Assignment 1: Basic Web Visualization

Date Posted: January 19, 2024

Due Date: February 2, 2024 11:59pm

Description

In this assignment you will create a simple webpage with several basic visualizations, using HTML, CSS, SVG, and JavaScript (only in part 2). This assignment is split into two parts: (1) Visualization by Hand (HTML & CSS only) and (2) Visualization with JavaScript.

In part one, you will only be allowed to use HTML and CSS to manually create the specified visualizations. The goal of this part is to get you comfortable with the basic building blocks of web visualization, primarily the SVG, that we will later manipulate with JavaScript and D3 to create visualizations. It is important that you understand how these pieces work before moving forward.

In part two, you will use JavaScript to create visualizations programmatically. You will use JavaScript to create and manipulate DOM elements based on the given data. The goal of this assignment is to familiarize you with manipulating SVG objects programmatically and the idea of creating data driven visual elements, before introducing you to D3.

Submission Requirements

Provided Template

In the GitHub repository, you will find three files: `spec.pdf`, `index.html` and `scores.js`. `spec.pdf` simply contains a copy of the assignment specification (i.e. this document). The `index.html` provides an outline for your submission. It contains an SVG tag for each assigned visualization that you will modify for each visualization. The `scores.js` file will only be used in Part 2 and is discussed in the Part 2 description.

Submitted Files

You will create your visualizations in `index.html`, it should be the only HTML file in your submission. You may add any additional JavaScript and CSS files as you see fit. You may **not** use any existing JavaScript files or libraries that are not provided in the template or written by you! You should add appropriate documentation via commenting to all JavaScript code, whether it's in a separate `.js` file or written directly in your `index.html` file. This is a good habit to get into long term and will ultimately help us accurately grade your assignments. Finally, you should include a `README.md` file that provides a text description of how to run your program and any parameters that you used. Also, document any idiosyncrasies, behaviors, or bugs of note that you want us to be aware of. When you have completed your assignment, please submit the link to your repository on Canvas.

Note, web development can be tricky and rendering can differ across browsers! I don't expect you to test

your solution across every browser to ensure it renders properly. However, I will largely be testing on Firefox and I recommend you use a common browser (Chrome, Firefox, etc.). If you are using a different browser, it may be worth noting in the README file. If there are discrepancies in rendering because of browser differences that result in loss of points, please bring them to me and we will work to resolve them.

Part 1: Visualization by Hand (HTML & CSS Only)

In this part of the assignment, you will create four visualizations: 2 bar charts, 1 line chart and 1 pie chart. You should create your visualizations in the corresponding SVG elements specified in the template. Each SVG element should be of size 400 x 400 and use the provided `id` attributes:

- bar chart with magenta bars: `magenta_bar_chart`
- bar chart with cyan bars: `cyan_bar_chart`
- line chart: `line_chart`
- pie chart: `pie_chart`

1a. Bar Charts (20 pts)

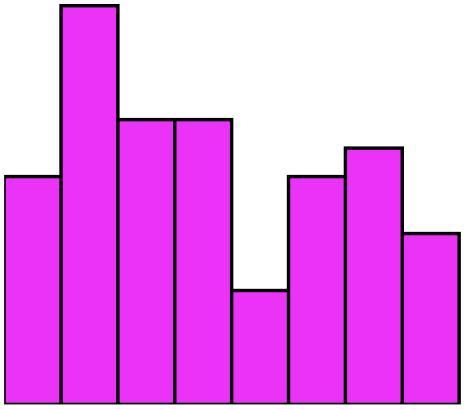
Your bar charts should take all of the horizontal space in the SVG (i.e. all 400 pixels). The individual bars should have the following heights, in order: `200`, `350`, `250`, `250`, `100`, `200`, `225`, `150`.

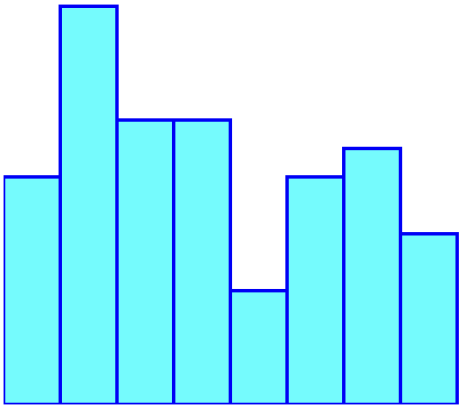
You should create a bar chart with magenta bars in the SVG element named `magenta_bar_chart`, and a bar chart with cyan bars in the SVG element named `cyan_bar_chart`.

In addition, the SVG elements inside each of the main elements should be IDENTICAL. In other words: any declarations that will influence the color of the bars need to be done with the attributes of the different **outer SVG elements** and your **CSS declarations**. It is important that you learn how to do this kind of thing because it will enable you, later on, to write better abstractions in d3. Better abstractions mean you can try more things faster. Trying more things, as we discussed, means you will end up with better visualizations.

To simplify specifying the position and heights of bars, you may also want to consider using a group `<g>` for the bars and then an appropriate transform.

The two bar charts should look like the following, using appropriate fill and stroke styles to recreate the appearance:

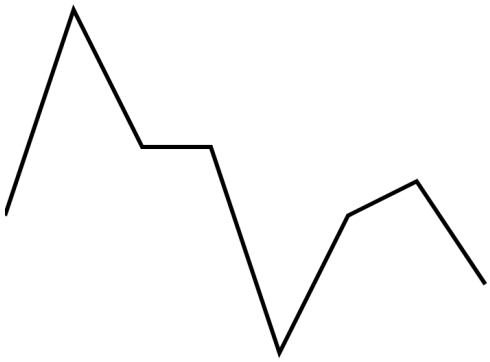




1b. Line Chart (15 pts)

Your line chart will be constructed using the same data as the bar charts. You can either use SVG *line* elements to construct it or compose a single *path* element for the entire polyline. The heights of the vertices in the polyline should be the same as the heights of the individual bars in your bar charts.

Your line chart should look very similar to this:

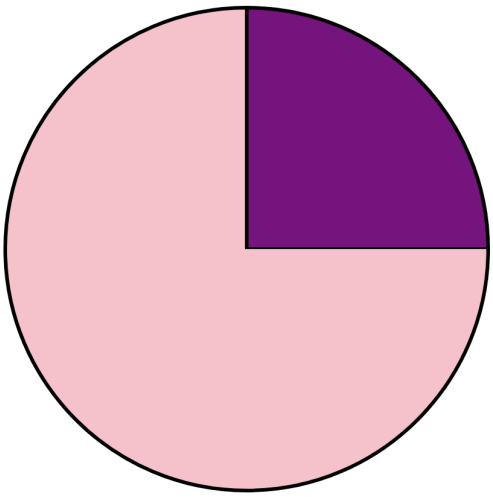


****EXTRA CREDIT (10 pts):** Construct a second line chart using the method you did not use in the first chart (i.e. if you used SVG line elements for the first chart, use a single path element.) If you choose to do this, create a 400 x 400 pixel SVG with id `line_ec`, placing it after the `line_chart` SVG. The resulting line chart should nearly identical to the chart in the previous question.

1c. Pie Chart (15 pts)

Your pie chart should have two wedges. The first wedge should span an arc of 270 degrees with two line segments to connect to the center. This wedge should be pink. The second wedge should span the remaining 90 degrees and should be purple. Hint: You will want to look at the svg path element, specifically the arc command.

Your pie chart should look very similar to this:



Unlike for the bar charts, since each wedge is styled differently you can separately specify their style per SVG element as opposed to for the entire SVG outer canvas. As a result you may want to create separate id's for each wedge in addition to the id for the chart.

Part 2: Visualization with JavaScript

In this part you will create a scatterplot of the data in the `scores.js` file by using JavaScript to create and manipulate SVG elements. You will use the data in The `scores.js` contains an array of 271 elements. Each element of the array is an object containing the SAT Math (SATM), SAT Verbal (SATV), ACT, and GPA scores for students at Calvin College. Note, you may **NOT** use D3 or any external libraries/code to create your visualization.

```
var scores = [
 { SATM:430, SATV:470, ACT:15, GPA: 2.239 },
 { SATM:560, SATV:350, ACT:16, GPA: 2.488 },
 { SATM:400, SATV:330, ACT:17, GPA: 2.982 },
 { SATM:410, SATV:450, ACT:17, GPA: 2.155 },
// 263 more rows
 { SATM:700, SATV:680, ACT:35, GPA: 3.911 },
 { SATM:720, SATV:770, ACT:35, GPA: 3.981 },
 { SATM:750, SATV:730, ACT:35, GPA: 3.882 },
 { SATM:790, SATV:780, ACT:35, GPA: 3.887 }
];
```

2a. Scatterplot Visualization (30 pts)

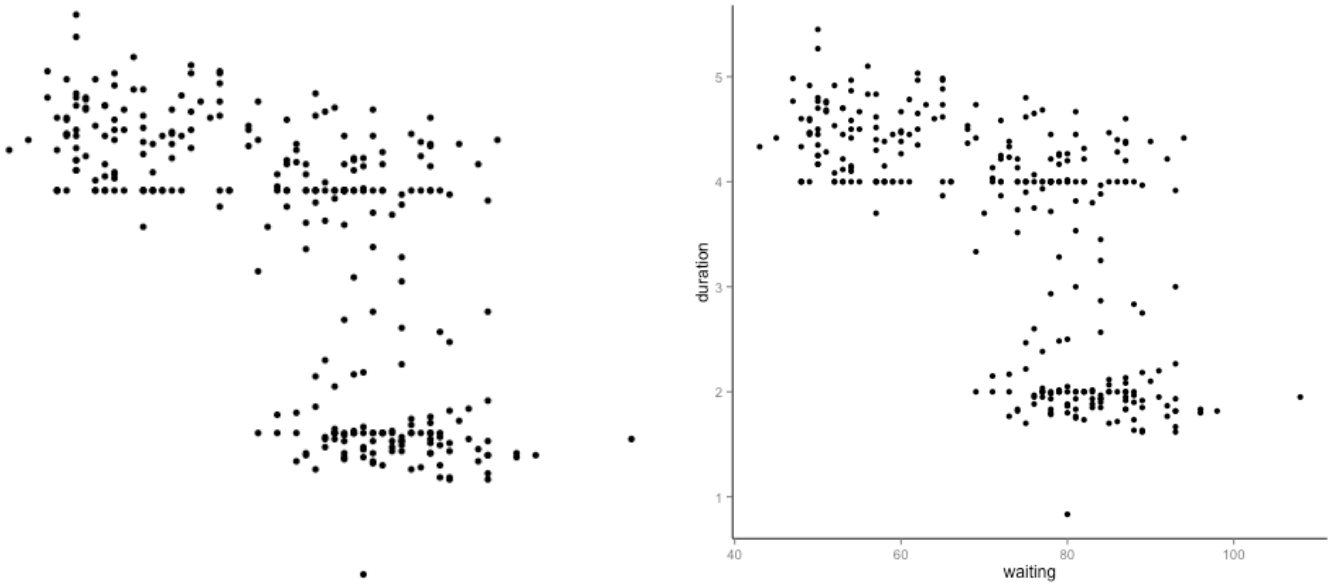
You will create a scatterplot of circles, using the following conditions:

- `SATM` should be on the x-axis
- `SATV` should be on the y-axis
- The radius of the circle should represent `ACT` scores
- The color of the circle should represent the `GPA`

I purposely left the "color" vague. I'd like you to think about the best way to use this visual signal.

You will create your scatterplot in the SVG with id `scatter` in the `index.html` file. The SVG should be 500 x 500 pixels.

****EXTRA CREDIT (15 pts): Axis Annotations.** Add axis lines, labels, and tick marks to the X and Y axes of your visualization. In other words, to get this extra credit, instead of looking like the plot on the left, it should look like the plot on the right:



(Of course, the dataset I just used in the example above is not the same as the one you have, so the values for the tick marks, labels, etc. will all be different).

Some things you will need to consider:

- margins for your axes, tick marks, and axis labels
- try to pick good values for the tick marks, and a good number of them: not too many, and not too few.

2b. Reflect on the Scatterplot Design (20 pts)

One of the goals of this course is to get you to think about how best to construct visualizations. We will discuss guidelines and principles to create effective design throughout the course. For now, I want you to reflect on the design of the scatterplot in the previous question and address the following questions in a short paragraph or two.

- Why did you choose to color the circles as you did? (We will discuss color choice in more detail in upcoming lectures).
- In reality, this visualization is OK, but not ideal. What are some problems with the design of this visualization? In what ways is it ineffective?

Grading

The points for each question are specified in the question.

In Part 1, the visualizations will be graded based on how well they match the described visualization. The charts don't have to look exactly like the ones here, but the general style should match: fill and stroke colors count, as do positioning of the elements, and respecting the constraints of the assignments (using the right elements and right declarations, as described above).

In Part 2, full points will be given to visualizations that correctly position each data point, correctly map the radii of each data point, and correctly map data to color. For the written questions, I don't expect you to use any principles of visualization (as we haven't learned them yet!) but you should make an honest effort to reflect on the visualization.