データ可視化 個人練習課題

ソーシャル・データサイエンス学部 2 年

5123017X　桑原 周平

1　本稿における可視化の目的

世界地図上の各国に対応する位置に、その国における”特定の年”において”特定の精神疾患”に悩まされる人の人口に占める割合を可視化することで、時系列による変化、場所による差異を明確化し、課題の所在を探る際に参照できるような図を作成する。

2　使用データの紹介

2.1　データの用意

Our World in Data の「Browse by topic」で「Health」→「Mental Health」を選択。そこの Interactive Charts on Mental Health の 2 ページ目に出てくる、”Mental illnesses prevalence, 2021”のテーブルを可視化することとした。こちらのデータは、列ラベルとして左から Country/area(国名)、Depressive disorders(鬱病)、Schizophrenia(統合失調症)、Bipolar disorder(双極性障害)、Eating disorders(摂食障害)、Anxiety disorders(不安障害) というそれぞれの精神障害に悩まされる人の国民に占める割合の推定値を、各国に対して 1990 年から 2021 年の 32 年間の各年における数値に分けて表示したものとなっている。32 年分の合計 32 個のテーブルを処理する良い方法が思いつかなかったため、今回は最初期の 1990 年のテーブルと、最新年度の 2021 年のテーブルの 2 つのみをピックアップし、それぞれをまずは「1990.csv」、「2021.csv」として保存した。さらに、世界地図上にこれらのデータを表示することを視野に入れ、各国の緯度、経度が参照できるファイルを探したところ、アマノ技研の「世界の首都の位置データ」のページから、国名 (幸い英語表記で 1990.csv、2021.csv と国名表記が一致)、首都名、首都の緯度、経度が分かるデータを用意した。

2.2　データの整形/統合

上述の通り、用意した csv ファイルは以下の 3 つ (ただし、 3 つ目は用意したデータから国名、首都名、緯度、経度以外の列を削除したもの)。

- 1990.csv
- 2021.csv
- location.csv

これら 3 つのデータを一つにまとめることで、各国のデータを 1990、2021 のメンタルヘルスのデータと位置情報にアクセスしやすくなることを期待し、次のように「data.csv」として統合した。こちらは、Country、Schizophrenia(1990)、Anxietydisorders(1990)、Bipolardisorder(1990)、Eatingdisorders(1990)、Schizophrenia(2021)、Anxietydisorders(2021)、Bipolardisorder(2021)、Eatingdisorders(2021)、Capital、Latitude、Longitude というラベルから成り、 上に列挙した 3 つのファイルのどれか一つにでも欠けている国は削除した。ゆえに欠損値を持たない。

3　可視化の流れ (processing 上で実行)

本節の手順

1. マップ上に、15 度ごとの緯線と経線と重なるように、手動で調整し、直線を引く
2. 15 度ごとの緯線と経線を近似的に 15 分割することで、1 度ごとの緯線、経線を引く。
3. 前の操作により、例えば東経 138 度、北緯 27 度に該当する点に円を描画することができるようになったので、これを応用し、各国の首都の緯度と経度を小数点以下切り捨てにより整数にし、その座標に円を描画する。
4. 描画した円のサイズが各国の Schizophrenia(1990) を参照した値と対応するようにする。
5. 同様に他の精神疾患や 2021 年のデータについても、その値に応じた円を正しい位置に描画する、ということが可能になったので、インタラクティブな可視化をする。

3.1 15 度ごとの線を描画

15 度ごとの緯線、経線が描かれているという理由で、今回は以下のマップを使用。

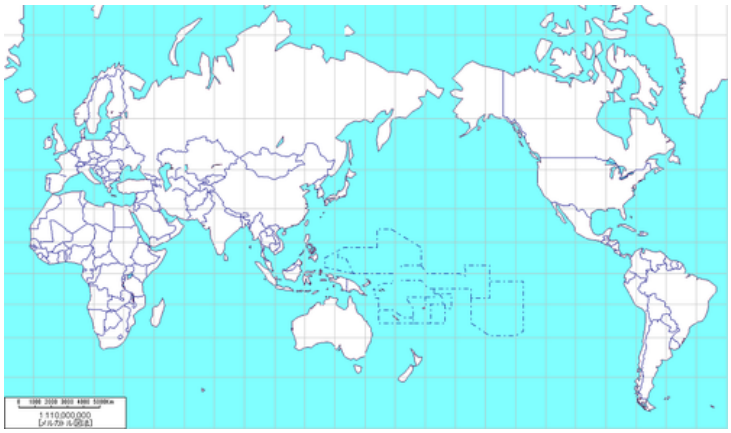

図 1　用いる地図画像

このマップに対し、processing 上では、提出フォルダ内の version1.pde に記載されたコードを実行した。これにより、次のように 15 度ごとの線をなぞることができた。

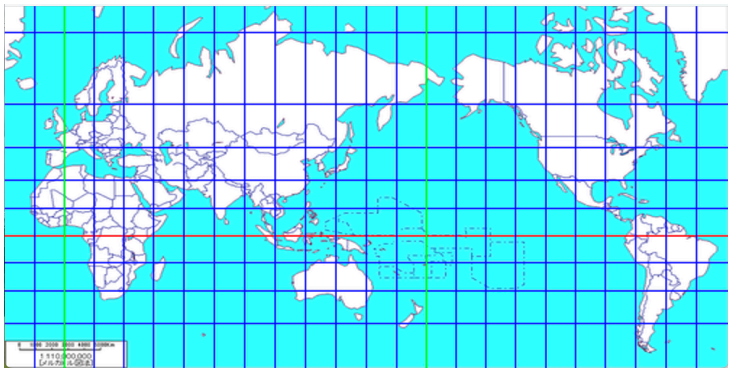

図 2　15 度線付き

3.2　15 分割の線を描画

手動で位置を調節し 15 度ごとの緯線と経線を引けたので、これらを全て隣り合う 2 本の間をさらに 15 分割し、1 度ごとの経線と緯線を設定する。これが近似的に今回の地図における座標となる。実行したコードは、提出フォルダ以内の version2.pde 参照。

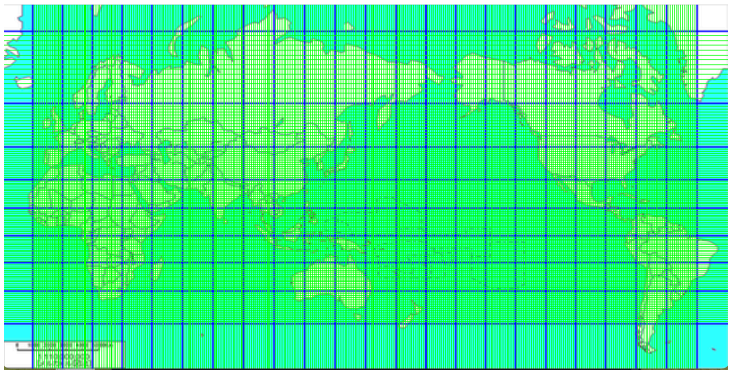

図 3　1 度ごとの緯線/経線つき

これにより、次のように 1 度ごとの線を引くことができた。ただし、当然線があまりに多いので、後々円による可視化を進める際には、noStroke で見えない線とし、円の位置を指定するための見えない座標として活用する。

3.3　円と国名を描画

地図上に座標を指定するための線は全て引くことができたので、それを用いて指定した位置に円を描画する。この時、国の位置データから取得される緯度と経度を小数点以下切り捨てしたため、多少のずれが想定された。実際の結果が下図。

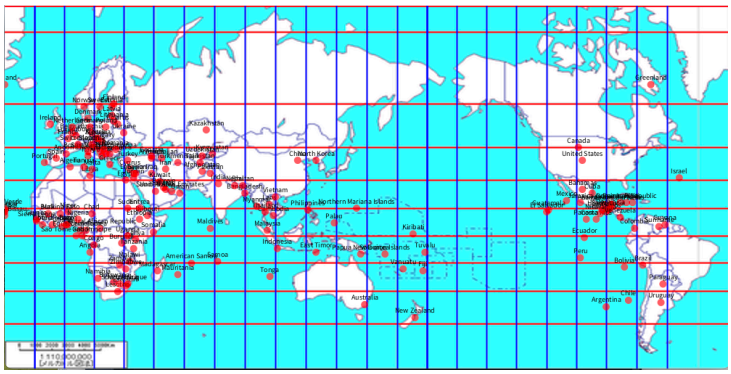

図 4　円の位置どり

3.4　データを参照し円のサイズに反映

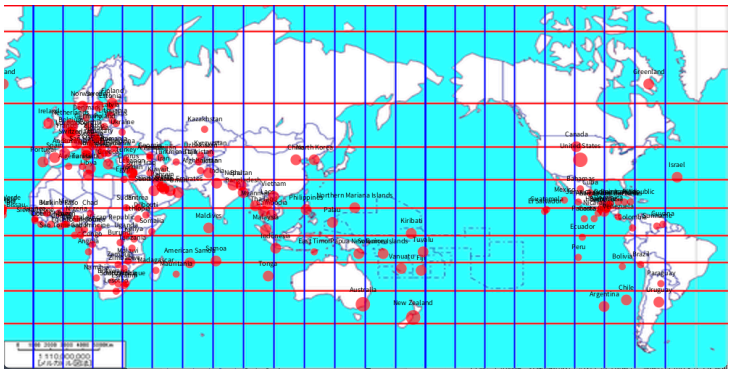

図 5　円のサイズ反映後

円のサイズを、実際にデータを参照して調整。その際、パーセント表示を数値に変換していなかったことが原因で円が表示されなくなってしまうことが続いて苦労した。

3.5　諸々の機能を追加して完成

諸々の機能 (具体的には工夫の項目で以下に言及) を追加し、完成。ソースコード、図は後に掲載。

4　工夫した点

1. 円が集中しているところは、どの国のデータなのか判別がつかない。そこで、円にカーソルを合わせると国名と参照した数値が表示されるようにした。
2. 大きさによる表現だけでなく、色分けにより視覚的によりわかりやすい図にした。
3. インタラクティブな操作により、1990 と 2021 年、精神疾患の選択などを可能にし、参照したい図へのアクセスを容易にした。具体的には、キーボードの d を押すと精神疾患が切り替わり、y を押すと年が切り替わるようになっている。つまり合計 10 種類の図にアクセスできるようになっている。

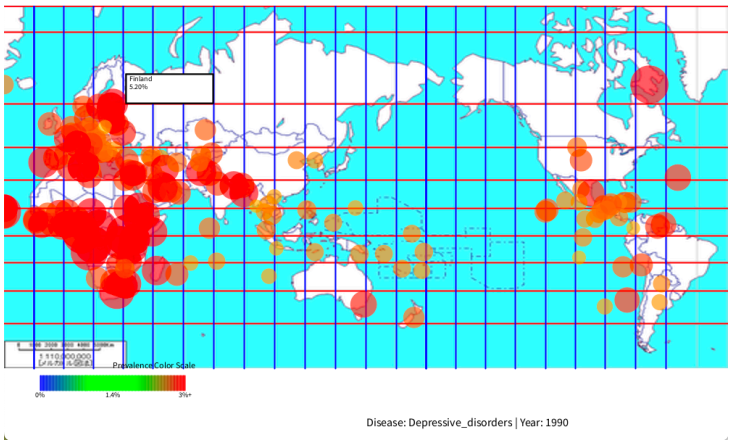

図 6　完成図 1

上の図では、1990 年のうつ病のデータを表示し、フィンランドの円にカーソルを合わせ、その国名とデータを表示している。

d や y を入力することにより、以下のように別の図を表示することも可能。

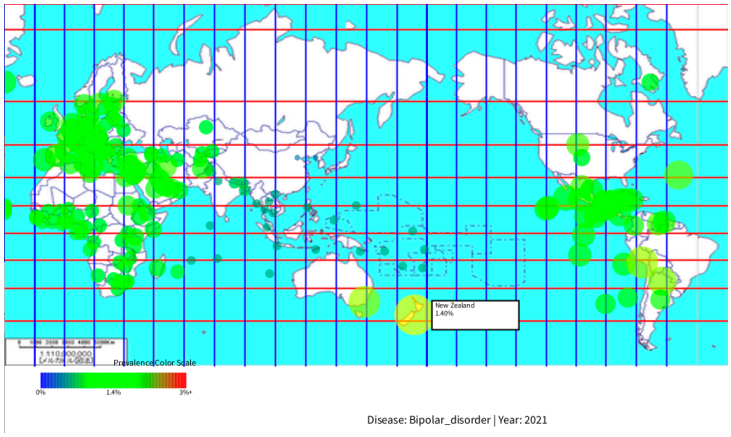

図 7　完成図 2

上の図では、2021 年の双極性障害のデータを表示し、ニュージーランドの円にカーソルを合わせ、その国名とデータを表示している。

5　processing で実行したコード (参考)

ソースコード 1　processing コード

```
1
2 PImage mapImage; // 地図画像
3
4 // 緯線・経線の座標リスト
5 float[] latitudeY;
```

```
6  float[] longitudeX;
7
8  Table data; // データを格納するオブジェクトTable
9  String[] diseases = { "Schizophrenia", "Depressive_disorders", "Anxiety_disorders", "Bi
10 int[] years = { 1990, 2021 };
11 int currentDiseaseIndex = 0;
12 int currentYearIndex = 1;
13
14 void setup() {
15   size(1000, 600); // サイズ調整して色分け表示領域を確保
16
17   // 地図画像を読み込む
18   mapImage = loadImage("map.png");
19
20   // 緯線と経線の位置を事前に計算して保存
21   calculateInvisibleGrid();
22
23   // ファイルを読み込むCSV
24   data = loadTable("data.csv", "header");
25
26   textAlign(CENTER, CENTER); // テキストの配置を中央揃え
27   textSize(10); // ラベルのフォントサイズ
28 }
29
30 void draw() {
31   background(255);
32
33   // 地図画像を描画
34   image(mapImage, 0, 0, width, height - 100);
35
36   // 太い緯線と経線を描画
37   drawMainLines();
38
39   // data.を参照して円をプロットcsv
40   plotCapitals();
41
42   // 現在の病気名と年を表示
43   displayCurrentSelection();
44
45   // 色分け基準の凡例を表示
```

```
46    drawColorLegend();
47  }
48
49  // 病気名と年を表示
50  void displayCurrentSelection() {
51    fill(0);
52    textSize(16);
53    text("Disease:" + diseases[currentDiseaseIndex] + "|Year:" + years[currentYearInde
54  }
55
56  // 色分け基準の凡例を表示
57  void drawColorLegend() {
58    float legendX = 50;
59    float legendY = height - 90;
60    float legendWidth = 200;
61    float legendHeight = 20;
62
63    textSize(12);
64    fill(0);
65    text("Prevalence Color Scale", legendX + legendWidth / 2, legendY - 10);
66
67    for (float i = 0; i <= 1; i += 0.01) {
68      int c = lerpColor(
69        lerpColor(color(0, 0, 255), color(0, 255, 0), constrain(i * 3, 0, 1)),
70        lerpColor(color(255, 255, 0), color(255, 0, 0), constrain((i - 0.33) * 3, 0, 1)),
71        constrain((i - 0.66) * 3, 0, 1)
72      );
73      stroke(c);
74      line(legendX + i * legendWidth, legendY, legendX + i * legendWidth, legendY + legend
75    }
76
77    fill(0);
78    textSize(10);
79    textAlign(CENTER);
80    text("0%", legendX, legendY + legendHeight + 10);
81    text("1.4%", legendX + legendWidth / 2, legendY + legendHeight + 10);
82    text("3%+", legendX + legendWidth, legendY + legendHeight + 10);
83  }
84
85  // 太い緯線と経線を描画する関数
```

```
86  void drawMainLines() {
87    strokeWeight(2);
88
89    // 太い緯線（赤道や度ごと）を描画15
90    stroke(255, 0, 0); // 赤い線
91    for (int i = 0; i <= 150; i += 15) {
92      float y = latitudeY[i]; // 度ごとの緯度座標15
93      line(0, y, width, y);
94    }
95
96    // 太い経線（度ごと）を描画15
97    stroke(0, 0, 255); // 青い線
98    for (int i = 0; i <= 360; i += 15) {
99      float x = longitudeX[i]; // 度ごとの経度座標15
100     line(x, 0, x, height - 100);
101   }
102 }
103
104 // 見えない格子（緯線・経線の座標）を計算する関数
105 void calculateInvisibleGrid() {
106   latitudeY = new float[151]; // ° ～° −7575 本(151)
107   longitudeX = new float[361]; // ° ～° −180180 本(361)
108
109   // 緯線の座標計算° から° まで (−7575)
110   float equatorY = (height - 100) / 2 + 67;
111   float north15Y = equatorY - ((height - 100) / 13);
112   float south15Y = equatorY + ((height - 100) / 13.2);
113   float north30Y = equatorY - ((height - 100) / 6.5);
114   float south30Y = equatorY + ((height - 100) / 6.5);
115   float north45Y = equatorY - ((height - 100) / 4.1);
116   float south45Y = equatorY + ((height - 100) / 4.1);
117   float north60Y = equatorY - ((height - 100) / 2.75);
118   float north75Y = equatorY - ((height - 100) / 1.78);
119
120   float[] majorLatitudeY = {
121     south45Y, south30Y, south15Y, equatorY, north15Y, north30Y, north45Y, north60Y, nort
122   };
123   int[] majorLatitudeDegrees = {
124     −45, −30, −15, 0, 15, 30, 45, 60, 75
125   };
```

```
126
127   for (int i = 0; i < majorLatitudeY.length - 1; i++) {
128     float startY = majorLatitudeY[i];
129     float endY = majorLatitudeY[i + 1];
130     for (int j = 0; j <= 15; j++) {
131       int degree = majorLatitudeDegrees[i] + j;
132       latitudeY[degree + 75] = map(j, 0, 15, startY, endY);
133     }
134   }
135
136   // 経線の座標計算° から° まで (−180180)
137   float west45X = 955;
138   float west60X = 913;
139   float west75X = 871;
140   float west90X = 829;
141   float west105X = 788;
142   float west120X = 747;
143   float west135X = 705;
144   float west150X = 664;
145   float west165X = 623;
146   float ew180X = 582;
147   float west15X = 41;
148   float longitudeXCoord = 82;
149   float east15X = 123;
150   float east30X = 164;
151   float east45X = 205;
152   float east60X = 247;
153   float east75X = 289;
154   float east90X = 331;
155   float east105X = 373;
156   float east120X = 415;
157   float east135X = 457;
158   float east150X = 499;
159   float east165X = 541;
160   float east180X = 582;
161
162   float[] majorLongitudeX = {
163     west45X, west60X, west75X, west90X, west105X, west120X, west135X, west150X, west165X
164     ew180X, west15X, longitudeXCoord, east15X, east30X, east45X, east60X, east75X, east9
165     east105X, east120X, east135X, east150X, east165X, east180X
```

```
166     };
167     int[] majorLongitudeDegrees = {
168       -45, -60, -75, -90, -105, -120, -135, -150, -165, -180, -15, 0, 15, 30, 45,
169       60, 75, 90, 105, 120, 135, 150, 165, 180
170     };
171
172     for (int i = 0; i < majorLongitudeX.length - 1; i++) {
173       float startX = majorLongitudeX[i];
174       float endX = majorLongitudeX[i + 1];
175       for (int j = 0; j <= 15; j++) {
176         int degree = majorLongitudeDegrees[i] + j;
177         longitudeX[degree + 180] = map(j, 0, 15, startX, endX);
178       }
179     }
180 }
181
182 // data.を参照して円をプロットする関数csv
183 void plotCapitals() {
184     String column = diseases[currentDiseaseIndex] + "_" + years[currentYearIndex];
185     for (TableRow row : data.rows()) {
186       String country = row.getString("Country");
187       float longitude = row.getFloat("Longitude");
188       float latitude = row.getFloat("Latitude");
189       String prevalenceStr = row.getString(column);
190       float prevalence = float(prevalenceStr.replace("%", "")) / 100.0;
191
192       float sizeMultiplier = 8000;
193       if (diseases[currentDiseaseIndex].equals("Depressive_disorders") || diseases[current
194         sizeMultiplier = 800;
195       } else if (diseases[currentDiseaseIndex].equals("Bipolar_disorder")) {
196         sizeMultiplier = 4000;
197       }
198       float size = prevalence * sizeMultiplier;
199
200       int intLongitude = int(floor(longitude));
201       int intLatitude = int(floor(latitude));
202       float x = longitudeX[intLongitude + 180];
203       float y = latitudeY[intLatitude + 75];
204
205       int c;
```