Lab 3 - Computational Statistics

Gustav Sternelöv

February, 18, 2016

Assignment 1

1.1-1.2

The code below shows the function written for selecting one city from the list of all cities. The probability for the respective city to be chosen is connected to the number of inhabitants in the city. Returned by the function is a number, *cityNum*, which is the row number for the selected city.

```r
cityFunc <- function(data){
  data$prob <- data$Population / sum(data$Population)
  data$cumProbs <- cumsum(data$prob)
  unifV <- runif(1,0,1)
  dataNew <- subset(data, data$cumProbs >= unifV)
  cityNum <- which(data$cumProbs==min(dataNew$cumProbs))
  return(cityNum)
}
```

1.3-1.4

The following program is used for selecting 20 cities from the list. After each evaluation of the function is one city selected and this city is added to the new data frame *newData*. In the next step is the selected city erased from the original list of cities so it cannot be chosen two times. This procedure is repeated until the new data frame contains 20 cities.

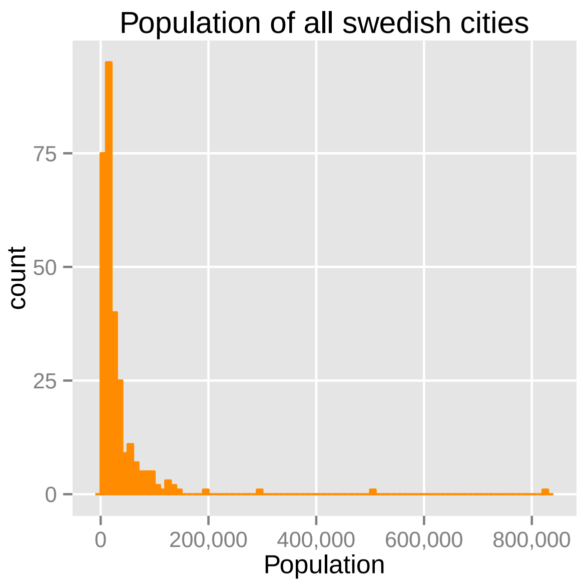
```r
selectData <- data1
newData <- as.data.frame(matrix(seq(20),nrow=20,ncol=3))
for(i in 1:20){
  set.seed(i)
  cityNum <- cityFunc(selectData)
  newData[i, ] <- selectData[cityNum, 1:3]
  selectData <- selectData[-cityNum, ]
}
```

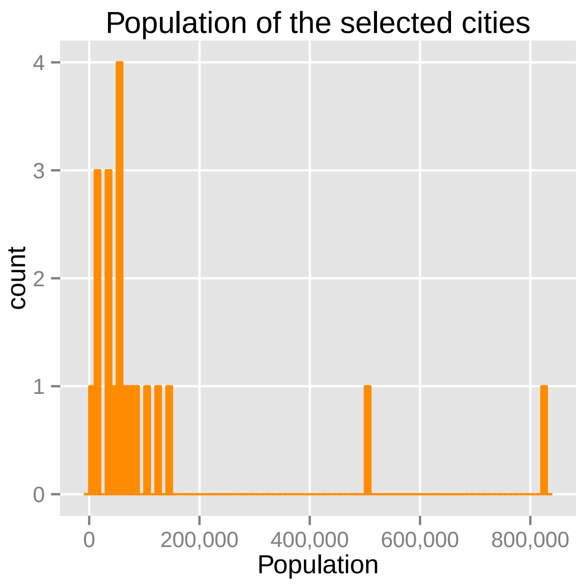
The names of the selected cities and the number of inhabitants in each city is presented in the table below. Among the cities that has been selected it can be noted that both Stockholm and Gothenburg, the two most populated swedish cities, are included. Of the ten cities with highest population are four included in the list. In general do the selected citites have a rather high number of inhabitants with a median population of 53601.5. The median population for all swedish cities is 15282, so the median population is 3.5 times as high for the selected cities.

```
##           City Population
## 1  Katrineholm      32303
## 2   Södertälje      85270
## 3    Stockholm     829417
## 4     Göteborg     507330
## 5     Nyköping      51209
## 6    Falköping      31419
## 7        Luleå      73950
## 8         Lund     109147
## 9    Linköping     144690
## 10     Svedala      19625
## 11   Jönköping     126331
## 12   Norrtälje      55927
## 13    Forshaga      11401
## 14     Ödeshög       5314
## 15     Kungälv      40727
## 16   Uddevalla      51518
## 17        Håbo      19452
## 18       Falun      55685
## 19        Täby      63014
## 20   Sandviken      36978
```

1.5

The number of inhabitants in all swedish cities and the number of inhabitants in the selected cities are compared with the following histograms.





As mentioned in *1.4* it is easily concluded that on average are the selected cities bigger than the average swedish city. This is a result of the selection procedure described in *1.1-1.2*. The probability for a city to be among the chosen cities is connected to its population since a higher number of inhabitants gives a higher probability for being selected. As a result of this selection procedure are the obtained histograms for the respective data sets rather dissimilar to each other.

Assignment 2

2.1

For generating values from the double exponential distribution from the uniform[0,1] distribtion according to the inverse CDF method, the following steps has to be performed.
To start, the pdf for the double exponential distribution is

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

From the pdf are two integrals computed, one when $x < \mu$ and one when $x \geq \mu$. The first then is

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

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

And the second is

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

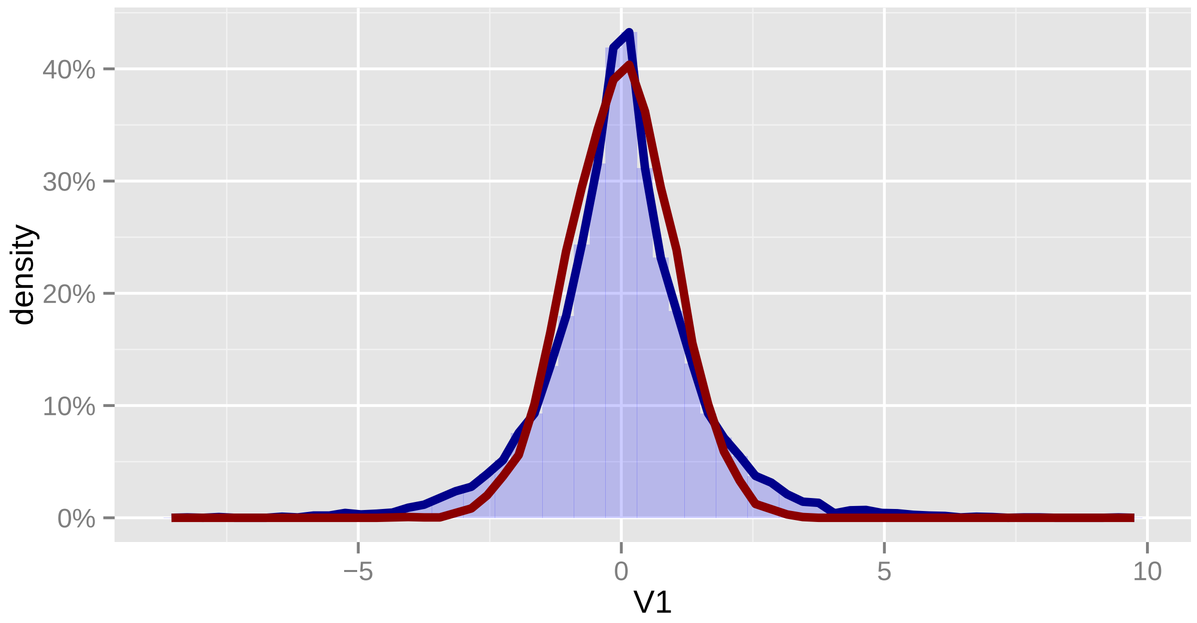
$$= 1 - \frac{1}{2}exp(-\alpha(x-\mu))$$

Next step is to solve for x in the respective cases.

$$x = ln(2U) + \mu$$

$$x = \mu - ln(2-2U) * \frac{1}{\alpha}$$

The equations above can be used for generating values from the double exponential distribution by using values from the uniform[0,1] distribution. As an example is 10000 values generated from DE(0,1) with the inverse CDF method. The result is shown in the histogram below. The blue area and the blue line visualizes the generated DE(0,1) values and the red line visualizes 10000 generated N(0,1) values via the *rnorm* function.



Compared to the normal distribution is the double exponential distribution expected to have fatter tails and a higher peak around zero. As can be seen in the histogram above is this the case for the DE(0,1) values generated with the inverse CDF method, hence the result appears to be reasonable.

2.2

The following steps are performed in order to use the acceptance/rejection method with DE(0,1) as majorizing density and N(0,1) as target density.
¤ First, the value of the majorizing constant c is calculated.
This is done by using the Laplace distributed values from *2.1*. The value of c must be higher than all ratios of $f_y(x)$ and $f_x(x)$ where f_y is the pdf for the double exponential distribution and f_x is the pdf for the normal distribution.

$$c > \frac{f_x(x)}{f_y(x)}$$

Then, the value of c is determined to be 1.315489.
¤ The first step of the acceptance/rejection algorithm is to generate Y.
This is done with the inverse CDF method explained in *2.1*
¤ Then the generated Y values is inserted into $f_y(Y)$ and $f_x(Y)$.
This is done by using the probability density function for f_y, the double exponential distribution, and f_x, the normal distribution.
¤ Generate a U[0,1] value.
¤ Test if Y should be accepted or rejected.
If the Y should be kept or not is decided with the following rule. If

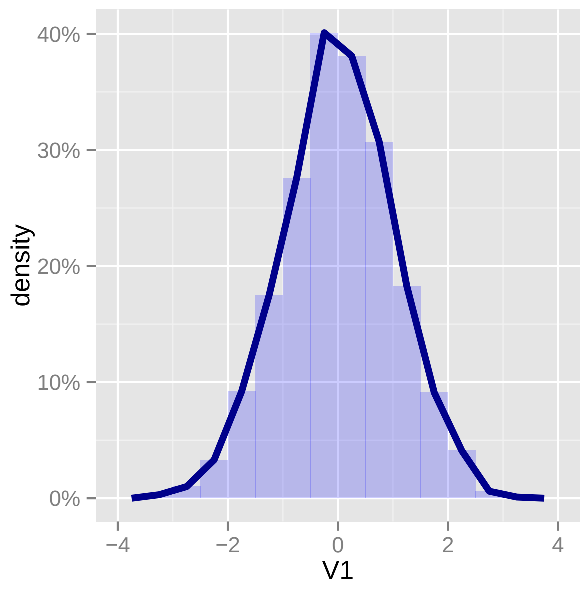
$$U \leq \frac{f_x(Y)}{c * f_y(Y)}$$

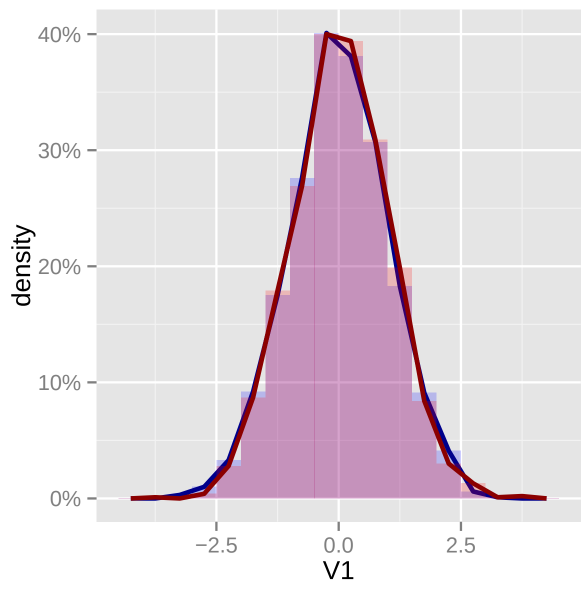
keep Y, else return to step one and generate a new Y value.

The implemented acception/rejection algorithm is tested and 2000 N(0,1) values are generated. In the histogram below are these values illustrated with the blue line and the red line is 2000 N(0,1) values generated with the *rnorm* function.
The acceptance rate is $\frac{1}{c}$, hence the rejection rate is $1 - \frac{1}{c}$. The expected rejection rate then is

$$1 - \frac{1}{1.315489} = 0.24$$

The 2000 N(0,1) values generated with the acceptance/rejection method required 2604 Y values to be generated. So 604 values were rejected, resulting in an rejection rate of 0.23. In other words are the obtained rejection rate very close to the expected rejection rate. Given this results does the chosen value for c seem to be reasonable.
In the histogram below to the left are the values generated from the acception/rejection method plotted. In the histogram to the right are these values, the blue line, compared to 2000 N(0,1) values, the red line, generated with the *rnorm* function.





As can be seen in the histogram to the right are the obtained histograms very similar to each other. Both the acceptance/rejection algorithm and the *rnorm* function produces values that seem to be N(0,1) distributed.

Appendix

```r
library(ggplot2)
library(scales)
require(XLConnect)
wb = loadWorkbook("C:/Users/Gustav/Documents/Computational-Statistics/Lab3/population.xls")
df = readWorksheet(wb, sheet = "Table", header = FALSE)

data1 <- df[, 1:3]
names(data1) <- c("cityCode", "City", "Population")
data1 <- subset(data1, data1$cityCode > 30)
cityFunc <- function(data){
  data$prob <- data$Population / sum(data$Population)
  data$cumProbs <- cumsum(data$prob)
  unifV <- runif(1,0,1)
  dataNew <- subset(data, data$cumProbs >= unifV)
  cityNum <- which(data$cumProbs==min(dataNew$cumProbs))
  return(cityNum)
}
selectData <- data1
newData <- as.data.frame(matrix(seq(20),nrow=20,ncol=3))
for(i in 1:20){
  set.seed(i)
  cityNum <- cityFunc(selectData)
  newData[i, ] <- selectData[cityNum, 1:3]
  selectData <- selectData[-cityNum, ]
}
options(scipen = 999)
```

```r
Cities <- newData[, 2:3]
names(Cities) <- c("City", "Population")
Cities
library(gridExtra)
All <- ggplot(data1, aes(Population)) + geom_histogram(binwidth=10000, fill="darkorange", col="darkorang
  scale_x_continuous(labels=comma) + theme(panel.grid.minor=element_blank())
Sel <- ggplot(newData, aes(V3)) + geom_histogram(binwidth=10000, fill="darkorange", col="darkorange") +
  scale_x_continuous(labels=comma) + theme(panel.grid.minor=element_blank()) + labs(x = "Population") +
grid.arrange(All, Sel, ncol=2)
## Assignment 2 ##
# 10 000 random uniform values
set.seed(190216)
randoms <- runif(10000, 0, 1)
randIndex <- randoms < 0.5
# The uniformed values are transformed by using the inverse CDF method.
# The transformed values follows a DE(0,1) distribution.
DEval1 <- data.frame(V1=log(2*randoms[randIndex]))
DEVal2 <- data.frame(V1= -log(2-2*randoms[!randIndex]))
DEval <- data.frame(rbind(DEval1, DEVal2))
set.seed(160219)
NormTest <- data.frame(V1=rnorm(10000, 0, 1))
ggplot(DEval, aes(V1,..density..)) + geom_histogram(fill="blue", alpha=0.2,binwidth=0.3) +
  scale_y_continuous(labels = percent_format()) +
  geom_freqpoly(size=1.5, col="darkblue",binwidth=0.3) +
  geom_freqpoly(data = NormTest,size=1.5, col="darkred",binwidth=0.3)
c <- 1.315489
i <- 0
j <- 1
y <- data.frame(V1=0)
set.seed(311015)
while (length(y[,1])<2000) {
  i <- i+1
  #set.seed(i)
  Utest <- runif(1,0,1)
  if(Utest < 0.5){
    DEy <- log(2*Utest)
  }else{
    DEy <- -log(2-2*Utest)
  }
  fy_y <- 1/2 * exp(-1 * abs(DEy-0))
  fx_y <- dnorm(DEy, 0, 1)
  #set.seed(i+10)
  Uselect <- runif(1,0,1)
  if (Uselect <= fx_y/(c*fy_y)) {
    y[j,] <- DEy
    j <- j+1
  }else{
    j <- j
  }
}
set.seed(311015)
NormTest <- data.frame(V1=rnorm(2000, 0, 1))
ARnm <- ggplot(y, aes(V1,..density..)) + geom_histogram(fill="blue", alpha=0.2,binwidth=0.5) +
  scale_y_continuous(labels = percent_format()) +
  geom_freqpoly(size=1.5, col="darkblue",binwidth=0.5)
RNnm <- ggplot(y, aes(V1,..density..)) + geom_histogram(fill="blue", alpha=0.2,binwidth=0.5) +
  scale_y_continuous(labels = percent_format()) +
  geom_freqpoly(size=1, col="darkblue",binwidth=0.5) +
  geom_histogram(data = NormTest, fill = "red", alpha = 0.2,binwidth=0.5)+
  geom_freqpoly(data = NormTest,size=1, col="darkred",binwidth=0.5)
grid.arrange(ARnm,RNnm, ncol=2)
```

```
## NA
```