# Homework 04 T-tests, Sampling Distributions, and the Bootstrap

Due by 11:59pm, Friday, February 14, 2025, 11:59pm

S&DS 230/530/ENV 757

**(1) Practice with Loops.** *(10 points)* For this problem, use loops even if you could do the task without them.

(1.1) A Tribonacci sequence is a series of integers where each number after the third number is found by adding together the three integers before it. Starting with 0, 1, 1, the sequence goes:

0, 1, 1, 2, 4, 7, ...

Write a loop that fills a vector called `myTrib` with this sequence going up to a total length of 21 numbers. Display the final value of `myTrib` AND write a line of code that calculates how many digits are in the final number in `myTrib`.

(1.2) Here is the link to the World Bank data from 2024:

http://www.reuningscherer.net/s&ds230/data/WB_2024.csv

Read the data into a dataframe called `wb`. Write a loop to fill a vector called `compVals` having length equal to the number of columns in the World Bank data frame. The i-th entry in `compVals` should be a number (>= 0) equal to the total number of non-missing values in the i-th column of World Bank data frame. Make a histogram of `compVals` and label as appropriate. In addition, make a histogram that displays the units on the horizonal axis in terms of percentages (not proportions) for each variable (rather than totals). Write a sentence or two about what these plots tell you about the dataset.

*(For full credit, use only one for-loop to do part 1.2)*

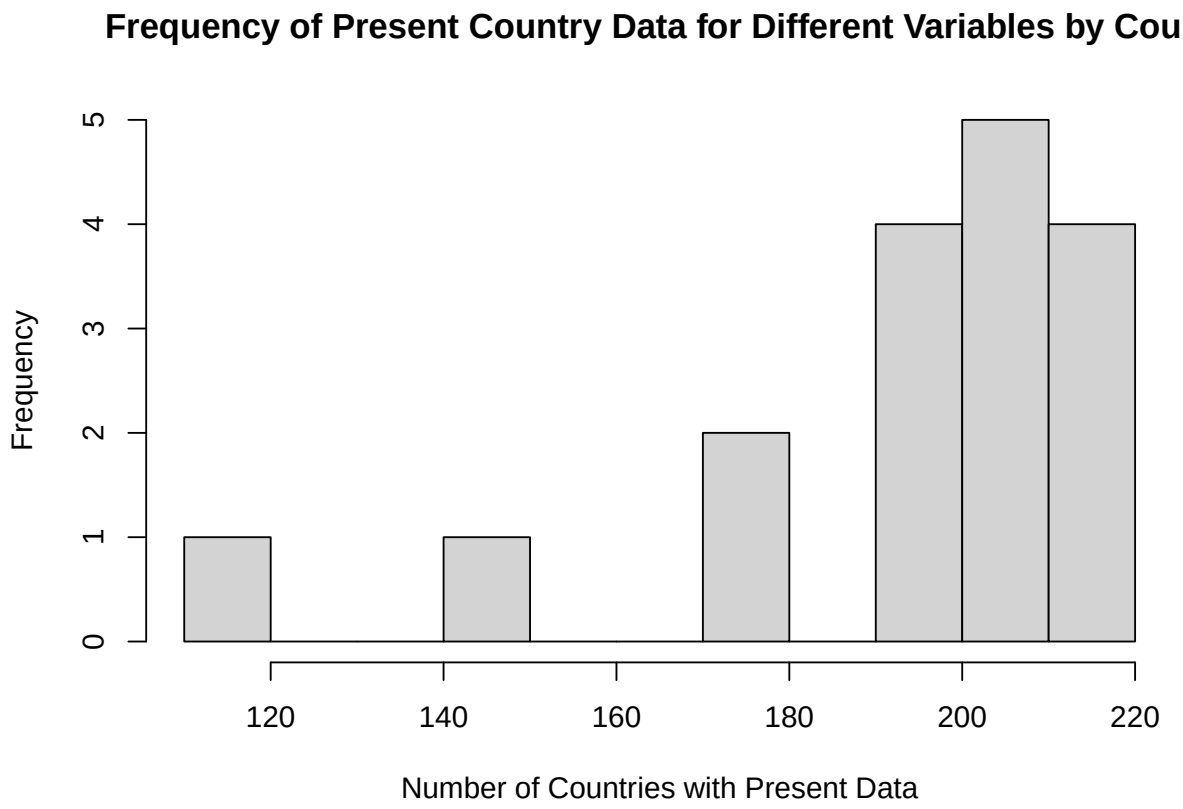
```
myTrib <- c(0, 1, 1)
for (i in 4:21) {
  myTrib[i] = myTrib[i-1] + myTrib[i-2] + myTrib[i-3]
}
myTrib
```

```
##  [1]     0     1     1     2     4     7    13    24    44    81   149   274
## [13]   504   927  1705  3136  5768 10609 19513 35890 66012
```

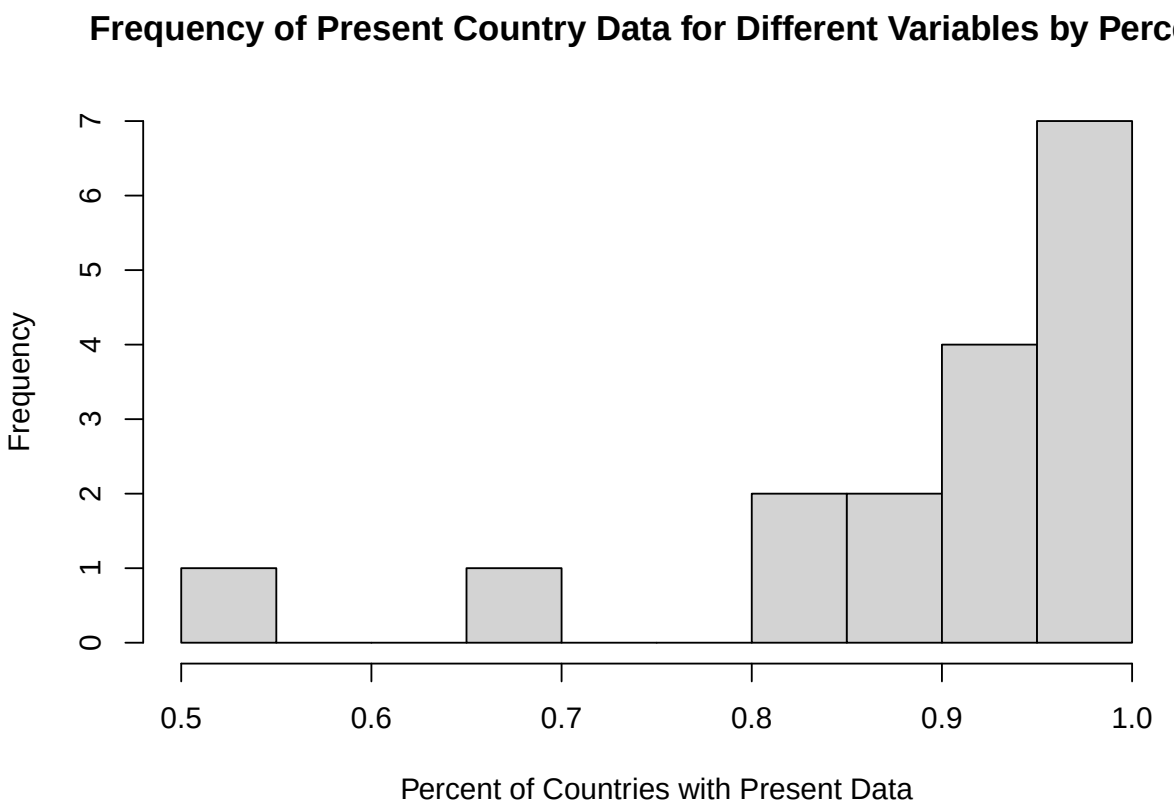
```
nchar(as.character(myTrib[21]))
```

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

```r
wb <- read.csv('http://www.reuningscherer.net/s&ds230/data/WB_2024.csv')
compVals <- c()
for (i in 1:17) {
  compVals[i] = sum(!is.na(wb[,i]))
}
hist(compVals,
     breaks=10,
     xlab="Number of Countries with Present Data",
     main="Frequency of Present Country Data for Different Variables by Count")
```




```r
compPers <- compVals / 217
hist(compPers,
     breaks=10,
     xlab="Percent of Countries with Present Data",
     main="Frequency of Present Country Data for Different Variables by Percent")
```




*Most of the countries in the dataset record/report data about most of the different variables in the dataset, except for 2 variables.*

**(2) Simulations with the Exponential Distribution** *(40 points).*

For this problem, we'll investigate the sampling distributional characteristics of three statistics. In particular, suppose we take a sample of size 20 from an exponential distribution. We can use the CLT to say something about how far the sample mean is likely to be from the true mean, but how far are the sample median or the sample variance likely to be from the true values in an exponential distribution where we take a sample of size 20? Also, what do we expect the distribution of these statistics to look like?

(2.1) *(10 points)* First, let's get a quick sense of what an exponential distribution looks like where the mean is 4. By the way, it's handy to know that for an exponential distribution with mean 4, the variance is 16 and the median is 4*ln(2). You can read about the exponential distribution HERE.

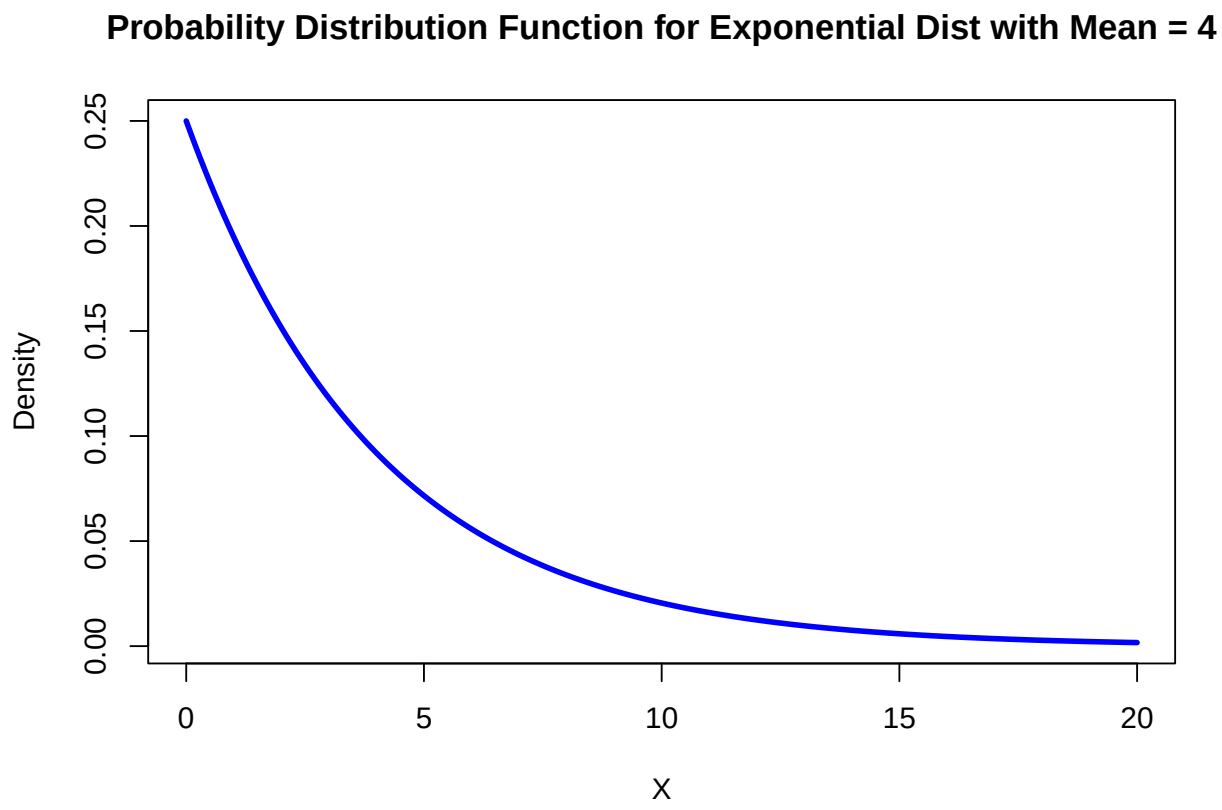
The code below gives a quick plot of this distribution. Your job is to succinctly answer what each part of the code does. You'll probably need to get help on the function `dexp()`, `seq()` and on a few of the graphics parameters in `par()`.

Then, add additional code that repeats the original code but uses the option `by = 4` in the `seq()` function. Why did I choose `by = .1` in the original code?

Finally, modify the code to produce a plot that shows the shape of exponential distributions with means of 1, 2, and 4 all on the same plot using different colors and line types for each distribution. Be sure to add a legend to your plot.

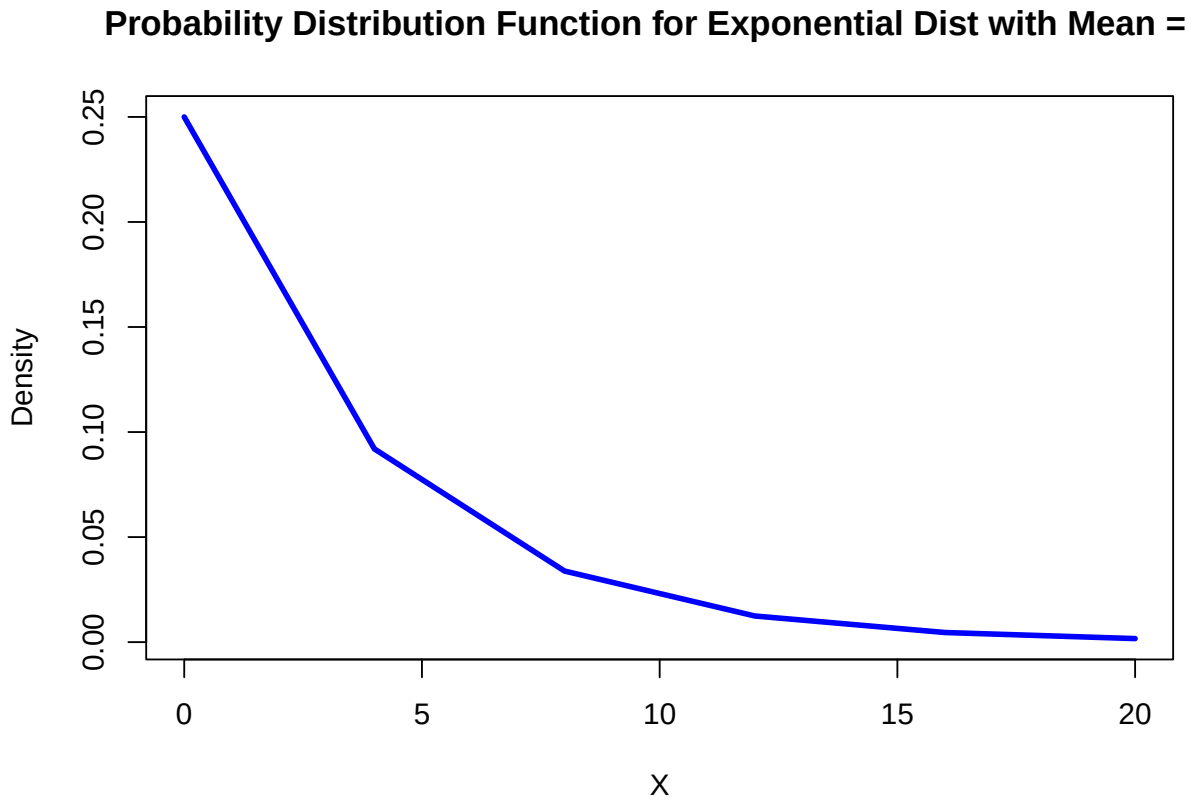
```
#Get exponential probabilities - note that rate = .5 gives us mean of 2 (mean is 1/rate)
probs <- dexp(seq(0, 20, by = .1), rate = .25)
```

```r
#dexp: gets the y values of an exponential distribution at a specified rate for
#      various x values
#rate: specifies the rate of the exp dist, which is one over the expected time
#      between two events
#by: the increments between adjacent x levels

#Plot sampling distribution
plot(seq(0, 20, by = .1), probs,
     type = 'l',
     lwd = 3,
     col = "blue",
     main = "Probability Distribution Function for Exponential Dist with Mean = 4",
     xlab = "X",
     ylab = "Density")
```



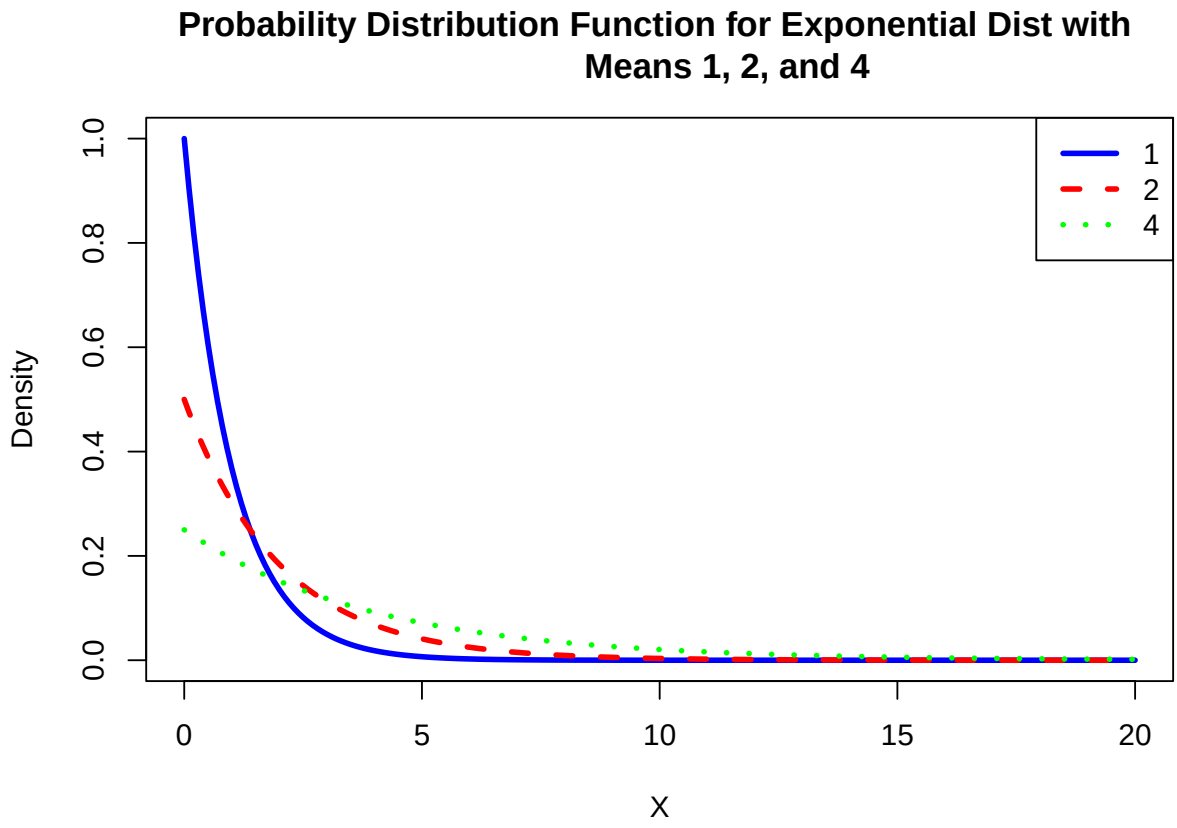

```r
#type = 'l': converts the scatterplot into a line graph, which connects
#            adjacent points with a line
#lwd: line width

#Add code below based on instructions
plot(seq(0, 20, by = 4),
     dexp(seq(0, 20, by = 4), rate = .25),
     type = 'l',
     lwd = 3,
```

```
     col = "blue",
     main = "Probability Distribution Function for Exponential Dist with Mean = 4",
     xlab = "X",
     ylab = "Density")
```




```
plot(seq(0, 20, by = 0.1),
     dexp(seq(0, 20, by = 0.1), rate = 1),
     type = 'l',
     lwd = 3,
     col = "blue",
     main = "Probability Distribution Function for Exponential Dist with
             Means 1, 2, and 4",
     xlab = "X",
     ylab = "Density")
points(seq(0, 20, by = 0.1),
       dexp(seq(0, 20, by = 0.1), rate = 0.5),
       type=  'l',
       lwd = 3,
       lty = 2,
       col = "red")
points(seq(0, 20, by = 0.1),
       dexp(seq(0, 20, by = 0.1), rate = 0.25),
       type=  'l',
       lty = 3,
```

```
       lwd = 3,
       col = "green")
legend("topright",
       legend = c("1", "2", "4"),
       col = c("blue", "red", "green"),
       lwd = 3,
       lty = c(1, 2, 3))
```




*The first line gets the y levels of an exponential distribution with rate 0.25 at various x levels, namely the points generated by seq(0, 20, by = .1), which are 0, 0.1, 0.2, . . . , 19.9, 20. Then these points are graphed on a line graph. The graph is really a very dense scatter plot but the density makes it appear to be a graph of a continuous function. Setting "by" to 4 makes the graph much more discrete.*

(2.2) *(4 points)* Following the example in class 8, get a random sample of 20 observations from an exponential distribution with mean 4. Repeat this process 10000 times. Save your results in a matrix called `samples` with 10,000 rows and 20 columns. Display the dimension of `samples'. Show the first 3 rows of`samples' but round the values to two decimal places.

```
# To make grading easier, please leave the following line of code in your assignment
set.seed(230)
samples <- matrix(nrow=10000, ncol=20)
for (i in 1:10000) {
  samples[i,] = rexp(20, 0.25)
}
dim(samples)
```

```
## [1] 10000    20
```

```
round(head(samples, 3), digits=2)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,] 5.29 9.12 1.03 9.29 5.59 0.13 0.41 1.71 4.10  3.40  1.66  0.42  3.56  0.15
## [2,] 1.80 1.28 0.51 5.28 7.92 0.82 0.17 2.97 8.35 14.77  0.11  0.53  2.76  5.12
## [3,] 2.40 4.94 3.63 2.16 0.26 2.49 7.93 1.20 9.42  7.67  2.75  3.70  1.27  1.99
##      [,15] [,16] [,17] [,18] [,19] [,20]
## [1,]  5.87  2.38  1.60  3.76  9.61  2.19
## [2,]  1.14  7.47  4.03  4.78 10.15  4.85
## [3,]  0.30  3.50  1.30  1.83  3.90  3.51
```

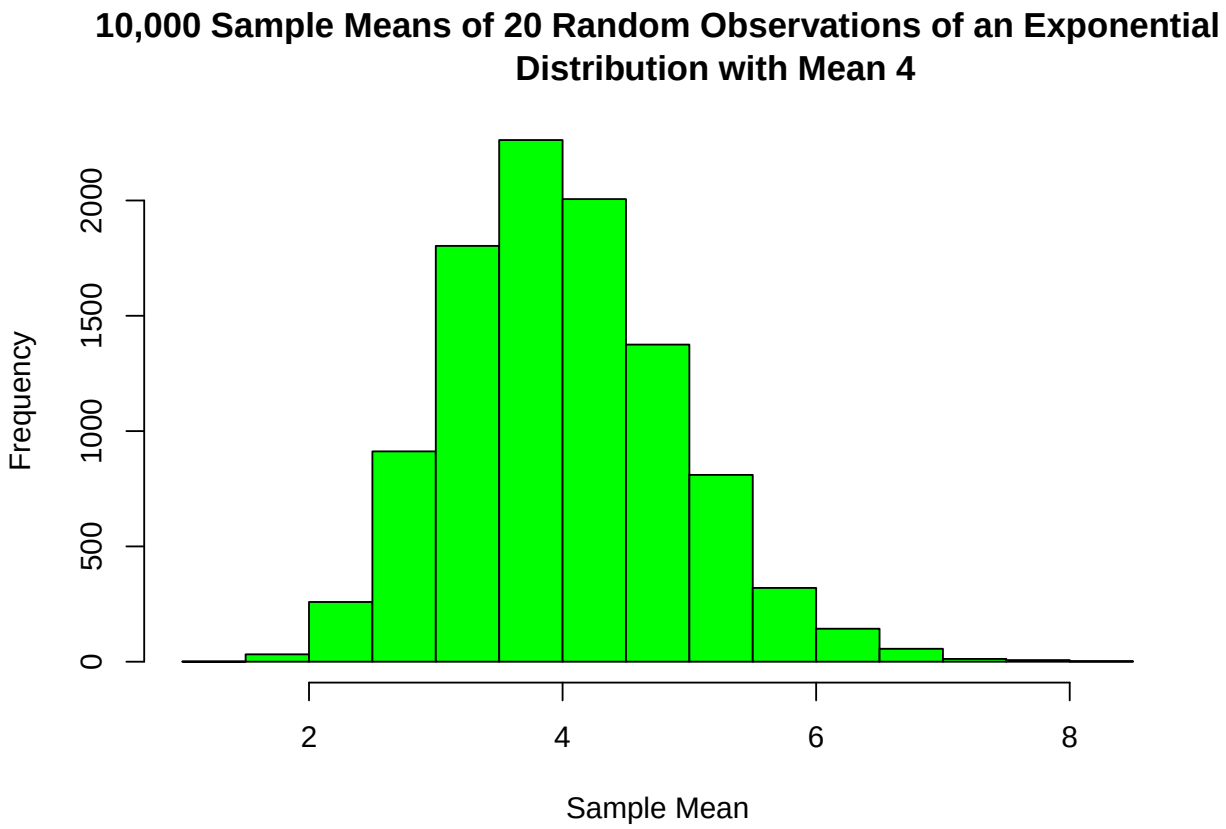
(2.3) *(4 points)* Calculate the sample mean for each sample of size 20 (i.e. calculate the mean for each row of `samples`). Repeat this process to get the sample median and the sample variance for each sample of size 20. Save these values in objects called, respectively, `smeans, smedians, svariance`.

```
smeans <- c()
smedians <- c()
svariance <- c()
for (i in 1:10000) {
  smeans[i] = mean(samples[i,])
  smedians[i] = median(samples[i,])
  svariance[i] = var(samples[i,])
}
```

(2.4) *(9 points)*

- Create a sample histogram of the sample means (make the bars green, make sure you label your axes and put on a clear title).
- Make a normal quantile plot of the sample means using the `qqPlot()` function in the `car` package. Comment on whether the CLT seems to be in effect.
- Get summary statistics OF THE SAMPLE MEANS and save this to an object called `ans1`. Using code, display only the element of `ans1` that is the sample mean, rounded to two decimal places. Is this the value you expect?
- Calculate and display the sample standard deviation of the sample means (use the function `sd()`)and display rounded to two decimal places. Then, use code to calculate the value you'd expect based on the CLT, again rounded to two decimal places. Are the two values similar?

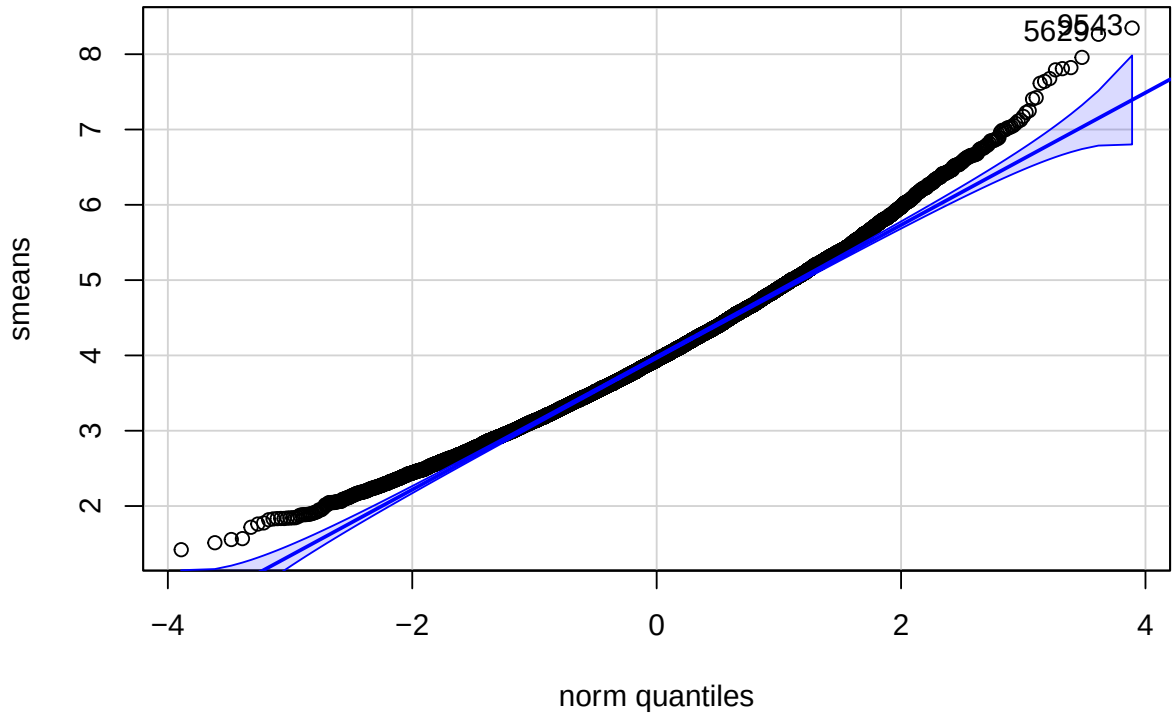
```
hist(smeans,
     col='green',
     xlab="Sample Mean",
     main="10,000 Sample Means of 20 Random Observations of an Exponential
           Distribution with Mean 4")
```




```
library("car")
```

```
## Loading required package: carData
```

```
qqPlot(smeans)
```




```
## [1] 9543 5629
```

```
ans1 <- summary(smeans)
round(ans1['Mean'], digits=2)
```

```
## Mean
## 4.01
```

```
round(sd(smeans), digits=2)
```

```
## [1] 0.89
```

```
round(4/sqrt(20), digits=2)
```

```
## [1] 0.89
```

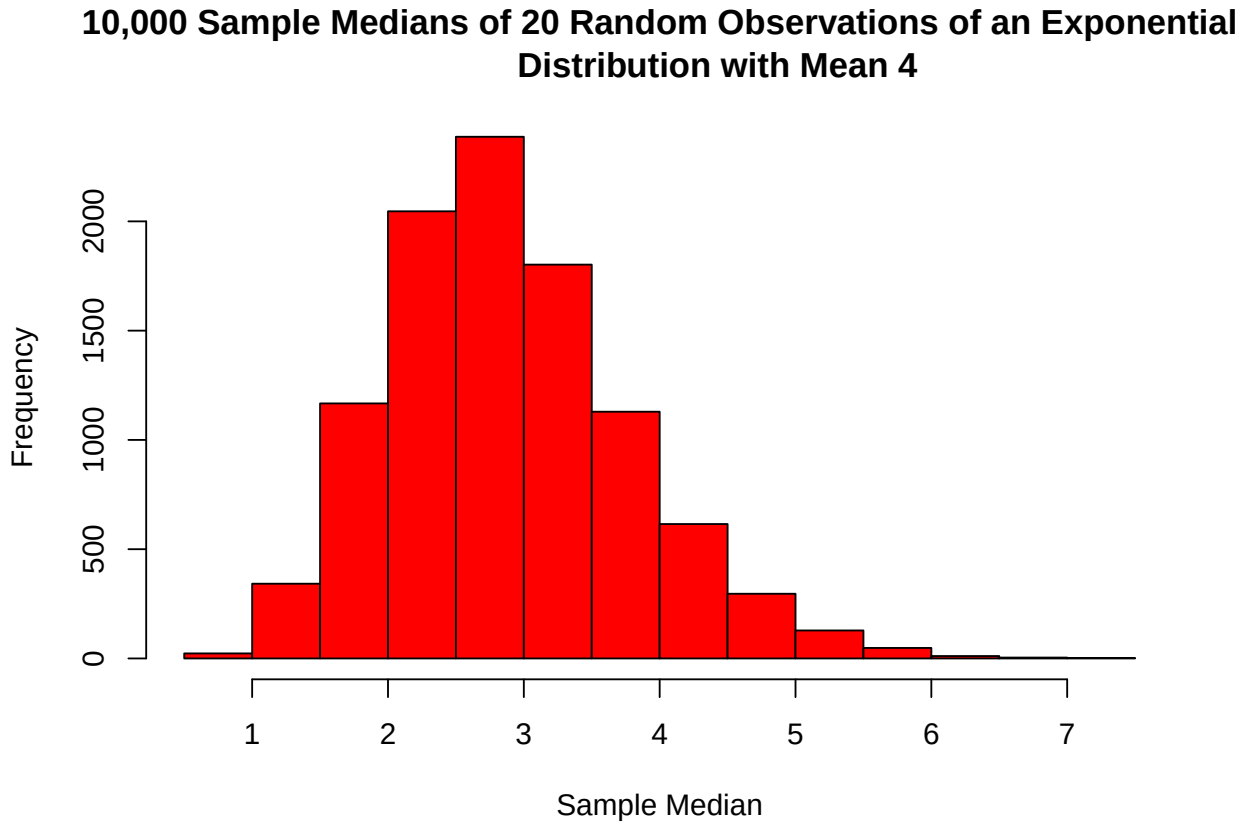
*Based on the normal quantile plot, the CLT does seem to apply to our sample means since the majority of the sample means, specifically those between -2 and 2 standard deviations, seems to lie within the 95% confidence interval of the normal quantile plot. The mean of the sample means is 4, which is what we would expect.*

*The standard deviation of the sample means does match what we would expect it to be, which is the standard deviation of the exponential distribution, 4, divided by the square root of the number of observations, 20.*

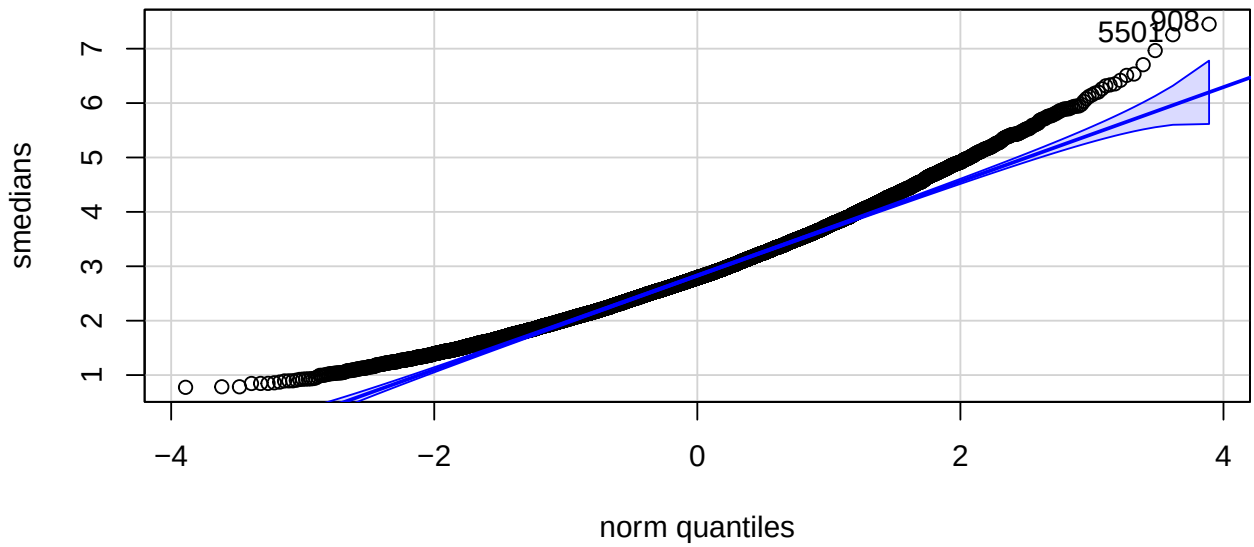
(2.5) *(7 points)*

- Create a sample histogram of the sample MEDIANS (make the bars red, make sure you label your axes and put on a clear title).
- Make a normal quantile plot of the sample medians using the `qqPlot()` function in the `car` package. Do the medians seem normally distributed?
- Display summary statistics OF THE SAMPLE MEDIANS. Is the median of the sample medians the value you expect?
- Calculate and display the sample standard deviation of the sample medians and display rounded to two decimal places. Is this value similar to the value you would expect? *Note that (despite what AI may tell you - I was given an incorrect answer twice), the standard deviation of the sample median is the same as the standard deviation of the sample mean.*

```
hist(smedians,
     col="red",
     xlab="Sample Median",
     main="10,000 Sample Medians of 20 Random Observations of an Exponential
           Distribution with Mean 4")
```




```
par(mar = c(10, 4, 4, 0))
qqPlot(smedians)
```




```
## [1]  908 5501
```

```
summary(smedians)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7747  2.2452  2.7884  2.8817  3.4133  7.4508
```

```
round(sd(smedians), digits=2)
```

```
## [1] 0.88
```

*I would say that the sample medians do look normally distributed, although I am a little hesitant due to the fact that as the sample medians approach +/- 2 standard deviations, the medians leave the 95% confidence interval.*

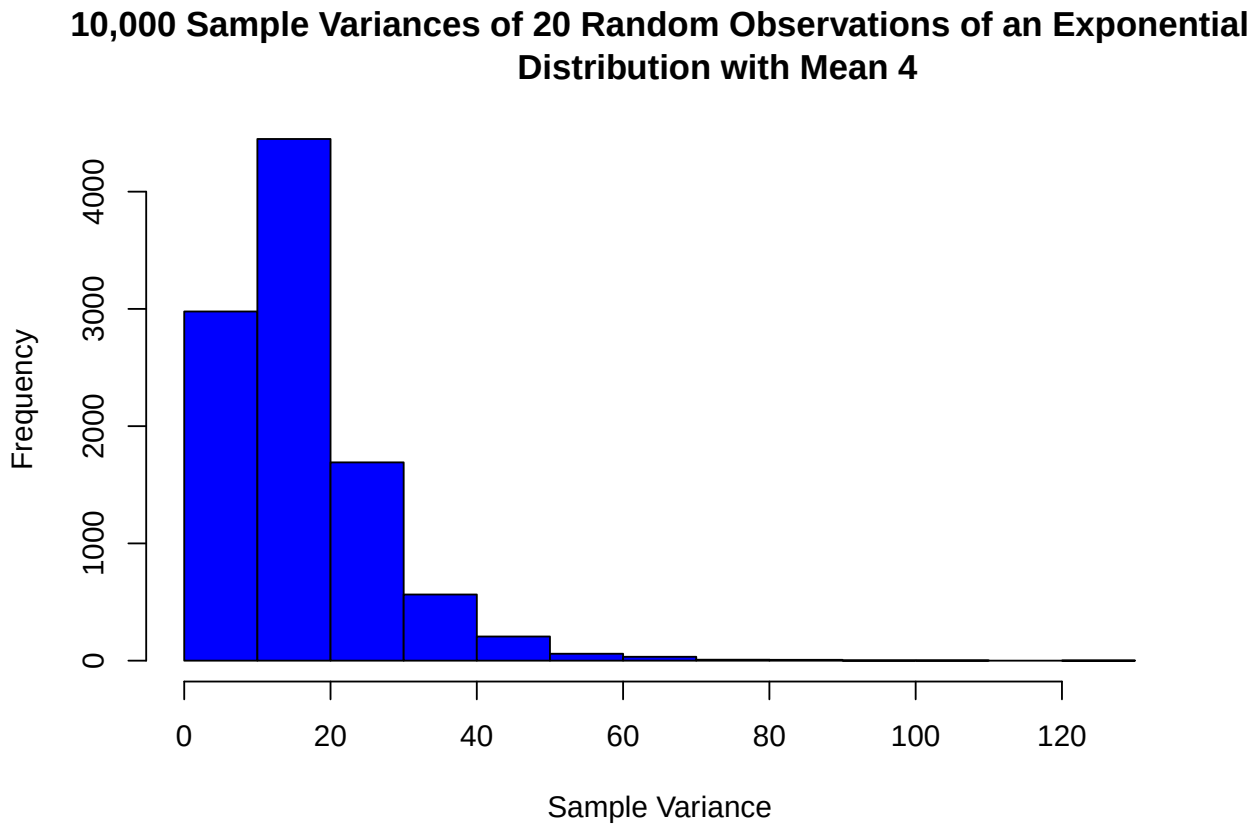
*The median of the sample medians, 2.7884, matches the theoretical median of the exponential distribution which is 2.77. The standard deviation of the sample medians also matches the standard deviation of the sample means, 0.89, which is what we would expect.*

(2.6) *(6 points)*

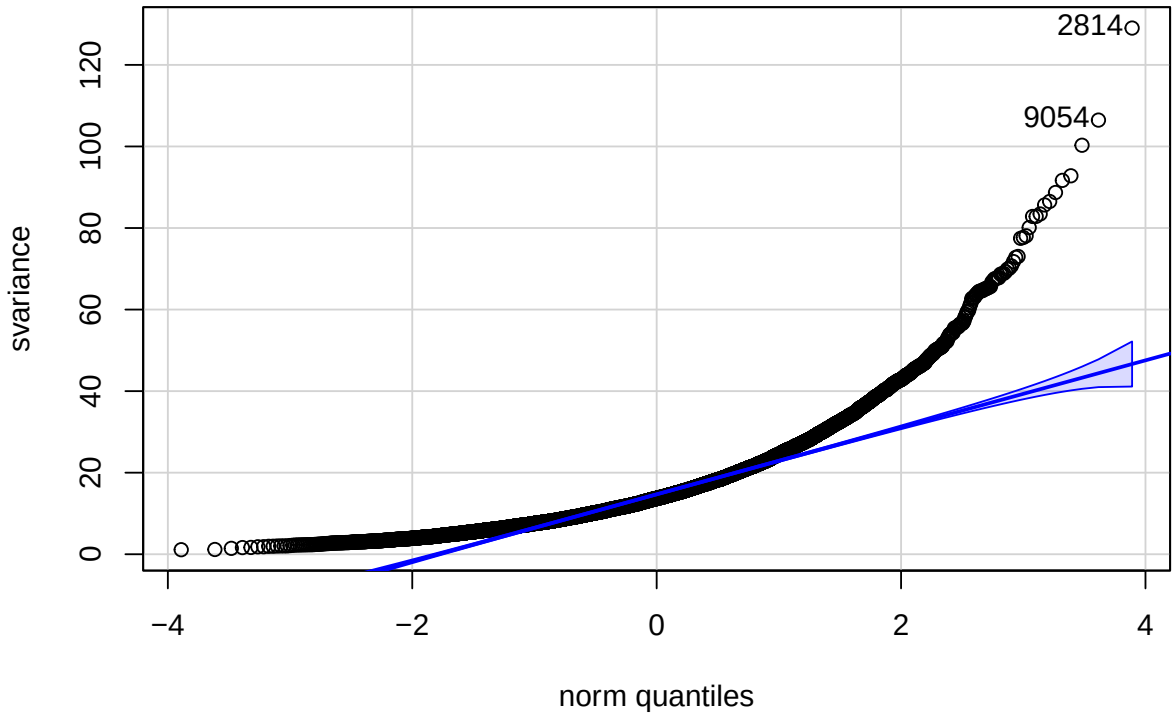
- Create a sample histogram of the sample VARIANCES (make the bars blue, make sure you label your axes and put on a clear title).

- Make a normal quantile plot of the sample VARIANCES using the `qqPlot()` function in the `car` package. Do the variances seem normally distributed?
- Display summary statistics OF THE SAMPLE VARIANCES Is the mean of the sample variances the value you expect?

```
hist(svariance,
     col="blue",
     xlab="Sample Variance",
     main="10,000 Sample Variances of 20 Random Observations of an Exponential
           Distribution with Mean 4")
```




```
qqPlot(svariance)
```




```
## [1] 2814 9054
```

```
summary(svariance)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.108   9.149  13.697  16.087  20.230 129.035
```

*The sample variances do not look normally distributed as they do not follow a linear relationship with the normal quantile values. However, the mean of the sample variances, 16.087, is close to 16, the theoritical variance of the exponential distribution.*

**(3) Iron Concentration and the Bootstrap** *(50 points, 5 points each part, parts 3.5 and 3.8 count double.*

This problem examines results of a study looking at iron concentration in the the water runoff of watersheds with different mining histories above different rock types. The data is HERE. The variables are `Rock`, `Mine`, `Iron` (concentration) and `LogIron` which is the natural log of iron concentration.

(3.1) Read the data into an object called `iron`. Show the first 6 rows of the data. Show the unique values of `Rock` and `Mine`. Then modify `iron` so that it only contains data for Unmined locations. How many observations remain? How many remaining observations do you have for each rock type? Incidentally, the rock types are Limestone and Sandstone.

```r
iron <- read.csv("http://reuningscherer.net/S&DS230/data/ironminerock.csv")
head(iron, 6)
```

```
##   Rock    Mine Iron LogIron
## 1 Sand Unmined 0.20   -1.61
## 2 Sand Unmined 0.25   -1.39
## 3 Sand Unmined 0.04   -3.22
## 4 Sand Unmined 0.06   -2.81
## 5 Sand Unmined 1.20    0.18
## 6 Sand Unmined 0.30   -1.20
```

```r
unique(iron$Rock)
```

```
## [1] "Sand" "Lime"
```

```r
unique(iron$Mine)
```

```
## [1] "Unmined" "Reclaim" "Aband"
```

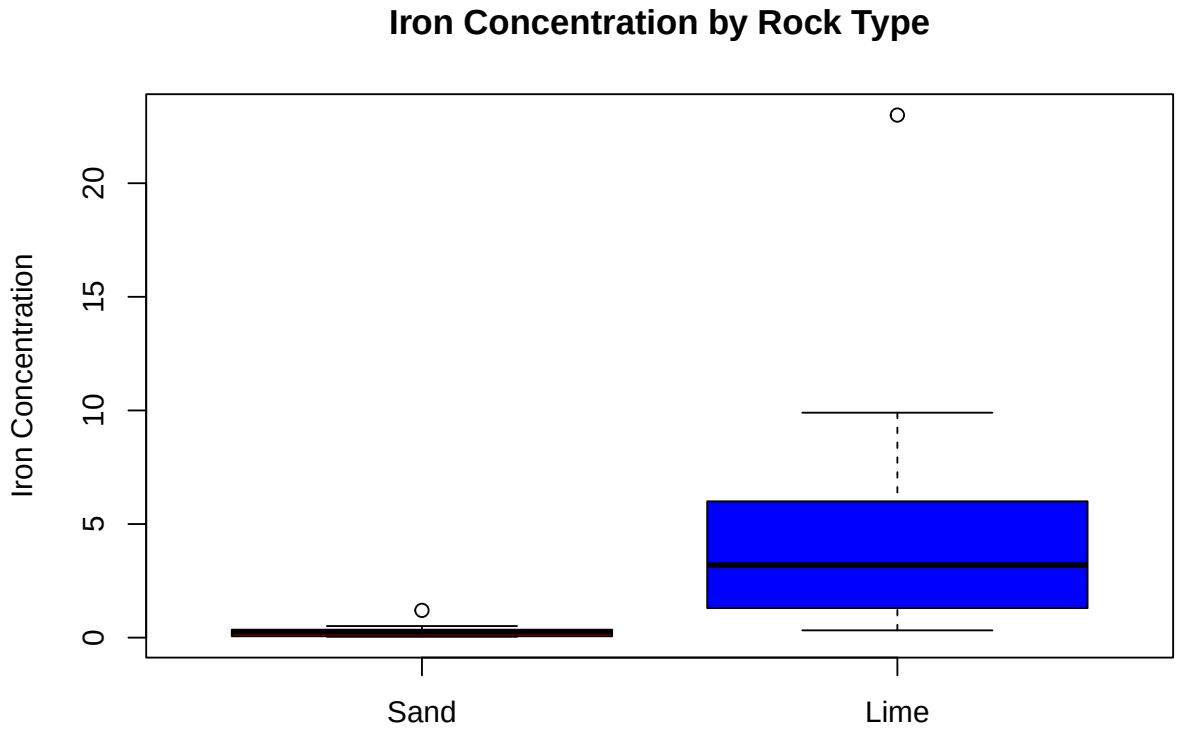
```r
iron <- iron[iron$Mine == "Unmined",]
iron
```

```
##    Rock    Mine  Iron LogIron
## 1  Sand Unmined  0.20   -1.61
## 2  Sand Unmined  0.25   -1.39
## 3  Sand Unmined  0.04   -3.22
## 4  Sand Unmined  0.06   -2.81
## 5  Sand Unmined  1.20    0.18
## 6  Sand Unmined  0.30   -1.20
## 7  Sand Unmined  0.05   -3.00
## 8  Sand Unmined  0.35   -1.05
## 9  Sand Unmined  0.27   -1.31
## 10 Sand Unmined  0.51   -0.67
## 11 Sand Unmined  0.41   -0.89
## 12 Sand Unmined  0.20   -1.61
## 13 Sand Unmined  0.04   -3.22
## 40 Lime Unmined  1.60    0.47
## 41 Lime Unmined  0.32   -1.14
## 42 Lime Unmined  5.60    1.72
## 43 Lime Unmined  0.87   -0.14
## 44 Lime Unmined  2.30    0.83
## 45 Lime Unmined  3.20    1.16
## 46 Lime Unmined  6.60    1.89
## 47 Lime Unmined  9.90    2.29
## 48 Lime Unmined  0.52   -0.65
## 49 Lime Unmined 23.00    3.14
## 50 Lime Unmined  1.30    0.26
## 51 Lime Unmined  6.00    1.79
## 52 Lime Unmined  6.00    1.79
```

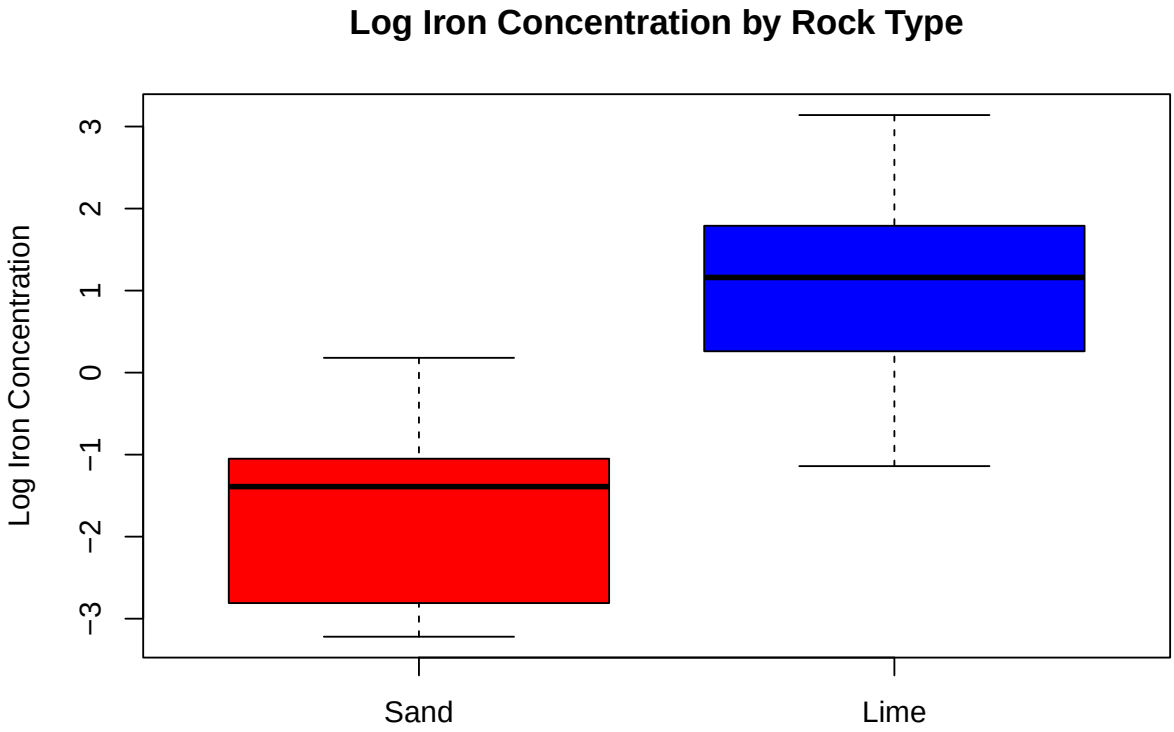
*There are 26 observations for unmined locations. 13 of them are sand and 13 are lime.*

(3.2) Make side by side boxplots of iron concentration by rock type. Make this same plot for the natural log of the iron concentration. Write a sentence or two about what you observe. Which scale do you prefer?

```
boxplot(iron[iron$Rock == "Sand",]$Iron,
        iron[iron$Rock == "Lime",]$Iron,
        col=c("red", "blue"),
        names=c("Sand", "Lime"),
        ylab="Iron Concentration",
        main="Iron Concentration by Rock Type")
```




```
boxplot(iron[iron$Rock == "Sand",]$LogIron,
        iron[iron$Rock == "Lime",]$LogIron,
        col=c("red", "blue"),
        names=c("Sand", "Lime"),
        ylab="Log Iron Concentration",
        main="Log Iron Concentration by Rock Type")
```




*The iron concentration boxplot for the sand is incredibly thin because there are much larger data poitns for the lime. However, this problem doesn't exist for the log iron concentration boxplot since the log concentration values are much closer together. I prefer the log scale.*

(3.3) Calculate summary statistics for iron concentration by treatment on the raw scale and the log scale.

```
by(iron$Iron, iron$Rock, summary)
```

```
## iron$Rock: Lime
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.32    1.30    3.20    5.17    6.00   23.00
## ------------------------------------------------------------
## iron$Rock: Sand
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0400  0.0600  0.2500  0.2985  0.3500  1.2000
```

```
by(iron$LogIron, iron$Rock, summary)
```

```
## iron$Rock: Lime
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -1.140   0.260   1.160   1.032   1.790   3.140
## ------------------------------------------------------------
## iron$Rock: Sand
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -3.220  -2.810  -1.390  -1.677  -1.050   0.180
```

(3.4) Calculate a two-sample t-test comparing mean log iron concentration between treatments. Save results in an object called `test1` and display the results. Use alpha = .01 (i.e. make a 99% CI). Is there evidence of a difference between rock types?

Create an object called 'test2' that has similar results for the raw iron concentration. Is there evidence of a difference between rock types?

```
test1 <- t.test(iron$LogIron
                ~ iron$Rock,
                conf.level = 0.99)
test1
```

```
##
##  Welch Two Sample t-test
##
## data:  iron$LogIron by iron$Rock
## t = 5.9803, df = 23.524, p-value = 3.88e-06
## alternative hypothesis: true difference in means between group Lime and group Sand is not equal to 0
## 99 percent confidence interval:
##  1.439543 3.977381
## sample estimates:
## mean in group Lime mean in group Sand
##           1.031538          -1.676923
```

```
test2 <- t.test(iron$Iron
                ~ iron$Rock,
                conf.level = 0.99)
test2
```

```
##
##  Welch Two Sample t-test
##
## data:  iron$Iron by iron$Rock
## t = 2.873, df = 12.062, p-value = 0.01395
## alternative hypothesis: true difference in means between group Lime and group Sand is not equal to 0
## 99 percent confidence interval:
##  -0.3029436 10.0460206
## sample estimates:
## mean in group Lime mean in group Sand
##          5.1700000          0.2984615
```

*Yes, there is evidence that there is a difference in log iron concentration between rock types since the p value of the 2 sample t test was 3.88 x 10^-6 which is less than 0.01. For the raw iron concentration, the p value of the 2 sample t test was 0.01395, which is small enough to reject the null hypothesis at 95% confidence, but not necessarily for 99% confidence.*

(3.5) Get 10,000 boostrap samples from the data and calculate the difference the in mean log iron concentration between rock types for each bootstrap sample. Save these means in an object called `diffLogIron`. In your code, use variable names that make sense.

```
# To make grading easier, please leave the following line of code in your assignment
set.seed(230)
smeandiffs <- c()
for (i in 1:10000) {
```

```
  s <- iron[sample(nrow(iron), 26, replace = TRUE),]
  sand_mean <- mean(s[s$Rock == "Sand",]$LogIron)
  lime_mean <- mean(s[s$Rock == "Lime",]$LogIron)
  smeandiffs[i] <- lime_mean - sand_mean
}
```

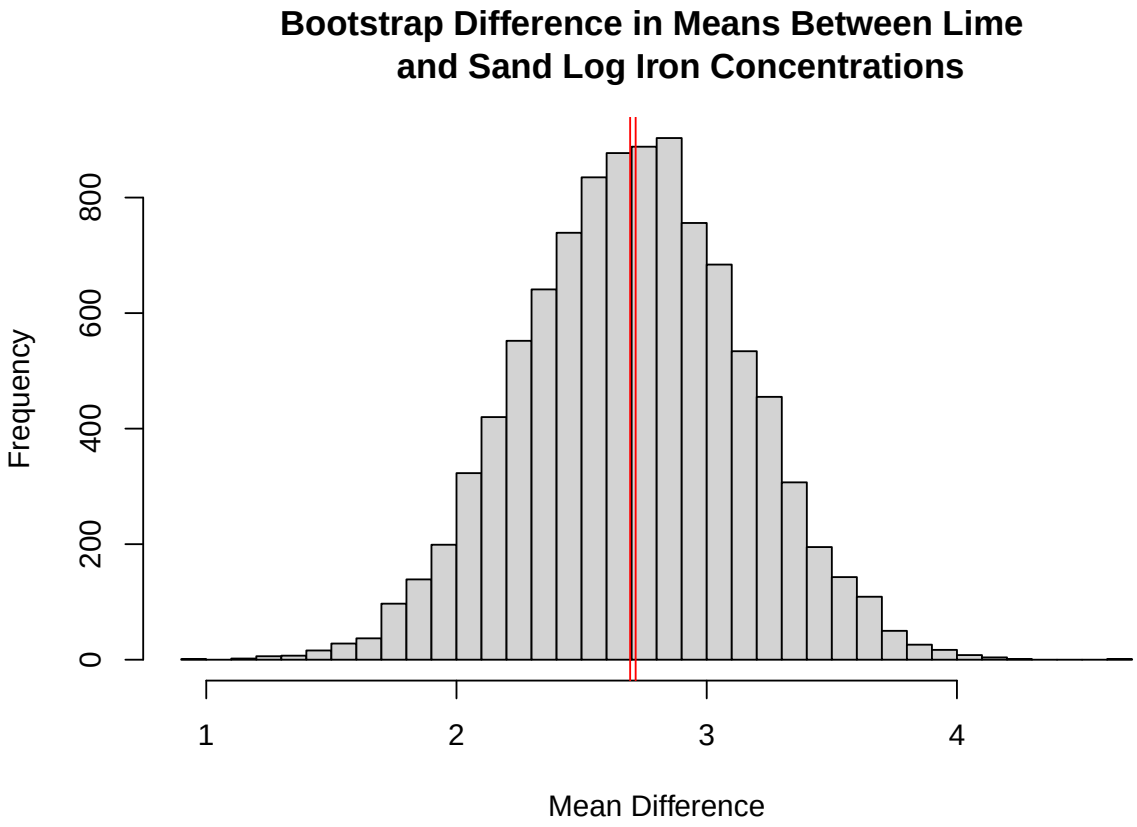
(3.6) Calculate a 99% Bootstrap confidence interval. Show your results.

```
test3 <- t.test(smeandiffs, conf.level = 0.99)
test3 # 99% confidence interval: [2.694, 2.716]
```

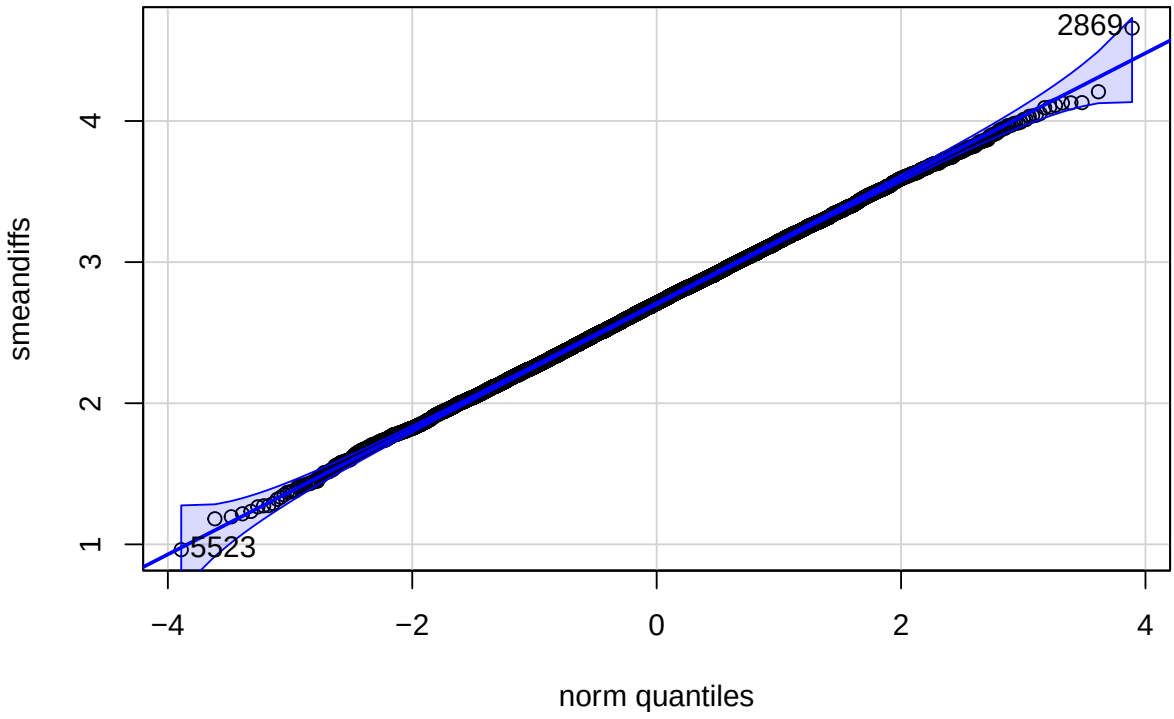
```
##
##  One Sample t-test
##
## data:  smeandiffs
## t = 614.69, df = 9999, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 99 percent confidence interval:
##  2.693813 2.716490
## sample estimates:
## mean of x
##  2.705152
```

(3.7) Make a histogram of bootstrap differences in means and add vertical lines for the theoretical and bootstrapped confidence intervals. Make a normal quantile plot of the bootstrapped differences. Discuss what you observe in both plots.

```
hist(smeandiffs,
     xlab="Mean Difference",
     main="Bootstrap Difference in Means Between Lime
     and Sand Log Iron Concentrations",
     breaks = 30)
abline(v=2.694, col="red", lwd=1)
abline(v=2.716, col="red", lwd=1)
```




```
qqPlot(smeandiffs)
```




```
## [1] 2869 5523
```

*The histogram of the bootstrap differences of the means looks normally distributed, and the normal quantile plot supports this since the majority of the points lie in the 95% confidence interval of the quantile plot.*

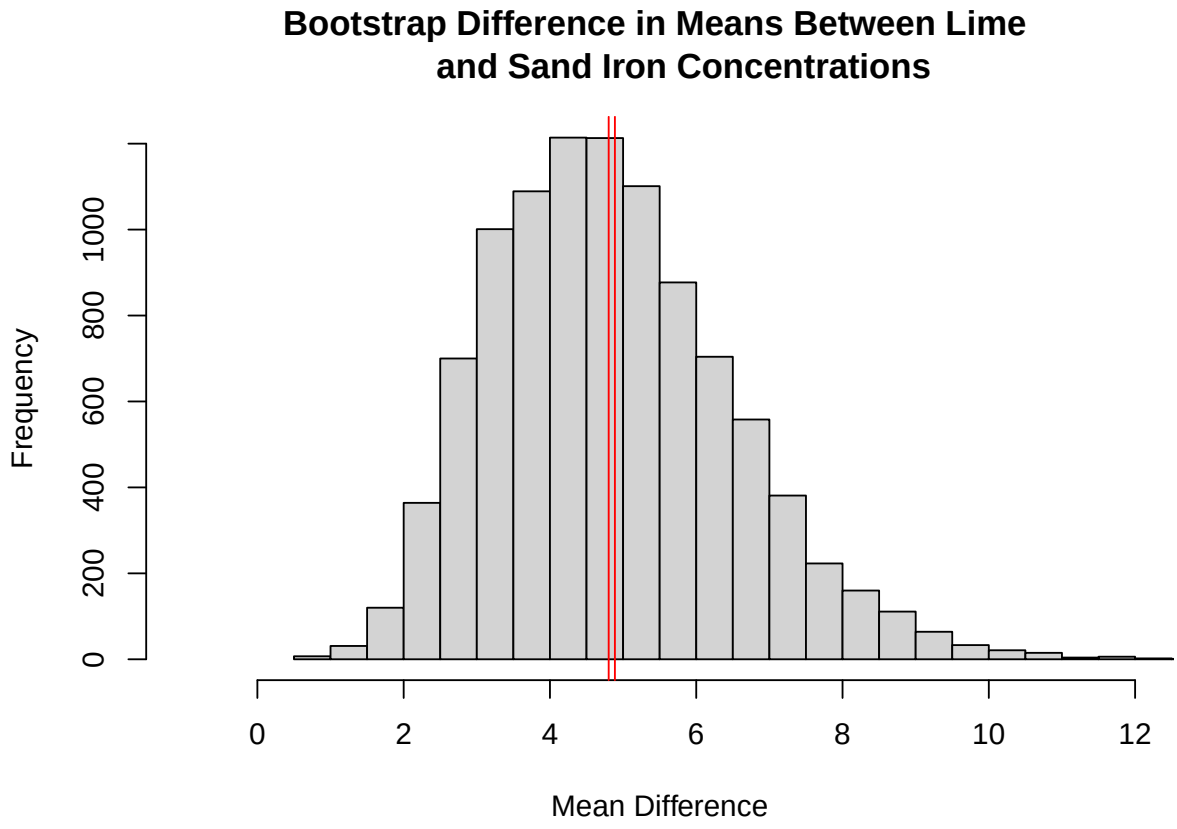
3.8) Finally, repeat 3.5 through 3.7 for iron concentrations on the raw scale. Discuss your results. Note that you'll need to add the option `xlim = c(-1, 12)` to your histogram.

```
set.seed(230)
smeandiffs2 <- c()
for (i in 1:10000) {
  s <- iron[sample(nrow(iron), 26, replace = TRUE),]
  sand_mean <- mean(s[s$Rock == "Sand",]$Iron)
  lime_mean <- mean(s[s$Rock == "Lime",]$Iron)
  smeandiffs2[i] <- lime_mean - sand_mean
}
test4 <- t.test(smeandiffs2, conf.level = 0.99)
test4 # 99% confidence interval: [4.803, 4.888]
```

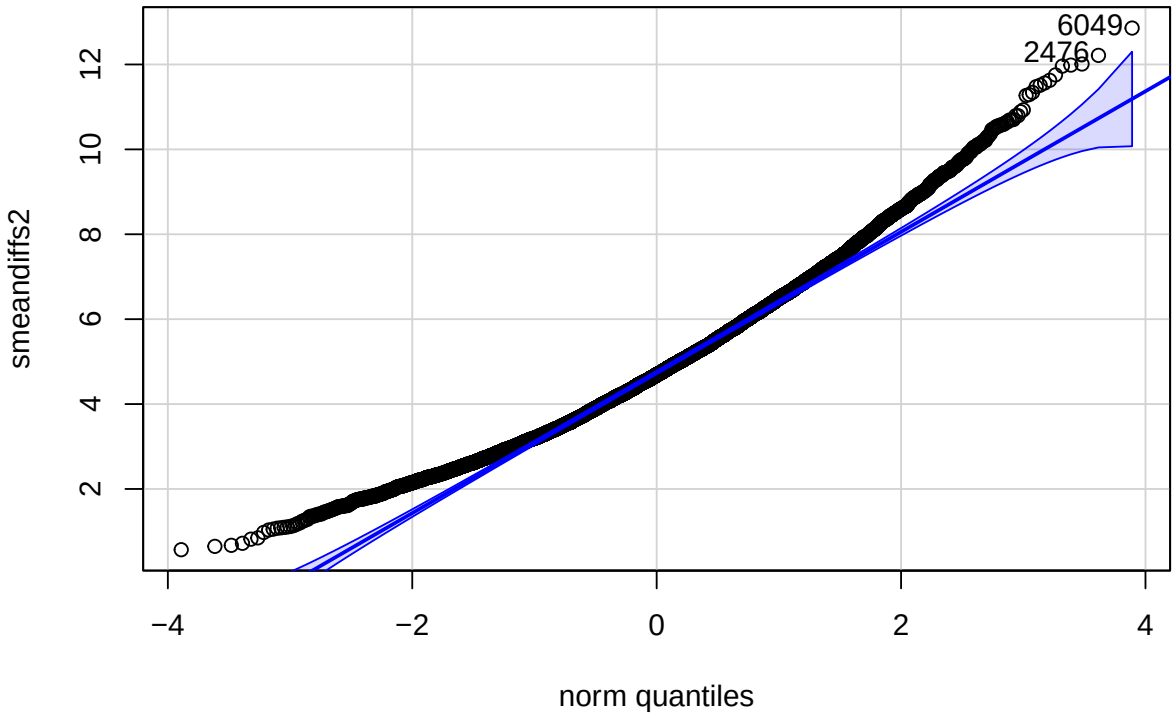
```
##
##  One Sample t-test
##
## data:  smeandiffs2
## t = 294.63, df = 9999, p-value < 2.2e-16
```

```
## alternative hypothesis: true mean is not equal to 0
## 99 percent confidence interval:
##  4.803381 4.888127
## sample estimates:
## mean of x
##  4.845754
```

```
hist(smeandiffs2,
     xlab="Mean Difference",
     main="Bootstrap Difference in Means Between Lime
     and Sand Iron Concentrations",
     breaks = 30,
     xlim = c(-1, 12))
abline(v=4.803, col="red", lwd=1)
abline(v=4.888, col="red", lwd=1)
```




```
qqPlot(smeandiffs2)
```




```
## [1] 6049 2476
```

*The distribution of the difference in means between the sand and lime iron concentrations appears to be close to normally distributed but with a right skew. The normal quantile plot supports this as most of the points closer to zero standard deviations are within the 95% confidence interval, but those that are 2 or more standard deviations away are not.*