Intro to R for Assignment 1

- If you are using RStudio, you should see 4 windows: a project window in the upper left, console in the lower left, plots in the lower right, and environment in the upper right. You can save your code in the upper left. Commands do not run automatically; to execute, you can highlight and press Ctrl+Enter, use the run botton, or copy and paste into the lower left and hit enter to execute.
- If you prefer to use R, there will be a single window open, the **R Console**. You can enter commands in this window, and the output will appear here as well. Any commands entered into this window will run automatically. It is probably wise to open a **Script Window**. In the PC version of R, you do this by selecting **File**, then **New Script**. You would highlight the commands you'd like to run, hit **Ctrl-R** to run them.
- To look at the help file for any command, simply enter a question mark before the command name. For example, if I wanted to know more about the command **hist**, I would enter **?hist**.
- If you are not sure of the name of a particular command, R allows for fuzzy matching. You would enter two question marks followed by the search term. For example, if I wasn't sure of the command to generate a histogram, I might enter **??histogram** to see a list of possible commands.

Entering Data

- For now, we will enter numerical data as vectors.
- For a small number of observations, it is easy enough to enter the data directly. Suppose we have a set of five steel bolts, and we measure their weights in grams:

  12.3, 12.5, 12.7, 11.5, 15.3

- It is good practice to use a descriptive name for your vector, rather than a single letter. Names must not contain spaces, but may contain periods. I might choose **bolt.weight** as the name for my vector.
- To create my vector, I use the following command:

```
bolt.weight <- c(12.3, 12.5, 12.7, 12.1, 12.6)
```

- We can read the symbols <- as meaning "is defined as". The **c** before our data is the command R uses to create a vector. It is also fine to use = instead of <- when defining a vector of data.

- For a large number of observations, we can scan in the data.

  Suppose I had the following list of odd numbers that I found in a document:

  $$1 \;\; 3 \;\; 5 \;\; 7 \;\; 9 \;\; 11 \;\; 13 \;\; 15 \;\; 17 \;\; 19$$

  Suppose I want to create a vector called **odd.numbers** containing these numbers. I begin by entering into the console window:

  ```
  odd.numbers <- scan()
  ```

- I can then copy and paste the numbers from the document into R. I can do this with data separated either by spaces or by line breaks.
- Hitting enter on a blank line stops the process of reading in numbers. At that point, R will tell you how many numbers it has read in.
- The **scan()** command is especially useful if you are copying and pasting a column or row of data from a spreadsheet.
- If we wish to see our vectors now in the computer's memory, we just need to enter the vector name. When I enter **bolt.weight**, I get the following output:

  ```
  [1] 12.3 12.5 12.7 11.5 15.3
  ```

- The **[1]** at the beginning of the line is a counter (i.e. this line begins with observation number one. If the data had several lines worth of observations, there would be a counter at the beginning of each line to tell the reader which observation is at the start of that line.

Numerical Summaries

- If we wanted to find the mean and median of our bolt data, we would only need to enter the command **mean(bolt.weight)** and **median(bolt.weight)** .
- The commands **var(bolt.weight)** and **sd(bolt.weight)** would give us the sample variance and sample standard deviation (respectively) for our observations.

Visual Summaries

- To create a histogram of the bolt data, I could enter the command **hist(bolt.weight)**. You can use various options to make your histogram even better looking. The options **main** and **xlab** allow you to enter a customized title for your histogram and x-axis.
- With the command below, the title of the histogram will read "Bolt Study", and below the x-axis it will read "weight of bolts (in g)".

```
hist(bolt.weight, main = "Bolt Study", xlab = "weight of bolts (in g)")
```

- The **boxplot** command is used to create boxplots, and the **plot** command is used to create scatterplots. Please use ?plot to check more about labels.
- If I enter **boxplot(bolt.data)**, I will see a boxplot of the bolt data.
- If I wish to compare two (or more) sets of data with boxplots, I can create side-by-side boxplots. For example, if I enter **boxplot(bolt.data,odd.numbers)**, R will put the boxplots for both data sets on the same axes.
- If I wanted to label the boxplots on the x-axis as **bolts** and **odd numbers** I could enter the command:

```
boxplot(bolt.weight, odd.numbers, names = c("bolts","odd numbers"))
```

- As with the **histogram** function, you can add a title to a boxplot using the main option in the boxplot function:

```
boxplot(bolt.weight, odd.numbers, names = c("bolts","odd numbers"),
  main = "Boxplots of bolt weights and odd numbers")
```

- The default is to have a vertical boxplot. If you wish to have a horizontal boxplot instead, you can use the HORIZONTAL option in the boxplot function:

```
boxplot(bolt.weight, odd.numbers, names = c("bolts","odd numbers"),
  main = "Boxplots of bolt weights and odd numbers",horizontal=TRUE)
```

- If I wish to know what cutoffs R is using for the boxplots, I can use the **summary** command. When I enter the command **summary(bolt.weight)** I get the following output:

```
Min.  1st Qu. Median  Mean  3rd Qu.  Max.
12.10   12.30  12.50  12.44  12.60   12.70
```

- The bottom of the box is at the first quartile (12.30) and the top of the box is at the third quartile (12.60). The interquartile range is thus 12.60 - 12.30 = 0.30. You could also find this using IQR(bolt.weight). The line in the middle of the box is at the median (12.50). The smallest observation is 12.10, and the largest observation is 12.70.
- If we want to calculate the correlation coefficient of two variables, we use the **cor** command, for example, we can use **cor(a,b)** to get the correlation coefficient of two variables **a,b**.

Exporting Images

- For the Windows version of R, if you right-click on your graphic that you've created, you can select **Copy as bitmap**. Then, in a Word or Open Office document, you can paste it in, using **Ctrl-V**.
- If you are using a Mac, it is probably easiest to hold down **Command-Shift-4** then click and drag the mouse to highlight the image. When you release the mouse button, a screenshot of the highlighted area will be made to the desktop of your computer.
- If you are using RStudio rather than R, the graphing window has menu bar where **Export** is one of the options. Select **Export**, and then follow the prompts to save your graph as an image.
- You can also export images from within R. Suppose I wanted to export the boxplot from the previous page. I could enter:

```
boxplot(bolt.weight, odd.numbers, names = c("bolts","odd numbers"),
  main = "Boxplots of bolt weights and odd numbers",horizontal=TRUE)

dev.copy(jpeg,file.choose())

dev.off()
```

- In the **dev.copy()** command, I am specifying the image to be exported as a JPG file. The **file.choose()** option will cause R to open a Windows file explorer window where you can choose the location and title of your image file.

  The **dev.off()** command is required to complete the exporting of your file.