Stat 50 - Lab 2

Data Analysis using an Excel file

Dr. Jorge Basilio

Jan 27, 2020

Getting Started

- Navigate to the Labs folder and find the Lab_2 folder. Inside you will find a Juptyper Notebook which can run and execute R code titled **"Stat50-Lab_2-YourLastName_YourFirstName-W20"**. Re-name the file with the obvious modifications.
- At the beginning of your Jupyter Notebook, double-click on the "Lab2" text. Replace the text "FirstName LastName" with your actual first and last name. Click "run cell" (looks like a play button) or hit `shift+enter`.
- By looking at this document, you are encouraged to copy and paste lines of code and modify them :-)

PART 1

Reading "big" data from a file

Many interesting data sets collected are large and it would be a huge pain to input the data manually like we did in Lab 1. Even "copy and paste" would be a challange if the data set contains hundreds of items. If you download a data set from trusted sources (like a government or academic institution), then you will want to first import the data set so you can use R to study it.

We need a special package that can read "Excel" files (files that end with `.xls` or `.xlsx`) called a "library". To load a library we need to know the name of it. The library we want to load is called "readxl".

```
#install.packages("readxl")
library(readxl)
```

Type that into a new cell in your lab notebook.

Next, we want R to read the data in the file "06-Freshman15.xlsx". To do this, type the following code in a new line:

```
fresh_15_full <- read_excel("06-Freshman15.xlsx")
```

To see what is in the file "06-Freshman15.xlsx", we use the function `str()`

```
str(fresh_15_full) # print out table of data listing first few values
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    67 obs. of  5 variables:
##  $ SEX      : chr  "M" "M" "M" "M" ...
##  $ WT SEPT  : num  72 97 74 93 68 59 64 56 70 58 ...
##  $ WT APRIL : num  59 86 69 88 64 55 60 53 68 56 ...
##  $ BMI SEPT : num  22 19.7 24.1 27 21.5 ...
```

```
##  $ BMI APRIL: num  18.1 17.4 22.4 25.6 20.1 ...
```

```
names(fresh_15_full) # print out names of variables
```

```
## [1] "SEX"       "WT SEPT"   "WT APRIL"  "BMI SEPT"  "BMI APRIL"
```

```
dim(fresh_15_full) # print out dimension
```

```
## [1] 67  5
```

The 67 observations tells us there are 67 pieces of data for each variable. Note that the first value is the variable names so we subtract 1. There are also 5 variables.

We see that the variables are `SEX` in first row, `WT SEPT` in second row, etc.

1. Explain what each of the five variables are. Be sure to write your answer using a markdown cell.

Next, there's another way we can see our data. Using `head()` and `tail()` will show the first 5 and last 5 rows of data.

```
head(fresh_15_full) # displays first few rows of data and every column
```

```
## # A tibble: 6 x 5
##   SEX   `WT SEPT` `WT APRIL` `BMI SEPT` `BMI APRIL`
##   <chr>     <dbl>      <dbl>      <dbl>       <dbl>
## 1 M            72         59       22.0        18.1
## 2 M            97         86       19.7        17.4
## 3 M            74         69       24.1        22.4
## 4 M            93         88       27.0        25.6
## 5 F            68         64       21.5        20.1
## 6 M            59         55       18.7        17.4
```

```
tail(fresh_15_full) # displays last few rows of data and every column
```

```
## # A tibble: 6 x 5
##   SEX   `WT SEPT` `WT APRIL` `BMI SEPT` `BMI APRIL`
##   <chr>     <dbl>      <dbl>      <dbl>       <dbl>
## 1 F            55         60       21.6        23.8
## 2 M            65         71       22.5        24.4
## 3 M            75         82       23.7        25.8
## 4 F            42         49       15.1        17.7
## 5 M            74         82       22.6        25.3
## 6 M            94        105       36.6        40.9
```

Not all that useful, but still nice to see more of the data.

More accurate view of entries

To get a more accurate view, sometimes we want to see all the data in a specific variable. We do this using the code `name_of_tabledata$"variable_name"` where we first put the name of the table data we saved and then we follow it with a dollar sign (`$`) and then the name of the variable we want to see.

```
fresh_15_full$"SEX" # displays all the data in variable "SEX"
```

```
##  [1] "M" "M" "M" "M" "F" "M" "F" "F" "F" "F" "F" "M" "M" "F" "F" "F" "F" "M" "F"
## [20] "M" "F" "F" "M" "F" "M" "M" "M" "F" "M" "F" "F" "M" "M" "F" "M" "M" "F" "F"
## [39] "F" "F" "M" "M" "M" "F" "F" "F" "F" "F" "F" "F" "M" "M" "M" "M" "F" "F" "F"
## [58] "M" "M" "F" "M" "F" "M" "M" "F" "M" "M"
```

Let's say we want to know only one data value. We can add square brackets with the entry number we are interested in. For example:

```
fresh_15_full$"SEX"[10] # displays the 10th entry in the data in variable "SEX"
```

```
## [1] "F"
```

You can check that this is correct by looking at the full data set and counting to the 10th spot.

What if we want a range of data values from 10 to 20? This is easy to do as well:

```
fresh_15_full$"SEX"[10:20] # displays all the entries in the 10th to 20th spots
```

```
##  [1] "F" "F" "M" "M" "F" "F" "F" "F" "M" "F" "M"
```

Again, you can check that this is correct by looking at the full data set above.

How to see data from different variables (columns)

What if we want to see what the **second column** of data is. We can use the command `name_of_tabledata[,number]`. The comma is not a typo! This will make more sense shortly.

```
fresh_15_full[,2] # displays (part of) 2nd column
```

```
## # A tibble: 67 x 1
##    `WT SEPT`
##        <dbl>
##  1        72
##  2        97
##  3        74
##  4        93
##  5        68
##  6        59
##  7        64
##  8        56
##  9        70
## 10        58
## # ... with 57 more rows
```

You can check this is the second column by opening up the excel file to be sure. Notice it only gives us the first few data values.

What about a number before the comma? What does `name_of_tabledata[number,]` do?

```
fresh_15_full[10,] # displays entire 10th row (every column)
```

```
## # A tibble: 1 x 5
##   SEX   `WT SEPT` `WT APRIL` `BMI SEPT` `BMI APRIL`
##   <chr>     <dbl>      <dbl>      <dbl>       <dbl>
## 1 F            58         56       21.9        21.0
```

If we want to see the data from rows 10 to 20 we can use the colon as before:

```
fresh_15_full[10:20,] # displays entire rows 10 to 20 (every column)
```

```
## # A tibble: 11 x 5
##    SEX   `WT SEPT` `WT APRIL` `BMI SEPT` `BMI APRIL`
##    <chr>     <dbl>      <dbl>      <dbl>       <dbl>
##  1 F            58         56       21.9        21.0
##  2 F            50         47       17.6        16.9
```

```
##  3 M            71         69       24.6        23.8
##  4 M            67         66       20.7        20.2
##  5 F            56         55       21.0        20.4
##  6 F            70         68       27.3        26.7
##  7 F            61         60       23.3        22.9
##  8 F            53         52       19.5        19.2
##  9 M            92         92       24.7        24.7
## 10 F            57         58       20.7        20.8
## 11 M            67         67       20.5        20.6
```

Next, what if we want to see rows 1 and 2 and columns 1 and 2 only? We can **remove columns 4 and 5** as follows:

```
fresh_15_full[c(1,2), c(-4,-5)] # displays row 1 and 2, removes columns 4 and 5
```

```
## # A tibble: 2 x 3
##   SEX   `WT SEPT` `WT APRIL`
##   <chr>     <dbl>      <dbl>
## 1 M            72         59
## 2 M            97         86
```

Notice that we use the code `c(1,2)` **before the comma** to tell R to grab the data from **rows** 1 and 2.

The code `c(-4,-5)` **after the comma** to tell R to REMOVE the data from **columns** 4 and 5–it REMOVES it because of the minus sign.

2. Find:
   [a] Display all the entries in "WT SEPT".
   [b] Display the 55th entry in "WT SEPT".
   [b] Display the entire rows 45 to 55.
   [b] Display rows 45 to 55 for only columns 4 and 5

Summary Statistics

Once we have the excel file saved to R using the name `fresh_15_full` and have learned how to read off specific data entries, it is very easy to have R compute the **5 number summary** using the `summary()` command.

```
summary(fresh_15_full) # computes 5# Summary statistics for each column
```

```
##      SEX                WT SEPT         WT APRIL         BMI SEPT
##  Length:67          Min.   :42.00   Min.   : 47.00   Min.   :15.08
##  Class :character   1st Qu.:56.50   1st Qu.: 58.00   1st Qu.:19.96
##  Mode  :character   Median :64.00   Median : 66.00   Median :21.73
##                     Mean   :65.06   Mean   : 66.24   Mean   :22.03
##                     3rd Qu.:70.50   3rd Qu.: 70.00   3rd Qu.:23.16
##                     Max.   :97.00   Max.   :105.00   Max.   :36.57
##    BMI APRIL
##  Min.   :16.89
##  1st Qu.:20.23
##  Median :22.31
##  Mean   :22.48
##  3rd Qu.:23.86
##  Max.   :40.86
```

3. Find:
   [a] Give the **5 number summaries** for the data in variable "WT SEPT" and "WT APRIL"

[b] Which month has the larger **median**?
[c] Compute the **range** for both months. Which is bigger?
[d] What does part (c) imply about the **spread** of the weights of freshmen? Use the range rule of thumb to make your argument–you do not need to compute the standard deviation.

Descriptive Statistics

Now, that we know some tricks for grabbing the different parts of the data we can learn how R computes descriptive statistics.

```
table(fresh_15_full$SEX) # summarizes freq of categorical data
```

```
##
##  F  M
## 35 32
```

4. Are there more male or females in the data set?

We already know how to get the 5 number summary of any of the variables (columns). What if we want to study other descriptive statistics, like the mean, median, standard deviation, etc?

Now, that we know how to get the data from a specific variable/column, this is easy in R:

```
mean(fresh_15_full$"WT SEPT") # mean of "WT SEPT" variable
```

```
## [1] 65.0597
```

```
median(fresh_15_full$"WT SEPT") # median of "WT SEPT" variable
```

```
## [1] 64
```

```
sd(fresh_15_full$"WT SEPT") # standard deviation of "WT SEPT" variable
```

```
## [1] 11.28539
```

```
var(fresh_15_full$"WT SEPT") # variance of "WT SEPT" variable
```

```
## [1] 127.36
```

5. Compare the exact standard deviation given above with the estimate found in problem 3(d) using the range rule of four.

Part 2

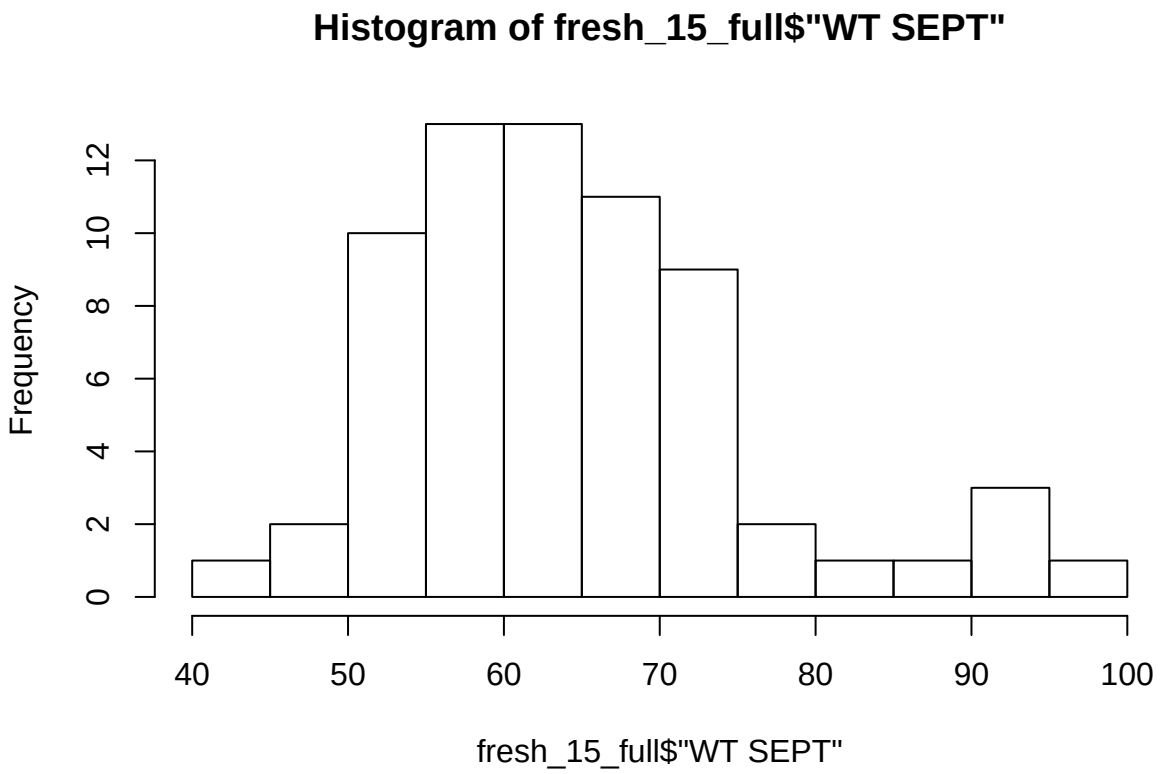
Data Visualization

Now that we've studied the basics of descriptive statistics, let's see how we can visualize the data better.

Histograms

We start with **histograms**. R will automatically create class sizes for us (yay!). You can, of course, customize the class sizes but this is beyond the score of this lab and is a bit advanced for us.

```
hist(fresh_15_full$"WT SEPT") # 5 Number Summary of "WT SEPT"
```

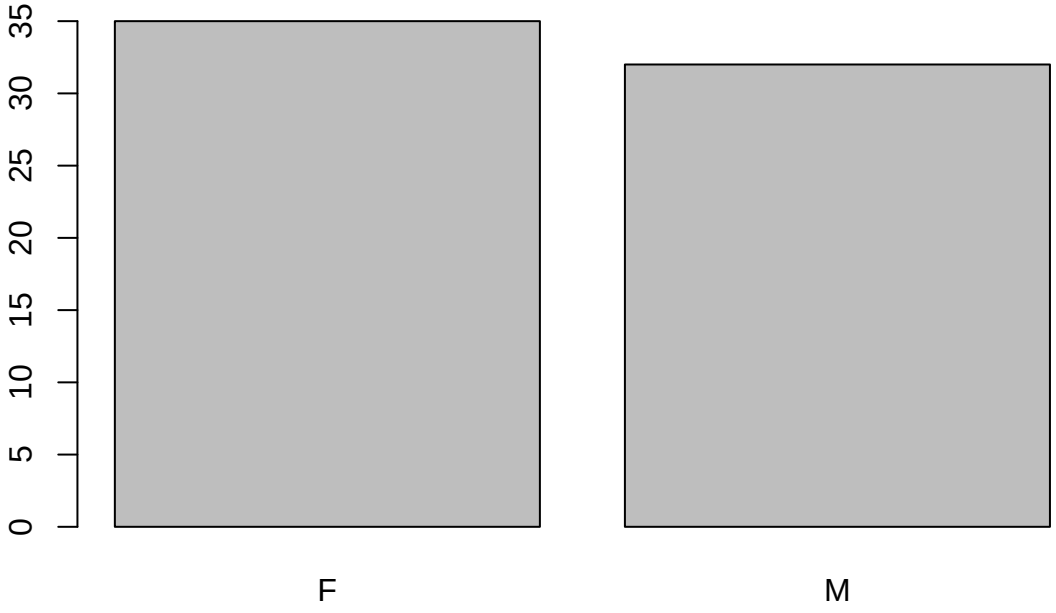


6. Create a histogram for each of the other variables.

Bar Plots

Bar plots are for displaying categorical data (recall: bars don't touch). If we wan to display the frequency of male vs females, we can use:

```
# box plots
barplot(table(fresh_15_full$SEX))
```

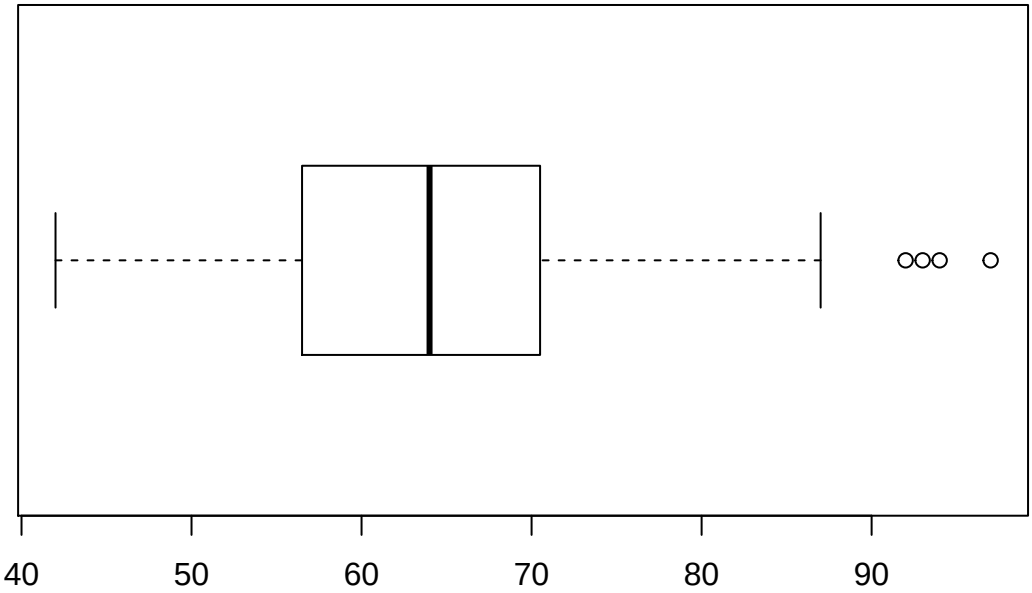


Notice this is a little tricky. We need to actually use `table(data$variable)` before the `barplot()` command is invoked.

Box Plots

Box plots display the five number summary. Recall that in Lab 1 we learned that R makes box plots verical automatically. Since we prefer to have them displayed horizontally we must add extra instructions in the form of `horizontal=TRUE` after a comma:

```
# box plot
boxplot(fresh_15_full$"WT SEPT", horizontal=TRUE)
```

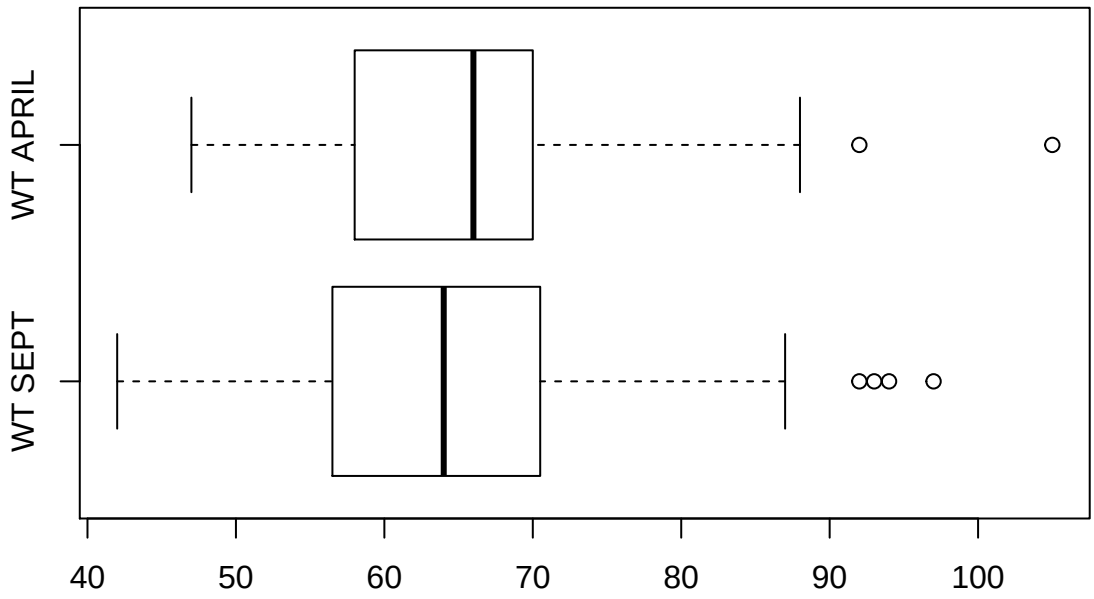


Notice that R's box plots are a little different than the ones we discussed in class. This is because the little circles (dots) on the right are considered **outliers**. This is discussed in the book and is given by a formula using the **IQR (inter-quartile range)**. If you are interested, you can consult our textbook.

Finally, we reach a nice conclusion to our analyses. Earlier we used the summary data to compare the weights of freshment in September and, again, in April of their first year of college.

We would like to plot both box plots at the same time so we can visually compare the data sets. We do it like this:

```
# box plot
# compare before and after freshman year weights
boxplot(fresh_15_full[,c(2,3)], horizontal=TRUE)
```



Notice that we are comparing columns 2 and 3, which is why we used the code `c(2,3)` **after** the comma.

7. By comparing the box plots of the weights of freshman in September and in April, do you believe the "freshman 15" myth is true? Please include the two box plots in your own notebook and use them to explain your answer.

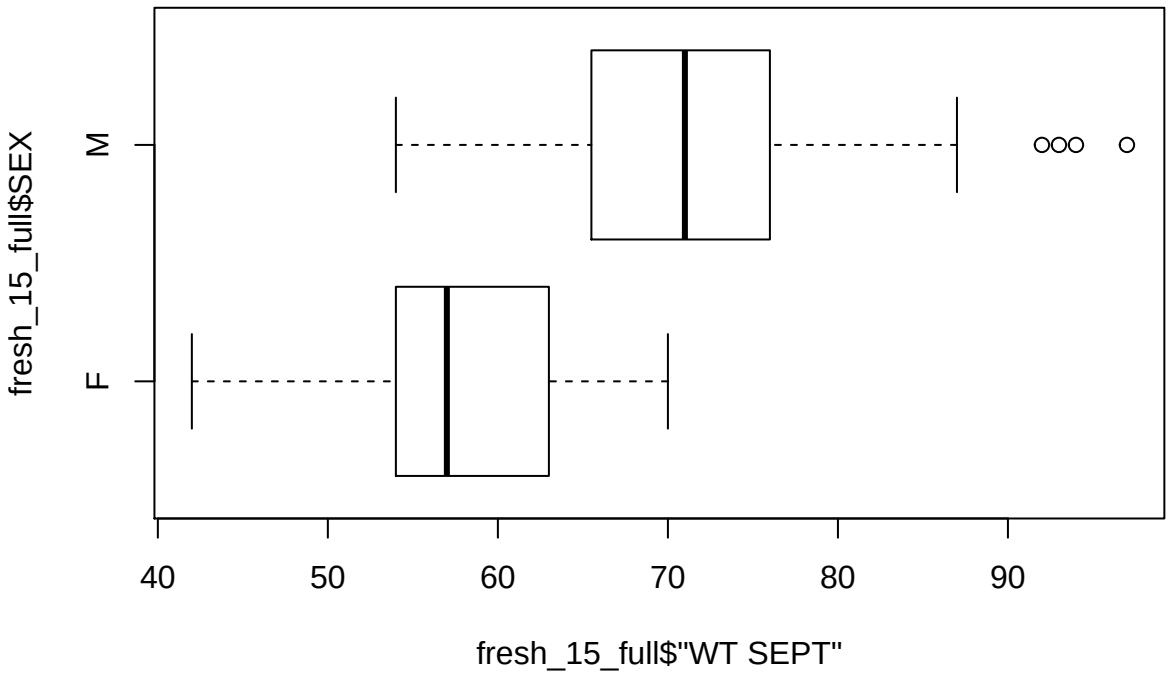
Part 3

Bonus comparison

What if we want to dig a little deeper and compare the "freshman 15" myth between females vs males?

Here is how we can tell R to create a box plot that compares the data for "WT SEPT" between "M" and "F":

```
# box plot
# compare two box plots between M vs F
# first piece before the tilde (~) is the box plots from the "WT SEPT" column
# the second after the tilde (~) is the grouping variable (SEX, M/F)
boxplot(fresh_15_full$"WT SEPT" ~ fresh_15_full$SEX, horizontal=TRUE)
```



This syntax is typically in the format of `y ~ x` where `y` is the response variable (vertical variable) and `x` is the grouping variable (horizontal variable). We haven't yet introduced this terminology, but it will be used in Chapter 10 we when we start comparing two pieces of data.

8. Create four box plots comparing the weights in September and in April between males and females. The first two will be the same code as above. You will need to modify the code to get the other two box plots for April.
   [a] By looking at only the male data in September and in April, is the "freshman 15" myth true for males?
   [b] By looking at only the female data in September and in April, is the "freshman 15" myth true for females?

Some extra "styling" in the display

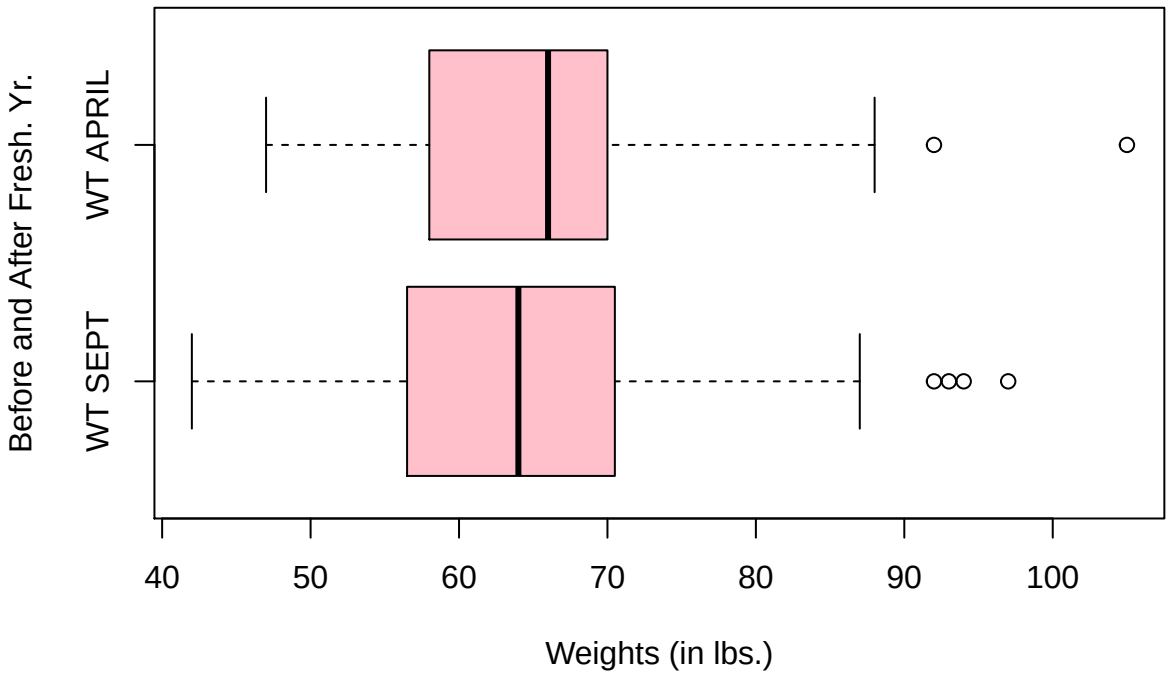
R allows for almost infinite customization. It's not necessarily easy to learn, but here are a few basics to make the box plots look a bit better by adding some "style".

Add color in the box plot using `col="pink"`.

Add a vertical label using `ylab="Before and After Fresh. Yr."`

Add a horizontal label using `xlab="Weights (in lbs.)"`

```
# box plot
# compare before and after freshman year weights
boxplot(fresh_15_full[,c(2,3)], col="pink", ylab="Before and After Fresh. Yr.", xlab="Weights (in lbs.)"
```



You've done it! You've now finished Lab_2! ;-)