Design Process

In this document, I first provide a summary of my design process followed by challenges I faced and key insights gathered from the data analysis. I first sketched out the initial layout of the webpage.

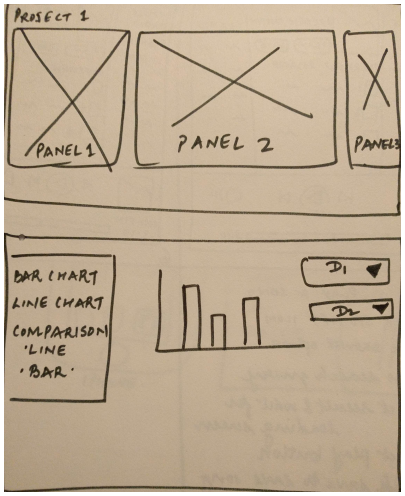

Fig 1. Sketch of basic UI layout

The project requirements provided the direction for the UI design. For instance, the requirement to showcase the different types of charts: bar chart, line chart, comparison of charts highlight that the user should be given the ability to switch between the charts. Following are the reasons for my design decision: -

a. Since both the panels are placed side-by-side, users can quickly shift their attention to both the panels. For example, when the user selects the bar chart option on the left panel, the associated the bar chart is displayed on the right panel for any given region.
b. The design follows other commonly used applications such as Windows Explorer, therefore reducing the users cognitive load of learning a new interface.
c. Since the user is presented with the options at once, they simply have to look at the options rather than remember, following closely with Neilsen's heuristics "Recognition rather than recall."

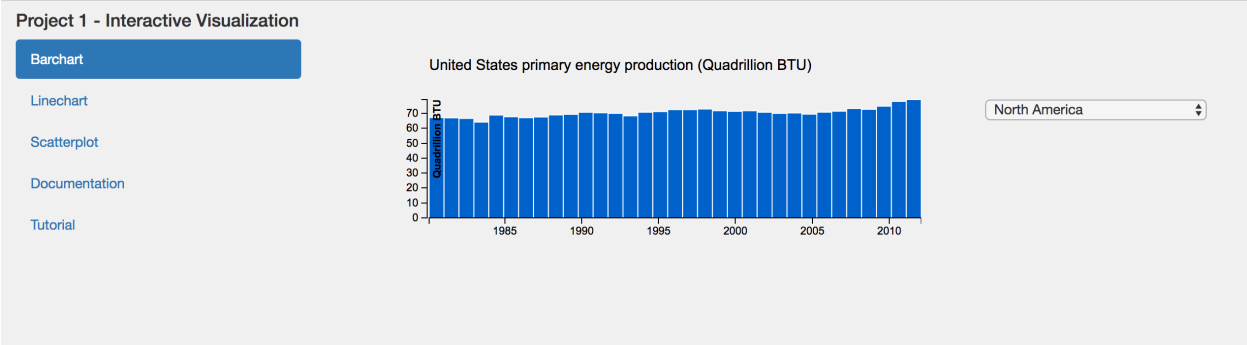

Fig 1. The options on the left follow other common websites to leverage user's familiarity with the interface.

My design process can be broken down into three steps. First, sketch the layout of the webpage. Second, implement the front page of the design without delving too much into the logic. Third, iterate within constraints.

***Step 1**: Sketch the Main Page*

Initially the idea of the main page was to include the basic components of the page. Some of the components included a dropdown to allow the user to switch between the different charts, a dropdown for the different types of energy consumption for a particular region and so on. The selection for the different types of energy production/consumption and regions were placed on the right of the chart area. However, the user had to move their cursor from the extreme left to the extreme right to switch between the regions. To eliminate this additional time, I placed the dropdowns and categorized all the user selections to one area of the UI i.e. left panel.

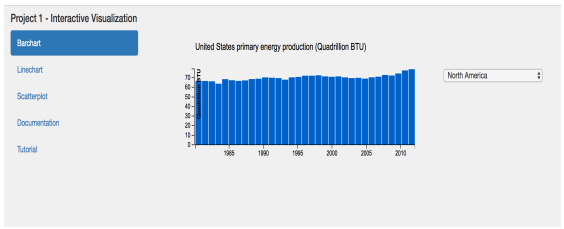

Fig 2a. Dropdown to the right panel of the UI of the UI

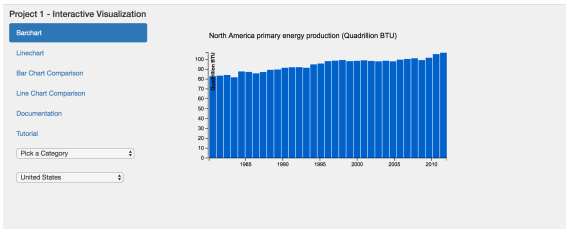

Fig 2b. Re-arranged dropdown to the left panel

***Step 2**: Implement Designs*

Writing the line chart function was relatively simple, parsing the data from various files to populate the dropdown buttons was another challenge altogether. I took several approaches to tackle this challenge. One included parsing the files based on if-else-if condition. However, I ran into several problems with it. Some of which included values not being read when the user selected a category from the dropdown. Although these conditions were written within the *d3.onchange()* function, the problem persisted. So I took another approach of separating the parsing logic into a separate .js file and referred to it onPage load. However, even with this approach only the first value from the dropdown got plotted. So finally, I went back to my first approach. In this iteration, however, I separated the dropdown selection logic into a separate function and referred to this function in the Papa parse logic. With this strategy, I got to load all the 13 variables on the page based on the user selection.

```
    d3.select("#selectionWidget")
    .on('change', function() {
    d3.select("#main").selectAll('g').remove();
    var selectionWidget = document.getElementById('selectionWidget');
    var option = selectionWidget.options[selectionWidget.selectedIndex].value;
    console.log("selection widget is" +option);

if (option === 'Primary Energy Production')
{
    Papa.parse('data/total_primary_energy_production.csv',
    {

        download: true,
        header: true,
        dynamicTyping: true,
        complete: function(results)
        {
            // loop through all the rows in file
            for (var row=0; row < results.data.length; row++)
            {
                var record = results.data[row];

                // make an object to store data for the current locality
                var locality = {
                    name: record.Locality,
                    energyProduction: []

                }
                console.log('locality: ' + locality);
```

Fig 3a. Parsing the data using several if-else-if loops to read the csv files

```
                drawLineChart(localityName);
            });

        }

    });
}

else if (option === 'Primary Energy Consumption')
{

                Papa.parse('data/total_primary_energy_consumption.csv',
                {

                download: true,
                header: true,
                dynamicTyping: true,
                complete: function(results)
                {
                    // loop through all the rows in file
                    for (var row=0; row < results.data.length; row++)
                    {
                        var record = results.data[row];

                        // make an object to store data for the current locality
                        var locality = {
                            name: record.Locality,
                            energyProduction: []

                        }
                        console.log('locality: ' + locality);
```

Fig 3b. Parsing data with if-else-if loops to read the csv files

***Step 3**: Iterate within constraints*

Even though this problem was not a part of the design, my limited JS knowledge constrained my ability to implement my designs. For instance, when the page loaded I could not get a chart to load unless the user made a selection. Even after several attempts, I failed to find a solution to the problem. So I decided to instead inform the user of this glitch. I introduced another dropdown option, *Pick a category*. This category informed the user of the action desired of them to load the respective charts.

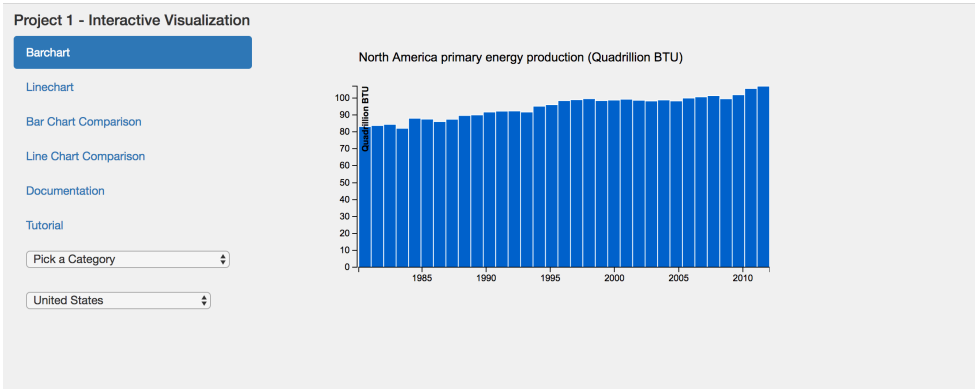

Fig 4. Introduced a "Pick a category" option in the dropdown

Key Insights

One of the charts that caught my attention upfront was the CO2 emission trends for Serbia. From the chart we see that CO2 emission showed a sharp rise in the year 2005. In order to understand this trend line, I searched the internet to gather more information about Serbia during this particular year. It was interesting to note that the temperature in Serbia is on the rise. As a result, permafrost zones (zones at or below freezing) have been releasing large amounts of green house gases in the atmosphere. This is one of the reasons for the sharp increase in the CO2 emission in Serbia.

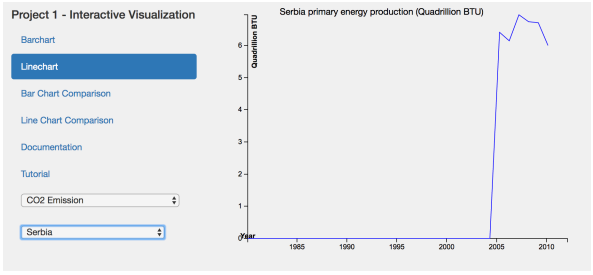

Fig 5. Serbia shows a sharp increase in the year 2005

Using the line chart, we can summarize the coal consumption of several regions worldwide. Regions such as United States consumed large amounts of coal until the year 2005, after which their coal consumption plummeted for the first time in approximately 30 years. However, in Central & South America regions coal consumption has increased rapidly since the year 2000. Colombia is the 10th largest producer of coal.

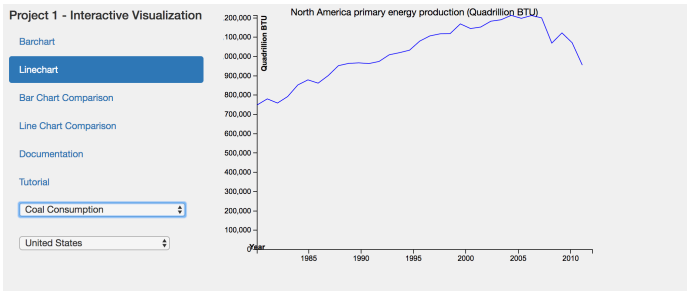

Fig 6. North America's coal consumption has decreased in the last 5 years.

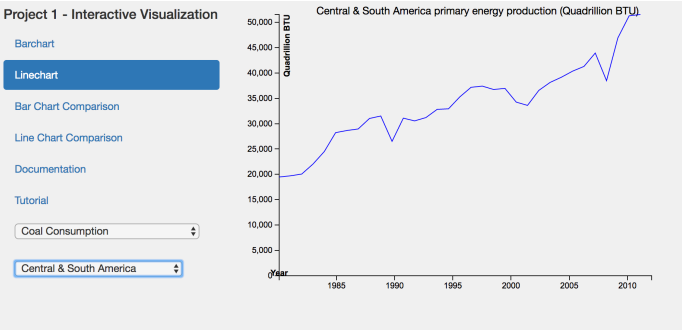

Fig 7. Central and South America's coal consumption has seen a dramatic increase in the past 5 years.

Conclusion & Limitations

Plotting the line and bar chart gave me a detailed perspective on the energy trends for the regions worldwide. However, plotting a scatterplot for two variables would have given me richer insights into the energy trends. One of the biggest challenges I faced was donning the hat of a designer and developer in a single project. There were several occasions when my designs could not be implemented as thought out due to my limited knowledge with JavaScript. Overall, visualizing the energy production and consumption data put forth interesting trends from the past decade.