Lab 1

Ming Zhong UNI:mz2692

September 13, 2018

Instructions

Before you leave lab today make sure that you upload an RMarkdown file to the canvas page (this should have a .Rmd extension) as well as the html output after you have knitted the file (this will have a .html extension). Note that since you have already knitted this file, you should see both a **Lab1_UNI.html** and a **Lab1_UNI.Rmd** file in your GR5206 folder. Click on the **Files** tab to the right to see this. The files 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.

Background: The Normal Distribution

Recall from your probability class that a random variable X is normally-distributed with mean μ and variance σ^2 (denoted $X \sim N(\mu, \sigma^2)$) if it has a probability density function, or *pdf*, equal to

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

In *R* we can simulate $N(\mu, \sigma^2)$ random variables using the `rnorm()` function. For example,

```
rnorm(n = 5, mean = 10, sd = 3)
```

```
## [1]  8.120639 10.550930  7.493114 14.785842 10.988523
```

outputs 5 normally-distributed random variables with mean equal to 10 and standard deviation (this is σ) equal to 3. If the second and third arguments are ommited the default rates are **mean = 0** and **sd = 1**, which is referred to as the "standard normal distribution".

Tasks

Sample means as sample size increases

1) Generate 100 random draws from the standard normal distribution and save them in a vector named **normal100**. Calculate the mean and standard deviation of **normal100**. In words explain why these values aren't exactly equal to 0 and 1.

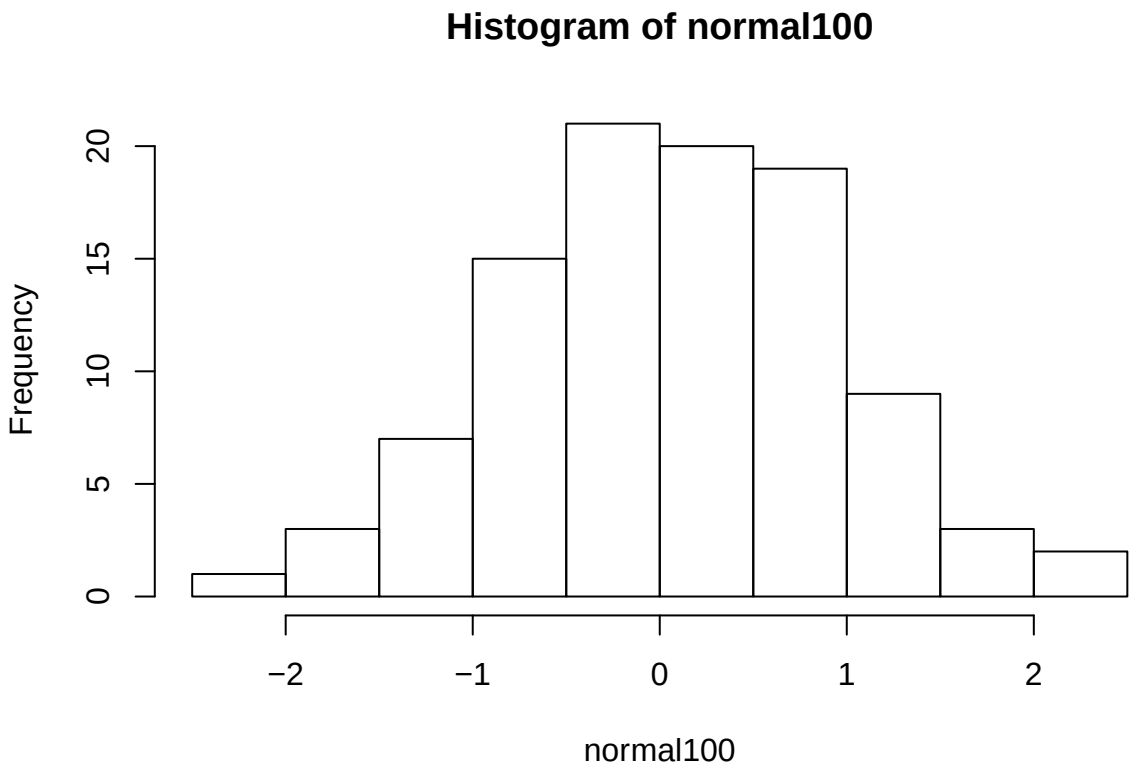
```
# Generate 100 random number from standard normal distribution.
  normal100 <- rnorm(n=100, mean=0, sd=1)
# Calculate the mean and standard deviation.
  mean=mean(normal100)
  sd=sd(normal100)
```

Answer:

The sample size of the distribution is rather small. So, the estimation is not accurate enough to get 0 and 1 exactly.

2) The function **hist()** is a base *R* graphing function that plots a histogram of its input. Use **hist()** with your vector of standard normal random variables from question (1) to produce a histogram of the standard normal distribution. Remember that typing **?hist** in your console will provide help documents for the **hist()** function. If coded properly, these plots will be automatically embedded in your output file.

```
#Genertate histogram for 100 numbers.
hist(normal100)
```



3) Repeat question (1) except change the number of draws to 10, 1000, 10,000, and 100,000 storing the results in vectors called **normal10**, **normal1000**, **normal10000**, **normal100000**.

```
#Generate 10,1000,10000,100000 numbers from standard normal distribution respectively.
normal10=rnorm(10,0,1)
normal1000=rnorm(1000,0,1)
normal10000=rnorm(10000,0,1)
normal100000=rnorm(100000,0,1)
```

4) We want to compare the means of our four random draws. Create a vector called **sample_means** that has as its first element the mean of **normal10**, its second element the mean of **normal100**, its third element the mean of **normal1000**, its fourth element the mean of **normal10000**, and its fifth element the mean of **normal100000**. After you have created the **sample_means** vector, print the contents of the vector and use the **length()** function to find the length of this vector. (it should be five). There are, of course, multiple ways to create this vector. Finally, explain in words the pattern we are seeing with the means in the **sample_means** vector.

```r
#Calculate means of datasets and group them into sample_means.
sample_means<-c(mean(normal10),mean(normal100),mean(normal1000),mean(normal10000),mean(normal100000))
print(sample_means)
```

```
## [1]  0.493735437  0.082566589 -0.026875723 -0.006719807 -0.001114476
```

```r
#Check the length of
length(sample_means)
```

```
## [1] 5
```

Answer:

The mean is geting closer to actual mean zero.

Sample distribution of the sample mean

5) Let's push this a little farther. 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**.

```r
#Generate 1 million random numbers from normal distribution
normal1mil=rnorm(1000000,3,2)

#Calculate the mean and sd; print them out
mean=mean(normal1mil)
sd=sd(normal1mil)
print(mean)
```

```
## [1] 3.000241
```

```r
print(sd)
```

```
## [1] 1.999961
```

6) 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.

```r
#Generate logical list as filter fit
fit=c(normal1mil>3)

#use fit logical list to filter out and get entries greater than 3
fit=normal1mil[fit]
```

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

```r
#Generate matrix
normal1mil_mat=matrix(normal1mil,nrow=100,ncol=10000)
```

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

```r
#Calculate the mean
mean(normal1mil_mat[,1234])
```

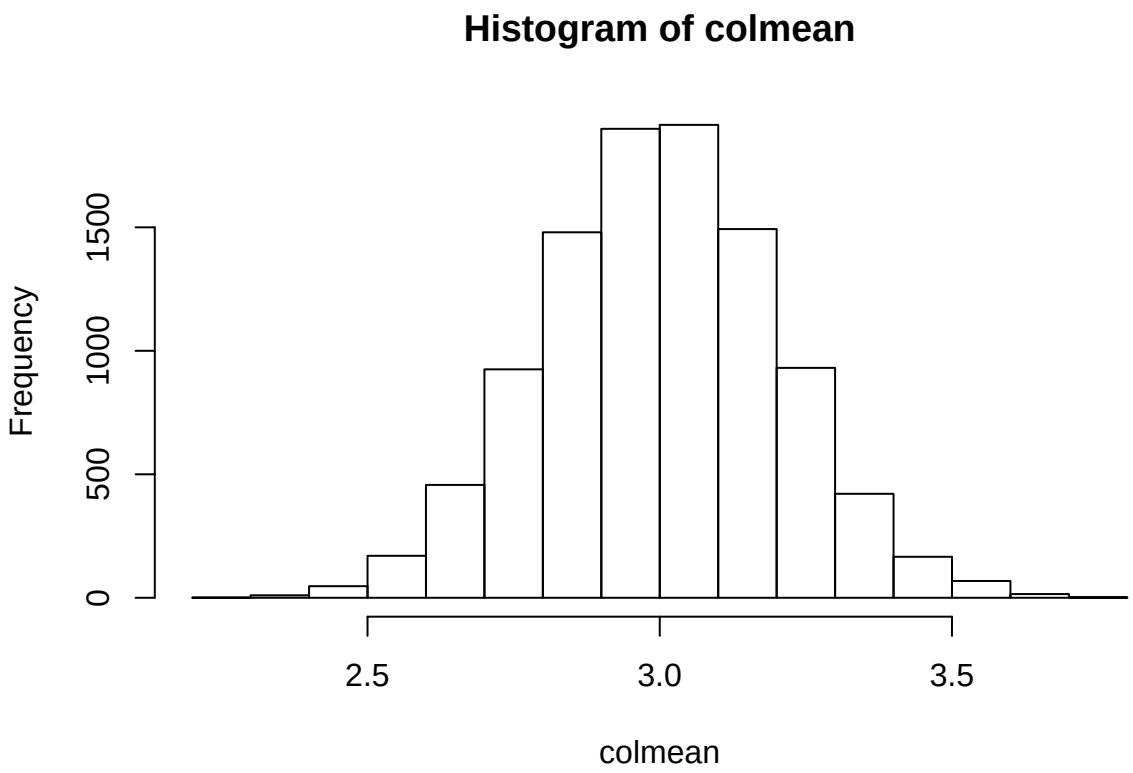
```
## [1] 3.056545
```

9) Use the **colSums()** functions to calculate the *means* of each column of **normal1mil_mat**. Remember, **?colSums** will give you help documents about this function. Save the vector of column means with an appropriate name as it will be used in the next task.

```
#Using ColumSums and ColMeans to get the column means list
colmean1=colMeans(normal1mil_mat,na.rm=FALSE,dims=1)
colmean=colSums(normal1mil_mat,na.rm=FALSE,dims=1)/100
```

10) Finally, produce a histogram of the column means you calculated in task (9). What is the distribution that this histogram approximates (i.e. what is the distribution of the sample mean in this case)?

```
#Generate histogram
hist(colmean)
```




```
sd(colmean)
```

```
## [1] 0.2011195
```

Answer:

The distribution is a proxy for noraml distribution with mean equal to 3.