D3.js Workshop

A quick, deep-dive into D3.js for data visualisation

CS5346 Information and Data Visualisation

Part III

Advanced D3

Goals

Learning objectives for today

Learning Objectives

- To plot a bar chart based on previous data.
- To try out mapping with D3.
- To learn about alternative libraries for D3.

Bar charts in D3

Bar Charts

Making Bar Charts the D3.js Way

Workshop 3 Part 1

Our Data

- Covid-19 cases in Singapore from January to April 2020.
- Dataset available at ourworldindata.org:
  - https://ourworldindata.org/coronavirus/country/singapore?country=~SGP
- Things to take note:
  - Data format - CSV
  - Many columns
  - Many different data types

1. Load the dataset

2. Create the drawing area

3. Creating our scales.

Create a Scale with Date

- Note that the data type in the first column is a **date time data type**.
- First have to parse the data points as a date before creating a scale.

```
const xScaleTime = d3.scaleTime()
    .domain([d3.min(dataset, (d) => new Date(d['date'])),
             d3.max(dataset, (d) => new Date(d['date']))])
    .range([margins.left, bounded.width]);
```

Create a Scale with Normal Numbers

- The yValue axis is the same.

```
const yScale = d3.scaleLinear()
      .domain([d3.min(dataset, (d) => parseInt(d['new_cases'])),
               d3.max(dataset, (d) => parseInt(d['new_cases']))])
      .range([bounded.height, margins.bottom]);
```

4. Drawing our points.

Draw our points!

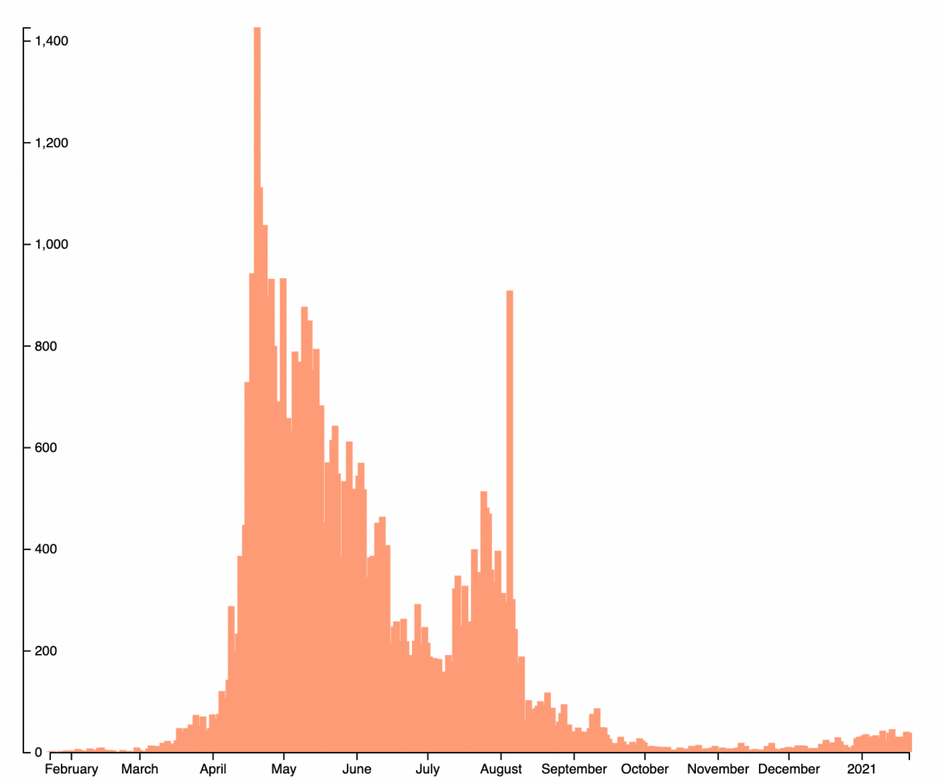
- We insert the points into the group container that we created earlier.

```
group.selectAll('rect')
    .data(dataset)
    .enter()
    .append('rect')
    .attr('x', d => ((xScaleTime(new Date(d['date'])) - (barWidth / 2))))
    .attr('y', d => (yScale(parseInt(d['new_cases']))))
    .attr('width', barWidth)
    .attr('height', (d) => {
        return (bounded.height - yScale(d['new_cases']))
    })
    .style('fill', 'lightsalmon')
```

5. Draw other stuff

Yay we’re done!

- Now we have some cool looking bar chart.
- Try adding the title on your own.



Draw other stuff

- Same as before.

Making maps

Making maps

- Today we'll try out making maps with d3.js.
- This will allow you to try out new forms of visualisation.
- Playing around with GeoJSON.

What is GeoJSON?

- GeoJSON is a format for encoding a variety of geographic data structures.
- Same as JSON in JavaScript, subset of JSON notation.
  - Structured dictionary.
  - Specified in RFC 7946 as the GeoJSON Specification.

Steps Involved

- To create a map, we have to take a look at the raw GeoJSON file, which encodes features such as Point, LineString, etc.
- Same steps as in Part I.

    1. Loading the dataset.
    2. Creating the drawing area (bounding box).
    3. Creating a scale.
    4. Drawing our points.
    5. Creating any additional (legends, axes, etc.).

Geospatial Visualisations

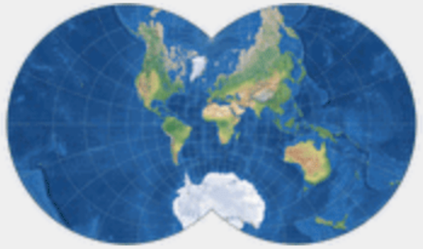
- Entirely different kind of data.
- Uses **GeoJSON** or **KML** files.
  - Encodes geometric features, including Points, Lines, Polygons, and so on.
- We have to use **projections** to transform GeoJSON data into coordinates on the screen.

Understanding Map Projections

- The central question: how do we flatten a 3D spherical globe into a 2D plane?
- Systematic transformation of latitudes and longitudes into Cartesian coordinates on a plane.
- It can get pretty... complicated.

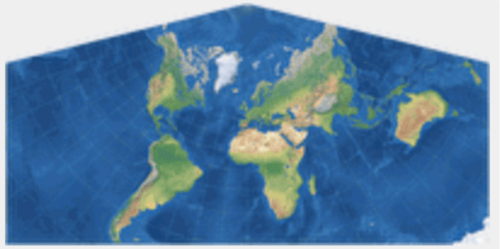
☐ Kharchenko-Shabanova

☐ Lagrange

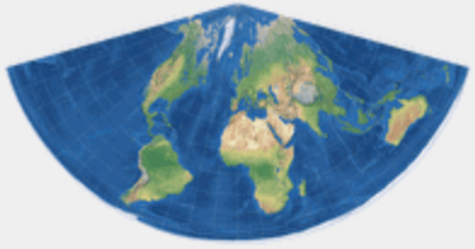


☐ Lagrange (120°)

☐ Lambert Cylindrical



☑ Lambert CC



☐ Lambert Equal-Area Conic

☐ Larrivée

☐ Laskowski Tri-Optimal

☐ McBryde P3

☐ McBryde Q3

☐ McBryde S2

☐ McBryde S3

☑ McBryde S3 (i.)

☐ McBryde-Thomas #1

☐ McBryde-Thomas #2

☐ McBryde-Thomas FPP

☐ McBryde-Thomas FPQ

☐ McBryde-Thomas FPS

☐ McBryde-Th. FPQ (i.)

☐ Mercator

Geospatial Visualisations

- D3 has built-in projection functions. It also has various external plugins that we can use.
- We have to set properties for each projection:
  - Scale - sets scale of the map (1 = smallest)
  - Rotate - sets longitude of the projection.
  - Center - sets single standard parallel.
  - Translate - pixel offset (ensures the centre of the projection is in the centre of the viewing area)

Maps

Let's give it a shot!

Workshop 3 Part 2

Our Data

- National Map Polygon, provided by the Singapore Land Authority (SLA).
- Dataset available at data.gov.sg:
  - https://data.gov.sg/dataset/national-map-polygon
- Things to take note:
  - Data format - KML or GeoJSON (choose GeoJSON)
  - Number of columns - many
  - Data types - geographic coordinates (mainly)

1. Load the dataset

Load the dataset

- GeoJSON is treated in the same way as JSON in d3.js.

```
d3.json('response.json')
.then((dataset) => {
```

Load the dataset

- OK. But it can get complicated. Our dataset is not formatted properly.
- So we reformat it!

```
let geoJson = JSON.parse(geoJsonObject.geojson)
features.push({
    type: 'Feature',
    properties: {
        Name: geoJsonObject.pln_area_n,
        dataPoint: dataPoint
    },
    geometry: geoJson
})
```

2. Create the drawing area

3. Creating our scales.

Create Scales

- A scale in geographic system is a projection.
- Thankfully d3.js has in-built functions for this.

```
const projection = d3.geoMercator()
    .scale(60000)
    .center([104, 1.4])

const path = d3.geoPath().projection(projection)
```

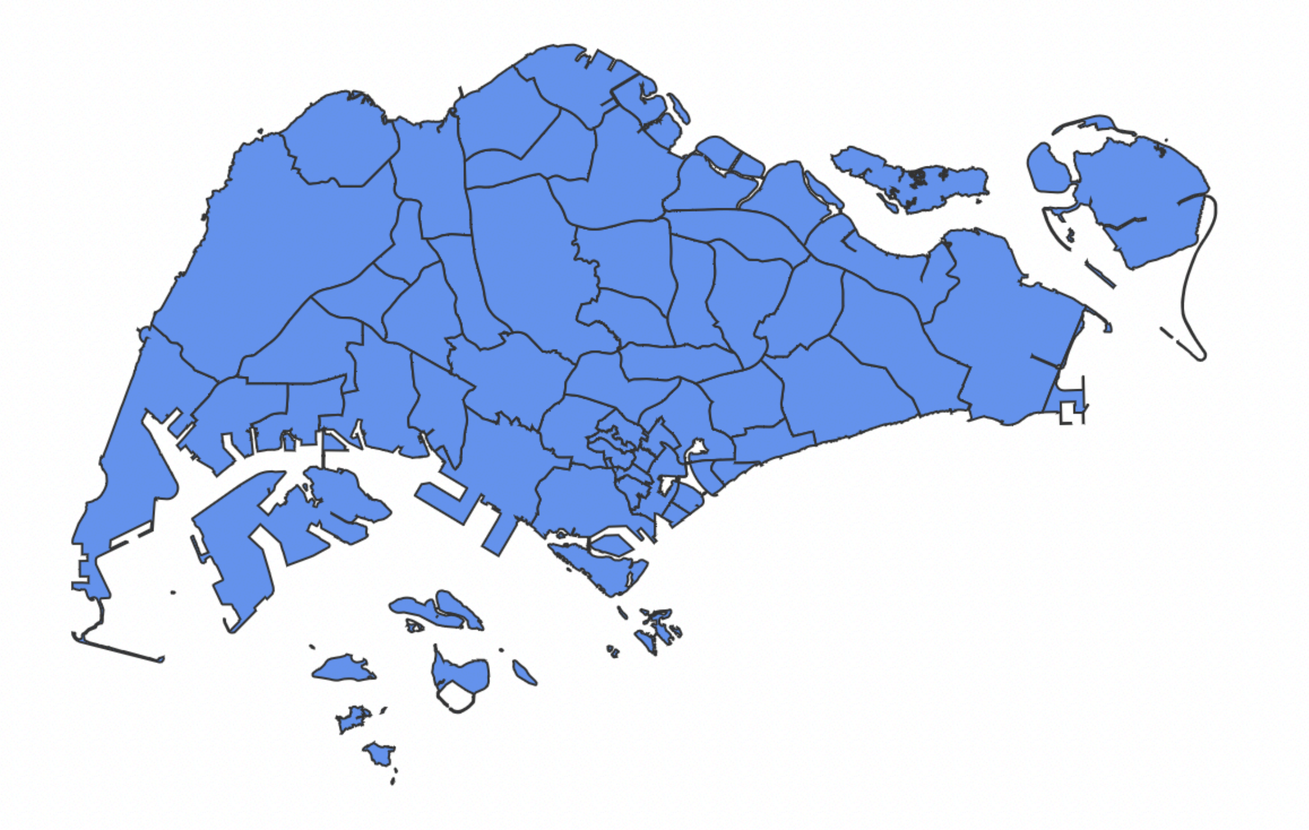
4. Drawing our points.

Select and Bind

- We need to use the ‘d’ attribute to draw the map.
- The ‘d’ attribute represents an SVG path.

```
group.selectAll('path')
     .data(features)
     .enter()
     .append('path')
     .attr('d', path)
     .attr('fill', 'cornflowerblue')
     .attr('stroke', '#333')
).
```

We’re done!



Advanced stuff

Layering

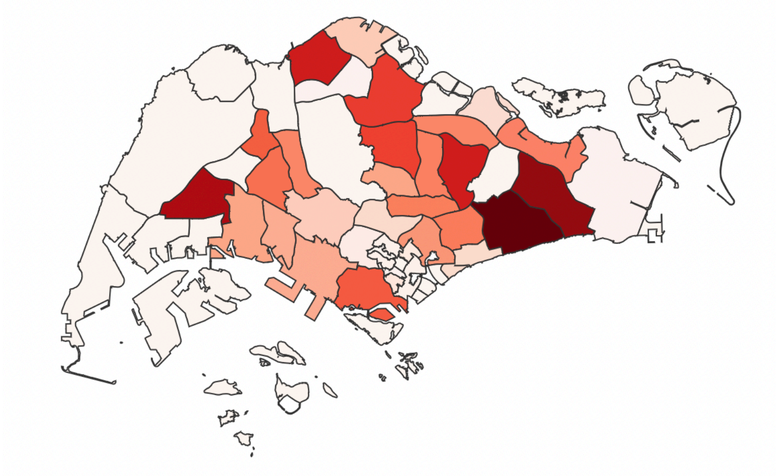
Creating Information *on top of* Maps

Workshop 3 Part 3

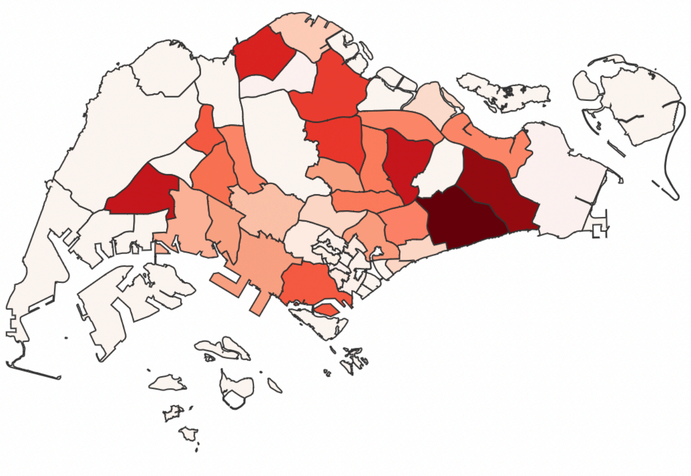
Advanced stuff (Part 3)

- Let’s try to layer some interesting information on top of our base map.
- This needs some advanced stuff, like getting an API token.
- **You need to register for a token to use the map properly.**

Singles in 2010

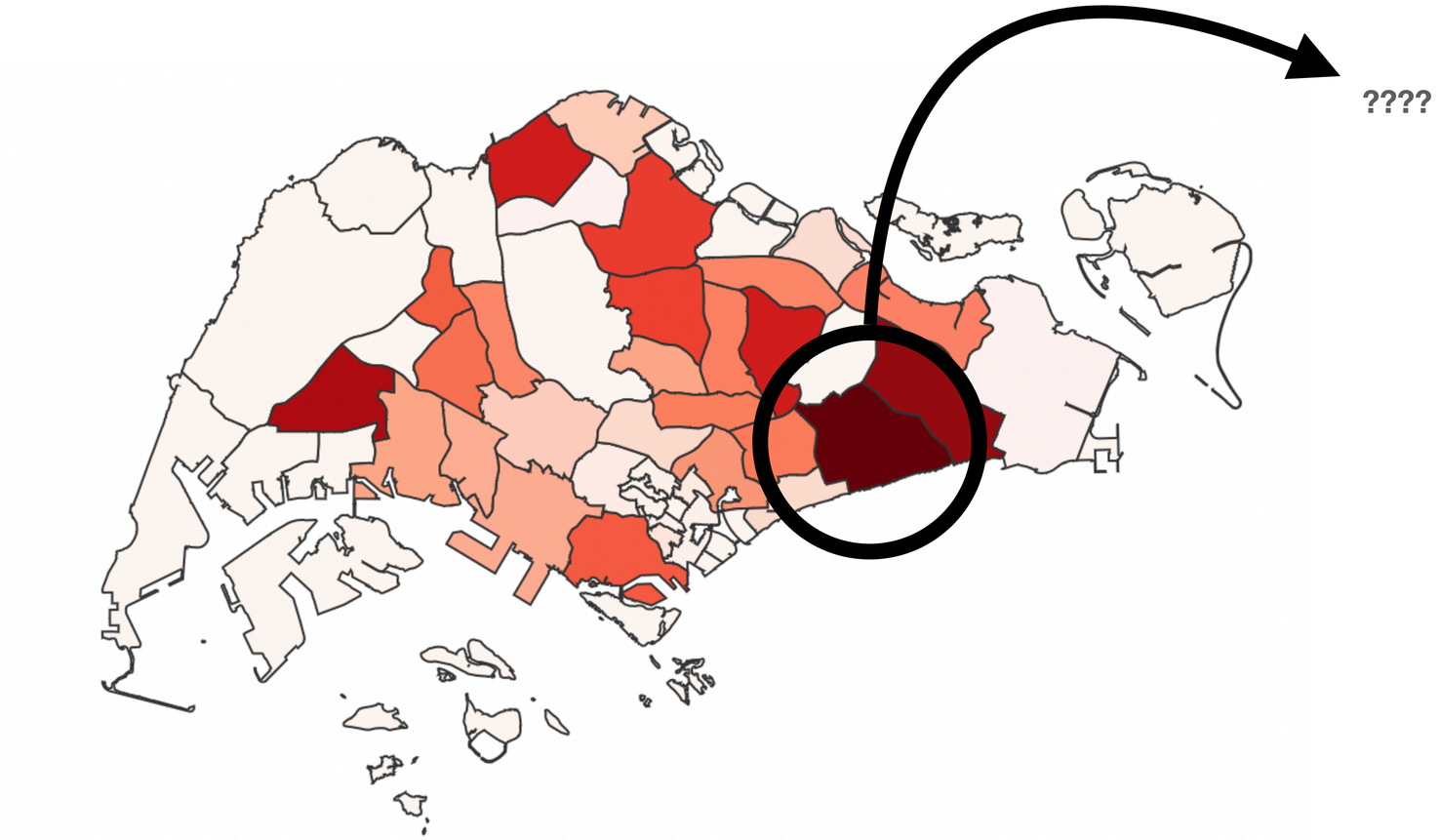


Males



Females

Singles in 2010



What is D3?

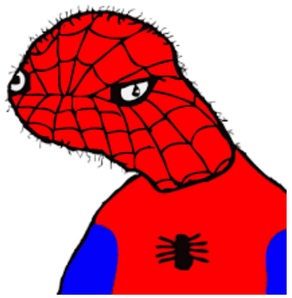
- A powerful library that allows us to declaratively manipulate the DOM.
- Arguably the **best open-source visualisation package for the web, …but**

- Does NOT come with pre-made charts.

Alternatives to D3

Alternatives to D3

- There are many alternatives to D3.
- Open-source libraries and proprietary software.
- Remember what we learnt at the start of the workshop.
  - D3 **is not a charting library**.
  - D3 provides you with the **tools to make charts**, but you have to make those charts on your own.
  - With great power, comes great responsibility.



Alternatives to D3

- Why you will want to use D3:
  - Customise your own visualisations.
  - Exercise more control and fine-tune a visualisation.
  - Data pre-processing using non-D3, JavaScript libraries (note that D3 has limited data cleaning capabilities)

Alternatives to D3

- Why you will NOT want to use D3:
  - Overpowered and complicated, like using a bazooka.
  - “Steep” learning curve.

Alternatives to D3

- Open-source alternatives
  - University of Washington - Vega and VegaLite
  - Uber - Vis.gl framework
  - Chart.js
- Proprietary alternatives
  - Tableau
  - ESRI ArcGIS (only for mapping)

Geospatial Visualisations

- There are many libraries other than D3 that offer mapping capabilities:
  - Leaflet.js (most well-known one)
  - Cesium
  - Turf.js is used for geospatial analysis.
  - Service providers:
    - Google Maps
    - OpenLayers, OpenStreetMap
    - MapBox

Resources online

Resources for D3.js

- ObservableHq — created by Mike Bostock!
  - https://observablehq.com/@d3/learn-d3 (good tutorial)
- D3 Graph Gallery
- D3.js for the Impatient by Philipp K. Janert
- Nathan Vander Wilt's talk on 'building apps with d3' [2013]
  - https://www.youtube.com/watch?v=hhSlX_r7GZA

Thank you!