Python による効率的な地理的可視化とその幅広い利用に向けた研究

弘前大学人文社会科学部社会経営課程　地域行動コース
中村　優翔
(主査　増山　篤　教授、副査　曽我　亨　教授)

1. はじめに

本研究は、幅広く多くの人が Python による地理的可視化を実行できるようにすることを目的とするものである。

近年ではプログラミングによるデータ処理が再評価されている。かつて、コンピュータを扱うには複雑なコマンドを打つ必要があり、高度な知識やスキルを要した。時代が進み、直感的な操作でコンピュータを動かせるようになり、多くの人にとってプログラミングの必要性はなくなった。しかし、依然としてプログラミングには二つの大きな利点がある。一つは、データ処理の過程と結果に対して透明性と再現性を保証できることである。もう一つは、人の手では捌ききれないほどの大量のデータも扱うことができるということである。

プログラミングによるデータ処理においては、さまざまな言語を用いうる。それら言語の中でも、近年特に脚光を浴びているのは Python である。Python はシンプルな構文と直感的な記述方法により扱いやすく、プログラミング初学者にも適している。その汎用性の高さから、有志によって多数の機能拡張ライブラリが開発されており、それらの中には地理空間データを扱うものもある。

地理空間データを扱う上で「可視化」は特に重要である。広くデータの「可視化」は特に有意義な知見や洞察をデータ利用者に対してもたらしてきた。直近 400 年間に起きたデータ視覚化に関する技術革新の多数はたびたび科学と社会の重要な問題解決などに貢献してきた(フレンドリーら,2021)。例えば、スノウ(1855)が作成したコレラの感染者および死亡者の居住地を視覚化した地図は、コレラの感染と井戸ポンプとの関係を示唆することで当時の公衆衛生への新たな知見の発見につながった。

このような「可視化」は主に人の手で行われてきたが、技術の発展に伴い、コンピュータが使用されるようになった。その方法として特にプログラミングの分野においては統計データ解析に強い R を用いたものが一般的であるが、近年特に脚光を浴びている Python においても手引き書(マクレイン,2023)が発売されるなど地理空間データの可視化需要は増大している。本研究は一般的に使用される地理的可視化および初学者に向けた簡便なコード記述を可能とすることを目的とする。

2. 研究方法

本研究の流れは次の通りである。まず、地理的可視化を実行する上で必要な環境を誰も使用できるように Docker を用いた構築の手順を示す。次に、ケーススタディとして作成した環境にて既存のライブラリを用いた交通事故統計データの可視化を行い、実行コード行数および実行時間を記録する。その次に、コーディングにおける技術的・時間的コストの削減を行うために既存のライブラリをもとに新たに関数を作成し再度検証を行う。最後に、全体のまとめとして成果を述べる。

3. 環境構築

Python を含むいずれのプログラミング言語においても、プログラム開発を行う際には、0S やソフトウェア、ライブラリの競合等に起因する予期せぬエラーが生じないように配慮した上で、そのための環境をコンピュータ上に構築する必要がある。そのため、本研究では Docker を用いた環境構築の手順を示す。

Docker の利用には Docker file、image、container、Docker Hub の関係を知る必要があるが、これらを図示したものが図 1 である。

本研究では Docker Hub 上に公開されている既存の miniconda 環境の image をもとに地理的可視化に必要なライブラリをインストールした image を作成し、実行環境である container を起動する手順を示す。具体的には、まず、image の作成に必要な Docker file の書き方を示す。ここでは、既存の image を Docker Hub から引き出し、新たにライブラリのインストール、フォントのダウンロードおよび適用の方法を示す。次に、作成した Docker file に基づいて image を作成する手順を示す。その次に、container を起動する際に用いる docker-compose.yml の書き方を示す。最後に、作成した docker-compose.yml を用いて実行環境である container を起動する手順を示す。

インストールしたライブラリをまとめたものが表 1 である。本研究では、可視化にあたって Plotnine を用いる。これは R の ggplot2 に即した関数設定がされているため、R ユーザーにとっても理解しやすいコーディングを行えることを狙いとして使用する。また、コーディングおよび実行には JupyterLab を用いる。

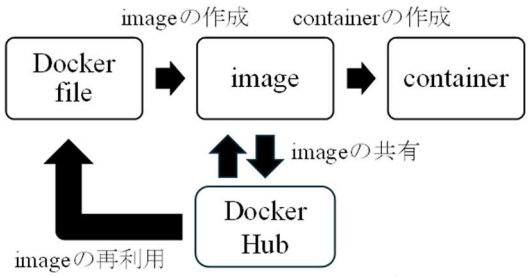

図 1 Docker の関係

表 1 インストールしたライブラリ一覧

ライブラリ名	主な用途
Pandas	表データの操作
GeoPandas	地理空間データの操作
Shapely	地理空間データの作成
NumPy	数値計算
SciPy	統計処理
Matplotlib	可視化
plotnine	可視化
JupyterLab	コード記述及び実行
JupyterLab-Language-Pack-ja-JP	jupyterlabの日本語化

4. ケーススタディ

ケーススタディでは前章で構築した Python プログラミング環境およびインストールした既存のライブラリ及び作成した関数を用いて日本国内の全市区町村を対象として可視化を試みる。本研究では、コロプレス地図、ドット分布マップ、比例シンボルマップ、ヒートマップ、アニメーションマップを作成した。

4-1. 使用するデータと可視化内容

ケースタディに使用するデータは次の通りである。国勢調査小地域ポリゴンデータ(以降、小地域データ)について「政府統計の総合窓口(e-stat)」(https://www.e-stat.go.jp/)から令和 2 年国勢調査基本単位区境界データ(世界測地系・平面直角座標系)について掲載中の全データ(1896 データ)をシェープファイルで入手した。また、交通事故統計データ(以降、事故データ)について警察庁ホームページ内の交通事故統計情報のオープンデータから 2019 年から 2023 年のデータ(5 データ)を CSV ファイルで入手した。

また、これらのデータを用いて表 2 に示す可視化を行った。なお、プログラムの動作環境を表 3 に示す。

4-2. 既存のライブラリを用いた検証

本節では既存のライブラリを使用して可視化のコードを作成、実行する手順を示す。可視化のプログラムを作成するにあたり流れを示したものが図 2 である。1 つのファイルを読み込み、下準備をした後にデータの欠損があるなど可視化に値しない場合はエラー処理としてテキストファイルを書き出す。また、正常に処理できた場合は可視化を行う。この動作をすべてのファイルに対して実行することで全市区町村における可視化を行う。

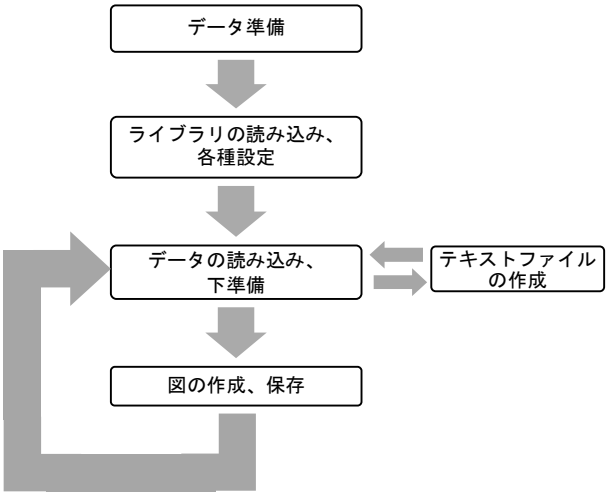

図 2 可視化の流れ

表 2 使用するデータと可視化内容

	コロプレス地図	ドット分布マップ	比例シンボルマップ	ヒートマップ	アニメーションマップ
使用データ	国勢調査小地域	国勢調査小地域 交通事故統計	国勢調査小地域 交通事故統計	国勢調査小地域 交通事故統計	国勢調査小地域 交通事故統計
可視化内容	人口密度	交通事故発生地点	人口密度 交通事故発生件数	交通事故発生の密集度	交通事故発生の密集度(年別)

表 3 プログラムの動作環境

OS	Windows11Home24H2
CPU	Intel(R)Core(TM)i5-14400F
メモリ	32GB
SSD	1TB
グラフィックボード	NVIDIAGeForceRTX4060 8GB

今回の流れでデータを処理する際には二つの注意点がある。一つは、扱うデータの問題であるが、事故データの緯度経度は度分秒表記である。このまま用いた場合、正しく座標を抽出できないため。度表記に変換する必要がある。

もう一つは、一度の処理を終了した後、作成した地図を格納した変数を削除し、メモリ解放する必要がある。これを行わない場合、使用メモリ量が増大するため、途中で処理が止まってしまい、エラーの原因となる。

以上に注意して可視化を行った。図 3 はコードを用いて作成した青森県弘前市における交通事故発生の密度を示したヒートマップである。本研究では、1 つのファイルごとに処理を行っているため、地域によっては縮尺に差が生じるが、どの市区町村についても図 3 と同様の可視化を行うことができた。

4-3. 新たに関数の作成による再検証

前節では既存のライブラリを用いて可視化を行うコードを作成したが、これらライブラリをそのまま使うと、長く複雑なコーディングが必要である。幅広い利用を見据えた場合、この課題を少しでも軽減する必要がある。

そのため、本節では Python 初学者でも簡便な可視化を実行できるように既存のライブラリから新たに関数を作成、使用して可視化を行う手順を示す。本研究では引数に応じて汎用的な可視化を行えるように関数を作成した。作成した関数の一覧が表 4 である。これらを用いて再度可視化を行う。

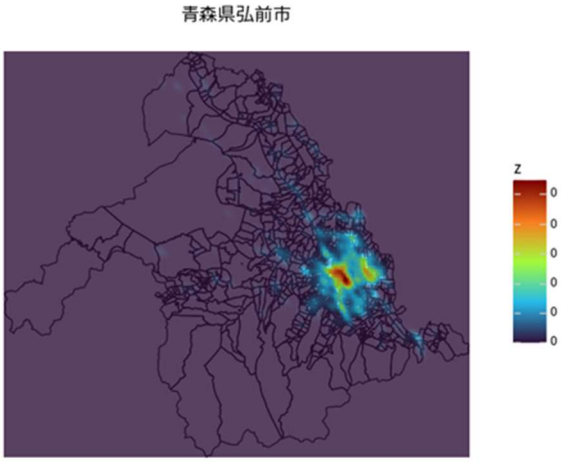

図 3 青森県弘前市における交通事故発生密度の可視化
出典:令和 2 年国勢調査基本単位区境界データ(政府統計の総合窓口(e-stat))及び交通事故統計情報のオープンデータ(警察庁)を加工し筆者作成

表 4 作成した関数の一覧

	関数	用途
データの下準備の利便化	make_file_list	指定したフォルダ内に存在する特定の拡張子を持つファイルのパスリストを作成
	bind_df	リスト内のパスからファイルを読み込み、結合
	make_list	データ内の要素をリストとして取得
	get_quantiles	指定した分位数を計算
	dms_to_dd	度分秒表記の緯度経度を度表記に変換
	add_xy	データにポイント座標を追加
	count_point	ポリゴン内に存在するポイントを数えて新たに列を追加
可視化の利便化	set_theme	可視化のレイアウトを調整
	make_choro_core	コロプレス地図を作成する際の中心的処理
	make_choro	コロプレス地図を作成
	make_dot_core	ドット分布マップを作成する際の中心的処理
	make_dot_dist	ドット分布マップを作成
	make_prop	比例シンボルマップを作成
	make_kde	カーネル密度推定によるヒートマップを作成
	make_ani	アニメーションを作成

表 5 実行コード行数と実行時間の比較

		コロプレス地図	点分布マップ	比例シンボルマップ	ヒートマップ	アニメーションマップ
既存のライブラリを用いた場合のコード	コード行(行)	31(23)	62(53)	83(74)	84(73)	108(94)
	実行時間(分)	9.9	11.7	13.4	114.2	509.6
新たに関数を作成した場合のコード	コード行(行)	8	21	23	15	25
	実行時間(分)	11.2	10.5	12.8	78.0	327.4
比較	コード行の削減率(%)	65.2%	60.4%	68.9%	79.5%	73.4%
	実行時間の削減率(%)	-13.1%	10.3%	4.5%	31.7%	35.8%

注 1)()内の数字はライブラリのインポートを除いた行数である。
注 2)新たに関数を作成した場合のコード行数は関数の定義づけに必要なコード行数を除いている。

4-4. 比較

本節では、実行コード行数と実行時間を比較することでプログラミングによる地理的可視化の利便性と関数化による利便性の向上がみられたことを示す。二つの結果を比較した表が表 5 である。いずれの可視化においても関数化の有無に限らず比較的短時間で可視化を実行していることがわかる。また、適切に関数化を行ったことでコード行数が 6〜8 割程度削減されている。また、実行時間についても効率的に処理が行えるようになったことで 3 割程度短縮される場合もあることがわかった。

5. まとめ

本研究における成果は四つである。まず、Docker によって、オペレーティングシステムに依存せず、Python による地理的可視化を実行するための環境構築手順を具体的に示した。これまで Python に限らずプログラミングによる処理を行う際には Macintosh や Windows といった OS の違いによって動作に影響が生じていた。しかしながら、本研究では Docker を用いることで誰もが同様の実行環境を得られるようになった。

次に、Python による地理的可視化を実行する上でどのようなライブラリを利用し、また、それらライブラリを用いてコーディングする際の注意点などを明らかにした。本研究では、これまで Python による可視化のライブラリとして主流ではなかった Plotnine を用いて可視化を行ったが、R 言語の主な可視化ライブラリである ggplot2 と似たメソッドを用いることから R に慣れ親しんだユーザーにもコーディングが容易であることを示した。また、ケーススタディでは繰り返し処理を用いた。実行中に作成した地図を格納した変数に対してメモリ解放を随時指示しなければ、メモリ使用量が増大し、可視化の失敗につながることが明らかになった。

その次に、さらにコーディングの負担を削減して、効率的にさまざまな種類の地理的可視化が実行できる関数を作成した。特に可視化を行う関数の作成では考えうる利用者の関心に沿うような引数を設けることで、汎用的かつ簡便な関数を示した。

最後に、ケーススタディを通じ、比較的軽いコーディング作業により、十分に短い実行時間で数多く地図作成が実行できることを示した。作成した関数によって、コード行数が 6〜8 割程度軽減された。また、ヒートマップの作成など計算量が多い可視化は実行時間が 3 割程度軽減された。

本研究で作成した Docker 環境およびソースコード並びに Python コードは Docker Hub(https://hub.docker.com/r/nakamura734/geobeginner) および GitHub(https://github.com/nakamura734/geobeginner) にて公開した。URL にアクセスしてダウンロード等することで誰でも同様の過程および結果を得られる。本研究の成果がさらなる学術の発展に寄与し、実社会における地理的可視化のニーズにも応えることを期待し、本稿を締めくくることにしたい。

参考文献

小口 高(2020),「地図と GIS の歴史的発展」,農村計画学会誌,38 巻,4 号,p.436-439

ブランズドン, C.・コーマー, L.(湯谷啓明・工藤和奏・市川太祐 訳)(2018),『R による地理空間データ解析入門』,共立出版,2018.
An Introduction to R for Spatial Analysis and Mapping.

フレンドリー,M.・ウェイナー,H.(飯島貴子 訳)(2021),『データ視覚化の人類史　グラフの発明から時間と空間の可視化まで』,青土社,2021.A History of Data Visualization and Graphic Communication.

マクレイン, B. P.,(2023),『Python による地理空間データ分析 : 例題で学ぶロケーションインテリジェンス』.広川類訳.オライリー・ジャパン.

ラブレス, R.・ノウォサド, J.・ミンチェフ, J.(2019), Geocomputation with R. CRC Press.

増山 篤(2019),「空間アクセシビリティ分析における R の活用可能性」,GIS-理論と応用,29 巻,2 号,p.101-108

読売新聞社(2024),「令和 6 年能登半島地震被災状況マップ(平面版) 各地の被災状況 俯瞰マップ」,https://www.yomiuri.co.jp/topics/noto-earthquake-situation-map-planar, (最終閲覧日:2024/09/13)

Dupin,C.(1826),Carte figurative de l’instruction populaire de la France. Brussels. https://gallica.bnf.fr/ark:/12148/btv1b530830640?lang=FR,(最終閲覧日:2024/05/23)

Snow, J.(1855). On the mode of communication of cholera. London. https://archive.org/details/b28985266/page/n57/mode/1up,(最終閲覧日:2024/05/27)