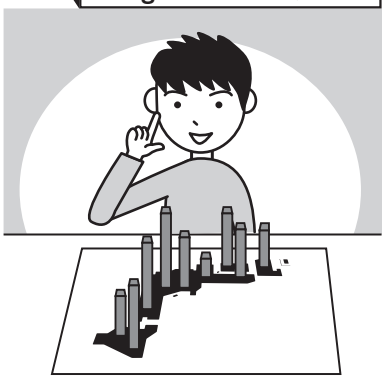
Google Colabで試せます



地理はデータ・サイエンス！

私も地図マスター

第3回　国勢調査の人口分布データを入手して統計地図を作ってみよう

小野原 彩香，岩崎 亘典

図1　地図の上に人口分布を重ね表示した階級区分図

今回からは，地図に統計データをもりこんだ，いわゆる主題図の描き方を学びます．主題図という名前は聞き慣れない人もいるかと思いますが，身近なところでは，毎日のように報道されている，都道府県ごとの新型コロナ感染者数のマップなどです．

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．

リスト1　必要なライブラリのインストールとインポート

```
!pip install GDAL
!pip install geopandas
!pip install matplotlib
!pip install folium
!pip install pixiedust

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

● 統計データを地図上で表現してみる

主題図のうち，コロプレスマップ（階級区分図）は白地図に別の情報を付け加えて，情報の大小や種類によって色分けして見やすくしたものです．領域ごとの違いが一目で把握できる事から，さまざまな分野で利用されています．今回は、政府統計のe-Statにある国勢調査データをAPIで取得して，コロプレスマップを作成します（**図1**）．

地図の作成に必要な準備

● 政府統計のe-StatのAPI利用には登録が必要

コロプレスマップに使用するデータを加工するのは意外と手間が掛かります．そこで，APIの仕組みを導入します．整理したデータを，必要な部分だけ持ってこられるのがAPIです．今回はAPIでデータを取得し，そのデータをコロプレスマップの形式で表示させる地図を作ります．

今回は政府統計のe-Stat（`https://www.e-stat.go.jp/`）のAPIを利用して，国勢調査のデータを入手します．API機能を使うためには，登録が必要なので，

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

からユーザー登録をしてください．また，

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

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，のちにPythonコードの中に記述する必要があるので控えておきます．

● 必要なライブラリ

必要なライブラリをインストールし，インポートします（**リスト1**）．ライブラリとは，これからの作業に必要なプログラムを全部入れた道具箱のようなものです．以下に各々のライブラリについて説明します．

①GDAL：地理情報のラスタとベクタを取り扱う
②geopandas：pandasというデータ解析系に特化し

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/2303py/

リスト2　取得したいデータのIDを調べる

```
app_id = "                                                    "
#冒頭で取得したアプリケーションID                    e-Statの登録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}&statsCod
e={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
```

リスト3　国勢調査データを入手

```
get_type="getSimpleStatsData"
stats_data_id="0003445078" # 令和2年の国勢調査ID
url = base_url + "{Get_type}?appId={Appid}&statsDat
      aId={Stats_data_id}".format(Get_type=get_type,
          Appid=app_id, Stats_data_id=stats_data_id)
print(url) # URLが正しく生成されるか確認する
```

たライブラリの拡張版で，幾何学的（図形や空間）な解析を可能にする
③matplotlib：グラフ描画
④folium：leaflet.jsというインタラクティブな地図の描画を可能にするJavaScriptで書かれたライブラリをPyhton環境に導入する
⑤pixiedust：データをGUIのように可視化できる

APIを使ってデータを取得

● まず取得したいデータのIDを調べる

今回はe-Stat APIのバージョン3.0を使用しています．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の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．もし，他の調査データ，例えば，労働力調査であれば00200531です．e-Statの仕様書では，

```
http(s)://api.e-stat.go.jp/rest/<バージョン>/app/getStatsList?<パラメータ群>
```

としてリクエスト用URLを指定するように指示があります．従って**リスト2**のように指定して，統計表情報に関する以下のリクエスト用URLを取得して，確認して取得したいデータのIDを調べられます．

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

● 国勢調査データを入手

今回は，令和2年国勢調査 人口等基本集計（主な内容：男女・年齢・配偶関係，世帯の構成，住居の状態，母子・父子世帯，国籍など）の中の，男女別人口－全国，都道府県，市区町村［2000年（平成12年）市区町村含む］のデータを使用することにします．従って，国勢調査の中の個別の調査データを参照するURLを生成します．このコードを**リスト3**に示し，以下に説明します．

▶使用する調査データ番号を指定

2行目のstat_data_idの箇所で使用する調査データ番号を指定します．こちらは国勢調査全体のリスト（**リスト2**の実行結果）で番号を探すこともできます．一方，e-Statのサイト（https://www.e-stat.go.jp/dbview?sid=0003445078）のように，直接，e-Stat内を検索して，表示したいデータを検索することもできます．

▶データ形式を選ぶ

e-StatのAPIでは，データ形式はXML，JSON，JSONP，CSVの4つを選ぶことができます（2022年11月現在）．今回，データ形式はCSVを選択しています．

リスト3を実行すると以下のリクエスト用URLが生成されます．

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

e-Statの仕様書で指定されたリクエスト用URLの形式は以下のようになっているので，getSimpleStatsDataの部分をgetStatsDataと変更すればXML形式，json/getStatsDataとすればJSON形式，jsonp/getStatsDataとすればJSONP形式でデータを取得できます．

```
http(s)://api.e-stat.go.jp/rest/<バージョン>/app/getSimpleStatsData?<パラメータ群>
```

● 取得したデータから必要なものをピックアップ

都道府県ごとの人口を地図上に表示させるため，リンク先のデータから地図表示に必要なデータのみをピックアップしてきます．このコードを**リスト4**に示します．

リスト4の3行目のcd_cat_01で，総数の人口を表示するか，男性の人口にするか，女性の人口にするかを指定します．また，4行目のlv_areaは集計単位で，今回は都道府県ごとの人口を指定します．5行目のsection_header_flgは，セクションヘッダで今回は無しにしておきます．

リスト4を実行すると次のURLが生成されます．

リスト4　地図表示に必要なデータのみをピックアップ

```
get_type="getSimpleStatsData"
stats_data_id="0003445078"
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}&statsDat
    aId={Stats_data_id}&cdCat01={cdcat01}&lvArea={l
    v_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

リスト5　取得したURLの中身を読み出す

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

（a）ソースコード

```
"tab_code","表章事項","cat01_code","男女","area_code","
全国，都道府県，市区町村（2000年市区町村含む）","time_code","時間
軸（年次）","unit","value","annotation""2020_01","人口","
0","総数","01000","北海道","2020000000","2020年","人","
5224614""2020_01","人口","0","総数","02000","青森県","
2020000000","2020年","人","1237984""2020_01","人口","
0","総数","03000","岩手県","2020000000","2020年","人
","1210534""2020_01","人口","0","総数","04000","宮城
県","2020000000","2020年","人","2301996""2020_01","
人口","0","総数","05000","秋田県","2020000000","2020年","
人","959502""2020_01","人口","0","総数","06000","山形
県","2020000000","2020年","人","1068027""2020_01","
人口","0","総数","07000","福島県","2020000000","2020年","
人","1833152""2020_01","人口","0","総数","08000","茨城
県","2020000000","2020年","人","2867009""2020_01","
人口","0","総数","09000","栃木県","2020000000","2020年","
人","1933146",""
⋮
（以下省略）
```

（b）実行結果

表1　APIで取得した人口分布のデータ

pcode	name	population	
0	1	北海道	5224614
1	2	青森県	1237984
2	3	岩手県	1210534
3	4	宮城県	2301996
4	5	秋田県	959502
5	6	山形県	1068027
6	7	福島県	1833152
7	8	茨城県	2867009
8	9	栃木県	1933146
9	10	群馬県	1939110
10	11	埼玉県	7344765
11	12	千葉県	6284480
12	13	東京都	14047594

（以下省略）

リスト6　次にデータを加工しやすい形に成形（改行で分割）

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

（a）ソースコード

```
['"2020_01","人口","0","総数","01000","北海道",
            "2020000000","2020年","人","5224614",""',
 '"2020_01","人口","0","総数","02000","青森県",
            "2020000000","2020年","人","1237984",""',
 '"2020_01","人口","0","総数","03000","岩手県",
            "2020000000","2020年","人","1210534",""',
 '"2020_01","人口","0","総数","04000","宮城県",
            "2020000000","2020年","人","2301996",""',
 '"2020_01","人口","0","総数","05000","秋田県",
             "2020000000","2020年","人","959502",""',
 '"2020_01","人口","0","総数","06000","山形県",
            "2020000000","2020年","人","1068027",""',
 '"2020_01","人口","0","総数","07000","福島県",
            "2020000000","2020年","人","1833152",""',
 '"2020_01","人口","0","総数","08000","茨城県",
            "2020000000","2020年","人","2867009",""',
 '"2020_01","人口","0","総数","09000","栃木県",
            "2020000000","2020年","人","1933146",""',
 '"2020_01","人口","0","総数","10000","群馬県",
            "2020000000","2020年","人","1939110",""',
 '"2020_01","人口","0","総数","11000","埼玉県",
            "2020000000","2020年","人","7344765",""',
 '"2020_01","人口","0","総数","12000","千葉県",
            "2020000000","2020年","人","6284480",""',
⋮
（以下省略）
```

（b）実行結果

リスト7　次にデータを加工しやすい形に成形［表頭の3種類のラベル（**表1**の列ラベル）を用意］

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

リスト8　最終的に引数dfにデータを渡す

```
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)
```

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

リスト9　ベースとなる地図の作成

```
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10　都道府県境界のあるベクタ・データの読み込み

```
!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
```

● 取得したデータの取り出し

取得したURLの中身を読み出します（**リスト5**）．次にデータを加工しやすい形に成形します（**リスト6**，**リスト7**）．最終的に引数dfに成形したデータを渡します（**リスト8**）．この結果，**表1**が出力されます．

地図の描画

● ①ベースとなる地図の作成

成形したデータ（df）を地図上に示します．このコードを**リスト9**に示します．

まず，ベースとなる地図を読み込みます．ここでは，foliumというパッケージを使って，leaflet.jsというインタラクティブな地図の描画を行います．なお，デフォルトで背景地図には，“OpenStreetMap”“Stamen Terrain”，“Stamen Toner”，“Stamen Watercolor”“CartoDB positron”，“CartoDB dark_matter”が使えます．最終的にmapという引数に背景地図の情報を渡しておきます．**リスト9**を実行すると**図2**が表示されます．

● ②都道府県を塗り分けるためには都道府県境界のあるベクタ・データが必要

背景地図が表示できたら，次に必要なのは，人口データを表示させるための枠になる都道府県ごとに境界が示されたベクタ・データです．今回はgeojsonという形式のファイルで読み込んでいきます．今回はあらかじめ，

```
https://github.com/wata909/interface2022/raw/main/GIS_DATA/japan.geojson
```

の場所に都道府県境界のあるベクタ・データのgeojsonファイルを置いておきました．それをcolaboratory環境に読み込みます．このコードを**リスト10**に示します．もし，皆さんが各自で似たような地図を作りたい場合は，コンピュータ内やクラウド上にgeojsonファイルを置き，それを読み込むことで同じように地図データを取り込むことが可能です．このコードを**リスト10**に示します．

無事，geojsonファイルが読み込めて，都道府県ごとの位置情報が付加されているリスト（**表2**）が表示されれば，成功です．

図2　まずはベースとなる白地図を作成した

● ③人口規模で都道府県を色分け

いよいよ人口規模に従って，都道府県を色分けする作業に取り掛かれます．folium.Choroplethの関

表2　都道府県ごとの位置情報が付加されたデータ

	coc	nam	CODE	pref_code	geometry
0	JPN	AICHI	23	23	MULTIPOLYGON Z (((136.96091 35.41453 0.00000, …
1	JPN	AKITA	5	5	MULTIPOLYGON Z (((140.87970 40.51147 0.00000, …
2	JPN	AOMORI	2	2	MULTIPOLYGON Z (((140.91240 41.55253 0.00000, …
3	JPN	CHIBA	12	12	MULTIPOLYGON Z (((139.77620 36.09087 0.00000, …
4	JPN	EHIME	38	38	MULTIPOLYGON Z (((132.99680 34.28980 0.00000, …

（以下省略）

コラム　階級の決め方

小野原 彩香

● 統計データの階級分けで地図の印象が変わる

地図を自分で作れるようになると，気になることがいくつか出てきます．今回のようなコロプレスマップの場合，階級の色分けの仕方によって，出来上がる地図の印象がかなり違うことに気づきます．また，表示させたい統計データの大きさの幅やばらつき具合に偏りがあると，どうやって色分けしたらよいかに悩みます．つまり，データ自体は客観的でも，それを分かりやすく地図という図版にする段階で，何らかの恣意性が含まれることにも気づくでしょう．階級の決め方にも絶対的ではありませんが，幾つか決まりがあるので**表A**に示します．これ以外にも地図の用途によって，さまざまな階級の分け方が存在します．

地図をつくる上で重要なことは，自分がその地図で表現したいことは何かを明確にすることと，色分けの凡例をつけるのを忘れないようにすることです．

● いろいろな地図を簡単に作れるようになった

紙の地図しか無い頃には，筆者はしょっちゅう道に迷っていました．自分の位置と周りの建物の位置を，紙の地図上にイメージする必要がありましたし，しかも，地図の一部が古いと，目の前の現実と一致しません．地図を使いこなすのは随分高度な技術が必要だなと思っていました．

今や，自分の場所はスマホが教えてくれるのは当たり前になりました．自分が回ると地図アプリの地図もぐるぐる回ります．地図を作ることも，今では随分簡単にできるようになりました．今回，使用したAPIを使えば，かなりのバリエーションの統計地図を作ることができます．

表A　階級の決め方の例

階級の決め方	内　容
等間隔	各階級のサイズが同じになるように分ける
等量分類	各階級の内部にあるデータの数が同じになるように分ける
自然分類（Jenks）	各階級内の分散は最小に，階級間の分散は最大になるように分ける
基準値からの偏差	ある値（平均値など基準値）からどれだけ離れているか（偏差）によって階級を決める
キリのいい数字	100，200，300のようにキリの良い数字を階級値とする

リスト11　人口規模で都道府県を色分け

```
# 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)
```

数を使い，パラメータを指定していくことで，思った地図を描くことができます．

コロプレスマップは先ほど作った背景地図“map”に色分けの情報を追加することで作成できます．

また，表示に使用しているleafletはインタラクティブな地図表示なので，左上の＋－ボタンで地図の拡大／縮小が可能ですし，地図上でドラッグ＆ドロップ操作をすることで，地図表示の位置を自由に変えることが可能です．このコードを**リスト11**に示し，実行した結果得られる地図を**図1**に示します．

＊

APIの統計データを使って，それを地図上に読み込み，表示させる手順と，その表示にはコロプレスマップを使用する方法を解説しました．次回は，複数の統計データを同時に地図上に表示する方法について学んでいきます．

◆参考・引用＊文献◆

(1) 政府統計の総合窓口（Stat）API仕様［バージョン令和元年独立行政法人7月3.0］．
https://www.e-stat.go.jp/api/sites/default/files/uploads/2019/07/API-specVer3.0.pdf
（閲覧日：2022年11月14日）

おのはら・あやか，いわさき・のぶすけ