School of Software and Electrical Engineering

COS30045 Data Visualisation

Exercise 7.1 D3 Paths - Line and Area charts

ILO	Create web-based interactive visualisations using real-world data sets.
Aim:	Use paths to draw line and area charts
Resources:	*Textbook:* Chapter 11 Paths - 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.

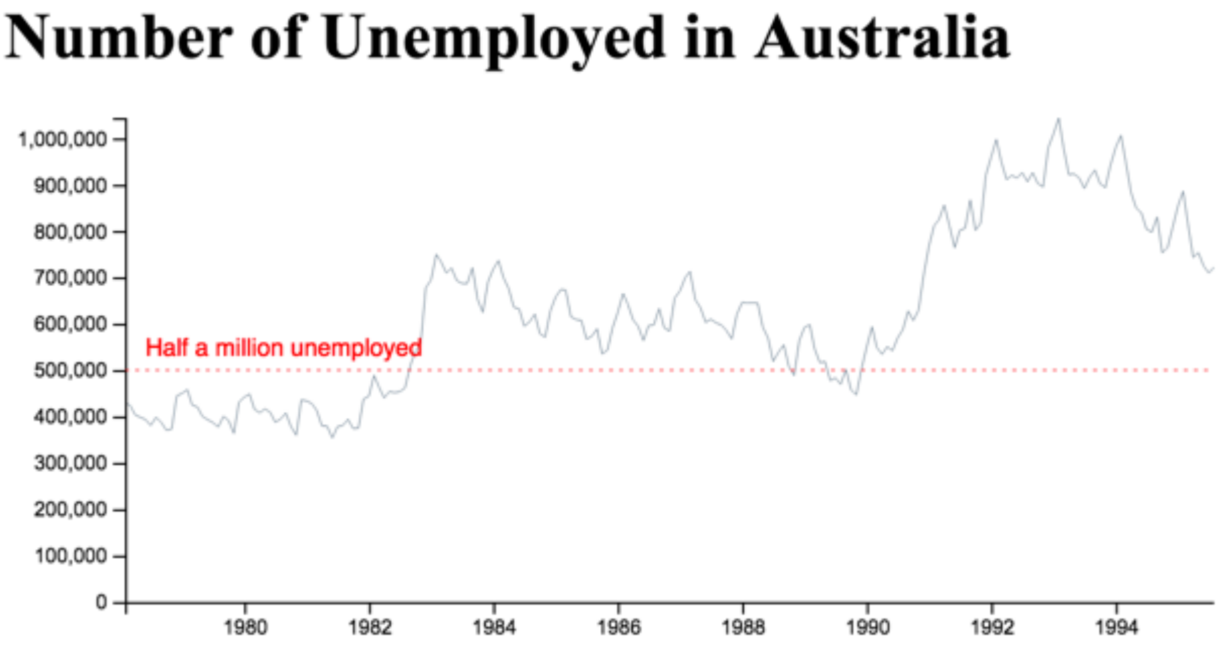
Code in this Task based on Murray Ch 11

Overview

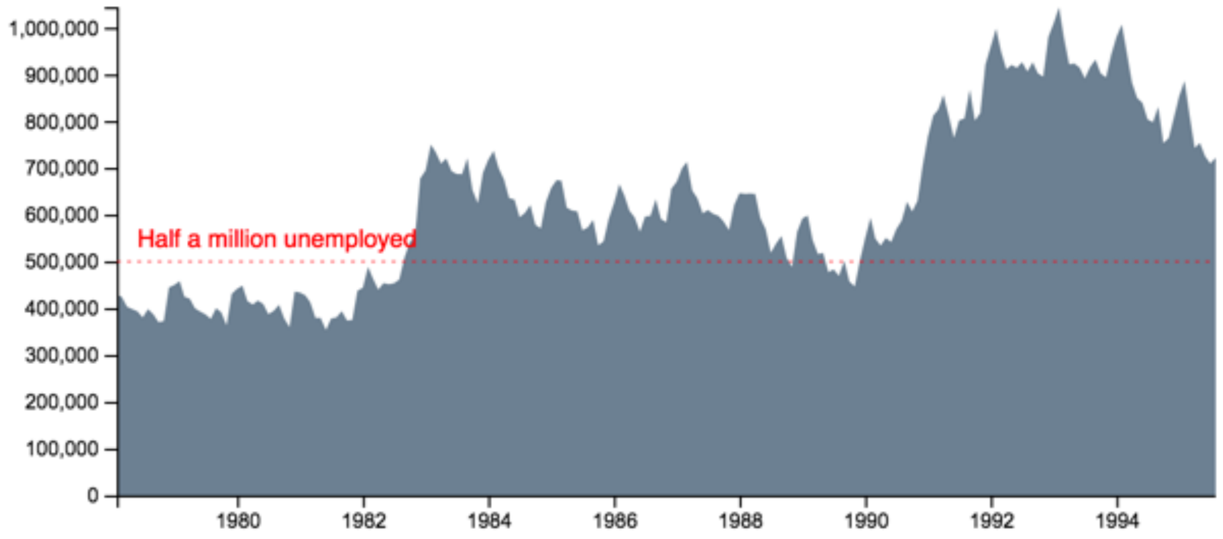
In this exercise we will be returning to our Importing Data from CSV exercise to import and plot a time series. Time series are a common type of chart and make use of the `path` functions of D3. `Path` covers any non-standard, irregular shapes (e.g., not rectangles and circles). In this task we will plot data for the monthly unemployment figures for Australia from 1978 to 1995.

Requirements

- ☐ line chart generated using imported data
- ☐ scaled x and y axis
- ☐ annotation (eg., half a million line)
- ☐ fill the area under the chart with a colour



Number of Unemployed in Australia

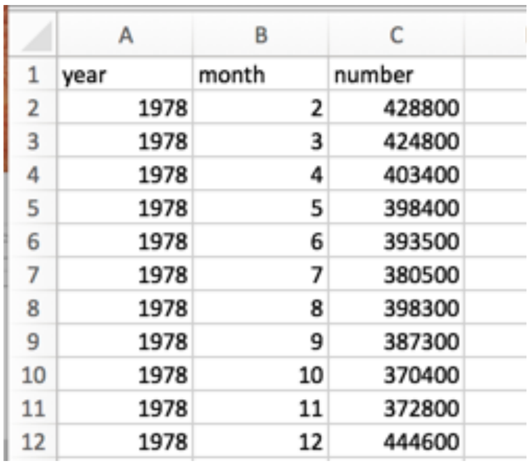


COS30045 Data Visualisation
Joe Bloggs

Step 1: Set data input file

Start with the code from Importing Data from CSV exercise. We want a slightly larger SVG canvas to put our line chart on so change w to 600 and h to 300. Also we have no 'bar-Padding' as we don't have Bars so you can delete the padding var for the minute.

The data we will be using is in CSV format and contains three columns - year, month and number.

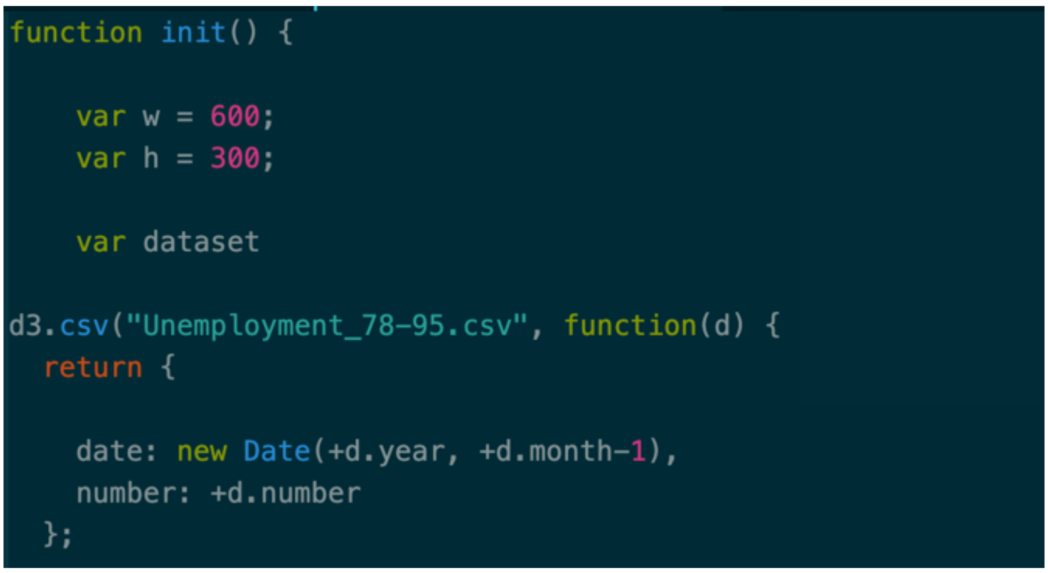


	A	B	C
1	year	month	number
2	1978	2	428800
3	1978	3	424800
4	1978	4	403400
5	1978	5	398400
6	1978	6	393500
7	1978	7	380500
8	1978	8	398300
9	1978	9	387300
10	1978	10	370400
11	1978	11	372800
12	1978	12	444600

The file is called Unemployment_78-95.csv

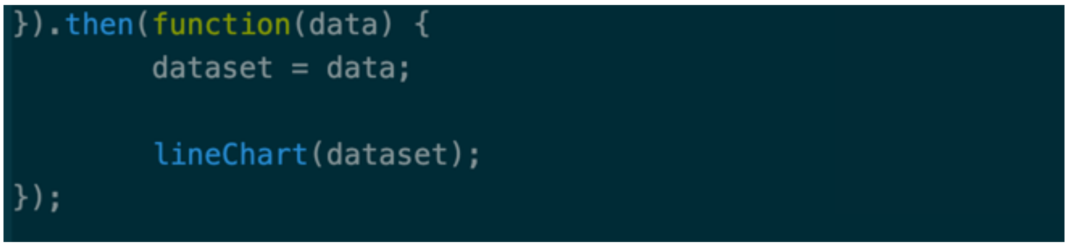
In the Importing Data from CSV exercise we only had one column of data, but now we have three and we need to tell D3 what they are and how to handle them using a function.

We can put the year and month together to get a Java-Script `Date` object using `new Date`. The -1 is there because JavaScript's month counting starts at 0 (ie Jan = 0), but our date column starts at 1.



```
function init() {

    var w = 600;
    var h = 300;

    var dataset

d3.csv("Unemployment_78-95.csv", function(d) {
  return {

    date: new Date(+d.year, +d.month-1),
    number: +d.number
  };
```

After that we load the data into our data set.



```
}).then(function(data) {
        dataset = data;

        lineChart(dataset);
});
```

To check the data has gone ok, you could add in:

```
console.table(dataset, ["date", "number"]);
```

which will print the data to the console.

Step 2: Set up the Scales

Now it is time to build our code for our line chart. In the Importing Data from CSV exercise we build a bar chart, you can delete most of the barChart() code (except for setting up the initial SVG).

The Scale set up is very similar to the Importing Data from CSV exercise except for a couple of key changes. Firstly, our xScale is not a straight number, it is a date. So instead of using `d3.scaleLinear()`, use D3's special `d3.scaleTime()`. Also remember that to assess the data you need to refer to the column heading (e.g., `d.number` or `d.date`).

```
xScale = d3.scaleTime()
               .domain([
                    d3.min(dataset, function(d) { return d.date; }),
                    d3.max(dataset, function(d) { return d.date; })
                ])
               .range([0, w]);

yScale = d3.scaleLinear()
                .domain([0, d3.max(dataset, function(d) { return d.number; })
                ])
                .range([h , 0]);
```

Step 3: Set up the Line

We will use `d3.line()` to generate our line. We need to tell it where to fid the x and y values. In this example the date data will be plotted along the x axis and the number of unemployed on the y axis.

```
line = d3.line()
             .x(function(d) { return xScale(d.date); })
             .y(function(d) { return yScale(d.number); });
```

Step 4: Set up the SVG and Path

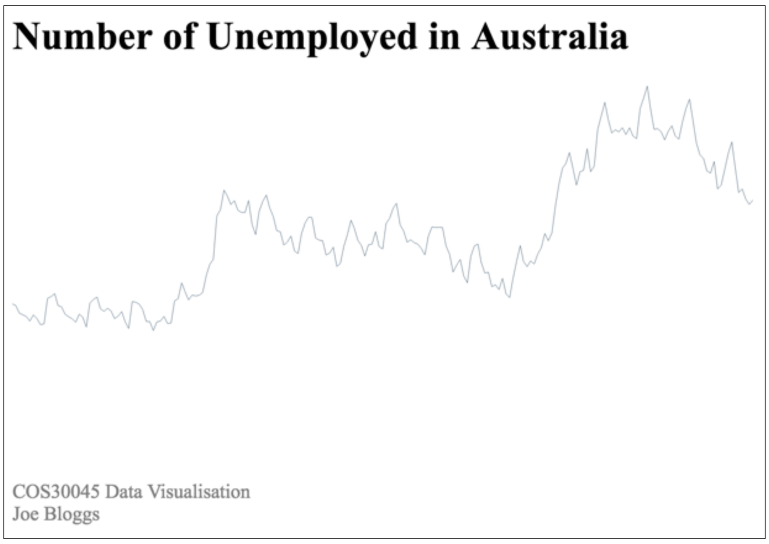
The final step in getting our line drawn is setting up the SVG canvas is the same as usual. However, the format for binding the data is different. `data()` is used to bind each single data value to a different html element. However, in this case we want to bind the data to a single path element. We use `datum()` to do this.

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

        svg.append("path")
            .datum(dataset)
            .attr("class", "line")
            .attr("d", line);
```

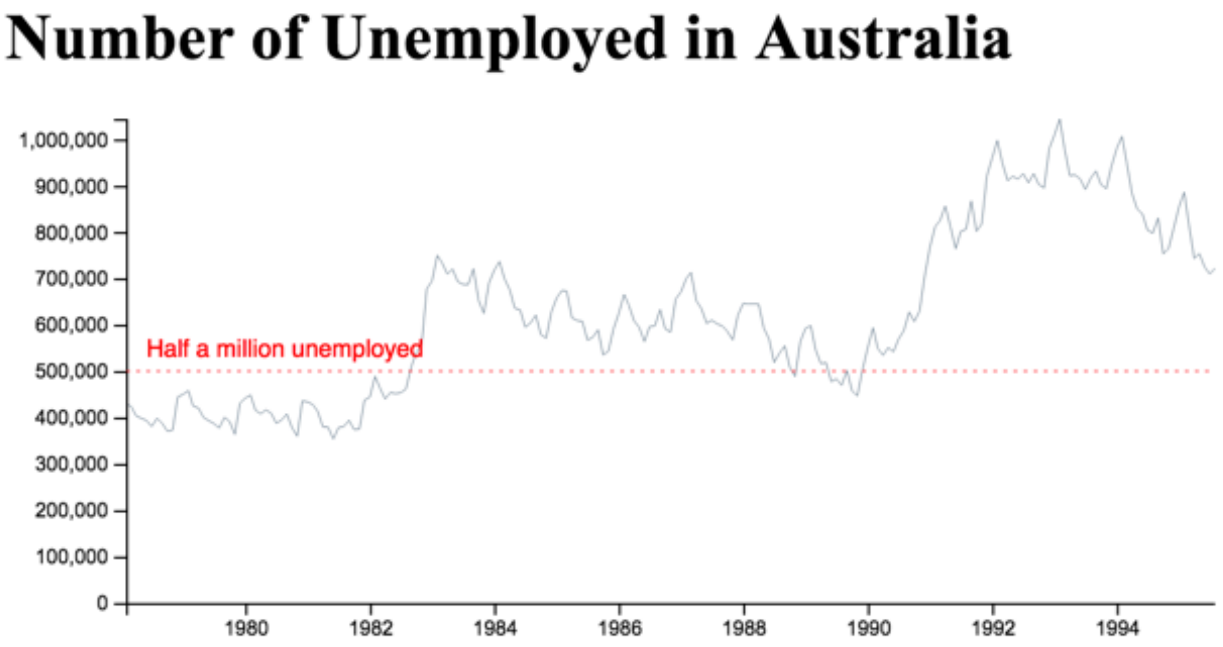
To make it look nice you can use the class attribute to specify some CSS styling on the line.

```
.line {
    fill: none;
    stroke: slategrey;
    stroke-width: 0.5;
}
```

Save and run the file and you should have a nice line chart like this…



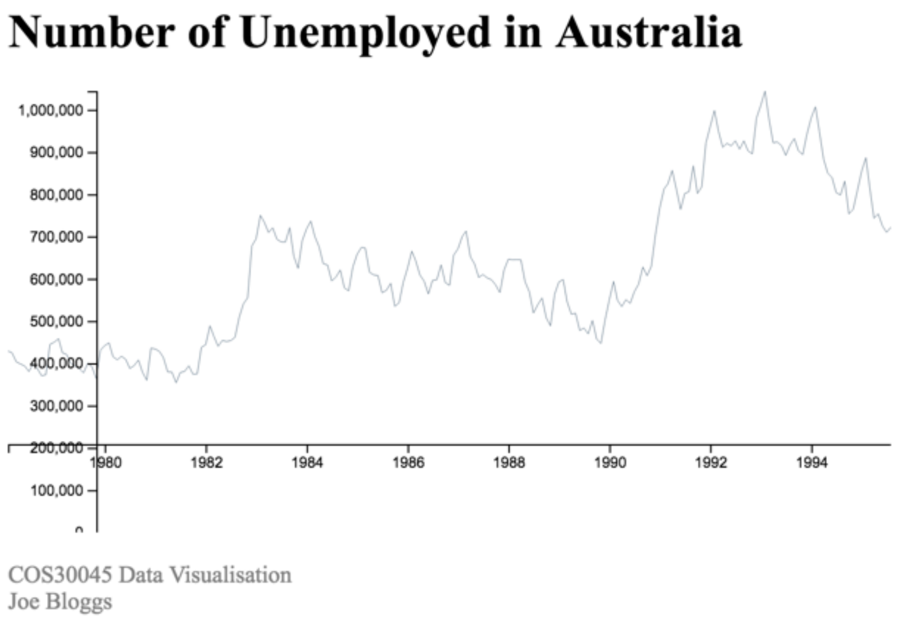
Unfortunately this is not very meaningful on it's own so we need to add some axis to help users interpret the data.



Step 5: Add Axis

Go back to Adding Axis to Charts and grab the code for generating axis', you should be able to do it with editing. However, you will need to add some padding to the range values in your x and y scales to make sure sure they fit.

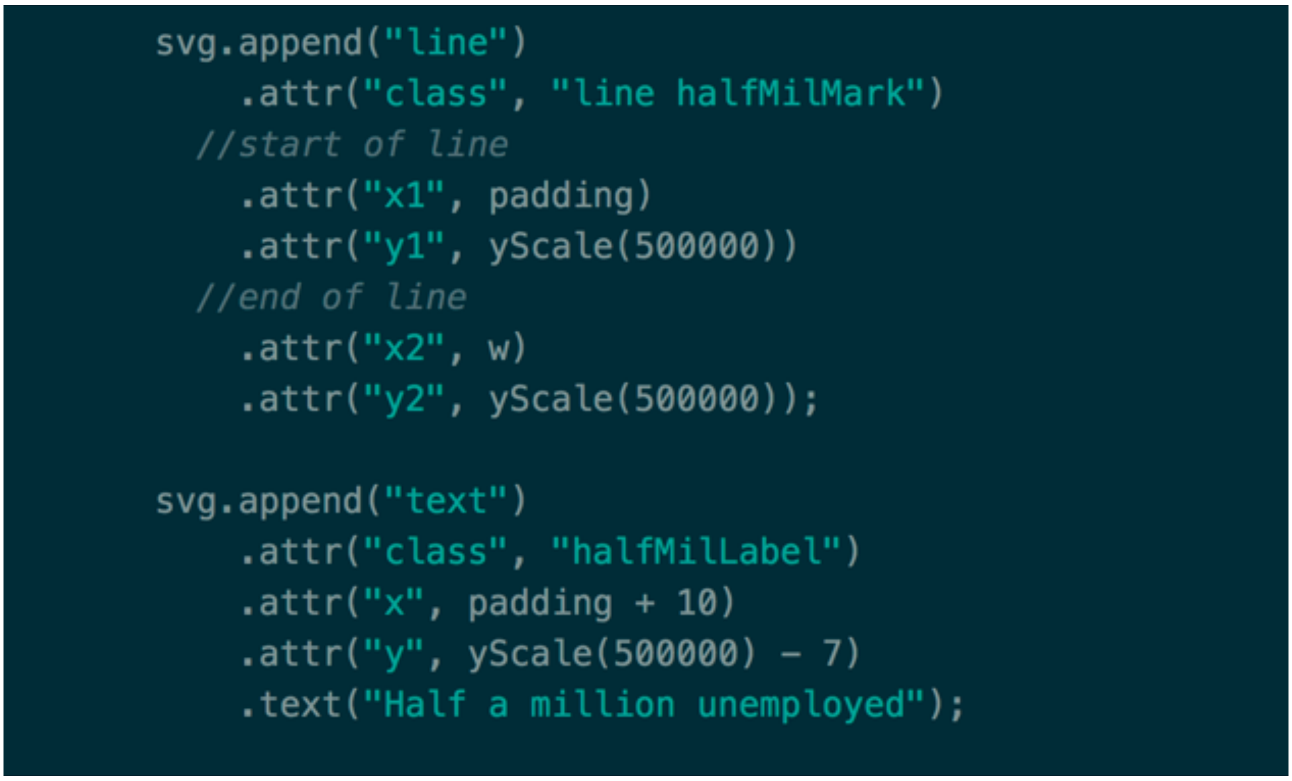
Without padding they will look like this (which is not so good):



Step 5: Add some annotations

Much of what we want to do with data visualisations is to draw peoples attention to important parts of the data.

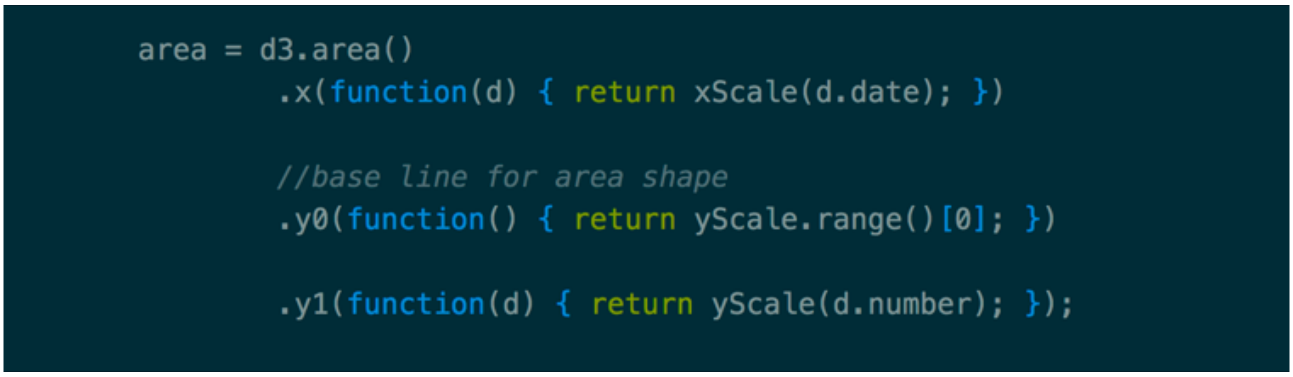
In this case a line has been drawn at half a million unemployed and adding in some CSS styling.



```
svg.append("line")
    .attr("class", "line halfMilMark")
  //start of line
    .attr("x1", padding)
    .attr("y1", yScale(500000))
  //end of line
    .attr("x2", w)
    .attr("y2", yScale(500000));

svg.append("text")
    .attr("class", "halfMilLabel")
    .attr("x", padding + 10)
    .attr("y", yScale(500000) - 7)
    .text("Half a million unemployed");
```

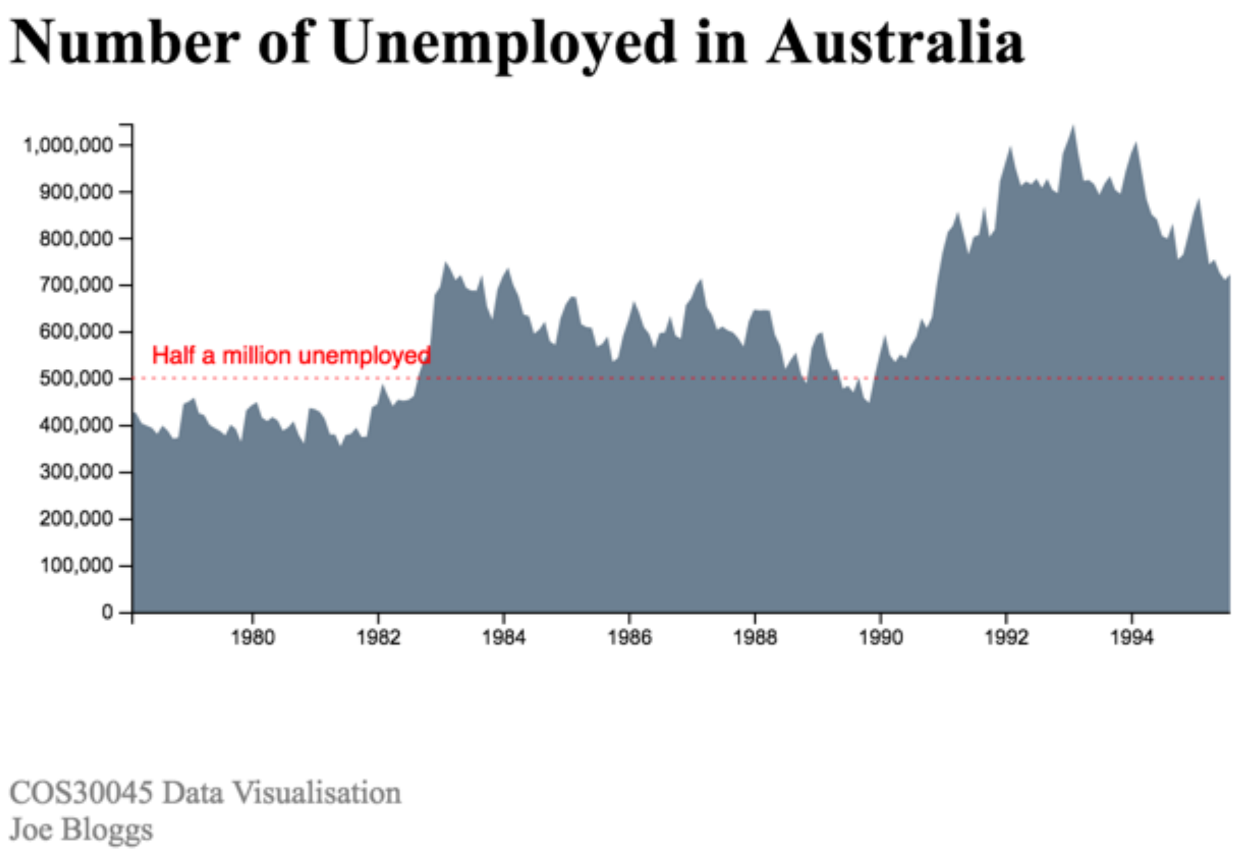
Step 7: Turn your line chart into an area chart

Turning a line chart into and area chart is not difficult. It uses the same path feature, only instead of generating a line you generate an area. The specifications are very similar, however, for and area you need to specify the bottom of the shape. In this case it is the 0 value of the yScale (i.e., the bottom of the chart).



```
area = d3.area()
        .x(function(d) { return xScale(d.date); })

        //base line for area shape
        .y0(function() { return yScale.range()[0]; })

        .y1(function(d) { return yScale(d.number); });
```

You will also need to update the append ("path") to area instead of line. With a bit of styling your chart could look something like this:


Number of Unemployed in Australia
Half a million unemployed
COS30045 Data Visualisation
Joe Bloggs