LAB EXERCISE 2

Learning Outcomes:

1. To create line chart with SVG and D3.js
2. To create multiple line charts

Lesson 1: Creating Line Chart

Create new HTML file in VS Code. Type ! and press Tab key to get skeleton of HTML code.

Step 1: Load d3.js in <head> section of HTML

<head>

```
<!-- Load d3.js -->
<script src="https://d3js.org/d3.v6.js"></script>
```

</head>

Step 2: Create a div to put the graph in <body> section of HTML

<body>

```
<div id="my_dataviz"></div>
```

…

Start writing the scripts in <body> section of HTML. There are few code fragments within the <script> compound.

1. Dimension and margin of the graph
2. SVG
3. Parsing the data
4. Adding X axis and Y Axis
5. Adding bars

Step 3: Set the dimensions and margins of the graph

```
<script>

const margin = {top: 10, right: 30, bottom: 30, left: 60},
    width = 460 - margin.left - margin.right,
    height = 400 - margin.top - margin.bottom;…

</script>
```

Step 3: Drawing graph with SVG

Append the SVG object to the body of the page. Add the following code to the script. The coordinate is based on the margin set in Step 3.

```
const 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})`);
"translate(" + margin.left + "," + margin.top + ")");
```

Padding can be applied with a **<g>** element translated by the desired value. The SVG <g> element is used to group SVG shapes together. Once grouped you can transform the whole group of shapes as if it was a single shape.

For more details about <g> attribute → https://jenkov.com/tutorials/svg/g-element.html

Step 4: Parse the data from github and format the date variable

```
d3.csv("https://raw.githubusercontent.com/holtzy/data_to_viz/master/Example_dataset/3_TwoNumOrdered_comma.csv",function(d) {
    return { date : d3.timeParse("%Y-%m-%d")(d.date), value : d.value }
  }).then(

// Now we can use this dataset:
  function(data) {
```

NOTE: the following code fragments will be within the { }

Step 5: Add X axis

```
const 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));
```

Step 6: Add Y axis

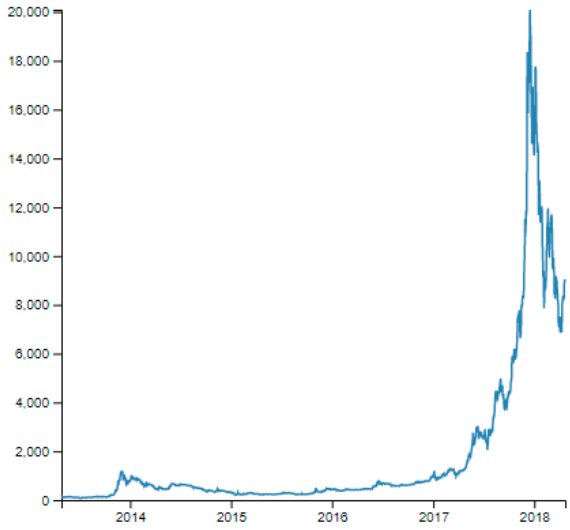
```
const y = d3.scaleLinear()
      .domain([0, d3.max(data, function(d) { return +d.value; })])
      .range([ height, 0 ]);
    svg.append("g")
      .call(d3.axisLeft(y));
```

Step 7: Add the line and close the scripts

```
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) })
        )
})

</script>
```

NOTE: Color code reference → https://htmlcolorcodes.com/

Sample Output



Lesson 2: Creating Group Line Chart

Coding Style #1

```
year,sex,name,n,prop
1880,F,Helen,636,0.00651612638826278
1880,F,Amanda,241,0.00246916109995492
1880,F,Betty,117,0.00119872136387853
1880,F,Dorothy,112,0.00114749395516577
1880,F,Linda,27,0.000276628007048891
1880,F,Deborah,12,0.000122945780910618
1880,F,Jessica,7,7.17183721978607e-05
1881,F,Helen,612,0.00619088564058469
1881,F,Amanda,263,0.0026604622932578
1881,F,Betty,112,0.00113297253553184
1881,F,Dorothy,109,0.00110262505690152
1881,F,Linda,38,0.000384401395984017
1881,F,Deborah,14,0.00014162156694148
1881,F,Jessica,7,7.081078347074e-05
1882,F,Helen,838,0.00724311990042871
1882,F,Amanda,288,0.00248928225694925
1882,F,Betty,123,0.00106313096390541
1882,F,Dorothy,115,0.000993984234545706
1882,F,Linda,36,0.000311160282118656
```

View full dataset here:
https://raw.githubusercontent.com/holtzy/data_to_viz/master/Example_dataset/5_OneCatSevNumOrdered.csv

Refer to Lesson 1 for Step 1 to step 3.

Step 4: Parse the data from internet

```
d3.csv("https://raw.githubusercontent.com/holtzy/data_to_viz/master/Example_dataset/5_OneCatSevNumOrdered.csv").then( function(data) {
```

Step 5: Group the data

```
const sumstat = d3.group(data, d => d.name);
```

Step 6: Add X axis

```
const x = d3.scaleLinear()
    .domain(d3.extent(data, function(d) { return d.year; }))
    .range([ 0, width ]);
  svg.append("g")
    .attr("transform", `translate(0, ${height})`)
    .call(d3.axisBottom(x).ticks(5));
```

Step 7: Add Y axis

```
onst y = d3.scaleLinear()
    .domain([0, d3.max(data, function(d) { return +d.n; })])
    .range([ height, 0 ]);
  svg.append("g")
    .call(d3.axisLeft(y));
```

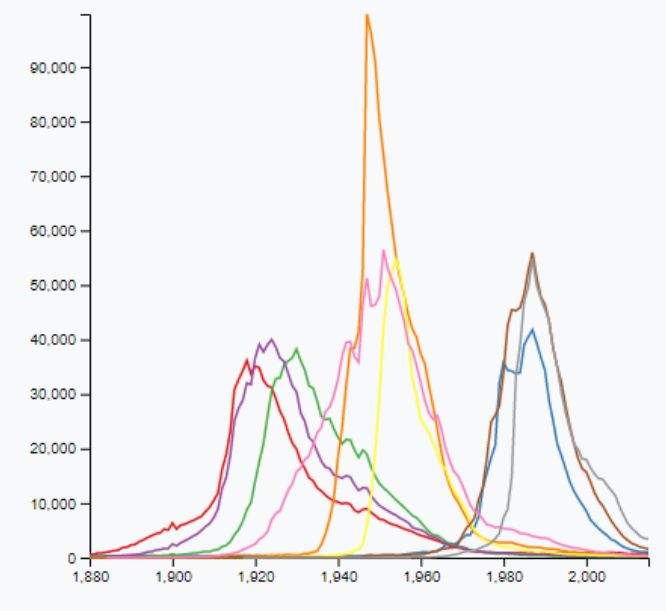
Step 8: Add color palette, one color per line

```
const color = d3.scaleOrdinal()
      .range(['#e41a1c','#377eb8','#4daf4a','#984ea3','#ff7f00','#ffff33','#a
      65628','#f781bf','#999999'])
```

Step 10: Draw the line

```
vg.selectAll(".line")
      .data(sumstat)
      .join("path")
        .attr("fill", "none")
        .attr("stroke", function(d){ return color(d[0]) })
        .attr("stroke-width", 1.5)
        .attr("d", function(d){
          return d3.line()
            .x(function(d) { return x(d.year); })
            .y(function(d) { return y(+d.n); })
            (d[1])
        })
})

</script>
```

Sample Output



Formatting Year to remove commas.

Go back to **Step 5**. Add the following code before adding X axis.

```
var parseTime = d3.timeParse("%Y");
```

```
// format the data
data.forEach(function(d) {
d.year = parseTime(d.year);
d.n = +d.n;
});
```

Modify **Step 6**

You should use **scaleTime** for x axis, NOT scaleLinear:

```
const x = d3.scaleTime()
          …
```

Save and view your chart on live server.

References:

https://d3-graph-gallery.com/

Coding Style #2

```
date,close,open
1-May-12,68.13,34.12
30-Apr-12,63.98,45.56
27-Apr-12,67.00,67.89
26-Apr-12,89.70,78.54
25-Apr-12,99.00,89.23
24-Apr-12,130.28,99.23
23-Apr-12,166.70,101.34
20-Apr-12,234.98,122.34
19-Apr-12,345.44,134.56
18-Apr-12,443.34,160.45
17-Apr-12,543.70,180.34
16-Apr-12,580.13,210.23
13-Apr-12,605.23,223.45
12-Apr-12,622.77,201.56
11-Apr-12,626.20,212.67
10-Apr-12,628.44,310.45
9-Apr-12,636.23,350.45
5-Apr-12,633.68,410.23
4-Apr-12,624.31,430.56
3-Apr-12,629.32,460.34
2-Apr-12,618.63,510.34
30-Mar-12,599.55,534.23
29-Mar-12,609.86,578.23
28-Mar-12,617.62,590.12
27-Mar-12,614.48,560.34
26-Mar-12,606.98,580.12
```

Step 1: Save the data as stockdata.csv

Step 2: Create CSS Style and D3 in <head> section

```
<head>
<style> /* set the CSS */
```

```
  .line {
    fill: none;
    stroke: steelblue;
    stroke-width: 2px;
  }

  </style>

  <script src="https://d3js.org/d3.v6.min.js"></script>
  </head>
```

Step 3: Set the margin in <body><script> section

```
// set the dimensions and margins of the graph
var margin = {top: 20, right: 20, bottom: 30, left: 50},
    width = 960 - margin.left - margin.right,
    height = 500 - margin.top - margin.bottom;
```

Step 4: Parse the time and set the ranges

```
// parse the date / time
var parseTime = d3.timeParse("%d-%b-%y");

// set the ranges
var x = d3.scaleTime().range([0, width]);
var y = d3.scaleLinear().range([height, 0]);
```

Step 5: Define the lines

```
// define the 1st line
var valueline = d3.line()
    .x(function(d) { return x(d.date); })
    .y(function(d) { return y(d.close); });

// define the 2nd line
var valueline2 = d3.line()
    .x(function(d) { return x(d.date); })
    .y(function(d) { return y(d.open); });
```

Step 6: Append the svg

```
var svg = d3.select("body").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 + ")");
```

Step 7: Get and format the data

```
// Get the data
d3.csv("stockdata.csv").then(function(data) {
```

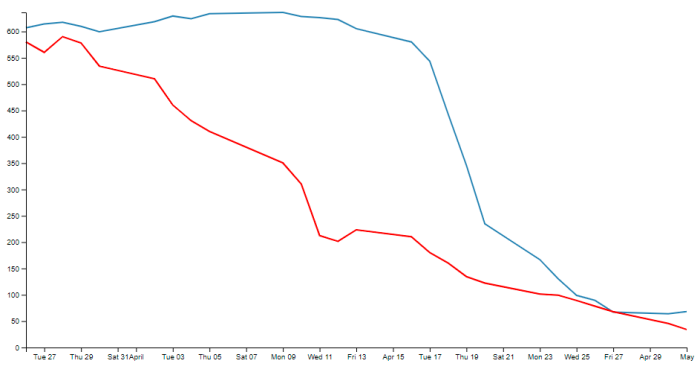
```
  // format the data
  data.forEach(function(d) {
      d.date = parseTime(d.date);
      d.close = +d.close;
      d.open = +d.open;
  });

  // Scale the range of the data
  x.domain(d3.extent(data, function(d) { return d.date; }));
  y.domain([0, d3.max(data, function(d) {
            return Math.max(d.close, d.open); })]);
```

Step 8: Add the lines

```
// Add the valueline path.
  svg.append("path")
      .data([data])
      .attr("class", "line")
      .attr("d", valueline);

  // Add the valueline2 path.
  svg.append("path")
      .data([data])
      .attr("class", "line")
      .style("stroke", "red")
      .attr("d", valueline2);
```

Step 9: Add the X axis and Y axis

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

  // Add the Y Axis
  svg.append("g")
      .call(d3.axisLeft(y));

});
</script>
```



SAMPLE OUTPUT

Reference: https://bl.ocks.org/

LAB TASK

1. Evaluate the group line chart in Lesson 2 based on design principles for data visualization. Compare with small multiple line chart in https://d3-graph-gallery.com/graph/line_smallmultiple.html
2. Refer to the following dataset > https://gist.githubusercontent.com/benlcollins/2d7fb36cd0c295c00882/raw/6776a4cf60432d1e325d09dc6cddcd5d62405d26/stock_data.csv
   a. Create group line chart and small multiple line chart.
   b. Consider appropriate pre-attentive attributes such as label, color etc.
   c. Write a summary of insights based on the graphs created in (a).

Submission item: HTML file.

Submission platform: ULearn

Deadline: 1st October 2023