MSDS-650 Data Analytics

Week One Lab Assignment

Regis University

Name:

Date:

The first week learning objectives are intended to make sure you have R installed and working properly. Specifically, one objective is to "Characterize problems that can be addressed using data analytics." This lab will give you more practice on these objectives.

Your Week One Assignment had you perform some basic data collection and preparation, and exploratory data analysis. Later in the program you may take a course on each of those. This week we will extend the assignment to some real-world examples.

**Always include the code you executed with results when pasting from console. For example:**

> head(iris) #confirm the above commands worked

X5.1 X3.5 X1.4 X0.2 Iris.setosa

1 4.9 3.0 1.4 0.2 Iris-setosa

2 4.7 3.2 1.3 0.2 Iris-setosa

3 4.6 3.1 1.5 0.2 Iris-setosa

4 5.0 3.6 1.4 0.2 Iris-setosa

5 5.4 3.9 1.7 0.4 Iris-setosa

6 4.6 3.4 1.4 0.3 Iris-setosa

Before you start the lab, bring up some documentation. Ignore the warning the first command may give you.

Run

> vignette(package="data.table")

> vignette("datatable-faq")

> vignette("datatable-intro")

**1. Load directly from a URL.**

In the assignment you may have used

> iris <- read.csv(file.choose(), header=T)

to load your data set. You can also specify an absolute file path by enclosing it in quotes. In Windows you have to replace back-slashes in the path with forward-slashes. For example,

> iris <- read.csv("D:/MSDS650/iris.data", header=T)

Now research and/or work with your classmates to learn how to download directly from a URL Download a set of raw network packets from <https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/a%20part%20of%20training%20and%20testing%20set/UNSW_NB15_training-set.csv>. The download takes about three minutes on an Ethernet connection. It may take longer over WiFi.

See <http://www.secrepo.com/> for more datasets if you are interested in cybersecurity.

**a. Read the csv file from the URL. Paste the command here:**

**b. Paste the internal structure of your object (see the Week One Assignment for code) below. For these assignments include your console output with the command you executed:**

**2. How many TCP, UDP, SCTP, and ARP packets are there in the data set?**

**a. To use the data.table package, you have to convert your object to a data table. If you have not done so already, convert the object you created upon reading the data into a data table. Omit this step if you combined it with the read.csv() command.**

**b. Your assignment had you count irises grouped by sepal area. Examine your data object to see which variable TCP, UDP, SCTP, and ARP belong to. This is the variable to group by. Following the example, count occurrences of all 131 unique members of this field. Your output will show you the top and bottom five results. One of the four will not show up in the top or bottom five. Think about how to see all the results.**

TCP:

UDP:

SCTP:

ARP:

**3. What is the mean packet size by protocol? Use the spkts variable. Only provide the first five from your console output.**

**4. Repeat Step 3, but this time provide the five protocols with the greatest mean packet size. Do not sort manually and show your work.**

**a. Read "Introduction to the data.table package in R" page 2-3 to see how keys work. You could use setkey(), but this only supports sorting in ascending order. Of course, you could take the last values, but this is an inelegant solution. Setorder() supports sorting in ascending and descending order. Type ?setorder in the RStudio console to see how to sort in descending order.**

**5. You used lapply() in the Week One Assignment. It is a member of the apply() family, which is an alternative to writing loops in R. lapply() is for evaluating functions applied to lists. It outputs a list regardless of the data input class. But let’s explore apply() first.**

**a. First, use a text editor to create a small matrix on disk. You can use any file extension you like, but either .txt or .csv would be most intuitive.**

**1, 2, 3**

**4, 5, 6**

**7, 8, 9**

**Use read.csv() or read.table() to read your file into an object called “my\_data”. Paste your read function and my\_data below. This will take two commands:**

**Note: if you get this warning, there is something wrong with your file, not your R command:**

Warning message:

In read.table(file = file, header = header, sep = sep, quote = quote, :

incomplete final line found by readTableHeader on '3by3.txt'

**Troubleshoot it. R warning and error messages can be cryptic. This is a good example, but there is a hint is in the message.**

**b. Write an apply() functions to calculate the sums by row:**

**c. Write an apply() functions to calculate the sums by column:**

**d. Write an apply function to calculate the sums by row and column. Use margin = c(1,2):**

The output is confusing. It may look like the sum function didn’t do anything, but it did. It summed up every element by itself, which equals that element.

**Now replace “sum” in your apply command with a simple function and** **run again. Use**

function(x) x + 1:

You should see that your apply() function took each element of your object individually and added one.

For more about R functions and the apply() family, see:

Functions:

<https://www.youtube.com/watch?v=i2VH5jIL76Y>

Apply Family of Functions in R Part 1: apply():

<https://www.youtube.com/watch?v=f0U74ZvLfQo>

Apply Family of Functions in R Part 2: lapply() and sapply():

<https://www.youtube.com/watch?v=ejVWRKidi9M>

Apply Family of Functions in R Part 3: tapply():

<https://www.youtube.com/watch?v=HmBPDTtb6Bg>