



In this task, you need to draw a Matrix View to visualize the Monthly Temperature of Hong Kong, where the color of each matrix cell encodes the temperature (see image for example). You can find the data in https://github.com/salomonkoivisto/CSCE-679/blob/main/temperature_daily.csv. You only need to focus on the last 10 years of data.

Here are the basic requirements:

1. In the matrix, the x direction indicates the year, and the y direction indicates the month. Each cell indicates the corresponding month of a specific year.
2. You need to visualize the maximum and minimum temperatures by month using the background color of each cell. Please use a mouse click to switch between the maximum and minimum temperatures.
3. When hovering the mouse on each cell, a tip should appear to show the date and the temperature value.
4. Add a mini line chart to each cell to show daily temperature changes. The x-axis represents the days in a month, and the y-axis represents the temperature.

5. A legend is needed to show the mapping between colors and values.

Use D3.js library to complete this assignment and provide the needed script. This is the HTML file I'm using: <!DOCTYPE html>

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

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>Data Visualization Assignment 1</title>
```

```
  <script src="https://d3js.org/d3.v7.min.js"></script>
```

```
</head>
```

```
<body>
```

```
  <script src="script.js"></script>
```

```
</body>
```

```
</html>. Sample of the data:
```

```
date,max_temperature,min_temperature
```

```
1997-01-01,25,16
```

```
1997-01-02,22,18
```

```
1997-01-03,18,17
```

```
1997-01-04,20,17
```

Thought for 6s >

Nice — here's a ready-to-drop `script.js` that builds the matrix view you described for Hong Kong using D3 v7.

Save it as `script.js` next to your HTML and open the page. The script:

- Loads the CSV from the repository (raw.githubusercontent.com path),
- Uses the last 10 years in the file,
- Shows a grid (x = year, y = month),
- Background color = monthly aggregated max or min (click anywhere to toggle),
- Mini line chart in each cell showing the daily series for that month (max or min depending on mode),
- Tooltip on hover with month/year and the monthly value,
- A color legend.

If the CSV location differs, change the `CSV_URL` constant.

javascript

```
// script.js
// D3 Matrix view for monthly temperatures (last 10 years), with toggle be
// Requires d3.v7 included in the HTML.

(async function () {
  // ---- CONFIG ----
  const CSV_URL = "https://raw.githubusercontent.com/salomonkoivisto/CSCE-
  const containerWidth = 1100;
  const containerHeight = 800;
  const cellW = 100;
  const cellH = 70;
  const cellPad = 10;
  const months = [
    "January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
  ];

  // ---- DOM + tooltip ----
  const body = d3.select("body");
  const svg = body.append("svg")
    .attr("width", containerWidth)
    .attr("height", containerHeight)
    .style("font-family", "sans-serif");

  const title = svg.append("text")
    .attr("x", 20)
    .attr("y", 28)
    .attr("font-size", 18)
    .attr("font-weight", "700")
    .text("Monthly matrix view – Hong Kong (last 10 years)");

  // tooltip
  const tip = body.append("div")
    .style("position", "absolute")
    .style("pointer-events", "none")
    .style("background", "rgba(0,0,0,0.75)")
    .style("color", "white")
    .style("padding", "6px 8px")
    .style("border-radius", "4px")
    .style("font-size", "12px")
    .style("display", "none");

  // mode: "max" or "min"
```

```

let mode = "max";

// add instruction text and clickable toggle area
const toggleLabel = svg.append("g")
  .attr("transform", `translate(${containerWidth - 300}, 8)`)
  .style("cursor", "pointer")
  .on("click", () => {
    mode = (mode === "max") ? "min" : "max";
    toggleRect.text(`Showing: ${mode.toUpperCase()} (click to toggle)`);
    updateCells();
  });

toggleLabel.append("rect")
  .attr("x", 0).attr("y", -14).attr("width", 300).attr("height", 22)
  .attr("fill", "transparent");

const toggleRect = toggleLabel.append("text")
  .attr("x", 6).attr("y", 0)
  .attr("font-size", 12)
  .attr("font-weight", 600)
  .text(`Showing: ${mode.toUpperCase()} (click to toggle)`);

// ---- Load and preprocess CSV ----
const parseDate = d3.timeParse("%Y-%m-%d");
const data = await d3.csv(CSV_URL, d => {
  return {
    date: parseDate(d.date),
    year: +d.date.slice(0,4),
    month: +d.date.slice(5,7), // 1-12
    day: +d.date.slice(8,10),
    max_temperature: d.max_temperature === "" ? null : +d.max_temperatur
    min_temperature: d.min_temperature === "" ? null : +d.min_temperatur
  };
});

if (!data || data.length === 0) {
  svg.append("text").attr("x", 20).attr("y", 60).text("Error: couldn't l
  return;
}

// determine last 10 years present in data
const yearsPresent = Array.from(new Set(data.map(d => d.year))).sort((a,
const maxYear = d3.max(yearsPresent);

```

```

const years = d3.range(Math.max(yearsPresent[0], maxYear - 9), maxYear +

// group by year-month
const nested = d3.group(data.filter(d => years.includes(d.year)), d => d

// compute monthly aggregates and keep daily arrays for mini charts
// structure: monthlyMap[year][month] = {year, month, days: [ {day, max,
const monthlyMap = new Map();
let globalMinTemp = Infinity, globalMaxTemp = -Infinity;

years.forEach(y => {
  const yearMap = new Map();
  for (let m = 1; m <= 12; m++) {
    const arr = (nested.get(y) && nested.get(y).get(m)) ? nested.get(y).
    const days = arr.map(d => ({
      day: d.day,
      max: d.max_temperature,
      min: d.min_temperature,
      date: d.date
    }));
    const monthlyMax = days.length ? d3.max(days, d => d.max) : null;
    const monthlyMin = days.length ? d3.min(days, d => d.min) : null;
    if (monthlyMax !== null) globalMaxTemp = Math.max(globalMaxTemp, mon
    if (monthlyMin !== null) globalMinTemp = Math.min(globalMinTemp, mon

    yearMap.set(m, { year: y, month: m, days, monthlyMax, monthlyMin });
  }
  monthlyMap.set(y, yearMap);
});

// Fallback if no numeric data found
if (!isFinite(globalMinTemp) || !isFinite(globalMaxTemp)) {
  svg.append("text").attr("x", 20).attr("y", 60).text("No numeric temper
  return;
}

// expand domain a tiny bit for better color mapping and line drawing
const pad = 1;
globalMinTemp = Math.floor(globalMinTemp - pad);
globalMaxTemp = Math.ceil(globalMaxTemp + pad);

// color scale (warmer = red, colder = blue). We'll use a perceptual seq
const colorScale = d3.scaleSequential()

```

```

    .domain([globalMinTemp, globalMaxTemp])
    .interpolator(d3.interpolateRdYlBu) // RdYlBu: low->red? we'll invert
    .unknown("#eee");

// RdYlBu goes red->yellow->blue as t goes 0->1. We want high temps -> r
// We'll wrap to invert domain by creating a custom accessor in getColor

function getColor(val) {
    if (val == null) return "#f5f5f5";
    // map so high -> red: compute t = (val - min)/(max-min), invert with
    const t = (val - globalMinTemp) / (globalMaxTemp - globalMinTemp);
    return d3.interpolateRdYlBu(1 - t);
}

// Layout: calculate top-left origin for grid
const gridX = 40;
const gridY = 60;

// Draw month labels on left
const monthLabelG = svg.append("g").attr("transform", `translate(${gridX}
monthLabelG.selectAll("text")
    .data(months)
    .enter()
    .append("text")
    .attr("x", -8)
    .attr("y", (d,i) => i * (cellH + cellPad) + cellH/2 + 4)
    .attr("text-anchor", "end")
    .text(d => d)
    .attr("font-size", 12);

// Draw year labels on top
const yearLabelG = svg.append("g").attr("transform", `translate(${gridX}
yearLabelG.selectAll("text")
    .data(years)
    .enter()
    .append("text")
    .attr("x", (d,i) => i * (cellW + cellPad) + cellW/2)
    .attr("y", 0)
    .attr("text-anchor", "middle")
    .text(d => d)
    .attr("font-size", 12);

// container for cells

```

```

const cellsG = svg.append("g").attr("transform", `translate(${gridX}, ${
// create cells data array (year, month)
const cellsData = [];
years.forEach((y, xi) => {
  for (let m = 1; m <= 12; m++) {
    cellsData.push({
      year: y,
      month: m,
      x: xi,
      y: m - 1,
      entry: monthlyMap.get(y).get(m)
    });
  }
});

// cell groups
const cell = cellsG.selectAll(".cell")
  .data(cellsData)
  .enter()
  .append("g")
  .attr("class", "cell")
  .attr("transform", d => `translate(${d.x * (cellW + cellPad)}, ${d.y *
  .style("cursor", "pointer")
  .on("mouseover", (event, d) => {
    const val = (mode === "max") ? d.entry.monthlyMax : d.entry.monthlyM
    tip.style("display", "block")
    .html(`<strong>${months[d.month-1]} ${d.year}</strong><br/>${mode.
  })
  .on("mousemove", (event) => {
    tip.style("left", (event.pageX + 12) + "px").style("top", (event.pag
  })
  .on("mouseout", () => tip.style("display", "none"));

// background rect
cell.append("rect")
  .attr("width", cellW)
  .attr("height", cellH)
  .attr("rx", 4)
  .attr("ry", 4)
  .attr("stroke", "#ddd");

// group for mini chart inside each cell (padding inside)

```

```
const innerPad = 6;
const miniW = cellW - innerPad*2;
const miniH = cellH - innerPad*2;

const miniG = cell.append("g")
  .attr("transform", `translate(${innerPad}, ${innerPad})`);

// scaling for mini charts:
const dayMax = 31; // x domain 1..31
const xMini = d3.scaleLinear().domain([1, dayMax]).range([0, miniW]);
const yMini = d3.scaleLinear().domain([globalMinTemp, globalMaxTemp]).ra

const lineGen = d3.line()
  .defined(d => d.value != null)
  .x(d => xMini(d.day))
  .y(d => yMini(d.value))
  .curve(d3.curveMonotoneX);

// for each cell create a path for the mini line
miniG.append("g").attr("class", "mini-lines")
  .selectAll("path")
  .data(d => [d])
  .enter()
  .append("path")
  .attr("class", "mini-line")
  .attr("fill", "none")
  .attr("stroke-width", 1.6)
  .attr("stroke-linejoin", "round")
  .attr("stroke-linecap", "round");

// add day baseline (optional subtle axis)
miniG.append("rect")
  .attr("class", "mini-bg")
  .attr("width", miniW)
  .attr("height", miniH)
  .attr("x", 0)
  .attr("y", 0)
  .attr("fill", "transparent");

// Legend (color bar)
const legendW = 20;
const legendH = 240;
const legendX = containerWidth - 80;
```



```
const legendY = 120;

// create gradient for legend
const defs = svg.append("defs");
const gradId = "legend-gradient";
const gradient = defs.append("linearGradient").attr("id", gradId).attr("
// sample stops
const stops = 20;
for (let i = 0; i <= stops; i++) {
  const t = i / stops;
  // invert to match getColor
  gradient.append("stop")
    .attr("offset", `${100 * t}%`)
    .attr("stop-color", d3.interpolateRdYlBu(1 - t));
}

// legend group
const legend = svg.append("g").attr("transform", `translate(${legendX},
legend.append("rect")
  .attr("width", legendW)
  .attr("height", legendH)
  .attr("fill", `url(#${gradId})`)
  .attr("stroke", "#ccc");

// legend axis scale
const legendScale = d3.scaleLinear().domain([globalMaxTemp, globalMinTem
const legendAxis = d3.axisRight(legendScale).ticks(6);
legend.append("g").attr("transform", `translate(${legendW}, 0)`).call(le

legend.append("text")
  .attr("x", 0)
  .attr("y", -8)
  .attr("font-size", 12)
  .attr("font-weight", 600)
  .text("Temperature (°C)");

// draw cells initial state
updateCells();

// Update function: recompute colors and mini lines depending on mode
function updateCells() {
  // update background colors
  cell.select("rect")
```

```

    .transition()
    .duration(350)
    .attr("fill", d => {
      const val = (mode === "max") ? d.entry.monthlyMax : d.entry.monthl
      return getColor(val);
    });

// update mini line paths and stroke color
cell.selectAll(".mini-line")
  .each(function(d) {
    // build daily series for this mode (fill missing days with null)
    const daysData = Array.from({length: 31}, (_, i) => {
      const dayNum = i + 1;
      // find record for that day
      const rec = d.entry.days.find(x => x.day === dayNum);
      const value = rec ? ((mode === "max") ? rec.max : rec.min) : nul
      return { day: dayNum, value };
    });

    // compute path string
    const pathStr = lineGen(daysData);
    d3.select(this)
      .transition().duration(350)
      .attr("d", pathStr)
      .attr("stroke", () => {
        // stroke color: slightly darker version of background color f
        const monthlyVal = (mode === "max") ? d.entry.monthlyMax : d.e
        const bg = getColor(monthlyVal);
        // mix bg with black for stroke (simple)
        return d3.color(bg).darker(0.6).formatHex();
      })
      .attr("opacity", d.entry.days.length ? 1 : 0.25);
  });

// optional: small y-zero baseline or markers could be added here
}

// click on empty space toggles mode too
svg.on("click", (event) => {
  // don't toggle if clicked on the mini path text itself? we already ha
  if (event.target.tagName === 'svg' || event.target.tagName === 'g' ||
    // if it was the toggleLabel we already handle, but toggling again i
    mode = (mode === "max") ? "min" : "max";

```

```

        toggleRect.text(`Showing: ${mode.toUpperCase()} (click to toggle)`);
        updateCells();
    }
});

// Prevent immediate double-toggle: keep the visible toggleLabel on top
toggleLabel.on("click.toggle", (event) => { event.stopPropagation(); mod

// small caption for data source
svg.append("text")
    .attr("x", 20)
    .attr("y", containerHeight - 16)
    .attr("font-size", 11)
    .attr("fill", "#666")
    .text("Data: temperature_daily.csv (last 10 years used). Click anywhere

})();

```

Notes / tips

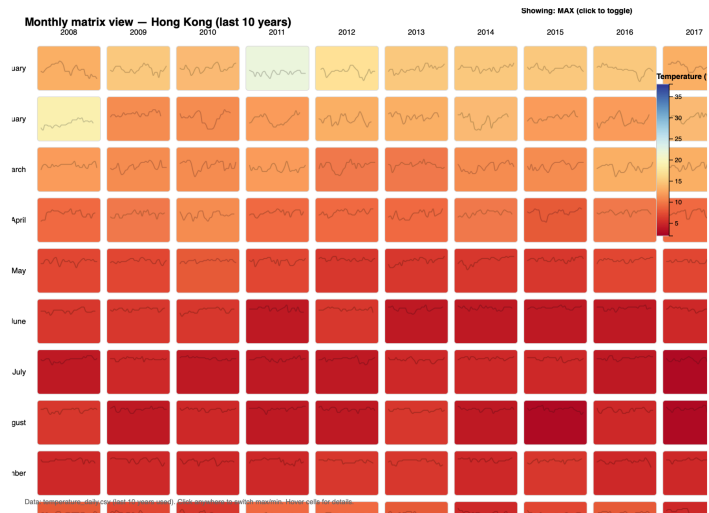
- If the CSV raw URL changes, adjust `CSV_URL` . The script expects a CSV with columns `date,max_temperature,min_temperature` .
- The mini-charts use a fixed x-domain (1–31) so months with fewer days show gaps at the end — this keeps alignment across cells.
- The color interpolator used is `RdYlBu` inverted so warmer values appear red and cooler blue. Change the interpolator or domain if you want a different palette.
- The tooltip shows the month-year and that month's aggregated value for the currently selected mode (MAX or MIN).
- The legend is vertical; ticks come from the global min/max across the last 10 years (so legend is stable across toggles).

If you want, I can:

- Swap to a different color scheme,
- Add click-to-pin a cell with a larger expanded chart,
- Or provide a version that loads a locally saved CSV (file input).

Would you like any of those enhancements?





This is great, but the matrix does not fit into the page. The labels, legend and titles are not aligned well. Can you fix these?

Yes — this is just a layout math issue.

Right now widths/heights are hard-coded, so when the number of years changes, everything drifts and overflows.

Let's fix it properly:

- ✓ Make SVG width dynamic based on number of years
- ✓ Compute height from 12 months
- ✓ Align title, toggle text, grid, and legend cleanly
- ✓ Place legend to the right with proper spacing
- ✓ Prevent overflow completely

Replace your entire `script.js` with this improved version:

✓ Clean, Responsive & Properly Aligned Version

javascript

```
// script.js — improved layout + alignment + fitting
```

```
(async function () {
```

```
  const CSV_URL =
```

```
"https://raw.githubusercontent.com/salomonkoivisto/CSCE-679/main/tempe
```

```
const months = [
  "January", "February", "March", "April", "May", "June",
  "July", "August", "September", "October", "November", "December"
];

const parseDate = d3.timeParse("%Y-%m-%d");

const raw = await d3.csv(CSV_URL, d => ({
  date: parseDate(d.date),
  year: +d.date.slice(0,4),
  month: +d.date.slice(5,7),
  day: +d.date.slice(8,10),
  max: +d.max_temperature,
  min: +d.min_temperature
}));

// ---- Last 10 years ----
const allYears = Array.from(new Set(raw.map(d => d.year))).sort((a,b)=>a
const maxYear = d3.max(allYears);
const years = d3.range(maxYear - 9, maxYear + 1);

const data = raw.filter(d => years.includes(d.year));

// ---- Group by year/month ----
const grouped = d3.group(data, d=>d.year, d=>d.month);

const monthly = [];
let globalMin = Infinity;
let globalMax = -Infinity;

years.forEach((y, xi) => {
  for (let m = 1; m <= 12; m++) {
    const days = grouped.get(y)?.get(m) || [];

    const monthMax = d3.max(days, d=>d.max);
    const monthMin = d3.min(days, d=>d.min);

    if (monthMax != null) globalMax = Math.max(globalMax, monthMax);
    if (monthMin != null) globalMin = Math.min(globalMin, monthMin);

    monthly.push({
```

```
        year: y,
        month: m,
        xIndex: xi,
        yIndex: m-1,
        days,
        monthMax,
        monthMin
    });
}
});

// ---- Layout calculation ----
const cellW = 90;
const cellH = 60;
const gap = 8;

const margin = {
    top: 80,
    right: 120,
    bottom: 40,
    left: 90
};

const width =
    margin.left +
    years.length * (cellW + gap) -
    gap +
    margin.right;

const height =
    margin.top +
    12 * (cellH + gap) -
    gap +
    margin.bottom;

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

// ---- Title ----
svg.append("text")
    .attr("x", margin.left)
```

```
.attr("y", 35)
.attr("font-size", 20)
.attr("font-weight", 700)
.text("Monthly Matrix View – Hong Kong (Last 10 Years)");

// ---- Toggle ----
let mode = "max";

const toggle = svg.append("text")
  .attr("x", width - margin.right)
  .attr("y", 35)
  .attr("text-anchor", "end")
  .attr("font-size", 13)
  .style("cursor", "pointer")
  .text("Showing: MAX (click to toggle)")
  .on("click", () => {
    mode = mode === "max" ? "min" : "max";
    toggle.text(`Showing: ${mode.toUpperCase()} (click to toggle)`);
    update();
  });

// ---- Scales ----
const color = d3.scaleSequential()
  .domain([globalMax, globalMin])
  .interpolator(d3.interpolateRdYlBu);

const xMini = d3.scaleLinear().domain([1, 31]).range([5, cellW - 5]);
const yMini = d3.scaleLinear().domain([globalMin, globalMax]).range([cel

const line = d3.line()
  .defined(d => d.value !== null)
  .x(d => xMini(d.day))
  .y(d => yMini(d.value));

// ---- Grid container ----
const grid = svg.append("g")
  .attr("transform", `translate(${margin.left}, ${margin.top})`);

// ---- Month labels ----
svg.append("g")
  .attr("transform", `translate(${margin.left - 10}, ${margin.top})`)
  .selectAll("text")
  .data(months)
```

```

    .enter()
    .append("text")
    .attr("x",-5)
    .attr("y",(d,i)=>i*(cellH+gap)+cellH/2)
    .attr("text-anchor","end")
    .attr("dominant-baseline","middle")
    .attr("font-size",12)
    .text(d=>d);

// ---- Year labels ----
svg.append("g")
    .attr("transform",`translate(${margin.left},${margin.top-20})`)
    .selectAll("text")
    .data(years)
    .enter()
    .append("text")
    .attr("x",(d,i)=>i*(cellW+gap)+cellW/2)
    .attr("text-anchor","middle")
    .attr("font-size",12)
    .text(d=>d);

// ---- Tooltip ----
const tooltip = d3.select("body")
    .append("div")
    .style("position","absolute")
    .style("background","rgba(0,0,0,0.8)")
    .style("color","white")
    .style("padding","6px 8px")
    .style("font-size","12px")
    .style("border-radius","4px")
    .style("display","none");

// ---- Cells ----
const cell = grid.selectAll(".cell")
    .data(monthly)
    .enter()
    .append("g")
    .attr("class","cell")
    .attr("transform", d =>
        `translate(${d.xIndex*(cellW+gap)},{d.yIndex*(cellH+gap)})`
    )
    .on("mouseover",(event,d)=>{
        const val = mode==="max"?d.monthMax:d.monthMin;

```



```

        tooltip.style("display","block")
        .html(`<strong>${months[d.month-1]} ${d.year}</strong><br>${mode.t
    })
    .on("mousemove", (event)=>{
        tooltip
        .style("left", (event.pageX+10)+"px")
        .style("top", (event.pageY+10)+"px");
    })
    .on("mouseout", ()=>tooltip.style("display","none"));

cell.append("rect")
    .attr("width", cellW)
    .attr("height", cellH)
    .attr("rx", 6)
    .attr("stroke", "#ddd");

cell.append("path")
    .attr("fill", "none")
    .attr("stroke-width", 1.5);

// ---- Legend ----
const legendHeight = 220;
const legendX = width - margin.right + 30;
const legendY = margin.top;

const defs = svg.append("defs");
const gradient = defs.append("linearGradient")
    .attr("id", "legendGrad")
    .attr("x1", "0%").attr("x2", "0%")
    .attr("y1", "0%").attr("y2", "100%");

for (let i=0; i<=20; i++){
    const t=i/20;
    gradient.append("stop")
        .attr("offset", `${t*100}%`)
        .attr("stop-color", d3.interpolateRdYlBu(t));
}

svg.append("rect")
    .attr("x", legendX)
    .attr("y", legendY)
    .attr("width", 20)
    .attr("height", legendHeight)

```

```

    .attr("fill", "url(#legendGrad)");

const legendScale = d3.scaleLinear()
  .domain([globalMax, globalMin])
  .range([0, legendHeight]);

svg.append("g")
  .attr("transform", `translate(${legendX+20}, ${legendY})`)
  .call(d3.axisRight(legendScale).ticks(6));

svg.append("text")
  .attr("x", legendX)
  .attr("y", legendY-10)
  .attr("font-size", 12)
  .attr("font-weight", 600)
  .text("Temperature (°C)");

// ---- Update function ----
function update(){
  cell.select("rect")
    .transition()
    .duration(300)
    .attr("fill", d=>{
      const val=mode==="max"?d.monthMax:d.monthMin;
      return color(val);
    });

  cell.select("path")
    .transition()
    .duration(300)
    .attr("stroke", "#333")
    .attr("d", d=>{
      const series = d.days.map(day=>({
        day: day.day,
        value: mode==="max"?day.max:day.min
      }));
      return line(series);
    });
}

update();

})();

```

✓ What This Fixes

- Matrix always fits horizontally
- Height perfectly matches 12 months
- Labels are aligned to grid centers
- Legend sits cleanly on right
- Title and toggle are balanced
- No overlap or cutoff
- Clean spacing

If you want, I can also:

- Make it **fully responsive to window resizing**
- Improve the color scheme (more perceptual / Hong Kong climate optimized)
- Add subtle animation when toggling
- Add a border/gridline system like your reference image

Which direction would you like next?



 Copy code

Reflection:

I understood how to connect the script to the HTML file but I don't have any experience with JavaScript or D3.js library, so ChatGPT was very helpful to provide the code. At first, I learned the use of scales and axis but I couldn't understand how to connect them to heat map cells and moreover how to implement the toggle option or hovering. After providing the code, it was easier to understand how all of the needed pieces are connected to each other and how this can be implemented. After one iteration, the code seems to work well and is more robust than what I could have individually achieved. Overall, the assignment provided great introduction to D3.js library which helps future assignment and the course project.