

# D3.js入門

西田 直樹

# はじめに

- 必要なもの: モダンなブラウザ
  - ChromeかFirefoxが無難
  - Opera…?
- スライドとサンプルコードはすべてGitHubにアップロードしてあります
  - [https://github.com/domitry/KC3\\_D3js](https://github.com/domitry/KC3_D3js)
- 手元でスライドを見ながら進めてください.

# 自己紹介

- 西田 直樹 @domitry
- 専門: 生物物理
- A member of SciRuby, E-cell project
- Simulation, Visualization



# Agenda

- D3.jsとは
- D3.jsができること
- JavaScriptの基本
- D3.jsの基本
- やってみよう
- 終わり

# D3.js とは

- Data Driven Documents -> d3
- データ可視化のためのナウでヤングなライブラリ
- 詳しくは <http://d3js.org/>

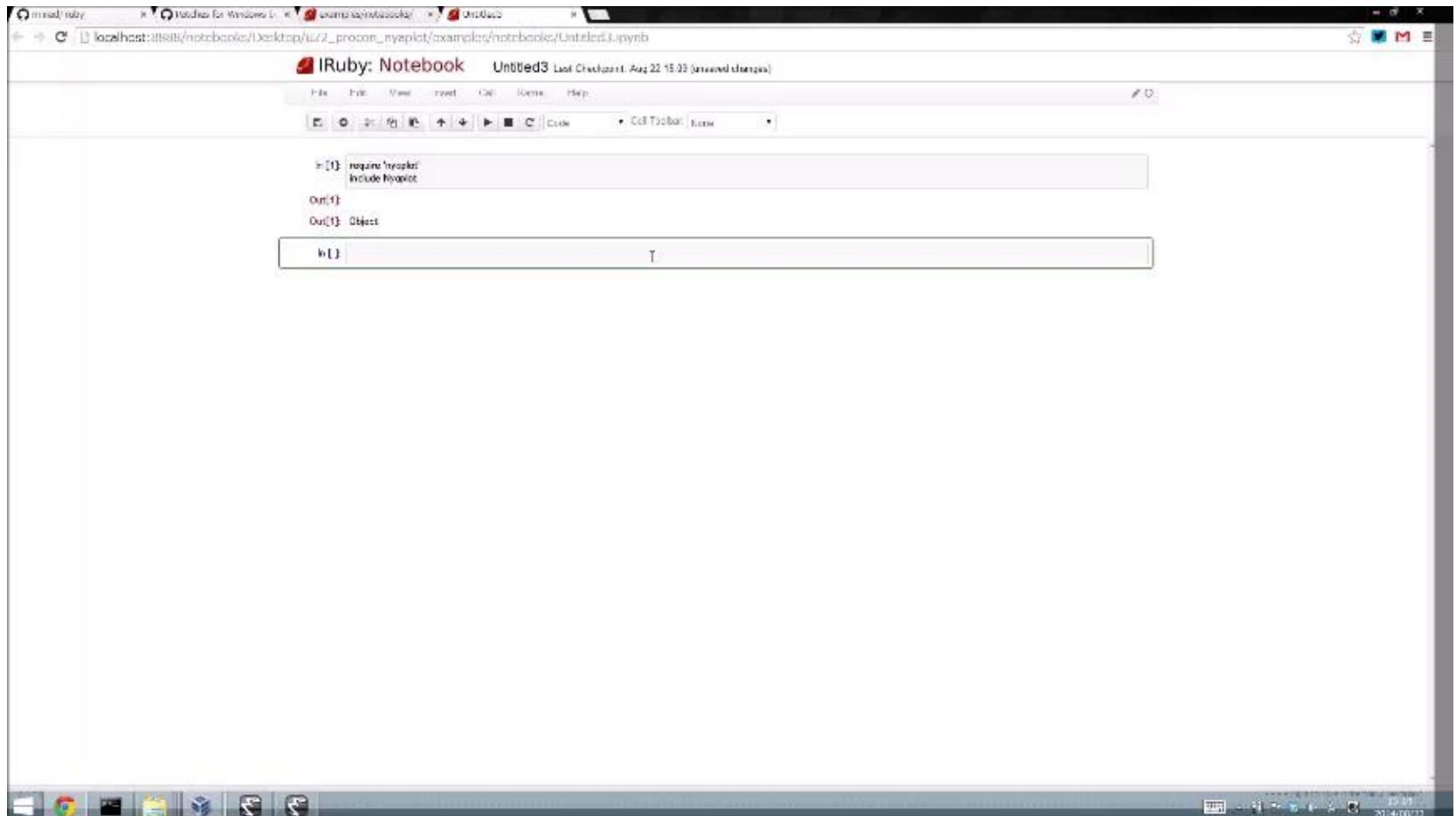


# D3.jsがで き る こ と

- 色々。jQueryの代替のようなことから高レベルなSVGの構築など
- 統計な人たちからBioinformatics屋さんまで
- 他言語との連携が熱い

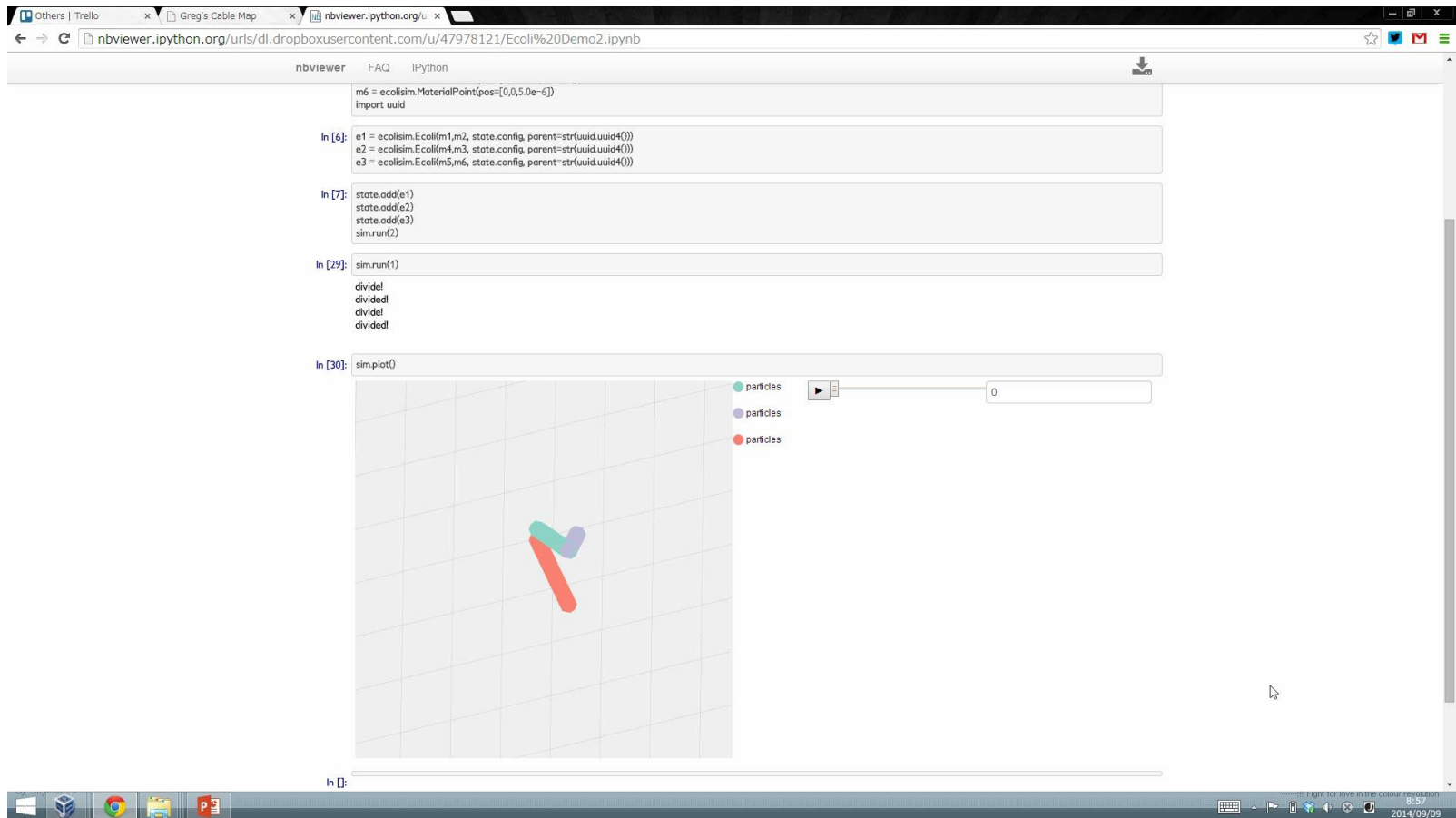
# Nyaplot

- D3.js+Ruby



# Elegans

- D3.js+WebGL





# JavaScriptの基本

# Question

- プログラミングしたことある人？

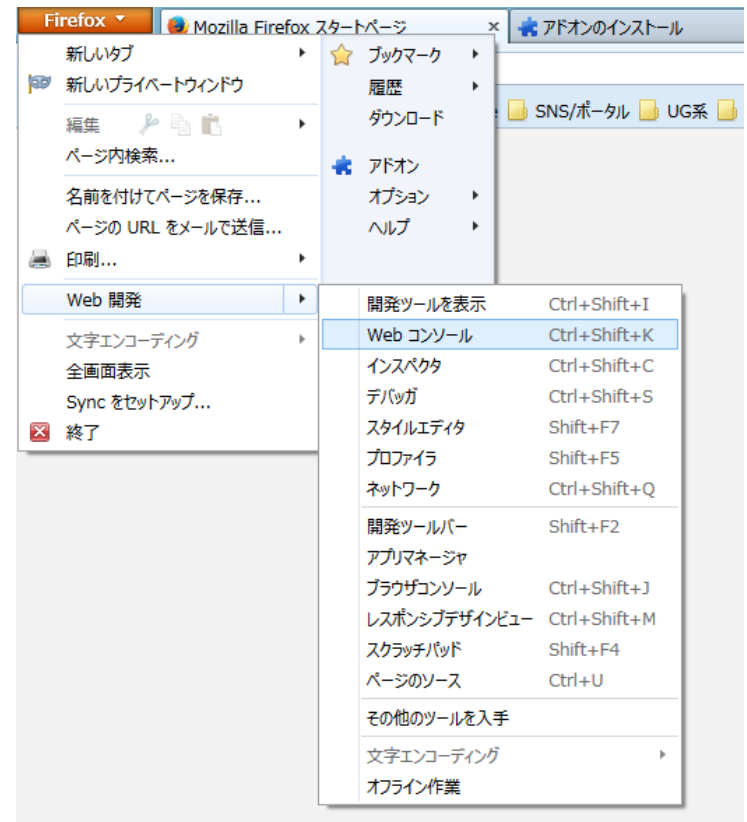
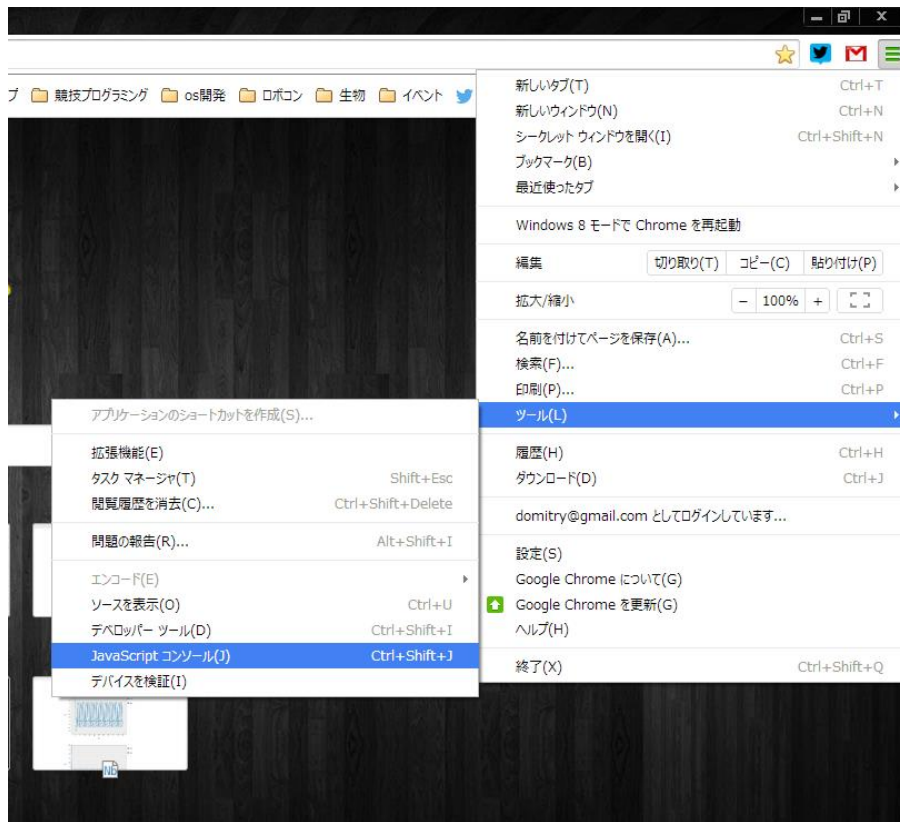
# Question

- プログラミングしたことある人？
- C言語書いたことある人？

# Question

- プログラミングしたことある人？
- C言語書いたことある人？
- JavaScript書いたことある人？

# JavaScriptコンソール



# JavaScriptコンソール

- コードを入力するとその場で実行
- Shift+Enterで複数行入力

```
console.log( “Hello, world!” );
```

# 変数と型

- JavaScriptは動的型付け言語
  - Number -> intやfloatの区別なし
  - Array
  - Object -> 実は割と皆Object, 今回はHash的に使う
  - Function

```
typeof {a: 3, b: 4, c: 10} //->?  
typeof [10, 20, 30] //->?
```

# 制御文の数々

- 基本的にCと一緒に！（と思っていたら痛い目にあうこともある）
  - if
  - switch, case
  - for
  - while
  - break, continue



# 関数

- functionで定義
- 引数の指定に型はいらない

```
function Str2Int(str){  
    return ParseInt(str);  
}
```

```
Str2Num( “3” ); //-> 3
```

# 無名関数・クロージャ

- window.onloadに無名関数を代入するとページのロードが終わった時点で実行される。

```
window.onload = function(){  
    console.log( "Hello, world!" );  
};
```

# 無名関数・クロージャ

- 先ほどのStr2Intは次のように書ける。
- 関数はFunction型のオブジェクト

```
var Str2Int = function(str){  
    return parseInt(str);  
};  
  
Str2Num( "3" ); //-> 3  
console.log(typeof Str2Int); //-> 'function'
```

# 無名関数・クロージャ

- スコープを汚したくないときは次のようにも書く

```
var hoge = (function(){  
    var a = 5;  
    var b = 3;  
    return a*b;  
})();  
  
console.log(typeof a); //-> 'undefined'
```

# D3.jsの基本

# D3.jsの基本

- ここで解説するもの
  - selector
  - data, datum
  - scale
- サンプルコードは下記URLを開いてJavaScript  
コンソールで実行してください
  - <http://goo.gl/DDwji8>

# Selector

- jQueryのセレクタと同じようなもの
  - `select(name)`
  - `selectAll(name)`

```
var hoge = d3.select( “.hoge” ); // hogeというクラスの要素を一つ選択  
hoge = d3.select( “svg” ); // svgタグの要素を一つ選択
```

```
var child = hoge.select( “rect” );
```

```
var rects = d3.selectAll( “rect” ); // rectタグの要素をすべて選択
```

# Selector

- 便利なメソッド
  - append(node\_name)
  - attr(name, val)
  - style(name, val)

```
hoge.append( "rect" ).attr({ 'x' : 100, 'y' : 100, 'width' :  
200, 'height' :200, 'fill' : '#f00' });
```

```
hoge.selectAll( "rect" ).attr( "fill" , "#0f0" );
```



# data, datum

- DOMノードとデータを紐づける
  - data – 複数のDOMノードにまとめて紐づけ
  - datum – 一つのDOMノードに一つのデータを紐づけ
- 紐づけたデータにはattrやstyle経由でアクセスできる

```
hoge.selectAll( "rect" ).data([0, 100]);  
hoge.select( "rect" ).datum([0, 100]);
```

# dataの使い方

- データの要素数だけDOMノードがない場合  
enter()で”まだ存在しないノード”を選択。

```
d3.selectAll( "rect" ).remove();  
d3.select( "svg" ).selectAll( "rect" ).data([1, 10, 100]).enter()  
  .append( "rect" ).attr({  
    x: function(d){return d/10;},  
    y: 0,  
    width: function(d){return d;},  
    height: function(d){return d;}  
  });
```

# dataの使い方

- データの要素数だけDOMノードがない場合  
enter()で”まだ存在しないノード”を選択。
- appendでrectをくっつける。

```
d3.selectAll( "rect" ).remove();  
d3.select( "svg" ).selectAll( "rect" ).data([1, 10, 100]).enter()  
.append( "rect" ).attr({  
    x: function(d){return d/10;},  
    y: 0,  
    width: function(d){return d;},  
    height: function(d){return d;}  
});
```

# Scale

- データから実際の座標に変換
  - domain – 入力データ
  - range – 出力座標

```
var scale = d3.scale.linear().domain([0, 2]).range([0, 100]);  
scale(1);  
scale(2);  
scale(1.5);
```

# Scale

- データの種類によって色々なscaleが定義
  - linear – 連続データ, 線形対応
  - log – 連続データ, 対数目盛のイメージ
  - ordinal – 離散データ

```
var scale = d3.scale.ordinal().domain([ “hoge” , “nya” ,  
    “nyaa” ]).range([0, 1, 2]);  
scale( “hoge” );  
scale( “nya” );  
scale( “nyaa” );
```

# Scale

- 座標だけでなく色もrangeに指定できる

In [17] Nyaplot::Colors.jet

Out[17]:

rgb(215,48,39)	rgb(244,109,67)	rgb(253,174,97)	rgb(254,224,144)	rgb(255,255,191)	rgb(224,243,248)	rgb(171,217,233)	rgb(116,173,209)	rgb(69,117,180)
----------------	-----------------	-----------------	------------------	------------------	------------------	------------------	------------------	-----------------

```
var scale = d3.scale.linear().domain([0, 100]).range([ "#fff" ,  
  "#f00"  ]);  
scale( "30" );  
scale( "50" );  
  
var scale = d3.scale.category10();  
scale( "hoge" );  
scale( "nya" );  
scale( "hogehege" )
```

# Underscore.jsの基本 (おまけ)

# \_.map

- 第二引数のfunctionに要素を一つ一つ代入, 返り値を配列にまとめて返す

```
_.map([0, 1, 2], function(val){  
    return val*2;  
}); // -> [0, 2, 4]
```



# \_.reduce

- valには左から順番に要素の値が入る
- memoは最初は{}, 次からfunctionの返り値
  - {}, {'1':10}, {'1':10, '2':9}

```
_.reduce([10, 9, 6], function(memo, val, i){  
    memo[String(i)] = val;  
    return memo;  
}, {}); // -> { '1' :10, '2' :9, '3' :6}
```

# 便利関数色々

- max, min, each, isUndefined, isArray, etc.
- 詳しくは<http://underscorejs.org/>

補足

# Colorbrewer

- 用途別のカラーセット
- <http://colorbrewer2.org/>
- The Apache License

Out[17]:

rgb(215,48,39)	rgb(244,109,67)	rgb(253,174,97)	rgb(254,224,144)	rgb(255,255,191)	rgb(224,243,248)	rgb(171,217,233)	rgb(116,173,209)	rgb(69,117,180)

Out[5]:

rgb(141,211,199)	rgb(255,255,179)	rgb(190,186,218)	rgb(251,128,114)	rgb(128,177,211)	rgb(253,180,98)	rgb(179,222,105)	rgb(252,205,229)	rgb(217,217,217)

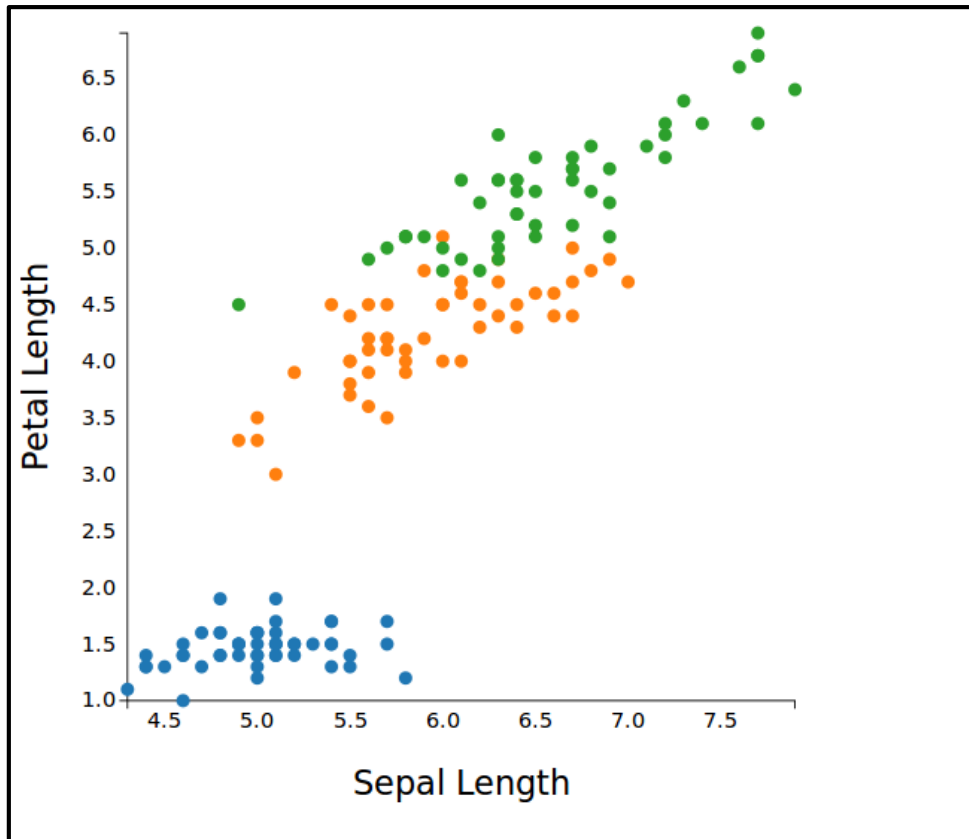
やってみよう

# Attention

- 腕に自信がある人は手元のpdfを見ながら先に進んでもらって構いません(後半の課題は解説しません)
- スライドに書いてある通りのことをやってもらわなくても結構です
- つまったら私かチューターに声をかけてください

# Assignments#-1

- iris.html



# Assignments#-1

- iris.csv
  - 言わずと知れた超有名データセット
  - あやめ(iris)の花びら(Petal)の長さと, がく片(Sepal)の長さに相関があるか調べる



Credit: katorisi | License: CC-BY-SA-3.0 GFDL



# Assignments#-1

- やること
  - データを解析してscaleを作る(★☆☆)
  - 散布図(scatter)を作る(★☆☆)
  - 種(Speciesごとに)色分けをする(★☆☆)
  - 軸にラベルをつける(★☆☆)

# Assignments#-1 scaleを作る

- 31行目のdomainを以下のように書き換えてみる

```
return d3.scale.linear().domain([-1, 3]).range([width, 0]);
```

# Assignments#-1 scaleを作る

- templates/iris.htmlの19行目にブレイクポイント
- 元のデータ(hpi)と加工後のデータ(data)を比較

The screenshot shows a web browser's developer console. On the left, a file explorer shows the project structure with files like `iris.html` and `u d3.v3.min.js`. The main console area displays a list of 14 objects (Object 6 through Object 14). Below this, a JavaScript error is shown: `Uncaught SyntaxError: Unexpected token < in kt/i.send/< d3.v3.min.js:1`. The error message is partially obscured by the file explorer. The console also shows the source code of `iris.html` at line 19, which is highlighted in blue. The code at line 19 is:

```
var width = 500, height = 500;
var padding = [80, 10];
```

The code continues with comments and DOM manipulation for a D3.js plot:

```
// prepare dom node
plot = d3.select(".plot").append("g").attr("transform", "translate(" + padding + ")");
plot.append("g").attr("class", "x axis").attr("transform", "translate(0, " + height + ")");
plot.append("g").attr("class", "y axis").attr("transform", "translate(0,0)");
plot.append("g").attr("class", "field").append("path");

// scales
```

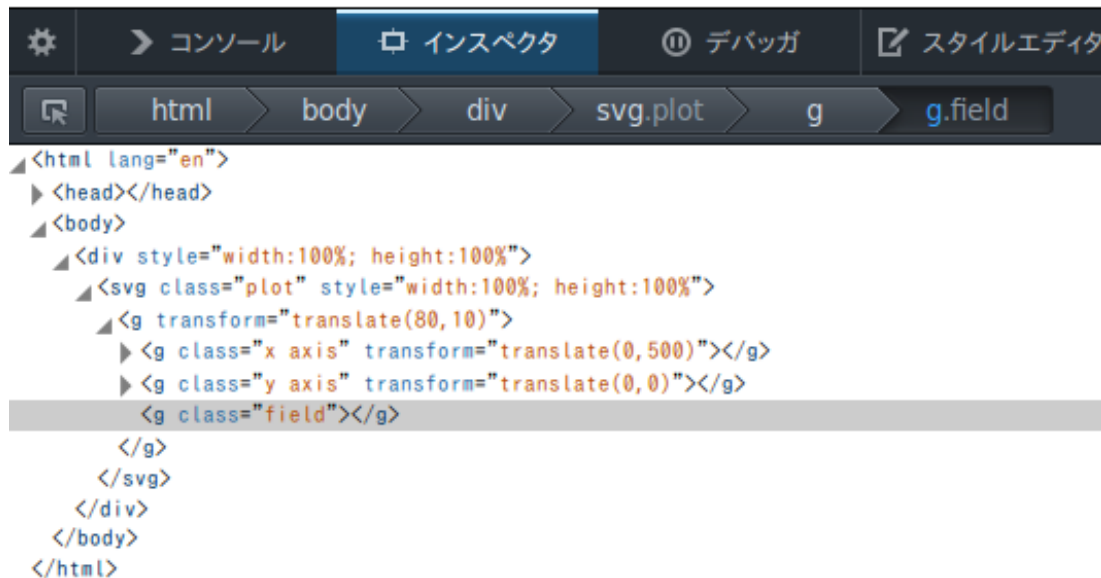
# Assignments#-1 scaleを作る

- 31, 36行目にブレークポイント, pl, slの中身を見してみる
- 両方ともdomainを正しく指定するだけで動くはず

```
_.min([0,1,2]); //-> 0  
_.max([0,1,2]); //-> 2
```

# Assignments#-1 散布図を作る

- インспекタを開く
  - class属性がfieldのgroupにcircleを追加すればよい



```
<html lang="en">
  <head></head>
  <body>
    <div style="width:100%; height:100%">
      <svg class="plot" style="width:100%; height:100%">
        <g transform="translate(80,10)">
          <g class="x axis" transform="translate(0,500)"></g>
          <g class="y axis" transform="translate(0,0)"></g>
          <g class="field"></g>
        </g>
      </svg>
    </div>
  </body>
</html>
```

# Assignments#-1 散布図を作る

- class属性がfieldのgroupにcircleを追加

```
d3.select( ".field" )  
    .selectAll( "circle" )  
    .data(data)  
    .enter()  
    .append( "circle" )  
    .attr({  
        'cx' :0,  
        'cy' :0,  
        'r'  :5,  
        'fill' : " #000"  
    });
```

# Assignments#-1 散布図を作る

- circleの位置を変える
  - dにはdataの要素が一つずつ入る
  - x, yは先ほど作ったscale

```
d3.select( ".field" )  
    .selectAll( "circle" )  
    .data(data)  
    .enter()  
    .append( "circle" )  
    .attr({  
        'cx' :function(d){return x(d.s1);},  
        'cy' :function(d){return y(d.p1);},  
        'r'   :5,  
        'fill' :'" #000"  
    });
```

# Assignments#-1 色分けをする

- color scaleを用意

```
var color_scale = d3.scale.category10();
```



# Assignments#-1 色分けをする

- 16行目, 元のデータからSpeciesの情報を抜き出してdataに入れておく

```
return { "sl":row['Sepal Length'], "pl":row['Petal Length']  
        "species" : ?};
```

# Assignments#-1 色分けをする

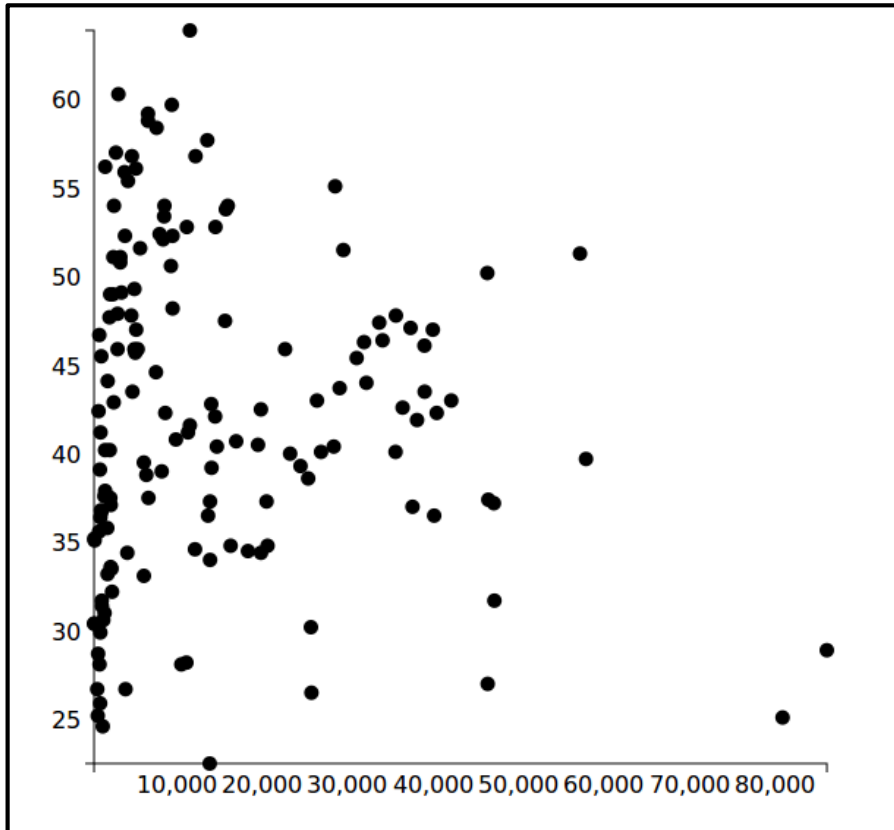
- fill属性を無名関数に変える.

```
d3.select( ".field" )  
    .selectAll( "circle" )  
    .data(data)  
    .enter()  
    .append( "circle" )  
    .attr({  
        'cx' :function(d){return x(d.sl);},  
        'cy' :function(d){return y(d.pl);},  
        'r'   :5,  
        'fill' : " #000"  
    });
```

# Questions?

# Assignments#0

- hpi.html



# Assignments#0

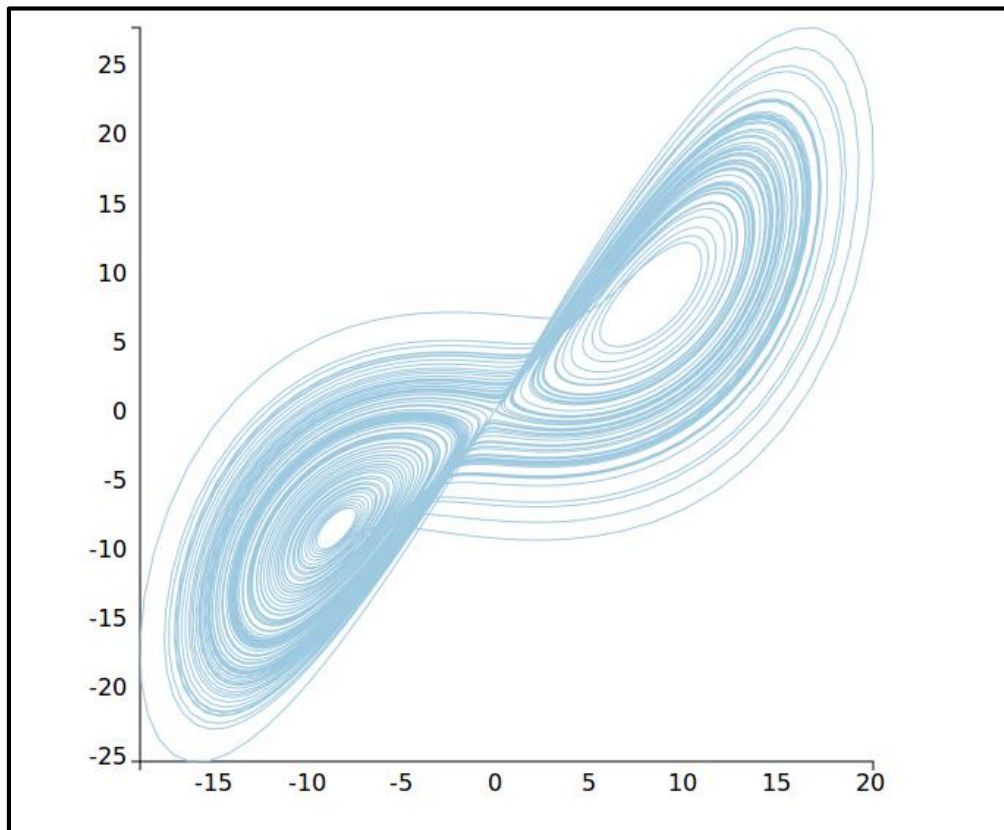
- hpi.csv
  - Happy Planet Index 地球幸福度指数
  - これからは経済力で競う時代じゃない, 大事なのは国民の幸福度だ! の指標の一つ.
  - GDPと人口(Population), HPIに相関がありそうか見てみる.
  - 出展: <http://www.happyplanetindex.org/>

# Assignments#0

- やること
  - Assignment#-1と同じように散布図を作成してみる。(☆☆☆)
    - GDPを横軸, HPIを縦軸にする。
- やること(Optional)
  - 人口で色分けしてみる。(★★☆)
    - d3.scale.linear()を使う。
    - Assignment#-1と違いちゃんとrangeとdomainを指定しないといけない

# Assignments#1

- [attractors.html](#)



# Assignments#1

- `attractors.html`
  - 有名なLorenz Attractor
  - オイラー法で微分方程式を解いてプロット



# Assignments#1

- やること
  - templates/attractors.htmlを完成させる
- やること(Optional)
  - setIntervalを使って時間によって線が伸びていくようにする(★★☆)
  - tによって線の色を変える(★☆☆)
  - Chuaなど別のアトラクタを実装する(☆☆☆)
  - アトラクタを切り替えられるようにする(★★★)

# Assignments#2

- wind.html



# Assignments#2

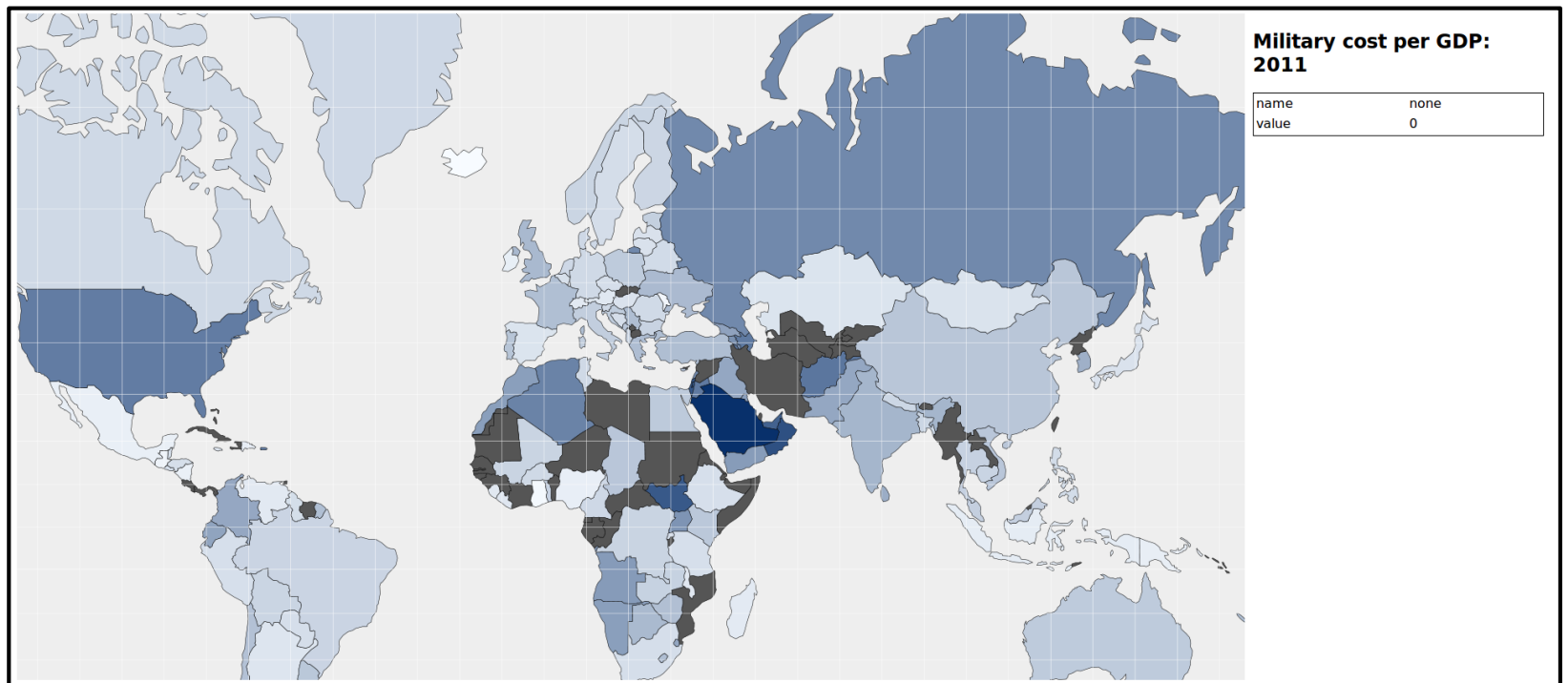
- wind.csv
  - NASAの全世界風向・風力データを手頃に加工したもの
  - lon, latは緯経度
  - uwnd, vwndの単位はm/s

# Assignments#2

- やること
  - templates/wind.htmlに付け足して地図の上に風向きを表す線をつけてみる.(★☆☆)
- やること(Optional)
  - 線の色を風量によって変える(★☆☆)
  - 線の先に矢印をつける等(★★☆)

# Assignments#3

- [gdp.html](#)



# Assignments#3

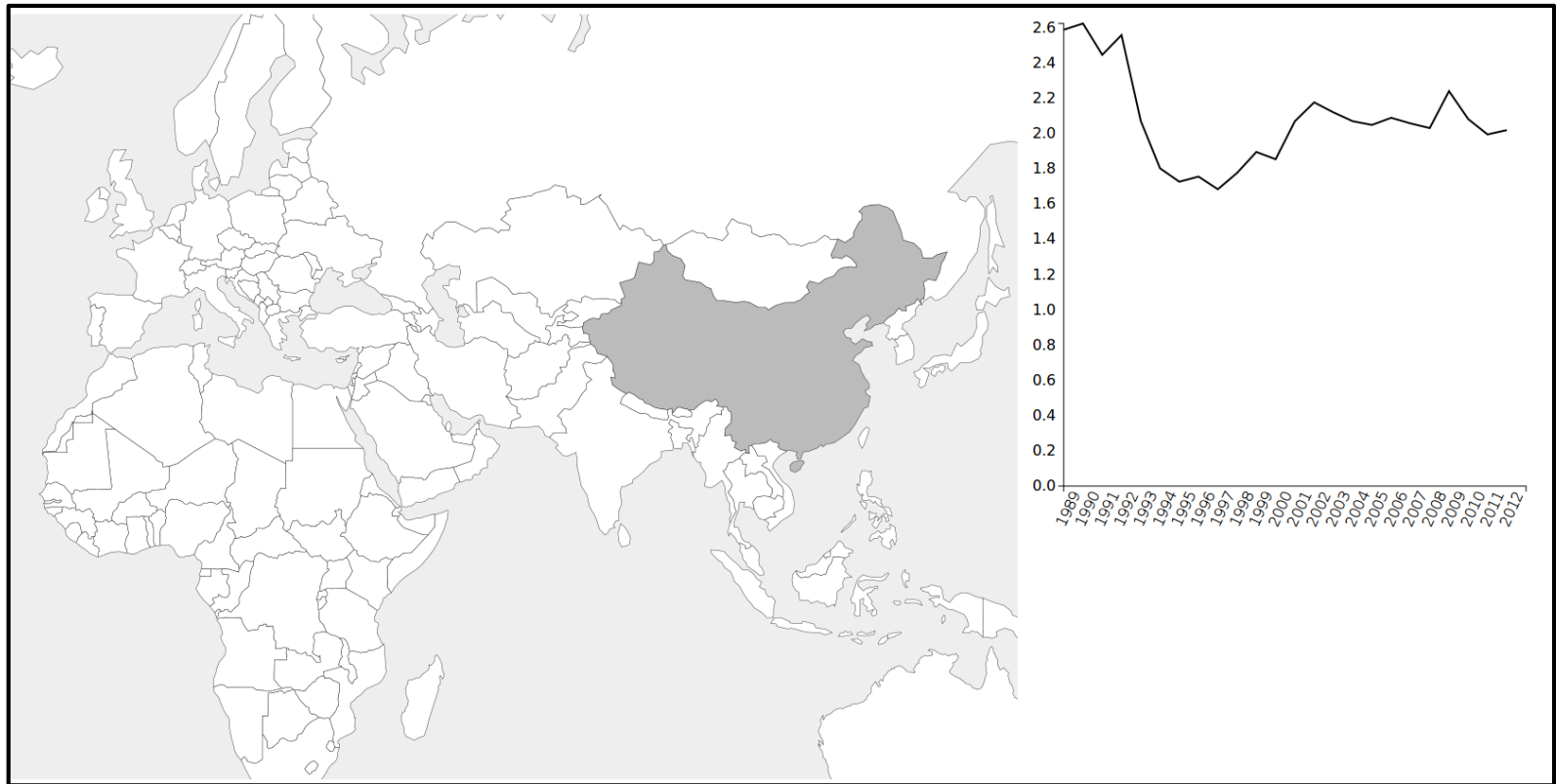
- gdp.csv
  - UN dataから, 軍事費の対GDP比を国, 地域別に集計したもの
  - 1980年代から2012年までばらばら
  - 国名もばらばら(できるかぎり修正しました)

# Assignments#3

- やること
  - templates/gdp.htmlを改変して軍事費によって地図を塗り分けてみる(★★☆)
- やること(Optional)
  - 左の地図をクリックしたら対応する国名と値を右の表に表示する(★★☆)
  - 表の下に年度による値の推移を折れ線グラフとして表示(★★★)

# Assignment#4

- `multiple_pane.html`





# Assignments#4

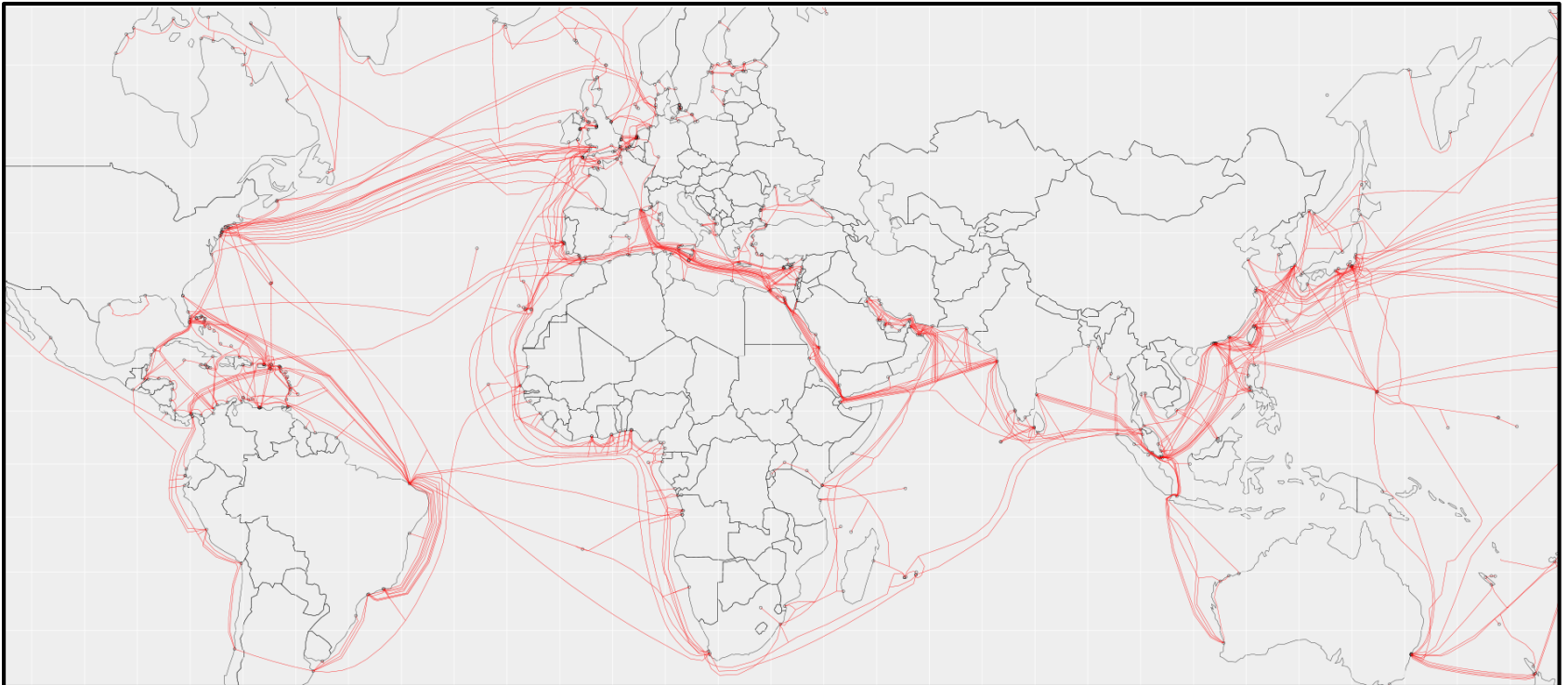
- gdp.csv
  - さっきと一緒に

# Assignments#4

- やること
  - templates/multiple\_pane.htmlを改変して左の地図をクリックしたら右に折れ線グラフを表示するようにする(★★☆)
- やること(Optional)
  - 複数の国を選択できるようにする(★★☆)
  - 国によって折れ線グラフ, 地図の色を変えるようにする(★☆☆)
  - 凡例を折れ線グラフの下に表示(★★☆)

# Assignment#5

- [cable.html](#)



# Assignment#5

- gregs\_cable\_map.geo.json
  - 全世界の海底ケーブルのデータ
  - ケーブルの座標と接続点の座標
  - データ元:
  - GPL v3

# Assignment#5

- やること
  - templates/cable.htmlを改造して海底ケーブルを地図上に表示する(☆☆☆)
- やること(Optional)
  - ケーブルによって色を変える(★☆☆)
  - 地名を地図上に表示する(★☆☆)
  - 各ポイントにマウスを乗せるとマウスの位置に地名が出るようにする等(★★★)