# NM2207

Session 12 Codealong

# Overview of what we will do today:

# Create an interactive choropleth map displaying life expectancy data for different countries over time.

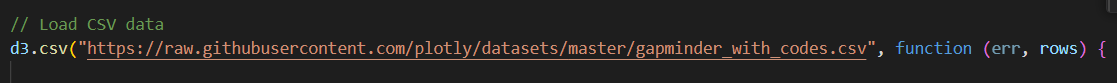
* Plotly boxplots
* Plotly maps
* Plotly menus

Please upload the codealong (all 5 parts) to your nm2207 page by the end of Saturday of Week 12.

**Warm up**

**Introduction to D3.js, CSV data loading, and data filtering**

* d3 data loading



See previous notes.

* Data filtering with array functions

A screenshot of a computer

Description automatically generated with medium confidence

This line of code creates a new constant variable called singaporeData. It filters the data array to get only the rows for which the country property is equal to "Singapore".

data.filter() is a built-in JavaScript method for arrays. It takes a callback function as an argument. This callback function should return either true or false for each element in the array. If the callback function returns true, the element is included in the resulting filtered array. If it returns false, the element is excluded.

In this case, the callback function is defined as an arrow function.

This function takes a single argument, row, which represents an element from the data array. The function checks whether the country property of the row object is equal to "Singapore". If it is, it returns true, and the row is included in the singaporeData array. If not, it returns false, and the row is excluded.

As a result, singaporeData will contain only the rows from the data array where the country property is "Singapore".

**Plot types**

**Box plots (Codealong 1 for today)**

Text

Description automatically generated

**Lines (Codealong 1 for today)**

**Text

Description automatically generated**

**Simple Map (See Week 5’s code.)**

**Text

Description automatically generated**

**Maps and Layout Objects:**

The layout object is used to specify the visual appearance and behavior of the plot. It includes properties such as the title, axis labels, color scheme, and font size, among others.

The updatemenus property is used to create a dropdown menu that allows the user to control the animation. It is an array of objects, where each object represents a single menu item. Each menu item has properties such as the label, the method to call when the item is clicked, and the arguments to pass to the method.

The sliders property is used to create a slider that allows the user to manually control the animation. It is an array of objects, where each object represents a single step in the slider. Each step has properties such as the label to display, the value to use for the animation, and the options for how the slider should be displayed.

**Map with time features (Codealong 2 for today)**

**Text

Description automatically generated**

The data array contains the information required by Plotly to create a choropleth map visualization. The locations, z, and text properties are initially set to display the data from the first frame, but they will be updated as the user interacts with the animation controls or the slider.

* type: This property specifies the type of the plot. In this case, it is set to 'choropleth', which means the plot is a choropleth map.
* locationmode: This property is set to 'world' to define the geographic scope of the map. It tells Plotly to display a world map.
* locations: This property is set to frames[0].data[0].locations, which is an array of location codes (ISO Alpha-3 codes) corresponding to the countries in the dataset. The codes are taken from the first frame in the frames array.
* z: This property is set to frames[0].data[0].z, which is an array of numeric values (in this case, life expectancies) associated with the countries in the dataset. The values are taken from the first frame in the frames array.
* text: This property is set to frames[0].data[0].text, which is an array of text labels (in this case, country names) associated with the countries in the dataset. The labels are taken from the first frame in the frames array.
* zauto: This property is set to false to disable the automatic scaling of the color scale based on the z values. This allows you to manually define the minimum and maximum values of the color scale using the zmin and zmax properties.
* zmin: This property is set to 30, defining the minimum value for the color scale. Countries with a life expectancy of 30 will be colored with the lowest color of the scale.
* zmax: This property is set to 90, defining the maximum value for the color scale. Countries with a life expectancy of 90 will be colored with the highest color of the scale.
* The data array contains the information required by Plotly to create a choropleth map visualization. The locations, z, and text properties are initially set to display the data from the first frame, but they will be updated as the user interacts with the animation controls or the slider.

**Part 1: Box plots and how to label them**

Examine 01.boxPlots and the code used to create its traces.

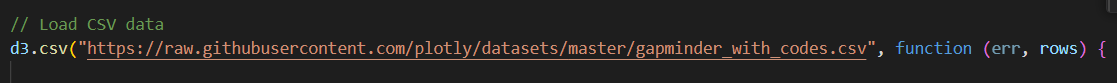
*From the Plotly website: “A trace is just the name we give a collection of data and the specifications of which we want that data plotted. Notice that a trace will also be an object itself, and these will be named according to how you want the data displayed on the plotting surface.”*

1. Now, examine 00.labeledPlot. Notice how it has a title, subtitle, legends and tooltips on hover. Use the code to add some of these features into your box plot.
2. Create a second plot next to or below the box plot. Here, you will plot the trend of life expectancy in these three countries from 1952-2007. It’s not that hard! Using the screenshots above, change the “type” of the “traces” to change the boxplot into a line plot. Does this address your goal?

**Part 2: Loading and processing data**

We need to

* Load data, using d3.csv



* Filter data for all the countries, for one year at a time.

Text

Description automatically generated

**Part 3: Preparing frames of data**

* Use the filtered data to populate ONE frame at a time in a for loop. Each frame has the locations of all the countries and their life expectancy values in that year.
* Put this yearly data into an object array called frames.

Text

Description automatically generated

**Part 4 Create the map**

Now, we want to create a map. Refer to the notes on the structure of a map’s data object, above.

Text

Description automatically generated

**Part 5 Explore the controls**

There’s more to a map in d3 than just a map, though. We now have slides and controls.

In this code, the updatemenus property is an array with a menu item. The item, labeled "Play", triggers the "animate" method when clicked, with the argument of the current year and some options for how the animation should be displayed.

In this code, the sliders property has a single slider with steps for each year in the animation. The current value of the slider is displayed with a label that says "Year:" followed by the year number. When the user moves the slider, the "animate" method is called with the value of the slider as the argument, and some options for how the animation should be displayed.

* Add a second button called “Pause” to the array in the updatemenus property to stop the animation. It should look exactly like the Play button, but with an "animate" method with an argument of null, effectively pausing the animation. Set both the duration properties to be 0 as well.

**Reflection:**

How did data transformations change the way we understood life expectancy in the three countries?

**Original commentary from Week 11 (on a slightly different version of code)**

Now, lets explore the 02.mapEvent folder. This uses d3 to populate and update the map.

Lets explore the functions and calls we observe:

* **Line 2, d3.csv** : This seems to be the command that replaces our “fetch” command to create a promise. Instead of following up with “then”,the second argument to the function is what should be executed once the data is collected.
* You’ll notice that this function(err rows) is anonymous (it has no name) and it takes two arguments, err and rows. Where does its closing bracket end? That’s right, at the very end of the code.
* Trace the progress of the code. When is the filter\_and\_unpack function called?
* Add console.log statements to inspect what the value is of frames[i] after it is written, in line 16.
* Examine slider\_steps. It is the property of the slider which is updated in line 17. When is it finally used to initialize the “layout” property of the page?
* **Line 117, Plotly.newPlot:** This appears to have replaced the way we used to add a new chart to our webpage, by referencing a canvas element in the html. Notice what arguments are passed in this function call.Examine each of them. Here we see a “promise” function being used, so that the complete data is added to the charts only AFTER everything has been correctly rendered as a plot.

**Here's a detailed description of the code in main.js that I generated using Chat-GPT, and this is a demonstration of how Chat-GPT can be useful for your programming tasks.**

This code is an example of a Plotly visualization in JavaScript. It creates an interactive choropleth map of world life expectancy from 1952 to 2007, using data from Gapminder. The map is animated to show changes in life expectancy over time, and it includes a slider and play/pause buttons to control the animation.

Here's a step-by-step explanation of the code:

1. The **d3.csv** function reads the CSV data from the provided URL. The callback function handles the data manipulation and visualization.
2. The **filter\_and\_unpack** function filters rows of the data based on the specified year and then extracts the values of a specified key (column) in the filtered data.
3. The **frames** array stores the data for each year as separate frames for the animation. The **slider\_steps** array stores the configuration for the slider steps.
4. The loop iterates through the years, from 1952 to 2007, in increments of 5. For each year, it calls the **filter\_and\_unpack** function to get the life expectancy and location data for that year. These values are then used to create a new frame and a slider step.
5. The **data** array defines the initial choropleth map with the first frame's data, setting the location mode to 'world', the color scale (zmin and zmax), and other map properties.
6. The **layout** object specifies the overall layout of the plot, including the map's appearance (colors, borders, etc.), the title, and the configurations for the animation controls (play/pause buttons and slider).
7. The **Plotly.newPlot** function creates the plot in the specified HTML div ('myDiv') using the data and layout objects. After the plot is created, the **Plotly.addFrames** function adds the animation frames to the plot.

When the visualization is rendered in a browser, the user can interact with the map, play and pause the animation, and use the slider to jump to specific years.