

Computer Lab 3 Computational Statistics

Phillip Hölscher & Zijie Feng

12 2019

Contents

Question 1: Cluster sampling	1
1. Import necessary information to R.	1
2. Use a uniform random number generator	1
3. Use the function you have created in step 2	2
4. Run the program.	2
5. Plot one histogram showing the size of all cities of the country.	3
Question 2: Different distributions	4
1. Write a code generating double exponential distribution $DE(0, 1)$	4
2. Use the Acceptance/rejection method	6

Question 1: Cluster sampling

An opinion pool is assumed to be performed in several locations of Sweden by sending inter-viewers 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.

```
# set working directory
# load the data
data = read.csv2("population.csv")
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).

```
# create the select city function - based on max
selectCity = function(data){
  # proportional to the number of inhabitants of the city
  data$proportionalPopulation = data$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$prob_proportionalPopulation = data$proportionalPopulation * data$rn_uniform
  # take one city - max prob_proportionalPopulation
  city_index = which.max(data$prob_proportionalPopulation)
```

```

# 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()
selected_cities

```

##	Municipality	Population
## 1	Stockholm	829417
## 2	Göteborg	507330
## 3	Malmö	293909
## 4	Uppsala	194751
## 5	Linköping	144690
## 6	Västerås	135936
## 7	Örebro	134006
## 8	Norrköping	129254
## 9	Helsingborg	128359
## 10	Jönköping	126331
## 11	Umeå	114075
## 12	Lund	109147
## 13	Borås	102458
## 14	Huddinge	95798
## 15	Eskilstuna	95577
## 16	Sundsvall	95533
## 17	Halmstad	91087

```
## 18      Nacka      88085
## 19  Södertälje    85270
## 20    Botkyrka    81195
```

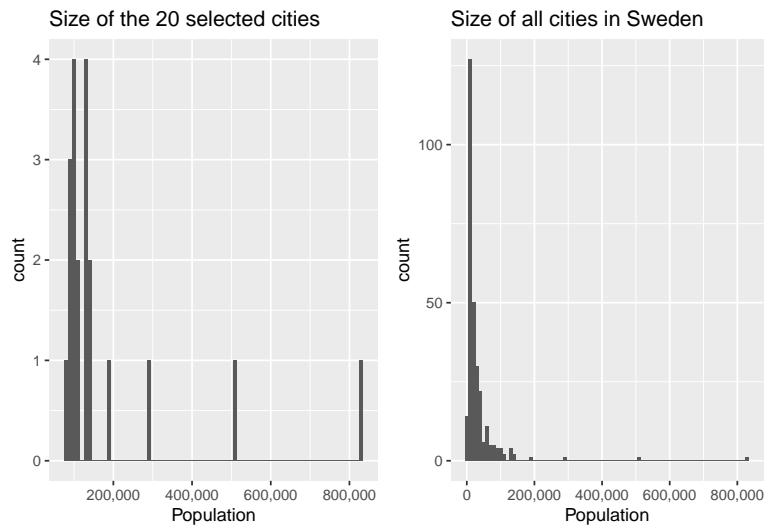
5. Plot one histogram showing the size of all cities of the country.

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

```
# 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 = 2)
```



Question 2: Different distributions

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

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

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.

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 is

$$F(u) = \frac{1}{2} + \frac{1}{2} \text{sgn}(u - \mu) [1 - \exp(-\alpha |u - \mu|)], \quad u > 0$$

and the deduction

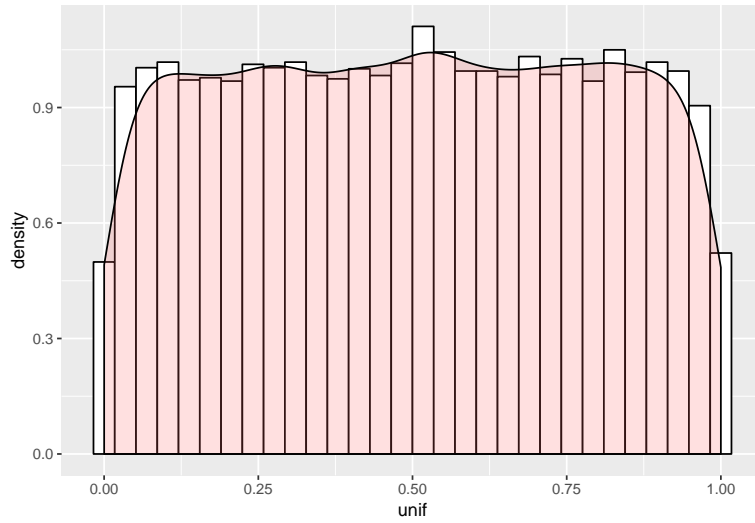
$$P(X < y) = P(F^{-1}(u) < y) = p(u < F(y)) = F(y),$$

we can assume that X satisfies

$$X = F^{-1}(u) = \mu - \frac{\text{sgn}(u - 0.5)}{\alpha} \ln(1 - 2|u - 0.5|).$$

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

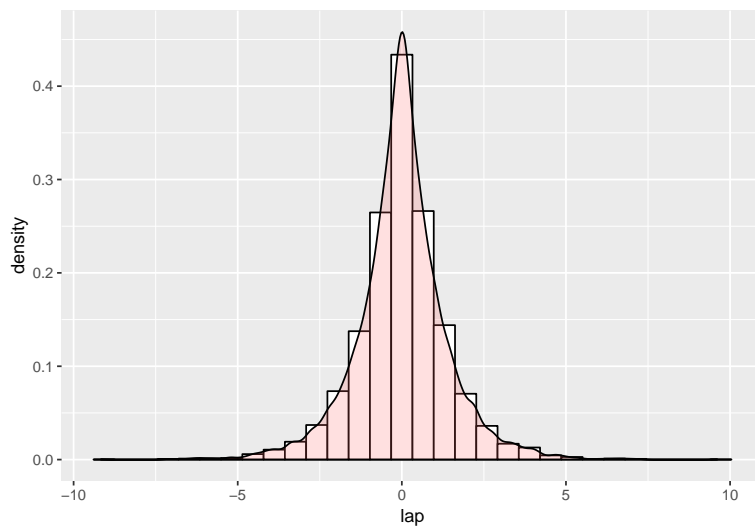
```
# generate random numbers
rm(list=ls())
n=10000
set.seed(12345)
x_rand <- runif(n=n, min=0, max=1)
data1 <- data.frame(unif=x_rand)
ggplot(data1, aes(x=unif)) +
  geom_histogram(aes(y=..density..),
                 colour="black",
                 fill="white",
                 bins=30) +
  geom_density(alpha=.2, fill="#FF6666")
```



```
laplace_distribution = function(mu, alpha, p){
  result <- mu-(1/alpha)*sign(p-0.5)*log(1-2*abs(p-0.5))
  # result = alpha/2 * exp(-alpha * (abs(x_rand - mu)))
  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")
```



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

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

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 can let f_Y cover f_X . 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 treating `data1$unif` as u , if it satisfies

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

we would accept that Y_i follows $N(0,1)$.

```
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()
  }
  u <- data1[i,1]
  Y <- data1[i,2]
  q <- pdf_lap(Y)
  p <- pdf_norm(Y)
  if(u<=p/(c*q)){
    accept <- c(accept,Y)
  }
}
```

```

}
data2 <- data.frame(val=accept, id="esti")
# ggplot(data = data2, aes(x=val)) +
#   geom_histogram(aes(y=..density..),
#     colour="black",
#     fill="white",
#     bins=30)+
#   geom_density(alpha=.2, fill="#FF6666")

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~.)

```

