Project 1 Documentation

By: Terrance Miller

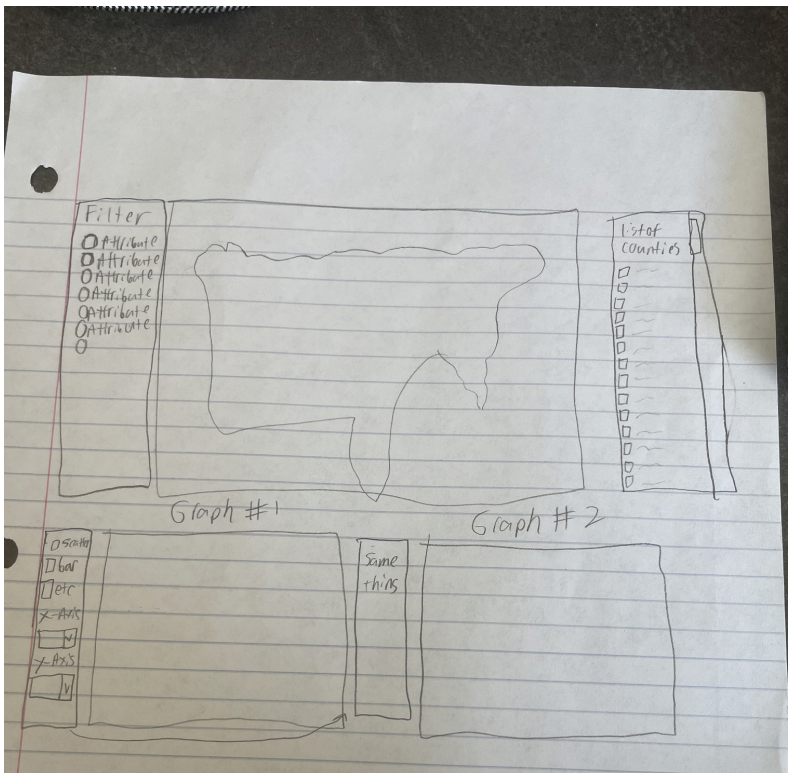
What is this application?

This is a data visualization application that can be utilized to understand health related statistics and how they're distributed/differentiated across the country.

What data is this application using?

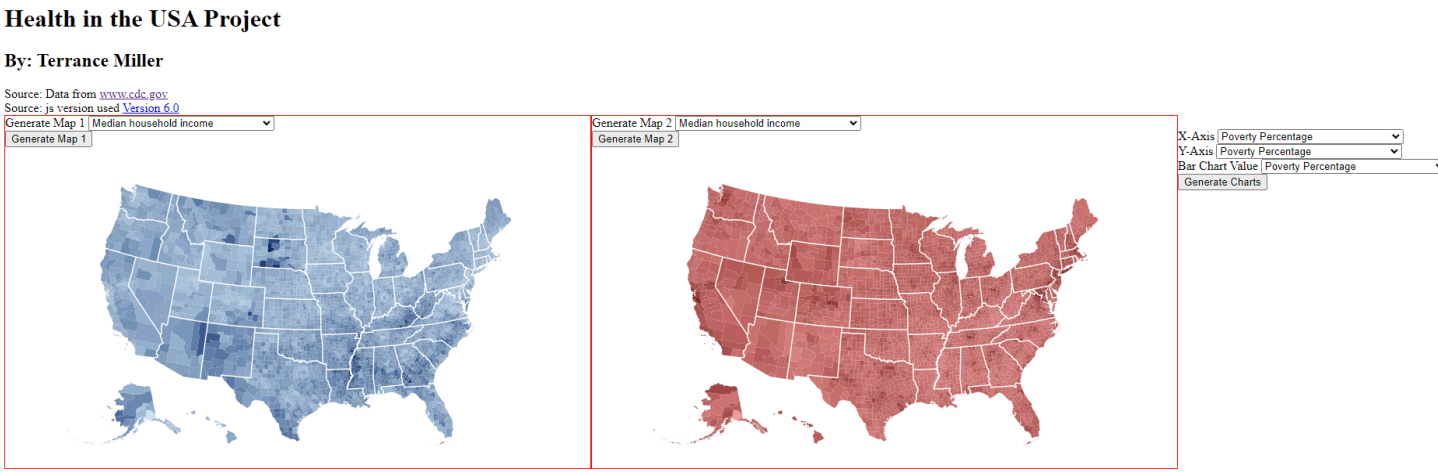
This application uses health and population data from the US Heart and Stroke Atlas. The data can be found here, https://www.cdc.gov/dhdsp/maps/atlas/index.htm. This data contains population health data per county data like, percent who have suffered a stroke or percent who have coronary heart disease. It also contains health facility access data including number of hospitals, number of primary care physicians, and park access. There is also general population data for example, elderly percentage, income data, and poverty percentage. This data can be used to study the correlation income has with general health. It also gives insight to how important health facilities like hospitals are to a region's health, and simple recreation facilities like parks being accessible to give the general public an area for healthier activities like hiking and sports.

Application sketches:



This is a rough sketch I made at the beginning of the design process for the project. The idea would be a list of radio buttons to select which data the choropleth map would be colored based off of. Then on the right hand side there would be a checkbox list of all the counties that would get updated based on which ones you highlight that way if you wanted to make sure you got certain counties in your brushed area you could double check. Then there would be two graphs with a form next to each where you could make them a bar or a scatter plot depending and pick what data you wanted on each axis. This I believed would be the optimal user interface because it would give the user as much freedom and accessibility as possible to customize the data as they saw fit. Then they could compare just exactly what they wanted to see. However as you will see my original sketch ended up being a bit too ambitious based on my skill level with d3.

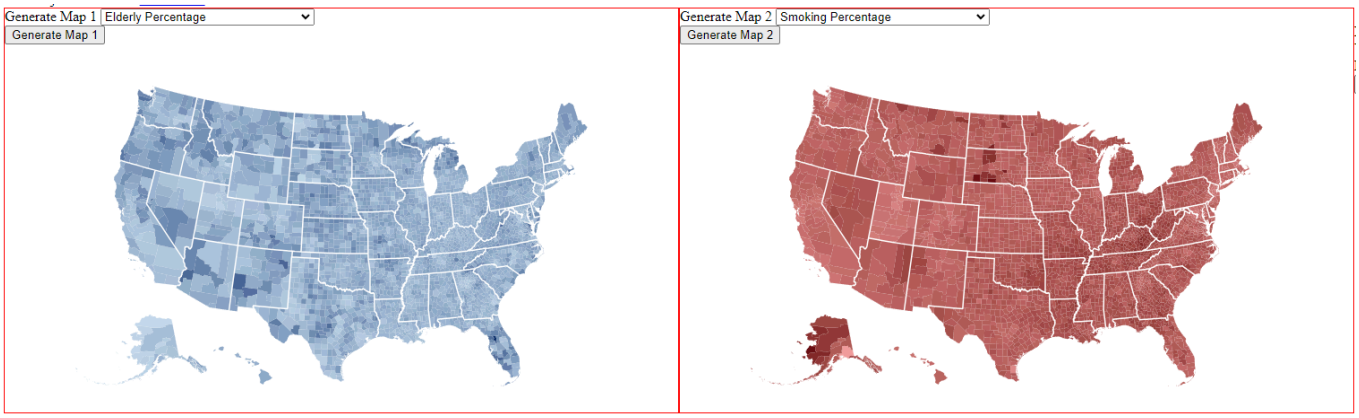
Visualization components:



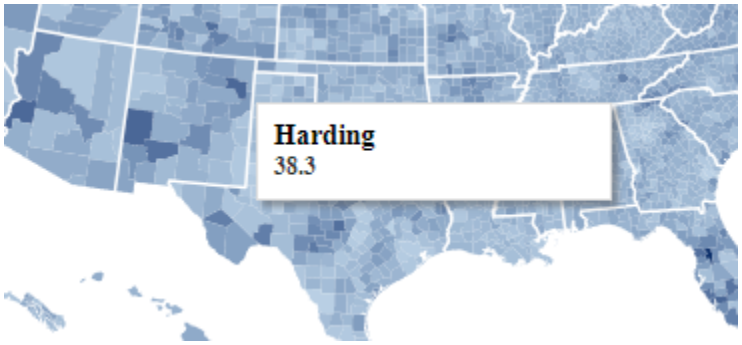
This is the initial view when you open the application. I have it by default create two choropleth maps of two different data attributes (Generate Map1's dropdown should default to poverty percentage not median household income) to give the user an understanding of what this application is made for. Unfortunately, I ran out of time trying to build linking and other dynamic features and didn't have enough time to touch up on the html and css so the UI doesn't look great. Its labels could also use work to help translate to the user exactly what each thing is doing. When you press the generate charts button you will generate a scatter plot and bar graph based on what values you choose for the drop down menus. The drop down menus contain every value of the data set except urban_rural_status which is built into the colors of the graphs. I didn't initialize the graphs by default because I figured it would spark their curiosity in the data so they would engage with the application more early and then they could produce graphs relevant to them.

I'll break down the 3 visual components here:

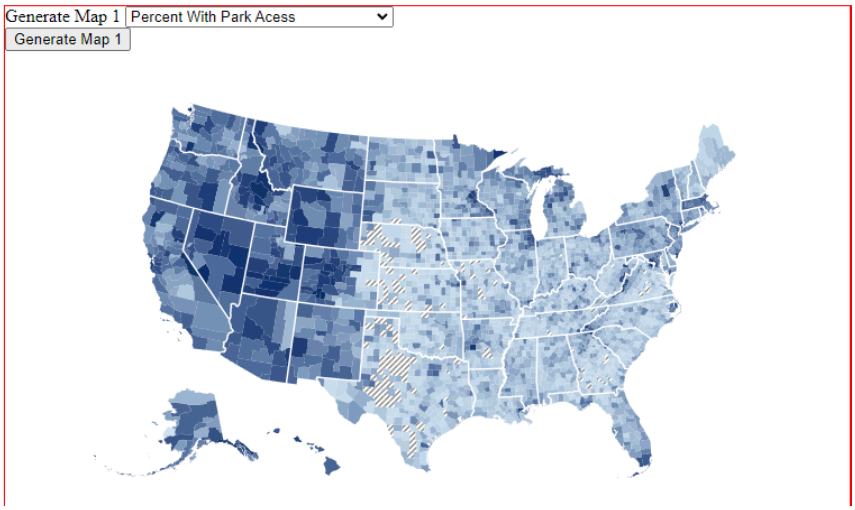
Choropleth Maps



I used two choropleth maps so users can compare any two data sets and how they distribute throughout the country. I made sure to use two different colors for the two maps so the user doesn't get confused looking between the two and the different data attributes can separate themselves. Each generated map button and dropdown are connected to each other so you can update one map or the other you don't have to do both at once.

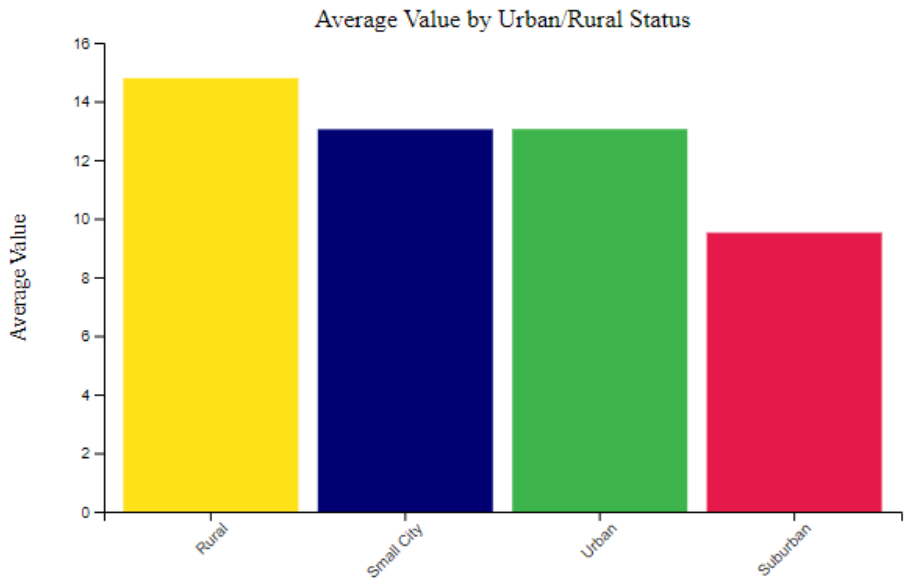


Both maps utilize the tooltip feature, tooltips made the most sense for the maps because then users could identify counties they live in or near and see their specific value based on the attribute selected. Tooltips are better used here than the scatterplot because it makes it easier for users to find specific areas that might be relevant to them. Tooltips wouldn't make as much sense in the scatter plot because it's just a mass of dots so the only thing the user might be curious about are the extremes but you can already find the extremes through the choropleth maps thanks to the color scaling. The user will be able to quickly identify very dark or light regions.



It is also coded to stripe out any region without data as to not skew the color distribution. I should've implemented the same for values of -1 but I forgot to and didn't remember until working on this documentation sadly.

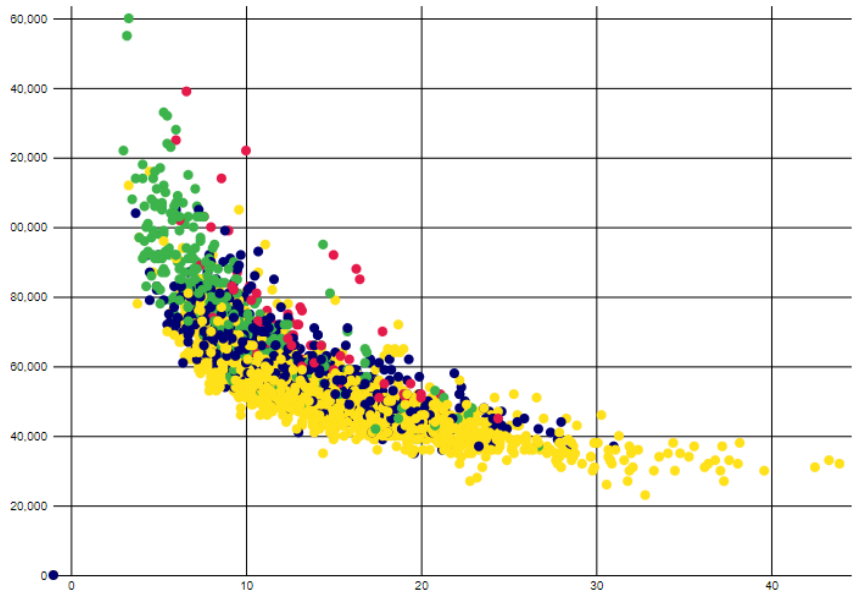
Bar Graph



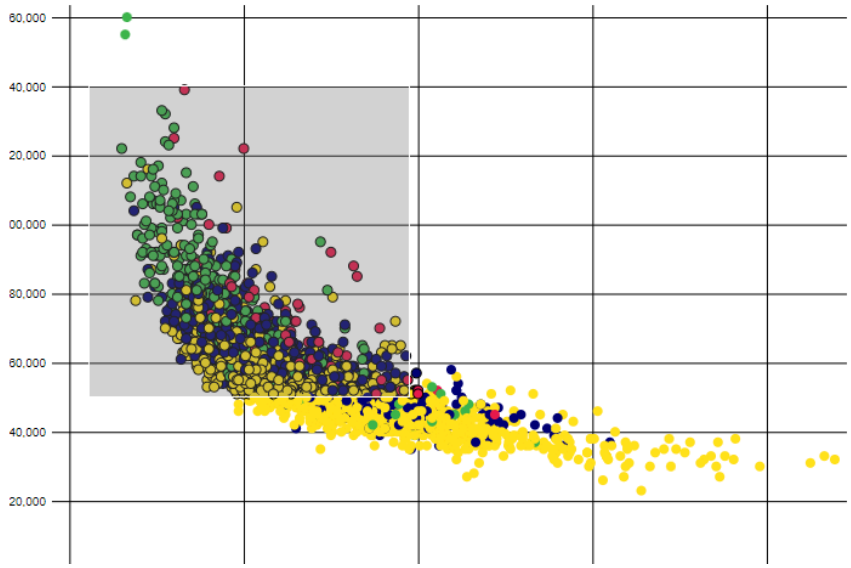
After you press the generate graphs button you will generate a scatter plot and a bar graph based on the drop down selection value. The color coordination for the urban/rural status is the same as the scatterplots so it also works as a legend for the scatterplot. The overall visuals in the application will give you a picture of how health data is distributed throughout the country but this bar graph helps understand just exactly the differences between people who live in the city versus people who live outside the cities. You can sort of pick up on the distribution based on the scatterplot but this bar graph helps summarize what you're seeing and give you a clearer picture.

Which is why it's positioned to the right of the scatterplot. Unfortunately I couldn't get the bar graph to link to the scatterplot but that was an intended feature. The value selected for this example was poverty percentage.

Scatter Plot

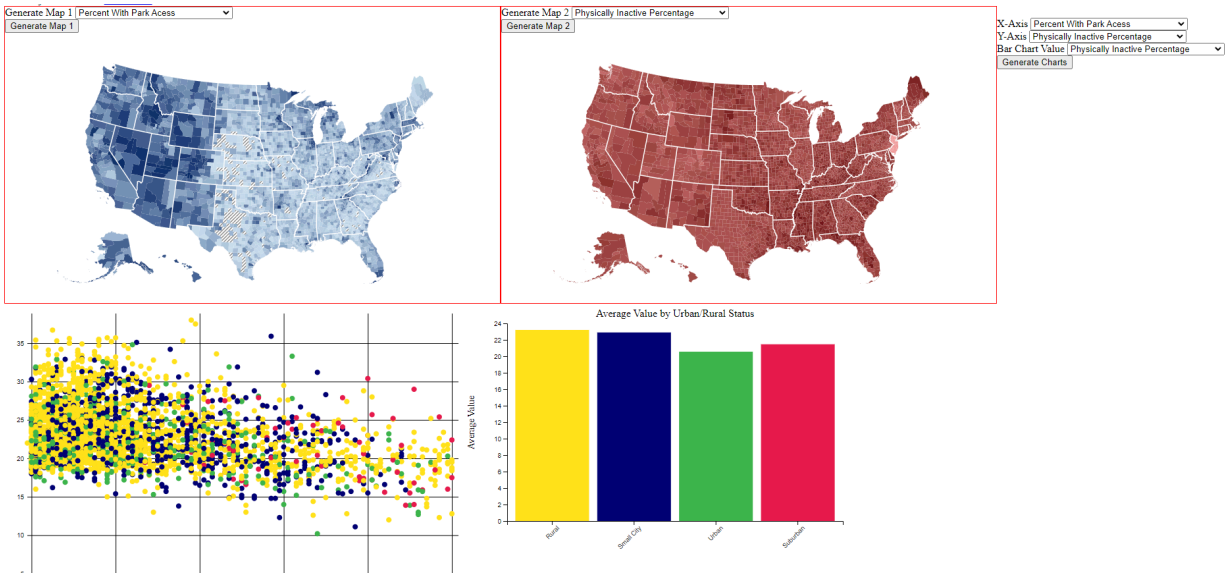


For this example I made the y-axis median household income and the x axis poverty percentage. Like mentioned in the bar graph section each dot is color coordinated based on the urban/rural status of the county it represents. You can select any of the data attributes outside of the urban rural status to compare. Axis labels aren't included however since you set the axis values the user will already know the axis values, so outside of making an optimal UI experience they aren't necessary to understand the data.



I was able to add a brush feature for the scatterplot that would ideally have linked with my other visuals but unfortunately I couldn't get that to work in time. It does at least help highlight the points and more clearly separate them.

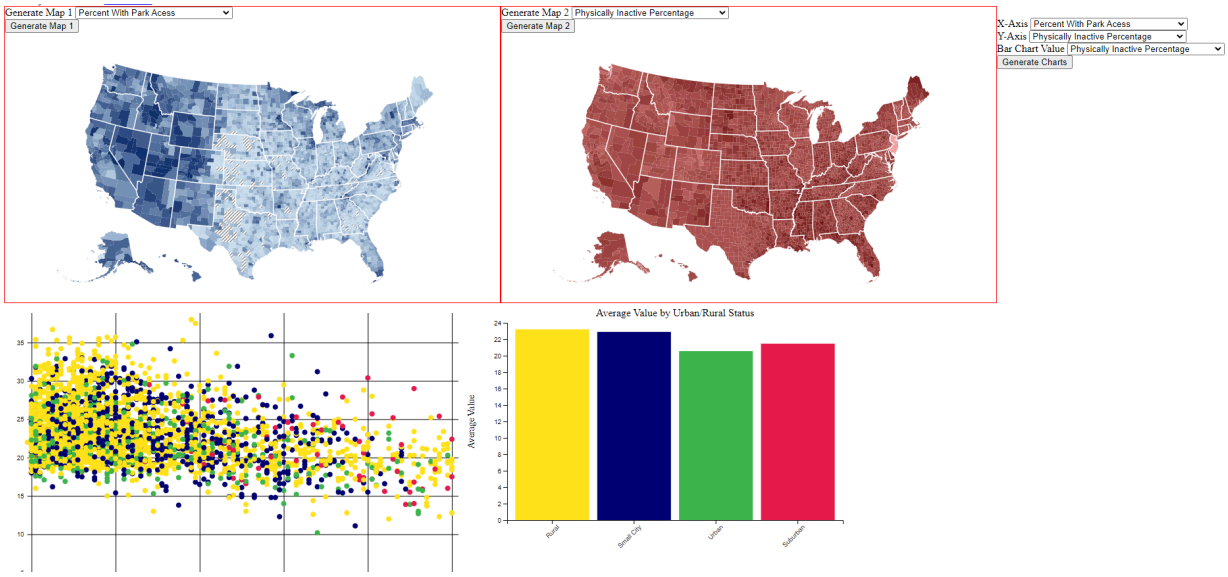
Overall Visualizations



This is how the application looks when all 4 visuals are displayed and correlated. Each visualization is unique in what it's showing but also has a connection with the others. I believe that these 3 visualization types complement each other very well and help the user understand the data in front of them, and with a bit of cleanup in the UI could be a great application.

Application Visual Discoveries:

For this section I'll work through 2 combinations of data attributes and discuss what my application helps me understand.



Comparing attributes:

X axis: Percent with Park Access

Y axis: Physically Inactive Percentage (bargraph value)

Choropleth observation:

From the map we can see the massive disparity of park access between the east coast and west coast. This isn't too surprising considering the geographically layout of the two regions but I didn't realize just how big the disparity was.

Scatterplot observation:

Surprisingly there isn't a disparity in physical activity based on park access. There is definitely some difference especially when you compare the two ends of the plot but it's not as significant as I figured it would be.

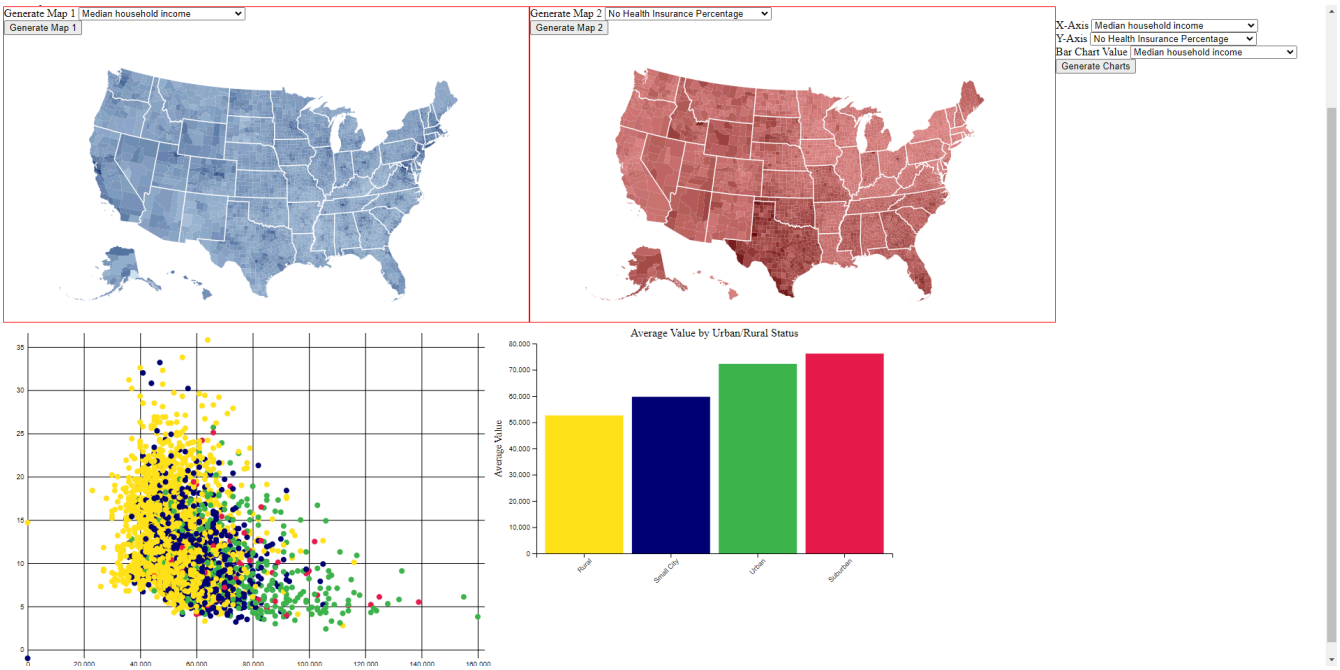
Bar Graph observation:

This data also surprised me. I definitely thought that rural areas would be more active than urban, but funny enough it's the opposite. Maybe because private transport isn't very available in urban areas so people walk more. However the distribution is pretty consistent no matter what type of region you live in.

Overall observation:

From this combination of data I can assume that people on the west coast are more physically active than those on the east and most likely healthier. Where you live by region

doesn’t determine your physical activity level much but living closer to a park will make you more active.



Comparing attributes:

X axis: Median household income (bargraph value)

Y axis: No Health Insurance Percentage

Choropleth observation:

The most staggering observation from the maps is how many people in Texas don’t have health insurance. It would make sense considering there are so many immigrants there, who most likely dont have the money to afford insurance, since they probably migrated here in the first place for economic reasons.

Scatterplot observation:

The less money you make the more likely you are to not have health insurance. Also if you don’t have health insurance you’re more likely to live in a rural area. I'm curious if that's because of the conservative nature of those areas or just because those areas have lower income levels.

Bar Graph observation:

This graph doesn’t surprise me much, especially since those income levels most likely directly correlate to cost of living and cost of living is often cheaper in more rural areas. I

was surprised to see that suburban areas make more money than urban areas. I guess that would make sense though since they're more home owners in the suburbs.

Overall observation:

It was interesting how there are high percentages of no health insurance by the border and other rural areas, but not in the cities where I feel like poverty is just as prevalent. Maybe there's just more people in the cities to balance out the averages. The highest incomes seem to come from cities, especially ones right by the coast, which makes sense. It shouldn't be too surprising that your income level determines how likely you are to have health care which is probably why there are so many pushes for universal health care.

Code Process/Structure:

To run my code I simply used the live server feature in vs code, however if you want to access it from my github pages the link is:
https://tmiller649.github.io/Data-Vis-Project-1/Data-Vis-Project1-TerranceMiller/src/index.html

My code uses d3 version 6.0 which is also linked on my website for reference. I created 6 files a style.css, index.html, choroplethMap.js, scatterplot.js, main.js, and barchart.js, so 3 standard files and then a file for the three visualization types I utilized. There isn't too much to discuss about my index and stylesheet because I didn't have time to clean those up as explained earlier but I will briefly discuss unique features for my javascript files. I followed a similar coding structure to the examples we discussed in class.

For this section I'll break down each code file briefly and give a general overview of any unique features.

ChoroplethMap.js

```
initVis(mapDataString, colorRange) {
  let vis = this;

  vis.colorScale = d3.scaleLinear()
  .domain(d3.extent(vis.data.objects.counties.geometries, d => d.properties[mapDataString]))
    .range(colorRange)
    .interpolate(d3.interpolateHcl);
```

For this class I had the initVis actually take in parameters of the mapDataString which I get from the value of my assigned dropdown menu. Then I have it taken in a color range because both my maps have different distinct colors.

```
.attr("d", vis.path)
.attr('fill', d => {
    if(d.properties[mapDataString]){
      return vis.colorScale(d.properties[mapDataString]);
    }
    else{
      return 'url(#lightstripe)';
    }
  });
```

The code for my choropleth map is pretty standard to a normal map set up outside of we have a check to make sure we have a proper value for the property we're checking. If not then we just stripe out the region so we don't skew the visual.

```
vis.counties
    .on('mousemove', (event, d) => {
      const popDensity = d.properties[mapDataString] ? `<strong>${d.properties[mapDataStr
      d3.select('#tooltip')
        .style('display', 'block')
        .style('left', (event.pageX + vis.config.tooltipPadding) + 'px')
        .style('top', (event.pageY + vis.config.tooltipPadding) + 'px')
        .html(`
          <div class="tooltip-title">${d.properties.name}</div>
          <div>${popDensity}</div>
        `);
    })
    .on('mouseleave', () => {
      d3.select('#tooltip').style('display', 'none');
    });
```

Here I show the tooltip hovering over each county. The popDensity should just say general mapDataValue but that's a small detail I missed because I ran out of time and needed to move the code into submission but will definitely be something to clean up after this project is graded.

scatterplot.js

There's not much to highlight for the scatterplot code outside of how the domain for the color scale is each connected to a value of urban_rural_status with each color being very distinct from the others.

```
// Initialize scales
vis.colorScale = d3.scaleOrdinal()
    .range(['#ffe119', '#000075', '#3cb44b', '#e6194B'])
    .domain(['Rural','Small City','Suburban', 'Urban']);
```

There is also my brushing feature on the scatterplot:

```
        // Add brushing
    d3.select("#scatterplot")
        .data(vis.data)
        .call( d3.brush()
        .extent( [ [0,0], [vis.width, vis.height] ] )
        .on("start brush", updateChart)

    function updateChart() {
    let extent = d3.brushSelection(this)
    point.classed("selected", function(d){return isBrushed(extent,  vis.xScale(vis.xValue(d)), vis.ySca
  }

  // A function that return TRUE or FALSE according if a dot is in the selection or not
  function isBrushed(brush_coords, cx, cy) {
    var x0 = brush_coords[0][0],
        x1 = brush_coords[1][0],
        y0 = brush_coords[0][1],
        y1 = brush_coords[1][1];
   return x0 -25 <= cx && cx <= x1 -25 && y0 -25 <= cy && cy <= y1 - 25;


  }
```

It calculates what point's are inside the brush area, and classify them as selected. For now being classified as selected only gives them the important tag but with more time it should be able to branch out and give them different features and links since they're all under the same class when brushed thanks to the updateChart function.

Barchart.js

Using the same method of grabbing the string value from a drop down menu the barchart class also calculates the average value based off the urban rural status.

```
//Calculates average data based on the 4 regions
const averages = d3.rollup(this.data, v => d3.mean(v, d => d[yValueDataString]), d => d.urban_rural_status);

//moves averages in an array connected to urban_rural_status to use later
const averageArray = Array.from(averages, ([key, value]) => ({ urban_rural_status: key, yValueDataString: value }));
```

It then stores the averages in another array that connects the averages to the urban_rural_status. We can then pull that for the four bars on the bar graph based on average data.

```
const y = d3.scaleLinear()
    .domain([0, d3.max(averageArray, d => d.yValueDataString)])
    .nice()
    .range([vis.height, 0]);

// Add x-axis
```

Were now determining the max Y value based on our array of average values instead of the values themselves since some extremes in the data could make our bar graph look bad. The .nice() rounds our data into a normal number.

Main.js

My main.js code is probably my sloppiest because I was working on getting proof of concept for my features, so I went with the easiest way of just declaring them and erasing the previous graph and making a new one when values are changed. The main.js initially creates two choropleth graphs as explained in earlier sections. There's not much else to discuss about this file because I was unable to get dynamic features like linking properly working, just simple on click functions for my buttons.

```
//Generate button to generate your scatter and bar plot
d3.select('#GeneratePlot').on('click', function() {
  d3.selectAll("#scatterplot > *").remove();
  d3.selectAll("#barchart > *").remove();
  let data, scatterplot, barchart;
  d3.csv('data/national_health_data.csv')
  .then(_data => {
```

Future Works

As previously mentioned there were many features I tried to implement but couldn't figure it out in time, some of them and new ideas I thought of while filing out the documentation are:

- Adding a brush for the choropleth maps to better filter the data
- Cleaning up the general UI
- Producing two bar graphs one for each axis not sure why I didn't think of it before now.
- Linking my brush on my scatterplot with the bar chart to filter out unbrushed data points.
- Adding an on click event for each bar on the bar chart to filter out regions from the scatterplot.

Challenges

Many of the future works I tried to implement but couldn't figure out in time. One of the biggest challenges I found with using d3 is that debugging can be very difficult sometimes. The error messages can be very unhelpful and I would spend lots of time on issues that often had simple fixes.

Another challenge I found is the consistent disparity of versions with online resources. Most online resources use different versions and there are drastic differences in each version which can make it hard to troubleshoot or implement new features.

Overall I thoroughly enjoyed this project, it was very challenging but rewarding. I didn't realize just how much time it would take to meet all the goals for this application and while I allotted dozens of hours I definitely need several more to make this application exactly what I wanted. Shortcomings aside I feel like this project has helped me build a strong base with D3. In future projects I think I'll work on driving certain aspects to completion instead of doing part of them and then jumping to something else. I feel like If I just would've perfected each visualization before moving on to the next I would've saved myself some time and made the linking process much easier.