

HW3

Part I

Problem 1

- 1.1.1 In Mathematical/Graph Coordinate Space, the x axis increases from left to right and the y axis increases from down to up. However, in the SVG Coordinate Space, the y axis increases form up to down.
- 1.1.2 `enter()` is used to selects the missing DOM elements compared to the data. `exit()` is used to select those DOM elements that are extra compared to the data.
- 1.1.3 “Transform” refers to the methods to makes changes to the SVG graph, including “translate”, “rotate”, “slew”, etc.
“Translate” is used to move the SVG graph without changing its size and shape. All points on this element move in the same direction and the same distance relative to the transformation reference point.
- 1.1.4 The return value is [5, 6, 7, 8, 9]. The function add 5 to the index of each element in the list.

- 1.1.5 The codes are:

```
<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>1.1.5</title>
    <script src="https://d3js.org/d3.v5.min.js"></script>
  </head>

  <body>
    <script>
      var data = ['big', 'data', 'assignment', 2, 'submission'];
      var body = d3.select("body")
        .selectAll("span")
        .data(data)
        .enter()
        .append("span")
        .text(function(d) {
          return d + " ";
        });
    </script>
  </body>

</html>
```

And

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>1.1.5_2</title>
8      <script src="https://d3js.org/d3.v5.min.js"></script>
9  </head>
10
11 <body>
12 <p>D3 Tutorials </p>
13
14 <script>
15     var myData = ['big', 'data', 'assignment', 2, 'submission'];
16
17     var p = d3.select("body")
18         .selectAll("p")
19         .data(myData)
20         .text(function (d,i) {
21             return d;
22         });
23 </script>
24 </body>
25
26 </html>

```

The outputs are:

big data assignment 2 submission

And

big

In the first snippet, as the body is empty, enter() gets the part of the data that is not yet bound to any element and holds place for creating DOM elements and then the program fills the places basing on the function.

In the second snippet, because enter() is not used, D3.js only processes the first element

'big' of the array by default and assigns it to all <p> elements.

Problem 2

1.2.1 The code is:

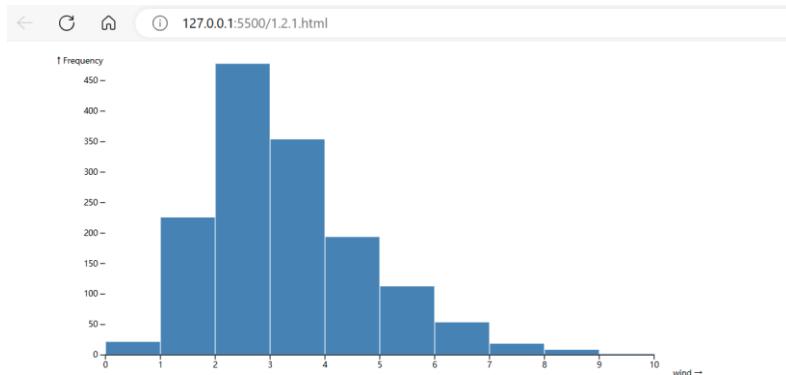
```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Seattle Weather Data Visualization</title>
8      <script src="https://d3js.org/d3.v5.min.js"></script>
9  </head>
10
11 <body>
12     <div id="chart"></div>
13     <script>
14         const width = 800
15         const height = 400
16         const margin = {
17             top: 10,
18             bottom: 30,
19             left: 60,
20             right: 60
21         }
22
23         var svg = d3.select('#chart')
24             .append('svg')
25             .attr('width', width + margin.left + margin.right)
26             .attr('height', height + margin.top + margin.bottom)
27             .append('g')
28             .attr('transform', `translate(${ margin.left }, ${ margin.top })`)
29
30         d3.csv("seattle-weather.csv").then(function(data){
31             var histogram = d3.histogram()
32                 .value(function(d) {return d.wind})
33                 .domain([0,10])
34                 .thresholds(10);
35
36             var bins = histogram(data)
37             console.log(bins);
38             //define x axis and y axis
39             const x = d3.scaleLinear()
40                 .domain([bins[0].x0, bins[bins.length - 1].x1])
41                 .range([margin.left, width - margin.right]);
42             const y = d3.scaleLinear()
```

```

42 |     const y = d3.scaleLinear()
43 |     .domain([0, d3.max(bins, (d) => d.length)])
44 |     .range([height - margin.bottom, margin.top]);
45 |
46 |
47 //add rectangulars to the figure
48 svg.append("g")
49     .attr("fill", "steelblue")
50     .selectAll()
51     .data(bins)
52     .join("rect")
53     .attr("x", (d) => x(d.x0) + 1)
54     .attr("width", (d) => x(d.x1) - x(d.x0) - 1)
55     .attr("y", (d) => y(d.length))
56     .attr("height", (d) => y(0) - y(d.length));
57
58 //add axis to the figure
59 svg.append("g")
60     .attr("transform", `translate(0,${height - margin.bottom})`)
61     .call(d3.axisBottom(x).ticks(width / 80).tickSizeOuter(0))
62     .call((g) => g.append("text")
63         .attr("x", width)
64         .attr("y", margin.bottom - 4)
65         .attr("fill", "currentColor")
66         .attr("text-anchor", "end")
67         .text("wind →"));
68
69 svg.append("g")
70     .attr("transform", `translate(${margin.left},0)`)
71     .call(d3.axisLeft(y).ticks(height / 40))
72     .call((g) => g.select(".domain").remove())
73     .call((g) => g.append("text")
74         .attr("x", -margin.left)
75         .attr("y", 10)
76         .attr("fill", "currentColor")
77         .attr("text-anchor", "start")
78         .text("↑ Frequency"));
79 });

```

The output is:



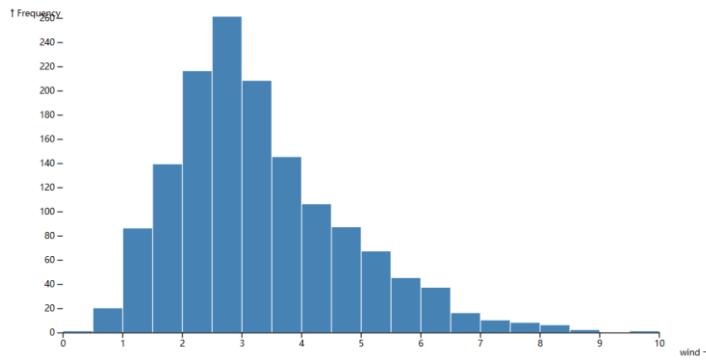
1.2.2 The code is:

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Seattle Weather Data Visualization</title>
8      <script src="https://d3js.org/d3.v5.min.js"></script>
9  </head>
10
11 <body>
12     <div id="chart"></div>
13     <script>
14         const width = 800
15         const height = 400
16         const margin = {
17             top: 10,
18             bottom: 30,
19             left: 60,
20             right: 60
21         }
22
23         var svg = d3.select('#chart')
24             .append('svg')
25             .attr('width', width + margin.left + margin.right)
26             .attr('height', height + margin.top + margin.bottom)
27             .append('g')
28             .attr('transform', `translate(${margin.left}, ${margin.top})`)
29
30         d3.csv("seattle-weather.csv").then(function(data)[
31             var histogram = d3.histogram()
32                 .value(function(d) {return d.wind})
33                 .domain([0,10])
34                 .thresholds(20);
35
36             var bins = histogram(data)
37             console.log(bins);
38             //define x axis and y axis
39             const x = d3.scaleLinear()
40                 .domain([bins[0].x0, bins[bins.length - 1].x1])
41                 .range([margin.left, width - margin.right]);
42             const y = d3.scaleLinear()
43
44             const y = d3.scaleLinear()
45                 .domain([0, d3.max(bins, (d) => d.length)])
46                 .range([height - margin.bottom, margin.top]);
47
48             //add rectangulars to the figure
49             svg.append("g")
50                 .attr("fill", "steelblue")
51                 .selectAll()
52                 .data(bins)
53                 .join("rect")
54                 .attr("x", (d) => x(d.x0) + 1)
55                 .attr("width", (d) => x(d.x1) - x(d.x0) - 1)
56                 .attr("y", (d) => y(d.length))
57                 .attr("height", (d) => y(0) - y(d.length));
58
59             //add axis to the figure
60             svg.append("g")
61                 .attr("transform", `translate(0,${height - margin.bottom})`)
62                 .call(d3.axisBottom(x).ticks(width / 80).tickSizeOuter(0))
63                 .call((g) => g.append("text")
64                     .attr("x", width)
65                     .attr("y", margin.bottom - 4)
66                     .attr("fill", "currentColor")
67                     .attr("text-anchor", "end")
68                     .text("wind →"));
69
70             svg.append("g")
71                 .attr("transform", `translate(${margin.left},0)`)
72                 .call(d3.axisLeft(y).ticks(height / 40))
73                 .call((g) => g.select(".domain").remove())
74                 .call((g) => g.append("text")
75                     .attr("x", -margin.left)
76                     .attr("y", 10)
77                     .attr("fill", "currentColor")
78                     .attr("text-anchor", "start")
79                     .text("↑ Frequency"));

```

The output is:



1.2.3 The code is:

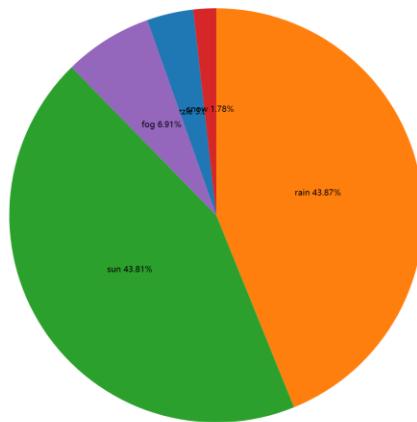
```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Seattle Weather Data Visualization</title>
8      <script src="https://d3js.org/d3.v5.min.js"></script>
9  </head>
10 <body>
11     <svg width="400" height="400" id="pie-chart"></svg>
12     <script>
13
14         var width = 1600;
15         var height = 1600;
16         var radius = Math.min(width, height) / 2;
17         var svg = d3.select('#pie-chart')
18             .attr('width', width)
19             .attr('height', height)
20             .append('g')
21             .attr('transform', 'translate(' + width / 2 + ',' + height / 2 + ')');
22
23         d3.csv("seattle-weather.csv").then(function(data){
24             var weatherCounts = data.reduce(function(counts, data) {
25                 counts[data.weather] = (counts[data.weather] || 0) + 1;
26                 return counts;}, {});
27             var totalObservations = Object.values(weatherCounts).reduce(function(sum, count) {
28                 return sum + count;
29             }, 0);
30             var weatherPercentages = {};
31             for (var weather in weatherCounts) {
32                 var count = weatherCounts[weather];
33                 var percentage = (count / totalObservations * 100).toFixed(2);
34                 weatherPercentages[weather] = percentage;
35             }
36
37             var pie = d3.pie()
38                 .value(function(d) { return d.value; });
39
40             var pieData = pie(Object.entries(weatherPercentages).map(function([weather, percentage]) {
41                 return { weather: weather, value: percentage };
42             }));
43         });
44     </script>
45 
```

```

44      var arcs = svg.selectAll('arc')
45        .data(pieData)
46        .enter()
47        .append('g')
48        .attr('class', 'arc');
49
50        var outerRadius = width / 4;
51        var innerRadius = 0;
52        var arc = d3.arc()
53          .innerRadius(innerRadius)
54          .outerRadius(outerRadius);
55
56        var color = d3.scaleOrdinal(d3.schemeCategory10);
57
58        arcs.append('path')
59          .attr('d', arc)
60          .attr('fill', function(d, i) { return color(i); });
61
62        arcs.append('text')
63          .attr('transform', function(d) { return 'translate(' + arc.centroid(d) + ')'; })
64          .attr('text-anchor', 'middle')
65          .text(function(d) { return d.data.weather + ' ' + d.data.value + '%'; });
66
67
68    });
69  </script>
70 </body>

```

The output is:



1.2.4 The code is:

```

1   <!DOCTYPE html>
2   <html lang="en">
3     <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Line Graph</title>
7       <script src="https://d3js.org/d3.v7.min.js"></script>
8       <style>
9         /* Add some basic styles if needed */
10      </style>
11    </head>
12    <body>
13      <svg width="800" height="400" id="line-chart"></svg>
14      <script>
15        // Set the dimensions and margins of the graph
16        var margin = { top: 50, right: 50, bottom: 50, left: 100 },
17        width = 800 - margin.left - margin.right,
18        height = 400 - margin.top - margin.bottom;
19
20        // Parse the date
21        var parseDate = d3.timeParse("%Y-%m-%d");
22        d3.csv("seattle-weather.csv").then(function(data){
23          // Format the data
24          data.forEach(function(d) {
25            d.date = parseDate(d.date);
26            d.precipitation = +d.precipitation;
27          });
28
29          // Set the ranges
30          var xScale = d3.scaleTime().domain(d3.extent(data, function(d) { return d.date; })).range([0, width]);
31          var yScale = d3.scaleLinear().domain([0, d3.max(data, function(d) { return d.precipitation; })]).range([height, 0]);
32
33          // Define the line
34          var line = d3.line()
35            .x(function(d) { return xScale(d.date); })
36            .y(function(d) { return yScale(d.precipitation); });
37
38          // Append the SVG object to the body of the page
39          var svg = d3.select("#line-chart")
40            .attr("width", width + margin.left + margin.right)
41            .attr("height", height + margin.top + margin.bottom)
42            .append("g")

```

```

43            .attr("transform", "translate(" + margin.left + "," + margin.top + ")");
44
45            // Add the line
46            svg.append("path")
47              .data([data])
48              .attr("class", "line")
49              .attr("d", line)
50              .attr("stroke", "black")
51              .attr("fill", "transparent");
52
53            // Add the X Axis
54            svg.append("g")
55              .attr("class", "x-axis")
56              .attr("transform", "translate(0," + height + ")")
57              .call(d3.axisBottom(xScale));
58
59            // Add the Y Axis
60            svg.append("g")
61              .attr("class", "y-axis")
62              .call(d3.axisLeft(yScale));
63
64            // Add X axis label
65            svg.append("text")
66              .attr("class", "x-label")
67              .attr("text-anchor", "middle")
68              .attr("x", width / 2)
69              .attr("y", height + margin.bottom)
70              .text("Date");
71
72            // Add Y axis label
73            svg.append("text")
74              .attr("class", "y-label")
75              .attr("text-anchor", "middle")
76              .attr("transform", "rotate(-90)")
77              .attr("x", -height / 2)
78              .attr("y", -margin.left / 2)
79              .text("Precipitation");
80
81            // Add title
82            svg.append("text")
83              .attr("class", "title")
84              .attr("text-anchor", "middle")

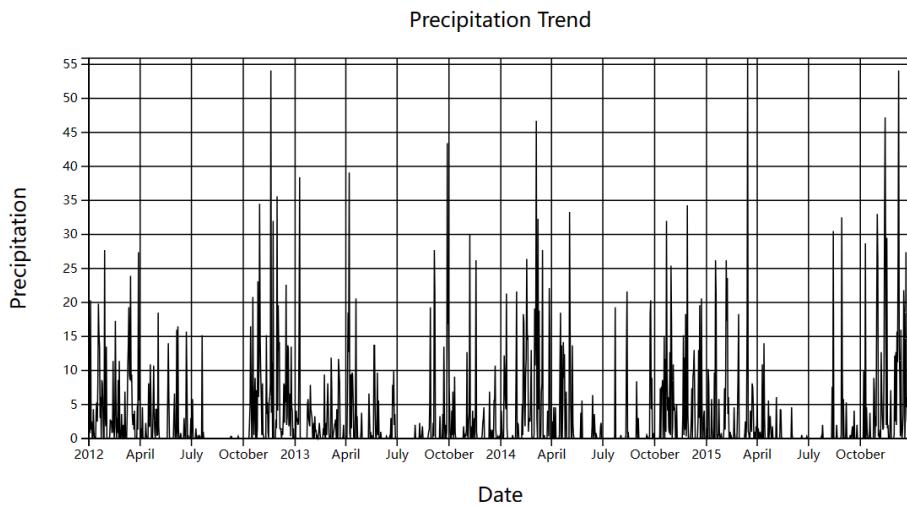
```

```

84     .attr("text-anchor", "middle")
85     .attr("x", width / 2)
86     .attr("y", -margin.top / 2)
87     .text("Precipitation Trend");
88
89     // Add gridlines for y-axis
90     svg.append("g")
91     .attr("class", "gridlines")
92     .call(d3.axisLeft(yScale).tickSize(-width).tickFormat(""));
93
94     // Add gridlines for x-axis
95     svg.append("g")
96     .attr("class", "gridlines")
97     .attr("transform", "translate(0," + height + ")")
98     .call(d3.axisBottom(xScale).tickSize(-height).tickFormat(""));
99   });
100  </script>
101 </body>
102 </html>

```

The output is:



1.2.5 The code is:

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>paragraphs</title>
7    <script src="https://d3js.org/d3.v7.min.js"></script>
8
9  </head>
10
11 <body>
12   <p> From 1.2.1, we can find that there has the greatest possibility that the value of wind is in the domain 2-3.</p>
13   <p> From 1.2.2, there has the greatest possibility that the value of wind is in the domain 2.5-3.</p>
14   <p> From 1.2.3, it is easy to find that rainy days and sunny days have the highest probability of occurrence,
15   | both accounting for about 43% and the probability of snowy days is the smallest.</p>
16   <p> From 1.2.4, it can be found that in the statistical data, there were three days when the precipitation exceeded 50mm,
17   | once in 2012 and twice in 2015.</p>
18
19 </body>
20 </html>

```

The output is:

From 1.2.1, we can find that there has the greatest possibility that the value of wind is in the domain 2-3.

From 1.2.2, there has the greatest possibility that the value of wind is in the domain 2.5-3.

From 1.2.3, it is easy to find that rainy days and sunny days have the highest probability of occurrence, both accounting for about 43% and the probability of snowy days is the smallest.

From 1.2.4, it can be found that in the statistical data, there were three days when the precipitation exceeded 50mm, once in 2012 and twice in 2015.

1.2.6 The code is:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>paragraphs</title>
7   <script src="https://d3js.org/d3.v7.min.js"></script>
8   <style>
9     p {
10       font-size: 16px;
11       color: #rgba(34, 92, 133, 0.828);
12       margin-bottom: 10px;
13     }
14
15     .section-number {
16       font-weight: bold;
17     }
18   </style>
19 </head>
20
21
22 <body>
23   <p> From <span class="section-number">1.2.1</span>, we can find that there has the greatest possibility that
24     the value of wind is in the domain 2-3.</p>
25   <p> From <span class="section-number">1.2.2</span>, there has the greatest possibility that the value of wind is
26     in the domain 2.5-3.</p>
27   <p> From <span class="section-number">1.2.3</span>, it is easy to find that rainy days and sunny days have the
28     highest probability of occurrence, both accounting for about 43% and the probability of snowy days is the smallest.</p>
29   <p> From <span class="section-number">1.2.4</span>, it can be found that in the statistical data, there were three
30     days when the precipitation exceeded 50mm, once in 2012 and twice in 2015.</p>
31 </body>
32 </html>
```

The output is:

From 1.2.1, we can find that there has the greatest possibility that the value of wind is in the domain 2-3.

From 1.2.2, there has the greatest possibility that the value of wind is in the domain 2.5-3.

From 1.2.3, it is easy to find that rainy days and sunny days have the highest probability of occurrence, both accounting for about 43% and the probability of snowy days is the smallest.

From 1.2.4, it can be found that in the statistical data, there were three days when the precipitation exceeded 50mm, once in 2012 and twice in 2015.

Problem 3

1.3.1 The code is:

```

1   <!DOCTYPE html>
2   <html lang="en">
3
4   <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Seattle Weather Data Visualization</title>
8     <script src="https://d3js.org/d3.v5.min.js"></script>
9   </head>
10
11  <body>
12    <div id="chart"></div>
13
14    <p>
15      <label># bins </label>
16      <input type="number" min="5" max="25" step="5" value="20" id="nBin">
17    </p>
18    <script>
19      const width = 800
20      const height = 400
21      const margin = [
22        top: 10,
23        bottom: 30,
24        left: 60,
25        right: 60
26      ]
27
28      var svg = d3.select('#chart')
29      .append('svg')
30      .attr('width', width + margin.left + margin.right)
31      .attr('height', height + margin.top + margin.bottom)
32      .append('g')
33      .attr('transform', `translate(${margin.left}, ${margin.top})`)
34
35      const x = d3.scaleLinear()
36      .range([margin.left, width - margin.right]);
37
38      const y = d3.scaleLinear()
39      .range([height - margin.bottom, margin.top]);
40
41      d3.csv("seattle-weather.csv").then(function(data){
42        function update(nBin){
43
44          var histogram = d3.histogram()
45          .value(function(d) {return d.wind})
46          .domain([0,10])
47          .thresholds(x.ticks(nBin));
48          var bins = histogram(data);
49
50          //define x axis and y axis
51          x.domain([bins[0].x0, bins[bins.length - 1].x1]);
52          y.domain([0, d3.max(bins, (d) => d.length)]);
53
54          svg.selectAll(".x-axis").remove();
55          svg.selectAll(".y-axis").remove();
56          //add axis to the figure
57          svg.append("g")
58            .attr("class", "x-axis")
59            .attr("transform", `translate(0,${height - margin.bottom})`)
60            .call(d3.axisBottom(x).ticks(width / 80).tickSizeOuter(0))
61            .call((g) => g.append("text")
62              .attr("x", width)
63              .attr("y", margin.bottom - 4)
64              .attr("fill", "currentColor")
65              .attr("text-anchor", "end")
66              .text("Wind ↗"));
67
68          svg.append("g")
69            .attr("class", "y-axis")
70            .attr("transform", `translate(${margin.left},0)`)
71            .call(d3.axisLeft(y).ticks(height / 40))
72            .call((g) => g.select(".domain").remove())
73            .call((g) => g.append("text")
74              .attr("x", -margin.left)
75              .attr("y", 10)
76              .attr("fill", "currentColor")
77              .attr("text-anchor", "start")
78              .text("↑ Frequency"));

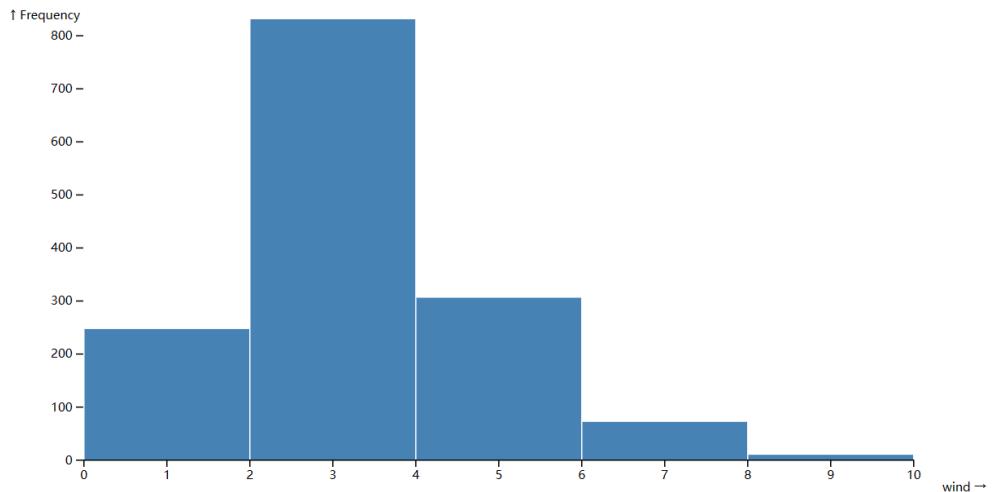
```

```

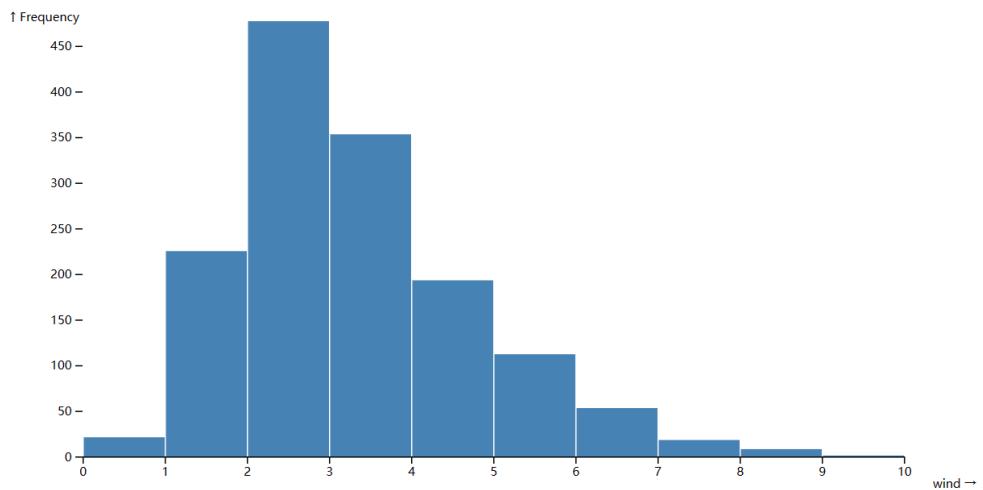
79  var u = svg.selectAll("rect")
80    .data(bins);
81
82  u.enter()
83    .append("rect")
84    .merge(u)
85    .transition()
86    .duration(1000)
87    .attr("x", (d) => x(d.x0) + 1)
88    .attr("width", (d) => x(d.x1) - x(d.x0) - 1)
89    .attr("y", (d) => y(0) - y(d.length))
90    .attr("height", (d) => y(0) - y(d.length))
91    .attr("fill", "steelblue");
92
93  u.exit().remove();
94 }
95
96 update(20)
97
98 </script>
99 </body>
100 </html>
101 });
102
103 </script>
104 </body>
105 </html>

```

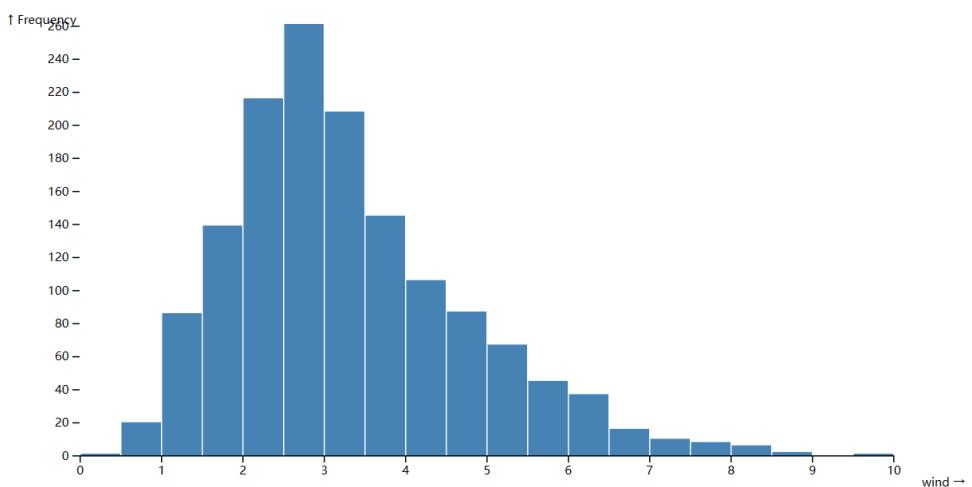
The output is:



bins



bins ▾



bins ▾

1.3.2 The code is:

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Seattle Weather Data Visualization</title>
8      <script src="https://d3js.org/d3.v5.min.js"></script>
9  </head>
10 <body>
11     <div id="chart"></div>
12     <p>
13         <label># bins </label>
14         <input type="number" min="5" max="25" step="5" value="20" id="nBin">
15     </p>
16     <p>
17         <label>Select Variable:</label>
18         <select id="variable">
19             <option value="precipitation">Precipitation</option>
20             <option value="temp_max">Max Temperature</option>
21             <option value="temp_min">Min Temperature</option>
22             <option value="Wind">Wind</option>
23         </select>
24     </p>
25     <script>
26         const width = 800
27         const height = 400
28         const margin = {
29             top: 10,
30             bottom: 30,
31             left: 60,
32             right: 60
33         }
34
35         var svg = d3.select('#chart')
36         .append('svg')
37         .attr('width', width + margin.left + margin.right)
38         .attr('height', height + margin.top + margin.bottom)
39         .append('g')
40         .attr('transform', `translate(${ margin.left }, ${ margin.top })`)
41
42

```

```

43     const x = d3.scaleLinear()
44         .range([margin.left, width - margin.right]);
45
46     const y = d3.scaleLinear()
47         .range([height - margin.bottom, margin.top]);
48
49
50     d3.csv("seattle-weather.csv").then(function(data){
51         var selectedVariable = "precipitation";
52         var minValue = d3.min(data, function(d) { return parseFloat(d[selectedVariable]); });
53         var maxValue = d3.max(data, function(d) { return parseFloat(d[selectedVariable]); });
54
55         function update(nBin){
56             var selectedData = data.map(function(d) {
57                 return parseFloat(d[selectedVariable]);
58             });
59             var histogram = d3.histogram()
60                 .value(function(d) {return d})
61                 .domain([minValue,maxValue])
62                 .thresholds(x.ticks(nBin));
63             var bins = histogram(selectedData);
64
65             //define x axis and y axis
66             x.domain([bins[0].x0, bins[bins.length - 1].x1]);
67             y.domain([0, d3.max(bins, (d) => d.length)]);
68
69             svg.selectAll(".x-axis").remove();
70             svg.selectAll(".y-axis").remove();
71             //add axis to the figure
72             svg.append("g")
73                 .attr("class", "x-axis")
74                 .attr("transform", `translate(0,${height - margin.bottom})`)
75                 .call(d3.axisBottom(x).ticks(width / 80).tickSizeOuter(0))
76                 .call((g) => g.append("text")
77                     .attr("x", width)
78                     .attr("y", margin.bottom - 4)
79                     .attr("fill", "currentColor")
80                     .attr("text-anchor", "end")
81                     .text("→"));
82

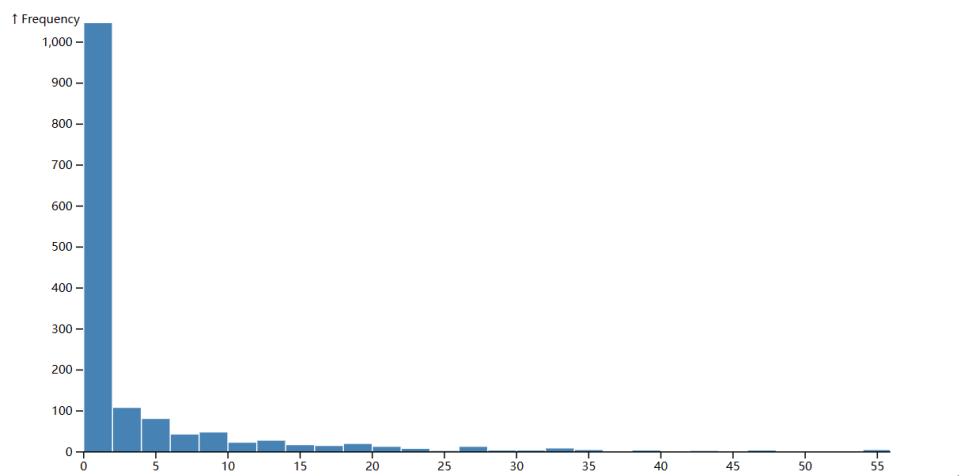
```

```

83     svg.append("g")
84         .attr("class", "y-axis")
85         .attr("transform", `translate(${margin.left},0)`)
86         .call(d3.axisLeft(y).ticks(height / 40))
87         .call((g) => g.select(".domain").remove())
88         .call((g) => g.append("text")
89             .attr("x", -margin.left)
90             .attr("y", 10)
91             .attr("fill", "currentColor")
92             .attr("text-anchor", "start")
93             .text("↑ Frequency"));
94
95     var u = svg.selectAll("rect")
96         .data(bins);
97
98     u.enter()
99         .append("rect")
100        .merge(u)
101        .transition()
102        .duration(1000)
103        .attr("x", (d) => x(d.x0) + 1)
104        .attr("width", (d) => x(d.x1) - x(d.x0) - 1)
105        .attr("y", (d) => y(d.length))
106        .attr("height", (d) => y(0) - y(d.length))
107        .attr("fill", "steelblue");
108
109    u.exit().remove();
110 }
111
112 update(20)
113
114 d3.select("#nBin").on("input", function(){
115     update(+this.value)
116 });
117
118 d3.select("#variable").on("change", function() {
119     selectedVariable = this.value;
120     update(+document.getElementById("nBin").value); // get bins
121 });
122
123 // get initial value
124 update(+document.getElementById("nBin").value);

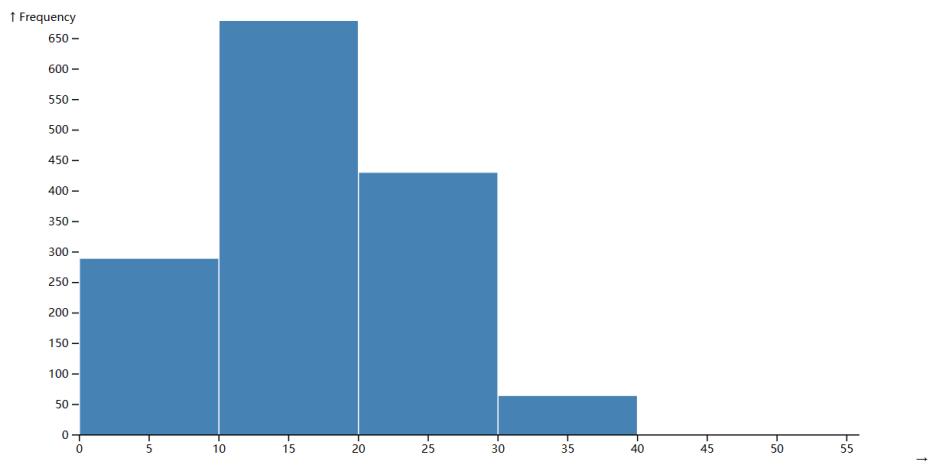
```

The output is:



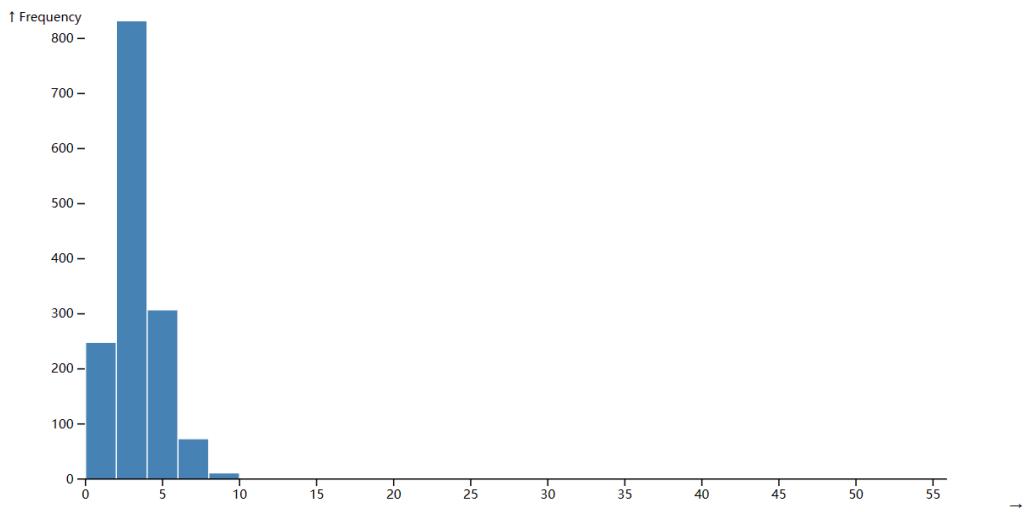
bins

Select Variable:



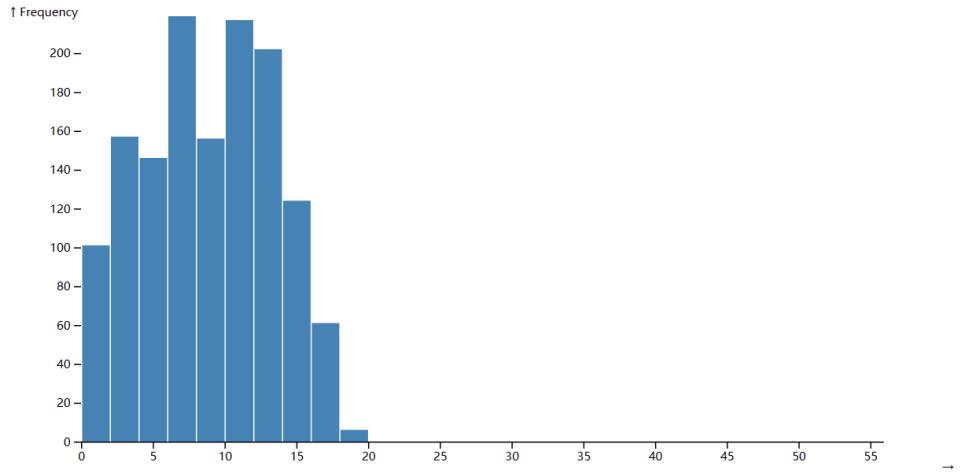
bins ▾

Select Variable: ▾



bins ▾

Select Variable: ▾



bins Select Variable:

Part II

Problem 1

2.1.1 The code is:

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Data Visualization</title>
8      <script src="https://d3js.org/d3.v5.min.js"></script>
9  </head>
10
11 <body>
12
13     <!-- open CSV FILE
14     <input type="file" accept=".csv" id="open"/> -->
15
16     <!-- GENERATE HTML TABLE -->
17     <table id="data-table"></table>
18
19     <script>
20
21         d3.csv("auto-mpg.csv").then(function(data) {
22             var table = "<table><thead><tr><th>mpg</th><th>cylinders</th><th>displacement</th><th>horsepower</th><th>weight</th><th>acceleration</th><th>model year</th><th>origin</th></thead><tbody>";
23             for (var i = 0; i < Math.min(5, data.length); i++) {
24                 table += "<tr><td>" + data[i].mpg + "</td><td>" + data[i].cylinders + "</td><td>" + data[i].displacement + "</td><td>" + data[i].horsepower + "</td><td>" + data[i].weight + "</td><td>" + data[i].acceleration + "</td><td>" + data[i].modelYear + "</td><td>" + data[i].origin + "</td></tr>";
25             }
26             table += "</tbody></table>";
27
28             document.getElementById("data-table").innerHTML = table;
29         });
30
31     </script>
32
33 </body>
34
35 </html>
```

```
>weight</th><th>acceleration</th><th>model-year</th></tr></thead><tbody>;  
ment + "</td><td>" + data[i].horsepower + "</td><td>" + data[i].weight + "</td><td>" + data[i].acceleration + "</td><td>" + data[i]["  
  
th><th>model-year</th></tr></thead><tbody>;  
orsepower + "</td><td>" + data[i].weight + "</td><td>" + data[i].acceleration + "</td><td>" + data[i][["model-year"]]+</td></tr>;
```

The output is:

mpg	cylinders	displacement	horsepower	weight	acceleration	model-year
18	8	307	130	3504	12	70
15	8	350	165	3693	11.5	70
18	8	318	150	3436	11	70
16	8	304	150	3433	12	70
17	8	302	140	3449	10.5	70

2.1.2 The code is:

```

1   <!DOCTYPE html>
2   <html lang="en">
3
4   <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Seattle Weather Data Visualization</title>
8     <script src="https://d3js.org/d3.v5.min.js"></script>
9   </head>
10
11  <body>
12
13    <!-- open CSV FILE
14    <input type="file" accept=".csv" id="open"/> -->
15
16    <!-- GENERATE HTML TABLE -->
17    <table id="data-table"></table>
18
19    <script>
20
21      d3.csv("auto-mpg.csv").then(function(data) {
22        var table = "<table><thead><tr><th>model</th><th>count</th></tr></thead><tbody>";
23        var carCounts = data.reduce(function(counts, item) {
24          counts[item["model-year"]] = (counts[item["model-year"]] || 0) + 1;
25          return counts;
26        }, {});
27        for (var model in carCounts) {
28          var count = carCounts[model];
29          table += "<tr><td>" + model + "</td><td>" + count + "</td></tr>";
30
31        table += "</tbody></table>";
32
33        document.getElementById("data-table").innerHTML = table;
34      });
35
36    </script>
37
38  </body>
39
40 </html>

```

The output is:

model	count
70	29
71	28
72	28
73	40
74	27
75	30
76	34
77	28
78	36
79	29
80	29
81	29
82	31

2.1.3 The code is:

```

1   <!DOCTYPE html>
2   <html lang="en">
3
4   <head>
5       <meta charset="UTF-8">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <title>Data Visualization</title>
8       <script src="https://d3js.org/d3.v5.min.js"></script>
9   </head>
10
11  <body>
12
13      <!-- open CSV FILE
14      <input type="file" accept=".csv" id="open"/> -->
15
16      <!-- GENERATE HTML TABLE -->
17      <table id="data-table"></table>
18
19      <script>
20
21          d3.csv("auto-mpg.csv").then(function(data) {
22              var table = "<table><thead><tr><th>model</th><th>count</th></tr></thead><tbody>";
23              var carCounts = data.reduce(function(counts, item) {
24                  var cylinders = parseInt(item["cylinders"]);
25                  counts[item["model-year"]] = (counts[item["model-year"]] || 0) + cylinders;
26                  return counts;
27              }, {});
28              for (var model in carCounts) [
29                  var count = carCounts[model];
30                  table += "<tr><td>" + model + "</td><td>" + count +"</td></tr>";}
31
32              table += "</tbody></table>";
33
34              document.getElementById("data-table").innerHTML = table;
35          });
36
37      </script>
38
39  </body>
40
41  </html>

```

The output is:

model count	
70	196
71	156
72	163
73	255
74	142
75	168
76	192
77	153
78	193
79	169
80	120
81	134
82	130

2.1.4 The code is:

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Data Visualization</title>
8      <script src="https://d3js.org/d3.v5.min.js"></script>
9  </head>
10
11 <body>
12
13     <!-- open CSV FILE
14     <input type="file" accept=".csv" id="open"/> -->
15
16     <!-- GENERATE HTML TABLE -->
17     <table id="data-table"></table>
18
19 <script>
20
21     d3.csv("auto-mpg.csv").then(function(data) {
22         var table = "<table><thead><tr><th>acceleration</th><th>count</th></tr></thead><tbody>";
23         var carCounts = data.reduce(function(counts, item) {
24
25             counts[item["acceleration"]] = (counts[item["acceleration"]] || 0) + 1;
26             return counts;
27         }, {});
28         for (var acceleration in carCounts) {
29             var count = carCounts[acceleration];
30             table += "<tr><td>" + acceleration + "</td><td>" + count + "</td></tr>";
31
32         table += "</tbody></table>";
33
34         document.getElementById("data-table").innerHTML = table;
35     });
36
37 </script>
38
39 </body>
40
41 </html>
```

Part of the output is:

acceleration	count
8	1
9	1
10	4
11	7
12	10
13	12
14	16
15	14
16	16
17	14
18	8
19	12
21	5
11.5	7
10.5	1
8.5	2
9.5	2
15.5	21
14.5	23
20.5	3
17.5	4
12.5	8
13.5	15
18.5	5
19.5	6
23.5	1
16.5	13
16.9	4
14.9	7
17.7	3
15.3	3
13.9	2
12.8	3

Problem 2

The code for this problem is:

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Data Visualization</title>
8      <script src="https://d3js.org/d3.v5.min.js"></script>
9  </head>
10 <body>
11     <svg width="800" height="800" id="pie-chart"></svg>
12     <svg width="800" height="400" id="line-chart"></svg>
13     <svg width="800" height="400" id="chart"></svg>
14
15     <script>
16         var widthPie = 800;
17         var heightPie = 800;
18         var radiusPie = Math.min(widthPie, heightPie) / 2;
19         var svgPie = d3.select('#pie-chart')
20             .attr('width', widthPie)
21             .attr('height', heightPie)
22             .append('g')
23             .attr('transform', 'translate(' + widthPie / 2 + ', ' + heightPie / 2 + ')');
24
25         d3.csv("auto-mpg.csv").then(function(data){
26             var carCounts = data.reduce(function(counts, item) {
27                 counts[item["model-year"]] = (counts[item["model-year"]] || 0) + 1;
28                 return counts;
29             }, {});
30             var totalObservations = Object.values(carCounts).reduce(function(sum, count) {
31                 return sum + count;
32             }, 0);
33             var carPercentages = {};
34             for (var model in carCounts) {
35                 var count = carCounts[model];
36                 var percentage = (count / totalObservations * 100).toFixed(2);
37                 carPercentages[model] = percentage;
38
39             var pie = d3.pie()
40                 .value(function(d) { return d.value; });
41
42             var pieData = pie(Object.entries(carPercentages).map(function([model, percentage]) {
43                 return { model: model, value: percentage };
44             }));
45
46             var arcs = svgPie.selectAll('arc')
47                 .data(pieData)
48                 .enter()
49                 .append('g')
50                 .attr('class', 'arc');
51
52             var outerRadius = widthPie / 4;
53             var innerRadius = 0;
54             var arc = d3.arc()
55                 .innerRadius(innerRadius)
56                 .outerRadius(outerRadius);
57
58             var color = d3.scaleOrdinal(d3.schemePaired);
59
60             arcs.append('path')
61                 .attr('d', arc)
62                 .attr('fill', function(d, i) { return color(i); });
63
64             arcs.append('text')
65                 .attr('transform', function(d) { return 'translate(' + arc.centroid(d) + ')'; })
66                 .attr('text-anchor', 'middle')
67                 .style('font-size', '10px')
68                 .text(function(d) { return d.data.model + ' ' + d.data.value + '%'; });
69
70         });
71     </script>
72
73
74
75     <script>
76         // Set the dimensions and margins of the graph
77         var marginLine = { top: 50, right: 50, bottom: 50, left: 100 },
78             widthLine = 800 - marginLine.left - marginLine.right,
79             heightLine = 400 - marginLine.top - marginLine.bottom;
80
81         // Parse the date
82         d3.csv("auto-mpg.csv").then(function(data){
83             var carCounts = data.reduce(function(counts, item) {
```

```

81 // Parse the data
82 d3.csv("auto-mpg.csv").then(function(data){
83     var carCounts = data.reduce(function(counts, item) {
84         var cylinders = parseInt(item["cylinders"]);
85         counts[item["model-year"]] = (counts[item["model-year"]] || 0) + cylinders;
86         return counts;
87     }, {});
88     var arrayOfEntries = Object.entries(carCounts);
89     // Set the ranges
90     var xScale = d3.scaleLinear()
91         .domain(d3.extent(arrayOfEntries, function(d) { return d[0]; }))
92         .range([0, widthLine]);
93
94     var yScale = d3.scaleLinear()
95         .domain([0, d3.max(arrayOfEntries, function(d) { return d[1]; })])
96         .range([heightLine, 0]);
97
98     // Define the line
99     var line = d3.line()
100        .x(function(d) { return xScale(d[0]); }) // Convert the x-value to a number
101        .y(function(d) { return yScale(d[1]); });
102
103    // Append the SVG object to the body of the page
104    var svgline = d3.select("#line-chart")
105        .attr("width", widthLine + marginLine.left + marginLine.right)
106        .attr("height", heightLine + marginLine.top + marginLine.bottom)
107        .append("g")
108        .attr("transform", "translate(" + marginLine.left + "," + marginLine.top + ")");
109
110    // Add the line
111    svgline.append("path")
112        .datum(arrayOfEntries)
113        .attr("class", "line")
114        .attr("id", line)
115        .attr("stroke", "blue")
116        .attr("fill", "transparent");
117
118    // Add the X Axis
119    svgline.append("g")
120        .attr("class", "x-axis")
121        .attr("transform", "translate(0," + heightLine + ")")
122        .call(d3.axisBottom(xScale));
123
124    // Add the Y Axis
125    svgline.append("g")
126        .attr("class", "y-axis")
127        .call(d3.axisLeft(yScale));
128
129    // Add X axis label
130    svgline.append("text")
131        .attr("class", "x-label")
132        .attr("text-anchor", "middle")
133        .attr("x", widthLine / 2)
134        .attr("y", heightLine + marginLine.bottom)
135        .text("Model Year");
136
137    // Add Y axis label
138    svgline.append("text")
139        .attr("class", "y-label")
140        .attr("text-anchor", "middle")
141        .attr("transform", "rotate(-90)")
142        .attr("x", -heightLine / 2)
143        .attr("y", -marginLine.left / 2)
144        .text("Cylinders");
145
146    // Add title
147    svgline.append("text")
148        .attr("class", "title")
149        .attr("text-anchor", "middle")
150        .attr("x", widthLine / 2)
151        .attr("y", -marginLine.top / 2)
152        .text("Cylinders vs Model Year");
153
154    // Add gridlines for y-axis
155    svgline.append("g")
156        .attr("class", "gridlines")
157        .call(d3.axisLeft(yScale).tickSize(-widthLine).tickFormat(""));
158
159    // Add gridlines for x-axis
160    svgline.append("g")
161        .attr("class", "gridlines")
162        .attr("transform", "translate(0," + heightLine + ")")
163        .call(d3.axisBottom(xScale).tickSize(-heightLine).tickFormat(""));
164
165});
</script>

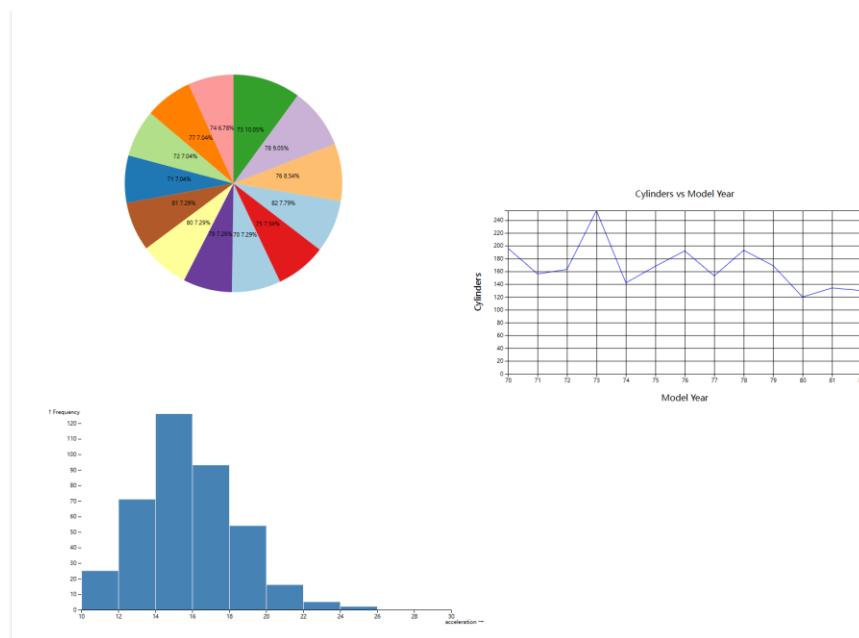
```

```

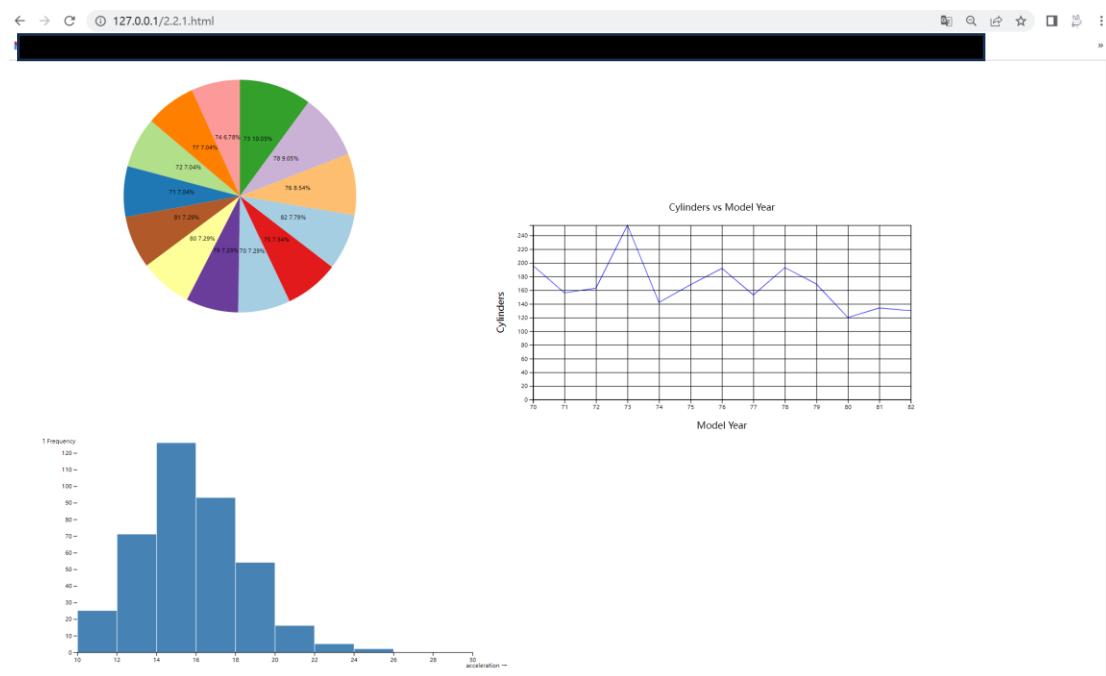
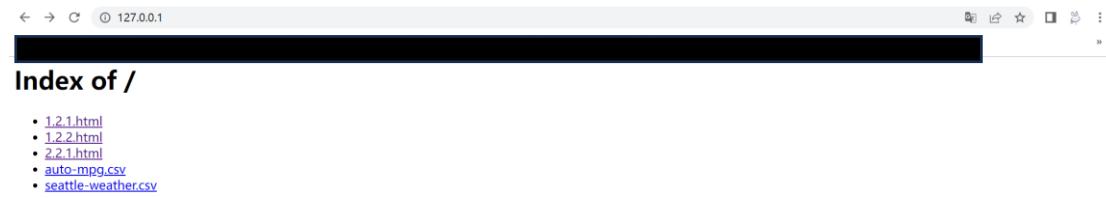
167 |     <script>
168 |         var widthBar = 800;
169 |         var heightBar = 400;
170 |         var marginBar = {
171 |             top: 10,
172 |             bottom: 30,
173 |             left: 60,
174 |             right: 60
175 |         };
176 |
177 |         var svgBar = d3.select('#chart')
178 |             .attr('width', widthBar + marginBar.left + marginBar.right)
179 |             .attr('height', heightBar + marginBar.top + marginBar.bottom)
180 |             .append('g')
181 |             .attr('transform', `translate(${marginBar.left}, ${marginBar.top})`);
182 |
183 |
184 |         d3.csv("auto-mpg.csv").then(function(data){
185 |             var histogram = d3.histogram()
186 |                 .value(function(d) { return d.acceleration; })
187 |                 // .domain(d3.extent(data, function(d) { return d.acceleration; }))
188 |                 .domain([10,30])
189 |                 .thresholds(10);
190 |
191 |             var bins = histogram(data)
192 |             console.log(bins);
193 |             //define x axis and y axis
194 |             const x = d3.scaleLinear()
195 |                 .domain([bins[0].x0, bins[bins.length - 1].x1])
196 |                 .range([marginBar.left, widthBar - marginBar.right]);
197 |             const y = d3.scaleLinear()
198 |                 .domain([0, d3.max(bins, (d) => d.length)])
199 |                 .range([heightBar - marginBar.bottom, marginBar.top]);
200 |
201 |
202 |             //add rectangulars to the figure
203 |             svgBar.append("g")
204 |                 .attr("fill", "steelblue")
205 |                 .selectAll()
206 |                 .data(bins)
207 |                 .join("rect")
208 |                 .attr("x", (d) => x(d.x0) + 1)
209 |                 .attr("width", (d) => x(d.x1) - x(d.x0) - 1)
210 |                 .attr("y", (d) => y(d.length))
211 |                 .attr("height", (d) => y(0) - y(d.length));
212 |
213 |
214 |             //add axis to the figure
215 |             svgBar.append("g")
216 |                 .attr("transform", `translate(0,${heightBar - marginBar.bottom})`)
217 |                 .call(d3.axisBottom(x).ticks(widthBar / 80).tickSizeOuter(0))
218 |                 .call((g) => g.append("text"))
219 |                     .attr("x", widthBar)
220 |                     .attr("y", marginBar.bottom - 4)
221 |                     .attr("fill", "currentColor")
222 |                     .attr("text-anchor", "end")
223 |                     .text("acceleration →");
224 |
225 |             svgBar.append("g")
226 |                 .attr("transform", `translate(${marginBar.left},0)`)
227 |                 .call(d3.axisLeft(y).ticks(heightBar / 40))
228 |                 .call((g) => g.select(".domain").remove())
229 |                 .call((g) => g.append("text"))
230 |                     .attr("x", -marginBar.left)
231 |                     .attr("y", 10)
232 |                     .attr("fill", "currentColor")
233 |                     .attr("text-anchor", "start")
234 |                     .text("↑ Frequency"));
235 |
236 |
237 |
238 |         </script>
239 |     </body>

```

The outputs are:



Problem 3



Part III

Problem 1

3.1.1 The code is:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Les Misérables Character Co-occurrence</title>
7      <script src="https://d3js.org/d3.v5.min.js"></script>
8      <style>
9          /* Add any custom styles for the graph container */
10         #graph-container {
11             width: 800px;
12             height: 600px;
13             margin: auto;
14         }
15     </style>
16 </head>
17 <body>
18
19 <svg width = "1600" height = "1200" id="graph"></svg>
20
21 <script>
22     d3.json("miserables.json").then(function(data) {
23
24         // Create svg
25         var svg = d3.select("#graph");
26
27         var width = +svg.attr("width");
28         var height = +svg.attr("height");
29
30         // Create simulation
31         var simulation = d3.forceSimulation(data.nodes)
32             .force("link", d3.forceLink(data.links).id(function(d) { return d.index; }))
33             .force("charge", d3.forceManyBody())
34             .force("center", d3.forceCenter(400, 300));
35
36         // Create links
37         var link = svg.append("g")
38             .selectAll("line")
39             .data(data.links)
40             .enter().append("line")
41             .attr("stroke", "#999")
42             .attr("stroke-width", 2);
43
44             .attr("stroke", "#999")
45             .attr("stroke-width", 2);
46
47         var color = d3.scaleOrdinal(d3.schemeCategory10);
48
49         // Create nodes
50         var node = svg.selectAll(".node")
51             .data(data.nodes)
52             .enter().append("circle")
53             .attr("class", "node")
54             .attr("r", 8)
55             .attr("fill", function(d) { return color(d.group); })
56             .call(d3.drag()
57                 .on("start", dragstarted)
58                 .on("drag", dragged)
59                 .on("end", dragended));
60
61
62         // Tick function to update positions of nodes and links
63         function ticked() {
64             link
65                 .attr("x1", function(d) { return d.source.x; })
66                 .attr("y1", function(d) { return d.source.y; })
67                 .attr("x2", function(d) { return d.target.x; })
68                 .attr("y2", function(d) { return d.target.y; });
69
70             node
71                 .attr("cx", function(d) { return d.x; })
72                 .attr("cy", function(d) { return d.y; });
73
74             label
75                 .attr("x", function(d) { return d.x; })
76                 .attr("y", function(d) { return d.y; });
77         }
78
79         // Start the simulation
80         simulation.on("tick", ticked);
81     
```

```

68     .attr("cx", function(d) { return d.x; });
69     .attr("cy", function(d) { return d.y; });
70
71     label
72     .attr("x", function(d) { return d.x; })
73     .attr("y", function(d) { return d.y; }));
74   }
75
76   // Start the simulation
77   simulation.on("tick", ticked);
78
79   // Drag functions
80   function dragstarted(d) {
81     if (!d3.event.active) simulation.alphaTarget(0.3).restart();
82     d.fx = d.x;
83     d.fy = d.y;
84   }
85
86   function dragged(d) {
87     d.fx = d3.event.x;
88     d.fy = d3.event.y;
89   }
90
91   function dragended(d) {
92     if (!d3.event.active) simulation.alphaTarget(0);
93     d.fx = null;
94     d.fy = null;
95   }
96 }
97 </script>
98
99 </body>
100 </html>

```

The output is:



3.1.2 The code is:

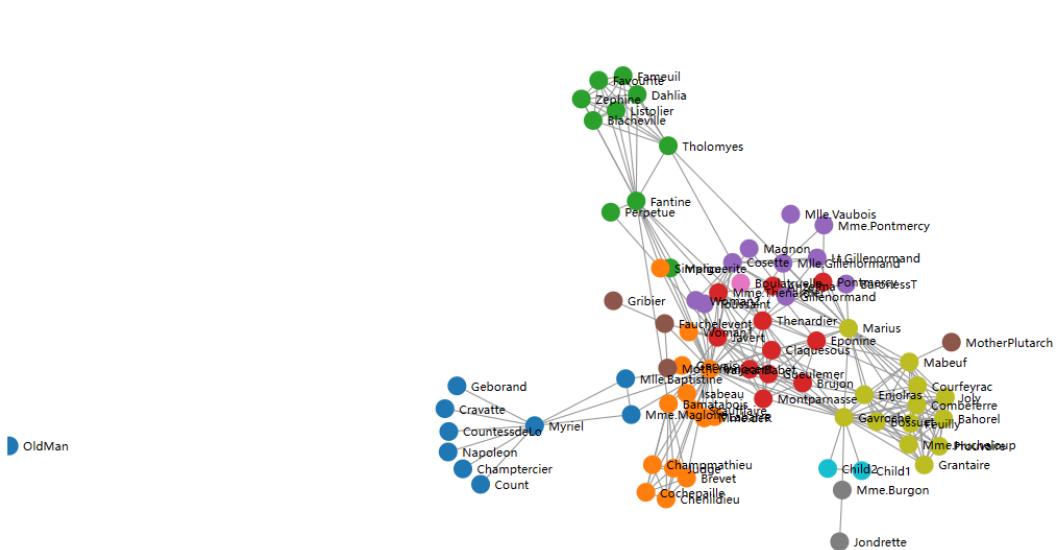
```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Les Misérables Character Co-occurrence</title>
7      <script src="https://d3js.org/d3.v5.min.js"></script>
8      <style>
9          /* Add any custom styles for the graph container */
10     #graph-container {
11         width: 800px;
12         height: 600px;
13         margin: auto;
14     }
15     </style>
16  </head>
17  <body>
18
19  <svg width = "1600" height = "1200" id="graph"></svg>
20
21  <script>
22      d3.json("miserables.json").then(function(data) {
23
24          // Create svg
25          var svg = d3.select("#graph");
26
27          var width = +svg.attr("width");
28          var height = +svg.attr("height");
29
30          // Create simulation
31          var simulation = d3.forceSimulation(data.nodes)
32              .force("link", d3.forceLink(data.links).id(function(d) { return d.index; }))
33              .force("charge", d3.forceManyBody())
34              .force("center", d3.forceCenter(400, 300));
35
36          // Create links
37          var link = svg.append("g")
38              .selectAll("line")
39              .data(data.links)
40              .enter().append("line")
41              .attr("stroke", "#999")
42              .attr("stroke-width", 2);
43
44              .attr("stroke-width", 2);
45
46          var color = d3.scaleOrdinal(d3.schemeCategory10);
47
48          // Create nodes
49          var node = svg.selectAll(".node")
50              .data(data.nodes)
51              .enter().append("circle")
52              .attr("class", "node")
53              .attr("r", 8)
54              .attr("fill", function(d) { return color(d.group); })
55              .call(d3.drag()
56                  .on("start", dragstarted)
57                  .on("drag", dragged)
58                  .on("end", dragended));
59
60          // Add labels to nodes
61          var label = svg.selectAll(".label")
62              .data(data.nodes)
63              .enter().append("text")
64              .attr("class", "label")
65              .text(function(d) { return d.name; })
66              .attr("font-size", "10px")
67              .attr("dx", 12)
68              .attr("dy", 4);
69
70          // Tick function to update positions of nodes and links
71          function ticked() {
72              link
73                  .attr("x1", function(d) { return d.source.x; })
74                  .attr("y1", function(d) { return d.source.y; })
75                  .attr("x2", function(d) { return d.target.x; })
76                  .attr("y2", function(d) { return d.target.y; });
77
78              node
79                  .attr("cx", function(d) { return d.x; })
80                  .attr("cy", function(d) { return d.y; });
81
82              label
83                  .attr("x", function(d) { return d.x; })
84                  .attr("y", function(d) { return d.y; });
85
86          }
87
88      
```

```

79     label
80       .attr("x", function(d) { return d.x; })
81       .attr("y", function(d) { return d.y; });
82   }
83
84   // Start the simulation
85   simulation.on("tick", ticked);
86
87   // Drag functions
88   function dragstarted(d) {
89     if (!d3.event.active) simulation.alphaTarget(0.3).restart();
90     d.fx = d.x;
91     d.fy = d.y;
92   }
93
94   function dragged(d) {
95     d.fx = d3.event.x;
96     d.fy = d3.event.y;
97   }
98
99   function dragended(d) {
100     if (!d3.event.active) simulation.alphaTarget(0);
101     d.fx = null;
102     d.fy = null;
103   }
104 });
105 </script>
106
107 </body>
108 </html>

```

The output is:



3.1.3 The code is:

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Les Misérables Character Co-occurrence</title>
7      <script src="https://d3js.org/d3.v5.min.js"></script>
8      <script src="fisheye.js"></script>
9
10     <style>
11         /* Add any custom styles for the graph container */
12     #graph-container {
13         width: 800px;
14         height: 600px;
15         margin: auto;
16     }
17     </style>
18 </head>
19 <body>
20
21     <svg width = "1600" height = "1200" id="graph"></svg>
22
23     <script>
24         d3.json("miserables.json").then(function(data) {
25
26             // Create svg
27             var svg = d3.select("#graph");
28
29             var width = +svg.attr("width");
30             var height = +svg.attr("height");
31
32
33             // Create simulation
34             var simulation = d3.forceSimulation(data.nodes)
35                 .force("link", d3.forceLink(data.links).id(function(d) { return d.index; }))
36                 .force("charge", d3.forceManyBody())
37                 .force("center", d3.forceCenter(400, 300));
38
39             // Create links
40             var link = svg.append("g")
41                 .selectAll("line")
42                 .data(data.links)
43
44             var link = svg.append("g")
45                 .selectAll("line")
46                 .data(data.links)
47
48             var color = d3.scaleOrdinal(d3.schemeCategory10);
49
50             // Create nodes
51             var node = svg.selectAll(".node")
52                 .data(data.nodes)
53                 .enter().append("circle")
54                 .attr("class", "node")
55                 .attr("r", 8)
56                 .attr("fill", function(d) { return color(d.group); })
57                 .call(d3.drag()
58                     .on("start", dragstarted)
59                     .on("drag", dragged)
60                     .on("end", dragended));
61
62             // Add labels to nodes
63             var label = svg.selectAll(".label")
64                 .data(data.nodes)
65                 .enter().append("text")
66                 .attr("class", "label")
67                 .text(function(d) { return d.name; })
68                 .attr("font-size", "10px")
69                 .attr("dx", 12)
70                 .attr("dy", 4);
71
72             //Update positions of nodes and links
73             function ticked() {
74                 link
75                     .attr("x1", function(d) { return d.source.x; })
76                     .attr("y1", function(d) { return d.source.y; })
77                     .attr("x2", function(d) { return d.target.x; })
78                     .attr("y2", function(d) { return d.target.y; });
79
80                 node
81                     .attr("cx", function(d) { return d.x; })
82                     .attr("cy", function(d) { return d.y; });

```

```

82     label
83         .attr("x", function(d) { return d.x; })
84         .attr("y", function(d) { return d.y; });
85     }
86
87     // Start the simulation
88     simulation.on("tick", ticked);
89
90     // Drag functions
91     function dragstarted(d) {
92         if (!d3.event.active) simulation.alphaTarget(0.3).restart();
93         d.fx = d.x;
94         d.fy = d.y;
95     }
96
97     function dragged(d) {
98         d.fx = d3.event.x;
99         d.fy = d3.event.y;
100    }
101
102    function dragended(d) {
103        if (!d3.event.active) simulation.alphaTarget(0);
104        d.fx = null;
105        d.fy = null;
106    }
107
108    var fisheye = d3.fisheye.circular()
109        .radius(200)
110        .distortion(2);
111    svg.on("mousemove", function() {
112        fisheye.focus(d3.mouse(this));
113
114        // Apply fisheye distortion to nodes
115        node.each(function(d) {
116            d.fisheye = fisheye(d);
117        });
118
119        // Update node positions
120        node.attr("cx", function(d) { return d.fisheye.x; })
121            .attr("cy", function(d) { return d.fisheye.y; });
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137

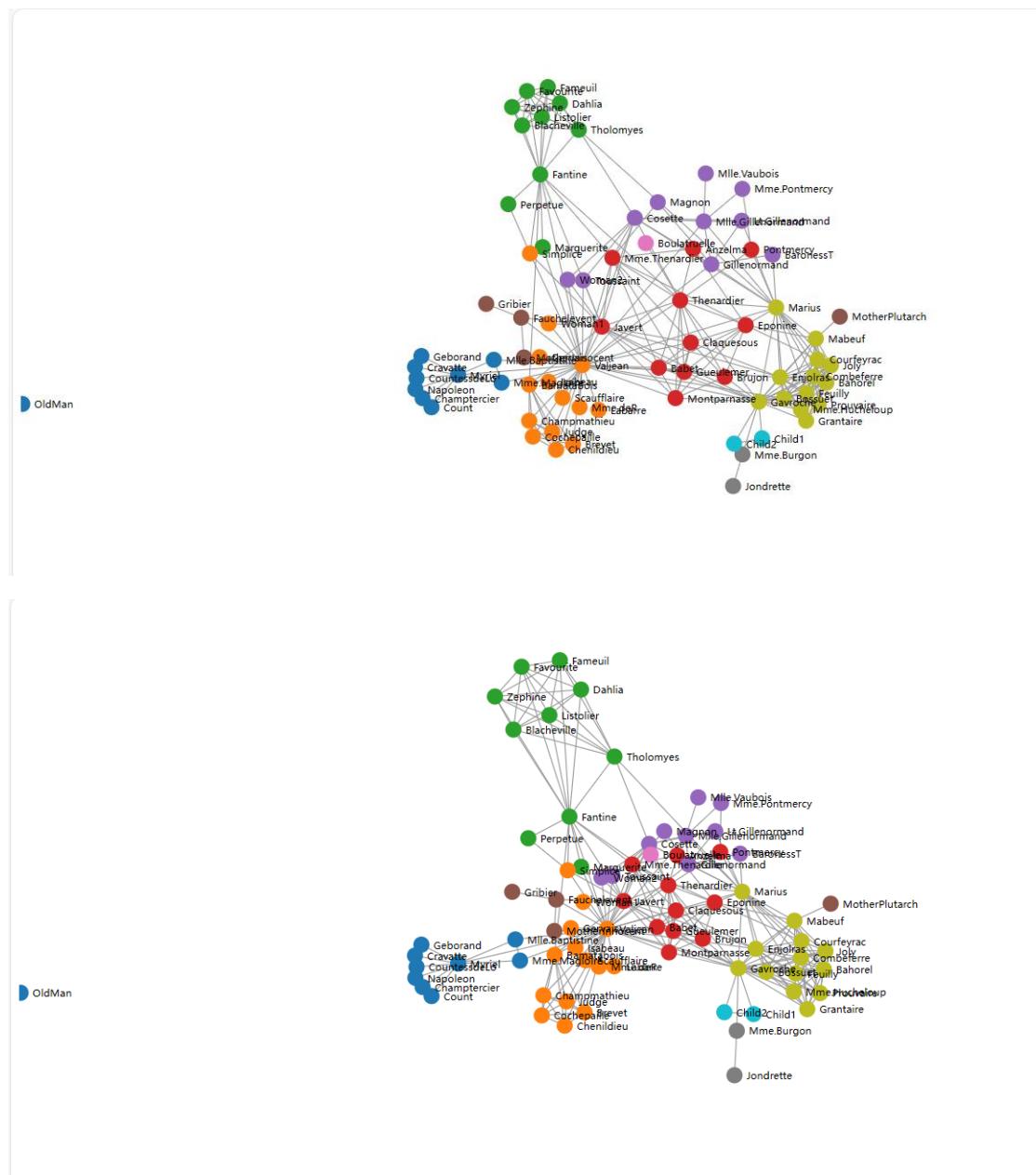
```

```

114    // Apply fisheye distortion to nodes
115    node.each(function(d) {
116        d.fisheye = fisheye(d);
117    });
118
119    // Update node positions
120    node.attr("cx", function(d) { return d.fisheye.x; })
121        .attr("cy", function(d) { return d.fisheye.y; });
122
123    // Update link positions
124    link.attr("x1", function(d) { return d.source.fisheye.x; })
125        .attr("y1", function(d) { return d.source.fisheye.y; })
126        .attr("x2", function(d) { return d.target.fisheye.x; })
127        .attr("y2", function(d) { return d.target.fisheye.y; });
128
129    // Update label positions
130    label.attr("x", function(d) { return d.fisheye.x; })
131        .attr("y", function(d) { return d.fisheye.y; });
132    });
133
134    </script>
135
136    </body>
137    </html>

```

The output is:



3.1.4 The code is:

```

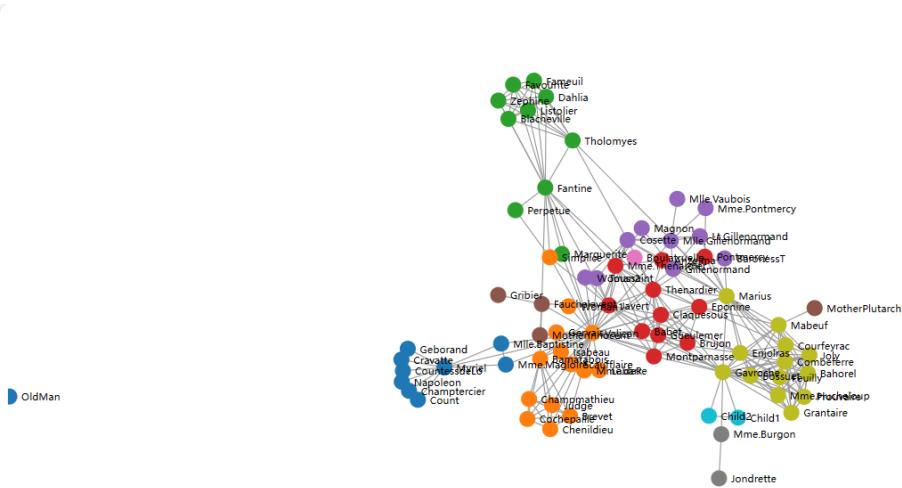
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Les Misérables Character Co-occurrence</title>
7       <script src="https://d3js.org/d3.v5.min.js"></script>
8       <script src="fisheye.js"></script>
9
10  </head>
11  <body>
12
13  <svg width = "1200" height = "600" id="graph"></svg>
14  <p>
15  |     Boulatruelle's group has the smallest number of people, with only one person.
16  |</p>
17  <p>
18  |     The group dyed orange has the largest number of people, with fourteen people.
19  |</p>
20  <p>
21  |     MotherPlutarch, Gribier, and Jondrette are all related to only one person.
22  |</p>
23  <p>
24  |     Oldman is not connected with anyone.
25  |</p>
26  <script>
27  d3.json("miserables.json").then(function(data) {
28
29     // Create svg
30     var svg = d3.select("#graph");
31
32     var width = +svg.attr("width");
33     var height = +svg.attr("height");
34
35
36     // Create simulation
37     var simulation = d3.forceSimulation(data.nodes)
38         .force("link", d3.forceLink(data.links).id(function(d) { return d.index; }))
39         .force("charge", d3.forceManyBody())
40         .force("center", d3.forceCenter(600, 300));
41
42     // Create links
43     var link = svg.append("g")
44         .selectAll("line")
45         .data(data.links)
46         .enter().append("line")
47         .attr("stroke", "#999");
48
49     var color = d3.scaleOrdinal(d3.schemeCategory10);
50
51     // Create nodes
52     var node = svg.selectAll(".node")
53         .data(data.nodes)
54         .enter().append("circle")
55         .attr("class", "node")
56         .attr("r", 8)
57         .attr("fill", function(d) { return color(d.group); })
58         .call(d3.drag()
59             .on("start", dragstarted)
60             .on("drag", dragged)
61             .on("end", dragended));
62
63     // Add labels to nodes
64     var label = svg.selectAll(".label")
65         .data(data.nodes)
66         .enter().append("text")
67         .attr("class", "label")
68         .text(function(d) { return d.name; })
69         .attr("font-size", "10px")
70         .attr("dx", 12)
71         .attr("dy", 4);
72
73     //update positions of nodes and links
74     function ticked() {
75         link
76             .attr("x1", function(d) { return d.source.x; })
77             .attr("y1", function(d) { return d.source.y; })
78             .attr("x2", function(d) { return d.target.x; })
79             .attr("y2", function(d) { return d.target.y; });
80

```

```

81     node
82       .attr("cx", function(d) { return d.x; })
83       .attr("cy", function(d) { return d.y; }));
84
85     label
86       .attr("x", function(d) { return d.x; })
87       .attr("y", function(d) { return d.y; }));
88   }
89
90   // Start the simulation
91   simulation.on("tick", ticked);
92
93   // Drag functions
94   function dragstarted(d) {
95     if (!d3.event.active) simulation.alphaTarget(0.3).restart();
96     d.fx = d.x;
97     d.fy = d.y;
98   }
99
100  function dragged(d) {
101    d.fx = d3.event.x;
102    d.fy = d3.event.y;
103  }
104
105  function dragended(d) {
106    if (!d3.event.active) simulation.alphaTarget(0);
107    d.fx = null;
108    d.fy = null;
109  }
110
111  var fisheye = d3.fisheye.circular()
112    .radius(200)
113    .distortion(2);
114  svg.on("mousemove", function() {
115    fisheye.focus(d3.mouse(this));
116
117    // Apply fisheye distortion to nodes
118    node.each(function(d) {
119      d.fisheye = fisheye(d);
120    });
121
122    // Update node positions
123    node.attr("cx", function(d) { return d.fisheye.x; })
124      .attr("cy", function(d) { return d.fisheye.y; });
125
126    // Update link positions
127    link.attr("x1", function(d) { return d.source.fisheye.x; })
128      .attr("y1", function(d) { return d.source.fisheye.y; })
129      .attr("x2", function(d) { return d.target.fisheye.x; })
130      .attr("y2", function(d) { return d.target.fisheye.y; });
131
132    // Update label positions
133    label.attr("x", function(d) { return d.fisheye.x; })
134      .attr("y", function(d) { return d.fisheye.y; });
135  });
136}
137</script>
138
139</body>
140</html>
```

The output is:



Boulatruele's group has the smallest number of people, with only one person.

The group dyed orange has the largest number of people, with fourteen people.

MotherPlutarch, Gribier, and Jondrette are all related to only one person.

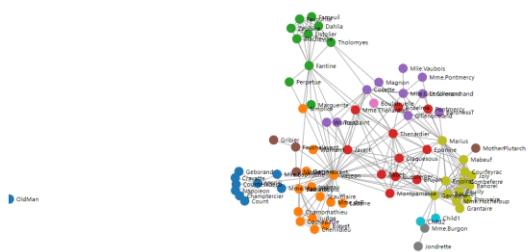
Oldman is not connected with anyone.

Problem 2



Index of /

- [1.2.1.html](#)
 - [1.2.2.html](#)
 - [2.2.1.html](#)
 - [3.1.3.html](#)
 - [auto-mpg.csv](#)
 - [fisheye.js](#)
 - [miserables.json](#)
 - [seattle-weather.csv](#)



Boulatruelle's group has the smallest number of people, with only one person.

The group dyed orange has the largest number of people, with fourteen people.

MotherPlutarch, Gribier, and Jondrette are all related to only one person.

Oldman is not connected with anyone.