

EECS6893 Big Data Analytics HW3

Nov 10, 2022



Lijie Huang
lh3158

Part I Tasks

Problem 1

1.1.1

What's the difference between SVG Coordinate Space and Mathematical / Graph Coordinate Space?

Answer:

SVG coordinate Space is similar to Mathematical / Graph Coordinate Space but the Y coordinate of SVG Coordinate Space increases from top to bottom.

1.1.2

What is `enter()` and `exit()` in d3.js?

Answer:

enter() creates the initial join of data to elements, creating one circle element for every data element in the array. *exit()* removes any circle elements no longer needed and create any new circle elements by using *enter()*.

1.1.3

What is transform and translate in SVG?

Answer:

SVG provides options to transform a single SVG shape element or group of SVG elements. SVG transform supports Translate, Scale, Rotate and Skew. Translate takes two options, *tx* refers to translation along the x-axis and *ty* refers to the translation along the y-axis.

1.1.4

Try to understand the idea of anonymous function and its use in d3.js.

If there is a list `a = [a,c,b,d,e]`, what is the return value of this anonymous function:
`a.map(function(d,i) {return i+5})`?

Answer:

The return value should be `[5, 6, 7, 8, 9]`.

1.1.5

Compare the 2 code snippets below

What will be the output of the 2 code snippets will it be different or same. If it is different, give reasoning on why it is different.

Code snippet I:

```
<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>
```

Code snippet II:

```
<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>
```

The output of the first code will be

Big

The output of the second code will be

Big Data Assignment 2 Submission

They are different because *enter()* in the second code creates the initial join of all data to elements.

Problem 2

Create webpages with HTML, CSS and plot various plots as asked in each subpart with d3.js. (follow tutorial for help). Use the Seattle weather dataset and modify the sample code for getting the output. Paste the code snippets and output pictures in your submission file.

1.2.1

Plot basic histogram for the wind variable using D3, set bins = 10. Label the X and Y axis properly.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="https://d3js.org/d3.v4.js"></script>
</head>
<body>
  <!-- Create a div where the graph will take place -->
  <div id="histogram"></div>
</body>
</html>

<script>
  // set the dimensions and margins of the graph
  var margin = {top: 20, right: 50, bottom: 50, left: 60},
      width = 460 - margin.left - margin.right,
      height = 400 - margin.top - margin.bottom;

  // append the svg object to the body of the page
  var svg = d3.select("#histogram")
    .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 + ")");

  // getting the data in csv format
  d3.csv("https://raw.githubusercontent.com/vega/vega/main/docs/data/seattle-weather.csv",
  function (data) {
    // X axis
    var x = d3.scaleLinear()
      .domain([0,10])
      .range([0, width]);
    svg.append("g")
      .attr("transform", "translate(0," + height + ")")
      .call(d3.axisBottom(x));

    //setting the parameters for the histogram
    var binNum = 10
    var histogram = d3.histogram()
      .value(function(d) {return d.wind;})
      .domain(x.domain())
      .thresholds(x.ticks(binNum)); // change this parameter to 25
    var bins = histogram(data);

    // Y axis
    var y = d3.scaleLinear()
```

```

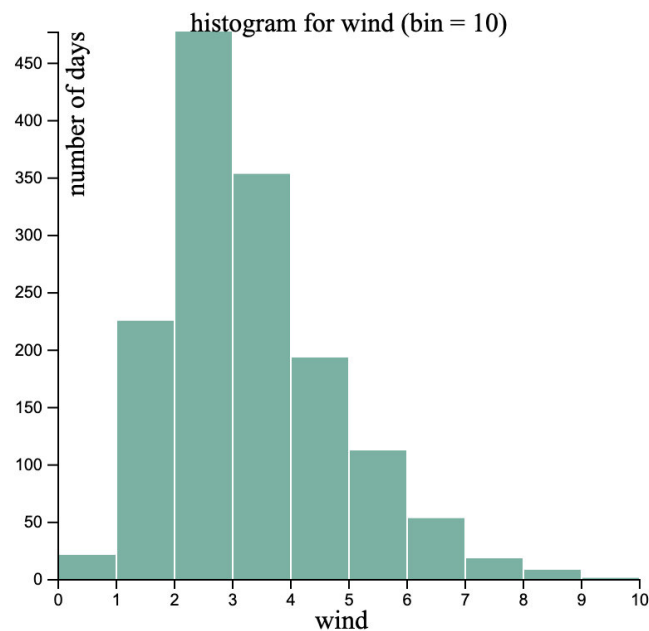
        .range([height, 0]);
    y.domain([0, d3.max(bins, function(d) {return d.length;})]);
    svg.append("g")
        .call(d3.axisLeft(y));

    // setting rectangle
    svg.selectAll("rect")
        .data(bins)
        .enter()
        .append("rect")
            .attr("x", 1)
            .attr("transform", function(d) {return "translate(" + x(d.x0) + "," + y(d.length) +
    ");";})

            .attr("width", function(d) {return x(d.x1) - x(d.x0) - 1;})
            .attr("height", function(d) {return height - y(d.length);})
            .style("fill", "#69b3a2");

    // add notations
    svg.append("g")
        .append('text')
            .attr("x", 80)
            .attr("y", 0)
            .text("histogram for wind (bin = " + binNum + ")")
    svg.append("g")
        .append('text')
            .attr("x", width / 2 - 20)
            .attr("y", 380 - margin.top)
            .text("wind")
    svg.append("g")
        .call(y)
        .append("text")
            .attr("transform", "rotate(-90)")
            .attr("dy", "1em")
            .attr("text-anchor", "end")
            .text("number of days")
    });
</script>

```

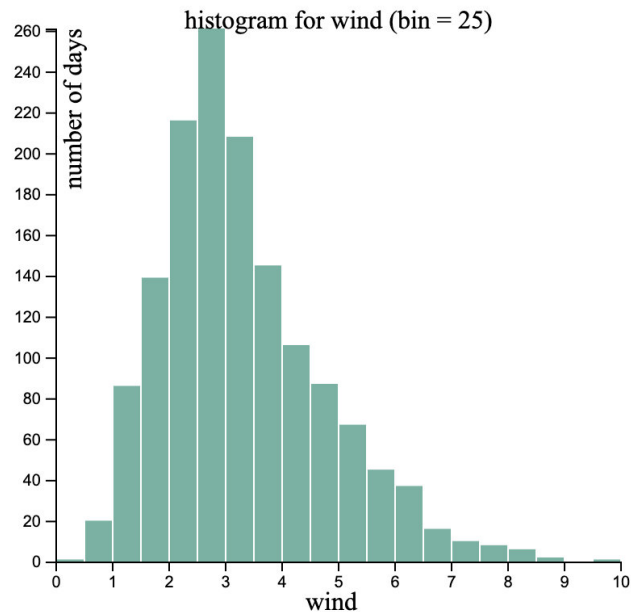


1.2.2

Now, change the number of bins to 25.

change binNum variable to 25 of the line below in the previous codes.

```
var binNum = 25
```



1.2.3

Now, plot a pie chart using D3 to show the distribution of the weather variable, properly label the sectors with percentage and annotate each sector with corresponding weather variable.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="https://d3js.org/d3.v4.js"></script>
</head>
<body>
  <svg width="960" height="500"></svg>
</body>
</html>

<script>
// appending the svg object to the svg id of the page
var svg = d3.select("svg"),
    width = +svg.attr("width"),
    height = +svg.attr("height"),
    radius = Math.min(width, height) / 2,
```

```

    g = svg.append("g").attr("transform", "translate(" + width / 2 + "," + height / 2 + ")");

    // setting color scale for the pie chart
    var color = d3.scaleOrdinal(["#98abc5", "#8a89a6", "#7b6888", "#6b486b", "#a05d56", "#d0743c",
    "#ff8c00"]);

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

    var path = d3.arc()
        .outerRadius(radius - 60)
        .innerRadius(0);

    var label = d3.arc()
        .outerRadius(radius)
        .innerRadius(radius - 60);

    d3.csv("https://raw.githubusercontent.com/vega/vega/main/docs/data/seattle-weather.csv", function
    (data) {
        var weatherList = data.map(function(d){return d.weather;});
        var weatherCount = weatherList.reduce((pre, cur) => {
            if (cur in pre) pre[cur]++;
            else pre[cur] = 1;
            return pre;
        }, {});

        var newData = [];
        for (var weather in weatherCount) {
            var d = {};
            d['weather'] = (weather);
            d['count'] = (weatherCount[weather]);
            d['percentage'] = (weatherCount[weather] / data.length);
            newData.push(d);
        }

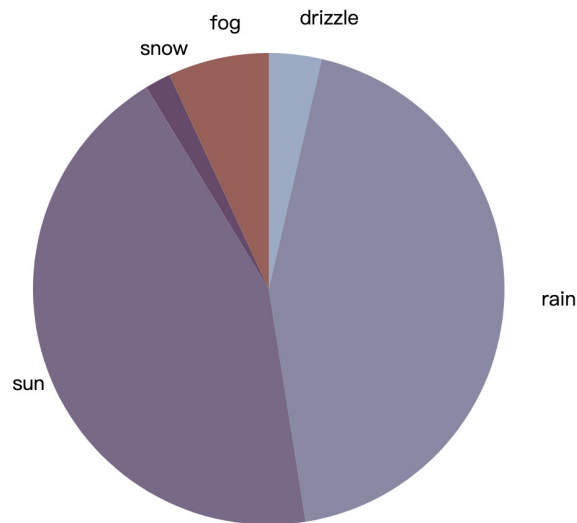
        var arc = g.selectAll(".arc")
            .data(pie(newData))
            .enter().append("g")
            .attr("class", "arc");

        arc.append("path")
            .attr("d", path)
            .attr("fill", function(d) {return color(d.data.weather);});

        arc.append("text")
            .attr("transform", function(d) {return "translate(" + label.centroid(d) + ")";})
            .attr("dy", "0.35em")
            .text(function(d){return d.data.weather;});

        svg.append("text")
            .attr("x", width / 2 - 100)
            .attr("y", height)
            .text("Pie chart of weather variable")
    });
</script>

```



Pie chart of weather variable

1.2.4

Now, plot a line graph using D3 to show the trend of the precipitation variable with respect to the date variable. Label the X and Y axis properly.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="https://d3js.org/d3.v4.js"></script>
</head>
<body>
  <div id = "lineGraph"></div>
</body>
</html>

<script>
  var margin = {top: 20, right: 50, bottom: 50, left: 60},
      width = 1000 - margin.left - margin.right,
      height = 400 - margin.top - margin.bottom;

  var svg = d3.selectAll("#lineGraph")
    .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("https://raw.githubusercontent.com/vega/vega/main/docs/data/seattle-weather.csv",
    function(d) {
      return {date : d3.timeParse("%Y-%m-%d")(d.date), value : d.precipitation}
    },
```



```

function(data) {
  // setting the X axis
  var x = d3.scaleTime()
    .domain(d3.extent(data, function(d){return d.date;}))
    .range([0, width]);

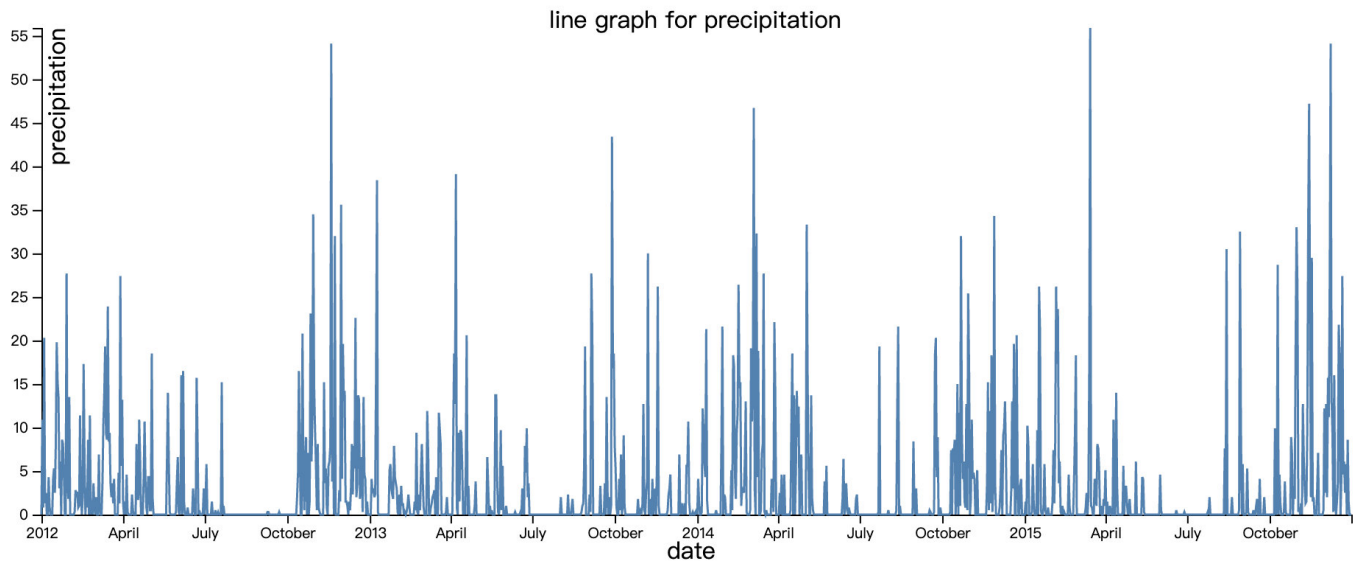
  svg.append("g")
    .attr("transform", "translate(0,"+height+"")")
    .call(d3.axisBottom(x));

  // setting the Y axis
  var y = d3.scaleLinear()
    .domain([0, d3.max(data, function(d){return +d.value;})])
    .range([height, 0]);

  svg.append("g")
    .call(d3.axisLeft(y));

  // adding the line
  svg.append("path")
    .datum(data)
    .attr("fill", "none")
    .attr("stroke", "steelblue")
    .attr("stroke-width", 1.5)
    .attr("d", d3.line()
      .x(function(d){return x(d.date)})
      .y(function(d){return y(d.value)})
    )
  },
)
svg.append("g")
  .append('text')
  .attr("x", width / 2 - 100)
  .attr("y", 0)
  .text("line graph for precipitation");
svg.append("g")
  .append('text')
  .attr("x", width / 2 - 20)
  .attr("y", 380 - margin.top)
  .text("date");
svg.append("g")
  .append("text")
  .attr("x", 0)
  .attr("transform", "rotate(-90)")
  .attr("dy", "1em")
  .attr("text-anchor", "end")
  .text("precipitation");
</script>

```



1.2.5

Add a paragraph to the webpage using the

tag to note down the main observations from the above obtained plots in 1.2.1, 1.2.2, 1.2.3, 1.2.4 (for eg. distribution, count, important understanding etc).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>
    main observation
  </h1>
  <p>
    1.2.1 The most common seen wind degree in Seattle is around 2.0 to 3.0. with more than 460
days.
    The number of days gradually decreases when the wind degree increases.
  </p>
  <p>
    1.2.2 When the number of bins is set to 25, the trend of bar charts is the same with
the chart in 1.2.1. With more bins, the differences between wind degrees can be observed
more clearly.
  </p>
  <p>
    1.2.3 The most days in Seattle are sun and rain.
  </p>
  <p>
    1.2.4 Precipitation in Seattle varies amongs continues days. Precipitation is commonly the
least
    between August and October every year.
```

```
</p>
</body>
</html>
```

main observation

1.2.1 The most common seen wind degree in Seattle is around 2.0 to 3.0. with more than 460 days. The number of days gradually decreases when the wind degree increases.

1.2.2 When the number of bins is set to 25, the trend of bar charts is the same with the chart in 1.2.1. With more bins, the differences between wind degrees can be observed more clearly.

1.2.3 The most days in Seattle are sun and rain.

1.2.4 Precipitation in Seattle varies amongs continues days. Precipitation is commonly the least between August and October every year.

1.2.6

Use the internal style sheet to change the style (font size, font color,indentation) of the paragraph of the web page you created in 1.2.5.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>
    main observation
  </h1>
  <p style="color: #69b3a2">
    1.2.1 The most common seen wind degree in Seattle is around 2.0 to 3.0. with more than 460
days.
    The number of days gradually decreases when the wind degree increases.
  </p>
  <p style="background-color: yellow">
    1.2.2 When the number of bins is set to 25, the trend of bar charts is the same with
    the chart in 1.2.1. With more bins, the differences between wind degrees can be observed
    more clearly.
  </p>
  <p style="font-size:x-large">
    1.2.3 The most days in Seattle are sun and rain.
  </p>
  <p style="font-weight: 800; color:red">
    1.2.4 Precipitation in Seattle varies amongs continues days. Precipitation is commonly the
least
    between August and October every year.
  </p>
</body>
```

```
</html>
```

main observation

1.2.1 The most common seen wind degree in Seattle is around 2.0 to 3.0. with more than 460 days. The number of days gradually decreases when the wind degree increases.

1.2.2 When the number of bins is set to 25, the trend of bar charts is the same with the chart in 1.2.1. With more bins, the differences between wind degrees can be observed more clearly.

1.2.3 The most days in Seattle are sun and rain.

1.2.4 Precipitation in Seattle varies amongs continues days. Precipitation is commonly the least between August and October every year.

Problem 3

1.3.1

Using the same histogram from 1.2.1 add an interactive component such that we can toggle through the bins size. (We should be able to select any bin size from range 5-25 with increments of 5)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="https://d3js.org/d3.v4.js"></script>
</head>
<body>
  <div id="my_dataviz"></div>
  <p>
    <label># bins</label>
    <input type="number" min="0" max="100" step="5" value="20" id="nBin">
  </p>
</body>
</html>

<script>
// set the dimensions and margins of the graph
var margin = {top: 10, right: 30, bottom: 30, left: 40},
    width = 460 - margin.left - margin.right,
    height = 400 - margin.top - margin.bottom;

// append the svg object to the body of the page
var svg = d3.select("#my_dataviz")
  .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 + ")");

// get the data
d3.csv("https://raw.githubusercontent.com/vega/vega/main/docs/data/seattle-weather.csv",
function(data) {

    // X axis: scale and draw:
    var x = d3.scaleLinear()
        .domain([0, 10]) // can use this instead of 1000 to have the max of data: d3.max(data,
function(d) { return +d.price })
        .range([0, width]);
    svg.append("g")
        .attr("transform", "translate(0," + height + ")")
        .call(d3.axisBottom(x));

    // Y axis: initialization
    var y = d3.scaleLinear()
        .range([height, 0]);
    var yAxis = svg.append("g")

    svg.append('text')
        .attr("x", 120)
        .attr("y", 10)
        .text("histogram for wind");
    svg.append('text')
        .attr("x", width / 2 - 20)
        .attr("y", 400 - margin.top)
        .text("wind");
    svg.append("text")
        .call(y)
        .attr("transform", "rotate(-90)")
        .attr("dy", "1em")
        .attr("text-anchor", "end")
        .text("number of days");

    // A function that builds the graph for a specific value of bin
    function update(nBin) {

        // set the parameters for the histogram
        var histogram = d3.histogram()
            .value(function(d) { return d.wind; }) // I need to give the vector of value
            .domain(x.domain()) // then the domain of the graphic
            .thresholds(x.ticks(nBin)); // then the numbers of bins

        // And apply this function to data to get the bins
        var bins = histogram(data);

        // Y axis: update now that we know the domain
        y.domain([0, d3.max(bins, function(d) { return d.length; })]); // d3.hist has to be called
        before the Y axis obviously
        yAxis
            .transition()
            .duration(1000)
            .call(d3.axisLeft(y));
    }
}

```

```

// Join the rect with the bins data
var u = svg.selectAll("rect")
    .data(bins)

// Manage the existing bars and eventually the new ones:
u
    .enter()
    .append("rect") // Add a new rect for each new elements
    .merge(u) // get the already existing elements as well
    .transition() // and apply changes to all of them
    .duration(1000)
    .attr("x", 1)
    .attr("transform", function(d) { return "translate(" + x(d.x0) + "," + y(d.length) + ")";

})

    .attr("width", function(d) { return x(d.x1) - x(d.x0) - 1 ; })
    .attr("height", function(d) { return height - y(d.length); })
    .style("fill", "#69b3a2")

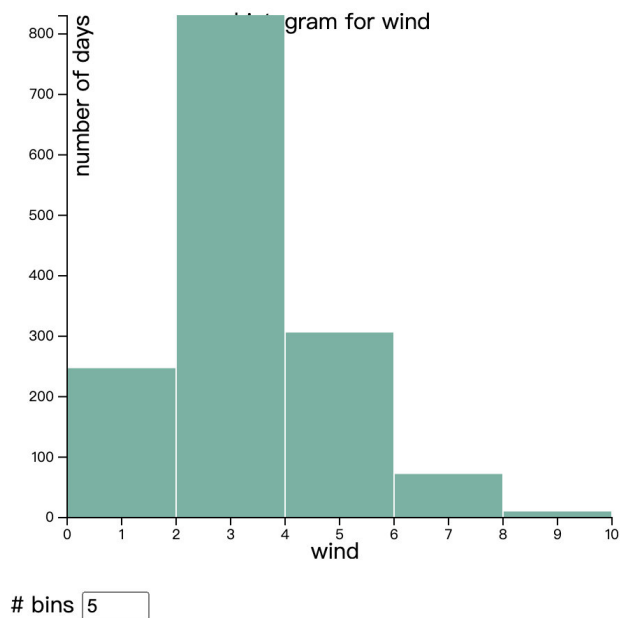
// If less bar in the new histogram, I delete the ones not in use anymore
u
    .exit()
    .remove()
}

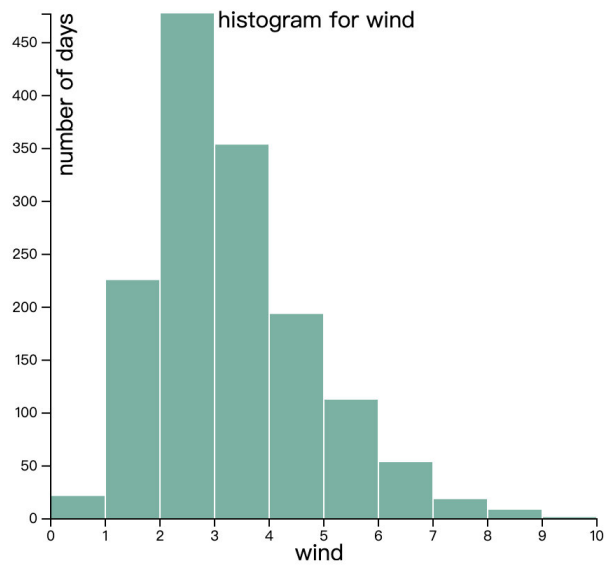
// Initialize with 20 bins
update(20)

// Listen to the button -> update if user change it
d3.select("#nBin").on("input", function() {
    update(+this.value);
});

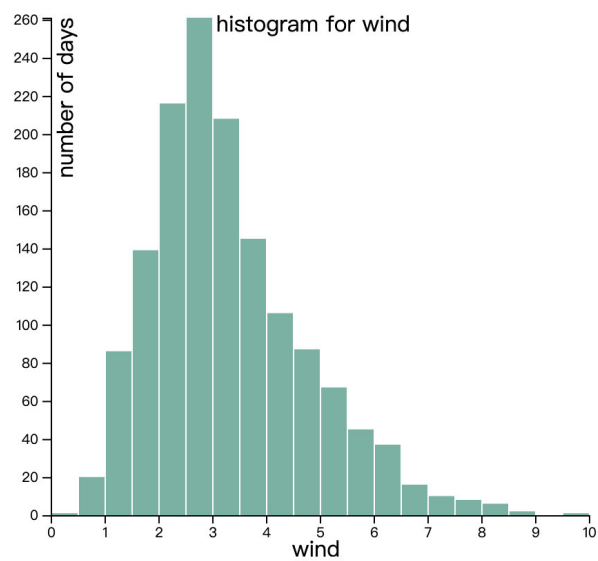
});
</script>

```

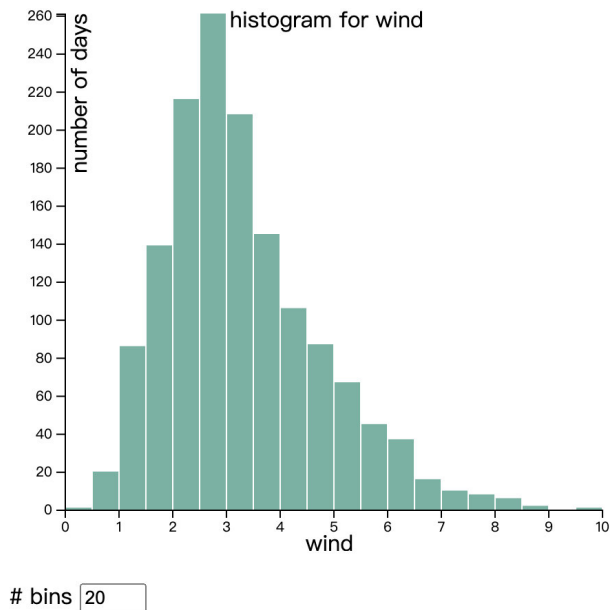




bins



bins



1.3.2

Add a drop down menu with precipitation, temp_max, temp_min and wind variables to get histograms for each of these variables upon selection of each variable from the dropdown menu.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="https://d3js.org/d3.v4.js"></script>
</head>
<body>
  <div id="my_dataviz"></div>
  <div id="dropdownButton"></div>
</body>
</html>

<script>
  // set the dimensions and margins of the graph
  var margin = {top: 10, right: 30, bottom: 30, left: 40},
      width = 460 - margin.left - margin.right,
      height = 400 - margin.top - margin.bottom;

  // append the svg object to the body of the page
  var svg = d3.select("#my_dataviz")
    .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 + ")");

  // add dropdown button
```



```

variableGroup = ["wind", "temp_max", "temp_min", "precipitation"];
curVar = "wind"; // default variable

var dropdownButton = d3.select("#dropdownButton")
    .append('select')

dropdownButton.selectAll("myOptions")
    .data(variableGroup)
    .enter()
    .append("option")
    .text(function(d) {return d;})
    .attr("value", function(d){return d;})

// get the data
d3.csv("https://raw.githubusercontent.com/vega/vega/main/docs/data/seattle-weather.csv",
function(data) {
    // Y axis: initialization
    var y = d3.scaleLinear()
        .range([height, 0]);
    var yAxis = svg.append("g");

    // var x = d3.scaleLinear()
    //     .range([0, width]);
    // var xAxis = svg.append("g").
    //     attr("transform", "translate(0," + height + ")");

    // A function that builds the graph for a specific value of bin
    function update(curVar) {
        svg.selectAll("g.tick").remove();
        svg.selectAll("text").remove();

        var minX = d3.min(data, function(d) {
            if (curVar == "wind") return d.wind;
            if (curVar == "temp_max") return d.temp_max;
            if (curVar == "temp_min") return d.temp_min;
            if (curVar == "precipitation") return d.precipitation;
        })

        var maxX = d3.max(data, function(d) {
            if (curVar == "wind") return d.wind;
            if (curVar == "temp_max") return d.temp_max;
            if (curVar == "temp_min") return d.temp_min;
            if (curVar == "precipitation") return d.precipitation;
        })

        var x = d3.scaleLinear()
            .domain([minX, maxX])
            .range([minX, width]);

        svg.append("g")
            .attr("transform", "translate(0," + height + ")")
            .call(d3.axisBottom(x));

        // set the parameters for the histogram
        var histogram = d3.histogram()
            .value(function(d){
                if (curVar == "wind") return d.wind;
                if (curVar == "temp_max") return d.temp_max;
                if (curVar == "temp_min") return d.temp_min;
            })
    }
}

```

```

        if (curVar == "precipitation") return d.precipitation;
    })
    .domain(x.domain())
    .thresholds(x.ticks(10));

    // And apply this function to data to get the bins
    var bins = histogram(data);

    // Y axis: update now that we know the domain
    y.domain([0, d3.max(bins, function(d) { return d.length; })]); // d3.hist has to be called
    before the Y axis obviously
    yAxis
        .transition()
        .duration(1000)
        .call(d3.axisLeft(y));

    // Join the rect with the bins data
    var u = svg.selectAll("rect")
        .data(bins)

    // Manage the existing bars and eventually the new ones:
    u
        .enter()
        .append("rect") // Add a new rect for each new elements
        .merge(u) // get the already existing elements as well
        .transition() // and apply changes to all of them
        .duration(1000)
        .attr("x", 1)
        .attr("transform", function(d) { return "translate(" + x(d.x0) + "," + y(d.length) + ")";

    })

        .attr("width", function(d) { return x(d.x1) - x(d.x0) - 1 ; })
        .attr("height", function(d) { return height - y(d.length); })
        .style("fill", "#69b3a2")

    // If less bar in the new histogram, I delete the ones not in use anymore
    u.exit().remove();

    svg.append("text")
        .attr("x", 120)
        .attr("y", 10)
        .text("histogram of " + curVar)

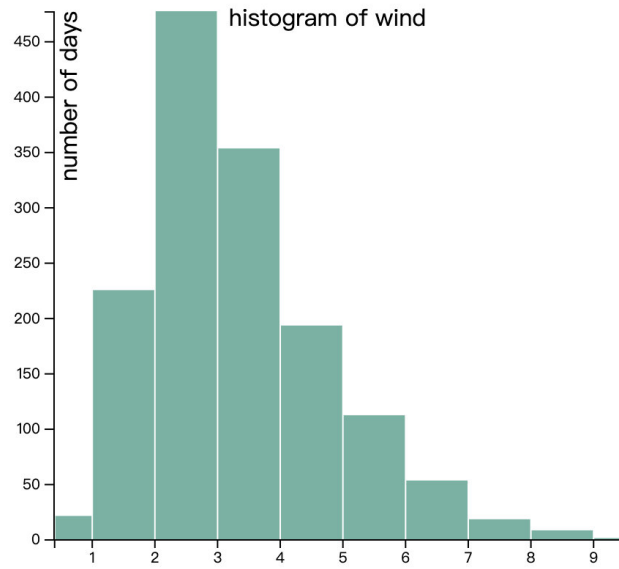
    svg.append("text")
        .call(y)
        .attr("transform", "rotate(-90)")
        .attr("dy", "1em")
        .attr("text-anchor", "end")
        .text("number of days")

    }
    update(curVar);

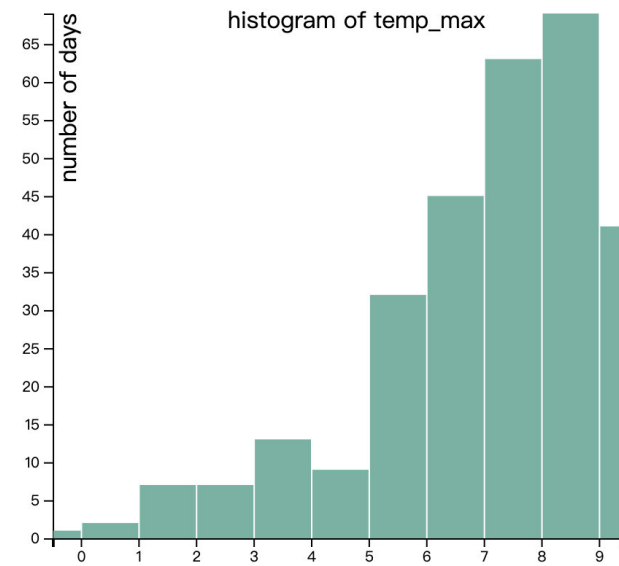
    // Listen to the button -> update if user change it
    dropdownButton.on("change", function(d){
        curVar = d3.select(this).property("value");
        update(curVar);
    })
    });

```

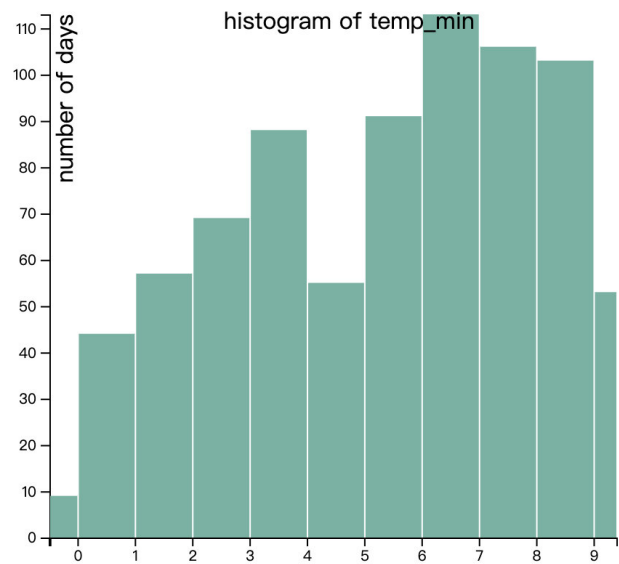
</script>



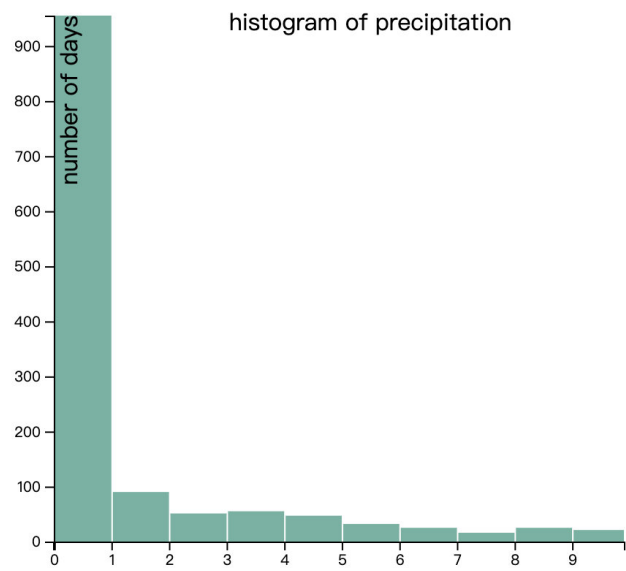
wind



temp_max



temp_min ▾



precipitation ▾

Part II Task

Problem 1

2.1.1

Load the csv file to a JavaScript array. Display the top 5 rows of the JavaScript array in the webpage as a table in HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="https://d3js.org/d3.v4.js"></script>
</head>
<body>

</body>
</html>

<script>
  d3.csv("https://raw.githubusercontent.com/plotly/datasets/master/auto-mpg.csv", function(data){
    var newData = [];
    for (var index in data) {
      var d = {};
      d["mpg"] = (data[index].mpg);
      d["cylinders"] = (data[index].cylinders);
      d["displacement"] = (data[index].displacement);
      d["horsepower"] = (data[index].horsepower);
      d["weight"] = (data[index].weight);
      d["acceleration"] = (data[index].acceleration);
      d["model-year"] = (data[index]["model-year"]);
      newData.push(d);
    }

    function tabulate(data, cols) {
      var table = d3.select('body').append('table');
      var thead = table.append('thead');
      var tbody = table.append('tbody');

      thead.append('tr')
        .selectAll('th')
        .data(cols)
        .enter()
        .append('th')
        .text(function (col){return col;});

      var rows = tbody.selectAll('tr')
        .data(data)
        .enter()
```

```

        .append( 'tr' );

    var cells = rows.selectAll( 'td' )
        .data( function ( row ) {
            return cols.map( function ( col ) {
                return { col: col, value: row[ col ] };
            } );
        } )
        .enter()
        .append( 'td' )
        .text( function ( d ) { return d.value; } );

    return table;
}
tabulate( newData.slice( 0, 5 ), [ "mpg", "cylinders", "displacement", "horsepower", "weight",
    "acceleration", "model-year" ] );
} )
</script>

```

	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

Create a new array to get the count of the number of cars of each model year and display the thus obtained Javascript array as a table in HTML .

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="https://d3js.org/d3.v4.js"></script>
</head>
<body></body>
</html>

<script>

    d3.csv( "https://raw.githubusercontent.com/plotly/datasets/master/auto-mpg.csv", function( data ) {
        var modelYears = data.map( function ( d ) { return d[ "model-year" ]; } );
    } );

```

```

    var modelYearCounts = modelYears.reduce((pre, cur) => {
        if(cur in pre) pre[cur]++;
        else pre[cur] = 1;
        return pre;
    }, {});
    var newData = [];
    for (var index in Object.keys(modelYearCounts)) {
        var d = {};
        d["model-year"] = (Object.keys(modelYearCounts)[index]);
        d["count"] = modelYearCounts[d["model-year"]];
        newData.push(d);
    }

function tabulate(data, cols) {
    var table = d3.select('body').append('table');
    var thead = table.append('thead');
    var tbody = table.append('tbody');

    thead.append('tr')
        .selectAll('th')
        .data(cols)
        .enter()
        .append('th')
        .text(function (col){return col;});

    var rows = tbody.selectAll('tr')
        .data(data)
        .enter()
        .append('tr');

    var cells = rows.selectAll('td')
        .data(function (row) {
            return cols.map(function(col){
                return {col: col, value: row[col]};
            });
        })
        .enter()
        .append('td')
        .text(function(d){return d.value;});

    return table;
}
tabulate(newData, ["model-year", "count"]);
})
</script>

```

model-year 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

Create a new array to get the total number of cylinders for each model year and display the thus obtained Javascript array as a table in HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="https://d3js.org/d3.v4.js"></script>
</head>
<body></body>
</html>

<script>
  d3.csv("https://raw.githubusercontent.com/plotly/datasets/master/auto-mpg.csv", function(data){
    var modelYears = data.map(function(d){return d["model-year"];});
    var modelYearCounts = modelYears.reduce((pre, cur) => {
      if(cur in pre) pre[cur]++;
      else pre[cur] = 1;
      return pre;
    }, {});
    var newData = [];
    for (var index in Object.keys(modelYearCounts)) {
      var d = {};
      d["model-year"] = (Object.keys(modelYearCounts)[index]);
      var modelYearTotal = data.filter(function(data){return data["model-year"] == d["model-
year"]});
      var modelYearTotalCylinders = 0;
      for (var i in modelYearTotal) {
        modelYearTotalCylinders += parseInt(modelYearTotal[i].cylinders);
      }
    }
  });
}
```



```

        d["total num of cylinders"] = modelYearTotalCylinders;
        newData.push(d);
    }
    console.log(newData);

function tabulate(data, cols) {
    var table = d3.select('body').append('table');
    var thead = table.append('thead');
    var tbody = table.append('tbody');

    thead.append('tr')
        .selectAll('th')
        .data(cols)
        .enter()
        .append('th')
        .text(function (col){return col;});

    var rows = tbody.selectAll('tr')
        .data(data)
        .enter()
        .append('tr');

    var cells = rows.selectAll('td')
        .data(function (row) {
            return cols.map(function(col){
                return {col: col, value: row[col]};
            });
        })
        .enter()
        .append('td')
        .text(function(d){return d.value;});

    return table;
}
tabulate(newData, ["model-year", "total num of cylinders"]);
})
</script>

```

model-year total num of cylinders

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

Create a new array to get the count of the number of cars of each acceleration and display the thus obtained Javascript array as a table in HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="https://d3js.org/d3.v4.js"></script>
</head>
<body></body>
</html>

<script>
  d3.csv("https://raw.githubusercontent.com/plotly/datasets/master/auto-mpg.csv", function(data){
    var accelerations = data.map(function(d){return d["acceleration"];});
    var accelerationCounts = accelerations.reduce((pre, cur) => {
      if(cur in pre) pre[cur]++;
      else pre[cur] = 1;
      return pre;
    }, {});
    var newData = [];
    for (var index in Object.keys(accelerationCounts)) {
      var d = {};
      d["acceleration"] = (Object.keys(accelerationCounts)[index]);
      d["count"] = accelerationCounts[d["acceleration"]];
      newData.push(d);
    }

    function tabulate(data, cols) {
      var table = d3.select('body').append('table');
      var thead = table.append('thead');
      var tbody = table.append('tbody');

      thead.append('tr')
        .selectAll('th')
        .data(cols)
        .enter()
        .append('th')
        .text(function (col){return col;});

      var rows = tbody.selectAll('tr')
        .data(data)
        .enter()
        .append('tr');

      var cells = rows.selectAll('td')
        .data(function (row) {
          return cols.map(function(col){
            return {col: col, value: row[col]};
          });
        });
    }
  })
}
```

```

        .enter()
        .append('td')
        .text(function(d){return d.value;});

    return table;
}
tabulate(newData, ["acceleration", "count"]);
})
</script>

```

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

Problem 2

Now that you have preprocessed the data and have new required arrays you will have to visualize the data in a single HTML webpage using various plots in D3.

2.2.1

Using the array created in 2.1.2, plot a pie chart using D3 along with the table from 2.1.2 to get the distribution of the model years amongst the cars, properly label the sectors with percentage and annotate each sector with corresponding model year.

2.2.2

Using the array created in 2.1.3, plot a line graph using D3 along with the table from 2.1.3 to see the total number of cylinders for each model year, label the x and y axis and sort the x axis in decreasing order.

2.2.3

Using the array created in 2.1.4, plot a histogram using D3 along with the table from 2.1.4 for acceleration with bins equal to 10 label x and y axis appropriately.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="https://d3js.org/d3.v4.js"></script>
</head>
<body>
  <div id = "pieChart">
    <h1>Problem 2.2.1</h1>
  </div>
  <div id = "lineGraph">
    <h1>Problem 2.2.2</h1>
  </div>
  <div id = "histogram">
    <h1>Problem 2.2.3</h1>
  </div>
</body>
</html>
<script>
  var svg1 = d3.select("#pieChart")
    .append("svg")
    .attr("width", 960)
    .attr("height", 500);
  var width = +svg1.attr("width");
  var height = +svg1.attr("height");
  var radius = Math.min(width, height) / 2;
  var g = svg1.append("g").attr("transform", "translate(" + width / 2 + "," + height / 2 + ")");

  // setting color scale for the pie chart
  var color = d3.scaleOrdinal(["#98abc5", "#8a89a6", "#7b6888", "#6b486b", "#a05d56", "#d0743c",
    "#ff8c00"]);

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

  var path = d3.arc()
    .outerRadius(radius - 60)
    .innerRadius(0);

  var label = d3.arc()
    .outerRadius(radius)
    .innerRadius(radius - 60);

  d3.csv("https://raw.githubusercontent.com/plotly/datasets/master/auto-mpg.csv", function (data)
  {
    var modelYears = data.map(function(d){return d["model-year"];});
  });
</script>
```

```

var modelYearCounts = modelYears.reduce((pre, cur) => {
  if(cur in pre) pre[cur]++;
  else pre[cur] = 1;
  return pre;
}, {});

var newData = [];
for (var index in Object.keys(modelYearCounts)) {
  var d = {};
  d["model-year"] = (Object.keys(modelYearCounts)[index]);
  d["count"] = modelYearCounts[d["model-year"]];
  d["percentage"] = (modelYearCounts[d["model-year"]] / data.length);
  newData.push(d);
}

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

arc.append("path")
  .attr("d", path)
  .attr("fill", function(d) {return color(d.data["model-year"]);});

arc.append("text")
  .attr("transform", function(d) {return "translate(" + label.centroid(d) + ")";})
  .attr("dy", "0.35em")
  .text(function(d){return d.data["model-year"];});

arc.append("text")
  .attr("transform", function(d) {return "translate(" + label.centroid(d) + ")";})
  .attr("dy", "1em")
  .attr("dx", "1.5em")
  .text(function(d){return d.data["percentage"];});

svg1.append("text")
  .attr("x", +svg1.attr("width") / 2 - 100)
  .attr("y", +svg1.attr("height"))
  .text("Pie chart of model-year variable")
});
</script>

<script>
// set the dimensions and margins of the graph
// You can change these values these are just sample values given
var margin = {top: 20, right: 50, bottom: 50, left: 60},
    width = 820 - margin.left - margin.right,
    height = 400 - margin.top - margin.bottom;

// append the svg object to the body of the page
var svg2 = d3.select("#lineGraph")
  .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 + ")");

// uncomment the function and complete this function to plot required graphs

```

```

d3.csv("https://raw.githubusercontent.com/plotly/datasets/master/auto-mpg.csv", function(data)
{
    var modelYearList = data.map(function(d) { return d["model-year"]; });
    var modelYearCountList = modelYearList.reduce((pre, cur) => {
        if(cur in pre) pre[cur]++;
        else pre[cur] = 1;
        return pre;
    }, {});
    var newData = [];
    var len = Object.keys(modelYearCountList).length;
    console.log(len);
    for(var index=len-1; index >= 0; index--) {
        var d = {};
        d["model-year"] = (Object.keys(modelYearCountList)[index]);
        var modelYearTotal = data.filter(function(data) { return data["model-year"] ==
d["model-year"]; });
        var modelYearTotalCylinders = 0;
        for(var i in modelYearTotal) {
            modelYearTotalCylinders += parseInt(modelYearTotal[i].cylinders);
        }
        d["total num of cylinders"] = modelYearTotalCylinders;
        newData.push(d);
    }
    console.log(newData);

    var x = d3.scaleLinear()
        .domain(d3.extent(newData, function(d) { return d["model-year"]; }).reverse())
        .range([0, width]);

    svg2.append("g")
        .attr("transform", "translate(0, "+ height + ")")
        .call(d3.axisBottom(x));

    var y = d3.scaleLinear()
        .domain([0, d3.max(newData, function(d) { return d["total num of cylinders"]; })])
        .range([height, 0]);

    svg2.append("g")
        .call(d3.axisLeft(y));
    svg2.append("path")
        .datum(newData)
        .attr("fill", "none")
        .attr("stroke", "steelblue")
        .attr("stroke-width", 1.5)
        .attr("d", d3.line()
            .x(function(d) { return x(d["model-year"]); })
            .y(function(d) { return y(d["total num of cylinders"]); })
        )
    svg2.append("text") // text label for the x axis
        .attr("x", width / 2 - 30)
        .attr("y", height + 30)
        .text("model year");
    svg2.append("g")
        .call(y)
        .append("text")
        .attr("transform", "rotate(-90)")
        .attr("dy", "1em")
        .attr("text-anchor", "end")
        .text("total num of cylinders");
}

```

```

    svg2.append("text")          // text label for the title
    .attr("x", width/2)
    .attr("y", 0)
    .style("text-anchor", "middle")
    .text("Line graph about total num of cylinders with model year");
  });
</script>

<script>
  // set the dimensions and margins of the graph
  // You can change these values these are just sample values given
  var margin = {top: 20, right: 50, bottom: 50, left: 60},
      width = 460 - margin.left - margin.right,
      height = 400 - margin.top - margin.bottom;

  // append the svg object to the body of the page
  var svg = d3.select("#histogram")
    .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 + ")");

  // getting the data in csv format
  d3.csv("https://raw.githubusercontent.com/plotly/datasets/master/auto-mpg.csv", function (data)
  {
    var minX = d3.min(data, function(d) {return +d.acceleration;})
    var maxX = d3.max(data, function(d) {return +d.acceleration;})

    // X axis
    var x = d3.scaleLinear()
      .domain([minX,maxX])
      .range([0, width]);
    svg.append("g")
      .attr("transform", "translate(0," + height + ")")
      .call(d3.axisBottom(x));

    //setting the parameters for the histogram
    var binNum = 10
    var histogram = d3.histogram()
      .value(function(d) {return d.acceleration;})
      .domain(x.domain())
      .thresholds(x.ticks(binNum)); // change this parameter to 25
    var bins = histogram(data);

    // Y axis
    var y = d3.scaleLinear()
      .range([height, 0]);
    y.domain([0, d3.max(bins, function(d) {return d.length;})]);
    svg.append("g")
      .call(d3.axisLeft(y));

    // setting rectangle
    svg.selectAll("rect")
      .data(bins)
      .enter()
      .append("rect")
      .attr("x", 1)

```

```

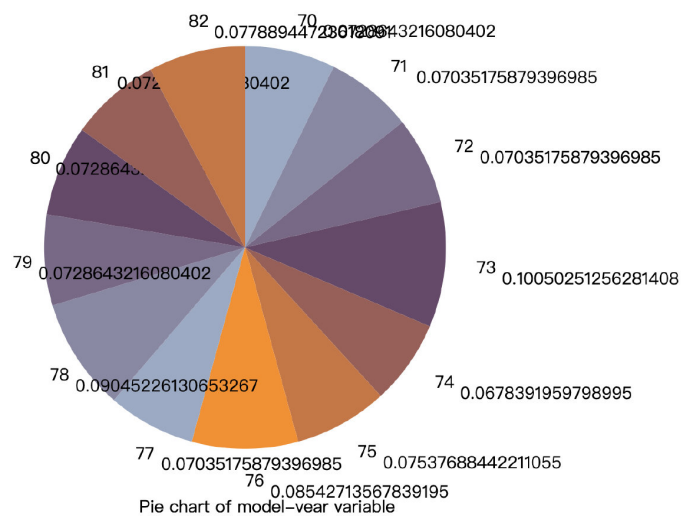
        .attr("transform", function(d) {return "translate(" + x(d.x0) + "," + y(d.length) +
    ");});

    .attr("width", function(d) {return x(d.x1) - x(d.x0) - 1;});
    .attr("height", function(d) {return height - y(d.length);});
    .style("fill", "#69b3a2");
svg.append("g")
    .append('text')
    .attr("x", 80)
    .attr("y", 0)
    .text("histogram for acceleration (bin = " + binNum + ")")
svg.append("g")
    .append('text')
    .attr("x", width / 2 - 20)
    .attr("y", 380 - margin.top)
    .text("acceleration")
svg.append("g")
    .call(y)
    .append("text")
    .attr("transform", "rotate(-90)")
    .attr("dy", "1em")
    .attr("text-anchor", "end")
    .text("count")

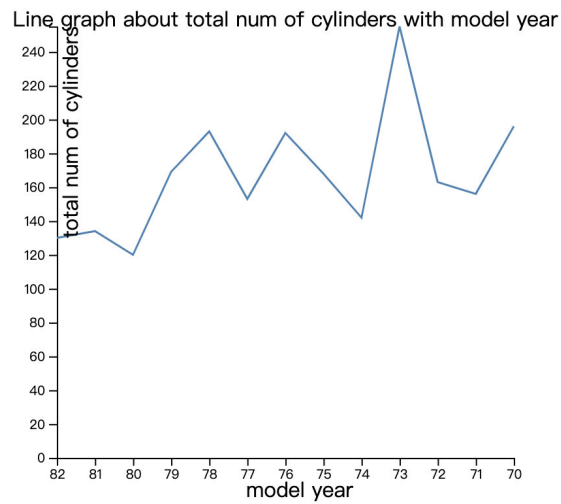
});
</script>

```

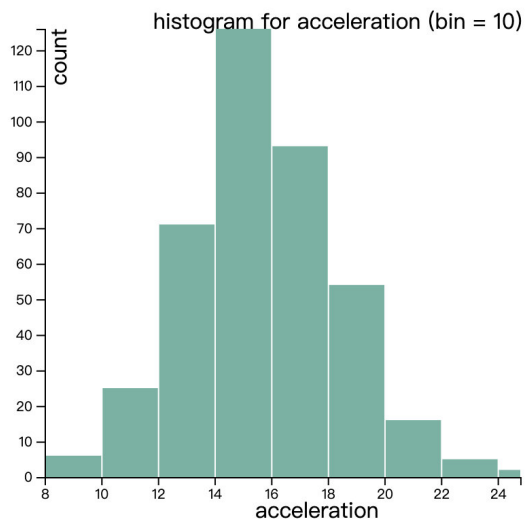
Problem 2.2.1



Problem 2.2.2



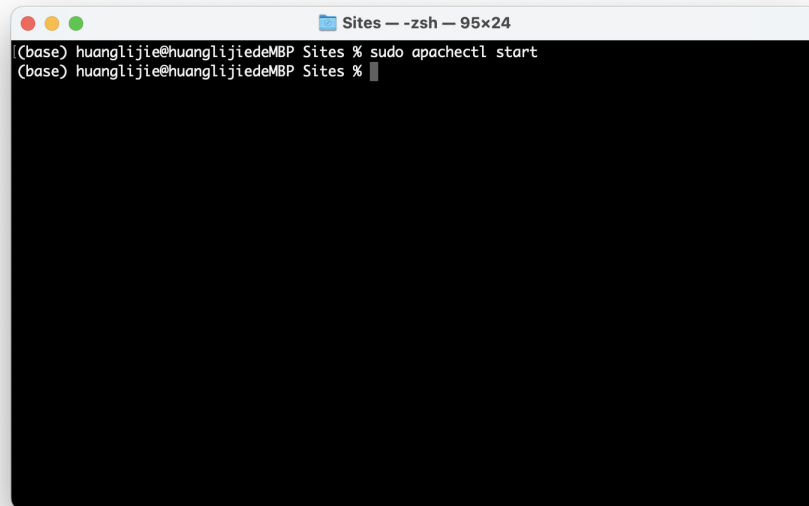
Problem 2.2.3



Problem 3

2.3.1

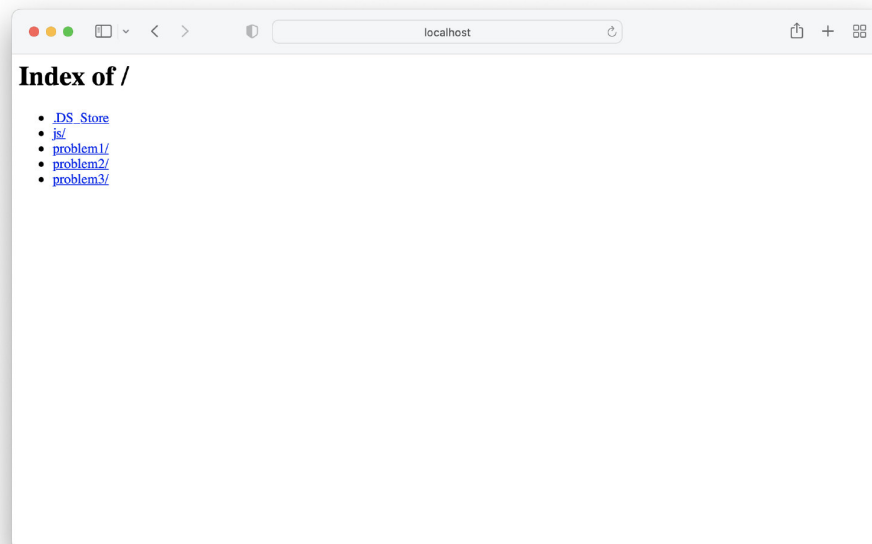
Follow the tutorial for installing Apache HTTP server



```
Sites --zsh-- 95x24
(base) huanglijie@huanglijiedeMBP Sites % sudo apachectl start
(base) huanglijie@huanglijiedeMBP Sites %
```

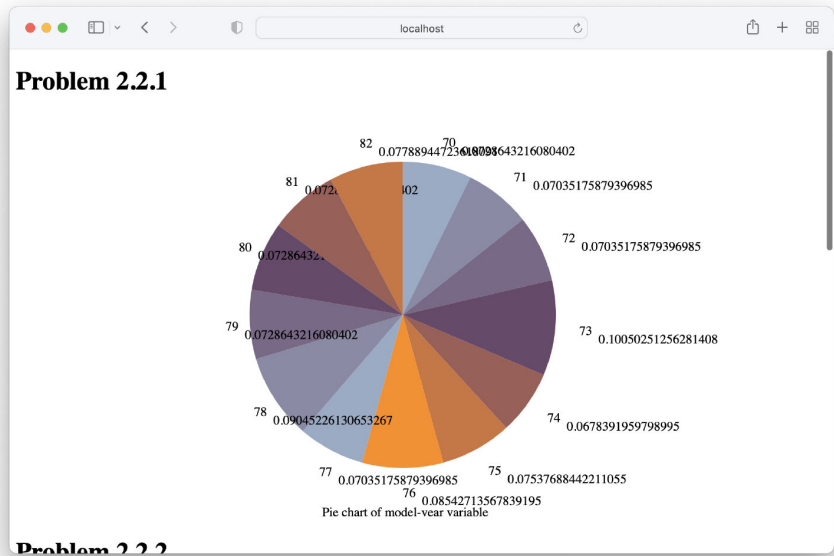
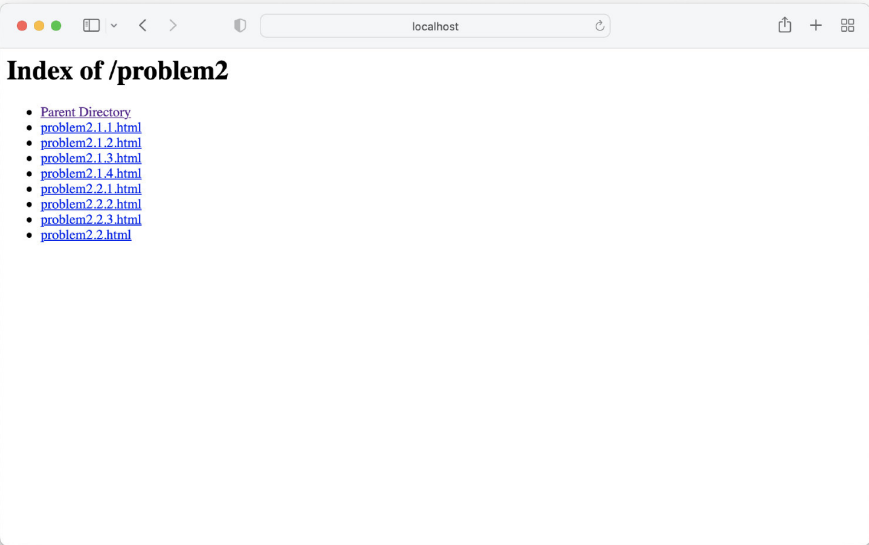
2.3.2

Create a Virtual host



2.3.3

Host your webpage (webpage from Part II problem 2) on the virtual host.



Part III Task

Problem 1

3.1.1

Plot a Network graph that shows character co-occurrence in Les Misérables.

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>

.node {
  stroke: #fff;
  stroke-width: 1.5px;
}

.link {
  stroke: #999;
  stroke-opacity: .6;
}

</style>
<body>
<script src="http://d3js.org/d3.v3.js"></script>
<script>

  var width = 960,
      height = 500;

  var color = d3.scale.category20();

  var force = d3.layout.force()
    .charge(-120)
    .linkDistance(30)
    .size([width, height]);

  var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height);

  d3.csv("https://gist.githubusercontent.com/timelyportfolio/5049980/raw/66a239b4aa325c05c7a19bd96bf093632591e809/les_mis.csv", function (error, data) {
    //set up graph in same style as original example but empty
    graph = { "nodes": [], "links": [] };

    //loop through each link record from the csv data
    //add to the nodes each source and target; we'll reduce to unique values later
    //add to the links each source, target record with the value (if desired, multiple value
    fields can be added)
    data.forEach(function (d) {
      graph.nodes.push({ "name": d.source, "group": +d.groupsource });
```

```

graph.nodes.push({ "name": d.target, "group": +d.grouptarget });

graph.links.push({ "source": d.source, "target": d.target, "value": +d.value });
});

//use this as temporary holding while we manipulate graph.nodes
//this will contain a map object containing an object for each node
//within each node object there will be a child object for each instance that node appear
//however, using rollup we can eliminate this duplication
var nodesmap = d3.nest()
    .key(function (d) { return d.name; })
    .rollup(function (d) { return { "name": d[0].name, "group": d[0].group
}; })

    .map(graph.nodes);

//thanks Mike Bostock https://groups.google.com/d/msg/d3-js/pl297cFtIQk/Eso4q\_eBulIJ
//this handy little function returns only the distinct / unique nodes
graph.nodes = d3.keys(d3.nest()
    .key(function (d) { return d.name; })
    .map(graph.nodes));

//it appears d3 with force layout wants a numeric source and target
//so loop through each link replacing the text with its index from node
graph.links.forEach(function (d, i) {
    graph.links[i].source = graph.nodes.indexOf(graph.links[i].source);
    graph.links[i].target = graph.nodes.indexOf(graph.links[i].target);
});

//this is not in the least bit pretty
//will get graph.nodes in its final useable form
//loop through each unique node and replace with an object with same numeric key and
name/group as properties
//that will come from the nodesmap that we defined earlier
graph.nodes.forEach(function (d,i) { graph.nodes[i]={ "name": nodesmap[d].name, "group":
nodesmap[d].group }; })

force
    .nodes(graph.nodes)
    .links(graph.links)
    .start();

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

var node = svg.selectAll(".node")
    .data(graph.nodes)
    .enter().append("circle")
    .attr("class", "node")
    .attr("r", 5)
    .style("fill", function (d) { return color(d.group); })
    .call(force.drag);

node.append("title")
    .text(function (d) { return d.name; });

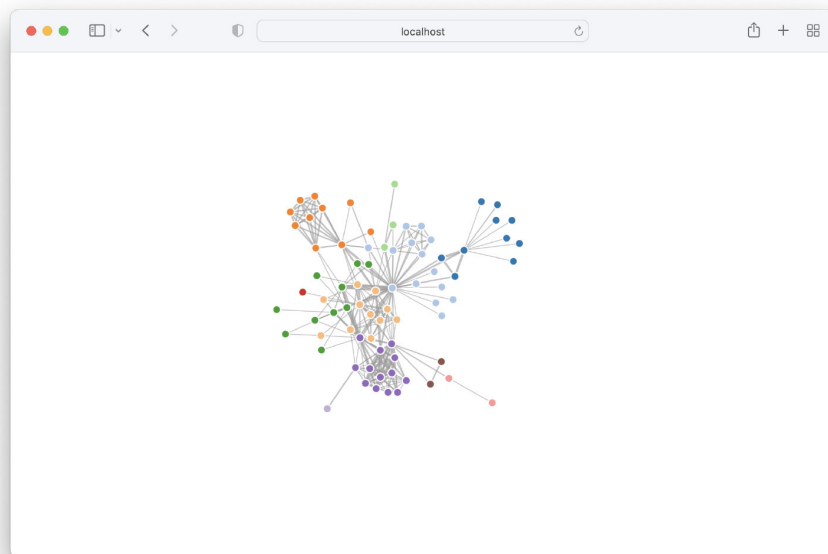
```

```

    force.on("tick", function () {
      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; });
    });
  });
</script>
</body>
</html>

```



3.1.2

Add labels to each node. (i.e. each node should show the name of the character)

```

<!DOCTYPE html>
<meta charset="utf-8">
<style>

.node {
  stroke: #fff;
  stroke-width: 1.5px;
}

.link {
  stroke: #999;
  stroke-opacity: .6;
}

</style>
<body>

```

```

<script src="http://d3js.org/d3.v3.js"></script>
<script>

    var width = 960,
        height = 500;

    var color = d3.scale.category20();

    var force = d3.layout.force()
        .charge(-120)
        .linkDistance(30)
        .size([width, height]);

    var svg = d3.select("body").append("svg")
        .attr("width", width)
        .attr("height", height);

    d3.csv("https://gist.githubusercontent.com/timelyportfolio/5049980/raw/66a239b4aa325c05c7a19bd96bf093632591e809/les_mis.csv", function (error, data) {
        //set up graph in same style as original example but empty
        graph = { "nodes": [], "links": [] };

        //loop through each link record from the csv data
        //add to the nodes each source and target; we'll reduce to unique values later
        //add to the links each source, target record with the value (if desired, multiple value
        fields can be added)
        data.forEach(function (d) {
            graph.nodes.push({ "name": d.source, "group": +d.groupsource });
            graph.nodes.push({ "name": d.target, "group": +d.grouptarget });

            graph.links.push({ "source": d.source, "target": d.target, "value": +d.value });
        });

        //use this as temporary holding while we manipulate graph.nodes
        //this will contain a map object containing an object for each node
        //within each node object there will be a child object for each instance that node appear
        //however, using rollup we can eliminate this duplication
        var nodesmap = d3.nest()
            .key(function (d) { return d.name; })
            .rollup(function (d) { return { "name": d[0].name, "group": d[0].group
}; })
            .map(graph.nodes);

        //thanks Mike Bostock https://groups.google.com/d/msg/d3-js/pl297cFtIQk/Eso4q_eBulIJ
        //this handy little function returns only the distinct / unique nodes
        graph.nodes = d3.keys(d3.nest()
            .key(function (d) { return d.name; })
            .map(graph.nodes));

        //it appears d3 with force layout wants a numeric source and target
        //so loop through each link replacing the text with its index from node
        graph.links.forEach(function (d, i) {
            graph.links[i].source = graph.nodes.indexOf(graph.links[i].source);
            graph.links[i].target = graph.nodes.indexOf(graph.links[i].target);
        });

        //this is not in the least bit pretty

```

```

    //will get graph.nodes in its final useable form
    //loop through each unique node and replace with an object with same numeric key and
name/group as properties
    //that will come from the nodesmap that we defined earlier
    graph.nodes.forEach(function (d,i) { graph.nodes[i]={ "name": nodesmap[d].name, "group":
nodesmap[d].group }; })

    force
        .nodes(graph.nodes)
        .links(graph.links)
        .start();

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

    var node = svg.selectAll(".node")
        .data(graph.nodes)
        .enter().append("circle")
        .attr("class", "node")
        .attr("r", 5)
        .style("fill", function (d) { return color(d.group); })
        .call(force.drag);

    var label = svg.selectAll(null)
        .data(graph.nodes)
        .enter()
        .append("text")
        .text(function(d){return d.name;})
        .style("text-anchor", "middle")
        .style("font-size", 8);

    node.append("title")
        .text(function (d) { return d.name; });

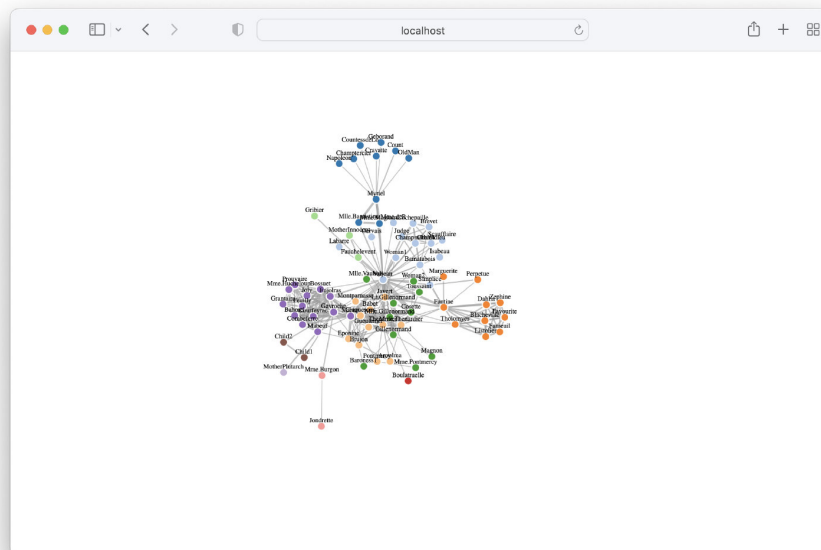
    force.on("tick", function () {
        link.attr("x1", function (d) { return d.source.x; })
            .attr("y1", function (d) { return d.source.y; })
            .attr("x2", function (d) { return d.target.x; })
            .attr("y2", function (d) { return d.target.y; });

        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 - 5;})
    });
});

</script>
</body>
</html>

```

3.1.3

It can be difficult to observe micro and macro features simultaneously with complex graphs. If you zoom in for detail, the graph is too big to view in its entirety. Use fisheye distortion (magnifies the local region around the mouse, while leaving the larger graph unaffected for context) for the above network graph. (you can use D3's fisheye plugin).

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>

.node {
  stroke: #fff;
  stroke-width: 1.5px;
}

.link {
  stroke: #999;
  stroke-opacity: .6;
}

</style>
<body>
<script src="http://d3js.org/d3.v3.js"></script>
<script src="fisheye.js"></script>
<script>

  var width = 960,
      height = 500;

  var color = d3.scale.category20();

  var force = d3.layout.force()
    .charge(-120)
```

```

        .linkDistance(30)
        .size([width, height]);

var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height);

var fisheye = d3.fisheye.circular()
    .radius(200)
    .distortion(2);

d3.csv("https://gist.githubusercontent.com/timelyportfolio/5049980/raw/66a239b4aa325c05c7a19bd96bf093632591e809/les_mis.csv", function (error, data) {
    //set up graph in same style as original example but empty
    graph = { "nodes": [], "links": [] };

    //loop through each link record from the csv data
    //add to the nodes each source and target; we'll reduce to unique values later
    //add to the links each source, target record with the value (if desired, multiple value
    fields can be added)
    data.forEach(function (d) {
        graph.nodes.push({ "name": d.source, "group": +d.groupsource });
        graph.nodes.push({ "name": d.target, "group": +d.grouptarget });

        graph.links.push({ "source": d.source, "target": d.target, "value": +d.value });
    });

    //use this as temporary holding while we manipulate graph.nodes
    //this will contain a map object containing an object for each node
    //within each node object there will be a child object for each instance that node appear
    //however, using rollup we can eliminate this duplication
    var nodesmap = d3.nest()
        .key(function (d) { return d.name; })
        .rollup(function (d) { return { "name": d[0].name, "group": d[0].group
}; })
        .map(graph.nodes);

    //thanks Mike Bostock https://groups.google.com/d/msg/d3-js/pl297cFtIQk/Eso4q_eBulIJ
    //this handy little function returns only the distinct / unique nodes
    graph.nodes = d3.keys(d3.nest()
        .key(function (d) { return d.name; })
        .map(graph.nodes));

    //it appears d3 with force layout wants a numeric source and target
    //so loop through each link replacing the text with its index from node
    graph.links.forEach(function (d, i) {
        graph.links[i].source = graph.nodes.indexOf(graph.links[i].source);
        graph.links[i].target = graph.nodes.indexOf(graph.links[i].target);
    });

    //this is not in the least bit pretty
    //will get graph.nodes in its final useable form
    //loop through each unique node and replace with an object with same numeric key and
    name/group as properties
    //that will come from the nodesmap that we defined earlier
    graph.nodes.forEach(function (d,i) { graph.nodes[i]={ "name": nodesmap[d].name, "group":
nodesmap[d].group }; })

```

```

force
  .nodes(graph.nodes)
  .links(graph.links)
  .start();

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

var node = svg.selectAll(".node")
  .data(graph.nodes)
  .enter().append("circle")
  .attr("class", "node")
  .attr("r", 5)
  .style("fill", function (d) { return color(d.group); })
  .call(force.drag);

var label = svg.selectAll(null)
  .data(graph.nodes)
  .enter()
  .append("text")
  .text(function(d){return d.name;})
  .style("text-anchor", "middle")
  .style("font-size", 8);

svg.on("mousemove", function() {
  fisheye.focus(d3.mouse(this));
  node.each(function(d) { d.fisheye = fisheye(d); })
    .attr("cx", function(d) { return d.fisheye.x; })
    .attr("cy", function(d) { return d.fisheye.y; })
    .attr("r", function(d) { return d.fisheye.z * 4.5; });

  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; });

  label.attr("x", function (d) { return d.fisheye.x; })
    .attr("y", function (d) { return d.fisheye.y - 5; });
});

node.append("title")
  .text(function (d) { return d.name; });

force.on("tick", function () {
  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; });

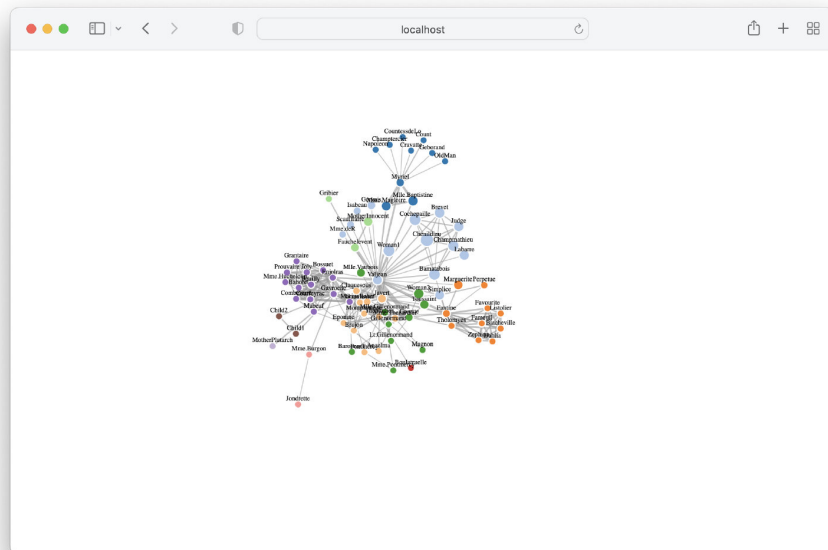
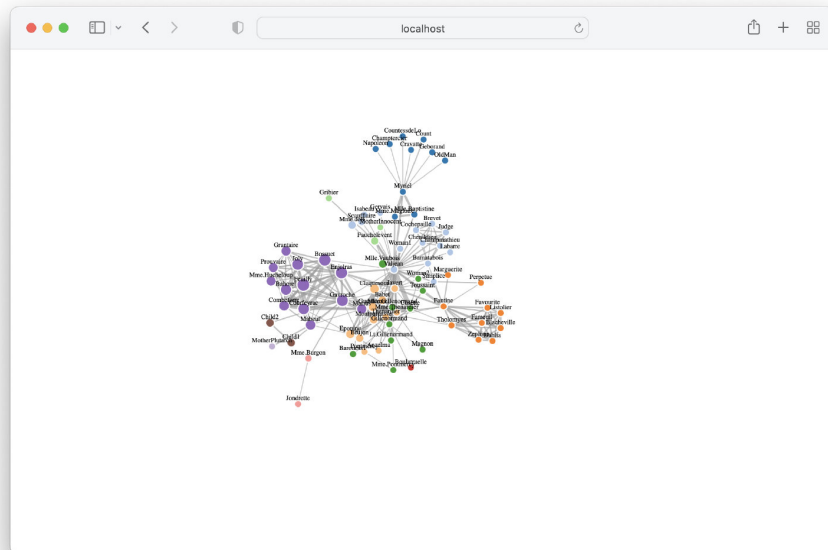
  label.attr("x", function(d){return d.x;})
    .attr("y", function(d){return d.y - 5;})

```

```

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

```



Problem 2

3.2.1

Add a

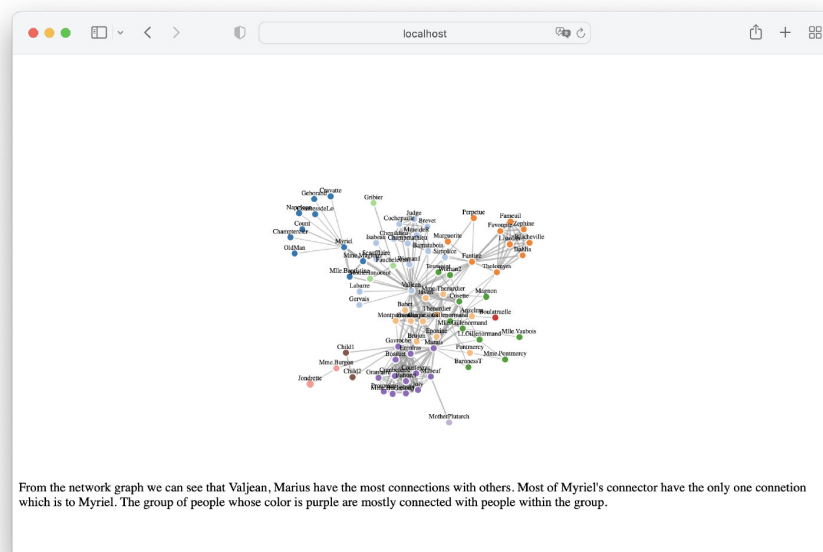
tag to your HTML file and infer what you see from the graph.

<p>

From the network graph we can see that Valjean, Marius have the most connections with others. Most of Myriel's connector have the only one connection which is to Myriel.

The group of people whose color is purple are mostly connected with people within the group.

</p>



3.2.2

Host your webpage (from 3.1.3) on the virtual host.

