GEOGRAPHIC VISUALIZATIONS

InfoVis

GEOGRAPHIC VISUALIZATIONS

Concepts, tools and basic map visualizations

Mireia Ribera

Data Science Master Degree

Geographic concepts

Projection and scale

When we want to visualize a map, we must decide at what level of detail (the scale) and with which projection we will draw it. It is also important to know the date.

The projection consists in how to convert the 3D geographic information into 2D. Some example projections are:

Figure 1. Albers USA -- .project(type='albersUsa') --, specific for USA. Source: https://altair-viz.github.io/gallery/world_projections.html

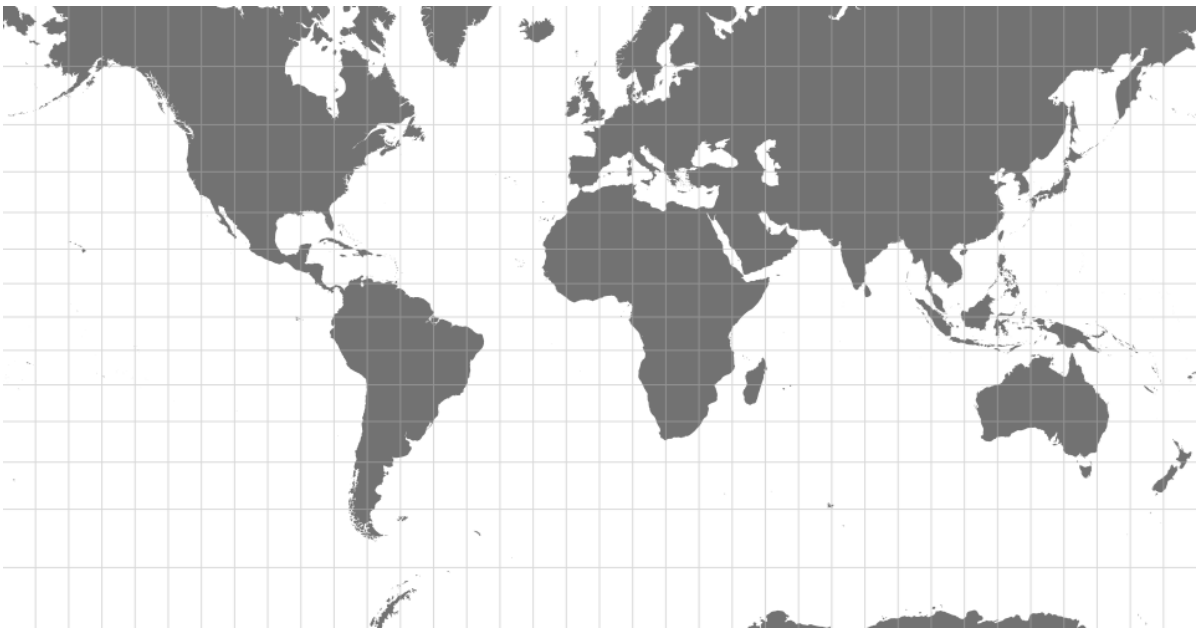

Figure 2. Mercator -- .project(type='mercator') --, or UTM, for most of the world. Source: https://altair-viz.github.io/gallery/world_projections.html

The level of detail is indicated by the source map as a proportion between a distance on a map and the corresponding distance on the ground. A scale of 500.000 means that every cm in the map describes 5 km in the real ground. Smaller scales will show more detail but will create bigger files.

Another important aspect is to use a map dated the same year as the data, because country or region boundaries may slightly vary on time.

Digital map file formats

There are three main digital file formats for maps: shapefiles, geojson and topojson.

Shapefiles

This is the standard format for professional GIS. It is a vectorial graphic format that includes polygons, lines and points. Apart from the geographical information, shapefiles include metadata on names, types, relations… that will be useful for visualizations, and in fact they come as a bunch of related files, the most important being the .shp file with the geographical information and the .dbf file with the metadata. This format was created by ESRI (Environmental Systems Research Institute), a USA consulting firm created in 1969 and that now commercializes ArcGIS, the most widely used professional GIS software.
Shapefiles can include information as longitude and latitude or they can be projected already. If this is the case, you must be careful to change the projection if that is needed.

GeoJSON

(http://geojson.org)

This is a JSON format to draw maps on the web. It has the following structure:

FeatureCollection: grouping of all the objects
- Feature: each object (this will be our map unit and can be a country, a state, a county…)
  - properties: feature metadata such as name, inhabitants…
  - geometry: geographical data
  - id: (optional) unique identifier

TopoJSON

(http://github.com/topojson/topojson)
Source: Adaptation of Mike Bostock, command-line cartography https://medium.com/@mbostock/command-line-cartography-part-3-1158e4c55a1e

TopoJSON is a variant of GeoJSON created by Mike Bostock that reduces the file size by simplifying, quantizing and compressing the information.

It has the following structure:

Topology
- transform
- objects
  - *ObjectName*
    - GeometryCollection: equivalent to FeatureCollection
      - geometries: each object (our map unit)
        - type: polygon, line or dot
        - arcs: link to the topological information (slice of arcs array)
        - properties: geometries metadata such as name, inhabitants…
  - arcs: array with path descriptions

TopoJSON represents lines and polygons as sequences of arcs rather than sequences of coordinates. Contiguous polygons have shared borders whose coordinates are duplicated in GeoJSON. By representing lines and polygons as sequences of arcs, repeating an arc does not require repeating coordinates, and the file size is compressed. TopoJSON can be quantized and represent coordinates by small integers instead of floating-point values (say 224.3021507 into 224), and then delta-encoded such that each successive x- and y- value is relative to the previous one. Quantization is expressed in powers of 10 like 1e6 (1000000) , 1e4 (10000) …

```
{ "type": "Topology",
  "transform": { "scale": [1,1], "translate": [0,0] },
  "objects": {
    "two-squares": {
      "type": "GeometryCollection",
      "geometries": [
        { "type": "Polygon",
          "arcs": [[0,1]],
          "properties": {"name": "left"}},
        { "type": "Polygon",
          "arcs": [[2,-1]],
          "properties": {"name": "right"}}
  ]}},
  "arcs":[
    [[1,2],[0,-2]],
    [[1,0],[-1,0],[0,2],[1,0]],
    [[1,2],[1,0],[0,-2],[-1,0]]
]}
```

Figure 3. Code of two lines in TopoJSON

The first thing to take into account is that the coordinate system starts bottom left, and increases upwards and rightwards. Then the arc [1,2],[0,-2] is the black one, that starts at position 1,2, and then draws a line at the same x (0), and decreasing the y in 2 (-2), reaching [1,0]. The arc [1,0],[-1,0],[0,2][1,0] is the blue one, that starts at position [1,0], draws a line til the point that decreases the x in 1, same y, reaching [0,0]; then same x, increases the y in 2, reaching [0,2]; then x+1, same y [1,2]

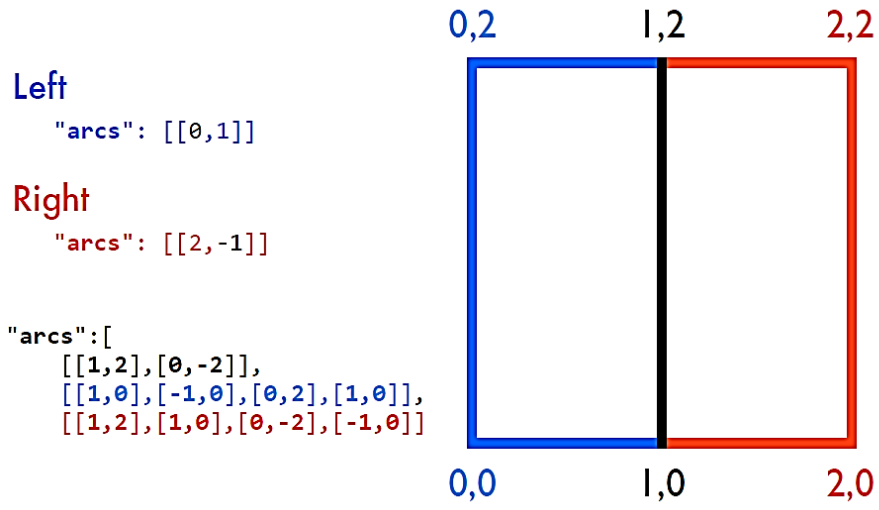

Figure 4. The compression and delta encoding exemplified

Referring these arcs are done by their indexes, arcs 0,1 means the first and the second. Arcs 2,-1 refers the third and-1 refers to the reversed first arc (from 1,0 to 1,2 [1,0][0,2]), taking into account that when more than arc is referenced,

the first position of a subsequent arc must be equal to the last position of the previous. Then the final sequence is : [1,2][1,0][0,-2][-1,0][0,0][0,2]

Original sequence of points in GeoJSON	Quantized points in TopoJSON (transformed by scale, and translation)	Delta-encoded
[545.7796789342211, 348.96136952241613]	[0, 403]	[0, 403]
[545.9825061954095, 349.29419494812123]	[231, 741]	[231, 338]
[546.3281879653109, 349.53210438248560]	[625, 982]	[394, 241]
[546.3147336879572, 348.77969898749300]	[610, 219]	[-15,-763]
[546.5844757927035, 348.76960903081610]	[917, 208]	[307, -11]
[546.5889751031176, 348.76842131978400]	[922, 207]	[5, -1]

You can also simplify the detail of the map to reduce file size. See Visvalingham algorithm in action.

Figure 5. Effects of oversimplification

In Altair (as it uses vega-lite coming from d3) we will use TopoJSON file format, as it is smaller in size and easily displayed on the web. Tableau accepts Shapes, GeoJSON and TopoJSON

Maps in Tableau

The files MapExercise, MapExercise2 and MapExercise3 show the basic possibilities of drawing maps with Tableau

Maps in Altair

The file Maps.ipynb shows the basic possibilities of drawing maps with Altair

Geographic data sources

Most countries publish their geographic information as open data, but sometimes you need to dig a little bit to find the correct file.

Some example sources are:
USA Census bureau https://www.census.gov/geographies/mapping-files.html
Diva-GIS program with http://diva-gis.org/Data with free country level data and geographic information on rivers, mountains, inhabitants, climate…
Anychart has also a good collection of maps in Geojson, Topojson, and Shp formats
https://www.anychart.com/download/cdn/?v=8.7.0 (GeoData)

Tools

QGIS or Mapshaper

In order to have a preview of our maps we recommend to use the online tool Mapshaper [http://mapshaper.org] or to install QGIS, a free open software desktop tool [http://www.qgis.org/en/site/forusers/alldownloads.html?highlight=installation].
With both tools, when visualizing a map, if you choose the "i" tool and select a region you'll be able to see the metadata for this region.
You can use QGIS to open a projected shapefile and save it as geoJSON with another projection. This is necessary if you are working with maps from Catalan government.

NPM modules

SHAPEFILE [HTTPS://GITHUB.COM/MBOSTOCK/SHAPEFILE]

Npm install –g shapefile
This is a parser created by Mike Bostock to convert shapefiles into geojson

> shp2json *file.shp* –o *file.json*

TOPOJSON SERVER [HTTPS://GITHUB.COM/TOPOJSON/TOPOJSON-SERVER]

Npm install-g topojson-server
Converts GeoJSON files to TopoJSON

> geo2topo *filegeo.json* > *filetopo.json*

You can also quantize during the conversion
> geo2topo –q *1e9 filegeo.json* > *filetopo.json*

The bigger the quantize parameter, the more detailed the map will result.

TOPOJSON SIMPLIFY [HTTPS://GITHUB.COM/TOPOJSON/TOPOJSON-SIMPLIFY]

Npm install-g topojson-simplify

>toposimplify –s value
With value between 0 and 1, the bigger the simpler

> toposimplify *filegeo.json* –s 0.3 > *filetopo.json*

We can follow all this process with Brazil shapefiles, taken from http://diva-gis.org/Data