



# COS30045 Data Visualisation

## Exercise 4.3 D3 Stacked Bar Chart

<b>ILO</b>	Create web-based interactive visualisations using real-world data sets.
<b>Aim:</b>	Generate a Stacked Bar Chart using D3
<b>Resources:</b>	<i>Textbook:</i> <a href="#">Chapter 13 Layouts Murray (2017) Interactive Data Visualisation (2nd Ed) on Pro-Quest</a>
<b>Demonstration</b>	If you are required to demonstrate this exercise we will be looking for: <ul style="list-style-type: none"><li>- code that is appropriate for exercise, well formatted and commented</li><li>- code that runs correctly and meets the requirements specified in this exercise</li><li>- an explain programming features and concepts in the code</li><li>- the ability to successfully edit code to change a specified feature of the program</li></ul>

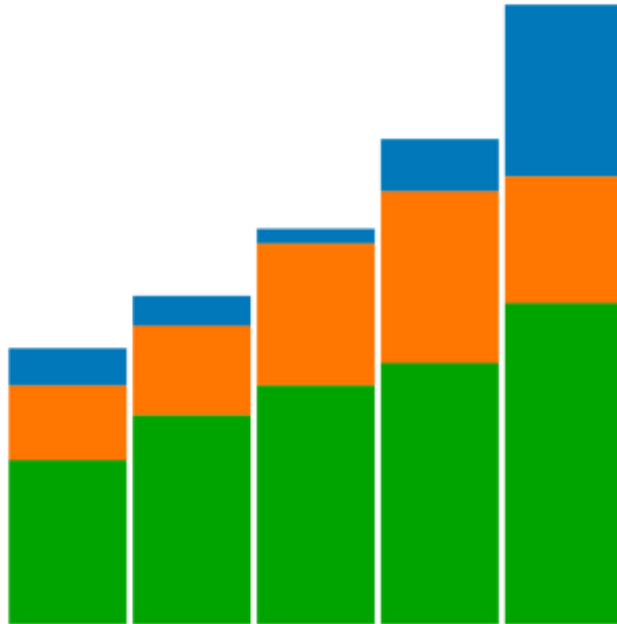
**Note:** The functions handling scale have changed between D3 v3 and D3 v4. This is something to be aware of if you are doing your own research into this topic. Make sure you use Murray Ed 2. Code examples from Ed 1 will not work.

Code in this Task based on Murray Ch 13

## Overview

In this exercise we will use D3 to draw a stacked bar chart.

### Stacked Bar Chart



#### *Requirements*

- ☐ staked chart created from data in an array

## Step 1: Set up the data

First we need some data to display. We will be using the fruit data set from Murray. We will also be displaying this chart on a SVG with a width and height of 300.

```
var dataset = [  
  { apples: 5, oranges: 10, grapes: 22 },  
  { apples: 4, oranges: 12, grapes: 28 },  
  { apples: 2, oranges: 19, grapes: 32 },  
  { apples: 7, oranges: 23, grapes: 35 },  
  { apples: 23, oranges: 17, grapes: 43 }  
];
```

## Step 2: Set up the stack

The data set above is arranged in columns. But to generate our stacked chart we need to organise them by categories. We can use `d3.stack()` to generate our stacks, and specify the categories of interest using `keys()`.

```
var stack = d3.stack()  
  .keys(["apples", "oranges", "grapes"]);  
  
var series = stack(dataset);
```

We can then generate a series which holds the array in a format that adds each category to the next.

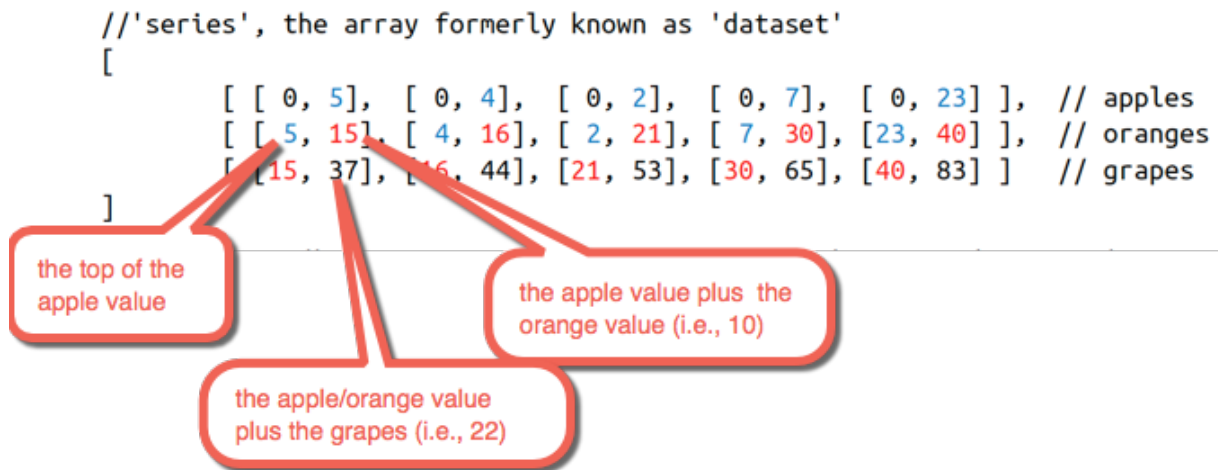


Fig 1: Series array from Murray

## Step 3: Set up the SVG and set up the arcs

As per usual set up the SVG canvas and bind the data to a group set.

```
var svg = d3.select("body")
    .append("svg")
    .attr("width", w)
    .attr("height", h);

var groups = svg.selectAll("g")
    .data(series)
    .enter()
    .append("g")
    .style("fill", function(d, i) {
        return color(i);
    });
```

Note that the data being read into the SVG is the series not the dataset. Also it uses a colour method as per the Pie Chart exercise.

## Step 4: Draw the rectangles

Bind rectangles to the group data. Note, as per earlier tasks you will need to add in an xScale and yScale. The main complication with this data set is that you need to access the values from an array so you will need to use the index to reference them.

```
var rects = groups.selectAll("rect")
    .data(function(d) { return d; })
    .enter()
    .append("rect")
    .attr("x", function(d, i) {
        return xScale(i);
    })
    .attr("y", function(d, i) {
        return yScale(d[1]);
    })
    .attr("height", function(d) {
        return yScale(d[0]) - yScale(d[1]);
    })
    .attr("width", xScale.bandwidth());
```

```
var yScale = d3.scaleLinear()
    .domain([0, d3.max(dataset2, function(d) {
        return d.apples + d.oranges + d.grapes;
    })
    ])
    .range([h, 0]);
```