Maps w/ D3.js

Matthew Zagaja
@mzagaja

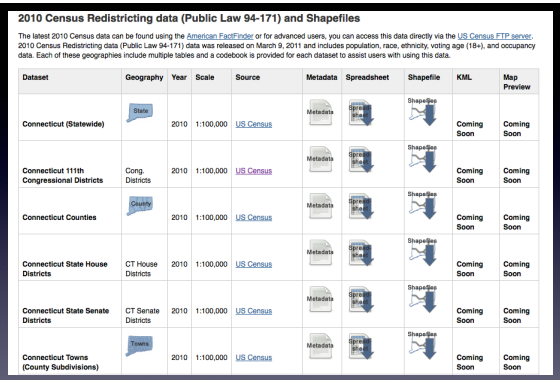
Shapefiles

Developed by ESRI and used by GIS software like ArcGIS. They spatially describe vector features and these features have attributes that describe them.

.shp - shape format is the geometry

.shx - shape index is a postiional index of the feature geometry to alow seeking forwards and backwards quickly

---



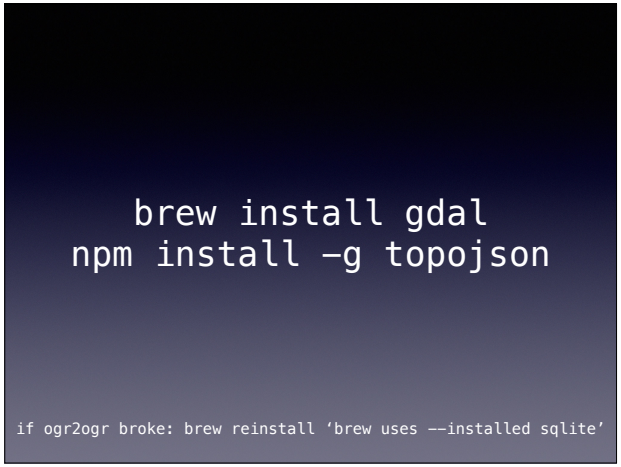
2010 Census Redistricting data (Public Law 94-171) and Shapefiles

The latest 2010 Census data can be found using the American FactFinder or for advanced users, you can access this data directly via the US Census FTP server. 2010 Census Redistricting data (Public Law 94-171) data was released on March 9, 2011 and includes population, race, ethnicity, voting age (18+), and occupancy data. Each of these geographies include multiple tables and a codebook is provided for each dataset to assist users with using this data.

Dataset	Geography	Year	Scale	Source	Metadata	Spreadsheet	Shapefile	KML	Map Preview
Connecticut (Statewide)	State	2010	1:100,000	US Census	Metadata	Spreadsheet	ShapeSep	Coming Soon	Coming Soon
Connecticut 111th Congressional Districts	Cong. Districts	2010	1:100,000	US Census	Metadata	Spreadsheet	ShapeSep	Coming Soon	Coming Soon
Connecticut Counties	County	2010	1:100,000	US Census	Metadata	Spreadsheet	ShapeSep	Coming Soon	Coming Soon
Connecticut State House Districts	CT House Districts	2010	1:100,000	US Census	Metadata	Spreadsheet	ShapeSep	Coming Soon	Coming Soon
Connecticut State Senate Districts	CT Senate Districts	2010	1:100,000	US Census	Metadata	Spreadsheet	ShapeSep	Coming Soon	Coming Soon
Connecticut Towns (County Subdivisions)	Towns	2010	1:100,000	US Census	Metadata	Spreadsheet	ShapeSep	Coming Soon	Coming Soon

http://magic.lib.uconn.edu/connecticut_data.html#2010_pl_data

Other Sources: http://www.naturalearthdata.com, census.gov



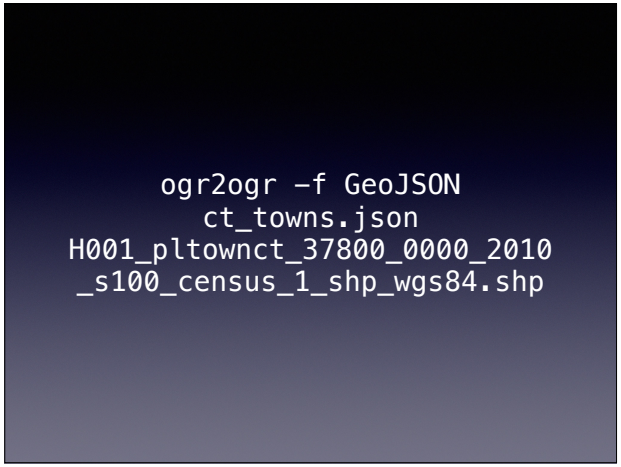
GDAL - Geospatial Data Abstraction Library. Translates data from GIS formats. Open source command line tools.

Need Node.js installed for npm install.

which ogr2ogr

TopoJSON is an extension of GeoJSON that encodes topology. TopoJSON is usually much smaller than GeoJSON. TopoJSON allows for computing boundary lines and has other interesting applications.

"ogr2ogr -f GeoJSON - where "" subunits.json shapefile.shp"



Use WGS84 version, always. It's the coordinate system that is common and will make your life easier.

```
topojson --id-property NAME10 \
  -p -o topo_ct_towns.json \
       ct_towns.json
```

https://github.com/mbostock/topojson

“TopoJSON eliminates redundancy, allowing related geometries to be stored efficiently in the same file. For example, the shared boundary between California and Nevada is represented only once, rather than being duplicated for both states. A single TopoJSON file can contain multiple feature collections without duplication, such as states and counties. Or, a TopoJSON file can efficiently represent both polygons

---

www.mapshaper.org

Smoothes boundary lines to reduce file size and rendering load. Put your TopoJSON in here, then try again. This kills the IDs FYI.

D3.js

D3.js is a JavaScript library for manipulating documents based on data. D3 allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document. For example, you can use D3 to generate an HTML table from an array of numbers. Or, use the same data to create an interactive SVG bar chart with smooth transitions and interaction.

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>

/* CSS goes here. */

</style>
<body>
<script src="http://d3js.org/d3.v3.min.js"></script>
<script src="http://d3js.org/topojson.v1.min.js"></script>
</script>
<script>

/* JavaScript goes here. */

</script>
```

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>

/* CSS goes here. */

</style>
<body>
<script src="http://d3js.org/d3.v3.min.js"></script>
<script src="http://d3js.org/topojson.v1.min.js"></script>
<script>
d3.json("topo_ct_towns.json", function(error, ct) {
  console.log(ct);
});

</script>
```

This uses D3 to load the data into the console and makes it so you can navigate it.

---

python -m SimpleHTTPServer 8008 &

```
<script>
var width = 960,
    height = 800;

var projection = d3.geo.albers()
    .rotate([75, 0])
    .center([2.2, 41.5])
    .scale(37000)
    .translate([width / 2, height / 2])
    .precision(.1);


var path = d3.geo.path()
    .projection(projection);
…
</script>
```

Scale and shape are not preserved in Albers, but distortion is minimal between standard parallels.

The primary mechanism for displaying geographic data is d3.geo.path. This class is similar to d3.svg.line and the other SVG shape generators: given a geometry or feature object, it generates the path data string suitable for the "d" attribute of an SVG path element.

The Projection

https://github.com/mbostock/d3/wiki/Geo-Projections

http://wrobstory.github.io/2013/06/creating-with-d3-dymo.html

---

```
<script>
…
var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height);

    d3.json("connecticut.json", function(error,
connecticut) {
        svg.selectAll(".subunit")
            .data(topojson.feature(connecticut,
connecticut.objects.ct_towns).features)
          .enter().append("path")
            .attr("class", function(d) { return
"subunit " + d.id; })
            .attr("d", path);
});
</script>
```

http://mbostock.github.io/d3/tutorial/circle.html

selectAll will select al elements that match the specified selector. In this case it returns an empty selection, binds the data to it, then enter makes it appear.

Scalable Vector Graphics (SVG)

- The files are generally much smaller than bitmaps, resulting in quicker download times.
- The graphics can be scaled to fit different display devices without the pixelation associated with enlarging bitmaps.
- The graphics are constructed within the browser, reducing the server load and network response time generally associated with web imagery. That is, a typically small formulaic description is sent

---

SVG code	Illustration
`<rect x="0" y="0" width="200" height="150" fill="#FFFF00" stroke="blue" stroke-width="5" />`	

http://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html

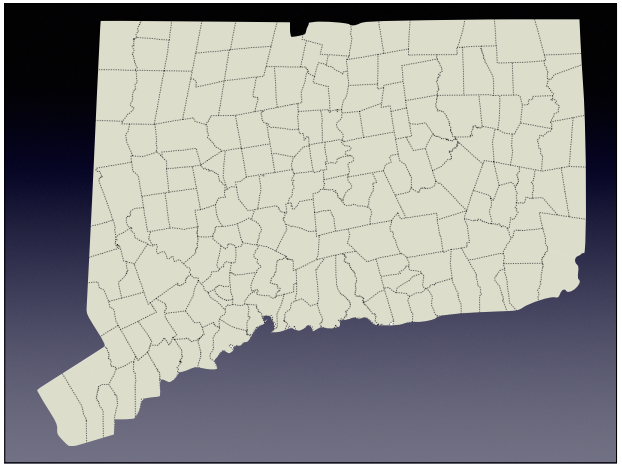
This is the picture of CT you get with that code, in SVG.

```
<script>
…
          svg.append("path")
              .datum(topojson.mesh(connecticut,
connecticut.objects.ct_towns, function(a, b) { return a
!== b && a.id !== "County subdivisions not
defined"; }))
              .attr("d", path)
              .attr("class", "subunit-boundary");
</script>
```

We'll use topojson.mesh to compute the boundaries from the topology. This requires two arguments: the topology and a constituent geometry object. An optional filter can reduce the set of returned boundaries, taking two arguments a and b representing the two features on either side of the boundary. For exterior boundaries such as coastlines, a and b are the same. Thus by filtering on a === b or a !== b, we obtain exterior or interior boundaries exclusively.

```
.subunit {
  fill: #ddc;
}

.subunit-boundary {
  fill: none;
  stroke: black;
  stroke-dasharray: 2,2;
  stroke-linejoin: round;
}

.subunit.County.subdivisions.not.defined {
  display: none;
}
```

Styled in CSS.



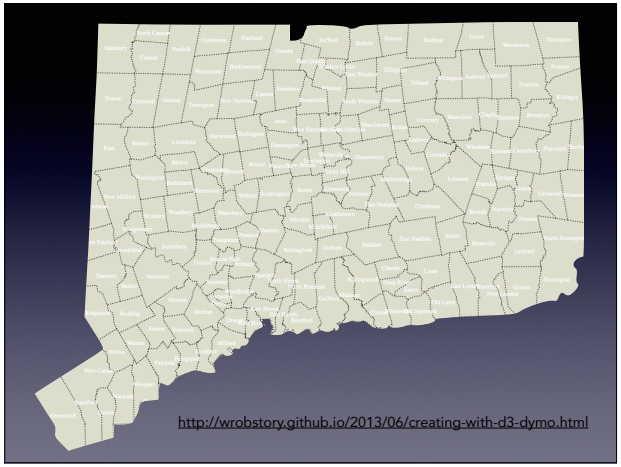
```
<script>
svg.selectAll(".subunit-label")
      .data(topojson.feature(connecticut,
connecticut.objects.ct_towns).features)
    .enter().append("text")
      .attr("class", function(d) { return "subunit-
label " + d.id; })
      .attr("transform", function(d) { return
"translate(" + path.centroid(d) + ")"; })
      .attr("dy", ".35em")
      .text(function(d) { return d.id; });
</script>
```

How to add labels.

Cheat and make “County subdivisions not defined white.

```
.subunit-label {
  fill: white;
  fill-opacity: 1;
  font-size: 10px;
  font-weight: 300;
  text-anchor: middle;
```

How to add labels.



http://wrobstory.github.io/2013/06/creating-with-d3-dymo.html

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/
2.0.3/jquery.min.js"></script>

d3.json("ct-all.json", function(error, results) {
  var ctjson = results;
  var state_data = [];
  var color =
d3.scale.linear().domain([0,1]).range(['red', 'blue'])
```

https://code.google.com/p/general-election-2008-data/source/browse/trunk/json/votes/2008/ct-all.json

Making a choropleth.

d3.json to load the json data.

we setup a ctjson to hold the results, a state data variable as an empty array, and then a color variable that will change based on the JSON.

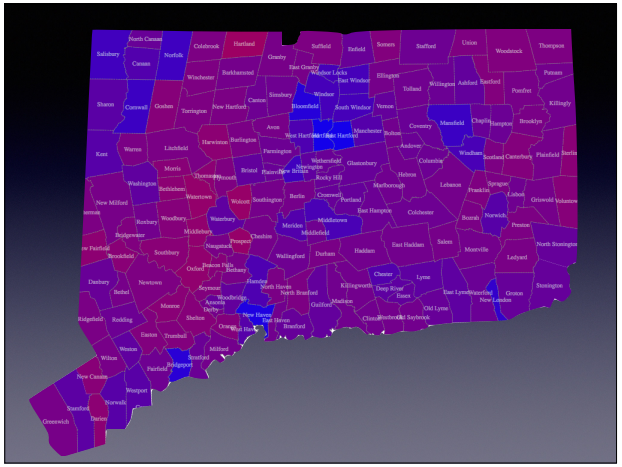
```
function colorTown(town) {
var selector = '.subunit.' +
town.name.split(' ').join('.');
d3.selectAll(selector).style("fill",
color(town.obamapercent));
}
```

```
  $.each(ctjson.locals, function(key, data) {
    var president = data.races.President[''].votes[0].id;
    var town = data.name;
    if(data.races.President[''].votes[0].id == 1918) {
      var ObamaPercent =
(data.races.President[''].votes[0].votes /
data.races.President[''].total);
      console.log(data.name + " " + ObamaPercent);
    }
    else {
      var ObamaPercent =
(data.races.President[''].votes[1].votes /
data.races.President[''].total);
      console.log(ObamaPercent + " " + data.name);
    }
    var datarow = {"name": town,
                   "president": president,
                   "obamapercent": ObamaPercent};
  state_data.push(datarow);
  });

state_data.forEach(colorTown);

});
```

https://www.dashingd3js.com/using-json-to-simplify-code

https://www.dashingd3js.com/creating-svg-elements-based-on-data

Using jQuery we go through each town (ctjson.locals) and in each town we extract the president and town names, then if Obama won we get the Obama percent by dividing the number of votes for



More D3ish way to do this:

http://bl.ocks.org/mbostock/4060606

Questions?
