Math 136 - Lab 2

Data Analysis using CSV files

Dr. Jorge Basilio

March 18, 2020

Learning Objectives:

Learn how to read "big data" from a CSV file in R

Learn how to see different data in R

Learn how to see summary statistics in R

Learn how to compute descriptive statistics in R

Learn Data Visualization in R

Learn some "styling" and colors in R

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 **"Math136-Lab_2-YourLastName_YourFirstName-S20"**. 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

Section 1 - Reading "big data" from a CSV 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 challenge if the data set contains thousands of entries with multiple variables. 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.

First, we want R to read the data in the file "06-Freshman15.csv". Notice that this file is already in the Lab 2 folder on CoCalc.

To have R read this file, and save it to the variable name `fresh_15_full`, type the following code in a new line:

```
fresh_15_full <- read.csv("06-Freshman15.csv")
```

A CSV file is short for "comma separated file" and is basically a file that stores data that is separated by commas.

To see what is in the file "06-Freshman15.csv", we use the function `str()` (tells us the structure of the file):

```
str(fresh_15_full) # see the 'structure' and see the first few values
```

```
## 'data.frame':    67 obs. of  5 variables:
##  $ SEX      : Factor w/ 2 levels "F","M": 2 2 2 2 1 2 1 1 1 1 ...
##  $ WT.SEPT  : int  72 97 74 93 68 59 64 56 70 58 ...
##  $ WT.APRIL : int  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 ...
```

R treats this as a **data frame**.

The 67 observations tells us there are 67 pieces of data for each variable. There are also 5 variables.

We can also see the variable names: "SEX", "WT.SEPT", etc.

If all you want to see is the variable names, you can use the `names()` function:

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

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

This is nice because we sometimes don't need to see the individual values of the data but need to remember the variable names.

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
```

```
##   SEX WT.SEPT WT.APRIL BMI.SEPT BMI.APRIL
## 1   M      72       59    22.02     18.14
## 2   M      97       86    19.70     17.44
## 3   M      74       69    24.09     22.43
## 4   M      93       88    26.97     25.57
## 5   F      68       64    21.51     20.10
## 6   M      59       55    18.69     17.40
```

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

```
##    SEX WT.SEPT WT.APRIL BMI.SEPT BMI.APRIL
## 62   F      55       60    21.64     23.81
## 63   M      65       71    22.51     24.45
## 64   M      75       82    23.69     25.80
## 65   F      42       49    15.08     17.74
## 66   M      74       82    22.64     25.33
## 67   M      94      105    36.57     40.86
```

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

To see ALL of the data, we simply type the name `fresh_15_full`:

```
fresh_15_full
```

```
##    SEX WT.SEPT WT.APRIL BMI.SEPT BMI.APRIL
```

```
## 1    M      72       59    22.02     18.14
## 2    M      97       86    19.70     17.44
## 3    M      74       69    24.09     22.43
## 4    M      93       88    26.97     25.57
## 5    F      68       64    21.51     20.10
## 6    M      59       55    18.69     17.40
## 7    F      64       60    24.24     22.88
## 8    F      56       53    21.23     20.23
## 9    F      70       68    30.26     29.24
## 10   F      58       56    21.88     21.02
## 11   F      50       47    17.63     16.89
## 12   M      71       69    24.57     23.85
## 13   M      67       66    20.68     20.15
## 14   F      56       55    20.97     20.36
## 15   F      70       68    27.30     26.73
## 16   F      61       60    23.30     22.88
## 17   F      53       52    19.48     19.24
## 18   M      92       92    24.74     24.69
## 19   F      57       58    20.69     20.79
## 20   M      67       67    20.49     20.60
## 21   F      58       58    21.09     21.24
## 22   F      49       50    18.37     18.53
## 23   M      68       68    22.40     22.61
## 24   F      69       69    28.17     28.43
## 25   M      87       88    23.60     23.81
## 26   M      81       82    26.52     26.78
## 27   M      60       61    18.89     19.27
## 28   F      52       53    19.31     19.75
## 29   M      70       71    20.96     21.32
## 30   F      63       64    21.78     22.22
## 31   F      56       57    19.78     20.23
## 32   M      68       69    22.40     22.82
## 33   M      68       69    22.76     23.19
## 34   F      54       56    20.15     20.69
## 35   M      80       82    22.14     22.57
## 36   M      64       66    20.27     20.76
## 37   F      57       59    22.15     22.93
## 38   F      63       65    23.87     24.67
## 39   F      54       56    18.61     19.34
## 40   F      56       58    21.73     22.58
## 41   M      54       56    18.93     19.72
## 42   M      73       75    25.88     26.72
## 43   M      77       79    28.59     29.53
## 44   F      63       66    21.89     22.79
## 45   F      51       54    18.31     19.28
## 46   F      59       62    19.64     20.63
## 47   F      65       68    23.02     24.10
## 48   F      53       56    20.63     21.91
## 49   F      62       65    22.61     23.81
## 50   F      55       58    22.03     23.42
## 51   M      74       77    20.31     21.34
## 52   M      74       78    20.31     21.36
## 53   M      64       68    19.59     20.77
## 54   M      64       68    21.05     22.31
```

```
## 55   F      57       61    23.47     25.11
## 56   F      64       68    22.84     24.29
## 57   F      60       64    19.50     20.90
## 58   M      64       68    18.51     19.83
## 59   M      66       71    21.40     22.97
## 60   F      52       57    17.72     19.42
## 61   M      71       77    22.26     23.87
## 62   F      55       60    21.64     23.81
## 63   M      65       71    22.51     24.45
## 64   M      75       82    23.69     25.80
## 65   F      42       49    15.08     17.74
## 66   M      74       82    22.64     25.33
## 67   M      94      105    36.57     40.86
```

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 in quotes.

```
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 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 M M F M F M M F M M
## Levels: F M
```

Note: sometimes you don't need to use quotes in the variable name. But to be safe just always use them.

```
# testing without quotes
fresh_15_full$SEX
```

```
##  [1] M M M M F M F F F F F M M F F F F M F 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 M M F M F M M F M M
## Levels: F M
```

This is why knowing the `names()` function is really important. It will tells us what the different column variables are so we can call on them to see or manipulate the data.

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
## Levels: F M
```

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
## Levels: 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 2nd column
```

```
##  [1] 72 97 74 93 68 59 64 56 70 58 50 71 67 56 70 61 53 92 57 67 58 49 68 69 87
## [26] 81 60 52 70 63 56 68 68 54 80 64 57 63 54 56 54 73 77 63 51 59 65 53 62 55
## [51] 74 74 64 64 57 64 60 64 66 52 71 55 65 75 42 74 94
```

You can check this is the second column by opening up the csv file to be sure or scrolling up to see the full data set.

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

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

```
##    SEX WT.SEPT WT.APRIL BMI.SEPT BMI.APRIL
## 10   F      58       56    21.88     21.02
```

Notice that it shows one row of the data, row 10 in this case. But it shows every variable value for row 10. Thus, the 10th entry in "SEX" is `F`; the 10th entry in "WT.SEPT" is `58`, the 10th entry in "WT.SEPT" is `56`, etc.

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)
```

```
##    SEX WT.SEPT WT.APRIL BMI.SEPT BMI.APRIL
## 10   F      58       56    21.88     21.02
## 11   F      50       47    17.63     16.89
## 12   M      71       69    24.57     23.85
## 13   M      67       66    20.68     20.15
## 14   F      56       55    20.97     20.36
## 15   F      70       68    27.30     26.73
## 16   F      61       60    23.30     22.88
## 17   F      53       52    19.48     19.24
## 18   M      92       92    24.74     24.69
## 19   F      57       58    20.69     20.79
## 20   M      67       67    20.49     20.60
```

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
```

```
##   SEX WT.SEPT WT.APRIL
## 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.

Recall that we are using the **vector** notation where the syntax `c()` is how we store vectors, or lists, into `R`.

Note: There's usually more than one way to do something in R. So be creative!

2. Do the following:
   [a] Display all the entries in "WT.SEPT".

[b] Display the 55th entry in "WT.SEPT".
[c] Display the entire rows 45 to 55.
[d] Display rows 45 to 55 for only columns 4 and 5.

Section 2 - Summary Statistics

Once we have the csv 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()` function.

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

```
##  SEX       WT.SEPT         WT.APRIL         BMI.SEPT       BMI.APRIL
##  F:35   Min.   :42.00   Min.   : 47.00   Min.   :15.08   Min.   :16.89
##  M:32   1st Qu.:56.50   1st Qu.: 58.00   1st Qu.:19.96   1st Qu.:20.23
##         Median :64.00   Median : 66.00   Median :21.73   Median :22.31
##         Mean   :65.06   Mean   : 66.24   Mean   :22.03   Mean   :22.48
##         3rd Qu.:70.50   3rd Qu.: 70.00   3rd Qu.:23.16   3rd Qu.:23.86
##         Max.   :97.00   Max.   :105.00   Max.   :36.57   Max.   :40.86
```

Notice we get a five-number summary for each variable. For the first variable, `SEX`, all we get is total counts in the summary (because it is qualitative data).

3. Find:
   [a] Have R display the **5 number summaries** for only the data in variable "WT.SEPT" and "WT.APRIL".
   [b] Using the summaries from part (a), which month has the larger **median**?
   [c] Compute the **range** for the weight for both months using R. 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. The range rule of thumb is a rough estimate for the standard deviation. It says that $s \approx range/4$.

Section 3 - 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
```

Notice it is pretty similar to what we did in Lab 1 but now we use `fresh_15_full$SEX`. We use the `dataframe_name$"variable_name"` format.

4. Are there more male or females in the data set? (Reminder: use a `markdown` cell in your answer.)

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
```

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

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

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

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

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

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

Compare the median with the values in the five-number summaries above–they should agree.

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

Part 2

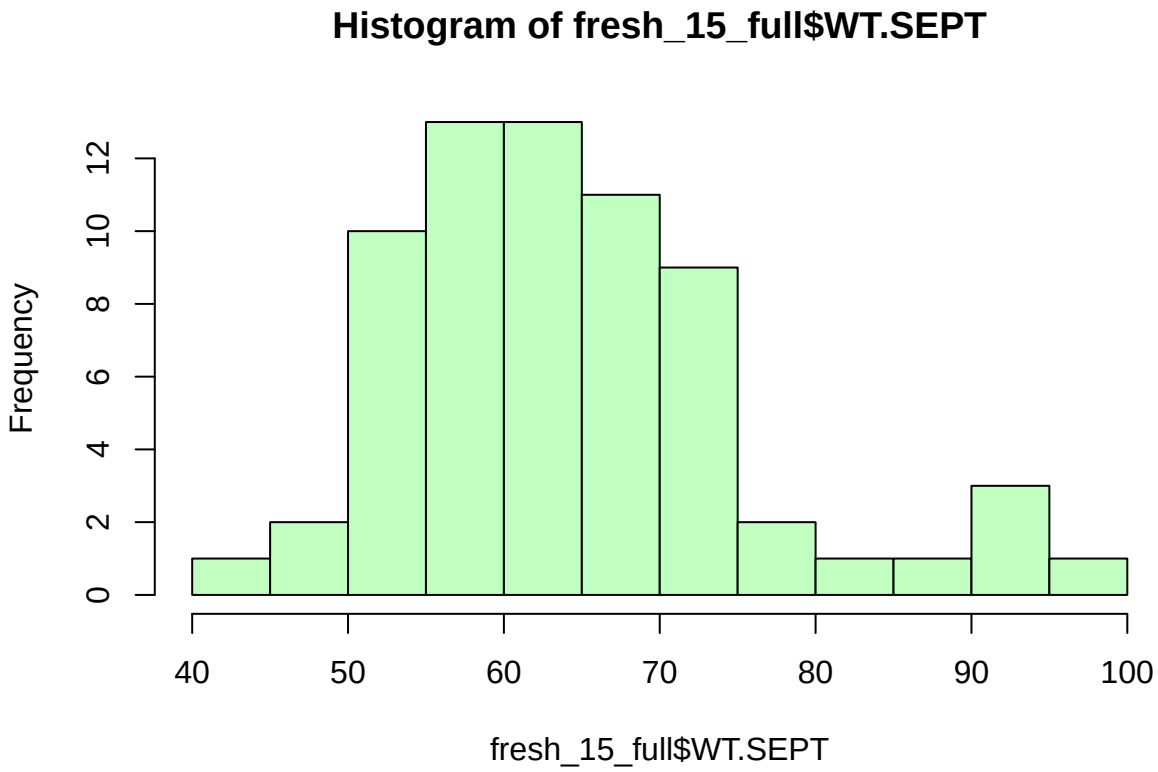
Section 4 - Data Visualization

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 scope of this lab and is a bit too advanced for us. You can also cutsomize the color of the bars using the `col="color_code"`.

```r
hist(fresh_15_full$"WT.SEPT", col="darkseagreen1") # Histogram of "WT SEPT"
```



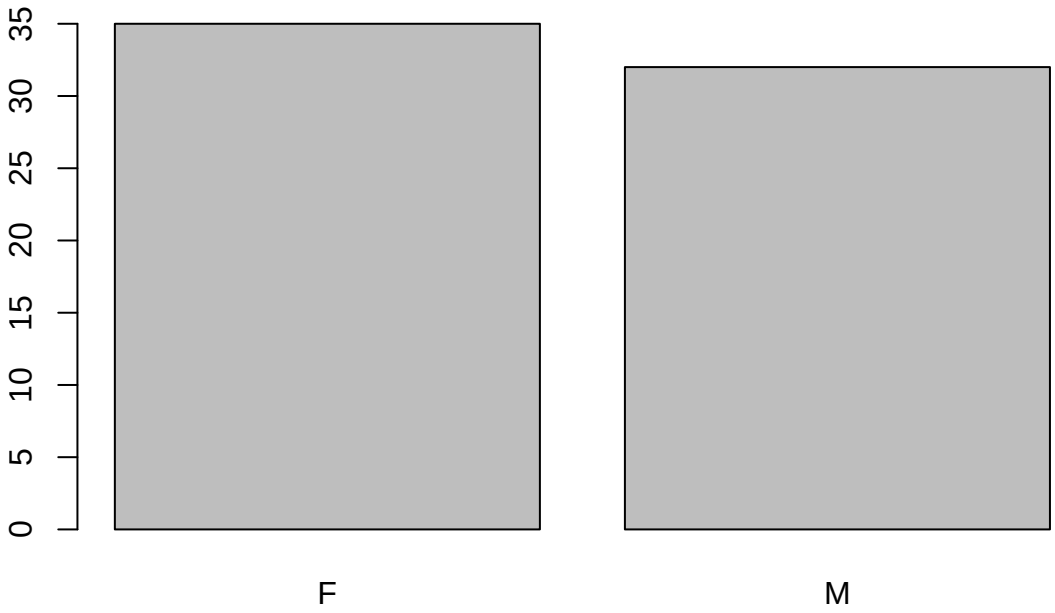
If you want to choose different colors: check this out.

6. Create a histogram for each of the other three quantitative 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))
```

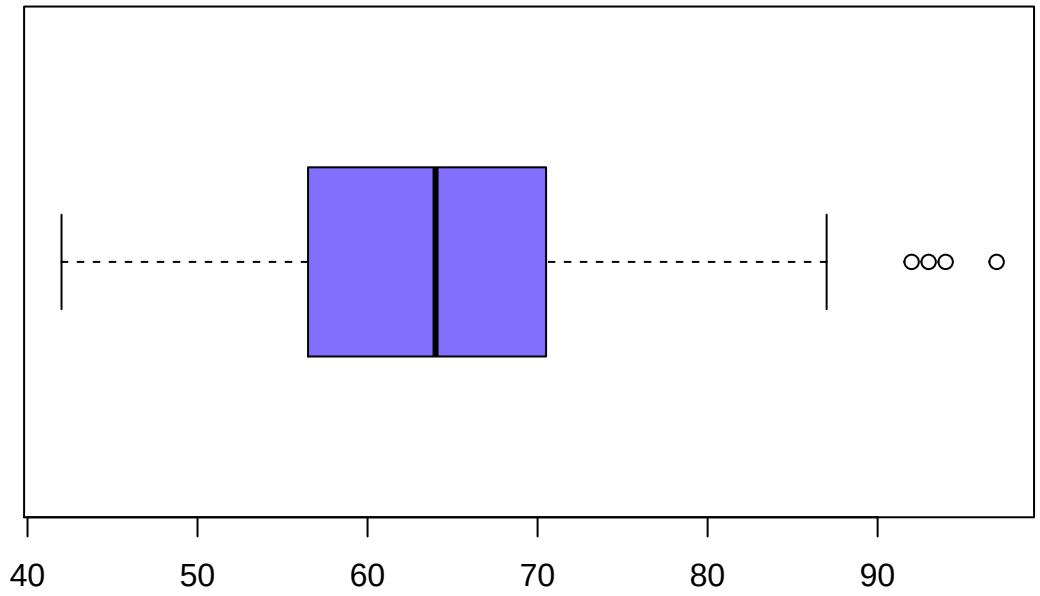


Notice this is a little tricky. We need to actually use `table(data$variable)` before the `barplot()` function is invoked.

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, col="slateblue1")
```



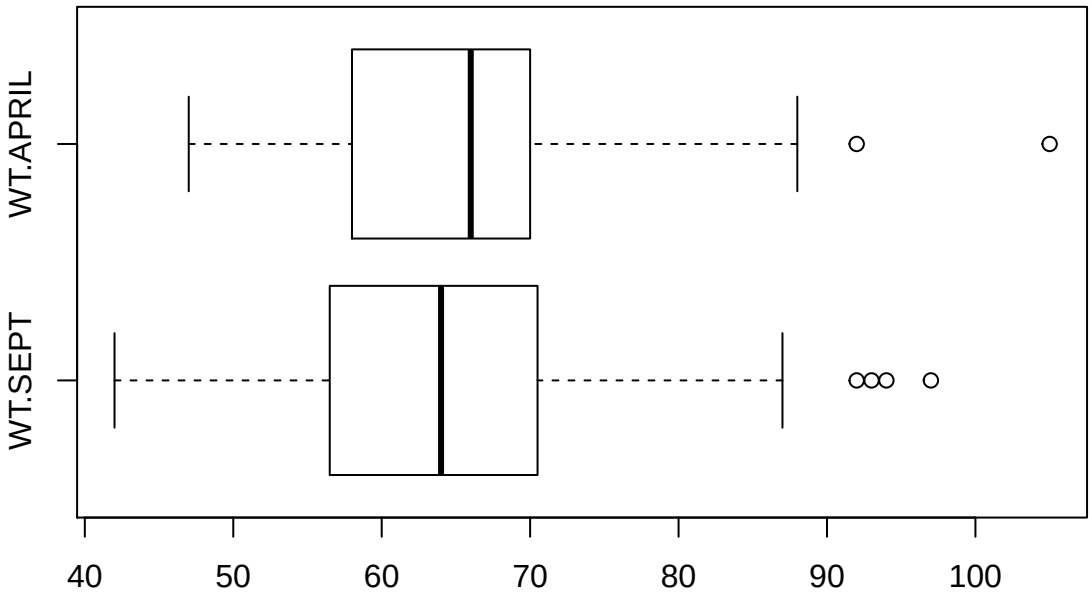
Notice that R's box plots automatically include **outliers**. This is because the little circles (dots) on the right are considered *outliers*. This is discussed in the book using the **IQR (inter-quartile range)**.

Recall outliers are values of the data above $Q_3 + 1.5 \cdot IQR$ and below $Q_1 - 1.5 \cdot IQR$.

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:

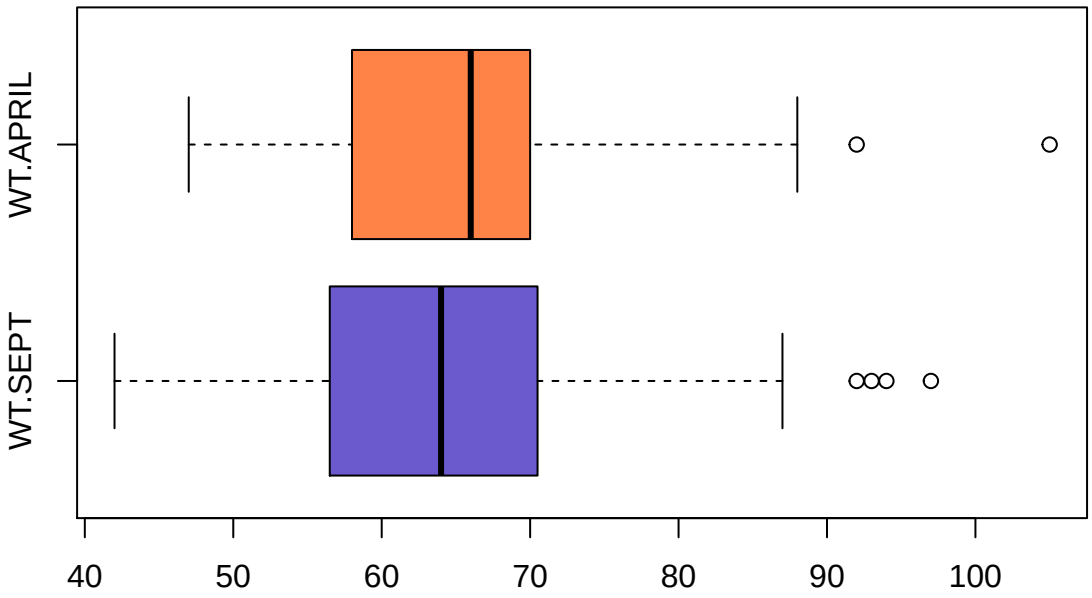
```
# box plot
# compare before and after freshman year weights
boxplot(fresh_15_full[,c(2,3)], horizontal=TRUE)
```



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

Bonus: if you want color for each box plot, you'll need to use `c("color1","color2")`:

```
# box plot
# compare before and after freshman year weights
boxplot(fresh_15_full[,c(2,3)], horizontal=TRUE, col=c("slateblue","sienna1"))
```



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 refer them (and the statistics they represent) to justify your answer.

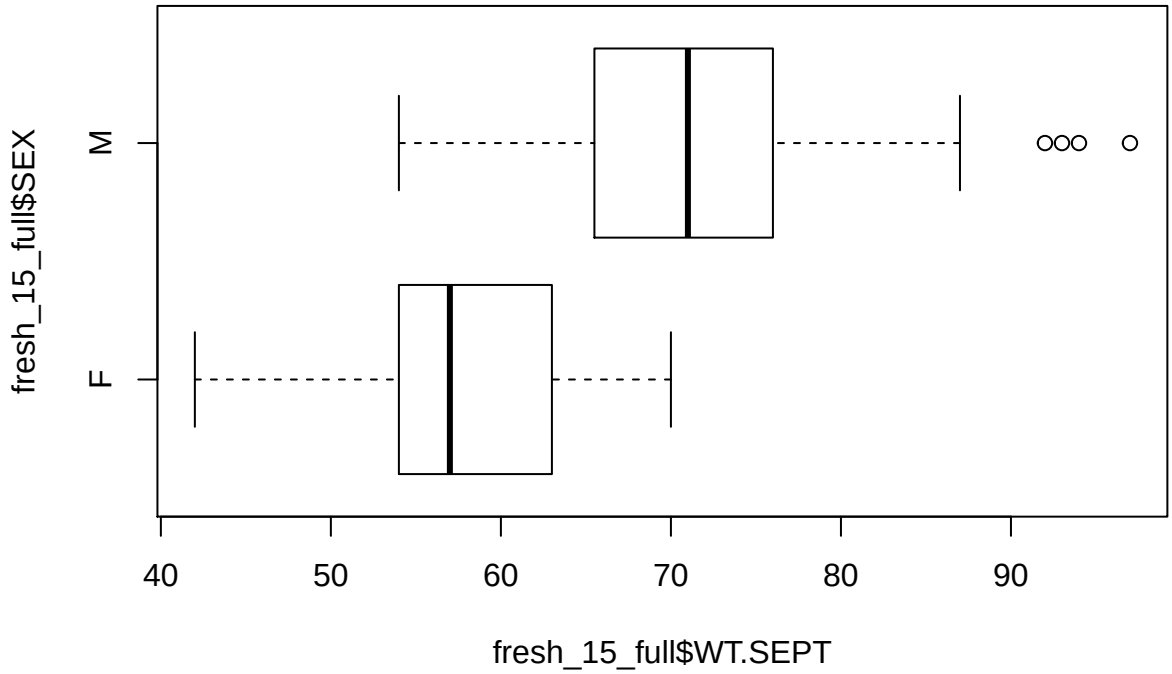
Part 3

Another 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":

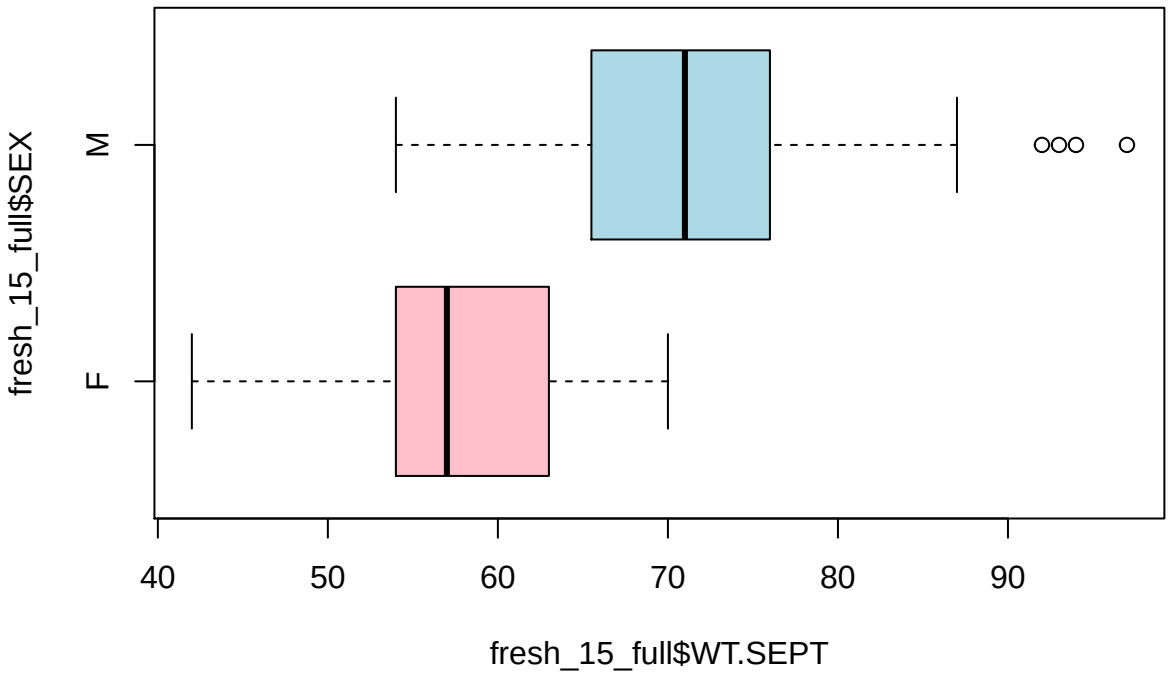
```
# box plot
# compare two box plots between M vs F
# first piece before the tilde (~) is the box plots from the "WT SEPT" column
# the second after the tilde (~) is the grouping variable (SEX, M/F)
boxplot(fresh_15_full$"WT.SEPT" ~ fresh_15_full$SEX, horizontal=TRUE)
```



This syntax is typically in the format of `y ~ x` where `y` is the response variable (vertical variable) and `x` is the explanatory (or grouping) variable (horizontal variable).

Style it with color:

```
# 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, col=c("pink", "lightblue"))
```



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".

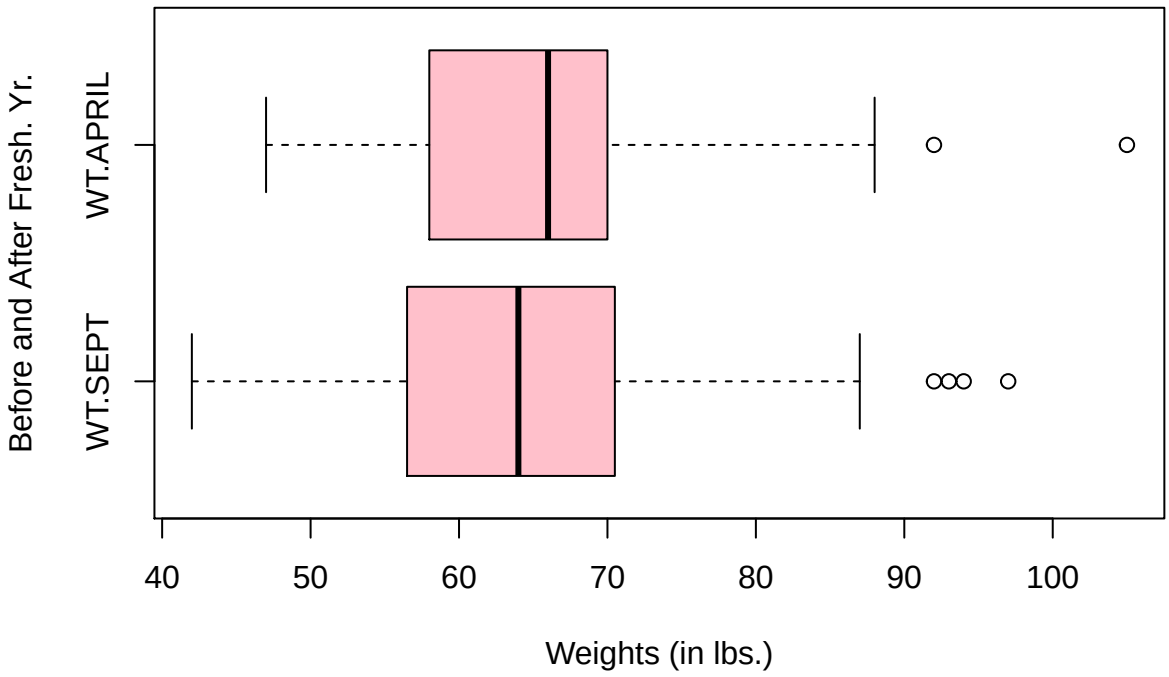
Colors We've already talked about adding color in the box plot using `col="pink"` for example.

The following website has lots of color codes using in R: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

Labels 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! ;-)