# Assignment 4

2023-05-10

In this assignment, we are going to analyze some sample data collected by a researcher named Tammy.

Please do the following exercises writing and/or running the given R scripts, and answers the questions asked as you proceed.

## Introduction

In some parts of the U.S., black cherry trees are an important agricultural crop. Tammy is a forestry researcher who studies cherry tree diseases. She hypothesizes that cherry trees in the Allegheny Mountains of West Virginia grow to 76 ft. on average, with a standard deviation of about 6 ft.

Furthermore, she believes that the wild black cherry tree heights are normally distributed. She randomly measured the height (in feet) of 1000 black cherry trees in the Jefferson National Forest (JNF) of West Virginia to test her hypotheses.

Clear the memory.

```
rm(list = ls())
```

Make the packages "dplyr" and "ggplot2"and their associated functions available to use. Do not forget to run these functions.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(ggfortify)
library(readr)
```

In the space below, please enter the data set "cherry_trees.csv" and view it.

Do not forget to run (execute) the script! Please assign this data frame,"cherry_trees.csv", to the object "cherry_trees".

```
library(readr)
cherry_trees <- read_csv("cherry_trees.csv")
```

```
## Rows: 1000 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (2): TreeNumber, Treeheight
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# View(cherry_trees)
```

## Part 1

First, as always, we will explore and plot the data.Notice that this is a large amount of data; 1,000 trees.Examining this data by eye is too difficult and time consuming. Even if we did explore it that way, we may miss something anyway.

## Explore the data.

Explore what is in the data using what we learned in Tutorial 2. Run the following scripts. Then in the space below, briefly describe in 1 - 2 well written sentences the data set using its name, data type, its number of rows and columns, column headings, and the number of data items in the data set. You might start with:

```
mode(cherry_trees)   # Show the data type
```

```
## [1] "list"
```

```
names(cherry_trees)   # Show column names
```

```
## [1] "TreeNumber" "Treeheight"
```

```
head(cherry_trees)   # Show first 6 rows of data
```

```
## # A tibble: 6 x 2
##   TreeNumber Treeheight
##        <dbl>      <dbl>
## 1          1       77.7
## 2          2       77.0
## 3          3       76.6
## 4          4       76.6
## 5          5       82.0
## 6          6       73.6
```

```
dim(cherry_trees)   # Show number of rows and columns
```

```
## [1] 1000    2
```

```
str(cherry_trees)   # Show structure of data set
```

```
## spc_tbl_ [1,000 x 2] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ TreeNumber: num [1:1000] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Treeheight: num [1:1000] 77.7 77 76.6 76.6 82 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   TreeNumber = col_double(),
##   ..   Treeheight = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
tail(cherry_trees)   # Show last 6 rows of data
```

```
## # A tibble: 6 x 2
##   TreeNumber Treeheight
##        <dbl>      <dbl>
## 1        995       80.4
## 2        996       73.3
## 3        997       65.4
## 4        998       66.8
## 5        999       78.0
## 6       1000       74.0
```

```
glimpse(cherry_trees)   # Shows a horizontal view of the data
```

```
## Rows: 1,000
## Columns: 2
## $ TreeNumber <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ~
## $ Treeheight <dbl> 77.65790, 76.96107, 76.56895, 76.61958, 81.98007, 73.57565,~
```

```
tbl_df(cherry_trees)  # Shows a vertical view of the data
```

```
## Warning: `tbl_df()` was deprecated in dplyr 1.0.0.
## i Please use `tibble::as_tibble()` instead.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## # A tibble: 1,000 x 2
##    TreeNumber Treeheight
##         <dbl>      <dbl>
##  1          1       77.7
##  2          2       77.0
##  3          3       76.6
##  4          4       76.6
##  5          5       82.0
##  6          6       73.6
##  7          7       74.4
##  8          8       90.7
##  9          9       84.3
## 10         10       79.8
## # i 990 more rows
```

The data frame consists of 1000 entries of tree heights. There are 1000 rows of float data points describing tree height, indexed by integer observation number.

### Q1. What do the 1000 observations in the data frame on cherry_trees consist of?

The dataframe consists of 1000 observations of tree heights.

### Q2 Suggest two (2) plots that we might use to explore the data for its approximate distributional form and two plots that we may use to check for outliers.

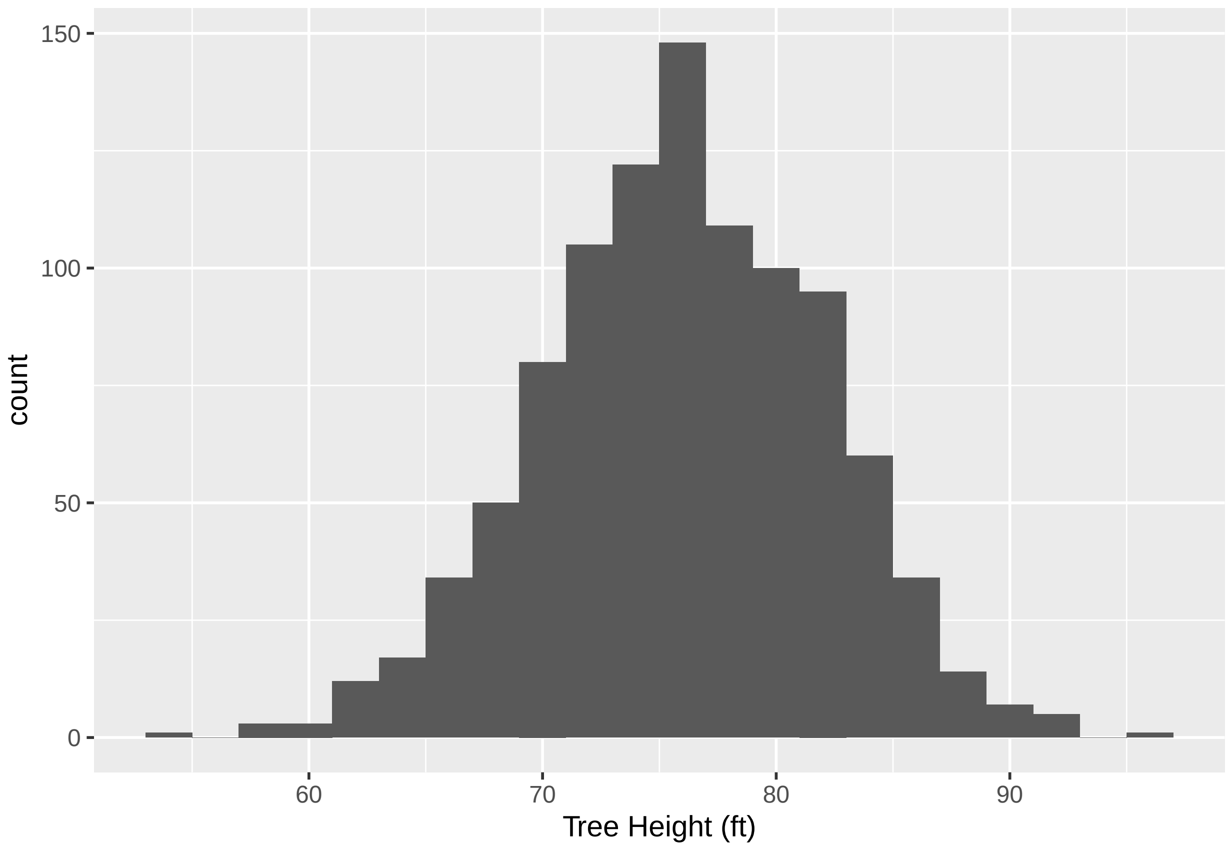
1.) Distribution Form:

a.) Histogram b.0 Cumulative Distribution Function

2.) Outliers a.) QQ plot b.) IQR boxplot

The script below creates and prints a quick histogram.

```
bin <- 50   # Set the number of bins
bw <- 2   # Set the binwidth
print(Height_histogram <-
        ggplot(data = cherry_trees, aes(x=Treeheight)) +
        geom_histogram(bins = bin, binwidth = bw) +
        scale_x_continuous(name = "Tree Height (ft)"))
```



The more complicated script below creates and prints the same histogram while overlaying a normal distribution on it. From the code, can you tell what the mean and standard deviation of the overlaid normal distribution are?

```
summary(cherry_trees$Treeheight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   54.35   71.96   76.10   75.97   80.22   95.82
```

```
mean(cherry_trees$Treeheight)
```

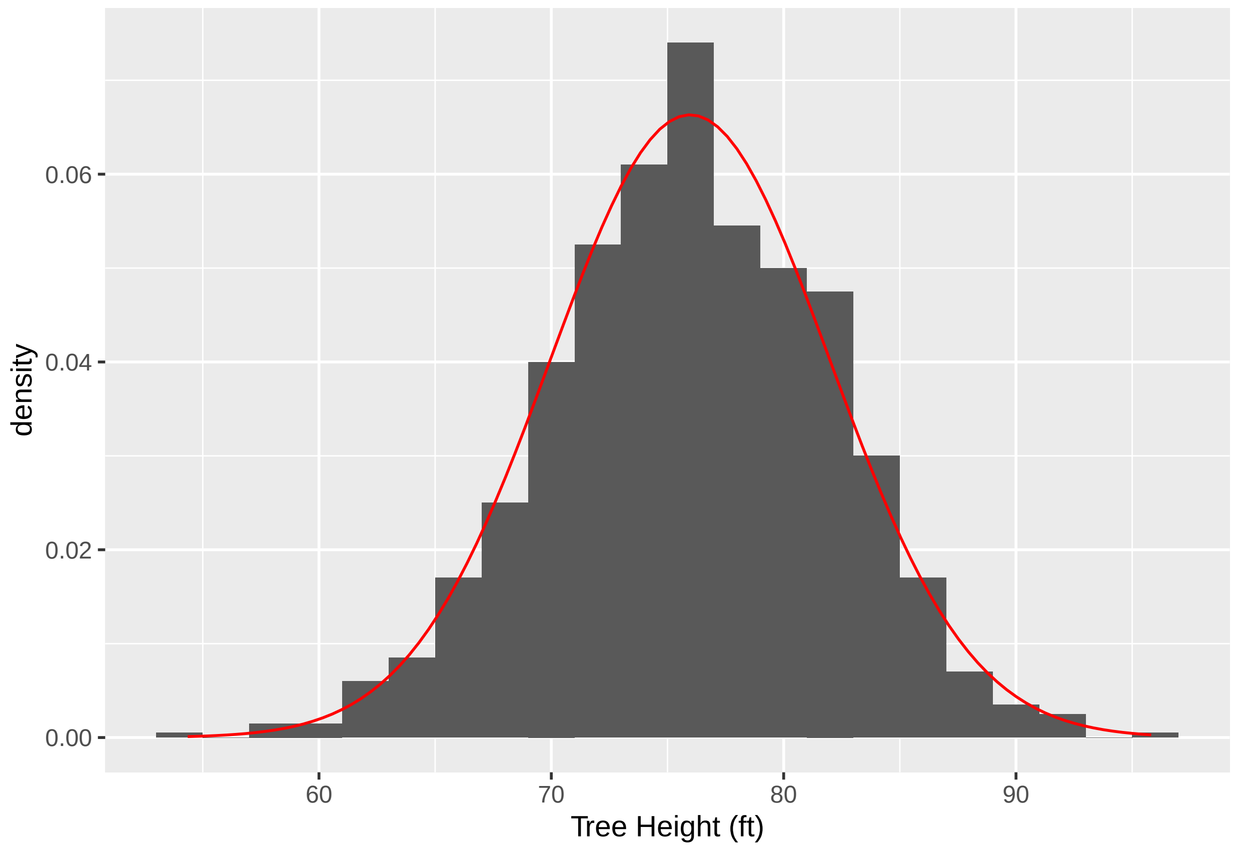
```
## [1] 75.97134
```

```
bin = 50  # Set the number of bins
bw = 2    # Set the binwidth

n_obs <- sum(!is.na(cherry_trees$Treeheight))  # Find the number of observations
n_obs
```

```
## [1] 1000
```

```
Height_histogram <-
  ggplot(data = cherry_trees, aes(x=Treeheight)) +
  geom_histogram(aes(y = ..density..), binwidth = bw) +
  stat_function(fun = dnorm, args = list(mean = mean(cherry_trees$Treeheight),
              sd = sd(cherry_trees$Treeheight)),color="red") +
  scale_x_continuous(name = "Tree Height (ft)")

Height_histogram    # Print the modified histogram
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

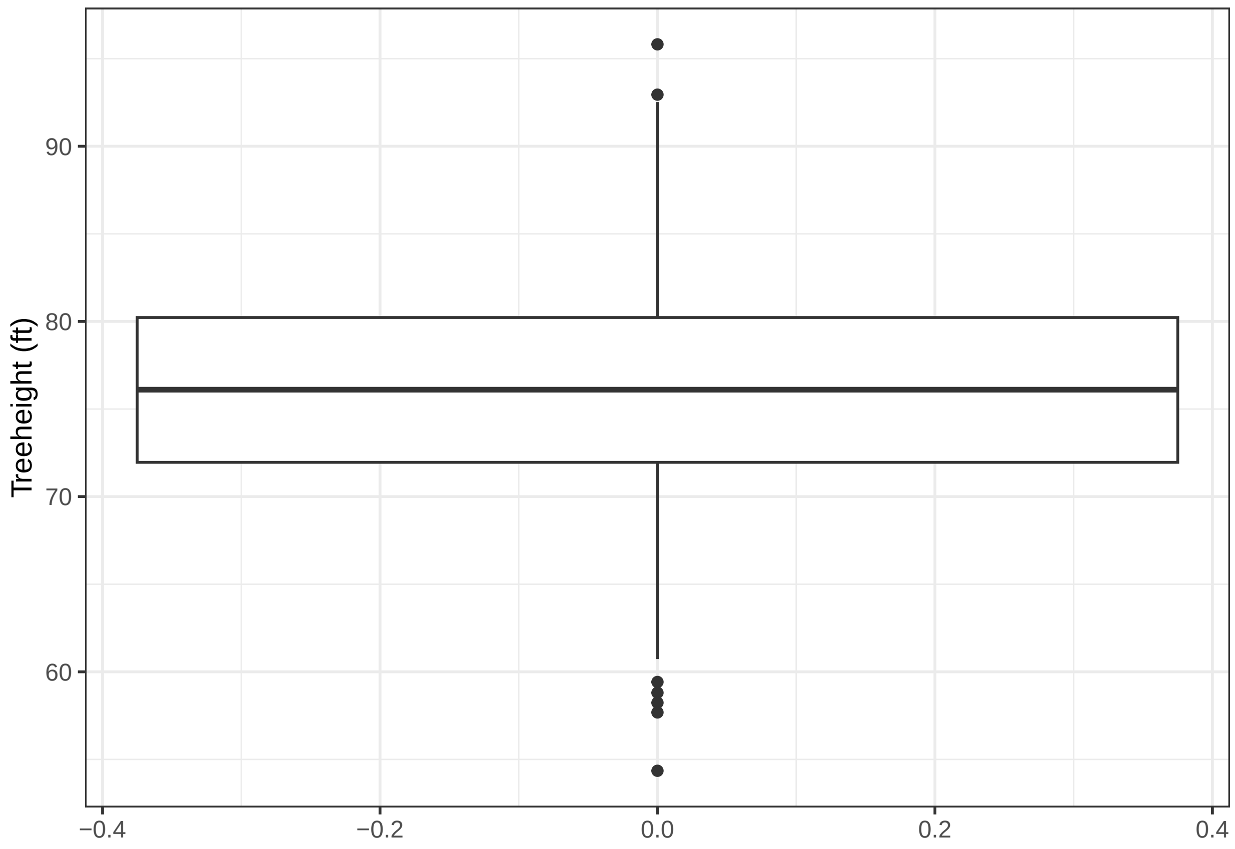


**Q3. From the histograms what do you conclude about the normality of the data? About outliers?**

The distribution appears normal with a mean and median around 75.There appear to be just a few outliers on both tails.

## Explore the data for outliers.

```
print( Height_Boxplot <- ggplot(cherry_trees, aes(y=Treeheight)) +
         geom_boxplot() +
         ylab("Treeheight (ft)") +
         theme_bw())
```



We will use a function from Assignment 2 to locate and identify potential outliers. You are welcome to create your own method for doing this.

```r
is_outlier <- function(x) {
  return(x < quantile(x,0.25) - 1.5 * IQR(x) |
           x > quantile(x, 0.75) + 1.5 * IQR(x))
}
```

Locate the outlier knowing some thing about the data set and an analysis of the box plot. Run the scripts below.Make sure to use help on the components of the following functions that you do not know.

```r
Input <- cherry_trees$Treeheight

print(outlier <- is_outlier(Input))
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [157] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [181] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [193]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [205] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [217] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [229] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [241] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [253] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [265] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [277] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [289] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [301] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [313] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [325] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [337] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [349] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [361] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [373] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [385] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [397] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [409] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
##  [421] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [433] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [445] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [457] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [469] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [481] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [493] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [505] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [517] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [529] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [541] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [553] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [565] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [577] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [589] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [601] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [613] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [625] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [637] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [649] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [661] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
##  [673] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [685] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [697] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
##  [709] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [721]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [733] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [745] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [757] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
##  [769] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [781] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [793] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [805] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [817] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [829] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [841] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [853] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [865] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [877] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [889] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [901] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [913] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [925] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [937] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [949] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [961] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [973] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [985] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [997] FALSE FALSE FALSE FALSE
```

```
print(outlier <- as.numeric(is_outlier(Input)))
```

```
##    [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [149] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [186] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [223] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [260] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [297] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [334] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [371] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [408] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [445] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [482] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [519] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [556] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [593] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [630] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [667] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [704] 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [741] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [778] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [815] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [852] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [889] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [926] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [963] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1000] 0
```

```
?as.numeric
```

The script below will identify individual data values that are larger or equal to 91 ft or (the "|" = "or") smaller than or equal to 60 feet. Trees can certainly be very short, so the second test is really needed.But this script was written to show how you might create a search based on two conditions. Please feel free to alter it to correspond to your interpretation of what appears in the boxplot.

```
Indivdual_Num <- which(Input >= 91 | Input <= 60)

# Locate the position in data set of any potential outliers.

?which()

which(is_outlier(Input))
```

```
## [1] 193 389 420 671 706 721 762
```

Locate the outlier(s) in the data set and identify its value(s) based on using the rule of thumb. This requires putting together some R functions to accomplish this, but it works very well.

```
cherry_trees[c(which(is_outlier(Input))),]
```

```
## # A tibble: 7 x 2
##   TreeNumber Treeheight
##        <dbl>      <dbl>
## 1        193       54.3
## 2        389       58.8
## 3        420       92.9
## 4        671       57.7
## 5        706       95.8
## 6        721       58.2
## 7        762       59.4
```

**Q4. What does this analysis identify as potential outliers in this data set?**

Here we used r scripts to find outliers in the data. We identified outliers by finding data points that fall one standard deviation above or below the third or first quantile, respectively.

**Q5. Please mention what you would do to review these identified data values to see if they are "good" data or represent errors. One thing would be to check how high cherry trees can grow. You may also notice that the results from the boxplot agree with what the histograms displayed. The boxplot makes it easiest to discover outliers.**

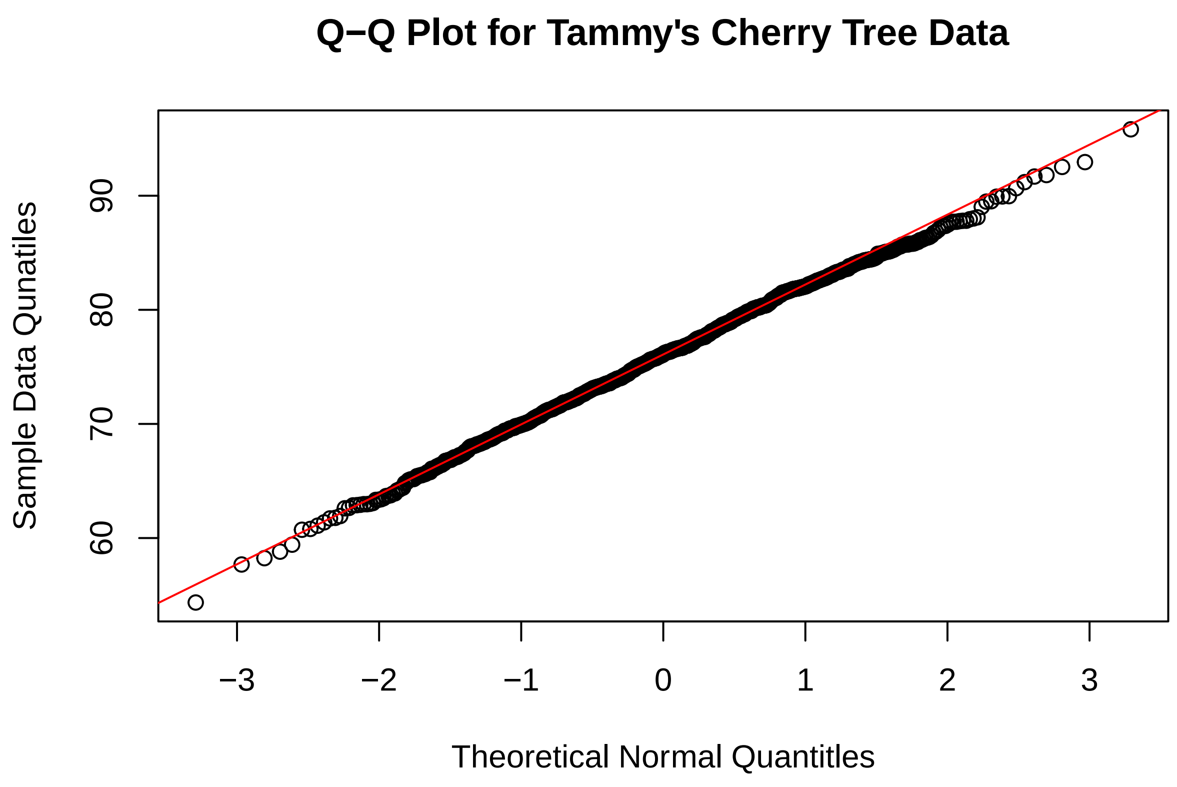
First I would check to see how high cherry trees grow, and then the age of the trees or think about the growth rate of trees and where the data was collected. Next I would think about how the data was collected and if there are potential errors in the measuring method.

As we learned in class, a helpful tool for assessing the normality of data is the Q-Q plot. Please refer to the analysis section in Assignment 3 to recall using a Q-Q plot for assessing the normality of data. In that assignment, we generated data from various probability distributions. Now we are going to use a Q-Q plot to assess the normality of experimental or sampled data.

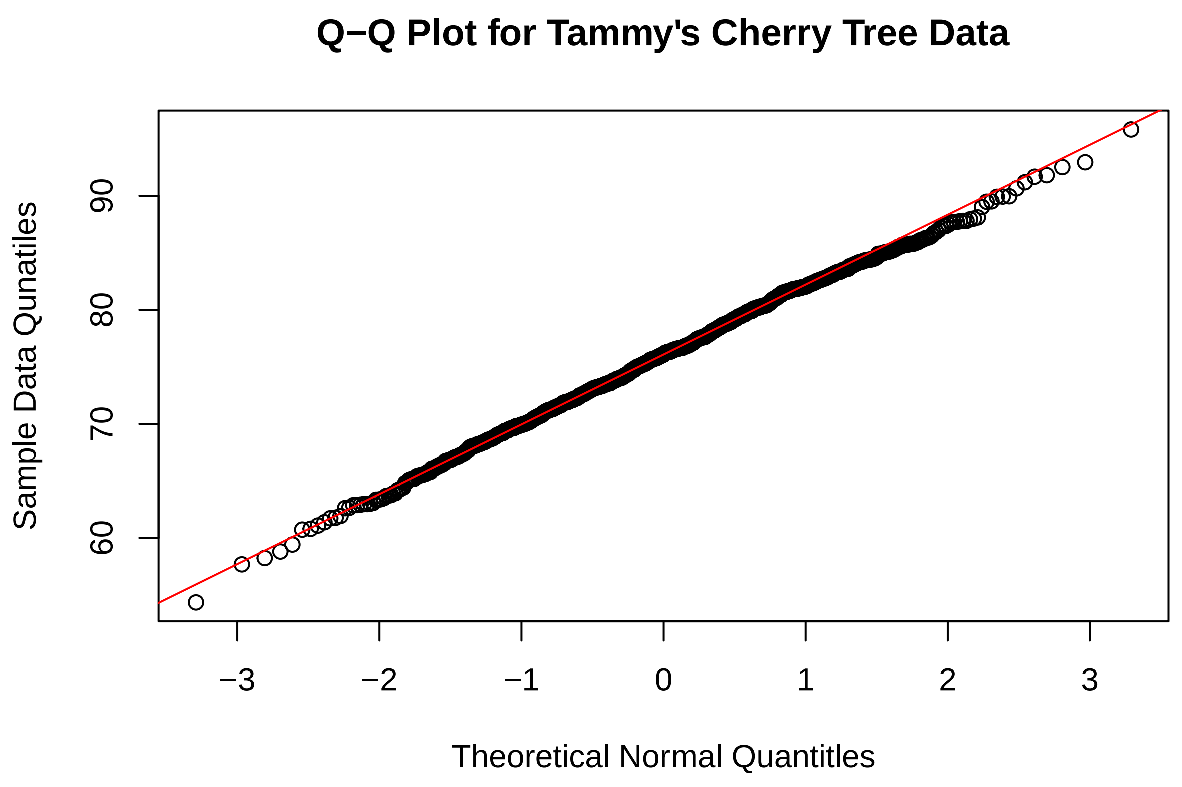
When we study the ANOVA later in the course, we will learn a set of functions to simultaneously produce four diagnostic plots to check for normality, but until then we will use the two core R functions "qqnorm()" and "qqline()" do this. We will compare the quantiles from Tammy's sample data to the quantiles for data generated from a theoretical normal distribution.

Please run the script below and answer the questions asked after it.

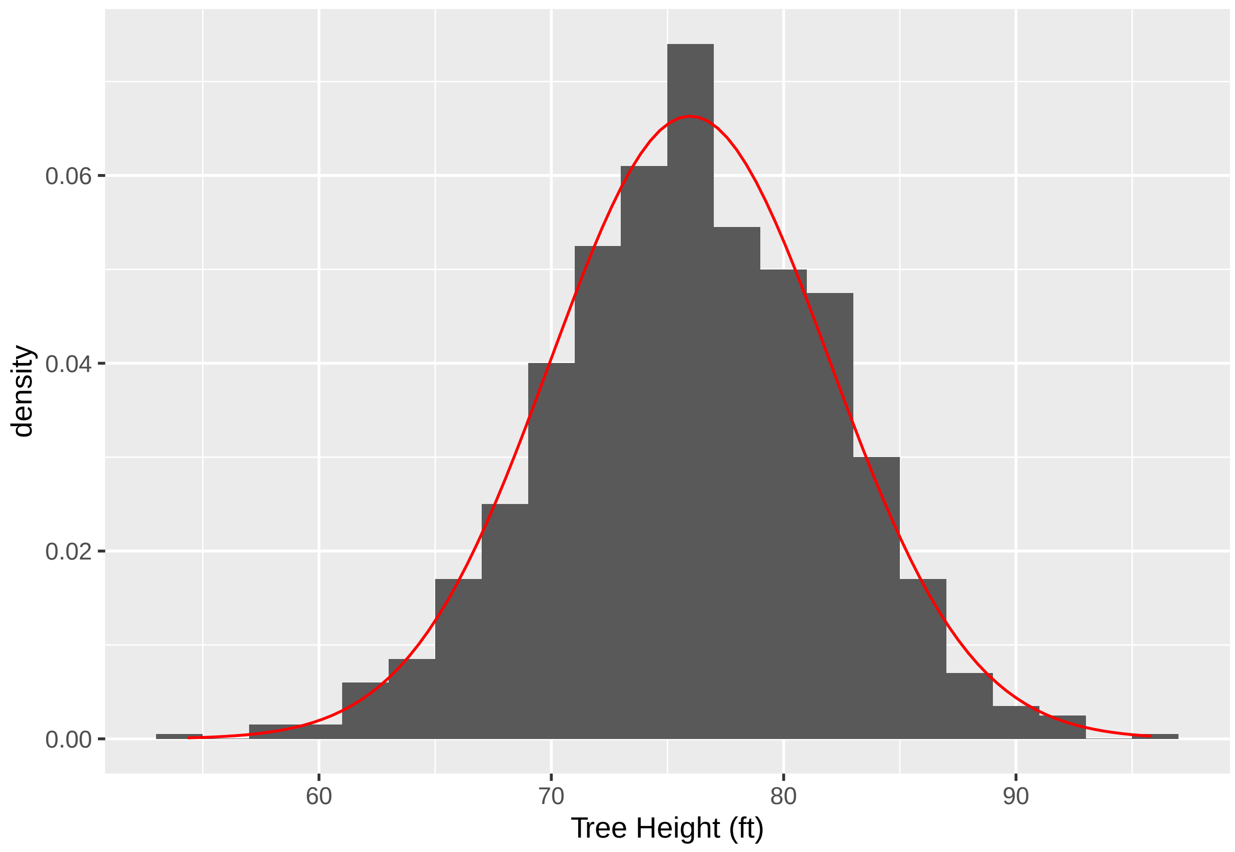
```
#  Get help.


?qqnorm()

qqnorm(Input, main="Q-Q Plot for Tammy's Cherry Tree Data",
       xlab="Theoretical Normal Quantitles",
       ylab="Sample Data Qunatiles")
qqline(Input,col = "red")
```



```
#  Plot the Q-Q plot and the histogram together.


qqnorm(Input, main="Q-Q Plot for Tammy's Cherry Tree Data",
       xlab="Theoretical Normal Quantitles",
       ylab="Sample Data Qunatiles")
qqline(Input,col = "red")
```



```
Height_histogram     # Print the modified histogram
```



Given the histogram of the data and what you know about Q-Q plots from Assignment 3 and class:

**Q6.Can we assume that these data fit the normal distribution? Use the histogram plot of your data and the Q-Q plot to back up your answer.**

Yes, we can assume this data is normally distributed. The data fits under a theoretical normal distribution, and the outliers on the tails do not deviate from theoretical quantiles.

**Q7. Do we need to use the central limit theorem in this case? If you do, what assumptions are you making?**

Yes. Assumptions made are that the sample size is large enough (n = 1000), sampling is random and samples are i.i.d., and variance is finite (it is, we can calculate it)

**Q8. Why have we gone to all of the effort checking the normality of this data? If it had turned out that the data was not at least approximately normally distributed, what theorem would you invoke?**

With normal data we can estimate statistics of the population. If this data had not turned out to be normal, we could invoke the Chebyschev inequality to apply confidence intervals to our distributions normality.

## Part 2

**Q1. What are the mean, median, variance, and standard deviation of Tammy's sample? Do not forget the units!**

We can use the following script from Assignments 2 to generate these statistics and much more. Run this script and answer the following question.

```
sumData <- cherry_trees %>%
  summarise(
    countHT = n(),   # Number of trees
    meanHT = mean(Treeheight),   # Mean
    medianHt = median(Treeheight),   # Median
    varHT = var(Treeheight),   # Variance
    sdevHT = sd(Treeheight),   # Standard deviation
    semHT = sd(Treeheight)/sqrt(n()),   # Standard error of the mean (SEM)

    # Alternative SEM if there are missing values
    sem_altHT = sd(Treeheight)/sqrt(sum(!is.na(Treeheight))),

    iqrHT = IQR(Treeheight),   # Interquartile range
    quant_25_HT = quantile(Treeheight, 0.25),   # First quartile
    quant_50_HT = quantile(Treeheight, 0.50),   # The median
    quant_75_HT = quantile(Treeheight, 0.75),   # Third quartile
    minHT = min(Treeheight),  # Minimum
    maxHT = max(Treeheight))   # Maximum
```

Display your summaries here:

```
sumData
```

```
## # A tibble: 1 x 13
##   countHT meanHT medianHt varHT sdevHT semHT sem_altHT iqrHT quant_25_HT
##     <int>  <dbl>    <dbl> <dbl>  <dbl> <dbl>     <dbl> <dbl>       <dbl>
## 1    1000   76.0     76.1  36.2   6.01 0.190     0.190  8.27        72.0
## # i 4 more variables: quant_50_HT <dbl>, quant_75_HT <dbl>, minHT <dbl>,
## #   maxHT <dbl>
```

**Q1. Do you notice anything interesting about the mean and the median? Does this make you think something about the potential normality of the data, even without having the value of the mode?**

They are very close, probably don't even need to know the mode to say about normality.

## Part 3

We will now develop confidence intervals for the mean.Please answer the following questions in the space below:

**Q1. On what do we determine confidence intervals; sample statistics or population parameters?**

population parameters

**Q2. In the situation here, what distribution would we use, the normal or Student's t, in creating confidence intervals? Why?**

Student's t, as it is more flexible with degrees of freedom.

**Q3. What are the appropriate degrees of freedom for the critical value?**

n - 1 = 999

## Create confidence intervals (CIs).

Remember that we have assigned the tree height data to the Object, "Input", as "Input <-cherry_trees$Treeheight".Wecan used this object in defining the CI. We know that we need the mean, standard deviation and number of data items to calculate the CI. Let's check them by printing them out.

```
Input <- cherry_trees$Treeheight
mean = mean(Input)
sd = sd(Input)
n = length(Input)
```

**Q1. Calculate the SEM (standard error of the mean) and assign it to the object "SEM". Print the result.**

```
SEM = sd / sqrt(n)
```

**Q2. Assign the variable (object) "Alpha95" to the appropriate alpha value for a 95% CI. Print the result.**

```
Alpha95 = 0.05
print(Alpha95)
```

```
## [1] 0.05
```

**Q3. Assign the appropriate degrees of freedom to "DOF". Hint: make appropriate use the core R function "length()". Print the result.**

```
DOF = n - 1
```

**Q4. Create a 95 % CI for Tammy's data. R gives us some easy ways to do this. One way is to use "qt()" that we mentioned in Assignment 3.**

```
#  Get help on "qt()"
?qt()
```

**Q5. Using the correct alpha and degrees of freedom, calculate and print qt(alpha/2,degrees of freedom) in the space below.**

```
print(qt(Alpha95/2, DOF))
```

```
## [1] -1.962341
```

**Q6. Using what you learned from this and the objects we have defined above, please calculate the upper and lower CI for Tammy's data. Assign the lower CI to "LowerCI_95" and the upper CI to "UpperCI_95". Report your CI interval, i.e., [LowerCI, UpperCI] after your script.**

```
library(glue)
LowerCI_95 = mean + qt(Alpha95/2, DOF)
UpperCI_95 = mean - qt(Alpha95/2, DOF)

glue('[{LowerCI_95}, {UpperCI_95}]')
```

```
## [74.0089944947666, 77.9336774170335]
```

You might suspect that there is another way of determining such CIs using Student's t-distribution. Presaging something to come later in the course,we will do just that. Run the script below and compare the results for the 95 % CI it calculates to what we calculated above. Yes, that is the R function that does what some of you may have heard of as the "t-test".

```
t.test(Input, conf.level=0.95)
```

```
##
##  One Sample t-test
##
## data:  Input
## t = 399.42, df = 999, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  75.59809 76.34459
## sample estimates:
## mean of x
##  75.97134
```

**Q7. Please calculate the 80%, 90%, and 99% CIs for Tammy's data. Do not forget to define the appropriate alpha level for each CI! In each case, print the alpha level and the lower and upper CIs, and check your calculations with the appropriate call to the function "t.test()".**

```
print(t.test(Input, conf.level=0.80))
```

```
##
##  One Sample t-test
##
## data:  Input
## t = 399.42, df = 999, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 80 percent confidence interval:
##  75.72742 76.21526
## sample estimates:
## mean of x
##  75.97134
```

```
print(t.test(Input, conf.level=0.90))
```

```
##
##  One Sample t-test
##
## data:  Input
## t = 399.42, df = 999, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 90 percent confidence interval:
##  75.65818 76.28449
## sample estimates:
## mean of x
##  75.97134
```

```
print(t.test(Input, conf.level=0.95))
```

```
##
##  One Sample t-test
##
## data:  Input
## t = 399.42, df = 999, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  75.59809 76.34459
## sample estimates:
## mean of x
##  75.97134
```

```
print(t.test(Input, conf.level=0.99))
```

```
##
##  One Sample t-test
##
## data:  Input
## t = 399.42, df = 999, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 99 percent confidence interval:
##  75.48046 76.46221
## sample estimates:
## mean of x
##  75.97134
```

**Q8. Please report the 80%, 90%, 95%, and 99% CIs intervals. You can do this easily once you have run the scripts by copying and pasting the results in the areas provided.**

80 % CI = [75.72742 76.21526]

90% CI = [75.65818 76.28449]

95% CI = [75.59809 76.34459]

99% CI = [75.48046 76.46221]

```
CONGRATULATIONS!!
```

Look at what you can now do using R. There are just a few more questions.Please continue with the assignment below.

**Q9. Explain in 1-2 sentences what the 95% confidence interval for the mean represents.**

Based on our sample data, we can say with 95% confidence that the mean of means falls within our lower and upper bound

## Part 4

Another researcher, Tom, randomly samples 50 trees in the JNF.From his sample, he calculates a mean tree height of 76.49 ft. with a standard deviation of 5.73 ft. Please answer the following questions.

**Q1. Is the mean of Tom's sample contained within any of the confidence intervals for Tammy's sample mean?**

No

**Q2. Do you think that Tammy's sample of tree heights is representative of the population of cherry trees in the entire forest? Do you think Tammy's or Tom's random sample is more representative of the population of wild black cherry trees in Allegheny Mountains of West Virginia? Why?**

Tammy's is more representative of the forest than Tom's is, since she has sampled more trees. I can't really say much more without actually looking at the data, but it isn't likely that Tom's is very good.

**Q3. Propose a sampling scheme for randomly sampling 100 wild black cherry trees from the JNF. Note: Many answers are possible.**

1.) Define a boundary to stay within. If the boundary is encountered at any time, rotate 45 degrees clockwise and continue forward. 2.) Start walking until you find a black cherry tree not yet sampled. 3.) Sample it. 4.) Turn 45 degrees clockwise and start walking. 5.) Stop at the third cherry tree you encounter, not including any tree already sampled. 6.) Repeat 2 - 6 until 100 trees are sampled.

The End of Assignment #4.