F21DV Lab 3 Report

Demonstrated to: Benjamin Kenwright

On date: 25/03/2022

1. Tech Used

The application is written using a combination of HTML, CSS and JavaScript using version 7 of the D3 library for data visualisation. To render the map I have used the Mapbox API and used D3 to visualise data on the map; this will be described in more detail in the Map section. The d3-simple-slider library is used to create the slider (https://github.com/johnwalley/d3-simple-slider).

2. Data Processing

I have used the prescribed COVID-19 dataset provided by 'Our World in Data'. I initially read in the data from a csv file stored in the same directory as the main HTML file for the dashboard using the d3.csv('file_name') function.

```
d3.csv('owid-covid-data.csv').then( function(data) {
```

Once the function has read in the data as an array of objects it passes the result to the .then(function(data) { function where it is used in the creation of the dashboard.

2.1 Grouping by month

My initial idea was to display on the map the number of deaths each month for each country, using circles where the radius represented the number of deaths. In order to do this, I needed to sort the unordered dataset by month. To do this I initialised an empty object "monthGroup". I then use a for loop to cycle through all objects in the data array, take their date property and convert it o a date object. From the date object I take the month and year and combine them into a string with the format "0 - 2020" for January 2020. This string is then used to look up the index of that month in the "months" object created before reading the data in. This object maps the "0 - 2020" to a human readable format and gives it an index so that the months can be stored ad displayed in order along the slider.

```
var months = { "0 - 2020": {format: "Jan - 2020", index: 0},
               "1 - 2020": {format: "Feb - 2020", index: 1},
               "2 - 2020": {format: "Mar - 2020", index: 2},
               "3 - 2020": {format: "Apr - 2020", index: 3},
               "4 - 2020": {format: "May - 2020", index: 4},
               "5 - 2020": {format: "Jun - 2020", index: 5},
               "6 - 2020": {format: "Jul - 2020", index: 6},
               "7 - 2020": {format: "Aug - 2020", index: 7},
               "8 - 2020": {format: "Sep - 2020", index: 8},
               "9 - 2020": {format: "Oct - 2020", index: 9},
               "10 - 2020": {format: "Nov - 2020", index: 10},
               "11 - 2020": {format: "Dec - 2020", index: 11},
               "0 - 2021": {format: "Jan - 2021", index: 12},
               "1 - 2021": {format: "Feb - 2021", index: 13},
               "2 - 2021": {format: "Mar - 2021", index: 14},
               "3 - 2021": {format: "Apr - 2021", index: 15},
               "4 - 2021": {format: "May - 2021", index: 16},
               "5 - 2021": {format: "Jun - 2021", index: 17},
               "6 - 2021": {format: "Jul - 2021", index: 18},
               "7 - 2021": {format: "Aug - 2021", index: 19},
               "8 - 2021": {format: "Sep - 2021", index: 20},
               "9 - 2021": {format: "Oct - 2021", index: 21},
               "10 - 2021": {format: "Nov - 2021", index: 22},
               "11 - 2021": {format: "Dec - 2021", index: 23},
               "0 - 2022": {format: "Jan - 2021", index: 24},
               "1 - 2022": {format: "Feb - 2022", index: 25},
               "2 - 2022": {format: "Mar - 2022", index: 26}
             }
```

This index returned is then used as a key to create an object property in “monthGroup”, and the value is set to be an empty array. This creates a list of all the months in “monthGroup” and the values are empty arrays.

```
// For loop to create a key value pair in the monthGroup object
// for each month in the data where the value is an empty array.
for(i in data) {
    var date = new Date(data[i].date)
    var keyTemp = date.getUTCMonth() + " - " + date.getUTCFullYear()
    var key = months[keyTemp];
    if(!isNaN(date)) {
        monthGroup[key.index] = [];
        monthDates[key] = new Date(date.getUTCFullYear(), date.getUTCMonth());
    }
}
```

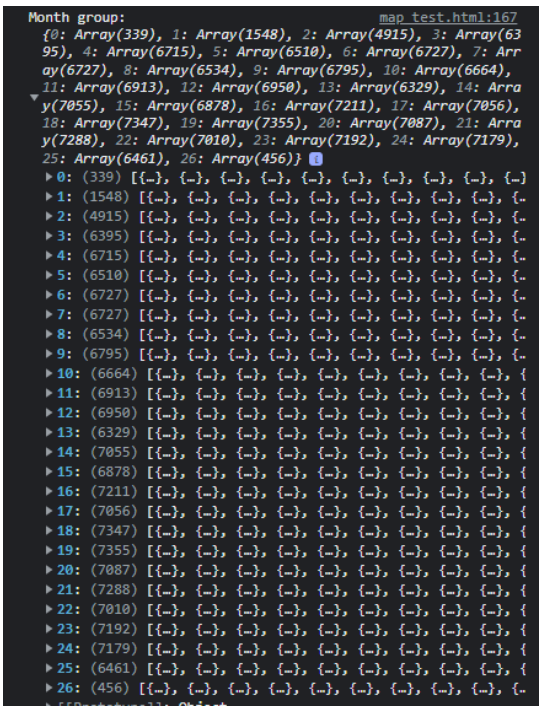
Function to create an object with an empty array for each month in the dataset

The next step is to populate the object with data corresponding to the correct month. I do this by using the same for loop but this time pushing the object into the array corresponding to the correct date.

```
// For loop to populate the empty arrays in monthGroup with the
// data objects from the corresponding month
for(i in data) {
    var date = new Date(data[i].date)
    var keyTemp = date.getUTCMonth() + " - " + date.getUTCFullYear()
    var key = months[keyTemp];
    if(!isNaN(date)) {
        monthGroup[key.index].push(data[i]);
    }
}
```

Function to group data objects by month

The result of this is an object where the keys are all the months the data covers and the values are arrays of all the data objects that correspond to that month.



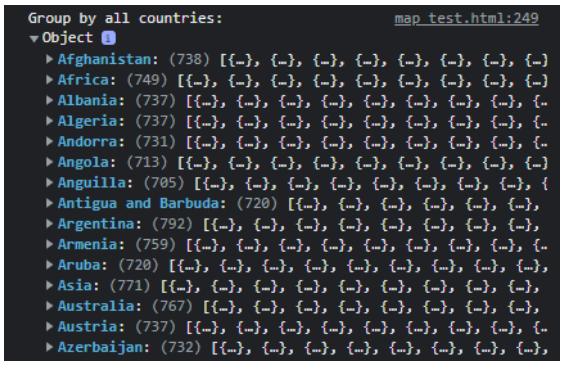
2.2 Grouping by country

As the data will be shown by country, I created a function that will group the data given as an argument by country. Using the Array.prototype.reduce() function I loop through each item in the dataset. In the first iteration, an empty array is created and the array element is pushed to the empty array and a key value pair is created in "gourps" where the key is the country name of that array object. In the second and subsequent iterations, the "groups" object is passed in to the callback and either a new key:value pair is created for the new array object in that iteration or the array object is added to an existing array in the "groups" object if that country has already been encountered in the loop. The result is an object where the keys are the country names and the values are arrays of all the data objects corresponding to that country from the input data.

```
// Function to create an object where data objects are stored
// by country.
function groupByCountry(data) {
    const newGroups = data.reduce(function(groups, item) {
    const group = (groups[item.location] || []);
    group.push(item);
    groups[item.location] = group;
    return groups;
    }, {});

    return newGroups;
}
```

Function to group data by country in an object

This is used in the dashboard for the bar chart where all the data is grouped by country at once and in the map where a particular month of data is passed in and sorted by country.



2.3 Coordinate data

For simplicity and speed, the data for each countries coordinates is stored as a data object in a separate JavaScript file ("newCoords.js") and linked in the <head> section of the document. To get the coordinates, I used the Mapbox API forward Geocoding functionality which accepts a location name and returns its coordinates. I used the groupByCountry function to create a list of all countries and then iterated through the array, calling the API at each iteration to the return the coordinates. This took a while an interfered with the visualisations and so I felt it was easier to run it once and store the result.

```
coordinates =
    {"Afghanistan": {
        "long": "66.0264709817607",
        "lat": "33.8388058603423"
    },
    "Albania": {
        "long": "20.1566908111252",
        "lat": "40.6410895558590"
    },
```

Coordinate dataset

2.4 Preparing data for visualisation

Once the data is grouped by country, the final data transformation takes place before being used in visualisation. The final data objects used by each visualisation contain the country name, the total number of deaths for that country in the input data (could be for a month or for all data), the number of people vaccinated, the country population, the gdp per capita, and the longitude and latitude of the country.

To do this, I use a for loop to loop through all of the countries in the input data object and then use a nested for loop to loop through all the data objects in that countries data array of the key:value pair. For each data object, the total deaths variable is incremented by the number of “new_deaths” (if not null) in each data object. Then, if the country name is in the array of 16 countries being used “countrySelection” and object is created with all the values listed above.

The gdp_per_capita is taken from the first data object as it is fixed for all data objects for each country and the population is take from the last array element for the same reason. To get the number of people vaccinated, a for loop iterates backward through the array form the final element until it find a non-null value larger than 0. I found that sometimes in the last object of the array the number of people vaccinated was 0 so the loop finds the most recent vaccination figure. The latitude and longitude are then taken from the coordinates data object and the final object is pushed to the deaths array and returned.

```
// Function to created the data objects used in the data visualisation
function sumDeaths(data) {

    const deaths = [];

    for(i in data) {

        const countryInd = i;
        // If the country is one in focus, pass if not
        if(countrySelection.includes(i)) {

            const country = data[countryInd];
            var totalDeaths = 0;
            var totalVax = 0;
            // Loop through country array and total deaths
            for(d of country) {
                const deaths = parseInt(d.new_deaths)
                if(!isNaN(deaths)) {
                    totalDeaths += deaths;

                }
            }

            const income = country[0].gdp_per_capita
            var vaccinated = 0;
            var pop = pop = +country[country.length-1].population;
            // Loop back through the array to get the most recent vax figure
            for(let v = 1; v < country.length; v++) {
                var vaxTemp = +country[country.length-v].people_fully_vaccinated
                if(!isNaN(vaxTemp) && vaxTemp > 0) {
                    vaccinated = vaxTemp;
                    break;
                }
            }
            const lng = coordinates[i].long;
            const lt = coordinates[i].lat;
            deaths.push({country : i,
                         deaths: totalDeaths,
                         vaxxed: vaccinated,
                         population: pop,
                         gdp: +income,
                         long: +lng,
                         lat: +lt})
        }

    }

    return deaths;
}
```

Function to create the data used in visualisations

3. Visualisations

3.1 Map

As mentioned in section 1, I have used Mapbox to render a map on the dashboard as an interactive map can be rendered using their API which also offers other benefits such as a geocoding API. The map is contained in a div that is initiated at the beginning of the script tag by calling new mapboxgl.Map and using my access token to render a map including street details. A function called project is then created which maps real world coordinate data passed to it as inputs to the coordinates on the map rendered on screen so that country data is mapped to the correct country.

Once the data has been transformed and prepared, circles are created where the radius is the number of deaths in the data divided by 800 so the circles are a more reasonable size. Given more time I would have normalised this data by country population to give a more balanced view of the data. The “cx” and “cy” attributes of the circles are then set by calling the project function, using the countries coordinate data as inputs. This correctly places the centre of the circle over the correct country.

An update function is also created so that the circle data can be updated when a user uses the slider to select a new month.

```
// Function to update the circle data
function updateCircle(data) {

    var newColor = d3.scaleSequential().domain([0,d3.max(data, function(d) { return +d.deaths})]).range(["white", "red"]);

    svg.selectAll("circle")
        .data(data)
        .transition()
        .duration(500)
        .attr("cx", function (d) {
        var longLat = [+d.long, +d.lat];
            return project(longLat).x; // Project used to map real coordinates to map coordinates
        })
        .attr("cy", function (d) {
            var longLat = [+d.long, +d.lat];
            return project(longLat).y;
        })
        .attr("r", function(d) {
            return d.deaths / 500;
        })
        .attr("fill", function(d) {
            return newColor(d.deaths);
        });

    render();
};
```

Function to re-render the circles when new data is selected

The map is also re-rendered when the user scrolls and zoom around the map, resizing the circles to be relative to the position and zoom.

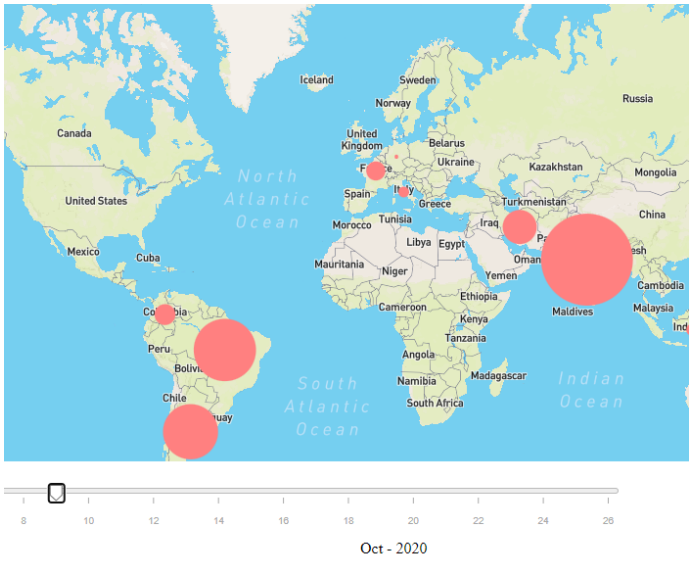
```
// Re-render the circles when the user scrolls and/or zooms
map.on("viewreset", render);
map.on("move", render);
map.on("moveend", render);
render(); // Render initially
```

3.1.1 Slider

The slider allows a user to scroll through different months of the pandemic and see how the death toll changes in each country every month. When the slider is moved, the 'onchange' event handler is called and the value of the slider at that point is passed in as an argument. This triggers the update of the data to a new month by changing the values of the "deaths" array using the relevant months data in a chined function call of sumDeaths(groupByCountry(newMonthData)). After the new data is produced, the map circles are re-rendered, and the pie chart is updated. The month is also displayed as text to the user under the slider.

```
.on('onchange', function(val) {
    const month = dateTicks[val];

    var newMonthData = monthGroup[month]; // Get data for new month from the month grouping

    deaths = sumDeaths(groupByCountry(newMonthData)); // Group by country and create data objects

    updateCircle(deaths); // Re-render circles on the map

    updateCurrentCountry(currentCountry); // Update the pie chart

    // Get the human readable format of the new month
    for(i in months) {
        if(months[i].index == val) {
            var displayMonth = months[i].format;
        }
    }

    // Update text under the slider to show new month
    d3.select('#value').text(displayMonth);
});
```

Function to update the data on the map and the circle when a user moves the slider



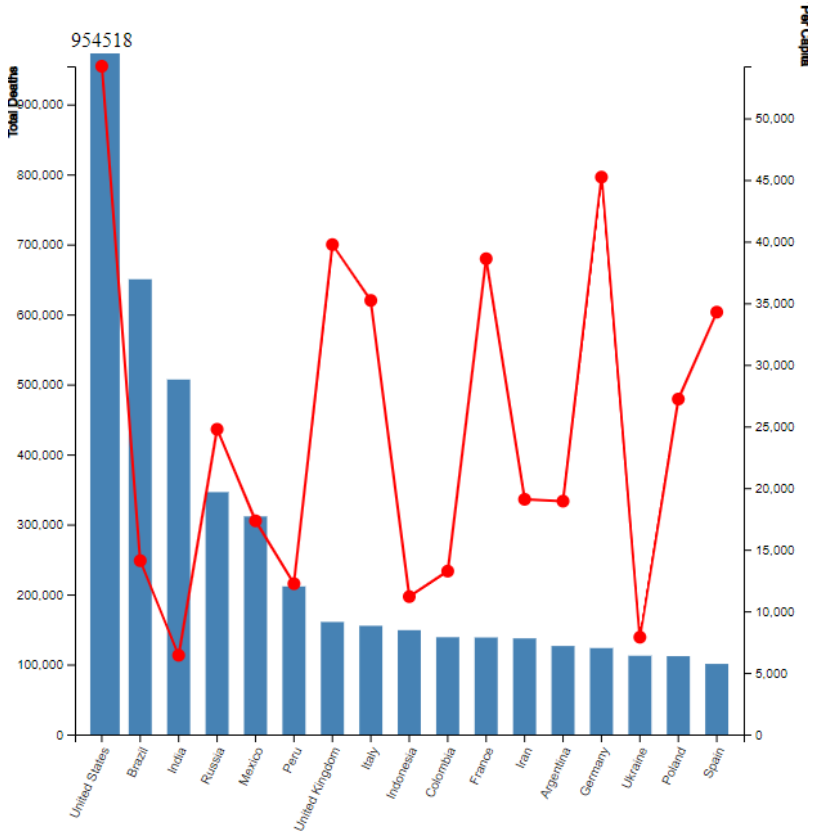
Deaths by country for October 2020

Deaths by country for November 2020

3.2 Bar and Line Chart

The bar chart displays the total deaths for the entire dataset for the countries whose total death tolls were above 100,000. I restricted it to those over 100,000 so there was a reasonable number of data points on the bar chart and it was still readable. On the right-hand y axis is shown the GDP er capita of each of those countries. This chart is intended to display the link, if any, between a country's wealth and the death toll. Based on the visualisation, there does not seem to be a correlation between country wealth and death toll.

Bar chart tooltips example



I have included tooltips on the bars and line chart dots so that when the user hovers over a particular data point, the value of that data point is displayed above.

```
//mouseover event handler function
function onMouseOverBar(d, i) {
    d3.select(this)
        .transition() // adds animation
        .duration(400)
        .attr('width', x.bandwidth() + 5)
        .attr("y", function(d) { return y(d.deaths) - 10; })
        .attr("height", function(d) { return height - y(d.deaths) + 10; });

    g.append("text")
        .attr('class', 'val')
        .attr("text-anchor", "middle")
        .attr('x', function() {
            return x(i.country) + x.bandwidth() / 2;
        })
        .attr('y', function() {
            return y(i.deaths) - 15;
        })
        .text( function(d) { return i.deaths; } ); // Value of the text
    }
```

Function to show tooltips on bars of the bar chart

3.3 Pie Chart

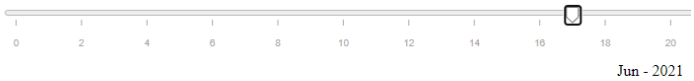
For the pie chart, I have chosen to display for each of the 16 countries the proportion of vaccinated people vs unvaccinated per month. The pie chart is controlled by the slider in the same way as the map. A variable called currentCountry contains a string of the country currently shown on the piechart. It is initiated to the first country in the drop down list which is Argentina. When the user selects a new country from the drop down an event handler is triggered that gets the name of country selected by the user and calls updateCurrentCountry, passing in the selected name as an

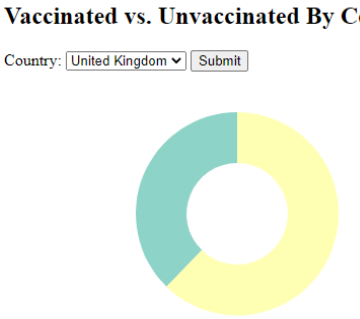
argument. In updateCurrentCountry, the currentCountry variable is updated to the new value and the current months dataset is search in a for loop in order to return the current months data of the new currentCountry. Once found, the new data is passed into updatePie and the pie chart is update. There is not data for each country for every month and so if not data is found, a message is shown to the user asking them to pick another country.

```
// Function to select new country data and call the updating of the pie chart.
function updateCurrentCountry(newCountry) {
    currentCountry = newCountry;

    // Looop through the current months deaths dataset and select data from the new country
    for(i in deaths) {
            if(deaths[i].country == currentCountry) {
                var newPieData = i;
                break;

            }
            else {
                var newPieData = -1;
            }
    }
    // If data is found, update the pie chart, else show user an error message
    if(newPieData > -1) {
        document.getElementById("errorMsg").textContent = "";
        updatePie(deaths[i]);
    }
    else {
        document.getElementById("errorMsg").textContent = "No data for " + currentCountry +
                                                           " for this month, please chose another.";
    }


}
```

Function to update the current country and then pie chart

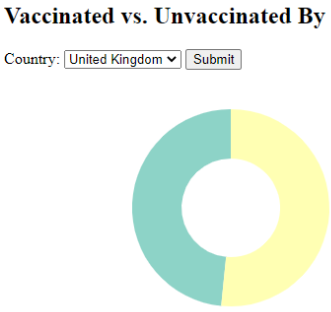
Error message saying data for selected country is unavailable





UK Vaccination data for May 2021

Green is vaccinated and yellow is unvaccinated



UK Vaccination data for June 2021

Green is vaccinated and yellow is unvaccinated

4. Conclusion

Due to illness, I was unable to complete the dashboard to the standard I would have wished, for example adding text labels and a legend to the pie chart, cleaning up the formatting on the bar/line chart, adding tool tips to the map, cluster analysis and adding cross-layout brushing, bidirectional interaction and faceted selection. I will endeavour to include these elements in my submission for the next lab.