Lab 2

YI CHEN, yc3356

September 26, 2017

Instructions

Before you leave lab today make sure that you upload a .pdf file to the canvas page (this should have a .pdf extension). This should be the PDF output after you have knitted the file, we don't need the .Rmd file (don't upload the one with the .Rmd extension). The file you upload to the Canvas page should be updated with commands you provide to answer each of the questions below. You can edit this file directly to produce your final solutions. Note, however, in the file you upload you should the above header to have the date, your name, and your UNI. Similarly, when you save the file you should replace **UNI** with your actualy UNI.

Tasks

The work we do today, will build on the work we did in the previous lab.

Sample distribution of the sample mean

1) Generate 1 million random draws from a normal distribution with $\mu = 3$ and $\sigma^2 = 4$ and save them in a vector named **normal1mil**. Calculate the mean and standard deviation of **normal1mil**.

```
normal1mil <- rnorm(1000000,mean = 3,sd = 2)
mean1 <- mean(normal1mil)
sd1 <- sd(normal1mil)
paste('the mean is: ',as.character(mean1),', and the standard deviation is:
', as.character(sd1))

## [1] "the mean is:  3.00009381551907 , and the standard deviation is:
2.00037053176673"
```

2) Find the mean of all the entries in **normal1mil** that are greater than 3. You may want to generate a new vector first which identifies the elements that fit the criteria.

```
normal_greater_3 <- normal1mil[normal1mil>3]
mean_greater_3 <- mean(normal_greater_3)
paste('the mean of all the entires in normal1mil is: ',
as.character(mean_greater_3))

## [1] "the mean of all the entires in normal1mil is:  4.59573644058866"
```

3) Create a matrix **normal1mil_mat** from the vector **normal1mil** that has 10,000 columns (and therefore should have 100 rows).

```
normal1mil_mat <- matrix(normal1mil,ncol = 10000,nrow = 100)
```

4) Calculate the mean of the 1234^{th} column.

```
results <- vector()
for(i in 1:3){
        results <- c(results,mean(normal1mil_mat[,i]))
}
results

## [1] 3.217775 2.924384 3.059347
```

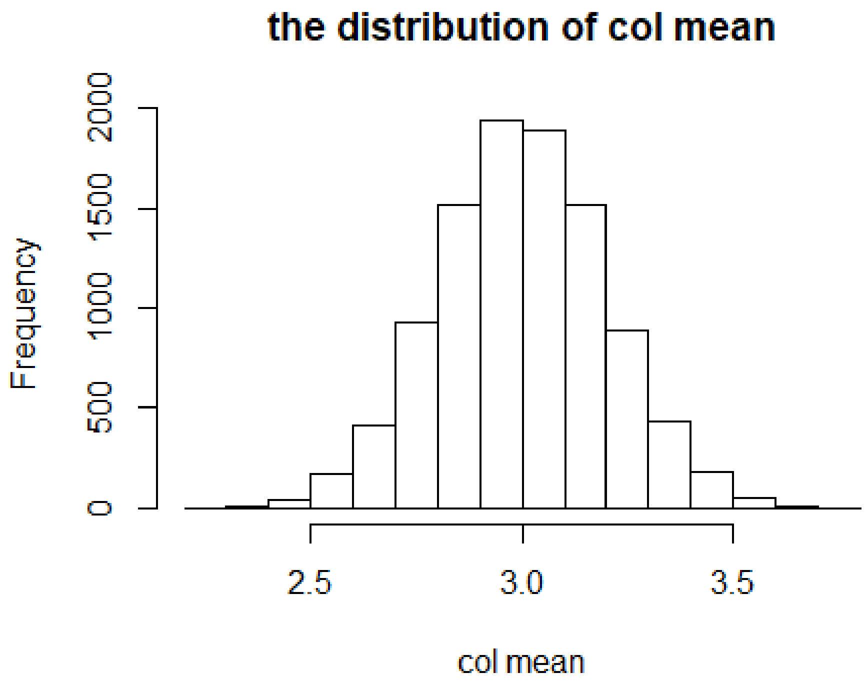
5) Use the **colSums()** functions to calculate the *means* of each column of **normal1mil_mat**. Save the vector of column means as **col_means** as it will be used in the next task. Use the **apply()** function to do the same thing, saving the results in a new vector **col_means2**, then compare the two with **all(col_means == col_means2)**.

```
col_means <- (colSums(normal1mil_mat))/(nrow(normal1mil_mat))
col_means1 <- colMeans(normal1mil_mat)
col_means2 <- apply(normal1mil_mat,2,mean)
all(col_means1 == col_means2)

## [1] TRUE
```

6) Finally, produce a histogram of the column means you calculated in task (5). What is the distribution that this histogram approximates (i.e. what is the distribution of the sample mean in this case)?

```
hist(col_means2,main = 'the distribution of col mean',xlab = 'col mean')
```



- explain: this distribution is approximate the normal distribution with mean equals to 3 and standard deviation equals to 4

7) Only complete the following questions if you have time. We'd like to count the number of values beyond 3 standard deviations from the mean in each column of our matrix and store them in a vector called **num_3devs** (that should have length 10,000). (a) Write code that loops over the columns and calculates this value for each column filling in the vector **num_3devs** as it iterates. (b) Use a smart call to the **apply()** function to do the same thing. (c) Make a table of these values.

```r
# (a)
num_3devs <- vector()
for(i in 1:ncol(normal1mil_mat)){
        mean_of_this_col <- mean(normal1mil_mat[,i])
        sd_of_this_col <- sd(normal1mil_mat[,i])
        uplimit <- mean_of_this_col + 3 * sd_of_this_col
        downlimit <- mean_of_this_col - 3 * sd_of_this_col
        judge <- (normal1mil_mat[,i] < downlimit) | (normal1mil_mat[,i] >
uplimit)
        over_3dev <-  normal1mil_mat[judge,i]
        result <- length(over_3dev)
        num_3devs <- c(num_3devs,result)
}

#(b)
judge <- function(x){
        m <- mean(x)
        s <- sd(x)
        uplimit <- m + 3 * s
        downlimit <- m - 3 * s
        n <- vector()
        for(i in 1:length(x)){
                if(x[i] < downlimit | x[i] > uplimit){
                    n <- c(n,x[i])
                }
        }
        length(n)
}


num_3devs <- apply(normal1mil_mat,2,judge)

#(c)
table(num_3devs)
```

```
## num_3devs
##    0    1    2    3 
## 7990 1899  110    1
```