R Workshop, Day 2

Toby Trotta, Applied Research Lab

Spring 2024

Day 2: Data Analysis, Graphing, Regression, and Hypothesis Testing

Day 1 Review:

The first step I tend to do when opening a new RStudio session is immediately load the packages I need. Since we plan to import a data frame, we need **tidyverse**. We plan to also do some more data manipulation, so we need **dplyr**.

Load the libraries:

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
```

It's generally good practice to keep all loaded packages in the {r setup} chunk above at the top of your markdown documents. Not necessary, of course, but if you run that entire chunk with all packages, you dont need to run individual lines of code later on. Remember, we do not need to install these packages again! One and done. Keep in mind that these packages are updated somewhat regularly. To see which packages need updated, go to the "Packages" tab in the Files pane to the right and click "Update". You'll probably have some, but no need to do it right now.

Import the `cleanedData.csv` and file from Day 1: (You'll already have this in your environment if we load our workspace from Day 1, but just for practice, go ahead and try importing. I'm going to show you another trick:)

```
cleanedData <- read_csv('cleanedData.csv')
```

```
## Rows: 200 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Gender, Class
## dbl (3): Height, Weight, Age
##
```

```
## 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.
```

A tip for large datasets with many variables: So we don't have to go column by column changing each variable to a factor, we use the following "algorithm":

```
1) Using the console below, type spec(df). This will return a list of the variable names and types (doul
2) Copy (CTRL + C) the cols(x = ..., y = ..., etc.) output provided.
3) In the cell you used `read_csv()` in (directly above), add an additional argument after the "cleanedl
```

```
cleanedData <- read_csv('cleanedData.csv', col_types = cols(
  Height = col_double(),
  Weight = col_double(),
  Gender = col_character(), # The Gender and Class variables are imported as
                            # character data since it text (after the
                            # recoding from Day 1).
  Class = col_character()
  )
)
```

This `spec(cleanedData)` output, like mentioned, provides a list of the data types. We can manually change each of these in the cell above (col_string, col_integer, col_factor, col_float, etc.).

From Day 1, we discussed how factors work and when to use them. We also determined that Gender and Class should be factors. Now, instead of executing `cleanedData$x <- as.factor(cleanedData$x)` for each variable, we can instead import the variables as factors automatically:

```
cleanedData <- read_csv('cleanedData.csv', col_types = cols(
  Height = col_double(),
  Weight = col_double(),
  Gender = col_factor(), # Here! Changed from col_double() to col_factor
  Class = col_factor() # Same thing here.
  )
)
```

```
head(cleanedData) # Now, Class and Gender are factors!
```

```
## # A tibble: 6 x 5
##   Height Weight Gender Class      Age
##    <dbl>  <dbl> <fct>  <fct>    <dbl>
## 1   150.   167. Female Freshman    17
## 2   151.   180. Female Freshman    18
## 3   151.   222. Female Freshman    18
## 4   151.   137. Female Freshman    17
## 5   151.   223. Female Freshman    17
## 6   153.   193. Female Freshman    17
```

Descriptive Statistics:

Providing the **descriptive statistics** is a fundamental part of research; our purpose in providing such statistics (mean, standard deviation, count, skewness, etc.) is used to summarize and describe (wink wink, nudge nudge) the main features of your dataset. Example questions we can answer with descriptive statistics, using our dataset, for example, would be "What is the mean height of all students?"

Using the `describe()` function (from the `psych` library):

A *very* convenient package and function for quick descriptive statistics is `describe()` from the `psych` library. You likely won't have this, so, as practice, try installing and loading this new library:

```
install.packages('psych')
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

Within the `psych` library is `describe()`. The syntax is `df %>% describe()` or `describe(df)`.

```
cleanedData %>% describe()
```

```
##         vars   n   mean    sd median trimmed   mad    min    max  range skew
## Height     1 200 173.58 12.61 173.37  173.26 12.35 150.46 199.46  49.00 0.21
## Weight     2 200 147.72 41.78 146.95  147.18 46.47  77.15 223.24 146.09 0.12
## Gender*    3 200   1.49  0.50   1.00    1.48  0.00   1.00   2.00   1.00 0.06
## Class*     4 200   1.66  1.00   1.00    1.47  0.00   1.00   4.00   3.00 1.22
## Age        5 200  18.42  1.40  18.00   18.23  1.48  17.00  22.00   5.00 1.02
##         kurtosis   se
## Height     -0.71 0.89
## Weight     -1.07 2.95
## Gender*    -2.01 0.04
## Class*      0.09 0.07
## Age         0.25 0.10
```

Q: Do you notice anything special? What data types are we trying to find the descriptive statistics on?

Also within the `psych` library is `describeBy()`. Here, we have the option to group the variables, rather than reporting the overall mean, etc. as before:

```
cleanedData %>% describeBy(group = 'Gender')
```

```
##
##  Descriptive statistics by group
## Gender: 1
##        vars   n   mean    sd median trimmed   mad    min    max  range skew
## Height    1 103 165.19  8.48 165.07  165.23 10.86 150.46 179.76  29.30 0.00
## Weight    2 103 150.90 41.91 150.49  151.09 47.04  77.15 223.24 146.09 0.00
## Gender    3 103   1.00  0.00   1.00    1.00  0.00   1.00   1.00   0.00  NaN
## Class     4 103   1.56  0.97   1.00    1.35  0.00   1.00   4.00   3.00 1.53
## Age       5 103  18.37  1.41  18.00   18.17  1.48  17.00  22.00   5.00 1.05
##        kurtosis   se
## Height    -1.18 0.84
## Weight    -1.09 4.13
## Gender      NaN 0.00
## Class      0.99 0.10
## Age        0.18 0.14
## ------------------------------------------------------------
## Gender: 2
##        vars  n   mean    sd median trimmed   mad    min    max  range skew
## Height    1 97 182.49  9.87 183.02  182.39 13.82 165.38 199.46  34.08 0.11
## Weight    2 97 144.34 41.60 139.39  143.15 46.95  78.38 220.88 142.50 0.25
```

```
## Gender    3 97   2.00  0.00   2.00    2.00  0.00   2.00   2.00   0.00  NaN
## Class     4 97   1.76  1.03   1.00    1.61  0.00   1.00   4.00   3.00 0.93
## Age       5 97  18.48  1.39  18.00   18.30  1.48  17.00  22.00   5.00 0.98
##        kurtosis   se
## Height    -1.30 1.00
## Weight    -1.04 4.22
## Gender      NaN 0.00
## Class     -0.59 0.10
## Age        0.27 0.14
```

ex) Use the `describeBy()` function to find the descriptive statistics based on class standing (ie. our Group variable):

```
cleanedData %>% describeBy(group = 'Class')
```

```
##
##  Descriptive statistics by group
## Class: 1
##        vars   n   mean    sd median trimmed   mad    min    max  range  skew
## Height    1 128 172.76 12.54 172.05  172.42 12.93 150.46 199.46  49.00  0.25
## Weight    2 128 149.20 40.49 149.36  148.76 45.37  77.58 223.24 145.66  0.09
## Gender    3 128   1.45  0.50   1.00    1.43  0.00   1.00   2.00   1.00  0.22
## Class     4 128   1.00  0.00   1.00    1.00  0.00   1.00   1.00   0.00   NaN
## Age       5 128  17.56  0.50  18.00   17.58  0.00  17.00  18.00   1.00 -0.25
##        kurtosis   se
## Height    -0.73 1.11
## Weight    -1.03 3.58
## Gender    -1.97 0.04
## Class       NaN 0.00
## Age       -1.95 0.04
## ------------------------------------------------------------
## Class: 2
##        vars  n   mean    sd median trimmed   mad    min    max  range  skew
## Height    1 29 175.80 13.00 174.10  175.66 13.44 153.19 198.99  45.79  0.30
## Weight    2 29 132.56 36.53 131.98  131.75 50.80  77.15 209.28 132.13  0.13
## Gender    3 29   1.48  0.51   1.00    1.48  0.00   1.00   2.00   1.00  0.07
## Class     4 29   2.00  0.00   2.00    2.00  0.00   2.00   2.00   0.00   NaN
## Age       5 29  19.28  0.80  19.00   19.32  1.48  18.00  20.00   2.00 -0.50
##        kurtosis   se
## Height    -0.91 2.41
## Weight    -1.08 6.78
## Gender    -2.06 0.09
## Class       NaN 0.00
## Age       -1.31 0.15
## ------------------------------------------------------------
## Class: 3
##        vars  n   mean    sd median trimmed   mad    min    max range  skew
## Height    1 26 176.61 11.72 174.04  176.97  7.61 152.40 196.30  43.9  0.04
## Weight    2 26 153.80 46.76 142.73  154.86 62.39  78.38 220.88 142.5 -0.04
## Gender    3 26   1.69  0.47   2.00    1.73  0.00   1.00   2.00   1.0 -0.79
## Class     4 26   3.00  0.00   3.00    3.00  0.00   3.00   3.00   0.0   NaN
## Age       5 26  19.62  0.50  20.00   19.64  0.00  19.00  20.00   1.0 -0.45
##        kurtosis   se
## Height    -0.57 2.30
## Weight    -1.49 9.17
```

```
## Gender    -1.43 0.09
## Class       NaN 0.00
## Age       -1.87 0.10
## ------------------------------------------------------------
## Class: 4
##        vars  n   mean    sd median trimmed   mad    min    max  range  skew
## Height    1 17 171.33 13.57 172.93  170.93 20.57 152.58 196.05  43.48  0.15
## Weight    2 17 153.14 49.23 150.49  153.56 70.35  79.61 220.32 140.72  0.08
## Gender    3 17   1.47  0.51   1.00    1.47  0.00   1.00   2.00   1.00  0.11
## Class     4 17   4.00  0.00   4.00    4.00  0.00   4.00   4.00   0.00   NaN
## Age       5 17  21.65  0.49  22.00   21.67  0.00  21.00  22.00   1.00 -0.56
##        kurtosis    se
## Height    -1.20  3.29
## Weight    -1.60 11.94
## Gender    -2.10  0.12
## Class       NaN  0.00
## Age       -1.78  0.12
```

It might not be useful in this case, but we can group by multiple variables using a list `c()` under the `group =` argument:

```
cleanedData %>% describeBy(group = c('Gender', 'Class'))
```

```
##
##  Descriptive statistics by group
## Gender: 1
## Class: 1
##        vars  n   mean    sd median trimmed   mad    min    max  range  skew
## Height    1 71 164.78  8.21 164.83  164.80  9.54 150.46 179.76  29.30  0.04
## Weight    2 71 150.74 40.81 150.49  150.44 45.89  77.58 223.24 145.66  0.10
## Gender    3 71   1.00  0.00   1.00    1.00  0.00   1.00   1.00   0.00   NaN
## Class     4 71   1.00  0.00   1.00    1.00  0.00   1.00   1.00   0.00   NaN
## Age       5 71  17.56  0.50  18.00   17.58  0.00  17.00  18.00   1.00 -0.25
##        kurtosis   se
## Height    -1.01 0.97
## Weight    -0.98 4.84
## Gender      NaN 0.00
## Class       NaN 0.00
## Age       -1.96 0.06
## ------------------------------------------------------------
## Gender: 2
## Class: 1
##        vars  n   mean    sd median trimmed   mad    min    max  range  skew
## Height    1 57 182.70  9.54 183.11  182.66 12.00 165.38 199.46  34.08  0.03
## Weight    2 57 147.28 40.36 144.02  146.60 47.91  80.62 220.13 139.50  0.07
## Gender    3 57   2.00  0.00   2.00    2.00  0.00   2.00   2.00   0.00   NaN
## Class     4 57   1.00  0.00   1.00    1.00  0.00   1.00   1.00   0.00   NaN
## Age       5 57  17.56  0.50  18.00   17.57  0.00  17.00  18.00   1.00 -0.24
##        kurtosis   se
## Height    -1.18 1.26
## Weight    -1.17 5.35
## Gender      NaN 0.00
## Class       NaN 0.00
## Age       -1.98 0.07
## ------------------------------------------------------------
```

```
## Gender: 1
## Class: 2
##        vars  n   mean    sd median trimmed   mad    min    max  range  skew
## Height    1 15 167.46  8.04 167.69  167.78 11.58 153.19 177.52  24.33 -0.24
## Weight    2 15 132.68 39.99 136.96  132.98 58.82  77.15 184.31 107.16 -0.09
## Gender    3 15   1.00  0.00   1.00    1.00  0.00   1.00   1.00   0.00   NaN
## Class     4 15   2.00  0.00   2.00    2.00  0.00   2.00   2.00   0.00   NaN
## Age       5 15  19.47  0.74  20.00   19.54  0.00  18.00  20.00   2.00 -0.87
##        kurtosis    se
## Height    -1.50  2.08
## Weight    -1.74 10.33
## Gender      NaN  0.00
## Class       NaN  0.00
## Age       -0.78  0.19
## ------------------------------------------------------------
## Gender: 2
## Class: 2
##        vars  n   mean    sd median trimmed   mad    min    max  range  skew
## Height    1 14 184.74 11.33 184.39  184.94 17.80 168.18 198.99  30.81 -0.12
## Weight    2 14 132.43 33.94 131.08  130.42 25.68  79.67 209.28 129.61  0.48
## Gender    3 14   2.00  0.00   2.00    2.00  0.00   2.00   2.00   0.00   NaN
## Class     4 14   2.00  0.00   2.00    2.00  0.00   2.00   2.00   0.00   NaN
## Age       5 14  19.07  0.83  19.00   19.08  1.48  18.00  20.00   2.00 -0.12
##        kurtosis   se
## Height    -1.58 3.03
## Weight    -0.27 9.07
## Gender      NaN 0.00
## Class       NaN 0.00
## Age       -1.64 0.22
## ------------------------------------------------------------
## Gender: 1
## Class: 3
##        vars n   mean    sd median trimmed   mad    min    max range  skew
## Height    1 8 167.06  9.30 168.91  167.06  6.85 152.40 177.35 24.95 -0.56
## Weight    2 8 178.28 33.84 189.81  178.28 25.94 129.21 208.54 79.34 -0.44
## Gender    3 8   1.00  0.00   1.00    1.00  0.00   1.00   1.00  0.00   NaN
## Class     4 8   3.00  0.00   3.00    3.00  0.00   3.00   3.00  0.00   NaN
## Age       5 8  19.88  0.35  20.00   19.88  0.00  19.00  20.00  1.00 -1.86
##        kurtosis    se
## Height    -1.39  3.29
## Weight    -1.72 11.96
## Gender      NaN  0.00
## Class       NaN  0.00
## Age        1.70  0.12
## ------------------------------------------------------------
## Gender: 2
## Class: 3
##        vars  n   mean    sd median trimmed   mad    min    max  range skew
## Height    1 18 180.85 10.23 176.63  180.81  8.83 165.92 196.30  30.38 0.29
## Weight    2 18 142.92 48.35 129.70  142.09 59.20  78.38 220.88 142.50 0.32
## Gender    3 18   2.00  0.00   2.00    2.00  0.00   2.00   2.00   0.00  NaN
## Class     4 18   3.00  0.00   3.00    3.00  0.00   3.00   3.00   0.00  NaN
## Age       5 18  19.50  0.51  19.50   19.50  0.74  19.00  20.00   1.00 0.00
##        kurtosis    se
```

```
## Height    -1.62  2.41
## Weight    -1.42 11.40
## Gender      NaN  0.00
## Class       NaN  0.00
## Age       -2.11  0.12
## ------------------------------------------------------------
## Gender: 1
## Class: 4
##        vars n   mean    sd median trimmed   mad    min    max  range  skew
## Height    1 9 162.92 10.83 157.24  162.92  6.39 152.58 179.37  26.79  0.56
## Weight    2 9 158.21 50.83 150.49  158.21 72.91  87.15 215.37 128.22 -0.14
## Gender    3 9   1.00  0.00   1.00    1.00  0.00   1.00   1.00   0.00   NaN
## Class     4 9   4.00  0.00   4.00    4.00  0.00   4.00   4.00   0.00   NaN
## Age       5 9  21.56  0.53  22.00   21.56  0.00  21.00  22.00   1.00 -0.19
##        kurtosis    se
## Height    -1.67  3.61
## Weight    -1.88 16.94
## Gender      NaN  0.00
## Class       NaN  0.00
## Age       -2.17  0.18
## ------------------------------------------------------------
## Gender: 2
## Class: 4
##        vars n   mean    sd median trimmed   mad    min    max  range  skew
## Height    1 8 180.79  9.68 176.71  180.79  6.72 171.62 196.05  24.43  0.49
## Weight    2 8 147.43 50.17 138.98  147.43 36.49  79.61 220.32 140.72  0.31
## Gender    3 8   2.00  0.00   2.00    2.00  0.00   2.00   2.00   0.00   NaN
## Class     4 8   4.00  0.00   4.00    4.00  0.00   4.00   4.00   0.00   NaN
## Age       5 8  21.75  0.46  22.00   21.75  0.00  21.00  22.00   1.00 -0.95
##        kurtosis    se
## Height    -1.70  3.42
## Weight    -1.49 17.74
## Gender      NaN  0.00
## Class       NaN  0.00
## Age       -1.21  0.16
```

```
# Our data consists of two genders and four groups, hence we have
# 8 pairs of descriptive statistics.
```

Using the `summarize()` function:

The `summarize()` (or `summarise()`) function is included within the dplyr package. We have the liberty of creating our own statistics about the data. We see that `describe()` and `describeBy()` only provide the mean, standard deviation, count, etc. The syntax with `summarize()` is more involved, however, as with the other functions we've implemented, we practically tell it what we want:

With `summarize()`, we create our own table:

```
cleanedData %>% summarize()
```

```
## # A tibble: 1 x 0
```

Some helpful arguments are:

```
* mean(var) -- will return the mean
* sd(var) -- will return the standard deviation
* n() -- will return the count, is typically used in tandem with `group_by()`
```

* group_by() is not included within summarize, but used as a quick way to group the data by factor.

Let's do an example.

```
cleanedData %>% summarize(MeanHeight = mean(Height),
            StdDevHeight = sd(Height))
```

```
## # A tibble: 1 x 2
##   MeanHeight StdDevHeight
##        <dbl>        <dbl>
## 1       174.         12.6
```

Challenge: Try using the `summarize()` function, grouped by `Class`, to find the count of participants in each Class and percentage per Class. (If 45/100 Males responded, the percentage would 0.45 = 45%). The numerator is simply the count of participants; how do we get the denominator?

```
cleanedData %>% group_by(Class) %>%
  summarize(Count = n(),
            Percent = n() / nrow(cleanedData))
```

```
## # A tibble: 4 x 3
##   Class     Count Percent
##   <fct>     <int>   <dbl>
## 1 Freshman    128   0.64
## 2 Sophomore    29   0.145
## 3 Junior       26   0.13
## 4 Senior       17   0.085
```

ex) Try using the `summarize()` function to find the mean and standard deviation of the weights, grouped by Gender:

```
cleanedData %>% group_by(Gender) %>%
  summarize(MeanWeight = mean(Weight),
            SdWeight = sd(Weight))
```

```
## # A tibble: 2 x 3
##   Gender MeanWeight SdWeight
##   <fct>       <dbl>    <dbl>
## 1 Female       151.     41.9
## 2 Male         144.     41.6
```

ex) Try using the `summarize()` function to find the percentage of participants by age: (`group_by()` works fine with Age in this case, even if it's not a factor variable! Why?)

```
cleanedData %>% group_by(Age) %>%
  summarize(Count = n(),
            Percent = n() / nrow(cleanedData))
```

```
## # A tibble: 6 x 3
##     Age Count Percent
##   <dbl> <int>   <dbl>
## 1    17    56   0.28
## 2    18    78   0.39
## 3    19    19   0.095
## 4    20    30   0.15
## 5    21     6   0.03
## 6    22    11   0.055
```

Standardization

A key topic in any statistics course is **standardization**. One of the important reasons we standardize data is to maintain the accuracy and information in the data while converting to a similar (or standard) scale, something that generally falls between -3 and 3 standard deviations.

Q: What do we call data that falls outside -3 and 3 standard deviations?

A: We call these values **outliers**.

To standardize data, we follow the formula:

(Observation within the sample - Mean of the sample) / (Standard Deviation of the Sample)

Let's standardize our Weight data:

```
cleanedData <- cleanedData %>% mutate(WeightSTD = (Weight - mean(Weight)) / sd(Weight))
head(cleanedData)
```

```
## # A tibble: 6 x 6
##   Height Weight Gender Class      Age WeightSTD
##    <dbl>  <dbl> <fct>  <fct>    <dbl>     <dbl>
## 1   150.   167. Female Freshman    17     0.463
## 2   151.   180. Female Freshman    18     0.769
## 3   151.   222. Female Freshman    18     1.78
## 4   151.   137. Female Freshman    17    -0.264
## 5   151.   223. Female Freshman    17     1.81
## 6   153.   193. Female Freshman    17     1.09
```

ex) Try standardizing the Height data. Store this in a separate variable like we did above so we don't lose the original data!

```
cleanedData <- cleanedData %>% mutate(HeightSTD = (Height - mean(Height)) / sd(Height))
head(cleanedData)
```

```
## # A tibble: 6 x 7
##   Height Weight Gender Class      Age WeightSTD HeightSTD
##    <dbl>  <dbl> <fct>  <fct>    <dbl>     <dbl>     <dbl>
## 1   150.   167. Female Freshman    17     0.463     -1.83
## 2   151.   180. Female Freshman    18     0.769     -1.83
## 3   151.   222. Female Freshman    18     1.78      -1.83
## 4   151.   137. Female Freshman    17    -0.264     -1.79
## 5   151.   223. Female Freshman    17     1.81      -1.78
## 6   153.   193. Female Freshman    17     1.09      -1.66
```

Introduction to `ggplot2`/Creating Graphics

Graphics (boxplots, scatterplots, barplots) provide the reader a visual of the distribution of your data, rather than just having numbers the body of your text. The most popular package for graphing is `ggplot2`, the package in R for creating statistical graphs.

`ggplot2` is conveniently included in the `tidyverse` package, so we do not need to install any additional packages.

How do we built a `ggplot()`? The no matter which graphic you are creating, we **always** begin with:

```
df %>% ggplot()
```

Why do we provide visuals of the data? Data visualization helps to tell stories by curating data into a form easier to understand, highlighting the trends and outliers, which is very helpful for individuals reading your research that are perhaps new to the field, or at their leisure.

If we put this into a chunk, we will see either below or in our Plots tab (in the Files pane) and empty grey graph has been created:

```
cleanedData %>% ggplot()
```


The convenient and streamlined nature of `ggplot()` allows to add the elements we want to a graph, ie. we provide R the building blocks. This might seem daunting, but don't forget, if you need Help with any functions (what arguments can it take, what types of data are allowed), use the console and type `?functionName`. Let's see what's included and necessary in ggplot:

```
?ggplot
```

The necessary arguments are `data`, the data frame (which we use `%>%`to bypass) and a *mapping*.

The mapping we will extend, for instruction sake, into the respective plots. However, to explain its purpose, `mapping = aes()` is an argument that is used either in `ggplot(mapping = aes())` or its subsequent layers. Included within the aesthetic mapping (hence `aes()`) are the variable types along the axes: for example, if we wanted to create a scatterplot with Weight on the x-axis and Height on the y-axis, our aesthetic mapping would be: `mapping = aes(x = Weight, y = Height)`.

Note: There is one portion of the help that I want to draw attention to: "The data = and mapping = specifications in the arguments are optional (and are often omitted in practice), so long as the data and the mapping values are passed into the function in the right order." What this means is that we can exclude explicitly defining `data =` and `mapping =`, further condensing the code (perhaps for readability), into something like: `df %>% ggplot(aes())`. This is just a tip; once you become more familiar with the general pattern function arguments follow, it is generally more efficient to omit these `something =` definitions since there aren't keyboard shortcuts for these.

When implementing a `ggplot()`, we "add" our elements. The elements that we add are:

```
* `geom_typeOfGraph()` : there are over 20 different options, including `geom_histogram()`, `geom_boxplo
```

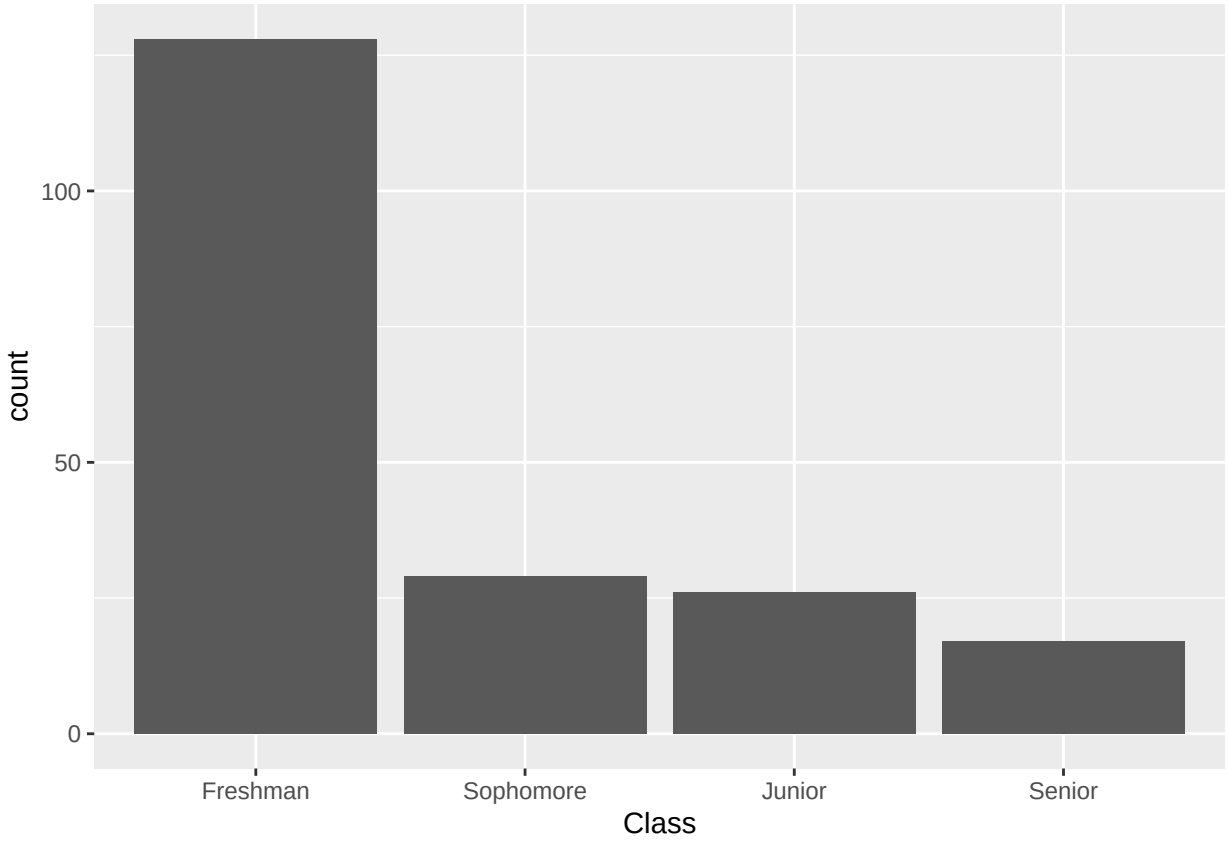
* xlab("X label"), ylab("Y label") : these are not necessary but are nice if your variable names don't
* ggtitle(label = "Title", subtitle = "Subtitle") : these automatically populate in the top-left corner

The order in which we add the elements (xlabel, ylabel, theme, title) does not matter. However, we must specify the type of graph (`geom_boxplot()`, `geom_histogram()`, etc) before we add the labels, title, and theme.

Barplots:

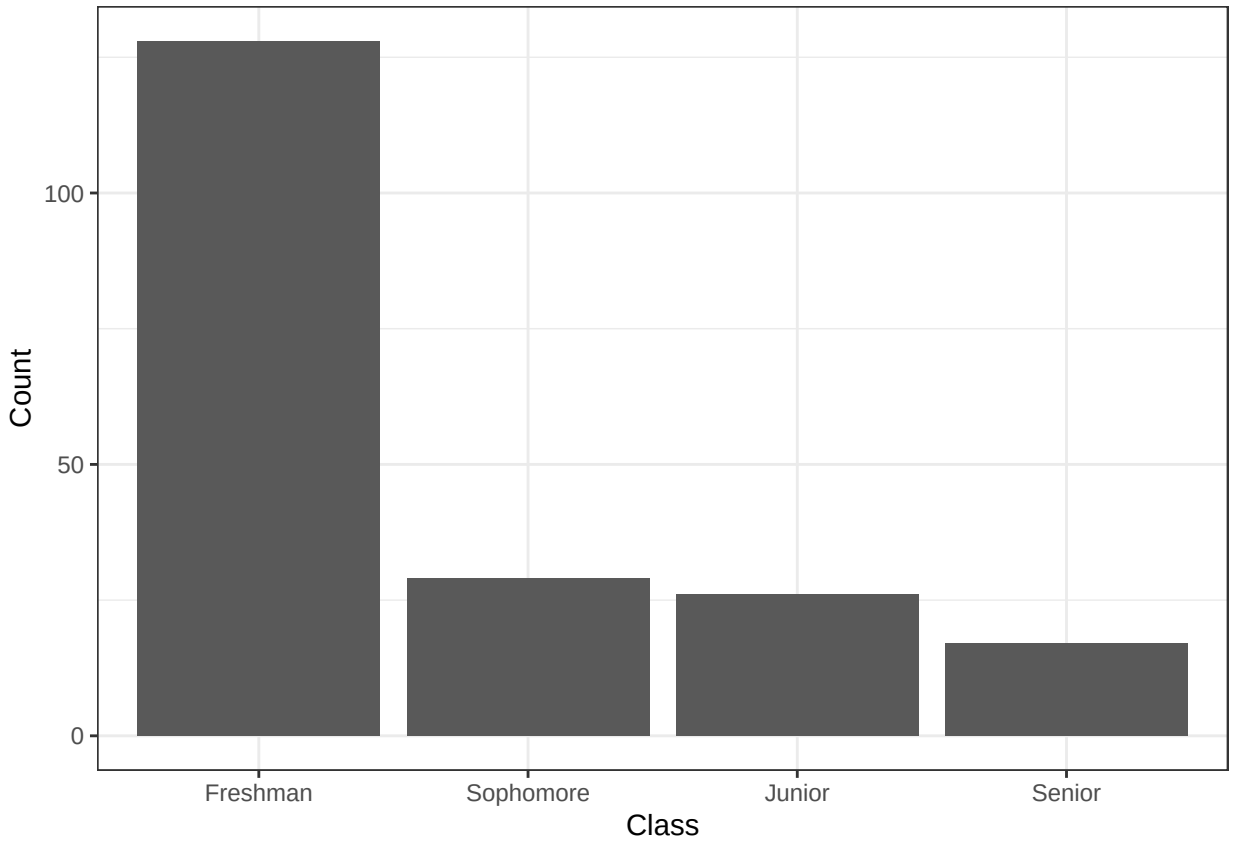
A barplot will provide a visual for counts of data. If we wanted to see the distribution of participants (by gender or group, for example):

```
cleanedData %>% ggplot() +
  geom_bar(mapping = aes(x = Class))
```



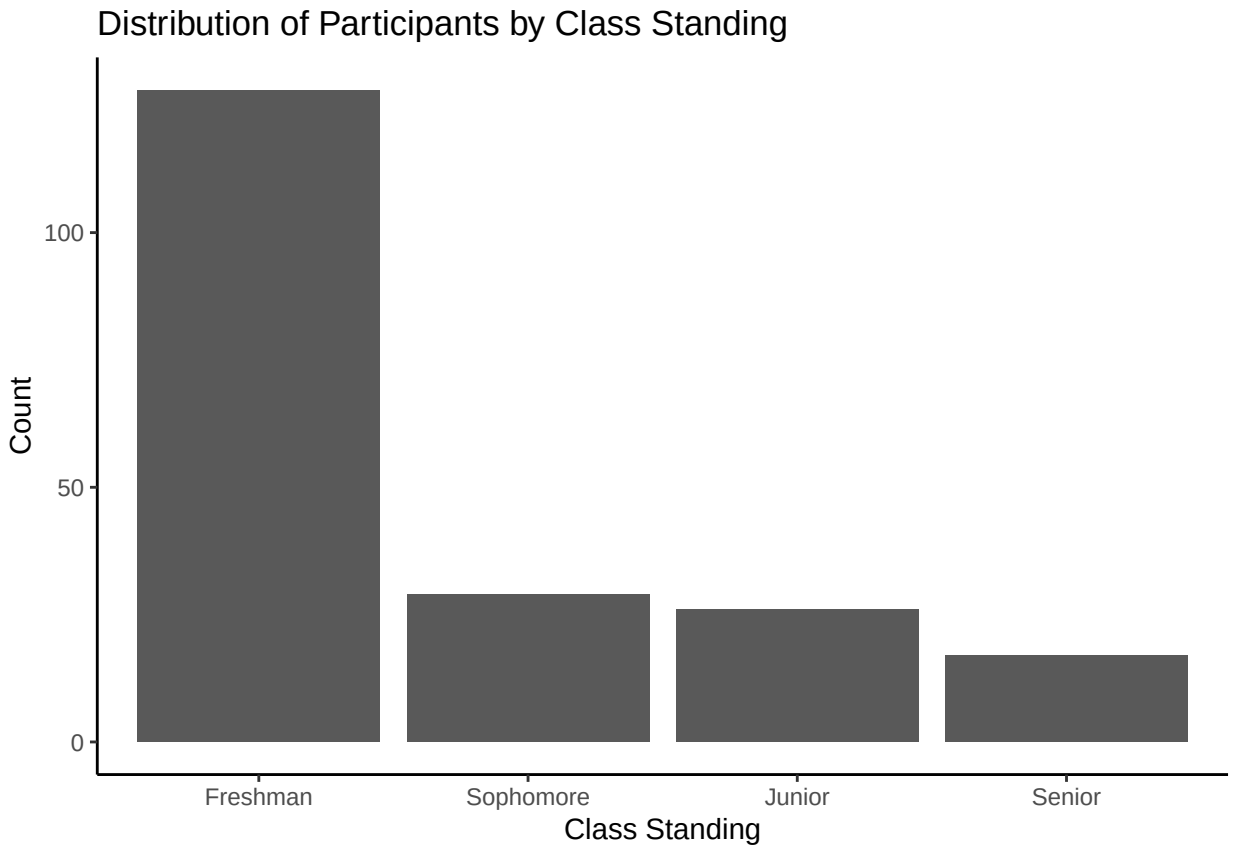
There is a default theme applied to all ggplots without providing any other aesthetic mappings. It is a pretty mundane greyscale with a grid background. We can change this easily using `ggtheme`, a class of themes that we can add to our graph. For example:

```
cleanedData %>% ggplot() +
  geom_bar(aes(x = Class)) +
  xlab("Class") +
  ylab("Count") +
  theme_bw()
```



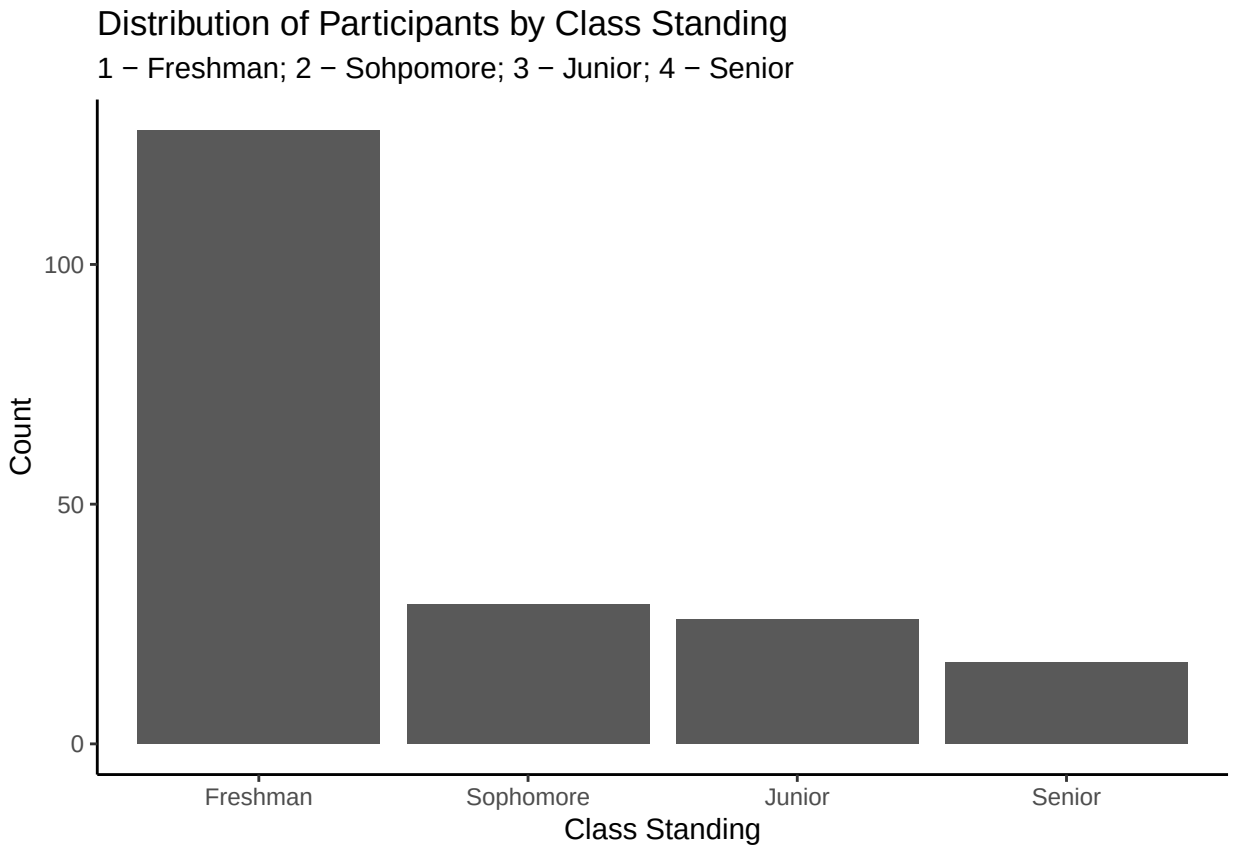
Now let's add a title:

```
cleanedData %>% ggplot() +
  geom_bar(mapping = aes(x = Class)) +
  xlab("Class Standing") +
  ylab("Count") +
  theme_classic() +
  ggtitle(label = "Distribution of Participants by Class Standing")
```



If we did not rename the factors, our x-axis would include values 1, 2, 3, and 4. If you decided not to convert, it would be appropriate to include a note on the graph describing what each number represents. We can do this by adding a subtitle:

```
cleanedData %>% ggplot() +
  geom_bar(mapping = aes(x = Class)) +
  xlab("Class Standing") +
  ylab("Count") +
  theme_classic() +
  ggtitle(label = "Distribution of Participants by Class Standing",
          subtitle = "1 - Freshman; 2 - Sohpomore; 3 - Junior; 4 - Senior")
```



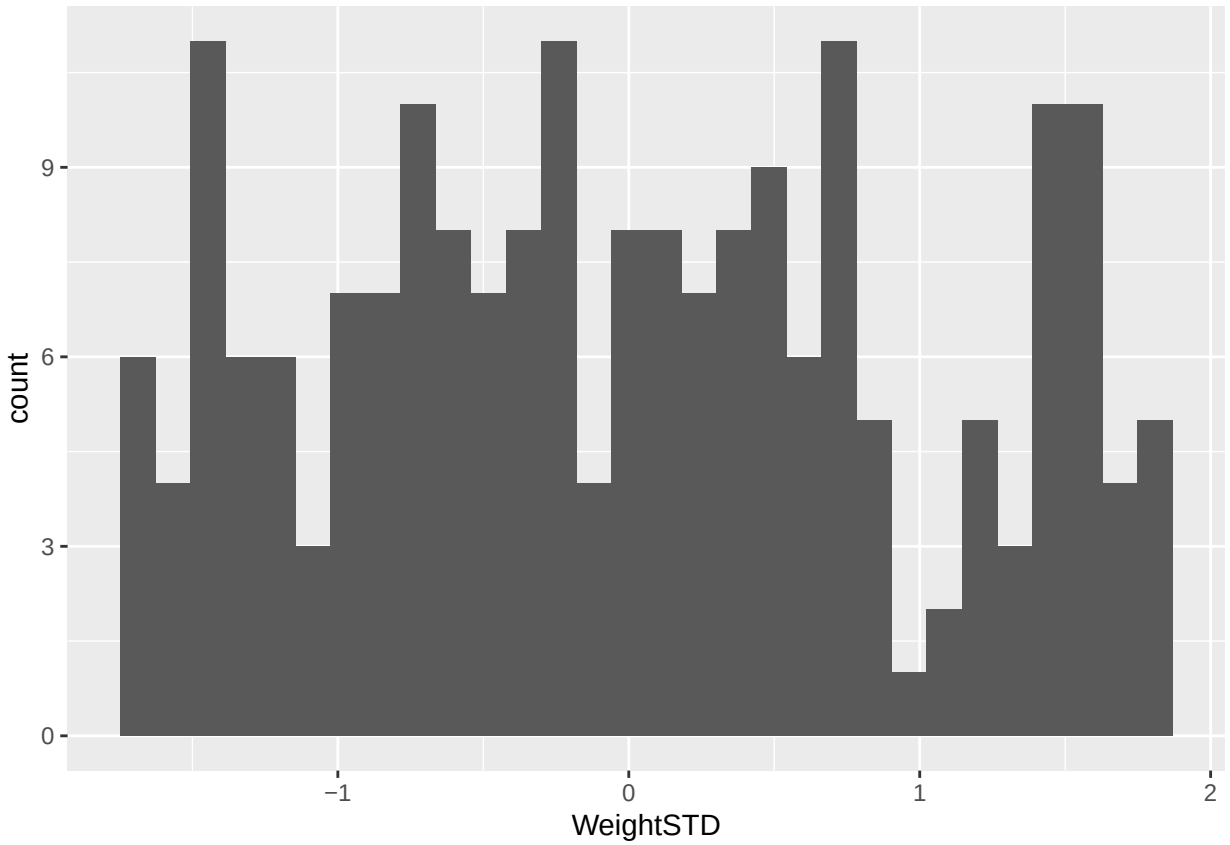
Histograms:

Histograms and bar plots both provide column-based data representation but provide different purposes. While bar plots are used to compare categorical (factor) data, histograms display the distribution of continuous data.

With histograms, we include an additional argument: `bins`. The number provided to the `bins` argument will output that number of columns. The complete syntax is: `df %>% ggplot() + geom_histogram(mapping = aes(x = Var), bins = #)`

```
cleanedData %>% ggplot() +
  geom_histogram(mapping = aes(x = WeightSTD)) # If we do not specify the bins, a default of 30 will be
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



There is quite the science behind determining the number of bins. If we have too many bins (relative the amount of data we have), we may not be able to discern the data versus the "noise" (the randomness included in the data) or compare to another common statistical distribution specific patterns. Whereas if we have too few bins, the data will look a bit more block-y.

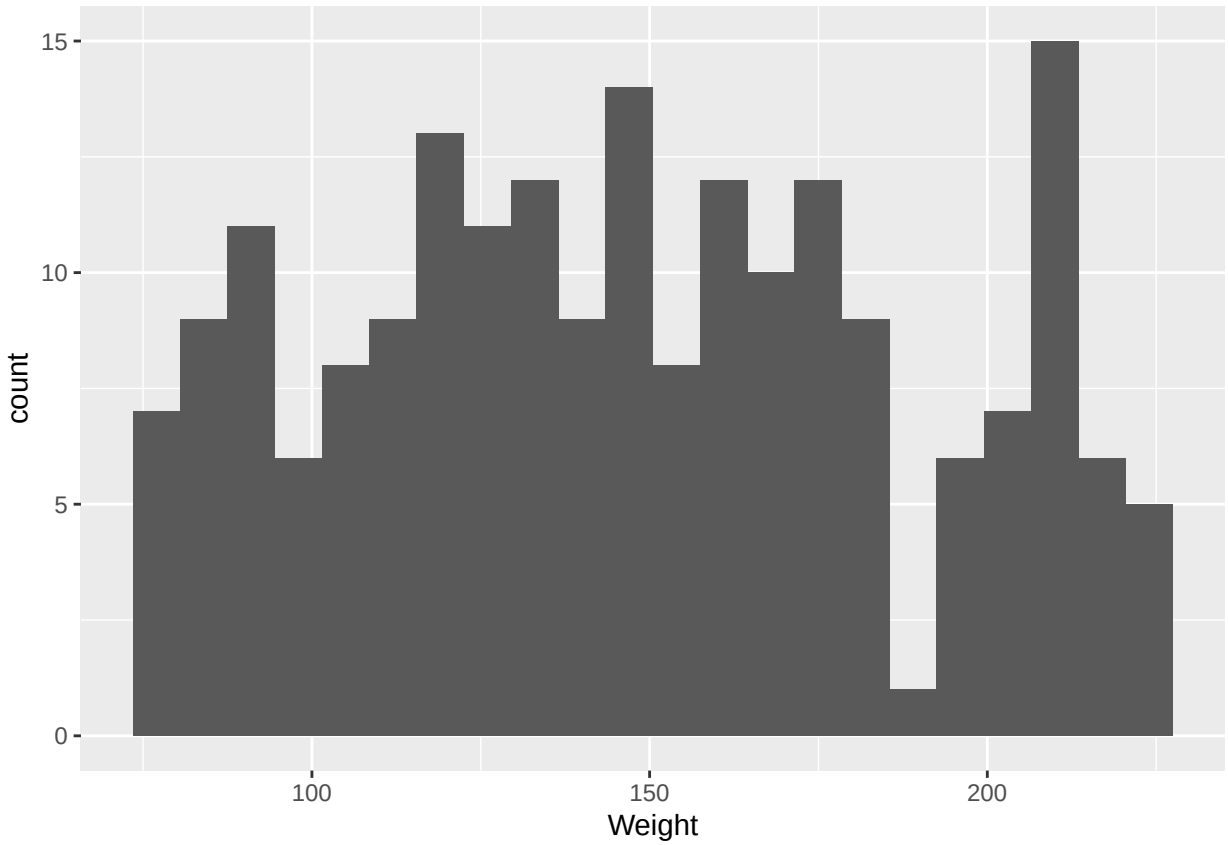
Instead of defining the number of bins, a robust and well-practiced rule is the Freedman-Diaconis rule. In this case, we specify the bin **width** rather than the number. The Freedman-Diaconis rule is defined as:

BinWidth = (Maximum of Data - Minimum of Data) * sampleSize ^ (1/3) / (2 * IQR)

- We don't have the ability to pipe the dataframe into this, so we have to specify the variable using `df$Var`:

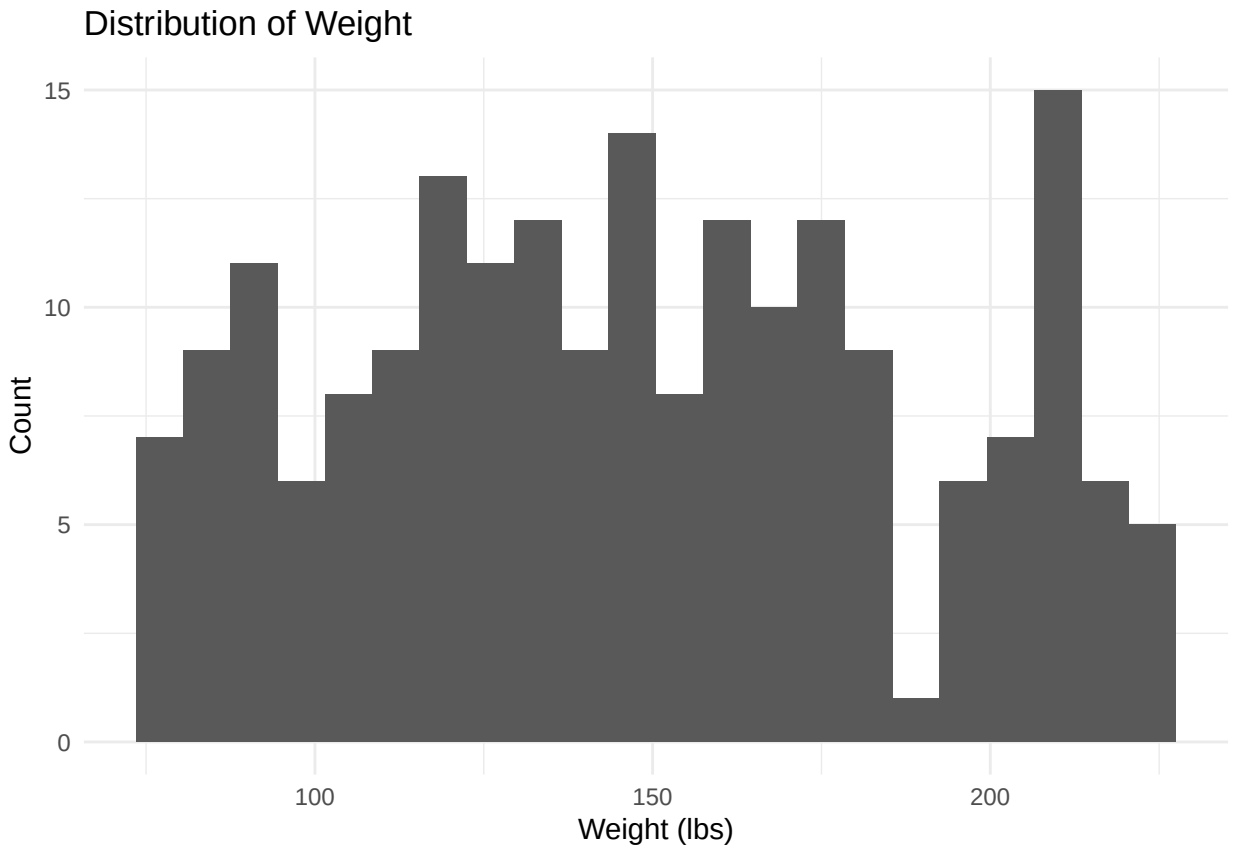
```
bw <- ceiling((max(cleanedData$Weight) - min(cleanedData$Weight)) /
  (2 * IQR(cleanedData$Weight) * nrow(cleanedData)^(-1/3)))
```

```
cleanedData %>% ggplot() +
  geom_histogram(mapping = aes(x = Weight), binwidth = bw)
```



Let's make it pretty. Type `theme` and wait for the options to pop up; go ahead and pick one that sounds cool! Add a title, as well.

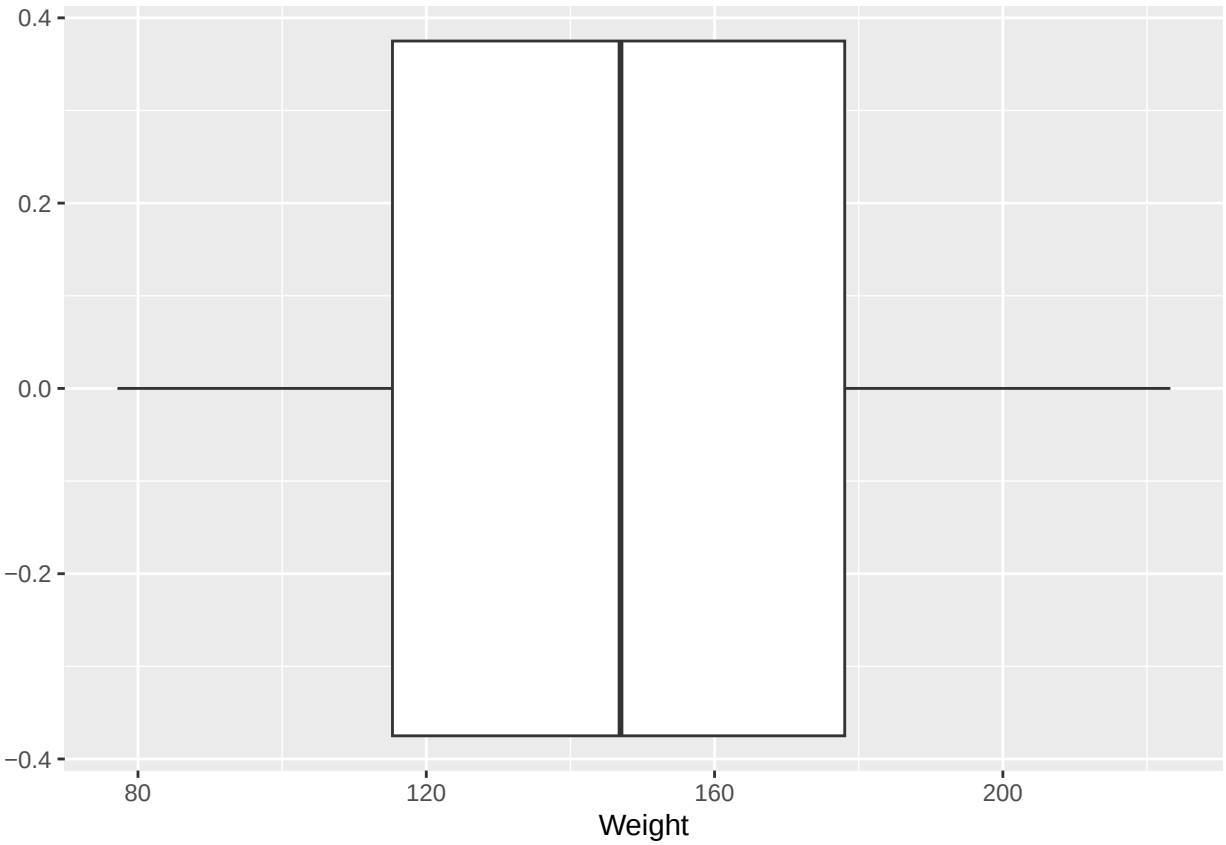
```
cleanedData %>% ggplot() +
  geom_histogram(mapping = aes(x = Weight), binwidth = bw) +
  xlab('Weight (lbs)') +
  ylab('Count') +
  ggtitle('Distribution of Weight') +
  theme_minimal()
```



Boxplots:

Boxplots, also known as box-and-whisker plots, are a type of graph that we can use to visualize the range (as well as observe any outliers) of a quantitative variable. Using `geom_boxplot()`, let's find the distribution of our Weight variable:

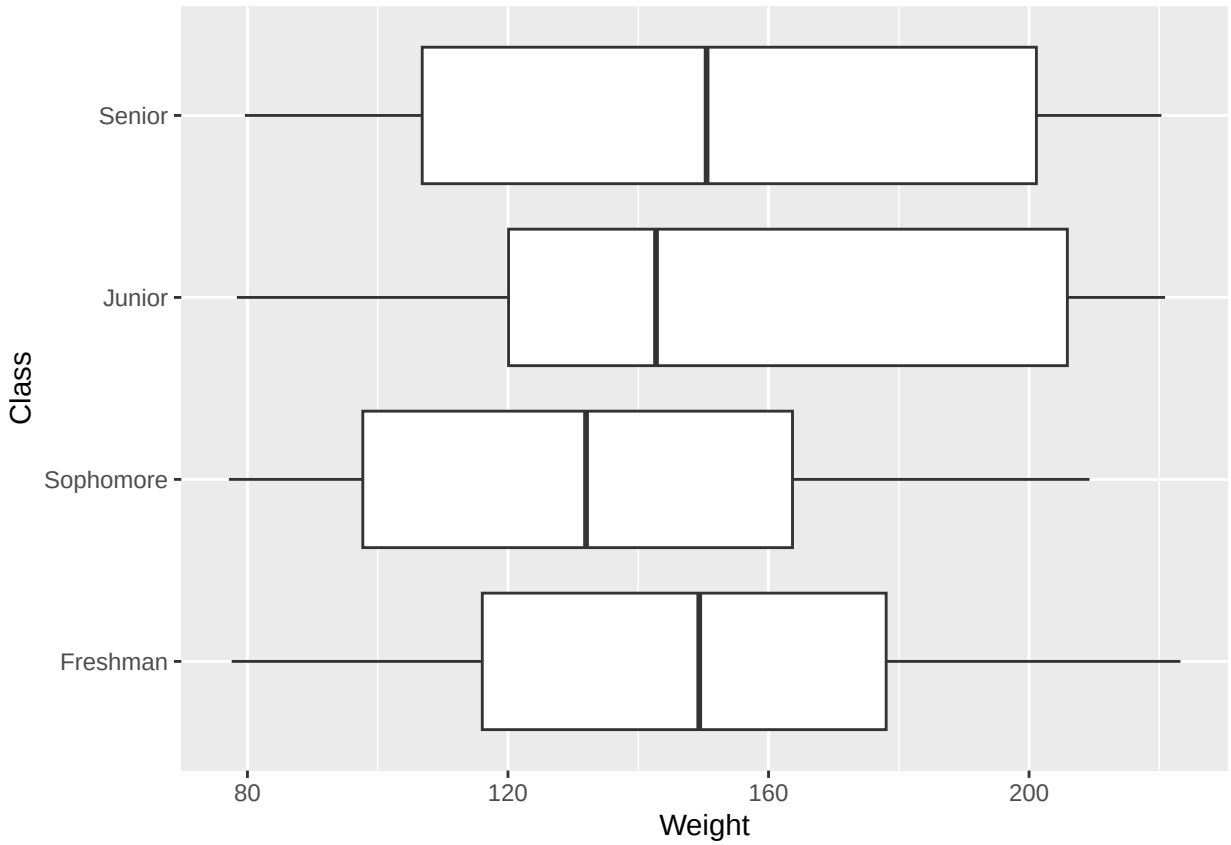
```
cleanedData %>% ggplot() +
  geom_boxplot(mapping = aes(x = Weight))
```



To understand how this works, we have a center "box", which is described as the interquartile range (IQR). The left-ward side of the box is the first quartile (25th percentile, 25% mark), the middle bar is the median (50th percentile, 50% mark), and the right-ward side of the box is the third quartile (75th percentile, 75% mark). The lines on either side, the "whiskers", are used to visualize the distance between the 25th percentile and the minimum (to the left, or downward if vertically plotted) and between the 75th percentile and the maximum (to the right, or upward if vertically plotted).

We can dig deeper to see the distribution of Weight by Class, using the factor variable as our y-mapping:

```
cleanedData %>% ggplot() +
  geom_boxplot(mapping = aes(x = Weight, y = Class))
```

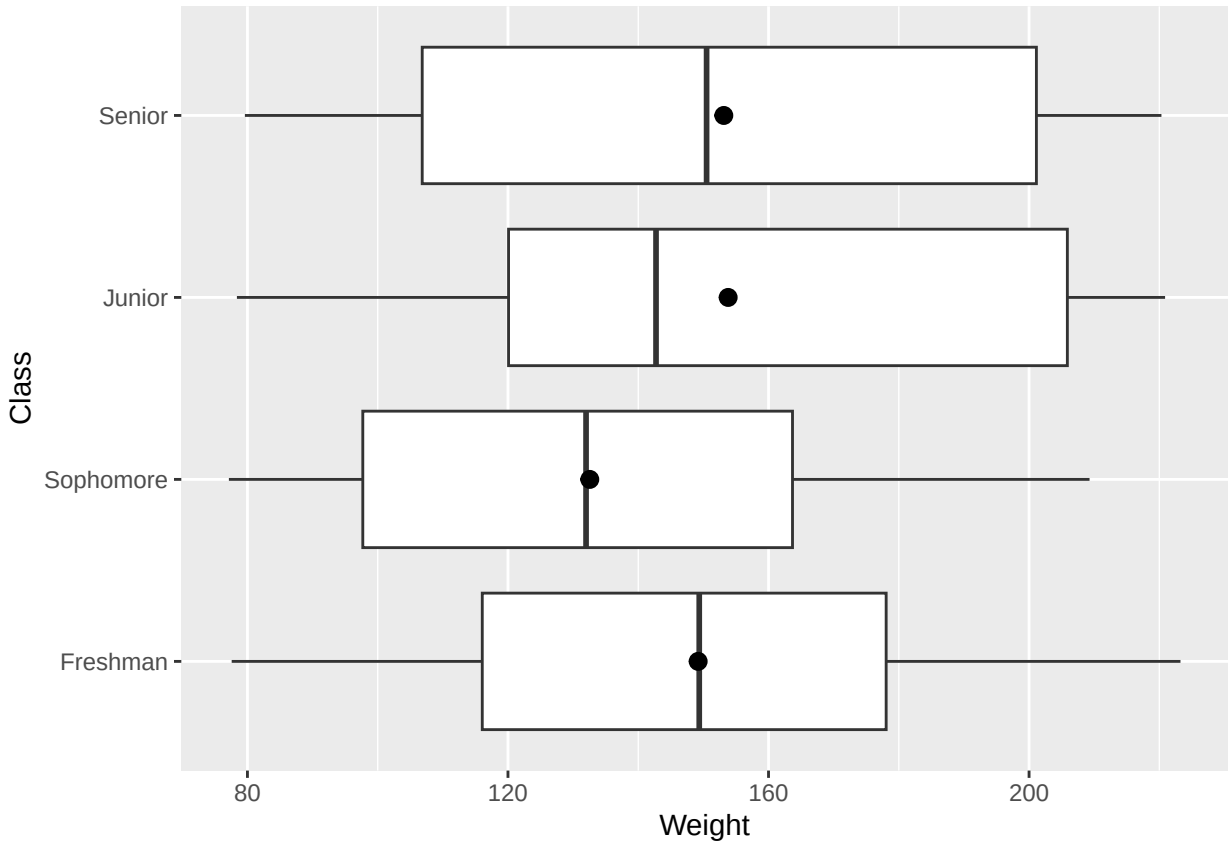


If we wanted to add a marker for the mean value of each group, we need to make some adjustments:

```
# Move the aesthetic mapping to ggplot() instead of within geom_boxplot()
cleanedData %>% ggplot(mapping = aes(x = Weight, y = Class)) +
  geom_boxplot() +
  stat_summary(fun.y = "mean")
```

```
## Warning: The `fun.y` argument of `stat_summary()` is deprecated as of ggplot2 3.3.0.
## i Please use the `fun` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_segment()`).
```

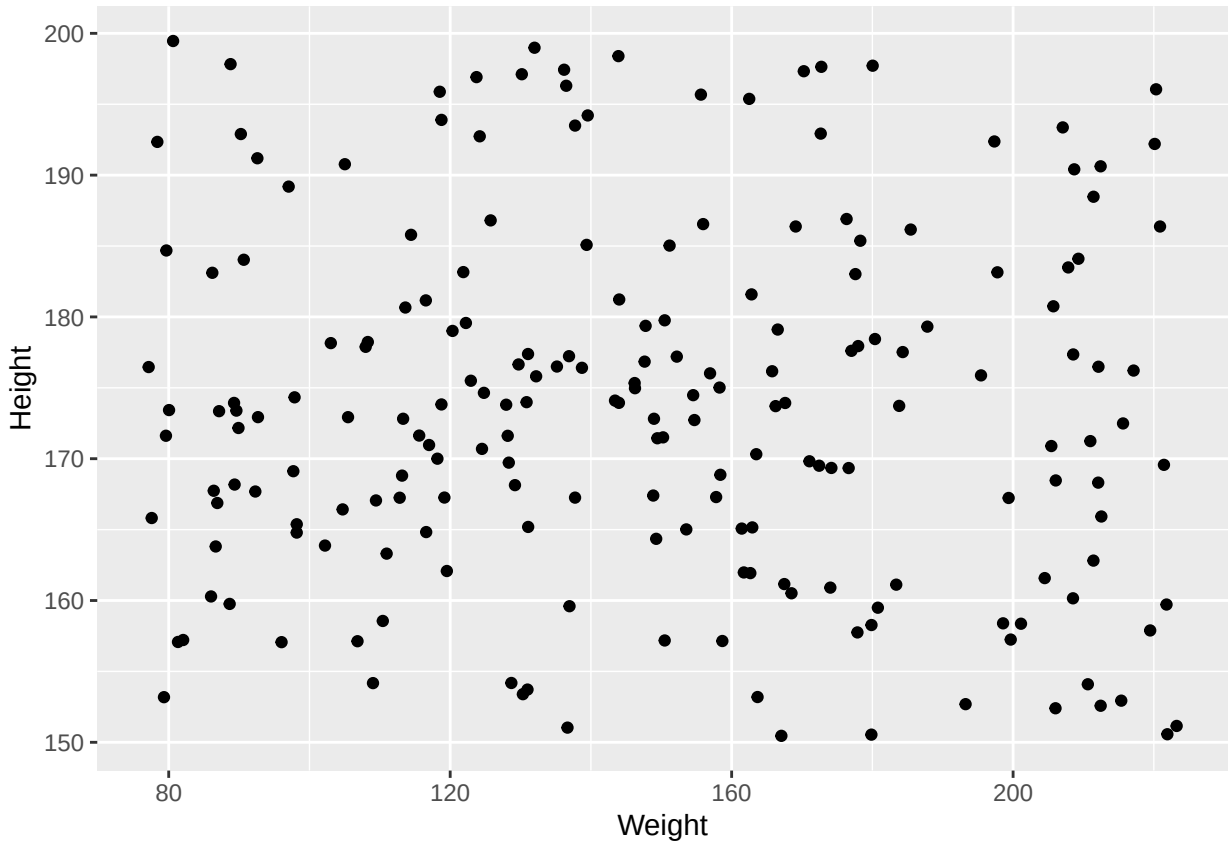



```
# This plots a function of y, where y is our grouping
# variable, and we tell it which statistic.
```

Scatterplots:

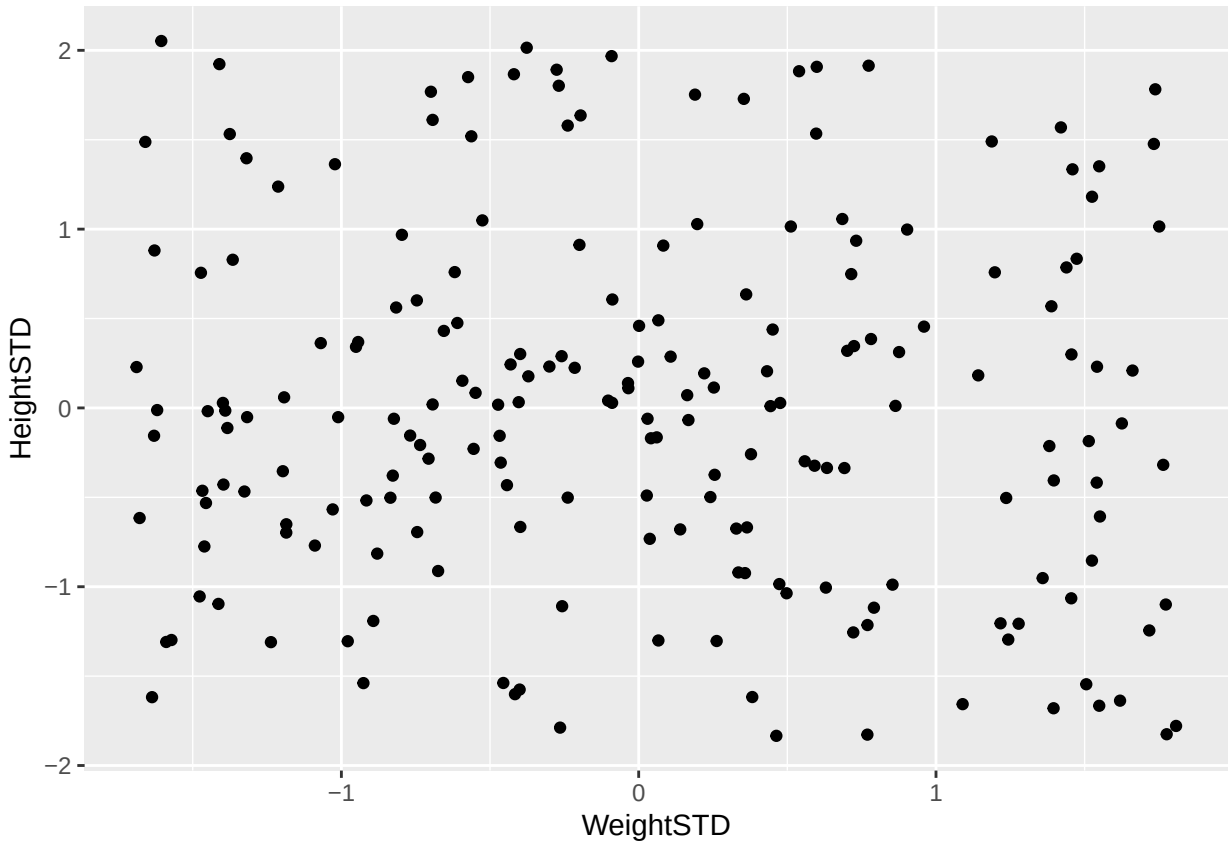
Scatterplots provide the paired distribution of, typically, two quantitative data types. If we wanted to plot the Weight and Height pairs, we use `geom_point(mapping = aes(x = Var1, y = Var2))`:

```
cleanedData %>% ggplot() +
  geom_point(mapping = aes(x = Weight, y = Height))
```



Our scales are off, so for consistency's sake (even though it will provide an identical plot), let's see the standardized Weight vs. standardized Height. Go ahead and try this on your own!

```
cleanedData %>% ggplot() +
  geom_point(mapping = aes(x = WeightSTD, y = HeightSTD))
```



Scatterplots are one of the several ways we can see if there is any specific relationship between the data: whether that be linear, quadratic, logarithmic, etc. Given this is all simulated data, not conditioned on either variable, it just looks completely randomized. But we can still find a line of best fit!

Linear Regression:

Linear regression is a very common and useful statistical technique used to mathematically describe the relationship between quantitative variables. In R, we use the function `lm()`, which stands for "linear model". The reason the function is described `lm` instead of `lr` (for linear regression) is because the `lm()` function does not only perform simple linear regression (ie. regression between two continuous variables). At least for purposes of this workshop, we will exclusively show how to perform simple linear regression. This is the only portion I go into some theory, so bear with me!

The formal mathematics involved in a simple linear regression model (with one independent variable) is

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

The first coefficient, Beta0, is the intercept. The second coefficient, Beta1, is the slope. The final term is used theoretically. We assume that each observation can be predicted with these coefficients, with errors Epsilon (the e-looking symbol), also called "noise." Obviously we can't predict each observation with 100% accuracy (unless the data fall on directly on said line). This "noise" is a random variable (a mathematical formalization of a quantity or object which depends on random event) following a Normal distribution having a mean of 0 and a variance.

Mathematically, we represent this as: (Use `CTRL + ENTER` to see the math text)

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

The expected value (ie. the mean) of our random variable Epsilon is 0, so it provides no information to our prediction, $\hat{y}$.

The purpose of linear regression is to find the line of best fit. This line of best fit minimizes the sum of squared distances between the observed and expected (predicted) values These are called the **residuals**. When we take the expected value of our predictor, y, our new model is:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

Anyways – when we use the `lm()` function, we are provided the estimators for the linear regression model for both the intercept and the slope.

The syntax included in the `lm()` function required a response (dependent, DV) variable by (using `~`) your independent variable (IV): `lm(DV ~ IV, data = df)`.

We can add as many variables as we want using `lm(DV ~ IV1 + IV2 + ..., data = df)`. This is called multiple linear regression; we have multiple independent variables that we are incorporating into our model. For graphical purposes, let's first do one dependent variable and one independent variable. Let's do Weight by Height:

```
model_lm <- lm(Weight ~ Height, data = cleanedData)
model_lm
```

```
##
## Call:
## lm(formula = Weight ~ Height, data = cleanedData)
##
## Coefficients:
## (Intercept)       Height
##    184.2585      -0.2105
```

From the output, we are instantly given the coefficients of the linear model. We can access the intercept and coefficient (for Height) by:

```
model_lm$coefficients[[1]]
```

```
## [1] 184.2585
```

```
model_lm$coefficients[[2]]
```

```
## [1] -0.2105081
```

However, there is an underlying test within the `lm()` function. We implicitly test for the significance of these estimators to our model. We have two hypotheses we test:

$$H_0 : \beta_0 = \beta_1 = 0 H_A : \beta_i \neq 0 \text{ for } i \in \{0, 1\}$$

If either one of our coefficients were 0, or more specifically not significantly different than 0, the variable associated with the coefficient is not significant to the model, ie. it is not a good predictor.

To see the test, we use `summary(model)`:

```
summary(model_lm)
```

```
##
## Call:
## lm(formula = Weight ~ Height, data = cleanedData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -72.686 -32.545  -0.532  30.413  77.335
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 184.2585    40.9013   4.505 1.13e-05 ***
## Height       -0.2105     0.2350  -0.896    0.371
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 41.81 on 198 degrees of freedom
## Multiple R-squared:  0.004036,   Adjusted R-squared:  -0.0009944
## F-statistic: 0.8023 on 1 and 198 DF,  p-value: 0.3715
```

An intercept will always be included, no matter the model, so the "significance" of that is, to be frank, useless; we are only interested in the coefficients attached to x.

In this case, we see that the the p-value associated Height ($p = 0.371$) is greater than our significance level – we'll say 0.05 – this is a very standard significance level. In this case, we would interpret as follows:

"At the 5% significance level, we fail to reject the null hypothesis. The data do not suggest that Height is significant to our model."

This is a very basic conclusion, one that should be adapted to your project, but generally follows the same rules.

Let's create a third model with two independent variables: Height and Age.

```
model_lm2 <- lm(Weight ~ Height + Age, data = cleanedData)
summary(model_lm2)
```

```
##
## Call:
## lm(formula = Weight ~ Height + Age, data = cleanedData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -72.220 -32.741  -0.459  30.881  76.489
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 178.1461    55.9921   3.182   0.0017 **
## Height       -0.2115     0.2357  -0.897   0.3707
## Age           0.3408     2.1259   0.160   0.8728
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 41.91 on 197 degrees of freedom
## Multiple R-squared:  0.004166,   Adjusted R-squared:  -0.005944
## F-statistic: 0.412 on 2 and 197 DF,  p-value: 0.6629
```

Here, we see that, again, neither Height nor Age are significant to the model.

If you decide you want to test for significant interaction between variables, we use an asterisk * instead of +:

```
model_lm3 <- lm(Weight ~ Height * Age, data = cleanedData)
model_lm3
```

```
##
```

```
## Call:
## lm(formula = Weight ~ Height * Age, data = cleanedData)
##
## Coefficients:
## (Intercept)       Height          Age   Height:Age
##   -76.06182      1.26278     14.16776     -0.08018
```

```
summary(model_lm3)
```

```
##
## Call:
## lm(formula = Weight ~ Height * Age, data = cleanedData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -71.466 -32.475  -0.995  30.282  82.932
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -76.06182  523.91520  -0.145    0.885
## Height        1.26278    3.03012   0.417    0.677
## Age          14.16776   28.41311   0.499    0.619
## Height:Age   -0.08018    0.16429  -0.488    0.626
##
## Residual standard error: 41.99 on 196 degrees of freedom
## Multiple R-squared:  0.005374,   Adjusted R-squared:  -0.00985
## F-statistic: 0.353 on 3 and 196 DF,  p-value: 0.787
```

We test for interaction when we want to test whether the association between two or more variables depends on the value of a different variable. In essence, we create a new variable that incorporates the influence on two variables as it pertains to the model. There may be instances where the interaction between variables is significant but the variables alone are not. Interaction in models is a bit more complicated, so, if interested, I like the following resources:

For the purpose and implementation of interaction effects:

- https://statisticsbyjim.com/regression/interaction-effects/
- https://online.stat.psu.edu/stat501/lesson/8/8.6

For instances where your interaction terms are significant and your variables alone are not and how to interpret:

- https://www.theanalysisfactor.com/interactions-main-effects-not-significant/

Correlations:

One thing to be careful with when performing linear regression is the correlation between independent variables. **Correlation** quantifies the strength of the linear relationship between a pair of variables. For a correlation matrix, we select the numeric variables:

```
is.num <- sapply(cleanedData, is.numeric) # `sapply()` is a command used to
# apply a function to a data frame or matrix.
# Here, our function is "is.numeric" and apply a boolean value to each column.
is.num
```

```
##    Height    Weight    Gender     Class       Age WeightSTD HeightSTD
##      TRUE      TRUE     FALSE     FALSE      TRUE      TRUE      TRUE
```

We can use this newly created "list" `is.num` to select the columns where the data is numeric, denoted by "TRUE":

```
numericData <- cleanedData[, is.num]
```

This method is similar to what we did on Day 1 when we selected all rows and certain columns. Here, R is smart enough to understand that you want any variable having "TRUE":

```
head(numericData)
```

```
## # A tibble: 6 x 5
##   Height Weight   Age WeightSTD HeightSTD
##    <dbl>  <dbl> <dbl>     <dbl>     <dbl>
## 1   150.   167.    17     0.463     -1.83
## 2   151.   180.    18     0.769     -1.83
## 3   151.   222.    18     1.78      -1.83
## 4   151.   137.    17    -0.264     -1.79
## 5   151.   223.    17     1.81      -1.78
## 6   153.   193.    17     1.09      -1.66
```

```
corr <- cor(numericData)
corr
```

```
##                Height       Weight          Age    WeightSTD   HeightSTD
## Height     1.00000000 -0.063527714 0.025392033 -0.063527714  1.00000000
## Weight    -0.06352771  1.000000000 0.009780534  1.000000000 -0.06352771
## Age        0.02539203  0.009780534 1.000000000  0.009780534  0.02539203
## WeightSTD -0.06352771  1.000000000 0.009780534  1.000000000 -0.06352771
## HeightSTD  1.00000000 -0.063527714 0.025392033 -0.063527714  1.00000000
```

Anything near 1 is perfectly positively correlated: as one variable increases, the other increases; near -1 is perfectly negatively correlated: as one variable increases, the other decreases; and near 0 is that there is no relationship between these variables.

This is a quick way to see which variables would be fine predictors in the linear model, however, we must be concerned with the concept of **multicollinearity**. Multicollinearity occurs when two or more independent variables have a high correlation with one another in a regression model. Variables that are highly correlated may bias the data, predicting values that are higher or lower than the true predicted value. To bypass this, we simply reconsider which variables we want to include. In practive, we remove the variables that are highly correlated to others and leave the more significant ones in the set. For example, we wouldn't want to consider both Weight and the standardized Weight (WeightSTD), why?

```
# We obviously wouldn't implement this as a formal model,
# but for demonstration sake:

summary(lm(Weight ~ WeightSTD, data = cleanedData))
```

```
## Warning in summary.lm(lm(Weight ~ WeightSTD, data = cleanedData)): essentially
## perfect fit: summary may be unreliable
```

```
##
## Call:
## lm(formula = Weight ~ WeightSTD, data = cleanedData)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -4.762e-13  3.200e-16  1.730e-15  3.270e-15  1.630e-14
##
## Coefficients:
```

```
##              Estimate Std. Error   t value Pr(>|t|)
## (Intercept) 1.477e+02  2.413e-15 6.122e+16   <2e-16 ***
## WeightSTD   4.178e+01  2.419e-15 1.727e+16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.412e-14 on 198 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 2.984e+32 on 1 and 198 DF,  p-value: < 2.2e-16
```

```
# We are even given a warning that the fit may be unreliable.
```

Independent Samples T-Test

When performing an independent samples T-test, there are several assumptions we must meet:

```
* The distribution of values in both populations is normally distributed.
* The values in both populations are independent from each other.
* Both populations have the same variance (there is another test we use for unequal variances)
```

1) Testing for Normality:

We test the hypotheses:

H0 : The sample is drawn from a normal distribution. HA : The sample is not drawn from a normal distribution.

I won't bore you with the mathematics, but we can test these hypotheses simply with `shapiro.test(df$Var)`. Since we are testing for significant difference in weight between groups, we check the normality of the Weight.

You should still have two datasets in your workspace from Day 1 (if you saved it). If not, here's the code to save some time:

```
male <- cleanedData %>% filter(Gender == "Male")
female <- cleanedData %>% filter(Gender == "Female")
```

```
shapiro.test(male$Weight)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  male$Weight
## W = 0.9526, p-value = 0.00151
```

```
shapiro.test(female$Weight)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  female$Weight
## W = 0.96149, p-value = 0.004334
```

In the event where our data is *not* normally distributed, like in our case, we typically divert to two methods:

```
a) We can transform the data (typically a Box-Cox transformation -- you would need the `MASS` package --
b) We perform a non-parametric test (like the Wilcoxon Rank-Sum or Kruskal-Wallis test)
```

2) Testing for Homogeneity of Variances (otherwise known as Equal Variances)

A rule of thumb for equal variances is that if the ratio of the larger variance to the smaller variance is less than 4, then the variances are approximately equal. But that takes too much time… We can use Levene's

Test instead. The function `leveneTest(DV ~ IV, data = df)` is found in the `car` package, so we should install and load the `car` package:

```
install.packages('car')
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
library(car)
```

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

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:psych':
##
##     logit
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

The hypotheses of this test are:

H0 : The variance of group 1 is equal to group 2. HA : The variance of group 1 is not equal to group 2.

Again, I won't bore you with the formal mathematics, so let's run the test:

```
leveneTest(Weight ~ Gender, data = cleanedData)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   1   0.026 0.8722
##       198
```

If your variances between groups are not equal (ie. our Pr(>F) < significance level, typically 0.05), that's fine; the unequal variance T-test (also known as Welch's Test) will be performed if `var.equal = FALSE` is included. By default, R assumes the variances are NOT equal, performing Welch's test. That said, you must specify whether the variances are equal (TRUE) but is not a necessary argument if they are assumed to be not equal (from Levene's test).

3) Independence of Groups:

When "testing" for independence for quantitative data, this is just something we assume to be true and argue by fact. In this case, it's pretty simple to understand: the observation of each of these weights are independent – the weight of one individual is not influenced by another individual… ideally.

Ignoring the fact that our normality assumption was violated (and in such case we would either perform a transformation or a nonparametric* test), I still want to show you the syntax involved and the output.

```
t.test(Weight ~ Gender, var.equal = TRUE, data = cleanedData)
```

```
##
##  Two Sample t-test
##
## data:  Weight by Gender
## t = 1.1109, df = 198, p-value = 0.2679
```

```
## alternative hypothesis: true difference in means between group Female and group Male is not equal to
## 95 percent confidence interval:
##  -5.08769 18.21527
## sample estimates:
## mean in group Female   mean in group Male
##             150.9016             144.3378
```

```
# We include var.equal = TRUE since our Levene's test returned not significant.
```

- Nonparametric tests assume that the data is not drawn from a known distribution, ie. there are no parameters (ie. mean, standard deviation) that can be used to model the data. Nonparametric tests often use the median as the measure of central tendency rather than the mean.

To demonstrate the output involved for `var.equal = FALSE` (or if this argument is excluded entirely), we see the test performed is "Welch Two Sample t-test":

```
t.test(Weight ~ Gender, data = cleanedData)
```

```
##
##  Welch Two Sample t-test
##
## data:  Weight by Gender
## t = 1.1112, df = 197.44, p-value = 0.2678
## alternative hypothesis: true difference in means between group Female and group Male is not equal to
## 95 percent confidence interval:
##  -5.085324 18.212903
## sample estimates:
## mean in group Female   mean in group Male
##             150.9016             144.3378
```

Analysis of Variance (ANOVA) + Post-Hoc Testing

The Analysis of Variance (ANOVA) class of models used to analyze the difference in means between more than two groups, an extension of the two-samples T-test. In this case, we'll use our Class variable (since there are 4). The assumptions of the T-test are the same for ANOVA.

1) Normality:

For four or fewer groups, it's the most straightforward to perform four Shapiro tests in R. You should still have the four datasets from Day 1, one for each class standing. If not, here is the code to save some time:

```
freshmen <- cleanedData %>% filter(Class == "Freshman")
sophomores <- cleanedData %>% filter(Class == "Sophomore")
juniors <- cleanedData %>% filter(Class == "Junior")
seniors <- cleanedData %>% filter(Class == "Senior")
```

```
shapiro.test(freshmen$Weight)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  freshmen$Weight
## W = 0.96562, p-value = 0.002459
```

```
shapiro.test(sophomores$Weight)
```

```
##
##  Shapiro-Wilk normality test
##
```

```
## data:  sophomores$Weight
## W = 0.95843, p-value = 0.3006
```

```
shapiro.test(juniors$Weight)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  juniors$Weight
## W = 0.90705, p-value = 0.02257
```

```
shapiro.test(seniors$Weight)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  seniors$Weight
## W = 0.904, p-value = 0.07926
```

Here we see both instances of significance:

The distribution of Weight among freshmen and juniors is significant, implying the distributions do not follow a normal distribution, whereas for sophomores and seniors, they are not significant (implying they do follow a normal distribution) – these conclusions were drawn at the 5% (0.05) significance level. If we were instead testing at the 0.01 significance level, freshmen, juniors, and seniors would be significant.

2) Homogeneity of Variance:

Here, we test the hypotheses:

H0 : The group variances are equal. HA : At least one pair of group variances is not equal.

** The language of the alternate hypothesis is very specific: At least one pair. Levene doesn't care which ones.

```
leveneTest(Weight ~ Class, data = cleanedData)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   3  1.6076 0.1889
##       196
```

Fligner-Killeen's Test:

The Fligner-Killeen's test is one of the many tests for homogeneity of variances which is most robust against departures from normality. It would be appropriate for our example.

The R function `fligner.test()` can be used to compute the test:

```
fligner.test(Weight ~ Class, data = cleanedData)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Weight by Class
## Fligner-Killeen:med chi-squared = 5.9483, df = 3, p-value = 0.1142
```

3) Independence Between Groups

This is something we just assume dependent on the data. In this case, again we assume the observations are independent among groups.

Now we can perform ANOVA now that the assumptions have been examined:

The hypotheses for an ANOVA hypothesis test are:

H0 : The mean measurement of each group is equal. HA : At least one pair of means are not equal. * Again, note the specific language. This will tell us if at least one group is significantly different from the other; we perform a post-hoc test to determine which pairs specifically.

To clarify the types of variables: mean differences in the Weight – a quantitative variable – between groups – a factor. Do to so, we use `aov(DV ~ IV, data = df)` – an anagram for analysis of variance (without the extra "n" and "a")

```
model_AOV <- aov(Weight ~ Class, data = cleanedData)

model_AOV # This output doesn't provide anything useful for any conclusions,
```

```
## Call:
##    aov(formula = Weight ~ Class, data = cleanedData)
##
## Terms:
##                     Class Residuals
## Sum of Squares    8405.5  339037.4
## Deg. of Freedom        3       196
##
## Residual standard error: 41.59065
## Estimated effects may be unbalanced
```

```
          # but we are provided some information:
```

For the test statistic and p-value, we use `summary(model)`, as we did with our linear model:

```
summary(model_AOV)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## Class         3   8405    2802    1.62  0.186
## Residuals   196 339037    1730
```

Here, if we are testing at the 5% significance level (0.05), we would fail to reject the null hypothesis. Our p-value (Pr(>F)) is greater than our level of significance, otherwise known as "alpha".

As we just discussed, the ANOVA model does not provide us the pairs and mean differences. To find this information, we use the Tukey HSD (honestly significant difference) post-hoc test. By default, this will find the 95% family-wise confidence level. In statistics, the family-wise error rate is the likelihood of making at least one Type I error (rejecting the null hypothesis when it is, in fact, true) in studies that test multiple hypotheses or make multiple comparisons.

```
TukeyHSD(model_AOV)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Weight ~ Class, data = cleanedData)
##
## $Class
##                             diff        lwr       upr     p adj
## Sophomore-Freshman -16.6383111 -38.802088  5.525466 0.2127241
## Junior-Freshman      4.6045835 -18.578256 27.787423 0.9555122
## Senior-Freshman      3.9382355 -23.881467 31.757938 0.9830880
## Junior-Sophomore    21.2428945  -7.863848 50.349637 0.2351898
## Senior-Sophomore    20.5765466 -12.342965 53.496059 0.3699525
## Senior-Junior       -0.6663479 -34.280412 32.947716 0.9999512
```

If we want to specify the confidence level (if you were, instead, testing at the 1% significance level):

```
TukeyHSD(model_AOV, conf.level = 0.99)
```

```
##   Tukey multiple comparisons of means
##     99% family-wise confidence level
##
## Fit: aov(formula = Weight ~ Class, data = cleanedData)
##
## $Class
##                            diff       lwr      upr     p adj
## Sophomore-Freshman -16.6383111 -43.61406 10.33743 0.2127241
## Junior-Freshman      4.6045835 -23.61147 32.82064 0.9555122
## Senior-Freshman      3.9382355 -29.92139 37.79786 0.9830880
## Junior-Sophomore    21.2428945 -14.18320 56.66899 0.2351898
## Senior-Sophomore    20.5765466 -19.49011 60.64320 0.3699525
## Senior-Junior       -0.6663479 -41.57835 40.24565 0.9999512
```

There aren't any instances where a pair is significantly different. There are two ways we can determine that:

1) The most straightforward is looking at the adjusted p-value. If it is less than your significance level, the pair is significantly different.
2) We can look at the lower (lwr) and upper (upr) bounds of our confidence interval: If both are positive or both are negative, the pair is significantly different – you are checking to see if 0 is included in this interval. Why?

For two (or more) factors, we add the variables similar to the linear model:

```
model_AOV2 <- aov(Weight ~ Class * Gender, data = cleanedData)
# * for both individual items AND interaction.
summary(model_AOV2)
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## Class          3   8405    2802   1.624  0.185
## Gender         1   2560    2560   1.484  0.225
## Class:Gender   3   5234    1744   1.011  0.389
## Residuals    192 331244    1725
```

If interaction significant, main effects moot

We can still apply the Tukey pairwise comparisons function, but we must specify the factor of interest using the additional argument `which = "Factor"`:

```
TukeyHSD(model_AOV2)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Weight ~ Class * Gender, data = cleanedData)
##
## $Class
##                            diff        lwr       upr     p adj
## Sophomore-Freshman -16.6383111 -38.776867  5.500245 0.2117687
## Junior-Freshman      4.6045835 -18.551876 27.761043 0.9553454
## Senior-Freshman      3.9382355 -23.849811 31.726282 0.9830222
## Junior-Sophomore    21.2428945  -7.830727 50.316516 0.2341911
## Senior-Sophomore    20.5765466 -12.305505 53.458599 0.3688126
## Senior-Junior       -0.6663479 -34.242162 32.909466 0.9999510
```

```
##
## $Gender
##                   diff       lwr      upr     p adj
## Male-Female -7.062945 -18.65414 4.528246 0.2309001
##
## $`Class:Gender`
##                                         diff        lwr       upr     p adj
## Sophomore:Female-Freshman:Female -18.0571938  -54.22989  18.11550 0.7904107
## Junior:Female-Freshman:Female     27.5353037  -19.93760  75.00821 0.6359931
## Senior:Female-Freshman:Female      7.4708169  -37.56945  52.51108 0.9996108
## Freshman:Male-Freshman:Female     -3.4637763  -26.10213  19.17457 0.9997724
## Sophomore:Male-Freshman:Female   -18.3131812  -55.53715  18.91079 0.8025392
## Junior:Male-Freshman:Female       -7.8148497  -41.40686  25.77716 0.9965409
## Senior:Male-Freshman:Female       -3.3136522  -50.78655  44.15925 0.9999989
## Junior:Female-Sophomore:Female    45.5924975  -10.13630 101.32130 0.1986940
## Senior:Female-Sophomore:Female    25.5280108  -28.14366  79.19968 0.8286761
## Freshman:Male-Sophomore:Female    14.5934175  -22.34596  51.53280 0.9279573
## Sophomore:Male-Sophomore:Female   -0.2559874  -47.55976  47.04778 1.0000000
## Junior:Male-Sophomore:Female      10.2423441  -34.25986  54.74454 0.9967663
## Senior:Male-Sophomore:Female      14.7435417  -40.98526  70.47234 0.9923609
## Senior:Female-Junior:Female      -20.0644868  -81.91803  41.78906 0.9748293
## Freshman:Male-Junior:Female      -30.9990800  -79.05874  17.06058 0.5003005
## Sophomore:Male-Junior:Female     -45.8484849 -102.26532  10.56835 0.2057889
## Junior:Male-Junior:Female        -35.3501534  -89.43951  18.73920 0.4826464
## Senior:Male-Junior:Female        -30.8489559  -94.49573  32.79782 0.8144180
## Freshman:Male-Senior:Female      -10.9345932  -56.59289  34.72370 0.9958545
## Sophomore:Male-Senior:Female     -25.7839982  -80.16974  28.60174 0.8310391
## Junior:Male-Senior:Female        -15.2856666  -67.25304  36.68171 0.9856410
## Senior:Male-Senior:Female        -10.7844691  -72.63802  51.06908 0.9994591
## Sophomore:Male-Freshman:Male     -14.8494049  -52.81884  23.12003 0.9315380
## Junior:Male-Freshman:Male         -4.3510734  -38.76731  30.06516 0.9999370
## Senior:Male-Freshman:Male          0.1501242  -47.90953  48.20978 1.0000000
## Junior:Male-Sophomore:Male        10.4983315  -34.86251  55.85917 0.9966512
## Senior:Male-Sophomore:Male        14.9995291  -41.41731  71.41637 0.9921289
## Senior:Male-Junior:Male            4.5011976  -49.58816  58.59056 0.9999964
```

Chi-Square

The final test we'll discuss is the chi-square test. The chi-square test is a hypothesis test used to determine if two categorical (factor) variables are independent. This is known as the Test of Independence or Pearson's Chi-square Test. There are some assumptions we must meet before we can perform this test:

```
1) The data should be able to be represented as count data (frequencies)
2) The data must consist of one categorical variable
    - One categorical variable -- goodness of fit; Two categorical variables -- tests of independence
3) The data must be mutually exclusive -- this is different from independent!*
4) The observations must be independent.
5) The expected frequencies should be at least 5 in 80% of the cells. (Fisher's if not)
6) Ideally, the sample size should be greater than 5 * number of cells.
```

- The difference between Independence and Mutual Exclusivity:
  - Independence is the fundamental concept of influence between data. If we can answer "Have the events or recorded observations been affected by others' events or observations?" This is generally the more difficult concept to understand.

- Mutual exclusivity is the other fundamental concept that describes events that cannot happen at the same time. For purposes of our data, one cannot be both a freshman and a senior at the same time.

Assumptions 1-4 are dependent on the data types themselves. We should just be able to answer these questions "verbally."

For assumption 5, we must run the actual Chi-sq test `chisq.test(x = df$Var1, y = df$Var2)` to check:

```
model_chi <- chisq.test(x = cleanedData$Gender,
                        y = cleanedData$Class)
```

Assumption 5 is directly associated with the expected values. To find this, we run `model_chi$expected`:

```
model_chi$expected
```

```
##                   cleanedData$Class
## cleanedData$Gender Freshman Sophomore Junior Senior
##             Female    65.92    14.935  13.39  8.755
##             Male      62.08    14.065  12.61  8.245
```

For understanding sake, each cell has an expected value based on the total counts of the data. The cells' expected values for each pair can be found by:

(Total of the row * Total of the column) / (Total sample size)

A quick way to find the observed (count distribution) is by `model_chi$observed`:

```
model_chi$observed
```

```
##                   cleanedData$Class
## cleanedData$Gender Freshman Sophomore Junior Senior
##             Female       71        15      8      9
##             Male         57        14     18      8
```

To demonstrate, the expected value for Freshman & Female is:

```
((71 + 57) * (71 + 15 + 8 + 9)) / nrow(cleanedData)
```

```
## [1] 65.92
```

We see that each cell is greater than 5 (obviously satisfying the "at least 80% of cells are greater than 5" condition)

Finally, for assumption 6, a quick way to test is:

Sample Size > (Number of Levels in Factor 1) * (Number of Levels in Factor 2) * 5

```
# 2 levels in Gender, 4 levels in Class:
nrow(cleanedData) > 2 * 4 * 5
```

```
## [1] TRUE
```

Let's take look at the `chisq.test()` output again and make some conclusions:

```
model_chi
```

```
##
##  Pearson's Chi-squared test
##
## data:  cleanedData$Gender and cleanedData$Class
## X-squared = 5.2955, df = 3, p-value = 0.1514
```

The hypotheses for the Chi-square test are:

H0 : There is no significant difference between the observed and expected frequencies. HA : There is a significant difference between the observed and expected frequencies.

In layman's terms, we are testing to see if the frequencies in one group is influenced by the frequencies in another group, ie. the null hypothesis is, in essence, "There is no relationship between the two variables."

In this case, our p-value is 0.1514, which is greater than our significance level of 0.05, thus we fail to reject the null hypothesis. The data do not suggest there is a significant relationship between Gender and Class.

Let's finally apply what we've all learned these last two workshops!

The data set we've been using was all simulated data; there probably won't be a significant pattern for us to discuss, so let's pivot to an applied dataset!

Description of the data set: This data, taken from Kaggle (an open-source, collaborative website to students and data scientists who want to share their work with the world :D), includes MRI scanned data from an Alzheimer's Dementia study. This dataset was created by OASIS (Open-Access Series of Imaging Studies). (Not that it's incredibly important for this workshop, but if you want to try these techniques on your own, Kaggle has thousands of data sets you can download and practice with!)

For more information about this dataset: https://www.oasis-brains.org/#data (This file is OASIS-1)

```
* ID : Patient ID in study
* M/F : Gender (either Male or Female)
* Hand : Dominant handedness (Left or Right)
* Delay : Time between visits
* Age : The patient's age
* Educ : The level of education the patient has received (: less than high school grad., 2: high school
* SES : Socioeconomic status
* MMSE : Mini-mental state examination
* CDR : Classification of Dementia (0 = Non, 0.5 = Very mild, 1 = Mild, and 2 = Moderate)
* eTIV : Estimated total intracranial volume
* nWBV : Normalized whole brain volume
* ASF : Atlas scaling factor
```

I've provided the list of variables above and a quick description of what is represents.

Q: Which variables should be factors?

A: M/F, Education, Hand, SES, and CDR.

You should have the `oasis_cross-sectional.csv` dataset available to you, so let's import the data using `read_csv()`. Use `spec()` to see the data specifications and change the appropriate variables to factors:

```
oasis_raw <- read_csv('oasis_cross-sectional.csv', col_types = cols(
    ID = col_character(),
    `M/F` = col_character(),
    Hand = col_character(),
    Age = col_double(),
    Educ = col_double(),
    SES = col_double(),
    MMSE = col_double(),
    CDR = col_factor(),
    eTIV = col_double(),
    nWBV = col_double(),
    ASF = col_double(),
    Delay = col_character()
  )
)
```

Let's rename the Gender column:

```
colnames(oasis_raw)
```

```
##  [1] "ID"    "M/F"   "Hand"  "Age"   "Educ"  "SES"   "MMSE"  "CDR"   "eTIV"
## [10] "nWBV"  "ASF"   "Delay"
```

```
# We see the "M/F" variable, which we want to rename "Gender", is the second
# item in the list. We can access and change this variable
# name using the brackets, like in Day 1:

colnames(oasis_raw)[2] <- "Gender"
colnames(oasis_raw)
```

```
##  [1] "ID"     "Gender" "Hand"   "Age"    "Educ"   "SES"    "MMSE"   "CDR"
##  [9] "eTIV"   "nWBV"   "ASF"    "Delay"
```

Check above to see the CDR definitions (0 = Non-demented, etc.). Change the factors to their appropriate name, no longer having a number associated with the dementia classification:

```
# Tip: Use levels(oasis_raw$CDR). Make sure to check the order!
levels(oasis_raw$CDR)
```

```
## [1] "0"   "0.5" "1"   "2"
```

```
levels(oasis_raw$CDR) <- c('Non-demented', 'Very Mildly Demented',
                           'Mildly Demented', 'Moderately Demented')
levels(oasis_raw$CDR)
```

```
## [1] "Non-demented"         "Very Mildly Demented" "Mildly Demented"
## [4] "Moderately Demented"
```

Now that we've done some "cleaning," let's find the descriptive statistics. To begin, let's find the overall statistics:

```
oasis_raw %>% describe()
```

```
##         vars   n    mean     sd  median trimmed    mad     min     max  range
## ID*        1 436  218.50 126.01  218.50  218.50 161.60    1.00  436.00 435.00
## Gender*    2 436    1.39   0.49    1.00    1.36   0.00    1.00    2.00   1.00
## Hand*      3 436    1.00   0.00    1.00    1.00   0.00    1.00    1.00   0.00
## Age        4 436   51.36  25.27   54.00   50.83  38.55   18.00   96.00  78.00
## Educ       5 235    3.18   1.31    3.00    3.22   1.48    1.00    5.00   4.00
## SES        6 216    2.49   1.12    2.00    2.47   1.48    1.00    5.00   4.00
## MMSE       7 235   27.06   3.70   29.00   27.77   1.48   14.00   30.00  16.00
## CDR*       8 235    1.56   0.73    1.00    1.44   0.00    1.00    4.00   3.00
## eTIV       9 436 1481.92 158.74 1475.50 1477.83 156.41 1123.00 1992.00 869.00
## nWBV      10 436    0.79   0.06    0.81    0.80   0.06    0.64    0.89   0.25
## ASF       11 436    1.20   0.13    1.19    1.19   0.13    0.88    1.56   0.68
## Delay*    12 436   14.65   1.83   15.00   15.00   0.00    1.00   15.00  14.00
##           skew kurtosis   se
## ID*       0.00    -1.21 6.03
## Gender*   0.47    -1.78 0.02
## Hand*      NaN      NaN 0.00
## Age       0.00    -1.60 1.21
## Educ     -0.01    -1.24 0.09
## SES       0.16    -1.16 0.08
## MMSE     -1.59     1.85 0.24
## CDR*      1.02     0.04 0.05
```

```
## eTIV     0.26    -0.18 7.60
## nWBV    -0.52    -0.90 0.00
## ASF      0.28    -0.20 0.01
## Delay*  -5.66    32.54 0.09
```

```
oasis_raw %>% describeBy("CDR")
```

```
##
##  Descriptive statistics by group
## CDR: 1
##        vars   n    mean     sd  median trimmed    mad     min     max range
## ID        1 135  218.33 124.65  219.00  218.21 158.64    1.00  436.00 435.0
## Gender    2 135    1.28   0.45    1.00    1.23   0.00    1.00    2.00   1.0
## Hand      3 135    1.00   0.00    1.00    1.00   0.00    1.00    1.00   0.0
## Age       4 135   69.07  13.88   71.00   69.45  14.83   33.00   94.00  61.0
## Educ      5 135    3.44   1.26    4.00    3.50   1.48    1.00    5.00   4.0
## SES       6 133    2.33   1.06    2.00    2.28   1.48    1.00    5.00   4.0
## MMSE      7 135   29.10   1.13   29.00   29.30   1.48   25.00   30.00   5.0
## CDR       8 135    1.00   0.00    1.00    1.00   0.00    1.00    1.00   0.0
## eTIV      9 135 1441.53 152.36 1415.00 1432.50 131.95 1123.00 1818.00 695.0
## nWBV     10 135    0.77   0.05    0.77    0.77   0.05    0.64    0.85   0.2
## ASF      11 135    1.23   0.13    1.24    1.23   0.12    0.96    1.56   0.6
## Delay    12 135   15.00   0.00   15.00   15.00   0.00   15.00   15.00   0.0
##         skew kurtosis    se
## ID     -0.04    -1.13 10.73
## Gender  0.96    -1.08  0.04
## Hand     NaN      NaN  0.00
## Age    -0.25    -0.82  1.19
## Educ   -0.23    -1.15  0.11
## SES     0.33    -0.99  0.09
## MMSE   -1.40     1.60  0.10
## CDR      NaN      NaN  0.00
## eTIV    0.51    -0.19 13.11
## nWBV   -0.43    -0.64  0.00
## ASF     0.00    -0.24  0.01
## Delay    NaN      NaN  0.00
## ------------------------------------------------------------
## CDR: 2
##        vars  n    mean     sd  median trimmed    mad     min    max  range
## ID        1 70  222.80 129.79  235.00  223.09 155.67    3.00  433.0 430.00
## Gender    2 70    1.44   0.50    1.00    1.43   0.00    1.00    2.0   1.00
## Hand      3 70    1.00   0.00    1.00    1.00   0.00    1.00    1.0   0.00
## Age       4 70   76.21   7.19   76.00   76.04   7.41   62.00   92.0  30.00
## Educ      5 70    2.94   1.28    3.00    2.93   1.48    1.00    5.0   4.00
## SES       6 57    2.67   1.12    3.00    2.68   1.48    1.00    5.0   4.00
## MMSE      7 70   25.64   3.50   27.00   26.04   2.97   14.00   30.0  16.00
## CDR       8 70    2.00   0.00    2.00    2.00   0.00    2.00    2.0   0.00
## eTIV      9 70 1485.37 186.61 1457.50 1473.52 159.38 1171.00 1992.0 821.00
## nWBV     10 70    0.73   0.04    0.73    0.73   0.04    0.64    0.8   0.15
## ASF      11 70    1.20   0.14    1.20    1.20   0.14    0.88    1.5   0.62
## Delay    12 70   15.00   0.00   15.00   15.00   0.00   15.00   15.0   0.00
##         skew kurtosis    se
## ID     -0.07    -1.18 15.51
## Gender  0.23    -1.98  0.06
## Hand     NaN      NaN  0.00
```

```
## Age     0.16    -0.59  0.86
## Educ    0.23    -1.17  0.15
## SES    -0.08    -1.18  0.15
## MMSE   -1.00     0.55  0.42
## CDR      NaN      NaN  0.00
## eTIV    0.59    -0.17 22.30
## nWBV   -0.26    -0.42  0.00
## ASF    -0.04    -0.54  0.02
## Delay    NaN      NaN  0.00
## ------------------------------------------------------------
## CDR: 3
##        vars  n    mean     sd  median trimmed    mad     min     max  range
## ID        1 28  221.79 143.21  236.00  221.42 209.05   26.00  431.00 405.00
## Gender    2 28    1.32   0.48    1.00    1.29   0.00    1.00    2.00   1.00
## Hand      3 28    1.00   0.00    1.00    1.00   0.00    1.00    1.00   0.00
## Age       4 28   77.75   6.99   78.00   77.54   8.15   65.00   96.00  31.00
## Educ      5 28    2.57   1.32    2.00    2.50   1.48    1.00    5.00   4.00
## SES       6 24    2.88   1.30    3.00    2.90   1.48    1.00    5.00   4.00
## MMSE      7 28   21.68   3.75   22.00   21.67   2.97   15.00   29.00  14.00
## CDR       8 28    3.00   0.00    3.00    3.00   0.00    3.00    3.00   0.00
## eTIV      9 28 1481.64 120.82 1490.00 1478.29 111.94 1274.00 1732.00 458.00
## nWBV     10 28    0.71   0.03    0.70    0.71   0.03    0.66    0.76   0.11
## ASF      11 28    1.19   0.10    1.18    1.19   0.09    1.01    1.38   0.36
## Delay    12 28   15.00   0.00   15.00   15.00   0.00   15.00   15.00   0.00
##         skew kurtosis    se
## ID      0.00    -1.61 27.06
## Gender  0.72    -1.53  0.09
## Hand     NaN      NaN  0.00
## Age     0.42    -0.12  1.32
## Educ    0.51    -1.07  0.25
## SES    -0.24    -1.51  0.26
## MMSE    0.06    -0.69  0.71
## CDR      NaN      NaN  0.00
## eTIV    0.23    -0.65 22.83
## nWBV    0.32    -1.04  0.01
## ASF     0.10    -0.74  0.02
## Delay    NaN      NaN  0.00
## ------------------------------------------------------------
## CDR: 4
##        vars n    mean    sd  median trimmed   mad     min     max  range skew
## ID        1 2  314.50 27.58  314.50  314.50 28.91  295.00  334.00  39.00    0
## Gender    2 2    1.50  0.71    1.50    1.50  0.74    1.00    2.00   1.00    0
## Hand      3 2    1.00  0.00    1.00    1.00  0.00    1.00    1.00   0.00  NaN
## Age       4 2   82.00  5.66   82.00   82.00  5.93   78.00   86.00   8.00    0
## Educ      5 2    2.00  1.41    2.00    2.00  1.48    1.00    3.00   2.00    0
## SES       6 2    3.50  0.71    3.50    3.50  0.74    3.00    4.00   1.00    0
## MMSE      7 2   15.00  0.00   15.00   15.00  0.00   15.00   15.00   0.00  NaN
## CDR       8 2    4.00  0.00    4.00    4.00  0.00    4.00    4.00   0.00  NaN
## eTIV      9 2 1456.50 78.49 1456.50 1456.50 82.28 1401.00 1512.00 111.00    0
## nWBV     10 2    0.68  0.03    0.68    0.68  0.03    0.66    0.70   0.04    0
## ASF      11 2    1.21  0.07    1.21    1.21  0.07    1.16    1.25   0.09    0
## Delay    12 2   15.00  0.00   15.00   15.00  0.00   15.00   15.00   0.00  NaN
##        kurtosis    se
## ID        -2.75 19.50
```

```
## Gender    -2.75  0.50
## Hand        NaN  0.00
## Age       -2.75  4.00
## Educ      -2.75  1.00
## SES       -2.75  0.50
## MMSE        NaN  0.00
## CDR         NaN  0.00
## eTIV      -2.75 55.50
## nWBV      -2.75  0.02
## ASF       -2.75  0.05
## Delay       NaN  0.00
```

If we examine these descriptive statistics further, we see that there are 135 non-demented, 70 very mildly demented, 28 mildly demented, and 2 moderately demented. I think we would all agree these sample sizes very different. It may be advantageous to group the non-demented and demented (very mildly, mildly, and moderately) patients.

Try mutating the CDR variable to have Non-demented patients remain in their own group and another other patient be grouped as Demented. Tip: Use an `ifelse()` command within mutate. Don't overthink it! Remember to sound out what you want – the code will practically write itself for you:

```
oasis_raw <- oasis_raw %>% mutate(CDR2 = ifelse(CDR == "Non-demented",
                                                # If CDR is "Non-demented"
                                                "Non-demented",
                                                # Store "Non-demented"
                                                "Demented"))
                                                # Otherwise, store "Demented"

head(oasis_raw)
```

```
## # A tibble: 6 x 13
##   ID    Gender Hand    Age  Educ   SES  MMSE CDR    eTIV  nWBV   ASF Delay CDR2
##   <chr> <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <fct> <dbl> <dbl> <dbl> <chr> <chr>
## 1 OAS1~ F      R        74     2     3    29 Non-~  1344 0.743  1.31 N/A   Non-~
## 2 OAS1~ F      R        55     4     1    29 Non-~  1147 0.81   1.53 N/A   Non-~
## 3 OAS1~ F      R        73     4     3    27 Very~  1454 0.708  1.21 N/A   Deme~
## 4 OAS1~ M      R        28    NA    NA    NA <NA>   1588 0.803  1.10 N/A   <NA>
## 5 OAS1~ M      R        18    NA    NA    NA <NA>   1737 0.848  1.01 N/A   <NA>
## 6 OAS1~ F      R        24    NA    NA    NA <NA>   1131 0.862  1.55 N/A   <NA>
```

```
# Remember what happens when we mutate! Be sure to check
# the data type and coerce when appropriate!

oasis_raw$CDR2 <- as.factor(oasis_raw$CDR2)
head(oasis_raw)
```

```
## # A tibble: 6 x 13
##   ID    Gender Hand    Age  Educ   SES  MMSE CDR    eTIV  nWBV   ASF Delay CDR2
##   <chr> <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <fct> <dbl> <dbl> <dbl> <chr> <fct>
## 1 OAS1~ F      R        74     2     3    29 Non-~  1344 0.743  1.31 N/A   Non-~
## 2 OAS1~ F      R        55     4     1    29 Non-~  1147 0.81   1.53 N/A   Non-~
## 3 OAS1~ F      R        73     4     3    27 Very~  1454 0.708  1.21 N/A   Deme~
## 4 OAS1~ M      R        28    NA    NA    NA <NA>   1588 0.803  1.10 N/A   <NA>
## 5 OAS1~ M      R        18    NA    NA    NA <NA>   1737 0.848  1.01 N/A   <NA>
## 6 OAS1~ F      R        24    NA    NA    NA <NA>   1131 0.862  1.55 N/A   <NA>
```

The syntax included in the `lm()` function required a response (dependent) variable by (using ~) the sum

of your independent variables: `lm(Dependent ~ IndVar1 + IndVar2, data = df)`. If we wanted to model eTIV by nWBV:

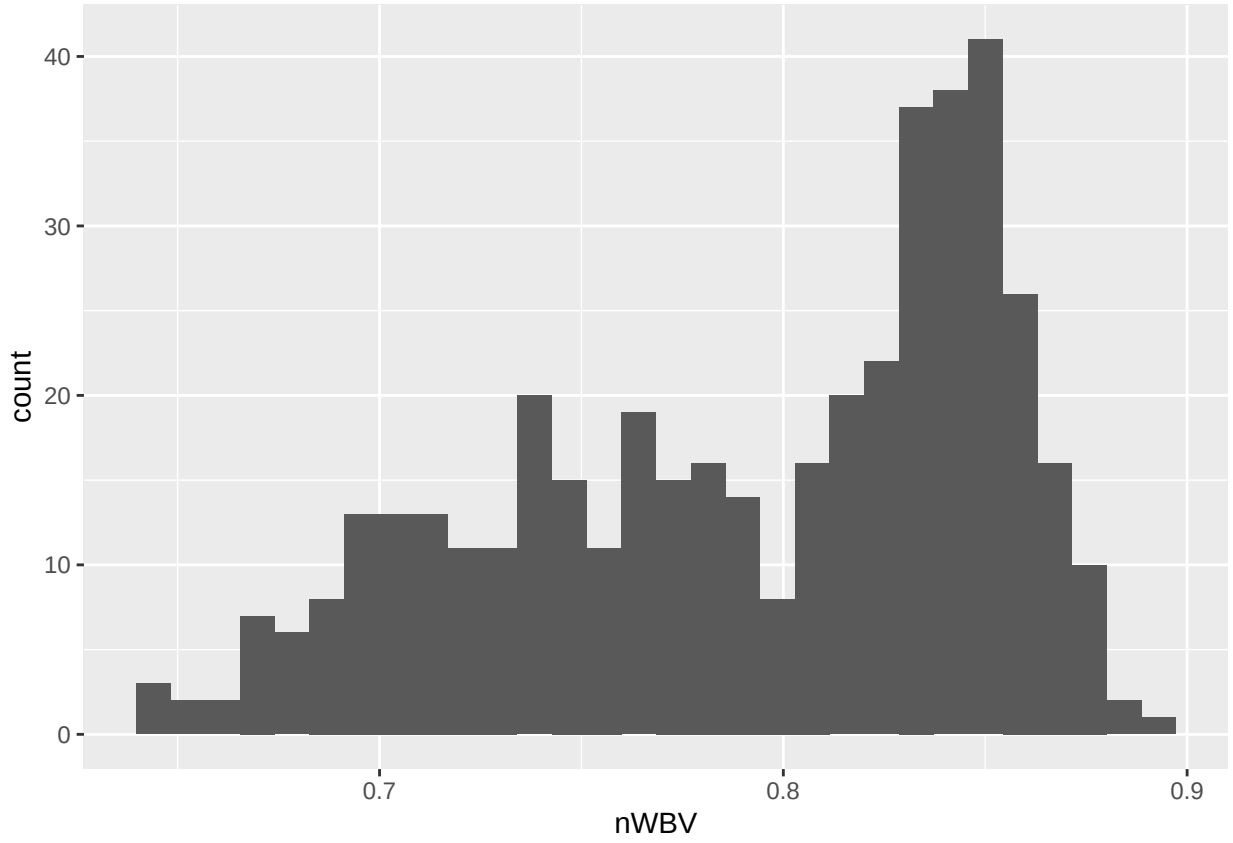
```
model1 <- lm(eTIV ~ nWBV, data = oasis_raw)
model1
```

```
##
## Call:
## lm(formula = eTIV ~ nWBV, data = oasis_raw)
##
## Coefficients:
## (Intercept)         nWBV
##     1502.23       -25.65
```

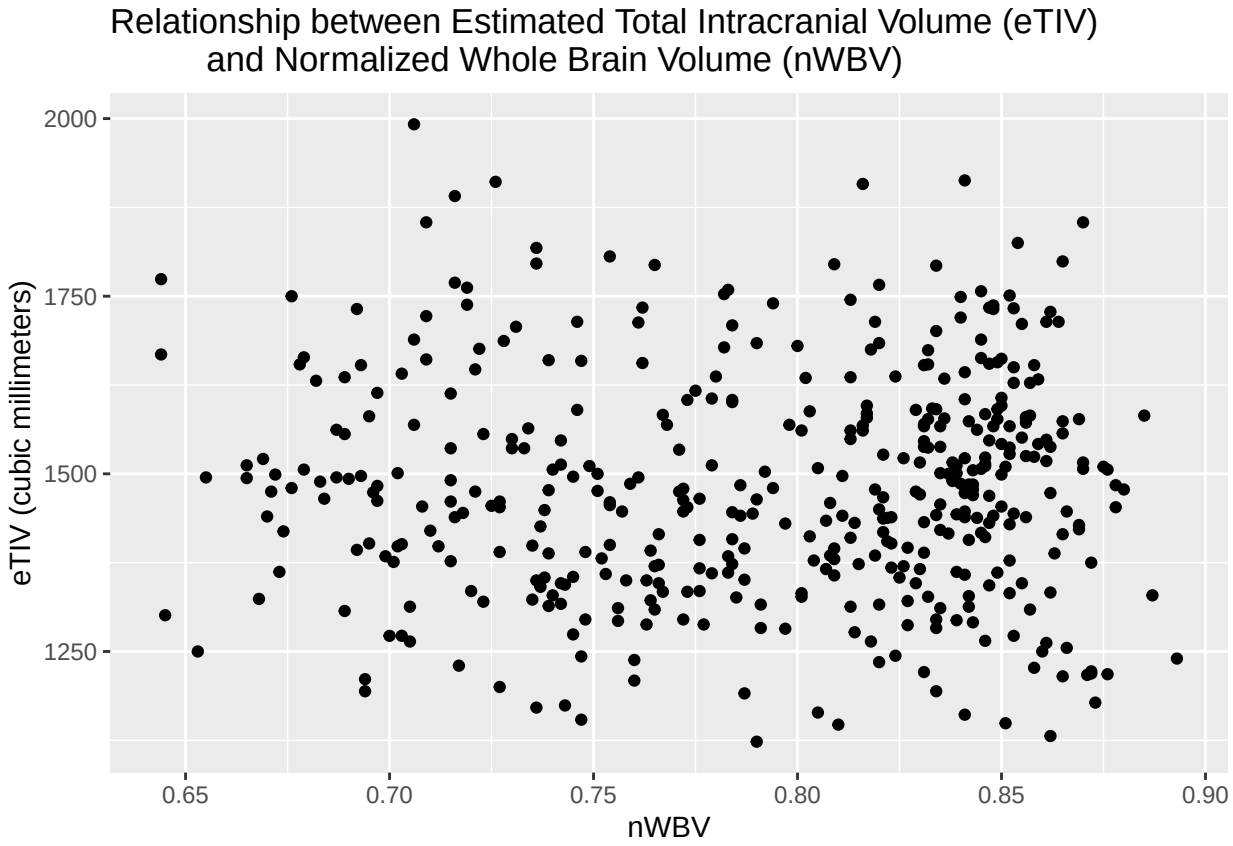
1) What does the distribution of eTIV by nWBV look like? (What kind of variable types are they, and which plot can accurately represent them?)

```
oasis_raw %>% ggplot() + geom_histogram(mapping = aes(x = nWBV))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
oasis_raw %>% ggplot() +
  geom_point(aes(x = nWBV, y = eTIV)) +
  xlab('nWBV') +
  ylab('eTIV (cubic millimeters)') +
  ggtitle(label = "Relationship between Estimated Total Intracranial Volume (eTIV)
          and Normalized Whole Brain Volume (nWBV)")
```




```
model_4 <- lm(eTIV ~ nWBV, data = oasis_raw)
summary(model_4)
```

```
##
## Call:
## lm(formula = eTIV ~ nWBV, data = oasis_raw)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -358.96 -113.57   -6.62   97.73  507.88
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1502.23     100.93  14.884   <2e-16 ***
## nWBV          -25.65     127.12  -0.202     0.84
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 158.9 on 434 degrees of freedom
## Multiple R-squared:  9.38e-05,   Adjusted R-squared:  -0.00221
## F-statistic: 0.04071 on 1 and 434 DF,  p-value: 0.8402
```

Are Gender and CDR related?

Q: What data types are Gender and CDR?

A: These are both factors/categorical variables.

Since we have two _____ variables, we use the Chi-square test for independence.

```r
chisq.test(x = oasis_raw$Gender, y = oasis_raw$CDR2)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  oasis_raw$Gender and oasis_raw$CDR2
## X-squared = 3.6955, df = 1, p-value = 0.05456
```

Correlations:

```r
# First, we select the numeric data: sapply(x, is.numeric)
is.num <- sapply(oasis_raw, is.numeric)
oasis_numeric <- oasis_raw[, is.num] # Select the numeric columns

# We have a bunch of NA values; to remove the NA values
# for the numeric data, we include some logic.
# The function `is.na(x)` will return a boolean vector,
# where FALSE -> the observation is not "NA"; TRUE -> the observation is "NA"

# Note: We associate 0 to be False and 1 to be True,
# so when using colSum(is.na(x)), we map True to 1 and sum
# to see how many observations were NA.
is.na(oasis_numeric)
```

```
##          Age  Educ   SES  MMSE  eTIV  nWBV   ASF
##   [1,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##   [2,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##   [3,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##   [4,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##   [5,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##   [6,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##   [7,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##   [8,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##   [9,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [10,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [11,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [12,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [13,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [14,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##  [15,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [16,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [17,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [18,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [19,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [20,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [21,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [22,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [23,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [24,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [26,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [27,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [28,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [29,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [30,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
##  [31,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [32,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [33,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [34,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [35,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [36,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [38,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##  [39,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [40,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [41,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [42,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [43,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##  [44,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [45,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [46,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [47,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [48,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [50,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [51,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [52,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [53,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [54,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [55,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [56,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##  [57,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [58,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [59,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [60,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [62,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [63,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [64,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [65,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [66,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [67,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [68,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [69,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [70,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##  [71,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [72,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [73,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [74,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [75,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [76,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [77,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [78,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [79,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [80,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##  [81,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [82,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [83,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [84,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

```
##  [85,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [86,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [87,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [88,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [89,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [90,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [91,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [92,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##  [93,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [94,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [95,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [96,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [97,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [98,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [99,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [100,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [101,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [102,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [103,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [104,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [105,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [106,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [107,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [108,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [110,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [111,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [112,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [113,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [114,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [115,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [116,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [117,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [118,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [119,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [120,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [121,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [122,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [123,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [124,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [125,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [126,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [127,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [128,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [129,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [130,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [131,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [132,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [133,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [134,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [135,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [136,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [137,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [138,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

```
## [139,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [140,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [141,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [142,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [143,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [144,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [145,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [146,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [147,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [148,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [149,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [150,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [151,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [152,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [153,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [154,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [155,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [156,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [157,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [158,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [159,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [160,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [161,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [162,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [163,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [164,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [165,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [166,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [167,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [168,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [169,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [170,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [171,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [172,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [173,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [174,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [175,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [176,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [177,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [178,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [179,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [180,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [181,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [182,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [183,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [184,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [185,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [186,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [187,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [188,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [189,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [190,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [191,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [192,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

```
## [193,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [194,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [195,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [196,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [197,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [198,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [199,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [200,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [201,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [202,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [203,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [204,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [205,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [206,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [207,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [208,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [209,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [210,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [211,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [212,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [213,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [214,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [215,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [216,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [217,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [218,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [219,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [220,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [221,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [222,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [223,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [224,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [225,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [226,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [227,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [228,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [229,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [230,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [231,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [232,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [233,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [234,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [235,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [236,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [237,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [238,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [239,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [240,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [241,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [242,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [243,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [244,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [245,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [246,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [247,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [248,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [249,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [250,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [251,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [252,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [253,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [254,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [255,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [256,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [257,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [258,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [259,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [260,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [261,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [262,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [263,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [264,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [265,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [266,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [267,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [268,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [269,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [270,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [271,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [272,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [273,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [274,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [275,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [276,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [277,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [278,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [279,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [280,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [281,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [282,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [283,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [284,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [285,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [286,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [287,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [288,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [289,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [290,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [291,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [292,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [293,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [294,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [295,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [296,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [297,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [298,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [299,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [300,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

```
## [301,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [302,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [303,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [304,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [305,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [306,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [307,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [308,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [309,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [310,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [311,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [312,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [313,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [314,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [315,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [316,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [317,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [318,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [319,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [320,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [321,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [322,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [323,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [324,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [325,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [326,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [327,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [328,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [329,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [330,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [331,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [332,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [333,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [334,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [335,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [336,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [337,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [338,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [339,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [340,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [341,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [342,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [343,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [344,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [345,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [346,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [347,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [348,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [349,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [350,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [351,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [352,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [353,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [354,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

```
## [355,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [356,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [357,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [358,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [359,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [360,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [361,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [362,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [363,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [364,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [365,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [366,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [367,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [368,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [369,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [370,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [371,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [372,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [373,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [374,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [375,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [376,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [377,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [378,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [379,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [380,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [381,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [382,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [383,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [384,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [385,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [386,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [387,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [388,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [389,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [390,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [391,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [392,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [393,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [394,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [395,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [396,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [397,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [398,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [399,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [400,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [401,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [402,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [403,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [404,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [405,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [406,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [407,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [408,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [409,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [410,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [411,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [412,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [413,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [414,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [415,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [416,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [417,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [418,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [419,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [420,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [421,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [422,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [423,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [424,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [425,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [426,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [427,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [428,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [429,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [430,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [431,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [432,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [433,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [434,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [435,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [436,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

```
colSums(is.na(oasis_numeric))
```

```
##  Age Educ  SES MMSE eTIV nWBV  ASF
##    0  201  220  201    0    0    0
```

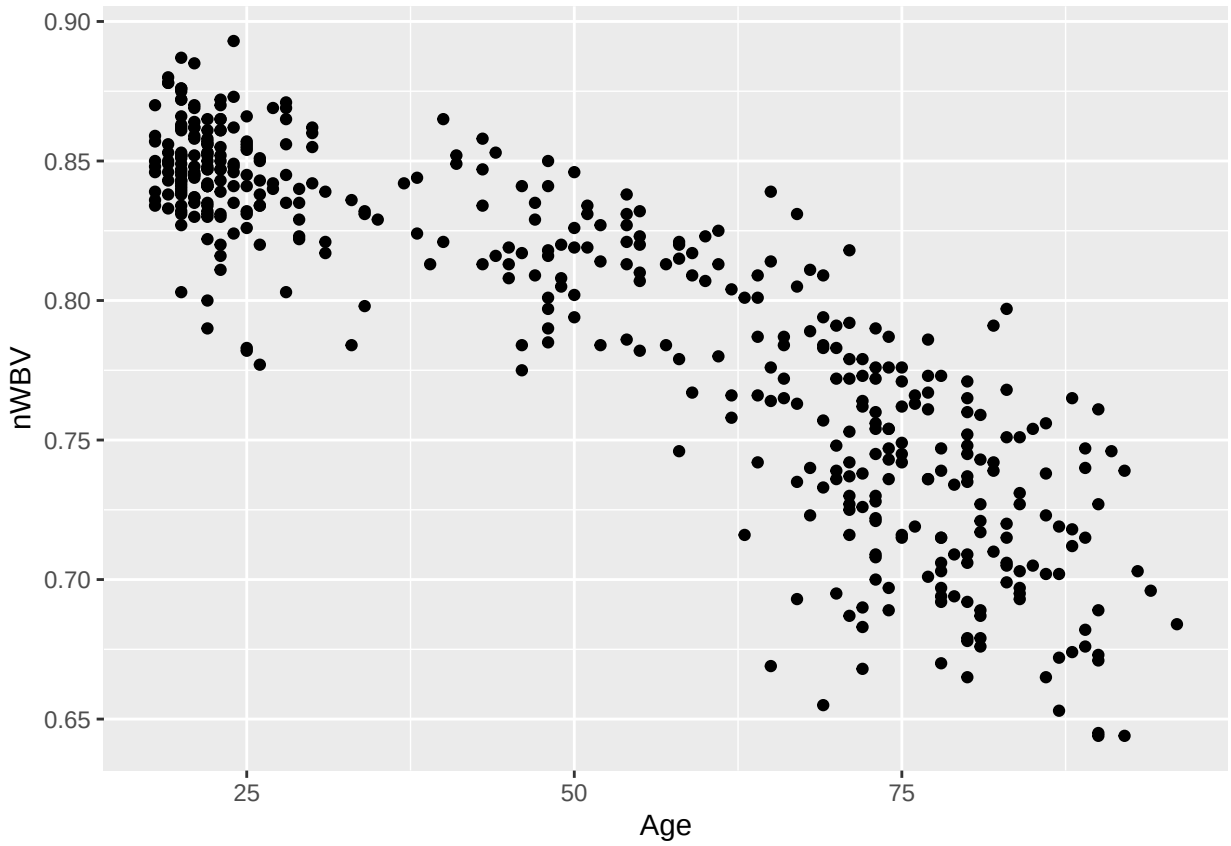
```
# There are 220 observations in SES that are NA and 201 observations in MMSE.
# We can't pair these with other obsevations, so we lose
# this information in our correlation plot.

# We use the colSums(is.na(oasis_numeric)) and "filter"
# the columns that have 0 NA values:
oasis_numeric2 <- oasis_numeric[, colSums(is.na(oasis_numeric)) == 0]

cor(oasis_numeric2)
```

```
##              Age         eTIV         nWBV        ASF
## Age   1.0000000 -0.140984368 -0.874100069  0.1375142
## eTIV -0.1409844  1.000000000 -0.009684861 -0.9756661
## nWBV -0.8741001 -0.009684861  1.000000000  0.0137713
## ASF   0.1375142 -0.975666144  0.013771295  1.0000000
```

We see that Age and nWBV are strongly negatively associated, as well as eTIV and ASF. Let's plot the pairs and see what that looks like:

First, plot Age and nWBV (two quantitative variables, we use scatterplot):

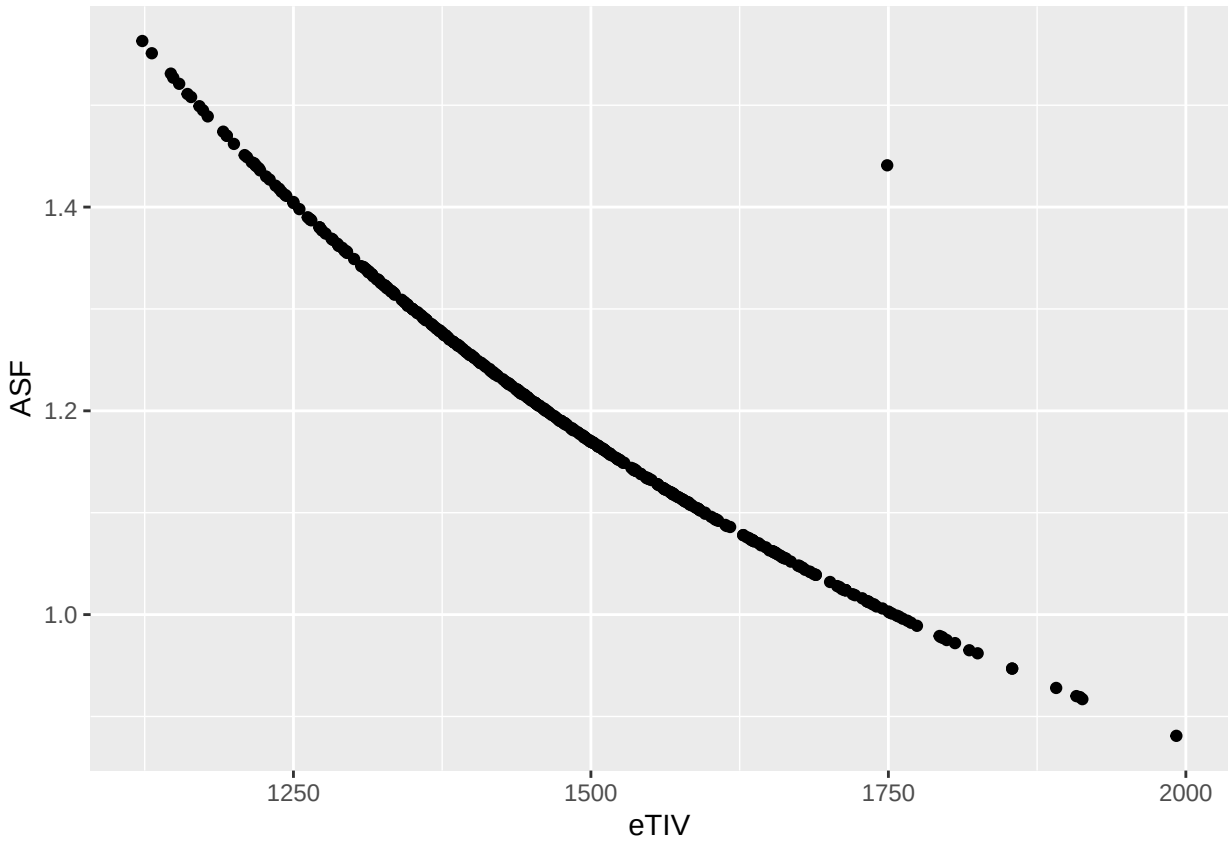
```
oasis_numeric2 %>% ggplot() +
  geom_point(mapping = aes(x = Age, y = nWBV))
```



There is a very clear pattern; as Age increases, the normalized whole brain volume decreases.

For eTIV and ASF, we will see something similar:

```
oasis_numeric2 %>% ggplot() +
  geom_point(mapping = aes(x = eTIV, y = ASF))
```



What's so interesting about this graph?

This should hopefully provide insight as to why we use our graphics, as well as numeric associations (correlations), to paint the whole picture; each individually provides part, but together they are very powerful. For demonstration, examine the summary of the linear model below:

```
summary(lm(eTIV ~ ASF, data = oasis_raw))
```

```
##
## Call:
## lm(formula = eTIV ~ ASF, data = oasis_raw)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -19.13 -16.44 -10.23   6.17 558.47
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2924.87      15.65   186.8   <2e-16 ***
## ASF         -1203.57      12.98   -92.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 34.85 on 434 degrees of freedom
## Multiple R-squared:  0.9519, Adjusted R-squared:  0.9518
## F-statistic:  8593 on 1 and 434 DF,  p-value: < 2.2e-16
```

```
# Here, a warning doesn't populate, but we should be careful. The Atlas Scaling
# Factor is defined as a normalization technique to measure the standardized
```

```
# total intracranial volume for comparison, classification and predication.
# Since the ASF is just a scaled version of the total intracranial volume,
# of course they are linearly related. That one observation does
# spark some interest/concern, though...
```

1) Create a linear model that predicts the normalized whole brain volume by age (nWBV ~ Age):

```
lm_oasis1 <- lm(nWBV ~ Age, dat = oasis_raw)
summary(lm_oasis1)
```

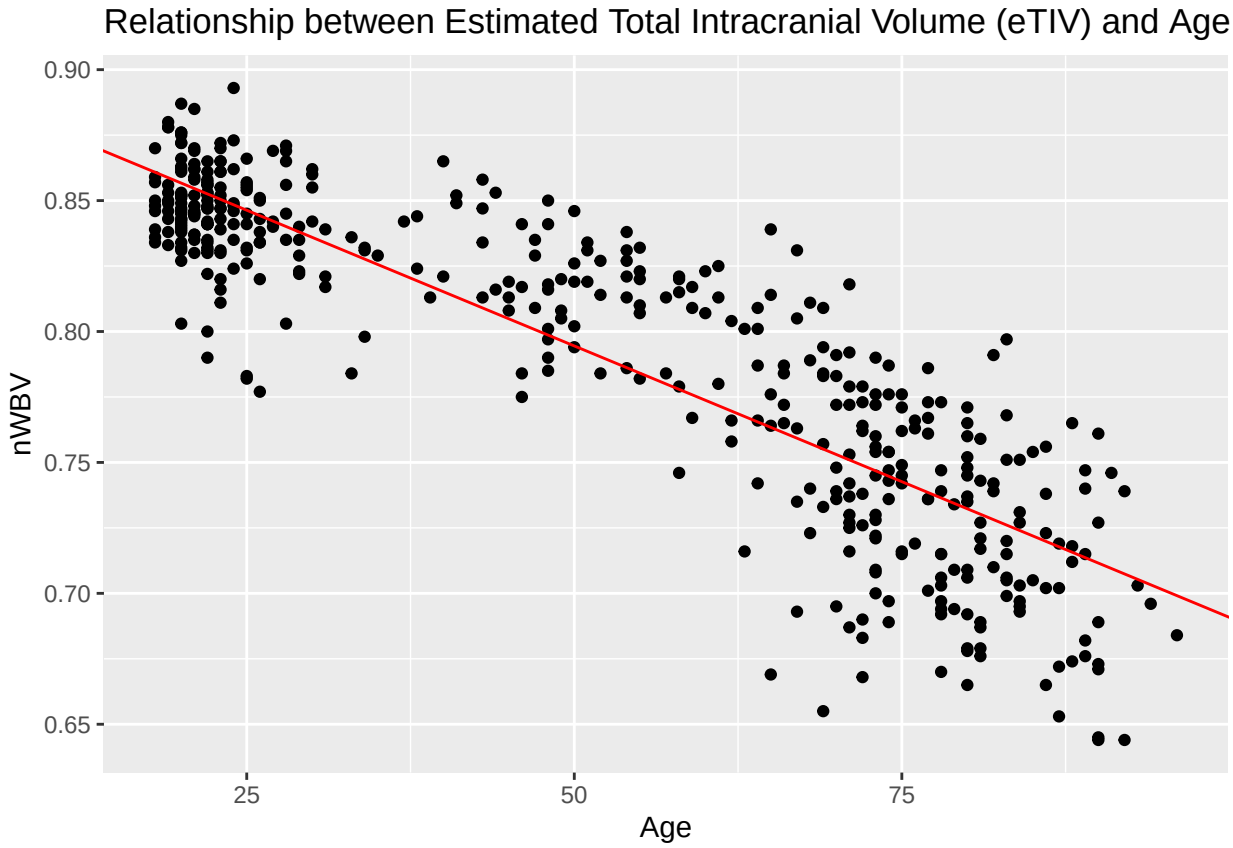
```
##
## Call:
## lm(formula = nWBV ~ Age, data = oasis_raw)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.100093 -0.017628  0.000502  0.021101  0.075614
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.8981476  0.0031648  283.80   <2e-16 ***
## Age         -0.0020733  0.0000553  -37.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02915 on 434 degrees of freedom
## Multiple R-squared:  0.7641, Adjusted R-squared:  0.7635
## F-statistic:  1405 on 1 and 434 DF,  p-value: < 2.2e-16
```

We can add the line of best fit to our scatterplot from above! We add another `geom` to our `ggplot()`. In this case, we have a function of the form

$$y = ax + b$$

2) Add the line of best fit (from the coefficients) to our plot. To do so, use `geom_abline(intercept = b, slope = a)`:

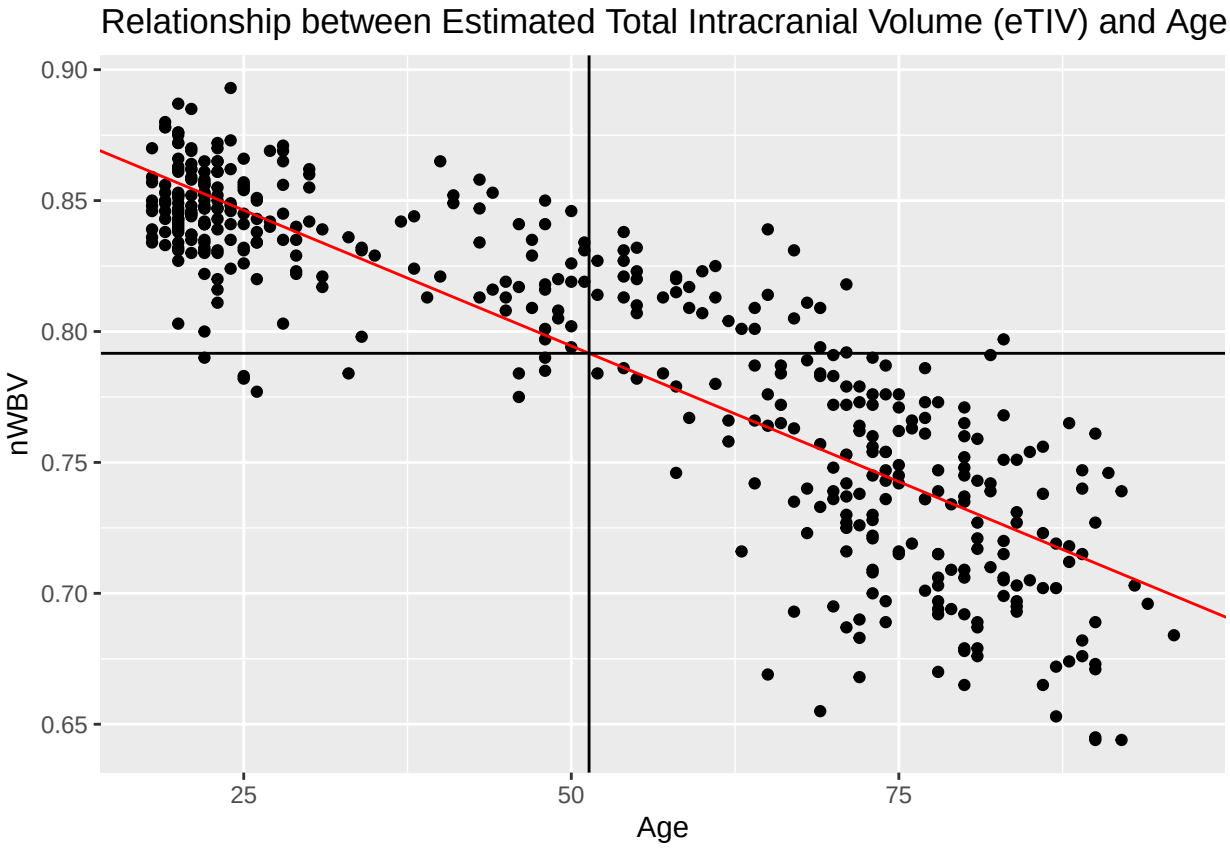
```
oasis_raw %>% ggplot() +
  geom_point(mapping = aes(x = Age, y = nWBV)) +
  geom_abline(intercept = lm_oasis1$coefficients[[1]],
              slope = lm_oasis1$coefficients[[2]],
              color = "red") +
  ggtitle(label = "Relationship between Estimated Total Intracranial Volume (eTIV) and Age")
```




```
* `geom_vline()` will plot a vertical line, otherwise known as the x-intercept.
* `geom_hline()` will plot a horizontal line, otherwise known as the y-intercept.
```

It might not be incredibly common, but a technique or use is including a line to represent the mean of each sample:

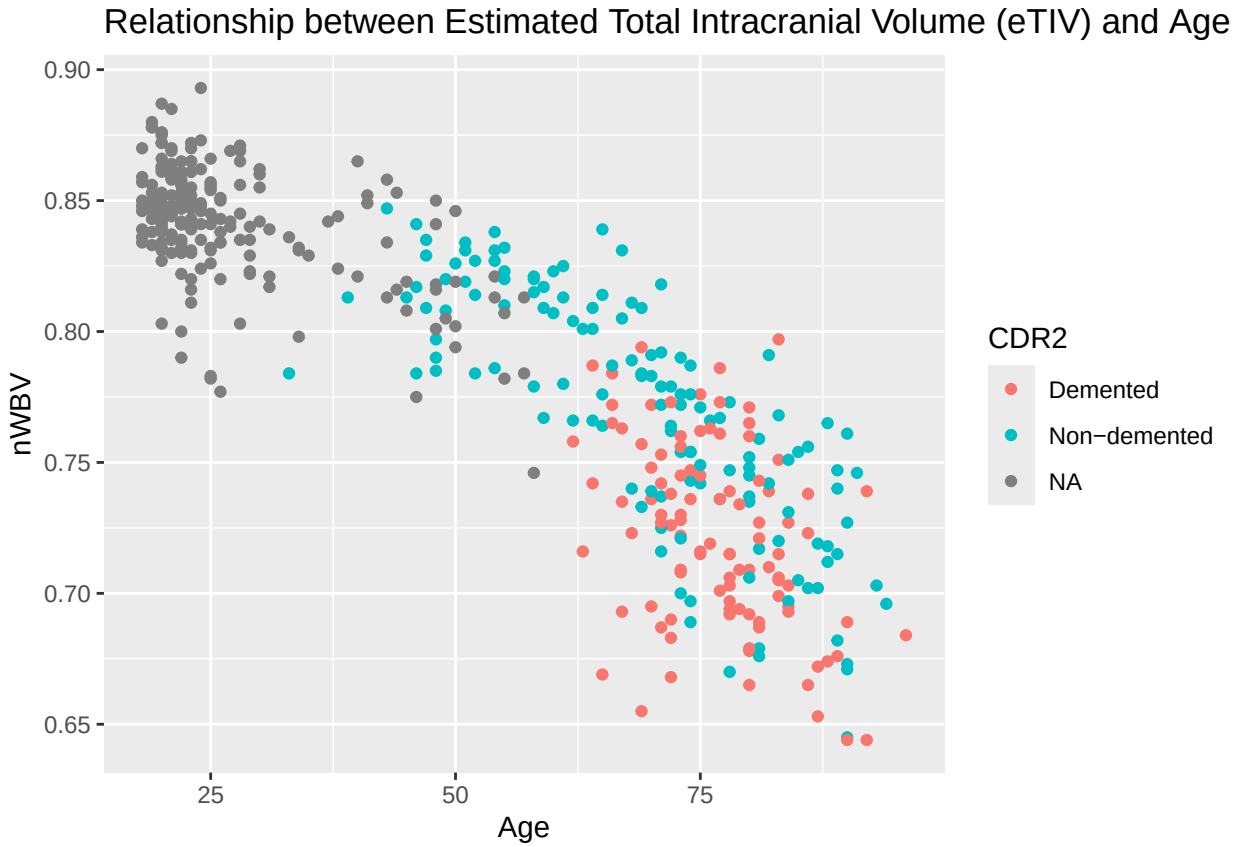
```
oasis_raw %>% ggplot() +
  geom_point(mapping = aes(x = Age, y = nWBV)) +
  geom_abline(intercept = lm_oasis1$coefficients[[1]],
              slope = lm_oasis1$coefficients[[2]],
              color = "red") +
  geom_vline(xintercept = mean(oasis_raw$Age)) +
  geom_hline(yintercept = mean(oasis_raw$nWBV)) +
  ggtitle(label = "Relationship between Estimated Total Intracranial Volume (eTIV) and Age")
```



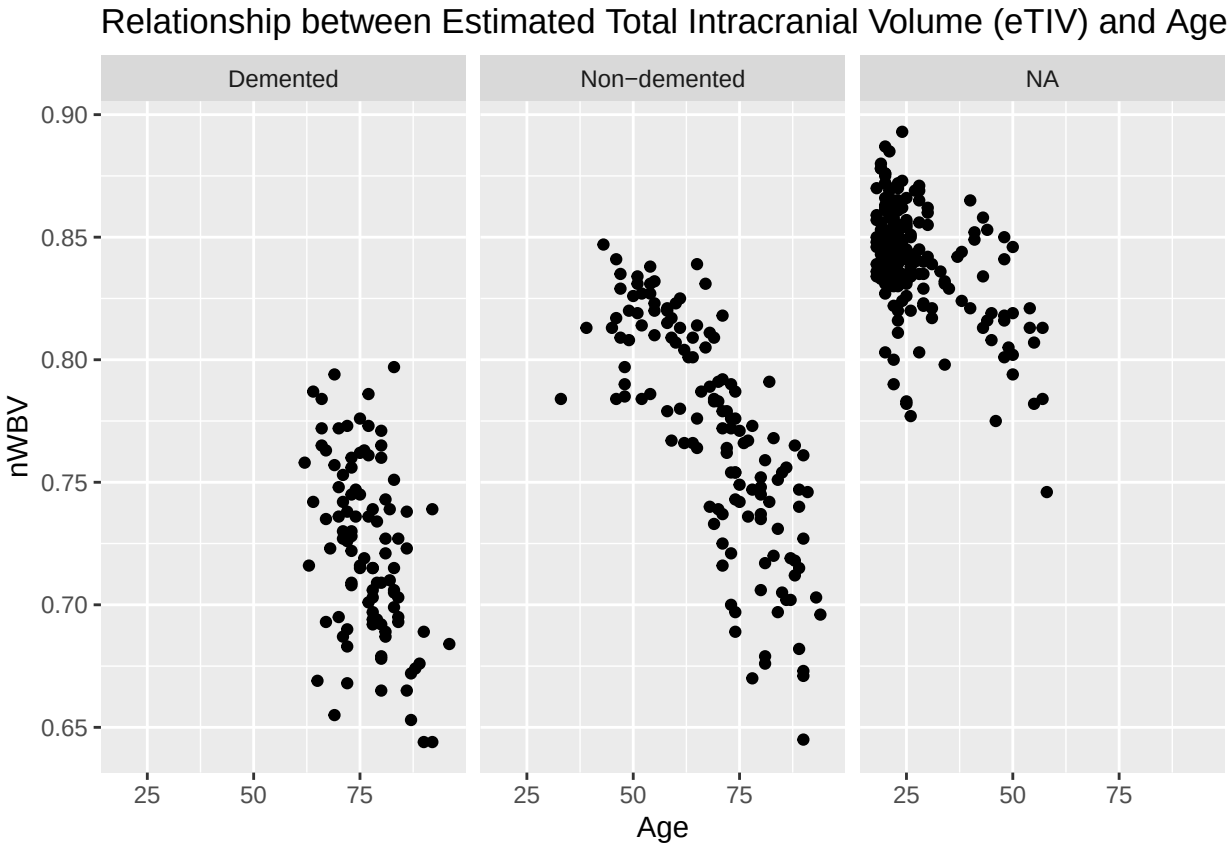
An important "paradox" when evaluating or visualizing linear models is Simpson's Paradox. Simpson's Paradox is the concept that an underlying association between Independent and Dependent variables changes when examining by factors. We see that the overall association is negative, but what if we break it up into factors dependent on their dementia classification?

There are two methods: We can color the data points based on CDR, or we can use `facet_wrap()` to create separate scatterplots, one for each CDR classification:

```
# Method 1: http://www.sthda.com/english/wiki/ggplot2-point-shapes

oasis_raw %>% ggplot() +
  geom_point(mapping = aes(x = Age, y = nWBV, color = CDR2)) +
  ggtitle(label = "Relationship between Estimated Total Intracranial Volume (eTIV) and Age")
```




```
# Method 2: The argument included in `facet_wrap()` is the factor variable you
# want to break up the data by. We include the factor variable in
# `facet_wrap()` with quotations around the variable.

oasis_raw %>% ggplot() +
  geom_point(mapping = aes(x = Age, y = nWBV)) +
  facet_wrap("CDR2") +
  ggtitle(label = "Relationship between Estimated Total Intracranial Volume (eTIV) and Age")
```



Consider the research question "Does the mean estimated total intracranial volume differ significantly by dementia classification?"

What type of test would we run? (While it may look like we have 3 different classifications for CDR, NA is not a factor. So in this case, we only have two groups: Non-demented and Demented):

1) Test the assumptions:

```
# Normality:
dem <- oasis_raw %>% filter(CDR2 == "Demented")
nondem <- oasis_raw %>% filter(CDR2 == "Non-demented")

shapiro.test(dem$eTIV)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dem$eTIV
## W = 0.97013, p-value = 0.02263
```

```
shapiro.test(nondem$eTIV)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  nondem$eTIV
## W = 0.9684, p-value = 0.003143
```

```
# Homogeneity of Variance:
leveneTest(eTIV ~ CDR2, data = oasis_raw)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   1  0.3154 0.5749
##       233
```

In this case, we fail the the normality assumption, leaving the actual t-test moot. But for instruction's sake, run the t-test and draw a conclusion!

```
t.test(eTIV ~ CDR2, data = oasis_raw)
```

```
##
##  Welch Two Sample t-test
##
## data:  eTIV by CDR2
## t = 1.9785, df = 200.95, p-value = 0.04923
## alternative hypothesis: true difference in means between group Demented and group Non-demented is no
## 95 percent confidence interval:
##   0.1427879 84.2905454
## sample estimates:
##     mean in group Demented mean in group Non-demented
##                   1483.750                   1441.533
```

At the 5% significance level, we reject the null hypothesis. The data suggest that the mean eTIV is significantly different by CDR classification, either Demented or Non-demented.

For formality's sake, we should briefly discuss the non-parametric alternative. The unpaired two-samples Wilcoxon test (also known as Wilcoxon rank sum test or Mann-Whitney test) is a non-parametric alternative to the unpaired two-samples t-test, which can be used to compare two independent groups of samples. It's used when your data are not normally distributed. The syntax is `wilcox.test(x = NumericalVar1, y = NumericalVar2, alternative = "two.sided")`

** We have the argument `alternative = "two-sided"` because we are testing for significant differences in either direction, meaning that one group median is significantly greater than OR significantly less than the other group median.

The hypotheses of the Wilcoxon rank sum test are:

H0 : The two populations are equal. HA : The two populations are not equal.

Instead of using the mean value, the non-parametric alternative considers the median when performing the test.

```
wilcox.test(dem$eTIV, nondem$eTIV, alternative = "two.sided")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  dem$eTIV and nondem$eTIV
## W = 7752.5, p-value = 0.05182
## alternative hypothesis: true location shift is not equal to 0
```

We see the `t.test` and `wilcox.test` actually provide contradicting output: we can assert (reject the null hypothesis) that the mean eTIV differs significantly between CDR Non-demented and demented groups for the t.test. For the Wilcoxon Rank sum test, we do not assert (fail to reject the null hypothesis) that the populations differ significantly; ie. the populations are statistically equal.

Conclusion:

I hope you enjoyed this workshop! R (either in RStudio or R, itself) is a very powerful tool. As far as many statistical or graphical packages & functions, most follow a similar pattern. Once you become more

acquainted with the system, of course, the easier it becomes. There's a big learning curve for students who, perhaps, have little-to-no experience with programming. Don't feel ashamed if you don't get it the first time – practice, practice, practice! There are many open-access datasets (via Kaggle or within R) that you can retrieve to help. A useful, but perhaps more technical, resource I have found reliable is the book "R for Data Science", which can be found online (for free!) – https://r4ds.had.co.nz/. Of course, paper copies are available, if you prefer and can be found online, either Amazon or other booksellers. R (and consequently RStudio) can be as complex or simple as you wish; don't let the new system overwhelm you. I hope this now acts as a resource if you decide to transition to RStudio for your analyses.