F20DV Data Visualisation

Lab 3 Report

18/03/22

Callum Taylor

Demonstrated by

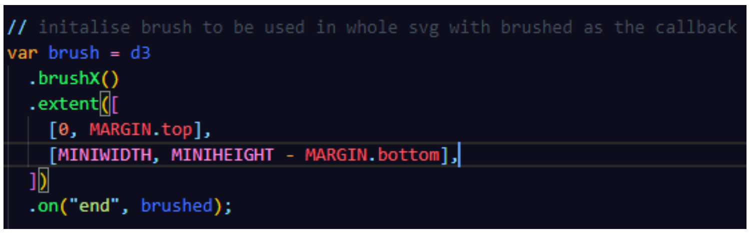
Shuangjiang Xue – 18/03/22

Github Repo - https://github.com/taybluetooth/f21dv-lab-3

Table of Contents

Cross Layout Brushing

Brushing in data visualisations can be extremely useful in very dense charts containing large quantities of data. This can be achieved quite simply in d3.js by using the d3.brush method. This is done by declaring the brush and the extent it can be used, normally being set as the width and height of a chart. An event handler is then attached to this brush which has a callback function, which is fired after finishing a selection. This can be seen in the code below.



```
// initalise brush to be used in whole svg with brushed as the callback
var brush = d3
  .brushX()
  .extent([
    [0, MARGIN.top],
    [MINIWIDTH, MINIHEIGHT - MARGIN.bottom],
  ])
  .on("end", brushed);
```

Figure 1 d3.js brush method.

I added this feature to be used between the new cases chart and a navigator which uses the brushing. A user selects a section of the navigator and this is then updated in the new cases chart. The image below shows an example chart of Afghanistan before any brushing.

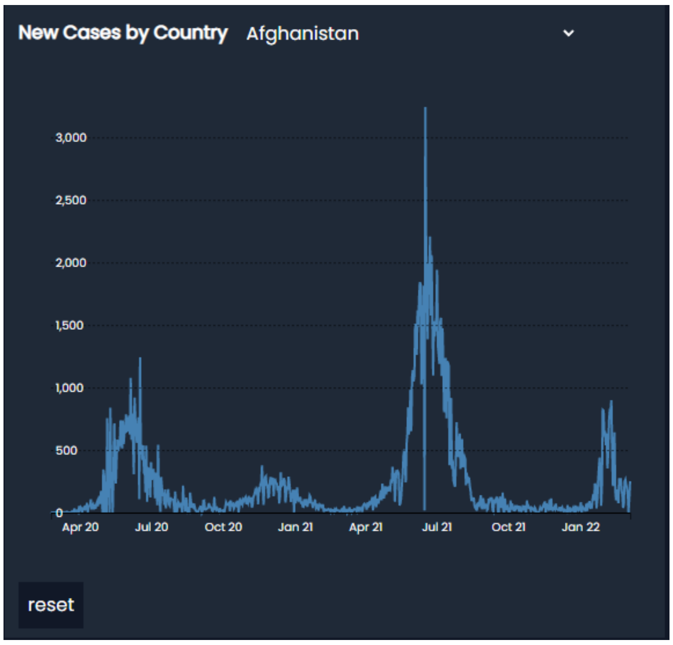


Figure 2 New cases in Afghanistan.

The user then selects a section of the chart by dragging a rectangle over the navigator. I used the brushX variant of the brush method which only allows dragging of the X axis. This was to ensure no data was missed on the Y axis.

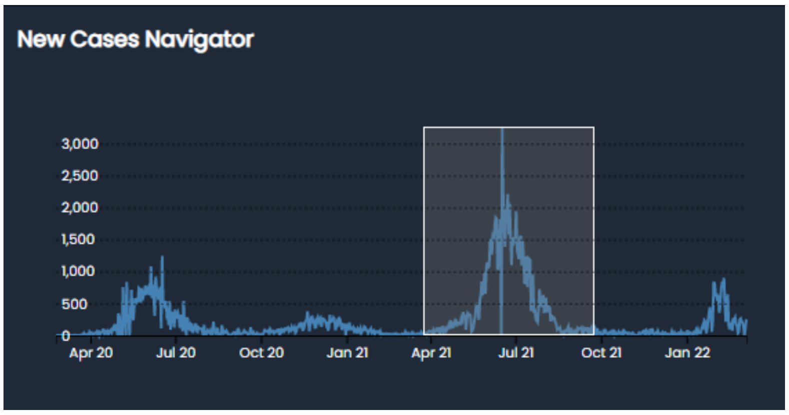


Figure 3 Navigator chart brushing.

The cases chart then transitions to its new updated state as seen below. The user can reset this if they wish by pressing the reset button, which resets the chart to its original state.

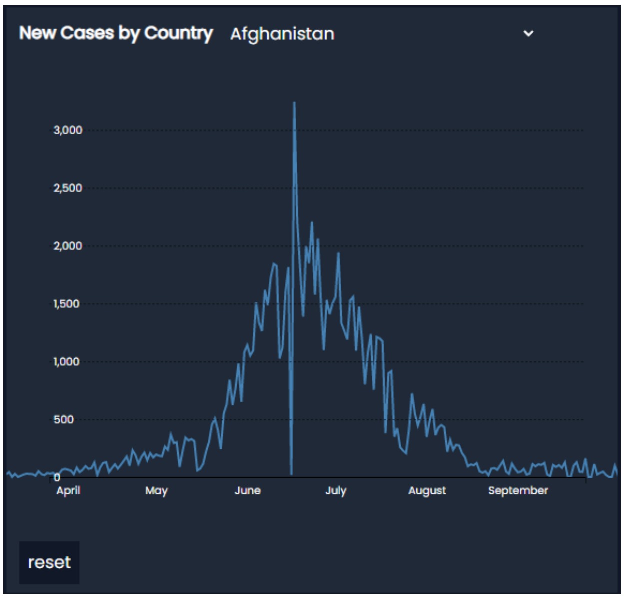


Figure 4 Updated cases for Afghanistan between April and September.

Bidirectional Interaction

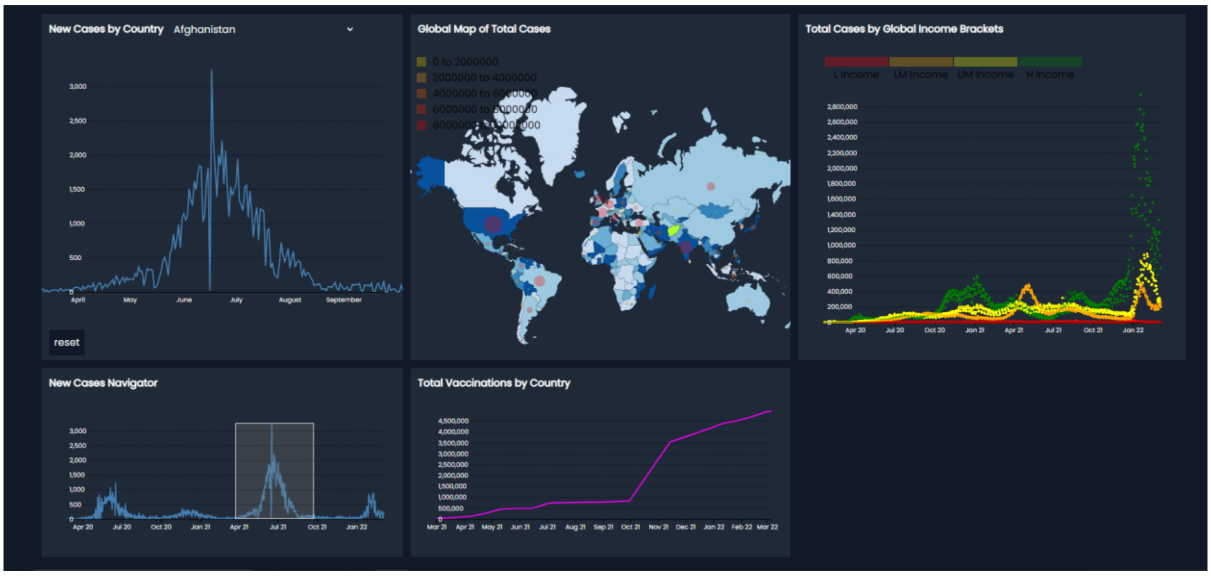


Figure 5 View of the dashboard.

Within the dashboard, there are 5 panels that are all linked with each other, and all (except the income chart) simultaneously update upon interaction with a singular chart. The user can use the drop-down menu seen in the new cases chart to update which country/continent/income bracket wishes to be viewed. This will then update all other charts.

Similarly, the world map feature can be zoomed and panned around, if the user wishes to select a country, they simply need to click on it, this will then update the dropdown showing which country is selected and updates the other charts. The country changes to a bright lime green colour to make it clear on the map as to which one is selected.

The user can also hover over one of the income bracket dots in the scatter graph which then updates the charts based on which income bracket has been selected. In a future update, I would like to add this hover feature to the legend I made at the top of the graph for ease of access.

World Map Generation

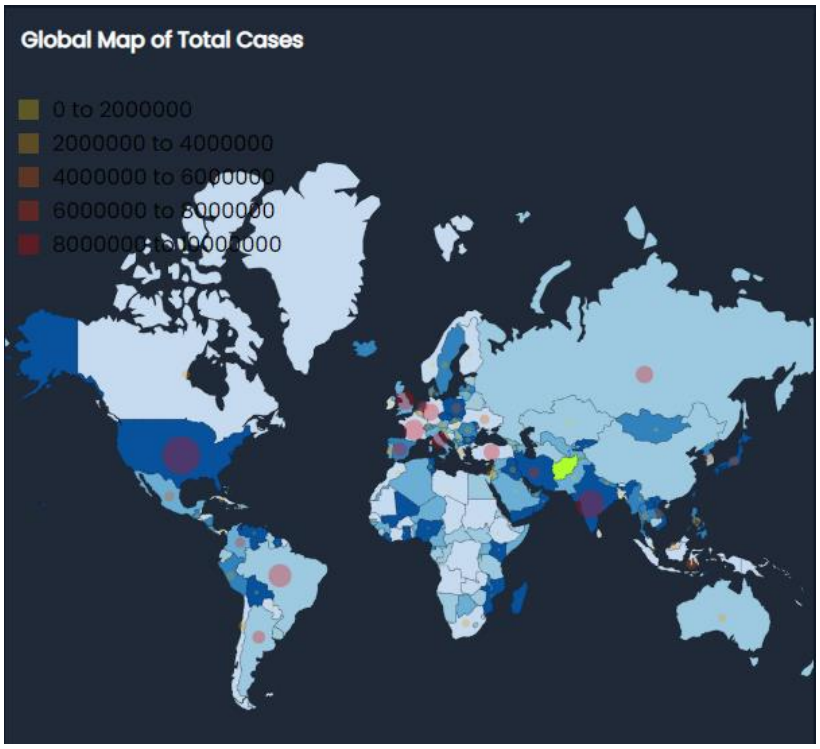


Figure 6 World map view.

The world map was created by using a combination of data. The countries were generated using the d3 geomercator function which generates a world map based on the Mercator projection.

The circles indicating the cases were generated by gathering the latitude and longitude coordinates of each country's central point. The circles' positions were then mapped to these points and the cases were then used to scale each country's circle based on size and colour, with larger and redder indicating higher cases. This is easily visible in the legend in the top left of the world map.

The map can be panned and zoomed in using the mouse, and each country can be selected by clicking on it.

In future I would like to add a small overlay when you hover a country which indicates which country the user has selected as well as the exact number of cases/deaths etc.

Lab Future Plans

In future, I would like to add several more features to the dashboard.

For starters, all dashboards should be fully mobile responsive and function as intended on mobile devices/tablets. This would involve using dynamic media queries and adding touch instead of clicking events to the panels to ensure the user could interact properly. This would best be achieved using a framework such as React JS which is great for web app development due to being responsive first.

Another feature that would be nice to add is the changing of colour schemes to improve accessibility. The dark scheme in my opinion works well and should be fine for most eyes. However in the event it isn’t, there should be some sort of toggle to change the colours and background used.