CMPSC 301
Data Analytics
Spring 2020

Lab 2: Introduction to R

Objectives

To enhance the understanding of the basic R functionality, including the use of R Studio and producing data visualizations.

Reading Assignment

Please read Chapters assigned for this week's lessons which you will find in the class slides, in addition to reviewing your notes. Please take some time to gain experience with using Markdown to write your work. See *Mastering Markdown* `https://guides.github.com/features/mastering-markdown/` for more details about Markdown.

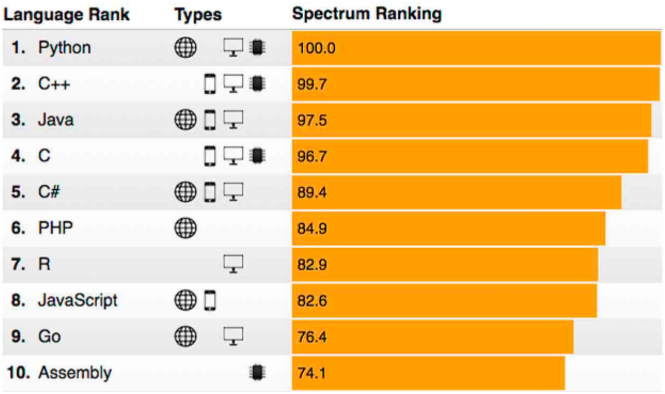
GitHub Starter Link

`https://classroom.github.com/a/lkJAQGDK`

To use this link, please follow the steps below.

- Click on the link and accept the assignment.
- Once the importing task has completed, click on the created assignment link which will take you to your newly created GitHub repository for this lab.
- Clone this repository (bearing your name) and work on the lab locally.
- As you are working on your lab, you are to commit and push regularly. You can use the following commands to add a single file, you must be in the directory where the file is located (or add the path to the file in the command):
  - `git commit <nameOfFile> -m ''Your notes about commit here''`
  - `git push`

  Alternatively, you can use the following commands to add multiple files from your repository:
  - `git add -A`
  - `git commit -m ''Your notes about commit here''`
  - `git push`



Language Rank	Types	Spectrum Ranking
1. Python	🌐 🖥 ▪	100.0
2. C++	📱 🖥 ▪	99.7
3. Java	🌐 📱 🖥	97.5
4. C	📱 🖥 ▪	96.7
5. C#	🌐 📱 🖥	89.4
6. PHP	🌐	84.9
7. R	🖥	82.9
8. JavaScript	🌐 📱	82.6
9. Go	🌐 🖥	76.4
10. Assembly	▪	74.1

Figure 1: IEEE's list of 2018's most popular languages.

Exploring R Programming

As shown in Figure 1 (IEEE's listing of 2018's top programming languages, `https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages`), R has become increasingly popular as an extensive language for industrial and academic applications. Thanks to its community of developers who write its open source libraries, R continues to grow. Besides its popularity, R is the most recommended language to learn when getting started with data analytics as it is created with statistics and data in mind. In this lab you are invited to complete some questions out of the book where you are to implement code in your responses.

In this lab you are asked to read the assigned sections of the book and complete all exercises from Chapters 3, 4 and 6 of the "R for Data Science" textbook (online version). As you remember, the chapter numbering in the printed version of the book is not the same as the numbering in the online version of the book. This lab's assigned exercises correspond to the following exercises in the online version of the book, found in `http://r4ds.had.co.nz/`.

For each question, please be sure to complete all its parts. The questions to answer are the following and have been taken from the book's web site.

{3.2.4, 3.3.1, 3.5.1, 3.6.1, 3.7.1, 4.4, 6.3 (number 1: See Twitter application note below)}

Note: For the question 6.3.1 (Twitter application), the link will take you to a listing of *tips and tricks* from the R community which may be useful in your own work. Find an interesting tip, explain it what it does and then discuss how you might use it in your own work.

Submission Information

Your answers to the assigned questions are to be typed up using the Markdown file `writing/responses.md`. Please be sure to add and label each question and its parts.

Steps to Plot Using GGplot()

Please follow the below steps to prepare some data to apply to the `ggplot()` plotting function. You will use these same steps to plot your own data.

1. Load your *tidyverse* library by entering,

   ```
   library(tidyverse)
   ```

   or using,

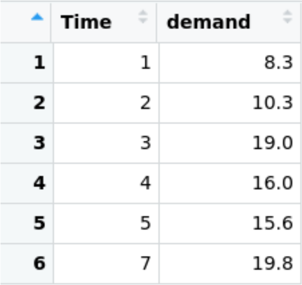
   ```
   install.packages("tidyverse")
   ```

   if the library has not already been installed on your machine. Note: You do not need to reinstall again, however, the library must be loaded to use its functions or keywords.
   Enter `? tidyverse` for more help.

2. Use `data()` to obtain a list of data sets which are included in R.

3. Each dataset has a name and a short description. Choose any set and assign it to a variable to make your coding work easier. `data <- name-of-your-dataset`. For example, to load the *BOD* dataset, use the following R code, `myData <- BOD`.

4. Inspect at the dataset *BOD* by entering `View(myData)` or, `View(BOD)` . This command allows you to determine what types of information are in its columns. Your output should match that of Figure 2.



	Time	demand
1	1	8.3
2	2	10.3
3	3	19.0
4	4	16.0
5	5	15.6
6	7	19.8

Figure 2: The output of `View(myData)` or, `View(BOD)`. Please note the capital '`V`' in '`View()`' command

5. Make a ggplot to study the *Time* by *Demand* relationship in the `BOD` dataset. Enter the following code.

   ```
   ggplot(data = myData) + geom_point(mapping = aes(x = myData$Time, y = myData$demand))
   ```

   Your (scatter) plot will appear like the one in Figure 3.
   Equally, you could try other types of plots such as,

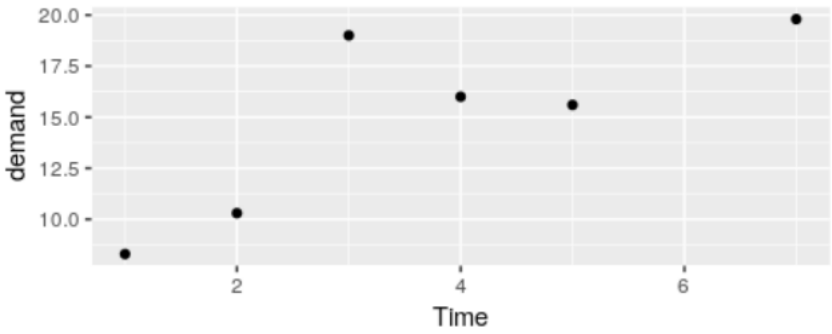


Figure 3: The output of the plot using `ggplot()`.

```
ggplot(data = myData) + geom_area(mapping = aes(x = myData$Time, y = myData$demand))
```

and your (area) plot will appear like the one in Figure 4. Note, in rStudio, if use a dollar-sign after the name of the dataset, (i.e., `BOD$`) and push `TAB` then you will see the column names in a pop-up panel. This coding will be helpful when assigning your x and y variables in `ggplot()`. A similar trick to show you what types of plots you can make is to enter `geom_` and enter `TAB`.

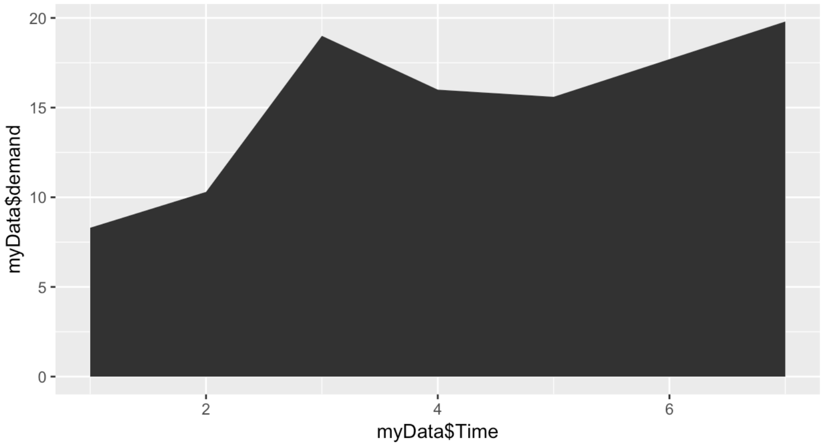


Figure 4: The output of an area plot using `ggplot()`.

After applying the above code into *rStudio* and checking for typographical errors, your plot should now resemble that of Figure 3.

6. Now you want to add a *smooth-line* to get an idea about the general trend in the data in terms of time. Your code is the following and must fit on one line.

```
ggplot(data = BOD)
+ geom_point(mapping = aes(x = Time, y = demand))
+ geom_smooth(mapping = aes(x = Time, y = demand))
```

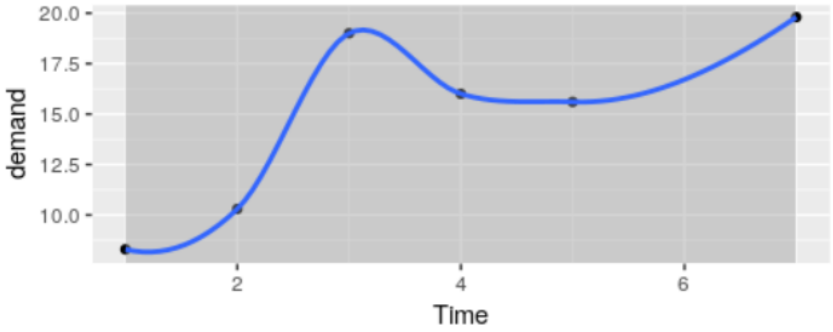


Figure 5: The second plot using `ggplot()`. Here we note that a smooth-line allows us to readily get an idea about the data without adding extra work in analysis.

After applying the above code into *rStudio* and checking for typographical errors, your plot should now resemble that of Figure 5.

Questions

Choose a new data set and make three plots that describe some type of pattern. For this task, if you are interested, you may elect to use an interactive plot from the `plotly` library (mentioned in the classDocs slides). To select your data, use the `data()` command to list the available datasets from which you are to create your plots. Note, it should be mentioned that not all the datasets will be equally convenient for applying to `ggplot()`. This because some of the datasets are there to give you practice in rearranging the values before a meaningful plot can be made. Other datasets contain data that must may not lend to plotting.

Below are the same questions that you will find in your reflection file in your working repository.

1. Please provide:
   (a) Full code for loading your chosen dataset.
   (b) Full code to plotting variables (i.e., points) from the data.
2. What trend did you see? Why do you think it was a trend in the data?
3. How did you decide to plot the x and y axes points that you chose?
4. If you enhanced the view by adding size and color aggregates to your plotting code, how did these additions help you to see something that was previously "invisible?"

Required Deliverables

1. Your **labeled** answers to the assigned exercises from Section "Exploring R Programming" are to be placed in `writing/responses.md`
2. Your plotting exercises are also to be placed at the end of your `writing/responses.md` submission.

When you have finished, please ensure that the GitHub web site has your pushed work by visiting your repository at the site. Please see the instructor if you have any questions about assignment submission.