Covid19 Tracking Dashboard

Tyler Jubenville

May 12, 2020

Purpose

The goal of this shiny app is to display real time information on the current prevalence of Covid19 worldwide.

Availability

The app is hosted through shinyapps.io and can be found here: https://tajubenv.shinyapps.io/CovidDashboard/.

The app may also be run by pulling it from the University of Minnesota github, which can be found here: https://github.umn.edu/juben002/PUBH7462_Shiny_App

Data Source

The data for this project is pulled from the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE) github: https://github.com/CSSEGISandData/COVID-19

App functions:

Covid19 Map

The map in this app will show the total number of confirmed cases, deaths, and recovered cases on a world map. The user may select which type of case they want to view using a drop down menu on the left. The user may move around the map using a mouse and the zoom buttons. Mousing over a specific circle will display the total number of cases within that country.

Any Countries Cases Graph

All countries within the data will be available for the user to select. By default the 5 countries with the greatest number of confirmed cases are selected. The user may select which countries they would like to appear on a cumulative graph. The cumulative graph will plot either confirmed cases, confirmed deaths, or recovered cases, depending on the user's selection.

Date Slider

In addition to the two options above, the app includes a date slider, so the user may examine how the app looked at any point between the first confirmed case and the current date. The date slider will affect both the map and the graph.

Programming challenges:

Combining the Data

The main challenge here was getting the data to reformat properly and then merge it all together. I had some initial difficulty with this, but using the relatively new pivot_longer function made this much easier. The data source also changed slightly while I was developing the app, which added some confusion, as it updating while I was debugging the app.

In addition, the methodology of the app changed from taking the raw data to aggregating by country. As each data point originally included the coordinates at the state or province level, it was necessary to instead generate new latitude and longitude data for each country. The rgeos package was used to calculate the centroids for each country, which were then used as the final coordinates.

Date Slider

I had some issues getting the date slider to work correctly with dynamic dates. This is actually how I learned that you can declare UI components within the server section, and then set the values dynamically. Once I got that section working correctly, my leaflet map became very slow to respond to changes in the slider. This was because I had incorrectly set up my leaflet call and was reinitializing the entire map each time the date input was changed. This led me to correctly initializing the map once and manually clearing the markers and drawing new markers at each input change, which drastically improved the overall responsiveness.

PickerInput from shinyWidgets

This is not really a programming challenge, but more of an easy solution to a difficult problem. The "pickerInput" function from the shinyWidgets package gives a combination of checkboxes and a drop down menu that makes it easy to display all countries for this project. It makes it very simple for the user to search for what they need and was extremely easy to implement on the programming side.

Map Displaying Outside of Limits

The final coding challenge I had was limiting the leaflet plot so it would not display the world map multiple times over. This involved setting a default zoom value that would properly display about one set of the world. By default, leaflet allows the user to scroll out of the main area where circle markers are drawn. Addressing this required two changes to the initial leaflet call. The first is specifying the minZoom and maxZoom within the providerTileOptions. The second is using the setMaxBounds function to limit the area where the user may drag. If the user drags outside of the intended view area, the map forces the user back to the desired frame. The updated initialization code is shown below:

```
leaflet() %>%
       addTiles(options = providerTileOptions(minZoom = 1.3,
                                              maxZoom = 10)) %>%
       setView(lng = 0, lat = 0, zoom = 2) %>%
```

```
      setMaxBounds(lng1 = -180, lng2 = 180,
                   lat1 = -90, lat2 = 90)
```

Division of labor

I completed this project alone, so overall there was very little division of labor.

Future work:

There are several features I would implement within this app with more time. The first would be to include population data for all of the countries included. I would then add selection for either per capita analysis or raw number analysis.

For the cumulative cases graph, I would like to add the ability to shift the dates of each country individually, so users could overlay countries from when they had their first cases in order to compare the response by country.

Finally, I would like to do a more thorough cleaning of the data and get as much of the analysis to a province/state or even county level if possible. Right now it is all at the country level, so zooming in further on the map has a limited usefulness.