JOSIP JURAJ STROSSMAYER UNIVERSITY OF OSIJEK

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

Project task in the subject

DATA VISUALIZATION

Comparative Visualization of Energetic Transition: Climate change impact on production of energy

Student: PALLET Thomas, Computer Science

Mentor: JOSIP Job

In Osijek, June 2023

CONTENT

1. KV1 - Defining a project task ... 3
1.1. Terms of reference ... 3
1.2. Data ... 3
1.3. Data processing ... 3
1.4. Relevant display types for the data used ... 3
2. KV2 - Data visualization design. ... 4
2.1. Questions answered by visualization ... 4
2.2. Data visualization sketch ... 4
2.3. Existing solutions and examples ... 4
2.4. Customization of data ... 4
2.5. Colours and data ... 4
3. KV3 - Creating a prototype data visualization ... 5
3.1. Basic functionalities and behaviors ... 5
3.2. Advanced functionalities and behaviors: ... 5
3.3. Implementation of basic functionalities ... 5
3.4. Implementation of basic behavior ... 5
4. KV4 - Creating the final data visualization ... 6
4.1. Implementation of basic functionalities ... 6
4.2. Implementation of basic behavior ... 6
4.3. Implementation of advanced functionality ... 6
4.4. Implementation of advanced behavior ... 6
5. KV5 - Completion of the project task and writing documentation ... 7
5.1. Possible modifications and refinements of the decision - in agreement with the teacher ... 7
5.2. Preparation of documents - project documentation ... 7
Literature ... 8
Annex I ... 9

1. KV1 - Defining a Project Task

1.1. Project Task

Task name: Comparative Visualization of Energetic Transition: Climate change impact on production of energy

Problem description: How can data visualization comparing CO2 emissions and renewable energy production between countries effectively highlight disparities, identify best practices, and show efforts to address climate change on a global scale?

Task description: The main objective of the project is to visually represent and compare the impact of climate change and the progress of the energetic transition in France and selected countries over several decades. By examining the data on CO2 emissions and renewable energy production, the project aims to provide a comprehensive view of the different trajectories and challenges faced by both advanced countries like France and other nations.

Objective of the project: The purpose this project is to effectively communicate the complex dynamics of climate change and the energetic transition. By comparing the data on CO2 emissions and renewable energy production between countries, the project seeks to highlight disparities, identify best practices and an assessment during the ecological transition.

To achieve these objectives, the project will collect and analyze relevant data from multiple countries, with a particular focus on France as a benchmark for comparison. This may include historical data on CO2 emissions, renewable energy production, energy consumption, and other relevant indicators.

Link to git repository of the project: https://github.com/TheMysters/VD-Project

1.2. Data

Z-1.2.1. *Two* datasets are used to achieve the objective of the project.

The first one is about CO2 emissions of each country [2].

The second one is about production of renewable energy of the world [3].

Z-1.2.2. The first dataset combines records of CO2 emissions by country over more than 200 years. It is compiled from various sources, including the United Nations Framework Convention on Climate Change (UNFCCC) and the International Energy Agency (IEA).

The second set of data gathers records concerning the sources of different renewable energy production as well as the amount produced by country from 2000 to 2021. It is compiled from the IRENA (International Renewable Energy Agency).

1.3. Data processing

Z-1.3.1. *Process the collected data and link it to create a complete set of data. This includes cleaning and processing of data, as well as checking their consistency, topicality, integrity, i.e. quality and correctness.*

For the CO2 dataset, I selected only the column that contained information about country, year and co2 emission.

For the Renewable energy dataset, I had to merge the producer type column because the information was not relevant, only the number corresponding was.

The source from each dataset seems reliable. The only work I had to do to have a complete set of data was selecting the column that I wanted and change some name of column to have the same for countries in my world map. For the heat map, If the country doesn't have a record at the year required, a null value is added to make my code work.

1.4. Relevant display types for the data used

It is possible to use different ways to display data that would be appropriate for climate change and energetic transition such as line charts, stacked charts, heat map.

Z-1.4.1. To understand the dynamic of climate change and the energetic transition, there are several possible ways:

1. Line charts to display trends in CO2 emissions and renewable energy production over time.

2. Stacked area charts to showcase the contribution of different renewable energy sources to overall renewable energy production over time.

3. World map with gradient of color: to represent CO2 emissions and renewable energy production on a geographical scale.

2. KV2 - Data visualization design.

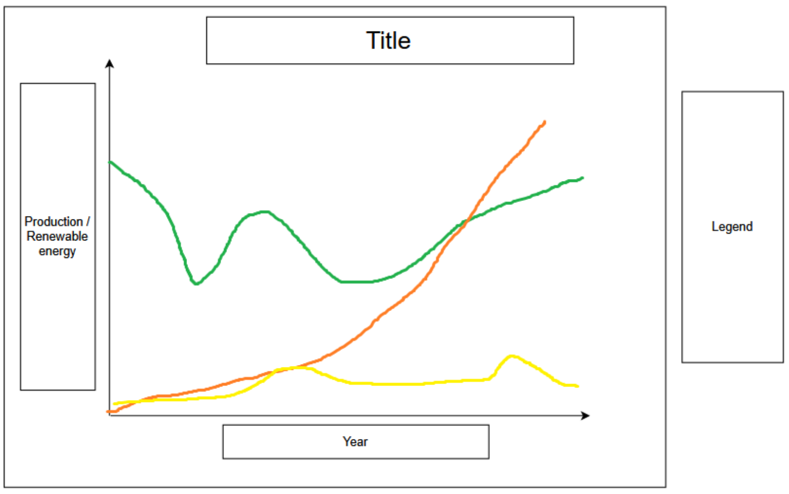
2.1. Questions that visualization answers

Z-2.1.1. Each type of graph will be able to answer to a specific question:

- Line charts:
  1. How have CO2 emissions changed in France and other countries over the past few decades?
  2. What is the trend in renewable energy production in France compared to other countries?
- Stacked area charts:
  1. How has CO2 emissions changed over time?
  2. Which renewable energy sources country's contribution have shown significant growth or decline in their contribution to energy production?
- Heat map:
  1. What are the regions or countries with the highest CO2 emissions?
  2. Which areas have made substantial progress in renewable energy production?
  3. Are there any geographical patterns or disparities in emissions and renewable energy production across different countries or regions?

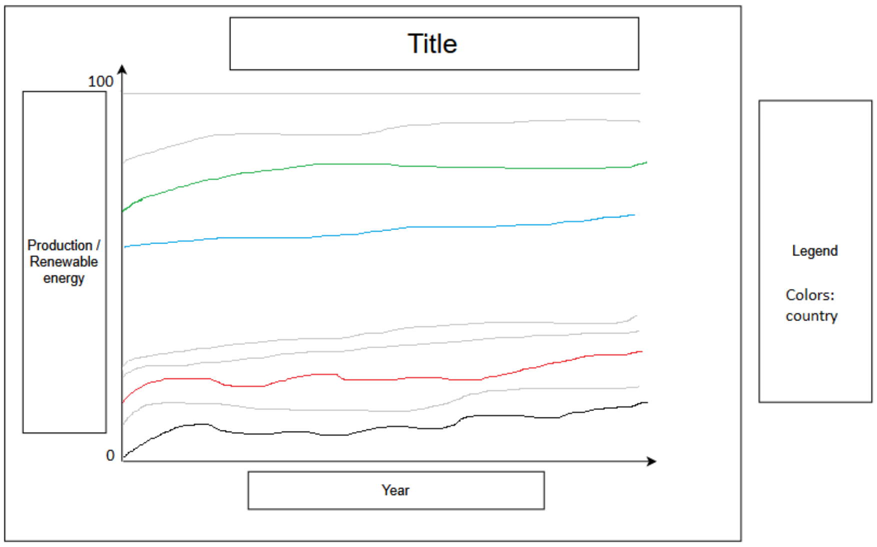
2.2. Data visualization draft

Z-2.2.1. -For the line charts, here is a draft of what it should looks like:

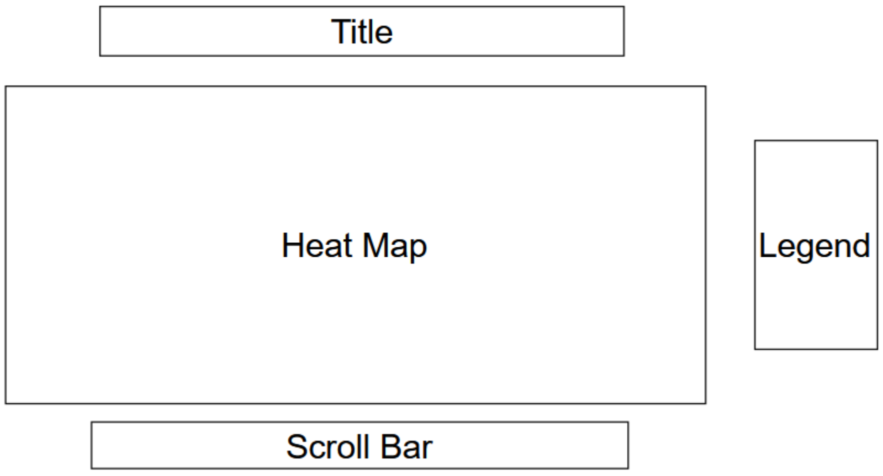


The x axis will be displaying year and the y-axis the value for the corresponding year. A legend will be displayed to know what color represents which country.

-For the stacked charts, it is the same idea but it will display percentage in the y-axis:



-For the heat map, it will display a map of the world and all the countries will be colorized based on a color gradient that will be implemented according to the distribution of the value of the emission or production of renewable energy for each year. A scroll bar will be availabe to go through the year.



2.3. Existing solutions and examples

Z-2.3.1. They are plenty of projects about data visualization of climate change and renewable energy.

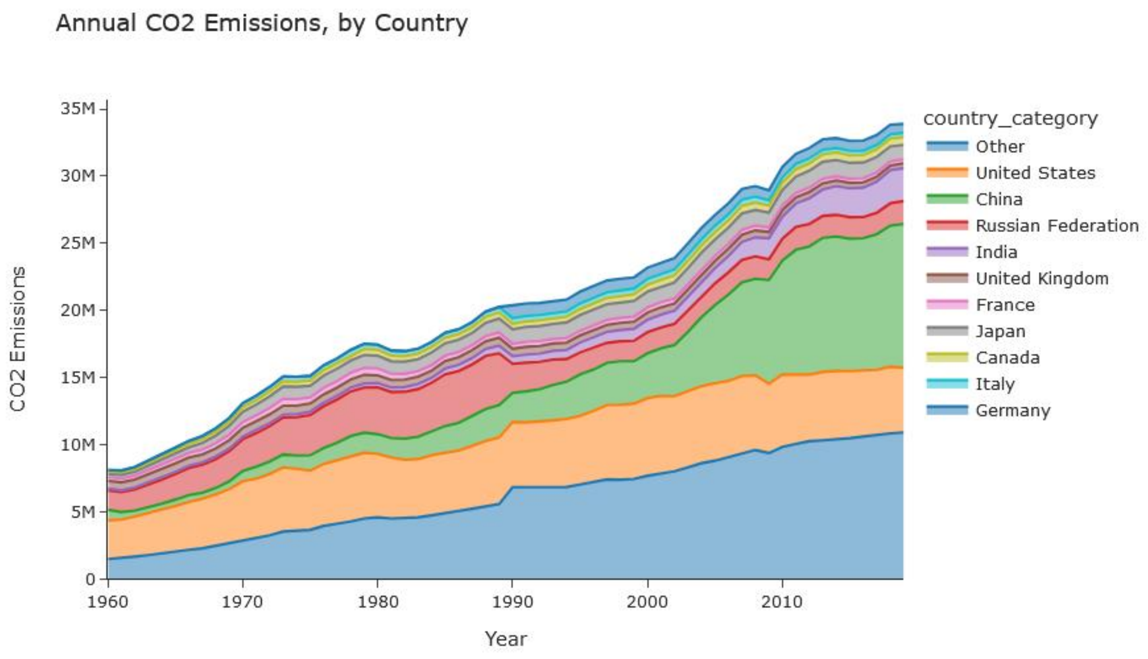
For example, there is Global Carbon Atlas [1] project that visualize data about global CO2 emissions and many GitHub projects or Kaggle projects about visualization of CO2 emissions and renewable energy production, mainly using python.

Z-2.3.2. Code for CO2 emissions can be found in the Kaggle website as CO2 Emission Plots [2].
Code for CO2 emissions and renewable energy production can be found in the Kaggle website as Renewable Energy World Wide: 1965 - 2022 [3].

Z-2.3.3. No part of the code will be used in the project. Some parts will be the same but with using d3 in JavaScript.
Example of a code in python:

```
px.area(df_country_10, x='year', y='value', line_group='country_category', color='country_category',
                   category_orders={
                    "country_category": ['Other'] + countries
            },
                labels={
                     "value": "CO2 Emissions",
                     "year": "Year",
                     "country_name": "Country"
                 },
              title = 'Annual CO2 Emissions, by Country',
       template="simple_white")
```

This code uses Ploty to create an area plot that visualizes the annual CO2 emissions for different countries over time, grouped by country category. The plot has a simple white background, and the legend is ordered to display "Other" followed by the individual country names.



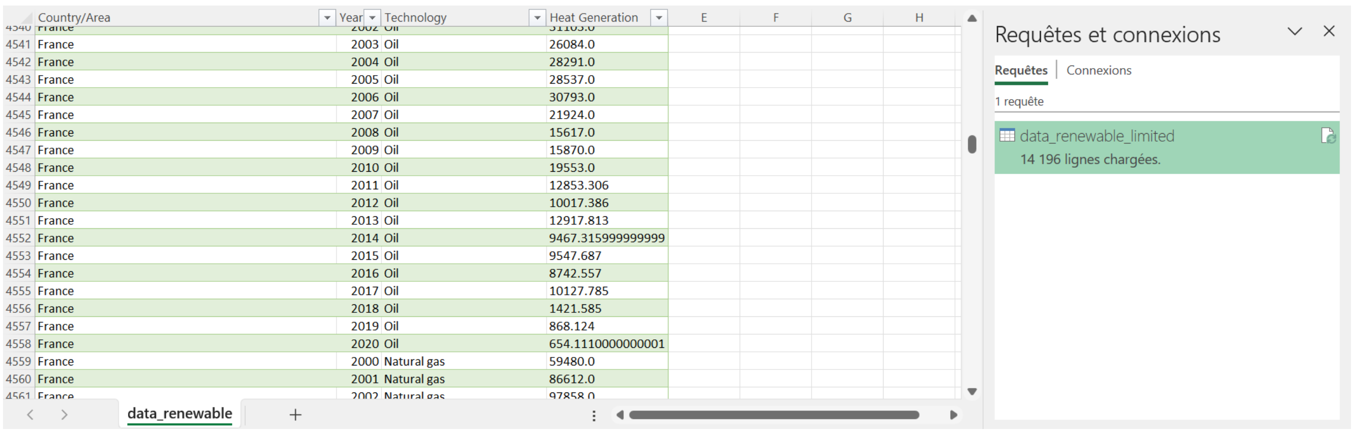
2.4. Customizing data

Z-2.4.1. The operations applied on the dataset are selection of some columns to reduce the dataset and take only the column useful for the visualization. A merge has been applied to a dataset to restrain the amount of information.

Z-2.4.2. The data is loaded from a csv file in JavaScript and select the column that it is needed to display it into a chart. To select the data, a variable contains the content of the csv file in JavaScript, and it is used during all the processing of the data.

Z-2.4.3. The data will be displayed with excel.

Z-2.4.4. Here is an image of one of the datasets loaded from excel with a csv format, the merge of the previous column was successful because the quantity of the dataset has been reduced by 2.



	Country/Area	Year	Technology	Heat Generation
4540	France	2002	Oil	31103.0
4541	France	2003	Oil	26084.0
4542	France	2004	Oil	28291.0
4543	France	2005	Oil	28537.0
4544	France	2006	Oil	30793.0
4545	France	2007	Oil	21924.0
4546	France	2008	Oil	15617.0
4547	France	2009	Oil	15870.0
4548	France	2010	Oil	19553.0
4549	France	2011	Oil	12853.306
4550	France	2012	Oil	10017.386
4551	France	2013	Oil	12917.813
4552	France	2014	Oil	9467.315999999999
4553	France	2015	Oil	9547.687
4554	France	2016	Oil	8742.557
4555	France	2017	Oil	10127.785
4556	France	2018	Oil	1421.585
4557	France	2019	Oil	868.124
4558	France	2020	Oil	654.1110000000001
4559	France	2000	Natural gas	59480.0
4560	France	2001	Natural gas	86612.0
4561	France	2002	Natural gas	97858.0

2.5. Colors and data

Z-2.5.1. Depending on the chart considered, I used a gradient of color or defined colors in an array because of a low number of countries that I wanted to highlight.

For some chart, the user can move his mouse over some part of the chart to have more information about the country like the value displayed. Moreover, the user can for the heat map increase or decrease the year considered and without updating the URL, the heat map will be updated.

3. KV3 - Creating prototype data visualization.

3.1. Basic functionalities and behaviors

Z-3.1.1. The dataset must be loaded in the html document every time the corresponding chart has to be loaded. This dataset will be parsed and then the data will be organized according to what the chart require.

The data must be displayed clearly and effectively, this includes legend, labels, and title.

A tooltip will be setup to display more information about the visualization when the user overs his mouth to the chart.

Z-3.1.2. Basic type of behavior will be the use of tooltip, animation regarding the user visualizing the data over the year.

Z-3.1.3. For some charts, the user will be able to have more information about the data displayed by overing the mouse into the chart. Information such as year, country or value will be displayed.

For the heat map, the user will have a scroll bar at his disposal to go through the year and see.

3.2. Advanced functionalities and behaviors:

Z-3.2.1. Statical information such as range, minimum, maximum (possibility to display mean and standard deviation) about the data will be displayed for each chart. These metrics provide an overview of the central tendency, dispersion, and distribution of the data.

Labels or data values directly on the chart elements to provide specific numerical information for each data point. This helps users understand the exact values associated with the chart elements, facilitating analysis and comparison.

Utilize tooltips that appear when hovering over chart elements to display additional statistical information for each data point. This could include specific values, percentages, or statistical measures relevant to the chart's context.

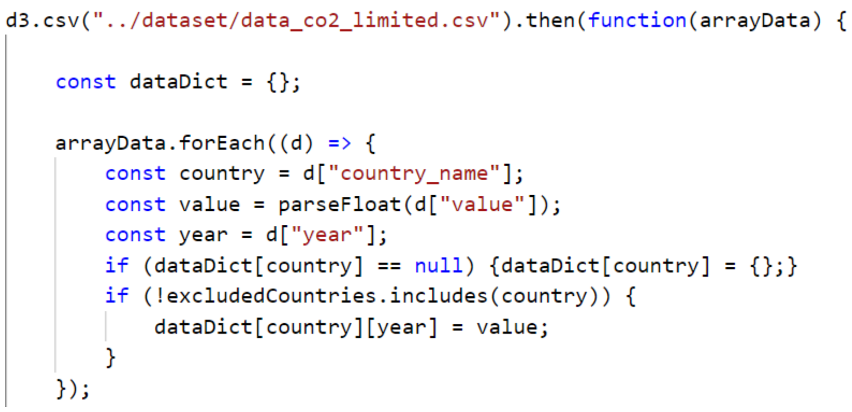
Thanks to all the chart, a comparative analysis is made and will highlight for the user what he should be aware.

Z-3.2.2. Some charts that can be updated will support real-time data updates and year update.

Z-3.2.3. The tooltip will help the user to analyze the data.

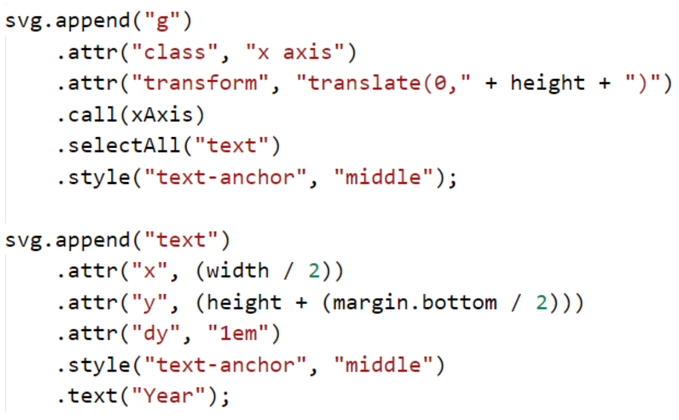
3.3. Implementation of basic functionalities

Z-3.3.1. -For parsing the data, *d3.csv* will be user and then, each column will be added into a dictionary. For example, for co2 emissions, here is the code for parsing and here is the corresponding object:



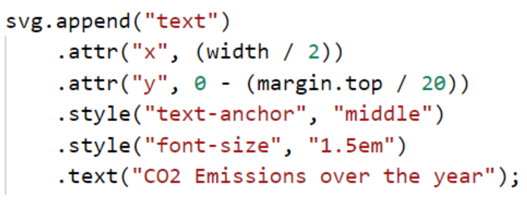
```
d3.csv("../dataset/data_co2_limited.csv").then(function(arrayData) {

    const dataDict = {};

    arrayData.forEach((d) => {
        const country = d["country_name"];
        const value = parseFloat(d["value"]);
        const year = d["year"];
        if (dataDict[country] == null) {dataDict[country] = {};}
        if (!excludedCountries.includes(country)) {
            dataDict[country][year] = value;
        }
    });
```

-For clear presentation of the data, here is how the labels, legends and title are implemented:



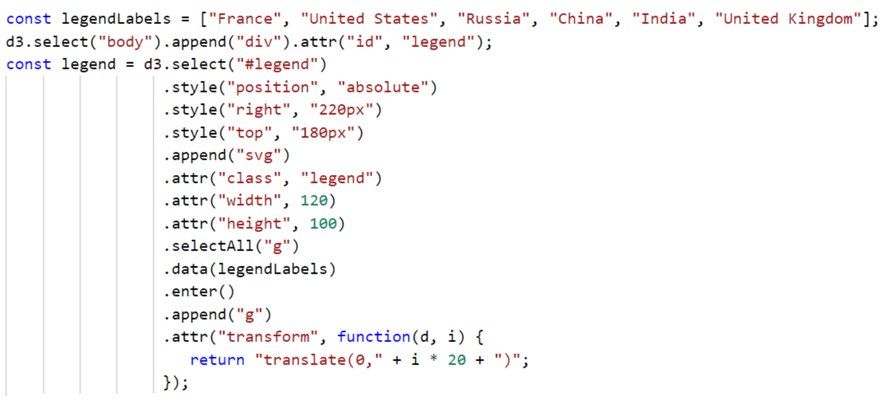
```
svg.append("g")
    .attr("class", "x axis")
    .attr("transform", "translate(0," + height + ")")
    .call(xAxis)
    .selectAll("text")
    .style("text-anchor", "middle");

svg.append("text")
    .attr("x", (width / 2))
    .attr("y", (height + (margin.bottom / 2)))
    .attr("dy", "1em")
    .style("text-anchor", "middle")
    .text("Year");
```

Implementation of the x-axis



```
svg.append("text")
    .attr("x", (width / 2))
    .attr("y", 0 - (margin.top / 20))
    .style("text-anchor", "middle")
    .style("font-size", "1.5em")
    .text("CO2 Emissions over the year");
```

Implementation of the title



```
const legendLabels = ["France", "United States", "Russia", "China", "India", "United Kingdom"];
d3.select("body").append("div").attr("id", "legend");
const legend = d3.select("#legend")
                .style("position", "absolute")
                .style("right", "220px")
                .style("top", "180px")
                .append("svg")
                .attr("class", "legend")
                .attr("width", 120)
                .attr("height", 100)
                .selectAll("g")
                .data(legendLabels)
                .enter()
                .append("g")
                .attr("transform", function(d, i) {
                    return "translate(0," + i * 20 + ")";
                });
```

Implementation of the legend, after this execution, a rectangle is created and is filled with 3 values: min, mean, max.

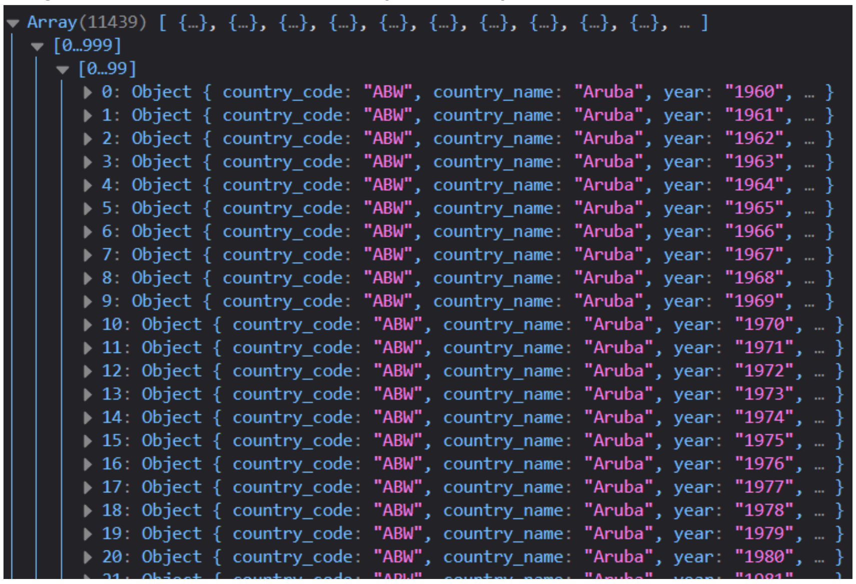
For the scroll bar of the heat map, here is the implementation:

```
const drag = d3.drag().on("drag", function (event) {
    const x = event.x;
    if (x >= 0 && x <= 300) {
      const value = Math.round((x / 300) * (maxValue - initialValue) + initialValue);
      document.getElementById("value").textContent = value;
      d3.select(this).style("margin-left", `${x}px`);
      update();
    }
  });

button.call(drag);
```

When the scroll bar is created, it is added to a lign element and then calculated given the initial position and ending position the value of the element that contain the id = "value" which will inform the update() function because this value has been updated.

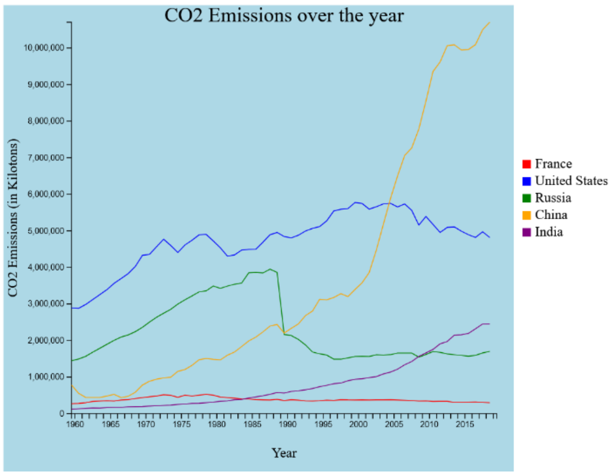
Z-3.3.2. Cf. previous pictures.

-For parsing the data, here is the arrayData object:

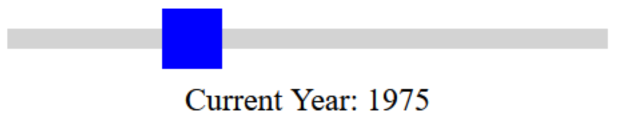


```
▼ Array(11439) [ {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, … ]
  ▼ [0…999]
    ▼ [0…99]
      ▶ 0: Object { country_code: "ABW", country_name: "Aruba", year: "1960", … }
      ▶ 1: Object { country_code: "ABW", country_name: "Aruba", year: "1961", … }
      ▶ 2: Object { country_code: "ABW", country_name: "Aruba", year: "1962", … }
      ▶ 3: Object { country_code: "ABW", country_name: "Aruba", year: "1963", … }
      ▶ 4: Object { country_code: "ABW", country_name: "Aruba", year: "1964", … }
      ▶ 5: Object { country_code: "ABW", country_name: "Aruba", year: "1965", … }
      ▶ 6: Object { country_code: "ABW", country_name: "Aruba", year: "1966", … }
      ▶ 7: Object { country_code: "ABW", country_name: "Aruba", year: "1967", … }
      ▶ 8: Object { country_code: "ABW", country_name: "Aruba", year: "1968", … }
      ▶ 9: Object { country_code: "ABW", country_name: "Aruba", year: "1969", … }
      ▶ 10: Object { country_code: "ABW", country_name: "Aruba", year: "1970", … }
      ▶ 11: Object { country_code: "ABW", country_name: "Aruba", year: "1971", … }
      ▶ 12: Object { country_code: "ABW", country_name: "Aruba", year: "1972", … }
      ▶ 13: Object { country_code: "ABW", country_name: "Aruba", year: "1973", … }
      ▶ 14: Object { country_code: "ABW", country_name: "Aruba", year: "1974", … }
      ▶ 15: Object { country_code: "ABW", country_name: "Aruba", year: "1975", … }
      ▶ 16: Object { country_code: "ABW", country_name: "Aruba", year: "1976", … }
      ▶ 17: Object { country_code: "ABW", country_name: "Aruba", year: "1977", … }
      ▶ 18: Object { country_code: "ABW", country_name: "Aruba", year: "1978", … }
      ▶ 19: Object { country_code: "ABW", country_name: "Aruba", year: "1979", … }
      ▶ 20: Object { country_code: "ABW", country_name: "Aruba", year: "1980", … }
```

-For legends, labels and title, here is the basic page of a line chart:

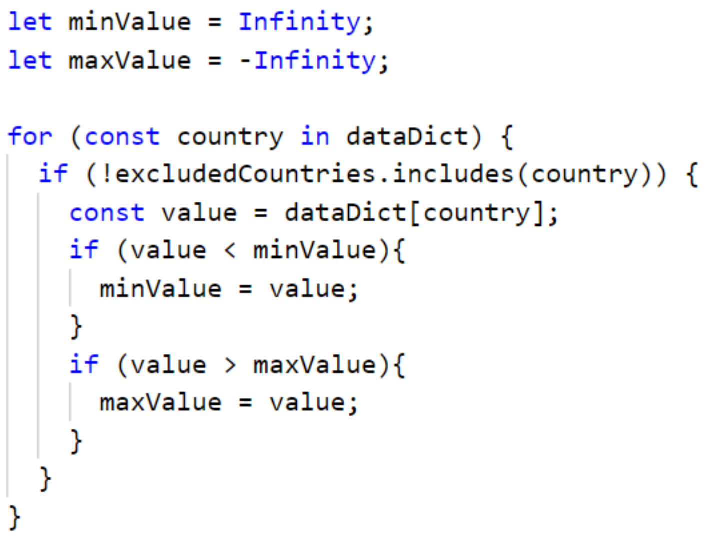


-For the scroll bar, here is a picture of the scroll bar used for the heatmap:



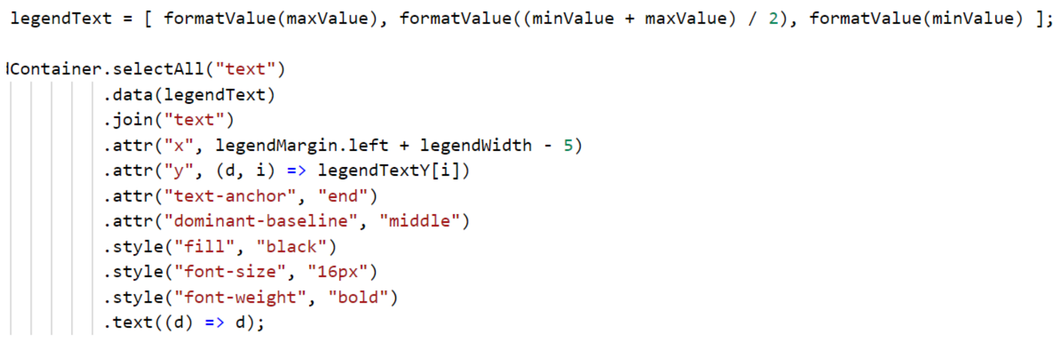
3.4. Implementation of basic behavior

Z-3.4.1. -To display statistical information about the data while rendering a good gradient legend, I decided to merge both features into one. First, it is important to find the minimum and maximum values for the dataset. Once this is done, it is needed to display it in the rectangular legend.



```
let minValue = Infinity;
let maxValue = -Infinity;

for (const country in dataDict) {
  if (!excludedCountries.includes(country)) {
    const value = dataDict[country];
    if (value < minValue){
      minValue = value;
    }
    if (value > maxValue){
      maxValue = value;
    }
  }
}
```

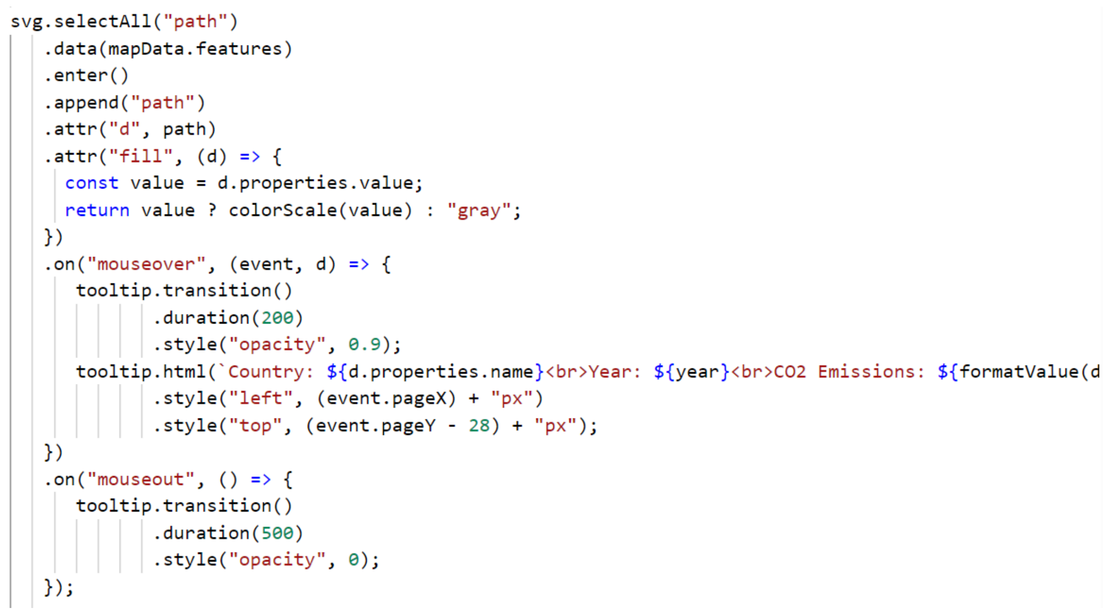
Implementation of research of a min and max value for the dataset (with a year that has been set by the user).



```
legendText = [ formatValue(maxValue), formatValue((minValue + maxValue) / 2), formatValue(minValue) ];

lContainer.selectAll("text")
          .data(legendText)
          .join("text")
          .attr("x", legendMargin.left + legendWidth - 5)
          .attr("y", (d, i) => legendTextY[i])
          .attr("text-anchor", "end")
          .attr("dominant-baseline", "middle")
          .style("fill", "black")
          .style("font-size", "16px")
          .style("font-weight", "bold")
          .text((d) => d);
```

Now, the formatValue() function will allow data to be displayed with a shorter version (using Giga, Mega and Kilo). Then, the legendContainer is a rectangle and the text is added to it.

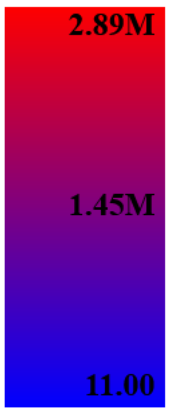
-For the tooltip, here is implementation used for the heatmap:



```
svg.selectAll("path")
    .data(mapData.features)
    .enter()
    .append("path")
    .attr("d", path)
    .attr("fill", (d) => {
      const value = d.properties.value;
      return value ? colorScale(value) : "gray";
    })
    .on("mouseover", (event, d) => {
      tooltip.transition()
              .duration(200)
              .style("opacity", 0.9);
      tooltip.html(`Country: ${d.properties.name}<br>Year: ${year}<br>CO2 Emissions: ${formatValue(d
              .style("left", (event.pageX) + "px")
              .style("top", (event.pageY - 28) + "px");
    })
    .on("mouseout", () => {
      tooltip.transition()
              .duration(500)
              .style("opacity", 0);
    });
```

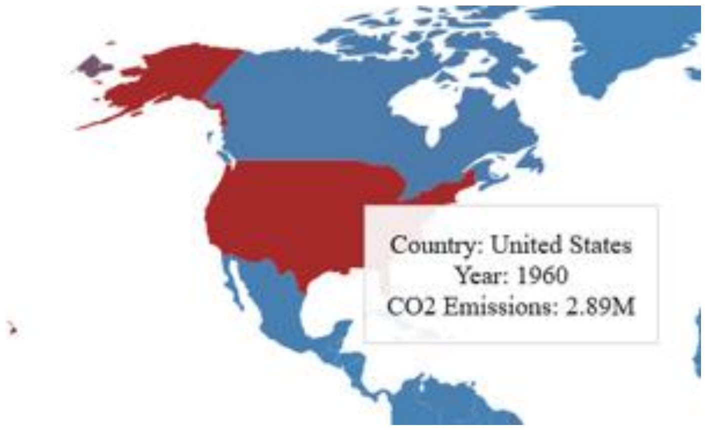
The tooltip will display information such as name of the country, year, value for the country overed by the user's mouse.

Z-3.4.2. -For statistical information about data and labels:



At the top, it the max value, then the central point and at the bottom the min value.

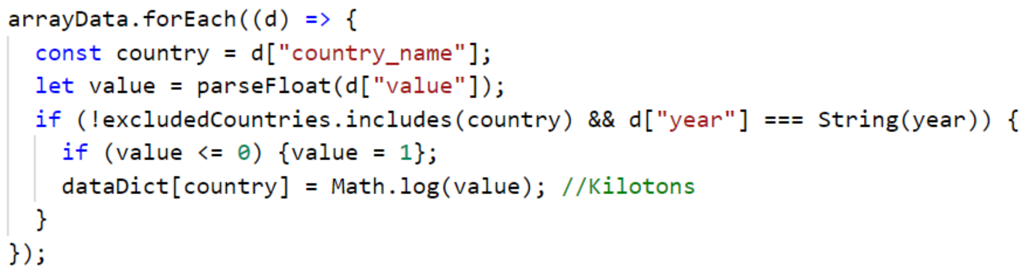
-For the tooltip, here is the final visualization for the heatmap:



4. KV4 - Creating the final data visualization

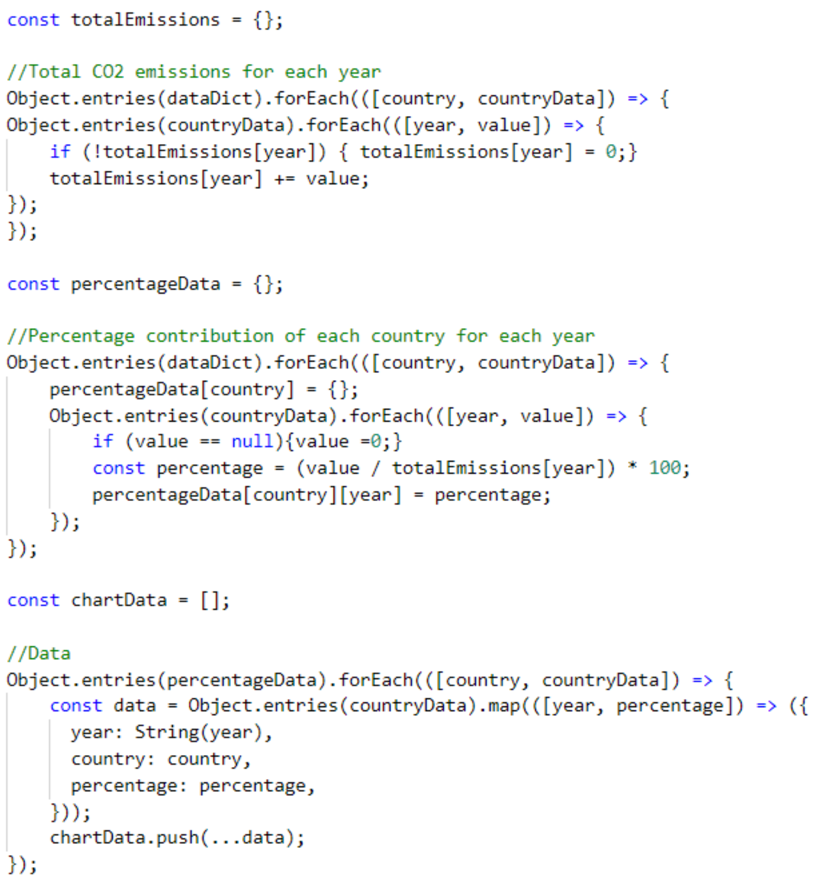
4.1. Implementation of basic functionalities

Z-4.1.1. -For parsing the data, *d3.csv* will be user and then, each column will be added into a dictionary. For example, for co2 emissions, I changed the data in the dictionary by adding a log function that will decrease the gradient of color and colorized more the heat map. It is relevant because of how the distribution of the data is (China and United State of America have values that are higher than the other).



```
arrayData.forEach((d) => {
  const country = d["country_name"];
  let value = parseFloat(d["value"]);
  if (!excludedCountries.includes(country) && d["year"] === String(year)) {
    if (value <= 0) {value = 1};
    dataDict[country] = Math.log(value); //Kilotons
  }
});
```

-To plot the stacked chart, the data needed to be prepared. There were a need to know the percentage of contribution for each country and calculate the total amount of emission for each year. Once that was done, the data needed to be a dictionnary with a first key about year. The entries of each element after using the key of the year would be the name of the country following by the percentage.

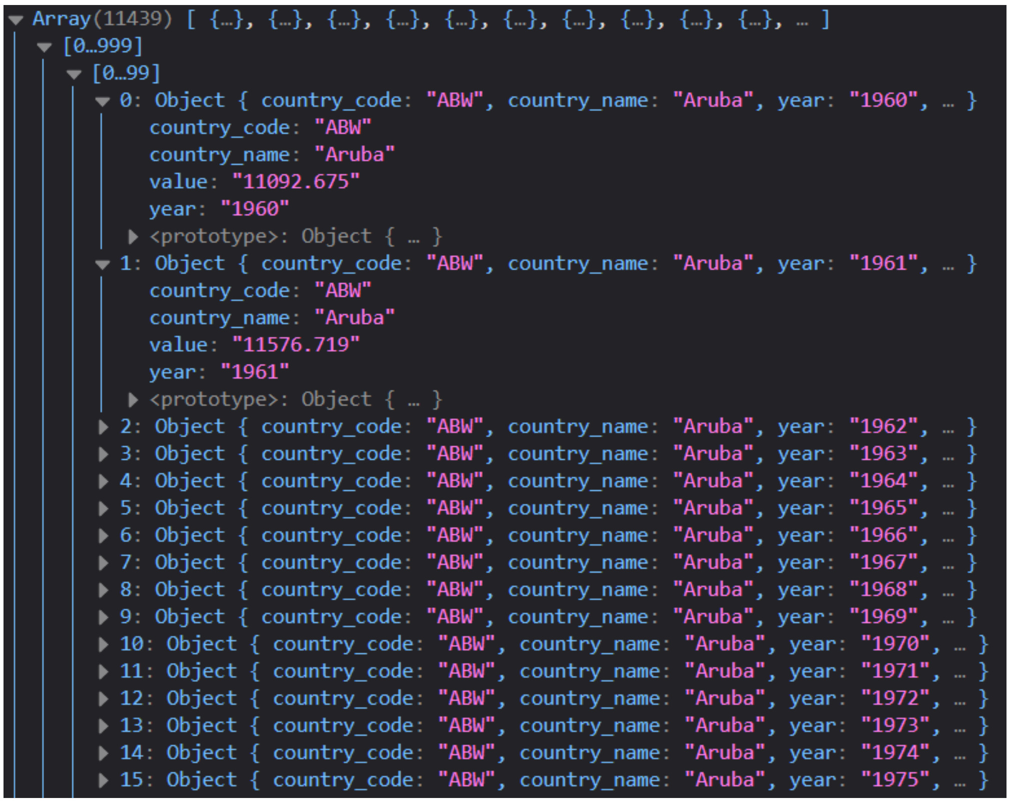


```
const totalEmissions = {};

//Total CO2 emissions for each year
Object.entries(dataDict).forEach(([country, countryData]) => {
Object.entries(countryData).forEach(([year, value]) => {
    if (!totalEmissions[year]) { totalEmissions[year] = 0;}
    totalEmissions[year] += value;
});
});

const percentageData = {};

//Percentage contribution of each country for each year
Object.entries(dataDict).forEach(([country, countryData]) => {
    percentageData[country] = {};
    Object.entries(countryData).forEach(([year, value]) => {
        if (value == null){value =0;}
        const percentage = (value / totalEmissions[year]) * 100;
        percentageData[country][year] = percentage;
    });
});

const chartData = [];

//Data
Object.entries(percentageData).forEach(([country, countryData]) => {
    const data = Object.entries(countryData).map(([year, percentage]) => ({
      year: String(year),
      country: country,
      percentage: percentage,
    }));
    chartData.push(...data);
});
```

And then a group by year:

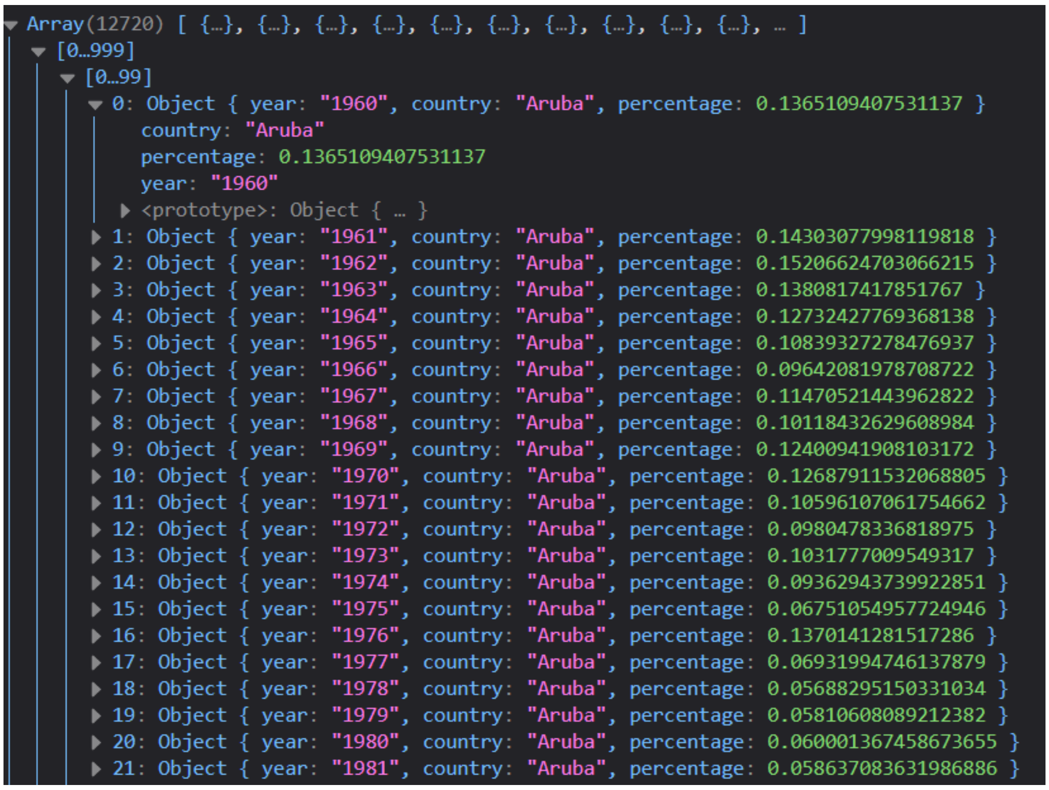
```
const sumstat = d3.group(chartData, d => d.year);
```

Z-4.1.2. Cf. previous pictures.
-For parsing the data, here is the arrayData object:



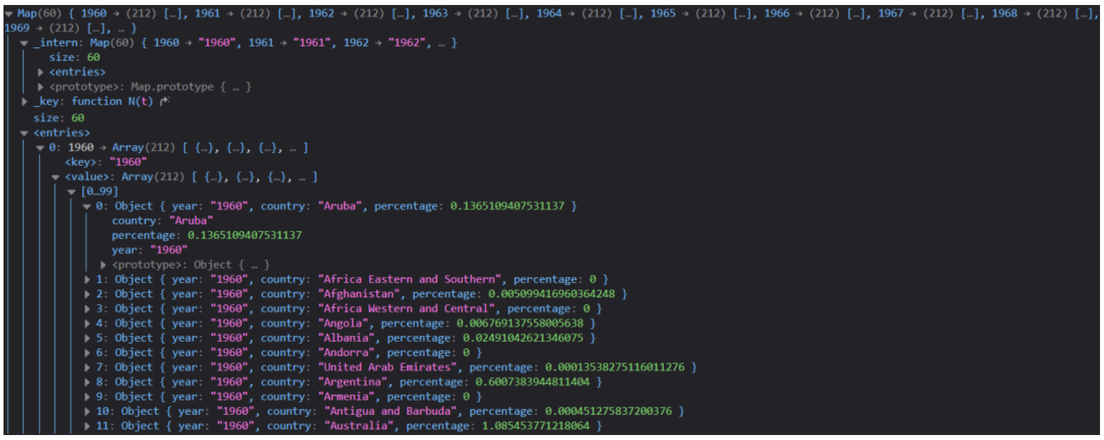
```
▼ Array(11439) [ {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, … ]
  ▼ [0…999]
    ▼ [0…99]
      ▼ 0: Object { country_code: "ABW", country_name: "Aruba", year: "1960", … }
          country_code: "ABW"
          country_name: "Aruba"
          value: "11092.675"
          year: "1960"
        ▶ <prototype>: Object { … }
      ▼ 1: Object { country_code: "ABW", country_name: "Aruba", year: "1961", … }
          country_code: "ABW"
          country_name: "Aruba"
          value: "11576.719"
          year: "1961"
        ▶ <prototype>: Object { … }
      ▶ 2: Object { country_code: "ABW", country_name: "Aruba", year: "1962", … }
      ▶ 3: Object { country_code: "ABW", country_name: "Aruba", year: "1963", … }
      ▶ 4: Object { country_code: "ABW", country_name: "Aruba", year: "1964", … }
      ▶ 5: Object { country_code: "ABW", country_name: "Aruba", year: "1965", … }
      ▶ 6: Object { country_code: "ABW", country_name: "Aruba", year: "1966", … }
      ▶ 7: Object { country_code: "ABW", country_name: "Aruba", year: "1967", … }
      ▶ 8: Object { country_code: "ABW", country_name: "Aruba", year: "1968", … }
      ▶ 9: Object { country_code: "ABW", country_name: "Aruba", year: "1969", … }
      ▶ 10: Object { country_code: "ABW", country_name: "Aruba", year: "1970", … }
      ▶ 11: Object { country_code: "ABW", country_name: "Aruba", year: "1971", … }
      ▶ 12: Object { country_code: "ABW", country_name: "Aruba", year: "1972", … }
      ▶ 13: Object { country_code: "ABW", country_name: "Aruba", year: "1973", … }
      ▶ 14: Object { country_code: "ABW", country_name: "Aruba", year: "1974", … }
      ▶ 15: Object { country_code: "ABW", country_name: "Aruba", year: "1975", … }
```

-For parsing the data to create a stacked chart, here is the chartData variable:



```
▼ Array(12720) [ {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, … ]
  ▼ [0…999]
    ▼ [0…99]
      ▼ 0: Object { year: "1960", country: "Aruba", percentage: 0.1365109407531137 }
          country: "Aruba"
          percentage: 0.1365109407531137
          year: "1960"
        ▶ <prototype>: Object { … }
      ▶ 1: Object { year: "1961", country: "Aruba", percentage: 0.14303077998119818 }
      ▶ 2: Object { year: "1962", country: "Aruba", percentage: 0.15206624703066215 }
      ▶ 3: Object { year: "1963", country: "Aruba", percentage: 0.1380817417851767 }
      ▶ 4: Object { year: "1964", country: "Aruba", percentage: 0.12732427769368138 }
      ▶ 5: Object { year: "1965", country: "Aruba", percentage: 0.10839327278476937 }
      ▶ 6: Object { year: "1966", country: "Aruba", percentage: 0.09642081978708722 }
      ▶ 7: Object { year: "1967", country: "Aruba", percentage: 0.11470521443962822 }
      ▶ 8: Object { year: "1968", country: "Aruba", percentage: 0.10118432629608984 }
      ▶ 9: Object { year: "1969", country: "Aruba", percentage: 0.12400941908103172 }
      ▶ 10: Object { year: "1970", country: "Aruba", percentage: 0.12687911532068805 }
      ▶ 11: Object { year: "1971", country: "Aruba", percentage: 0.10596107061754662 }
      ▶ 12: Object { year: "1972", country: "Aruba", percentage: 0.0980478336818975 }
      ▶ 13: Object { year: "1973", country: "Aruba", percentage: 0.1031777009549317 }
      ▶ 14: Object { year: "1974", country: "Aruba", percentage: 0.09362943739922851 }
      ▶ 15: Object { year: "1975", country: "Aruba", percentage: 0.06751054957724946 }
      ▶ 16: Object { year: "1976", country: "Aruba", percentage: 0.1370141281517286 }
      ▶ 17: Object { year: "1977", country: "Aruba", percentage: 0.06931994746137879 }
      ▶ 18: Object { year: "1978", country: "Aruba", percentage: 0.05688295150331034 }
      ▶ 19: Object { year: "1979", country: "Aruba", percentage: 0.05810608089212382 }
      ▶ 20: Object { year: "1980", country: "Aruba", percentage: 0.060001367458673655 }
      ▶ 21: Object { year: "1981", country: "Aruba", percentage: 0.05863708363198688 }
```

And here the group by year, the sumstat variable:



```
▼ Map(60) { 1960 → (212) […], 1961 → (212) […], 1962 → (212) […], 1963 → (212) […], 1964 → (212) […], 1965 → (212) […], 1966 → (212) […], 1967 → (212) […], 1968 → (212) […],
1969 → (212) […], … }
  ▼ _intern: Map(60) { 1960 → "1960", 1961 → "1961", 1962 → "1962", … }
      size: 60
    ▶ <entries>
    ▶ <prototype>: Map.prototype { … }
  ▶ _key: function N(t)
    size: 60
  ▼ <entries>
    ▼ 0: 1960 → Array(212) [ {…}, {…}, {…}, … ]
        <key>: "1960"
      ▼ <value>: Array(212) [ {…}, {…}, {…}, … ]
        ▼ [0…99]
          ▼ 0: Object { year: "1960", country: "Aruba", percentage: 0.1365109407531137 }
              country: "Aruba"
              percentage: 0.1365109407531137
              year: "1960"
            ▶ <prototype>: Object { … }
          ▶ 1: Object { year: "1960", country: "Africa Eastern and Southern", percentage: 0 }
          ▶ 2: Object { year: "1960", country: "Afghanistan", percentage: 0.005099416960364248 }
          ▶ 3: Object { year: "1960", country: "Africa Western and Central", percentage: 0 }
          ▶ 4: Object { year: "1960", country: "Angola", percentage: 0.006769137558005638 }
          ▶ 5: Object { year: "1960", country: "Albania", percentage: 0.02491042621346075 }
          ▶ 6: Object { year: "1960", country: "Andorra", percentage: 0 }
          ▶ 7: Object { year: "1960", country: "United Arab Emirates", percentage: 0.00013538275116011276 }
          ▶ 8: Object { year: "1960", country: "Argentina", percentage: 0.6007383944811404 }
          ▶ 9: Object { year: "1960", country: "Armenia", percentage: 0 }
          ▶ 10: Object { year: "1960", country: "Antigua and Barbuda", percentage: 0.000451275837200376 }
          ▶ 11: Object { year: "1960", country: "Australia", percentage: 1.085453771218064 }
```

4.2. Implementation of basic behavior

Z-4.2.1. -For clear presentation of the data, 2 values were added: mean and std of the data considered. This is useful for the user to understand how the production or emission is different for region of the world and country.

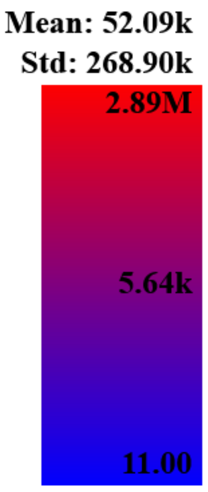
```
let minValue = Infinity;
let maxValue = -Infinity;
let sum = 0;
let squaredDiffSum = 0;
let count = 0;

for (const country in dataDict) {
  if (!excludedCountries.includes(country)) {
    const value = dataDict[country];
    if (value < minValue) {
      minValue = value;
    }
    if (value > maxValue) {
      maxValue = value;
    }
    sum += Math.exp(value);
    count++;
  }
}

const mean = sum / count;

for (const country in dataDict) {
  if (!excludedCountries.includes(country)) {
    const value = dataDict[country];
    squaredDiffSum += (Math.exp(value) - mean) ** 2;
  }
}

const std = Math.sqrt(squaredDiffSum / count);
```

```
const legendContainer = d3.select("body").select("div").select("svg.legend-svg");
const legendTextY = [legendMargin.top - 30, legendMargin.top - 10, legendMargin.top + 10, (legendHeight / 2) + legendMargin.top , legendHeight + legendMargin.top - 10]
const legendText = ["Mean: " + formatValueBis(mean), "Std: " + formatValueBis(std), formatValue(maxValue), formatValue((minValue + maxValue) / 2), formatValue(minValue

legendContainer.selectAll("text")
               .data(legendText)
               .join("text")
               .attr("x", legendMargin.left + legendWidth - 5 )
               .attr("y", (d, i) => legendTextY[i])
               .attr("text-anchor", "end")
               .attr("dominant-baseline", "middle")
               .style("fill", "black")
               .style("font-size", "16px")
               .style("font-weight", "bold")
               .text((d) => d);
```

Now, the formatValue() function will allow data to be displayed with a shorter version (using Giga, Mega and Kilo). Then, the legendContainer is a rectangle colorized with a gradient of color and the text is added to it.

Z-4.2.2. At the top, it the max value, then the central point and at the bottom the min value.

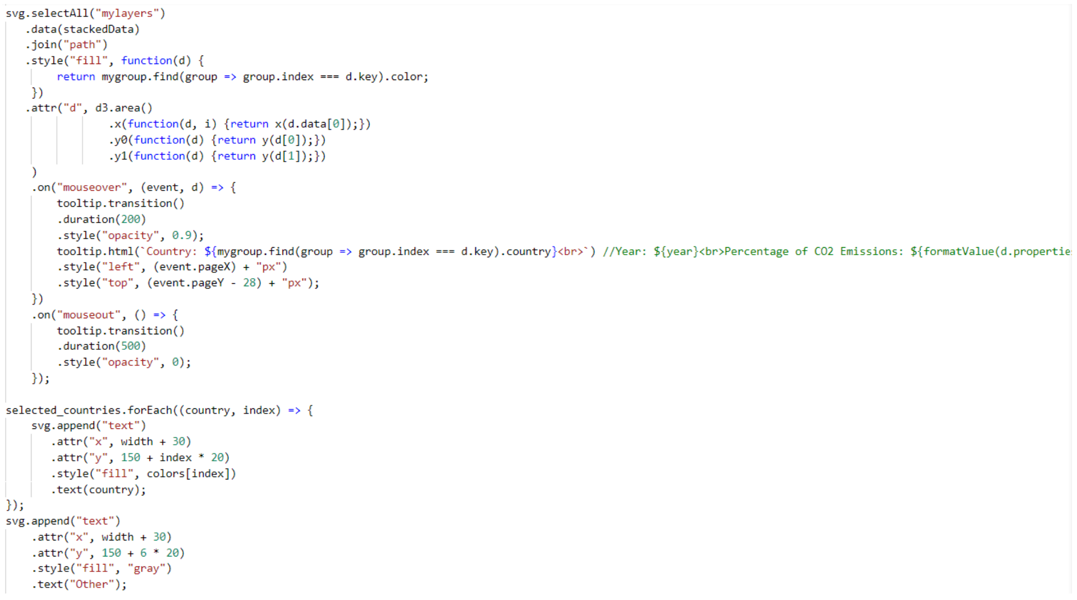
On top of the rectangle, the mean value and the std value are added.



4.3. Implementation of advanced functionality

Z-4.3.1. For the stacked chart, an interaction with the user would be possible, letting him choose which country he wants to see. But, to make that possible without implementing it, the tooltip will be enough because the user will be able to identify which country it is.

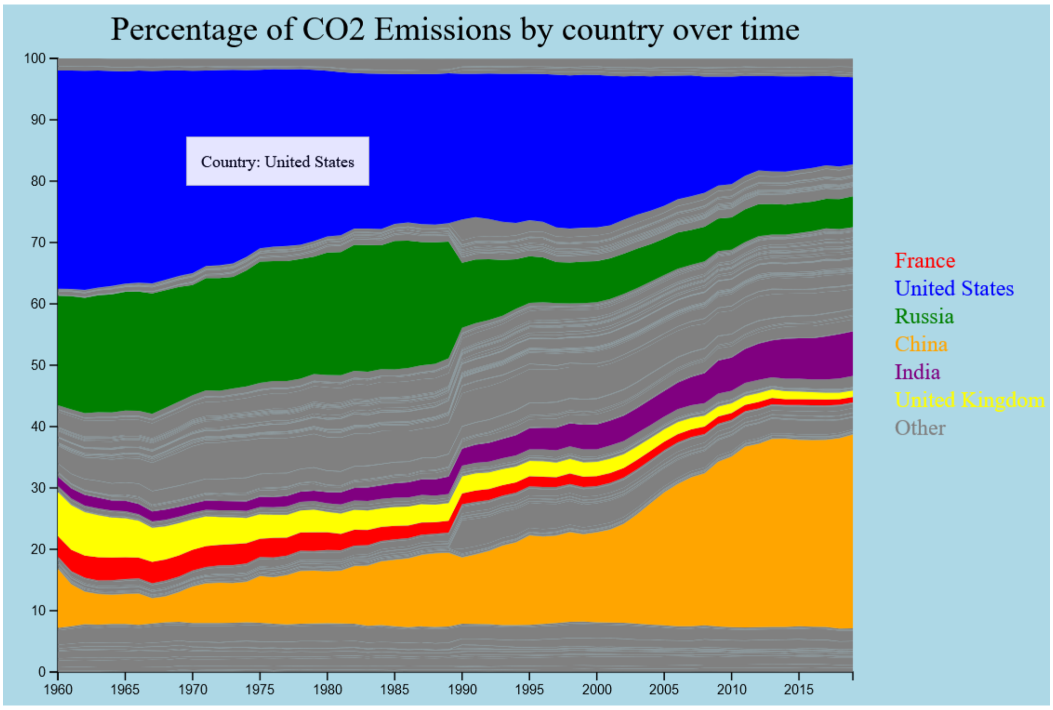
Z-4.3.2. To implement this tooltip, here is the code:



```
svg.selectAll("mylayers")
   .data(stackedData)
   .join("path")
   .style("fill", function(d) {
       return mygroup.find(group => group.index === d.key).color;
    })
   .attr("d", d3.area()
             .x(function(d, i) {return x(d.data[0]);})
             .y0(function(d) {return y(d[0]);})
             .y1(function(d) {return y(d[1]);})
    )
    .on("mouseover", (event, d) => {
        tooltip.transition()
        .duration(200)
        .style("opacity", 0.9);
        tooltip.html(`Country: ${mygroup.find(group => group.index === d.key).country}<br>`) //Year: ${year}<br>Percentage of CO2 Emissions: ${formatValue(d.propertie
        .style("left", (event.pageX) + "px")
        .style("top", (event.pageY - 28) + "px");
    })
    .on("mouseout", () => {
        tooltip.transition()
        .duration(500)
        .style("opacity", 0);
    });

selected_countries.forEach((country, index) => {
    svg.append("text")
       .attr("x", width + 30)
       .attr("y", 150 + index * 20)
       .style("fill", colors[index])
       .text(country);
});
svg.append("text")
    .attr("x", width + 30)
    .attr("y", 150 + 6 * 20)
    .style("fill", "gray")
    .text("Other");
```

This code implements the legend too, and add information about the year and country overed.

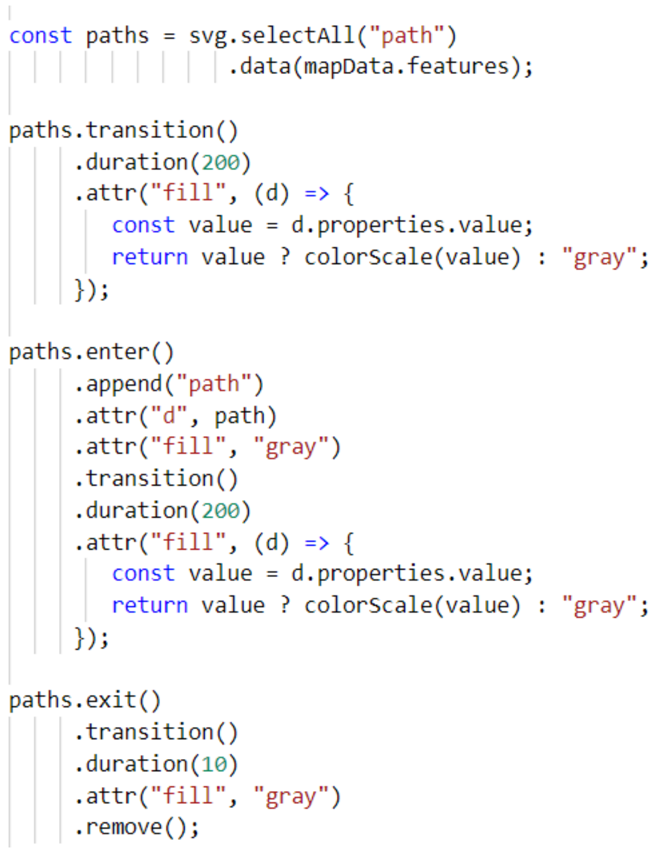
Z-4.3.3. Here is the view from the user:



4.4. Implementing advanced behavior

Z-4.4.1. For the heatmap, when the user uses the scroll bar, the map will be updated in real-time, this is Real-time Data Processing, it enables the system to process and analyze data in real-time, allowing for immediate insights and responses.

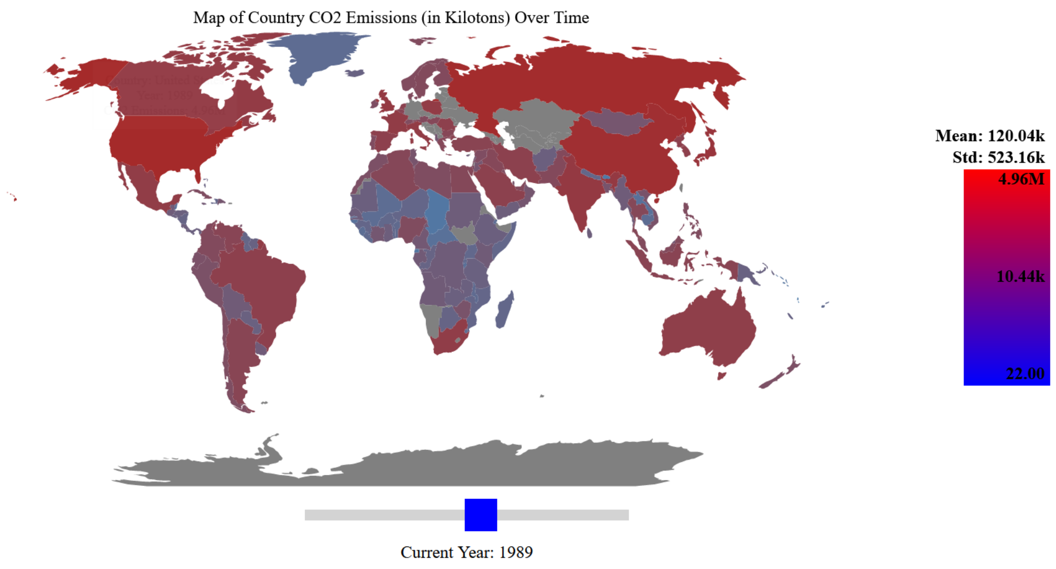
Z-4.4.2. Here is the implementation of the previous behavior:

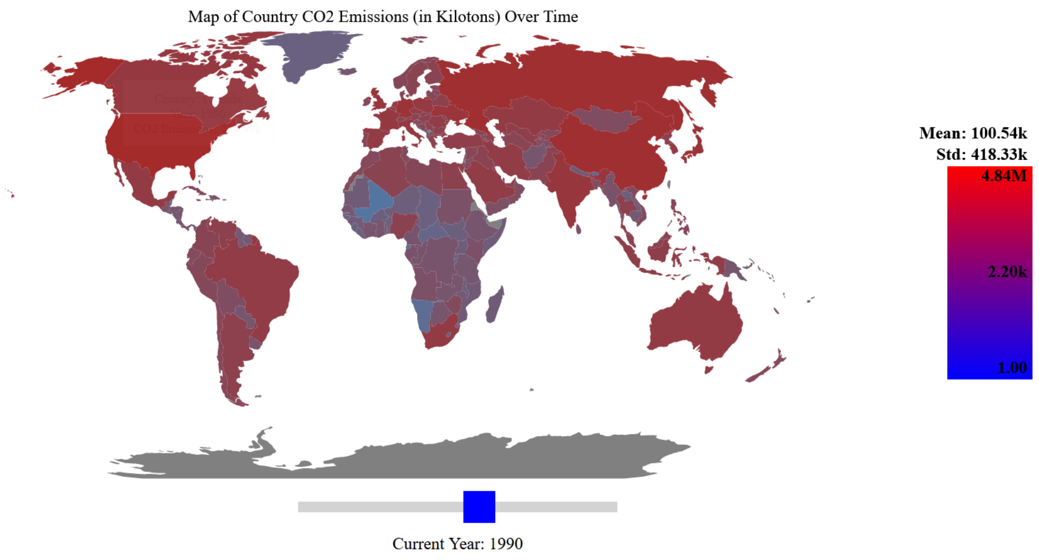


```
const paths = svg.selectAll("path")
                 .data(mapData.features);

paths.transition()
    .duration(200)
    .attr("fill", (d) => {
      const value = d.properties.value;
      return value ? colorScale(value) : "gray";
    });

paths.enter()
    .append("path")
    .attr("d", path)
    .attr("fill", "gray")
    .transition()
    .duration(200)
    .attr("fill", (d) => {
      const value = d.properties.value;
      return value ? colorScale(value) : "gray";
    });

paths.exit()
    .transition()
    .duration(10)
    .attr("fill", "gray")
    .remove();
```

All the paths are selected and then, a transition() and an exit() function are realized to make the transition smooth.

Z-4.4.3. Pictures between transition of year with the use of the scroll bar:





5. KV5 - Completing the project task and writing documentation

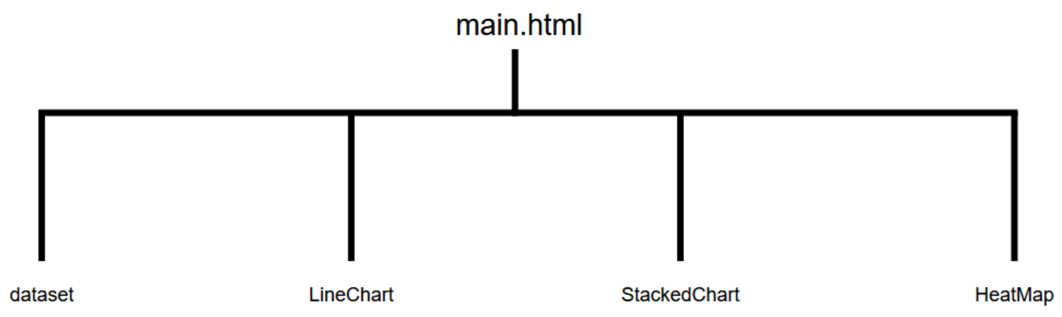
5.1. Possible modifications and refinements of the solution - in agreement with the teacher

[This task refers to potential changes and refinements that need to be made on the solution of the terms of reference, which are agreed with the teacher. It is possible that it is necessary to change some functionalities, corrections in the code or any other finishing to ensure a quality and complete solution.]

5.2. Preparation of documents - project documentation

[In this task, it is necessary to prepare project documentation that will describe the process of preparation and achieved results of the terms of reference. Project documentation usually includes a description of the project task, the necessary tools, the work process, a description of the design and visualization of data, reports on conducted tests and results, conclusion and the like. The goal is to keep the documentation clear, detailed and complete so that others can understand and use your solution.]

Z-5.2.1. Project hierarchy is the following:



All the branches are directories.

The dataset directory contains the csv files.

All the other contains two graphs (two html file) about the name of their directory. One for CO2 emissions and one for renewable energy production.

Z-5.2.2. List of technologies used:

- JavaScript (D3.js)
- HTML
- CSS
- Python
- JSON
- CSV
- Excel

Z-5.2.3. For example, the launch project from Visual Studio Code is simple to install and will be enough to launch the project.

Z-5.2.4. To use the project, it is needed to launch a local project. The file to launch is “*main.html*”.

Literature

[1] Global Carbon Atlas, accessed: June 17, 2023
[2] Country CO2 Emission plots, accessed: June 18, 2023
[3] Renewable Energy World Wide: 1965~2022, accessed: June 18, 2023

Annex I

Link to git repository of the project: https://github.com/TheMysters/VD-Project

Program code