Blair Ednie
CS-39AE

November 15, 2024

Project 2: D3.js Interactive Web Page

1. *"What type of visualization did you select and why?"*

    - Single Station Line Plot (from lab):
        - Shows detailed temperature trends over time for individual stations
        - Allows users to focus on specific locations through a dropdown selector
        - Gives context for the broader patterns shown in other views

    - Temperature vs. Elevation Scatter Plot
        - Visualizes the correlation between elevation and average maximum temperature
        - Includes a regression line to highlight the overall trend
        - Features interactive temperature-based filtering to help identify stations in specific temperature ranges
        - Makes it easy to spot outliers and clusters of stations with similar characteristics

    - Multi-station "Spaghetti" Plot
        - Shows all stations' temperature trends simultaneously
        - Uses a viridis color scale to encode elevation information
        - Implements elevation-based filtering to analyze temperature patterns within specific altitude ranges

    - I chose these two visualizations because one of the outstanding questions I had about this dataset, having worked with it in a previous homework assignment for this class, is the relationship between Avg. Daily Max Temp and the elevation. From the scatter plot, we can see that there is some correlation between elevation and lower temperatures, but not a very strong one.

    - For the Spaghetti Plot, I chose this" messy" visualization partly to see how D3.js can handle lots of plots at once on the same chart. Just like with the Python plotting packages, we can apply a colormap and create tooltips to help communicate facts about the data. Another reason I chose this visualization is that I thought it would be perfect for the elevation range filtering I had in mind.

2. *"A summary of the D3.js features you implemented."*
    - Data Handling:

    - CSV: Used d3.csv() to load external data from a GitHub repository
    - Data Parsing: Automatic conversion of numeric values using the + operator
    - Grouping: d3.group() to organize data by station names
    - Aggregation: d3.mean() for calculating average temperatures

- <u>Scales and Color:</u>
    - scaleLinear: For mapping years, temperatures, and elevations to pixel coordinates
    - scaleSequential: Combined with interpolateViridis for elevation-based color mapping
    - Domains: Dynamic domain calculation using d3.extent(), d3.min(), and d3.max()

- <u>SVG Elements and Visualization:</u>
    - SVG Container: Created three separate SVG containers with margins
    - Basic Elements: paths, circles, and lines for various plot elements
    - Line Generator: d3.line() for creating temperature trend lines
    - Axes: d3.axisBottom() and d3.axisLeft() with custom formatting
    - Labels: Text elements for axis labels and titles
    - Trendline: Custom-calculated regression line using statistics
    - Interactive Elements: Dots with hover effects and tooltips
    - Transitions: Used for smooth tooltip appearances/disappearances

- <u>Interactivity:</u>
    - Event Handlers: "mouseover", "mouseout", "change", and "click" events
    - Tooltips: Custom div-based tooltips with dynamic positioning
    - Filtering: Interactive filtering based on elevation and temperature ranges
    - Selection Updates: Dynamic updating of visual elements based on user input
    - Forms: Dropdown menus and number inputs with event listeners

3. *“Any challenges you encountered and how you overcame them.”*
    - Since I used AI tools (Claude 3.5 LLM) for this project, many of the challenges came from wrangling Claude rather than the code itself.
        - Context window problems: Claude would forget what we were doing, or generate the same code twice. We ran into problems especially when getting default values for the filter ranges, since this touches several parts of the code: colormap, buttons, CSS, functions, and requires getting the min and max from the data. In the end I had to break this down into small pieces and iterate (responsible for many of the 90+ deployments of the site)

    - Scatter plot trend line: Claude had fatal problems with managing D3 and something called “D3-regression,” which I didn’t end up using. D3-regression was conflicting with D3 causing the visualizations to error and display nothing. In the end I worked with Claude to calculate the trendline manually, avoiding the import.

    - Consistent formatting & style: Making sure across the three visualizations that the font sizes, labels, tooltips, colors, axes, and so on, are consistent was a challenge. This required reworking parts of the code to reuse CSS and to use consistent D3 methods and attributes.

- UI/UX challenges - Text entry vs. Dual range slider: In earlier iterations I used text entry boxes for the range endpoints, but this felt clunky. I thought dual range sliders would be a better user experience. Although it isn’t D3.js per se, I found https://refreshless.com/nouislider/, a package which allowed me to implement dual range sliders with ease. Then I was able to make this work with the D3.js features that were previously using text entry boxes.