合同会社 緑ＩＴ事務所

Midori IT Office, LLC

データビジュアライゼーション(1)

この記事は1年以上前に書かれました。
内容が古くなっている可能性がありますのでご注意下さい。

（オープン）データのビジュアライゼーションに関して、何回かに分けて記します。
第1回目は、D3.jsとGeoJSONを用いて、2012年の日本の県別人口をビジュアライズしてみることにします。

D3.jsは、データビジュアライゼーションのための、オープンソース（BSDライセンス）のJavaScriptライブラリです。**D3**は、**D**ata-**D**riven **D**ocumentsの略です。
GeoJSONは、おなじみのJSON（JavaScript Object Notation）を用いた、地理空間データ交換フォーマットです。
GeoJSONのデータを用いてD3.jsで日本地図を描き、人口の多寡によって県を塗り分ける、というのが今回のゴールです。

県別人口のデータは、政府統計の総合窓口（ｅ－Ｓｔａｔ）から入手できます。しかし、Excelファイルのため、余分な行と列を削除し、文字コードをUTF-8に変換し、カンマ区切りのCSVファイルする必要があります。先頭行では、列名を
ken,population
と定義します。ファイル名はpop2012.csvとします。

日本地図のGeoJSONファイルの作成は、

- Shape形式の地図データの入手
- GeoJSON形式への変換

という2つのステップが必要です。以下の作業は、
こちらのページ[1]を参考にさせて頂きました。

まず、Natural Earthから、Shape形式の地図データを入手します。Natural Earthは、たくさんの地図データをpublic domainで提供しています。今回は、
http://www.naturalearthdata.com/downloads/10m-cultural-vectors/
の「Admin 1 – States, Provinces」にある
ne_10m_admin_1_states_provinces.zip
をダウンロードします。

次に、GDAL（Geospatial Data Abstraction Library）に含まれるogr2ogrコマンドを使用して、Shape形式からGeoJSON形式に変換します。また、ダウンロードしたファイルには世界中の地図データが含まれていますので、-whereオプションを用いて日本の地図データのみを抽出します。

```
ogr2ogr -f GeoJSON -where geonunit=\"Japan\" japan.json ne_10m_admin_1_states_provinces.shp
```

GDALは、オープンソースの地理情報システムQGISにも同梱されていますので、QGISをインストールするのが手っ取り早いと思います。Windows環境で、QGISを含むオープンソースのGIS（Geographic Information System）を色々と試してみたいときは、OSGeo4Wがお勧めです。

変換したてできたjapan.jsonファイルは、なぜか静岡県のみ県名が入っていないので修正が必要です。
“name_local”: null
となっているので、
“name_local”: “静岡県”
と修正します。

人口と地図のデータができましたので、いよいよ本題に入ります。HTMLファイルの完全なリストは以下のとおりです。

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>D3.js & GeoJSON sample</title>
  <script type="text/javascript" src="d3.js"></script>
</head>
<body>
  <script type="text/javascript">
    var w = 800;
    var h = 600;

    var svg = d3.select("body")
      .append("svg")
      .attr({width:w, height:h});

    var projection = d3.geo.albers()
      .center([-15, 36])
      .rotate([210, 0])
```

```
        .parallels([50, 60])
        .translate([w/2, h/2])
        .scale([1500]);
      var path = d3.geo.path().projection(projection);

      d3.csv("pop2012.csv", function(data) {
        d3.json("japan.json", function(json) {
          for(var i=0; i<data.length; i++) {
            for(var j=0; j<json.features.length; j++) {
              if( data[i].ken == json.features[j].properties.name_local ) {
                json.features[j].properties.population = data[i].population;
              }
            }
          }

          svg.selectAll("path")
            .data(json.features)
            .enter()
            .append("path")
            .attr("d", path)
            .style("fill", function (d) {
              var population = d.properties.population;
              if( population > 10000000 )
                var c = "darkred";
              else if( population > 5000000 )
                var c = "orangered";
              else if( population > 2500000 )
                var c = "orange";
              else if( population > 1000000 )
                var c = "gold";
              else
                var c = "yellow";
              return c;
            })
            .style("stroke", "gray")
            .style("stroke-width", "0.5px");
        });
      });
    </script>
  </body>
  </html>
```

まず最初に

```
<script type="text/javascript" src="d3.js"></script>
```

でD3のライブラリを読み込みます。これでD3の豊富な機能を利用することができます。

今回は、HTML文書のbodyはscriptのみです。
早速、D3を使います。まず、

```
var w = 800;
var h = 600;
```

描画をするための領域であるSVG（Scalable Vector Graphics）要素の幅と高さの変数を定義し、D3を用いてSVG要素を作成します。次の

```
d3.select("body")
```

では、HTMLのbody要素を選択しています。選択されたbody要素に対して、

```
.append("svg")
```

でSVG領域を追加（append）します。さらに、

```
.attr({width:w, height:h});
```

で、SVG要素の幅と高さを設定します。これらを

```
var svg = d3.select("body").append("svg").attr({width:w, height:h});
```

のように1行で記述することもできます。
このように、ピリオドを使ってメソッドを連鎖させる記法を**チェイン構文**と言います。

次に、地図を描画するための準備をします。

```
var projection = d3.geo.albers()
  .center([-15, 36])
  .rotate([210, 0])
  .parallels([50, 60])
  .translate([w/2, h/2])
  .scale([1500]);
```

まずは投影法（projection）を定義します。投影法とは、3次元の球面である地球表面を2次元の平面上に表現するための方法です。ここでもチェイン構文を用いてたくさんの設定をしています。各メソッドに渡している値については、こちらのページ[2]を参考にさせて頂きました。
次の

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

では、地図を描画するための**パスジェネレータ**を定義します。

ここから少し難しくなります。

```
d3.csv("pop2012.csv", function(data) {
```

でCSVファイルを読み込みます。csv()の最初の引数は、読み込むCSVファイルの名前、2つ目の引数はCSVファイルを読み込んだときに実行するコールバック関数になります。
function(data) { 以降、</script>の前の行の } までが、コールバック関数の定義になります。
csv()のコールバック関数では、はじめに

```
  d3.json("japan.json", function(json) {
```

でJSONファイルを読み込みます。そして、json()のコールバック関数の定義が始まります。

```
    for(var i=0; i<data.length; i++) {
      for(var j=0; j<json.features.length; j++) {
        if( data[i].ken == json.features[j].properties.name_local ) {
          json.features[j].properties.population = data[i].population;
        }
      }
    }
```

では、CSVデータとJSONデータを突き合わせて、（CSVではken、GeoJSONではfeatures.properties.name_localに入っている）県名が一致したら、CSVに入っている人口の値をJSONデータに追加します。

いよいよ、描画が始まります。

```
    svg.selectAll("path")
```

で、path要素を選択します。しかし、この時点ではまだpath要素は存在しません。次に

```
      .data(json.features)
```

でjson.featuresデータをバインドします。しかし、json.featuresデータに対応するpath要素はまだ存在しないので、

```
      .enter()
```

でjson.featuresの各データに対応するプレースホルダ要素を取得し、

```
        .append("path")
```

でpath要素を追加します。これで、GeoJSONに入っていた都道府県（json.featuresの各データ）に対応するpath要素ができました。以降では、描画のための設定をします。

```
        .attr("d", path)
```

で、はじめに投影法を指定して作成しておいたパスジェネレータを属性として指定し、

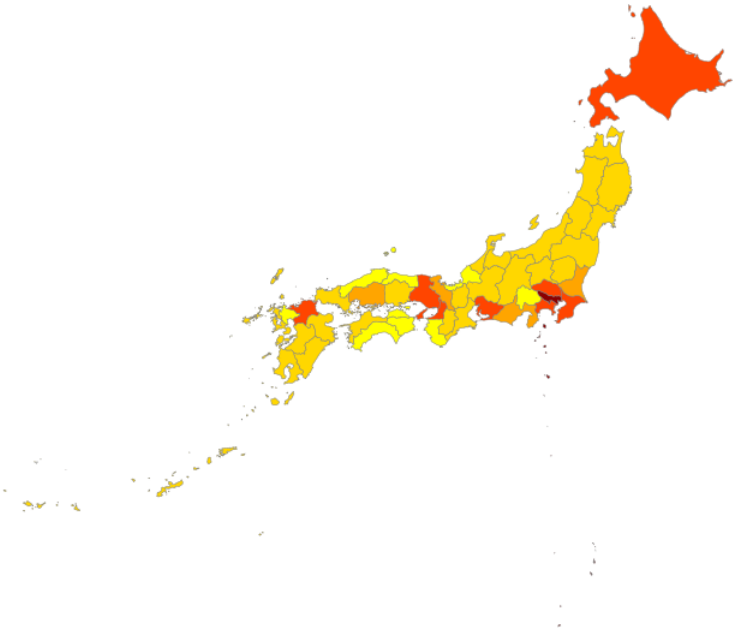
```
        .style("fill", function(feat) {
          var population = feat.properties.population;
          if( population > 10000000 )
            var c = "darkred";
          else if( population > 5000000 )
            var c = "orangered";
          else if( population > 2500000 )
            var c = "orange";
          else if( population > 1000000 )
            var c = "gold";
          else
            var c = "yellow";
          return c;
        })
```

人口に応じて塗りつぶしの色を指定します。
関数の引数featには、データバインドしたjson.featuresが渡されますので、feat.properties.populationで人口の値が得られます。この値によって異なる色を返します。
最後に、

```
        .style("stroke", "gray")
        .style("stroke-width", "0.5px");
```

線の色と太さを指定しておしまいです。

実行結果は、以下のようになります（画像キャプチャです。実際のHTMLの表示はこちら）。
余談ですが、山梨県の人口が意外に少ないことに驚きました。



Firefoxの場合は、同じフォルダにd3.js、japan.html、japan.json、pop2012.csvを置いてjapan.htmlを開けば上のような地図が表示されますが、Chromeでは何も表示されません。「デベロッパーツール」を表示すると「cannot load …. Cross origin requests are only supported for HTTP.」というようなエラーメッセージを見ることができます。Chromeを使用する場合は、Webサーバにファイルを置いてHTTP経由で参照するか、Chromeの起動時に「–allow-file-access-from-files」オプションを付ける必要があります。

ダウンロード
japan.zip

参考ページ

[1] D3.js で日本地図を描く, y_uti のブログ
[2] D3.jsとTopoJSONで地図を表示してみた その４, stylesslab-annex

カテゴリー: オープンデータ, ビジュアライゼーション | タグ: D3.js | 投稿日: 2014年3月4日
[https://midoriit.com/2014/03/%e3%83%87%e3%83%bc%e3%82%bf%e3%83%93%e3%82%b8%e3%83%a5%e3%82%a2%e3%83%a9%e3%82%a4%e3%82%bc%e3%83%bc%e3%82%b7%e3%83%a7%e3%83%b31.html] | 投稿者: 小池隆