Student Declaration of Authorship

HERIOT WATT UNIVERSITY
UK | DUBAI | MALAYSIA

Course code and name:	F21DV Data Visualization and Analytics
Type of assessment:	**Group / Individual** *(delete as appropriate)*
Coursework Title:	Lab 4
Student Name:	Jonathan Lee
Student ID Number:	H00255553

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):* *Jonathan Lee*

Date: *01/04/2022*

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.

F21DV Coursework Lab 4 Report

Jonathan Song Yang, Lee (H00255553)

Demonstrated to: Dr. Benjamin Kenwright (08/04/2022)

Contents

1 Introduction

Lab 4 focuses on using the general visualization skill set obtained from the past 3 Labs, to design a complex, dynamic and interactive visualization webpage.

GitHub Repo: https://github.com/jonleesy/F21DV-Coursework

GitHub Pages: https://jonleesy.github.io/F21DV-Coursework/public

1.1 Set Up

1.1.1 Layout Set Up

The application uses `CSS` grid to drive its layout, and through the use of `CSS` and `JavaScript` DOM, a generalized function was used to create divs as shown in line 8 - 13 in listing 1.1, where in line 20 onwards shows its implementation.

```
1  /**
2   * Function adds a div systematically to an existing div
3   * @param {*} targetClass div's class where div is being added to
4   * @param {*} className name of class of newly added div
5   * @param {*} idName name of id of newly added div
6   * @param {*} textDisplay text to display in the div
7   */
8  function addDiv(targetClass, className = '', idName = '') {
9    d3.select(targetClass)
10     .append('div')
11         .attr('class', className)
12         .attr('id', idName);
13 }
14
15 /**
16  * main entry point to the module
17  */
18 export async function setup() {
19     // add the divs
20     addDiv('body', 'grid-container');
21     addDiv('.grid-container', 'main-container');
22     addDiv('.main-container', 'main-container-div', 'map-container');
23     addDiv('.main-container', 'main-container-div', 'top-right-container');
```

Listing 1.1: Abstract of Systematic Div Creation

1.1.2 Modules Set Up

The application is set up to be event-driven, where independent modules were used to process different segments within the application, and functions could be reused through different call-back contexts. Within each module, there exist an entry point `setup()` function, and for some an additional `update()` function.

```
import { setupContent } from './content.js';
import { setup } from './functions.js';
import { setupLine } from './line.js';
import { setupMap } from './map.js';
import { setupScatter } from './scatter.js';
import { setupSlider } from './slider.js';

// set up the divs systematically
setup();

// set up the map
await setupMap();

// set up the plots
await setupScatter('top-right-container');
await setupLine();

// setup slider
await setupSlider();

// setup content
setupContent();
```

Listing 1.2: Abstract of Entry Point Code

From listing 1.2, it could be seen that the entry point for all modules is through a `setup()` function.

1.1.3 Testing Set Up

Testings were mainly user driven. However, to ensure that code quality is up to standard, a local instance of `ESLint`, `Prettier` and `SonarQube` was used to scan for vulnerability in code and ensure that best programming practices were used. The user testing and development was also done on a Mozilla based web browser.

Figure 1.1: JSDoc Module and Copyright

As shown in figure 1.1.3, I have also added on `JSDoc` style `@module` and `@copyright` descriptions to the top of each module file specifying its purposes, after the discussion during the demo.

1.1.4 Data Set Up

`Python` was used to pre-process the World Development Indicator dataset to shorten loading times. The data set would not be provided due to its sheer size, but the example of python code used to

filter the data set is as below in listing 1.3, only filtering for what is needed within the application.

```
import pandas as pd

# read csv using pd
# Data will have to be download from https://databank.worldbank.org/source/world-
    ↪ development-indicators# and redirected.
df = pd.read_csv('wdi_raw.csv')

# filter df
list_of_series_code = ['NY.GDP.PCAP.KD', 'EN.POP.DNST']
new_df = df[df['Indicator Code'].isin(list_of_series_code)]

# write to csv
new_df.to_csv('wdi.csv', index=False)
```

Listing 1.3: Python Pre-Processing

2 The Application and Its Feature

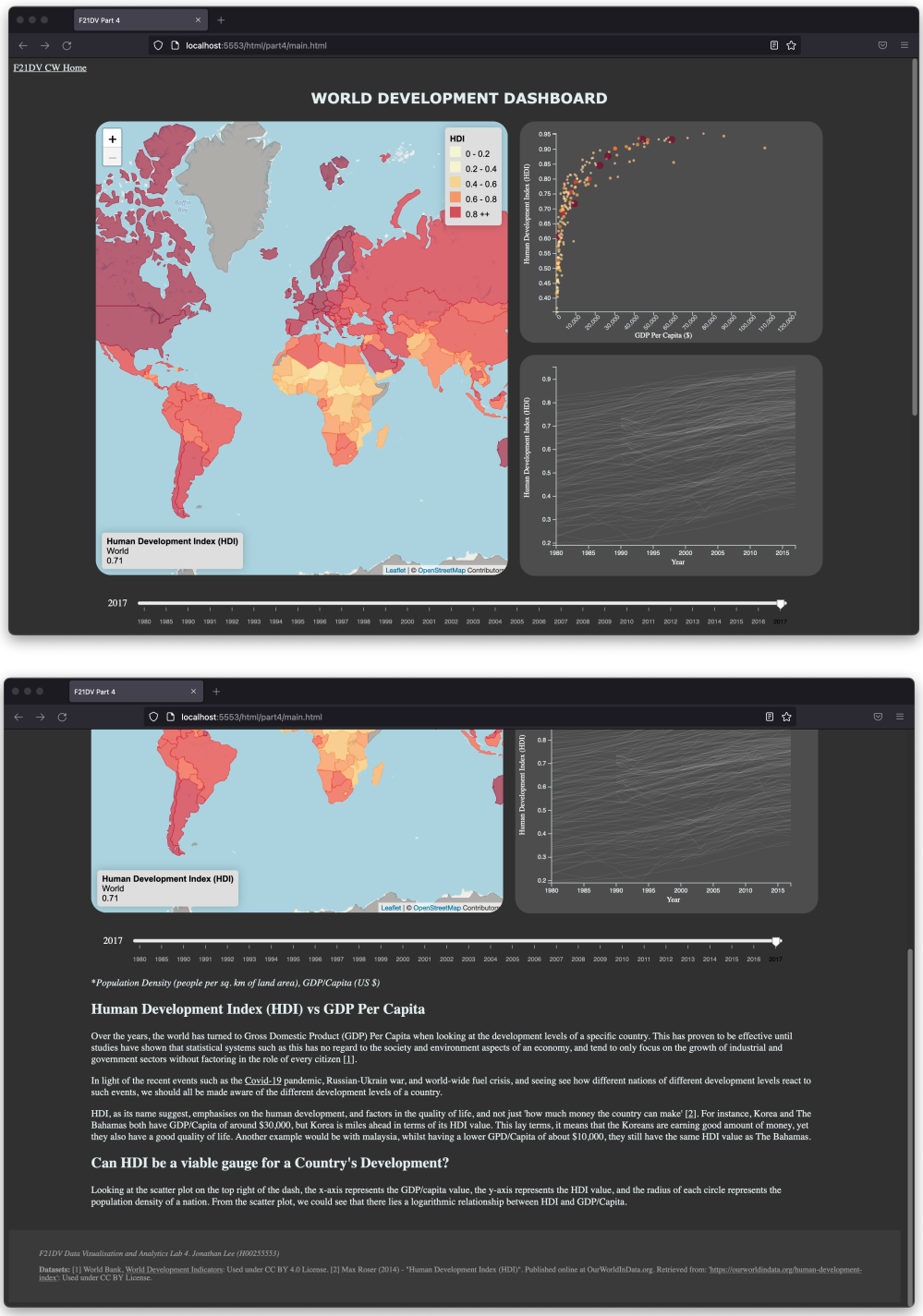

Figure 2.1: Main Dashboard Layout

Figure 2 shows the main layout of the application dashboard, mainly consisting of the map, scatter plot, multi-line chart, timeline slider, and a write-up of the dashboard.

The main idea behind choosing to look into the two main development indicator, HDI and GDP/-Capita, was inspired by the recent world events happening around us, and hoping that this would heighten our concerns to the different development levels of different countries. For instance, a coun-

try with low development index generally responded poorly to the Covid-19 pandemic. With lack of reach for vaccinations in the early days, and high population densities, these countries were hit with high virus reproduction rates and relatively worse symptoms as a result of infection. On the other hand a well-developed country with low population density and reach for advanced medical technologies were able to roll out vaccination campaigns quickly, and prevent a far worse transmission of the virus.

These are all factors that should be put towards the calculation of a country's development index, and not just in layman terms 'how much money does the country make in a year'. Hence, this dashboard focuses on the comparison of the use widely used Gross Domestic Product (GDP) per Capita (US $) and Human Development Index (HDI) as a more viable indicator for development.

2.1 Map

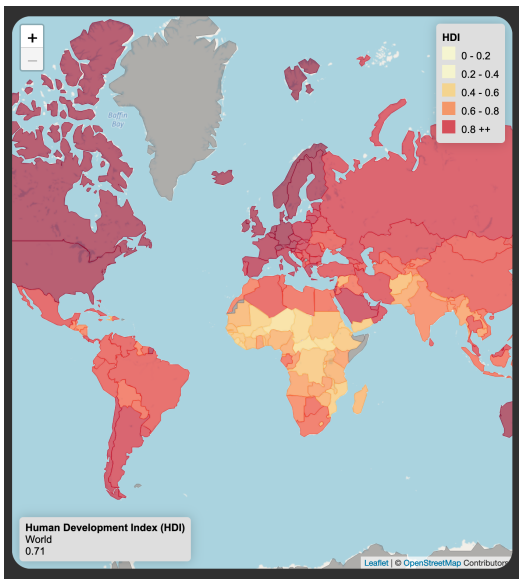

Figure 2.2: Map Container

The map shown in figure 2.1 was implemented using the `leaflet` and `d3.js` library, as shown in listing 2.1.

```
1     // create the map
2     map = L.map('map-container', {
3         center: mapCenter,
4         zoom: mapZoom,
5         maxZoom: 5,
6         minZoom: 2,
7         maxBoundsViscosity: 0.5,
8     });
9
10    // add mapjson layer
11    dataLayer = L.geoJson(geoJsonData, {
12        onEachFeature: onEachFeature,
13        style: defaultMapSettings
14        });
15    dataLayer.addTo(map);
```

Listing 2.1: Abstract of map setup

Upon the setup of the map, using the `L.control()` and `L.onAdd()` functions, we have created both the Tooltip on the bottom left, and legend on the top right. Each layer of the map were also assigned unique class values such as `'map-path-${countryCode}'`, and this is for the ease of performing style updates on individual layers.

2.1.1 Map, Scatter, Line Mouse Event

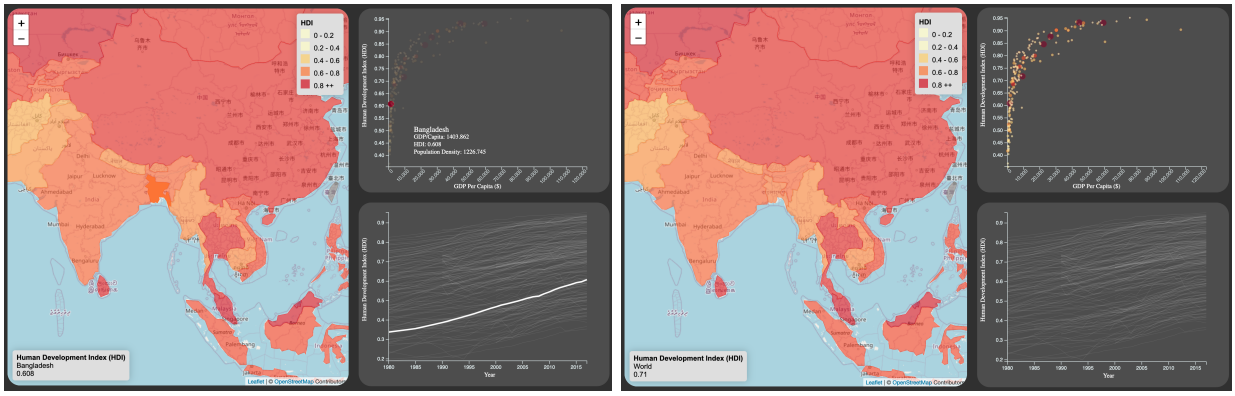

Figure 2.3: Map Mouse Update MouseOver and MouseOut

From figure 2.1.1, we could see the parts of graph being updated as a result of the `mouseOn` and `mouseOut` events. Due to the nature of the computer's OS, the physical mouse is not visible during a screenshot, but the mouse is actually on the Bangladesh polygon on the map.

Upon mouse over, the dynamic dashboard updates the following things:

- Map: Highlight the selected country.
- Scatter Plot: Highlight the selected country, blur the rest of the countries, and show a detailed tooltip of the country's data, including the population density.
- Line Chart: Highlight the selected country.

This is the same for all three visualizations on the dashboard. If the scatter plot is being interacted with, it would highlight the corresponding country on the map and update the line graph to show the country's change in HDI. The same would go for the line chart.

2.1.2 Map Zoom and Borders

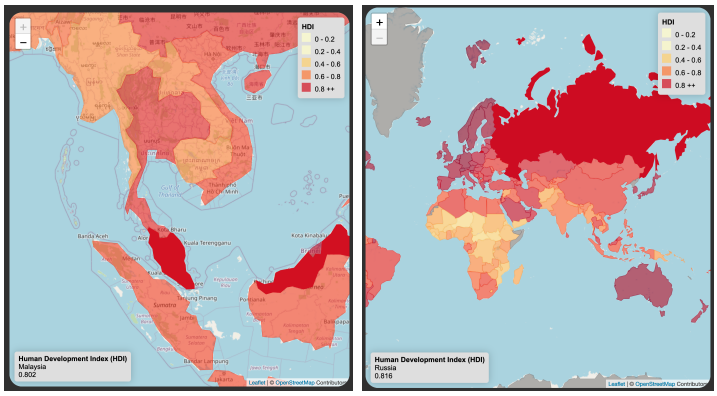

Figure 2.4: Map Minimum and Maximum Zoom

A border is being set for the map where if the user drags past the border, it would automatically bounce back. This is very hard to be shown, but the event could be tested out on the GitHub Page. From figure 2.1.2, the user could see the minimum zoom level for the map just so that user could visualize changes to multiple countries. A maximum zoom was also set since the `GeoJson` file used was one of a smaller scale to improve dashboard loading performance, hence, not being able to show in detail a country's detail. The zoom controls also greyed-out once a maximum zoom is reached.

2.2 Scatter Plot

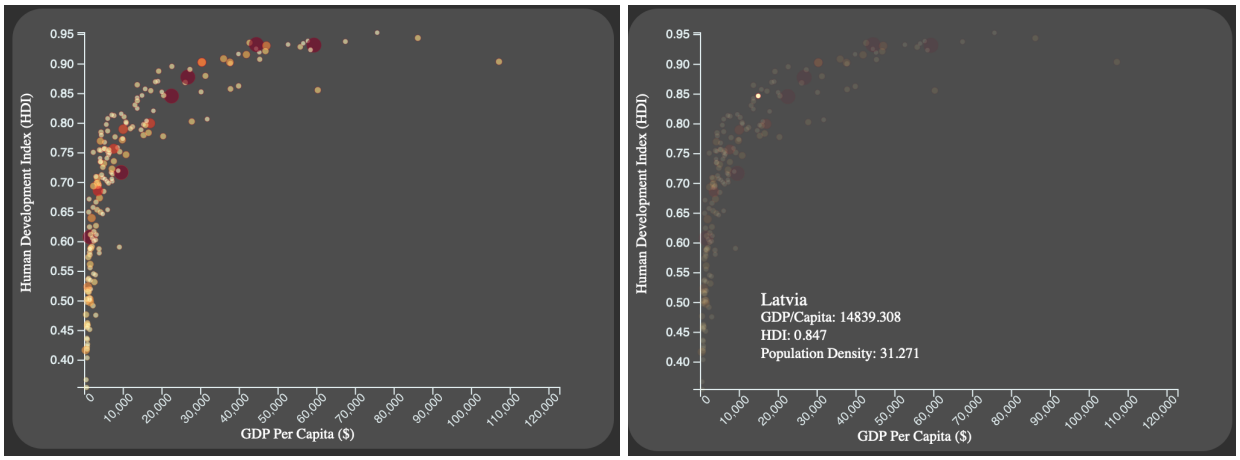

Figure 2.5: Map Minimum and Maximum Zoom

Figure 2.2 shows the scatter plot and what happens when the user hovers on a circle. The scatter plot was made solely using the `d3.js` library. This was done using a `enter()-append()-merge()-exit()` function. This ensures that upon updating the year, the plots will transition its position.

```
const u = scatterSvg.selectAll('circle')
                    .data(yearHdi);
...
// enter-append function
u.enter()
    .append('circle')
        .on('mouseover', mouseOver)
```

```
...
.merge(u)
    .transition()
        .duration(400)
            .attr('r', d => {
...
// remove extra stuff
u.exit()
    .transition()
            .duration(400)
            .remove();
```

Listing 2.2: Scatter Plot Update

The plot also shows three different types of data. The x-axis shows the GDP per Capita of the country, the y-axis shows the HDI, and the size of the circles are scaled to show the population density of a country. Once again, a unique id value has been assigned to each circle, to enable an effective way to update its style dynamically.

From the scatter plot, we can conclude that HDI is indeed a viable development index as it shows a logarithmic correlation with the GDP/Capita indicator. This also shows that population density plays no role in determining how developed a country is. For instance, Hong Kong and Singapore are one of the most dense country in the world, yet they are two of the best performing countries in terms of HDI and GDP/Capital value. GDP/Capita also fails to capture the development levels of a country, as explained within the dashboard itself.

2.3 Line Chart

From figure 2.1.1, we could see the sole purpose of the line chart. is to show growth, using HDI values, for each individual country. There is an average upward trend for all countries, and this is just what we need to see in all parts of the world, where the countries should all be moving forwards, not back.

```
// add lines
for (let item of await getListOfCountry()) {
    ...
    // add the path
    linePlotSvg.append('path')
        .attr('class', 'hdi-line')
        .attr('id', `hdi-line-${item}`)
    ...
```

Listing 2.3: Line Set Up

The way the lines were plotted was also slightly different. As shown in listing 2.3, I have used a for loop for plot a line for each country there is, using the country code as a unique identifier to ease the update process.

2.4 Slider

Figure 2.6: Slider

From figure 2.4, we could see a figure of the slider where it allows the user to slider along the slider to update the map and scatter plot the show data for the relevant years.

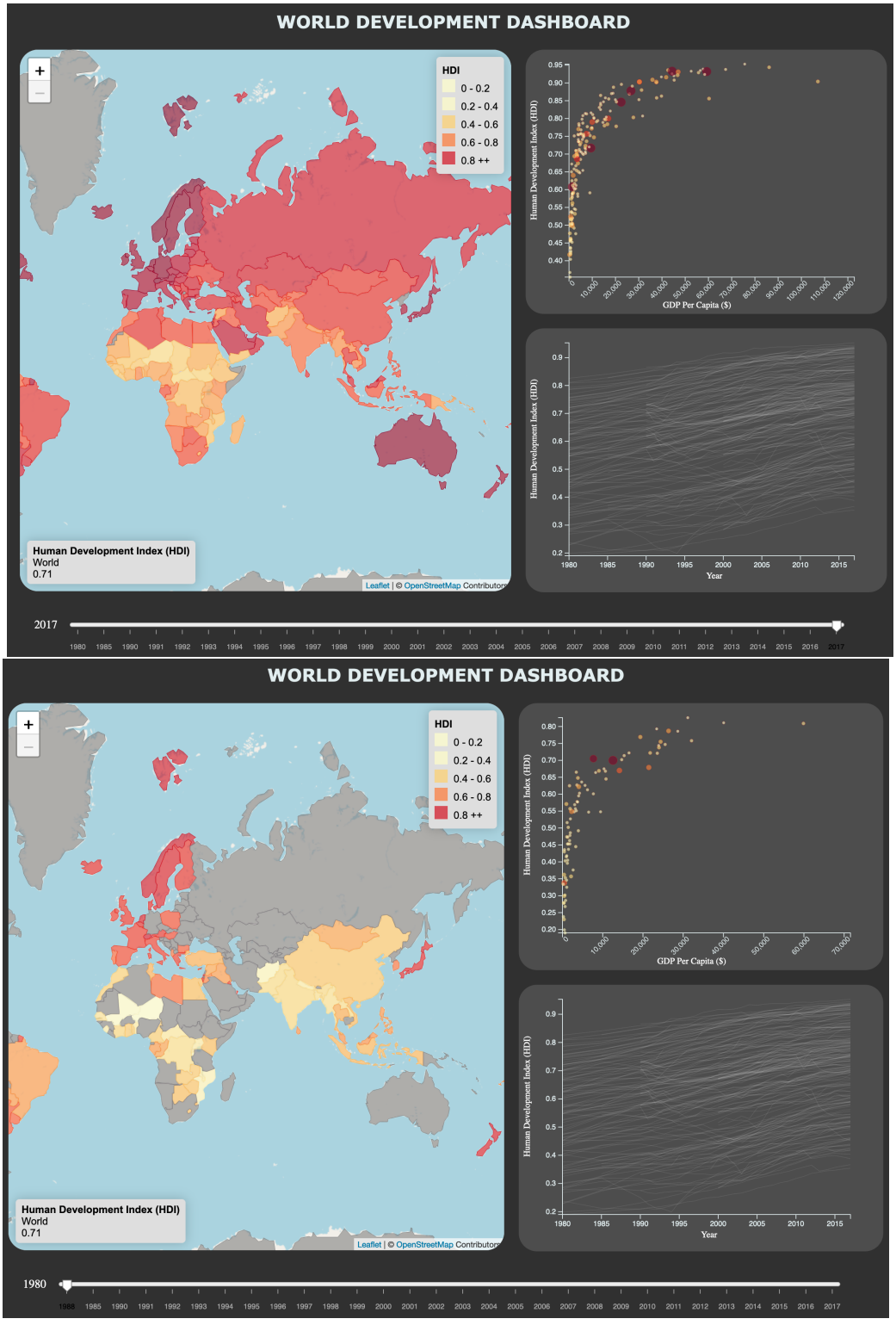

Figure 2.7: Slider update

Figure 2.4 shows what happens the map and scatter plot upon update of the slider. The default slider position is as 2017, and upon sliding the controls to 1980, the map and scatter plot immediately shows a few extra countries without HDI data. This is due to the fact that HDI was not really a thing up till the early 2000s, as shown in the line chart—multiple countries have lines only starting at 2000.

```
/**
 * functions aims at updating the charts and maps upon slider movement
 * @param {*} newYear the updated year from the slider
 */
export function updateDashboard(newYear) {
    // set a new year
    window.year = newYear;

    // update functions.
    updateScatter();
    updateMap();
}
```

Listing 2.4: Update function for the dash

The update is done by an `updateDashboard()` function, as shown in listing 2.4. Upon change of the slider, the changed value (new year value) is being passed on to the `updateDashboard()` function, where it updates a global `window.year` variable, and it is just a remaining task of simply calling the `update()` function for each of the map and scatter module.

2.5 Contents and Footer Section

Figure 2.8: Content and Footer

The content and footer section is simply just a basic html knowledge, where its purpose is to provide valuable explanation for the dashboard. I have created a generic function that can be used to add divs systematically into the container. This is as shown as below, both the function, and its implementation.

```
/**
 * this function systematically adds div (or paragraph) to the content section
 * @param {*} text text to add
 * @param {*} type ttext size, h1, h2, h3,... etc. default: 'p'
 */
function addText(text, type = 'p') {
    d3.select('.content-container')
        .append('div')
            .append(type)
                .html(text);
}

// href constants
const covid19 = `<a class="active" href='../../html/part3/main.html'>Covid-19</a>`;
const ref1 = `<a href='https://datatopics.worldbank.org/world-development-indicators/stories/statistical-performance-indicators.html'>[1]</a>`;
const ref2 = `<a href='https://medium.com/4thought-studios/human-development-index-the-new-gdp-34ce23fc8bd1'>[2]</a>`;
```

```
17
18 /**
19  * main function for the content module
20  */
21 export function setupContent() {
22     addText(`*<em>Population Density (people per sq. km of land area), GDP/Capita (US
        ↪  \$)</em>`)
23     addText('Human Development Index (HDI) vs GDP Per Capita', 'h2');
24     addText(`Over the years, the world has turned to Gross Domestic Product (GDP) Per
        ↪  Capita
25             when looking at the development levels of a specific country. This has
                ↪ proven to
26             be effective until studies have shown that statistical systems such as
                ↪ this has
27             no regard to the society and environment aspects of an economy, and tend
                ↪ to only
28             focus on the growth of industrial and government sectors without
                ↪ factoring in the
29             role of every citizen ${ref1}.`);
30     addText(`In light of the recent events such as the ${covid19} pandemic, Russian-
        ↪ Ukrain war,
31             and world-wide fuel crisis, and seeing see how different nations of
                ↪ different development
32             levels react to such events, we should all be made aware of the different
                ↪  development
33             levels of a country.`);
34     addText(`HDI, as its name suggest, emphasises on the human development, and
        ↪ factors in the
35             quality of life, and not just 'how much money the country can make' ${
                ↪ ref2}. For instance,
36             Korea and The Bahamas both have GDP/Capita of around \$30,000, but Korea
                ↪ is miles ahead
37             in terms of its HDI value. This lay terms, it means that the Koreans are
                ↪ earning good
38             amount of money, yet they also have a good quality of life. Another
                ↪ example would be
39             with malaysia, whilst having a lower GPD/Capita of about \$10,000, they
                ↪ still have the
40             same HDI value as The Bahamas.`)
41     addText(`Can HDI be a viable gauge for a Country's Development?`, 'h2')
42     addText(`Looking at the scatter plot on the top right of the dash, the x-axis
        ↪ represents the
43             GDP/capita value, the y-axis represents the HDI value, and the radius of
                ↪ each circle
44             represents the population density of a nation. From the scatter plot, we
                ↪ could see that
45             there lies a logarithmic relationship between HDI and GDP/Capita.`)
46 }
```

Listing 2.5: Content Section

3 Conclusion

In conclusion, through the 4 Labs, I have now a more robust understanding of Visualizations and particularly through the most intuitive and accessible manner—through a web application. My understanding of `d3.js`, best programming practices, and DOM has been deepened as well.