COS30045 Data Visualisation

Exercise 5.1 D3 Updating the data

ILO	Create web-based interactive visualisations using real-world data sets.
Aim:	Use D3 to update your data.
Resources:	*Textbook:* Chapter 9 Updates, Transitions and Motion - Murray (2017) Interactive Data Visualisation (2nd Ed) on ProQuest
Demonstration	If you are required to demonstrate this exercise we will be looking for: - code that is appropriate for exercise, well formatted and commented - code that runs correctly and meets the requirements specified in this exercise - an explain programming features and concepts in the code - the ability to successfully edit code to change a specified feature of the program

Note: The functions handling scale have changed between D3 v3 and D3 v4. This is something to be aware of if you are doing your own research into this topic. Make sure you use Murray Ed 2. Code examples from Ed 1 will not work.

Overview

In this tutorial we will start with your code from Drawing with Data - Bar Chart. At the end of this exercise you should end up with a bar chart that refreshes with random data (up to 25) every time you press the update button. It should look a bit like this:
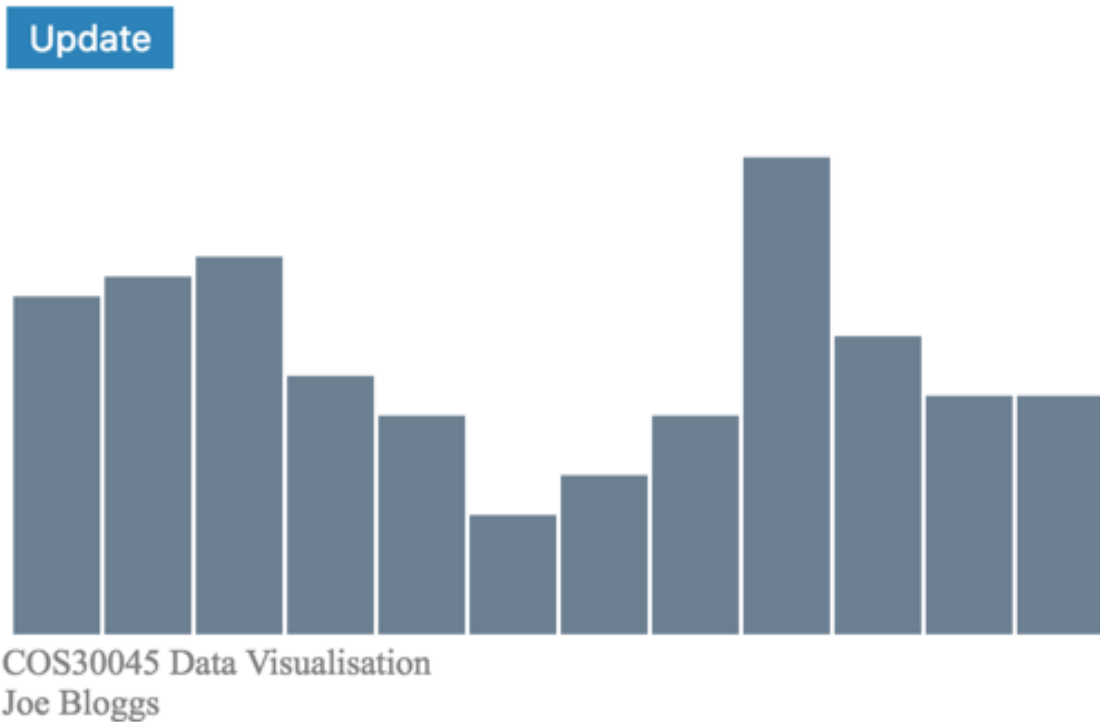
> ### *Requirements*
>
> ☐ code produces a scaled bar chart that the user can refresh with random data

Step 1: Start with the code from Task 2.2

We will start with the code from the previous bar chart task. Unlike your scatter plot from 3.1 and 3.2 this chart is not yet scalable. So our first job is to make it scaleable.

Step 3: Scaleable Ordinal Axis

In the scatter plot example we use `scaleLinear ()` to generate the scale for us. However, `scaleLinear()` is designed for use with continuous variables. Although the columns in this bar chart are also numbers, most bar charts we build will use categorial data. In this case we are going to pretend that our data is ordinal in nature (i.e. has some kind of order to it such as high, medium and low).

We can use `scaleBand()` to generate an ordinal scale for out x-axis. As per using `scaleLinear()` we need to specify the input domain and the output range. If you were using true ordinal data you could use the category labels to specify the domain input bands as in:

```
.domain(["high", "medium", "low"])
```

However, because we are using a set if of numbers we can generate numbers from 0 to range for our bars using `d3.range()`. As before we also need to specify the out put range relative to the current size of our svg using `range()`.

Use `scaleBand()` to create an ordinal scaleable x-axis based on the range of the data set.

D3 will now calculate the width each 'band' needs to be in pixels to fit the length of the input domain.

```
var dataset = [14, 5, 26, 23, 9, 21, 7, 19, 2          ];

var xScale = d3.scaleBand()                    Ordinal scale
                                               method
              .domain(d3.range(dataset.length))
              .range([0,w]);
                                                    calculate the range
                                                    of the domain

var svg1 = d3.select("body")
            .append("svg")
            .attr("width", w)                   specify the size of the range
            .attr("height", h);                 the domain needs to be
                                                mapped too.
```

Go ahead now and update the x attribute of our generated rectangles to be relative to our new `xScale(i)`.

```
.attr("x", function(d, i) {
    return xScale(i);
})
```

If you save and run this you will see that nothing has changed visually in the chart.

At the moment our width of our rectangles is determined by dividing the width of the svg canvas by the data set length (minus a bit of padding).

```
.attr("width", w / dataset.length - barPadding)
```

However, we can now make use of our scale to calculate the width of our bars using `bandwidth()`.

```
.attr("width", xScale.bandwidth())
```

If you save and run this you will see that something has changed visually in the chart.

Currently our scale does not specify any padding. So lets go back to the scale and add padding. This time we can use `paddingInner()` to calculate the padding rather than hard coding the padding as a variable.

```
var xScale = d3.scaleBand()
                .domain(d3.range(dataset.length))
                .range([0,w])
                .paddingInner(0.05);
```
generates a padding value of 5% of the band width.

Add padding to your chart that changes relative to the data set.

Now save your file and note that the padding is back! However, the rectangles are a bit fuzzy. If you check the DOM you will see that the calculated values for the bandwidth are to many decimal places, and because this does not line up neatly with full pixels it looks a bit tragic.

```
▼<svg width="500" height="250">
    <rect x="0" y="194" width="39.7489539748954" height="56" fill="slategrey"></rect>
    <rect x="41.84100418410042" y="230" width="39.7489539748954" height="20" fill="slategrey"></rect>
    <rect x="83.68200836820084" y="146" width="39.7489539748954" height="104" fill="slategrey"></rect>
    <rect x="125.52301255230125" y="158" width="39.7489539748954" height="92" fill="slategrey"></rect>
```

So you can use `rangeRound()` instead of `range()` to round the bandwidths to whole numbers.

```
▼<svg width="500" height="250">
    <rect x="5" y="194" width="39" height="56" fill="slategrey"></rect>
    <rect x="46" y="230" width="39" height="20" fill="slategrey"></rect>
    <rect x="87" y="146" width="39" height="104" fill="slategrey"></rect>
    <rect x="128" y="158" width="39" height="92" fill="slategrey"></rect>
```

Now your bars will look crisp and clear.

Step 4: Add Scale for y-axis

Now you have your x-axis scaleable, it's time to do the same for your y-axis. In constructing your yScale remember that:

- the y data is quantitative data

- the highest value in the data set domain can be found using `d3.max()`

- you need to use yScale in defining both the x position and the height of the rectangles.

Add a scalable y-axis to your chart.

Step 4: Updating the data: adding an event listener

A key aspect of using D3 is that we can make *interactive* charts, that is the user can make some change to their view of the data. In your JavaScript exercise we used a button to change the image file being displayed. Now we are going to use a button to update the data in our bar chart.

Go grab the code from your JavaScript Review exercise and add your button into this program (note you will need to modify it a bit). To start with we will be using the button to update the data once with a new static data set. When your button is added it should look something like this:

Update Data

`Update Once`

COS30045 Data Visualisation
Joe Bloggs
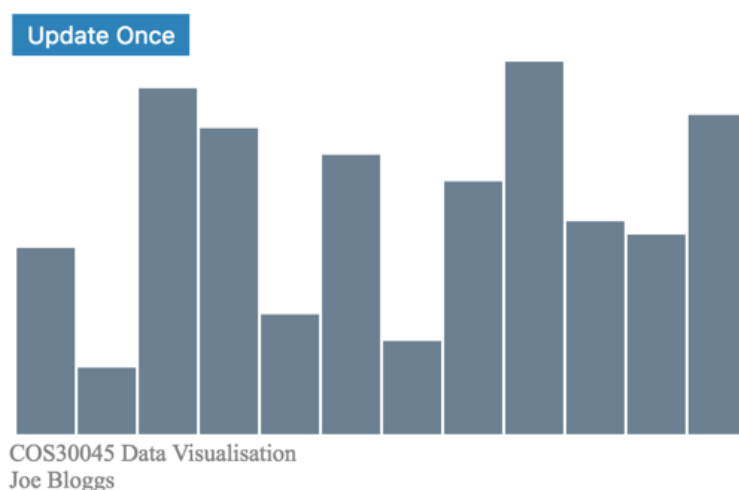
You can use D3 to listen for a click on your new button. Test that your button works with a pop up alert.

```
d3.select("button")
    .on("click", function() {
        alert("Hey, the button works!")
    });
```

Once your button is working, it's time to up make it do something more interesting, like update the data set. Firstly, add your new data set. Make the data different to the one you started with.

```
var dataset = [24, 10, 29, 19, 8, 15, 20, 12, 9, 6, 21, 28];
```

Next you need to update rectangles with the data from the new data set and update the y and height values with the new data.

```
svg1.selectAll("rect")
    .data(dataset)
    .attr("y", function(d) {
        return h - yScale(d);
    })
    .attr("height", function(d) {
        return yScale(d);
    })
```

Step 5: Updating the data over and over again!

Instead of just having one hard coded set of data, let's add in a randomly generated data set.

Add in some code for randomly generating numbers when the button is pushed.

```
var numValues = dataset.length;

dataset = [];

for (var i = 0; i < numValues; i++) {
    var newNumber = Math.floor(Math.random()* maxValue);
    dataset.push(newNumber);
}
```

Make sure you remember to add in a new variable for maxValue. Set it to 25. Now you should be able to get a new set of data between 0 and 24 every time you click the update button.

Try adding Axis labels to your chart. Murray Chapter 9 (p 158) for some hints about how to make your labels move with your updated data.