**In-Class Activity #10: Descriptive Statistics Using R/RStudio**

|  |
| --- |
| **Submission Instructions****What to submit:**Submit three files on Canvas: (1) **descriptivesOutput.txt,** (2) **histogram.pdf,** and (3) **your answer to the “Try It Yourself” question on the last page**. These first two files will be created if you follow the steps until Step 20 (on Page 6). * **If Canvas does not allow you to submit the descriptivesOutput.txt file** – copy the content of the descriptivesOutput.txt file to a Word file, name the Word file descriptivesOutput.docx, andsubmit the Word file instead.
 |

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: (1) present descriptive statistics, (2) test the difference between means, and (3) plot a histogram. Therefore, this walkthrough doesn’t present any new concepts in statistics – it is to acquaint you with the syntax of an R script.

**Part 1. Getting Started with R/RStudio**

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.
2. Browse for the NBA14Salaries.csv file. This is the data file. Open the file in Excel and 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 NBA14Salaries.csv file.
2. 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!
3. Go to the File menu and select “Open File…”.
4. Browse for the Descriptives.r file and open the file.

**Part 2. Look Through the R Script**

1. 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:



1. There are a lot of **comments** in the file to explain how everything works. This file is 113 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 double 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, you will need to change what’s in-between the quotes on the right.

Now check line 26:

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.

|  |
| --- |
| **Variable Assignment:**Another way to assign a value to a variable is to use the “=” sign. For example, the following are equivalent:NUM\_BREAKS <- 25NUM\_BREAKS = 25 |

Notice that the variable names are black, the assignment symbol is grey, 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 29-31:

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

require("psych") }

R is a development platform that allows for anyone to create special modules (addons), called packages, that add new features. We’re going to use a package called “psych.” The psych package provides functions for presenting descriptive statistics.

1. The if (!require("psych")) condition checks whether the psych package was previously installed in your computer. If already installed, the psych package will be loaded.
2. If the psych package is not yet installed, 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 installed on your computer. If it is already installed, it won’t load it a second time.

|  |
| --- |
| **Functions:**Both require() and install.packages() are **functions**. A function performs an action, like installing a package or loading a library. **You know it’s a function because there are parentheses after the command.** Zero, one, or more values go inside the parentheses as input (arguments), depending on what you want the function to do – those are the values that the function needs to complete its job. |

1. Now go to line 40:

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.

**Part 3. Now let’s run the script.**

1. **Set Working Directory**. We need to tell RStudio where to find the input file by setting the working directory. The easiest way to 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**. This step is very important!!!



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 depending on your OS and the directory you use. That’s ok. Trust RStudio!):



1. Now run the script. Go to the **Code** menu and select **Run Region/Run All** (see the screenshot below).



1. You’ll know if it worked because you’ll see a histogram in the bottom right corner of the screen:



1. That’s it. You have successfully run your first R script using RStudio!
2. 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.
3. Note that the histogram and the console output are not the only outputs. The R script also generated some files in your working directory.

For example, if you check your working directory (the folder where your files are stored), there will be two files created: **descriptivesOutput.txt** and **histogram.pdf**.

(If you are not sure what the working directory is, type **getwd()** in the console and it will tell you the location of the working directory.)

**Part 4. More about the R Script**

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 in the R script, we have line 46:

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 61:

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:

> summary(dataSet$Salary)

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

 35000 1036000 2511000 4142000 5586000 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.

> summary(dataSet)

 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

|  |
| --- |
| **The summary() function:**Notice that the form of results returned by the **summary()** function depends on the data type of the fields.* Character values. Name and Position are of character values. For character values, the summary () function will return the number of observations for each possible value.
* Numeric values. Salary has numeric values. For numeric values, the summary () function will return the min, 1st quartile, median, mean, 3rd quartile, and max.
 |

1. Now look at line 66.

describe(dataSet$Salary)

**describe()** is a function provided by the *psych* package that presents more summary statistics such as standard deviation (sd). The output from summary looks something like this:

> describe(dataSet$Salary)

 vars n mean sd median trimmed mad min max range skew kurtosis se

1 1 327 4141913 4610687 2511432 3264683 2541009 35000 30453805 30418805 2.07 5.21 254971.6

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

	1. Providing descriptive statistics for salary, grouped by player position using the **describeBy()** function (line 73).
	2. Selecting the players who are point guard and small forwards (line 87).
	3. Performing a t-test using the **t.test()** function (line 93).
	4. Plotting the histogram inside RStudio using the **hist()** function (line 106) and creating a PDF with the graphic (lines 111-113).

**Part 5. View the Output Files: descriptivesOutput.txt and histogram.pdf**

1. Find your working directory (the folder where your 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 describe(dataSet$Salary):**

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

 vars n mean sd median trimmed mad min max range skew kurtosis se

1 1 327 4141913 4610687 2511432 3264683 2541009 35000 30453805 30418805 2.07 5.21 254971.6

We can see that the mean, minimum, maximum, median values are the same as provided by the summary() function. But the describe() function displays a few additional statistics, such as the number of obervations (n= 327), standard deviation (sd = 4610687), range (30418805), and skewness (2.07).

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

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

group: PG

 vars n mean sd median trimmed mad min max range skew kurtosis se

1 1 110 4076415 4594908 2175554 3164817 2055895 35000 21466718 21431718 1.83 3.06 438107.3

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

group: SF

 vars n mean sd median trimmed mad min max range skew kurtosis se

1 1 102 4193529 4474942 2801280 3366262 2818547 35000 21679893 21644893 1.74 2.93 443085.3

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

group: SG

 vars n mean sd median trimmed mad min max range skew kurtosis se

1 1 115 4158784 4780810 2653080 3271857 2710311 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 87 we defined the variable **subset** as containing only the data where Position was PG and SF.



Now let’s take a closer look at the results from t-test:

|  |
| --- |
| Welch Two Sample t-testdata: subset$Salary by subset$Positiont = -0.188, df = 209.488, p-value = 0.8511alternative hypothesis: true difference in means is not equal to 095 percent confidence interval: -1345478 1111250sample estimates:mean in group PG mean in group SF  4076415 4193529  |

From the output, we see that the **alternative hypothesis (H1)** is: **true difference in means is not equal to 0**.

The **null hypothesis (H0)** is simply the opposite: **there is no difference between the means**.

We can see that the p-value is 0.8511. A p-value that is larger than 0.05 indicates that we **fail to reject** **the null hypothesis (H0)** that there is no difference between the means. In other words, with a p-value of 0.8511>0.05, we conclude that the two player groups, statistically, have the same average salary.

**Try It Yourself:**

Now we want to compare point guards (PG) to shooting guards (SG). The only change you need to make is in Line 87: change 'SF' to 'SG', and Line 87 will now look like this:

subset <- dataSet[ which(dataSet$Position=='PG' | dataSet$Position=='SG'), ]

Now re-run the script. Go to the Code menu and select Run Region/Run All…

Based on the new output, are these average salaries significantly different in a statistical sense?

Hint:

*A small p-value (typically ≤ 0.05) indicates strong evidence against the null hypothesis, so you reject the null hypothesis.*

*A large p-value (> 0.05) indicates insufficient evidence against the null hypothesis, so you fail to reject the null hypothesis.*