MIS2502: Data Analytics

In-Class Exercise: Getting Familiar with R/RStudio

This exercise involves running an R script for the first time using RStudio and learning the basics of the environment and the language. So here’s some basic terminology:

* **R** is “a programming language and software environment for statistical computing and graphics” (source: Wikipedia).
* **RStudio** is an integrated development environment (IDE) for R. It runs “on top” of R and makes R easier to use.
* A **script** is a set of commands in a text file that carry out a set of operations.
* A **variable** is a symbol that contains a value. Values can be numbers or letters and are assigned to variables. For example:
age <- 30; assigns the value 30 to the variable age
name <- “Bob”; assigns the value Bob to the variable name
* A **function** performs a more complex operation, like plotting a chart (plot()), or displaying something on the screen (print()). You know you’re looking at a function and not a variable if there are parentheses after the symbol. For example, consider:
print(name); print is the function, name is the variable
print(“Bob”); print is the function, “Bob” is the value
print(30); print is the function, 30 is the value

**Part 1: Run the Test Script
(if you want to make sure R works on your own computer, you can just do Part 1)**

1. Download the files TestScript.r and USBalanceOfTrade.xlsx from the Community Site post where you got these instructions. Save those files to a place where you can find them again.

**I suggest you create a new folder on your hard drive called RFiles (i.e., C:\RFiles). If you’re using a lab computer, just create the RFiles folder on a flash drive.**
2. Open USBalanceOfTrade.xlsx in Excel and verify the data is there. It’s a list of the total value of US Exports and Imports every five years (except 2013) starting in 1960.


3. R can read Excel files, but it is easier for it to read a plain text file. So save the file as a comma-separated file (csv). Go to File/Save As… and then select “CSV (Comma delimited) (\*.csv)” from the drop-down menu.
**Make sure you save the file in the same location as the other files you just downloaded.**
4. Start RStudio. RStudio is an Integrated Development Environment (IDE) that makes it easier to use R. **Always use RStudio instead of starting R directly**. It will make your life much easier!
5. Go to the File menu and select “Open File…”
6. Browse for the TestScript.r file. It will open and look like this:


7. Notice that on line 15 it identifies USBalanceOfTrade.csv as your input file. It does this by assigning a value to a variable. In this case the value is “USBalanceOfTrade.csv” and the variable is INPUT\_FILENAME.

Now when we refer to INPUT\_FILENAME on line 23, it will retrieve the value USBalanceOfTrade.csv.
8. We still need to tell RStudio where to find the input file. The easiest way to do this is to go to the Session menu and select Set Working Directory/To Source File Location.



It will create and execute a setwd() function (as in “set working directory”) in the Console window at the bottom of the screen:



Yours may look a little different. That’s ok. Trust RStudio!
9. Now run the script. Go to the Code menu and select Run Region/Run All…


10. If everything works, the script will run and you’ll see this plot in the bottom right of the screen:


11. That’s it. You have successfully run your first R script using RStudio!

**Part 2: Try Changing Values in the Script**

Let’s play around with variables and values.

1. Go to line 16 and change the LINE\_COLOR variable to blue. Make sure the word “blue” is still in quotes. Then rerun the script (Code/Run Region/Run All).
2. Go to line 17 and change the LINE\_WIDTH to 20. The number will NOT be in quotes, and don’t put them in! Then rerun the script.
3. Now check out line 61 of the script:

lines(tradeBalance, col=LINE\_COLOR, lwd=LINE\_WIDTH);

The lines() function draws a line on the chart connecting the plotted points. To see the plotted points by themselves, put a # sign in front of lines():

**#**lines(tradeBalance, col=LINE\_COLOR, lwd=LINE\_WIDTH);

This instructs R to skip the line. Now rerun the script.

**Try It Yourself:**

Go to lines 52 and 53. The plot() function creates a scatterplot of the points in your data set. Look at the comments in the script above the plot() function to understand how it works and then change the title to “An Important Graph!” Rerun the script to make sure it works.

**Part 3: Descriptive Statistics Using R**

Now that you’ve gotten started with RStudio and R, let’s dive into a script that does some real analysis.

The script we’ll be working with does some pretty simple things: present descriptive statistics, plot a histogram, and test the difference between means. So this walkthrough doesn’t present any new concepts in statistics – it is to acquaint you with the syntax of an R script.

**Get Set Up**

1. Download the files Descriptives.r and NBA14Salaries.csv from the Community Site post where you got these instructions. Save those files to a place where you can find them again.

**I suggest you use the folder you created back in Step 1 (i.e., C:\RFiles).**
2. Start RStudio.
3. Go to the File menu and select “Open File…”
4. Browse for the NBA14Salaries.csv file. This is the data file. When you open the file you’ll see this:



The file is a list of NBA players’ names, salaries (2014), and the position they play.

Each row of data is on a separate line. Each column of data is separated by a comma (,) – this is why it is called a comma-separated (or comma delimited) file.

*You can also open (and edit) this file in Excel, but any formulas you enter will be converted to their values and any formatting will be lost when you save the file in CSV format.*

1. Close the file by going to the File menu and selecting Close.
2. Go to the File menu and select “Open File…”

**Look Through the R Script**

1. Browse for the Descriptives.r file. This is the R script file – this contains all of the commands R uses to analyze your data. When you open the file you’ll see this:



There are a lot of **comments** in the file to explain how everything works. This file is 105 lines long but most of those are comments – so pay attention to them!

Comments start with the **#** symbol. That tells R to ignore what’s on the rest of the line – you could remove all the comment lines and it wouldn’t affect the script. For example, check out line 14:

# INPUT\_FILENAME The name of the file that contains the data (CSV format)

This is just telling you what the variable INPUT\_FILENAME is used for. Notice that comment lines are color-coded in green.

1. Lines 11 through 27 are pretty typical of the R scripts you’ll use in this course. This is a section of variables that allow you to customize the settings for the rest of the analysis. Most of the changes you’ll make to the R scripts in this course will be limited to this section of the file.

*Don’t go changing things outside of the “Variables” section of the file unless you’re instructed to do so or you really know what you’re doing. Otherwise you can create a mess. If you feel the urge to play around, at least make a backup copy of the script before you start!*
2. So look closely at lines 21 through 27. Those are **creating and assigning values to variables**. Variables hold values that can be numbers or strings (i.e., letters, numbers, and symbols).

String values have quotes around them – numeric values do not. But otherwise, variable assignment statements always have the same format:


So if you wanted to change the value of INPUT\_FILENAME, change what’s in-between the quotes on the right, like this (BUT DON’T DO IT – We need to work with NBA14Salaries.csv!):

So line 24:

NUM\_BREAKS <- 25;

Assigns a value of 25 to the variable NUM\_BREAKS. When you look at the comment (line 17), you see that NUM\_BREAKS is the number of buckets (bars) that will appear in our histogram.

Notice that the variable names are black, the assignment symbol is blue, string values are green, and numeric values are blue. The color-coding is handy when something doesn’t work – it helps you figure out if you’ve made a typo!

1. Scroll down to lines 29 and 30:

if (!require("psych")) { install.packages("psych")

require("psych") };

R is also a development platform that allows for anyone to create special modules, called packages, that add new features. We’re going to use a package called “psych.”

The **install.packages("psych")** statement tells R to download a package and install it. So when you run the script you’ll see a dialog box:



It will do this every time that it detects that the package is not loaded on your computer. If it is already installed, it won’t load it a second time.

Another thing: packages() is a **function**. A function performs an action, like installing a package or loading a library. You know it’s a function because there are parenthesis after the command. Zero, one, or more values go inside the parenthesis, depending on what you want the function to do – those are the values that the function needs to complete its job.

1. Now let’s run the script. Set the working directory to the location of your R script by going to the Session menu and select Set Working Directory/To Source File Location.
2. Go to the Code menu and select Run Region/Run All.
3. You’ll know if it worked because you’ll see a histogram in the bottom right corner of the screen:



1. But that’s not the only output. It generated some files that it placed in your working directory (we’ll look at those later) and sent a lot of output to your console window (bottom left of the screen).

Scroll up through the Console output and you’ll see the results of various tests on this data. Locate the “Welch Two Sample t-test”:



We’ll discuss what this means later – just verify that it generated this output for now.
2. Now go to line 36:

dataSet <- read.csv(INPUT\_FILENAME);

This reads the data from our input file (NBA14Salaries.csv, check the variable settings), then assigns it to the variable dataSet. Now when we reference dataSet, we are talking about our NBA player data.

1. We want our output to go to a file as well as the screen (the console). This will make it easier to read later. So we have line 42:

sink(OUTPUT\_FILENAME, append=FALSE, split=TRUE);

The sink() function redirects the output to the file OUTPUT\_FILENAME. We also instruct R to NOT append (append=FALSE) – it will overwrite the old file each time – and to also send the output to the screen (split=TRUE) so we can see it’s doing what it should.

*Note that this time we sent three values to the sink() function, separated by commas – other functions like setwd() only took one value. With multiple values, order is important, so make sure you read the comments in the script carefully if you’re going to change anything!!*

1. You can read the rest of the comments to see what each command does, but there’s one more thing about syntax to know. Check out line 58:

summary(dataSet$Salary);

summary() is a function that presents summary statistics about a data set, or an individual data field (column). So by using dataSet$Salary, we’re telling summary() to pick out Salary from the rest of the data and just analyze that. The output from summary looks something like this:

Min. 1st Qu. Median Mean 3rd Qu.

35000 1036000 2511000 4142000 5586000

Max.

30450000

1. You can type commands directly into the console window. So try it – scroll to the bottom of the console window (bottom left window in RStudio) and type:

summary(dataSet);

Then press Enter.

You’ll see the following output - a summary of all three data fields (Name, Salary, and Position) in the data set.

 Name Salary Position

 Alan Anderson : 2 Min. : 35000 PG:110

 Andre Iguodala: 2 1st Qu.: 1036212 SF:102

 Arron Afflalo : 2 Median : 2511432 SG:115

 Avery Bradley : 2 Mean : 4141913

 Beno Udrih : 2 3rd Qu.: 5586120

 Brandon Heath : 2 Max. :30453805

 (Other) :315

1. Now, read through the rest of the script and the comments. Pay special attention to where the following things occur:

	1. Computing the standard deviation of the Salary variable (line 59).
	2. Providing descriptive statistics for salary, grouped by player position (line 66).
	3. Selecting the players who are point guard and small forwards (line 80).
	4. Performing a t-test (line 85).
	5. Plotting the histogram (line 98) and creating a PDF with the graphic (lines 103-105).

**View the Output**

1. Find your working directory (the folder where you files are stored). Focus on two files: descriptivesOutput.txt and histogram.pdf.
2. Open descriptivesOutput.txt. You can do this in RStudio, Word, Notepad, or any other editor.
3. We’ll take a look at the sections of output, one by one:

**Output from summary(dataSet$Salary):**

These are the summary descriptive statistics generated by the summary() function for the Salary data field. Ignore the comment fields in the output.

 Min. 1st Qu. Median Mean 3rd Qu. Max.

 35000 1036000 2511000 4142000 5586000 30450000

This displays the minimum value (i.e., the lowest paid NBA player makes $35,000), the maximum value ($30,450,000), the mean salary ($4,142,000), the median salary ($2,511,000), and the salaries for the first and third quartiles.

**Output from sd(dataSet$Salary):**

Just a simple computation of standard deviation of Salary:

[1] 4610687

**Output from describeBy(dataSet$Salary,dataSet$Position):**

This is similar to summary(), but splits the data set into groups, organized by player position:

group: PG

 vars n mean sd median trimmed mad

1 1 110 4076415 4594908 2175554 3164817 2055895

 min max range skew kurtosis se

1 35000 21466718 21431718 1.83 3.06 438107.3

--------------------------------------

group: SF

 vars n mean sd median trimmed mad

1 1 102 4193529 4474942 2801280 3366262 2818547

 min max range skew kurtosis se

1 35000 21679893 21644893 1.74 2.93 443085.3

--------------------------------------

group: SG

 vars n mean sd median trimmed mad

1 1 115 4158784 4780810 2653080 3271857 2710311

 min max range skew kurtosis se

1 35000 30453805 30418805 2.45 8.2 445812.9

There are lots of stats here, but you should recognize mean, standard deviation (sd), and median. And we learn from this that point guards’ average salary is $4,076,415, small forwards’ average salary is $4,193,529, and shooting guards’ average salary is $4,175,784.

The question is: are these average salaries significantly different in a statistical sense?

**Output from
hist(dataSet$Salary, breaks=NUM\_BREAKS, col=HIST\_BARCOLOR, xlab=HISTLABEL):**Open the histogram.pdf file to see the output from this command.

****The data is not normally distributed, which given the size of our data (over 100 players in each group) is unlikely to be a problem for our t-test. But it is good information to have.

**Output from t.test(subset$Salary~subset$Position):**

Now back to the descriptivesOutput.txt file. This performs a t-test, comparing point guards (PG) to small forwards (SF). It excludes shooting guards because on line 85 we defined the variable **subset** as containing only the data where Position was PG and SF.

Welch Two Sample t-test

data: subset$Salary by subset$Position

t = -0.188, df = 209.488, p-value = 0.8511

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

 -1345478 1111250

sample estimates:

mean in group PG mean in group SF

 4076415 4193529

We can see that the p-value is 0.8511, indicating that by any standard we should **not** reject the null hypothesis that there is no difference between the means. In other words, the two player groups, statistically, have the same average salary.