Kian Afkhami

ENGL 393

Instructions Final Draft

**Reflection:**

This assignment was something I had a lot of motivation to complete, since I was able to choose the main focus and I used it to help me complete a personal project. I spent extra time working on the code because when I originally created this project, I enjoyed working on it a lot and wanted to continue working on it. This assignment was a part of that, as writing instructions is not something I would normally do until the project was complete/nearly complete and there is more I want to add to this project, like vaccination data that did not exist when I first wrote the code in Fall 2020.

The peer review was very helpful for me. The comments Alyssa Comising gave were very helpful in finding small errors, like spelling mistakes, as well as larger issues, like the organization of actions and explanations in the steps(I only had one peer reviewer). What was also useful was reading Alyssa's instructions. We both had similar topics and use some common language (eg: dataframe), so the pairing made sense as both of our instructions were comparable. Both of our instructions had issues organizing the explanations and the actions within the steps. Reading Alyssa's highlighted this for me, and I rewrote/reorganized my instructions so the actions are more distinct. I don't believe I received any feedback from the instructor, if I did get feedback from the instructor, I did not see it and I checked the discussion and the comments/rubric for the first draft.

There were a number of resources I used while writing the instructions. The link included in the first draft assignment( https://pressbooks.bccampus.ca/technicalwriting/chapter/writinginstructions/) was useful, especially when differentiating between different parts of the instructions. I used slightly different syntax (italics and indentation) but it was a good guide for figuring out how to split up parts of the instructions successfully. It was also useful in determining how to organize the instructions. Of course, I constantly consulted with the code to ensure the instructions were as accurate and precise as possible. I also looked at other R guides, to brush up on my R skills and to ensure I was getting the syntax right.

The comments Alyssa gave me were very insightful. She pointed out parts of the instructions that she thought were good, such as the explanations on what the next few steps were going to do. Her pointing these out helped me since I knew to add more of them or expand on them. She also noted that my instructions were a mix of explanations and actions, a problem both of our instructions had, so I made sure to separate those and make it more clear. Also, she pointed out that my audience may be too broad, so I tried to make it more narrow by focusing on two different group, those in a group project and those who would want to use the functions if I ever publicly release this project, which I hope to do in the future.

**Audience Analysis:**

The intended audience for this set of instructions is students/anyone who is familiar with R who would like to create customizable COVID-19 graphs displaying cases, deaths, and case to death ratio data for countries(US state data and vaccination rates coming soon). The original purpose of these instructions was for other students in a group project I was doing to create these graphs. I wrote the code as a part of the project and the other students wrote analysis on graphs created with the code. The instructions gave the other students the ability to create their own graphs without me needing to help, ideally. In the group of about 15 students, I wrote the code and most other students did analysis on the graphs. Instructions would allow them to create their own graphs as effortlessly as possible. I have continued to work on this project since and it has changed and expanded considerably. The instructions explain how to use a set of functions I wrote in R, which create graphs using COVID-19 related data.

Note: Included in my submission is my current version of COVID\_Graphs.rmd. I do not think this is required, but I wanted to include it as part of the final product since the code is not available any other way, it can probably just be ignored if including it was unnecessary.

**How to Create Graphs Using COVID\_Graphs.rmd**

**Introduction:**

COVID\_Graphs.rmd is a file that contains functions which can be used to create case and death graphs. The functions use up to date data from Johns Hopkins University to create graphs in as simple a manner as possible. You can change the title, countries, type of graph, and cumulative/daily data. Coming soon is the ability to use US state data and vaccination data. All of this can be done with just one line of code(per graph).

**Warnings/Cautions:**

The first time the code is run, after COVID\_Graphs.rmd is installed, there may be some libraries that have not been installed. If you see an error code like this when running, you have this issue:

Background pattern

Description automatically generated with low confidence

To fix it, in the console, run *install.packages("****package\_name****")* for every uninstalled package. For example, if you run into the error above, run *install.packages("****ggvis****").* Note that *ggvis* is not used in the project, just as an example here. Replace package\_name with the package you are missing, keeping the quotation marks.

If you see any errors like this, you can ignore them:



**Technical Background:**

Experience with R and RStudio is strongly recommended, although not necessarily requires. If looking for some R basics, here is a good link to learn: http://sthda.com/english/wiki/r-basics-quick-and-easy. You do not necessarily need most/any of these skills to make graphs, but you will need to understand R syntax, as well as how to use RStudio in the most simple ways.

**Materials:**

All that is required is a computer that can run R and RStudio, and both R and RStudio installed on that computer. If you don't have these installed, install them on your own or use these tutorials as guides:

MacOS: https://www.youtube.com/watch?v=0eBD6Jxb2hs

Windows: https://www.youtube.com/watch?v=gkTourWAvX0

**Instructions:**

**1: Install COVID\_Graphs.rmd**

COVID\_Graphs.rmd is a .rmd file that contains a series of functions to create graphs. This is a guide on how to use those functions to make graphs.

Download COVID\_Graphs.rmd (are included with instructions. If I ever made this project public, these instructions and the file would be uploaded to a Github repository). Import it into an existing directory or create a new directory. Open the file and ensure there are no errors. If there are errors, see the Warnings/Cautions section.

Steps 2-5 will show you how to create a single graph using one line of code in this syntax:

function\_name(countries, cumulative, title). Do not try to run the code until all of these steps have been completed.

**2: Choose a graph type**

There are 3 different types of functions, which create 3 different types of graphs:

Cases: create\_cases\_plot()

Deaths: create\_deaths\_plot()

Ratio of Cases to Deaths: create\_ratio\_plot()

Choose one of these and type it into the workspace at line 332.

You should have something that looks like this so far:

Graphical user interface, text, application

Description automatically generated

**3: Create Country List**

The first field of the function is the countries. You can use any of the countries on this list https://github.com/CSSEGISandData/COVID-19 (look at data sources for countries, practically every single country should work). Some countries might have strange names, for example the US is *"us"* and South Korea is *"korea, south".* If you know the exact names for the country/countries you want, skip step 3a.

**3a: Find Country List**

To view a full list of country names, in the console run *view(global\_cases)* after step 6. Step 6 generates *global\_cases*, so it does not exist until that step. If looking for a specific country, jump to step 6a, then come back and run *view(global\_cases)* in the console. This will open a data frame showing all the global case data, listing every country in the region column. If you want to use data for the whole world, use *"total"*.

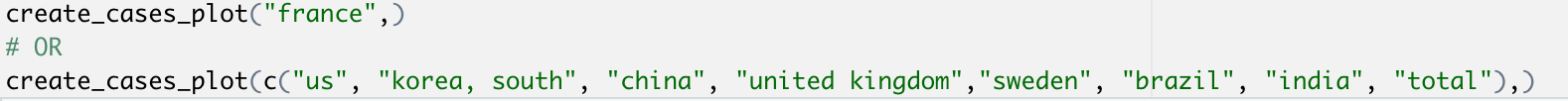
**3b: Make Country List**

If you only want to use one country, in the parenthesis of your function call, put the name of that country(the way it is shown in *global\_cases*) in quotation marks, make sure it is in lower case. Add a comma after that.

If you want to use multiple countries, you have to make a list. In the parenthesis,type *c(* then add whatever country you want in quotes, followed by a comma, then as many countries as you want , then *),* .

It should look something like this *c("name", "name", "name")*, with as many countries as you want. This also work with just one country.

You should have something that looks like this:

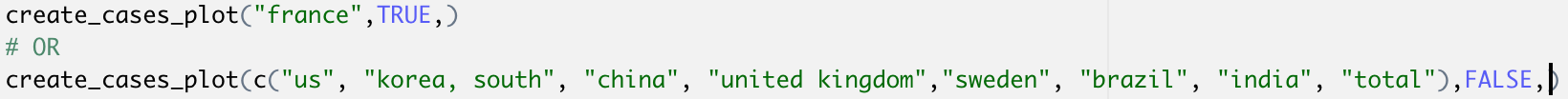


**4: Choose Cumulative**

The data can be presented as daily data(new) or cumulative data(total) for either the cases or deaths graph. If creating a ratio graph, skip this step.

If you want cumulative data, type *TRUE,* in the parenthesis of your function call after countries. If you don't want cumulative, type *FALSE,* . Don't forget the comma after!

You should have something that looks like this:

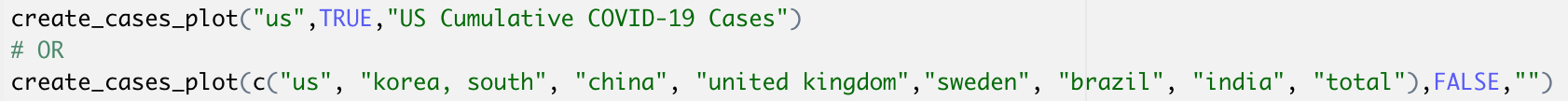


**5: Choose Title**

The final part of your function call is the title.

If you don't want a title, just put *""* after the last comma in your function call. If you want a title, add it after the last comma in your function call, surrounded by quotes. Your function call is now complete!

You should have something that looks like this:



The example plots are now complete! These examples look like this:

Chart, line chart

Description automatically generatedChart, histogram

Description automatically generated

Steps 6-8 show how to add lines and points to graphs in order to accentuate those graphs, for analysis or otherwise. You can skip to Step 9 if you do not want to add these graph features.

To add these features, after the function call for the graph, put a *+* and then add the function for the line/point.

Lines and points will have this syntax: + *function\_name(location, color)*

**6: View Dataframes**

In order to best add points, it is important to view dataframes in order to find points of interest. Dataframes contain the exact data values that are represented on graphs. It is sometimes important to know the exact values so that points and graphs line up perfectly. If you do not need to view dataframes(you already know exactly where to put points/lines), you can skip this step.

**6a: Pull Data and Generate Data Frames:**

The code block from lines 12-297 needs to be run in order to generate dataframes. Run the code in this block. The simplest way to do this is by using the run button.

First, click into the block, as if you were going to edit code within that code block.

At the top right of the code, you should see a run button. Click on *Run*, then select *Run Current Chunk*.

Graphical user interface, text, application

Description automatically generated

Alternatively, you can click into the code chunk, then use the keyboard shortcut *Command + Shift + Enter* on Mac or *Control + Shift + Enter* on Windows to run that chunk.

After you run the code, you should see a green bar appear on the left side of your screen. Once that has disappeared, the entire code block has been run and you should be able to open dataframe.

The green bar looks like this:

Graphical user interface, text, application

Description automatically generated

**6b: Choose a Dataframe**

Each dataset has a different dataframe. Here is a key of all the dataframes. Ther daily dataframes contain the points on the graph, while the weekly dataframes contain the weeky average, which is displayed as a line on the graph. Identify which dataframe you would like to open, the titles are self-explanatory:

*global\_cases*

*global\_deaths*

*global\_ratio*

To view a list of all countries, use any of the dataframes and look at the *regions* column.

Note: There are more than these dataframes, but they are not directly relevant to finding datapoints, although you may access them.

**6c: Open and Understand a Dataframe**

To open a dataframe, in the console(at the bottom of the screen) type *view(dataframe\_name)* and press enter. This should open a dataframe in a new tab.

A dataframe should look like this:

Table

Description automatically generated

The region column contains a list of all countries, the other columns show the number of new cases each day. Here you can find values for whatever value you are looking for to create a line or point with.

**7: Add lines**

Both horizontal and vertical lines can be added. You can add lines wherever you want, you can add as many lines as you want, or none at all.

Horizontal: + geom\_hline(yintercept = y, color = "color")Replace y with a y value and color in quotes with the name of a color.

Vertical: + geom\_vline(xintercept = as.Date("YYYY-MM-DD"), color = "color")

Replace YYYY-MM-DD with a date in that notation and color in quotes with the name of a color.

Add these statements after your function call to create your graph, like this:



These examples look like this:

Chart, line chart

Description automatically generatedChart, histogram

Description automatically generated

**8: Add points**

Points work similarly to lines, although they have different inputs. Points can be added wherever you want, however many you want.

The function call for a point looks like this:

+ geom\_point(aes(x= as.Date("YYYY-MM-DD"), y=y), color="color")

Replace YYYY-MM-DD with a date in that notation, y(after y=) with a y value, and color in quotes with the name of a color.

If you want to add a point, it should look something like this:



This example looks like this:

Chart, line chart

Description automatically generated

**9: Run Code/Generate Graphs**

In order to generate the graphs, two chunks of code need to be run.

First, run the chunk of code from line 12-297.

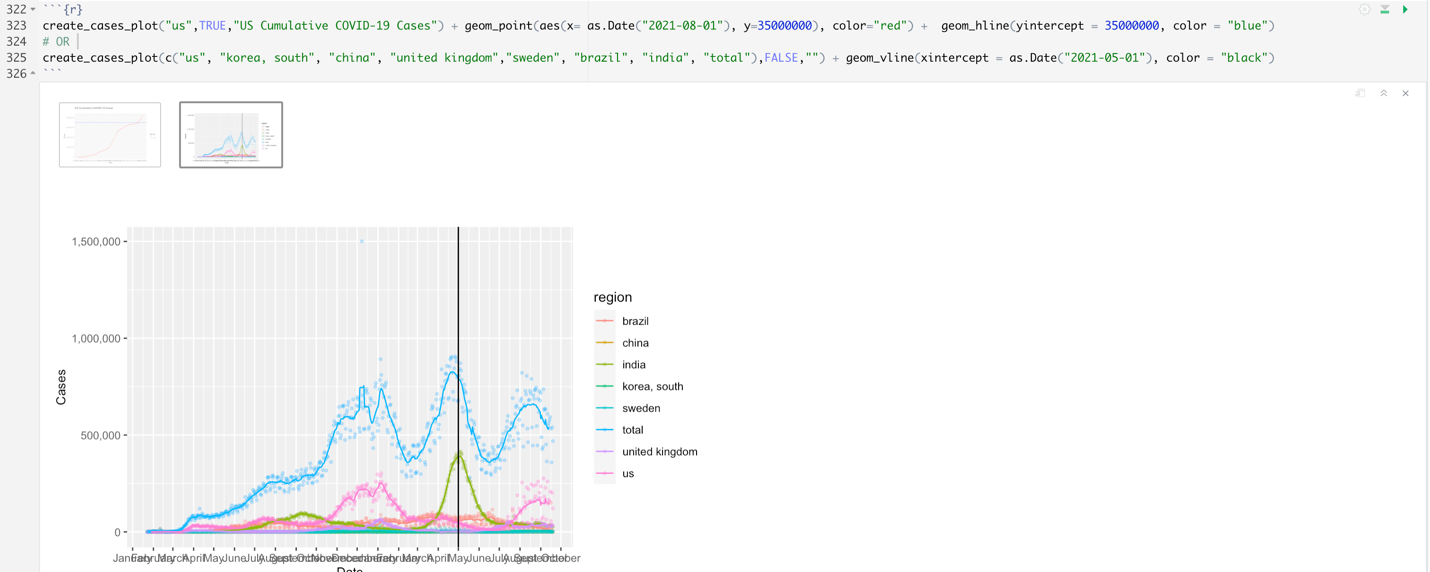
This fetches the data and generates the dataframes. This needs to be done every time you want data to be updated and the first time the code is run. For example, if you run this code block now, then wait 1 month and run the lines that contain the function calls, the data would be one month old. So, run this block every time you want to gather new data and if you ran it more than 1 day ago.

If you don't know how to run a code block, go to step 6a, then come back.

Now that you have done that, the code chunk at line 322 with your function calls needs to be run. This will generate the graphs. You can also make more blocks of code and run those if you desire. After you run code blocks, your graph/graphs should appear.

**10: Viewing/Saving Graphs**

You should now have something that looks like this:



If there were multiple graphs created in your code chunk, there will be thumbnails of the graphs, with one displayed at a time while in RStudio. If you are famliar with RStudio and would like to knit to HTML, you can do that now(don't worry about this if you don't know what it means). You can click on the thumbnail to view the other graph/graphs. If you want to export the graph/graphs, there are multiple ways to do it.

a: You could use a screenshot application to take a screenshot of the graph.

b: Right click on the graph and select save image and save it to your computer.

If you would like to change the aspect ratio/resolution of the graph, select the new window icon, shown in the top right of the image below next to the X and the two upward arrows. This will open the graphs in a new window, where they can be resized to your liking. Then use a or b to save the graph.

Graphical user interface, application

Description automatically generated

Chart, histogram

Description automatically generated

As you can see, the graph is higher resolution and in an aspect ratio I changed by resizing the window.

You are finished!

You can now create graphs using COVID\_Graphs.rmd

I have used this program throughout the pandemic, as it was the easiest way for me to view COVID data and it was extremely useful. I hope this provides some use for you as well.