# 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:

```html
<!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

```html
<!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_2</title>
  <script src="https://d3js.org/d3.v5.min.js"></script>
</head>

<body>
<p>D3 Tutorials </p>

<script>
    var myData = ['big', 'data', 'assignment', 2, 'submission'];

        var p = d3.select("body")
                  .selectAll("p")
                  .data(myData)
                  .text(function (d,i) {
                      return d;
                  });
</script>
</body>

</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.

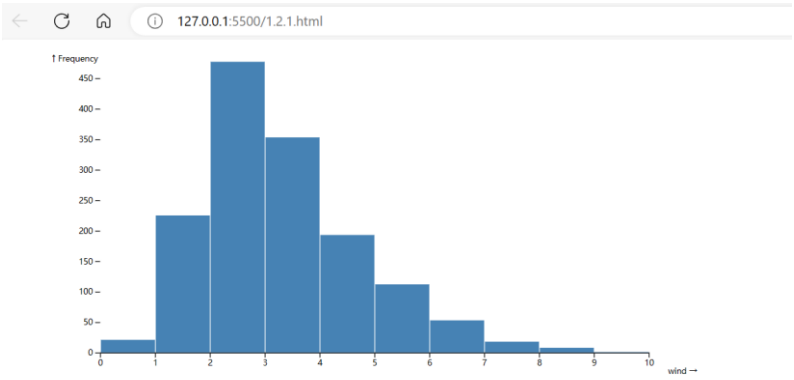
**Problem 2**

1.2.1 The code is:

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

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

<body>
    <div id="chart"></div>
    <script>
        const width = 800
        const height = 400
        const margin = {
            top: 10,
            bottom: 30,
            left: 60,
            right: 60
        }

        var svg = d3.select('#chart')
        .append('svg')
        .attr('width', width + margin.left + margin.right)
        .attr('height', height + margin.top + margin.bottom)
        .append('g')
        .attr('transform', `translate(${ margin.left }, ${ margin.top })`)

        d3.csv("seattle-weather.csv").then(function(data){
            var histogram = d3.histogram()
                .value(function(d) {return d.wind})
                .domain([0,10])
                .thresholds(10);

            var bins = histogram(data)
            console.log(bins);
        //define x axis and y axis
        const x = d3.scaleLinear()
                    .domain([bins[0].x0, bins[bins.length - 1].x1])
                    .range([margin.left, width - margin.right]);
        const y = d3.scaleLinear()
```

```
    const y = d3.scaleLinear()
                .domain([0, d3.max(bins, (d) => d.length)])
                .range([height - margin.bottom, margin.top]);


    //add rectangulars to the figure
    svg.append("g")
        .attr("fill", "steelblue")
        .selectAll()
        .data(bins)
        .join("rect")
        .attr("x", (d) => x(d.x0) + 1)
        .attr("width", (d) => x(d.x1) - x(d.x0) - 1)
        .attr("y", (d) => y(d.length))
        .attr("height", (d) => y(0) - y(d.length));

    //add axis to the figure
    svg.append("g")
        .attr("transform", `translate(0,${height - margin.bottom})`)
        .call(d3.axisBottom(x).ticks(width / 80).tickSizeOuter(0))
        .call((g) => g.append("text")
            .attr("x", width)
            .attr("y", margin.bottom - 4)
            .attr("fill", "currentColor")
            .attr("text-anchor", "end")
            .text("wind →"));

    svg.append("g")
        .attr("transform", `translate(${margin.left},0)`)
        .call(d3.axisLeft(y).ticks(height / 40))
        .call((g) => g.select(".domain").remove())
        .call((g) => g.append("text")
            .attr("x", -margin.left)
            .attr("y", 10)
            .attr("fill", "currentColor")
            .attr("text-anchor", "start")
            .text("↑ Frequency"));
    });
```

The output is:

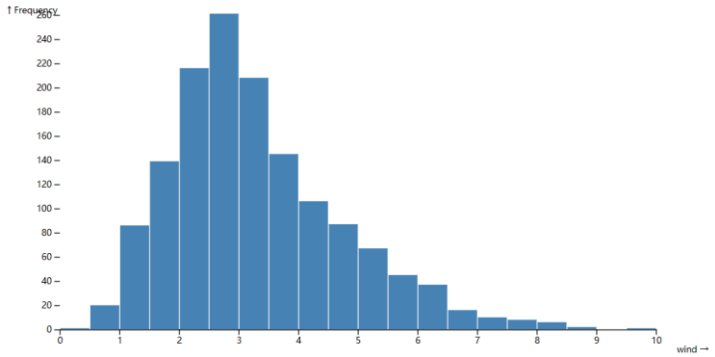



1.2.2 The code is:

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

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

<body>
    <div id="chart"></div>
    <script>
        const width = 800
        const height = 400
        const margin = {
            top: 10,
            bottom: 30,
            left: 60,
            right: 60
        }

        var svg = d3.select('#chart')
        .append('svg')
        .attr('width', width + margin.left + margin.right)
        .attr('height', height + margin.top + margin.bottom)
        .append('g')
        .attr('transform', `translate(${ margin.left }, ${ margin.top })`)

        d3.csv("seattle-weather.csv").then(function(data){
            var histogram = d3.histogram()
                .value(function(d) {return d.wind})
                .domain([0,10])
                .thresholds(20);

            var bins = histogram(data)
            console.log(bins);
        //define x axis and y axis
        const x = d3.scaleLinear()
                    .domain([bins[0].x0, bins[bins.length - 1].x1])
                    .range([margin.left, width - margin.right]);
        const y = d3.scaleLinear()
```

```
        const y = d3.scaleLinear()
                    .domain([0, d3.max(bins, (d) => d.length)])
                    .range([height - margin.bottom, margin.top]);


        //add rectangulars to the figure
        svg.append("g")
            .attr("fill", "steelblue")
            .selectAll()
            .data(bins)
            .join("rect")
            .attr("x", (d) => x(d.x0) + 1)
            .attr("width", (d) => x(d.x1) - x(d.x0) - 1)
            .attr("y", (d) => y(d.length))
            .attr("height", (d) => y(0) - y(d.length));

        //add axis to the figure
        svg.append("g")
            .attr("transform", `translate(0,${height - margin.bottom})`)
            .call(d3.axisBottom(x).ticks(width / 80).tickSizeOuter(0))
            .call((g) => g.append("text")
                .attr("x", width)
                .attr("y", margin.bottom - 4)
                .attr("fill", "currentColor")
                .attr("text-anchor", "end")
                .text("wind →"));

        svg.append("g")
            .attr("transform", `translate(${margin.left},0)`)
            .call(d3.axisLeft(y).ticks(height / 40))
            .call((g) => g.select(".domain").remove())
            .call((g) => g.append("text")
                .attr("x", -margin.left)
                .attr("y", 10)
                .attr("fill", "currentColor")
                .attr("text-anchor", "start")
                .text("↑ Frequency"));
        });
```

The output is:

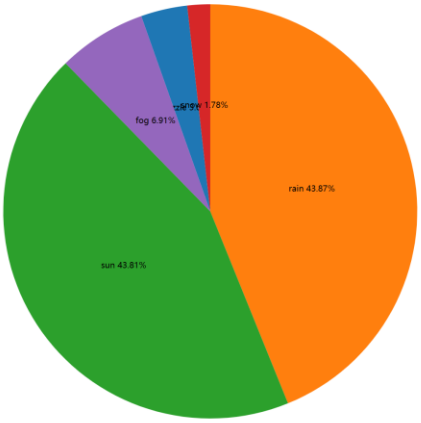



### 1.2.3 The code is:

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

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

<body>
    <svg width="400" height="400" id="pie-chart"></svg>
    <script>
        var width = 1600;
        var height = 1600;
        var radius = Math.min(width, height) / 2;
        var svg = d3.select('#pie-chart')
                    .attr('width', width)
                    .attr('height', height)
                    .append('g')
                    .attr('transform', 'translate(' + width / 2 + ',' + height / 2 + ')');

        d3.csv("seattle-weather.csv").then(function(data){
            var weatherCounts = data.reduce(function(counts, data) {
                                    counts[data.weather] = (counts[data.weather] || 0) + 1;
                                    return counts;}, {});
            var totalObservations = Object.values(weatherCounts).reduce(function(sum, count) {
                                        return sum + count;
                                        }, 0);
            var weatherPercentages = {};
            for (var weather in weatherCounts) {
                var count = weatherCounts[weather];
                var percentage = (count / totalObservations * 100).toFixed(2);
                weatherPercentages[weather] = percentage;}


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

            var pieData = pie(Object.entries(weatherPercentages).map(function([weather, percentage]) {
                        return { weather: weather, value: percentage };
                    }));
```

```
            var arcs = svg.selectAll('arc')
            .data(pieData)
            .enter()
            .append('g')
            .attr('class', 'arc');

            var outerRadius = width / 4;
            var innerRadius = 0;
            var arc = d3.arc()
            .innerRadius(innerRadius)
            .outerRadius(outerRadius);

            var color = d3.scaleOrdinal(d3.schemeCategory10);

            arcs.append('path')
            .attr('d', arc)
            .attr('fill', function(d, i) { return color(i); });

            arcs.append('text')
            .attr('transform', function(d) { return 'translate(' + arc.centroid(d) + ')'; })
            .attr('text-anchor', 'middle')
            .text(function(d) { return d.data.weather + ' ' + d.data.value + '%'; });


        });
    </script>
</body>
```

The output is:

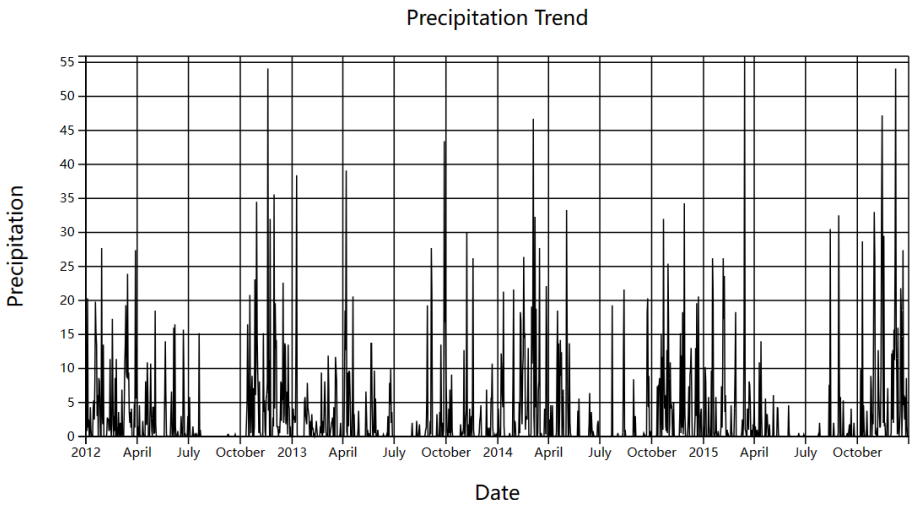



1.2.4 The code is:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Line Graph</title>
  <script src="https://d3js.org/d3.v7.min.js"></script>
  <style>
    /* Add some basic styles if needed */
  </style>
</head>
<body>
  <svg width="800" height="400" id="line-chart"></svg>
  <script>
        // Set the dimensions and margins of the graph
    var margin = { top: 50, right: 50, bottom: 50, left: 100 },
    width = 800 - margin.left - margin.right,
    height = 400 - margin.top - margin.bottom;

    // Parse the date
    var parseDate = d3.timeParse("%Y-%m-%d");
    d3.csv("seattle-weather.csv").then(function(data){
        // Format the data
        data.forEach(function(d) {
        d.date = parseDate(d.date);
        d.precipitation = +d.precipitation;
        });

        // Set the ranges
        var xScale = d3.scaleTime().domain(d3.extent(data, function(d) { return d.date; })).range([0, width]);
        var yScale = d3.scaleLinear().domain([0, d3.max(data, function(d) { return d.precipitation; })]).range([height, 0]);

        // Define the line
        var line = d3.line()
        .x(function(d) { return xScale(d.date); })
        .y(function(d) { return yScale(d.precipitation); });

        // Append the SVG object to the body of the page
        var svg = d3.select("#line-chart")
        .attr("width", width + margin.left + margin.right)
        .attr("height", height + margin.top + margin.bottom)
        .append("g")
        .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

        // Add the line
        svg.append("path")
        .data([data])
        .attr("class", "line")
        .attr("d", line)
        .attr("stroke", "black")
        .attr("fill", "transparent");

        // Add the X Axis
        svg.append("g")
        .attr("class", "x-axis")
        .attr("transform", "translate(0," + height + ")")
        .call(d3.axisBottom(xScale));

        // Add the Y Axis
        svg.append("g")
        .attr("class", "y-axis")
        .call(d3.axisLeft(yScale));

        // Add X axis label
        svg.append("text")
        .attr("class", "x-label")
        .attr("text-anchor", "middle")
        .attr("x", width / 2)
        .attr("y", height + margin.bottom)
        .text("Date");

        // Add Y axis label
        svg.append("text")
        .attr("class", "y-label")
        .attr("text-anchor", "middle")
        .attr("transform", "rotate(-90)")
        .attr("x", -height / 2)
        .attr("y", -margin.left / 2)
        .text("Precipitation");

        // Add title
        svg.append("text")
        .attr("class", "title")
        .attr("text-anchor", "middle")
```

```
        .attr("text-anchor", "middle")
        .attr("x", width / 2)
        .attr("y", -margin.top / 2)
        .text("Precipitation Trend");

        // Add gridlines for y-axis
        svg.append("g")
        .attr("class", "gridlines")
        .call(d3.axisLeft(yScale).tickSize(-width).tickFormat(""));

        // Add gridlines for x-axis
        svg.append("g")
        .attr("class", "gridlines")
        .attr("transform", "translate(0," + height + ")")
        .call(d3.axisBottom(xScale).tickSize(-height).tickFormat(""));
    });
  </script>
</body>
</html>
```

The output is:




### 1.2.5 The code is:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>paragraphs</title>
  <script src="https://d3js.org/d3.v7.min.js"></script>

</head>


<body>
    <p> From 1.2.1, we can find that there has the greatest posibility that the value of wind is in the domain 2-3.</p>
    <p> From 1.2.2, there has the greatest posibility that the value of wind is in the domain 2.5-3.</p>
    <p> 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.</p>
    <p> 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.</p>
</body>
</html>
```

The output is:

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

From 1.2.2, there has the greatest posibility 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:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>paragraphs</title>
  <script src="https://d3js.org/d3.v7.min.js"></script>
  <style>
    p {
        font-size: 16px;
        color: rgba(34, 92, 133, 0.828);
        margin-bottom: 10px;
    }

    .section-number {
        font-weight: bold;
    }
</style>
</head>


<body>
    <p> From <span class="section-number">1.2.1</span>, we can find that there has the greatest posibility that
        the value of wind is in the domain 2-3.</p>
    <p> From <span class="section-number">1.2.2</span>, there has the greatest posibility that the value of wind is
        in the domain 2.5-3.</p>
    <p> From <span class="section-number">1.2.3</span>, 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.</p>
    <p> From <span class="section-number">1.2.4</span>, 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.</p>
</body>
</html>
```

The output is:

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

From **1.2.2**, there has the greatest posibility 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.

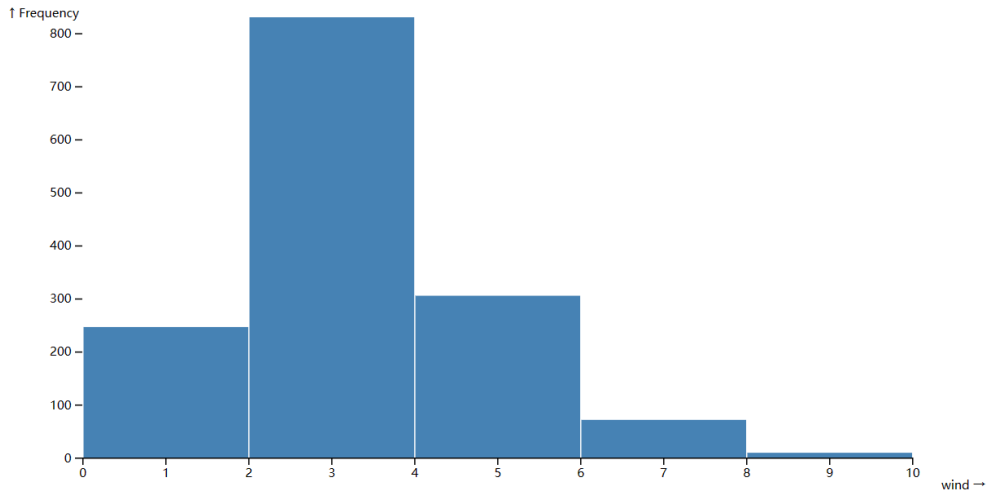
## Problem 3

### 1.3.1 The code is:

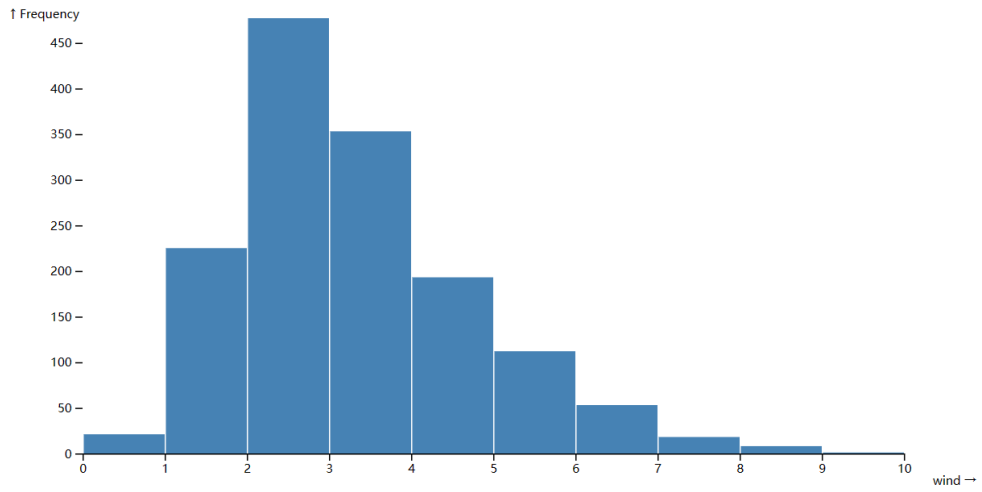
```html
<!DOCTYPE html>
<html lang="en">

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

<body>
    <div id="chart"></div>
    <p>
        <label># bins        </label>
        <input type="number" min="5" max="25" step="5" value="20" id="nBin">
    </p>
    <script>
        const width = 800
        const height = 400
        const margin = {
            top: 10,
            bottom: 30,
            left: 60,
            right: 60
        }

        var svg = d3.select('#chart')
        .append('svg')
        .attr('width', width + margin.left + margin.right)
        .attr('height', height + margin.top + margin.bottom)
        .append('g')
        .attr('transform', `translate(${ margin.left }, ${ margin.top })`)

        const x = d3.scaleLinear()
            .range([margin.left, width - margin.right]);

        const y = d3.scaleLinear()
            .range([height - margin.bottom, margin.top]);


        d3.csv("seattle-weather.csv").then(function(data){
            function update(nBin){
                var histogram = d3.histogram()
                .value(function(d) {return d.wind})
                .domain([0,10])
                .thresholds(x.ticks(nBin));
                var bins = histogram(data);

        //define x axis and y axis
                x.domain([bins[0].x0, bins[bins.length - 1].x1]);
                y.domain([0, d3.max(bins, (d) => d.length)]);

                svg.selectAll(".x-axis").remove();
                svg.selectAll(".y-axis").remove();
        //add axis to the figure
        svg.append("g")
            .attr("class", "x-axis")
            .attr("transform", `translate(0,${height - margin.bottom})`)
            .call(d3.axisBottom(x).ticks(width / 80).tickSizeOuter(0))
            .call((g) => g.append("text")
                .attr("x", width)
                .attr("y", margin.bottom - 4)
                .attr("fill", "currentColor")
                .attr("text-anchor", "end")
                .text("wind →"));

        svg.append("g")
            .attr("class", "y-axis")
            .attr("transform", `translate(${margin.left},0)`)
            .call(d3.axisLeft(y).ticks(height / 40))
            .call((g) => g.select(".domain").remove())
            .call((g) => g.append("text")
                .attr("x", -margin.left)
                .attr("y", 10)
                .attr("fill", "currentColor")
                .attr("text-anchor", "start")
                .text("↑ Frequency"));

```

```
        var u = svg.selectAll("rect")
          .data(bins);

        u.enter()
        .append("rect")
        .merge(u)
        .transition()
        .duration(1000)
        .attr("x", (d) => x(d.x0) + 1)
        .attr("width", (d) => x(d.x1) - x(d.x0) - 1)
        .attr("y", (d) => y(d.length))
        .attr("height", (d) => y(0) - y(d.length))
        .attr("fill", "steelblue");

        u.exit().remove();
        }

        update(20)

        d3.select("#nBin").on("input", function(){
            update(+this.value)
        });
    });

    </script>
</body>

</html>
```

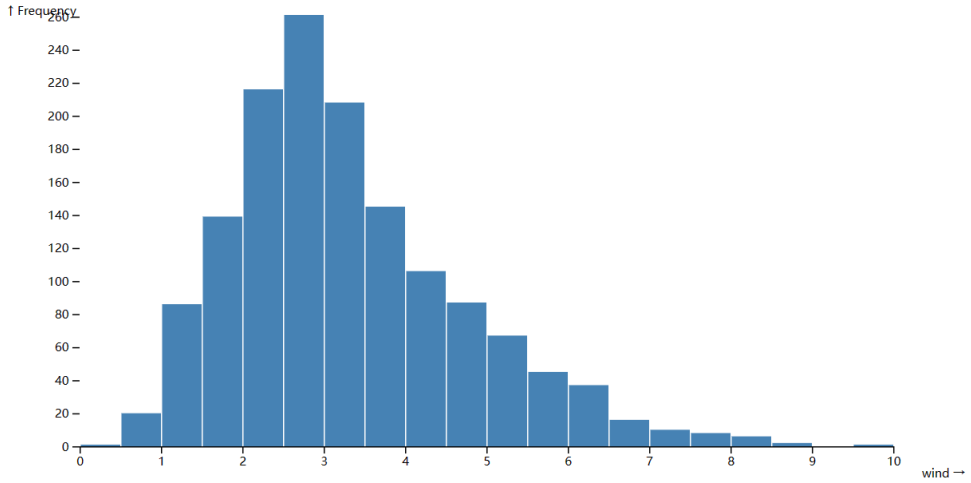
The output is:




# bins 5




# bins 10




# bins 15

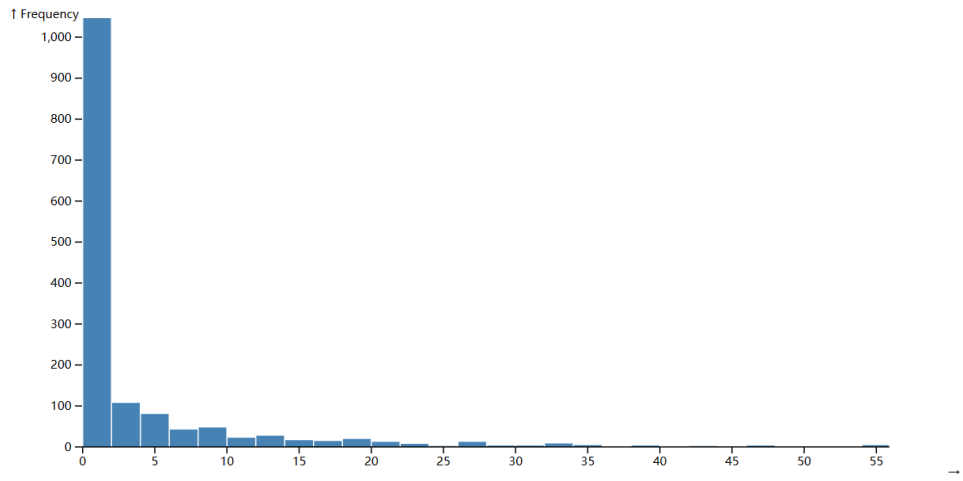
### 1.3.2 The code is:

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

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

<body>
    <div id="chart"></div>
    <p>
        <label># bins        </label>
        <input type="number" min="5" max="25" step="5" value="20" id="nBin">
    </p>
    <p>
        <label>Select Variable:</label>
        <select id="variable">
            <option value="precipitation">Precipitation</option>
            <option value="temp_max">Max Temperature</option>
            <option value="temp_min">Min Temperature</option>
            <option value="wind">Wind</option>
        </select>
    </p>
    <script>
        const width = 800
        const height = 400
        const margin = {
            top: 10,
            bottom: 30,
            left: 60,
            right: 60
        }

        var svg = d3.select('#chart')
        .append('svg')
        .attr('width', width + margin.left + margin.right)
        .attr('height', height + margin.top + margin.bottom)
        .append('g')
        .attr('transform', `translate(${ margin.left }, ${ margin.top })`)
```

```javascript
        const x = d3.scaleLinear()
            .range([margin.left, width - margin.right]);

        const y = d3.scaleLinear()
            .range([height - margin.bottom, margin.top]);


        d3.csv("seattle-weather.csv").then(function(data){
            var selectedVariable = "precipitation";
            var minDataValue = d3.min(data, function(d) { return parseFloat(d[selectedVariable]); });
            var maxDataValue = d3.max(data, function(d) { return parseFloat(d[selectedVariable]); });

            function update(nBin){
                var selectedData = data.map(function(d) {
            return parseFloat(d[selectedVariable]);
        });
                var histogram = d3.histogram()
                .value(function(d) {return d})
                .domain([minDataValue,maxDataValue])
                .thresholds(x.ticks(nBin));
                var bins = histogram(selectedData);

        //define x axis and y axis
                x.domain([bins[0].x0, bins[bins.length - 1].x1]);
                y.domain([0, d3.max(bins, (d) => d.length)]);

                svg.selectAll(".x-axis").remove();
                svg.selectAll(".y-axis").remove();
        //add axis to the figure
        svg.append("g")
            .attr("class", "x-axis")
            .attr("transform", `translate(0,${height - margin.bottom})`)
            .call(d3.axisBottom(x).ticks(width / 80).tickSizeOuter(0))
            .call((g) => g.append("text")
                .attr("x", width)
                .attr("y", margin.bottom - 4)
                .attr("fill", "currentColor")
                .attr("text-anchor", "end")
                .text("→"));
```

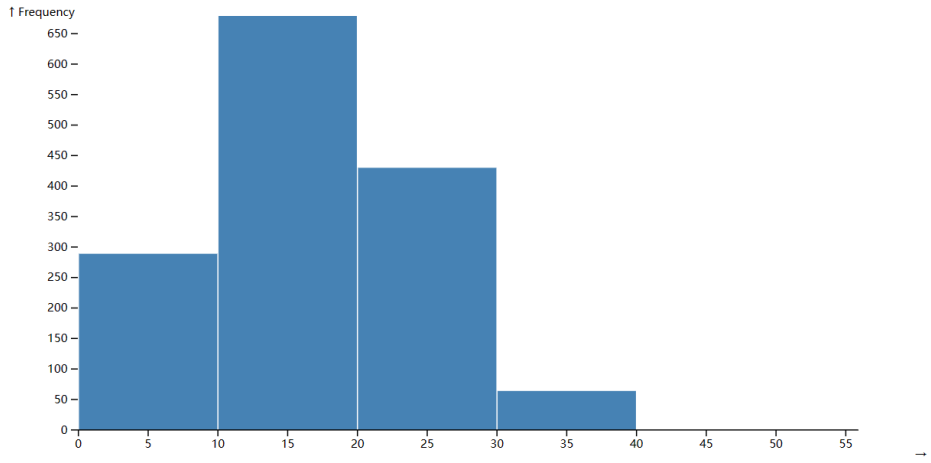
```
        svg.append("g")
            .attr("class", "y-axis")
            .attr("transform", `translate(${margin.left},0)`)
            .call(d3.axisLeft(y).ticks(height / 40))
            .call((g) => g.select(".domain").remove())
            .call((g) => g.append("text")
                .attr("x", -margin.left)
                .attr("y", 10)
                .attr("fill", "currentColor")
                .attr("text-anchor", "start")
                .text("↑ Frequency"));

        var u = svg.selectAll("rect")
          .data(bins);

        u.enter()
        .append("rect")
        .merge(u)
        .transition()
        .duration(1000)
        .attr("x", (d) => x(d.x0) + 1)
        .attr("width", (d) => x(d.x1) - x(d.x0) - 1)
        .attr("y", (d) => y(d.length))
        .attr("height", (d) => y(0) - y(d.length))
        .attr("fill", "steelblue");

        u.exit().remove();
        }

        update(20)

        d3.select("#nBin").on("input", function(){
            update(+this.value)
        });

        d3.select("#variable").on("change", function() {
        selectedVariable = this.value;
        update(+document.getElementById("nBin").value); // get bins
    });

    // get initial value
    update(+document.getElementById("nBin").value);
```

The output is:



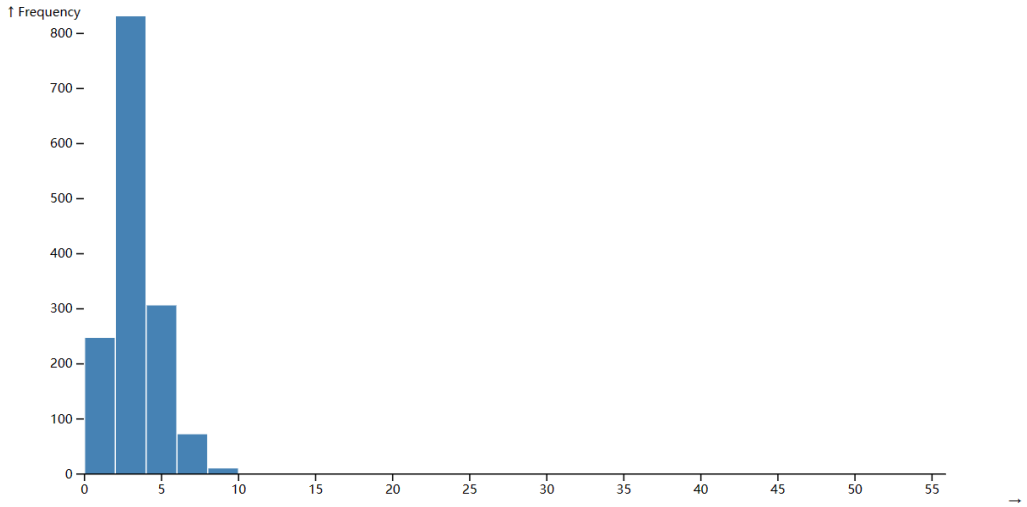

# bins 20

Select Variable: Precipitation



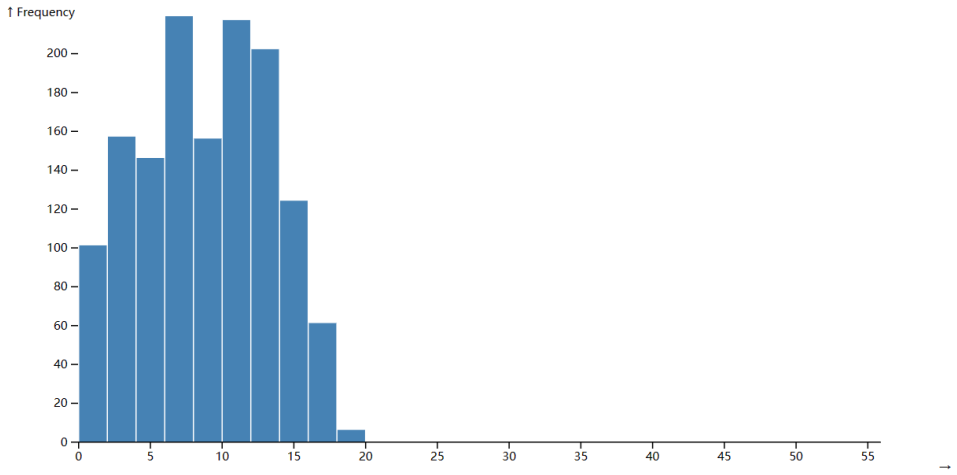

# bins 5

Select Variable: Max Temperature




# bins 20

Select Variable: Wind




# bins 25

Select Variable: Min Temperature

## Part II

### Problem 1

2.1.1 The code is:

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

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

<body>

    <!-- open CSV FILE
    <input type="file" accept=".csv" id="open"/> -->

    <!-- GENERATE HTML TABLE -->
    <table id="data-table"></table>

    <script>

        d3.csv("auto-mpg.csv").then(function(data) {
            var table = "<table><thead><tr><th>mpg</th><th>cylinders</th><th>displacement</th><th>horsepower</th><th>weight</th><th>a
            for (var i = 0; i < Math.min(5, data.length); i++) {
                table += "<tr><td>" + data[i].mpg + "</td><td>" + data[i].cylinders + "</td><td>" + data[i].displacement + "</td><td>
            }
            table += "</tbody></table>";

            document.getElementById("data-table").innerHTML = table;
        });

    </script>

</body>

</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:

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

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

<body>

    <!-- open CSV FILE
    <input type="file" accept=".csv" id="open"/> -->

    <!-- GENERATE HTML TABLE -->
    <table id="data-table"></table>

    <script>

        d3.csv("auto-mpg.csv").then(function(data) {
            var table = "<table><thead><tr><th>model</th><th>count</th></tr></thead><tbody>";
            var carCounts = data.reduce(function(counts, item) {
                counts[item["model-year"]] = (counts[item["model-year"]] || 0) + 1;
                return counts;
            }, {});
            for (var model in carCounts) {
                var count = carCounts[model];
                table += "<tr><td>" + model + "</td><td>" + count +"</td></tr>";}

            table += "</tbody></table>";

            document.getElementById("data-table").innerHTML = table;
        });

    </script>

</body>

</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:

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

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

<body>

    <!-- open CSV FILE
    <input type="file" accept=".csv" id="open"/> -->

    <!-- GENERATE HTML TABLE -->
    <table id="data-table"></table>

    <script>

        d3.csv("auto-mpg.csv").then(function(data) {
            var table = "<table><thead><tr><th>model</th><th>count</th></tr></thead><tbody>";
            var carCounts = data.reduce(function(counts, item) {
                var cylinders = parseInt(item["cylinders"]);
                counts[item["model-year"]] = (counts[item["model-year"]] || 0) + cylinders;
                return counts;
            }, {});
            for (var model in carCounts) {
                var count = carCounts[model];
                table += "<tr><td>" + model + "</td><td>" + count +"</td></tr>";}

            table += "</tbody></table>";

            document.getElementById("data-table").innerHTML = table;
        });

    </script>

</body>

</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:

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

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

<body>

    <!-- open CSV FILE
    <input type="file" accept=".csv" id="open"/> -->

    <!-- GENERATE HTML TABLE -->
    <table id="data-table"></table>

    <script>

        d3.csv("auto-mpg.csv").then(function(data) {
            var table = "<table><thead><tr><th>acceleration</th><th>count</th></tr></thead><tbody>";
            var carCounts = data.reduce(function(counts, item) {

                counts[item["acceleration"]] = (counts[item["acceleration"]] || 0) + 1;
                return counts;
            }, {});
            for (var acceleration in carCounts) {
                var count = carCounts[acceleration];
                table += "<tr><td>" + acceleration + "</td><td>" + count +"</td></tr>";}

            table += "</tbody></table>";

            document.getElementById("data-table").innerHTML = table;
        });

    </script>

</body>

</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:

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

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

<body>
    <svg width="800" height="800" id="pie-chart"></svg>
    <svg width="800" height="400" id="line-chart"></svg>
    <svg width="800" height="400" id="chart"></svg>
    <script>
        var widthPie = 800;
        var heightPie = 800;
        var radiusPie = Math.min(widthPie, heightPie) / 2;
        var svgPie = d3.select('#pie-chart')
            .attr('width', widthPie)
            .attr('height', heightPie)
            .append('g')
            .attr('transform', 'translate(' + widthPie / 2 + ',' + heightPie / 2 + ')');

        d3.csv("auto-mpg.csv").then(function(data){
            var carCounts = data.reduce(function(counts, item) {
                counts[item["model-year"]] = (counts[item["model-year"]] || 0) + 1;
                return counts;
            }, {});
            var totalObservations = Object.values(carCounts).reduce(function(sum, count) {
                                        return sum + count;
                                        }, 0);
            var carPercentages = {};
            for (var model in carCounts) {
                var count = carCounts[model];
                var percentage = (count / totalObservations * 100).toFixed(2);
                carPercentages[model] = percentage;}


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

```
            var pieData = pie(Object.entries(carPercentages).map(function([model, percentage]) {
                        return { model: model, value: percentage };
                    }));

            var arcs = svgPie.selectAll('arc')
            .data(pieData)
            .enter()
            .append('g')
            .attr('class', 'arc');

            var outerRadius = widthPie / 4;
            var innerRadius = 0;
            var arc = d3.arc()
            .innerRadius(innerRadius)
            .outerRadius(outerRadius);

            var color = d3.scaleOrdinal(d3.schemePaired);

            arcs.append('path')
            .attr('d', arc)
            .attr('fill', function(d, i) { return color(i); });

            arcs.append('text')
            .attr('transform', function(d) { return 'translate(' + arc.centroid(d) + ')'; })
            .attr('text-anchor', 'middle')
            .style('font-size', '10px')
            .text(function(d) { return d.data.model + ' ' + d.data.value + '%'; });
        });
    </script>



    <script>
            // Set the dimensions and margins of the graph
    var marginLine = { top: 50, right: 50, bottom: 50, left: 100 },
        widthLine = 800 - marginLine.left - marginLine.right,
        heightLine = 400 - marginLine.top - marginLine.bottom;

    // Parse the date
    d3.csv("auto-mpg.csv").then(function(data){
        var carCounts = data.reduce(function(counts, item) {
```

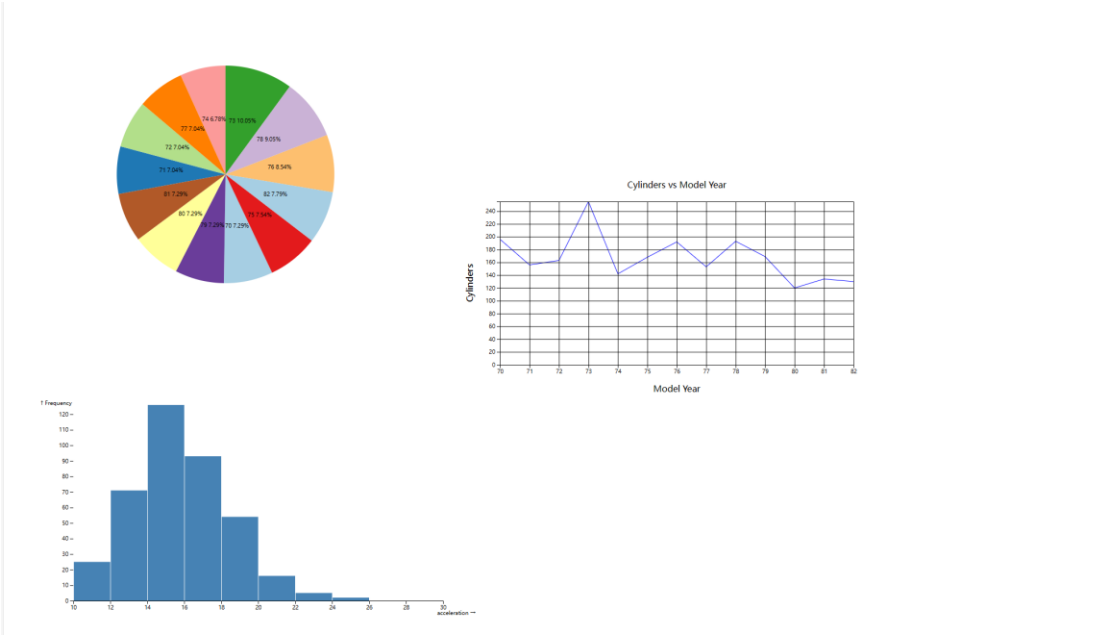
```
    // Parse the date
    d3.csv("auto-mpg.csv").then(function(data){
        var carCounts = data.reduce(function(counts, item) {
                var cylinders = parseInt(item["cylinders"]);
                counts[item["model-year"]] = (counts[item["model-year"]] || 0) + cylinders;
                return counts;
            }, {});
        var arrayOfEntries = Object.entries(carCounts);
        // Set the ranges
        var xScale = d3.scaleLinear()
            .domain(d3.extent(arrayOfEntries, function(d) { return d[0]; }))
            .range([0, widthLine]);

        var yScale = d3.scaleLinear()
            .domain([0, d3.max(arrayOfEntries, function(d) { return d[1]; })])
            .range([heightLine, 0]);

        // Define the line
        var line = d3.line()
        .x(function(d) { return xScale(d[0]); }) // Convert the x-value to a number
        .y(function(d) { return yScale(d[1]); });

        // Append the SVG object to the body of the page
        var svgLine = d3.select("#line-chart")
            .attr("width", widthLine + marginLine.left + marginLine.right)
            .attr("height", heightLine + marginLine.top + marginLine.bottom)
            .append("g")
            .attr("transform", "translate(" + marginLine.left + "," + marginLine.top + ")");

        // Add the line
        svgLine.append("path")
        .datum(arrayOfEntries)
        .attr("class", "line")
        .attr("d", line)
        .attr("stroke", "blue")
        .attr("fill", "transparent")

        // Add the X Axis
        svgLine.append("g")
        .attr("class", "x-axis")
        .attr("transform", "translate(0," + heightLine + ")")
        .call(d3.axisBottom(xScale));
```

```
        // Add the Y Axis
        svgLine.append("g")
        .attr("class", "y-axis")
        .call(d3.axisLeft(yScale));

        // Add X axis label
        svgLine.append("text")
        .attr("class", "x-label")
        .attr("text-anchor", "middle")
        .attr("x", widthLine / 2)
        .attr("y", heightLine + marginLine.bottom)
        .text("Model Year");

        // Add Y axis label
        svgLine.append("text")
        .attr("class", "y-label")
        .attr("text-anchor", "middle")
        .attr("transform", "rotate(-90)")
        .attr("x", -heightLine / 2)
        .attr("y", -marginLine.left / 2)
        .text("Cylinders");

        // Add title
        svgLine.append("text")
        .attr("class", "title")
        .attr("text-anchor", "middle")
        .attr("x", widthLine / 2)
        .attr("y", -marginLine.top / 2)
        .text("Cylinders vs Model Year");

        // Add gridlines for y-axis
        svgLine.append("g")
        .attr("class", "gridlines")
        .call(d3.axisLeft(yScale).tickSize(-widthLine).tickFormat(""));

        // Add gridlines for x-axis
        svgLine.append("g")
        .attr("class", "gridlines")
        .attr("transform", "translate(0," + heightLine + ")")
        .call(d3.axisBottom(xScale).tickSize(-heightLine).tickFormat(""));
    });
    </script>
```

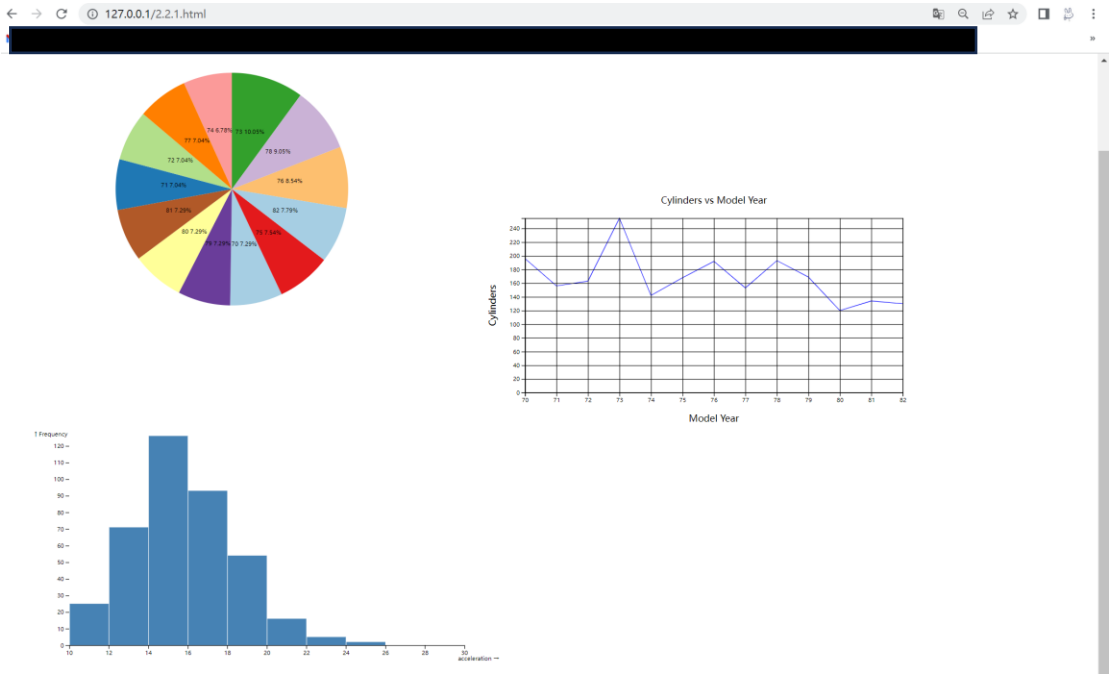
```
    <script>
        var widthBar = 800;
        var heightBar = 400;
        var marginBar = {
            top: 10,
            bottom: 30,
            left: 60,
            right: 60
        };

        var svgBar = d3.select('#chart')
            .attr('width', widthBar + marginBar.left + marginBar.right)
            .attr('height', heightBar + marginBar.top + marginBar.bottom)
            .append('g')
            .attr('transform', `translate(${marginBar.left}, ${marginBar.top})`);


        d3.csv("auto-mpg.csv").then(function(data){
        var histogram = d3.histogram()
            .value(function(d) { return d.acceleration; })
            // .domain(d3.extent(data, function(d) { return d.acceleration; }))
            .domain([10,30])
            .thresholds(10);

            var bins = histogram(data)
            console.log(bins);
        //define x axis and y axis
        const x = d3.scaleLinear()
                    .domain([bins[0].x0, bins[bins.length - 1].x1])
                    .range([marginBar.left, widthBar - marginBar.right]);
        const y = d3.scaleLinear()
            .domain([0, d3.max(bins, (d) => d.length)])
            .range([heightBar - marginBar.bottom, marginBar.top]);


        //add rectangulars to the figure
        svgBar.append("g")
            .attr("fill", "steelblue")
            .selectAll()
            .data(bins)
            .join("rect")
            .attr("x", (d) => x(d.x0) + 1)
```

```
        //add rectangulars to the figure
        svgBar.append("g")
            .attr("fill", "steelblue")
            .selectAll()
            .data(bins)
            .join("rect")
            .attr("x", (d) => x(d.x0) + 1)
            .attr("width", (d) => x(d.x1) - x(d.x0) - 1)
            .attr("y", (d) => y(d.length))
            .attr("height", (d) => y(0) - y(d.length));

        //add axis to the figure
        svgBar.append("g")
            .attr("transform", `translate(0,${heightBar - marginBar.bottom})`)
            .call(d3.axisBottom(x).ticks(widthBar / 80).tickSizeOuter(0))
            .call((g) => g.append("text")
                .attr("x", widthBar)
                .attr("y", marginBar.bottom - 4)
                .attr("fill", "currentColor")
                .attr("text-anchor", "end")
                .text("acceleration →"));

        svgBar.append("g")
            .attr("transform", `translate(${marginBar.left},0)`)
            .call(d3.axisLeft(y).ticks(heightBar / 40))
            .call((g) => g.select(".domain").remove())
            .call((g) => g.append("text")
                .attr("x", -marginBar.left)
                .attr("y", 10)
                .attr("fill", "currentColor")
                .attr("text-anchor", "start")
                .text("↑ Frequency"));
        });



    </script>
</body>
```

The outputs are:




## Problem 3

## Part III

### Problem 1

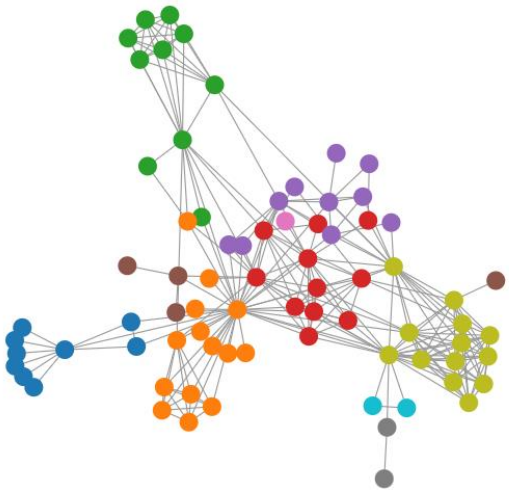
3.1.1 The code is:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Les Misérables Character Co-occurrence</title>
  <script src="https://d3js.org/d3.v5.min.js"></script>
  <style>
    /* Add any custom styles for the graph container */
    #graph-container {
      width: 800px;
      height: 600px;
      margin: auto;
    }
  </style>
</head>
<body>

<svg width = "1600" height = "1200" id="graph"></svg>

<script>
    d3.json("miserables.json").then(function(data) {

    // Create svg
    var svg = d3.select("#graph");

    var width = +svg.attr("width");
    var height = +svg.attr("height");

        // Create simulation
    var simulation = d3.forceSimulation(data.nodes)
        .force("link", d3.forceLink(data.links).id(function(d) { return d.index; }))
        .force("charge", d3.forceManyBody())
        .force("center", d3.forceCenter(400, 300));

    // Create links
    var link = svg.append("g")
        .selectAll("line")
        .data(data.links)
        .enter().append("line")
        .attr("stroke", "#999")
        .attr("strock-width", 2);
```

```
        .attr("stroke", "#999")
        .attr("strock-width", 2);

    var color = d3.scaleOrdinal(d3.schemeCategory10);

    // Create nodes
    var node = svg.selectAll(".node")
        .data(data.nodes)
        .enter().append("circle")
        .attr("class", "node")
        .attr("r", 8)
        .attr("fill", function(d) { return color(d.group); })
        .call(d3.drag()
        .on("start", dragstarted)
        .on("drag", dragged)
        .on("end", dragended));


    // Tick function to update positions of nodes and links
    function ticked() {
        link
        .attr("x1", function(d) { return d.source.x; })
        .attr("y1", function(d) { return d.source.y; })
        .attr("x2", function(d) { return d.target.x; })
        .attr("y2", function(d) { return d.target.y; });

        node
        .attr("cx", function(d) { return d.x; })
        .attr("cy", function(d) { return d.y; });

        label
        .attr("x", function(d) { return d.x; })
        .attr("y", function(d) { return d.y; });
    }

    // Start the simulation
    simulation.on("tick", ticked);
```

```
        .attr("cx", function(d) { return d.x; })
        .attr("cy", function(d) { return d.y; });

    label
        .attr("x", function(d) { return d.x; })
        .attr("y", function(d) { return d.y; });
}

// Start the simulation
simulation.on("tick", ticked);

// Drag functions
function dragstarted(d) {
    if (!d3.event.active) simulation.alphaTarget(0.3).restart();
    d.fx = d.x;
    d.fy = d.y;
}

function dragged(d) {
    d.fx = d3.event.x;
    d.fy = d3.event.y;
}

function dragended(d) {
    if (!d3.event.active) simulation.alphaTarget(0);
    d.fx = null;
    d.fy = null;
}
});
</script>

</body>
</html>
```

The output is:



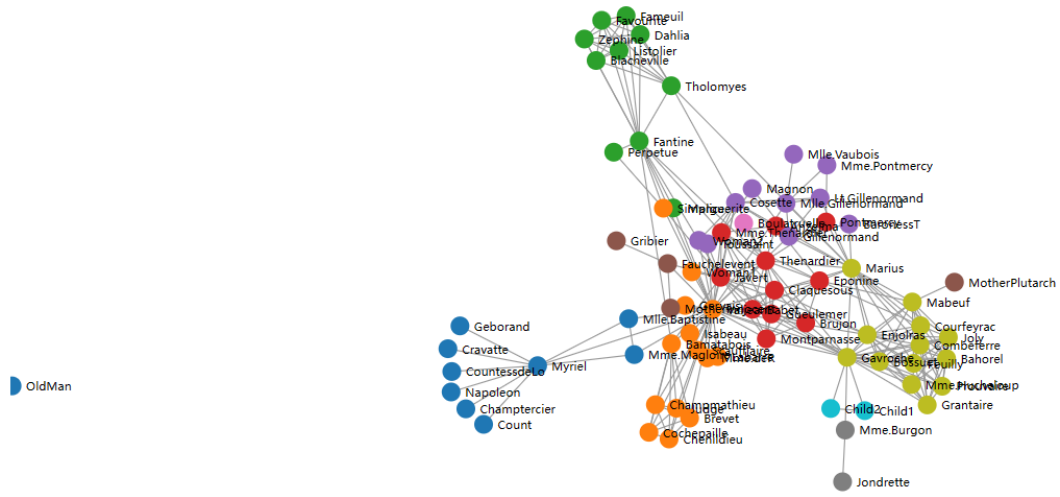
3.1.2 The code is:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Les Misérables Character Co-occurrence</title>
    <script src="https://d3js.org/d3.v5.min.js"></script>
    <style>
    /* Add any custom styles for the graph container */
    #graph-container {
        width: 800px;
        height: 600px;
        margin: auto;
    }
    </style>
</head>
<body>

<svg width = "1600" height = "1200" id="graph"></svg>

<script>
    d3.json("miserables.json").then(function(data) {

    // Create svg
    var svg = d3.select("#graph");

    var width = +svg.attr("width");
    var height = +svg.attr("height");

        // Create simulation
    var simulation = d3.forceSimulation(data.nodes)
        .force("link", d3.forceLink(data.links).id(function(d) { return d.index; }))
        .force("charge", d3.forceManyBody())
        .force("center", d3.forceCenter(400, 300));

    // Create links
    var link = svg.append("g")
        .selectAll("line")
        .data(data.links)
        .enter().append("line")
        .attr("stroke", "#999")
        .attr("strock-width", 2);
```

```
        .attr("strock-width", 2);

    var color = d3.scaleOrdinal(d3.schemeCategory10);

    // Create nodes
    var node = svg.selectAll(".node")
        .data(data.nodes)
        .enter().append("circle")
        .attr("class", "node")
        .attr("r", 8)
        .attr("fill", function(d) { return color(d.group); })
        .call(d3.drag()
        .on("start", dragstarted)
        .on("drag", dragged)
        .on("end", dragended));

    // Add labels to nodes
    var label = svg.selectAll(".label")
        .data(data.nodes)
        .enter().append("text")
        .attr("class", "label")
        .text(function(d) { return d.name; })
        .attr("font-size", "10px")
        .attr("dx", 12)
        .attr("dy", 4);
    // Tick function to update positions of nodes and links
    function ticked() {
        link
        .attr("x1", function(d) { return d.source.x; })
        .attr("y1", function(d) { return d.source.y; })
        .attr("x2", function(d) { return d.target.x; })
        .attr("y2", function(d) { return d.target.y; });

        node
        .attr("cx", function(d) { return d.x; })
        .attr("cy", function(d) { return d.y; });

        label
        .attr("x", function(d) { return d.x; })
        .attr("y", function(d) { return d.y; });
```

```
        label
        .attr("x", function(d) { return d.x; })
        .attr("y", function(d) { return d.y; });
    }

    // Start the simulation
    simulation.on("tick", ticked);

    // Drag functions
    function dragstarted(d) {
        if (!d3.event.active) simulation.alphaTarget(0.3).restart();
        d.fx = d.x;
        d.fy = d.y;
    }

    function dragged(d) {
        d.fx = d3.event.x;
        d.fy = d3.event.y;
    }

    function dragended(d) {
        if (!d3.event.active) simulation.alphaTarget(0);
        d.fx = null;
        d.fy = null;
    }
    });
</script>

</body>
</html>
```

The output is:




### 3.1.3 The code is:

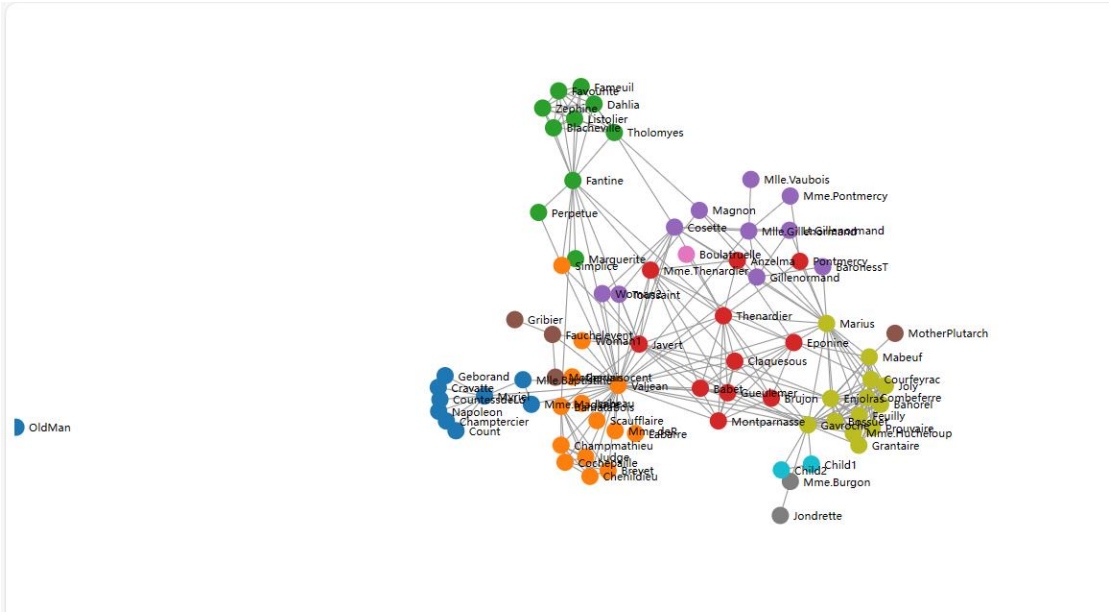
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Les Misérables Character Co-occurrence</title>
    <script src="https://d3js.org/d3.v5.min.js"></script>
    <script src="fisheye.js"></script>

    <style>
    /* Add any custom styles for the graph container */
    #graph-container {
        width: 800px;
        height: 600px;
        margin: auto;
    }
    </style>
</head>
<body>

<svg width = "1600" height = "1200" id="graph"></svg>

<script>
    d3.json("miserables.json").then(function(data) {

    // Create svg
    var svg = d3.select("#graph");

    var width = +svg.attr("width");
    var height = +svg.attr("height");


        // Create simulation
    var simulation = d3.forceSimulation(data.nodes)
        .force("link", d3.forceLink(data.links).id(function(d) { return d.index; }))
        .force("charge", d3.forceManyBody())
        .force("center", d3.forceCenter(400, 300));

    // Create links
    var link = svg.append("g")
        .selectAll("line")
        .data(data.links)
```

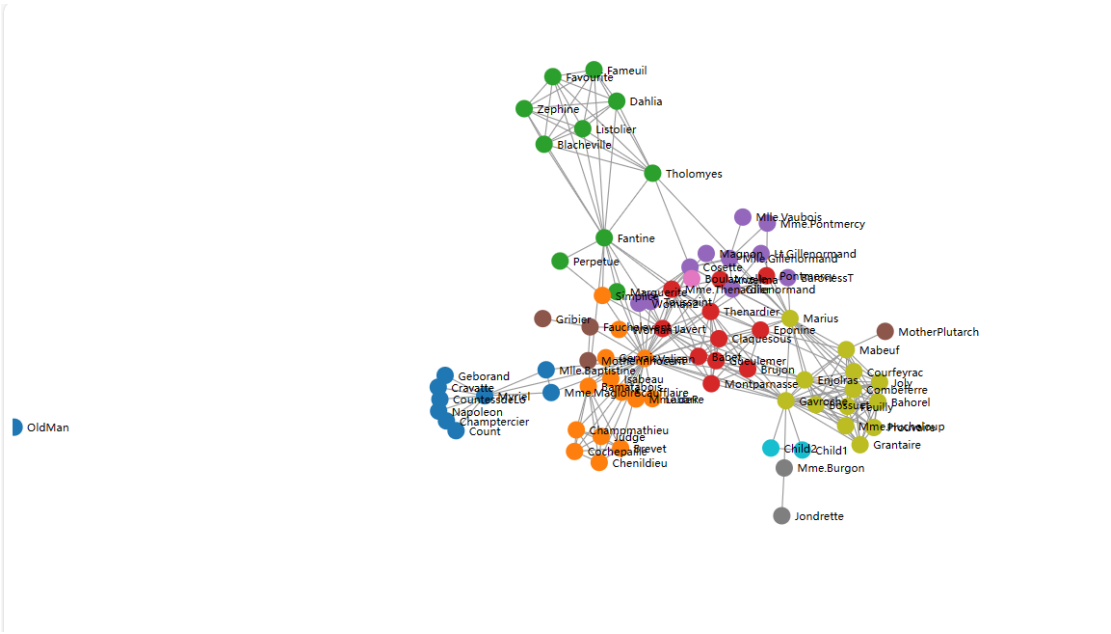
```
    var link = svg.append("g")
        .selectAll("line")
        .data(data.links)
        .enter().append("line")
        .attr("stroke", "#999");

    var color = d3.scaleOrdinal(d3.schemeCategory10);

    // Create nodes
    var node = svg.selectAll(".node")
        .data(data.nodes)
        .enter().append("circle")
        .attr("class", "node")
        .attr("r", 8)
        .attr("fill", function(d) { return color(d.group); })
        .call(d3.drag()
        .on("start", dragstarted)
        .on("drag", dragged)
        .on("end", dragended));

    // Add labels to nodes
    var label = svg.selectAll(".label")
        .data(data.nodes)
        .enter().append("text")
        .attr("class", "label")
        .text(function(d) { return d.name; })
        .attr("font-size", "10px")
        .attr("dx", 12)
        .attr("dy", 4);

    //update positions of nodes and links
    function ticked() {
        link
        .attr("x1", function(d) { return d.source.x; })
        .attr("y1", function(d) { return d.source.y; })
        .attr("x2", function(d) { return d.target.x; })
        .attr("y2", function(d) { return d.target.y; });

        node
        .attr("cx", function(d) { return d.x; })
        .attr("cy", function(d) { return d.y; });
```

```
        label
            .attr("x", function(d) { return d.x; })
            .attr("y", function(d) { return d.y; });
    }

    // Start the simulation
    simulation.on("tick", ticked);

    // Drag functions
    function dragstarted(d) {
        if (!d3.event.active) simulation.alphaTarget(0.3).restart();
        d.fx = d.x;
        d.fy = d.y;
    }

    function dragged(d) {
        d.fx = d3.event.x;
        d.fy = d3.event.y;
    }

    function dragended(d) {
        if (!d3.event.active) simulation.alphaTarget(0);
        d.fx = null;
        d.fy = null;
    }

    var fisheye = d3.fisheye.circular()
    .radius(200)
    .distortion(2);
    svg.on("mousemove", function() {
    fisheye.focus(d3.mouse(this));

    // Apply fisheye distortion to nodes
    node.each(function(d) {
        d.fisheye = fisheye(d);
    });

    // Update node positions
    node.attr("cx", function(d) { return d.fisheye.x; })
        .attr("cy", function(d) { return d.fisheye.y; });
```

```
    // Apply fisheye distortion to nodes
    node.each(function(d) {
        d.fisheye = fisheye(d);
    });

    // Update node positions
    node.attr("cx", function(d) { return d.fisheye.x; })
        .attr("cy", function(d) { return d.fisheye.y; });

    // Update link positions
    link.attr("x1", function(d) { return d.source.fisheye.x; })
        .attr("y1", function(d) { return d.source.fisheye.y; })
        .attr("x2", function(d) { return d.target.fisheye.x; })
        .attr("y2", function(d) { return d.target.fisheye.y; });

    // Update label positions
    label.attr("x", function (d) { return d.fisheye.x; })
        .attr("y", function (d) { return d.fisheye.y; });
    });
});
</script>

</body>
</html>
```

The output is:



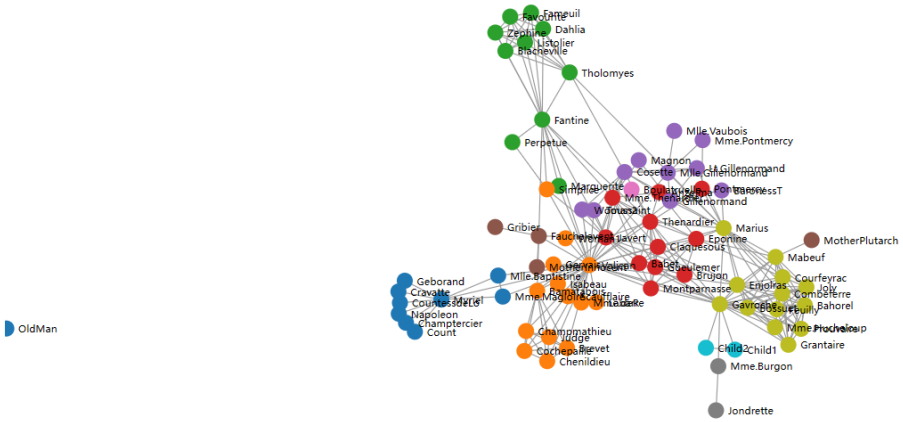

3.1.4 The code is:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Les Misérables Character Co-occurrence</title>
    <script src="https://d3js.org/d3.v5.min.js"></script>
    <script src="fisheye.js"></script>

</head>
<body>

<svg width = "1200" height = "600" id="graph"></svg>
<p>
    Boulatruelle's group has the smallest number of people, with only one person.
</p>
<p>
    The group dyed orange has the largest number of people, with fourteen people.
</p>
<p>
    MotherPlutarch, Gribier, and Jondrette are all related to only one person.
</p>
<p>
    Oldman is not connected with anyone.
</p>
<script>
    d3.json("miserables.json").then(function(data) {

    // Create svg
    var svg = d3.select("#graph");

    var width = +svg.attr("width");
    var height = +svg.attr("height");


        // Create simulation
    var simulation = d3.forceSimulation(data.nodes)
        .force("link", d3.forceLink(data.links).id(function(d) { return d.index; }))
        .force("charge", d3.forceManyBody())
        .force("center", d3.forceCenter(600, 300));

    // Create links
    var link = svg.append("g")
        .selectAll("line")
        .data(data.links)
        .enter().append("line")
        .attr("stroke", "#999");

    var color = d3.scaleOrdinal(d3.schemeCategory10);

    // Create nodes
    var node = svg.selectAll(".node")
        .data(data.nodes)
        .enter().append("circle")
        .attr("class", "node")
        .attr("r", 8)
        .attr("fill", function(d) { return color(d.group); })
        .call(d3.drag()
        .on("start", dragstarted)
        .on("drag", dragged)
        .on("end", dragended));

    // Add labels to nodes
    var label = svg.selectAll(".label")
        .data(data.nodes)
        .enter().append("text")
        .attr("class", "label")
        .text(function(d) { return d.name; })
        .attr("font-size", "10px")
        .attr("dx", 12)
        .attr("dy", 4);

    //update positions of nodes and links
    function ticked() {
        link
        .attr("x1", function(d) { return d.source.x; })
        .attr("y1", function(d) { return d.source.y; })
        .attr("x2", function(d) { return d.target.x; })
        .attr("y2", function(d) { return d.target.y; });
```

```
        node
        .attr("cx", function(d) { return d.x; })
        .attr("cy", function(d) { return d.y; });

        label
        .attr("x", function(d) { return d.x; })
        .attr("y", function(d) { return d.y; });
    }

    // Start the simulation
    simulation.on("tick", ticked);

    // Drag functions
    function dragstarted(d) {
        if (!d3.event.active) simulation.alphaTarget(0.3).restart();
        d.fx = d.x;
        d.fy = d.y;
    }

    function dragged(d) {
        d.fx = d3.event.x;
        d.fy = d3.event.y;
    }

    function dragended(d) {
        if (!d3.event.active) simulation.alphaTarget(0);
        d.fx = null;
        d.fy = null;
    }

    var fisheye = d3.fisheye.circular()
    .radius(200)
    .distortion(2);
    svg.on("mousemove", function() {
    fisheye.focus(d3.mouse(this));

    // Apply fisheye distortion to nodes
    node.each(function(d) {
        d.fisheye = fisheye(d);
    });
```

```
        // Update node positions
        node.attr("cx", function(d) { return d.fisheye.x; })
            .attr("cy", function(d) { return d.fisheye.y; });

        // Update link positions
        link.attr("x1", function(d) { return d.source.fisheye.x; })
            .attr("y1", function(d) { return d.source.fisheye.y; })
            .attr("x2", function(d) { return d.target.fisheye.x; })
            .attr("y2", function(d) { return d.target.fisheye.y; });

        // Update label positions
        label.attr("x", function (d) { return d.fisheye.x; })
            .attr("y", function (d) { return d.fisheye.y; });
        });
    });
    </script>

    </body>
    </html>
```

The output is:



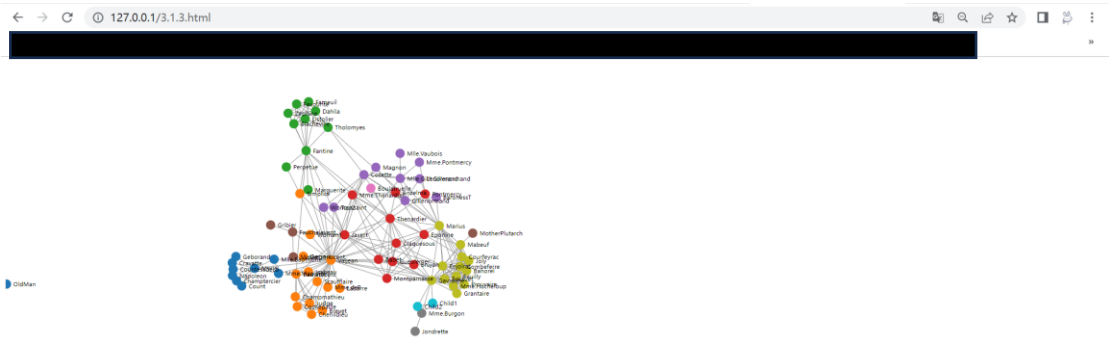

Boulatruelle's group has the smallest number of people, with only one person.

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.

## Problem 2




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.