D3.jsを使った
データビジュアライズ勉強会
「テレビ放送受信契約数」版

2014年7月17日

先端IT活用推進コンソーシアム
クラウド・テクノロジー活用部会

使用するデータを選択

- 使用するデータの選択条件
  - 住所もしくは緯度経度と、数値があるもの
  - 時間とともに変化すると、さらに面白い
- 今回使用するデータ
  - 横浜市統計情報ポータルサイト ＞ 分野別インデックス
    - http://www.city.yokohama.lg.jp/ex/stat/index2.html

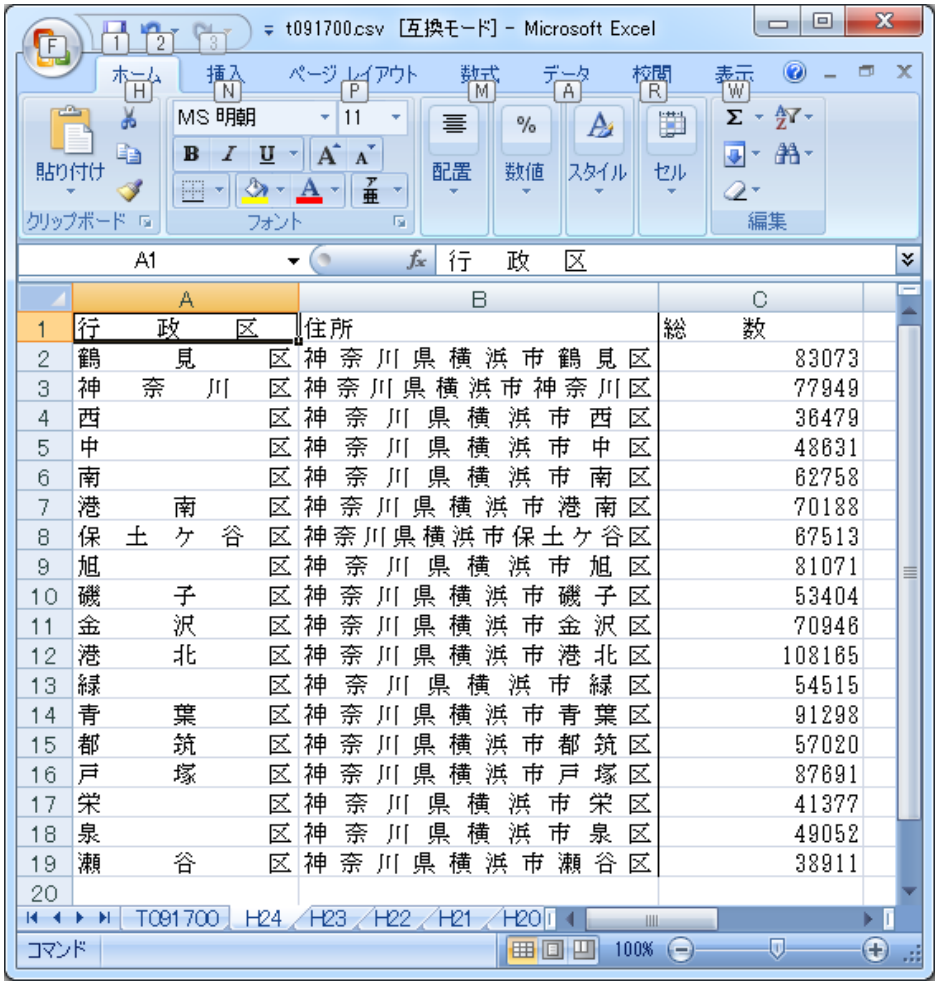
    →「電話・テレビ・郵便・インターネット」

    →「テレビ放送受信契約数」**※注：Excelファイル**
    - 平成18年から平成24年までの増減を可視化してみる
  - ExcelとJavaScriptに自信があれば、他データに挑戦してもOK
- Excelを持っていない人は
  - Excel Viewで開いて、テキストエディタにコピペ
  - もしくは、PDFからテキストエディタにコピペ

データ準備－１

- Excel→CSVに変換する
  - 「H24」シートを選択
  - CSVらしいフォーマットに変換
    - 不要な行＆列と「衛星契約」の列を削除
    - タイトル部分のセル結合を解除
    - 総数の「セルの書式設定」を「標準」にする
  - 「鶴見区」を「神奈川県横浜市鶴見区」にする
    - ユニークな地名であれば自動補完してくれるけど、念のため
    - タイトルは「住所」
    - =“神奈川県横浜市” & A2
  - 「名前を付けて保存」でCSVを選択
    - 保存したものをコピー。ファイル名は「H24.csv」
      - 保存したCSVファイルは、Excelがロックしているため
    - テキストエディタで開いて、意図通りか確認
      - タイトル部分の余計なスペース、行の最後の余計なカンマ、空行があれば削除

データ準備－1－最終形



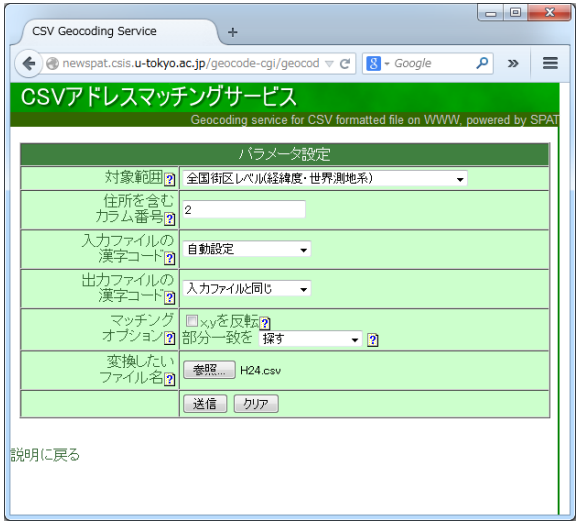
	A	B	C
1	行政区	住所	総数
2	鶴見区	神奈川県横浜市鶴見区	83073
3	神奈川区	神奈川県横浜市神奈川区	77949
4	西区	神奈川県横浜市西区	36479
5	中区	神奈川県横浜市中区	48631
6	南区	神奈川県横浜市南区	62758
7	港南区	神奈川県横浜市港南区	70188
8	保土ケ谷区	神奈川県横浜市保土ケ谷区	67513
9	旭区	神奈川県横浜市旭区	81071
10	磯子区	神奈川県横浜市磯子区	53404
11	金沢区	神奈川県横浜市金沢区	70946
12	港北区	神奈川県横浜市港北区	108165
13	緑区	神奈川県横浜市緑区	54515
14	青葉区	神奈川県横浜市青葉区	91298
15	都筑区	神奈川県横浜市都筑区	57020
16	戸塚区	神奈川県横浜市戸塚区	87691
17	栄区	神奈川県横浜市栄区	41377
18	泉区	神奈川県横浜市泉区	49052
19	瀬谷区	神奈川県横浜市瀬谷区	38911
20			

H24.csv

```
行政区,住所,総数
鶴見区,神奈川県横浜市鶴見区,83073
神奈川区,神奈川県横浜市神奈川区,77949
西区,神奈川県横浜市西区,36479
中区,神奈川県横浜市中区,48631
南区,神奈川県横浜市南区,62758
港南区,神奈川県横浜市港南区,70188
保土ケ谷区,神奈川県横浜市保土ケ谷区,67513
旭区,神奈川県横浜市旭区,81071
磯子区,神奈川県横浜市磯子区,53404
金沢区,神奈川県横浜市金沢区,70946
港北区,神奈川県横浜市港北区,108165
緑区,神奈川県横浜市緑区,54515
青葉区,神奈川県横浜市青葉区,91298
都筑区,神奈川県横浜市都筑区,57020
戸塚区,神奈川県横浜市戸塚区,87691
栄区,神奈川県横浜市栄区,41377
泉区,神奈川県横浜市泉区,49052
瀬谷区,神奈川県横浜市瀬谷区,38911
```

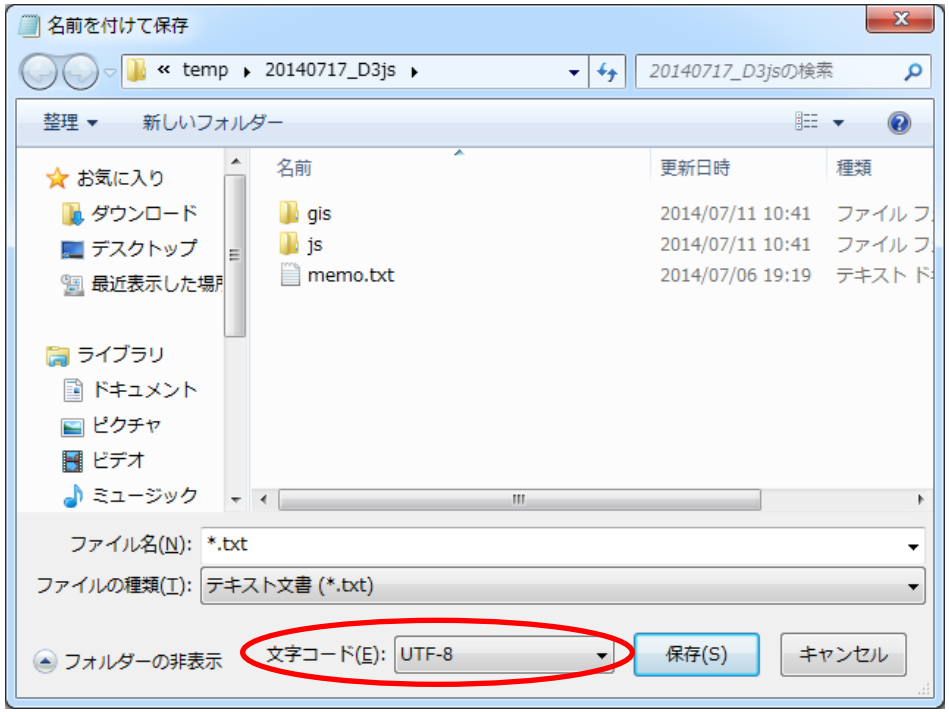
データ準備－2

- 住所を緯度経度に変換する
  - http://newspat.csis.u-tokyo.ac.jp/geocode/
  - 「今すぐサービスを利用する」をクリック
  - 各パラメータを設定
    - 住所を含むカラム番号：2
    - 変換したいファイル名：H24.csv
  - 「送信」を押すと、変換結果のCSVが落ちてくる
    - ダウンロード後、Excelで開いて内容を確認
    - 取得した緯度経度をGoogleMapsで確認
      - 「fY fX」の順で検索



データ準備－３

- 文字コード変換
  - テキストエディタで開き、文字コードを「UTF-8」に変更し保存
  - ファイル名は、「H24UTF8.csv」を指定



Windowsのメモ帳
「ファイル」→「名前を付けて保存」

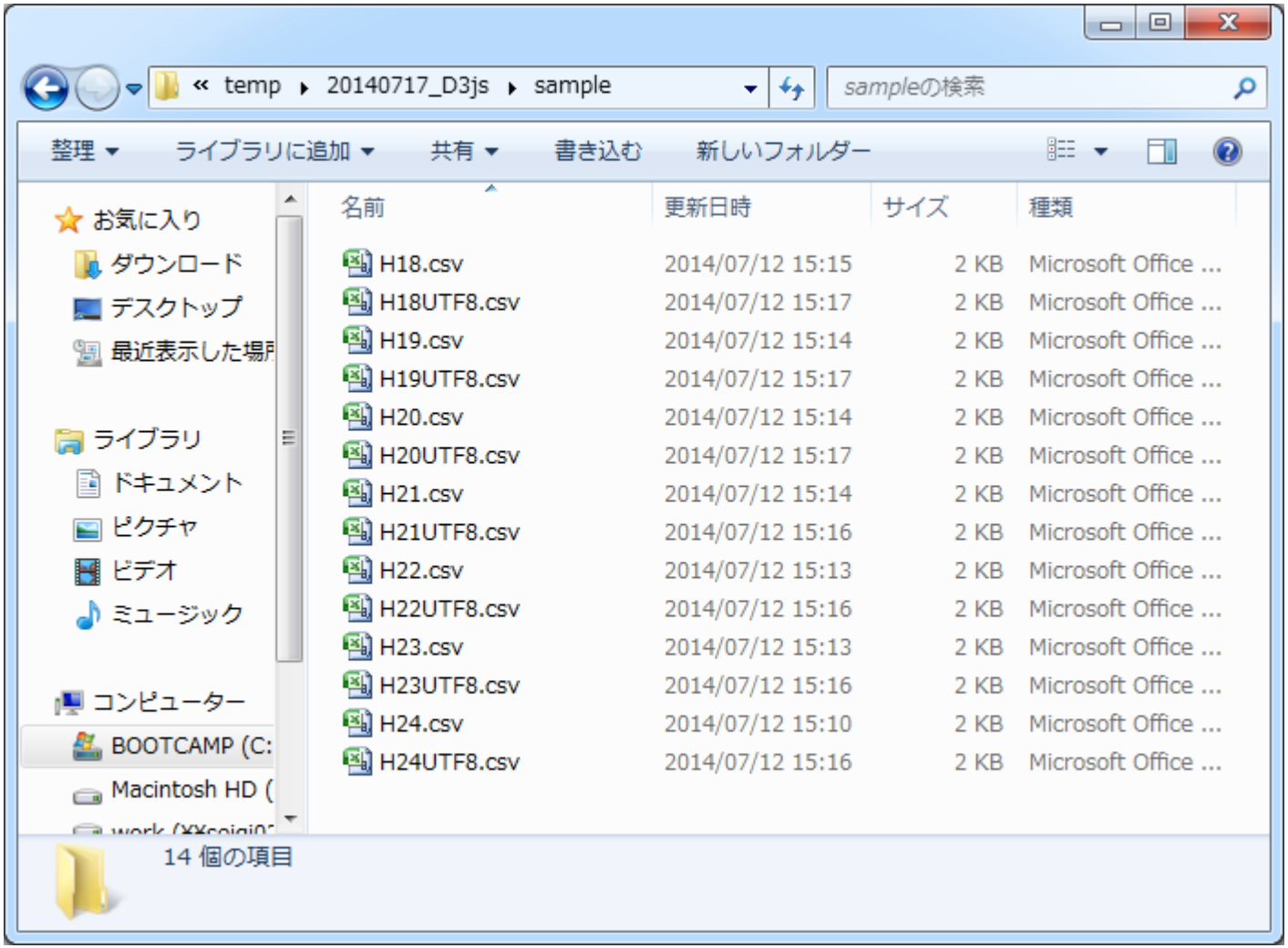
Macのテキストエディット
「テキストエディット」→「環境設定」

データ準備－4（余裕のある人だけ）

- H23～H18まで、同様の手順で行う
  もしくは
- H24.csvをコピーし、「総数」部分だけコピー
  - 他の項目が変化していない事を確認
  - 「形式を選択して貼り付け」で『値』だけをコピー
  - 住所→緯度経度変換サービスを呼び出す回数減
  - 全ファイルを「UTF-8」に変換
    - Excelが保存時に、CSVの文字コードはShift-JISにする
    - 文字コードの確認方法
      - 拡張子を.txt にして、ブラウザで開いてみる
      - エンコードを確認

データ準備－4（余裕のある人だけ）

- こんな状態になっているはず



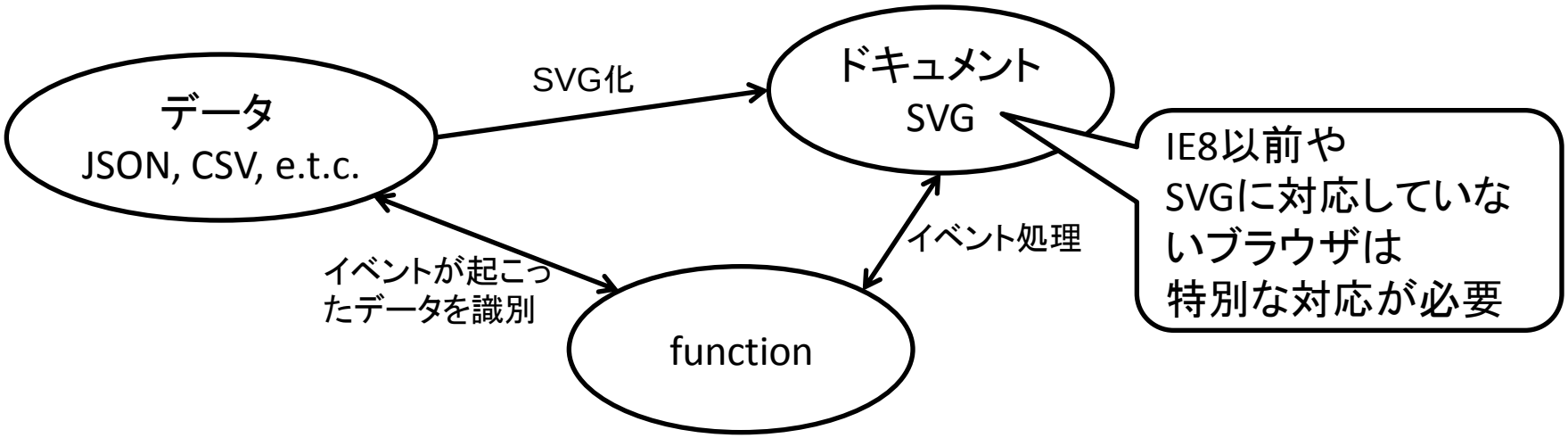
D3.jsで可視化しよう

D3.js とは

D3.js（またはD3:Data-Driven Documents、旧:Protovis[1]）は、2011年に開発が始まった[2]ウェブブラウザで動的コンテンツを描画するJavaScriptライブラリである。World Wide Web Consortium準拠のデータ可視化ツールとして、Scalable Vector Graphics（SVG）、JavaScript、HTML5、Cascading Style Sheetsを最大限に活用している。その他多くのライブラリとは対照的に、最終的に出力された結果に視覚的な調整ができる。[3]

ウィキペデアより

- データをドキュメント化して、関係も保持
  - SVGを効率良く生成するためのライブラリです



D3.jsを使ってビジュアライズ

- 今回は地図を軸に可視化してみる
  - D3.js は、地図描画の機能が充実

- 注意：地図は楽だけど、グラフを描くのは意外に面倒
  - 必要な部品がそろっているだけ
  - D3.js Examples → 各グラフを描くためのソースを参照
    - https://github.com/mbostock/d3/wiki/Gallery
  - 参考URL http://postd.cc/what-d3js-is-not/
- 参考書籍

  エンジニアのための データ可視化[実践]入門
  ~D3.jsによるWebの可視化
  - http://www.amazon.co.jp/dp/4774163260
  - D3.js はほとんど出てこないけど、「可視化」の理解が深まります

緯度・経度を扱う上での注意

0, 0

スクリーン

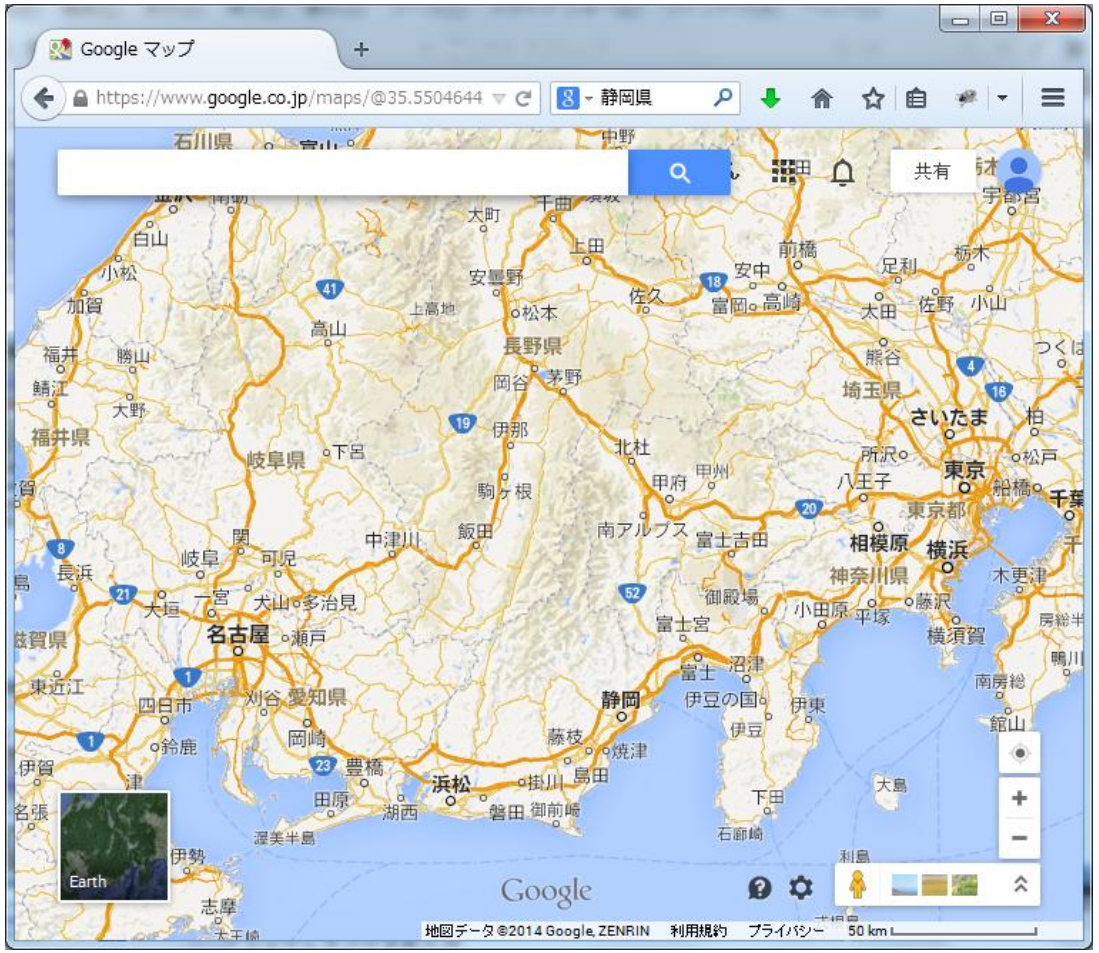
画面内に描画するため
・スクリーンの原点と、緯度・経度の原点が違う
・表示スケールの調整

経度・緯度

136.09, 34.36

0 , 0
136.09, 36.77

900 , 0
140.18, 36.77



0 , 600
136.09, 34.36

900 , 600
140.18, 34.36

開発環境の準備

- D3.js をダウンロード
  - http://d3js.org/
    - d3.zip をダウンロードし、d3.min.js を取り出す
    - js ディレクトリを作成し、格納
    - 今回は、すでに入っています
- テキストエディタ
  - お気に入りのエディタでOK
- ブラウザ
  - [F-12]を押せば、開発ツールが起動する
  - Chrome：「ツール」→「デベロッパーツール」
    - ローカルのデータファイルを参照するためには、起動オプションを追加
      「--allow-file-access-from-files」
  - Firefox＋firebug：「ツール」→「Web開発」→「Firebug」→「Firebugを開く」
  - Safari：「環境設定」→「詳細」→「メニューバーに”開発”メニューを表示」
  - IE（9以降）：「F12 開発者ツール」
    - ローカルだとうまく動作しない
      - データにアクセスしている部分をjQueryにすれば、動くらしい

地図データの準備

- 地図データ
  - D3.js の geo パッケージで読める形式が必要
    - GeoJSON：地理情報用に規格されたJSONデータ形式
    - TopoJSON：GeoJSONの拡張形式。D3.jsではプラグインが必要
      - 冗長性を排するので、データサイズが20%程度になる
  - Shape形式の地図を入手し、GeoJSONに変換
    - 入手元：国土交通省、国土数値情報
      - http://nlftp.mlit.go.jp/ksj/gml/datalist/KsjTmplt-N03.html
      - 神奈川県：N03-130401_14_GML.zip
    - このサイトで10%に圧縮＋GeoJSONに変換
      - http://www.mapshaper.org/
      - 10%というのはShapefileでの比（1,191KB→131KB）
      - Shapefile→GeoJSONに変換すると大きくなる（131KB→514KB）
  - 変換結果（GeoJSON形式、514KB）
    - http://cloud.aitc.jp/20140717_D3js/gis/kanagawa.json

D3.jsの超概要 http://ja.d3js.node.ws/

- セレクタ（W3C Selectorsを参照）
  - d3.select(**“#hoge”**) → <xxx id=“hoge”> を対象
  - d3.select(**“.hoge”**) → <xxx class=“hoge”>を対象
  - d3.select(**“hoge”**) → <hoge>を対象

```
signal = [
    { “cx”: 100, “cy”: 100, “color”: “#0000ff”, “title”:”青”,},
    { “cx”: 200, “cy”: 100, “color”: “#ffff00”, “title”:”黄”,},
    { “cx”: 300, “cy”: 100, “color”: “#ff0000”, “title”:”赤”,},
];
```

- セレクション
  - selectAll(), enter(), exit()
  - 繰り返し処理が楽に書ける
    - d3.select("#TEXT1").**selectAll("p")**.style("color", “#000000");
- 動的プロパティ
    - svg.selectAll(".node").data(signal).text(**function(d) { return d.title; }**);
    - svg.selectAll(".node").data(signal).text(**function(d, i) { return i; }**);
- データの結合
  - 更新：svg.selectAll(".node").data(signal).text(“hogehoge”);
  - 追加：svg.selectAll(".node").data(signal).**enter()**.append(“text").text(“piyopiyo”);
  - 削除：svg.selectAll(".node").data(signal).**exit()**.remove();
- アニメーション
    - d3.select("#TEXT4").**transition()**.style("background-color", "black");

その他の今回使用するメソッド

- ファイルの読み込み
  - d3.json(), d3.text(), d3.csv()
- 地図描画
  - d3.geo.mercator()
  - d3.geo.path().projection(projection);

SVG (Scalable Vector Graphics) について

- Wikipedia
  - http://ja.wikipedia.org/wiki/Scalable_Vector_Graphics

> **Scalable Vector Graphics**（スケーラブル・ベクター・グラフィックス、**SVG**）は、XMLをベースとした、2次元ベクターイメージ用の画像形式の1つである。アニメーションやユーザインタラクションもサポートしている。SVGの仕様はW3Cによって開発され、オープン標準として勧告されている。

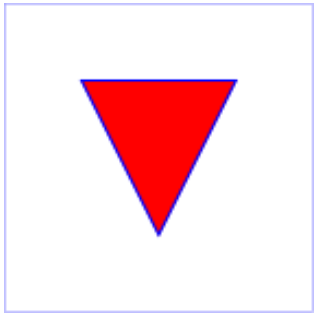
- SVG仕様
  - http://www.hcn.zaq.ne.jp/___/SVG11-2nd/

- 使用上の注意
  - 対応していないブラウザだと、何も表示されない
  - ブラウザによっては、微妙に見え方が違う

SVG (Scalable Vector Graphics) について

- 代表的な図形
  - パス

```
<path  d=“M 100 100 L 300 100 L 200 300 z”  fill=“red”
    stroke=“blue”  stroke-width="3" />
```



  - 円

```
<circle  cx=“600”  cy=“200”  r=“100”  fill=“red”  stroke=“blue”
  stroke-width="10"  />
```

  - 線

```
<line  x1=“100”  y1=“300”  x2=“300”  y2=“100” stroke=“green”
  stroke-width="5" />
```

  - テキスト

```
<text  x=“250”  y=“150”  font-family=“Verdana”  font-size=“55”
  fill="blue" >
  Hello, out there
</text>
```

以降は、ソースコードを見ながら解説

- D3.jsでデータとSVGが連動するサンプル
  - step1_1.html, step1_2.html
- D3.jsで地図を描画する
  - step2.html
- D3.jsでデータを取得する
  - step3.html
- 取得したデータをsvgで可視化する
  - step4.html
    - 緯度・経度の扱いに注意
- 地図に重ねてデータを描画する
  - 金沢区内公園を描画するサンプルコード
  - step5.html

step5.html を改造し、「テレビ放送受信契約数」に対応

```
35        ////////////////////////////////////////////////////////////////////////
36        // 金沢区内公園  http://www.city.yokohama.lg.jp/kanazawa/kz-opendata/
37        var csvfile = "./23-kz-park_utf8.csv";
38        d3.csv(csvfile, function(csv) {
39            points = csv;
40            svg = d3.select("svg");  // <svg>を選択
41
42            svg.selectAll(".node")      //  <circle class="node">を選択
43                .data(points).enter()       // データの増分を対象
44                .append("circle")           // svgのcircleを追加
45                .attr("class", "node")
46                .attr("cx", function(d) { return projection([d.経度, d.緯度])[0]; })    // 円の中心
47                .attr("cy", function(d) { return projection([d.経度, d.緯度])[1]; })    // 円の中心
48                .attr("r", function(d) { return 5; })                                  // 円の半径
49                .style("stroke", function(d) { return "#000000"; }) //
50                .style("stroke-width", "1px")                       // 枠線の太さ
51                .style("fill", function(d) {                        // 塗りつぶし
52                    if (d.名称.indexOf("公園")>=0){
53                        return "white";
54                    } else {
55                        return "red";
56                    }
57                })
58                .on('click', function(d) {
59                    console.log(d);
60                });
61            ;
62
63            // 住所を重ねて書く
64            svg.selectAll(".txt")
65                .data(points).enter()
66                .append("text")
67                .attr("class", "txt")
68                .attr("x",function(d) { return projection([d.経度, d.緯度])[0]; })
69                .attr("y",function(d) { return projection([d.経度, d.緯度])[1]; })
70                .style("font", "8pt sans-serif")
71                .text(function(d) { return d.住所; })
72            ;
73        });
74    });
```

ファイル名を「./H24UTF8.csv」にし、csvファイルを同ディレクトリにコピー

csvのタイトルに合わせる

「総数」から算出

とりあえず固定で「white」を返す

csvのタイトルに合わせる

余裕のある人用の拡張案

- 自分の位置情報を表示
  - HTML5のGeolocation API
- H18～H24のデータを使ってアニメーション
  - 変換が面倒な人は、sampleディレクトリの下を参照
  - JavaScriptのsetInterval()
  - D3.js の更新＋アニメーション機能

    .transition(). duration(ミリ秒).変化後のスタイル()
  - 静岡県雨量を使った例
    - http://cloud.aitc.jp/20140627_D3js/sample6.html
- 課題
  - 微妙な増減をどうやって伝えるか？

実装例：step6example.html

今後の拡張の方向

- 付加情報を表示
  - clickかmouseoverで、地名・住所・関連情報を重ねて表示
  - 文字をもっと見易くする（白抜き文字、など）
    - http://www.slideshare.net/kadoppe/inline-svg/53
- 他のデータも合わせて表示
  - 区の世帯数に対する割合
  - 区の世帯数の増減に対する割合
- アニメーション
  - 「もう1つ軸を足したい」時
- 拡大／縮小
  - 特定の区を詳細に見たい
  - 倍率に合わせて、文字のサイズなどを調整

Q&A

- 「テレビ放送受信契約数」対応実装例
  - step5example. js
  - step6example. js