Visualization Assignment 1

Aishwarya Vijayakumar

112673842

Demo Link:

https://youtu.be/oN_b79779uI

Introduction

In this assignment I created visualization graphs using d3 Javascript library. I used housing data set from King County. The dataset consists of 21 columns. I'm plotting graphs for 13 columns including 7 categorical columns and 6 numerical columns.

Functionalities

- User can pick a column name from the dropdown menu list to update the graph.
- For numerical columns, histograms are plotted by binning the values in a fixed range. Here, the Y-axis is the frequency of the column in that particular range value.
- The bin size for the histograms can be changed from 1 to 20 by using the slider present on the right side of the graph
- For categorical columns, bar graphs are plotted with frequency on the Y-axis and categories on X-axis
- The slider is disabled when the user chooses categorical columns since bin size is not required for this graph.
- On hovering over the graph, the value of the bar(Frequency) is displayed on top of the bar
- On hovering over the graph, the height and width of the bar increases with a change in color of the bar.

Implementation

- Data is loaded into the graphs from a csv file (house_data.csv) as shown in the code below

```
d3.csv("./data/house_data.csv").then(function(data) {
    csvInArr = data;
    beginFunc();
});
```

- The beginFunc() is called which generates the data arrays from the csv file and uses this to plot the graphs. The graphs are plotted based on the column category(numerical/categorical)
- For histograms, two functions are used to obtain the values on the x axis (by using bin counts) and their respective frequencies.

```
function generateDataForBarGraph(colName) {
    let dataArray = generateDataArrayHist(colName);
    let binArray = generateBinArrayHist(dataArray);
    return generateDataForHist(dataArray, binArray);
}
```

generateDataArrayHist() -> returns an array of all the values present in a particular column

generateBinArrayHist() -> returns an array of all the values between the maximum and minimum values in the array generated by the previous method. The values are incremented by the binSize value

These arrays are passed to generateDataForHist() to obtain an object consisting of the column values and its respective bin value.

These functions return the required arrays to plot the histograms. For plotting of the histogram, histChart() method is called. Here, I used scaleBand() function for scaling X-axis and scaleLinear() for scaling Y-axis. Used this to plot the graph in the SVG.

```
const yScale = d3.scaleLinear()
                 .domain([0,d3.max(dataArr.map(function(d) {
                     return d.y;
                 }))])
                 .range([graphHeight,0]);

const xScale = d3.scaleBand()
                 .domain(dataArr.map(function(d) {
                     return d.binVal;
                 }))
                 .range([0, svgWidth]);

const xAxis = d3.axisBottom()

                .scale(xScale);

const yAxis = d3.axisLeft()
                .scale(yScale);
```

- For Bar chart, I constructed an object by using dictionary. This object consists of the values in this format:

{xval: “ “, yval: “ “}

This object is passed to the barchart() function to plot the graph

```
    x.domain(data.map(function(d) { return d.xaxisvalues; }));
    y.domain([0, d3.max(data, function(d) { return d.yaxisvalues; })]);
```

Rest of the process is similar to the histogram construction.

- For hovering functionality, I used mouseover and mouseleave to get the desired results.

```
.on("mouseover", function(d) {
  let currRect = d3.select(this);
  let xVal = currRect.attr("x");
  let yVal = currRect.attr("y");

  currRect.style("fill","darkblue")
          .style("opacity", 1)
           .attr("width", x.bandwidth() +4)
           .attr("height", function(d) { return graphHeight - y(d.yaxisvalues)+10; });

  svg.append("text")
     .attr("class", "barText")
     .attr("x", xVal)
     .attr("y", yVal)
     .attr("transform", "translate(10,-5)")
     .text(d.yaxisvalues);
  })
```

```
.on("mouseleave", function(d) {
    let currRect = d3.select(this);
    currRect.style("fill", "steelblue")
            .style("opacity", 0.8)
            .style("z-index", 1)
            .attr("width", x.bandwidth())
            .attr("y", y(d.yaxisvalues))
            .attr("height",  function(d) { return graphHeight - y(d.yaxisvalues); });

    svg.select(".barText")
            .remove();

    })
```

- For changing the bin size, I took a slider and made slider to go from maximum bin frequency to minimum frequency. This would enable it to move from minimum bin size/width to maximum bin size/width as it goes from left to right.

```
    <div class="contchild2">
        <b>Change the Bin Frequency:</b>
        <input type="range" min="2" max="21" value="11"  class="slider" id="sliderId" onchange="
            binCountHandler()" />
    </div>
```

```
#sliderId{
  direction: rtl
}
```

References

[1] https://www.kaggle.com/shivachandel/kc-house-data/kernels

[2] https://bl.ocks.org/d3noob/bdf28027e0ce70bd132edc64f1dd7ea4