… CSCE 5320 Scientific Data Visualization …   
ICE-6

…Marks and Channels. . .

# Created a covid cases dataset that contains the quantitative value and visualized it by creating a Bar Chart in VizHub.

## Explanation-Introduction:

COVID-19 is an infectious illness that results from the SARS-CoV-2 virus. The provided dataset presents information about the deaths, recoveries, active cases, and total tests related to the coronavirus in different countries. The dataset was last updated one month ago and included eight columns: "Serial Number," "Country," "Total cases," "Total Deaths," "Total Recovered," "Active Cases," "Total Test," and "Population." There are approximately 232 rows in this dataset. We shall be using this dataset, cleaning it, and using the required attribute to visualize covid cases using a horizontal bar chart.

## Methodology:

Firstly, I have downloaded Excel with Covid Cases data. We can see that it has eight columns: "Serial Number," "Country," "Total cases," "Total Deaths," "Total Recovered," "Active Cases," "Total Test," and "Population."

I have converted the excel into a CSV file as shown below.

As we can see above, there is a serial number column that is not needed for the visualization or prediction of data. We shall remove that column. Below is what the dataset looks like after removing the column.

We can observe that a few rows have many N/A and NULL values, removing such rows. Below are a few such rows that have been highlighted with a red underline.

I have converted the numbers in text format to numeric values, which will make it easy for visualization. Below is the final CSV file without any junk data in it. This clean data has 7 columns and 196 rows.

Now we load this clean data into the GitHub Gist, I have created a GitHub Gist as “covid\_worldwide.csv” and paste the CSV data from my computer onto the GitHub Gist as shown below. I have saved it as a public Gist.

After saving, the file looks as shown below. In the final file, we have seven columns: “country”, "total\_cases," "total\_deaths," "total\_recovered," "active\_cases," "total\_test," and "population." It has 196 rows.

If we click the “Raw” button, the raw data is loaded on another tab of the browser as shown below.

I wrote a code using the D3 library in VizHub to create a horizontal bar chart that displays COVID-19 data for the top 15 countries with the most cases. Initially, I loaded the dataset using a gist URL, and only the active cases and countries were loaded. The height, width, and margins were all defined. To parse the data and convert the necessary columns to numbers, I passed a row function to the CSV function. The vertical scale, which maps the country names to their positions on the y-axis, was created using the scaleBand function. The horizontal scale, which maps the number of active cases to their positions on the x-axis, was created using the scaleLinear function. The message "Loading..." is displayed until the data is completely loaded. The returned function sends the visualized data as an SVG image. The xScale.ticks().the map function is used to create the tick marks and labels on the x-axis, and the yScale.domain().map function is used to create the labels on the y-axis. Finally, the data.map function is used to create rectangles for each country, representing active cases. The rectangles are positioned according to their country name on the y-axis and their respective data values on the x-axis using the translate function.

We use an HTML page to display the data. The code includes a style section with instructions for the body and message classes, defining font size, text alignment, and overflow. It also has a header with a first-level heading tag that presents the page's title. The body contains a div with an ID of "root" and a script tag linking to the bundle.js file. Furthermore, three script tags load external libraries, React, ReactDOM, and D3. Below is how the bar chart is displayed. For example, Japan has the highest active cases of around 10000000. The next highest is the USA with around 2000000 cases.

Now I have changed the code to include recovered cases and total cases. We are using the use effect function to get active, total, and recovered cases. I have changed the data.map function to create rectangles for each country, with different colors representing active cases(red), total cases(blue), and total recoveries(green). The setData function determines how many countries are displayed.

We can even visualize multiple columns in the dataset against the countries. Here in the below bar chart, we can observe total cases, recovered cases, and active cases. The number of cases is represented on the y-axis and countries are represented on the y-axis. We can observe that the USA has the highest total and recovered cases of around 100000000. Japan has the highest active cases of around 10000000. The active cases are represented using red columns, recovered ones are represented using green columns and total cases are represented using blue columns.

## Conclusion:

To summarize, the dataset given to us contains COVID-19 data for various countries, including deaths, recoveries, active cases, and total tests. We cleaned the dataset by removing unnecessary columns, eliminating rows with null values, and converting text into numeric values. The cleaned dataset was then loaded into a GitHub Gist, and we utilized the D3 library to create a horizontal bar chart that displays COVID-19 data for the top 15 countries with the highest number of cases. We used HTML to display the chart with style instructions, including a header with a first-level heading tag that presents the page's title. Additionally, we modified the code to display active, recovered, and total cases for each country, and we used the useEffect function to retrieve the necessary data. Finally, we created a bar chart that represents active cases in red, recovered cases in green, and total cases in blue, enabling us to monitor and mitigate the spread of COVID-19 effectively.

Link to data:

<https://gist.githubusercontent.com/nehabaddam/376eed68b3f6e35d637f9e136523b47b/raw/b1b61d04b3e176715e02c2659339c00f249d2371/covid_worldwide.csv>

VizHub Link: <https://vizhub.com/nehabaddam/5d189bd18f384ad9b9db776ad7a76de0?edit=files&file=index.js>

VizHub ID: nehabaddam

# I have downloaded the penguin data set and created a scatter plot based on the data in VizHub.

## Explanation-Introduction:

We are using penguin data set to create a scatter plot. This dataset consists of physical measurements of 3 different species of penguins on different islands. The measurements include bill length, bill depth, flipper length, body mass, and sex of the penguins. The bill length is measured in millimeters, while body mass is measured in grams. The dataset contains information on both male and female penguins. The scatter plot represents flipper length against bill length with body mass for different species represented using different colors.

## Methodology:

I have downloaded the penguin dataset and added it to my gist. I have used this data to create a scatter plot with X and Y labels for the plot.

The dataset has 8 columns, “species”, “island”, “bill\_length\_mm”, “flipper\_length\_mm”, “body\_mass\_g” and “sex”. There are 333 rows of data.

Below is the code, it uses D3.js, a widely-used JavaScript library for data visualization, to generate a scatter plot of penguin data. The data is contained in a CSV file and is loaded into the script with D3's csv function. A parseRow function is utilized to convert specific columns to numerical values.

The x and y scales for the plot are set using D3's scaleLinear function, with the flipper length and body mass columns used for the domain, respectively. The characteristics for each data point, including the x and y coordinates, species, color, bill length, and depth values, are established using an array of objects.

The plot itself is generated by utilizing D3's select, append, and join functions to create and format circles for each data point. The axis labels are also generated using D3's append function and repositioned with the translate and rotate functions.

Overall, this code presents a fundamental template for creating scatter plots with D3.js and can be customized for different datasets by modifying the CSV file and adapting the parseRow, xValue, yValue, and marker functions to match the corresponding data columns.

Below is how the to scatter plot appears without labels initially.

After adding the labels, the scatter plot is displayed as shown below.

The x-axis represents flipper\_length\_mm and the y-axis represents bill\_length\_mm. The 3 species of penguins are represented using different colors, the Adelie species uses orange, Gentoo is red, and Chinstrap is blue.

## Conclusion:

In conclusion, the code creates a scatter plot using the D3.js library. The scatter plot visualizes data from a penguin dataset hosted on GitHub, where each point represents a penguin's body mass and flipper length. The color of each point corresponds to the penguin's species. The code also includes axes with labels for both the x and y-axis, as well as a title for the plot. The plot is designed to be responsive, adapting to the size of the window.

Link to Penguin Dataset:

<https://gist.githubusercontent.com/nehabaddam/b43f07ea239cff419dac8237e26bdd69/raw/018bb6fd3afa5a65afb9c2ccbeb5b1c5b48238fb/penguins.csv>

VizHub Link:

<https://vizhub.com/nehabaddam/8cc9cc5d75f440928bc1e65e46355cd0?edit=files&file=index.html>

VizHub ID: nehabaddam