Lecture Exercise 5 (5/2)

Submit your team number

Question *Submitted May 2nd 2023 at 4:30:54 pm*

Please enter your team number.

12

1. Draw a scatterplot using a CSV file

1. Create **exercise5.html**. Make sure to import d3.min.js as well.

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

2. Load/process the dataset that we used for Lab 3 (cereal.csv). You can refer to the code from Lab3 as well. In this part, instead of using the rowConverter callback function, we will now use a function for processing the dataset by directly iterating over it.

Question 1 *Submitted May 2nd 2023 at 5:21:49 pm*

In the body, parse the dataset cereal.csv using a function to directly iterate over the data and convert data types as needed (refer to slide from lecture 7 if necessary). Copy and paste your code below.

```
let rowConverter = function(d){
    return {
        name : d.Name,
        manufacturer : d.Manufacturer,
        calories : +d.Calories,
        carbo : +d.Carbo,
        year : +d.Year
    }
}

function main(data) {
    console.log(data)
}

d3.csv("cereal.csv", rowConverter)
    .then(data => main(data))
```

Question 2 *Submitted May 2nd 2023 at 5:43:34 pm*

Let's define dynamic scaling for the scatter plot:

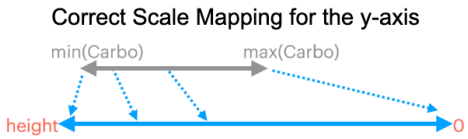
First, copy/paste the following code below inside the cereal.then() function after parsing the data (basically after data.forEach) and fill in your code to generate a scatterplot between Calories and Carbo. The x- and y- axis indicate Calories and Carbo respectively.

```
/* Draw scatterplot with Calories and Carbo */
const svgwidth = 300;
const svgheight = 300;

let svg = d3.select("body").append("svg")
    .attr("width", svgwidth)
    .attr("height", svgheight);

// Question 2: add your code below to draw a scatterplot
```

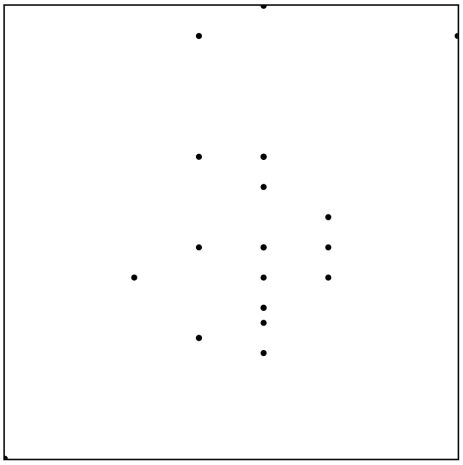
Define xScale and yScale using `d3.scaleLinear()` which will be used for the x- and y-axis respectively. You may find these two D3 methods useful when defining the domain and range.

- d3.min() extracts all Calories values from the data then finds the minimum
- d3.max() uses the same scheme as d3.min() to find the maximum

(Hint) the y-axis grows downward, not upward. In other words, the bottom dots have high y-coordinates on the <svg> canvas. The visualization of the correct mapping below would help.



After applying dynamic scaling to both x- and y-axis, the distribution of the point marks should look exactly as below (without the border). If different, likely your scaling is not done properly.



Copy and paste the code below:

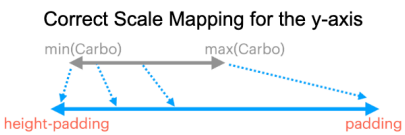
```
let xScale = d3.scaleLinear()
    .domain([d3.min(data, d => d.calories), d3.max(data, d => d.calories)])
    .range([0, svgwidth])
let yScale = d3.scaleLinear()
    .domain([d3.min(data, d => d.carbo), d3.max(data, d => d.carbo)])
    .range([0, svgheight])

svg.selectAll(".scatterpoint")
    .data(data).enter()
        .append("circle")
    .attr("class", "scatterpoint")
    .attr("cx", d => xScale(d.calories))
    .attr("cy", d => svgheight - yScale(d.carbo))
    .attr("r", 2)
```

Question 3 *Submitted May 2nd 2023 at 5:45:13 pm*

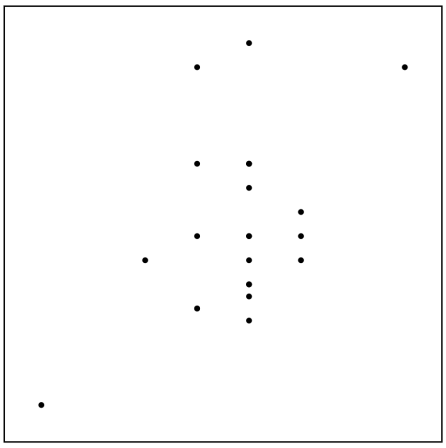
Notice that on the current scatterplot some circles around the edges are only partially shown. The reason is that the min(or max) cx/cy values are mapped to the smallest (or largest) pixel in the canvas, which hides some part of the circles. You can fix that too by adding some buffer space, called **padding**, around the edges.

Define a variable padding and modify xScale and yScale accordingly.

(Hint) The visualization below would help you find the answer for yScale. And you can apply the same strategy for xScale.



The new output should look as below (without the border). No points are hidden!



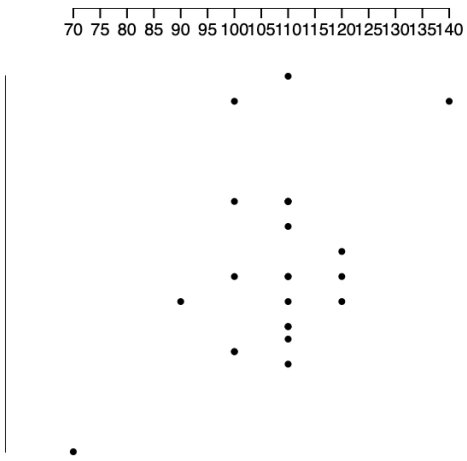
```
const padding = 25;
let xScale = d3.scaleLinear()
    .domain([d3.min(data, d => d.calories), d3.max(data, d => d.calories)])
    .range([padding, svgwidth - padding])
let yScale = d3.scaleLinear()
    .domain([d3.min(data, d => d.carbo), d3.max(data, d => d.carbo)])
    .range([padding, svgheight - padding])

svg.selectAll(".scatterpoint")
    .data(data).enter()
        .append("circle")
    .attr("class", "scatterpoint")
    .attr("cx", d => xScale(d.calories))
    .attr("cy", d => svgheight - yScale(d.carbo))
    .attr("r", 2)
```

Question 4 *Submitted May 2nd 2023 at 5:50:18 pm*

We will now add axes to the scatter plot.

- Adapt xAxis and yAxis to the current scatterplot. You can refer to the lecture slides today. You can find the full list of axis methods here.

Resulting scatterplot:



⚠ Notice that the axis doesn't look quite right for now - we will fix it in the next question.

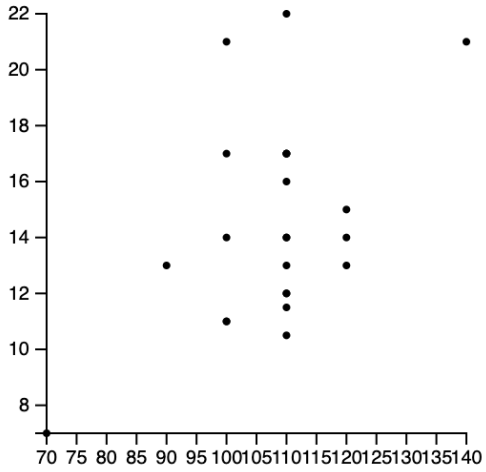
Copy paste the code below:

```
svg.append("g").call(d3.axisBottom(xScale))
    .attr("class", "xAxis")
svg.append("g").call(d3.axisRight(yScale))
    .attr("class", "yScale")
```

Question 5 *Submitted May 2nd 2023 at 5:54:31 pm*

Now let's correct the position of each axis. “transform” attribute is key. Take a look at the demo of "Translate" type transformation in this article and update the position of the axes.

Resulting scatterplot:



Copy paste the code below:

```
svg.append("g").call(d3.axisBottom(xScale))
.attr("class", "xAxis")
.attr("transform", `translate(${0},
svgheight − padding)‘)svg.append(”g”).call(d3.axisLeft(yScale)).attr(”class”,”yScale”).attr(”transf
{padding})`)
```

2. Color the dots for each manufacturer

Now we want to encode another attribute, Manufacturer, to the current scatterplot. Recall the channel rankings framework we learned in the last lecture: Manufacturer is an unordered attribute, so we are going to use the **color-hue** channel.

Encode Manufacturer by following the steps below:

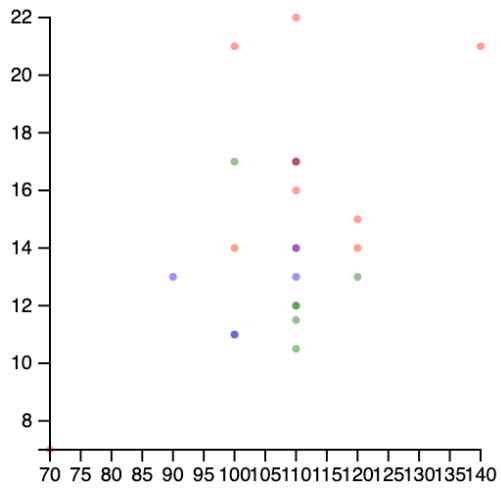
1) Set different color hues for each manufacturer using the color map below:

```
const colorMapping = {
  "G": "green",
  "K": "red",
  "P": "blue"
};
```

2) Set the opacity to be 0.5 (in case some dots are overlaid on top of others).

Define a colormap using **d3.scaleOrdinal()** with the domain and range. Then you can use the colormap to set the `fill` attribute of each circle.

Expected output:



Question *Submitted May 2nd 2023 at 6:38:20 pm*

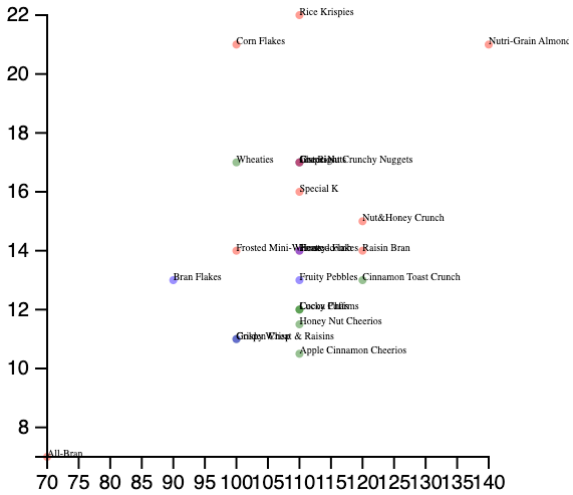
Copy and paste your code that selects different colors for different manufacturers.

```
        .attr("opacity", 0.5)
        .attr("fill", d => colorMapping[d.manufacturer])
```

3. Add a Label to the Scatterplot

Add labels of cereal names, on the bottom-right with font size "5px" of each data point.

Your output should be similar to:



Question *Submitted May 2nd 2023 at 6:45:51 pm*

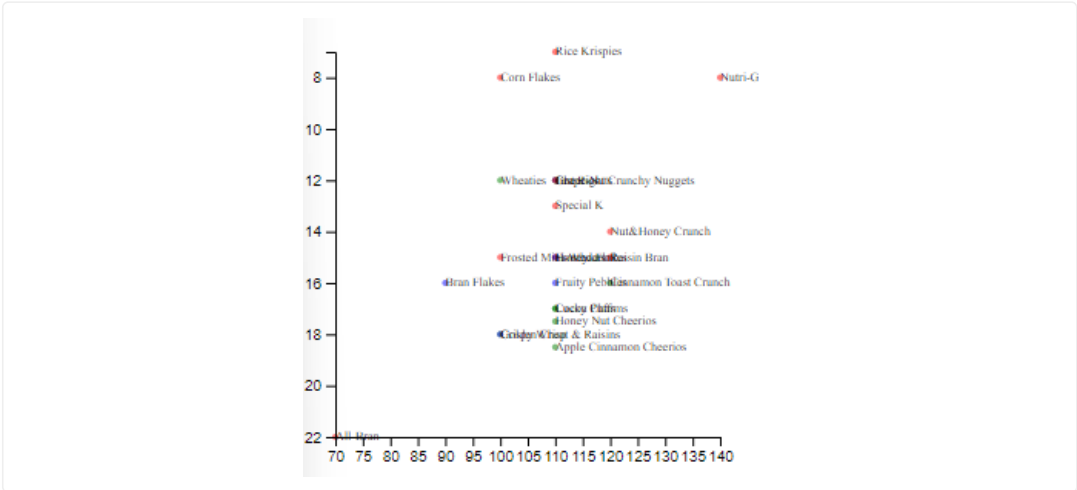
Copy and paste the part of your code which adds labels of cereal names.

```
svg.selectAll(".scatterpoint-label")
    .data(data).enter()
        .append("text")
    .attr("x", d => xScale(d.calories))
    .attr("y", d => svgheight - yScale(d.carbo))
    .attr("text-anchor", "right")
    .attr("dy", ".3em")
    .text(d => d.name)
    .style("font-size", "8px")
```

Upload Your Files

Question 1 *Submitted May 2nd 2023 at 6:46:13 pm*

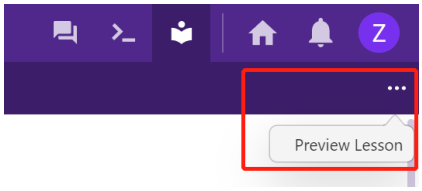
Upload the screenshot of your resulting webpage. You will need to click the "clip" button to upload a file into the Answer box.



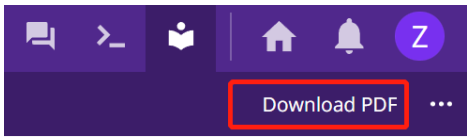
Question 2

You need to download the PDF of lecture exercise 3 and upload it with other files to the Gradescope. Follow the instructions on how to download PDF file:

1. Click on the ellipsis button and the Preview Lesson.



2. After that, click on the Download PDF button.



- ○ PDF downloaded!
- ○ Haven't done yet!

Question 3

Upload the following files to Gradescope. You need to make **a group submission,adding all present members in your team**, so that the present members get the participation credit.

Files to upload:

- exercise5.html
- PDF you downloaded as Q2

- [ ] Our team uploaded the the files on gradescope!
- [ ] Oops, our team did not upload the files on gradescope!

Feedback

Question

Was the activity today clear? If not, please share how the course can improve it. Your comments will help us design future lab content (and also future students).

No response