American University - DATA 312

Instructions for Homework 1-A

Unit 1 - Data Visualization
Part A - Getting Started with R

Pre-requisite work:
Please use the installation guide provided for this course to install R and RStudio on your computer. If you encounter differences between the video guide and the written directions, please follow the written directions and let your instructor know about the discrepancy.

Work Process:
Open RStudio. Open a new R Script file (Ctrl+Shift+N). Use the Console to experiment with commands, then cut/paste the commands which work properly into your script. Use Control-S often to save your script! Your script will not auto-save!

You will submit only this one R Script as your homework. (We will not be using .Rmd files on this assignment. Only submit an .R file!) Your instructor will run this script on their computer to verify that it works when commands are run in the precise order you have specified. As such, you should run and re-run your script prior to submission, until it does exactly what you intend for it to do.

Learning R is like learning a language. Some of the things you will be asked to do, you will be expected to manage fully on your own, while some of the tasks will be done outright for you. You should think of programming in R as being on a continuum. Sometimes you will write lines fully from "scratch" while other times you will make small adjustments to others' code to achieve your desired effect. You should be careful not to borrow too much from other authors unless you credit their work. In this assignment, you will be doing quite a bit of borrowing, as you are extremely new to R. In this case, you do not need to credit anyone, because the code you are copying is provided by your instructor for that purpose.

As part of this learning process, **you should type out all these steps by hand**, without copying/pasting from these instructions. You are more than welcome to cut/paste from your own typing, but you won't learn anything if you just cut/paste everything you do in R. Like learning a language, you must practice. Typing the code that you are trying to understand is like repeating phrases in a foreign language. At the end of the assignment, **you will be asked to certify that you have hand-typed all your instructions**.

You should be able to complete this assignment using these written instructions alone. However, most students will benefit from watching the video. If you are still struggling after reading these instructions and watching the video, please consult your instructor for assistance. Your instructor is more than happy to help you on this exciting journey!

Begin by opening RStudio. Open a new script file and name it HW1A_userid where userid is your AU id. This will save a file called HW1A_userid.R, which is a text file. If you should wish to view it outside of R (for example - to print it), you may need to temporarily change the extension to .txt so other software will open it. (You may need to show extensions in order to change them.)

1. Create a Heading in your file. Use the '#' symbol at the start of a line to create a comment line. Put your name, date, assignment name, and instructor's name at the top of your script. For each objective given in this lab, create a comment line which references what portion of this assignment

you are working on. ("Step 5", "Step 6" etc is fine.) Verbal questions can be answered in your comments as well.

2. To make sure you are starting from a clean R session, add these lines to the top of your script.

```
rm(list=ls())
gc()
```

3. Invoke all the libraries you will use in your script right at the top. Do not use the "install" command here, just the "library" command. If any of these libraries fail to load, please refer to the instructions for installing them, which are given in the installation guide.

```
library(tidyverse)
    library(usmap)
```

4. Read COVID data and population data from public online data sources.

```
url<-"https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-counties.csv"
nytcovcounty <- read_csv(url)

url2<-"https://usafactsstatic.blob.core.windows.net/public/data/covid-19/
covid_county_population_usafacts.csv"  ###(Type it on one line!)

pop2019 <- read_csv(url2)
```

5. Convert FIPS from a word to a number so they will match in your two data sets.

```
nytcovcounty$countyFIPS <- as.numeric(nytcovcounty$fips)
```

6. Select a state ***(other than Texas)*** to investigate for this assignment. Texas will be used as the example state. Do not intentionally choose a state that is the same as another classmate's choice.

7. Plot the entire USA with county boundaries using:

```
plot_usmap(regions="counties")
```

8. Plot the entire USA with only state boundaries. (Figure out how to do this by experimentation.)

9. To restrict your map to a single state, use include as follows. (Use your chosen state, not Texas.)

```
plot_usmap(regions="counties", include=c("TX"))
```

10. To plot a collection of states, you can adjust the include. For example, try
include=c("TX","OK","LA","AR") in the command above.
In your script, create a plot with between 3 and 7 states which are near each other and include your chosen state.

11. We would like to color our states or counties with data, naturally! After the include but still inside the plot_usmaps parentheses, add
,data=countypop, values="pop_2015"

to see what this does to your plot for your chosen state! Notice that the dataset (dataframe) you are using is referenced without quotes, while the column you have chosen (values) is referenced by putting that variable (column) name in quotes. Each row of the data set is called an "observation", while each column is called a "variable".

12. In your script, generate a plot containing between 3 and 7 states which uses the state population values from 2015 for coloring the map.

13. Perhaps the legend on this map is not what you really want or except it to be. Let's make a new plot! After the plot_usmap(.....) , type
    + theme(legend.position="right")

    Note that this goes after plot_usmap(...), not within its parentheses. It should either be all together on one line in your script, or if you must split it, be sure to leave the + on the first line.
    This is so important, that it's worth doing it the wrong way, just to see what happens. Go ahead and save it in your script incorrectly, highlight those lines, and run it. Then, fix it and run it again! When you leave the + on the initial line, this tells R that something else is coming. It says, "Please connect this line and the next one!" Without that clue, R will execute the line as given, but the next line won't make any sense!

14. Perhaps you still dislike this legend or scale. Make another map. This time, add:
    + scale_fill_continuous(name="Population", label=scales::comma)

15. Did you know that RStudio has many built-in help features? The legend.position option "right" is not the only option. Create a map of your multi-state area again, coloring the states by their state populations. Next, start typing "theme(legend.position..." but when you finish typing the "n" in "position", just leave your cursor right there and wait a moment. Options should appear as if by magic! Choose a new legend position from this list! Create your new map!

16. For this question, list all legend position options given by RStudio in the previous question.

BASIC DATA WRANGLNG

Data wrangling is an art which takes time and effort to learn. However, some of the basic data wrangling tasks are so common, you really ought to know how to handle them. This demo is so you will understand the left_join command we are about to use in this assignment.

Type the following into your script, then execute one section at a time to see what happens.

```
a <- c (10,20,30,40)
b <- c ('y', 'n', 'n', 'y')
s <- c ('P', "Q", 'R', 'Q')    #Note: You may use single or double quotes in R.
df1 <- data.frame(a,b,s)
View(df1)
```

```
s <- c ('P', 'Q', 'R')
r <- c ('Papa', 'Quebec', 'Romeo')
df2 <- data.frame(s, r)
View(df2)

df3 <- left_join(df1, df2, by=c("s"))
View(df3)
```

Note: the column used in "join" must have the same variable name at the top of both original data frames. This is perhaps the simplest joining task. Think of the second data frame as a dictionary or lookup table containing additional information which needs to be added to each existing row in your first data set. (You probably prefer 2019 population data rather than 2015 data.)

17. We will do this type of join as follows:
    ```
    mdata <- left_join(nytcovcounty, pop2019, by=c('countyFIPS'))
    ```

18. What does nytcovcounty stand for? Where does it come from? (Look that up!)

19. Look at mdata to see how it is organized. The min command can be used on a column of data:
    ```
    min(mdata$date)
    ```
    What is the first date in this data set?
    What is the last date in this date set?
    What if you ask for the average (mean) of the date column? What do you get?
    What happens if you ask for the maximum of the state column?

20. We would like to know the COVID rate per thousand in each US county.
    ```
    mdata$perThou <- 1000 * mdata$cases / mdata$population
    ```

21. Save a slice of your most recent COVID data under a new name:
    ```
    mostRecent <- mutate(mdata) %>% filter(mdata$date==max(mdata$date))
    ```

22. Create a map of your chosen state, with county boundaries, colored by "cases", using a good title, making sure to adjust *data=…* and *values=…* and *name=…* Make sure your legend does not cover your graphic, and control the other legend properties so your legend is meaningful to a typical reader. Create your title using:
    ```
    +labs(title="your title")
    ```

23. Create a map of your chosen state, with county boundaries, colored by "deaths", with a good title and legend.

24. Create a map of your chosen state, with county boundaries colored by "perThou", with a good title and legend.

Note: Do not worry about the fact that your graphics are all being created one on top of the other. You can still see all the maps you have drawn by clicking forwards and backwards through them with the blue arrows which above and to the left of your plots.

25. Some states might have some "gray" counties. Does your chosen state have any gray (not on the blue-scale) counties? If so, explain why. Did the population map have any gray counties? Why not? Look at the raw data to figure out why you have gray counties. (If you don't have any, experiment with Texas in order to answer this question.)

Mutating Data

For this next section, we will be mutating data. We don't really want to destroy the data we have, so we'll make a copy to tear apart and play with. This example uses Texas. Use your chosen state.

```
TX <- mutate(mdata) %>% filter(State=='TX')  # This %>% is a pipe and the subject of R jokes!
TX2 <- mutate(TX) %>% filter(TX$date==max(TX$date))

TX2 <- TX2[order(-TX2$cases) , ]  #BEWARE of that little comma!!!
View(TX2)
```

We could do this next task with a function, but this time we'll do it manually. Select the top 5 counties from this list, and type them into a variable called top5. Here is how to do it for Texas:

26. top5 <- c("Harris", "Dallas", "Tarrant", "Travis", "El Paso")

27. View(TX) # Just to remind yourself of what's in that data set

28. TX4 <- TX # Make a copy, because we're about to destroy it!
    View(TX4) # And keep that open as we do the next step!

29. Get rid of deaths, perThou, abbr, county, population, state, and fips. For example:
        TX4$county = NULL
    Make sure that the only remaining columns are: date, county, and cases.

30. We will be making timeseries plots for the top five counties. Reduce your data as follows:
        TX5 <- TX4 %>% filter(county %in% top5)

31. Create your very first timeseries plot!
        TX5 %>% ggplot(aes(x=date, y=cases, color=county)) + geom_point()

32. Perhaps you don't like the way that looks. Try again, using geom_smooth() instead of geom_point().

33. Use either geom_point() or geom_smooth() as you prefer, but this time, add a title and change the label on the horizontal axis to say "2020". Use:

```
    + labs(title="Your Title Here", x="2020")
```

34. We will now create a plot for these five counties using a logarithmic form of this data.

    ```
    TX5$logs <- log(TX5$cases)
    ```

    Please note that you will get an error if you ask for log(0). However, R allows you take log(NA), giving a result of NA. Thus, all empty data cells in your raw count data will remain as NA in your logarithmic data. That's pretty convenient!

35. Create a graphic using your logarithmic data for the same five counties you've been using. Give your plot a reasonable title, and control the scale label.

36. Please certify that you have typed out the commands in this assignment. (That is, you have not cut/pasted from these instructions).

Once you have completed all the above tasks and they are clearly commented by question number in your .R script, run and re-run your script. One way to do this is to highlight the first three non-commented lines and run them together, to clear out the variable space in R. Then, highlight the remainder of the script and run that together.

If that all works out, submit your .R file (and nothing else) on Canvas.

Again, do not hesitate to contact your instructor if you have difficulties after watching the video.