Homework 3: Creating Interactive Charts with D3.js

Instructor Hamza Elhamdadi
Course COMPSCI 571: Data Visualization & Exploration
Due Apr 14, 2025 at 11:59PM Eastern Time

In this assignment, you will use Javascript and D3.js to create a interactive visualizations of COVID-19 data.

Include your name, UMass email, and SPIRE ID in the HTML file in your submission. Also, make sure that your submission is a valid HTML5 file. You can check the validity of your HTML by uploading it to the W3C HTML Validator.

Download the starter code from Canvas. The file structure for the assignment will look like this:

```
hw3/
    hw3.html
    script.js
    data/
        covid_us.csv
        covid_utah.csv
        covid_ca.csv
        covid_ny.csv
```

All work for this assignment will be done in **`hw3.html`** and **`script.js`**.

Remember: For your browser to be able to access the data file using Javascript, you will need to be *serving* the hw3 directory, not just opening the HTML file in a browser.

We recommend using the Live Server extension in Visual Code to serve the hw3 directory. See the *HTML and CSS* lecture (Week 2) for instructions on how to do this.

If you are using a different development environment that does not provide a built-in server, you can start one using the following commands:

```
    $ cd path/to/hw3
    # for python 2
    $ python -m SimpleHTTPServer 8080
    # for python 3
    $ python -m http.server 8080
```

Then, you can view the page at https://localhost:8080.

If you are using the python server, it is a good idea to Hard Refresh whenever you make changes to your code. This will clear the browser cache on the current page and force the browser to reload your Javascript and CSS files. This is different from a regular "refresh" via simply pressing the *Refresh* button or pressing (CTRL or CMD)+R.

Part 1: Basic Events and Setup

In the the starter code file **`hw3.html`**, we provide three controls: two *Select* menus for choosing a dataset and metrics, and a *Random Subset* checkbox. Attach event listeners to the menus and checkbox in the `setup()` function so that they call `changeData()` when the value changes.

For this homework, you can add to the HTML so that you have `<svg>` and some `<g>` elements to work with, or add those necessary elements in the `setup()` function that will only be called once. The **`hw3.html`** file also has some styling that you can use for different charts.

The Data

Note: This section is informational only, no need to implement anything.

The provided starter code loads and parses the CSV data depending on which option the user selects. The `changeData()` function imports the **`.csv`** file based on the output of menus and output the corresponding data structure. The output is an array of objects, where the data has already been parsed:

```
[...,
{cases: 5842, date: "08/27", deaths: 144},
{cases: 5383, date: "08/28", deaths: 148},
...]
```

Make sure to familiarize yourself with the data structure, e. g., printing the object and exploring it in the console.

Each object in the array has the date, cases, and deaths of COVID cases. `changeData()` calls `update()` with the output data, and `update()` is where the visualization will update based on the data.

Implement each chart with the `data` input of `update()`.

Part II: Drawing the Visualizations with D3

The **bar chart**, **line chart**, and **area chart** use the date for x-axis and the selected metric of the *Data Metric* drop-down for the y-axis. You can retrieve the value of the drop-down with `d3.select("#metric").node().value ;`. The **scatterplot** uses the cases for x-axis, and the deaths for y-axis. With this in mind, you need to:

1. Construct scales for each chart. *[HINT: Recall that the y-axis for the bar, line, and area charts are the same.]*
2. Draw the axes for each chart.
3. Draw and update the elements based on the input data.

Using some constant "margin" can be helpful when placing an axis on a chart so that the axis labels are shown within the svg. Some globally-defined constants are provided that you might find useful, but make sure that the scales are also adjusted when margins are incorporated. The scales should use `[0, maximum]` of the data visualized currently. The axes need to be transformed to the appropriate place. The scales and axes should update based on any change in the data.

For the **bar chart** and the **scatterplot**, use the data to populate your `rect` or `circle` elements. Position and size them correctly with the scales.

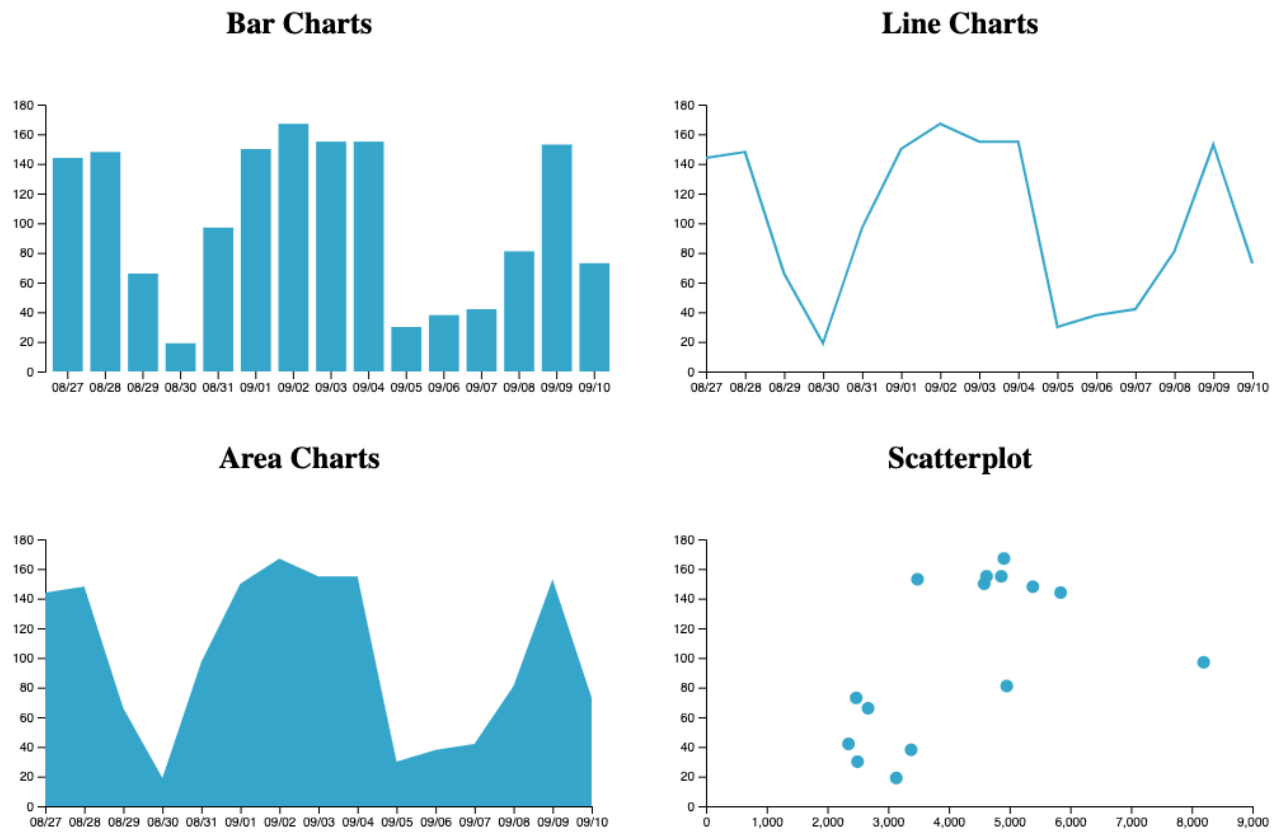
For the **line chart** and **area chart**, the drawing will use a single `path` element. You can add the path in your HTML or in the `setup()` function. Binding an array to a single element will be different from the bar chart and the scatterplot, requiring the use of `datum()` instead of `data()`.

- Note that `datum()` called with a data structure such as an array binds the whole array to a single element, instead of creating multiple elements. That makes sense for a line chart, because the `path` element for that is just one element that is re-shaped based on all of the data, as opposed to multiple bars.

Use `d3.line()` and `d3.area()` to generate a `d` attribute for the path. The starter code provides (commented-out) code for the line and area constructors in `update()`.

We suggest to have the code for each chart in separate update functions and call each chart update function in the main `update()` function. Adjust the chart update function inputs based on your need to draw the charts.

Your final result should look somewhat like this:



Extra Credit 1: More Events

We focus on interactivity in this class; you will usually implement interactivity with JavaScript. We suggest using D3 events for these.

First, make rectangles in bar chart and points in scatter plot change color when the mouse hovers over it using JavaScript, not CSS hover styling. One way to do this is to make a `.hovered` class that uses a different fill. Assign elements to have the class when the mouse is on the element, and remove the class when the mouse leaves.

Your second interactive component will be to add an *On Click* listener to circles. When clicked, the browser should console log the `x` and `y` coordinates of that point to the console.

1 Extra Credit 2: Transitions

For extra credit, animate each D3 transition (gradually change sizes, positions, and shapes when switching datasets, and fade new and old items with opacity). As we will learn later in the course, animation is very attention-grabbing; make sure your animations are tasteful and subtle.

2 Demo

You can check out a demo of how the interactions could look like here.

3 Submitting Your Work

Submit your code as a zip file on Gradescope. Include all code and data files and directories in your submission.

4 Grading

Your score on this assignment will consist of the following:

- 5%: The menus trigger the appropriate functions.
- 60%: Each chart is rendered properly on page load (15% for each chart)
- 15%: The charts, including the axis update based on data change.
- 10%: Elements are removed when *Random Subset* loads a smaller dataset.
- 10%: Elements are created and styled appropriately when *Random Subset* loads a larger dataset.
- +5%: Extra Credit 1: Elements change styles on hover and console log the data points on click.
- +5%: Extra Credit 2: All D3 transitions are animated tastefully.