Student Declaration of Authorship

HERIOT WATT UNIVERSITY
UK | DUBAI | MALAYSIA

Course code and name:	F20DVA/F21DVA Data Visualization and Analytics
Type of assessment:	**Individual**
Coursework Title:	Data Visualization and Analytics
Student Name:	Anshraj Rastogi
Student ID Number:	H00415334

Declaration of authorship. By signing this form:

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penalties that I will face should I not adhere to the University Regulations.

- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

Student Signature *(type your name):* *Anshraj Rastogi*

Date: *12/03/2023*

Copy this page and insert it into your coursework file in front of your title page.
For group assessment each group member must sign a separate form and all forms must be included with the group submission.

Your work will not be marked if a signed copy of this form is not included with your submission.

Coursework 1- **F20DVA/F21DVA – Data Visualization and Analytics**

Topic: **Data Visualization and Analytics**

Submitted by: **Ansh Raj Rastogi**

Video Link: 2023-03-15 17-07-01.mkv

INDEX

Topic	Page
1. Introduction	4
2. Design Considerations	4
3. The Application	5
4. Analysis	12
5. References	13

1.INTRODUCTION

This information dashboard's main goal is to give readers a comprehensive understanding of the handling and distribution of COVID-19. Also, it is a wonderful tool that enhanced my d3.js visualization skills. It is simpler for viewers to spot patterns, trends, and outliers throughout the dataset because to the visualization.

Viewers can access the COVID data for many nations using this dashboard, which enables users to see various features such as the rise in overall cases, new infections, death count, vaccination rates, etc.

The most recent version of d3, version 7, was employed for the construction of this web application.

2.DESIGN CONSIDERATIONS

Efficiency is one of the fundamental requirements for any application. Other developers should be able to understand the application's code easily because of its straightforward nature and scalability.

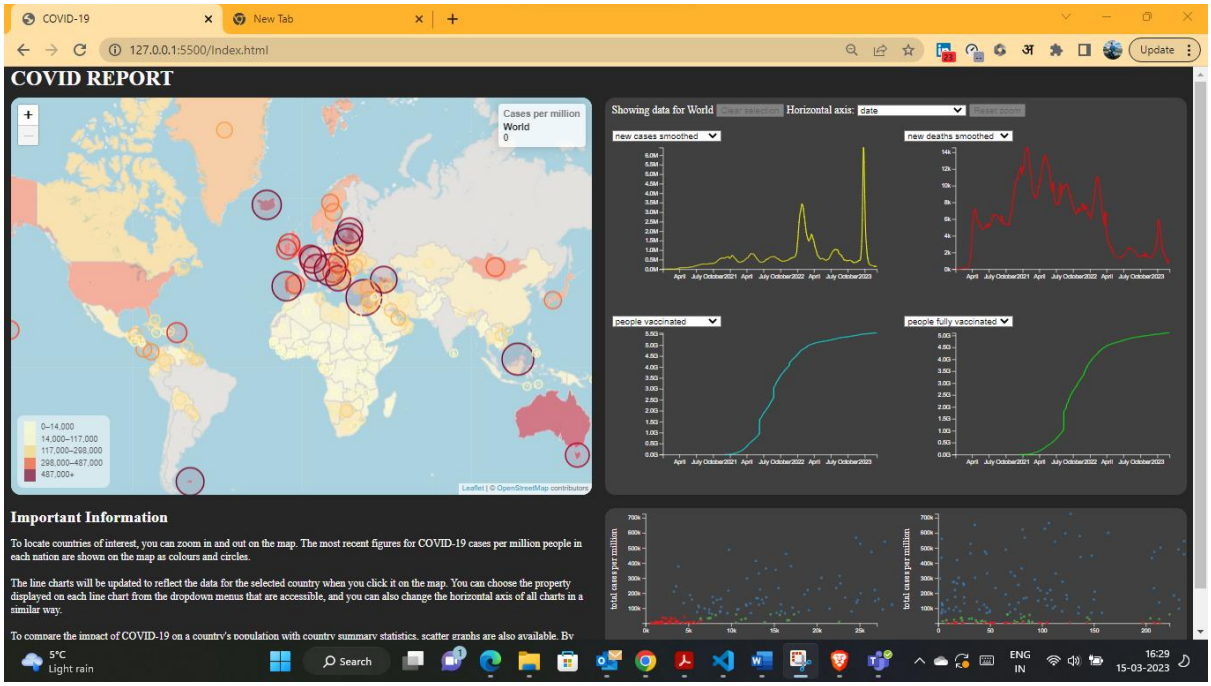
I have used various files to get the data, as well as various files for the map and various plots, to keep this throughout the code.

The dashboard's layout is provided by the main file index.html, and main page.css serves as its CSS.

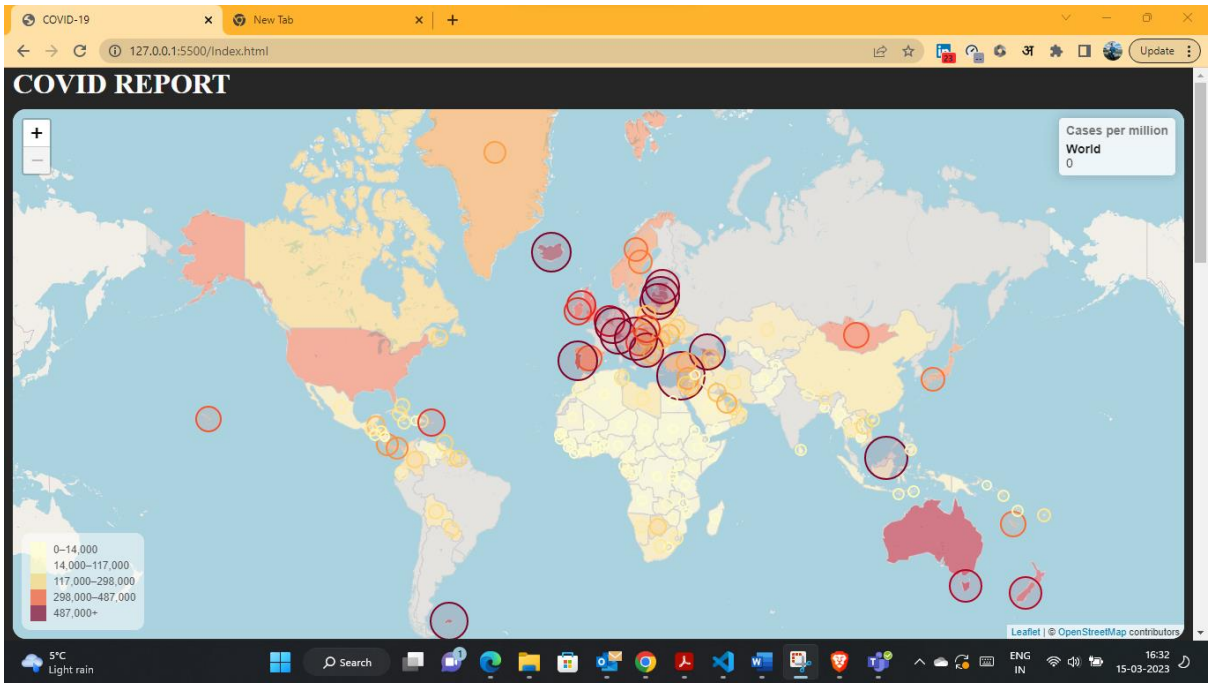
The main difficulty I encountered was trying to interpret the dataset due to its size and use it to generate several graphs.

3.THE APPLICATION

3.1-THE MAP



The map, line graph, and scatter plot are just a few of the numerous layouts that make up the main view of the Covid-19 report dashboard.



The world map in the image above can be seen in its entirety. It is color-coded based on cases per million, and the circles on the map illustrate where the Covid-19 hotspots are in the world. The upper right corner of the map also has a dynamic legend that, when the website loads, displays the Cases per Million over the world. However, as soon as the mouse pointer is moved to a particular country, the legend changes to show actual Cases per Million of just that nation. You may also zoom in and out of the map to find specific countries of interest.

```
const worldGeoData = await geoData();
const worldCovidData = await covidData();

const allCountryStats = Object.keys(worldCovidData).map(k => {
  const countryInfo = worldCovidData[k];
  const latestStats = countryInfo.data[countryInfo.data.length - 1];

  // Using normalised data for a consistent scale
  return latestStats.total_cases_per_million;
})

// Colouring countries by covid cases to show regional influence
// Using 0.9 quantile to reduce influence of outliers on colouring
const countryMax = d3.quantile(allCountryStats, 0.9);
const colorScale = d3.scaleSequential(d3.interpolateYlOrRd)
  .domain([0, countryMax]);
updateModel({mapColors: colorScale});
```

Importing data into the map utilising techniques from other files and using normalised data to ensure scale consistency.

```
// Add legend to map
legend.onAdd = () => {
    const div = L.DomUtil.create('div', 'info legend');

    // Divide the data range into suitable quantiles
    const grades = [
      0,
      d3.quantile(allCountryStats, 0.25),
      d3.quantile(allCountryStats, 0.5),
      d3.quantile(allCountryStats, 0.75),
      d3.quantile(allCountryStats, 0.9)
    ];

    function formatNum(number) {
      return (Math.round(number / 1000) * 1000).toLocaleString()
    }
```

The legend in the map's upper right corner was made using the legend.onAdd() function. The figure also demonstrates how the data range was divided into appropriate quantiles within array grades.

```
161
162     div.html(
163       '<h4>Cases per million</h4>'
164       + `<b>${country.location}</b><br/>` +
165       `${Math.round(latestStats.total_cases_per_million).toLocaleString()}`
166     );
167   };
168
169   info.addTo(map);
170   addModelListener(({ hoveredCountry }) => {
171     info.update(worldCovidData[hoveredCountry ?? 'OWID_WRL']);
172   });
173
174   function onMouseEnter(event) {
175     const { target } = event;
176
177     target.setStyle({
178       fillOpacity: 0.2,
179     });
180
181     updateModel({hoveredCountry: target.feature.properties.iso_a3});
182   }
```

The div.html is used to set values into the legend and add it to the map. The .addTo() method updates the value into the legend based on the country on which the mouse is hovered onto.

```
/**
 * updating to the data model.
 * @param {object} changes object with new values for attributes
 */
export function updateModel(changes) {
  const delta = {};

  for (const key in changes) {
    // keys existing in the model can only be udpated
    if (key in model) {
      // dispatches changed value to listeners only
      if (model[key] != changes[key]) {
        model[key] = changes[key];
        delta[key] = changes[key];
      }
    } else {
      throw `Bad model update atempted with key: ${key}`;
    }
  }

  for (const listener of listeners) {
    listener(delta);
  }
}
```

The updateModel function checks to see if the key is present in the model after receiving the input parameter as updates, which are the objects with new values for the attributes. The altered value is provided to the event listener if the key exists in the model alone. Also, if the key is absent from the model, an exception is thrown.

The array delta, which is supplied as a parameter to the method listener, is used to dispatch the altered values to the listener using the second for loop.

3.2-CHART UPDATE

```
updateHorizontalAxis() {
  // Only date dimension is non-numeric
  this.xScale = this.timeline ? d3.scaleTime() : d3.scaleLinear();
  this.xScale.range([0, innerWidth]);

  // By default, use full extent of horizontal axis to reflect missing data
  this.xScale.domain(this.bounds ?? this.domain);

  const x = d3.axisBottom(this.xScale)

  // Prevent large numeric values from creating long tick labels
  if (!this.timeline) {
    x.ticks(null, 's');
  }

  this.xAxis.transition()
    .duration(2000)
    .call(x);

  this.updateLine();
}

updateVerticalAxis() {
  // Quantity of interest should always be shown with respect to 0
  this.yScale.domain([0, d3.max(this.lineData, d => d[this.vertical])])
  this.yAxis.transition()
    .duration(2000)
    .call(d3.axisLeft(this.yScale).ticks(null, 's'));
```

According on the values the user chooses from the dropdown menus, the functions depicted in the above graphic are employed to update the vertical and horizontal axis. The graph is updated using the update line function based on the user's selection. Moreover, I've added a temporary transition effect so that we can see the graphs change and improve the user experience.

3.3-MAIN PAGE

```
<body>
  <main>
    <h1>COVID REPORT</h1>
    <section id="map"></section>
    <section id="instructions">
      <h2>Important Information</h2>
      <p>
        To locate countries of interest, you can zoom in and out on the map.
        The most recent figures for COVID-19 cases per million people in each nation are shown on the map as co
        and circles.
        <br />
        <br />
        The line charts will be updated to reflect the data for the selected country when you click it on the m
       You can choose the property displayed on each line chart from the dropdown menus that are accessible, an
       can also change the horizontal axis of all charts in a similar way.
        <br />
        <br />
       To compare the impact of COVID-19 on a country's population with country summary statistics,
        scatter graphs are also available.
        By clicking and dragging a box over them, points can be highlighted for comparison across
        the plots and coloured according to the results of the k-means clustering analysis.
      </p>

    </section>
    <section id="line-charts">
      <div id="toolbar">
        <label id="charts-title">Loading...</label>
        <button id="reset-selection">Clear selection</button>
        <label>Horizontal axis:</label>
        <select id="charts-select"></select>
        <button id="reset-bounds">Reset zoom</button>
      </div>
```

The Index.html file, which contains the primary designs for the dashboard's main page, is where the image above is taken from. This file gives the dashboard its fundamental organisation, choosing the many dashboard parts and the CSS that will be used for each one. The primary file utilised to execute our web application is this one. Additionally, it ties the CSS file to the application and imports the D3 js version 7 file.

3.4-MAIN JS FILE

The html file that contains the dashboard setup is the other most crucial file to load in all the visuals and transitions.

```
import { ClusterChart } from './modules/cluster_chart.js';
import { covidData } from './modules/datafetch.js';
import { LineChart } from './modules/line_chart.js';
import { makeMap } from './modules/map.js';
import { addModelListener, updateModel } from './modules/dashmodel.js';

// Load covid data on page load so that it's ready ASAP
covidData();

makeMap();

// Toolbar section above line charts
addModelListener(async ({ selectedCountry }) => {
  if (!selectedCountry) return;

  const data = await covidData();
  d3.select('#charts-title')
      .text(`Showing data for ${data[selectedCountry].location}`);
});

// Selection box to change horizontal axis of all line charts
d3.select('#charts-select')
    .on('change', event => {
      updateModel({
        axisValue: event.target.value,
        bounds: null,
      });
    })
  .selectAll('option')
    .data(covidData.toPlotAgainst)
  .join('option')
    .attr('value', d => d)
```

The image above demonstrates how all the JavaScript files were imported into the main.js file to use all the necessary functions and updates needed for the dashboard's smooth operation. When the page loads, the file first runs the methods makeMap() and covidData() to load the data and initialise the map with all of the default values.

The toolbar above the line charts shows the selected country and allows the user to change the label on the axis for the line chart. This also provides better accessibility to the users.

3.5-THE CLEAR SELECTION BUTTON

The clear selection button offers the following two features:

1. Resets the currently selected nation on the map to display data for the entire world.

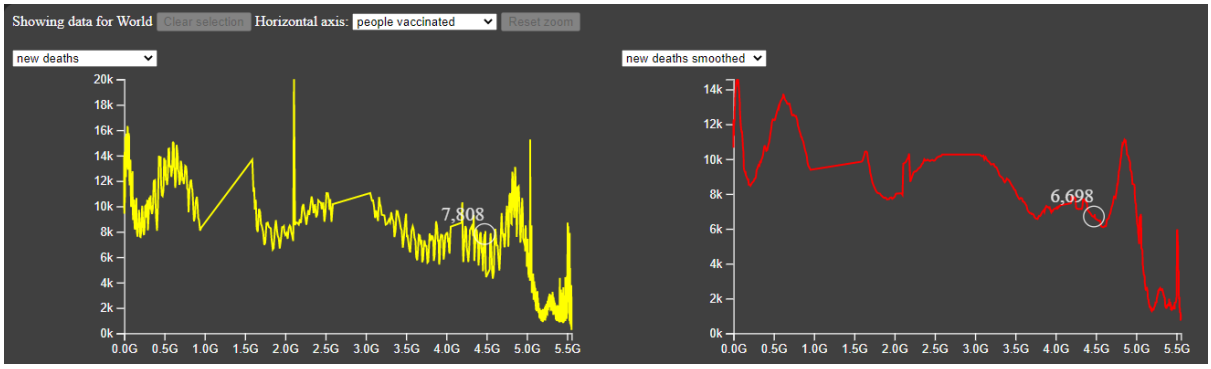
2. Corrects all line chart values, and the horizontal axis value is changed to Total Cases.

4. ANALYSIS

Choosing some specific countries, is there a relationship between the relative "wealth" (e.g., GDP) of a population and the spread of the pandemic?

The scatter plot reveals a group of nations with low death rates and low GDP per capita. The remaining nations are just dispersed throughout the graph. This does not demonstrate any connection between a nation's wealth and its population. It just demonstrates that there are a greater number of nations with low GDPs than there are nations with low mortality rates.

What effect did vaccinations have on the spread of cases/deaths? Did booster jabs also have an impact on the spread/transmissibility of the virus?



The graph above demonstrates that as more people have received vaccinations, deaths have decreased from their peak levels. The main conclusion is that over time, as the total population who received vaccinations increased, the number of new deaths declined, despite some peaks and declines at certain points in the graph.

5.References

- https://www.naturalearthdata.com/downloads/10m-cultural-vectors/
- https://bost.ocks.org/mike/map/
- https://www.tutorialsteacher.com/d3js/loading-data-from-file-in-d3js#d3.csv
- https://dev.to/jessesbyers/react-and-d3-dynamic-covid-19-visualizations-part-2-country-comparison-dashboard-f9n
- https://www.npmjs.com/package/world-atlas
- https://observablehq.com/@d3/world-map
- https://www.youtube.com/watch?v=tXIc9ki1ZVU&list=PLTgRMOcmRb3PezbW0FziWyoZCAGi_8XzZ&index=10
- covid-19-data/public/data at master · owid/covid-19-data (github.com)
- covid-19-data/public/data/vaccinations at master · owid/covid-19-data (github.com)
- Create With Data (d3indepth.com)
- Basic choropleth map in d3.js (d3-graph-gallery.com)
- Choropleth map with hover effect in d3.js (d3-graph-gallery.com)
- Bubblemap template for d3.js (d3-graph-gallery.com)