Lab 1

Christine Chong cc4190

May 23, 2017

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 pdf output after you have knitted the file (this will have a .pdf extension). Note that since you have already knitted this file, you should see both a **Lab1_UNI.pdf** and a **Lab1_UNI.Rmd** file in your GR4206 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

Problem One ———————————————————————————————————————

Question:

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.

Answer:

We can generate 100 random draws from the standard normal distribution by using the **rnorm()** function and assigning the argument **n** (the number of desired outputs) with the value of 100.

Afterwards, we can assign the output of the function to a variable name **normal100** using **<-**.

```
normal100 <- rnorm(n = 100)
```

In order to find the standard deviation of **normal100**, we can use the **sd()** function with **normal100** as the argument.

```
sd(normal100)
```

```
## [1] 0.8891336
```

In order to find the mean of **normal100**, we can use the **mean()** function with **normal100** as the argument.

```
mean(normal100)
```

```
## [1] 0.08256659
```

Because **normal100** is from random draws, the actual value of **mean(normal100)** and **sd(normal100)** will vary. The numbers will always be close to 0 and 1 but never reach those numbers because of the random draws. The more random draws there are, the closer the mean and standard deviation should be to 0 and 1.

Problem Two

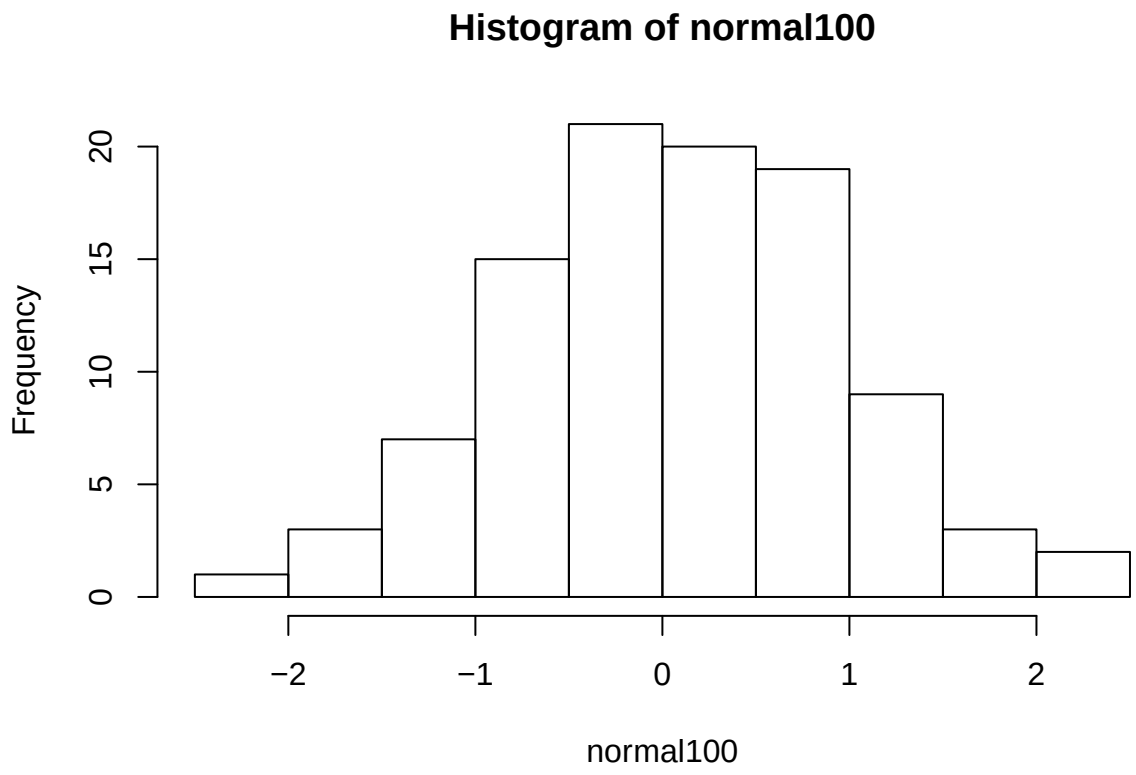
Question:

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.

Answer

In order to plot a histogram of **normal100**, we can use the function **hist()** with **normal100** as the argument.

```
hist(normal100)
```



Problem Three

Question:

Repeat problem (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**.

Answer

We can assign more variable names with a naming scheme related to the number of draws using the function **rnorm**, different numbers as the values of the **n** argument, and **<-**.

```
normal10 <- rnorm(n=10)
normal1000 <- rnorm(n=1000)
normal10000 <- rnorm(n=10000)
normal100000 <- rnorm(n=100000)
```

Problem Four ———————————————————————————————————————-

Question:

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.

Answer

In order to create a vector with a data frame mode named **sample_means** with declaring the different variables as the means of the different components (using the **mean()** function), we can use **data.frame()**.

```
sample_means <- data.frame(first=mean(normal10), second=mean(normal100), third=mean(normal1000), fourth=
```

Sample distribution of the sample mean

In order to print the vector named **sample_means**, we can use the **print()** function it after declaring it.

```
print(sample_means)
```

```
##       first     second       third       fourth        fifth
## 1 0.4937354 0.08256659 -0.02687572 -0.006719807 -0.001114476
```

In order to find the length of the vector, we can use the **length()** function with sample_means as the argument.

```
length(sample_means)
```

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

The means are getting closer to the value of 0.

Problem Five ———————————————————————————————————————

Question: 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**.

Answer

We can use the function **rnorm** with 1,000,000 as the value of the **n** argument with 3 as the value of the mean argument and 4 as the value of the the sd (standard deviation) argument.

```
normal1mil <- rnorm(n=1000000, mean=3, sd=4)
```

Then we can use the **mean()** function with the argument **normal1mil** to calculate the mean of **normal1mil**.

```
mean(normal1mil)
```

```
## [1] 3.000482
```

Then we can use the **sd()** function with the argument **normal1mil** to calculate the standard deviation of **normal1mil**.

```
sd(normal1mil)
```

```
## [1] 3.999923
```

Problem Six ———————————————————————————————————————

Question:

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.

Answer

To find the mean of all the entries in **normal1mil** that are greater than 3, we must first create a new vector called **greaterthan3** that filters all the values greater than 3 by using the $>$ relational operator.

```
greaterthan3 <- normal1mil[normal1mil > 3]
```

'Then we can use the **mean()** function with the vector **greaterthan3** as the argument to get the mean of the all the entries in **normal1mil** that are greater than 3.

```
mean(greaterthan3)
```

```
## [1] 6.192317
```

Problem Seven ———————————————————————————————————————

Question:

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

Answer

To create a matrix called **normal1mil_mat**, we can use the **matrix()** function using **normal1mil** as the value of the data argument, 100 as the value of the nrow argument and 10,000 as the value of the ncol argument.

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

Problem Eight ———————————————————————————————————————

Question:

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

Answer

To find the mean of the 1234^{th} column, we can use the **mean** function with **normal1mil_mat[,1234]** as the argument.

```
mean(normal1mil_mat[,1234])
```

```
## [1] 3.113089
```

Problem Nine ——-

Question:

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.

Answer

In order to calculate the **means** of each column of **normal1mil_mat**, we can use the **colSums()** function with **normal1mil_mat** as the argument and assign it a vector name of **milColSums** using **<-**.

```
milColSums <- colSums(normal1mil_mat)
```

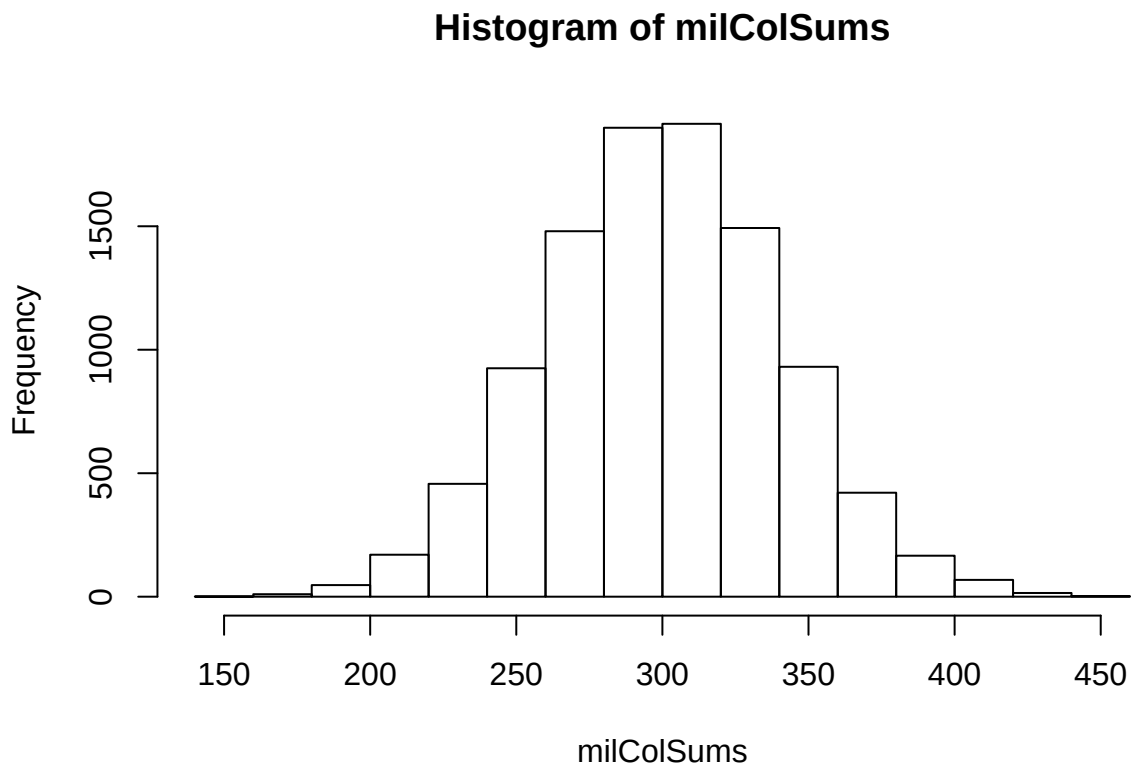
Problem Ten ——

Question:

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)?

Answer In order to produce a histogram of **milColSums**, we can use the **hist()** function with **milColSums** as the argument.

```
hist(milColSums)
```



The type of distribution seen from the histogram of **milColSums** is a normal distribution.