CS 448B - Alexa Siu & Junwon Park

Link to live visualization:
http://alexasiu.com/CS448B_A3/CS448B_A3.html

Github repository:
https://github.com/alexasiu/DynaSFRestScoreQuery

Interaction

Location filtering: Our visualization allows users to filter restaurants by distance between two locations. Initially, the user can click to specify each of the two locations which appear as two circles, blue and green in color. Then to update their locations, the circles can be dragged. Slider elements allow the user to change the radius of each circle. Data that lies at the intersection of these two circles appear in red while all other data points remain in gray. We chose red and gray as two contrasting colors to highlight this region.

Inspection score and risk filtering: An additional filter allows users to specify a desired inspection score level. Users can specify both the score value (0 to 100) or the discretized risk category levels (high, moderate, low). The data contains several inspections for each business, we decided the one with the latest date would be most relevant since users probably don't want to see outdated data.

Resetting: Users can use the *Clear* button to reset the visualization to its initial state.

Hovering for detail: Hovering over any of the data points reveals the restaurants name, address, inspection score, and risk category.

Code Structure & External Libraries

The index html file sets the layout for the visualization webpage. Then we have four javascript files that handle different tasks:

1. Data parsing (parseData.js):
   - reads, formats, and loads the specified data
2. Load map (loadMap.js)
   - draws the map
   - handles projection functions
3. Plot data (plotData.js)
   - creates the data and location A and B svgs
   - filters data showed based on user interaction
4. GUI:
   - creates GUI elements
   - handles input to GUI elements

External files for both css and javascript are inside external folders and also commented as such on the index file. We make use of the following:

1. JS:
   - d3.v5.min.js
   - dat.min.gui.js: for creating all the GUI elements
2. CSS:
   - dat-gui-light-theme.css: for changing the color scheme of our GUI elements

Work Distribution

Alexa was leaving out of town to a conference the last week of the assignment so she did the beginning base work. She set up the initial files structure, skeleton functions, and GUI elements. This amounted to writing the code that completed deliverable 1 i.e. clicking to set location A and B and filtering data to the intersection of the specified radius for each location. This included reading and formatting the data, plotting the map, refreshing the data, being able to change A and B, and being able to specify the radius of each in miles. Aspects that took most time were getting familiar with the workings of the projection functions and Promises for handling data.

Junwon then continued by adding more user interaction. This amounted to writing the code that completed deliverable 2 i.e. filtering by inspection score and risk category level. In addition, he added interactions for dragging the user-specified locations and for hovering to see more detailed information for each restaurant. Junwon also fixed some performance issues that came up from having a very high update rate (now limited to 0.1 s) and filtered for data duplicates to only plot the latest restaurant inspection. Aspects that took most time were setting up the event handlers for the different interaction events and decoupling issues between them.

Last Alexa and Junwon met to discuss final results and worked on some small improvements together such as performance and looks.

For a very detailed breakdown and ideation pictures that we discussed in our very first meeting, see this document: https://docs.google.com/document/d/1Gpm-JTjqUiT4hUH8_sDrBbWDKvPeiJAnwFyY1BnXj7w/edit?usp=sharing