Map

Data Visualization

Three Key Concept of Map in D3

- GeoJSON
  - A json-based format for specifying geographic data
  - D3 creates map based on GeoJSON data
- Map projections
  - Functions that convert from latitude/longitude coordinate to x and y coordinates
- Geographic path generators – **d3.geoPath()**
  - Functions that convert GeoJSON shapes into SVG "paths"
  - Similar to shape generator e.g. d3.line(), d3.area(), etc.
  - Note: d3.geoPath() only recognize **WGS84** geodetic system. Make sure the file you import uses WGS84

GeoJSON

- A JSON-based format for specifying geographic data
- A segment of GeoJSON data for Taipei City in “taiwan.json”
  - You can find “taiwan.json” in Ex06-01 folder

Properties: name, id, and other attributes of the region


```
{"type":"Feature","properties":{"COUNTYID":"63","NAME_1984":"臺北市","NAME_1984_ALIAS":"台北市","NAME_2010":"臺北
市","NAME_2010_ALIAS":"台北市","NAME_2014":"臺北市","NAME_2014_ALIAS":"台北
市","ISO3166":"TPE","SEGIS_COUNTY_ID":"63000","AREA_ID":"A","_id":18744641},"geometry":
{"type":"Polygon","coordinates":[[[121.570980505,25.197168115],[121.570972325,25.197035112],
[121.570933739,25.196892124],[121.570916396,25.196824628],[121.570896707,25.196769777],[121.570873123,25.196706945],
[121.570838015,25.196631625],[121.570810569,25.196563093],[121.570794274,25.196480668],[121.570809417,25.196392146],
[121.570871837,25.196258319],[121.57093985,25.196163419],[121.571050807,25.196073786],[121.571103112,25.196027289],
[121.571178172,25.19598511],[121.571279499,25.195927707],[121.571360862,25.19588545],[121.571414521,25.195845818],
[121.571479026,25.195770484],[121.571527208,25.195699925],[121.571567938,25.195637512],[121.571599922,25.195580893],
[121.571661955,25.195509035],[121.571694973,25.195436367],[121.5717422,25.195387625],[121.571887893,25.195346915],
[121.572022694,25.195338467],[121.572156089,25.195319697],[121.572267904,25.195290863],[121.572404217,25.195211251],
[121.572506625,25.195141223],[121.572623032,25.195080196],[121.572711925,25.195035561],[121.572904134,25.194985127],
[121.573071745,25.194978591],[121.573180199,25.194979625],[121.573264975,25.19500044],[121.573344693,25.195020172],
[121.57342179,25.195033045],[121.573534334,25.195055829],[121.57359519,25.195079218],[121.573724591,25.195135087],
[121.573824513,25.195156862],[121.573918002,25.195169545],[121.574003551,25.195155933],[121.574094063,25.195136515],
[121.574160911,25.195138034],[121.574246998,25.195162264],[121.574470178,25.195251468],[121.574590627,25.195298253],
```


Coordinates **(latitude/longitude)** of the contour of the region)

We have to map latitude and longitude to x y coordinates in pixel to draw them on the browser

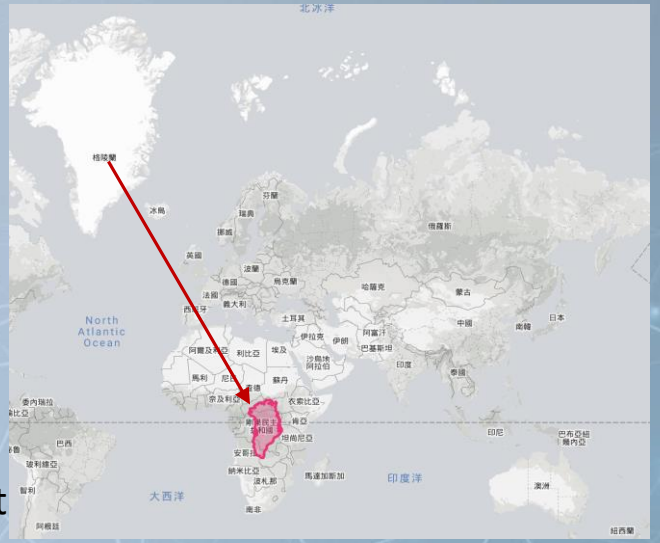
TopoJSON

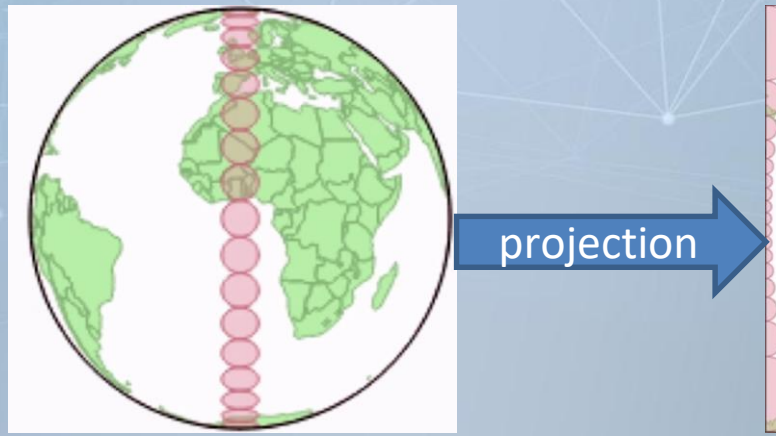
- Comparing with GeoJSON, TopoJSON is smaller

- To load and use topoJSON file to draw map
  - You need https://d3js.org/topojson.v3.min.js to convert topojson format to geoJson, then send to data to d3.geoPath
  - We do not use or introduce topoJSON in details in this lecture

Map Projection

- Function that covert from latitude/longitude coordinates to x and y coordinates
- Why?
  - "The true size of" website
    - https://thetruesize.com/
  - Our earth is a sphere. If we want to map it to a rectangle, the distortion must exist

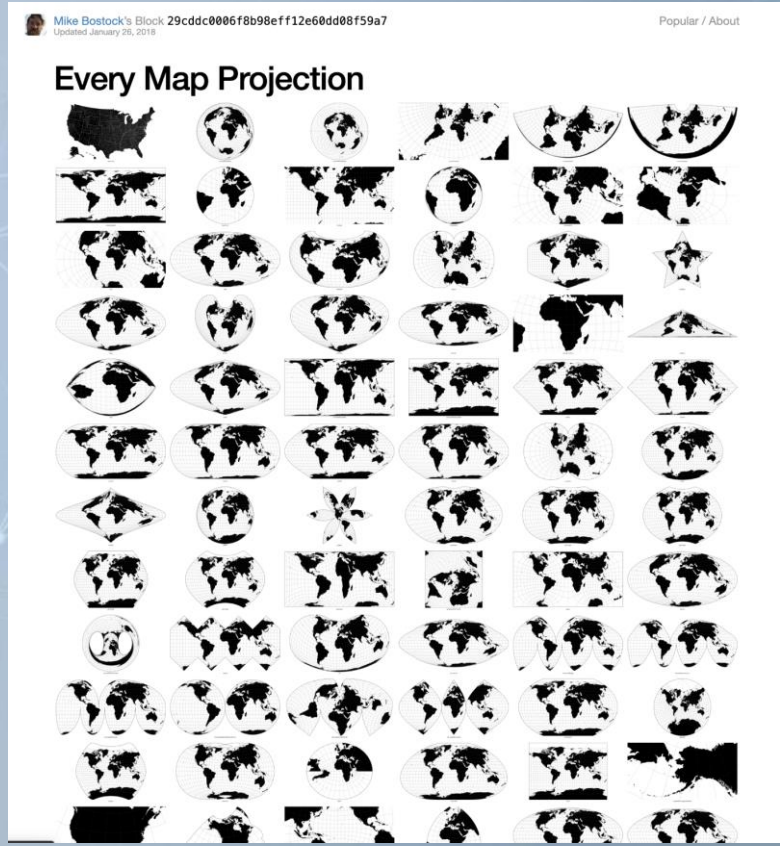




Map Projection

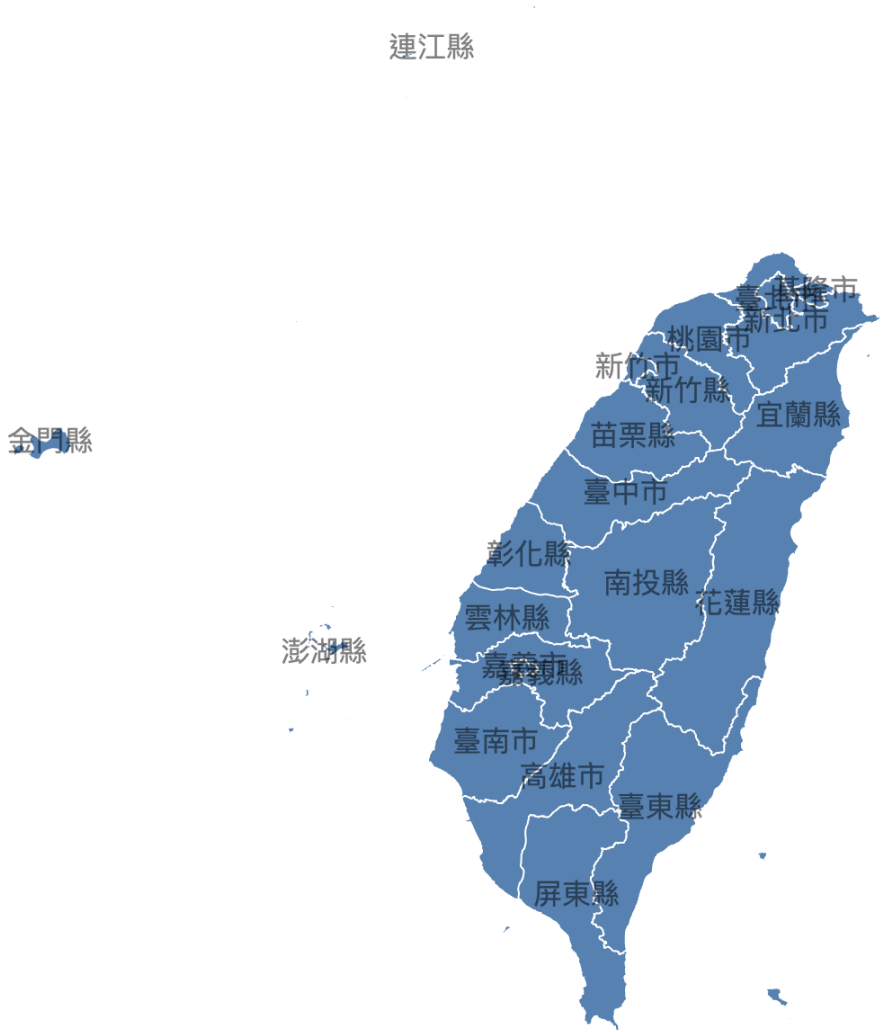
- D3 supports various map projection
  - https://github.com/d3/d3-geo-projection
  - Every projection will distort shape, area, distance and/or direction. Every projection also try to keep some attributes have as less distortion as possible
  - Think what properties you do not want to be distorted in your visualization. Then, choose a projection which has as less as possible distortion on these properties

https://bl.ocks.org/mbostock/29cddc0006f8b98eff12e60dd08f59a7



Ex08-01

- Load ”taiwan.json” and draw city/county with their name



Ex06-01

- In this example, we use d3.geoEquirectangular() as our projection function
  - .fitExtent(extent, GeoJSON)
    - The specified region will be scaled to fill the extent on the screen
  - [[0,0], [width, height]]:
    - map the top-left latitude/longitude in the GeoJSON to [0,0] and bottom-right latitude/longitude in the GeoJSON to [width, height]

Load taiwan.json. If the loading is done, run drawTaiwan()

Content of taiwan.json

```
d3.json("taiwan.json").then(drawTaiwan);

function drawTaiwan(taiwan) {
  var width = 1000;
  var height = 800;

  var projection = d3.geoEquirectangular()
      .fitExtent([[0,0], [width, height]], taiwan);

  var geoGenerator = d3.geoPath()
      .projection(projection);

  var paths = d3.select('svg')
      .selectAll('path')
      .data(taiwan.features)
      .enter()
      .append('path')
      .attr('stroke', "white")
      .attr('fill', 'steelblue')
      .attr('d', geoGenerator);

  var texts = d3.select('svg')
      .selectAll('text')
      .data(taiwan.features)
      .enter()
      .append('text')
      .attr('text-anchor', 'middle')
      .attr('alignment-baseline', 'middle')
      .attr('opacity', 0.5)
      .text(function(d) {
        return d.properties.NAME_2014;
      })
      .attr('transform', function(d) {
        var center = geoGenerator.centroid(d);
        return 'translate (' + center + ')';
      });
}
```

Ex06-01

- In this example, we use d3.geoEquirectangular() as our projection function
  - .fitExtent(extent, GeoJSON)
    - The specified region will be scaled to fill the extent on the screen
  - [[0,0], [width, height]]:
    - map the top-left latitude/longitude in the GeoJSON to [0,0] and bottom-right latitude/longitude in the GeoJSON to [width, height]
- Use “d3.geoPath()” to create the generator
  - Remember to set the projection function to it by .projection()

Load taiwan.json. If the loading is done, run drawTaiwan()

Content of taiwan.json

```
d3.json("taiwan.json").then(drawTaiwan);

function drawTaiwan(taiwan) {
  var width = 1000;
  var height = 800;

  var projection = d3.geoEquirectangular()
      .fitExtent([[0,0], [width, height]], taiwan);

  var geoGenerator = d3.geoPath()
      .projection(projection);

  var paths = d3.select('svg')
      .selectAll('path')
      .data(taiwan.features)
      .enter()
      .append('path')
      .attr('stroke', "white")
      .attr('fill', 'steelblue')
      .attr('d', geoGenerator);

  var texts = d3.select('svg')
      .selectAll('text')
      .data(taiwan.features)
      .enter()
      .append('text')
      .attr('text-anchor', 'middle')
      .attr('alignment-baseline', 'middle')
      .attr('opacity', 0.5)
      .text(function(d) {
        return d.properties.NAME_2014;
      })
      .attr('transform', function(d) {
        var center = geoGenerator.centroid(d);
        return 'translate (' + center + ')';
      });
}
```

Ex06-01

- Use taiwan.features as the data to draw the paths

Each element in the data array (Taiwan.features) will be sent to “geoGenerator” to generator the path descriptor

```
d3.json("taiwan.json").then(drawTaiwan);

function drawTaiwan(taiwan) {
  var width = 1000;
  var height = 800;

  var projection = d3.geoEquirectangular()
        .fitExtent([[0,0], [width, height]], taiwan);

  var geoGenerator = d3.geoPath()
        .projection(projection);

  var paths = d3.select('svg')
        .selectAll('path')
        .data(taiwan.features)
        .enter()
        .append('path')
        .attr('stroke', "white")
        .attr('fill', 'steelblue')
        .attr('d', geoGenerator);

  var texts = d3.select('svg')
        .selectAll('text')
        .data(taiwan.features)
        .enter()
        .append('text')
        .attr('text-anchor', 'middle')
        .attr('alignment-baseline', 'middle')
        .attr('opacity', 0.5)
        .text(function(d) {
          return d.properties.NAME_2014;
        })
        .attr('transform', function(d) {
          var center = geoGenerator.centroid(d);
          return 'translate (' + center + ')';
        });
}
```

22 cities/counties (so, 22 paths)

Data to generate a path

Ex06-01

Use Taiwan.features as the data to draw the paths

geoGenerator looks for data in the attribute with the name ‘”geometry”’ to generate the path

taiwan.features

```
▼Array(22) i
  ▼0:
    ▼geometry:
      ▼coordinates: Array(183)
        ▶[0 … 99]
        ▶[100 … 182]
          length: 183
        ▶__proto__: Array(0)
      type: "MultiPolygon"
      ▶__proto__: Object
    ▼properties:
        AREA_ID: "Z"
        COUNTYID: "9007"
        ISO3166: "LJF"
        NAME_1984: "連江縣"
        NAME_1984_ALIAS: ""
        NAME_2010: "連江縣"
        NAME_2010_ALIAS: ""
        NAME_2014: "連江縣"
        NAME_2014_ALIAS: ""
        SEGIS_COUNTY_ID: "9007"
        _id: 18744625
      ▶__proto__: Object
    type: "Feature"
  ▶__proto__: Object
▶1: {type: "Feature", properties: {…}, geometry: {…}}
▶2: {type: "Feature", properties: {…}, geometry: {…}}
▶3: {type: "Feature", properties: {…}, geometry: {…}}
▶4: {type: "Feature", properties: {…}, geometry: {…}}
▶5: {type: "Feature", properties: {…}, geometry: {…}}
▶6: {type: "Feature", properties: {…}, geometry: {…}}
▶7: {type: "Feature", properties: {…}, geometry: {…}}
▶8: {type: "Feature", properties: {…}, geometry: {…}}
▶9: {type: "Feature", properties: {…}, geometry: {…}}
▶10: {type: "Feature", properties: {…}, geometry: {…}}
▶11: {type: "Feature", properties: {…}, geometry: {…}}
▶12: {type: "Feature", properties: {…}, geometry: {…}}
▶13: {type: "Feature", properties: {…}, geometry: {…}}
▶14: {type: "Feature", properties: {…}, geometry: {…}}
▶15: {type: "Feature", properties: {…}, geometry: {…}}
▶16: {type: "Feature", properties: {…}, geometry: {…}}
▶17: {type: "Feature", properties: {…}, geometry: {…}}
▶18: {type: "Feature", properties: {…}, geometry: {…}}
▶19: {type: "Feature", properties: {…}, geometry: {…}}
▶20: {type: "Feature", properties: {…}, geometry: {…}}
▶21: {type: "Feature", properties: {…}, geometry: {…}}
  length: 22
▶__proto__: Array(0)
```

```
d3.json("taiwan.json").then(drawTaiwan);

function drawTaiwan(taiwan) {
  var width = 1000;
  var height = 800;

  var projection = d3.geoEquirectangular()
      .fitExtent([[0,0], [width, height]], taiwan);

  var geoGenerator = d3.geoPath()
      .projection(projection);

  var paths = d3.select('svg')
      .selectAll('path')
      .data(taiwan.features)
      .enter()
      .append('path')
      .attr('stroke', "white")
      .attr('fill', 'steelblue')
      .attr('d', geoGenerator);

  var texts = d3.s
      .selectAll
      .data(taiw
      .enter()
      .append('t
      .attr('tex
      .attr('alignment-baseline', 'middle')
      .attr('opacity', 0.5)
      .text(function(d) {
        return d.properties.NAME_2014;
      })
      .attr('transform', function(d) {
        var center = geoGenerator.centroid(d);
        return 'translate (' + center + ')';
      });
}
```

Each element in the data array (Taiwan.features) will be sent to “geoGenerator” to generator the path descriptor

Ex06-01

Put the cities/conties name on the map

taiwan.features

```
▼Array(22) ℹ
  ▼0:
    ▼geometry:
      ▼coordinates: Array(183)
        ▶[0 … 99]
        ▶[100 … 182]
          length: 183
        ▶__proto__: Array(0)
        type: "MultiPolygon"
      ▶__proto__: Object
    ▼properties:
        AREA_ID: "Z"
        COUNTYID: "9007"
        ISO3166: "LJF"
        NAME_1984: "連江縣"
        NAME_1984_ALIAS: ""
        NAME_2010: "連江縣"
        NAME_2010_ALIAS: ""
        NAME_2014: "連江縣"
        NAME_2014_ALIAS: ""
        SEGIS_COUNTY_ID: "9007"
        _id: 18744625
      ▶__proto__: Object
      type: "Feature"
    ▶__proto__: Object
  ▶1: {type: "Feature", properties: {…}, geometry: {…}}
  ▶2: {type: "Feature", properties: {…}, geometry: {…}}
  ▶3: {type: "Feature", properties: {…}, geometry: {…}}
  ▶4: {type: "Feature", properties: {…}, geometry: {…}}
  ▶5: {type: "Feature", properties: {…}, geometry: {…}}
  ▶6: {type: "Feature", properties: {…}, geometry: {…}}
  ▶7: {type: "Feature", properties: {…}, geometry: {…}}
  ▶8: {type: "Feature", properties: {…}, geometry: {…}}
  ▶9: {type: "Feature", properties: {…}, geometry: {…}}
  ▶10: {type: "Feature", properties: {…}, geometry: {…}}
  ▶11: {type: "Feature", properties: {…}, geometry: {…}}
  ▶12: {type: "Feature", properties: {…}, geometry: {…}}
  ▶13: {type: "Feature", properties: {…}, geometry: {…}}
  ▶14: {type: "Feature", properties: {…}, geometry: {…}}
  ▶15: {type: "Feature", properties: {…}, geometry: {…}}
  ▶16: {type: "Feature", properties: {…}, geometry: {…}}
  ▶17: {type: "Feature", properties: {…}, geometry: {…}}
  ▶18: {type: "Feature", properties: {…}, geometry: {…}}
  ▶19: {type: "Feature", properties: {…}, geometry: {…}}
  ▶20: {type: "Feature", properties: {…}, geometry: {…}}
  ▶21: {type: "Feature", properties: {…}, geometry: {…}}
    length: 22
  ▶__proto__: Array(0)
```

```
d3.json("taiwan.json").then(drawTaiwan);

function drawTaiwan(taiwan) {
  var width = 1000;
  var height = 800;

  var projection = d3.geoEquirectangular()
        .fitExtent([[0,0], [width, height]], taiwan);

  var geoGenerator = d3.geoPath()
        .projection(projection);

  var paths = d3.select('svg')
        .selectAll('path')
        .data(ta
        .enter(
        .append
        .attr('
        .attr('fill', 'steelblue')
        .attr('d', geoGenerator);

  var texts = d3.select('svg')
        .selectAll('text')
        .data(taiwan.features)
        .enter()
        .append('text')
        .attr('text-anchor', 'middle')
        .attr('alignment-baseline', 'middle')
        .attr('opacity', 0.5)
        .text(function(d) {
          return d.properties.NAME_2014;
        })
        .attr('transform', function(d) {
          var center = geoGenerator.centroid(d);
          return 'translate (' + center + ')';
        });
}
```

Calculate centroid of the cities/counties to place the name

Ex06-01

- Convert longitude and latitudes to x-y and draw a circle
- Send longitude/latitudes to the projection function. It returns an array with x and y.
  - 120.9575, 23.47: Yushan (玉山)

```
d3.select('svg').append('circle')
    .attr('cx', projection( [120.9575, 23.47 ])[0] )
    .attr('cy', projection( [120.9575, 23.47 ])[1] )
    .attr('fill', 'red')
    .attr('r', 5);
```