http://rubenhm.org/blog/

Research, Writings + Code

Coding Annoyances

Choropleth maps with D3, Python, and data from FRED

06 July 2016 • Rubén Hernández-Murillo •

I often need to create simple thematic maps of the Fourth Federal Reserve District (https://en.wikipedia.org/wiki/Federal_Reserve_Bank), for example of the unemployment rate by county. The data are readily available from FRED (https://fred.stlouisfed.org), but using ArcView is no fun.

I found a nice tutorial by Nathan Yau (http://flowingdata.com/2009/11/12/how-to-make-a-us-county-thematic-map-using-free-tools/) that showed how to edit an SVG map with IPython using BeautifulSoup, but it required to have an SVG map to work with already.

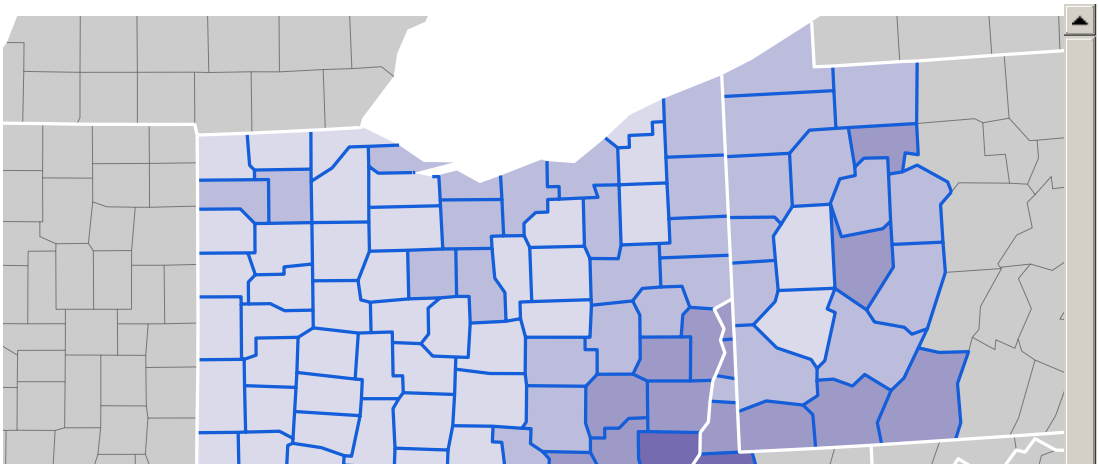
Then I found another tutorial by Mike Bostock (https://bost.ocks.org/mike/map/) that uses D3 (http://d3js.org) and TopoJSON (https://github.com/mbostock/topojson) to generate the SVG map from a script. Mike Bostock's multiple examples pages (https://bl.ocks.org/mbostock) provide tutorials for generating and manipulating all kinds of maps.

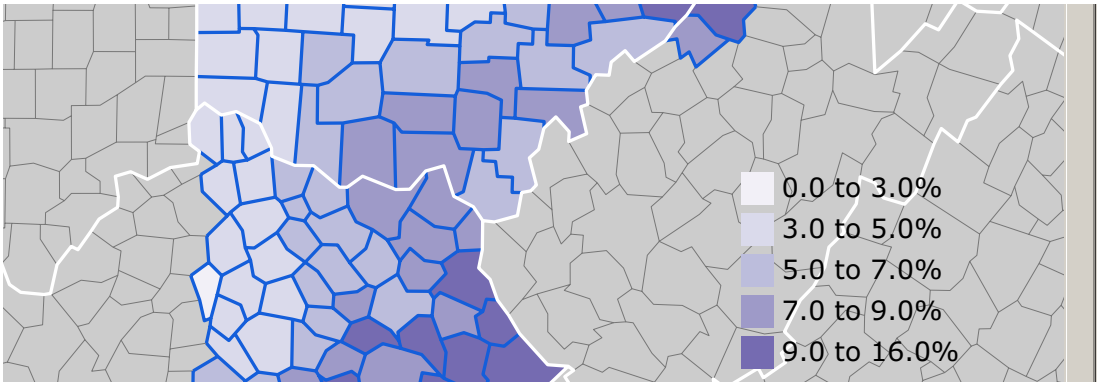
My map is based on the example for a simple Choropleth map of unemployment rates with thresholds (https://bl.ocks.org/mbostock/3306362) and the example that uses a selection of geo units is (https://bl.ocks.org/mbostock/5416405).

Map

The Fourth Federal Reserve Districts includes all of Ohio, counties in eastern Kentucky, counties in western Pennsylvania, and a few counties in the northern section of West Virginia that lies between Ohio and Pennsylvania.

Here is the final result:





Steps

1. Create a file with the definition of the selection.
2. Obtain data for each county from FRED.
3. Create the map.

Geographic definition

My file `d4ctydef.csv` (/downloads/choropleth/d4ctydef.csv) is a comma-delimited file with the FIPS (http://www.census.gov/geo/reference/codes/cou.html) codes of the counties in the District.

```
STATE,STATE_FIPS,FIPS,COUNTY_NAME,D4
KY,"21","21001",Adair County,FALSE
KY,"21","21003",Allen County,FALSE
KY,"21","21005",Anderson County,FALSE
KY,"21","21007",Ballard County,FALSE
KY,"21","21009",Barren County,FALSE
KY,"21","21011",Bath County,TRUE
KY,"21","21013",Bell County,TRUE
KY,"21","21015",Boone County,TRUE
KY,"21","21017",Bourbon County,TRUE
...
```

Downloading data from FRED

Automated downloading of data from FRED requires an API key and signing up for the service. Install the `fredapi` Python module from https://github.com/mortada/fredapi (https://github.com/mortada/fredapi) Now try the following in a Jupyter or IPython notebook.

```
# Import fredapi module
from fredapi import Fred
# Apply your personal api key
fred = Fred(api_key='abcdefghijklmnopqrstuv0123456789')

# Import pandas
import pandas as pd

# Get information on realease=116, which includes data on Unemployment in States and
Local Areas
df = fred.search_by_release(116)
df['title'].head(10) # df has information on 6,000+ series

# Restrict information to the unemployment series
state_df = df[df['title'].str.startswith('Unemployment Rate in')]

# Create a column with states by selecting the first two letters of the index, series id
# The following commands throw up an error/warning but it works
state_df['state'] = state_df.apply(lambda row: row.id[:2], axis=1)

# Restrict further to the states in the Fourth District
d4_states = state_df[state_df.state.isin(['OH','KY','PA','WV'])]

# Read District definitions to a pandas dataframe
d4def = pd.read_csv('../data/d4ctydef.csv')

# Select District counties only using the boolean variable d4def.D4
d4strict = d4def[d4def["D4"]]

# Search for county names in Fred series to obtain series id
d4_states.loc[d4_states["title"].str.contains("Adair County"),"id"]

# Get FRED id of unemployment series if county is in the District and create
# a dictionary with county fips, county name, and FRED id
d4_county_series = {}
for row in d4strict.itertuples():
        state,fips,county = row[1],row[3],row[4]
        seriesid = d4_states.loc[d4_states["id"].str.startswith(state) & d4_states["title
"].str.contains(county)]['id']
        d4_county_series[fips] = [seriesid[0], county + ', ' + state, '"' + str(fips) + '
"']

# Download the latest observation for each county and create
# a dictionary of FRED id, county fips, county name, and unemployment rate
end_time = max(d4_states.observation_end)
d4_cty_ur = {}
for fips, dictitem in d4_county_series.iteritems():
    seriesid, county, cty_fips = dictitem
    d4_cty_ur[fips] = [fips, county, fred.get_series(seriesid, observation_start=end_time
, observation_end=end_time)[0]]

# Write dictionary to tab-delimited file with header
with open('urd4.tsv', 'w') as f:
    f.write('id\tcounty\trate\n')
    [f.write('"{0}"\t"{1}"\t{2}\n'.format(fips, county, rate)) for key, [fips, county, ra
te] in d4_cty_ur.items()]
```

The file `urd4.tsv` looks like this:

```
id  county  rate
"21165" "Menifee County, KY"    8.0
"21011" "Bath County, KY"   6.6
"21013" "Bell County, KY"   8.4
"21015" "Boone County, KY"  3.7
"21017" "Bourbon County, KY"    4.6
"21019" "Boyd County, KY"   7.5
"21023" "Bracken County, KY"    5.7
...
```

Determine natural brakes in the data using the python module `jenks` from https://github.com/perrygeo/jenks (https://github.com/perrygeo/jenks).

```
# Determine cutoffs for choropleth map
from jenks import jenks

# Create list from dictionary and obtain cutoffs
# http://stackoverflow.com/questions/16228248/python-simplest-way-to-get-list-of-values-from-dict
values = [ d4_cty_ur[key][2] for key in d4_cty_ur]
cutoffs = jenks(values,4)
print cutoffs
[2.9000001, 4.8000002, 6.4000001, 8.3999996, 15.6]
```

Create the map

The TopoJSON file for the US corresponds to the file `topo/us-10m.json` and is generated according to the instructions in Mike Bostock's repository for the US-Atlas (https://github.com/mbostock/us-atlas).

Finally, the code for the SVG map is as follows:

```
<!DOCTYPE html>
<!-- This maps uses bits and pieces from multiple Mike Bostock's examples, including:
  https://bl.ocks.org/mbostock/5737662
  https://bl.ocks.org/mbostock/9943478
  https://bl.ocks.org/mbostock/9943478
  http://bl.ocks.org/mbostock/4090848
-->
<meta charset="utf-8">
<style>

/* CSS goes here. */
path {
  stroke-linejoin: round;
}

/* This is style for the different thresholds. */
.colour1 {fill: #f2f0f7;}
.colour2 {fill: #dadaeb;}
.colour3 {fill: #bcbddc;}
.colour4 {fill: #9e9ac8;}
.colour5 {fill: #756bb1;}

.label {
  font-family: Verdana;
  font-size: 16px;
}

.d4-county:hover {
```

```
    opacity: 0.3;
  }

  .counties {
    fill: #ccc;
  }

  .county-boundary {
    fill: none;
    stroke: #777;
    stroke-width: .35px;
  }

  .d4-county-boundary {
    fill: none;
    stroke: #145eda;
    stroke-width: 2px;
  }

  .state-boundary {
    fill: none;
    stroke: white;
    stroke-width: 2px;
  }

  </style>
  <body>
  <script src="//d3js.org/d3.v3.min.js" charset="utf-8"></script>
  <script src="//d3js.org/queue.v1.min.js"></script>
  <script src="//d3js.org/topojson.v1.min.js"></script>
  <script>

  /* JavaScript goes here. */

  /* These constraints on width and height produce a nice squarish size. */
  var width  = 960*0.7,
      height = 500*1.2;

  var formatNumber = d3.format(",.1f");

  /* The thresholds are harcoded here. The jenks breaks are rounded some. */
  var color = d3.scale.threshold()
      .domain([3.0, 5.0, 7.0, 9.0, 16.0])
      .range(["#f2f0f7", "#dadaeb", "#bcbddc", "#9e9ac8", "#756bb1", "#54278f"]);

  /* I played around with the projection options to limit the view to the Fourth District.
  */
  var projection = d3.geo.albers()
      .center([3,39.43])
      .rotate([85, 0])
      .parallels([29.5, 45.5])
      .scale(5800)
      .translate([width / 2, height / 2]);

  var path = d3.geo.path()
      .projection(projection);

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

```
88.
89. /* Here is the selection of counties in the Fourth District. */
90. var selected = {"21011": 1,"21013": 1,"21015": 1,"21017": 1,"21019": 1,"21023": 1,"21025":
    1,"21037": 1,"21043": 1,"21049": 1,"21051": 1,"21063": 1,"21065": 1,"21067": 1,"21069": 1,
    "21071": 1,"21079": 1,"21081": 1,"21089": 1,"21095": 1,"21097": 1,"21109": 1,"21113": 1,"2
    1115": 1,"21117": 1,"21119": 1,"21121": 1,"21125": 1,"21127": 1,"21129": 1,"21131": 1,"211
    33": 1,"21135": 1,"21137": 1,"21147": 1,"21151": 1,"21153": 1,"21159": 1,"21161": 1,"21165
    ": 1,"21173": 1,"21175": 1,"21181": 1,"21189": 1,"21191": 1,"21193": 1,"21195": 1,"21197":
    1,"21199": 1,"21201": 1,"21203": 1,"21205": 1,"21209": 1,"21235": 1,"21237": 1,"21239": 1,
    "39001": 1,"39003": 1,"39005": 1,"39007": 1,"39009": 1,"39011": 1,"39013": 1,"39015": 1,"3
    9017": 1,"39019": 1,"39021": 1,"39023": 1,"39025": 1,"39027": 1,"39029": 1,"39031": 1,"390
    33": 1,"39035": 1,"39037": 1,"39039": 1,"39041": 1,"39043": 1,"39045": 1,"39047": 1,"39049
    ": 1,"39051": 1,"39053": 1,"39055": 1,"39057": 1,"39059": 1,"39061": 1,"39063": 1,"39065":
    1,"39067": 1,"39069": 1,"39071": 1,"39073": 1,"39075": 1,"39077": 1,"39079": 1,"39081": 1,
    "39083": 1,"39085": 1,"39087": 1,"39089": 1,"39091": 1,"39093": 1,"39095": 1,"39097": 1,"3
    9099": 1,"39101": 1,"39103": 1,"39105": 1,"39107": 1,"39109": 1,"39111": 1,"39113": 1,"391
    15": 1,"39117": 1,"39119": 1,"39121": 1,"39123": 1,"39125": 1,"39127": 1,"39129": 1,"39131
    ": 1,"39133": 1,"39135": 1,"39137": 1,"39139": 1,"39141": 1,"39143": 1,"39145": 1,"39147":
    1,"39149": 1,"39151": 1,"39153": 1,"39155": 1,"39157": 1,"39159": 1,"39161": 1,"39163": 1,
    "39165": 1,"39167": 1,"39169": 1,"39171": 1,"39173": 1,"39175": 1,"42003": 1,"42005": 1,"4
    2007": 1,"42019": 1,"42031": 1,"42039": 1,"42049": 1,"42051": 1,"42053": 1,"42059": 1,"420
    63": 1,"42065": 1,"42073": 1,"42085": 1,"42111": 1,"42121": 1,"42123": 1,"42125": 1,"42129
    ": 1,"54009": 1,"54029": 1,"54051": 1,"54069": 1,"54095": 1,"54103": 1};
91.
92. /* Loading the JSON and unemployment data. */
93.
94. queue()
95.     .defer(d3.json, "us.json")
96.     .defer(d3.tsv, "urd4.tsv")
97.     .await(ready);
98.
99. function ready(error, us, unemployment) {
100.   if (error) throw error;
101.
102.   var cutoffs  = [0.0, 3.0, 5.0, 7.0, 9.0, 16.0]
103.   var rateById = {},
104.       nameById = {}
105.
106.   unemployment.forEach(function(d) { rateById[d.id] = +d.rate; })
107.   unemployment.forEach(function(d) { nameById[d.id] =  d.county; })
108.
109.   var counties = topojson.feature(us, us.objects.counties)
110.
111.   var selection = {type: "FeatureCollection", features: counties.features.filter(function
    (d) { return d.id in selected; })
112.       };
113.
114.   var exselection = {type: "FeatureCollection", features: counties.features.filter(functi
    on(d) { return d.id && !(d.id in selected); })
115.       };
116.
117. // Create path for Counties outside selection
118.   svg.append("path")
119.       .datum(exselection)
120.       .attr("class", "counties")
121.       .attr("d", path);
122.
123.   svg.append("path")
124.       .datum(topojson.mesh(us, us.objects.counties, function(a, b) {
125.         return a !== b // a border between two counties
```

```
            && (a.id === 53000 || b.id === 5300 // where a and b are in puget sound
              || a.id % 1000 && b.id % 1000) // or a and b are not in a lake
            && !(a.id / 1000 ^ b.id / 1000) // and a and b are in the same state
            && (
               ( !(a.id in selected) && !(b.id in selected) )
            || (  (a.id in selected) && !(b.id in selected) )
            || ( !(a.id in selected) &&  (b.id in selected) )
            );
      }))
      .attr("class", "county-boundary")
      .attr("d", path);

// Create path for Counties inside selection
// Here the counties are color-coded according to their unemployment rate.

  svg.append("g")
      .selectAll("path")
        .data(selection.features)
        .enter()
        .append("path")
        .attr("class", "d4-county")
        .attr("id", function(d) {
           return "F" + d.id ;}
            )
         .style("fill", function(d) { return color(rateById[d.id]); })
         .attr("d", path)
         .append("title")
         .text(function(d) { return nameById[d.id] + " (FIPS: " + d.id + ")"
           + "\nUnemployment rate: " + formatNumber(rateById[d.id]) + "%"; });

  svg.append("path")
      .datum(topojson.mesh(us, us.objects.counties, function(a, b) {
        return a !== b // a border between two counties
            && (a.id === 53000 || b.id === 5300 // where a and b are in puget sound
              || a.id % 1000 && b.id % 1000) // or a and b are not in a lake
            && !(a.id / 1000 ^ b.id / 1000) // and a and b are in the same state
            && (
               (  (a.id in selected) &&  (b.id in selected) )
            || (  (a.id in selected) && !(b.id in selected) )
            || ( !(a.id in selected) &&  (b.id in selected) )
            );
      }))
      .attr("class", "d4-county-boundary")
      .attr("d", path);

// Create path for State boundaries

  svg.append("path")
      .datum(topojson.mesh(us, us.objects.states, function(a, b) {
         return a !== b ; // a and b not in selection//
        }))
      .attr("class", "state-boundary")
      .attr("d", path);

// Add labels and color guide
  svg.append("g")
    .attr("class", "label").attr("id", "keycolor")
    .selectAll("rect")
    .data([1,2,3,4,5])
    .enter()
    .append("rect")
```

```
186.      .append("rect")
187.          .attr("x", 450)
188.          .attr("y", function (d) {return 350 + d*25; })
189.          .attr("width", 20).attr("height",20)
190.          .attr("class", function (d) {return "key colour" + d ;});
191.
192.    svg.append("g")
193.      .attr("class", "label").attr("id", "keytext")
194.      .selectAll("text")
195.      .data([1,2,3,4,5])
196.      .enter()
197.      .append("text")
198.          .attr("x", 475)
199.          .attr("y", function (d) {return 365 + d*25; })
200.          .text(function (d) {
201.            if (d==1)
202.              {return "0.0 to " + formatNumber(cutoffs[d]) + "%"}
203.            else
204.              {return formatNumber(cutoffs[d-1]) + " to " + formatNumber(cutoffs[d]) + "%" }
205.            ;});
206.
207.  }
208.
209.  d3.select(self.frameElement).style("height", height + "px");
210.
211.  </script>
```

Follow for updates.

(http://rubenhm.org/feed.xml) @rubenhm_org