MAGIST

Geog 413

Module 1 Exercise 1

Getting Started with R and RStudio

1. R is a computing (software) environment that was developed for data analysis and visualization. R includes a built-in series of operators for data storage, retrieval, manipulation and display. R also has a simple programming language to help execute a series of scripts (user commands).

2. R is open source and relies upon the production of scripts that are “donated” by users. These scripts are typically called packages and are individually loaded into R to perform particular kinds of operations.

3. While R is standalone and scripts (commands) can be written directly into the R console and executed, it is also possible to save and edit a chain of commands in a text editor and then execute these commands within R. RStudio is an integrated development environment (IDE) that allows you to write, save and read existing scripts, execute them and visualize output, and perform a variety of other tasks.

4. You have access to R and RStudio through the MAGIST site on AWS. However, your best bet is to download R and RStudio for free on your own desktop/notebook from:

https://www.r-project.org/

https://www.rstudio.com/

I am going to assume that you have downloaded R and Rstudio to your own computer from this point on.

A First Session in RStudio

1. Open RStudio by double clicking on the RStudio icon.

2. 3 or 4 panes will open within RStudio. The top-right pane is the Environment/History window. Clicking the History button shows the commands that you have entered in the current session of R. The Environment button shows recent datasets that you have accessed. The bottom-right pane is the Viewer/Help pane. Help commands are shown with the Help button, while graphics output is shown under the Viewer button. Note that graphics output can be saved in various formats using the Plots button. The left/bottom-left pane is the R-console. This is an interactive pane that allows you to enter commands and retrieve the output. A top-left pane is typically where you load existing scripts or programs. You can access these by clicking on the File menu in the menubar at the top of the RStudio window. Use your mouse and navigate around these different panes.

3. As you begin to work in R it will be important to develop a folder (directory) on your computer where you can save R scripts and the output from your analysis. To do this, click on the Session button in the top menubar, choose the Set Working Directory tab, select Choose Directory and specify the folder of your choice. (Hint: You might make a folder to hold your R files under a MAGIST or class directory/folder and then select this with the Choose Directory command in RStudio.)

3. As we learn R, we will use a variety of different packages (pre-written scripts) that help us with various tasks. We add these packages to our base version of R by installing them. dslabs is a package written to accompany the book, *Introduction to Data Science* by Rafael Irizarry. The URL for this free, online book is

https://rafalab.github.io/dsbook/ . We will use a few chapters from this book to help introduce R. I am giving you just a working knowledge of R through the class Exercises. If you want a little bit more, read through Chapters 1 and 2 of the Irizarry book.

There a number of introductions to R available from the internet. One of the most well-known is the Owen guide

https://cran.r-project.org/doc/contrib/Owen-TheRGuide.pdf

You can find many more “Introductions to R” with a Google search.

4. To install the package dslabs, go to the R-console and type.

```
> install.packages(“dslabs”)
```

followed by the enter/return key.

Don’t type the > , this is the prompt from R asking you to enter a command. Note that from this point on I won’t tell you to hit the enter key anymore, but you will do that after most all the commands below.

The dslabs package should now be installed. (Hint: You can also install packages by selecting install packages from the Tools button on the top menubar.) The dslabs package is now resident on your computer (or on the AWS site for R). You only need to install a package once unless you reinstall R. To see what packages are already installed use the command

```
> installed.packages()
```

5. To load a package into your current R workspace you use the library function

```
> library(dslabs)
```

No response from R suggests that the library has been loaded. Any issues would have resulted in an error message. Note that you have to load required libraries each time you start R.

6. Along with some code snippets, the dslabs package also contains some data files. To get a list of these, type in the R console

```
> data(package = ”dslabs”)
```

The data files in this package should be listed in the scripting pane (top-left). You might already have found out that spaces in most commands are ignored. However, the case of the letters you type is critical. So typing the command

```
> data(package=”Dslabs”)
```

would generate an error message. Try it and see.

7. To see all datasets that are loaded in different packages, including base R, type

```
> data()
```

Note that the datasets listed in the package “datasets” at the top of the output are loaded with the base R program. These datasets are always available to you. To access other datasets that come with different packages, you usually load those packages using the library function as above.

8. There are various ways to examine a dataset. To look at the file co2 (not CO2 which is a different file), just type the name of the file

```
> co2
```

You should see a list of the variables in the dataset and a series of observations on those variables.

9. If you have a large dataset, it is not very efficient to explore it using the filename only. In most cases, you might use the head command

```
> head(co2)
```

and the top 6 or so observations in the dataset will be listed.

10. The help command is useful for finding about the specific of commands and datafiles:

`> help("head")` or `> ?head` - tells you about the head command

`> help(co2)` - gives you information on the co2 dataset

11. Note that you could access the murders dataset in the dslabs package without using the library function with the commands

```
> data(murders, package="dslabs")
```

`> head(murders)` or just

```
> head(murders, package="dslabs")
```

There are several ways to do the same thing in R!

12. Let's write some simple R-script. Click on the File menu at the top of the page, select New File and then R Script. A new scripting pane (top-right) opens up. To run some code (or run a script), you enter the code in the scripting pane and then execute it by highlighting the script and clicking on the Run command (just above the scripting window). Try working through the following code. Execute the scripts one at a time – you will them reproduced in the R-console window below. Then you can highlight the script as a whole and run it all together. Make sure that you save your first R-script, by clicking on the File menu key at the top of the RStudio window and select save as. You should save the file to your R working directory. You can then reload it later and run it again. I add a series of comment lines (that start ##) in the script below:

```
##First, note that this is a comment line - useful in longer scripts
## Let us first load the dslabs library. After typing this command, highlight
## the command with the cursor and then click on the Run command just above this pane.
## You should see the command in the R-console pane where it is executed. You do not need to enter >

library(dslabs)

##And then load the dataset temp_carbon

data(temp_carbon)

##Another way of finding out information about the structure of
## this dataset is to use the str command

str(temp_carbon)
```

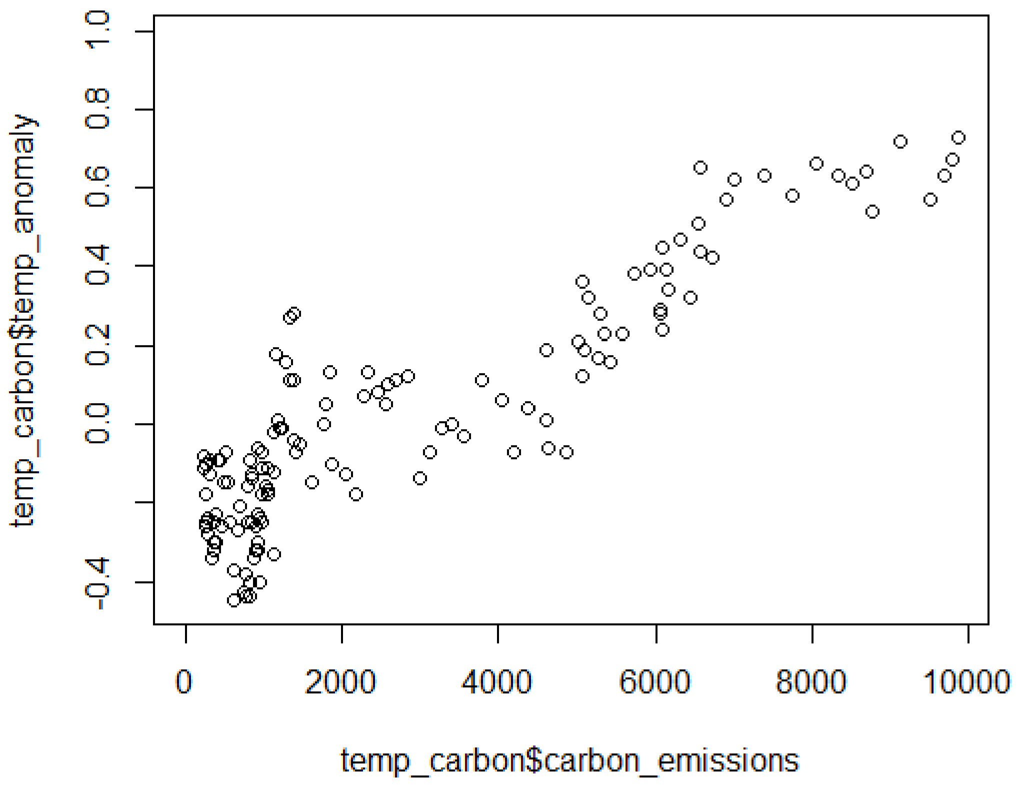
```
##Note that the datset contains 268 observations on 5 variables
##You can use the head command also

head(temp_carbon)

##Theory tells us that there is a positive relationship between carbon emissions
##and temperature anomalies (differences between temperatures and long-term average temperatures)
##To check this, generate a simple scatterplot
##By convention we place what we call the dependent variable on the Y-axis
##The dependent variable is a variable that we think has its values influenced by another
##independent variable. The dependent variable in this case is temperature anomalies

plot(temp_carbon$carbon_emissions, temp_carbon$temp_anomaly)

##You should see the following in the plot window
##Note that the $ operator in the plot command tells R to use the named variable in the named file
```



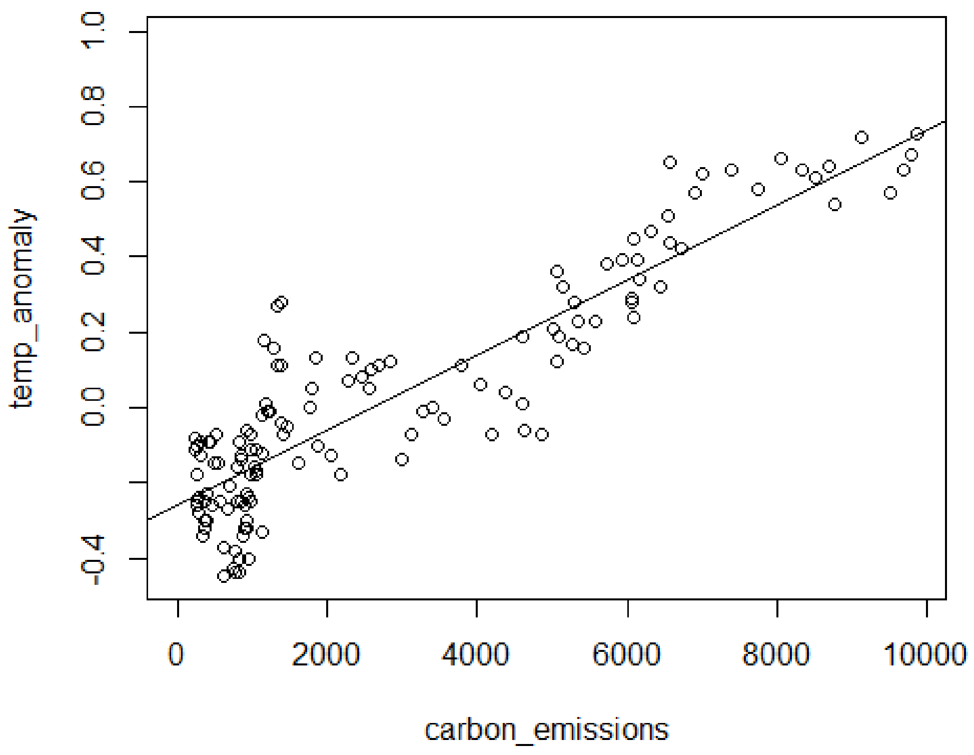
```
##How did I capture the image? From the plot window I clicked on the Export button and saved the file as
##an image (PNG) file. Then I just used the insert command in Word to paste it above.
```

```
##We can add a best-fit line to the plot using the regression command

reg1 <- lm(temp_anomaly~carbon_emissions, data=temp_carbon)

##And add this line to the plot as follows

with(temp_carbon, plot(carbon_emissions, temp_anomaly))
abline(reg1)
```



```
##Look at the plot, what does it say to you? Does it support the theoretical claim noted above?
##The regression line in the plot is also handy. The slope of that line is positive, meaning that
##as the value of carbon_emissions increase, so temp_anomaly values also increase.
##The slope of the line tells you that for every 1-unit increase in carbon_emissions, the temp_anomaly
##increases by the value 9.994e-05 (or 0.0000994). You ran a regression above and placed the output
##in an object named reg1. To look at that object, simply enter its name and run the script.

reg1

#In the R-console you will see the output

Call:
lm(formula = temp_anomaly ~ carbon_emissions, data = temp_carbon)
```

```
Coefficients:
    (Intercept)  carbon_emissions
     -2.591e-01        9.994e-05
```

```
##We will think more about what regression is doing later.
##Good job with your first R-script.
##Now save this script using the File menu and save as function.
##You can reload this script anytime and run individual pieces of it, or the whole thing.
```

Note that you can exit RStudio by right clicking the X at the top-right of the window, or by going to File and choosing to exiting RStudio.