Computational Statistics

Lab 3

Emil K Svensson and Rasmus Holm

2017-02-21

Question 1

1.1

```
pop <- read.csv2("../data/population.csv", encoding = "latin1")
```

1.2

```
cityUnif <- function(data){
    data$prob <- data$Population / sum(data$Population)
    data$cumprob <- cumsum(data$prob)
    return(which.min(data$cumprob < runif(1)))
}
```

1.3

```
n <- 20

rmpop <- pop
selpop <-  data.frame()

set.seed(123456)

for (i in 1:n) {
    popidx <- cityUnif(rmpop)
    selpop <- rbind(selpop, rmpop[popidx,])
    rmpop <- rmpop[-popidx,]
}
```

1.4

```
print(paste(as.character(selpop$Municipality),":",selpop$Population))
```

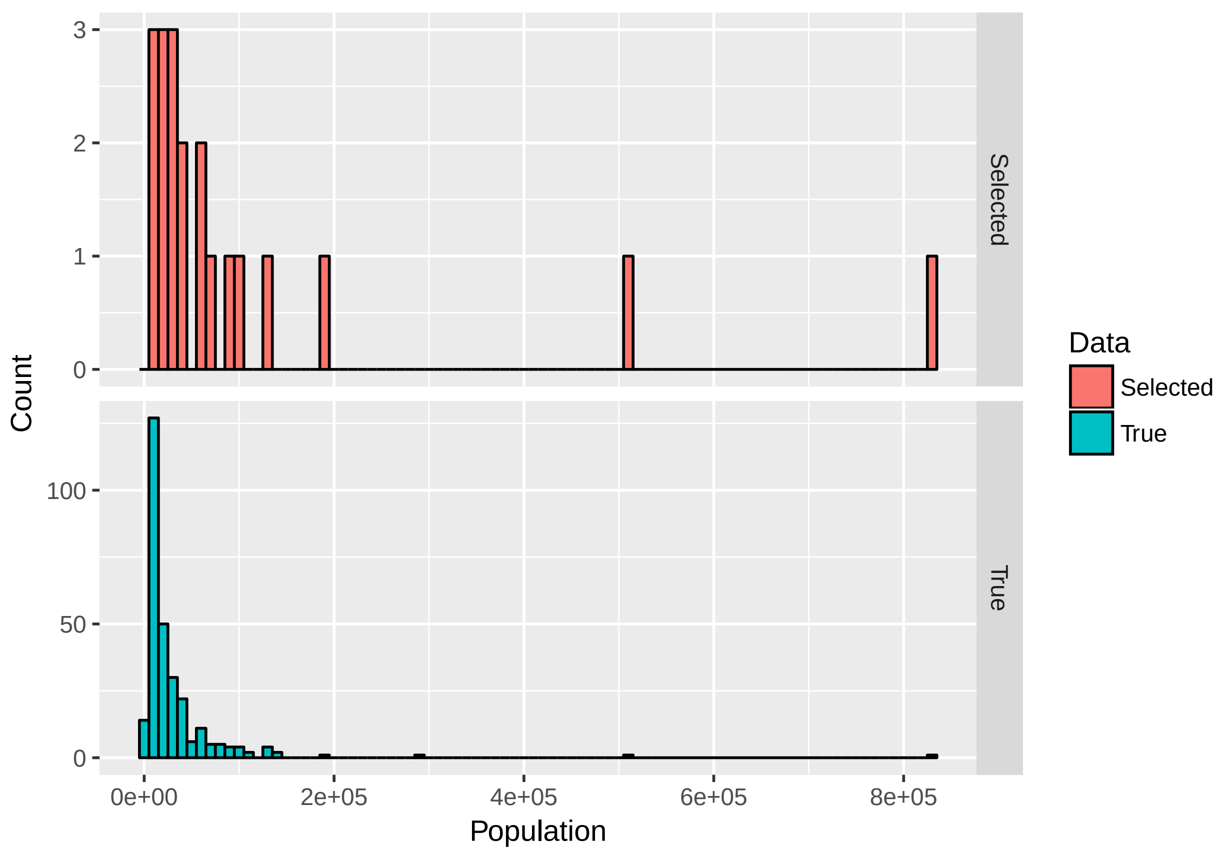
```
##  [1] "Karlskoga : 29742"    "Ulricehamn : 22753"   "Kalmar : 62388"
##  [4] "Jönköping : 126331"   "Ljungby : 27410"      "Täby : 63014"
##  [7] "Trelleborg : 41891"   "Stockholm : 829417"   "Luleå : 73950"
## [10] "Uppsala : 194751"     "Fagersta : 12249"     "Göteborg : 507330"
## [13] "Sundsvall : 95533"    "Gävle : 94352"        "Piteå : 40860"
## [16] "Ovanåker : 11530"     "Smedjebacken : 10758" "Katrineholm : 32303"
```

```
## [19] "Nybro : 19576"        "Hallsberg : 15235"
```

As seen in the output above, we in general select cities with large populations. This is logical since the addition of cities is based on the population, i.e. highly populated cities are more likely to be added, thus it is less likely to add smaller cities in the output.

1.5

```r
library(ggplot2)

true_pop <- data.frame(pop=pop$Population)
picked_pop <- data.frame(pop=selpop$Population)

true_pop$data <- "True"
picked_pop$data <- "Selected"

plot_data <- rbind(true_pop, picked_pop)

ggplot(plot_data, aes(pop, fill=data)) +
    geom_histogram(color="black", binwidth=10000) +
    facet_grid(data ~ ., scales="free_y") +
    xlab("Population") + ylab("Count") +
    labs(fill="Data")
```



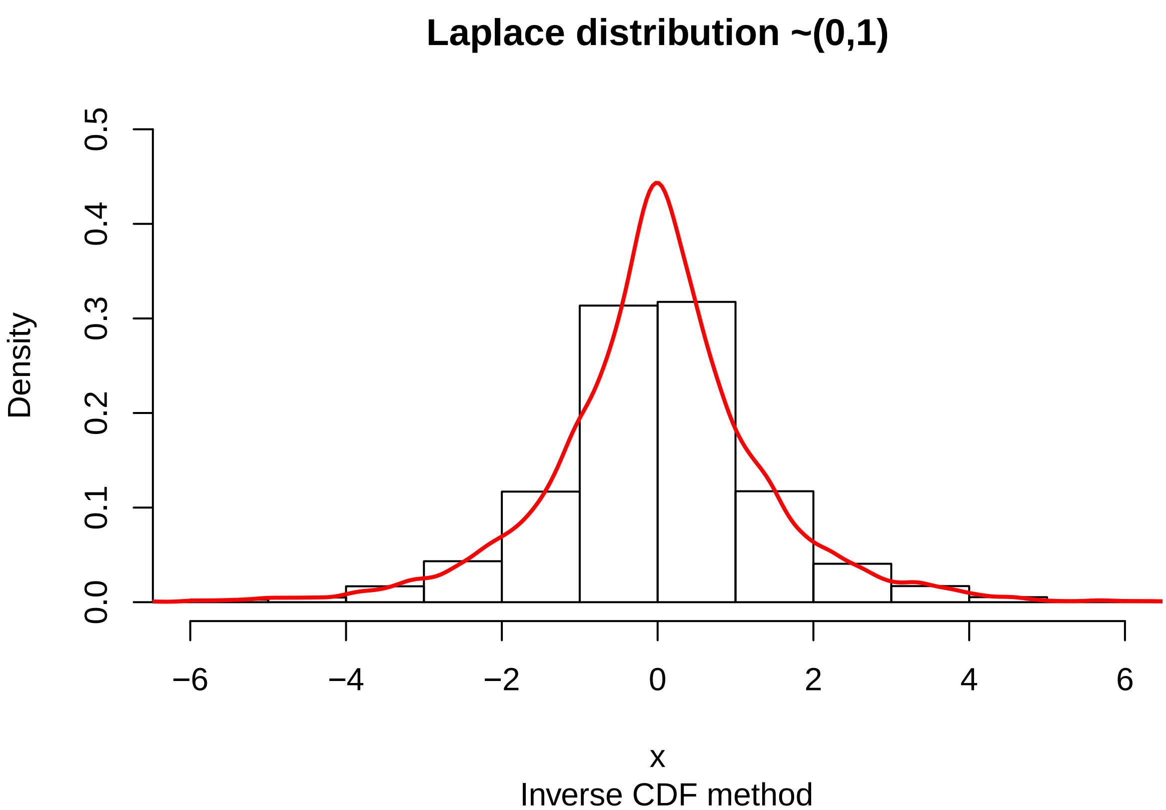
In the histograms above the width of a staple is 10000 and we can see that in the data set most cities are ≤ 50000 in population and the cities we selected are all ≤ 20000 in size. This is as described above expected but we can also see that it is expected in terms of the number of cities in that range.

Question 2

2.1

```
invLap <- function(){
    rng <- runif(1)

    if(rng > 0.5) {
        return(-log(2 - 2 * rng) )
    } else {
        return(log(2 * rng))
    }
}
```

```
plotdata <- sapply(1:10000,FUN = function(x) invLap())

hist(plotdata, prob = TRUE, ylim = c(0,0.5), xlim = c(-6,6),
     xlab = "x", main = "Laplace distribution ~(0,1)",
     sub = "  Inverse CDF method", breaks = 20)
lines(density(plotdata), col = "red", lwd = 2)
```



Yes, the histogram and the density-curve we included seems reasonable compared to the normal shape of a Laplace distribution.

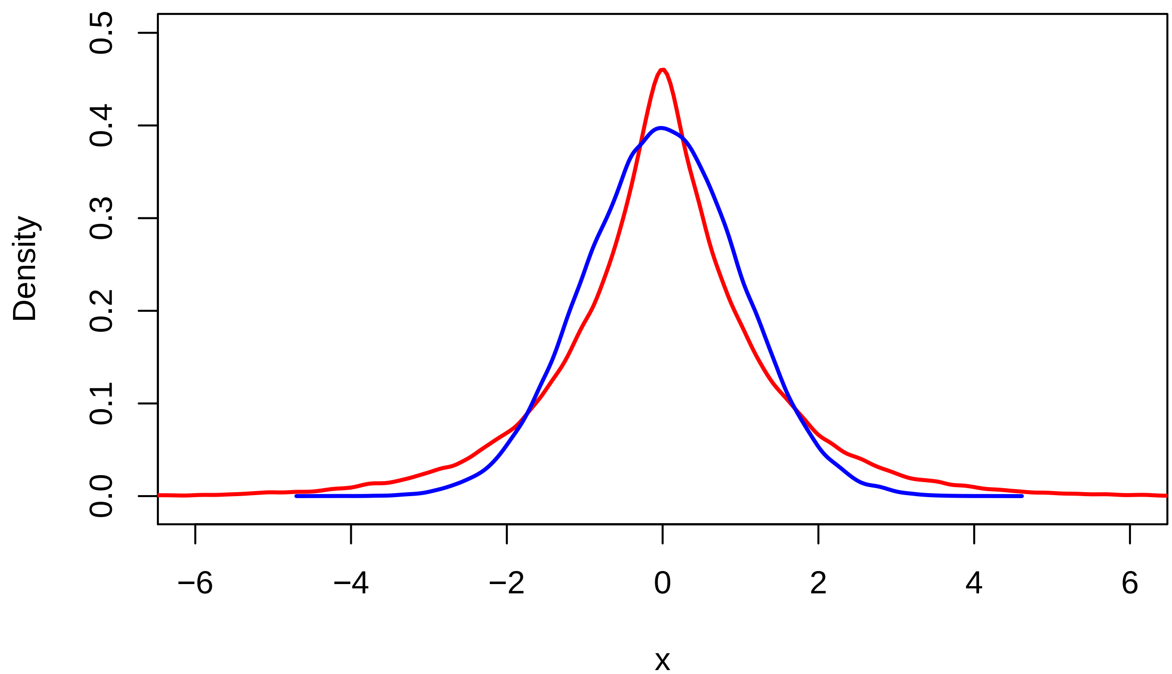
2.2

```
DE01 <- function(x) {
    (1 / 2) * exp(-abs(x))
}

ar <- function(c) {
  rej <- 0

  generated <- FALSE
  x <- c()

  while(!generated){
      y <- invLap() # Y ~ f_y
      u <- runif(1) # U(0,1)

      fx <- dnorm(y, mean = 0, sd = 1) # fx(y)
      fy <- DE01(y) #f_y

      if(u < fx / (c * fy)){
          return(c(y, rej))  #alt. set x <- y and generated = TRUE, to end the loop
      }

      rej <- rej + 1
  }
}
```

Here is our function.

Choosing the appropriate c

```
set.seed(123456)
n <- 100000
lap <- sapply(1:n,FUN = function(x) invLap())
nor <- rnorm(n, 0, 1)

plot(x = 0, y = 0, col ="white", xlim = c(-6,6),
     ylim = c(-0.01, 0.5), xlab = "x",
     ylab = "Density" )
lines(density(lap), col = "red", lwd = 2)
lines(density(nor), col = "blue", lwd = 2)
```



Our approach for approximating c was to try to make a laplace-curve that would as much as possible cover the whole density-curve for the normal distribution.

We know that

$$
\begin{aligned}
f_y(x) &= \frac{1}{2}e^{-x} \\
f_x(x) &= \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}.
\end{aligned}
$$

For $x \geq 0$ since it is symmetric

$$
\begin{aligned}
cf_y(x) &\geq f_x(x) \\
c &\geq \frac{2}{\sqrt{2\pi}}e^{-\frac{x^2}{2}+x}.
\end{aligned}
$$

To get the minimum c required we have to maximize $e^{-\frac{x^2}{2}+x}$ which is the same as to maximize $-\frac{x^2}{2}+x$.

$$
\begin{aligned}
g(x) &= -\frac{x^2}{2}+x \\
g'(x) &= 1-x \\
1-x &= 0 \Rightarrow x = 1
\end{aligned}
$$

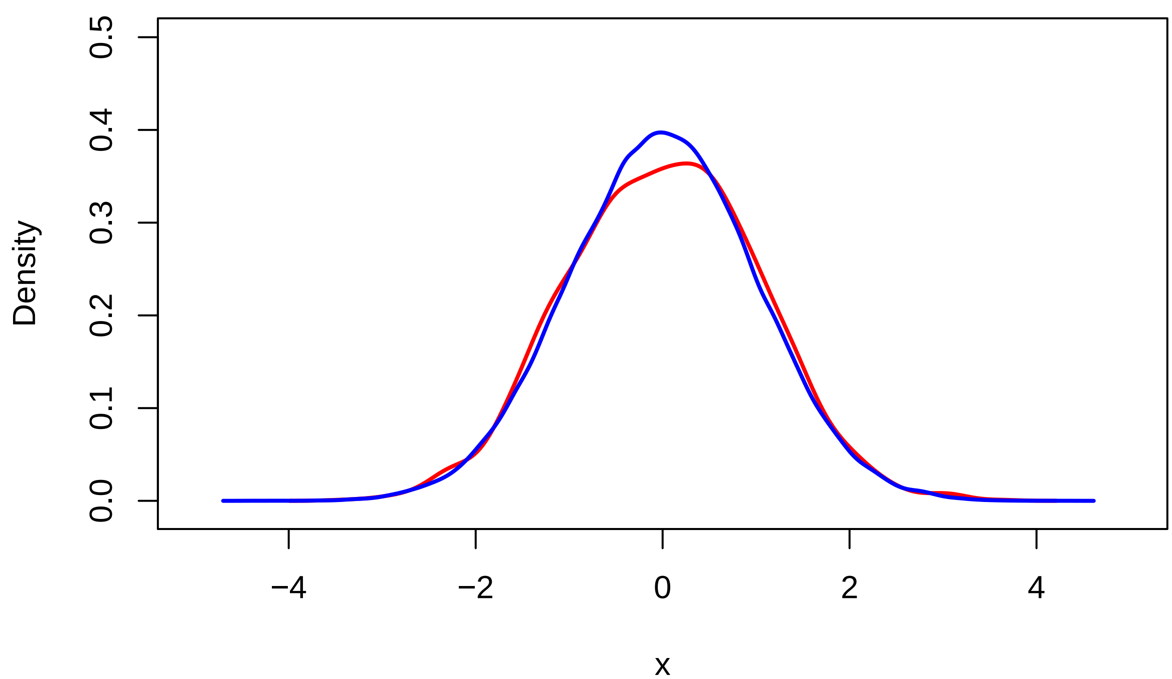
This gives us that

$$
\begin{aligned}
c &= \frac{2}{\sqrt{2\pi}} e^{-\frac{1^2}{2}+1} \\
&= \frac{2}{\sqrt{2\pi}} \sqrt(e) \\
&\approx 1.315
\end{aligned}
$$

Simulation and comparison

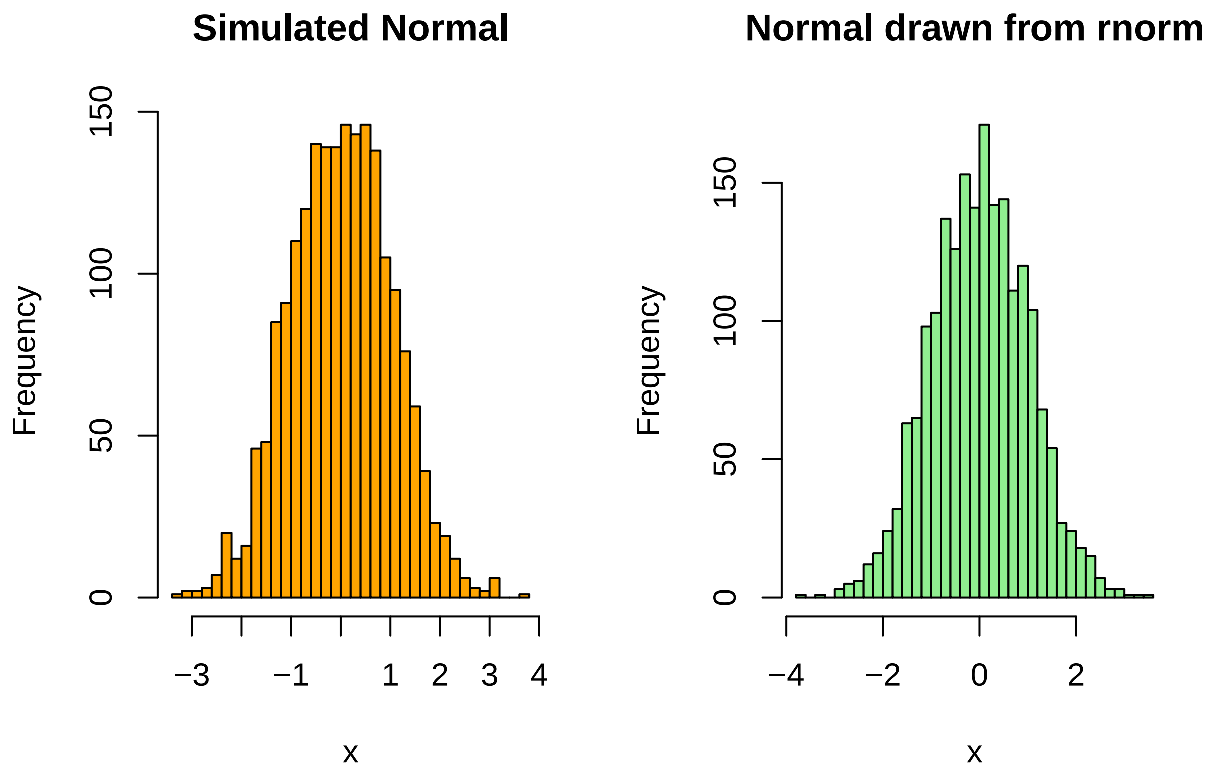
```
ARsim <- data.frame(rn = 1, rej = 1)
c <- 1.32

for(i in 1:2000){
    ARsim[i,] <- ar(c=c)
}
```

```
plot(x = 0, y = 0, col ="white", xlim = c(-5,5),
     ylim = c(-0.01,0.5), xlab = "x", ylab = "Density" )
lines(density(ARsim$rn), col="red", lwd=2)
lines(density(nor), col="blue", lwd=2)
```




```
par(mfrow = c(1,2))
hist(ARsim$rn, main = "Simulated Normal", xlab = "x",
```

```
     col = "orange",  breaks = 30)

set.seed(123456)
hist(rnorm(2000,0,1), main = "Normal drawn from rnorm", xlab = "x",
     col = "lightgreen", breaks = 30 )
```



The simulated one is a bit more rough around the edges and more narrow in the center in comparison to the histogram drawn from the rnorm.

The expected rejection rate is $\frac{n(M-1)}{nM}$ since the expected number of iterations before a accepted value is M so intuitively the number of rejections is $M-1$. n here is only the number of values that we use, but these cancel each other out so they are not necessary but we left them here for clarity. The theoretical rejection rate would be 0.242 and what we empirically found was 0.262 which is reasonably close.