Bharathkrishna Guruvayoor Murali
ID : 110945903

VISUALIZATION LAB 1 REPORT

Dataset : aircraft.csv

The data used contains various parameters that define the structure of an aircraft, like Aspect Ratio, Cost, Thrust, Lift-to-drag ratio, Weight and Cost. All these variables are represented using bar chart and Pie chart.

The file src/redraw.js contains the code to change the bin width and plot the graph according to the mode selected. Bin width is calculated by :
binWidth = (max_in_array - min_in_array) / binDivisions;

The frequencies of all the bins are calculated and stored in the 'frequencies' array. The values in this array is used to plot the graph.

Drawing the bar chart

The bars are plotted by adding rectangles 'rect' in the svg in HTML using d3. The data used is the 'frequencies' array. Colors are used in the graphs from d3.schemeCategory20. The width of the bars is set as width of svg divided by total number of elements in frequencies array. A padding of 10 is subtracted to maintain the width between the bars. A snippet performing these operations is shown below:

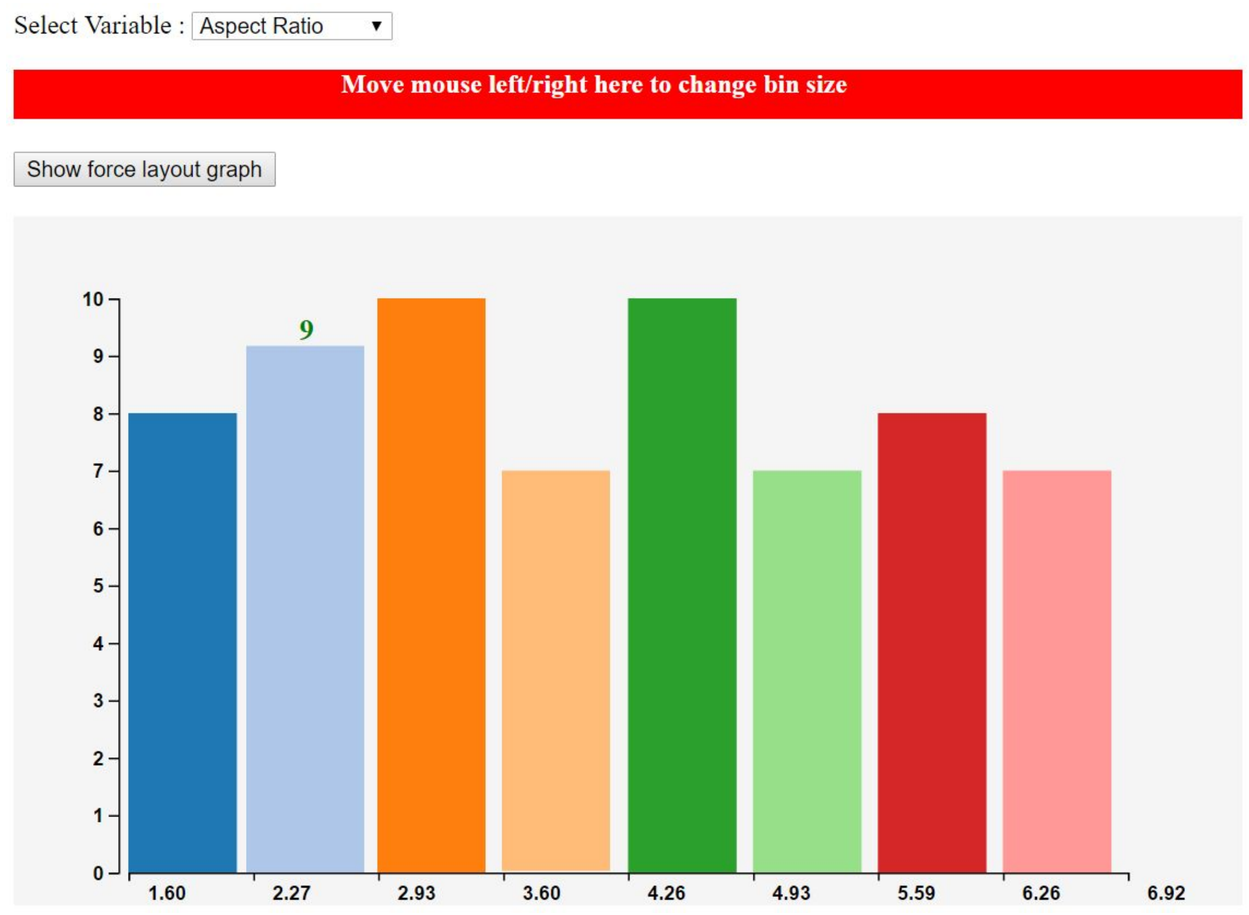
```
 bars = svg.selectAll("rect")
        .data(frequencies)
        .enter()
        .append("rect")
        .attr("x", 0)
        .attr("width", w / frequencies.length - 10)
        .attr("height", 400)
        .attr("x", function(d, i) {
            return i * (w / frequencies.length);
        })
        .attr("fill", function(d, i) {
            return c20[i % 20];
        });
```

As the coordinate system in svg is measured from top-left to bottom-right, the y coordinate for bar is given as **((max_freq - d) / max_freq) * 400 + 20**; where 20 is used as a padding for top and 400 is the height of the svg. A transition is used to have the bars come up in a duration of 2000ms. The code snippet is shown below:

```
 bars
        .attr('y', 420)
        .transition()
        .duration(2000)
        .attr("y", function(d) {
            return ((max_freq - d) / max_freq) * 400 + 20;
        })
```

Only on mouse-over, the value of the bar is displayed on top of the bar. The menu can be used to allow users to select a new variable and update chart.
On mouse-over, the bar is made wider and higher to focus on it. When mouse is moved out, the bar is reverted to original size. When mouse moves left on the div above the chart, the bin size is decreased . When mouse moves right on the div above the chart, bin width/size is increased. All the code related to bar chart can be found in src/bar_chart.js.
The code for changing the variables and change of bin size is present in src/main.js, and it works in both bar and pie charts.
The x axis represents the bin width and the y axis represents the frequency.
The x and y axis labels are also updated according to the change of variables and bin size.

BAR CHART



The code for rendering pie chart using d3 is present in src/pie_chart.js.

The radius is taken as the half of minimum of width and height.
radius = Math.min(w, h) / 2;
The inner radius of pie chart will be 0, and outer radius is kept as radius-40 to accommodate for padding on transitions.
VARIABLE SELECTION : An HTML select element is used to create a variable picker. Based on the variable selected, the chart is re-drawn by calculating the frequencies of the selected variable.

Drawing the pie chart

On mouse-click the bar chart is transformed into a pie chart (and back).

```
  var arc = d3.arc()
      .outerRadius(radius - 40)
      .innerRadius(0);
```

The labels are put at a point radius-70 for them to appear inside the arcs when they are mouse-hovered.

```
 var labelArc = d3.arc()
      .outerRadius(radius - 70)
      .innerRadius(radius - 70);
```

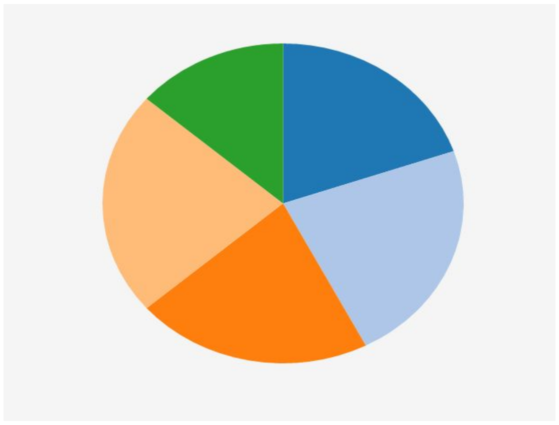
The same ‘frequencies’ array is used to plot the pie chart, sort(null) is used as we want to present the chart in the natural order of elements.

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

  var arcOver = d3.arc()
      .outerRadius(radius - 20)
      .innerRadius(0);
```

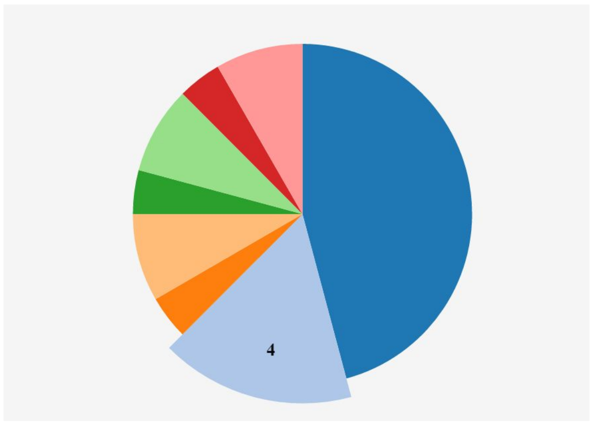
The pie chart is plotted inside the ‘g’ element, and the center is chosen at the (w/2,h/2) point, which is the center of the svg element.

```
  var mainGroup = svg.append("g")
      .attr("transform", "translate(" + w / 2 + "," + h / 2 + ")");

  var g = mainGroup.selectAll(".arc")
      .data(pie(frequencies))
      .enter().append("g")
      .attr("class", "arc");
```

The pie chart also has a transition. The chart shows the value of variable when hovered, it also increases the size of the current slice to focus on it.



PIE CHART



PIE CHART ON MOUSEOVER

EXTRA CREDIT PART :

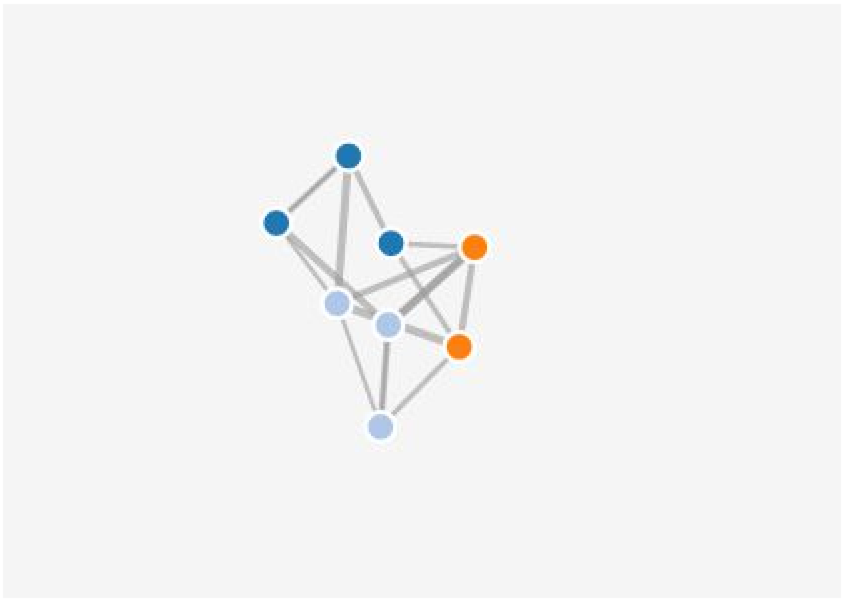
FORCED LAYOUT GRAPH

A json object has been created to plot a basic forced layout graph. The JSON is created such that the nodes are created which are divided into different groups. The links define mapping between different nodes and the weight associated between them.
The forced-layout is created using the forceSimulation function in d3.

```
 var simulation = d3.forceSimulation()
        .force("link", d3.forceLink().id(function(d) {
            return d.id; }))
        .force("charge", d3.forceManyBody())
        .force("center", d3.forceCenter(width / 2, height / 2));
```

The JSON is parsed and the links and nodes are created using the following snippet of code :
Graph.links shows the links part from Json and graph.nodes is the nodes part from json. The links are basically SVG lines and nodes are created using SVG circles.
Function callbacks are created for the drag events associated with the nodes, which change the x and y values of node to give the dragging effect.

```
 d3.json("force_directed.json", function(error, graph) {
        if (error) throw error;

        var link = svg.append("g")
            .attr("class", "links")
            .selectAll("line")
            .data(graph.links)
            .enter().append("line")
            .attr("stroke-width", function(d) {
              return Math.sqrt(d.value); });

        var node = svg.append("g")
            .attr("class", "nodes")
            .selectAll("circle")
            .data(graph.nodes)
            .enter().append("circle")
            .attr("r", 5)
            .attr("fill", function(d) {
              return color(d.group); })
            .call(d3.drag()
              .on("start", dragstarted)
              .on("drag", dragged)
              .on("end", dragended));
```



SOURCE CODE : https://github.com/bharos/visualization_bar_graph

REFERENCES

1. https://bl.ocks.org/ Pie chart examples
2. Bar chart tutorials from https://bl.ocks.org/d3noob/bdf28027e0ce70bd132edc64f1dd7ea4
3. Forced directed tutorial https://bl.ocks.org/mbostock/4062045
4. Visualization Lecture Slides from Professor Klaus Mueller