Assignment 2

VIKRAMADITYA REDDY VARKALA

Z1973679

Data Visualization(CSCI 627)

2023-09-27

1. Table (25 pts)

```
data=d3.csv("https://gist.githubusercontent.com/dakoop/9e67814b6073ebbf6e7f55e31b5781ce/raw
/5dad34b939d0d0789570064a75b145cc255f2811/newspaper-circulation.csv")
```

a. Data Processing (10 pts)

```
filterData = data.filter(d => d.Year !== "1940" && d.Weekday !== "--")
//filters out rows with 1940 and --
```

```
processData = filterData.map(d => ({ Year: parseInt(d.Year), Weekday:
parseInt(d.Weekday),Sunday: parseInt(d.Sunday), Average:
        (parseInt(d.Weekday) + parseInt(d.Sunday)) / 2}));
```

b. Table (15 pts)

Year	Weekday	Sunday	Average
1945	48384000	39860000	44122000
1946	50928000	43665000	47296500
1947	51673000	45151000	48412000
1948	52285000	46308000	49296500
1949	52846000	46399000	49622500
1950	53829000	46582000	50205500
1951	54018000	46279000	50148500
1952	53951000	46210000	50080500
1953	54472000	45949000	50210500
1954	55072000	46176000	50624000
1955	56147000	46448000	51297500
1956	57102000	47162000	52132000
1957	57805000	47044000	52424500

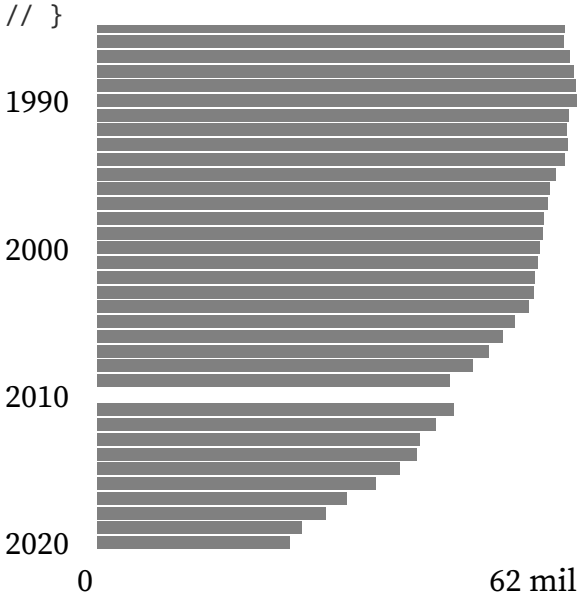
1957	57805000	47044000	52424500
1958	57418000	46955000	52186500
1959	58300000	47848000	53074000
1960	58882000	47699000	53290500
1961	59261000	48216000	53738500
1962	59849000	48888000	54368500
1963	58905000	46830000	52867500
1964	60412000	48383000	54397500
1965	60358000	48600000	54479000
1966	61397000	49282000	55339500
1967	61561000	49224000	55392500
1968	62535000	49693000	56114000
1969	62060000	49675000	55867500
1970	62108000	49217000	55662500
1971	62231000	49665000	55948000
1972	62510000	50001000	56255500
1973	63147000	51717000	57432000
1974	61877000	51679000	56778000
1975	60655000	51096000	55875500
1976	60977000	51565000	56271000
1977	61495000	52429000	56962000
1978	61990000	53990000	57990000
1979	62223000	54380000	58301500
1980	62202000	54676000	58439000
1981	61431000	55180000	58305500
1982	62487000	56261000	59374000
1983	62645000	56747000	59696000
1984	63340000	57574000	60457000
1985	62766000	58826000	60796000
1986	62502000	58925000	60713500
1987	62826000	60112000	61469000
1988	62695000	61474000	62084500
1989	62649000	62008000	62328500
1990	62328000	62635000	62481500
1991	60687000	62068000	61377500
1992	60164000	62160000	61162000
1993	59812000	62566000	61189000
1994	59305000	62295000	60800000
1995	58193000	61229000	59711000
1996	56983000	60798000	58890500
1997	56728000	60486000	58607000
1998	56182000	60066000	58124000
1999	55979000	59894000	57936500
2000	55773000	59421000	57597000

2001	55578000	59090000	57334000
2002	55186000	58780000	56983000
2003	55185000	58495000	56840000
2004	54626000	57754000	56190000
2005	53345000	55270000	54307500
2006	52329000	53179000	52754000
2007	50742000	51246000	50994000
2008	48597000	49115000	48856000
2009	45653000	46164000	45908500
2011	44421000	48510000	46465500
2012	43433000	44821000	44127000
2013	40712000	43292000	42002000
2014	40420000	42751000	41585500
2015	37711860	40955458	39333659
2016	34657199	37801888	36229543.5
2017	30948419	33971695	32460057
2018	28554137	30817351	29685744
2019	25952584	27389866	26671225
2020	24299333	25785036	25042184.5

```
table = { const table = document.createElement("table");
          const header = table.insertRow();

          //inserts header elements into row
          header.insertCell().textContent = "Year";
          header.insertCell().textContent = "Weekday";
          header.insertCell().textContent = "Sunday";
          header.insertCell().textContent = "Average";

          //fill the processData into cells
          processData.forEach(d => { const fill = table.insertRow();
                                     fill.insertCell().textContent = d.Year;
                                     fill.insertCell().textContent = d.Weekday;
```

```
                                    fill.insertCell().textContent = d.Sunday;
                                    fill.insertCell().textContent = d.Average;
                                   });
          return table;
         }


//references:
//https://www.w3schools.com/jsref/met_tablerow_insertcell.asp
//https://www.w3schools.com/jsref/prop_node_textcontent.asp
//https://www.w3schools.com/jsref/met_document_createelement.asp




// barData = data.map(d => {return {Year: d.Year,Average: d.Weekday === '--' ? null :
(parseInt(d.Weekday) + parseInt(d.Sunday)) / 2 };
//                          })
// .filter(d => d.Year !== "1940");


// function addEltToSvg(name, attrs, appendTo)
// {
//     var element = document.createElementNS("http://www.w3.org/2000/svg", name);
//     if (attrs === undefined) attrs = {};
//     for (var key in attrs) {
//         element.setAttributeNS(null, key, attrs[key]);
//     }
//     if (appendTo) {
//         appendTo.appendChild(element);
//     }
//     return element;
// }


// horizontalSvg = {
//   const divElt = html`<div id="chart"></div>`;
//   const svgElement = document.createElementNS("http://www.w3.org/2000/svg", "svg");
//   svgElement.setAttribute('width', 300);
//   svgElement.setAttribute('height', 600);
//   divElt.appendChild(svgElement);

//   const barHeight = 600 / barData.length;
//   const maxAverage = Math.max(barData.map(d => d.Average).filter(Boolean));

//     barData.forEach((d, i) => {
//         if (d.Average !== null)
```

```
//         {
//             const barWidth = (d.Average / maxAverage) * 250;
//             addEltToSvg(svgElement, 'rect', { x: 50, y: i * barHeight, width: barWidth,
height: barHeight - 1, fill: 'gray' });
//         }

//         addEltToSvg(svgElement, 'text', { x: 10, y: (i * barHeight) + (barHeight /
2),'font-size': 10 }, d.Year);
//         addEltToSvg(svgElement, 'text', { x: 300,y: 80, 'font-size': 10, });


//     return divElt;
// }
```

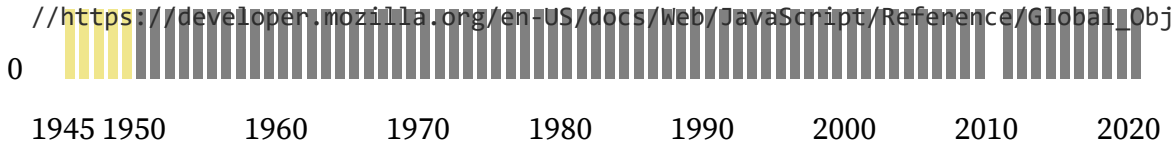



```
horizontalGraph = undefined
```

```
horizontalSvg = {
  const svg = document.createElementNS("http://www.w3.org/2000/svg", "svg");
  svg.setAttribute("width",300);
  svg.setAttribute("height",600);
```

```
  return svg;
}
//creates canvas of dimensions 300 x 600



horizontalGraph = {
    const max_avg = Math.max(...processData.map(d => d.Average)); //stores max averrage value
    const startYear = processData[0].Year; //stores year at index 0 position from processData
    const endYear = processData[processData.length - 1].Year; //stores year at last index
position from processData
    //above 3 constants for labeling the graph

    const totalYears = endYear- startYear + 1;
    const x_dim = 240/max_avg;
    const y_dim = 560/totalYears;

   //year has all years from 1945 to 2020 including 2010
    //year_data has years that are common in between years and processData(it doesnt have
year 2010)
    //i have used the years in year_data to plot average values

    for (let year = startYear; year <= endYear; year++) //increments year stating from
starting year to ending year - (1)
    {
        const i=year-startYear;  // To find index
        const year_data = processData.find(d =>d.Year===year); // looks for year in the
processData and compares it with year in eq 1
                      above.

        // plot the bar If exists
        if (year_data)
        {
            const rect = document.createElementNS("http://www.w3.org/2000/svg", "rect");
            rect.setAttribute("x","60"); //left some space for labels
            rect.setAttribute("y", y_dim*i); // finds position for bars
            rect.setAttribute("width", year_data.Average * x_dim);
            rect.setAttribute("height", y_dim-1);// Use the y_dim as the height
            rect.setAttribute("fill","gray");
            horizontalSvg.appendChild(rect);
        }

        // Print the label for starting year, ending year, and every decade
        //EXTRA CREDIT for labelling decade
        if (year === startYear || year === endYear || year % 10 === 0)
        {
            const text = document.createElementNS("http://www.w3.org/2000/svg", "text");
            text.setAttribute("x","30");
            text.setAttribute("y",y_dim * i + 9);
```

```
            text.setAttribute("text-anchor","middle");//keeps text at middle
            text.textContent = year;
            horizontalSvg.appendChild(text);
        }
    }

      // Minimum average  label
    const minavg_label = document.createElementNS("http://www.w3.org/2000/svg", "text");
    minavg_label.setAttribute("x", 50);
    minavg_label.setAttribute("y", 580);
    minavg_label.setAttribute("text-anchor","start"); //keeps text at the end
    minavg_label.textContent = "0"; // i kept value as  0 as shown in the sample example in
assignment 2.
    horizontalSvg.appendChild(minavg_label);

    // Maximum average label
    const maxavg_label = document.createElementNS("http://www.w3.org/2000/svg", "text");
    maxavg_label.setAttribute("x", 60 + max_avg * x_dim);
    maxavg_label.setAttribute("y", 580);
    maxavg_label.setAttribute("text-anchor","end"); //keeps text at the end
    maxavg_label.textContent = Math.round(max_avg/1000000) + " mil"; // rounding off max avg
value by dividing by million(to keeop text
                        similar to sample example)
    horizontalSvg.appendChild(maxavg_label);
}

// References:
//find method : https://www.w3schools.com/jsref/jsref_find.asp
//https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math
```




```
verticalGraph = undefined
```

```
verticalSvg = {
  const svg = document.createElementNS("http://www.w3.org/2000/svg", "svg");
  svg.setAttribute("width", 600);
  svg.setAttribute("height", 300);
  return svg;
}
//creates canvas of dimensions 600 x 300



verticalGraph = {

    const max_avg=Math.max(...processData.map(d => d.Average)); //stores max averrage value
    const startYear = processData[0].Year; //stores year at index 0 position from processData
    const endYear = processData[processData.length-1].Year; //stores year at last index
position from processData
    //above 3 constants for labeling the graph

    const padding = 2.2; // for the gap b/w each rect bar.
    const x_dim = 540/(endYear-startYear+1);
    const y_dim = 230/max_avg;
    const bar_dim = x_dim - padding;  // bar width


     //year has all years from 1945 to 2020 including 2010
     //year_data has years that are common in between years and processData(it doesnt have
year 2010)
    //i have used the years in year_data to plot average values

    for (let year = startYear; year <= endYear; year++) //increments year stating from
starting year to ending year - (1)
    {
      const i = year - startYear;  // To find index
```

```
        const year_data = processData.find(d => d.Year === year); // looks for year in the
    processData and compares it with year in eq 1
                        above.


        if (year_data)
        {
            const rect = document.createElementNS("http://www.w3.org/2000/svg", "rect");
            rect.setAttribute("x", (x_dim * i) + padding/2 + 30);
            rect.setAttribute("y", 260 - (year_data.Average * y_dim));
            rect.setAttribute("width", bar_dim);
            rect.setAttribute("height", year_data.Average * y_dim);
            rect.setAttribute("fill","gray");


          //interaction(highlight)
          //Math.floor(year_data.Year/10)*10 returns the starting point of a decade
          //year%10 modulues method can also be used here as the remainder with 0 will be the
    decade starting point.
          //if (year_data.Year-year_data.Year%10===decade-decade%10)

          if (Math.floor(year_data.Year/10)*10 === Math.floor(decade/10)*10)
          {
            rect.setAttribute("class", "highlighted");
          }

          verticalSvg.appendChild(rect);

        }

        ////EXTRA CREDIT for labelling decade
        // labels year ending with 0(decade) and start,end year.
        if (year % 10 === 0 || year === startYear || year === endYear)
        {
            const text = document.createElementNS("http://www.w3.org/2000/svg", "text");
            text.setAttribute("x", x_dim*i+ 30); //finds postion for labeling text.
            text.setAttribute("y", 290);
            text.setAttribute("text-anchor", "middle"); //keeps the text in the middle
            text.textContent = year;
            verticalSvg.appendChild(text);
        }
    }

    // Mininum average label.
    const minavg_label = document.createElementNS("http://www.w3.org/2000/svg", "text");
    minavg_label.setAttribute("x", 10);
    minavg_label.setAttribute("y", 260);
    minavg_label.setAttribute("text-anchor", "end"); //keeps text at the end
```

```
    minavg_label.textContent = "0"; // i kept value as  0 as shown in the sample example in
assignment 2.
    verticalSvg.appendChild(minavg_label);

    // Maximum average label
    const maxavg_label = document.createElementNS("http://www.w3.org/2000/svg", "text");
    maxavg_label.setAttribute("x",40);
    maxavg_label.setAttribute("y",30);
    maxavg_label.setAttribute("text-anchor","end");//keeps text at the end
    maxavg_label.textContent = Math.round(max_avg/1000000) + " mil"; // rounding off max avg
value by dividing by million(to keeop text
                        similar to sample example)
    verticalSvg.appendChild(maxavg_label);
}

//references:
//set class attribute for highlight:
https://www.w3schools.com/jsref/met_element_setattribute.asp
//find method : https://www.w3schools.com/jsref/jsref_find.asp
//https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math
//https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/floor

<style>
  .highlighted {
    fill: khaki;
  }
</style>
<!-- selects highlighted class and fills it with colour -->
```