CSE 564 Visualization

Spring 2019

Anish Saha - 112027821

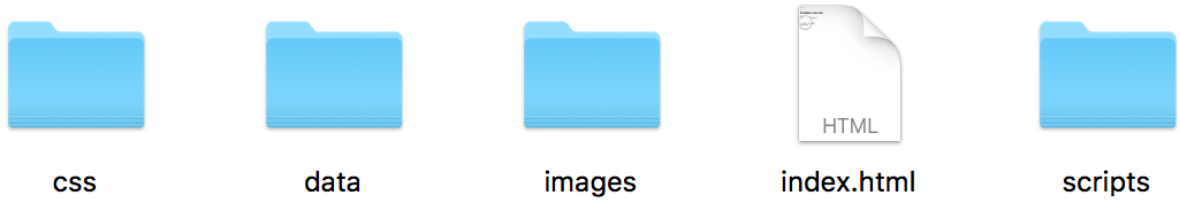
Assignment 1

INTRODUCTION

The main idea of the exercise is to load a .csv file and plot the frequency of a chosen variable as bar chart or pie chart using D3. There has been use of CSS and HTML as well to form the structure of the page layouts. On choosing a variable from a side pane, the variable is binned into buckets and plotted as a histogram. The number of buckets is user chosen with a slider for each selection. On clicking the graph area, the chart transforms to a pie chart. Both charts have highlights on mouse pointer over each portions of the charts. The entire implementation is described subsequently in detail.

DIRECTORY STRUCTURE

/ScreenCapture.mov - Contains a video walkthrough of all functionalities.
/README.txt - Contains important running information for various browser compatibilities.
/src/ - Contains the following files and folders:-



/src/css/ - Contains the styles.css that styles all static elements in the html.
/src/data/ - Contains the .csv datasets that you want to load.
/src/images/ - Contains images that may be used as source images.(Currently empty)
/src/scripts/ - Contains script.js , the javascript source file for the entire code.
/src/index.html - Main HTML pages where page containers are defined.

index.html

```
<head>
    <script src="https://d3js.org/d3.v4.min.js"></script>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
    <script type="text/javascript" src="scripts/script.js"></script>
</head>
```

Link the online d3 libraries. Link the style sheet and the javascript.

```
<div id="header">
    <label id="headLabel"></label>
</div>
```

Create the page header with the header labels that is dynamically assigned as “Bar Graph” or “Pie Chart” as per the type of graph being displayed as per the following code:

```
if(barPieToggle == 1){
    chartType="Y-axis";
    document.getElementById("headLabel").innerHTML = "Bar Graph";
}
else{
    chartType="Pie Chart";
    document.getElementById("headLabel").innerHTML = "Pie Chart";
}
```

The Side Panel contains the “Plot” Button for variable selection out of the available rows in the dataset.

The main content area is divided into two regions. “descBox” or the description box and the “graphHolder” that contains the plots. Data information is displayed in the description box while the plots are constructed in the graph holder area. The description box also contains a slider to choose number of bins for distribution calculation.

```
<div id="content">
    <div id="descBox">
        <h1 id="varName">Variable selected</h1>
        <p id="plotDesc">Distribution will be plotted</p>
        <label id="binValue"></label>
        <input type="range" min="5" max="20" value="12" class="slider" id="rangeSlider" oninput=adjust()>

    </div>
    <div id="graphHolder">

    </div>

</div>
```

Additional footer and footer labels are provided.

script.js

This is the main javascript file. All dynamic logic and drawing of d3 elements are created here. The first call is made to the method **readAndStart( ).** The .csv file is loaded here and the callback function **onLoad( )** is called to operate on the data. First, the options in the side panel are populated and depending on the loading condition (first time load or load after choosing a variable) the description box is populated partially.

Next, the data is bucketed based on the. Number of bins, this is a crucial phase and deals with some edge cases as well.

```
var dataset = [];
for (var c=0;c<numBins;c++)
    dataset.push(0);
console.log("Intialized to: "+dataset);
for (var i =0;i<datasetArray.length;i++){
    if (datasetArray[i]==d3.max(datasetArray)){
        dataset[numBins-1]+=1;
        continue;
    }
    var bin = (datasetArray[i]-d3.min(datasetArray))*numBins/(d3.max(datasetArray)-d3.min(datasetArray));
    if(Math.floor(bin)>numBins)
        console.log("index: "+i+" Chosen bin: " +Math.floor(bin)+ " for value "+datasetArray[i]);
    //console.log("Setting bin: "+bin+" count to "+dataset[Math.floor(bin)]+1);
    dataset[Math.floor(bin)]+=1;

}
```

The next part of the code depends on the variable **barPieToggle.** This determines the creation of a bar chart or pie chart.

For the **Bar Chart**, a main svg container is created as:-

```
var svg = d3.select("#graphHolder")
    .append("svg")
    .attr("width",SVGContainerWidth)
    .attr("height",SVGContainerHeight)
    .on("click",function(){
        if (barPieToggle == 1)
            barPieToggle=0;
        else
            barPieToggle=1;
        //alert("barPieToggle: "+barPie
        onLoad();
    });
```

Now, bars are constructed as rectangle SVG elements. X-position is determined using the transform attribute and height is decided by iterating through the data. Each rectangle is modified using the **onmouseover, onmouseout** attributes. Styling is also provided is the chaining for each element. Subsequently the axes and scales are also defined with their domain and ranges specified. Value display as text, slight height increase and a color change is done on mouse over and exit in the respective functions.

```
var barChart = svg.selectAll("rect")
    .data(dataset)
    .enter()
    .append("rect")
    .attr("y",function(d){
        //console.log("Y is: "+graphHeight-(d*normHeight)+(SVGContainerHeight-gr
        return graphHeight-(d*normHeight)+(SVGContainerHeight-graphHeight)/2;})
    .attr("x",(SVGContainerWidth-graphWidth)/2+barPadding/2)
    .attr("height",function(d){return d*normHeight;})
    .on("mouseover",onHover)
    .on("mouseout",onOut)
    .attr("width",barWidth - barPadding)
    .style("fill","#004d80")
    .attr("transform",function(d,i){
        var translate = [barWidth*i,0];
        return "translate("+ translate +")";
    })
```

For the **Pie Chart**, a “svg” element is created just as in the case of the bar chart. Arcs are constructed with a specified radius and fed to the **d3.pie( )** call while iterating thought the bucketed data. Color is chosen for each arc from a color array and opacity is maintained as 0.8 for highlights on mouse hover. Each arc contains **onmouseover, onmousemove** and **onmouseout** methods to increase arc radius, opacity and append text to the cursor position and display bucket frequency and bucket range. The main pie chart definition code is as below. For extensive cover of the mouse hover kindly refer to the main code file.

```
var color = d3.scaleOrdinal(d3.schemeCategory10);
var g = svg.append('g')
           .attr('transform', 'translate(' + (radius+(SVGContainerWidth-pieChartWidth)/2) + ',' + (radi

var arc = d3.arc()
            .innerRadius(0)
            .outerRadius(radius);

var pie = d3.pie()
            .value(function(d) { console.log(d);return d; })
            .sort(null);

var path = g.selectAll('path')
            .data(pie(dataset))
            .enter()
            .append("g")
            .append('path')
            .attr('d', arc)
            .attr('fill', (d,i) => color(i))
            .style('opacity', 0.8)
            .style('stroke', 'white')
            .on("mouseover", function(d,i) {
```

CONCLUSION

The exercise provides an extensive coverage of the basic implementation of D3 using SVG elements to create simple graphs for data visualization. Coupled with conventional html and css formatting, D3 Is a powerful tool to quickly visualize large datasets to get a comprehensive understanding of data.