c){
  x <- NA
  num.reject <- 0
    while (is.na(x)){
      u <- runif(1, 0, 1)
      y <- de_dist(u, 0, 1)
      U <- runif(1)
        if (y>0 && U <= sqrt(2/pi)/c*exp(-(y^2)/2)+y){
          x <- y
        } else if (y<=0 && U<=sqrt(2/pi)/c*exp(-(y^2)/2-y)){
            x <- y
              } else {
                num.reject <- num.reject + 1
            }
    }
  c(x,num.reject)
}


c <- sqrt(2*exp(0.5)/pi)

set.seed(12345)
nnormal <- sapply(rep(c,2000), generate_n)
hist(nnormal[1,], breaks = 50, freq = FALSE, xlim = c(-4,4),
     main = "Normal distribution generated by Accept/Reject method")
lines(density(nnormal[1,]))
average_rejection <- sum(nnormal[2,])/(ncol(nnormal)+sum(nnormal[2,]))
average_rejection
expected_rejection <- 1-(1/c)
expected_rejection
rejection_difference <- expected_rejection - average_rejection
rejection_difference
newnormal <- nnormal[1,]
random <- rnorm(2000, 0, 1)
df <- as.data.frame(cbind(newnormal, random))

ggplot(df, aes(newnormal, fill = "newnormal")) + geom_histogram(alpha = 0.4, bins = 50) +
  geom_histogram(aes(random, fill = "random"), alpha = 0.4, bins = 50)
```