



# F20DV Lab 3 Report

DEMONSTRATED TO AMIT PAREKH ON: 18/03/2022

CAMERON DOUGLAS – H00295794

## Features of COVID-19 Dashboard

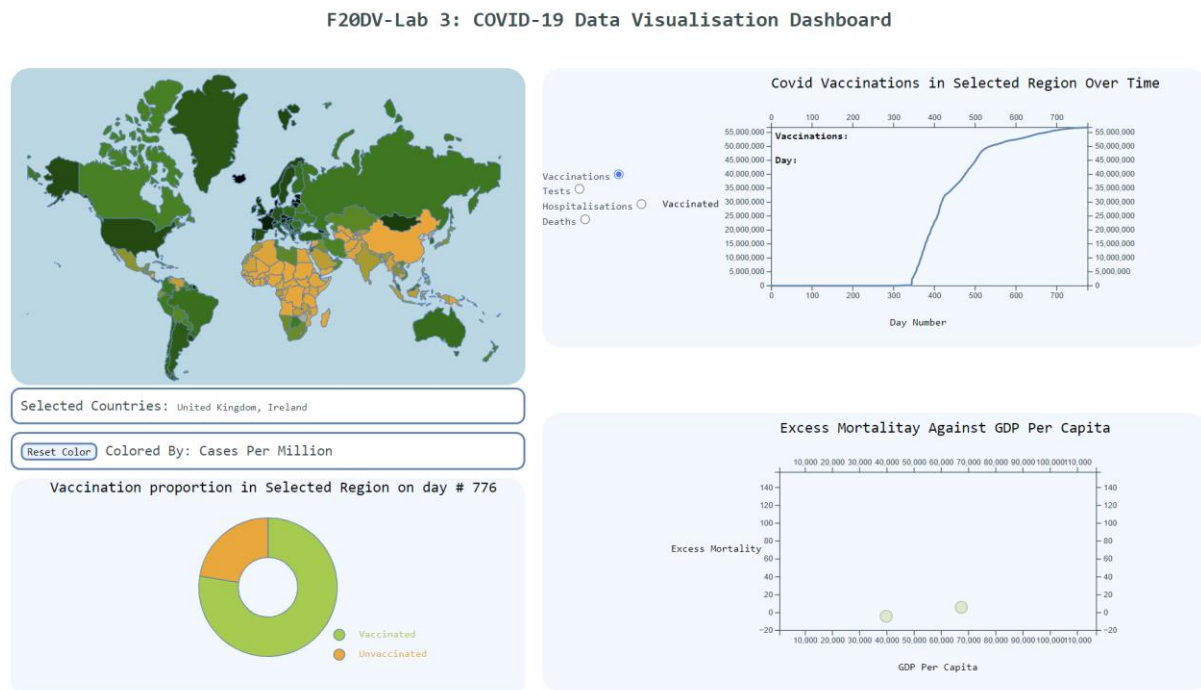


Figure 1 - COVID Data Dashboard

For my dashboard, I first created a map which would be used to select regions. I downloaded a geoJSON of the globe from <https://geojson-maps.ash.ms/>, which contains information and spatial data about each country.

```
"features": [
  { "type": "Feature", "properties": { "ADMIN": "Aruba", "ISO_A3": "ABW" },
    "geometry": { "type": "Polygon", "coordinates": [ [ [ -69.996937628999916,
      12.577582098000036 ], [ -69.936390753999945, 12.531724351000051 ], [ -
      69.924672003999945, 12.519232489000046 ], [ -69.915760870999918,
      12.497015692000076 ], [ -69.880197719999842, 12.453558661000045 ], [ -
      69.876820441999939, 12.427394924000097 ], [ -69.888091600999928,
      12.417669989000046 ], [ -69.908802863999938, 12.417792059000107 ], [ -
      69.930531378999888, 12.425970770000035 ], [ -69.945139126999919,
      12.440375067000009 ], [ -69.924672003999945, 12.440375067000009 ], [ -
      69.924672003999945, 12.447211005000014 ], [ -69.958566860999923,
      12.463202216000099 ], [ -70.027658657999922, 12.522935289000088 ], [ -
      70.048085089999887, 12.531154690000079 ], [ -70.058094855999883,
      12.537176825000088 ], [ -70.062408006999874, 12.546820380000057 ], [ -
      70.060373501999948, 12.556952216000113 ], [ -70.051096157999893,
      12.574042059000064 ], [ -70.048736131999931, 12.583726304000024 ], [ -
      70.052642381999931, 12.600002346000053 ], [ -70.059641079999921,
      12.614243882000054 ], [ -70.061105923999975, 12.625392971000068 ], [ -
      70.048736131999931, 12.632147528000104 ], [ -70.007150844999987,
      12.58551666900001 ], [ -69.996937628999916, 12.577582098000036 ] ] ] } } ...
```

Figure 2 - Example geoJSON data

In my file *map.js*, I use d3's built in projections and path generators to draw the graph, given the data loaded in *core.js* using d3's JSON function. In the function *update()* I draw the paths to the SVG, giving each path an ID containing it's relevant ISO code, which will allow me to identify countries by which paths are selected. I also set the colour of the path based on values obtained from the covid dataset. The initial colouring is based on cases per million (green showing low cases and orange showing high). This colouring indicates how geographical positioning affected COVID-19: countries in Africa and East Asia were disproportionately affected when compared with the rest of the world.

I implemented brushing as a means to interact with the map, the desired region is "brushed" and on mouse release the charts update with the specified data is accumulated from each selected country.

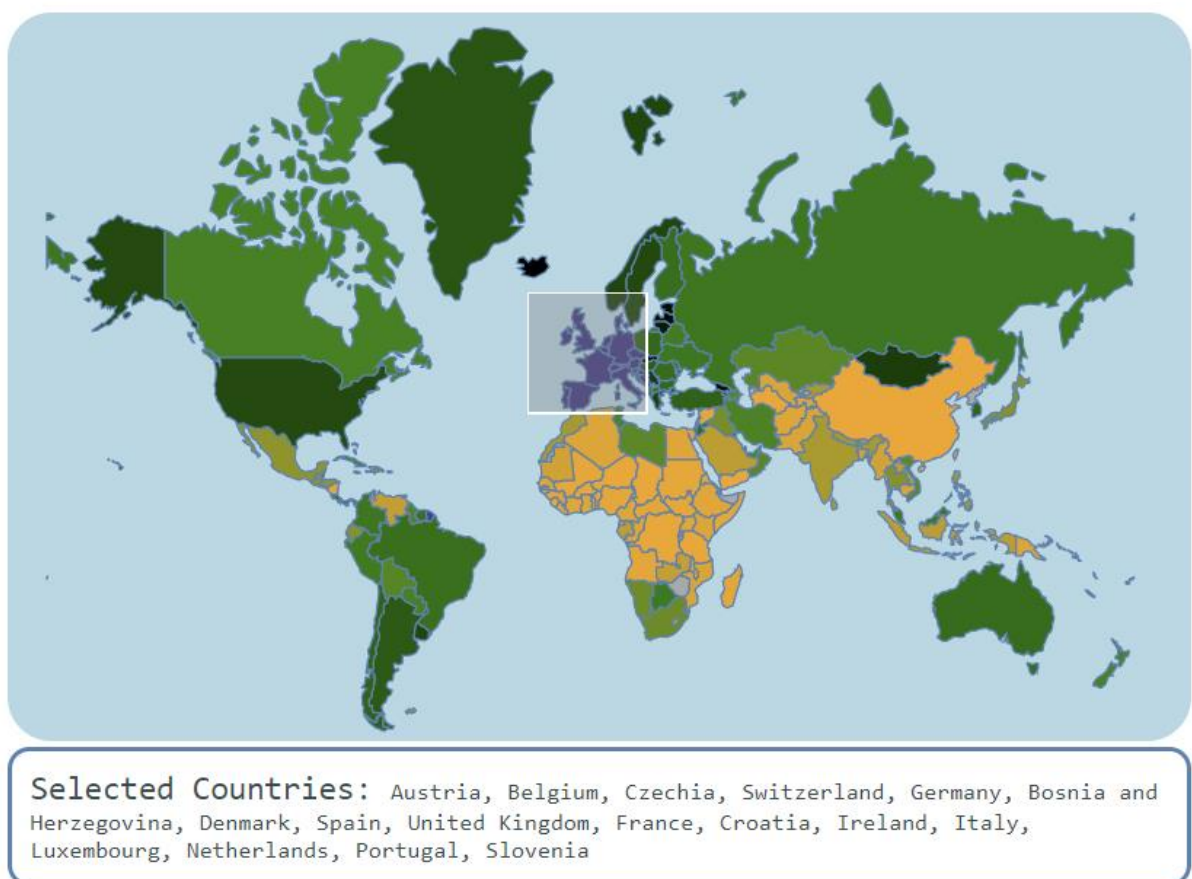


Figure 3 - Demonstrating brushing

There are three charts on the dashboard: a line chart, a pie chart and a scatter plot. The line chart shows covid related data for the selected region over time, the specific data can be selected using the radio buttons next to the chart:

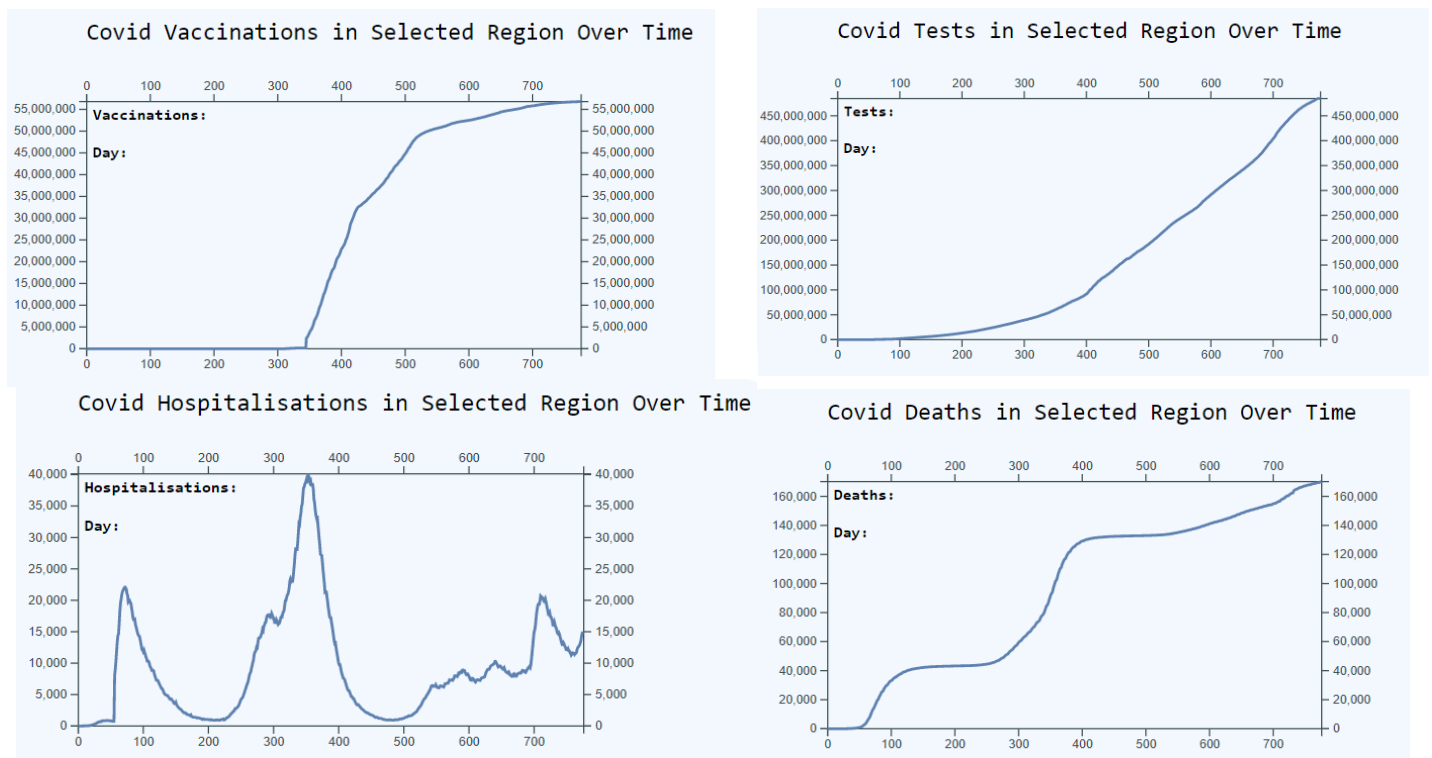


Figure 4 - Line charts showing COVID data

On mouseover, a marker will display on the pointer to show the specific day being selected, and the counters with the specific values in the top left corner update.

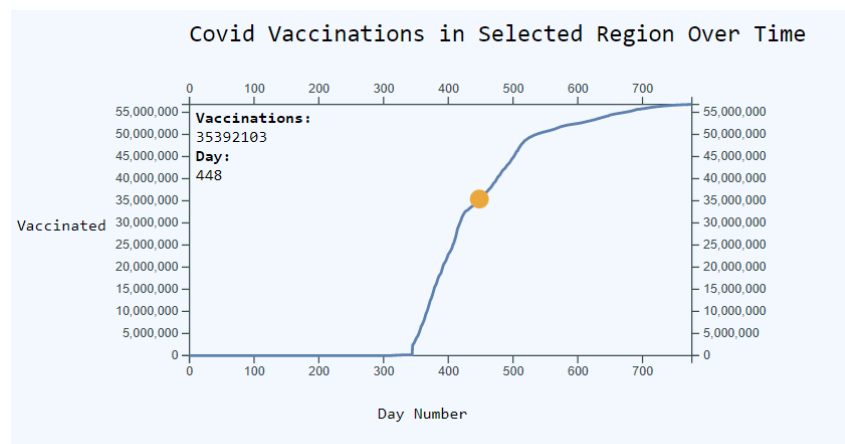
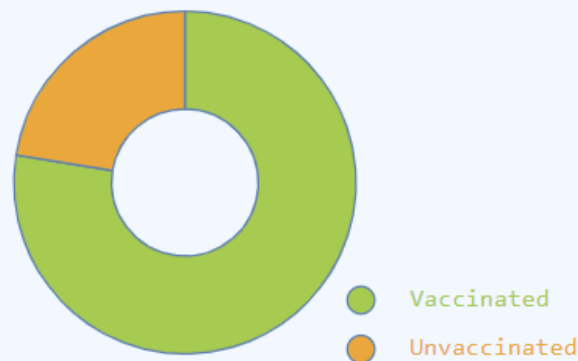


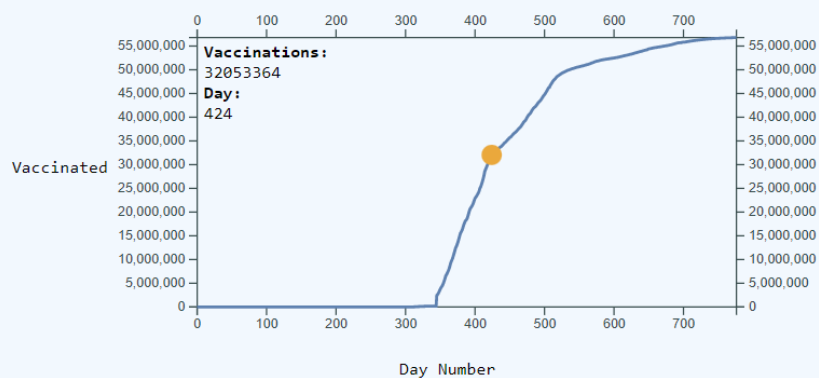
Figure 5 - Mouseover effect on chart

The Pie chart shows the proportion of vaccinated and unvaccinated people in the region on a given day. When looking at the pie chart with the graph of deaths and the map showing cases, we can see how vaccines affect the impact of COVID. Initially the pie chart will show the data for the last recorded day in the region, however this can be updated by clicking on a particular day on the line chart.

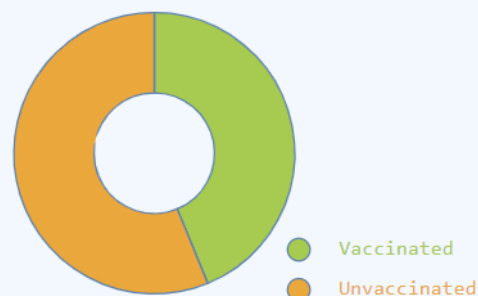
Vaccination proportion in Selected Region on day # 776



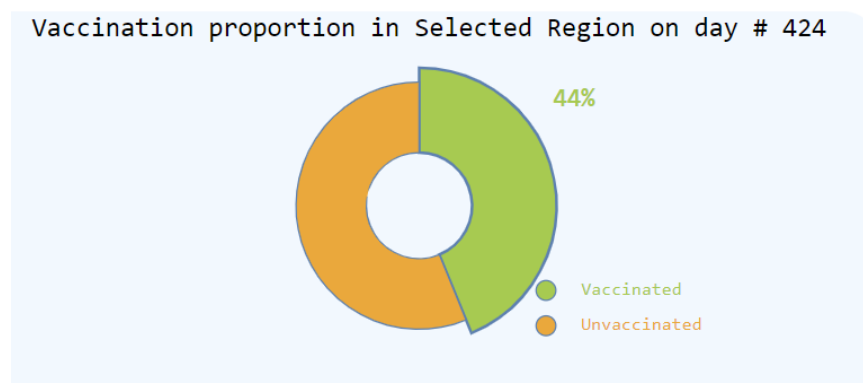
Covid Vaccinations in Selected Region Over Time



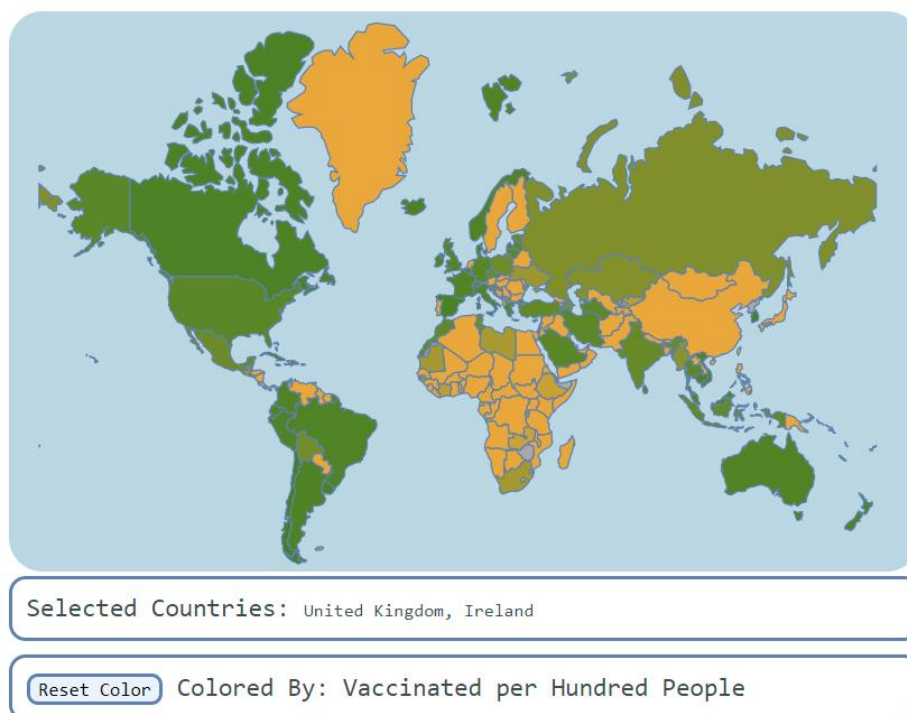
Vaccination proportion in Selected Region on day # 424



The segments of the pie chart can be moused over to reveal the specific value of the proportion:

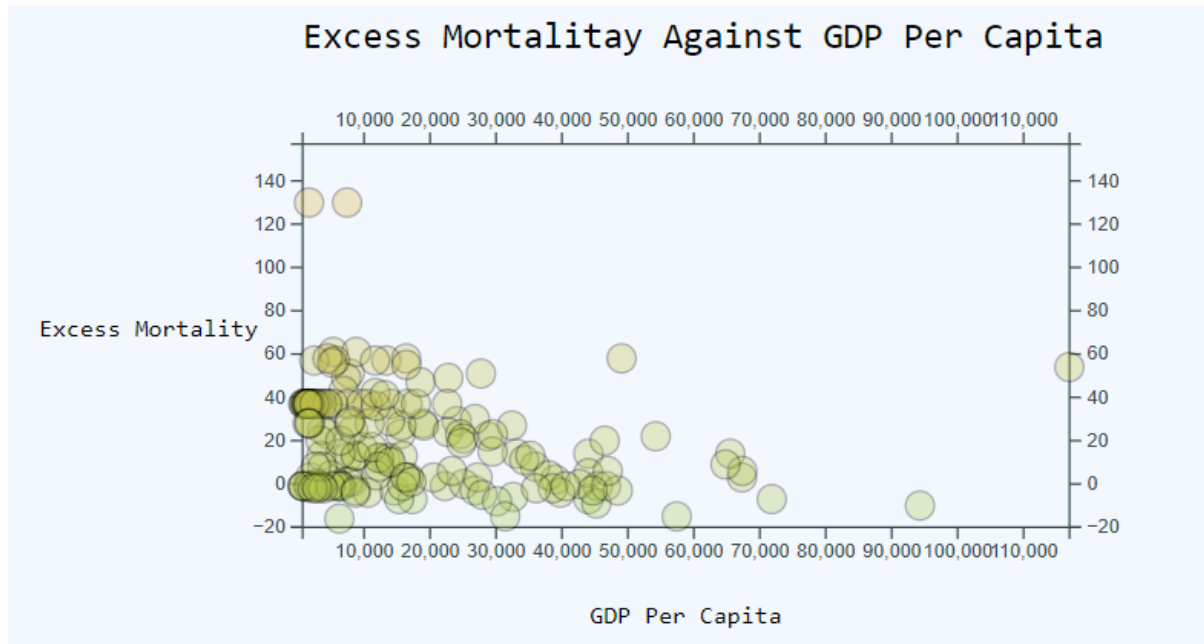


The segment can then be clicked to change the colour scale of the map from cases per million, to vaccinated per hundred:



This colouring is representative of the day that is selected, this can be updated by once again clicking the datapoints of the graph. This interaction shows on the map exactly how each country's vaccination progression occurred. This is how my dashboard narrates the evolution of the world's response to COVID-19.

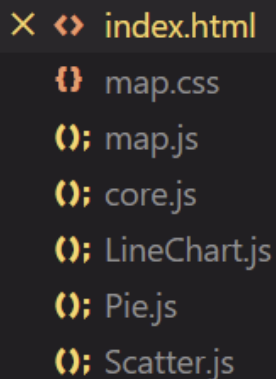
My final plot is a scatter plot, this plot shows a country's excess mortality against its GDP per capita. This gives a very strong inclination of the relation between wealth and effectiveness against COVID.



This graph, adds and removes datapoints based on the selected countries, this is the resultant plot of selecting the whole world and shows the general trend that richer countries have a lower excess mortality due to COVID 19.

*Note:* Some countries seem to have an exceptionally low excess mortality, -20% is the lowest value indicated on the axes which is surprisingly low, even for normal times, let alone during COVID. This will almost certainly be an error in the reported data.

## Code Structure



```
× ↺ index.html
  {} map.css
  {} map.js
  {} core.js
  {} LineChart.js
  {} Pie.js
  {} Scatter.js
```

For ease of readability, I split my code up into 5 separate JavaScript files: Core.js, Map.js, Pie.js, LineChart.js and Scatter.js. This helped me keep my code clean and made it easier to add and remove functionality without majorly affecting other aspects of the project.

In Core.js, both the COVID and geoJSON datasets are loaded, once they have been loaded, I call the *update()* function in Map.js and draw the map as described previously. Core.js then begin building datasets. The first thing I do once the COVID data is loaded is build a map which maps from iso\_code, to a list of lists, each sub-list is a row from the COVID dataset. Using this structure, I can return all of the data for each country by simply calling, *worldData.get(iso\_code)*. Once the data is loaded, I create an initial list of selected countries and call the *multiCountry()* function. The first call, has the parameters:

```
multiCountry(['GBR','IRL'], "Vaccinations", "max", true)
```

The first parameter is the list of countries (in this case the UK and Ireland). The second, indicates what data I want to be built (Vaccination). The third specifies a day, for this example, I have passed the string “max” which in the function, I interpret to be the last recorded day in the dataset (i.e. *worldData.get(iso\_code).length - 1*). The final is a boolean which indicates whether new countries have been selected, this enables me to update the list of currently selected countries if needed, otherwise the list is unchanged.

Within *multiCountry()* I then pass each iso code one at a time to helper functions which then build an accumulated dataset which is then passed to *updateChart()*, *initialisePie()* and *updateScatterChart()*. These functions are contained within each of the chart’s respective JavaScript file and take the data and draw the respective visualisations.

Any time a new selection is made or a new date is selected, *multiCountry()* is called to update the datasets and then redraw the visualisations. *Note: All the charts are drawn in the same manner as demonstrated in previous lab exercises so I won’t describe how this is done here.*