Faculty of Science, Engineering and Technology

COS30045 Data Visualisation

Exercise 4.5 D3 Choropleth

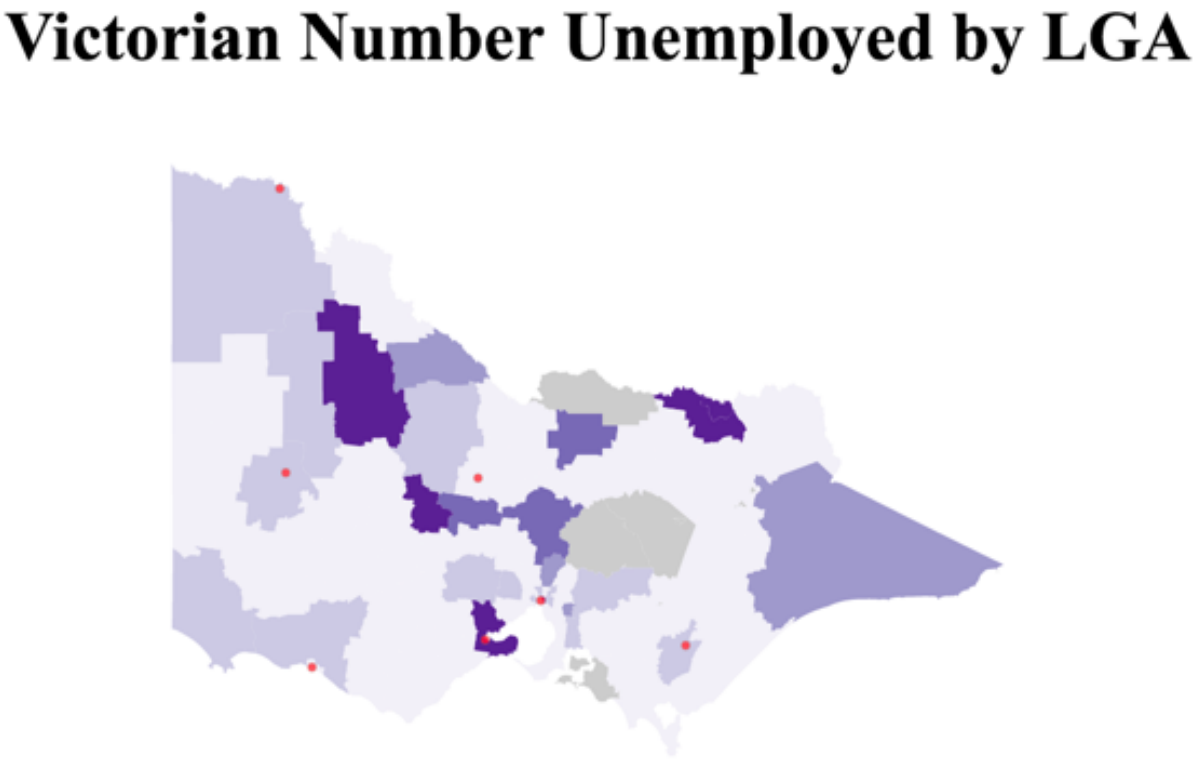
ILO	Create web-based interactive visualisations using real-world data sets.
Aim:	Build a choropleth
Resources:	*Textbook:* Chapter 14 Geomapping Murray (2017) Interactive Data Visualisation (2nd Ed) on ProQuest
Demonstration	If you are required to demonstrate this exercise we will be looking for: - code 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 14

Overview

In this exercise we will be attempting to get D3 to draw us some nice maps so that we can display data (in this case number of unemployed) geographically.



Requirements

- ☐ choropleth of Victorian LGAs showing unemployment data
- ☐ location of some Victorian towns marked

Step 1: Set the colour range

A key part of a choropleth is the use of saturation to encode data. Typically deeper colours indicate higher values and washed out colours indicate lower values.

D3 provides us with a way of scaling colours so that we end up with even bands. Use ColorBrewer to choose a colour scale range for use in your choropleth from your Geo Paths and Projections exercise.

```
var color = d3.scaleQuantize()
              .range([insert colour scheme here])
```

Step 2: Load in the data and map colour domain

In this task we are going to use unemployment data (number of unemployed per LGA see `vic_LGA_unemployment.csv`).

	A	B
1	LGA	unemployed
2	Alpine	221
3	Ararat	308
4	Ballarat	2432
5	Banyule	3207
6	Bass Coast	1141
7	Baw Baw	1188
8	Bayside	1699
9	Benalla	358

Use `d3.csv()` to load in the unemployment csv then set the colour domain so that the employment data is mapped to the colour range.

Note that all the code related to our map now needs to appear within the `d3.csv` function, including `d3.json()`.

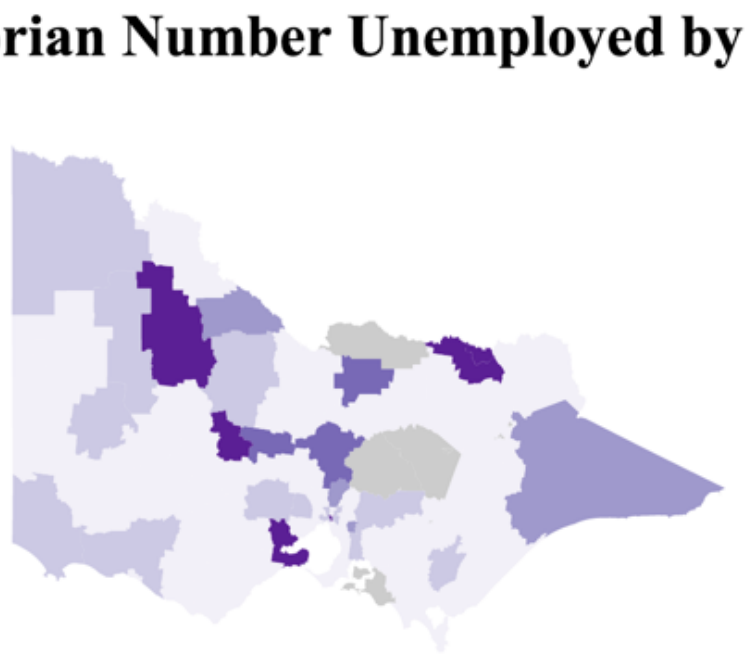
Step 3: Link map data and employment data

At the moment we have our data of interest in a CSV file and our map data in a separate JSON file. Our next step is to merge the CSV data with the JSON data so we can display our CSV data on our map. See below for Murray's preferred method for merging the two data sets. His example is for a different data set, you should adapt to make it work for our data set.

```
        d3.json("us-states.json", function(json) {

            //Merge the ag. data and GeoJSON
            //Loop through once for each ag. data value
            for (var i = 0; i < data.length; i++) {

                //Grab state name
                var dataState = data[i].state;

                //Grab data value, and convert from string to float
                var dataValue = parseFloat(data[i].value);

                //Find the corresponding state inside the GeoJSON
                for (var j = 0; j < json.features.length; j++) {

                var jsonState = json.features[j].properties.name;

                if (dataState == jsonState) {

                    //Copy the data value into the JSON
                    json.features[j].properties.value = dataValue;

                    //Stop looking through the JSON
                    break;

            }
        }
    }
```

Step 4: Add colour encoding for unemployment data

Use the `.style()` attribute and our `color` scale to change the fill of the shapes so that they encode the employment data.



Step 5: Add Victorian towns and cities

To help give a sense of place, add in some Victorian towns and cities. The file `VIC_city.csv` contains the longitude and latitude of a selection of Victorian cities/towns. Add in some small circles to indicate the place of these cities/towns.

Use `d3.csv()` to load in the city data. Then add some circles and position them according to their longitude and latitude. We can use the `projection` function we added earlier to map the longitude and latitude to x and y values on our SVG.

```
    .attr("cx", function(d) {
        return projection([d.lon, d.lat])[0];
    })
```