Computer Lab 3 Computational Statistics

Phillip Hölscher & Zijie Feng

1 2 2019

Contents

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.

```r
# load the data
data = read.csv2("population.csv", encoding = "latin1")
data[,1] <- as.character(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 functions present in R).

```r
# create the select city function - based on max
selectCity = function(data){
  # proportional to the number of inhabitants of the city
  data$proportinalPopulation = data$Population/sum(data$Population)
  # generate random numbers - include to dataset
  data$rn_uniform = runif(n=nrow(data),0,1)
  # probabilities proportional to the number of inhabitants
  data$prob_proportinalPopulation = data$proportinalPopulation * data$rn_uniform
  # take one city - max prob_proportinalPopulation
  city_index = which.max(data$prob_proportinalPopulation)
  # print(data[city_index,][1]) # make the print nicer

  return(data[city_index,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.

```
# create a function to select 20 cities
select20cities = function(data){
  cities_remain <- data
  selected_cities <- c()
  for (i in 1:20) {
  selected_city <- selectCity(cities_remain)
  selected_cities <- c(selected_cities,selected_city)
  cities_remain <- cities_remain[-which(cities_remain[,1]==selected_city),]
  }
  return(data.frame(Municipality=selected_cities))
}
```

4. Run the program.

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

```
# list of all selected cities
selected_cities <- select20cities(data)
idx <- match(selected_cities$Municipality, data[,1])
selected_cities$Population <- data[idx,2]
selected_cities <- selected_cities[order(selected_cities[,2],decreasing = TRUE),]
rownames(selected_cities) <- c()
```

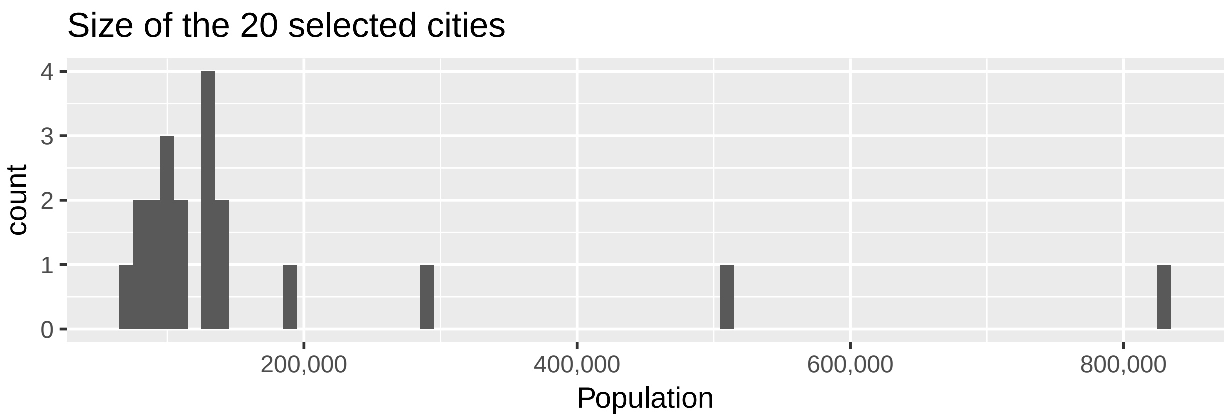
Municipality	Population
Stockholm	829417
Göteborg	507330
Malmö	293909
Uppsala	194751
Linköping	144690
Västerås	135936
Örebro	134006
Norrköping	129254
Helsingborg	128359
Jönköping	126331
Umeå	114075

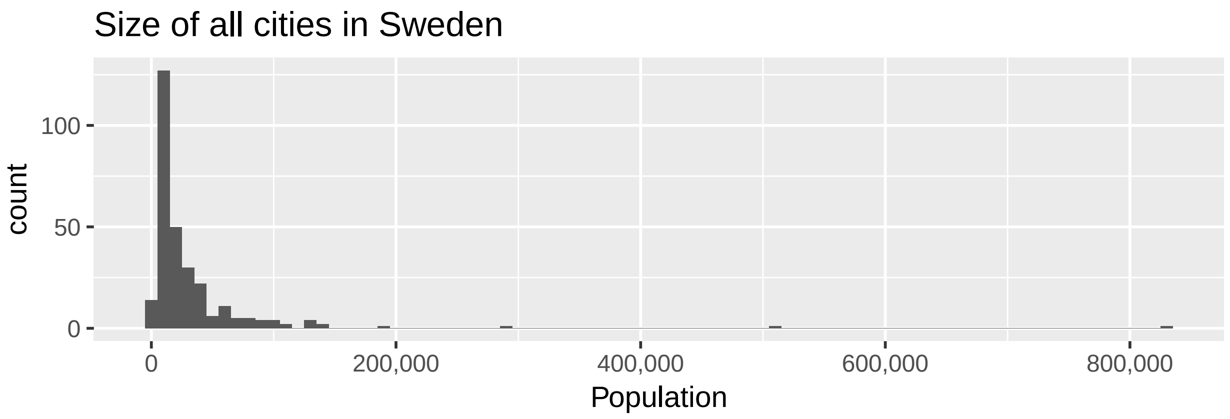
Municipality	Population
Lund	109147
Borås	102458
Eskilstuna	95577
Sundsvall	95533
Gävle	94352
Södertälje	85270
Karlstad	84736
Haninge	76237
Luleå	73950

The output above shows all the cities we selected, it is obvious that the cities with large population are quite easy to be selected, especially Stockholm, Goteborg, Malmo and Uppsala. This can be seen in a later visualization.

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 population of most of Swedish cities is around (0,100000), with relatively small size compared with other big cities. Most of the cities we randomly select belong to big-size cities with huge population ($\geq$ 100000), and only 7 cities' populations are smaller than 100000.

Question 2: Different distributions

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

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

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.

Firstly, we create a sequence of random numbers which are uniform distributed,

$$u \sim \text{Unif}(0, 1).$$

Since the cumulative distribution function (CDF) of Laplace distribution Y is

$$F_{DE}(x) = \frac{1}{2} + \frac{1}{2}\text{sgn}(\text{x} - \mu_{\text{DE}})[1 - \exp\left(-\alpha|\text{x} - \mu_{\text{DE}}|\right)] = \text{y}$$

and the deduction

$$P[F_Y^{-1}(u) \leq y] = P[u \leq F_Y(y)] = F_U[F_Y(y)] = F_Y(y),$$

We combine the duduction and u into the inverse CDF of DE(0,1), and we thereby get such formula as

$$F_{DE}^{-1}(u) = \mu_{DE} - \frac{\text{sgn}(\text{u} - \mu_{\text{Unif}})}{\alpha}\ln[1 + sgn(u - \mu_{\text{Unif}}) - sgn(u - \mu_{\text{Unif}})2u].$$

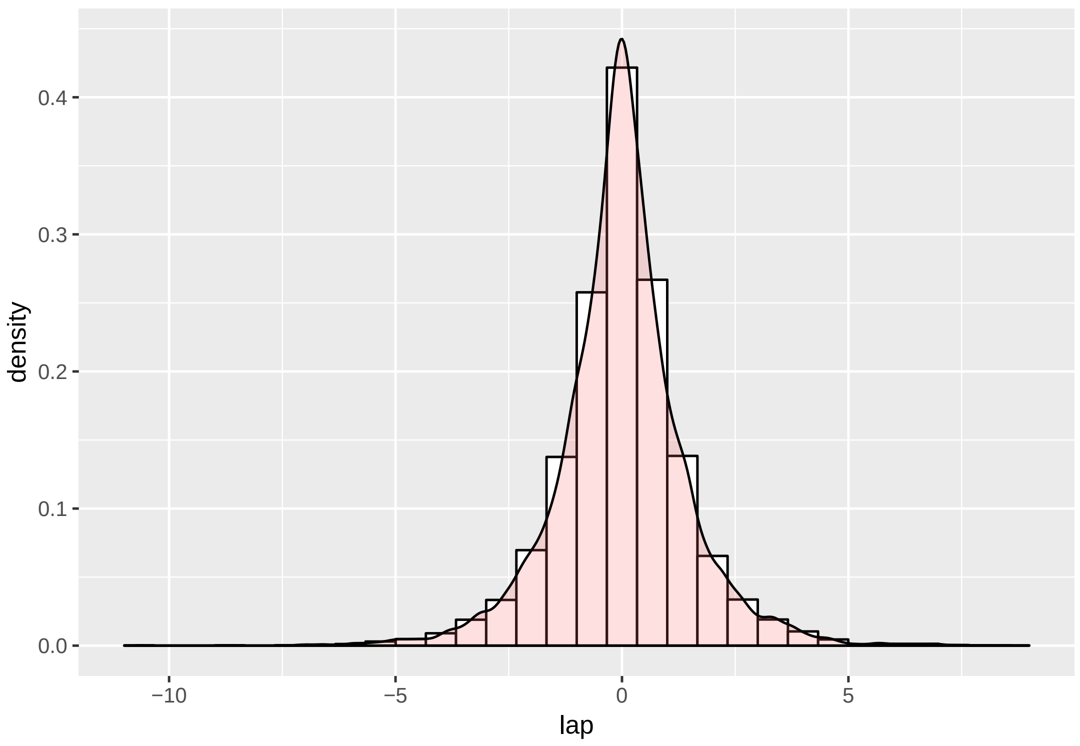
Substituting $\mu_{\text{Unif}} = 0.5$ into the formula,

$$Y = F_{DE}^{-1}(u) = \mu_{DE} - \frac{\text{sgn}(\text{u} - 0.5)}{\alpha}\ln(1 - 2|u - 0.5|).$$

Thus, we conclude that $Y \sim DE(0, 1)$ and thereby get a sequence y following Laplace distribution by the inverse CDF function of Laplace distribution.

```r
# generate random numbers
n=10000
set.seed(123456)
x_rand <- runif(n=n, min=0, max=1)
data1 <- data.frame(unif=x_rand)
```

```r
# the inverse laplace with mu = 0.5
laplace_distribution = function(mu, alpha, p){
  # result <- mu-(1/alpha)*sign(p-0.5)*log(1-2*abs(p-0.5))
  result <- mu-sign(p-0.5)*1/alpha*log(1+sign(p-0.5)-sign(p-0.5)*2*p)
  return(result)
}
```



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.

Firstly, we treat the sequene `data1$lap` as $Y \sim f_Y$. Then we find the probability density function (PDF) of Laplace distribution DE(0,1), which is

$$f_Y(x) = \frac{1}{2}\exp(- \mid x \mid).$$

Meanwhile, we find the PDF of normal distribution N(0,1),

$$f_X(x) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{x^2}{2}).$$

We want a c which is larger than f_X/f_Y. Since we have

$$\frac{f_X}{f_Y} = \sqrt{\frac{2}{\pi}}\exp(-\frac{x^2}{2} + |x|),$$

it will get its maximum when $x = \pm 1$. So

$$c = \max\frac{f_X}{f_Y} = \sqrt{\frac{2}{\pi}}\exp(\frac{1}{2}) \approx 1.315.$$

By creating a sequence $u \sim \text{Unif}(0,1)$, if it statisfies

$$u_i \leq \frac{f_X(Y_i)}{cf_Y(Y_i)},$$

```r
pdf_norm = function(x){
  exp(-(x**2)/2)/(sqrt(2*pi))
}
pdf_lap = function(x){
  (1/2)*exp(-abs(x))
}

c <- sqrt(2/pi)*exp(0.5)
n2 <- 2000
accept <- c()

for (i in 1:n) {
  if(length(accept)>=n2){
    break()
  }
  Y <- data1[i,2]
  u <- runif(1)
  g <- pdf_lap(Y)
  f <- pdf_norm(Y)

  if(u<=f/(c*g)){
    accept <- c(accept,Y)
  }
}
data2 <- data.frame(val=accept, id="esti")
reject_rate <- 1-n2/i
```

ejection rate R

```
## [1] 0.2595335
```

Rejection rate E(R):

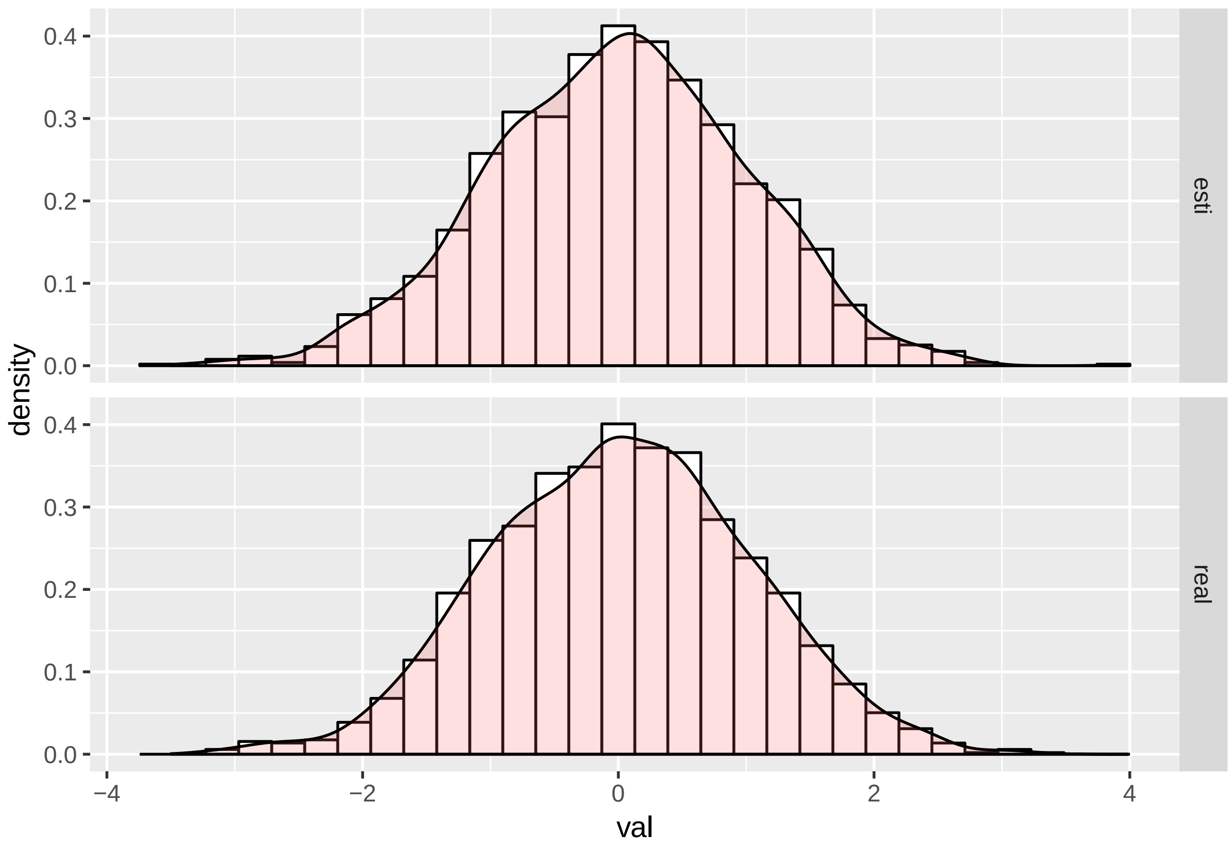
```r
1-1/c
```

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

The expected rejection rate should be $\mathrm{E(R)} = 1/\mathrm{c} \approx 0.24$. The real rejection rate R is around 0.25, which is also close to E(R). The following figures are based on our estimated normal distributed variables and samples from `rnorm()` respectively. It is right that such two figures like similar mutually.

```r
data3 <- data.frame(val=rnorm(n=n2,0,1),id ="real")
data3 <- rbind(data2,data3)

ggplot(data = data3, aes(x=val)) +
 geom_histogram(aes(y=..density..),
                colour="black",
                fill="white",
                bins=30)+
  geom_density(alpha=.2, fill="#FF6666")+
  facet_grid(id~.)
```


density
0.4
0.3
0.2
0.1
0.0
esti
real
−4
−2
0
2
4
val

Appendix

```r
knitr::opts_chunk$set(echo = TRUE, out.height = "200px")
#Sys.setlocale(locale="english")
# libraries used
rm(list=ls())
library(ggplot2)
#install.packages("gridExtra") # to put the plots togeter
library(gridExtra)

# set working directory
# use code for encoding
Sys.setlocale("LC_ALL", 'en_US.UTF-8')
# load the data
data = read.csv2("population.csv", encoding = "latin1")
data[,1] <- as.character(data[,1])
# create the select city function - based on max
selectCity = function(data){
  # proportional to the number of inhabitants of the city
  data$proportinalPopulation = data$Population/sum(data$Population)
  # generate random numbers - include to dataset
  data$rn_uniform = runif(n=nrow(data),0,1)
  # probabilities proportional to the number of inhabitants
  data$prob_proportinalPopulation = data$proportinalPopulation * data$rn_uniform
  # take one city - max prob_proportinalPopulation
  city_index = which.max(data$prob_proportinalPopulation)
  # print(data[city_index,][1]) # make the print nicer

  return(data[city_index,1])
}
# create a function to select 20 cities
select20cities = function(data){
  cities_remain <- data
  selected_cities <- c()
  for (i in 1:20) {
  selected_city <- selectCity(cities_remain)
  selected_cities <- c(selected_cities,selected_city)
  cities_remain <- cities_remain[-which(cities_remain[,1]==selected_city),]
  }
  return(data.frame(Municipality=selected_cities))
}
# list of all selected cities
selected_cities <- select20cities(data)
idx <- match(selected_cities$Municipality, data[,1])
selected_cities$Population <- data[idx,2]
selected_cities <- selected_cities[order(selected_cities[,2],decreasing = TRUE),]
rownames(selected_cities) <- c()
knitr::kable(selected_cities)

# histogram - 2 selected cities
histplot_20cities = ggplot(selected_cities)+
  geom_histogram(binwidth=10000,aes(x=Population)) +
  ggtitle("Size of the 20 selected cities") +
  scale_x_continuous(labels = scales::comma)

histplot_cities = ggplot(data)+
  geom_histogram(binwidth=10000,aes(x=Population)) +
  ggtitle("Size of all cities in Sweden")+
  scale_x_continuous(labels = scales::comma)
# put the plots together
grid.arrange(histplot_20cities, histplot_cities, ncol = 1)
# clean environment
rm(list=ls())
# generate random numbers
n=10000
set.seed(123456)
x_rand <- runif(n=n, min=0, max=1)
data1 <- data.frame(unif=x_rand)
ggplot(data1,aes(x=unif))+
   geom_histogram(aes(y=..density..),
                 colour="black",
                 fill="white",
                 bins=30)+
  geom_density(alpha=.2, fill="#FF6666")
# the inverse laplace with mu = 0.5
laplace_distribution = function(mu, alpha, p){
  # result <- mu-(1/alpha)*sign(p-0.5)*log(1-2*abs(p-0.5))
  result <- mu-sign(p-0.5)*1/alpha*log(1+sign(p-0.5)-sign(p-0.5)*2*p)
  return(result)
}

data1$lap <- laplace_distribution(0,1, x_rand)

ggplot(data = data1, aes(x=lap)) +
 geom_histogram(aes(y=..density..),
                 colour="black",
                 fill="white",
                 bins=30)+
  geom_density(alpha=.2, fill="#FF6666")
pdf_norm = function(x){
  exp(-(x**2)/2)/(sqrt(2*pi))
}
pdf_lap = function(x){
  (1/2)*exp(-abs(x))
}

c <- sqrt(2/pi)*exp(0.5)
n2 <- 2000
accept <- c()

for (i in 1:n) {
  if(length(accept)>=n2){
    break()
  }
  Y <- data1[i,2]
  u <- runif(1)
```

```r
  g <- pdf_lap(Y)
  f <- pdf_norm(Y)

  if(u<=f/(c*g)){
    accept <- c(accept,Y)
  }
}
data2 <- data.frame(val=accept, id="esti")
reject_rate <- 1-n2/i
reject_rate
1-1/c
data3 <- data.frame(val=rnorm(n=n2,0,1),id ="real")
data3 <- rbind(data2,data3)

ggplot(data = data3, aes(x=val)) +
 geom_histogram(aes(y=..density..),
                 colour="black",
                 fill="white",
                 bins=30)+
  geom_density(alpha=.2, fill="#FF6666")+
  facet_grid(id~.)
```