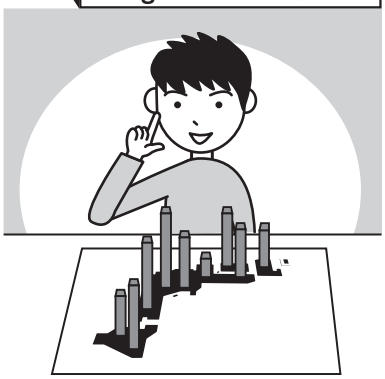
Google Colabで試せます



地理はデータ・サイエンス！

私も地図マスター

第4回　2つの統計情報を同時に地図上で確認する

小野原 彩香，岩崎 亘典

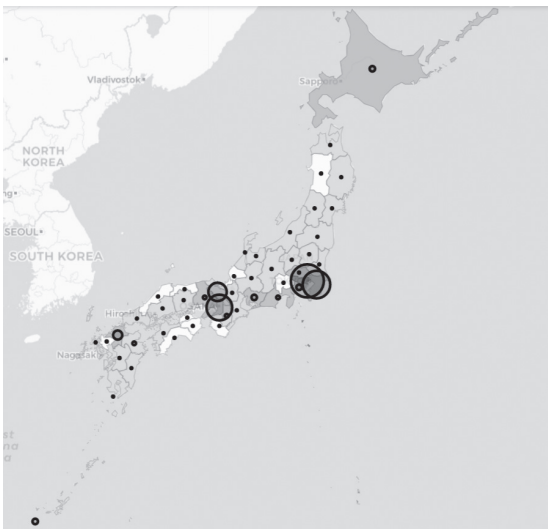


図1　都道府県別訪日外国人数マップ
都道府県別に訪日外国人数が円の大きさで示されている．人口の多さで色分けしている

最近は外国との往来が復活しつつあり，街で外国の方をよく見かけるようになりました．そもそも新型コロナ流行前までは，たくさんの外国の方が日本を訪れていました．外国の方々はどの都道府県にどれくらい訪れていたのでしょうか．

今回は，都道府県別訪日外国人数を円の大きさを使って地図上に表し，人口規模の多い大都市に集中するのかあるいは別の傾向が見られるのか可視化します．使用する言語はPython，開発環境はGoogle Colaboratoryを想定しています．詳しい使い方はサポート・ページをご覧ください．

都道府県別訪日外国人数を地図上に表したい

● 2つの統計データを同時に表示する

都道府県ごとの人口と訪日外国人数という，2つの統計情報を同時に地図上に表現します（**図1**）．

前回，コロプレスマップを作成しました．コロプレスマップとは，統計データを階級に分け，階級を色の濃淡や色で分ける図の描き方でした．今回はそこに，図形表現図を加えていきます．図形表現図とは，棒の長さや円の面積などで数量を表す地図です．

まず，必要なライブラリ（**リスト1**）をインストール，インポートしてください．

プログラムは本誌サポート・ページから入手できます．
https://interface.cqpub.co.jp/2305py/

リスト1　必要なライブラリ

```
#必要なモジュールのインポート
!pip install GDAL
!pip install geopandas
!pip install matplotlib
!pip install folium==0.12.1
!pip install pixiedus

import numpy as np
import pandas as pd
import urllib.request
import folium
from IPython.display import display
```

ステップ1…都道府県ごとの人口データ・マップの作成

政府統計のe-Stat（https://www.e-stat.go.jp/）の国勢調査データから都道府県ごとの人口データ(1)を取得し，人口の多さで色分けした地図（**図2**）を作成します．こちらについては，前回2023年3月号の記事を参考にして行ってください．以下に概要を説明します．

<第3回の記事はこちら>

▶①e-Satのウェブ・サイトでユーザ登録する

今回はe-StatのAPI機能を使ってデータを取得します．登録が必要なので次のウェブ・サイトからユーザ登録をしてください．

https://www.e-stat.go.jp/mypage/user/preregister

加えて，次のウェブ・サイトに従いアプリケーションIDの取得を行う必要もあります．このアプリケーションIDは，後に　Pythonコードの中に記述する必

図2　都道府県別人口比較マップ…色の濃い方が人口が多い

要があるので控えておきます．

https://www.e-stat.go.jp/api/apiinfo/api-guide

▶②APIを使って取得したいデータIDを調べる

国勢調査のデータをAPIで指定する番号は，https://www.e-stat.go.jp/statistics/00200521にあるように政府統計コードの欄に表示があります．この場合，国勢調査の番号は00200521です．

e-Statの仕様書より，ユーザ登録IDと国勢調査の番号などを指定した次のリクエストURLでデータのIDを調べることができます．このリクエストURLを生成するプログラムを**リスト2**（前回記事リスト2）に示します．今回は令和2年の国勢調査を使うので，IDは0003445078と分かりました．

▶③APIを使って，②の必要なデータのみ入手

データを地図上に表示させたいデータのみピックアップしてきます．このプログラムを**リスト3**（前回記事リスト4）に示します．

▶④③で取得したデータの読み出し，次の⑤で使いやすいように成形する

このプログラムを**リスト4**（前回記事のリスト5～リスト8）に示します．

▶⑤地図を作成し，地図中の都道府県ごとに人口に応じた色を塗る

リスト5（前回記事リスト9）で背景地図を作成し，**リスト6**（前回記事リスト10）で人口データを読み込み，**リスト7**（前回記事リスト11）で都道府県ごとに人口で色分けした地図を作成します．

リスト2　政府統計e-StatのAPIを使って取得したいデータIDを調べる

```
app_id = "　　e-Statの登録ID　　" #冒頭で取得したアプリケーションID
api_version = "3.0"
base_url = "https://api.e-stat.go.jp/rest/{API_version}/app/".format(API_version=api_version)

get_type = "getStatsList"
stats_code = "00200521" #国勢調査に割り当てられた番号
url = base_url + "{Get_type}?appId={appid}&statsCode={Stats_code}".format(Get_type=get_type,appid=app_id,
                                                                          Stats_code=stats_code)
print(url) # 確認して取得したいデータのIDを調べる
```

（a）ソースコード

```
https://api.e-stat.go.jp/rest/3.0/app/getStatsList?appId=　　e-Statの登録ID　　&statsCode=00200521
```

（b）実行結果

リスト3　必要なデータのみ入手

```
get_type="getSimpleStatsData"
stats_data_id="0003445078"  ← 令和2年国勢調査データのID
cd_cat_01="0" # 総数0，男1，女2
lv_area="2" # 集計レベル．全国レベル1，都道府県レベル2，市区町村レベル3
section_header_flg="2" # セクションヘッダー無し2
url = base_url + "{Get_type}?appId={Appid}&statsDataId={Stats_data_id}&cdCat01={cdcat01}&lvArea={
                 lv_Area}&sectionHeaderFlg={Section_header_flg}".format(Get_type=get_type,Appid=app_id,
          Stats_data_id=stats_data_id,cdcat01=cd_cat_01,lv_Area=lv_area, Section_header_flg=section_header_flg)
print(url)
```

データ指定に必要な要素

（a）ソースコード

```
https://api.e-stat.go.jp/rest/3.0/app/getSimpleStatsData?appId=　　e-Statの登録ID　　&statsDataId=
                                                                    0003445078&cdCat01=0&lvArea=2&sectionHeaderFlg=2
```

（b）実行結果

リスト4　データ読み出しと加工

```
d = urllib.request.urlopen(url).read().decode(
                                                    "utf8")
d

dlines = d.splitlines()[1:] #改行で分割
dlines

pcodes = []
names = []
populations = []

for line in dlines:
    line2 = line.replace('"', "").split(",")
    pcode = line2[4]
    name = line2[5]
    population = line2[9]
    pcode = pcode[0:2]
    population = int(population)
    pcodes.append(pcode)
    names.append(name)
    populations.append(population)

df = pd.DataFrame({"pcode" : pcodes,
       "name" : names, "population" : populations})
display(df)
```

ステップ2…
訪日外国人数のデータを読み込む

● データの読み込み

政府統計e-Statの訪日外国人消費動向調査の訪日外国人の都道府県別訪問人数（回答数）のデータ(2)を取得して読み込みます．最新版は2016年なのでそのデータを使います．ステップ1で準備した都道府県別人口データは令和2（2020）年のものですが，最新版でそろえることにします．

e-Statのトップ・ページで「訪日外国人消費動向調査」を検索すると，このデータの番号（政府統計コード）が00601030と分かります．**リスト2**のstats_codeを00601030と変更して実行します．得られたリクエストURLで表示された画面の中から，「集計結果　訪問地（都道府県47区分および地方運輸局等10区分）別　回答者属性および旅行内容　全体」の年確定値(3)を探します．このデータのIDは0003317305だと分かります．**リスト8**にデータ読み込みのプログラムと実行結果を示します．

リスト9で，読み込んだデータを使いやすいようにデータ項目で改行しています．

リスト5　背景地図作成

```
location = [32.99125000,138.45999999]
                                        #地図の中心位置を指定
tiles='CartoDB positron' #背景地図の指定
zoom_start = 5 #ズームレベル
map = folium.Map(location=location, tiles=tiles,
                                  zoom_start=zoom_start)
map
```

リスト6　人口データ読み込み

```
!pip install geojson
import geojson
import geopandas as gpd
jpn = "https://github.com/wata909/interface2022/
                      raw/main/GIS_DATA/japan.geojson"
fjpn = gpd.read_file(jpn) #ベクタファイルの読み込み
fjpn #読み込んだデータの確認
```

リスト7　都道府県ごとに人口に応じて色分け

```
# 地図に階級ごとに色を塗る
folium.Choropleth(
geo_data=jpn, #都道府県ごとの緯度経度情報のあるgeojsonデータ
name='choropleth',
data=df, #描画データ
columns=["pcode", "population"],
                                  # ["都道府県コード", "値の列"]
key_on='properties.pref_code',
threshold_scale=[500000, 1000000, 3000000,
10000000, 15000000],
fill_color='YlGnBu',  #色分けするための色指定
fill_opacity=0.7,  #塗りつぶしの濃さ
line_opacity=0.2, #境界ラインの濃さ
).add_to(map)
display(map)
```

リスト8　訪日外国人の都道府県別訪問人数のデータの読み込み

```
#都道府県別にデータを読み込む。手順は人口の場合と同様
get_type="getSimpleStatsData"
stats_data_id="0003317305" ←（データのID）
cd_cat_01="100" # 総数
cd_Time="2016000000" #2016年
cd_Tab=100 #特定の項目コード
section_header_flg="2" # セクションヘッダー無し＝2
url2 = base_url + "{Get_type}?appId={Appid}&statsDataId={Stats_data_id}&cdCat01={cdcat01}&cdTime={cd_Time}&
      cdTab={cd_Tab}&sectionHeaderFlg={Section_header_flg}".format(Get_type=get_type,Appid=app_id,Stats_data_id=
           stats_data_id,cdcat01=cd_cat_01,cd_Time=cd_Time,cd_Tab=cd_Tab, Section_header_flg=section_header_flg)
print(url2) #リンク先にエラーが出ないか確認する
```

（**a**）ソースコード

```
https://api.e-stat.go.jp/rest/3.0/app/getSimpleStatsData?appId=　e-Statの登録ID　&statsDataId=
                                0003317305&cdCat01=100&cdTime=2016000000&cdTab=100&sectionHeaderFlg=2
```

（**b**）実行結果

リスト9　使いやすいように改行

```
d2 = urllib.request.urlopen(url2).read().decode("utf8")
d2lines = d2.splitlines()[1:] #改行で分割
d2lines #データ内容確認
```

（a）ソースコード

```
['"100","回答数","100","全体","00100","全体","00100","北海道","2016000000","2016年","人","2720",""',
 '"100","回答数","100","全体","00100","全体","00110","青森県","2016000000","2016年","人","157",""',
…
 '"100","回答数","100","全体","00100","全体","00650","九州運輸局","2016000000","2016年","人","6066",""',
 '"100","回答数","100","全体","00100","全体","00660","沖縄総合事務局","2016000000","2016年","人","2561",""']
```

（b）実行結果

リスト10　訪日外国人の都道府県別訪問人数のデータの取り出し

```
pcodes2 = []
names2 = []
populations2 = []

for line2 in d2lines:
    line2_2 = line2.replace('"', "").split(",")
    pcode2 = line2_2[6]
    name2 = line2_2[7]
    population2 = line2_2[11]
    population2 = int(population2)
    pcodes2.append(pcode2)
    names2.append(name2)
    populations2.append(population2)

df2 = pd.DataFrame({"code2" : pcodes2, "name" :
            names2, "population2" : populations2})
df2 #データ内容確認
```

表1　道府県コード，都道府県名，実際の訪日外国人数のデータ列のみを取り出した

–	code2	name	population2
0	100	北海道	2720
1	110	青森県	157
⋮	⋮	⋮	⋮
55	650	九州運輸局	6066
56	660	沖縄総合事務局	2561

必要な情報である都道府県コード，都道府県名，実際の訪日外国人数のデータ列のみを取り出します（**リスト10**）．取り出したデータを**表1**に示します．

● 訪日外国人数を表す円の表示位置を決める

今回，都道府県の境界を外周とする多角形として地図を表現します．線で囲まれた多角形の面のことをポリゴンと言います．**リスト6**の引数fjpnには都道府県47個のポリゴン・データが格納されています．fjpnとして読み込んだgeojsonファイルの形式がポリゴン形式となっています．

各都道府県の境界内のどこかに訪日外国人のデータを表示させたいので，今回は都道府県の面（ポリゴン）の重心位置にそれを表示させます．その場所を特定するために重心を計算します．

fjpnに入っているデータは緯度経度座標系なので，*XY*座標系に変更した後，ポリゴンの重心を求めます．緯度経度座標系のまま重心を求めようとするとエラーが出るので注意してください．裏で実行されている計算アルゴリズムとデータ形式が合致しないためエラーが出てしまうのです．

リスト11　ベースとなるmapのCRSを確認

```
map.crs #mapのCRS確認
```

（a）ソースコード

```
'EPSG3857'
```

（b）実行結果

リスト12　ポリゴンの重心を求める

```
#一度緯度経度座標系をXY座標系に変換
fjpn.crs
fjpn = fjpn.to_crs(3857) #mapと同じCRSを設定
point = fjpn.centroid #ポリゴンの重心を求める
point.crs #pointのCRS確認
```

リスト13　データ内容の確認

```
point #データ内容確認
```

（a）ソースコード

```
0     POINT (15274159.486 4168980.823)
1     POINT (15629875.460 4830698.700)
⋮
46    POINT (15429819.511 4247920.089)
dtype: geometry
```

（b）実行結果

重心を求める一連のプログラムを**リスト11**～**リスト14**に示します．

まず，ベースとなる，都道府県ごとに人口に応じて色分けしたデータ（**リスト7**で引数mapに格納されている）のCRS（Coordinate Reference System：座標参照系）を確認します（**リスト11**）．CRSとはGISにおける位置の情報を決定するためのルールです．**リスト11**（a）の実行結果としてEPSG3857と表示されました．ここで表示されたEPSG3857がCRSです．これと同じCRSを**リスト12**でfjpnに設定し，緯度

リスト14　CRSを元のCRSに戻しておく

```
point4326 = point.to_crs(4326)
columns = ['point']
point4326 = pd.DataFrame(point4326,columns=columns)
point4326
```

表2
再びCRSを元のCRSに戻したデータ

	point
0	0POINT (137.21011 35.03752)
1	1POINT (140.40556 39.75704)
⋮	⋮
45	45POINT (131.57147 34.20181)
46	46POINT (138.60843 35.61606)

リスト15　人口のデータ・フレームと訪日外国人のデータ・フレームを合併

```
#dfにdf2を合併。都道府県名をキーにする
df3= pd.merge(df, df2,
          how="inner", on = "name")
df3 #データの内容確認
```

経度座標系をXY座標系に変換して，ポリゴンの重心を求めます．**リスト13**でデータ内容の確認をし，**リスト14**で再びCRSを元のCRSに戻しておきます．この結果を**表2**に示します．

ステップ3…人口と訪日外国人数のデータを1つにまとめる

リスト15で人口のデータ・フレームdfと訪日外国人のデータ・フレームdf2を都道府県名（＝name）を元に合併します．この結果を**表3**に示します．

リスト16で，後の処理のために都道府県を示すコード番号のデータの列名pcodeをpref_codeへ変更します．

リスト17で，各データの準備ができたので元の人口データ（fjpn）に重心データ（point4326）を合併します．この結果を**表4**に示します．

さらに，**リスト18**で観光客数（df3）をresultに合併します．この結果を**表5**に示します．また，次のコードでデータ構造を確認します．

```
result.dtypes #データ構造を確認
```

ステップ4…図形表現図の完成！

これでデータの準備が整ったので，Circle Marker関数を使って，元のコロプレスマップに訪

表3　人口のデータ・フレームと訪日外国人のデータ・フレームを合併させた

−	pcode	name	population	code2	population2
0	1	北海道	5224614	100	2720
1	2	青森県	1237984	110	157
⋮	⋮	⋮	⋮	⋮	⋮
45	46	鹿児島県	1588256	550	314
46	47	沖縄県	1467480	560	2561

リスト16　データ名をpcodeからpref_codeへ変換

```
df3 = df3.rename(columns={'pcode': 'pref_code'})
                                #あとの処理のために列名を変更
df3 #データの内容確認
```

リスト17　元のfjpnにpoint4326を合併させる

```
#point4326を元のfjpnに合併
result = pd.concat([fjpn,point4326], axis=1)
result #データの内容確認
```

リスト18　観光客数を結果に合併

```
#観光客数を合併
result = pd.merge(result, df3,
          how="inner", on = "pref_code")
result #データの内容確認
```

表4　元のfjpnにpoint4326を合併した

−	coc	nam	CODE	pref_code	geometry	point
0	JPN	AICHI	23	23	MULTIPOLYGON Z (((15246418.424 4220358.017 0.0...	POINT (137.21011 35.03752)
⋮	⋮	⋮	⋮	⋮	⋮	⋮
46	JPN	YAMANASHI	19	19	MULTIPOLYGON Z (((15403311.369 4296695.856 0.0...	POINT (138.60843 35.61606)

表5　1つにまとめた人口と訪日外国人数のデータ

−	coc	nam	CODE	pref_code	geometry	point	name	population	code2	population2
0	JPN	AICHI	23	23	MULTIPOLYGON Z (((15246418.424 4220358.017 0.0...	POINT (137.21011 35.03752)	愛知県	7542415	320	2836
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
46	JPN	YAMANASHI	19	19	MULTIPOLYGON Z (((15403311.369 4296695.856 0.0...	POINT (138.60843 35.61606)	山梨県	809974	280	1488

リスト19　都道府県別訪日外国人数マップの作成

```
for i,row in result.iterrows():
  tooltip = '外国人観光客数' + '<br>' +
str(row['population2']) +'人' +'<br>' + row['name']
  folium.CircleMarker(location=[row.point.y,
                                row.point.x],
        radius = row.population2/1000,
        tooltip =tooltip,
        color="black",
        fill_color="black"
).add_to(map)

display(map)
```

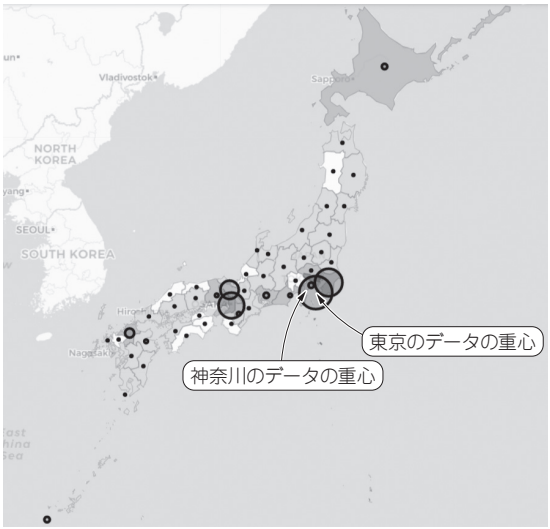


図3　東京への訪日外国人数を示す円の中心（重心）**が南に寄りすぎている**

日外国人数を表す円を重ねて表示します．

このプログラムを**リスト19**に示し，結果のコロプレスマップを**図3**に示します．**リスト19**ではデータを1行ずつ参照し，そのデータを使って円を描くので，forループ文とiterrows()を組み合わせます．locationは円を描くための中心座標で，これが先ほどpointとして作成した各都道府県の重心に該当します．radiusは円の大きさ，tooltipは対象の場所にマウスをオーバ・レイしたときに表示される対象についての説明の内容を示します．radiusに直接訪日外国人数を入れると，円が大きくなりすぎるので，1000で割って調整してあります．tooltipのコードでは「外国人観光客数」＋resultのpolutaion2の値（＝APIによって取得した外国人訪問数の具体的な数字）＋都道府県名が表示されるように設定してあります．colorは円の外周線の色，fill_colorは塗りつぶしの色です．ここでは黒を指定します．

▶重心の修正

東京には八丈島など中心部とは離れた地域にも所属

リスト20　東京の重心を修正

```
#東京都の場所がおかしいので、そこだけ変更
print(result.iloc[40,5]) #東京は４０行目にデータがある
from shapely.wkt import loads
string = 'POINT(139.7592747 35.6850655)'
geom = loads(string)
result.iloc[40,5]= geom
print(result.iloc[40,5])
```

（a）ソースコード

```
POINT (139.55064698090615 35.142159431705494)
POINT (139.7592747 35.6850655)
```

（b）実行結果

地域があるため，東京の重心を求めると，神奈川より南側に重心がきてしまいます（**図3**）．これを修正します（**リスト20**）．

▶図の再作成

この状態で次に**リスト19**のみを実行してしまうと，修正前の地図に上書きされてしまうので，**リスト5**と**リスト7**を実行した上で，**リスト19**を実行します．これで無事，**図1**に示した目的の地図が作成できました．

▶図の考察

図1から，やはり人口の多い地域に，訪日外国人も多く訪問している傾向が見て取れます．しかし，それほど人口の多くない京都府にも多くの外国の方が訪れているのが読み取れるでしょう．やはり，古都である京都府は一度は訪れてみたい魅力的な場所なのでしょうか．また，千葉県が人口に対して多少多いのは，東京ディズニー・リゾートへの訪問者でしょうか．いろいろと想像が膨らみます．細かい部分が分かりにくい場合は，拡大できるのがfoliumパッケージを用いた地図の利点です．

＊

次回は，今回の内容をさらに応用し，2つの統計情報について別の地図表現方法について探究します．

◆参考・引用*文献◆

(1) *国勢調査データ.
https://www.e-stat.go.jp/stat-search?page=1&toukei=00200521&survey=%E5%9B%BD%E5%8B%A2%E8%AA%BF%E6%9F%BB

(2) 訪日外国人消費動向調査.
https://www.e-stat.go.jp/statistics/00601030

(3) *訪日外国人消費動向調査　集計結果　訪問地（都道府県47区分および地方運輸局等10区分）別　回答者属性および旅行内容　全体.
https://www.e-stat.go.jp/dbview?sid=0003317305

おのはら・あやか，いわさき・のぶすけ