Web Mapping and Geovisualisation

Gabriele Filomena and Elisabetta Pietrostefani

2024-04-22

Table of contents

Welcome

This is the website for "Web Mapping and Geovisualisation" (module **ENVS456**) at the University of Liverpool. This course is designed and delivered by Dr. Gabriele Filomena and Dr. Elisabetta Pietrostefani from the Geographic Data Science Lab at the University of Liverpool, United Kingdom. The module has two main aims. It seeks to provide hands-on experience and training in:

- The design and generation of web-based mapping and geographical information tools.
- The use of software to access, analyse and visualize web-based geographical information.

 A compilation of this web course is hosted as a GitHub repository that you can access:

- As an html website.
- As a GitHub repository.

Contact

Gabriele Filomena - gfilo [at] liverpool.ac.uk Lecturer in Geographic Data Science Office 1xx, Roxby Building, University of Liverpool - 74 Bedford St S, Liverpool, L69 7ZT, United Kingdom.

Elisabetta Pietrostefani - e.pietrostefani [at] liverpool.ac.uk Lecturer in Geographic Data Science Office 6xx, Roxby Building, University of Liverpool - 74 Bedford St S, Liverpool, L69 7ZT, United Kingdom.

Syllabus

Week 1

- Lecture: Introduction to the module
- Lab: Powerful examples and Python Refresher

Week 2

- Lecture: Geovisualisation: Design Principles and Statistical Visualisation
- Lab: Static Maps

Week 3

- Lecture: The Web's Architecture
- Lab: Web Architectures and APIs

Week 4

- Lecture: Data architechtures & formats
- Lab: Data architechtures & Tiles

Week 5

- Lecture: Interactive Maps
- Lab: Interactive Maps

Week 6

- Lecture & Lab: Q&A & Assignment I Clinic
- **Assignment I Due**

Week 7

- Lecture: OpenStreetMap Data & Spatial Network
- Lab: Retrieving Data From OpenStreetMap

Week 8

- Lecture: Dashboards
- Lab: Designing Dashboard for Geovisualisation

Week 9

- Lecture: Advanced Tools
- Lab

Week 10

- Lecture & Lab: Review & Assignment II clinic
- **Assignment II Due (Week 11)**

Overview

Aims

This module aims to provide hands-on experience and training in: - The design and generation of (good looking) web-based mapping and geographical information tools. - The use of software to access, analyse and visualize web-based geographical information.

Learning Outcomes

By the end of the module, students should be able to:

(2) Visualise and represent geo-data through static and dynamic maps.
(3) Recognise and describe the component of web based mapping infrastructure.
(4) Collect Web-based data.
(5) Generate interactive maps and dashboards.
(6) Understand basic concepts of spatial network analysis.
(7) Manipulate geo-data through scripting in Python.

Feedback

Formal assessment. Two pieces of coursework (50%/50%). Equivalent to 2,500 words each

Verbal face-to-face feedback. Immediate face-to-face feedback will be provided during computer, discussion and clinic sessions in interaction with staff. This will take place in all live sessions during the semester. *Teams Forum*. Asynchronous written feedback will be provided via Teams. Students are encouraged to contribute by asking and answering questions relating to the module content. Staff will monitor the forum Monday to Friday 9am-5pm, but it will be open to students to make contributions at all times. Response time will vary depending on the complexity of the question and staff availability.

Assessments

Assignment I

- **Title:** Exploring APIs and Interactive Maps in Python.
- **Type:** Coursework.
- **Due: Thursday March 7th, Week 6**.
- 50% of the final mark.
- Electronic submission only.

In this assessment, you will have the opportunity to explore different sources and combine them in a map that can be explored interactively through a web browser. **The assignment aims to evaluate your knowledge and aptitude in the following areas**:

- Understanding of core "backend" concepts in web mapping such as tilesets, client-server architecture, and APIs.
- Ability to use the web as a resource for original data.
- Design skills to represent effectively a diverse set of geospatial data in a web map.

Design, data pan assemblage.

This assignment requires you to source data from the web in different formats, assemble them, and document the process. To be successful, you will need to demonstrate your understanding not only of the technical aspects involved in the process, but also of the conceptual notions that underpin them. Below are described the required components for your submission.

1. **Data**. Draft a list of potential spatial datasets that you can access by developing a simple API in Python. The data should contain spatial information or should be easy to link to other existing spatial data sources (e.g. combining different datasets). While you can make use of available datasets, **you are required to demonstrate the ability to develop your own API request**. Once you know which datasets you are interested and able to obtain, think about what data/variables deserve consideration and how they could be presented.

2. **Design**. Start by designing a map that represents the spatial units of the dataset (e.g. buildings, train stations, cities, etc.). Use `folium` to incorporate initial basic interactivity. At this point, check on your ideas with the module teachers, who will be able to assess whether potential problems may arise from your choices. Move to the incorporation of data, both for categorical (if appropriate) and numerical variables, and consider how to represent it through the map. This stage should draw some inspiration from the first weeks of the course, where we looked at examples of web maps and spent time discussing what made them good and why.
3. **Assemblage**. Embellish your map, consider the usage of widgets for facilitating the exploration of the dataset. Play and choose one of the available tilesets. Consider creating your own in `Mapbox`. Pay attention to the design aspects involved in this step too. For example, what is the extent of your map (not necessarily the extent of each of your data)? What are the zoom levels your map will allow? Do you have the same "map" for every zoom level? These are questions you will have to ask (and answer!) yourself to complete this stage successfully.

Once you have developed your API and created the interactive map, you will need to present it. An important aspect of this stage is that it is not really the code/map you need to present, but the *process* of creation you have followed and the *design* choices you have made that should go into the text. Additionally, you will need to provide evidence that you understand the concepts behind some of the technologies you have used.

In a Nutshell, Assignment 1 should contain:

Code

- Introductory Static Maps (2 to 3), presenting the topic and the geographic context.
- An API request written by your own.
- All the necessary steps for making your API work and for data cleaning/exploration.
- An interactive final map (and intermediate ones, if necessary, to present the final development). This should be fed with data obtained through the API request.

Up to **1,000 words** - distributed across the notebook, and included as Markdown Cells - referring to:

Map brief:

```
-   About 250 words introducing the map(s). This should cover what it tries to represent (wha
-   About 250 words discussing and motivating the sources of data you have used. Here you sho
```

Conceptual background:

```
-   About 250 words with your description of what your API is, how it works and how it has ma
-   About 250 words with your description of how your final interactive map works, its compon
```

Submission

You will submit through Canvas a `.html` file obtained from a Python `.ipynb` Jupyter Notebook file. To do so, in your `.ipynb` file, follow these steps: `File` –> `Save and Export as..` –> `HTML`. Prior to this step, the notebook needs to be rendered (i.e. all the cells should be executed). Other file formats will not be accepted.

Evaulation

The assignment will be evaluated based on four main pillars, on which you will have to be successful to achieve a good mark:

1. **Map design abilities.** This includes ideas that were discussed in week 2, 3 and 4.
2. **Technical skills**. This includes your ability to master technologies that allow you to create a compelling map, but also to access interesting and sophisticated data sources.
3. **Overall narrative**. This assesses your aptitude to introduce, motivate and justify your map, as well as you ability to bring each component of the assignment into a coherent whole that "fits together".
4. **Conceptual understanding of key technologies** presented in the course, in particular as regards the usage of APIs.

How is this assignment useful?

This assessment includes several elements that will help you improve critical aspects of your web mapping skills:

- **Design**: this is not about making maps, this is about making good maps. And behind every good map there is a set of conscious choices that you will have to think throug to be successful (what map? what data? how to present the data? etc.).
- **Technology**: at the end of the day, building good web maps requires solid understanding of current technology that goes beyond what the average person can be expected to know. In this assignment, you will need to demonstrate you are proficient in a series of tasks manipulating geospatial data in a web environment.
- **Presentation**: in many real-world contexts, your work is as good as it can come across to the audience it is intended to. This means that it is vital to be able to communicate not only what you are doing but why and on what building blocks it is based on.

Assignment II

- **Title:** *A dashboard that explores a Spatial Dataset.*

- **Type:** Coursework.
- **Due: Thursday May 2nd, Week 11, 2.00 pm**.
- 50% of the final mark.
- Electronic submission only in HTML.

This assignment requires you to build a dashboard for a **spatial data set of your choice**. To be successful, you will need to demonstrate your understanding not only of technical elements, but of the design process required to create a product that can communicate complex ideas effectively. There are three core building blocks you will have to assemble to build your dashboard: the main maps(s), base map, and widgets.

1. **The main map(s)**. Import your data and start building a dashboard with `panel`. Think about what you want to show, how, which interactive elements you will allow the user to access and how they will let them modify the experience of your dashboard. It goes without saying that the dashboard should include interactive map(s), besides allowing the user to play with the dataset. Interactive maps should be built with `folium` or (optionally) with `pydeck`. **Holoviews, Geoviews, or other tools can be used as well, but only for showing alternative or secondary maps.**
2. **The basemap**. Design your own basemap through scripting (e.g. assembling a basemap with OpenStreetMap features in a unique layer) or use available TileSets. Think about the data in the background, which colors, the zoom levels that will be allowed, and how it all comes together to create a backdrop for your main message that is conducent to the experience you want to create. Use the basemap to enhance the visualisation experience of the user.
3. **Additional widgets**. One of the advantages of dashboards in comparison to standard web maps is that they allow to bring elements of analysis to a more finished product. Think about what you want your users to be able to analyse, why, and how that will modify the main map.

DOs and DONTs

- Do not include "temporary" maps unless you really need to specifically show something to your reader.

- Do not include maps that have no actual differences, apart from few things (e.g. you changed the zoom level)
- Mix the accompanying text, in markdown cells, with the code.
- Do not include all the text at the beginning, and that's it.
- Provide some theoretical context and motivation to your topic.
- Present 2 or 3 NICE maps + the final one included in your dashboard.

Submission

Important, before exporting your `.ipynb` to a `.html` file: follow the steps described here to include an interactive dashboard in the static html that will be sumbitted as your assignment. This is to guarantee that your dashboard works in the submission file.

You will submit through Canvas a `.html` file obtained from a Python `.ipynb` Jupyter Notebook file. To do so, in your `.ipynb` file, follow these steps: `File` –> `Save and Export as..` –> `HTML`. Prior to this step, the notebook needs to be rendered (i.e. all the cells should be executed).

Other file formats will not be accepted.

Besides the necessary code, the report should include up to **1,000** words and the following:

- About 250 words for the overall idea of the dashboard. What do you want to communicate? What is the story you want to tell?
- About 250 words for the data used. Which datasets are you using? Why? What new information do they bring and how they complement each other?
- About 250 words to describe your design choices in the layers presented (e.g. choropleths).
- About 250 words to describe your design choices around interactivity, including both cartographic elements (e.g. zooming, panning) as well as additional interactivity built around components such as widgets.

Evaluation

The assignment will be evaluated based on four main pillars, on which you will have to be successful to achieve a good mark:

1. *Overall design of the experience.* It is very important you think through every step of preparing this assignment as if it was part of something bigger towards which it contributes. Because that is exactly what it is. Everything should have a reason to be there, and every aspect of the dashboard should be connected to each other following a common thread. And, of course, make this connection and holistic approach come alive in your report.
2. *Map design.* Critically introduce every aspect you have thought about when designing the maps, and explicitly connect it to the overall aim of the dashboard. Be clear in your descriptions and critical in how you engage every design choice.
3. *Interactivity design.* Your dashboard should use interactivity when necessary to deliver a more compelling and fuller experience that better gets your message across. Be sure to clearly lay out in your report which elements are used and why.

4. *Narrative around the description of the process.* Finally, the final mark will also take into account not only how good your dashboard is, but how well you are able to introduce it. Start with the key goals, and then unpack every element in an integrated and compelling way.

How is this assignment useful?

This assignment combines several elements that will help you improve critical aspects of web mapping:

- *Design*: this is not about making maps, this is about making good maps. And behind every good map there is a set of conscious choices that you will have to think through to be successful (what map? what data? how to present the data? etc.).
- *Technology*: at the end of the day, building good web maps requires familiarity with the state-of-the-art in terms of web mapping tools. In this assignment, you will need to demonstrate your mastery of some of the key tools that are leading both industry and academia.
- *Presentation*: in many real-world contexts, your work is as good as it can come across to the audience it is intended to. This means that it is vital to be able to communicate not only what you are doing but why and on what building blocks it is based on.

Marking Criteria

This course follows the standard marking criteria (the general ones and those relating to GIS assignments in particular) set by the School of Environmental Sciences. Please make sure to check the student handbook and familiarise with them. In addition to these generic criteria, the following specific criteria will be used in cases where computer code is part of the work being assessed:

- 0-15: the code does not run and there is no documentation to follow it.
- 16-39: the code does not run, or runs but it does not produce the expected outcome. There is some documentation explaining its logic.
- 40-49: the code runs and produces the expected output. There is some documentation explaining its logic.
- 50-59: the code runs and produces the expected output. There is extensive documentation explaining its logic.
- 60-69: the code runs and produces the expected output. There is extensive documentation, properly formatted, explaining its logic.
- 70-79: all as above, plus the code design includes clear evidence of skills presented in advanced sections of the course (e.g. custom methods, list comprehensions, etc.).

- 80-100: all as above, plus the code contains novel contributions that extend/improve the functionality the student was provided with (e.g. algorithm optimizations, novel methods to perform the task, etc.).

Setting up the Working Environment

Follow the instructions for yuour Operating System and test your installation. If you experience any issues, write a message on the Ms Teams channel of the module. Setting up the Python environment is necessary for:

- Executing the Jupyter Notebooks of the Lab sessions of the course.
- Preparing your own Jupyter Notebooks for the assignments (one each).

We will use `Minconda` to handle our working environment. *Miniconda is a free minimal installer for conda. It is a small bootstrap version of Anaconda that includes only conda, Python, the packages they both depend on, and a small number of other useful packages (like pip, zlib, and a few others)*

Set up Miniconda (and Python) on Ms Windows

Installation

1. Install Miniconda:
    - *Option 1*: On a UoL Machine: Download and install Miniconda from here. This will install Miniconda and Python in `C:\`. If this process is aborted because it requires administrator rights, press `Start`, select `Install University Applications`, type and choose `Miniconda`.
    - *Option 2*, **Recommended**: Install Miniconda on your personal Laptop: Follow the instructions here.
2. During the installation, leave the default settings. In particular, when asked whom to "Install Miniconda for", choose "Just for me".

Important: If you do choose to work on University Machines you will have to reinstall `Miniconda` every lab session unless you use a PC where Miniconda has been installed already.

Alternatively, you can work on the lab notebooks directly on the web. This does not require to install `Miniconda`. However, it represents a much slower option, especially when setting up the environment. To do so, you can access the data and the lab notebooks in the `\labs` directory from a virtual copy of the course repository here. If you opt for this option, you do not need to follow the rest of the instructions below.

Set up the Directories

1. Create a folder where you want to keep your work conducted throughout this course. For example, call it `envs456`. You can save it wherever you want. If you are working on a university machine, it could be worth creating it in `M:/`, which should your "virtual" hard-disk.
2. Download the data and the images for running and rendering the jupyter notebooks.
3. Unzip the folders and move the nested folders into the folder `envs456`.
4. Create another folder called `labs`

The folder structure should look like:

```
envs456/
  data/
  labs_img/
  labs/
```

Set up the Python Environment

1. Download the `envs456.yml` from GitHub by cliciking `Download raw file`, top right at this page
2. Save it in the folder `envs456` created before.
3. Type in the search bar and find the `Anaconda Prompt (miniconda 3)`. Launch it. The terminal should appear.

3. In the **Anaconda Terminal** write: `conda env create -n envs456 --file C:\envs456\envs456.yml` and press `Enter`; if the file is located elsewhere you'll need to use the corresponding file path.
4. If you are prompted any questions, press `y`. This process will install all the packages necessary to carry out the lab sessions.
5. In the **Anaconda Terminal** write `conda activate envs456` and press `Enter`. This activates your working environment.
6. *Necessary* on University machines, otherwise *Optional*: Configuration of Jupyter Notebooks

    - In the **Anaconda Terminal**, write `jupyter server --generate-config` and press enter. This, at least in Windows, should create a file to: `C:\Users\username\.jupyter\jupyte`
    - Open the file with a text editor (e.g. Notepad++), do a `ctrl-f` search for: `c.ServerApp.root_dir`, uncomment it by removing the and change it to `c.ServerApp.notebook_dir = 'C:\\your\\new\\path`, for example the directory where you created the `envs456` folder. In the University Machines, it is advised to work on the directory `M:\`.
    - Save the file and close it.

Start a Lab Session

1. Download the Jupyter Notebook of the session in your folder. Choose one jupyter notebook and click `Dowload raw file` as shown below

2. Save the file in the `labs` folder within your `envs456` folder on your machine.
3. Type in the search bar, find and open the `Anaconda Prompt (miniconda 3)`.
4. In the **Anaconda Terminal** write and run `conda activate envs456`.
5. In the **Anaconda Terminal** write and run `jupyter notebook`. This should open Jupyter Notebook in your default browser.

6. Navigate to your course folder in and double click on the notebook downloaded in step 1.
7. You can now work on your copy of the notebook.

Follow these instructions and test your installation **prior to the first Lab Session** (Wed, 31st of January). If you experience any issues, write a message on the Ms Teams channel of the module. Setting up the Python environment is necessary for:

- Executing the Jupyter Notebooks of the Lab sessions of the course.
- Preparing your own Jupyter Notebooks for the assignments (one each).

Set up Miniconda (and Python) on MAC

Installation

To install Miniconda on your personal laptop, Follow the instructions here. During the installation, leave the default settings. In particular, when asked whom to "Install Miniconda for", choose "Just for me".

Set up the Directories

1. Create a folder where you want to keep your work conducted throughout this course. For example, call it `envs456`. You can save it wherever you want. For example, Elisabetta has named her folder `envs456` and it's in her Dropbox in `Users/PIETROST/Library/CloudStorage/Dropbox/envs456`
2. Download the data and the images for running and rendering the jupyter notebooks.
3. Unzip the folders and move the nested folders into the folder `envs456`.
4. Create another folder called `labs`

The folder structure should look like:

```
envs456/
  data/
  labs_img/
  labs/
```

Set up the Python Environment

1. Download the `envs456.yml` from GitHub by cliciking `Download raw file`, top right, at this page
2. Save it in the folder `envs456` created before.
3. Type in the search bar and open the **Terminal**.
4. In the **Terminal** write `conda env create -n envs456 --file envs456.yml` and press `Enter`. This will need to be modified according to where you placed the `envs456` folder. For example, Elisabetta has named her folder `envs456` and it's in her Dropbox in `Users/PIETROST/Library/CloudStorage/Dropbox/envs456/envs456.yml`. If you created the `envs456` folder on your desktop, the path would be `Desktop/envs456`.

4. If you are prompted any questions, press `y`. This process will install all the packages necessary to carry out the lab sessions.

Start a Lab Session

1. Download the Jupyter Notebook of the session in your folder. Choose one jupyter notebook and click `Dowload raw file` as shown below

2. Save the file in the `labs` folder within your `envs456` folder on your machine.
3. Type in the search bar, find and open the **Terminal**.
4. In the **Terminal** write and run `conda activate envs456`.
5. In the **Terminal** write and run `jupyter notebook`.

6. This should open Jupyter Notebook in your default browser. You should see something like this:

7. Navigate to your folder. You can now work on your copy of the notebook.

1 Introduction & Python Refresher

The **Lecture slides** can be found here.

This **lab**'s notebook can be downloaded from here.

1.1 Part I: Powerful Web Mapping Examples

This part of the lab has two main components: 1. The first one will require you to find a partner and work together with her/him 2. And the second one will involve group discussion.

1.1.1 Paired Activity

In pairs, find **three** examples where web maps are used to communicate an idea. Complete the following sheet for each example:

- **Substantive**
  - `Title`: Title of the map/project
  - `Author`: Who is behind the project?
  - `Big idea`: a "one-liner" on what the project tries to accomplish –
  - `Message`: what does the map try to get accross
- **Technical**
  - `URL`:
  - `Interactivity`: does the map let you interact with it in any way? Yes/No
  - `Zoomable`: can you explore the map at different scales? Yes/No
  - `Tooltips`:
  - `Basemap`: Is there an underlying map providing geographical context? Yes/No. If so, who is it provided by?
  - `Technology`: can you guess what technology does this map rely on?

Post each sheet as a separate item on the Teams channel for Lab No.1

1.1.1.1 Example

The project "WHO Coronavirus (COVID-19) Dashboard"

- **Substantive**
  - `Title`: WHO Coronavirus (COVID-19) Dashboard
  - `Author`: World Health Organization
  - `Big idea`: Shows confirmed COVID-19 cases and deaths by country to date
  - `Message`: The project displays a map of the world where COVID-19 cases are shown by country. This element is used to show which countries have had more cases (large trends). A drop down button allows us to visualise the map by a) Total per 100,000 population b) % change in the last 7 days c) newly reported in the last 7 days d) newly reported in the last 24 hours.
- **Technical**
  - `URL`: https://covid19.who.int/
  - `Interactivity`: Yes
  - `Zoomable`: Yes
  - `Tooltips`: Yes
  - `Basemap`: No
  - `Technology`: Unknown

Here are a couple of other COVID-19 examples of web-maps that where basemaps and technology is easier to spot.

- "London School of Hygiene & Tropical Medicine - COVID-19 tracker"
- "Tracking Coronavirus in the United Kingdom: Latest Map and Case Count"

1.1.2 Class discussion

We will select a few examples posted and collectively discuss (some of) the following questions:

1. What makes them powerful, what "speaks" to us?
2. What could be improved, what is counter-intuitive?
3. What design elements do they rely on?
4. What technology do they use?

1.1.3 References

- For an excellent coverage of "visualisation literacy", Chapter 11 of Andy Kirk's "Data Visualisation" is a great start. Lab: Getting up to speed for web mapping
- A comprehensive overview of computational notebooks and how they relate to modern scientific work is available on Ch.1 of the GDS book.
- A recent overview of notebooks in Geography is available in Boeing & Arribas-Bel (2021)

1.2 Part II: Python/Pandas (Refresher)

Gabriele Filomena has prepared this notebook by readapting material shared on this repository.

1.2.1 Python

A quick overview of ubiquitous programming concepts including data types, for loops, if-then-else conditionals, and functions.

```
import numpy as np
import pandas as pd
```

```
# integers (int)
x = 100
type(x)
```

```
# floating-point numbers (float)
x = 100.5
type(x)
```

```
# sequence of characters (str)
x = 'Los Angeles, CA 90089'
len(x)
```

```
# list of items
x = [1, 2, 3, 'USC']
len(x)
```

```
# sets are unique
x = {2, 2, 3, 3, 1}
x
```

```
# tuples are immutable sequences
latlng = (34.019425, -118.283413)
type(latlng)
```

```
# you can unpack a tuple
lat, lng = latlng
type(lat)
```

```
# dictionary of key:value pairs
iceland = {'Country': 'Iceland', 'Population': 372520, 'Capital': 'Reykjavík', '% Foreign Po
type(iceland)
```

```
# you can convert types
x = '100'
print(type(x))
y = int(x)
print(type(y))
```

```
# you can loop through an iterable, such as a list or tuple
for coord in latlng:
    print('Current coordinate is:', coord)
```

```
# loop through a dictionary keys and values as tuples
for key, value in iceland.items():
    print(key, value)
```

```
# booleans are trues/falses
x = 101
x > 100
```

```
# use two == for equality and one = for assignment
x == 100
```

```
# if, elif, else for conditional branching execution
x = 101
if x > 100:
    print('Value is greater than 100.')
elif x < 100:
    print('Value is less than 100.')
else:
    print('Value is 100.')
```

```
# use functions to encapsulate and reuse bits of code
def convert_items(my_list, new_type=str):
    # convert each item in a list to a new type
    new_list = [new_type(item) for item in my_list]
    return new_list

l = [1, 2, 3, 4]
convert_items(l)
```

1.2.2 `pandas` Series and DataFrames

pandas has two primary data structures we will work with: `Series` and `DataFrame`.

1.2.2.1 Pandas Series

```
# a pandas series is based on a numpy array: it's fast, compact, and has more functionality
# it has an index which allows you to work naturally with tabular data
my_list = [8, 5, 77, 2]
my_series = pd.Series(my_list)
my_series
```

```
# look at a list-representation of the index
my_series.index.tolist()
```

```
# look at the series' values themselves
my_series.values
```

```
# what's the data type of the series' values?
type(my_series.values)
```

```
# what's the data type of the individual values themselves?
my_series.dtype
```

1.2.2.2 Pandas DataFrames

```
# a dict can contain multiple lists and label them
my_dict = {'hh_income'  : [75125, 22075, 31950, 115400],
           'home_value' : [525000, 275000, 395000, 985000]}
my_dict
```

```
# a pandas dataframe can contain one or more columns
# each column is a pandas series
# each row is a pandas series
# you can create a dataframe by passing in a list, array, series, or dict
df = pd.DataFrame(my_dict)
df
```

```
# the row labels in the index are accessed by the .index attribute of the DataFrame object
df.index.tolist()
```

```
# the column labels are accessed by the .columns attribute of the DataFrame object
df.columns
```

```
# the data values are accessed by the .values attribute of the DataFrame object
# this is a numpy (two-dimensional) array
df.values
```

1.2.3 Loading data in Pandas

Usually, you'll work with data by loading a dataset file into pandas. CSV is the most common format. But pandas can also ingest tab-separated data, JSON, and proprietary file formats like Excel .xlsx files, Stata, SAS, and SPSS.

Below, notice what pandas's `read_csv` function does:

1. Recognize the header row and get its variable names.
2. Read all the rows and construct a pandas DataFrame (an assembly of pandas Series rows and columns).
3. Construct a unique index, beginning with zero.
4. Infer the data type of each variable (i.e., column).

```
# load a data file
# note the relative filepath! where is this file located?
# use dtype argument if you don't want pandas to guess your data types
df = pd.read_csv('../data/GTD_2022.csv', low_memory = False)
```

```
to_replace = [-9, -99, "-9", "-99"]
for value in to_replace:
    df = df.replace(value, np.NaN)

df['eventid'] = df['eventid'].astype("Int64")
```

```
# dataframe shape as rows, columns
df.shape
```

```
# or use len to just see the number of rows
len(df)
```

```
# view the dataframe's "head"
df.head()
```

```
# view the dataframe's "tail"
df.tail()
```

```
# column data types
df.dtypes
```

```
# or
for dt in df.columns[:10]:
    print(dt, type(dt))
```

1.2.4 Selecting and slicing data from a DataFrame

```
# CHEAT SHEET OF COMMON TASKS
# Operation                       Syntax           Result
#------------------------------------------------------------
# Select column by name           df[col]          Series
# Select columns by name          df[col_list]     DataFrame
# Select row by label             df.loc[label]    Series
# Select row by integer location  df.iloc[loc]     Series
# Slice rows by label             df.loc[a:c]      DataFrame
# Select rows by boolean vector   df[mask]         DataFrame
```

1.2.4.1 Select DataFrame's column(s) by name

```
# select a single column by column name
# this is a pandas series
df['country']
```

```
# select multiple columns by a list of column names
# this is a pandas dataframe that is a subset of the original
df[['country_txt', 'year']]
```

```
# create a new column by assigning df['new_col'] to some values
# people killed every perpetrator
df['killed_per_attacker'] = df['nkill'] / df['nperps']

# inspect the results
df[['country', 'year', 'nkill', 'nperps', 'killed_per_attacker']].head(15)
```

1.2.4.2 Select row(s) by label

```
# use .loc to select by row label
# returns the row as a series whose index is the dataframe column names
df.loc[0]
```

```
# use .loc to select single value by row label, column name
df.loc[15, 'gname'] #group name
```

```
# slice of rows from label 5 to label 7, inclusive
# this returns a pandas dataframe
df.loc[5:7]
```

```
# slice of rows from label 17 to label 27, inclusive
# slice of columns from country_txt to city, inclusive
df.loc[17:27, 'country_txt':'city']
```

```
# subset of rows from with labels in list
# subset of columns with names in list
df.loc[[1, 350], ['country', 'gname']]
```

```
# you can use a column of identifiers as the index (indices do not *need* to be unique)
df_gname = df.set_index('gname')
df_gname.index.is_unique
```

```
df_gname.head(3)
```

```
# .loc works by label, not by position in the dataframe
try:
    df_gname.loc[0]
except KeyError as e:
    print('label not found')
```

```
# the index now contains gname values, so you have to use .loc accordingly to select by row l
df_gname.loc['Taliban'].head()
```

1.2.4.3 Select by (integer) position - Independent from actual Index

```
# get the row in the zero-th position in the dataframe
df.iloc[0]
```

```
# you can slice as well
# note, while .loc is inclusive, .iloc is not
# get the rows from position 0 up to but not including position 3 (ie, rows 0, 1, and 2)
df.iloc[0:3]
```

```
# get the value from the row in position 3 and the column in position 2 (zero-indexed)
df.iloc[3, 6] #country_txt
```

1.2.4.4 Select/filter by value

You can subset or filter a dataframe for based on the values in its rows/columns.

```
# filter the dataframe by urban areas with more than 25 million residents
df[df['nkill'] > 30].head()
```

```
# you can chain multiple conditions together
# pandas logical operators are: | for or, & for and, ~ for not
# these must be grouped by using parentheses due to order of operations
df[['country','nkill', 'nwound']][(df['nkill'] > 200) & (df['nwound'] > 10)].head()
# columns on the left-hand side are here used to slice the resulting output
```

```
# ~ means not... it essentially flips trues to falses and vice-versa
df[['country','nkill', 'nwound']][~(df['nkill'] > 200) & (df['nwound'] > 10)]
```

1.2.5 Grouping and summarizing

```
# group by terroristic group name
groups = df.groupby('gname')
```

```
# what is the median number of people killed per event across the different groups?
groups['nkill'].median().sort_values(ascending=False)
```

```
# look at several columns' medians by group
groups[['nkill', 'nwound', 'nperps']].median()
```

```
# you can create a new dataFrame by directly passing columns between "[[ ]]", after the grou
# to do so, you also need to pass a function that can deal with the values (e.g. sum..etc)
western_europe = df[df.region_txt == 'Western Europe']
western_europe.groupby('country_txt')[['nkill', 'nwound']].sum().sort_values('nkill', ascend
```

1.2.6 Indexes

Each `DataFrame` has an index. Indexes do not have to be unique (but that would be for the best)

```
# resetting index (when loading a .csv file pandas creates an index automatically, from 0 to
df.reset_index(drop = True).sort_index().head() # this does not assign the new index though,
```

```
#this does assign the new index to your df
df = df.reset_index(drop = True).sort_index()
df.head()
```

```
# index isn't unique
df.index.is_unique
```

```
# you can set a new index
# drop -> Delete columns to be used as the new index.
# append ->  whether to append columns to existing index.
df = df.set_index('eventid', drop=True, append=False)
df.index.name = None # remove the index "name"
df.head()

# this index is not ideal, but it's the original source's id
```

1.3 Part III: Geospatial Vector data in Python

Gabriele Filomena has prepared this notebook by readapting material shared on this repository. Copyright (c) 2018, Joris Van den Bossche.

```
%matplotlib inline

import geopandas as gpd
```

1.3.1 Importing geospatial data

GeoPandas builds on Pandas types `Series` and `Dataframe`, by incorporating information about geographical space.

- `GeoSeries`: a Series object designed to store shapely geometry object
- `GeoDataFrame`: object is a pandas DataFrame that has a column with geometry (that contains a *Geoseries*)

We can use the GeoPandas library to read many of GIS file formats (relying on the `fiona` library under the hood, which is an interface to GDAL/OGR), using the `gpd.read_file` function. For example, let's start by reading a shapefile with all the countries of the world (adapted from http://www.naturalearthdata.com/downloads/110m-cultural-vectors/110m-admin-0-countries/, zip file is available in the `/data` directory), and inspect the data:

```
countries = gpd.read_file("../data/ne_countries.zip")
# or if the archive is unpacked:
# countries = gpd.read_file("../data/ne_countries.shp")
```

```
countries.head()
```

```
countries.plot()
```

We observe that:

- Using `.head()` we can see the first rows of the dataset, just like we can do with Pandas.
- There is a `geometry` column and the different countries are represented as polygons
- We can use the `.plot()` (matplotlib) method to quickly get a *basic* visualization of the data

1.3.2 What's a GeoDataFrame?

We used the GeoPandas library to read in the geospatial data, and this returned us a `GeoDataFrame`:

```
type(countries)
```

A GeoDataFrame contains a tabular, geospatial dataset:

- It has a 'geometry' column that holds the geometry information (or features in GeoJSON).
- The other columns are the **attributes** (or properties in GeoJSON) that describe each of the geometries.

Such a `GeoDataFrame` is just like a pandas `DataFrame`, but with some additional functionality for working with geospatial data: * A `geometry` attribute that always returns the column with the geometry information (returning a `GeoSeries`). The column name itself does not necessarily need to be 'geometry', but it will always be accessible as the `geometry` attribute. * It has some extra methods for working with spatial data (area, distance, buffer, intersection, …) see here, for example.

```
countries.geometry.head()
```

```
type(countries.geometry)
```

```
countries.geometry.area
```

It's still a `DataFrame`, so we have all the `pandas` functionality available to use on the geospatial dataset, and to do data manipulations with the attributes and geometry information together. For example, we can calculate the average population over all countries (by accessing the 'pop_est' column, and calling the `mean` method on it):

```
countries['pop_est'].mean()
```

```
africa = countries[countries['continent'] == 'Africa']
```

```
africa.plot();
```

The rest of the tutorial is going to assume you already know some pandas basics, but we will try to give hints for that part for those that are not familiar.

Important:

- A `GeoDataFrame` allows to perform typical tabular data analysis together with spatial operations
- A `GeoDataFrame` (or *Feature Collection*) consists of:
  - **Geometries** or **features**: the spatial objects
  - **Attributes** or **properties**: columns with information about each spatial object

1.3.3 Geometries: Points, Linestrings and Polygons

Spatial **vector** data can consist of different types, and the 3 fundamental types are:

- **Point** data: represents a single point in space.
- **Line** data ("LineString"): represented as a sequence of points that form a line.
- **Polygon** data: represents a filled area.

And each of them can also be combined in multi-part geometries (See https://shapely.readthedocs.io/en/stable/m objects for extensive overview).

For the example we have seen up to now, the individual geometry objects are Polygons:

```
print(countries.geometry[2])
```

Let's import some other datasets with different types of geometry objects.

A dateset about cities in the world (adapted from http://www.naturalearthdata.com/downloads/110m-cultural-vectors/110m-populated-places/, zip file is available in the `/data` directory), consisting of `Point` data:

```
cities = gpd.read_file("../data/ne_cities.zip")
```

```
print(cities.geometry[0])
```

And a dataset of rivers in the world (from http://www.naturalearthdata.com/downloads/50m-physical-vectors/50m-rivers-lake-centerlines/, zip file is available in the `/data` directory) where each river is a `(Multi-)LineString`:

```
rivers = gpd.read_file("../data/ne_rivers.zip")
```

```
print(rivers.geometry[0])
```

1.3.4 The `shapely` library

The individual geometry objects are provided by the `shapely` library

```
from shapely.geometry import Point, Polygon, LineString
```

```
type(countries.geometry[0])
```

To construct one ourselves:

```
p = Point(0, 0)
```

```
print(p)
```

```
polygon = Polygon([(1, 1), (2,2), (2, 1)])
```

```
polygon.area
```

```
polygon.distance(p)
```

Important:

Single geometries are represented by `shapely` objects:

- If you access a single geometry of a GeoDataFrame, you get a shapely geometry object

- Those objects have similar functionality as geopandas objects (GeoDataFrame/GeoSeries). For example:
  - `single_shapely_object.distance(other_point)` -> distance between two points
  - `geodataframe.distance(other_point)` -> distance for each point in the geodataframe to the other point

1.3.5 Plotting

```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 1, figsize=(15, 10))
countries.plot(ax = ax, edgecolor='k', facecolor='none')
rivers.plot(ax=ax)
cities.plot(ax=ax, color='red')
ax.set(xlim=(-20, 60), ylim=(-40, 40))
```

1.3.6 Creating GeoDataFrames (withouth specifying the CRS)

```
gpd.GeoDataFrame({
    'geometry': [Point(1, 1), Point(2, 2)],
    'attribute1': [1, 2],
    'attribute2': [0.1, 0.2]})
```

```
# Creating a GeoDataFrame from an existing dataframe
# For example, if you have lat/lon coordinates in two columns:
df = pd.DataFrame(
    {'City': ['Buenos Aires', 'Brasilia', 'Santiago', 'Bogota', 'Caracas'],
     'Country': ['Argentina', 'Brazil', 'Chile', 'Colombia', 'Venezuela'],
     'Latitude': [-34.58, -15.78, -33.45, 4.60, 10.48],
     'Longitude': [-58.66, -47.91, -70.66, -74.08, -66.86]})
```

```
gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df.Longitude, df.Latitude))
gdf
```

2 Practice

Throughout the exercises in this course, we will work with several datasets about the city of Paris.

Here, we start with the following datasets:

- The administrative districts of Paris (https://opendata.paris.fr/explore/dataset/quartier_paris/): `paris_districts_utm.geojson`
- Real-time (at the moment I downloaded them ..) information about the public bicycle sharing system in Paris (vélib, https://opendata.paris.fr/explore/dataset/stations-velib-disponibilites-en-temps-reel/information/): `data/paris_bike_stations_mercator.gpkg`

Both datasets are provided as spatial datasets using a GIS file format.

Excercise 1:

We will start by exploring the bicycle station dataset (available as a GeoPackage file: `data/paris_bike_stations_mercator.gpkg`)

- Read the stations datasets into a GeoDataFrame called `stations`.
- Check the type of the returned object
- Check the first rows of the dataframes. What kind of geometries does this datasets contain?
- How many features are there in the dataset?

Hints

- Use `type(..)` to check any Python object type
- The `gpd.read_file()` function can read different geospatial file formats. You pass the file name as first argument.
- Use the `.shape` attribute to get the number of features

Exercise 2:

- Make a quick plot of the `stations` dataset.
- Make the plot a bit larger by setting the figure size to (12, 6) (hint: the `plot` method accepts a `figsize` keyword).

Exercise 3:

Next, we will explore the dataset on the administrative districts of Paris (available as a GeoJSON file: `../data/paris_districts_utm.geojson`)

- Read the dataset into a GeoDataFrame called `districts`.
- Check the first rows of the dataframe. What kind of geometries does this dataset contain?
- How many features are there in the dataset? (hint: use the `.shape` attribute)
- Make a quick plot of the `districts` dataset (set the figure size to (12, 6)).

Exercise 4:

What are the largest districts (biggest area)?

- Calculate the area of each district.
- Add this area as a new column to the `districts` dataframe.
- Sort the dataframe by the area column from largest to smallest values (descending).

Hints

- Adding a column can be done by assigning values to a column using the same square brackets syntax: `df['new_col'] = values`
- To sort the rows of a DataFrame, use the `sort_values()` method, specifying the colum to sort on with the `by='col_name'` keyword. Check the help of this method to see how to sort ascending or descending.

2.1 Part IV: Coordinate reference systems & Projections

Gabriele Filomena has prepared this notebook by readapting material shared on this repository.


```
countries = gpd.read_file("../data/ne_countries.zip")
cities = gpd.read_file("../data/ne_cities.zip")
rivers = gpd.read_file("../data/ne_rivers.zip")
```

2.1.1 Coordinate reference systems

Up to now, we have used the geometry data with certain coordinates without further wondering what those coordinates mean or how they are expressed.

> The **Coordinate Reference System (CRS)** relates the coordinates to a specific location on earth.

For an in-depth explanation, see https://docs.qgis.org/2.8/en/docs/gentle_gis_introduction/coordinate_referer

2.1.1.1 Geographic coordinates

Degrees of latitude and longitude.

E.g. 48°51 N, 2°17 E

The most known type of coordinates are geographic coordinates: we define a position on the globe in degrees of latitude and longitude, relative to the equator and the prime meridian. With this system, we can easily specify any location on earth. It is used widely, for example in GPS. If you inspect the coordinates of a location in Google Maps, you will also see latitude and longitude.

Attention!

in Python we use (lon, lat) and not (lat, lon)

- Longitude: [-180, 180]{{1}}
- Latitude: [-90, 90]{{1}}

2.1.2 Projected coordinates

`(x, y)` coordinates are usually in meters or feet

Although the earth is a globe, in practice we usually represent it on a flat surface: think about a physical map, or the figures we have made with Python on our computer screen. Going from the globe to a flat map is what we call a *projection*.

We project the surface of the earth onto a 2D plane so we can express locations in cartesian x and y coordinates, on a flat surface. In this plane, we then typically work with a length unit such as meters instead of degrees, which makes the analysis more convenient and effective.

However, there is an important remark: the 3 dimensional earth can never be represented perfectly on a 2 dimensional map, so projections inevitably introduce distortions. To minimize such errors, there are different approaches to project, each with specific advantages and disadvantages.

Some projection systems will try to preserve the area size of geometries, such as the Albers Equal Area projection. Other projection systems try to preserve angles, such as the Mercator projection, but will see big distortions in the area. Every projection system will always have some distortion of area, angle or distance.

Projected size vs actual size (Mercator projection):

2.1.3 Coordinate Reference Systems in Python / GeoPandas

A GeoDataFrame or GeoSeries has a `.crs` attribute which holds (optionally) a description of the coordinate reference system of the geometries:

```
countries.crs
```

For the `countries` dataframe, it indicates that it uses the EPSG 4326 / WGS84 lon/lat reference system, which is one of the most used for geographic coordinates.

It uses coordinates as latitude and longitude in degrees, as can you be seen from the x/y labels on the plot:

```
countries.plot()
```

The `.crs` attribute returns a `pyproj.CRS` object. To specify a CRS, we typically use some string representation:

- **EPSG code** Example: `EPSG:4326` = WGS84 geographic CRS (longitude, latitude)

For more information, see also http://geopandas.readthedocs.io/en/latest/projections.html.

2.1.3.1 Transforming to another CRS

We can convert a GeoDataFrame to another reference system using the `to_crs` function.

For example, let's convert the countries to the World Mercator projection (http://epsg.io/3395):

```
# remove Antartica, as the Mercator projection cannot deal with the poles
countries = countries[(countries['name'] != "Antarctica")]
countries_mercator = countries.to_crs(epsg=3395)  # or .to_crs("EPSG:3395")
countries_mercator.plot()
```

Note the different scale of x and y.

2.1.3.2 Why using a different CRS?

There are sometimes good reasons you want to change the coordinate references system of your dataset, for example:

- Different sources with different CRS -> need to convert to the same crs.

- Different countries/geographical areas with different CRS.
- Mapping (distortion of shape and distances).
- Distance / area based calculations -> ensure you use an appropriate projected coordinate system expressed in a meaningful unit such as meters or feet (**not degrees!**).

Important:

All the calculations (e.g. distance, spatial operations, etc.) that take place in `GeoPandas` and `Shapely` assume that your data is represented in a 2D cartesian plane, and thus the result of those calculations will only be correct if your data is properly projected.

2.2 Practice

Again, we will go back to the Paris datasets. Up to now, we provided the datasets in an appropriate projected CRS for the exercises. But the original data were actually using geographic coordinates. In the following exercises, we will start from there.

Going back to the Paris districts dataset, this is now provided as a GeoJSON file (`"../data/paris_districts.geojson"`) in geographic coordinates.

For converting the layer to projected coordinates, we will use the standard projected CRS for France is the RGF93 / Lambert-93 reference system, referenced by the `EPSG:2154` number.

Exercise: Projecting a GeoDataFrame

- Read the districts datasets (`../data/paris_districts.geojson"`) into a GeoDataFrame called `districts`.
- Look at the CRS attribute of the GeoDataFrame. Do you recognize the EPSG number?
- Make a plot of the `districts` dataset.
- Calculate the area of all districts.
- Convert the `districts` to a projected CRS (using the `EPSG:2154` for France). Call the new dataset `districts_RGF93`.
- Make a similar plot of `districts_RGF93`.
- Calculate the area of all districts again with `districts_RGF93` (the result will now be expressed in m^2).

Hints

- The CRS information is stored in the `.crs` attribute of a GeoDataFrame.
- Making a simple plot of a GeoDataFrame can be done with the `.plot()` method.
- Converting to a different CRS can be done with the `.to_crs()` method, and the CRS can be specified as an EPSG number using the `epsg` keyword.

3 Static Maps in Python

The **Lecture slides** can be found here.

This **lab**'s notebook can be downloaded from here.

3.1 Part I: Basic Maps

In this session, we will use the libraries `matplotlib` and `contextily` to plot the information represented into different `GeoDataFrames`. We will look into plotting `Point`, `LineString` and `Polygon GeoDataFrames`. Most of the plots here are rather ugly but, at this point, the goal is to get familiar with the parameters of the plot function and what can be done with them.

```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import geopandas as gpd
import pandas as pd
import osmnx as ox
import contextily as ctx
import seaborn as sns
```

3.1.1 Plotting Points

Load the data of terrorist attacks 1970-2020 and choose a country. Germany is used as a case study here but feel free to change the country. If you do so, also change the `crs` (see https://epsg.io).

```
attacks = pd.read_csv("../data/GTD_2022.csv", low_memory = False)
```

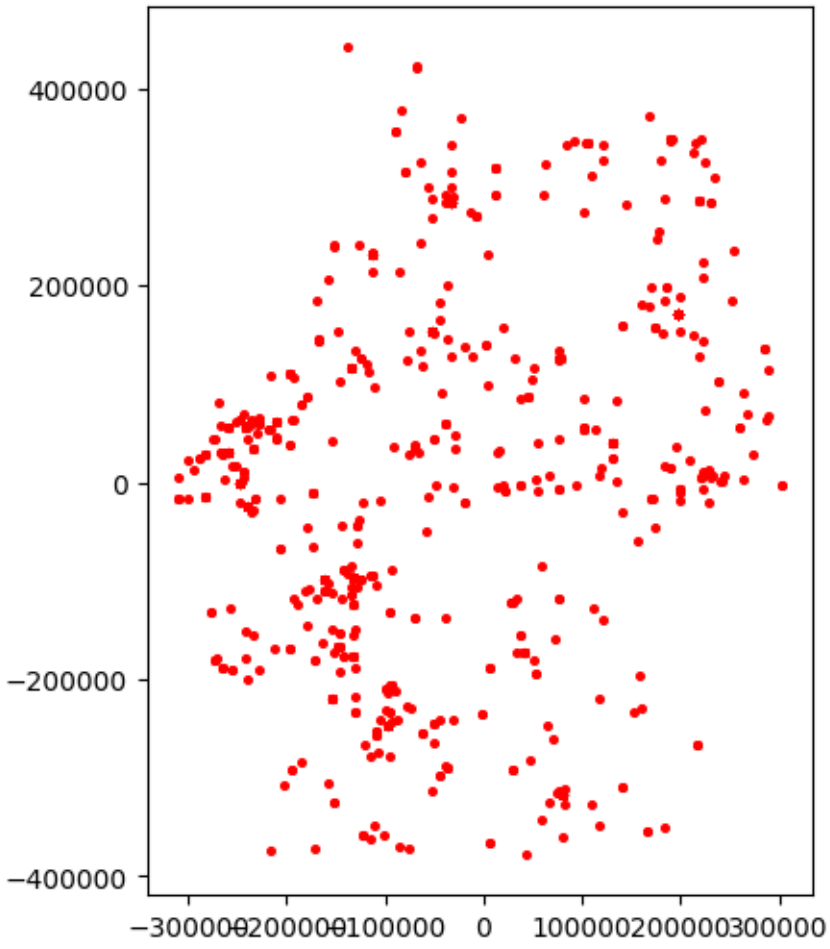
Creating the `GeoDataFrame` from the `DataFrame`

```
germany = ['West Germany (FRG)', 'Germany', 'East Germany (GDR)'] # Germany was split till 19
df = attacks[attacks.country_txt.isin(germany)].copy()

# Uncomment the lines below for other countries that haven't changed their denominations/boun
# country = 'France'
# df = attacks[attacks.country_txt == country].copy()#
wgs = 'EPSG:4326'
germany_crs = 'EPSG:4839'
gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df.longitude, df.latitude), crs = wgs)
gdf = gdf[~gdf.geometry.is_empty] # remove empty geometries
gdf.to_file("../data/germany.shp")
gdf = gdf.to_crs(germany_crs)
```

```
C:\Users\gfilo\AppData\Local\Temp\ipykernel_700\3815068564.py:11: UserWarning: Column names l
  gdf.to_file("data/germany.shp")
```

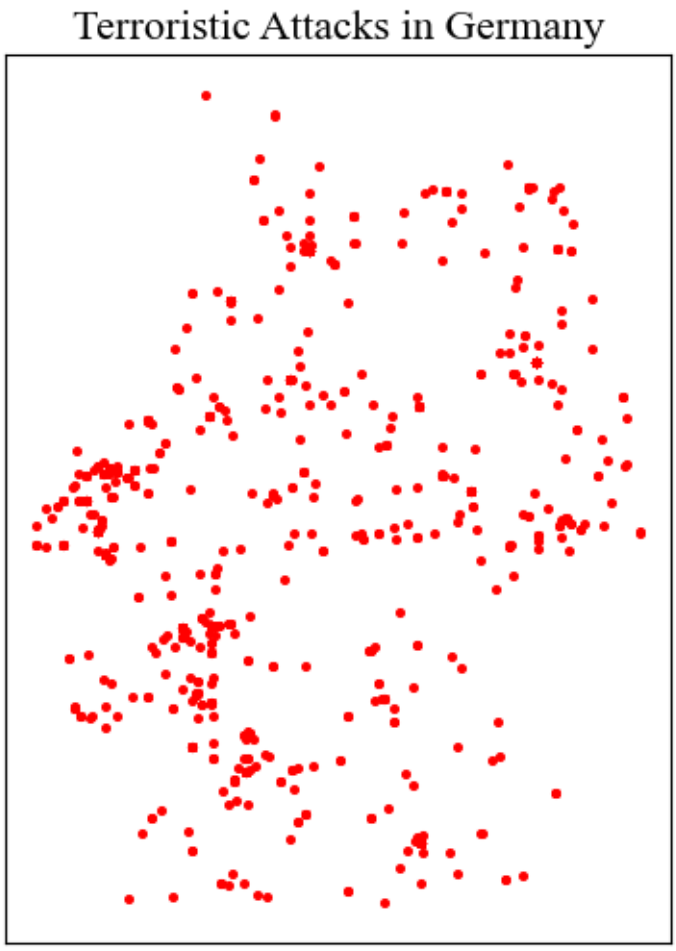
Basic plotting

```
# prepare the axis and coordinate
nr_rows = 1
nr_cols = 1
fig, ax = plt.subplots(nr_cols, nr_rows, figsize=(8, 6))
gdf.plot(ax=ax, color='red', markersize=7)
```



Slightly improving the plot:

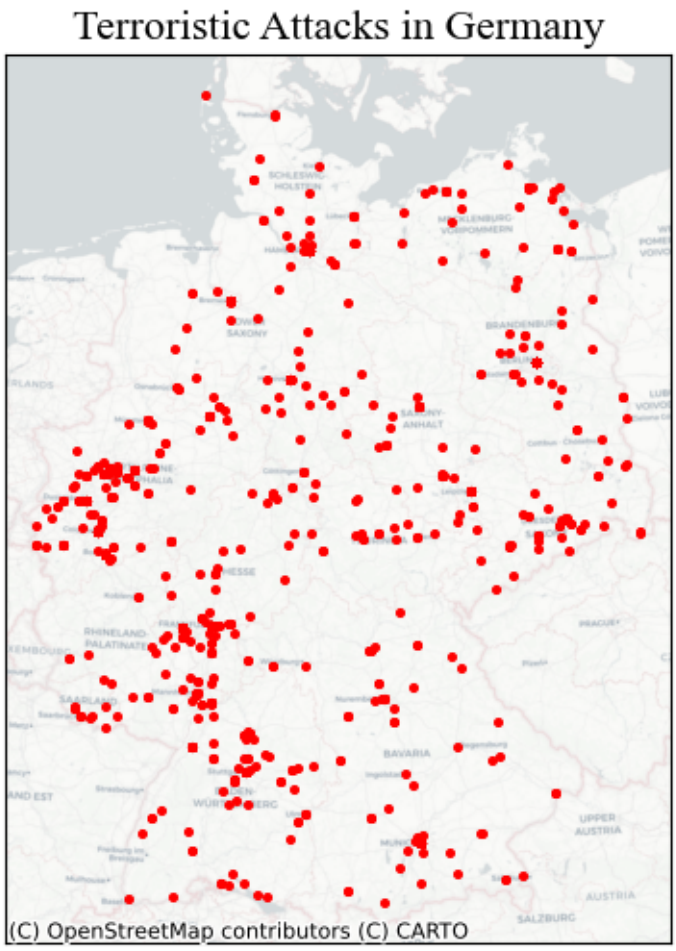
```
# removing ticks
ax.xaxis.set_ticklabels([])
ax.yaxis.set_ticklabels([])
ax.tick_params(axis= 'both', which= 'both', length=0)
title_parameters = {'fontsize':'16', 'fontname':'Times New Roman'}
ax.set_title("Terroristic Attacks in Germany", **title_parameters)
fig
```



3.1.1.1 Adding some context: Base Maps with `Contextily`

see providers and options here https://xyzservices.readthedocs.io/en/stable/introduction.html

```
source = ctx.providers.CartoDB.Positron
ctx.add_basemap(ax, crs= gdf.crs.to_string(), source= source)
# replot
fig
```




```
<Figure size 640x480 with 0 Axes>
```

3.1.1.2 Parameters specific to `Point` in the `plot` method

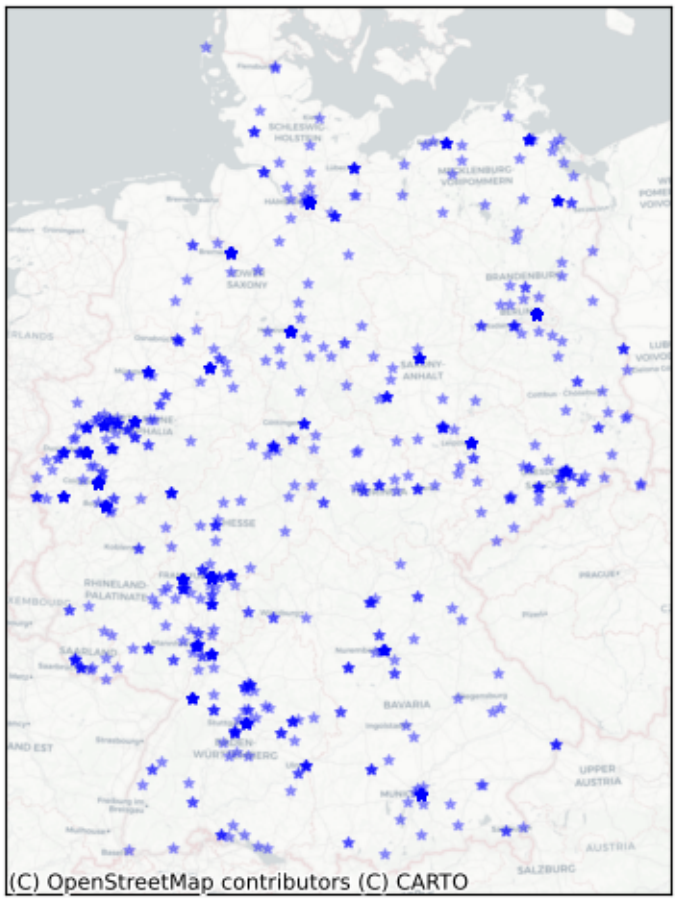
- `markersize`: numerical value (for now)
- `marker`: see https://matplotlib.org/stable/api/markers_api.html

3.1.1.2.1 Other properties, shape independent:

- `color`: https://matplotlib.org/3.1.0/gallery/color/named_colors.html
- `alpha`: regulates transparency of the shape: 0 to 1

```
# first, let's make a function

def ax_ticks_off(ax):
    ax.xaxis.set_ticklabels([])
    ax.yaxis.set_ticklabels([])
    ax.tick_params(axis= 'both', which= 'both', length=0)
```

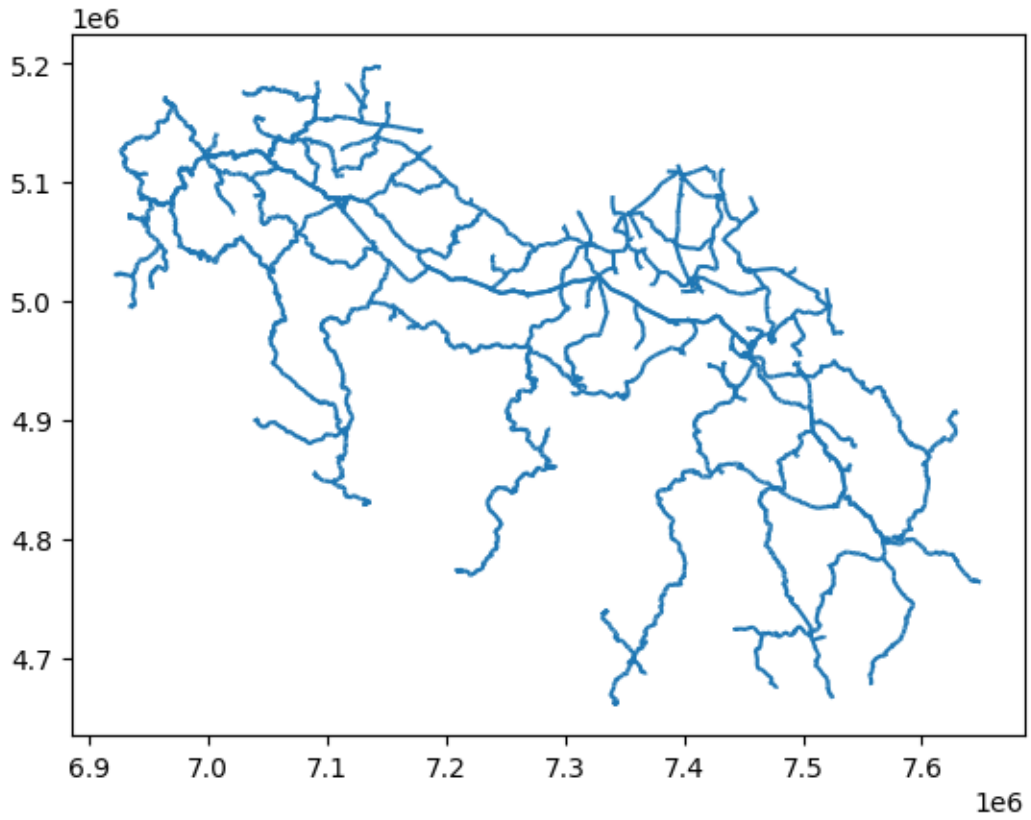
```
# prepare the axis and coordinate
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
gdf.plot(ax=ax, markersize = 15, color = 'blue', marker = '*', alpha = 0.3)
ctx.add_basemap(ax, crs= gdf.crs.to_string(), source= source)
ax_ticks_off(ax)
```



3.1.2 Plotting LineStrings

Let's import railway tracks in the Western Balkans (Slovenia, Croatia, Bosnia & Herzegovina, Montenegro, Serbia, Kosovo)

```
wb_crs = 'EPSG:31277'
lines_gdf = gpd.read_file("..\data\wb_railways.shp")
lines_gdf.plot()
```




```
# prepare the plot
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
lines_gdf.plot(ax=ax, linewidth = 0.8, color = 'blue', alpha = 1)
ctx.add_basemap(ax, crs= lines_gdf.crs.to_string(), source = ctx.providers.Esri.WorldGrayCan
ax_ticks_off(ax)
ax.set_title("Railway infrastructure in the West Balkans", **title_parameters) #parameters a
```

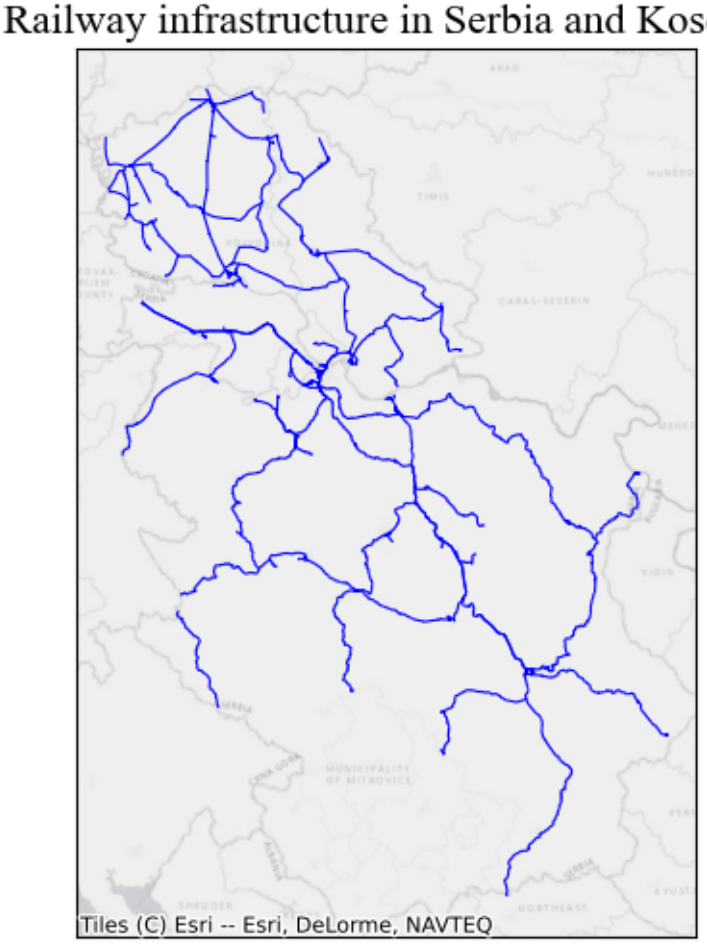
```
Text(0.5, 1.0, 'Railway infrastructure in the West Balkans')
```

One can also filter prior to plotting, based on the columns in the GeoDataFrame. First we download Serbia's Boundary with `OSMNX`, more on that later on. Then we filter `lines_gdf` with a `within` operation.

```
serbia = ox.geocode_to_gdf('Serbia')
serbia = serbia.to_crs(wb_crs)
serbia_lines = lines_gdf[lines_gdf.geometry.within(serbia.iloc[0].geometry)].copy() #there's
```

```
# prepare the plot
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
serbia_lines.plot(ax=ax, linewidth = 0.8, color = 'blue', alpha = 1)
ctx.add_basemap(ax, crs= lines_gdf.crs.to_string(), source = ctx.providers.Esri.WorldGrayCan
ax_ticks_off(ax)
ax.set_title("Railway infrastructure in Serbia and Kosovo", **title_parameters) #parameters
```

```
Text(0.5, 1.0, 'Railway infrastructure in Serbia and Kosovo')
```



3.1.2.1 Parameters specific to `LineString`:

- `linewidth`: numerical value (for now).
- `capstyle`: controls how Matplotlib draws the corners where two different line segments meet. See https://matplotlib.org/stable/gallery/lines_bars_and_markers/capstyle.html
- `joinstyle`': controls how Matplotlib draws the corners where two different line segments meet. https://matplotlib.org/stable/gallery/lines_bars_and_markers/joinstyle.html

```
# prepare the plot
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
serbia_lines.plot(ax=ax, linewidth = 0.9, color = 'black', alpha = 1, capstyle = 'round', jo
ax.set_axis_off() # we don't need the ticks function
ax.set_title("Railway infrastructure in Serbia", **title_parameters) #parameters as above
```

```
Text(0.5, 1.0, 'Railway infrastructure in Serbia')
```

Railway infrastructure in Serbia

3.1.3 Plotting Polygons

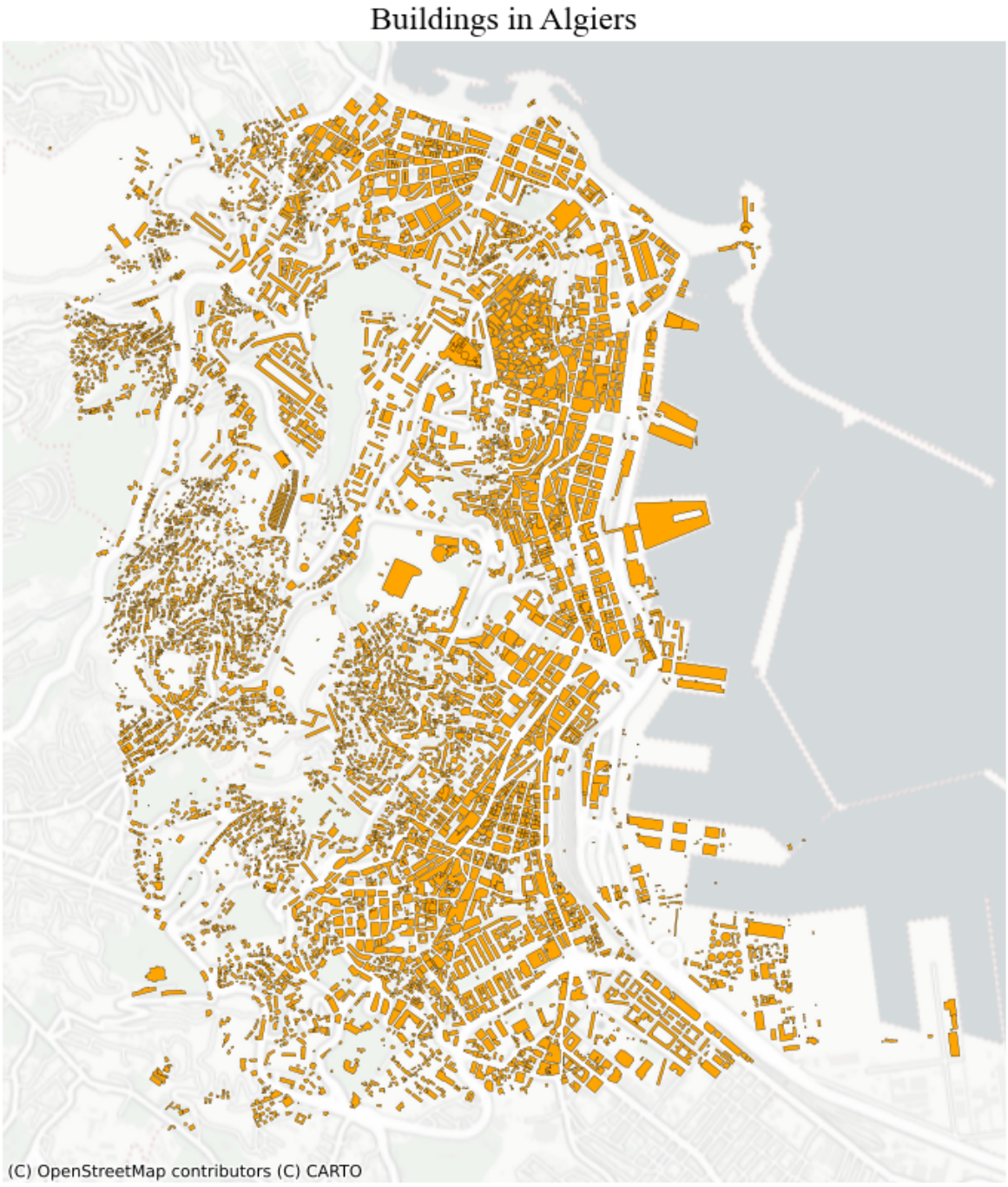
We are again using `OSMNX` to download data from `OpenStreetMap` automatically. In this case, we will get building footprints from the city of Algiers in Alageria.

3.1.3.1 Parameter specific to `Polygon`:

- `edgecolor`: the outline of the polygon, by default = `None` (often better).
- `linewidth`: the width of the outline of the polygon.

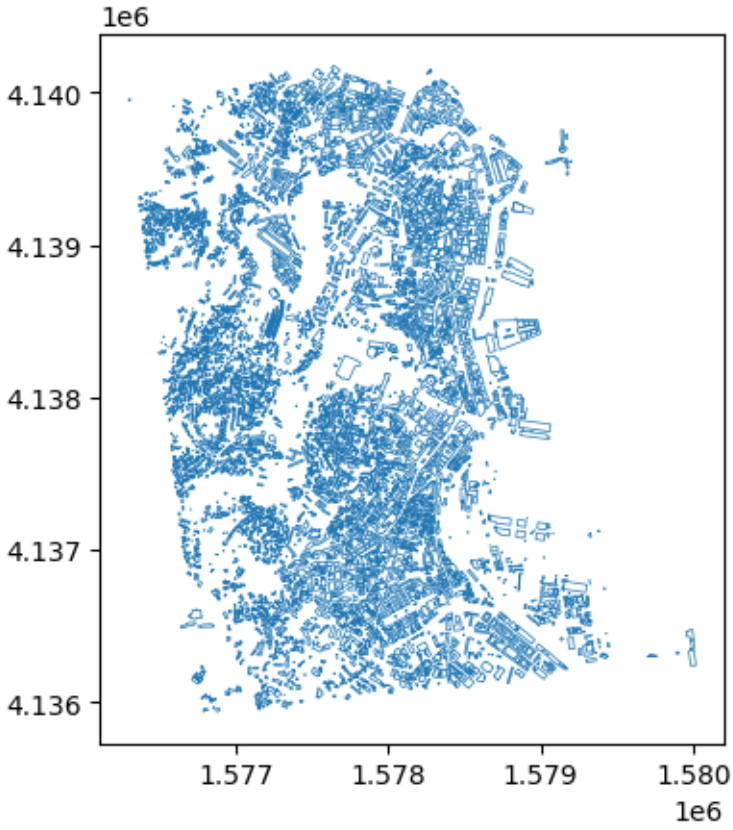
```
algeria_crs = 'EPSG:30729'
tags = {"building": True} #OSM tags
buildings = ox.features_from_address("Algiers, Algeria", tags = tags, dist = 2000)
buildings = buildings.reset_index()
 # sometimes building footprints are represented by Points, let's disregard them
buildings = buildings[(buildings.geometry.geom_type == 'Polygon') | (buildings.geometry.geom
buildings = buildings.to_crs(algeria_crs)
```

```
fig, ax = plt.subplots(1, 1, figsize=(15, 10))
ax.set_title("Buildings in Algiers", **title_parameters)
ax.set_axis_off() # we don't need the ticks function
buildings.plot(ax=ax, color = 'orange', edgecolor = 'black', lw = 0.2)
source = ctx.providers.CartoDB.PositronNoLabels
ctx.add_basemap(ax, crs= buildings.crs.to_string(), source= source)
```



For polygons, you can also plot just the boundaries of the geometries by:

```
buildings.boundary.plot(lw = 0.5)
```



3.1.4 Plotting more than one layer together

Let's also download roads for Algiers

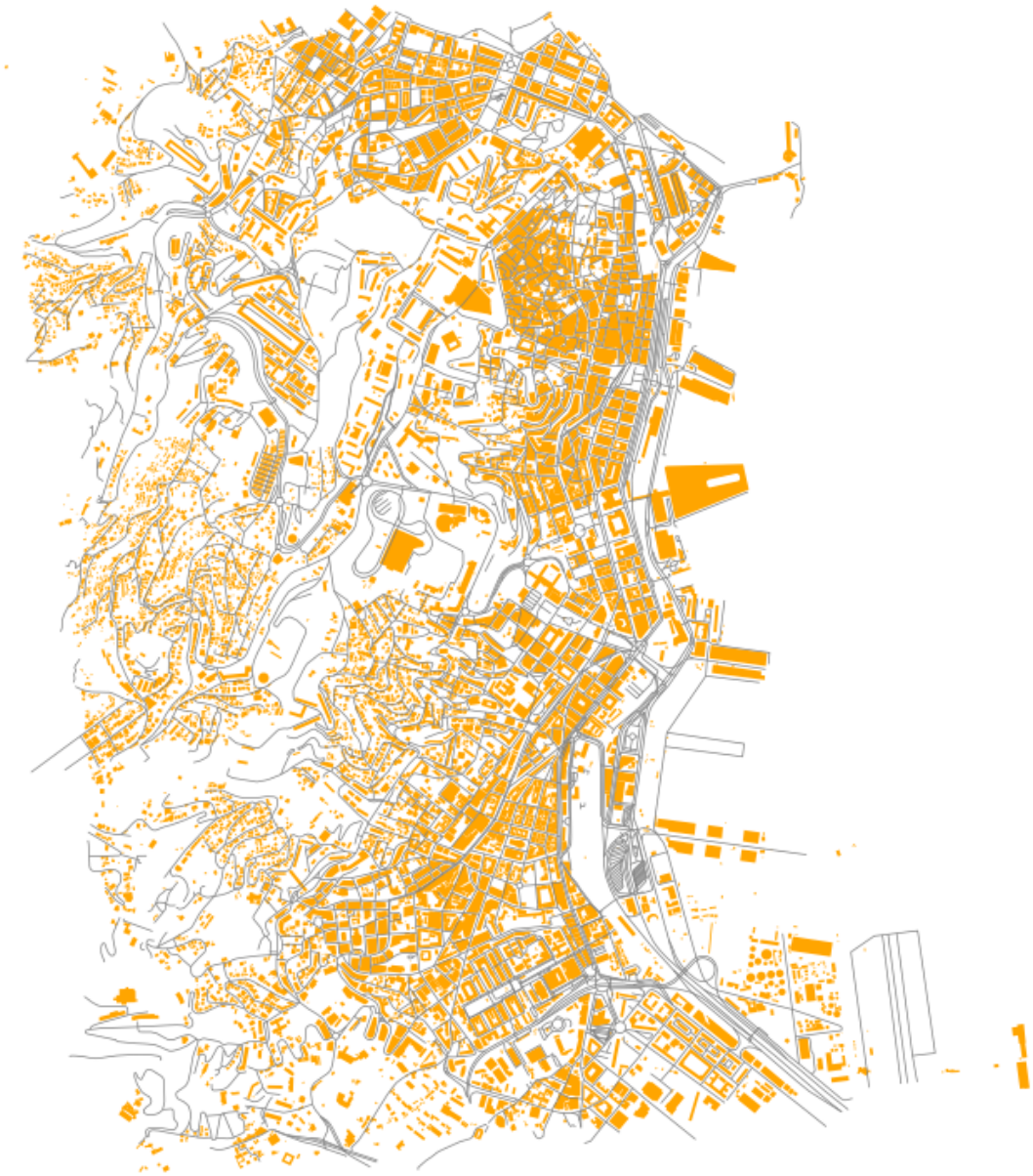
```
tags = {"highway": True} #OSM tags
roads = ox.features_from_address("Algiers, Algeria", tags = tags, dist = 2000)
roads = roads.reset_index()
roads = roads.to_crs(algeria_crs)
 # sometimes building footprints are represented by Points, let's disregard them
roads = roads[roads.geometry.geom_type == 'LineString']
```

And plot everything togehter. It's important to keep in mind that the last layer is always rendered on top of the others. In other words, they may cover the previous ones.

However, you can prevent this by passing arguments to the parameter `zorder` in the `plot` method. The layer with the higher zorder value will be plotted on top.

```
fig, ax = plt.subplots(1, 1, figsize=(15, 10))
ax.set_title("Buildings and Roads in Algiers", **title_parameters)
ax.set_axis_off() # we don't need the ticks function
# only roads within the extent of the buildings layer
roads[roads.geometry.within(buildings.unary_union.envelope)].plot(ax=ax, color = 'grey', lw =
buildings.plot(ax=ax, color = 'orange')
```

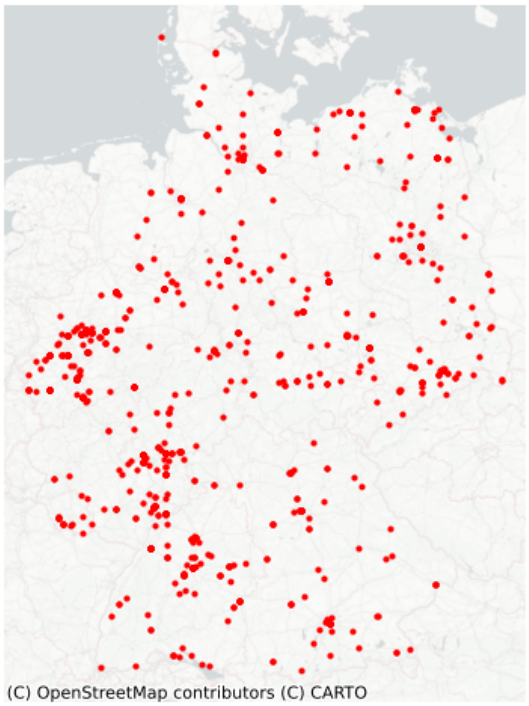
Buildings and Roads in Algiers

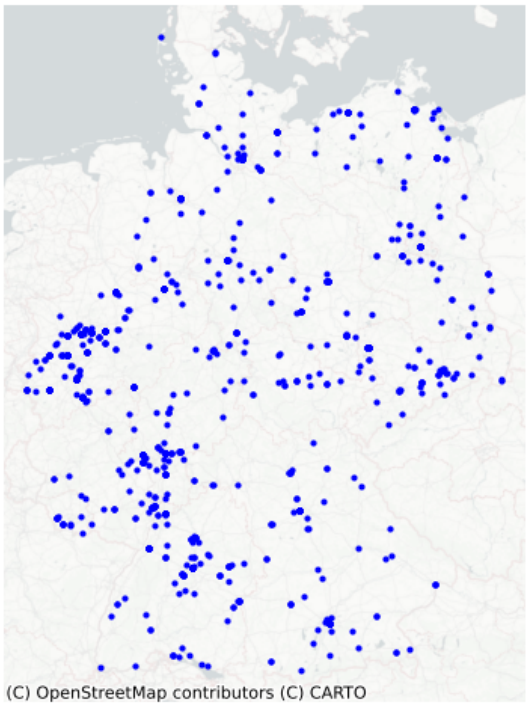


3.1.5 Sub-plots

To obtain multiple sub-plots, we manipulate the `nrows`, `ncols` parameters. We can use this approach to: * Plot the same layer with different properties.

```
fig, axes = plt.subplots(1, 2, figsize=(10, 6))
colors = ['red', 'blue']

for n, ax in enumerate(axes):
    gdf.plot(ax=ax, markersize = 4, color = colors[n])
    ax.set_axis_off()
    ctx.add_basemap(ax, crs= gdf.crs.to_string(), source= source)
```

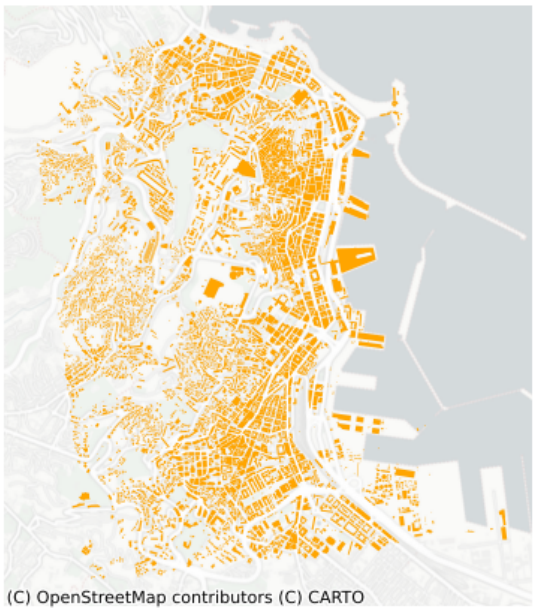




- Plot different layers.

```
fig, axes = plt.subplots(1, 2, figsize=(10, 6))
gdfs = [buildings, roads]
colors = ['orange', 'grey']

buildings.plot(ax=axes[0], color = 'orange', edgecolor = 'none')
roads.plot(ax=axes[1], color = 'gray', lw = 0.5)
```

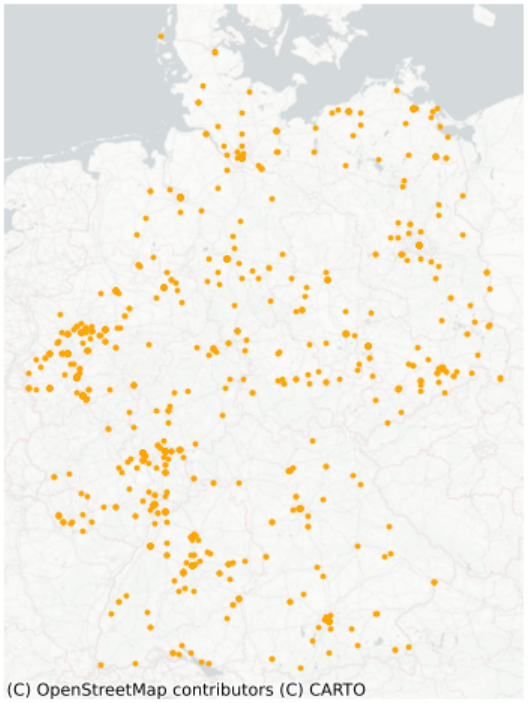
```
for ax in axes:
    ax.set_axis_off()
    ctx.add_basemap(ax, crs= buildings.crs.to_string(), source = source)
```

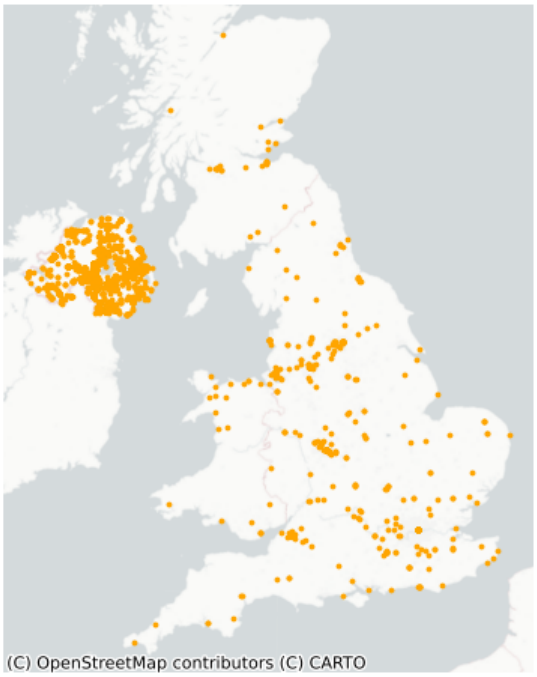


- Analyse phenomena across different geographical areas. For example, terrorism in Germany and in the UK.

```
# let's prepare the gdf for the UK
df_uk = attacks[attacks.country_txt == 'United Kingdom'].copy()
uk_crs = 'EPSG:27700'
gdf_uk = gpd.GeoDataFrame(df_uk, geometry=gpd.points_from_xy(df_uk.longitude, df_uk.latitude
gdf_uk = gdf_uk.to_crs(uk_crs)
```

```
fig, axes = plt.subplots(1, 2, figsize=(10, 6))
gdfs = [gdf, gdf_uk]

for n, ax in enumerate(axes):
    gdf_tmp = gdfs[n]
    gdf_tmp.plot(ax=ax, color = 'orange', markersize = 3)
    ax.set_axis_off()
    ctx.add_basemap(ax, crs= gdf_tmp.crs.to_string(), source= source)
```





Exercise:

- Think about the plots above and how they could be improved.
- Copy and paste the code and execute the functions playing with the different parameters.
- Produce a neat map using the `GeoDataFrame`s available in this notebook or the ones employed in the previous sessions, making use of the elements/parameters discussed here.
- Try out different tiles for the basemap to familiarise yourself with what's available.

3.2 Part II: Choropleth Mapping

```
import geoplot.crs as gcrs
import geoplot as gplt
```

Data

For this second part of the tutorial, we will use some data at the municipality level for Serbia. The data contains information regarding poverty level, average income, population and tourism. The data is taken from https://data.stat.gov.rs/?caller=SDDB&languageCode=en-US and

can be associated to the polygons representing the administrative boundaries of the municipalities. These boundaries can be found here https://data.humdata.org/dataset/geoboundaries-admin-boundaries-for-serbia?force_layout=desktop. While most of the data refers to 2023, the admin boundaries file traces back to 2017. Thus, it may contain obsolete information (few changes may occur).

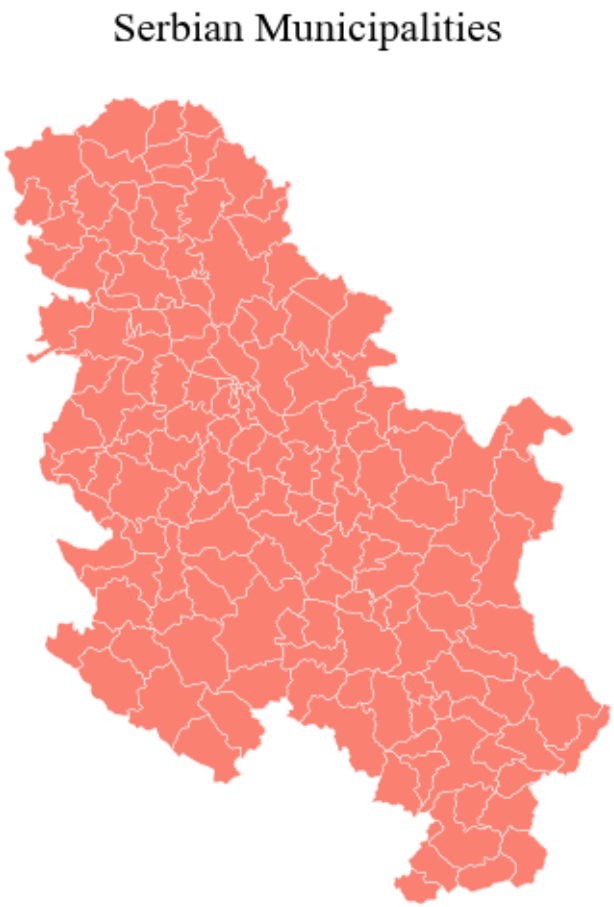
Later on, we will go back to the terrorism dataset.

```
# This will be different on your computer and will depend on where
# you have downloaded the files
serbia_crs = 'EPSG:31277'
wgs = 'EPSG:4326'
serbia_admin = gpd.read_file('../data/serbia_admin.shp')
serbia_admin.set_index('townID', inplace = True, drop = True)
serbia_admin = serbia_admin.to_crs(serbia_crs)
```

Let's plot the `GeoDataFrame` following the last session's steps.

```
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
serbia_admin.plot(ax = ax, color = 'salmon', linewidth = 0.3, edgecolor = 'white')
ax.set_axis_off()
title_parameters = {'fontsize':'16', 'fontname':'Times New Roman'}
ax.set_title("Serbian Municipalities", **title_parameters) #parameters as above
```

```
Text(0.5, 1.0, 'Serbian Municipalities')
```



The we load the data and merge it into the `GeoDataFrame`, before getting rid of municipalities that do not have a corresponding shape/record in the GeoDataFrame (probably the result of changes in the national subdivisions).

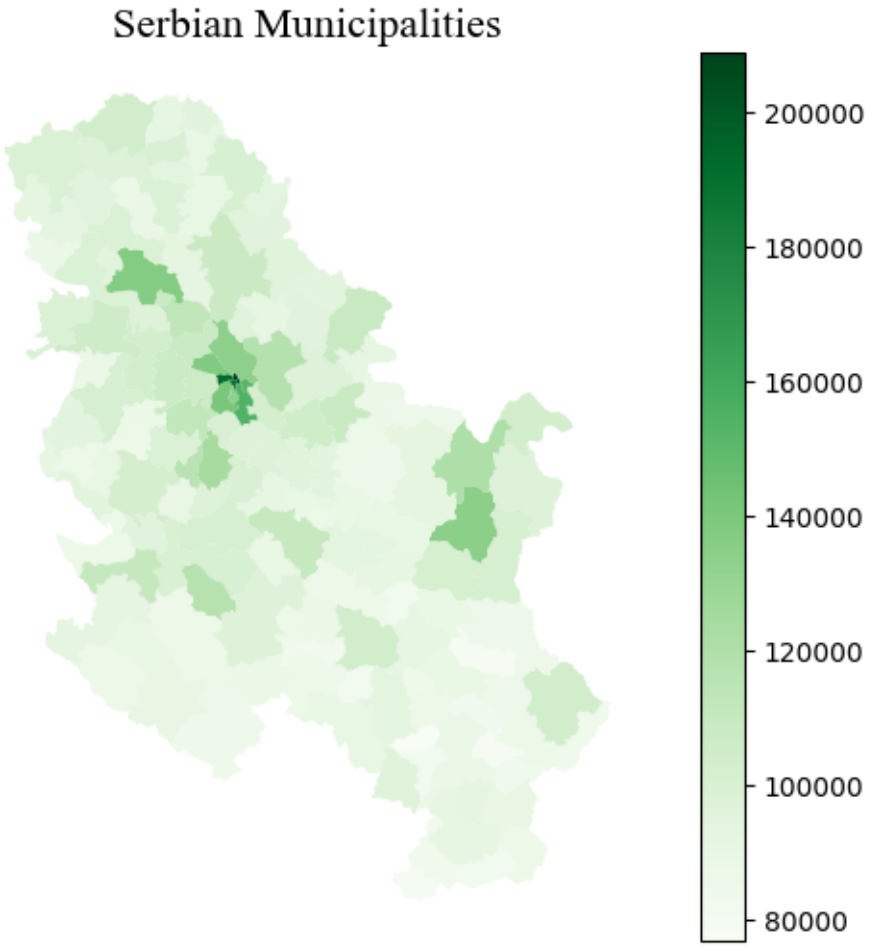
```
data = pd.read_csv("../data/serbia_data.csv") #some slavic characters
data.drop('name_en', axis = 1, inplace = True)
serbia_admin = pd.merge(serbia_admin, data, left_on = "townID", right_on = "id")
serbia_admin = serbia_admin[serbia_admin.id.notna()]
serbia_admin['id'] = serbia_admin['id'].astype('int64')
serbia_admin.head()

#let's save the so-obtained gdf for later (encoding for dealing with slavic characters).
serbia_admin.to_file("../data/serbia_data.shp", encoding='utf-8')
```

Creating a choropleth map is rather straightforward and can ben done by using few other

parameters. Reflect on what you see and whether the map below is informative.

```
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
serbia_admin.plot(ax = ax, column = 'gross', linewidth = 0.3, cmap = 'Greens', legend = True)
ax.set_axis_off()
title_parameters = {'fontsize':'16', 'fontname':'Times New Roman'}
ax.set_title("Serbian Municipalities", **title_parameters) #parameters as above
```

```
Text(0.5, 1.0, 'Serbian Municipalities')
```



3.2.1 Choropleth Maps for Numerical Variables

We are essentially using the same approach employed for creating basic maps, the method `plot`, but we now need to pass arguments to some new parameters to specify which column

is to be represented and how. As an optional argument, one can set legend to `True` and the resulting figure will include a colour bar.

- `column`: the name of the column representing the variable that we want to use to colour-code our shapes.
- `scheme`: the scheme used to colour the shapes based on the variable values.
- `cmap`: the colormap used to show variation.

3.2.1.1 Colormaps

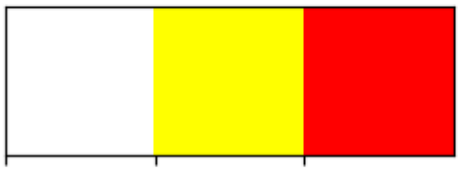
Built-in colour maps can be found here https://matplotlib.org/stable/gallery/color/colormap_reference.html. However one can create new ones as follows from a list of colours:

```
from seaborn import palplot
from matplotlib.colors import LinearSegmentedColormap

colors = [(0.00, 0.00, 0.00,1), (0.248, 0.0271, 0.569, 1), (0.0311, 0.258, 0.646,1),
            (0.019, 0.415, 0.415,1), (0.025, 0.538, 0.269,1), (0.0315, 0.658, 0.103,1),
            (0.331, 0.761, 0.036,1),(0.768, 0.809, 0.039,1), (0.989, 0.862, 0.772,1),
            (1.0, 1.0, 1.0)]
palplot(colors)
```

```
kindlmann = LinearSegmentedColormap.from_list('kindlmann', colors)
```

or from colour names:

```
colors = ["white", "yellow", "red"]
palplot(colors)
```

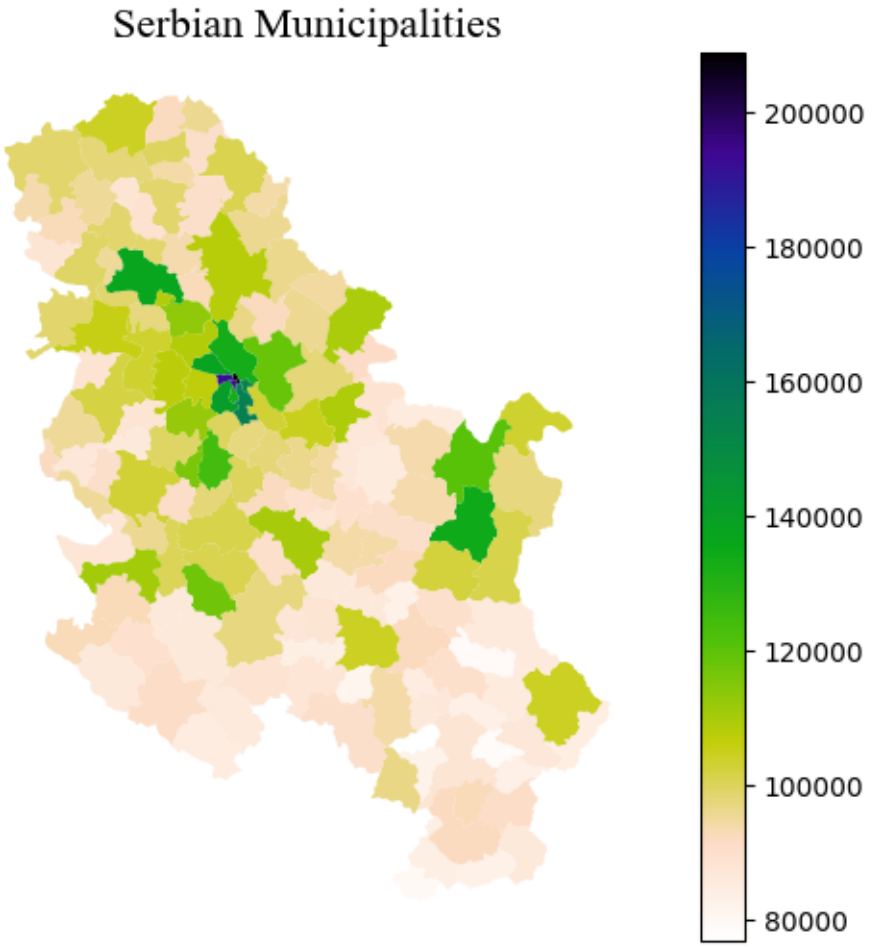


Let's try a new colormap and let's also set a number of classes to divide the data in, through the parameter `k`.

```
white_to_red = LinearSegmentedColormap.from_list("name", ["yellow","red"])
```

```
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
serbia_admin.plot(ax = ax, column = 'gross', linewidth = 0.3, cmap = kindlmann.reversed(), le
ax.set_axis_off()
title_parameters = {'fontsize':'16', 'fontname':'Times New Roman'}
ax.set_title("Serbian Municipalities", **title_parameters) #parameters as above
```

```
Text(0.5, 1.0, 'Serbian Municipalities')
```



With `GeoPandas`, when you use the `plot` method with `legend=True` the type of legend that appears depends on the data being visualized:

- Continuous Data: For columns with continuous data (like population estimates, temperatures, etc.), a colour bar is generated as the legend. This color bar represents a range of values with a gradient, indicating how data values correspond to colours on the map.
- Categorical Data: For columns with categorical data (like country names, types of land use, etc.), if you specify `legend=True`, `GeoPandas` will try to create a legend that categorizes these distinct values with different colours. However, creating legends for categorical data is not as straightforward as with continuous data and might require additional handling for a clear and informative legend (see below).

3.2.1.2 Scheme

It is important to keep in mind that choropleth maps strongly depend on the scheme that it is passed (or the default one) to classify the data in groups. The plot above only shows one municipality coloured in dark blue.

Look at the following plots and how three different classifiers produce different results for the same data.

Refer to https://geopandas.org/en/stable/gallery/choropleths.html and https://geographicdata.science/book/no for further details

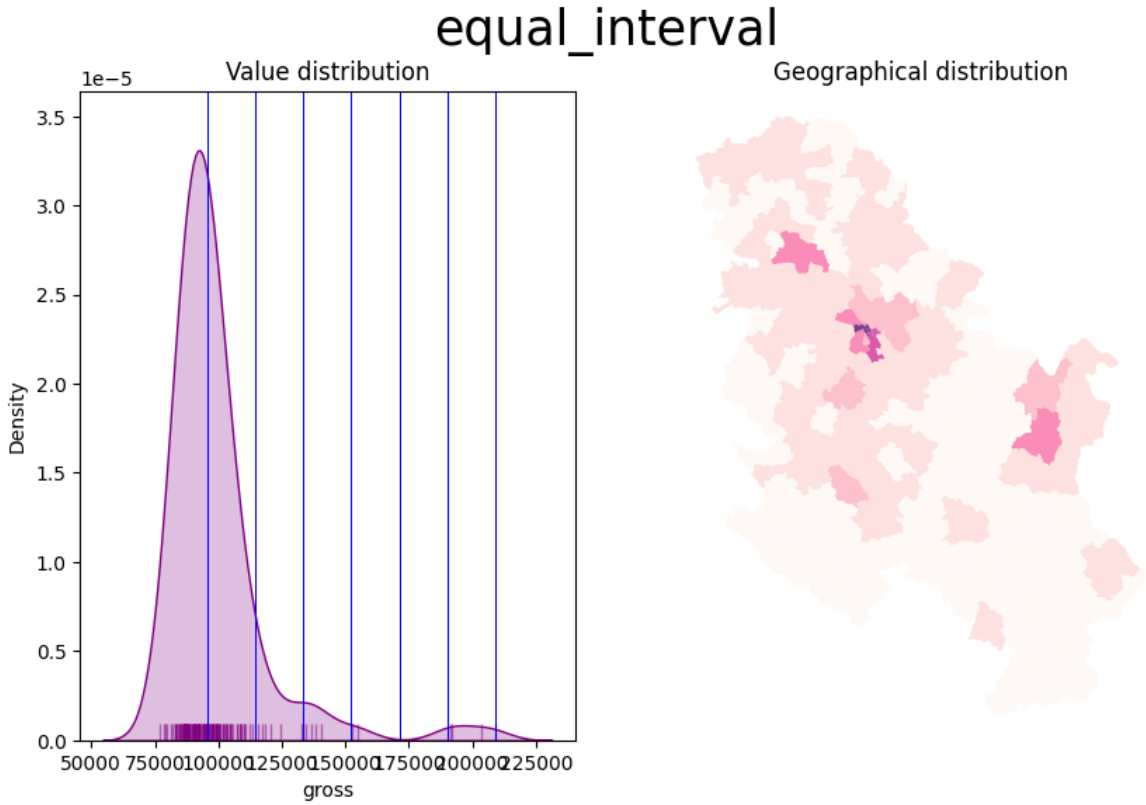
```
# Function for plotting the map and the distribution of the value in bins

from mapclassify import Quantiles, EqualInterval, FisherJenks

def plot_scheme(gdf, column, scheme, figsize=(10, 6)):
    '''
    Arguments
    ---------
    gdf: GeoDataFrame
        The GeoDataFrame to plot
    column: str
        Variable name
    scheme: str
        Name of the classification scheme to use
    figsize: Tuple
        [Optional. Default = (10, 6)] Size of the figure to be created.

    '''
    schemes = {'equal_interval': EqualInterval, 'quantiles': Quantiles, 'fisher_jenks': Fishe
    classification = schemes[scheme](gdf[column], k=7)
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=figsize)
    # KDE
```

```
    sns.kdeplot(gdf[column], fill=True, color='purple', ax=ax1)
    sns.rugplot(gdf[column], alpha=0.5, color='purple', ax=ax1)
    for cut in classification.bins:
        ax1.axvline(cut, color='blue', linewidth=0.75)
    ax1.set_title('Value distribution')
    # Map
    p = gdf.plot(column=column, scheme=scheme, alpha=0.75, k=7, cmap='RdPu', ax=ax2, linewid
    ax2.axis('equal')
    ax2.set_axis_off()
    ax2.set_title('Geographical distribution')
    fig.suptitle(scheme, size=25)
    plt.show()
```

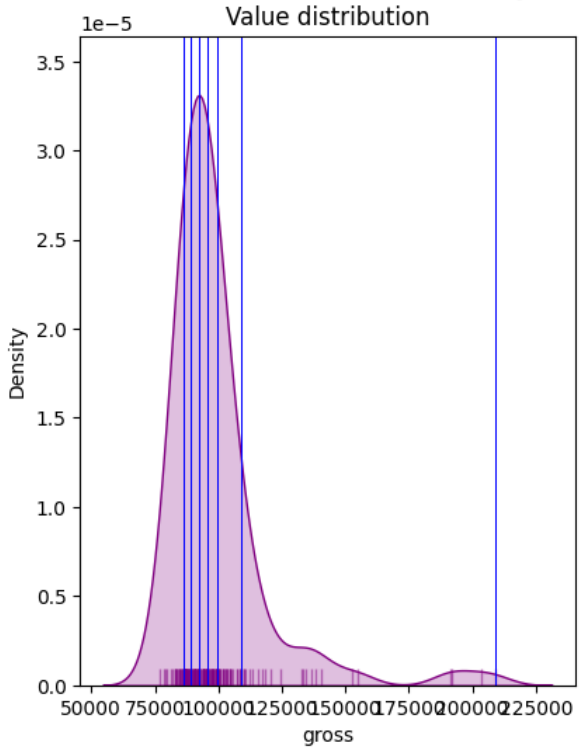
- The *Equal intervals* method splits the range of the distribution, the difference between the minimum and maximum value, into equally large segments and to assign a different colour to each of them according to a palette that reflects the fact that values are ordered.
- To obtain a more balanced classification, one can use the *Quantiles* scheme. This assigns the same amount of values to each bin: the entire series is laid out in order and break points are assigned in a way that leaves exactly the same amount of observations between each of them. This "observation-based" approach contrasts with the "value-based" method of equal intervals and, although it can obscure the magnitude of extreme values, it can be more informative in cases with skewed distributions.
- Amongst many other, the *Fisher Jenks* dynamically minimises the sum of the absolute deviations around class medians. The Fisher-Jenks algorithm is guaranteed to produce an optimal classification for a prespecified number of classes.

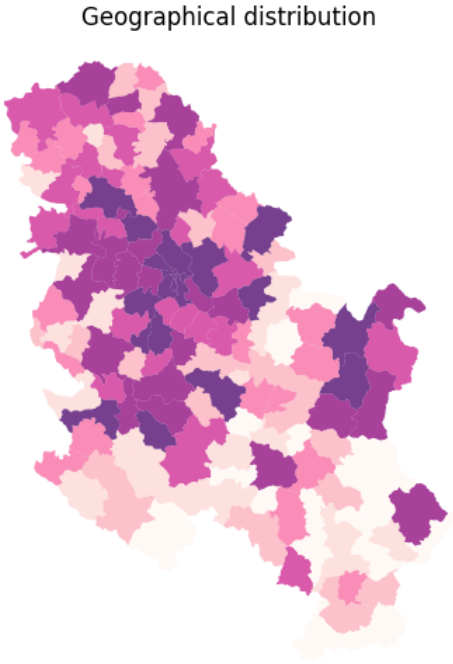
The only additional arguments to pass for producing a choropleth, therefore, are the actual variable we would like to classify and the number of segments we want to create, `k`. This is, in other words, the number of colours that will be plotted on the map so, although having several can give more detail, at some point the marginal value of an additional one is fairly limited, given the ability of the human brain to tell any differences.

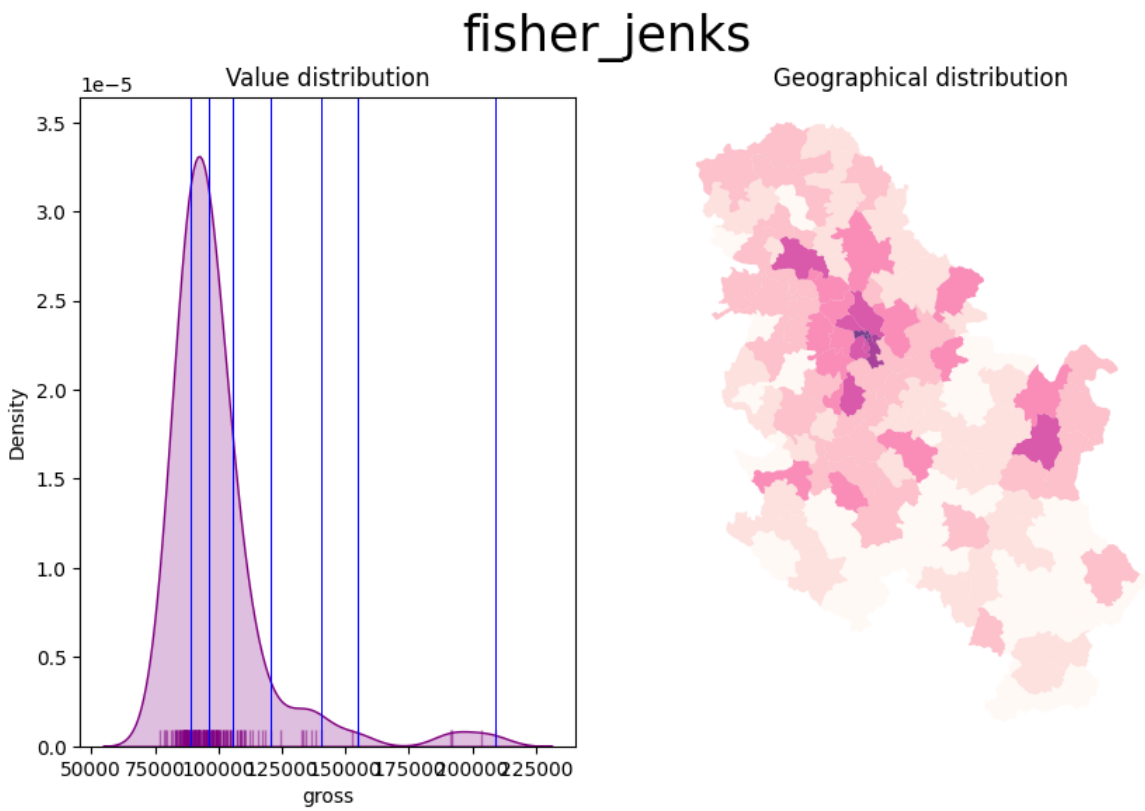
```
schemes = ['equal_interval', 'quantiles', 'fisher_jenks']
for scheme in schemes:
    plot_scheme(serbia_admin, 'gross', scheme)
```


equal_interval
Value distribution
Geographical distribution
1e−5
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Density
50000
75000
100000
125000
150000
175000
200000
225000
gross

quantiles


Value distribution
1e−5
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Density
50000 75000 100000 125000 150000 175000 200000 225000
gross


Geographical distribution



Also consider the Modifiable Areal Unit Problem and how the geographies of the administrative boundaries, in this case, may impact the visualisation.

For example, the most populated area is a municipality in the north that corresponds to the city of Novi Sad. Let's have a look at the data

```
serbia_admin[['name', 'pop', 'Province']].sort_values(by = 'pop', ascending = False).iloc[:1
```

	name	pop	Province
48	Novi Sad	341625.0	Južno-Bački
24	Novi Beograd	186667.0	Grad Beograd
19	Čukarica	154854.0	Grad Beograd
3	Kragujevac	154290.0	Šumadijski
26	Palilula	148292.0	Grad Beograd
34	Zemun	143173.0	Grad Beograd
32	Voždovac	137315.0	Grad Beograd
35	Zvezdara	130225.0	Grad Beograd
39	Leskovac	123201.0	Jablanički

	name	pop	Province
120	Subotica	121250.0	Severno-Bački

In our dataset, the city of Novi Sad is categorised as a municipality by itself, because the administrative boundaries file is not updated. In reality, "since 2002, when the new statute of the city of Novi Sad came into effect, Novi Sad is divided into two city municipalities, Petrovaradin and Novi Sad. From 1989 until 2002, the name Municipality of Novi Sad meant the whole territory of the present-day city of Novi Sad." (see: wikipedia).

On the contrary, Grad Beograd, that is Belgrade, is correctly split into different municipalities and its population, when visualised, is spread out across the different geometries of its municipalities. In other words, our map depends on the geometries of the areas and on how the data was collected. While it could be that these areas were indeed identified by population size in the first place, the point is that the fact that Novi Sad is not split into more areas, as Belgrade is, makes it stound out more clearly from the map (and to some extent a bit unfairly)

This may happen with different types of data, particularly with administrative boundaries and it is crucial to reflect on how Choropleth maps may be impacted. One can look for more granular data or consider to weight the continuous value with the extent of the area (i.e. obtaining density values).

3.2.1.3 An alternative to scheme: ColorMap Normalisation

The `mpl.colors.Normalize` function in `matplotlib` creates a normalization object, which adjusts data values into a range that is ideal for colour mapping in a colormap. This function is particularly beneficial in scenarios where precise control over the mapping of data values to colour representations is needed.

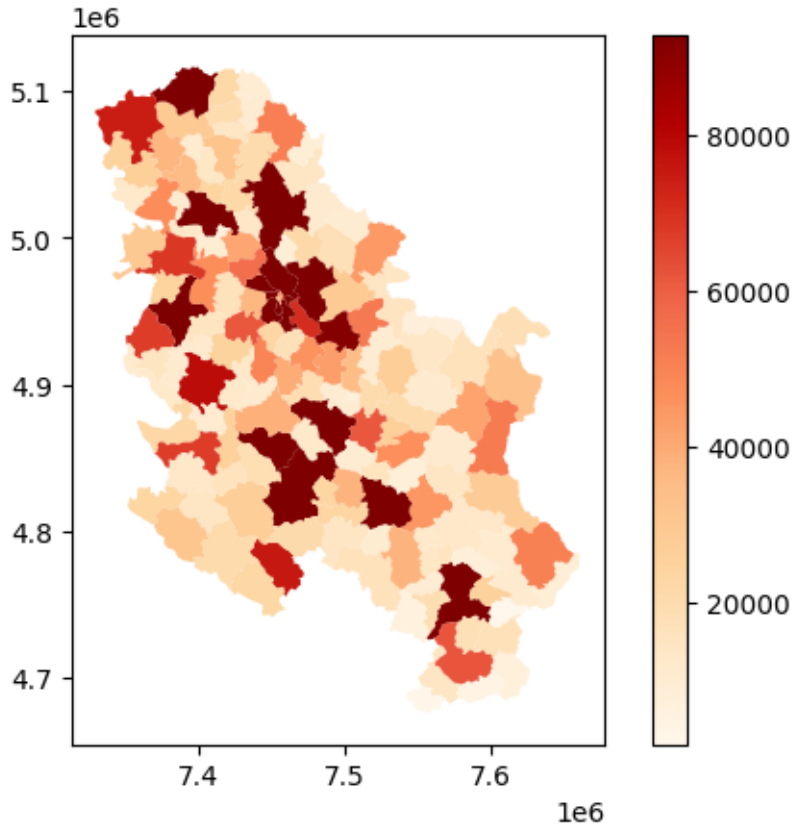
When employed in a plotting function, this normalization object ensures that the data values are scaled to fit a pre-defined range (for instance, `norm = mpl.colors.Normalize(vmin=0, vmax=40)`). Any values falling below 0 are mapped to the lowest colour on the colormap scale, while values exceeding 40 are mapped to the highest colour. This approach is especially useful when aiming to highlight differences within a specific data range; it can significantly enhance the visualization of data, by, for example, emphasizing temperature variations between 0°C and 40°C. This becomes crucial in instances where a few data points with high values (e.g., 50°C) might otherwise lead to a less informative visualization if not 'normalized' and treated as if they corresponded to 40° C values.

For our dataset, we can use as `vmax` the value corresponding to the 90th percentile.

```
serbia_admin['pop'].quantile(0.90)
```

```
93014.0
```

```
import matplotlib as mpl
fig, ax = plt.subplots(1, 1)
vmin = serbia_admin['pop'].min()
vmax = serbia_admin['pop'].quantile(0.90) #
norm = mpl.colors.Normalize(vmin=vmin, vmax=vmax)
serbia_admin.plot(ax = ax, column='pop', cmap='OrRd', legend=True, norm = norm)
```

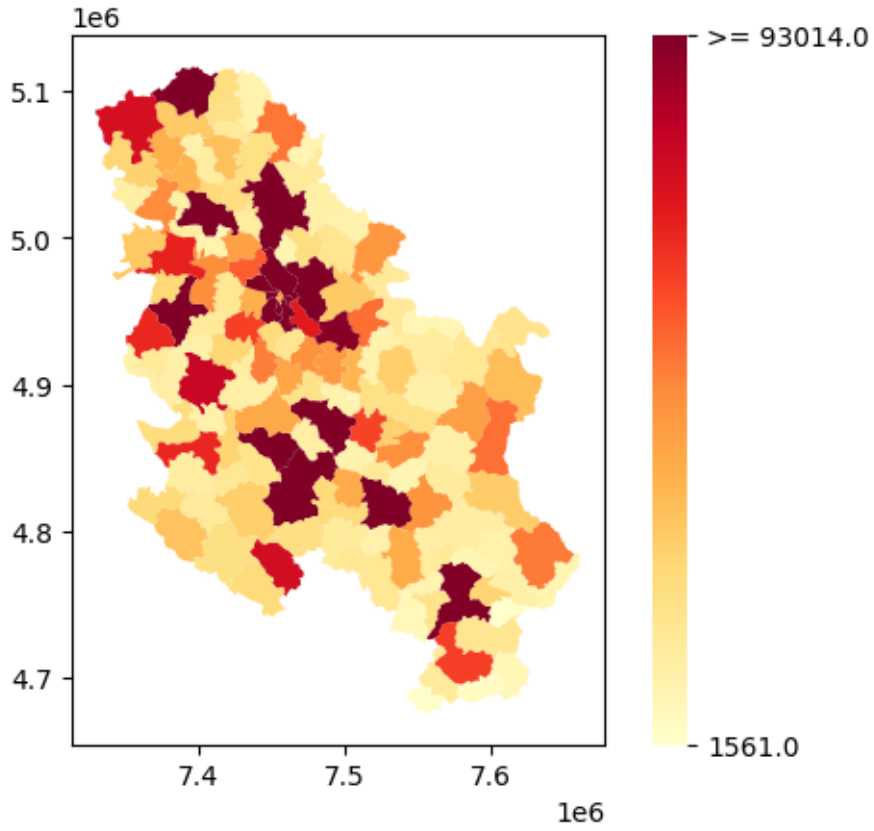


Important:

When passing `norm` in the `plot` method, do not pass the arguments to the `scheme` parameter. For continuous variables, `norm` maps each value directly to a color, making discrete categorization redundant. In other words, it allows for a direct mapping of data values to the color map, eliminating the need for intermediary classification schemes. `norm` ensures a smooth gradient in the color map without artificially segmenting the data.

3.2.1.4 Customising the colorbar

```
import matplotlib.cm as cm

fig, ax = plt.subplots(1, 1)
cmap = 'YlOrRd'
# we leave the legend out
serbia_admin.plot(column='pop', cmap=cmap, norm = norm, ax=ax)

# we add the colorbar separately passing the norm and cmap
cbar = fig.colorbar(cm.ScalarMappable(norm=norm, cmap=cmap), ax = ax)
cbar.outline.set_visible(False)

# updating ticks VALUES
ticks = [norm.vmin, norm.vmax]
cbar.set_ticks(ticks = ticks)

# updating ticks LABELS
cbar.ax.set_yticklabels([round(t,1) for t in ticks])
cbar.ax.set_yticklabels([round(t,1) if t < norm.vmax else ">= "+str(round(t,1)) for t in cba
```



Above, we removed the outline of the color bar. Then we set the tick values to the min and the max population values, based on our norm object. Then, for the vmax value's label we added a ">=" to remind us that other, higher values are displayed with the darkest color.

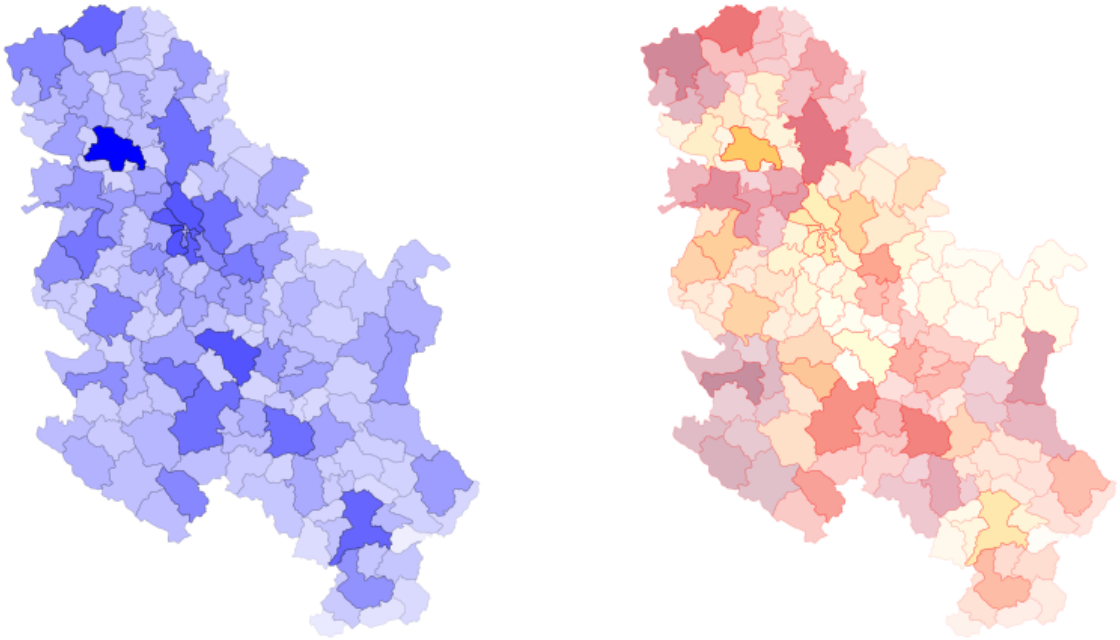
3.2.1.5 Varying alpha transparency based on an array

Finally, we can also convey variation in a continuous scale through transparency. `alpha` doesn't expect column names, so we cannot just pass the name of the column containing the variable. Instead, we have to create an array from 0.0 to 1.0 values. To so we can a) use normalisation methods, or b) rescale the original values within 0 to 1 based on the original min and max values.

For example, with square root normalization:

```
# 1. Create an alpha array based on a normalized value (e.g., population)
import numpy as np
pop_max = serbia_admin['pop'].max()
alpha = np.sqrt(serbia_admin['pop'] / pop_max)
```

```
# Plot with varying alpha values
fig, axes = plt.subplots(1, 2, figsize=(10, 6))
serbia_admin.plot(color = 'blue', ax=axes[0], alpha=alpha, edgecolor='black', linewidth = 0.3
serbia_admin.plot(cmap = 'YlOrRd', ax=axes[1], alpha=alpha, edgecolor='red', linewidth = 0.3)
for ax in axes:
    ax.set_axis_off()
```



Important:

`matplotlib` would not able to plot a color bar from variations in the alpha value since no column is passed directly. We would need, in this case, to build a color bar manually as demonstrated above.

3.2.2 Choropleth Maps for Categorical Variables

A choropleth for categorical variables assigns a different color to every potential value in the series based on certain colormaps (`cmap`). We don't need to specify a scheme in this case, but just to the categorical `column`. Using last's week GeoDataFrame, we can plot terrorist attacks in Germany, for example, by group.

```
gdf = gpd.read_file("../data/germany.shp").to_crs(germany_crs)
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
gdf.plot(ax = ax, column = 'gname', legend = True)
ax.set_axis_off()
title_parameters = {'fontsize':'16', 'fontname':'Times New Roman'}
ax.set_title("Terrorist Attacks in Germany, by Group", **title_parameters) #parameters as ab
```

```
Text(0.5, 1.0, 'Terrorist Attacks in Germany, by Group')
```

Terrorist Attacks in Germany, by Group

The map above is what you would get from datasets that are not cleaned/manipulated directly or when there are too many categories in the selected column. First, let's get a slimmer slice of the gdf that only contains attacks that cause a number of fatalities and wounded higher than 10.

```
condition = (gdf.nkill + gdf.nwound) > 10
gdf_filtered = gdf[condition].copy()
```

Then, let's build a function that creates a random color map based on the number of categories. This creates random `HUE`-based colors:

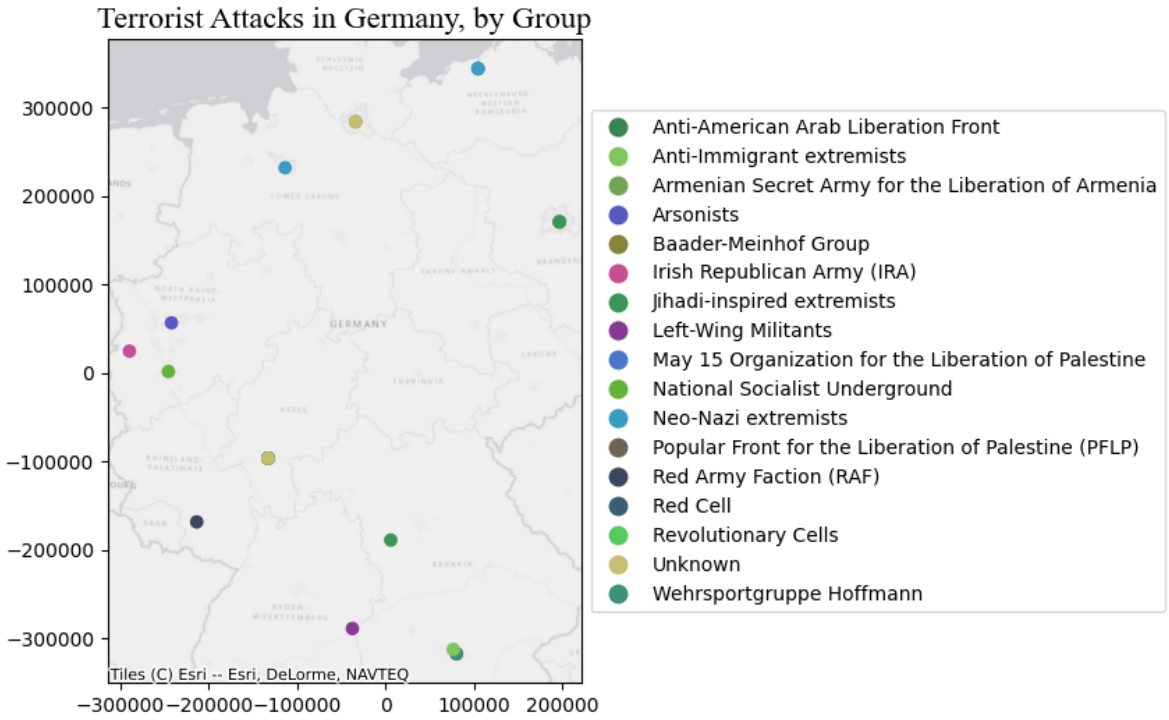
```
# Generate random colormap
def rand_cmap(nlabels):
    """
    It generates a categorical random color map, given the number of classes

    Parameters
    ----------
    nlabels: int
        The number of categories to be coloured.
    type_color: str {"soft", "bright"}
        It defines whether using bright or soft pastel colors, by limiting the RGB spectrum.

    Returns
    -------
    cmap: matplotlib.colors.LinearSegmentedColormap
        The color map.
    """
    # Generate color map for bright colors, based on hsv
    randHSVcolors = [(np.random.uniform(low=0.20, high=0.80),
                          np.random.uniform(low=0.20, high=0.80),
                          np.random.uniform(low=0.20, high= 0.80)) for i in range(nlabels)]

    random_colormap = LinearSegmentedColormap.from_list('new_map', randHSVcolors, N=nlabels)

    return random_colormap
```

```
cmap = rand_cmap(len(gdf_filtered.gname.unique()))
cmap
```

We also place the legend on the centre left. This is done automatically, but the legend and its items can be manipulated directly. Legends in `matplotlib` are extremely complex to personalise. However, do have a look at https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.legend.html#ma for both automatic and explicit manipulation.

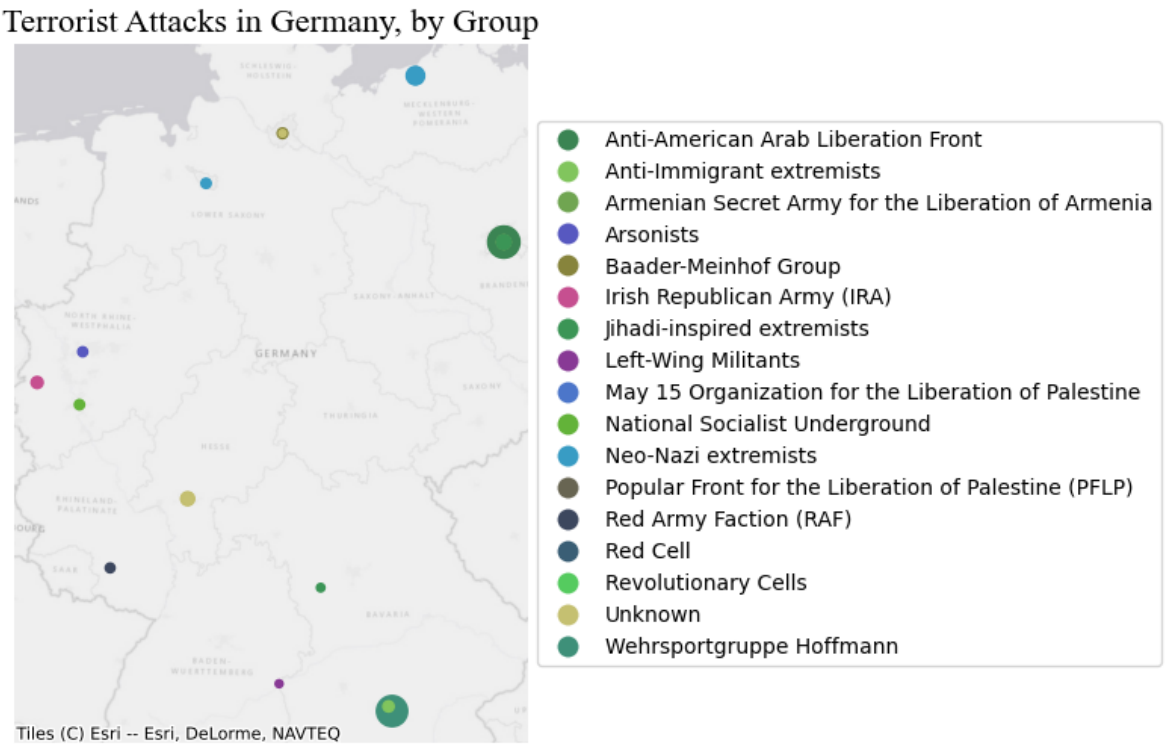
```
legend_kwds={"loc": "center left", "bbox_to_anchor": (1, 0.5)}
```

```
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
gdf_filtered.plot(ax = ax, column = 'gname', legend = True, cmap = cmap, legend_kwds = legend
title_parameters = {'fontsize':'16', 'fontname':'Times New Roman'}
ax.set_title("Terrorist Attacks in Germany, by Group", **title_parameters) #parameters as ab
ctx.add_basemap(ax, crs= gdf_filtered.crs.to_string(), source = ctx.providers.Esri.WorldGray
```



We can also convey the impact of the events through the `markersize`. This introduces the concept of *cartogram* (see below).

```
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
gdf_filtered.plot(ax = ax, column = 'gname', markersize = 'nwound', legend = True, cmap = cma
ax.set_title("Terrorist Attacks in Germany, by Group", **title_parameters) #parameters as abo
ctx.add_basemap(ax, crs= gdf_filtered.crs.to_string(), source = ctx.providers.Esri.WorldGray(
ax.set_axis_off()
```



3.3 Part III: Cartograms - Manipulating the Geometry size for showing the magnitude of a value

Cartograms are maps that represent the spatial distribution of a variable not by encoding it in a color palette but rather by modifying geographical objects. There are many algorithms to distort the shapes of geographical entities according to values, some of them are rather complex.

3.3.1 Polygons

You can obtain cartograms for `Polygon` with `geoplot`: see https://residentmario.github.io/geoplot/

`geoplot` functions pretty much work as `plot`

```
# this library needs the GeoDataFrame to be reverted to WGS
ax = gplt.cartogram(serbia_admin.to_crs(wgs), scale='pop', projection=gcrs.Mercator(), color
# see for projections that work with gplt https://scitools.org.uk/cartopy/docs/v0.15/crs/proj
gplt.polyplot(serbia_admin.to_crs(wgs), facecolor='lightgray', edgecolor='white', ax=ax, lw =
```

3.3.2 Points

For `Point` GeoDataFrames we can just go back to `plot` and pass a column name to `markersize`.

```
attacks = pd.read_csv("../data/GTD_2022.csv", low_memory = False)
country = 'Germany'
```

```
df = attacks[attacks.country_txt == country].copy()
wgs = 'EPSG:4326'
germany_crs = 'EPSG:4839'
gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df.longitude, df.latitude), crs = wgs)
gdf = gdf.to_crs(germany_crs)
```

```
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
gdf.plot(ax = ax, markersize = 'nwound', color = 'purple', legend = True)
ax.set_axis_off()
```

One can also convert polygons into points by using their centroids, and then define the size of the dot proportionally to the value of the variable we want to display.

3.3.3 LineString

For `LineString` we pass the column name to `linewidth`.

Let's load a shapefile of lines. These lines represent frequency of train connections from/to train stations in the region of Liguria (Italy) to other stations within or outside the region. Each line refers to a connection between two specific stations, through a certain type of service and contains information about the frequency of that type of service. For example, the cities of Savona and Finale Ligure might be connected by 5 InterCity trains and 50 regional services. These services correspond to 2 different records.

```
trains_freq = gpd.read_file("../data/trains_liguria.shp" )
trains_freq.crs
```

```
<Projected CRS: EPSG:3003>
Name: Monte Mario / Italy zone 1
Axis Info [cartesian]:
- X[east]: Easting (metre)
- Y[north]: Northing (metre)
Area of Use:
- name: Italy - onshore and offshore - west of 12°E.
- bounds: (5.93, 36.53, 12.0, 47.04)
Coordinate Operation:
- name: Italy zone 1
- method: Transverse Mercator
Datum: Monte Mario
- Ellipsoid: International 1924
- Prime Meridian: Greenwich
```

Let's check the type of services contained here.

```
trains_freq['train_type'].unique()
```

```
array(['REG', 'IC', 'FB', 'ICN', 'U', 'EC/EN', 'AV'], dtype=object)
```

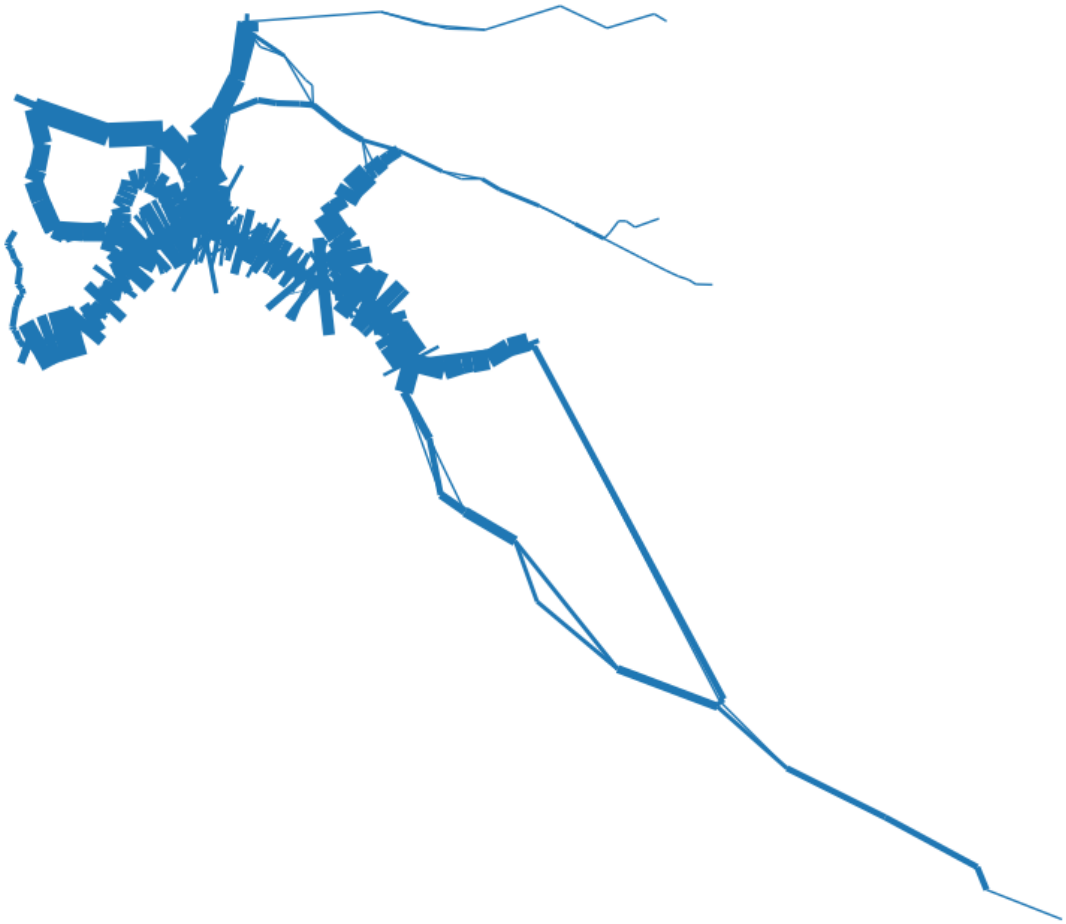
We have: - 'REG': regional trains. - 'IC': intercity trains. - 'FB': similar to IC, but slightly faster. - 'ICN': sleeper trains. - 'U': urban trains (Genoa). - 'EC/EN': international trains. - 'AV': High-speed trains.

Let's keep just regional, intercity, and high-speed trains.

```
to_keep = ['REG', 'IC', 'AV']
trains_freq = trains_freq[trains_freq['train_type'].isin(to_keep)]
```

The usage of `linewdith` is a bit different from `markersize` for some reason. We have to pass an `array` of N values, where N is equal to the GeoDataFrame size. In other words, we have to pass the column we want to use to regulate the line width directly as a list/array. Specifying the column name is not enough.

```
fig, ax = plt.subplots(1, 1, figsize=(10, 15))
trains_freq.plot(ax = ax, linewidth = trains_freq['freq'])
ax.set_axis_off()
```



As you can see, the default arguments and simply passing the column values do not produce pretty results. The first thing to look at is the values that are passed to `linewidth`. In some cases, the min and max values, as well as their distribution, are not ideal for visually conveying the magnitude of the variable attached to the geometry. One option is to use a multiplier factor (see below), or to rescale the values from 0 to 1, for example, and then, again, if necessary use a multiplier.

```
fig, ax = plt.subplots(1, 1, figsize=(15, 20))
lw = trains_freq['freq'] * 0.15
trains_freq.plot(ax = ax, linewidth = lw, capstyle = 'round', joinstyle = 'round', column =
ctx.add_basemap(ax, crs= trains_freq.crs.to_string(), source = ctx.providers.Esri.WorldGrayCa
ax.set_axis_off()
```

While this looks a bit better, this visualisation is not ideal because the frequencies are not snapped to the actual railway network. The lines represent, instead, connection between train stops and therefore their coordinates only include the ones corresponding to the stations where the different services call at. One can devise approaches to:

- Assigning the frequencies, or any other value, to the corresponding infrastructure's section. For example, the railway section between two stations could be associated with a value representing the total number of regional/local services travelling along it.
- Smoothing the lines representing the services by adding further coordinates along the line.

Both these processes go beyond the scopes of this lab and require several considerations depending on the data, the scale, and what information one wants to displays.

Exercise:

Today we've seen how to exploit `matplotlib` to plot `GeoDataFrame` layers. Go through the notebook again if you feel that there's something you need to review. You are not expected to remember each step/method/parameter. Rather, this notebook should be used as a reference for producing maps in Python. Do keep in mind that most of the maps above have been produced with just a bunch of rows, so each of them can be improved and embellished with some more effort.

Now, if you are not overwhelmed, have a look at the very last map and produce some nice visualisation using the same data. You can further improve its clarity, add a legend that refers to the line width, visualise only a certain type of services, or add information/context, for example. In the folder `\data` you can also find a `.shp` file containing all the train stations in Italy, should you need that.

3.3.3.1 Saving figures (check here for details)

```
fig.savefig("fig1.pdf", dpi='figure', format="pdf", bbox_inches = 'tight')
```

4 Web Architectures and APIs

The **Lecture slides** can be found here.

This **lab**'s notebook can be downloaded from here.

4.1 What do APIs actually do?

In this lab, we will unpack how Application Programming Interfaces ("APIs") work and we cover the basics of accessing an API using Python. Instead of downloading a data set, APIs allow programmers, statisticians (or students) to request data directly from a server to a local machine. When you work with web APIs, two different computers — a client and server — will interact with each other to request and provide data, respectively.

4.1.1 RESTful Web APIs are all around you.

Web APIs

- Allow you query a remote database over the internet
- Take on a variety of formats
- Adhere to a particular style known as Representation State Transfer or REST (in most cases)
- RESTful APIs are convenient because we use them to query database using URLs

Consider a simple Google search:

Ever wonder what all that extra stuff in the address bar was all about? In this case, the full address is Google's way of sending a query to its databases requesting information related to the search term *liverpool top attractions*.

In fact, it looks like Google makes its query by taking the search terms, separating each of them with a +, and appending them to the link https://www.google.com/#q=. Therefore, we should be able to actually change our Google search by adding some terms to the URL:

Learning how to use RESTful APIs is all about learning how to format these URLs so that you can get the response you want.

4.1.2 Group activity

Get into groups of 5 or 6 students. Using your friend the internet, look up answers to the following questions. Each group will be assigned one question and asked to present their findings in 5 min to discuss with the entire class.

1. What is a `URL` and how can it help us query data? What is a response status and what are the possible categories?
2. What is a `GET` request? How does a `GET` request work?
3. What are API keys and how do you obtain them? What kinds of restrictions to they impose on users? Find an example of an API key, what does it look like?
4. (For 2 groups) More and more APIs pop up every day. Do a bit of quick research and find 2 different examples of APIs that you would be interested in using. 2 groups, 2 or 3 APIs each.

There are two ways to collect data through APIs in Python:

- **Plug-n-play packages.** Many common APIs are available through user-written Python (or R) Packages. These packages offer functions that conveniently "wrap" API queries and format the response. These packages are usually much more convenient than writing our own query, so it is worth searching for a package that works with the API we need.
- **Writing our own API request.** If no wrapper function is available, web have to write our own API request and format the response ourselves using Python. This is tricky, but definitely doable.

4.2 API Python libraries

Exercise: `census` pair activity:

Some Python packages "wrap" API queries and format the response. Lucky us! In pairs, let's have a look at `census`, a wrapper or the United States Census Bureau's API You can also have a look at the different APIs available from the United States Census Bureau.

```
import pandas as pd
from census import Census
```

To get started working, set you Census API key. A key can be obtained from http://api.census.gov/data/key_signup.html.

```
census_api_key = "" # Replace 'YOUR_CENSUS_API_KEY_HERE' with your actual Census API key.
```

```
# Set API key
c = Census(census_api_key)
```

- Variables in tidycensus are identified by their Census ID, e.g. B19013_001.
- Entire tables of variables can be requested with the table argument, e.g. table = 'B19001'.
- Users can request multiple variables at a time, and set custom names with a named vector.

In Python we can use the library `census` to access the American Community Survey (ACS) 5-Year Data (2016-2020)

Exercise:

In pairs explore some of the different variables available in the 5-Year ACS (2016-2020). Make a note of 3 variables you would be interested in exploring. The ACS2 variablespage might help.

Let's explore income data for example.

```
# Retrieve income data by state using the ACS table B19001
# Note: The variable 'B19001_001E' represents "Estimate!!Total" in table B19001
income_data = c.acs5.state(('NAME', 'B19001_001E'), Census.ALL, year=2020)
income_data[:10] # just first ten "rows"
```

```
[{'NAME': 'Pennsylvania', 'B19001_001E': 5106601.0, 'state': '42'},
 {'NAME': 'California', 'B19001_001E': 13103114.0, 'state': '06'},
 {'NAME': 'West Virginia', 'B19001_001E': 734235.0, 'state': '54'},
 {'NAME': 'Utah', 'B19001_001E': 1003345.0, 'state': '49'},
 {'NAME': 'New York', 'B19001_001E': 7417224.0, 'state': '36'},
 {'NAME': 'District of Columbia', 'B19001_001E': 288307.0, 'state': '11'},
 {'NAME': 'Alaska', 'B19001_001E': 255173.0, 'state': '02'},
 {'NAME': 'Florida', 'B19001_001E': 7931313.0, 'state': '12'},
 {'NAME': 'South Carolina', 'B19001_001E': 1961481.0, 'state': '45'},
 {'NAME': 'North Dakota', 'B19001_001E': 320873.0, 'state': '38'}]
```

```
income_df = pd.DataFrame(income_data)
income_df.head()
```

	NAME	B19001_001E	state
0	Pennsylvania	5106601.0	42
1	California	13103114.0	06
2	West Virginia	734235.0	54

	NAME	B19001_001E	state
3	Utah	1003345.0	49
4	New York	7417224.0	36

Exercise:

Discuss the format of the data obtained with your partner and then use the function `acs5.state` to explore the 3 variables you discussed in the previous exercise.

You can also get more variables by passing the names in a `tuple`. The code below, for example, fetches all the columns from the B19001 table, which include various income brackets. The tuple in the request generates a list of column names (like 'B19001_001E', 'B19001_002E', ..., 'B19001_017E').

```
variables = tuple(f'B19001_{str(i).zfill(3)}E' for i in range(1, 18))
```

4.2.0.1 This is what the line above does:

- `range(1, 18)`: This part generates a sequence of numbers from 1 to 17. In Python, range(start, stop) generates numbers from start up to but not including stop.
- `for i in range(1, 18)`: This is a loop within a comprehension that iterates over each number in the range from 1 to 17.
- `str(i).zfill(3)`: For each number `i`, this converts `i` to a string. Then, `.zfill(3)` pads the string with zeros to make it 3 characters long. For example, if `i` is 2, `str(i).zfill(3)` becomes '002'.
- `f'B19001_{str(i).zfill(3)}E'`: This is an `f-string`, a way to format strings in Python. It inserts the zero-padded string into a larger string. So, for i = 2, you would get 'B19001_002E'.
- Finally, the comprehension is wrapped in `tuple(...)`, which converts the entire series of strings into a tuple.

```
wide_data = c.acs5.state(('NAME',) + variables, Census.ALL, year=2020)
# Convert to a Pandas DataFrame
wide_df = pd.DataFrame(wide_data)
wide_df.head()
```

	NAME	B19001_001E	B19001_002E	B19001_003E	B19001_004E	B19001_005E	B19001_
0	Pennsylvania	5106601.0	296733.0	206216.0	223380.0	227639.0	226678.0
1	California	13103114.0	614887.0	507398.0	435382.0	474093.0	454373.0
2	West Virginia	734235.0	62341.0	43003.0	45613.0	44635.0	40805.0

	NAME	B19001_001E	B19001_002E	B19001_003E	B19001_004E	B19001_005E	B19001_
3	Utah	1003345.0	36211.0	27395.0	28460.0	32497.0	36116.0
4	New York	7417224.0	471680.0	340614.0	303901.0	298025.0	276764.0

Let's make our query a bit more precise. We are going to query data on median household income and median age by county in the state of Louisiana from the 2016-2020 ACS. We can use the library `us` to get the code of each of the state by passing their name.

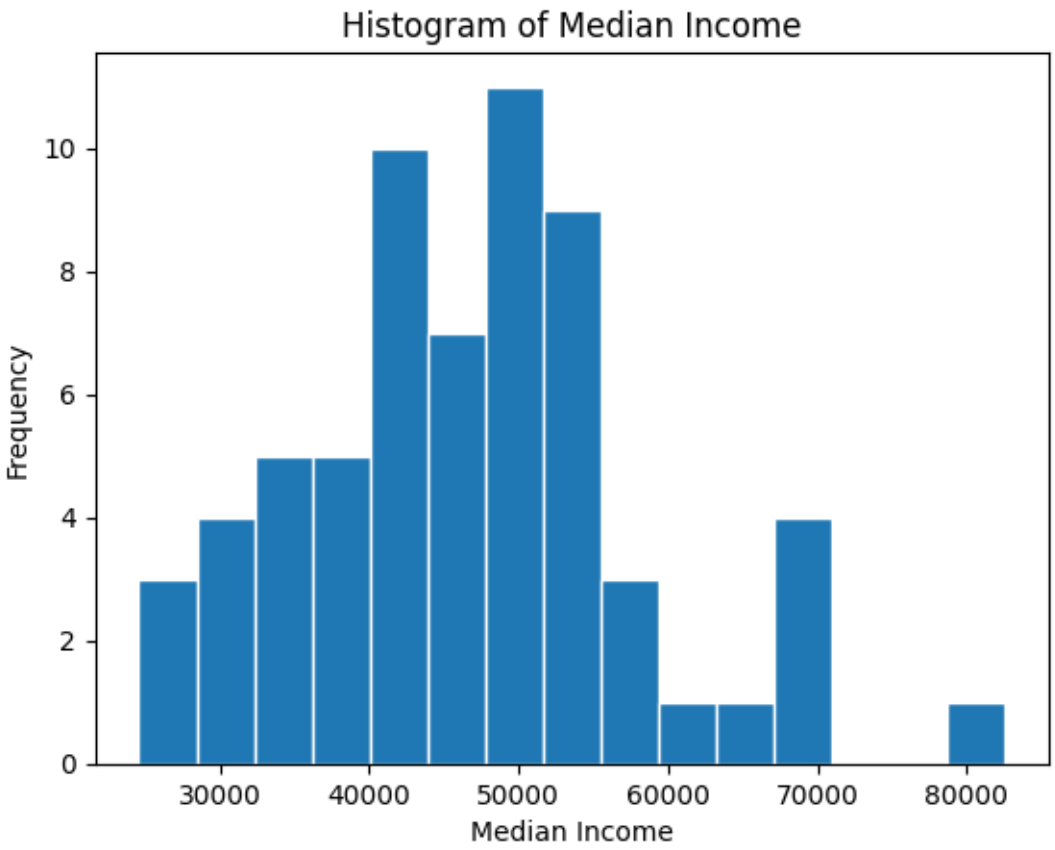
```
import us
# Get the state object for Louisiana
state_obj = us.states.lookup('Louisiana')
# Get the FIPS code (state code)
code = state_obj.fips
```

```
# Retrieve wide format data for median income and median age by county in Louisiana for the y
louisiana_data = c.acs5.state_county(('B19013_001E', 'B01002_001E'), code, Census.ALL, year=2
# Convert to a Pandas DataFrame
louisiana_df = pd.DataFrame(louisiana_data)
# Renaming the columns for clarity
louisiana_df.rename(columns={'B19013_001E': 'median_income', 'B01002_001E': 'median_age'}, in
louisiana_df.head()  # Display the first few rows
```

	median_income	median_age	state	county
0	82594.0	36.0	22	005
1	49256.0	37.5	22	011
2	42003.0	37.8	22	017
3	56902.0	44.9	22	023
4	36294.0	37.4	22	029

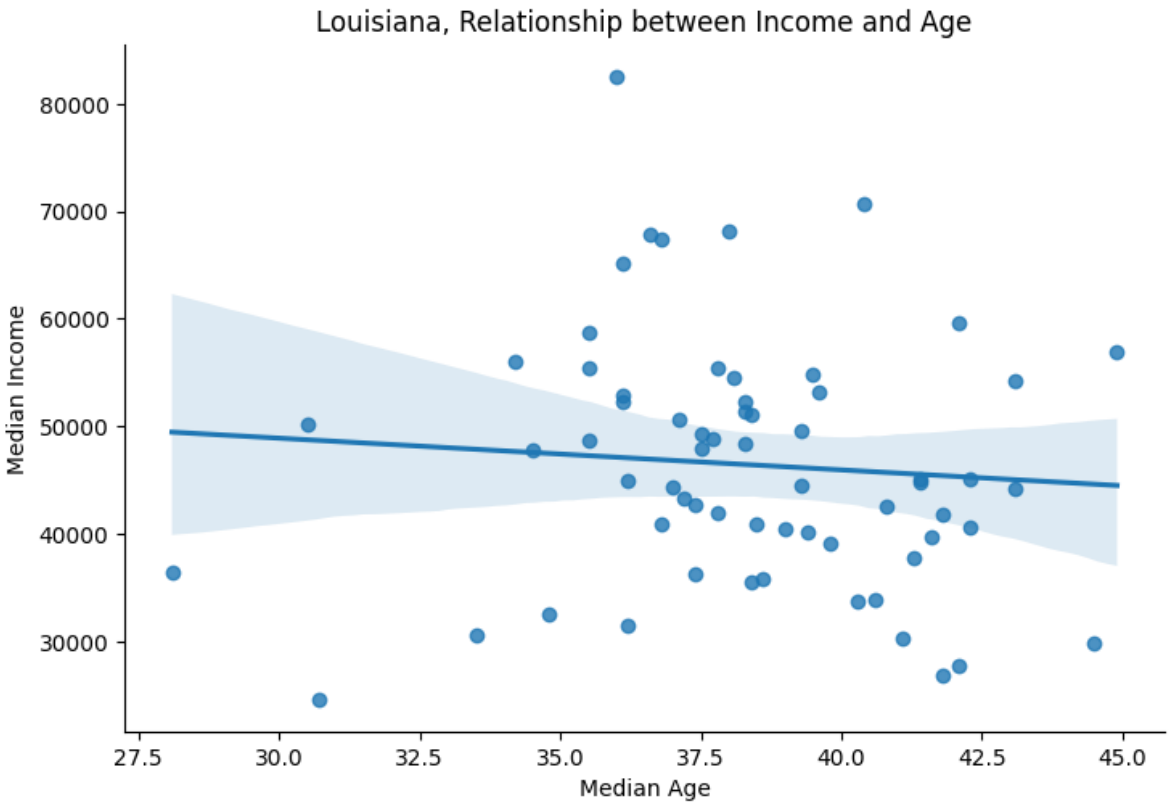
Let's plot one of our variables.

```
import matplotlib.pyplot as plt

# Plotting histogram for median income with 15 bins
plt.hist(louisiana_df['median_income'], bins=15, edgecolor='white')
plt.xlabel('Median Income')
plt.ylabel('Frequency')
plt.title('Histogram of Median Income')
plt.show()
```



We can also explore correlations between variables. Let's use `seaborn`.

```
import seaborn as sns

# Creating a scatter plot with a linear regression line
sns.lmplot(x='median_age', y='median_income', data=louisiana_df, aspect=1.5)
plt.xlabel('Median Age')
plt.ylabel('Median Income')
plt.title('Louisiana, Relationship between Income and Age')
plt.show()
```



Exercise:

In pairs, modify the state, variables and year parameters following the approach adopted above and produce some other simple scatter plots (cloud of points) that suggest correlations between your variables of interest.

4.3 Your own API request demo

The Python libraries commonly used for API requests are `requests` and `json`.

JSON stands for JavaScript Object Notation. Despite its association with JavaScript, JSON is widely used due to its readability and ease of use by computers, making it the primary format for data transmission through APIs. Most APIs return their responses in JSON format. The `json` library in Python allows you to parse and convert these JSON responses into Python data structures. JSON structures are composed of key-value pairs, similar to Python dictionaries.

In Python, to make an API request and handle the response, you would typically use the `requests` library. A standard API request involves sending a `GET` request to the server's URL, which specifies the data you wish to retrieve. For instance, to request the locations of

all the hire bike stations in London from the **Transport for London** API, you would use the `requests.get()` method. This method requires a URL that directs the request to the appropriate server.

```
import requests
import json

response = requests.get("https://api.tfl.gov.uk/BikePoint/")
response
```

```
<Response [200]>
```

Good! The response code 200 indicates a successful request

Most GET request URLs for API querying have three or four components:

1. Authentication Key/Token: A user-specific character string appended to a base URL telling the server who is making the query; allows servers to efficiently manage database access.
2. Base URL: A link stub that will be at the beginning of all calls to a given API; points the server to the location of an entire database.
3. Search Parameters: A character string appended to a base URL that tells the server what to extract from the database; basically, a series of filters used to point to specific parts of a database.
4. Response Format: A character string indicating how the response should be formatted; usually one of .csv, .json, or .xml.

```
# Making the API request
response = requests.get("https://api.tfl.gov.uk/BikePoint/")

# Parsing the response content as JSON and converting it to a DataFrame
bike_stations = pd.DataFrame(response.json())

# Printing the column names
print(bike_stations.columns)
```

```
Index(['$type', 'id', 'url', 'commonName', 'placeType', 'additionalProperties',
       'children', 'childrenUrls', 'lat', 'lon'],
      dtype='object')
```

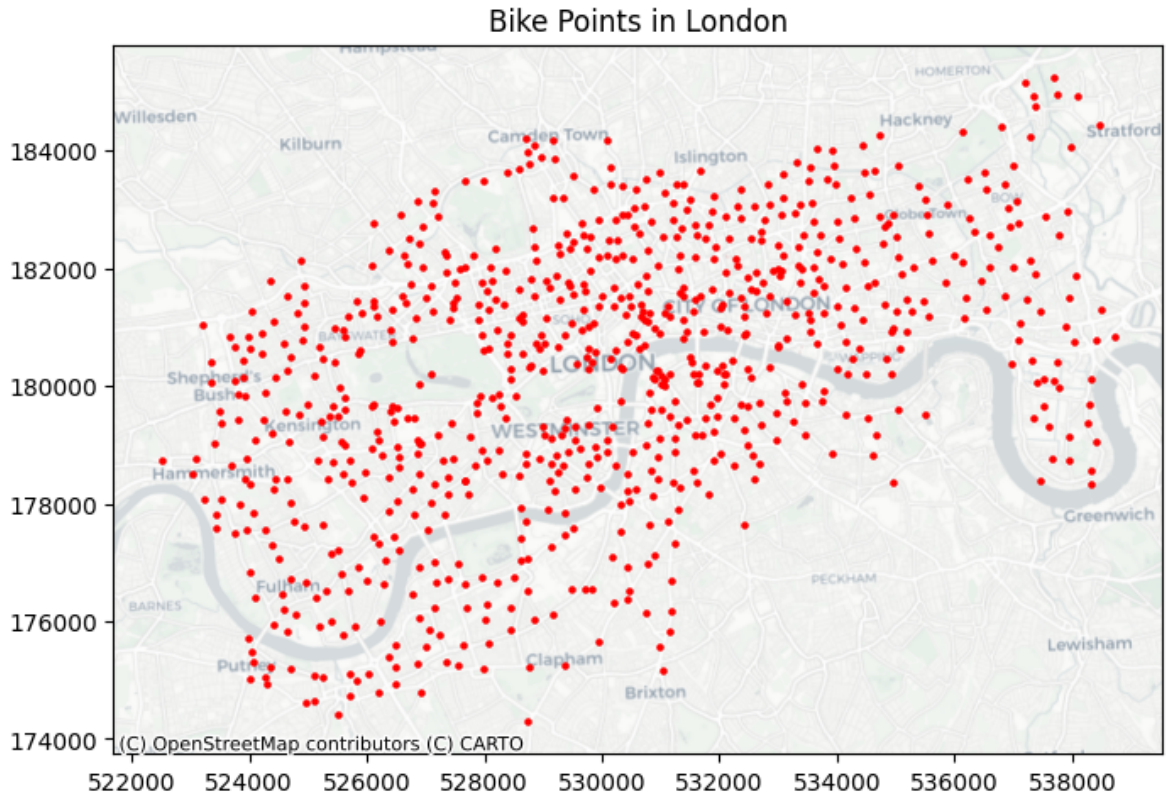
```
# Creating a new column 'Station ID' by extracting the numeric part from the 'id' column
bike_stations['Station ID'] = bike_stations['id'].str.extract(r'BikePoints_(\d+)', expand=Fa
bike_stations.head()
```

	$type	id	url	commonName
0	Tfl.Api.Presentation.Entities.Place, Tfl.Api.P...	BikePoints_1	/Place/BikePoints_1	River Street , Cle
1	Tfl.Api.Presentation.Entities.Place, Tfl.Api.P...	BikePoints_2	/Place/BikePoints_2	Phillimore Garde
2	Tfl.Api.Presentation.Entities.Place, Tfl.Api.P...	BikePoints_3	/Place/BikePoints_3	Christopher Stree
3	Tfl.Api.Presentation.Entities.Place, Tfl.Api.P...	BikePoints_4	/Place/BikePoints_4	St. Chad's Street
4	Tfl.Api.Presentation.Entities.Place, Tfl.Api.P...	BikePoints_5	/Place/BikePoints_5	Sedding Street, S

By now you should be able to visualise the data that you have obtained through this API.

```
%matplotlib inline
import geopandas as gpd
import contextily as ctx

# Convert DataFrame to GeoDataFrame
gdf = gpd.GeoDataFrame(bike_stations, geometry=gpd.points_from_xy(bike_stations.lon, bike_sta
# Setting the Coordinate Reference System (CRS) to WGS84 (EPSG:4326)
gdf.set_crs(epsg=4326, inplace=True)
# project to British Grid
gdf.to_crs(epsg= 27700, inplace = True)

# Plotting bikepoints on a map
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
gdf.plot(ax = ax, color='red', markersize=5)
source = ctx.providers.CartoDB.Positron
ctx.add_basemap(ax, crs= gdf.crs.to_string(), source= source)
ax.set_title('Bike Points in London')
```

```
Text(0.5, 1.0, 'Bike Points in London')
```



4.4 Geocoding API

Below is a short exploration of a the `geocoder` provided by the library `geopy`. The `geocoder` relies on various external geocoding services. When using `Nominatim`, it accesses data from OpenStreetMap. OpenStreetMap's data includes a vast array of geographical information sourced from contributors globally. This includes street addresses, points of interest, and other location-based data. When you provide an address to the Nominatim geocoder, it queries OpenStreetMap's databases to find the corresponding geographical coordinates.

In your own time, try to use it to automatically embed coordinates between addresses.

```
from geopy.geocoders import Nominatim

# Create a DataFrame with addresses
some_addresses = pd.DataFrame({
    'name': ["South Campus Teaching Hub", "Sefton Park", "Stanley Street"],
    'addr': ["140 Chatham St, Liverpool L7 7BA", "Sefton Park, Liverpool L17 1AP", "4 Stanley
})
```

```
# Initialize geocoder
geolocator = Nominatim(user_agent="geoapiExercises")

# Function for geocoding
def geocode(address):
    location = geolocator.geocode(address)
    return pd.Series([location.latitude, location.longitude], index=['latitude', 'longitude']

# Geocode the addresses
lat_longs = some_addresses['addr'].apply(geocode)

# Adding the results back to the DataFrame
some_addresses = some_addresses.join(lat_longs)
some_addresses
```

	name	addr	latitude	longitude
0	South Campus Teaching Hub	140 Chatham St, Liverpool L7 7BA	53.400188	-2.964437
1	Sefton Park	Sefton Park, Liverpool L17 1AP	53.389846	-2.923999
2	Stanley Street	4 Stanley St, Liverpool L1 6AA	53.407363	-2.987240

4.4.0.1 Reverse Geocoding

You can also reverse geo-code the data, open the output and see what the result is.

```
# Reverse geocoding (optional)
def reverse_geocode(lat, lon):
    location = geolocator.reverse((lat, lon))
    return location.address

# Apply reverse geocoding
some_addresses[['latitude', 'longitude']].apply(lambda x: reverse_geocode(x['latitude'], x['
```

```
0    148, Chatham Street, Canning / Georgian Quarte...
1    Sefton Park, Smithdown Road, Wavertree, Liverp...
2    Davies Street, Cavern Quarter, Liverpool, Live...
dtype: object
```

4.4.0.2 Geographic Data through APIs and the web

APIs can help us generate and create spatial data. In this lab, we have both:

- Written **our own API request.**
- Used **Plug-n-play packages.**

There are many APIs where we can get data these days. A few other examples include

- The London DataStore API
- Thames Water API
- London Air API
- Crime data

4.5 *Group activity answers*

1. Uniform Resource Location (`URL`) is a string of characters that, when interpreted via the Hypertext Transfer Protocol (`HTTP`). URLs point to a data resource, notably files written in Hypertext Markup Language (`HTML`) or a subset of a database.

   - 1xx informational response - the request was received, continuing process
   - 2xx successful - the request was successfully received, understood, and accepted
   - 3xx redirection - further action needs to be taken in order to complete the request
   - 4xx client error - the request contains bad syntax or cannot be fulfilled
   - 5xx server error - the server failed to fulfil an apparently valid request

2. `GET` requests a representation of a data resource corresponding to a particular `URL`. The process of executing the `GET` method is often referred to as a `GET request` and is the main method used for querying RESTful databases. `HEAD`, `POST`, `PUT`, `DELETE`: other common methods, though mostly never used for database querying.

   Surfing the web is basically equivalent to sending a bunch of `GET` requests to different servers and asking for different files written in `HTML`. Suppose, for instance, I wanted to look something up on Wikipedia. Your first step would be to open your web browser and type in http://www.wikipedia.org. Once you hit return, you would see the page below. Several different processes occurred, however, between you hitting "return" and the page finally being rendered:

   1. The web browser took the entered character string, used the command-line tool "Curl" to write a properly formatted `HTTP GET` request, and submitted it to the server that hosts the Wikipedia homepage.
   2. After receiving this request, the server sent an `HTTP` response, from which `Curl` extracted the HTML code for the page (partially shown below).

3. The raw `HTML` code was parsed and then executed by the web browser, rendering the page as seen in the window.
4. Most APIs requires a key or other user credentials before you can query their database. Getting credentials for an API requires that you register with the organization. Once you have successfully registered, you will be assigned one or more keys, tokens, or other credentials that must be supplied to the server as part of any API call you make. To make sure users are not abusing their data access privileges (e.g., by making many rapid queries), each set of keys will be given rate limits governing the total number of calls that can be made over certain intervals of time.

3. Most APIs requires a key before you can query their database. This usually requires you to register with the organization. Most APIs are set up for developers, so you will likely be asked to register an "application." All this really entails is coming up with a name for your app/bot/project and providing your real name, organization, and email. Note that some more popular APIs (e.g., Twitter, Facebook) will require additional information, such as a web address or mobile number. Once you have registered, you will be assigned one or more keys, tokens, or other credentials that must be supplied to the server as part of any API call you make. Most API keys limits the total number of calls that can be made over certain intervals of time. This is so users do not busing their data access privileges.

4.6 References

- Brief History of the Internet, by the Internet Society, is a handy (and free!) introduction to how it all came to be.
- Haklay, M., Singleton, A., Parker, C. 2008. "Web Mapping 2.0: The Neogeography of the GeoWeb". Geography Compass, 2(6):2011–2039
- A blog post from JoeMorrison commenting on the recent change of licensing for some of the core software from Mapbox
- Terman, R., 2020. Computational Tools for Social Science

5 Interactive Maps

Gabriele Filomena has readapted parts of this notebook to prepare this notebook. Copyright (c) Henrikki Tenkanen, Vuokko Heikinheimo, Håvard Wallin Aagesen, Christoph Fink, Kamyar Hasanzadeh.

The **Lecture slides** can be found here.

This **lab**'s notebook can be downloaded from here.

Online maps have been interactive for a long time: virtually all online maps allow to zoom in and out, to pan the map extent, and to select map features, or otherwise query information about them. Interactive content in web pages, such as online maps, are typically implemented using *JavaScript/ECMAScript*, a scripting language originally targeted at web pages, primarily, but used for many other applications.

In the open source realm, there exists a number of different *JavaScript* libraries for interactive web cartography, including Leaflet and OpenLayers. We will not write a single line of *JavaScript*; instead, we will take advantage of the *Folium* Python package: it helps create interactive *Leaflet* maps from data stored in `geopandas.GeoDataFrame`. Folium is a library that bridges the capabilities of data manipulation in Python with the interactive mapping strengths of Leaflet, a leading open-source JavaScript library for creating dynamic, interactive maps in the browser. Let's explore both Folium and Leaflet in more detail to understand their roles and relationship.

Leaflet is at the heart of Folium's mapping capabilities. It's an open-source JavaScript library used extensively in web mapping. Known for its simplicity, performance, and ease of use, Leaflet has become a popular choice among developers for embedding maps in web applications. It allows users to create maps with various interactive features like zooming, panning, markers, popups, and different layers.

Folium builds on Leaflet's foundation, bringing its interactive mapping capabilities into the Python environment. This integration is significant for several reasons:

- With Folium, Python users can use libraries like Pandas and directly create interactive Leaflet maps from within Python. This seamless integration means data can be manipulated, analyzed, and visualized using Python, without the need for extensive JavaScript knowledge.
- Folium abstracts the complexity of Leaflet, allowing users to create sophisticated maps using simple Python commands.

- Folium inherits Leaflet's interactivity, enabling the creation of feature-rich maps with various interactive elements such as markers, layers, and choropleths. These maps can be customized and embedded in web applications or Jupyter notebooks, making them highly versatile for data visualization tasks.
- Folium can render maps with different tilesets (like OpenStreetMap, Stamen Terrain), overlay data in various formats, and even incorporate advanced features like heatmaps or GeoJSON layers.

Folium acts as a bridge between Python and Leaflet, enabling Python users to incorporate and develop web maps within their environment. It leverages Leaflet's core functionalities and extends them into the Python ecosystem, making geospatial data visualization both powerful and accessible within Python workflows.

> Find more information about the capabilities of the *Folium* package on its official web pages: - Documentation - Example gallery - Quickstart tutorial

```
import folium
import pandas as pd
import geopandas as gpd
```

5.1 Simple interactive web maps with Folium

We will start by creating a simple interactive web map that contains nothing but a base map. This is so that we can get acustomed to how Folium's syntax works, and which steps we have to take. We create a `folium.Map` object, and specify centred around which `location` and at which initial zoom level (~0-20) a map shall be displayed. By setting `control_scale` to `True`, we make Folium display a scale bar.

```
location = (39.6508, 66.9654) # Samarkand, Uzbekistan
map = folium.Map(location=location, zoom_start=13, control_scale=True)
map
```

```
<folium.folium.Map at 0x148ed64e650>
```

5.1.0.1 Save the resulting map

To save this map to an HTML file that can be opened in any web browser, use `folium.Map.save()`:

```
map.save("base-map.html")
```

5.1.1 Basemap API

Folium, Leaflet.js and `contextily`, the library that we used for adding basemaps to our static maps in the 2nd session, rely on built-in tilesets borrowed from the `xyzservices` package. The XYZ protocol exposes maps as images for portions of the Earth, which we refer to as 'tiles.' The term 'XYZ' originates from the coordinates used to locate a specific tile. Think of the entire planet divided into squares, each identified by a unique combination of X and Y numbers. Now, add a third component (Z) for the zoom level: lower values use fewer tiles to cover the world, while higher zoom levels (higher Z) provide progressively smaller areas with more detail. Most XYZ APIs serve tiles directly over HTTP, enabling us to access them from the browser. In a nuthesell, XYZ Tiles is a method of expressing map tile mosaics, indexed by (x,y) offsets and zoom levels (z).

```
import xyzservices.providers as xyz
xyz
```

```
{'OpenStreetMap': {'Mapnik': {'url': 'https://tile.openstreetmap.org/{z}/{x}/{y}.png',
   'max_zoom': 19,
   'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
   'attribution': '(C) OpenStreetMap contributors',
   'name': 'OpenStreetMap.Mapnik'},
  'DE': {'url': 'https://tile.openstreetmap.de/{z}/{x}/{y}.png',
   'max_zoom': 18,
   'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
   'attribution': '(C) OpenStreetMap contributors',
   'name': 'OpenStreetMap.DE'},
  'CH': {'url': 'https://tile.osm.ch/switzerland/{z}/{x}/{y}.png',
   'max_zoom': 18,
   'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
   'attribution': '(C) OpenStreetMap contributors',
   'bounds': [[45, 5], [48, 11]],
   'name': 'OpenStreetMap.CH'},
  'France': {'url': 'https://{s}.tile.openstreetmap.fr/osmfr/{z}/{x}/{y}.png',
   'max_zoom': 20,
   'html_attribution': '&copy; OpenStreetMap France | &copy; <a href="https://www.openstreetm
   'attribution': '(C) OpenStreetMap France | (C) OpenStreetMap contributors',
   'name': 'OpenStreetMap.France'},
  'HOT': {'url': 'https://{s}.tile.openstreetmap.fr/hot/{z}/{x}/{y}.png',
   'max_zoom': 19,
```

```
  'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
  'attribution': '(C) OpenStreetMap contributors, Tiles style by Humanitarian OpenStreetMap
  'name': 'OpenStreetMap.HOT'},
 'BZH': {'url': 'https://tile.openstreetmap.bzh/br/{z}/{x}/{y}.png',
  'max_zoom': 19,
  'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
  'attribution': '(C) OpenStreetMap contributors, Tiles courtesy of Breton OpenStreetMap Tea
  'bounds': [[46.2, -5.5], [50, 0.7]],
  'name': 'OpenStreetMap.BZH'},
 'BlackAndWhite': {'url': 'http://{s}.tiles.wmflabs.org/bw-mapnik/{z}/{x}/{y}.png',
  'max_zoom': 18,
  'attribution': '(C) OpenStreetMap contributors',
  'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
  'name': 'OpenStreetMap.BlackAndWhite'}},
'MapTilesAPI': {'OSMEnglish': {'url': 'https://maptiles.p.rapidapi.com/{variant}/{z}/{x}/{y}
  'html_attribution': '&copy; <a href="http://www.maptilesapi.com/">MapTiles API</a>, &copy
  'attribution': '(C) MapTiles API, (C) OpenStreetMap contributors',
  'variant': 'en/map/v1',
  'apikey': '<insert your api key here>',
  'max_zoom': 19,
  'name': 'MapTilesAPI.OSMEnglish'},
 'OSMFrancais': {'url': 'https://maptiles.p.rapidapi.com/{variant}/{z}/{x}/{y}.png?rapidapi-
  'html_attribution': '&copy; <a href="http://www.maptilesapi.com/">MapTiles API</a>, &copy
  'attribution': '(C) MapTiles API, (C) OpenStreetMap contributors',
  'variant': 'fr/map/v1',
  'apikey': '<insert your api key here>',
  'max_zoom': 19,
  'name': 'MapTilesAPI.OSMFrancais'},
 'OSMEspagnol': {'url': 'https://maptiles.p.rapidapi.com/{variant}/{z}/{x}/{y}.png?rapidapi-
  'html_attribution': '&copy; <a href="http://www.maptilesapi.com/">MapTiles API</a>, &copy
  'attribution': '(C) MapTiles API, (C) OpenStreetMap contributors',
  'variant': 'es/map/v1',
  'apikey': '<insert your api key here>',
  'max_zoom': 19,
  'name': 'MapTilesAPI.OSMEspagnol'}},
'OpenSeaMap': {'url': 'https://tiles.openseamap.org/seamark/{z}/{x}/{y}.png',
 'html_attribution': 'Map data: &copy; <a href="http://www.openseamap.org">OpenSeaMap</a> co
 'attribution': 'Map data: (C) OpenSeaMap contributors',
 'name': 'OpenSeaMap'},
'OPNVKarte': {'url': 'https://tileserver.memomaps.de/tilegen/{z}/{x}/{y}.png',
 'max_zoom': 18,
 'html_attribution': 'Map <a href="https://memomaps.de/">memomaps.de</a> <a href="http://cre
 'attribution': 'Map memomaps.de CC-BY-SA, map data (C) OpenStreetMap contributors',
```

```
 'name': 'OPNVKarte'},
'OpenTopoMap': {'url': 'https://{s}.tile.opentopomap.org/{z}/{x}/{y}.png',
 'max_zoom': 17,
 'html_attribution': 'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">Ope
 'attribution': 'Map data: (C) OpenStreetMap contributors, SRTM | Map style: (C) OpenTopoMap
 'name': 'OpenTopoMap'},
'OpenRailwayMap': {'url': 'https://{s}.tiles.openrailwaymap.org/standard/{z}/{x}/{y}.png',
 'max_zoom': 19,
 'html_attribution': 'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">Ope
 'attribution': 'Map data: (C) OpenStreetMap contributors | Map style: (C) OpenRailwayMap (C
 'name': 'OpenRailwayMap'},
'OpenFireMap': {'url': 'http://openfiremap.org/hytiles/{z}/{x}/{y}.png',
 'max_zoom': 19,
 'html_attribution': 'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">Ope
 'attribution': 'Map data: (C) OpenStreetMap contributors | Map style: (C) OpenFireMap (CC-B
 'name': 'OpenFireMap'},
'SafeCast': {'url': 'https://s3.amazonaws.com/te512.safecast.org/{z}/{x}/{y}.png',
 'max_zoom': 16,
 'html_attribution': 'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">Ope
 'attribution': 'Map data: (C) OpenStreetMap contributors | Map style: (C) SafeCast (CC-BY-S
 'name': 'SafeCast'},
'Stadia': {'AlidadeSmooth': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y
  'min_zoom': 0,
  'max_zoom': 20,
  'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
  'attribution': '(C) Stadia Maps (C) OpenMapTiles (C) OpenStreetMap contributors',
  'variant': 'alidade_smooth',
  'ext': 'png',
  'name': 'Stadia.AlidadeSmooth'},
 'AlidadeSmoothDark': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y}{r}.{
  'min_zoom': 0,
  'max_zoom': 20,
  'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
  'attribution': '(C) Stadia Maps (C) OpenMapTiles (C) OpenStreetMap contributors',
  'variant': 'alidade_smooth_dark',
  'ext': 'png',
  'name': 'Stadia.AlidadeSmoothDark'},
 'OSMBright': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y}{r}.{ext}',
  'min_zoom': 0,
  'max_zoom': 20,
  'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
  'attribution': '(C) Stadia Maps (C) OpenMapTiles (C) OpenStreetMap contributors',
  'variant': 'osm_bright',
```

```
 'ext': 'png',
 'name': 'Stadia.OSMBright'},
'Outdoors': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y}{r}.{ext}',
 'min_zoom': 0,
 'max_zoom': 20,
 'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
 'attribution': '(C) Stadia Maps (C) OpenMapTiles (C) OpenStreetMap contributors',
 'variant': 'outdoors',
 'ext': 'png',
 'name': 'Stadia.Outdoors'},
'StamenToner': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y}{r}.{ext}'
 'min_zoom': 0,
 'max_zoom': 20,
 'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
 'attribution': '(C) Stadia Maps (C) Stamen Design (C) OpenMapTiles (C) OpenStreetMap contr
 'variant': 'stamen_toner',
 'ext': 'png',
 'name': 'Stadia.StamenToner'},
'StamenTonerBackground': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y}+
 'min_zoom': 0,
 'max_zoom': 20,
 'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
 'attribution': '(C) Stadia Maps (C) Stamen Design (C) OpenMapTiles (C) OpenStreetMap contr
 'variant': 'stamen_toner_background',
 'ext': 'png',
 'name': 'Stadia.StamenTonerBackground'},
'StamenTonerLines': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y}{r}.{e
 'min_zoom': 0,
 'max_zoom': 20,
 'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
 'attribution': '(C) Stadia Maps (C) Stamen Design (C) OpenMapTiles (C) OpenStreetMap contr
 'variant': 'stamen_toner_lines',
 'ext': 'png',
 'name': 'Stadia.StamenTonerLines'},
'StamenTonerLabels': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y}{r}.+
 'min_zoom': 0,
 'max_zoom': 20,
 'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
 'attribution': '(C) Stadia Maps (C) Stamen Design (C) OpenMapTiles (C) OpenStreetMap contr
 'variant': 'stamen_toner_labels',
 'ext': 'png',
 'name': 'Stadia.StamenTonerLabels'},
'StamenTonerLite': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y}{r}.{ex
```

```
 'min_zoom': 0,
 'max_zoom': 20,
 'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
 'attribution': '(C) Stadia Maps (C) Stamen Design (C) OpenMapTiles (C) OpenStreetMap contr
 'variant': 'stamen_toner_lite',
 'ext': 'png',
 'name': 'Stadia.StamenTonerLite'},
'StamenWatercolor': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y}.{ext}
 'min_zoom': 1,
 'max_zoom': 16,
 'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
 'attribution': '(C) Stadia Maps (C) Stamen Design (C) OpenMapTiles (C) OpenStreetMap contr
 'variant': 'stamen_watercolor',
 'ext': 'jpg',
 'name': 'Stadia.StamenWatercolor'},
'StamenTerrain': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y}{r}.{ext}
 'min_zoom': 0,
 'max_zoom': 18,
 'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
 'attribution': '(C) Stadia Maps (C) Stamen Design (C) OpenMapTiles (C) OpenStreetMap contr
 'variant': 'stamen_terrain',
 'ext': 'png',
 'name': 'Stadia.StamenTerrain'},
'StamenTerrainBackground': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y
 'min_zoom': 0,
 'max_zoom': 18,
 'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
 'attribution': '(C) Stadia Maps (C) Stamen Design (C) OpenMapTiles (C) OpenStreetMap contr
 'variant': 'stamen_terrain_background',
 'ext': 'png',
 'name': 'Stadia.StamenTerrainBackground'},
'StamenTerrainLabels': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y}{r}
 'min_zoom': 0,
 'max_zoom': 18,
 'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
 'attribution': '(C) Stadia Maps (C) Stamen Design (C) OpenMapTiles (C) OpenStreetMap contr
 'variant': 'stamen_terrain_labels',
 'ext': 'png',
 'name': 'Stadia.StamenTerrainLabels'},
'StamenTerrainLines': {'url': 'https://tiles.stadiamaps.com/tiles/{variant}/{z}/{x}/{y}{r}
 'min_zoom': 0,
 'max_zoom': 18,
 'html_attribution': '&copy; <a href="https://www.stadiamaps.com/" target="_blank">Stadia M
```

```
  'attribution': '(C) Stadia Maps (C) Stamen Design (C) OpenMapTiles (C) OpenStreetMap contr
  'variant': 'stamen_terrain_lines',
  'ext': 'png',
  'name': 'Stadia.StamenTerrainLines'}},
'Thunderforest': {'OpenCycleMap': {'url': 'https://{s}.tile.thunderforest.com/{variant}/{z}/
  'html_attribution': '&copy; <a href="http://www.thunderforest.com/">Thunderforest</a>, &co
  'attribution': '(C) Thunderforest, (C) OpenStreetMap contributors',
  'variant': 'cycle',
  'apikey': '<insert your api key here>',
  'max_zoom': 22,
  'name': 'Thunderforest.OpenCycleMap'},
 'Transport': {'url': 'https://{s}.tile.thunderforest.com/{variant}/{z}/{x}/{y}.png?apikey=
  'html_attribution': '&copy; <a href="http://www.thunderforest.com/">Thunderforest</a>, &co
  'attribution': '(C) Thunderforest, (C) OpenStreetMap contributors',
  'variant': 'transport',
  'apikey': '<insert your api key here>',
  'max_zoom': 22,
  'name': 'Thunderforest.Transport'},
 'TransportDark': {'url': 'https://{s}.tile.thunderforest.com/{variant}/{z}/{x}/{y}.png?apik
  'html_attribution': '&copy; <a href="http://www.thunderforest.com/">Thunderforest</a>, &co
  'attribution': '(C) Thunderforest, (C) OpenStreetMap contributors',
  'variant': 'transport-dark',
  'apikey': '<insert your api key here>',
  'max_zoom': 22,
  'name': 'Thunderforest.TransportDark'},
 'SpinalMap': {'url': 'https://{s}.tile.thunderforest.com/{variant}/{z}/{x}/{y}.png?apikey=
  'html_attribution': '&copy; <a href="http://www.thunderforest.com/">Thunderforest</a>, &co
  'attribution': '(C) Thunderforest, (C) OpenStreetMap contributors',
  'variant': 'spinal-map',
  'apikey': '<insert your api key here>',
  'max_zoom': 22,
  'name': 'Thunderforest.SpinalMap'},
 'Landscape': {'url': 'https://{s}.tile.thunderforest.com/{variant}/{z}/{x}/{y}.png?apikey=
  'html_attribution': '&copy; <a href="http://www.thunderforest.com/">Thunderforest</a>, &co
  'attribution': '(C) Thunderforest, (C) OpenStreetMap contributors',
  'variant': 'landscape',
  'apikey': '<insert your api key here>',
  'max_zoom': 22,
  'name': 'Thunderforest.Landscape'},
 'Outdoors': {'url': 'https://{s}.tile.thunderforest.com/{variant}/{z}/{x}/{y}.png?apikey={a
  'html_attribution': '&copy; <a href="http://www.thunderforest.com/">Thunderforest</a>, &co
  'attribution': '(C) Thunderforest, (C) OpenStreetMap contributors',
  'variant': 'outdoors',
```

```
  'apikey': '<insert your api key here>',
  'max_zoom': 22,
  'name': 'Thunderforest.Outdoors'},
 'Pioneer': {'url': 'https://{s}.tile.thunderforest.com/{variant}/{z}/{x}/{y}.png?apikey={ap
  'html_attribution': '&copy; <a href="http://www.thunderforest.com/">Thunderforest</a>, &co
  'attribution': '(C) Thunderforest, (C) OpenStreetMap contributors',
  'variant': 'pioneer',
  'apikey': '<insert your api key here>',
  'max_zoom': 22,
  'name': 'Thunderforest.Pioneer'},
 'MobileAtlas': {'url': 'https://{s}.tile.thunderforest.com/{variant}/{z}/{x}/{y}.png?apikey
  'html_attribution': '&copy; <a href="http://www.thunderforest.com/">Thunderforest</a>, &co
  'attribution': '(C) Thunderforest, (C) OpenStreetMap contributors',
  'variant': 'mobile-atlas',
  'apikey': '<insert your api key here>',
  'max_zoom': 22,
  'name': 'Thunderforest.MobileAtlas'},
 'Neighbourhood': {'url': 'https://{s}.tile.thunderforest.com/{variant}/{z}/{x}/{y}.png?apik
  'html_attribution': '&copy; <a href="http://www.thunderforest.com/">Thunderforest</a>, &co
  'attribution': '(C) Thunderforest, (C) OpenStreetMap contributors',
  'variant': 'neighbourhood',
  'apikey': '<insert your api key here>',
  'max_zoom': 22,
  'name': 'Thunderforest.Neighbourhood'}},
'CyclOSM': {'url': 'https://{s}.tile-cyclosm.openstreetmap.fr/cyclosm/{z}/{x}/{y}.png',
 'max_zoom': 20,
 'html_attribution': '<a href="https://github.com/cyclosm/cyclosm-cartocss-style/releases" t
 'attribution': 'CyclOSM | Map data: (C) OpenStreetMap contributors',
 'name': 'CyclOSM'},
'Jawg': {'Streets': {'url': 'https://{s}.tile.jawg.io/{variant}/{z}/{x}/{y}{r}.png?access-to
  'html_attribution': '<a href="http://jawg.io" title="Tiles Courtesy of Jawg Maps" target="
  'attribution': '(C) **Jawg** Maps (C) OpenStreetMap contributors',
  'min_zoom': 0,
  'max_zoom': 22,
  'subdomains': 'abcd',
  'variant': 'jawg-streets',
  'accessToken': '<insert your access token here>',
  'name': 'Jawg.Streets'},
 'Terrain': {'url': 'https://{s}.tile.jawg.io/{variant}/{z}/{x}/{y}{r}.png?access-token={ac
  'html_attribution': '<a href="http://jawg.io" title="Tiles Courtesy of Jawg Maps" target="
  'attribution': '(C) **Jawg** Maps (C) OpenStreetMap contributors',
  'min_zoom': 0,
  'max_zoom': 22,
```

```
  'subdomains': 'abcd',
  'variant': 'jawg-terrain',
  'accessToken': '<insert your access token here>',
  'name': 'Jawg.Terrain'},
 'Sunny': {'url': 'https://{s}.tile.jawg.io/{variant}/{z}/{x}/{y}{r}.png?access-token={acces
  'html_attribution': '<a href="http://jawg.io" title="Tiles Courtesy of Jawg Maps" target="
  'attribution': '(C) **Jawg** Maps (C) OpenStreetMap contributors',
  'min_zoom': 0,
  'max_zoom': 22,
  'subdomains': 'abcd',
  'variant': 'jawg-sunny',
  'accessToken': '<insert your access token here>',
  'name': 'Jawg.Sunny'},
 'Dark': {'url': 'https://{s}.tile.jawg.io/{variant}/{z}/{x}/{y}{r}.png?access-token={access
  'html_attribution': '<a href="http://jawg.io" title="Tiles Courtesy of Jawg Maps" target="
  'attribution': '(C) **Jawg** Maps (C) OpenStreetMap contributors',
  'min_zoom': 0,
  'max_zoom': 22,
  'subdomains': 'abcd',
  'variant': 'jawg-dark',
  'accessToken': '<insert your access token here>',
  'name': 'Jawg.Dark'},
 'Light': {'url': 'https://{s}.tile.jawg.io/{variant}/{z}/{x}/{y}{r}.png?access-token={acces
  'html_attribution': '<a href="http://jawg.io" title="Tiles Courtesy of Jawg Maps" target="
  'attribution': '(C) **Jawg** Maps (C) OpenStreetMap contributors',
  'min_zoom': 0,
  'max_zoom': 22,
  'subdomains': 'abcd',
  'variant': 'jawg-light',
  'accessToken': '<insert your access token here>',
  'name': 'Jawg.Light'},
 'Matrix': {'url': 'https://{s}.tile.jawg.io/{variant}/{z}/{x}/{y}{r}.png?access-token={acce
  'html_attribution': '<a href="http://jawg.io" title="Tiles Courtesy of Jawg Maps" target="
  'attribution': '(C) **Jawg** Maps (C) OpenStreetMap contributors',
  'min_zoom': 0,
  'max_zoom': 22,
  'subdomains': 'abcd',
  'variant': 'jawg-matrix',
  'accessToken': '<insert your access token here>',
  'name': 'Jawg.Matrix'}},
'MapBox': {'url': 'https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}{r}?access_token=
 'html_attribution': '&copy; <a href="https://www.mapbox.com/about/maps/" target="_blank">Ma
 'attribution': '(C) Mapbox (C) OpenStreetMap contributors Improve this map',
```

```
 'tileSize': 512,
 'max_zoom': 18,
 'zoomOffset': -1,
 'id': 'mapbox/streets-v11',
 'accessToken': '<insert your access token here>',
 'name': 'MapBox'},
'MapTiler': {'Streets': {'url': 'https://api.maptiler.com/maps/{variant}/{z}/{x}/{y}{r}.{ext
  'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
  'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
  'variant': 'streets',
  'ext': 'png',
  'key': '<insert your MapTiler Cloud API key here>',
  'tileSize': 512,
  'zoomOffset': -1,
  'min_zoom': 0,
  'max_zoom': 21,
  'name': 'MapTiler.Streets'},
 'Basic': {'url': 'https://api.maptiler.com/maps/{variant}/{z}/{x}/{y}{r}.{ext}?key={key}',
  'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
  'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
  'variant': 'basic',
  'ext': 'png',
  'key': '<insert your MapTiler Cloud API key here>',
  'tileSize': 512,
  'zoomOffset': -1,
  'min_zoom': 0,
  'max_zoom': 21,
  'name': 'MapTiler.Basic'},
 'Bright': {'url': 'https://api.maptiler.com/maps/{variant}/{z}/{x}/{y}{r}.{ext}?key={key}'
  'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
  'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
  'variant': 'bright',
  'ext': 'png',
  'key': '<insert your MapTiler Cloud API key here>',
  'tileSize': 512,
  'zoomOffset': -1,
  'min_zoom': 0,
  'max_zoom': 21,
  'name': 'MapTiler.Bright'},
 'Pastel': {'url': 'https://api.maptiler.com/maps/{variant}/{z}/{x}/{y}{r}.{ext}?key={key}'
  'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
  'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
  'variant': 'pastel',
```

```
 'ext': 'png',
 'key': '<insert your MapTiler Cloud API key here>',
 'tileSize': 512,
 'zoomOffset': -1,
 'min_zoom': 0,
 'max_zoom': 21,
 'name': 'MapTiler.Pastel'},
'Positron': {'url': 'https://api.maptiler.com/maps/{variant}/{z}/{x}/{y}{r}.{ext}?key={key
 'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
 'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
 'variant': 'positron',
 'ext': 'png',
 'key': '<insert your MapTiler Cloud API key here>',
 'tileSize': 512,
 'zoomOffset': -1,
 'min_zoom': 0,
 'max_zoom': 21,
 'name': 'MapTiler.Positron'},
'Hybrid': {'url': 'https://api.maptiler.com/maps/{variant}/{z}/{x}/{y}{r}.{ext}?key={key}'
 'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
 'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
 'variant': 'hybrid',
 'ext': 'jpg',
 'key': '<insert your MapTiler Cloud API key here>',
 'tileSize': 512,
 'zoomOffset': -1,
 'min_zoom': 0,
 'max_zoom': 21,
 'name': 'MapTiler.Hybrid'},
'Toner': {'url': 'https://api.maptiler.com/maps/{variant}/{z}/{x}/{y}{r}.{ext}?key={key}',
 'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
 'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
 'variant': 'toner',
 'ext': 'png',
 'key': '<insert your MapTiler Cloud API key here>',
 'tileSize': 512,
 'zoomOffset': -1,
 'min_zoom': 0,
 'max_zoom': 21,
 'name': 'MapTiler.Toner'},
'Topo': {'url': 'https://api.maptiler.com/maps/{variant}/{z}/{x}/{y}{r}.{ext}?key={key}',
 'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
 'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
```

```
 'variant': 'topo',
 'ext': 'png',
 'key': '<insert your MapTiler Cloud API key here>',
 'tileSize': 512,
 'zoomOffset': -1,
 'min_zoom': 0,
 'max_zoom': 21,
 'name': 'MapTiler.Topo'},
'Voyager': {'url': 'https://api.maptiler.com/maps/{variant}/{z}/{x}/{y}{r}.{ext}?key={key}
 'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
 'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
 'variant': 'voyager',
 'ext': 'png',
 'key': '<insert your MapTiler Cloud API key here>',
 'tileSize': 512,
 'zoomOffset': -1,
 'min_zoom': 0,
 'max_zoom': 21,
 'name': 'MapTiler.Voyager'},
'Basic4326': {'url': 'https://api.maptiler.com/maps/{variant}/{z}/{x}/{y}{r}.{ext}?key={key
 'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
 'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
 'variant': 'basic-4326',
 'ext': 'png',
 'key': '<insert your MapTiler Cloud API key here>',
 'tileSize': 512,
 'zoomOffset': -1,
 'min_zoom': 0,
 'max_zoom': 21,
 'name': 'MapTiler.Basic4326',
 'crs': 'EPSG:4326'},
'Outdoor': {'url': 'https://api.maptiler.com/maps/{variant}/{z}/{x}/{y}{r}.{ext}?key={key}
 'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
 'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
 'variant': 'outdoor',
 'ext': 'png',
 'key': '<insert your MapTiler Cloud API key here>',
 'tileSize': 512,
 'zoomOffset': -1,
 'min_zoom': 0,
 'max_zoom': 21,
 'name': 'MapTiler.Outdoor'},
'Topographique': {'url': 'https://api.maptiler.com/maps/{variant}/{z}/{x}/{y}{r}.{ext}?key=
```

```
  'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
  'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
  'variant': 'topographique',
  'ext': 'png',
  'key': '<insert your MapTiler Cloud API key here>',
  'tileSize': 512,
  'zoomOffset': -1,
  'min_zoom': 0,
  'max_zoom': 21,
  'name': 'MapTiler.Topographique'},
 'Winter': {'url': 'https://api.maptiler.com/maps/{variant}/{z}/{x}/{y}{r}.{ext}?key={key}'
  'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
  'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
  'variant': 'winter',
  'ext': 'png',
  'key': '<insert your MapTiler Cloud API key here>',
  'tileSize': 512,
  'zoomOffset': -1,
  'min_zoom': 0,
  'max_zoom': 21,
  'name': 'MapTiler.Winter'},
 'Satellite': {'url': 'https://api.maptiler.com/tiles/{variant}/{z}/{x}/{y}.{ext}?key={key}
  'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
  'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
  'variant': 'satellite-v2',
  'ext': 'jpg',
  'key': '<insert your MapTiler Cloud API key here>',
  'min_zoom': 0,
  'max_zoom': 20,
  'name': 'MapTiler.Satellite'},
 'Terrain': {'url': 'https://api.maptiler.com/tiles/{variant}/{z}/{x}/{y}.{ext}?key={key}',
  'html_attribution': '<a href="https://www.maptiler.com/copyright/" target="_blank">&copy;
  'attribution': '(C) MapTiler (C) OpenStreetMap contributors',
  'variant': 'terrain-rgb',
  'ext': 'png',
  'key': '<insert your MapTiler Cloud API key here>',
  'min_zoom': 0,
  'max_zoom': 12,
  'name': 'MapTiler.Terrain'}},
'TomTom': {'Basic': {'url': 'https://{s}.api.tomtom.com/map/1/tile/{variant}/{style}/{z}/{x}
  'variant': 'basic',
  'max_zoom': 22,
  'html_attribution': '<a href="https://tomtom.com" target="_blank">&copy;  1992 - 2023 Tom
```

```
  'attribution': '(C) 1992 - 2023 TomTom.',
  'subdomains': 'abcd',
  'style': 'main',
  'ext': 'png',
  'apikey': '<insert your API key here>',
  'name': 'TomTom.Basic'},
 'Hybrid': {'url': 'https://{s}.api.tomtom.com/map/1/tile/{variant}/{style}/{z}/{x}/{y}.{ext
  'variant': 'hybrid',
  'max_zoom': 22,
  'html_attribution': '<a href="https://tomtom.com" target="_blank">&copy;  1992 - 2023 Tom
  'attribution': '(C) 1992 - 2023 TomTom.',
  'subdomains': 'abcd',
  'style': 'main',
  'ext': 'png',
  'apikey': '<insert your API key here>',
  'name': 'TomTom.Hybrid'},
 'Labels': {'url': 'https://{s}.api.tomtom.com/map/1/tile/{variant}/{style}/{z}/{x}/{y}.{ext
  'variant': 'labels',
  'max_zoom': 22,
  'html_attribution': '<a href="https://tomtom.com" target="_blank">&copy;  1992 - 2023 Tom
  'attribution': '(C) 1992 - 2023 TomTom.',
  'subdomains': 'abcd',
  'style': 'main',
  'ext': 'png',
  'apikey': '<insert your API key here>',
  'name': 'TomTom.Labels'}},
'Esri': {'WorldStreetMap': {'url': 'https://server.arcgisonline.com/ArcGIS/rest/services/{va
  'variant': 'World_Street_Map',
  'html_attribution': 'Tiles &copy; Esri &mdash; Source: Esri, DeLorme, NAVTEQ, USGS, Intern
  'attribution': 'Tiles (C) Esri -- Source: Esri, DeLorme, NAVTEQ, USGS, Intermap, iPC, NRCA
  'name': 'Esri.WorldStreetMap'},
 'DeLorme': {'url': 'https://server.arcgisonline.com/ArcGIS/rest/services/{variant}/MapServe
  'variant': 'Specialty/DeLorme_World_Base_Map',
  'html_attribution': 'Tiles &copy; Esri &mdash; Copyright: &copy;2012 DeLorme',
  'attribution': 'Tiles (C) Esri -- Copyright: (C)2012 DeLorme',
  'min_zoom': 1,
  'max_zoom': 11,
  'name': 'Esri.DeLorme'},
 'WorldTopoMap': {'url': 'https://server.arcgisonline.com/ArcGIS/rest/services/{variant}/Map
  'variant': 'World_Topo_Map',
  'html_attribution': 'Tiles &copy; Esri &mdash; Esri, DeLorme, NAVTEQ, TomTom, Intermap, iP
  'attribution': 'Tiles (C) Esri -- Esri, DeLorme, NAVTEQ, TomTom, Intermap, iPC, USGS, FAO
  'name': 'Esri.WorldTopoMap'},
```

```
'WorldImagery': {'url': 'https://server.arcgisonline.com/ArcGIS/rest/services/{variant}/Ma
 'variant': 'World_Imagery',
 'html_attribution': 'Tiles &copy; Esri &mdash; Source: Esri, i-cubed, USDA, USGS, AEX, Ge
 'attribution': 'Tiles (C) Esri -- Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapp
 'name': 'Esri.WorldImagery'},
'WorldTerrain': {'url': 'https://server.arcgisonline.com/ArcGIS/rest/services/{variant}/Ma
 'variant': 'World_Terrain_Base',
 'html_attribution': 'Tiles &copy; Esri &mdash; Source: USGS, Esri, TANA, DeLorme, and NPS
 'attribution': 'Tiles (C) Esri -- Source: USGS, Esri, TANA, DeLorme, and NPS',
 'max_zoom': 13,
 'name': 'Esri.WorldTerrain'},
'WorldShadedRelief': {'url': 'https://server.arcgisonline.com/ArcGIS/rest/services/{varian
 'variant': 'World_Shaded_Relief',
 'html_attribution': 'Tiles &copy; Esri &mdash; Source: Esri',
 'attribution': 'Tiles (C) Esri -- Source: Esri',
 'max_zoom': 13,
 'name': 'Esri.WorldShadedRelief'},
'WorldPhysical': {'url': 'https://server.arcgisonline.com/ArcGIS/rest/services/{variant}/M
 'variant': 'World_Physical_Map',
 'html_attribution': 'Tiles &copy; Esri &mdash; Source: US National Park Service',
 'attribution': 'Tiles (C) Esri -- Source: US National Park Service',
 'max_zoom': 8,
 'name': 'Esri.WorldPhysical'},
'OceanBasemap': {'url': 'https://server.arcgisonline.com/ArcGIS/rest/services/{variant}/Ma
 'variant': 'Ocean/World_Ocean_Base',
 'html_attribution': 'Tiles &copy; Esri &mdash; Sources: GEBCO, NOAA, CHS, OSU, UNH, CSUMB
 'attribution': 'Tiles (C) Esri -- Sources: GEBCO, NOAA, CHS, OSU, UNH, CSUMB, National Ge
 'max_zoom': 13,
 'name': 'Esri.OceanBasemap'},
'NatGeoWorldMap': {'url': 'https://server.arcgisonline.com/ArcGIS/rest/services/{variant}/
 'variant': 'NatGeo_World_Map',
 'html_attribution': 'Tiles &copy; Esri &mdash; National Geographic, Esri, DeLorme, NAVTEQ
 'attribution': 'Tiles (C) Esri -- National Geographic, Esri, DeLorme, NAVTEQ, UNEP-WCMC, U
 'max_zoom': 16,
 'name': 'Esri.NatGeoWorldMap'},
'WorldGrayCanvas': {'url': 'https://server.arcgisonline.com/ArcGIS/rest/services/{variant}/
 'variant': 'Canvas/World_Light_Gray_Base',
 'html_attribution': 'Tiles &copy; Esri &mdash; Esri, DeLorme, NAVTEQ',
 'attribution': 'Tiles (C) Esri -- Esri, DeLorme, NAVTEQ',
 'max_zoom': 16,
 'name': 'Esri.WorldGrayCanvas'},
'ArcticImagery': {'url': 'http://server.arcgisonline.com/ArcGIS/rest/services/Polar/Arctic
 'variant': 'Arctic_Imagery',
```

```
 'html_attribution': 'Earthstar Geographics',
 'attribution': 'Earthstar Geographics',
 'max_zoom': 24,
 'name': 'Esri.ArcticImagery',
 'crs': 'EPSG:5936',
 'bounds': [[-2623285.8808999993, -2623285.8808999993],
  [6623285.8803, 6623285.8803]]},
'ArcticOceanBase': {'url': 'http://server.arcgisonline.com/ArcGIS/rest/services/Polar/Arcti
 'variant': 'Arctic_Ocean_Base',
 'html_attribution': 'Tiles &copy; Esri &mdash; Esri, DeLorme, NAVTEQ, TomTom, Intermap, iP
 'attribution': 'Tiles &copy; Esri &mdash; Esri, DeLorme, NAVTEQ, TomTom, Intermap, iPC, US
 'max_zoom': 24,
 'name': 'Esri.ArcticOceanBase',
 'crs': 'EPSG:5936',
 'bounds': [[-2623285.8808999993, -2623285.8808999993],
  [6623285.8803, 6623285.8803]]},
'ArcticOceanReference': {'url': 'http://server.arcgisonline.com/ArcGIS/rest/services/Polar/
 'variant': 'Arctic_Ocean_Reference',
 'html_attribution': 'Tiles &copy; Esri &mdash; Esri, DeLorme, NAVTEQ, TomTom, Intermap, iP
 'attribution': 'Tiles &copy; Esri &mdash; Esri, DeLorme, NAVTEQ, TomTom, Intermap, iPC, US
 'max_zoom': 24,
 'name': 'Esri.ArcticOceanReference',
 'crs': 'EPSG:5936',
 'bounds': [[-2623285.8808999993, -2623285.8808999993],
  [6623285.8803, 6623285.8803]]},
'AntarcticImagery': {'url': 'http://server.arcgisonline.com/ArcGIS/rest/services/Polar/Anta
 'variant': 'Antarctic_Imagery',
 'html_attribution': 'Earthstar Geographics',
 'attribution': 'Earthstar Geographics',
 'max_zoom': 24,
 'name': 'Esri.AntarcticImagery',
 'crs': 'EPSG:3031',
 'bounds': [[-9913957.327914657, -5730886.461772691],
  [9913957.327914657, 5730886.461773157]]},
'AntarcticBasemap': {'url': 'https://tiles.arcgis.com/tiles/C8EMgrsFcRFL6LrL/arcgis/rest/se
 'variant': 'Antarctic_Basemap',
 'html_attribution': 'Imagery provided by NOAA National Centers for Environmental Informati
 'attribution': 'Imagery provided by NOAA National Centers for Environmental Information (N
 'max_zoom': 9,
 'name': 'Esri.AntarcticBasemap',
 'crs': 'EPSG:3031',
 'bounds': [[-4524583.19363305, -4524449.487765655],
  [4524449.4877656475, 4524583.193633042]]}},
```

```
'OpenWeatherMap': {'Clouds': {'url': 'http://{s}.tile.openweathermap.org/map/{variant}/{z}/
  'max_zoom': 19,
  'html_attribution': 'Map data &copy; <a href="http://openweathermap.org">OpenWeatherMap</a
  'attribution': 'Map data (C) OpenWeatherMap',
  'apiKey': '<insert your api key here>',
  'opacity': 0.5,
  'variant': 'clouds',
  'name': 'OpenWeatherMap.Clouds'},
 'CloudsClassic': {'url': 'http://{s}.tile.openweathermap.org/map/{variant}/{z}/{x}/{y}.png
  'max_zoom': 19,
  'html_attribution': 'Map data &copy; <a href="http://openweathermap.org">OpenWeatherMap</a
  'attribution': 'Map data (C) OpenWeatherMap',
  'apiKey': '<insert your api key here>',
  'opacity': 0.5,
  'variant': 'clouds_cls',
  'name': 'OpenWeatherMap.CloudsClassic'},
 'Precipitation': {'url': 'http://{s}.tile.openweathermap.org/map/{variant}/{z}/{x}/{y}.png
  'max_zoom': 19,
  'html_attribution': 'Map data &copy; <a href="http://openweathermap.org">OpenWeatherMap</a
  'attribution': 'Map data (C) OpenWeatherMap',
  'apiKey': '<insert your api key here>',
  'opacity': 0.5,
  'variant': 'precipitation',
  'name': 'OpenWeatherMap.Precipitation'},
 'PrecipitationClassic': {'url': 'http://{s}.tile.openweathermap.org/map/{variant}/{z}/{x}/
  'max_zoom': 19,
  'html_attribution': 'Map data &copy; <a href="http://openweathermap.org">OpenWeatherMap</a
  'attribution': 'Map data (C) OpenWeatherMap',
  'apiKey': '<insert your api key here>',
  'opacity': 0.5,
  'variant': 'precipitation_cls',
  'name': 'OpenWeatherMap.PrecipitationClassic'},
 'Rain': {'url': 'http://{s}.tile.openweathermap.org/map/{variant}/{z}/{x}/{y}.png?appid={ap
  'max_zoom': 19,
  'html_attribution': 'Map data &copy; <a href="http://openweathermap.org">OpenWeatherMap</a
  'attribution': 'Map data (C) OpenWeatherMap',
  'apiKey': '<insert your api key here>',
  'opacity': 0.5,
  'variant': 'rain',
  'name': 'OpenWeatherMap.Rain'},
 'RainClassic': {'url': 'http://{s}.tile.openweathermap.org/map/{variant}/{z}/{x}/{y}.png?ap
  'max_zoom': 19,
  'html_attribution': 'Map data &copy; <a href="http://openweathermap.org">OpenWeatherMap</a
```

```
 'attribution': 'Map data (C) OpenWeatherMap',
 'apiKey': '<insert your api key here>',
 'opacity': 0.5,
 'variant': 'rain_cls',
 'name': 'OpenWeatherMap.RainClassic'},
'Pressure': {'url': 'http://{s}.tile.openweathermap.org/map/{variant}/{z}/{x}/{y}.png?appi
 'max_zoom': 19,
 'html_attribution': 'Map data &copy; <a href="http://openweathermap.org">OpenWeatherMap</
 'attribution': 'Map data (C) OpenWeatherMap',
 'apiKey': '<insert your api key here>',
 'opacity': 0.5,
 'variant': 'pressure',
 'name': 'OpenWeatherMap.Pressure'},
'PressureContour': {'url': 'http://{s}.tile.openweathermap.org/map/{variant}/{z}/{x}/{y}.p
 'max_zoom': 19,
 'html_attribution': 'Map data &copy; <a href="http://openweathermap.org">OpenWeatherMap</
 'attribution': 'Map data (C) OpenWeatherMap',
 'apiKey': '<insert your api key here>',
 'opacity': 0.5,
 'variant': 'pressure_cntr',
 'name': 'OpenWeatherMap.PressureContour'},
'Wind': {'url': 'http://{s}.tile.openweathermap.org/map/{variant}/{z}/{x}/{y}.png?appid={a
 'max_zoom': 19,
 'html_attribution': 'Map data &copy; <a href="http://openweathermap.org">OpenWeatherMap</
 'attribution': 'Map data (C) OpenWeatherMap',
 'apiKey': '<insert your api key here>',
 'opacity': 0.5,
 'variant': 'wind',
 'name': 'OpenWeatherMap.Wind'},
'Temperature': {'url': 'http://{s}.tile.openweathermap.org/map/{variant}/{z}/{x}/{y}.png?a
 'max_zoom': 19,
 'html_attribution': 'Map data &copy; <a href="http://openweathermap.org">OpenWeatherMap</
 'attribution': 'Map data (C) OpenWeatherMap',
 'apiKey': '<insert your api key here>',
 'opacity': 0.5,
 'variant': 'temp',
 'name': 'OpenWeatherMap.Temperature'},
'Snow': {'url': 'http://{s}.tile.openweathermap.org/map/{variant}/{z}/{x}/{y}.png?appid={a
 'max_zoom': 19,
 'html_attribution': 'Map data &copy; <a href="http://openweathermap.org">OpenWeatherMap</
 'attribution': 'Map data (C) OpenWeatherMap',
 'apiKey': '<insert your api key here>',
 'opacity': 0.5,
```

```
  'variant': 'snow',
  'name': 'OpenWeatherMap.Snow'}},
'HERE': {'normalDay': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapI
  'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
  'attribution': 'Map (C) 1987-2023 HERE',
  'subdomains': '1234',
  'mapID': 'newest',
  'app_id': '<insert your app_id here>',
  'app_code': '<insert your app_code here>',
  'base': 'base',
  'variant': 'normal.day',
  'max_zoom': 20,
  'type': 'maptile',
  'language': 'eng',
  'format': 'png8',
  'size': '256',
  'name': 'HERE.normalDay'},
 'normalDayCustom': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID
  'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
  'attribution': 'Map (C) 1987-2023 HERE',
  'subdomains': '1234',
  'mapID': 'newest',
  'app_id': '<insert your app_id here>',
  'app_code': '<insert your app_code here>',
  'base': 'base',
  'variant': 'normal.day.custom',
  'max_zoom': 20,
  'type': 'maptile',
  'language': 'eng',
  'format': 'png8',
  'size': '256',
  'name': 'HERE.normalDayCustom'},
 'normalDayGrey': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}/
  'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
  'attribution': 'Map (C) 1987-2023 HERE',
  'subdomains': '1234',
  'mapID': 'newest',
  'app_id': '<insert your app_id here>',
  'app_code': '<insert your app_code here>',
  'base': 'base',
  'variant': 'normal.day.grey',
  'max_zoom': 20,
  'type': 'maptile',
```

```
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.normalDayGrey'},
'normalDayMobile': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'normal.day.mobile',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.normalDayMobile'},
'normalDayGreyMobile': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{ma
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'normal.day.grey.mobile',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.normalDayGreyMobile'},
'normalDayTransit': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapI
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'normal.day.transit',
```

```
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.normalDayTransit'},
'normalDayTransitMobile': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'normal.day.transit.mobile',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.normalDayTransitMobile'},
'normalDayTraffic': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapI
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'traffic',
 'variant': 'normal.traffic.day',
 'max_zoom': 20,
 'type': 'traffictile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.normalDayTraffic'},
'normalNight': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}/{va
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
```

```
 'base': 'base',
 'variant': 'normal.night',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.normalNight'},
'normalNightMobile': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapI
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'normal.night.mobile',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.normalNightMobile'},
'normalNightGrey': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'normal.night.grey',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.normalNightGrey'},
'normalNightGreyMobile': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
```

```
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'normal.night.grey.mobile',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.normalNightGreyMobile'},
'normalNightTransit': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{ma
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'normal.night.transit',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.normalNightTransit'},
'normalNightTransitMobile': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{typ
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'normal.night.transit.mobile',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.normalNightTransitMobile'},
'reducedDay': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}/{va
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
```

```
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'reduced.day',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.reducedDay'},
'reducedNight': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}/{
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'reduced.night',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.reducedNight'},
'basicMap': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}/{varia
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'normal.day',
 'max_zoom': 20,
 'type': 'basetile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.basicMap'},
'mapLabels': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}/{vari
```

```
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'normal.day',
 'max_zoom': 20,
 'type': 'labeltile',
 'language': 'eng',
 'format': 'png',
 'size': '256',
 'name': 'HERE.mapLabels'},
'trafficFlow': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}/{va
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'traffic',
 'variant': 'normal.day',
 'max_zoom': 20,
 'type': 'flowtile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.trafficFlow'},
'carnavDayGrey': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}/
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'carnav.day.grey',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
```

```
 'name': 'HERE.carnavDayGrey'},
'hybridDay': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}/{vari
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'aerial',
 'variant': 'hybrid.day',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.hybridDay'},
'hybridDayMobile': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'aerial',
 'variant': 'hybrid.day.mobile',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.hybridDayMobile'},
'hybridDayTransit': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapI
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'aerial',
 'variant': 'hybrid.day.transit',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
```

```
 'format': 'png8',
 'size': '256',
 'name': 'HERE.hybridDayTransit'},
'hybridDayGrey': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}/
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'aerial',
 'variant': 'hybrid.grey.day',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.hybridDayGrey'},
'hybridDayTraffic': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapI
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'traffic',
 'variant': 'hybrid.traffic.day',
 'max_zoom': 20,
 'type': 'traffictile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.hybridDayTraffic'},
'pedestrianDay': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}/
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'pedestrian.day',
 'max_zoom': 20,
```

```
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.pedestrianDay'},
'pedestrianNight': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'base',
 'variant': 'pedestrian.night',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.pedestrianNight'},
'satelliteDay': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}/{
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'aerial',
 'variant': 'satellite.day',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HERE.satelliteDay'},
'terrainDay': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapID}/{va
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'app_id': '<insert your app_id here>',
 'app_code': '<insert your app_code here>',
 'base': 'aerial',
```

```
  'variant': 'terrain.day',
  'max_zoom': 20,
  'type': 'maptile',
  'language': 'eng',
  'format': 'png8',
  'size': '256',
  'name': 'HERE.terrainDay'},
 'terrainDayMobile': {'url': 'https://{s}.{base}.maps.api.here.com/maptile/2.1/{type}/{mapI
  'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
  'attribution': 'Map (C) 1987-2023 HERE',
  'subdomains': '1234',
  'mapID': 'newest',
  'app_id': '<insert your app_id here>',
  'app_code': '<insert your app_code here>',
  'base': 'aerial',
  'variant': 'terrain.day.mobile',
  'max_zoom': 20,
  'type': 'maptile',
  'language': 'eng',
  'format': 'png8',
  'size': '256',
  'name': 'HERE.terrainDayMobile'}},
'HEREv3': {'normalDay': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/
  'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
  'attribution': 'Map (C) 1987-2023 HERE',
  'subdomains': '1234',
  'mapID': 'newest',
  'apiKey': '<insert your apiKey here>',
  'base': 'base',
  'variant': 'normal.day',
  'max_zoom': 20,
  'type': 'maptile',
  'language': 'eng',
  'format': 'png8',
  'size': '256',
  'name': 'HEREv3.normalDay'},
 'normalDayCustom': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapI
  'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
  'attribution': 'Map (C) 1987-2023 HERE',
  'subdomains': '1234',
  'mapID': 'newest',
  'apiKey': '<insert your apiKey here>',
  'base': 'base',
```

```
 'variant': 'normal.day.custom',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.normalDayCustom'},
'normalDayGrey': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapID
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'normal.day.grey',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.normalDayGrey'},
'normalDayMobile': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapI
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'normal.day.mobile',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.normalDayMobile'},
'normalDayGreyMobile': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'normal.day.grey.mobile',
```

```
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.normalDayGreyMobile'},
'normalDayTransit': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{ma
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'normal.day.transit',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.normalDayTransit'},
'normalDayTransitMobile': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{typ
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'normal.day.transit.mobile',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.normalDayTransitMobile'},
'normalNight': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapID}/
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'normal.night',
 'max_zoom': 20,
```

```
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.normalNight'},
'normalNightMobile': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{ma
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'normal.night.mobile',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.normalNightMobile'},
'normalNightGrey': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapI
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'normal.night.grey',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.normalNightGrey'},
'normalNightGreyMobile': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'normal.night.grey.mobile',
 'max_zoom': 20,
 'type': 'maptile',
```

```
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.normalNightGreyMobile'},
'normalNightTransit': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'normal.night.transit',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.normalNightTransit'},
'normalNightTransitMobile': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{ty
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'normal.night.transit.mobile',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.normalNightTransitMobile'},
'reducedDay': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapID}/{v
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'reduced.day',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
```

```
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.reducedDay'},
'reducedNight': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapID}/
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'reduced.night',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.reducedNight'},
'basicMap': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapID}/{va
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'normal.day',
 'max_zoom': 20,
 'type': 'basetile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.basicMap'},
'mapLabels': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapID}/{v
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'normal.day',
 'max_zoom': 20,
 'type': 'labeltile',
 'language': 'eng',
 'format': 'png',
```

```
 'size': '256',
 'name': 'HEREv3.mapLabels'},
'trafficFlow': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapID}/
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'traffic',
 'variant': 'normal.day',
 'max_zoom': 20,
 'type': 'flowtile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.trafficFlow'},
'carnavDayGrey': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapID
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'carnav.day.grey',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.carnavDayGrey'},
'hybridDay': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapID}/{va
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'aerial',
 'variant': 'hybrid.day',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
```

```
 'name': 'HEREv3.hybridDay'},
'hybridDayMobile': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapI
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'aerial',
 'variant': 'hybrid.day.mobile',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.hybridDayMobile'},
'hybridDayTransit': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{map
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'aerial',
 'variant': 'hybrid.day.transit',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.hybridDayTransit'},
'hybridDayGrey': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapID}
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'aerial',
 'variant': 'hybrid.grey.day',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.hybridDayGrey'},
```

```
'pedestrianDay': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapID}
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'pedestrian.day',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.pedestrianDay'},
'pedestrianNight': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapI
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'base',
 'variant': 'pedestrian.night',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.pedestrianNight'},
'satelliteDay': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapID}/
 'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
 'attribution': 'Map (C) 1987-2023 HERE',
 'subdomains': '1234',
 'mapID': 'newest',
 'apiKey': '<insert your apiKey here>',
 'base': 'aerial',
 'variant': 'satellite.day',
 'max_zoom': 20,
 'type': 'maptile',
 'language': 'eng',
 'format': 'png8',
 'size': '256',
 'name': 'HEREv3.satelliteDay'},
'terrainDay': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{mapID}/{v
```

```
  'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
  'attribution': 'Map (C) 1987-2023 HERE',
  'subdomains': '1234',
  'mapID': 'newest',
  'apiKey': '<insert your apiKey here>',
  'base': 'aerial',
  'variant': 'terrain.day',
  'max_zoom': 20,
  'type': 'maptile',
  'language': 'eng',
  'format': 'png8',
  'size': '256',
  'name': 'HEREv3.terrainDay'},
 'terrainDayMobile': {'url': 'https://{s}.{base}.maps.ls.hereapi.com/maptile/2.1/{type}/{ma
  'html_attribution': 'Map &copy; 1987-2023 <a href="http://developer.here.com">HERE</a>',
  'attribution': 'Map (C) 1987-2023 HERE',
  'subdomains': '1234',
  'mapID': 'newest',
  'apiKey': '<insert your apiKey here>',
  'base': 'aerial',
  'variant': 'terrain.day.mobile',
  'max_zoom': 20,
  'type': 'maptile',
  'language': 'eng',
  'format': 'png8',
  'size': '256',
  'name': 'HEREv3.terrainDayMobile'}},
'FreeMapSK': {'url': 'https://{s}.freemap.sk/T/{z}/{x}/{y}.jpeg',
 'min_zoom': 8,
 'max_zoom': 16,
 'subdomains': 'abcd',
 'bounds': [[47.204642, 15.996093], [49.830896, 22.576904]],
 'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
 'attribution': '(C) OpenStreetMap contributors, visualization CC-By-SA 2.0 Freemap.sk',
 'name': 'FreeMapSK'},
'MtbMap': {'url': 'http://tile.mtbmap.cz/mtbmap_tiles/{z}/{x}/{y}.png',
 'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
 'attribution': '(C) OpenStreetMap contributors & USGS',
 'name': 'MtbMap'},
'CartoDB': {'Positron': {'url': 'https://{s}.basemaps.cartocdn.com/{variant}/{z}/{x}/{y}{r}
  'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
  'attribution': '(C) OpenStreetMap contributors (C) CARTO',
  'subdomains': 'abcd',
```

```
 'max_zoom': 20,
 'variant': 'light_all',
 'name': 'CartoDB.Positron'},
'PositronNoLabels': {'url': 'https://{s}.basemaps.cartocdn.com/{variant}/{z}/{x}/{y}{r}.png
 'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
 'attribution': '(C) OpenStreetMap contributors (C) CARTO',
 'subdomains': 'abcd',
 'max_zoom': 20,
 'variant': 'light_nolabels',
 'name': 'CartoDB.PositronNoLabels'},
'PositronOnlyLabels': {'url': 'https://{s}.basemaps.cartocdn.com/{variant}/{z}/{x}/{y}{r}.p
 'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
 'attribution': '(C) OpenStreetMap contributors (C) CARTO',
 'subdomains': 'abcd',
 'max_zoom': 20,
 'variant': 'light_only_labels',
 'name': 'CartoDB.PositronOnlyLabels'},
'DarkMatter': {'url': 'https://{s}.basemaps.cartocdn.com/{variant}/{z}/{x}/{y}{r}.png',
 'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
 'attribution': '(C) OpenStreetMap contributors (C) CARTO',
 'subdomains': 'abcd',
 'max_zoom': 20,
 'variant': 'dark_all',
 'name': 'CartoDB.DarkMatter'},
'DarkMatterNoLabels': {'url': 'https://{s}.basemaps.cartocdn.com/{variant}/{z}/{x}/{y}{r}.p
 'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
 'attribution': '(C) OpenStreetMap contributors (C) CARTO',
 'subdomains': 'abcd',
 'max_zoom': 20,
 'variant': 'dark_nolabels',
 'name': 'CartoDB.DarkMatterNoLabels'},
'DarkMatterOnlyLabels': {'url': 'https://{s}.basemaps.cartocdn.com/{variant}/{z}/{x}/{y}{r
 'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
 'attribution': '(C) OpenStreetMap contributors (C) CARTO',
 'subdomains': 'abcd',
 'max_zoom': 20,
 'variant': 'dark_only_labels',
 'name': 'CartoDB.DarkMatterOnlyLabels'},
'Voyager': {'url': 'https://{s}.basemaps.cartocdn.com/{variant}/{z}/{x}/{y}{r}.png',
 'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
 'attribution': '(C) OpenStreetMap contributors (C) CARTO',
 'subdomains': 'abcd',
 'max_zoom': 20,
```

```
  'variant': 'rastertiles/voyager',
  'name': 'CartoDB.Voyager'},
 'VoyagerNoLabels': {'url': 'https://{s}.basemaps.cartocdn.com/{variant}/{z}/{x}/{y}{r}.png
  'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
  'attribution': '(C) OpenStreetMap contributors (C) CARTO',
  'subdomains': 'abcd',
  'max_zoom': 20,
  'variant': 'rastertiles/voyager_nolabels',
  'name': 'CartoDB.VoyagerNoLabels'},
 'VoyagerOnlyLabels': {'url': 'https://{s}.basemaps.cartocdn.com/{variant}/{z}/{x}/{y}{r}.pn
  'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
  'attribution': '(C) OpenStreetMap contributors (C) CARTO',
  'subdomains': 'abcd',
  'max_zoom': 20,
  'variant': 'rastertiles/voyager_only_labels',
  'name': 'CartoDB.VoyagerOnlyLabels'},
 'VoyagerLabelsUnder': {'url': 'https://{s}.basemaps.cartocdn.com/{variant}/{z}/{x}/{y}{r}.p
  'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
  'attribution': '(C) OpenStreetMap contributors (C) CARTO',
  'subdomains': 'abcd',
  'max_zoom': 20,
  'variant': 'rastertiles/voyager_labels_under',
  'name': 'CartoDB.VoyagerLabelsUnder'}},
'HikeBike': {'HikeBike': {'url': 'https://tiles.wmflabs.org/{variant}/{z}/{x}/{y}.png',
  'max_zoom': 19,
  'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
  'attribution': '(C) OpenStreetMap contributors',
  'variant': 'hikebike',
  'name': 'HikeBike.HikeBike'},
 'HillShading': {'url': 'https://tiles.wmflabs.org/{variant}/{z}/{x}/{y}.png',
  'max_zoom': 15,
  'html_attribution': '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMa
  'attribution': '(C) OpenStreetMap contributors',
  'variant': 'hillshading',
  'name': 'HikeBike.HillShading'}},
'BasemapAT': {'basemap': {'url': 'https://mapsneu.wien.gv.at/basemap/{variant}/{type}/google
  'max_zoom': 20,
  'html_attribution': 'Datenquelle: <a href="https://www.basemap.at">basemap.at</a>',
  'attribution': 'Datenquelle: basemap.at',
  'type': 'normal',
  'format': 'png',
  'bounds': [[46.35877, 8.782379], [49.037872, 17.189532]],
  'variant': 'geolandbasemap',
```

```
 'name': 'BasemapAT.basemap'},
'grau': {'url': 'https://mapsneu.wien.gv.at/basemap/{variant}/{type}/google3857/{z}/{y}/{x}
 'max_zoom': 19,
 'html_attribution': 'Datenquelle: <a href="https://www.basemap.at">basemap.at</a>',
 'attribution': 'Datenquelle: basemap.at',
 'type': 'normal',
 'format': 'png',
 'bounds': [[46.35877, 8.782379], [49.037872, 17.189532]],
 'variant': 'bmapgrau',
 'name': 'BasemapAT.grau'},
'overlay': {'url': 'https://mapsneu.wien.gv.at/basemap/{variant}/{type}/google3857/{z}/{y}/
 'max_zoom': 19,
 'html_attribution': 'Datenquelle: <a href="https://www.basemap.at">basemap.at</a>',
 'attribution': 'Datenquelle: basemap.at',
 'type': 'normal',
 'format': 'png',
 'bounds': [[46.35877, 8.782379], [49.037872, 17.189532]],
 'variant': 'bmapoverlay',
 'name': 'BasemapAT.overlay'},
'terrain': {'url': 'https://mapsneu.wien.gv.at/basemap/{variant}/{type}/google3857/{z}/{y}/
 'max_zoom': 19,
 'html_attribution': 'Datenquelle: <a href="https://www.basemap.at">basemap.at</a>',
 'attribution': 'Datenquelle: basemap.at',
 'type': 'grau',
 'format': 'jpeg',
 'bounds': [[46.35877, 8.782379], [49.037872, 17.189532]],
 'variant': 'bmapgelaende',
 'name': 'BasemapAT.terrain'},
'surface': {'url': 'https://mapsneu.wien.gv.at/basemap/{variant}/{type}/google3857/{z}/{y}/
 'max_zoom': 19,
 'html_attribution': 'Datenquelle: <a href="https://www.basemap.at">basemap.at</a>',
 'attribution': 'Datenquelle: basemap.at',
 'type': 'grau',
 'format': 'jpeg',
 'bounds': [[46.35877, 8.782379], [49.037872, 17.189532]],
 'variant': 'bmapoberflaeche',
 'name': 'BasemapAT.surface'},
'highdpi': {'url': 'https://mapsneu.wien.gv.at/basemap/{variant}/{type}/google3857/{z}/{y}/
 'max_zoom': 19,
 'html_attribution': 'Datenquelle: <a href="https://www.basemap.at">basemap.at</a>',
 'attribution': 'Datenquelle: basemap.at',
 'type': 'normal',
 'format': 'jpeg',
```

```
  'bounds': [[46.35877, 8.782379], [49.037872, 17.189532]],
  'variant': 'bmaphidpi',
  'name': 'BasemapAT.highdpi'},
 'orthofoto': {'url': 'https://mapsneu.wien.gv.at/basemap/{variant}/{type}/google3857/{z}/{y
  'max_zoom': 20,
  'html_attribution': 'Datenquelle: <a href="https://www.basemap.at">basemap.at</a>',
  'attribution': 'Datenquelle: basemap.at',
  'type': 'normal',
  'format': 'jpeg',
  'bounds': [[46.35877, 8.782379], [49.037872, 17.189532]],
  'variant': 'bmaporthofoto30cm',
  'name': 'BasemapAT.orthofoto'}},
'nlmaps': {'standaard': {'url': 'https://service.pdok.nl/brt/achtergrondkaart/wmts/v2_0/{var
  'min_zoom': 6,
  'max_zoom': 19,
  'bounds': [[50.5, 3.25], [54, 7.6]],
  'html_attribution': 'Kaartgegevens &copy; <a href="https://www.kadaster.nl">Kadaster</a>'
  'attribution': 'Kaartgegevens (C) Kadaster',
  'variant': 'standaard',
  'name': 'nlmaps.standaard'},
 'pastel': {'url': 'https://service.pdok.nl/brt/achtergrondkaart/wmts/v2_0/{variant}/EPSG:38
  'min_zoom': 6,
  'max_zoom': 19,
  'bounds': [[50.5, 3.25], [54, 7.6]],
  'html_attribution': 'Kaartgegevens &copy; <a href="https://www.kadaster.nl">Kadaster</a>'
  'attribution': 'Kaartgegevens (C) Kadaster',
  'variant': 'pastel',
  'name': 'nlmaps.pastel'},
 'grijs': {'url': 'https://service.pdok.nl/brt/achtergrondkaart/wmts/v2_0/{variant}/EPSG:385
  'min_zoom': 6,
  'max_zoom': 19,
  'bounds': [[50.5, 3.25], [54, 7.6]],
  'html_attribution': 'Kaartgegevens &copy; <a href="https://www.kadaster.nl">Kadaster</a>'
  'attribution': 'Kaartgegevens (C) Kadaster',
  'variant': 'grijs',
  'name': 'nlmaps.grijs'},
 'water': {'url': 'https://service.pdok.nl/brt/achtergrondkaart/wmts/v2_0/{variant}/EPSG:385
  'min_zoom': 6,
  'max_zoom': 19,
  'bounds': [[50.5, 3.25], [54, 7.6]],
  'html_attribution': 'Kaartgegevens &copy; <a href="https://www.kadaster.nl">Kadaster</a>'
  'attribution': 'Kaartgegevens (C) Kadaster',
  'variant': 'water',
```

```
  'name': 'nlmaps.water'},
 'luchtfoto': {'url': 'https://service.pdok.nl/hwh/luchtfotorgb/wmts/v1_0/Actueel_ortho25/E
  'min_zoom': 6,
  'max_zoom': 19,
  'bounds': [[50.5, 3.25], [54, 7.6]],
  'html_attribution': 'Kaartgegevens &copy; <a href="https://www.kadaster.nl">Kadaster</a>'
  'attribution': 'Kaartgegevens (C) Kadaster',
  'name': 'nlmaps.luchtfoto'}},
'NASAGIBS': {'ModisTerraTrueColorCR': {'url': 'https://map1.vis.earthdata.nasa.gov/wmts-webm
  'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
  'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIBS
  'bounds': [[-85.0511287776, -179.999999975],
   [85.0511287776, 179.999999975]],
  'min_zoom': 1,
  'max_zoom': 9,
  'format': 'jpg',
  'time': '',
  'tilematrixset': 'GoogleMapsCompatible_Level',
  'variant': 'MODIS_Terra_CorrectedReflectance_TrueColor',
  'name': 'NASAGIBS.ModisTerraTrueColorCR'},
 'ModisTerraBands367CR': {'url': 'https://map1.vis.earthdata.nasa.gov/wmts-webmerc/{variant}
  'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
  'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIBS
  'bounds': [[-85.0511287776, -179.999999975],
   [85.0511287776, 179.999999975]],
  'min_zoom': 1,
  'max_zoom': 9,
  'format': 'jpg',
  'time': '',
  'tilematrixset': 'GoogleMapsCompatible_Level',
  'variant': 'MODIS_Terra_CorrectedReflectance_Bands367',
  'name': 'NASAGIBS.ModisTerraBands367CR'},
 'ViirsEarthAtNight2012': {'url': 'https://map1.vis.earthdata.nasa.gov/wmts-webmerc/{variant
  'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
  'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIBS
  'bounds': [[-85.0511287776, -179.999999975],
   [85.0511287776, 179.999999975]],
  'min_zoom': 1,
  'max_zoom': 8,
  'format': 'jpg',
  'time': '',
  'tilematrixset': 'GoogleMapsCompatible_Level',
  'variant': 'VIIRS_CityLights_2012',
```

```
 'name': 'NASAGIBS.ViirsEarthAtNight2012'},
'ModisTerraLSTDay': {'url': 'https://map1.vis.earthdata.nasa.gov/wmts-webmerc/{variant}/de
 'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
 'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIBS
 'bounds': [[-85.0511287776, -179.999999975],
  [85.0511287776, 179.999999975]],
 'min_zoom': 1,
 'max_zoom': 7,
 'format': 'png',
 'time': '',
 'tilematrixset': 'GoogleMapsCompatible_Level',
 'variant': 'MODIS_Terra_Land_Surface_Temp_Day',
 'opacity': 0.75,
 'name': 'NASAGIBS.ModisTerraLSTDay'},
'ModisTerraSnowCover': {'url': 'https://map1.vis.earthdata.nasa.gov/wmts-webmerc/{variant}/
 'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
 'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIBS
 'bounds': [[-85.0511287776, -179.999999975],
  [85.0511287776, 179.999999975]],
 'min_zoom': 1,
 'max_zoom': 8,
 'format': 'png',
 'time': '',
 'tilematrixset': 'GoogleMapsCompatible_Level',
 'variant': 'MODIS_Terra_NDSI_Snow_Cover',
 'opacity': 0.75,
 'name': 'NASAGIBS.ModisTerraSnowCover'},
'ModisTerraAOD': {'url': 'https://map1.vis.earthdata.nasa.gov/wmts-webmerc/{variant}/defaul
 'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
 'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIBS
 'bounds': [[-85.0511287776, -179.999999975],
  [85.0511287776, 179.999999975]],
 'min_zoom': 1,
 'max_zoom': 6,
 'format': 'png',
 'time': '',
 'tilematrixset': 'GoogleMapsCompatible_Level',
 'variant': 'MODIS_Terra_Aerosol',
 'opacity': 0.75,
 'name': 'NASAGIBS.ModisTerraAOD'},
'ModisTerraChlorophyll': {'url': 'https://map1.vis.earthdata.nasa.gov/wmts-webmerc/{varian
 'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
 'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIBS
```

```
 'bounds': [[-85.0511287776, -179.999999975],
  [85.0511287776, 179.999999975]],
 'min_zoom': 1,
 'max_zoom': 7,
 'format': 'png',
 'time': '',
 'tilematrixset': 'GoogleMapsCompatible_Level',
 'variant': 'MODIS_Terra_Chlorophyll_A',
 'opacity': 0.75,
 'name': 'NASAGIBS.ModisTerraChlorophyll'},
'ModisTerraBands721CR': {'url': 'https://gibs.earthdata.nasa.gov/wmts/epsg3857/best/MODIS_
 'max_zoom': 9,
 'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIBS
 'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
 'name': 'NASAGIBS.ModisTerraBands721CR',
 'time': ''},
'ModisAquaTrueColorCR': {'url': 'https://gibs.earthdata.nasa.gov/wmts/epsg3857/best/MODIS_A
 'max_zoom': 9,
 'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIBS
 'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
 'name': 'NASAGIBS.ModisAquaTrueColorCR',
 'time': ''},
'ModisAquaBands721CR': {'url': 'https://gibs.earthdata.nasa.gov/wmts/epsg3857/best/MODIS_Aq
 'max_zoom': 9,
 'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIBS
 'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
 'name': 'NASAGIBS.ModisAquaBands721CR',
 'time': ''},
'ViirsTrueColorCR': {'url': 'https://gibs.earthdata.nasa.gov/wmts/epsg3857/best/VIIRS_SNPP_
 'max_zoom': 9,
 'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIBS
 'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
 'name': 'NASAGIBS.ViirsTrueColorCR',
 'time': ''},
'BlueMarble3413': {'url': 'https://gibs.earthdata.nasa.gov/wmts/epsg3413/best/BlueMarble_Ne
 'max_zoom': 5,
 'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIBS
 'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
 'name': 'NASAGIBS.BlueMarble3413',
 'crs': 'EPSG:3413'},
'BlueMarble3031': {'url': 'https://gibs.earthdata.nasa.gov/wmts/epsg3031/best/BlueMarble_Ne
 'max_zoom': 5,
 'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIBS
```

```
  'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
  'name': 'NASAGIBS.BlueMarble3031',
  'crs': 'EPSG:3031'},
 'BlueMarble': {'url': 'https://gibs.earthdata.nasa.gov/wmts/epsg3857/best/BlueMarble_NextG
  'max_zoom': 8,
  'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIB
  'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
  'name': 'NASAGIBS.BlueMarble'},
 'ASTER_GDEM_Greyscale_Shaded_Relief': {'url': 'https://gibs.earthdata.nasa.gov/wmts/epsg385
  'max_zoom': 12,
  'attribution': 'Imagery provided by services from the Global Imagery Browse Services (GIB
  'html_attribution': 'Imagery provided by services from the Global Imagery Browse Services
  'name': 'NASAGIBS.ASTER_GDEM_Greyscale_Shaded_Relief'}},
'NLS': {'url': 'https://nls-{s}.tileserver.com/nls/{z}/{x}/{y}.jpg',
 'html_attribution': '<a href="http://geo.nls.uk/maps/">National Library of Scotland Histori
 'attribution': 'National Library of Scotland Historic Maps',
 'bounds': [[49.6, -12], [61.7, 3]],
 'min_zoom': 1,
 'max_zoom': 18,
 'subdomains': '0123',
 'name': 'NLS',
 'status': 'broken'},
'JusticeMap': {'income': {'url': 'https://www.justicemap.org/tile/{size}/{variant}/{z}/{x}/
  'html_attribution': '<a href="http://www.justicemap.org/terms.php">Justice Map</a>',
  'attribution': 'Justice Map',
  'size': 'county',
  'bounds': [[14, -180], [72, -56]],
  'variant': 'income',
  'name': 'JusticeMap.income',
  'status': 'broken'},
 'americanIndian': {'url': 'https://www.justicemap.org/tile/{size}/{variant}/{z}/{x}/{y}.png
  'html_attribution': '<a href="http://www.justicemap.org/terms.php">Justice Map</a>',
  'attribution': 'Justice Map',
  'size': 'county',
  'bounds': [[14, -180], [72, -56]],
  'variant': 'indian',
  'name': 'JusticeMap.americanIndian',
  'status': 'broken'},
 'asian': {'url': 'https://www.justicemap.org/tile/{size}/{variant}/{z}/{x}/{y}.png',
  'html_attribution': '<a href="http://www.justicemap.org/terms.php">Justice Map</a>',
  'attribution': 'Justice Map',
  'size': 'county',
  'bounds': [[14, -180], [72, -56]],
```

```
 'variant': 'asian',
 'name': 'JusticeMap.asian',
 'status': 'broken'},
'black': {'url': 'https://www.justicemap.org/tile/{size}/{variant}/{z}/{x}/{y}.png',
 'html_attribution': '<a href="http://www.justicemap.org/terms.php">Justice Map</a>',
 'attribution': 'Justice Map',
 'size': 'county',
 'bounds': [[14, -180], [72, -56]],
 'variant': 'black',
 'name': 'JusticeMap.black',
 'status': 'broken'},
'hispanic': {'url': 'https://www.justicemap.org/tile/{size}/{variant}/{z}/{x}/{y}.png',
 'html_attribution': '<a href="http://www.justicemap.org/terms.php">Justice Map</a>',
 'attribution': 'Justice Map',
 'size': 'county',
 'bounds': [[14, -180], [72, -56]],
 'variant': 'hispanic',
 'name': 'JusticeMap.hispanic',
 'status': 'broken'},
'multi': {'url': 'https://www.justicemap.org/tile/{size}/{variant}/{z}/{x}/{y}.png',
 'html_attribution': '<a href="http://www.justicemap.org/terms.php">Justice Map</a>',
 'attribution': 'Justice Map',
 'size': 'county',
 'bounds': [[14, -180], [72, -56]],
 'variant': 'multi',
 'name': 'JusticeMap.multi',
 'status': 'broken'},
'nonWhite': {'url': 'https://www.justicemap.org/tile/{size}/{variant}/{z}/{x}/{y}.png',
 'html_attribution': '<a href="http://www.justicemap.org/terms.php">Justice Map</a>',
 'attribution': 'Justice Map',
 'size': 'county',
 'bounds': [[14, -180], [72, -56]],
 'variant': 'nonwhite',
 'name': 'JusticeMap.nonWhite',
 'status': 'broken'},
'white': {'url': 'https://www.justicemap.org/tile/{size}/{variant}/{z}/{x}/{y}.png',
 'html_attribution': '<a href="http://www.justicemap.org/terms.php">Justice Map</a>',
 'attribution': 'Justice Map',
 'size': 'county',
 'bounds': [[14, -180], [72, -56]],
 'variant': 'white',
 'name': 'JusticeMap.white',
 'status': 'broken'},
```

```
 'plurality': {'url': 'https://www.justicemap.org/tile/{size}/{variant}/{z}/{x}/{y}.png',
  'html_attribution': '<a href="http://www.justicemap.org/terms.php">Justice Map</a>',
  'attribution': 'Justice Map',
  'size': 'county',
  'bounds': [[14, -180], [72, -56]],
  'variant': 'plural',
  'name': 'JusticeMap.plurality',
  'status': 'broken'}},
'GeoportailFrance': {'plan': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
  'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
  'attribution': 'Geoportail France',
  'bounds': [[-85.0, -175.0], [85.0, 175.0]],
  'min_zoom': 0,
  'max_zoom': 19,
  'apikey': 'essentiels',
  'format': 'image/png',
  'style': 'normal',
  'variant': 'GEOGRAPHICALGRIDSYSTEMS.PLANIGNV2',
  'name': 'GeoportailFrance.plan',
  'TileMatrixSet': 'PM'},
 'parcels': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SERVICE=WMT
  'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
  'attribution': 'Geoportail France',
  'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
  'min_zoom': 0,
  'max_zoom': 19,
  'apikey': 'essentiels',
  'format': 'image/png',
  'style': 'normal',
  'variant': 'CADASTRALPARCELS.PARCELLAIRE_EXPRESS',
  'name': 'GeoportailFrance.parcels',
  'TileMatrixSet': 'PM'},
 'orthos': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SERVICE=WMTS
  'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
  'attribution': 'Geoportail France',
  'bounds': [[-75.0, -179.5], [75.0, 179.5]],
  'min_zoom': 0,
  'max_zoom': 21,
  'apikey': 'ortho',
  'format': 'image/jpeg',
  'style': 'normal',
  'variant': 'ORTHOIMAGERY.ORTHOPHOTOS',
  'name': 'GeoportailFrance.orthos',
```

```
 'TileMatrixSet': 'PM'},
'Adminexpress_cog_carto_Latest': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQU
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'administratif',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ADMINEXPRESS-COG-CARTO.LATEST',
 'name': 'GeoportailFrance.Adminexpress_cog_carto_Latest',
 'TileMatrixSet': 'PM'},
'Adminexpress_cog_Latest': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'administratif',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ADMINEXPRESS-COG.LATEST',
 'name': 'GeoportailFrance.Adminexpress_cog_Latest',
 'TileMatrixSet': 'PM'},
'Limites_administratives_express_Latest': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'administratif',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LIMITES_ADMINISTRATIVES_EXPRESS.LATEST',
 'name': 'GeoportailFrance.Limites_administratives_express_Latest',
 'TileMatrixSet': 'PM'},
'Geographicalgridsystems_Slopes_Pac': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.5446, -63.1614], [51.0991, 56.0018]],
 'min_zoom': 0,
 'max_zoom': 15,
```

```
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'GEOGRAPHICALGRIDSYSTEMS.SLOPES.PAC',
 'name': 'GeoportailFrance.Geographicalgridsystems_Slopes_Pac',
 'TileMatrixSet': 'PM'},
'Hydrography_Bcae_Latest': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 17,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'HYDROGRAPHY.BCAE.LATEST',
 'name': 'GeoportailFrance.Hydrography_Bcae_Latest',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture_Latest': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE.LATEST',
 'name': 'GeoportailFrance.Landuse_Agriculture_Latest',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2007': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.419, -63.2635], [51.2203, 56.0237]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2007',
 'name': 'GeoportailFrance.Landuse_Agriculture2007',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2008': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
```

```
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.419, -63.2635], [51.2203, 56.0237]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2008',
 'name': 'GeoportailFrance.Landuse_Agriculture2008',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2009': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.419, -63.2635], [51.2203, 56.0237]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2009',
 'name': 'GeoportailFrance.Landuse_Agriculture2009',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2010': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2010',
 'name': 'GeoportailFrance.Landuse_Agriculture2010',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2011': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
```

```
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2011',
 'name': 'GeoportailFrance.Landuse_Agriculture2011',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2012': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2012',
 'name': 'GeoportailFrance.Landuse_Agriculture2012',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2013': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2013',
 'name': 'GeoportailFrance.Landuse_Agriculture2013',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2014': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2014',
 'name': 'GeoportailFrance.Landuse_Agriculture2014',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2015': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
```

```
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2015',
 'name': 'GeoportailFrance.Landuse_Agriculture2015',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2016': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2016',
 'name': 'GeoportailFrance.Landuse_Agriculture2016',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2017': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2017',
 'name': 'GeoportailFrance.Landuse_Agriculture2017',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2018': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2018',
```

```
 'name': 'GeoportailFrance.Landuse_Agriculture2018',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2019': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2019',
 'name': 'GeoportailFrance.Landuse_Agriculture2019',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2020': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2020',
 'name': 'GeoportailFrance.Landuse_Agriculture2020',
 'TileMatrixSet': 'PM'},
'Landuse_Agriculture2021': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDUSE.AGRICULTURE2021',
 'name': 'GeoportailFrance.Landuse_Agriculture2021',
 'TileMatrixSet': 'PM'},
'Prairies_Sensibles_Bcae': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
```

```
 'max_zoom': 16,
 'apikey': 'agriculture',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'PRAIRIES.SENSIBLES.BCAE',
 'name': 'GeoportailFrance.Prairies_Sensibles_Bcae',
 'TileMatrixSet': 'PM'},
'Elevation_Contour_Line': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Get
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'altimetrie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ELEVATION.CONTOUR.LINE',
 'name': 'GeoportailFrance.Elevation_Contour_Line',
 'TileMatrixSet': 'PM'},
'Elevation_Elevationgridcoverage_Shadow': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4069, -63.187], [50.9218, 55.8884]],
 'min_zoom': 0,
 'max_zoom': 15,
 'apikey': 'altimetrie',
 'format': 'image/png',
 'style': 'estompage_grayscale',
 'variant': 'ELEVATION.ELEVATIONGRIDCOVERAGE.SHADOW',
 'name': 'GeoportailFrance.Elevation_Elevationgridcoverage_Shadow',
 'TileMatrixSet': 'PM'},
'Elevation_Elevationgridcoverage_Threshold': {'url': 'https://wxs.ign.fr/{apikey}/geoporta
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 3,
 'max_zoom': 17,
 'apikey': 'altimetrie',
 'format': 'image/png',
 'style': 'ELEVATION.ELEVATIONGRIDCOVERAGE.THRESHOLD',
 'variant': 'ELEVATION.ELEVATIONGRIDCOVERAGE.THRESHOLD',
 'name': 'GeoportailFrance.Elevation_Elevationgridcoverage_Threshold',
 'TileMatrixSet': 'PM'},
```

```
'Elevation_Level0': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.51, -63.2529], [51.1388, 55.9472]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'altimetrie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ELEVATION.LEVEL0',
 'name': 'GeoportailFrance.Elevation_Level0',
 'TileMatrixSet': 'PM'},
'Elevation_Slopes': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-22.5952, -178.206], [50.9308, 167.432]],
 'min_zoom': 6,
 'max_zoom': 14,
 'apikey': 'altimetrie',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ELEVATION.SLOPES',
 'name': 'GeoportailFrance.Elevation_Slopes',
 'TileMatrixSet': 'PM'},
'Elevationgridcoverage_Highres_Quality': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wm
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'altimetrie',
 'format': 'image/png',
 'style': 'Graphe de source du RGE Alti',
 'variant': 'ELEVATIONGRIDCOVERAGE.HIGHRES.QUALITY',
 'name': 'GeoportailFrance.Elevationgridcoverage_Highres_Quality',
 'TileMatrixSet': 'PM'},
'Geographicalgridsystems_Slopes_Mountain': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.5446, -63.1614], [51.0991, 56.0018]],
 'min_zoom': 0,
 'max_zoom': 17,
 'apikey': 'altimetrie',
```

```
 'format': 'image/png',
 'style': 'normal',
 'variant': 'GEOGRAPHICALGRIDSYSTEMS.SLOPES.MOUNTAIN',
 'name': 'GeoportailFrance.Geographicalgridsystems_Slopes_Mountain',
 'TileMatrixSet': 'PM'},
'Geographicalgridsystems_1900typemaps': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wm
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[48.4726, 1.62941], [49.1548, 3.0]],
 'min_zoom': 10,
 'max_zoom': 15,
 'apikey': 'cartes',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'GEOGRAPHICALGRIDSYSTEMS.1900TYPEMAPS',
 'name': 'GeoportailFrance.Geographicalgridsystems_1900typemaps',
 'TileMatrixSet': 'PM'},
'Geographicalgridsystems_Bonne': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQU
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-0.49941, -55.9127], [7.88966, -50.0835]],
 'min_zoom': 0,
 'max_zoom': 10,
 'apikey': 'cartes',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'GEOGRAPHICALGRIDSYSTEMS.BONNE',
 'name': 'GeoportailFrance.Geographicalgridsystems_Bonne',
 'TileMatrixSet': 'PM'},
'Geographicalgridsystems_Etatmajor10': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[48.3847, 1.82682], [49.5142, 2.79738]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'cartes',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'GEOGRAPHICALGRIDSYSTEMS.ETATMAJOR10',
 'name': 'GeoportailFrance.Geographicalgridsystems_Etatmajor10',
 'TileMatrixSet': 'PM'},
'Geographicalgridsystems_Etatmajor40': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
```

```
 'attribution': 'Geoportail France',
 'bounds': [[41.1844, -6.08889], [51.2745, 10.961]],
 'min_zoom': 6,
 'max_zoom': 15,
 'apikey': 'cartes',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'GEOGRAPHICALGRIDSYSTEMS.ETATMAJOR40',
 'name': 'GeoportailFrance.Geographicalgridsystems_Etatmajor40',
 'TileMatrixSet': 'PM'},
'Geographicalgridsystems_Maps_Bduni_J1': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wm
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'cartes',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'GEOGRAPHICALGRIDSYSTEMS.MAPS.BDUNI.J1',
 'name': 'GeoportailFrance.Geographicalgridsystems_Maps_Bduni_J1',
 'TileMatrixSet': 'PM'},
'Geographicalgridsystems_Maps_Overview': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wm
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 1,
 'max_zoom': 8,
 'apikey': 'cartes',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'GEOGRAPHICALGRIDSYSTEMS.MAPS.OVERVIEW',
 'name': 'GeoportailFrance.Geographicalgridsystems_Maps_Overview',
 'TileMatrixSet': 'PM'},
'Geographicalgridsystems_Maps_Scan50_1950': {'url': 'https://wxs.ign.fr/{apikey}/geoportail
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 3,
 'max_zoom': 15,
 'apikey': 'cartes',
 'format': 'image/jpeg',
 'style': 'SCAN50_1950',
```

```
 'variant': 'GEOGRAPHICALGRIDSYSTEMS.MAPS.SCAN50.1950',
 'name': 'GeoportailFrance.Geographicalgridsystems_Maps_Scan50_1950',
 'TileMatrixSet': 'PM'},
'Geographicalgridsystems_Terrier_v1': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts'
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.2568, 8.36284], [43.1174, 9.75281]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'cartes',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'GEOGRAPHICALGRIDSYSTEMS.TERRIER_V1',
 'name': 'GeoportailFrance.Geographicalgridsystems_Terrier_v1',
 'TileMatrixSet': 'PM'},
'Geographicalgridsystems_Terrier_v2': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts'
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.2568, 8.36284], [43.1174, 9.75282]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'cartes',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'GEOGRAPHICALGRIDSYSTEMS.TERRIER_V2',
 'name': 'GeoportailFrance.Geographicalgridsystems_Terrier_v2',
 'TileMatrixSet': 'PM'},
'Landcover_Cha00_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[40.576, -9.88147], [51.4428, 11.6781]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.CHA00_FR',
 'name': 'GeoportailFrance.Landcover_Cha00_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Cha06_dom': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [47.1747, 55.9259]],
```

```
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - DOM',
 'variant': 'LANDCOVER.CHA06_DOM',
 'name': 'GeoportailFrance.Landcover_Cha06_dom',
 'TileMatrixSet': 'PM'},
'Landcover_Cha06_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[40.576, -9.88147], [51.4428, 11.6781]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.CHA06_FR',
 'name': 'GeoportailFrance.Landcover_Cha06_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Cha12_dom': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [47.1747, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - DOM',
 'variant': 'LANDCOVER.CHA12_DOM',
 'name': 'GeoportailFrance.Landcover_Cha12_dom',
 'TileMatrixSet': 'PM'},
'Landcover_Cha12_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[40.576, -9.88147], [51.4428, 11.6781]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.CHA12_FR',
 'name': 'GeoportailFrance.Landcover_Cha12_fr',
```

```
 'TileMatrixSet': 'PM'},
'Landcover_Cha18': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.4428, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover',
 'variant': 'LANDCOVER.CHA18',
 'name': 'GeoportailFrance.Landcover_Cha18',
 'TileMatrixSet': 'PM'},
'Landcover_Cha18_dom': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTil
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [47.1747, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - DOM',
 'variant': 'LANDCOVER.CHA18_DOM',
 'name': 'GeoportailFrance.Landcover_Cha18_dom',
 'TileMatrixSet': 'PM'},
'Landcover_Cha18_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[40.576, -9.88147], [51.4428, 11.6781]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.CHA18_FR',
 'name': 'GeoportailFrance.Landcover_Cha18_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Clc00r_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTil
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[40.576, -9.88147], [51.4428, 11.6781]],
 'min_zoom': 0,
 'max_zoom': 16,
```

```
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.CLC00R_FR',
 'name': 'GeoportailFrance.Landcover_Clc00r_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Clc00_dom': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [47.1747, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - DOM',
 'variant': 'LANDCOVER.CLC00_DOM',
 'name': 'GeoportailFrance.Landcover_Clc00_dom',
 'TileMatrixSet': 'PM'},
'Landcover_Clc00_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[40.576, -9.88147], [51.4428, 11.6781]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.CLC00_FR',
 'name': 'GeoportailFrance.Landcover_Clc00_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Clc06r_dom': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTil
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [47.1747, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - DOM',
 'variant': 'LANDCOVER.CLC06R_DOM',
 'name': 'GeoportailFrance.Landcover_Clc06r_dom',
 'TileMatrixSet': 'PM'},
'Landcover_Clc06r_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
```

```
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[40.576, -9.88147], [51.4428, 11.6781]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.CLC06R_FR',
 'name': 'GeoportailFrance.Landcover_Clc06r_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Clc06_dom': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTil
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [47.1747, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - DOM',
 'variant': 'LANDCOVER.CLC06_DOM',
 'name': 'GeoportailFrance.Landcover_Clc06_dom',
 'TileMatrixSet': 'PM'},
'Landcover_Clc06_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[40.576, -9.88147], [51.4428, 11.6781]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.CLC06_FR',
 'name': 'GeoportailFrance.Landcover_Clc06_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Clc12': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.4428, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
```

```
 'style': 'CORINE Land Cover - DOM',
 'variant': 'LANDCOVER.CLC12',
 'name': 'GeoportailFrance.Landcover_Clc12',
 'TileMatrixSet': 'PM'},
'Landcover_Clc12r': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.4428, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover',
 'variant': 'LANDCOVER.CLC12R',
 'name': 'GeoportailFrance.Landcover_Clc12r',
 'TileMatrixSet': 'PM'},
'Landcover_Clc12r_dom': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTil
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [47.1747, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - DOM',
 'variant': 'LANDCOVER.CLC12R_DOM',
 'name': 'GeoportailFrance.Landcover_Clc12r_dom',
 'TileMatrixSet': 'PM'},
'Landcover_Clc12r_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[40.576, -9.88147], [51.4428, 11.6781]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.CLC12R_FR',
 'name': 'GeoportailFrance.Landcover_Clc12r_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Clc12_dom': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
```

```
 'bounds': [[-21.4756, -63.3725], [47.1747, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - DOM',
 'variant': 'LANDCOVER.CLC12_DOM',
 'name': 'GeoportailFrance.Landcover_Clc12_dom',
 'TileMatrixSet': 'PM'},
'Landcover_Clc12_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[40.576, -9.88147], [51.4428, 11.6781]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.CLC12_FR',
 'name': 'GeoportailFrance.Landcover_Clc12_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Clc18': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SER
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.4428, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover',
 'variant': 'LANDCOVER.CLC18',
 'name': 'GeoportailFrance.Landcover_Clc18',
 'TileMatrixSet': 'PM'},
'Landcover_Clc18_dom': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [47.1747, 55.9259]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - DOM',
 'variant': 'LANDCOVER.CLC18_DOM',
```

```
 'name': 'GeoportailFrance.Landcover_Clc18_dom',
 'TileMatrixSet': 'PM'},
'Landcover_Clc18_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[40.576, -9.88147], [51.4428, 11.6781]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.CLC18_FR',
 'name': 'GeoportailFrance.Landcover_Clc18_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Clc90_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[40.576, -9.88147], [51.4428, 11.6781]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.CLC90_FR',
 'name': 'GeoportailFrance.Landcover_Clc90_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Grid_Clc00': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTil
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4825, -61.9063], [51.1827, 55.9362]],
 'min_zoom': 0,
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover',
 'variant': 'LANDCOVER.GRID.CLC00',
 'name': 'GeoportailFrance.Landcover_Grid_Clc00',
 'TileMatrixSet': 'PM'},
'Landcover_Grid_Clc00r_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.1779, -5.68494], [51.1827, 10.8556]],
 'min_zoom': 0,
```

```
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.GRID.CLC00R_FR',
 'name': 'GeoportailFrance.Landcover_Grid_Clc00r_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Grid_Clc00_dom': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=G
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4825, -61.9063], [16.6077, 55.9362]],
 'min_zoom': 0,
 'max_zoom': 12,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - DOM',
 'variant': 'LANDCOVER.GRID.CLC00_DOM',
 'name': 'GeoportailFrance.Landcover_Grid_Clc00_dom',
 'TileMatrixSet': 'PM'},
'Landcover_Grid_Clc00_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.1779, -5.68494], [51.1827, 10.8556]],
 'min_zoom': 0,
 'max_zoom': 12,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.GRID.CLC00_FR',
 'name': 'GeoportailFrance.Landcover_Grid_Clc00_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Grid_Clc06': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTi
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4825, -61.9063], [51.1827, 55.9362]],
 'min_zoom': 0,
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover',
 'variant': 'LANDCOVER.GRID.CLC06',
 'name': 'GeoportailFrance.Landcover_Grid_Clc06',
 'TileMatrixSet': 'PM'},
```

```
'Landcover_Grid_Clc06r': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetT
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4825, -61.9063], [51.2963, 55.9362]],
 'min_zoom': 0,
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover',
 'variant': 'LANDCOVER.GRID.CLC06R',
 'name': 'GeoportailFrance.Landcover_Grid_Clc06r',
 'TileMatrixSet': 'PM'},
'Landcover_Grid_Clc06r_dom': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4825, -61.9063], [16.6077, 55.9362]],
 'min_zoom': 0,
 'max_zoom': 12,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - DOM',
 'variant': 'LANDCOVER.GRID.CLC06R_DOM',
 'name': 'GeoportailFrance.Landcover_Grid_Clc06r_dom',
 'TileMatrixSet': 'PM'},
'Landcover_Grid_Clc06r_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=G
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.0278, -5.91689], [51.2963, 11.0883]],
 'min_zoom': 0,
 'max_zoom': 12,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.GRID.CLC06R_FR',
 'name': 'GeoportailFrance.Landcover_Grid_Clc06r_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Grid_Clc06_dom': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=G
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4825, -61.9063], [16.6077, 55.9362]],
 'min_zoom': 0,
 'max_zoom': 12,
 'apikey': 'clc',
```

```
 'format': 'image/png',
 'style': 'CORINE Land Cover - DOM',
 'variant': 'LANDCOVER.GRID.CLC06_DOM',
 'name': 'GeoportailFrance.Landcover_Grid_Clc06_dom',
 'TileMatrixSet': 'PM'},
'Landcover_Grid_Clc06_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.1779, -5.68494], [51.1827, 10.8556]],
 'min_zoom': 0,
 'max_zoom': 12,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.GRID.CLC06_FR',
 'name': 'GeoportailFrance.Landcover_Grid_Clc06_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Grid_Clc12': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTi
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4825, -61.9063], [51.2963, 55.9362]],
 'min_zoom': 0,
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover',
 'variant': 'LANDCOVER.GRID.CLC12',
 'name': 'GeoportailFrance.Landcover_Grid_Clc12',
 'TileMatrixSet': 'PM'},
'Landcover_Grid_Clc90_fr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.1779, -5.68494], [51.1827, 10.8556]],
 'min_zoom': 0,
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - France métropolitaine',
 'variant': 'LANDCOVER.GRID.CLC90_FR',
 'name': 'GeoportailFrance.Landcover_Grid_Clc90_fr',
 'TileMatrixSet': 'PM'},
'Landcover_Hr_Dlt_Clc12': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Get
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
```

```
 'attribution': 'Geoportail France',
 'bounds': [[-21.572, -62.3602], [51.4949, 55.8441]],
 'min_zoom': 0,
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - HR - type de forêts',
 'variant': 'LANDCOVER.HR.DLT.CLC12',
 'name': 'GeoportailFrance.Landcover_Hr_Dlt_Clc12',
 'TileMatrixSet': 'PM'},
'Landcover_Hr_Dlt_Clc15': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Get
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.572, -62.3602], [51.4949, 55.8441]],
 'min_zoom': 0,
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - HR - type de forêts',
 'variant': 'LANDCOVER.HR.DLT.CLC15',
 'name': 'GeoportailFrance.Landcover_Hr_Dlt_Clc15',
 'TileMatrixSet': 'PM'},
'Landcover_Hr_Gra_Clc15': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Get
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.3925, -61.8133], [51.4949, 55.84]],
 'min_zoom': 0,
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - HR - prairies',
 'variant': 'LANDCOVER.HR.GRA.CLC15',
 'name': 'GeoportailFrance.Landcover_Hr_Gra_Clc15',
 'TileMatrixSet': 'PM'},
'Landcover_Hr_Imd_Clc12': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Get
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.5758, -62.3609], [51.4952, 56.1791]],
 'min_zoom': 0,
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': "CORINE Land Cover - HR - taux d'imperméabilisation des sols",
```

```
 'variant': 'LANDCOVER.HR.IMD.CLC12',
 'name': 'GeoportailFrance.Landcover_Hr_Imd_Clc12',
 'TileMatrixSet': 'PM'},
'Landcover_Hr_Imd_Clc15': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Get
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.5758, -62.3609], [51.4952, 56.1791]],
 'min_zoom': 0,
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': "CORINE Land Cover - HR - taux d'imperméabilisation des sols",
 'variant': 'LANDCOVER.HR.IMD.CLC15',
 'name': 'GeoportailFrance.Landcover_Hr_Imd_Clc15',
 'TileMatrixSet': 'PM'},
'Landcover_Hr_Tcd_Clc12': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Get
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.572, -62.3602], [51.4949, 55.8441]],
 'min_zoom': 0,
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - HR - taux de couvert arboré',
 'variant': 'LANDCOVER.HR.TCD.CLC12',
 'name': 'GeoportailFrance.Landcover_Hr_Tcd_Clc12',
 'TileMatrixSet': 'PM'},
'Landcover_Hr_Tcd_Clc15': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Get
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.572, -62.3602], [51.4949, 55.8441]],
 'min_zoom': 0,
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - HR - taux de couvert arboré',
 'variant': 'LANDCOVER.HR.TCD.CLC15',
 'name': 'GeoportailFrance.Landcover_Hr_Tcd_Clc15',
 'TileMatrixSet': 'PM'},
'Landcover_Hr_Waw_Clc15': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Get
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.572, -62.3602], [51.4949, 55.8441]],
```

```
 'min_zoom': 0,
 'max_zoom': 13,
 'apikey': 'clc',
 'format': 'image/png',
 'style': 'CORINE Land Cover - HR - zones humides et surfaces en eaux permanentes',
 'variant': 'LANDCOVER.HR.WAW.CLC15',
 'name': 'GeoportailFrance.Landcover_Hr_Waw_Clc15',
 'TileMatrixSet': 'PM'},
'Areamanagement_Zfu': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'AREAMANAGEMENT.ZFU',
 'name': 'GeoportailFrance.Areamanagement_Zfu',
 'TileMatrixSet': 'PM'},
'Areamanagement_Zus': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'AREAMANAGEMENT.ZUS',
 'name': 'GeoportailFrance.Areamanagement_Zus',
 'TileMatrixSet': 'PM'},
'Communes_Prioritydisctrict': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 17,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'COMMUNES.PRIORITYDISCTRICT',
 'name': 'GeoportailFrance.Communes_Prioritydisctrict',
```

```
 'TileMatrixSet': 'PM'},
'Dreal_Zonage_pinel': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[47.2719, -5.15012], [48.9064, -1.00687]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'DREAL.ZONAGE_PINEL',
 'name': 'GeoportailFrance.Dreal_Zonage_pinel',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Enfants_0_17_Ans_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/w
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.ENFANTS.0.17.ANS.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Enfants_0_17_Ans_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Logements_Surface_Moyenne_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geo
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.LOGEMENTS.SURFACE.MOYENNE.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Logements_Surface_Moyenne_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Niveau_De_Vie_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
```

```
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.NIVEAU.DE.VIE.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Niveau_De_Vie_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Familles_Monoparentales_Secret': {'url': 'https://wxs.ign.fr/{apikey}/
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.FAMILLES.MONOPARENTALES.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Familles_Monoparentales_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Individus_25_39_Ans_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geop
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.INDIVIDUS.25.39.ANS.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Individus_25_39_Ans_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Individus_40_54_Ans_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geop
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.INDIVIDUS.40.54.ANS.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Individus_40_54_Ans_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Individus_55_64_Ans_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geop
```

```
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.INDIVIDUS.55.64.ANS.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Individus_55_64_Ans_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Logements_Apres_1990_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geo
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.LOGEMENTS.APRES.1990.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Logements_Apres_1990_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Logements_Avant_1945_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geo
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.LOGEMENTS.AVANT.1945.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Logements_Avant_1945_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Logements_Collectifs_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geo
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
```

```
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.LOGEMENTS.COLLECTIFS.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Logements_Collectifs_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Logements_Construits_1945_1970_Secret': {'url': 'https://wxs.ign.fr/{a
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.LOGEMENTS.CONSTRUITS.1945.1970.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Logements_Construits_1945_1970_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Logements_Construits_1970_1990_Secret': {'url': 'https://wxs.ign.fr/{a
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.LOGEMENTS.CONSTRUITS.1970.1990.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Logements_Construits_1970_1990_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Logements_Sociaux_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geopor
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.LOGEMENTS.SOCIAUX.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Logements_Sociaux_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Menages_1_Personne_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geopo
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
```

```
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.MENAGES.1.PERSONNE.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Menages_1_Personne_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Menages_5_Personnes_Ouplus_Secret': {'url': 'https://wxs.ign.fr/{apike
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.MENAGES.5.PERSONNES.OUPLUS.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Menages_5_Personnes_Ouplus_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Menages_Maison_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geoportai
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.MENAGES.MAISON.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Menages_Maison_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Menages_Pauvres_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geoportai
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.MENAGES.PAUVRES.SECRET',
```

```
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Menages_Pauvres_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Menages_Proprietaires_Secret': {'url': 'https://wxs.ign.fr/{apikey}/g
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.MENAGES.PROPRIETAIRES.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Menages_Proprietaires_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Part_Plus_65_Ans_Secret': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.PART.PLUS.65.ANS.SECRET',
 'name': 'GeoportailFrance.Insee_Filosofi_Part_Plus_65_Ans_Secret',
 'TileMatrixSet': 'PM'},
'Insee_Filosofi_Population': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'economie',
 'format': 'image/png',
 'style': 'INSEE',
 'variant': 'INSEE.FILOSOFI.POPULATION',
 'name': 'GeoportailFrance.Insee_Filosofi_Population',
 'TileMatrixSet': 'PM'},
'Debroussaillement': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
```

```
 'max_zoom': 18,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'DEBROUSSAILLEMENT',
 'name': 'GeoportailFrance.Debroussaillement',
 'TileMatrixSet': 'PM'},
'Forets_Publiques': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 3,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'FORETS PUBLIQUES ONF',
 'variant': 'FORETS.PUBLIQUES',
 'name': 'GeoportailFrance.Forets_Publiques',
 'TileMatrixSet': 'PM'},
'Geographicalgridsystem_Dfci': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'GEOGRAPHICALGRIDSYSTEM.DFCI',
 'name': 'GeoportailFrance.Geographicalgridsystem_Dfci',
 'TileMatrixSet': 'PM'},
'Landcover_Forestareas': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTi
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDCOVER.FORESTAREAS',
 'name': 'GeoportailFrance.Landcover_Forestareas',
 'TileMatrixSet': 'PM'},
```

```
'Landcover_Forestinventory_V1': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDCOVER.FORESTINVENTORY.V1',
 'name': 'GeoportailFrance.Landcover_Forestinventory_V1',
 'TileMatrixSet': 'PM'},
'Landcover_Forestinventory_V2': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDCOVER.FORESTINVENTORY.V2',
 'name': 'GeoportailFrance.Landcover_Forestinventory_V2',
 'TileMatrixSet': 'PM'},
'Landcover_Sylvoecoregions': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=G
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDCOVER.SYLVOECOREGIONS',
 'name': 'GeoportailFrance.Landcover_Sylvoecoregions',
 'TileMatrixSet': 'PM'},
'Landcover_Sylvoecoregions_Alluvium': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
```

```
 'format': 'image/png',
 'style': 'normal',
 'variant': 'LANDCOVER.SYLVOECOREGIONS.ALLUVIUM',
 'name': 'GeoportailFrance.Landcover_Sylvoecoregions_Alluvium',
 'TileMatrixSet': 'PM'},
'Protectedareas_Apb': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.APB',
 'name': 'GeoportailFrance.Protectedareas_Apb',
 'TileMatrixSet': 'PM'},
'Protectedareas_Apg': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.APG',
 'name': 'GeoportailFrance.Protectedareas_Apg',
 'TileMatrixSet': 'PM'},
'Protectedareas_Aphn': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-53.6279, -63.3725], [51.3121, 82.645]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.APHN',
 'name': 'GeoportailFrance.Protectedareas_Aphn',
 'TileMatrixSet': 'PM'},
'Protectedareas_Aplg': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
```

```
 'attribution': 'Geoportail France',
 'bounds': [[-53.6279, -63.3725], [51.3121, 82.645]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'PROTECTEDAREAS.APLG',
 'name': 'GeoportailFrance.Protectedareas_Aplg',
 'TileMatrixSet': 'PM'},
'Protectedareas_Bios': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTil
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.BIOS',
 'name': 'GeoportailFrance.Protectedareas_Bios',
 'TileMatrixSet': 'PM'},
'Protectedareas_Gp': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&S
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.GP',
 'name': 'GeoportailFrance.Protectedareas_Gp',
 'TileMatrixSet': 'PM'},
'Protectedareas_Inpg': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTil
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-53.6279, -63.3725], [51.3121, 82.645]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
```

```
 'variant': 'PROTECTEDAREAS.INPG',
 'name': 'GeoportailFrance.Protectedareas_Inpg',
 'TileMatrixSet': 'PM'},
'Protectedareas_Mnhn_Cdl_Parcels': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REC
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.MNHN.CDL.PARCELS',
 'name': 'GeoportailFrance.Protectedareas_Mnhn_Cdl_Parcels',
 'TileMatrixSet': 'PM'},
'Protectedareas_Mnhn_Cdl_Perimeter': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?R
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.MNHN.CDL.PERIMETER',
 'name': 'GeoportailFrance.Protectedareas_Mnhn_Cdl_Perimeter',
 'TileMatrixSet': 'PM'},
'Protectedareas_Mnhn_Conservatoires': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.MNHN.CONSERVATOIRES',
 'name': 'GeoportailFrance.Protectedareas_Mnhn_Conservatoires',
 'TileMatrixSet': 'PM'},
'Protectedareas_Mnhn_Rn_Perimeter': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?RE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-53.6279, -63.3725], [51.3121, 82.645]],
```

```
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.MNHN.RN.PERIMETER',
 'name': 'GeoportailFrance.Protectedareas_Mnhn_Rn_Perimeter',
 'TileMatrixSet': 'PM'},
'Protectedareas_Pn': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&S
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.PN',
 'name': 'GeoportailFrance.Protectedareas_Pn',
 'TileMatrixSet': 'PM'},
'Protectedareas_Pnm': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.PNM',
 'name': 'GeoportailFrance.Protectedareas_Pnm',
 'TileMatrixSet': 'PM'},
'Protectedareas_Pnr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.PNR',
 'name': 'GeoportailFrance.Protectedareas_Pnr',
```

```
 'TileMatrixSet': 'PM'},
'Protectedareas_Prsf': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTil
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 17,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'POINT RENCONTRE SECOURS FORET',
 'variant': 'PROTECTEDAREAS.PRSF',
 'name': 'GeoportailFrance.Protectedareas_Prsf',
 'TileMatrixSet': 'PM'},
'Protectedareas_Ramsar': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetT
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.RAMSAR',
 'name': 'GeoportailFrance.Protectedareas_Ramsar',
 'TileMatrixSet': 'PM'},
'Protectedareas_Rb': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.RB',
 'name': 'GeoportailFrance.Protectedareas_Rb',
 'TileMatrixSet': 'PM'},
'Protectedareas_Ripn': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTil
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
```

```
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.RIPN',
 'name': 'GeoportailFrance.Protectedareas_Ripn',
 'TileMatrixSet': 'PM'},
'Protectedareas_Rn': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&S
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-53.6279, -63.3725], [51.3121, 82.645]],
 'min_zoom': 0,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.RN',
 'name': 'GeoportailFrance.Protectedareas_Rn',
 'TileMatrixSet': 'PM'},
'Protectedareas_Rnc': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.RNC',
 'name': 'GeoportailFrance.Protectedareas_Rnc',
 'TileMatrixSet': 'PM'},
'Protectedareas_Rncf': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.RNCF',
 'name': 'GeoportailFrance.Protectedareas_Rncf',
 'TileMatrixSet': 'PM'},
'Protectedareas_Sic': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
```

```
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.SIC',
 'name': 'GeoportailFrance.Protectedareas_Sic',
 'TileMatrixSet': 'PM'},
'Protectedareas_Znieff1': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Get
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.ZNIEFF1',
 'name': 'GeoportailFrance.Protectedareas_Znieff1',
 'TileMatrixSet': 'PM'},
'Protectedareas_Znieff1_Sea': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.ZNIEFF1.SEA',
 'name': 'GeoportailFrance.Protectedareas_Znieff1_Sea',
 'TileMatrixSet': 'PM'},
'Protectedareas_Znieff2': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Get
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
```

```
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.ZNIEFF2',
 'name': 'GeoportailFrance.Protectedareas_Znieff2',
 'TileMatrixSet': 'PM'},
'Protectedareas_Znieff2_Sea': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.ZNIEFF2.SEA',
 'name': 'GeoportailFrance.Protectedareas_Znieff2_Sea',
 'TileMatrixSet': 'PM'},
'Protectedareas_Zpr': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-53.6279, -63.3725], [51.3121, 82.645]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'PROTECTEDAREAS.ZPR',
 'name': 'GeoportailFrance.Protectedareas_Zpr',
 'TileMatrixSet': 'PM'},
'Protectedareas_Zps': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDAREAS.ZPS',
 'name': 'GeoportailFrance.Protectedareas_Zps',
 'TileMatrixSet': 'PM'},
'Protectedsites_Mnhn_Reserves_regionales': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
```

```
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'environnement',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'PROTECTEDSITES.MNHN.RESERVES-REGIONALES',
 'name': 'GeoportailFrance.Protectedsites_Mnhn_Reserves_regionales',
 'TileMatrixSet': 'PM'},
'Ocsge_Constructions': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[14.2395, -61.6644], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'OCSGE.CONSTRUCTIONS',
 'name': 'GeoportailFrance.Ocsge_Constructions',
 'TileMatrixSet': 'PM'},
'Ocsge_Constructions_2002': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.366, -5.13902], [51.089, 9.55982]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.CONSTRUCTIONS.2002',
 'name': 'GeoportailFrance.Ocsge_Constructions_2002',
 'TileMatrixSet': 'PM',
 'status': 'broken'},
'Ocsge_Constructions_2010': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
```

```
 'variant': 'OCSGE.CONSTRUCTIONS.2010',
 'name': 'GeoportailFrance.Ocsge_Constructions_2010',
 'TileMatrixSet': 'PM'},
'Ocsge_Constructions_2011': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.CONSTRUCTIONS.2011',
 'name': 'GeoportailFrance.Ocsge_Constructions_2011',
 'TileMatrixSet': 'PM'},
'Ocsge_Constructions_2014': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.366, -5.13902], [51.089, 9.55982]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.CONSTRUCTIONS.2014',
 'name': 'GeoportailFrance.Ocsge_Constructions_2014',
 'TileMatrixSet': 'PM',
 'status': 'broken'},
'Ocsge_Constructions_2016': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.CONSTRUCTIONS.2016',
 'name': 'GeoportailFrance.Ocsge_Constructions_2016',
 'TileMatrixSet': 'PM'},
'Ocsge_Constructions_2017': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
```

```
 'bounds': [[14.2395, -61.6644], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.CONSTRUCTIONS.2017',
 'name': 'GeoportailFrance.Ocsge_Constructions_2017',
 'TileMatrixSet': 'PM'},
'Ocsge_Constructions_2019': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[43.3043, -0.291052], [44.0864, 1.2122]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.CONSTRUCTIONS.2019',
 'name': 'GeoportailFrance.Ocsge_Constructions_2019',
 'TileMatrixSet': 'PM'},
'Ocsge_Couverture': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[14.2395, -61.6644], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'OCSGE.COUVERTURE',
 'name': 'GeoportailFrance.Ocsge_Couverture',
 'TileMatrixSet': 'PM'},
'Ocsge_Couverture_2002': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTi
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.366, -5.13902], [51.089, 9.55982]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'OCSGE.COUVERTURE.2002',
```

```
 'name': 'GeoportailFrance.Ocsge_Couverture_2002',
 'TileMatrixSet': 'PM',
 'status': 'broken'},
'Ocsge_Couverture_2010': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetT
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.COUVERTURE.2010',
 'name': 'GeoportailFrance.Ocsge_Couverture_2010',
 'TileMatrixSet': 'PM'},
'Ocsge_Couverture_2011': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetT
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.COUVERTURE.2011',
 'name': 'GeoportailFrance.Ocsge_Couverture_2011',
 'TileMatrixSet': 'PM'},
'Ocsge_Couverture_2014': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetT
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.366, -5.13902], [51.089, 9.55982]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.COUVERTURE.2014',
 'name': 'GeoportailFrance.Ocsge_Couverture_2014',
 'TileMatrixSet': 'PM',
 'status': 'broken'},
'Ocsge_Couverture_2016': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetT
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
```

```
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.COUVERTURE.2016',
 'name': 'GeoportailFrance.Ocsge_Couverture_2016',
 'TileMatrixSet': 'PM'},
'Ocsge_Couverture_2017': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetT
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[14.2395, -61.6644], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.COUVERTURE.2017',
 'name': 'GeoportailFrance.Ocsge_Couverture_2017',
 'TileMatrixSet': 'PM'},
'Ocsge_Couverture_2019': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetT
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[43.3043, -0.291052], [44.0864, 1.2122]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.COUVERTURE.2019',
 'name': 'GeoportailFrance.Ocsge_Couverture_2019',
 'TileMatrixSet': 'PM'},
'Ocsge_Usage': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SERVIC
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[14.2395, -61.6644], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'OCSGE.USAGE',
```

```
 'name': 'GeoportailFrance.Ocsge_Usage',
 'TileMatrixSet': 'PM'},
'Ocsge_Usage_2002': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SI
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
 'attribution': 'Geoportail France',
 'bounds': [[41.366, -5.13902], [51.089, 9.55982]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'OCSGE.USAGE.2002',
 'name': 'GeoportailFrance.Ocsge_Usage_2002',
 'TileMatrixSet': 'PM',
 'status': 'broken'},
'Ocsge_Usage_2010': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SI
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.USAGE.2010',
 'name': 'GeoportailFrance.Ocsge_Usage_2010',
 'TileMatrixSet': 'PM'},
'Ocsge_Usage_2011': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SI
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.USAGE.2011',
 'name': 'GeoportailFrance.Ocsge_Usage_2011',
 'TileMatrixSet': 'PM'},
'Ocsge_Usage_2014': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SI
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
 'attribution': 'Geoportail France',
 'bounds': [[41.366, -5.13902], [51.089, 9.55982]],
```

```
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'OCSGE.USAGE.2014',
 'name': 'GeoportailFrance.Ocsge_Usage_2014',
 'TileMatrixSet': 'PM',
 'status': 'broken'},
'Ocsge_Usage_2016': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SI
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.USAGE.2016',
 'name': 'GeoportailFrance.Ocsge_Usage_2016',
 'TileMatrixSet': 'PM'},
'Ocsge_Usage_2017': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SI
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
 'attribution': 'Geoportail France',
 'bounds': [[14.2395, -61.6644], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.USAGE.2017',
 'name': 'GeoportailFrance.Ocsge_Usage_2017',
 'TileMatrixSet': 'PM'},
'Ocsge_Usage_2019': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SI
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
 'attribution': 'Geoportail France',
 'bounds': [[43.3043, -0.291052], [44.0864, 1.2122]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.USAGE.2019',
```

```
 'name': 'GeoportailFrance.Ocsge_Usage_2019',
 'TileMatrixSet': 'PM'},
'Ocsge_Visu_2016': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[43.2815, -0.318517], [44.0543, 1.22575]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.VISU.2016',
 'name': 'GeoportailFrance.Ocsge_Visu_2016',
 'TileMatrixSet': 'PM'},
'Ocsge_Visu_2019': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[43.2815, -0.321664], [44.1082, 1.22575]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'ocsge',
 'format': 'image/png',
 'style': 'nolegend',
 'variant': 'OCSGE.VISU.2019',
 'name': 'GeoportailFrance.Ocsge_Visu_2019',
 'TileMatrixSet': 'PM'},
'Hr_Orthoimagery_Orthophotos': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4013, -63.1607], [51.1124, 55.8464]],
 'min_zoom': 6,
 'max_zoom': 19,
 'apikey': 'ortho',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'HR.ORTHOIMAGERY.ORTHOPHOTOS',
 'name': 'GeoportailFrance.Hr_Orthoimagery_Orthophotos',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophos_Restrictedareas': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-22.9723, -178.309], [51.3121, 168.298]],
 'min_zoom': 6,
```

```
 'max_zoom': 16,
 'apikey': 'ortho',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOS.RESTRICTEDAREAS',
 'name': 'GeoportailFrance.Orthoimagery_Orthophos_Restrictedareas',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Bdortho': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?RE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-22.7643, -178.187], [51.1124, 168.19]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'ortho',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.BDORTHO',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Bdortho',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Coast2000': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[43.301, -5.21565], [51.1233, 2.60783]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'ortho',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.COAST2000',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Coast2000',
 'TileMatrixSet': 'PM',
 'status': 'broken'},
'Orthoimagery_Orthophotos_Ilesdunord': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[17.8626, -63.1986], [18.1701, -62.7828]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'ortho',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.ILESDUNORD',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Ilesdunord',
```

```
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Irc': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4013, -62.9717], [51.1124, 55.8464]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'ortho',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.IRC',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Irc',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Irc_express_2023': {'url': 'https://wxs.ign.fr/{apikey}/geoportai
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'ortho',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.IRC-EXPRESS.2023',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Irc_express_2023',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Ortho_asp_pac2022': {'url': 'https://wxs.ign.fr/{apikey}/geoporta
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.ORTHO-ASP_PAC2022',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Ortho_asp_pac2022',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Ortho_asp_pac2023': {'url': 'https://wxs.ign.fr/{apikey}/geoporta
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
```

```
 'apikey': 'ortho',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.ORTHO-ASP_PAC2023',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Ortho_asp_pac2023',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Ortho_express_2023': {'url': 'https://wxs.ign.fr/{apikey}/geopor
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'ortho',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.ORTHO-EXPRESS.2023',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Ortho_express_2023',
 'TileMatrixSet': 'PM'},
'Pcrs_Lamb93': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SERVIC
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[42.6976, -3.80779], [48.8107, 6.92319]],
 'min_zoom': 6,
 'max_zoom': 21,
 'apikey': 'ortho',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'PCRS.LAMB93',
 'name': 'GeoportailFrance.Pcrs_Lamb93',
 'TileMatrixSet': 'LAMB93_5cm_EPSG',
 'status': 'broken'},
'Thr_Orthoimagery_Orthophotos': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[40.4378, -6.92466], [51.9098, 11.4965]],
 'min_zoom': 6,
 'max_zoom': 21,
 'apikey': 'ortho',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'THR.ORTHOIMAGERY.ORTHOPHOTOS',
 'name': 'GeoportailFrance.Thr_Orthoimagery_Orthophotos',
 'TileMatrixSet': 'PM'},
```

```
'Orthoimagery_Orthophotos_1950_1965': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts'
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4013, -67.7214], [51.0945, 55.8464]],
 'min_zoom': 3,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.1950-1965',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_1950_1965',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_1980_1995': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts'
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.3125, -2.37153], [49.7785, 9.67536]],
 'min_zoom': 3,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/png',
 'style': 'BDORTHOHISTORIQUE',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.1980-1995',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_1980_1995',
 'TileMatrixSet': 'PM',
 'status': 'broken'},
'Orthoimagery_Orthophotos_Irc_express_2021': {'url': 'https://wxs.ign.fr/{apikey}/geoportai
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.IRC-EXPRESS.2021',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Irc_express_2021',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Irc_express_2022': {'url': 'https://wxs.ign.fr/{apikey}/geoportai
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4013, -62.9717], [51.1124, 55.8464]],
 'min_zoom': 0,
 'max_zoom': 19,
```

```
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.IRC-EXPRESS.2022',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Irc_express_2022',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Irc_2013': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[42.5538, -3.74871], [50.3767, 7.17132]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.IRC.2013',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Irc_2013',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Irc_2014': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[43.1508, -2.37153], [49.6341, 7.22637]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.IRC.2014',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Irc_2014',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Irc_2015': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[42.3163, -5.20863], [51.0945, 8.25674]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.IRC.2015',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Irc_2015',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Irc_2016': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?
```

```
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.3215, -3.74871], [50.1839, 9.66314]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.IRC.2016',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Irc_2016',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Irc_2017': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?l
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[42.9454, -0.185295], [46.4137, 7.74363]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.IRC.2017',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Irc_2017',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Irc_2018': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?l
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[42.3163, -5.19371], [51.1124, 8.25765]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.IRC.2018',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Irc_2018',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Irc_2019': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?l
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.3125, -3.74871], [50.1928, 9.66314]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
```

```
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.IRC.2019',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Irc_2019',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Irc_2020': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[42.9454, -2.68142], [49.4512, 7.74363]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.IRC.2020',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Irc_2020',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Ortho_asp_pac2020': {'url': 'https://wxs.ign.fr/{apikey}/geoporta
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.ORTHO-ASP_PAC2020',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Ortho_asp_pac2020',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Ortho_asp_pac2021': {'url': 'https://wxs.ign.fr/{apikey}/geoporta
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.ORTHO-ASP_PAC2021',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Ortho_asp_pac2021',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Ortho_express_2021': {'url': 'https://wxs.ign.fr/{apikey}/geopor
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
```

```
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 20,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.ORTHO-EXPRESS.2021',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Ortho_express_2021',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Ortho_express_2022': {'url': 'https://wxs.ign.fr/{apikey}/geopor
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4013, -63.1607], [51.1124, 55.8464]],
 'min_zoom': 6,
 'max_zoom': 19,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.ORTHO-EXPRESS.2022',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Ortho_express_2022',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Socle_asp_2018': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.SOCLE-ASP.2018',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Socle_asp_2018',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos_Urgence_Alex': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wm
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[43.8095, 7.07917], [44.1903, 7.64199]],
 'min_zoom': 6,
 'max_zoom': 20,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS.URGENCE.ALEX',
```

```
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos_Urgence_Alex',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2000': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2000',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2000',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2000_2005': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?F
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4013, -178.187], [64.0698, 55.8561]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2000-2005',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2000_2005',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2001': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[4.47153, -61.2472], [50.3765, 7.23234]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2001',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2001',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2002': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[4.49867, -61.2472], [50.3765, 9.68861]],
 'min_zoom': 0,
```

```
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2002',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2002',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2003': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2003',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2003',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2004': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4013, -178.187], [51.091, 55.8561]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2004',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2004',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2005': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4013, -178.187], [51.091, 55.8561]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2005',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2005',
 'TileMatrixSet': 'PM'},
```

```
'Orthoimagery_Orthophotos2006': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4013, -178.187], [51.091, 55.8561]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2006',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2006',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2006_2010': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?R
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2006-2010',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2006_2010',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2007': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2007',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2007',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2008': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4013, -178.187], [51.091, 55.8561]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
```

```
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2008',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2008',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2009': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2009',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2009',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2010': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2010',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2010',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2011': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2011',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2011',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2011_2015': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?I
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
```

```
 'attribution': 'Geoportail France',
 'bounds': [[-21.4013, -178.187], [51.0945, 55.8561]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2011-2015',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2011_2015',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2012': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2012',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2012',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2013': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2013',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2013',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2014': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
```

```
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2014',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2014',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2015': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2015',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2015',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2016': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2016',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2016',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2017': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4013, -63.1607], [50.3856, 55.8464]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2017',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2017',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2018': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[42.3163, -5.20863], [51.1124, 8.25765]],
```

```
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2018',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2018',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2019': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.3125, -3.74871], [50.1928, 9.66314]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2019',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2019',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Orthophotos2020': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUES
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[42.9454, -2.68142], [49.4512, 7.74363]],
 'min_zoom': 6,
 'max_zoom': 19,
 'apikey': 'orthohisto',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHOPHOTOS2020',
 'name': 'GeoportailFrance.Orthoimagery_Orthophotos2020',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Pleiades_2012': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmt
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.3539, -53.2686], [50.6037, 55.5544]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'satellite',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.PLEIADES.2012',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Pleiades_2012',
```

```
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Pleiades_2013': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmt
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'satellite',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.PLEIADES.2013',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Pleiades_2013',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Pleiades_2014': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmt
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'satellite',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.PLEIADES.2014',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Pleiades_2014',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Pleiades_2015': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmt
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'satellite',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.PLEIADES.2015',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Pleiades_2015',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Pleiades_2016': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmt
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.32, -54.1373], [50.6549, 55.8441]],
 'min_zoom': 0,
 'max_zoom': 18,
```

```
 'apikey': 'satellite',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.PLEIADES.2016',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Pleiades_2016',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Pleiades_2017': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmt
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4013, -63.1796], [51.1117, 55.8465]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'satellite',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.PLEIADES.2017',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Pleiades_2017',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Pleiades_2018': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmt
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4094, -63.1702], [51.0841, 55.8649]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'satellite',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.PLEIADES.2018',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Pleiades_2018',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Pleiades_2019': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmt
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4094, -63.1702], [51.1117, 55.8649]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'satellite',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.PLEIADES.2019',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Pleiades_2019',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Pleiades_2020': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmt
```

```
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-13.0169, -63.1724], [51.1117, 45.3136]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'satellite',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.PLEIADES.2020',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Pleiades_2020',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Pleiades_2021': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wm
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 19,
 'apikey': 'satellite',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.PLEIADES.2021',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Pleiades_2021',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Pleiades_2022': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wm
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.3733, -67.7132], [69.3108, 55.7216]],
 'min_zoom': 0,
 'max_zoom': 18,
 'apikey': 'satellite',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.PLEIADES.2022',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Pleiades_2022',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Rapideye_2010': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wm
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.2014, -5.80725], [50.9218, 10.961]],
 'min_zoom': 0,
 'max_zoom': 15,
 'apikey': 'satellite',
 'format': 'image/jpeg',
```

```
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.RAPIDEYE.2010',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Rapideye_2010',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Rapideye_2011': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wm
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.0227, -5.80725], [51.1752, 10.961]],
 'min_zoom': 0,
 'max_zoom': 15,
 'apikey': 'satellite',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.RAPIDEYE.2011',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Rapideye_2011',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Spot_2013': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?R
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[44.8809, 0.563585], [50.3879, 4.29191]],
 'min_zoom': 0,
 'max_zoom': 17,
 'apikey': 'satellite',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.SPOT.2013',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Spot_2013',
 'TileMatrixSet': 'PM',
 'status': 'broken'},
'Orthoimagery_Ortho_sat_Spot_2014': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?R
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-75.0, -179.5], [75.0, 179.5]],
 'min_zoom': 0,
 'max_zoom': 17,
 'apikey': 'satellite',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.SPOT.2014',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Spot_2014',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Spot_2015': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?R
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
```

```
 'attribution': 'Geoportail France',
 'bounds': [[-21.4104, -61.8141], [51.106, 55.856]],
 'min_zoom': 0,
 'max_zoom': 17,
 'apikey': 'satellite',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.SPOT.2015',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Spot_2015',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Spot_2016': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?RI
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
 'attribution': 'Geoportail France',
 'bounds': [[-21.4104, -61.85], [51.1123, 55.8562]],
 'min_zoom': 0,
 'max_zoom': 17,
 'apikey': 'satellite',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.SPOT.2016',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Spot_2016',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Spot_2017': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?RI
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
 'attribution': 'Geoportail France',
 'bounds': [[-21.4104, -61.8534], [51.1123, 55.8562]],
 'min_zoom': 0,
 'max_zoom': 17,
 'apikey': 'satellite',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.SPOT.2017',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Spot_2017',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Spot_2018': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?RI
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail I
 'attribution': 'Geoportail France',
 'bounds': [[41.2593, -5.57103], [51.1123, 10.7394]],
 'min_zoom': 0,
 'max_zoom': 17,
 'apikey': 'satellite',
 'format': 'image/jpeg',
 'style': 'normal',
```

```
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.SPOT.2018',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Spot_2018',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Spot_2019': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?RE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.2593, -5.57103], [51.1123, 10.7394]],
 'min_zoom': 0,
 'max_zoom': 17,
 'apikey': 'satellite',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.SPOT.2019',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Spot_2019',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Spot_2020': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?RE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.2593, -5.57103], [51.1123, 10.7394]],
 'min_zoom': 0,
 'max_zoom': 17,
 'apikey': 'satellite',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.SPOT.2020',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Spot_2020',
 'TileMatrixSet': 'PM'},
'Orthoimagery_Ortho_sat_Spot_2021': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?RE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[41.2593, -5.57103], [51.1123, 10.7394]],
 'min_zoom': 0,
 'max_zoom': 17,
 'apikey': 'satellite',
 'format': 'image/jpeg',
 'style': 'normal',
 'variant': 'ORTHOIMAGERY.ORTHO-SAT.SPOT.2021',
 'name': 'GeoportailFrance.Orthoimagery_Ortho_sat_Spot_2021',
 'TileMatrixSet': 'PM'},
'Bdcarto_etat_major_Niveau3': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[42.3263, -5.15012], [51.0938, 7.19384]],
```

```
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'sol',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'BDCARTO_ETAT-MAJOR.NIVEAU3',
 'name': 'GeoportailFrance.Bdcarto_etat_major_Niveau3',
 'TileMatrixSet': 'PM'},
'Bdcarto_etat_major_Niveau4': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[41.3252, -5.15047], [51.0991, 9.57054]],
 'min_zoom': 6,
 'max_zoom': 16,
 'apikey': 'sol',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'BDCARTO_ETAT-MAJOR.NIVEAU4',
 'name': 'GeoportailFrance.Bdcarto_etat_major_Niveau4',
 'TileMatrixSet': 'PM'},
'Buildings_Buildings': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTil
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4969, -63.9692], [71.5841, 55.9644]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'topographie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'BUILDINGS.BUILDINGS',
 'name': 'GeoportailFrance.Buildings_Buildings',
 'TileMatrixSet': 'PM'},
'Geographicalnames_Names': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4969, -63.9692], [71.5841, 55.9644]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'topographie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'GEOGRAPHICALNAMES.NAMES',
 'name': 'GeoportailFrance.Geographicalnames_Names',
```

```
 'TileMatrixSet': 'PM'},
'Hydrography_Hydrography': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4969, -63.9692], [71.5841, 55.9644]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'topographie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'HYDROGRAPHY.HYDROGRAPHY',
 'name': 'GeoportailFrance.Hydrography_Hydrography',
 'TileMatrixSet': 'PM'},
'Transportnetwork_Commontransportelements_Markerpost': {'url': 'https://wxs.ign.fr/{apikey
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 10,
 'max_zoom': 18,
 'apikey': 'topographie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'TRANSPORTNETWORK.COMMONTRANSPORTELEMENTS.MARKERPOST',
 'name': 'GeoportailFrance.Transportnetwork_Commontransportelements_Markerpost',
 'TileMatrixSet': 'PM'},
'Transportnetworks_Railways': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4969, -63.9692], [71.5841, 55.9644]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'topographie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'TRANSPORTNETWORKS.RAILWAYS',
 'name': 'GeoportailFrance.Transportnetworks_Railways',
 'TileMatrixSet': 'PM'},
'Transportnetworks_Roads': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=Ge
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4969, -63.9692], [71.5841, 55.9644]],
 'min_zoom': 6,
 'max_zoom': 18,
```

```
 'apikey': 'topographie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'TRANSPORTNETWORKS.ROADS',
 'name': 'GeoportailFrance.Transportnetworks_Roads',
 'TileMatrixSet': 'PM'},
'Transportnetworks_Runways': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4969, -63.9692], [71.5841, 55.9644]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'topographie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'TRANSPORTNETWORKS.RUNWAYS',
 'name': 'GeoportailFrance.Transportnetworks_Runways',
 'TileMatrixSet': 'PM'},
'Utilityandgovernmentalservices_All': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [71.5841, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 18,
 'apikey': 'topographie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'UTILITYANDGOVERNMENTALSERVICES.ALL',
 'name': 'GeoportailFrance.Utilityandgovernmentalservices_All',
 'TileMatrixSet': 'PM'},
'Hedge_Hedge': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SERVIC
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 7,
 'max_zoom': 18,
 'apikey': 'topographie',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'hedge.hedge',
 'name': 'GeoportailFrance.Hedge_Hedge',
 'TileMatrixSet': 'PM'},
'Securoute_Te_1te': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&S
```

```
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 4,
 'max_zoom': 17,
 'apikey': 'transports',
 'format': 'image/png',
 'style': 'RESEAU ROUTIER 1TE',
 'variant': 'SECUROUTE.TE.1TE',
 'name': 'GeoportailFrance.Securoute_Te_1te',
 'TileMatrixSet': 'PM'},
'Securoute_Te_2te48': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 17,
 'apikey': 'transports',
 'format': 'image/png',
 'style': 'RESEAU ROUTIER 2TE48',
 'variant': 'SECUROUTE.TE.2TE48',
 'name': 'GeoportailFrance.Securoute_Te_2te48',
 'TileMatrixSet': 'PM'},
'Securoute_Te_All': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 17,
 'apikey': 'transports',
 'format': 'image/png',
 'style': 'TOUS LES FRANCHISSEMENTS',
 'variant': 'SECUROUTE.TE.ALL',
 'name': 'GeoportailFrance.Securoute_Te_All',
 'TileMatrixSet': 'PM'},
'Securoute_Te_Oa': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SER
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 17,
 'apikey': 'transports',
 'format': 'image/png',
```

```
 'style': 'AUTRES FRANCHISSEMENTS',
 'variant': 'SECUROUTE.TE.OA',
 'name': 'GeoportailFrance.Securoute_Te_Oa',
 'TileMatrixSet': 'PM'},
'Securoute_Te_Pn': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&SE
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 17,
 'apikey': 'transports',
 'format': 'image/png',
 'style': 'FRANCHISSEMENTS PASSAGE A NIVEAU',
 'variant': 'SECUROUTE.TE.PN',
 'name': 'GeoportailFrance.Securoute_Te_Pn',
 'TileMatrixSet': 'PM'},
'Securoute_Te_Pnd': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&S
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 17,
 'apikey': 'transports',
 'format': 'image/png',
 'style': 'FRANCHISSEMENTS PASSAGE A NIVEAU DIFFICILE',
 'variant': 'SECUROUTE.TE.PND',
 'name': 'GeoportailFrance.Securoute_Te_Pnd',
 'TileMatrixSet': 'PM'},
'Securoute_Te_Te120': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 17,
 'apikey': 'transports',
 'format': 'image/png',
 'style': 'RESEAU ROUTIER TE120',
 'variant': 'SECUROUTE.TE.TE120',
 'name': 'GeoportailFrance.Securoute_Te_Te120',
 'TileMatrixSet': 'PM'},
'Securoute_Te_Te72': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail
 'attribution': 'Geoportail France',
```

```
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 17,
 'apikey': 'transports',
 'format': 'image/png',
 'style': 'RESEAU ROUTIER TE72',
 'variant': 'SECUROUTE.TE.TE72',
 'name': 'GeoportailFrance.Securoute_Te_Te72',
 'TileMatrixSet': 'PM'},
'Securoute_Te_Te94': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQUEST=GetTile&S
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 6,
 'max_zoom': 17,
 'apikey': 'transports',
 'format': 'image/png',
 'style': 'RESEAU ROUTIER TE94',
 'variant': 'SECUROUTE.TE.TE94',
 'name': 'GeoportailFrance.Securoute_Te_Te94',
 'TileMatrixSet': 'PM'},
'Transportnetworks_Roads_Direction': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?R
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[-21.4756, -63.3725], [51.3121, 55.9259]],
 'min_zoom': 15,
 'max_zoom': 18,
 'apikey': 'transports',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'TRANSPORTNETWORKS.ROADS.DIRECTION',
 'name': 'GeoportailFrance.Transportnetworks_Roads_Direction',
 'TileMatrixSet': 'PM'},
'Transports_Drones_Restrictions': {'url': 'https://wxs.ign.fr/{apikey}/geoportail/wmts?REQU
 'html_attribution': '<a target="_blank"href="https://www.geoportail.gouv.fr/">Geoportail F
 'attribution': 'Geoportail France',
 'bounds': [[40.576, -9.88147], [51.4428, 11.6781]],
 'min_zoom': 3,
 'max_zoom': 15,
 'apikey': 'transports',
 'format': 'image/png',
 'style': 'normal',
 'variant': 'TRANSPORTS.DRONES.RESTRICTIONS',
```

```
  'name': 'GeoportailFrance.Transports_Drones_Restrictions',
  'TileMatrixSet': 'PM'}},
 'OneMapSG': {'Default': {'url': 'https://maps-{s}.onemap.sg/v3/{variant}/{z}/{x}/{y}.png',
  'variant': 'Default',
  'min_zoom': 11,
  'max_zoom': 18,
  'bounds': [[1.56073, 104.11475], [1.16, 103.502]],
  'html_attribution': '<img src="https://docs.onemap.sg/maps/images/oneMap64-01.png" style="
  'attribution': '![](https://docs.onemap.sg/maps/images/oneMap64-01.png) New OneMap | Map d
  'name': 'OneMapSG.Default'},
 'Night': {'url': 'https://maps-{s}.onemap.sg/v3/{variant}/{z}/{x}/{y}.png',
  'variant': 'Night',
  'min_zoom': 11,
  'max_zoom': 18,
  'bounds': [[1.56073, 104.11475], [1.16, 103.502]],
  'html_attribution': '<img src="https://docs.onemap.sg/maps/images/oneMap64-01.png" style="
  'attribution': '![](https://docs.onemap.sg/maps/images/oneMap64-01.png) New OneMap | Map d
  'name': 'OneMapSG.Night'},
 'Original': {'url': 'https://maps-{s}.onemap.sg/v3/{variant}/{z}/{x}/{y}.png',
  'variant': 'Original',
  'min_zoom': 11,
  'max_zoom': 18,
  'bounds': [[1.56073, 104.11475], [1.16, 103.502]],
  'html_attribution': '<img src="https://docs.onemap.sg/maps/images/oneMap64-01.png" style="
  'attribution': '![](https://docs.onemap.sg/maps/images/oneMap64-01.png) New OneMap | Map d
  'name': 'OneMapSG.Original'},
 'Grey': {'url': 'https://maps-{s}.onemap.sg/v3/{variant}/{z}/{x}/{y}.png',
  'variant': 'Grey',
  'min_zoom': 11,
  'max_zoom': 18,
  'bounds': [[1.56073, 104.11475], [1.16, 103.502]],
  'html_attribution': '<img src="https://docs.onemap.sg/maps/images/oneMap64-01.png" style="
  'attribution': '![](https://docs.onemap.sg/maps/images/oneMap64-01.png) New OneMap | Map d
  'name': 'OneMapSG.Grey'},
 'LandLot': {'url': 'https://maps-{s}.onemap.sg/v3/{variant}/{z}/{x}/{y}.png',
  'variant': 'LandLot',
  'min_zoom': 11,
  'max_zoom': 18,
  'bounds': [[1.56073, 104.11475], [1.16, 103.502]],
  'html_attribution': '<img src="https://docs.onemap.sg/maps/images/oneMap64-01.png" style="
  'attribution': '![](https://docs.onemap.sg/maps/images/oneMap64-01.png) New OneMap | Map d
  'name': 'OneMapSG.LandLot'}},
 'USGS': {'USTopo': {'url': 'https://basemap.nationalmap.gov/arcgis/rest/services/USGSTopo/Ma
```

```
  'max_zoom': 20,
  'html_attribution': 'Tiles courtesy of the <a href="https://usgs.gov/">U.S. Geological Su
  'attribution': 'Tiles courtesy of the U.S. Geological Survey',
  'name': 'USGS.USTopo'},
 'USImagery': {'url': 'https://basemap.nationalmap.gov/arcgis/rest/services/USGSImageryOnly/
  'max_zoom': 20,
  'html_attribution': 'Tiles courtesy of the <a href="https://usgs.gov/">U.S. Geological Su
  'attribution': 'Tiles courtesy of the U.S. Geological Survey',
  'name': 'USGS.USImagery'},
 'USImageryTopo': {'url': 'https://basemap.nationalmap.gov/arcgis/rest/services/USGSImagery
  'max_zoom': 20,
  'html_attribution': 'Tiles courtesy of the <a href="https://usgs.gov/">U.S. Geological Su
  'attribution': 'Tiles courtesy of the U.S. Geological Survey',
  'name': 'USGS.USImageryTopo'}},
'WaymarkedTrails': {'hiking': {'url': 'https://tile.waymarkedtrails.org/{variant}/{z}/{x}/{
  'max_zoom': 18,
  'html_attribution': 'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">O
  'attribution': 'Map data: (C) OpenStreetMap contributors | Map style: (C) waymarkedtrails
  'variant': 'hiking',
  'name': 'WaymarkedTrails.hiking'},
 'cycling': {'url': 'https://tile.waymarkedtrails.org/{variant}/{z}/{x}/{y}.png',
  'max_zoom': 18,
  'html_attribution': 'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">O
  'attribution': 'Map data: (C) OpenStreetMap contributors | Map style: (C) waymarkedtrails
  'variant': 'cycling',
  'name': 'WaymarkedTrails.cycling'},
 'mtb': {'url': 'https://tile.waymarkedtrails.org/{variant}/{z}/{x}/{y}.png',
  'max_zoom': 18,
  'html_attribution': 'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">O
  'attribution': 'Map data: (C) OpenStreetMap contributors | Map style: (C) waymarkedtrails
  'variant': 'mtb',
  'name': 'WaymarkedTrails.mtb'},
 'slopes': {'url': 'https://tile.waymarkedtrails.org/{variant}/{z}/{x}/{y}.png',
  'max_zoom': 18,
  'html_attribution': 'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">O
  'attribution': 'Map data: (C) OpenStreetMap contributors | Map style: (C) waymarkedtrails
  'variant': 'slopes',
  'name': 'WaymarkedTrails.slopes'},
 'riding': {'url': 'https://tile.waymarkedtrails.org/{variant}/{z}/{x}/{y}.png',
  'max_zoom': 18,
  'html_attribution': 'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">O
  'attribution': 'Map data: (C) OpenStreetMap contributors | Map style: (C) waymarkedtrails
  'variant': 'riding',
```

```
  'name': 'WaymarkedTrails.riding'},
 'skating': {'url': 'https://tile.waymarkedtrails.org/{variant}/{z}/{x}/{y}.png',
  'max_zoom': 18,
  'html_attribution': 'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">Op
  'attribution': 'Map data: (C) OpenStreetMap contributors | Map style: (C) waymarkedtrails
  'variant': 'skating',
  'name': 'WaymarkedTrails.skating'}},
'OpenAIP': {'url': 'https://{s}.tile.maps.openaip.net/geowebcache/service/tms/1.0.0/openaip_
 'html_attribution': '<a href="https://www.openaip.net/">openAIP Data</a> (<a href="https:/
 'attribution': 'openAIP Data (CC-BY-NC-SA)',
 'ext': 'png',
 'min_zoom': 4,
 'max_zoom': 14,
 'tms': True,
 'detectRetina': True,
 'subdomains': '12',
 'name': 'OpenAIP'},
'OpenSnowMap': {'pistes': {'url': 'https://tiles.opensnowmap.org/{variant}/{z}/{x}/{y}.png'
  'min_zoom': 9,
  'max_zoom': 18,
  'html_attribution': 'Map data: &copy; <a href="https://www.openstreetmap.org/copyright">Op
  'attribution': 'Map data: (C) OpenStreetMap contributors & ODbL, (C) www.opensnowmap.org (
  'variant': 'pistes',
  'name': 'OpenSnowMap.pistes'}},
'AzureMaps': {'MicrosoftImagery': {'url': 'https://atlas.microsoft.com/map/tile?api-version=
  'html_attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map-
  'attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map-tile
  'apiVersion': '2.0',
  'variant': 'microsoft.imagery',
  'subscriptionKey': '<insert your subscription key here>',
  'language': 'en-US',
  'name': 'AzureMaps.MicrosoftImagery'},
 'MicrosoftBaseDarkGrey': {'url': 'https://atlas.microsoft.com/map/tile?api-version={apiVers
  'html_attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map-
  'attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map-tile
  'apiVersion': '2.0',
  'variant': 'microsoft.base.darkgrey',
  'subscriptionKey': '<insert your subscription key here>',
  'language': 'en-US',
  'name': 'AzureMaps.MicrosoftBaseDarkGrey'},
 'MicrosoftBaseRoad': {'url': 'https://atlas.microsoft.com/map/tile?api-version={apiVersion}
  'html_attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map-
  'attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map-tile
```

```
  'apiVersion': '2.0',
  'variant': 'microsoft.base.road',
  'subscriptionKey': '<insert your subscription key here>',
  'language': 'en-US',
  'name': 'AzureMaps.MicrosoftBaseRoad'},
 'MicrosoftBaseHybridRoad': {'url': 'https://atlas.microsoft.com/map/tile?api-version={apiV
  'html_attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map
  'attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map-tile
  'apiVersion': '2.0',
  'variant': 'microsoft.base.hybrid.road',
  'subscriptionKey': '<insert your subscription key here>',
  'language': 'en-US',
  'name': 'AzureMaps.MicrosoftBaseHybridRoad'},
 'MicrosoftTerraMain': {'url': 'https://atlas.microsoft.com/map/tile?api-version={apiVersion
  'html_attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map
  'attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map-tile
  'apiVersion': '2.0',
  'variant': 'microsoft.terra.main',
  'subscriptionKey': '<insert your subscription key here>',
  'language': 'en-US',
  'name': 'AzureMaps.MicrosoftTerraMain'},
 'MicrosoftWeatherInfraredMain': {'url': 'https://atlas.microsoft.com/map/tile?api-version=
  'html_attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map
  'attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map-tile
  'apiVersion': '2.0',
  'variant': 'microsoft.weather.infrared.main',
  'subscriptionKey': '<insert your subscription key here>',
  'language': 'en-US',
  'timeStamp': '2021-05-08T09:03:00Z',
  'name': 'AzureMaps.MicrosoftWeatherInfraredMain'},
 'MicrosoftWeatherRadarMain': {'url': 'https://atlas.microsoft.com/map/tile?api-version={api
  'html_attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map
  'attribution': 'See https://docs.microsoft.com/en-us/rest/api/maps/render-v2/get-map-tile
  'apiVersion': '2.0',
  'variant': 'microsoft.weather.radar.main',
  'subscriptionKey': '<insert your subscription key here>',
  'language': 'en-US',
  'timeStamp': '2021-05-08T09:03:00Z',
  'name': 'AzureMaps.MicrosoftWeatherRadarMain'}},
'SwissFederalGeoportal': {'NationalMapColor': {'url': 'https://wmts.geo.admin.ch/1.0.0/ch.sw
  'html_attribution': '<a target="_blank" href="https://www.swisstopo.admin.ch/">swisstopo</
  'attribution': '© swisstopo',
  'bounds': [[45.398181, 5.140242], [48.230651, 11.47757]],
```

```
  'min_zoom': 2,
  'max_zoom': 18,
  'name': 'SwissFederalGeoportal.NationalMapColor'},
 'NationalMapGrey': {'url': 'https://wmts.geo.admin.ch/1.0.0/ch.swisstopo.pixelkarte-grau/de
  'html_attribution': '<a target="_blank" href="https://www.swisstopo.admin.ch/">swisstopo</
  'attribution': '© swisstopo',
  'bounds': [[45.398181, 5.140242], [48.230651, 11.47757]],
  'min_zoom': 2,
  'max_zoom': 18,
  'name': 'SwissFederalGeoportal.NationalMapGrey'},
 'SWISSIMAGE': {'url': 'https://wmts.geo.admin.ch/1.0.0/ch.swisstopo.swissimage/default/curr
  'html_attribution': '<a target="_blank" href="https://www.swisstopo.admin.ch/">swisstopo</
  'attribution': '© swisstopo',
  'bounds': [[45.398181, 5.140242], [48.230651, 11.47757]],
  'min_zoom': 2,
  'max_zoom': 19,
  'name': 'SwissFederalGeoportal.SWISSIMAGE'},
 'JourneyThroughTime': {'url': 'https://wmts.geo.admin.ch/1.0.0/ch.swisstopo.zeitreihen/defa
  'html_attribution': '<a target="_blank" href="https://www.swisstopo.admin.ch/">swisstopo</
  'attribution': '© swisstopo',
  'bounds': [[45.398181, 5.140242], [48.230651, 11.47757]],
  'min_zoom': 2,
  'max_zoom': 18,
  'time': 18641231,
  'name': 'SwissFederalGeoportal.JourneyThroughTime'}},
'Gaode': {'Normal': {'url': 'http://webrd01.is.autonavi.com/appmaptile?lang=zh_cn&size=1&sca
  'max_zoom': 19,
  'attribution': '&copy; <a href="http://www.gaode.com/">Gaode.com</a>',
  'html_attribution': '&copy; <a href="http://www.gaode.com/">Gaode.com</a>',
  'name': 'Gaode.Normal'},
 'Satellite': {'url': 'http://webst01.is.autonavi.com/appmaptile?style=6&x={x}&y={y}&z={z}'
  'max_zoom': 19,
  'attribution': '&copy; <a href="http://www.gaode.com/">Gaode.com</a>',
  'html_attribution': '&copy; <a href="http://www.gaode.com/">Gaode.com</a>',
  'name': 'Gaode.Satellite'}},
'Strava': {'All': {'url': 'https://heatmap-external-a.strava.com/tiles/all/hot/{z}/{x}/{y}.p
  'max_zoom': 15,
  'attribution': 'Map tiles by <a href="https://labs.strava.com/heatmap">Strava 2021</a>',
  'html_attribution': 'Map tiles by <a href="https://labs.strava.com/heatmap">Strava 2021</a
  'name': 'Strava.All'},
 'Ride': {'url': 'https://heatmap-external-a.strava.com/tiles/ride/hot/{z}/{x}/{y}.png',
  'max_zoom': 15,
  'attribution': 'Map tiles by <a href="https://labs.strava.com/heatmap">Strava 2021</a>',
```

```
  'html_attribution': 'Map tiles by <a href="https://labs.strava.com/heatmap">Strava 2021</a
  'name': 'Strava.Ride'},
 'Run': {'url': 'https://heatmap-external-a.strava.com/tiles/run/bluered/{z}/{x}/{y}.png',
  'max_zoom': 15,
  'attribution': 'Map tiles by <a href="https://labs.strava.com/heatmap">Strava 2021</a>',
  'html_attribution': 'Map tiles by <a href="https://labs.strava.com/heatmap">Strava 2021</a
  'name': 'Strava.Run'},
 'Water': {'url': 'https://heatmap-external-a.strava.com/tiles/water/blue/{z}/{x}/{y}.png',
  'max_zoom': 15,
  'attribution': 'Map tiles by <a href="https://labs.strava.com/heatmap">Strava 2021</a>',
  'html_attribution': 'Map tiles by <a href="https://labs.strava.com/heatmap">Strava 2021</a
  'name': 'Strava.Water'},
 'Winter': {'url': 'https://heatmap-external-a.strava.com/tiles/winter/hot/{z}/{x}/{y}.png'
  'max_zoom': 15,
  'attribution': 'Map tiles by <a href="https://labs.strava.com/heatmap">Strava 2021</a>',
  'html_attribution': 'Map tiles by <a href="https://labs.strava.com/heatmap">Strava 2021</a
  'name': 'Strava.Winter'}},
'OrdnanceSurvey': {'Road': {'url': 'https://api.os.uk/maps/raster/v1/zxy/Road_3857/{z}/{x}/{
  'html_attribution': 'Contains OS data &copy Crown copyright and database right 2023',
  'attribution': 'Contains OS data (C) Crown copyright and database right 2023',
  'key': '<insert your valid OS MapsAPI Key. Get a free key here - https://osdatahub.os.uk/
  'min_zoom': 7,
  'max_zoom': 16,
  'max_zoom_premium': 20,
  'bounds': [[49.766807, -9.496386], [61.465189, 3.634745]],
  'name': 'OrdnanceSurvey.Road'},
 'Road_27700': {'url': 'https://api.os.uk/maps/raster/v1/zxy/Road_27700/{z}/{x}/{y}.png?key=
  'html_attribution': 'Contains OS data &copy Crown copyright and database right 2023',
  'attribution': 'Contains OS data (C) Crown copyright and database right 2023',
  'key': '<insert your valid OS MapsAPI Key. Get a free key here - https://osdatahub.os.uk/
  'crs': 'EPSG:27700',
  'min_zoom': 0,
  'max_zoom': 9,
  'max_zoom_premium': 13,
  'bounds': [[0, 0], [700000, 1300000]],
  'name': 'OrdnanceSurvey.Road_27700'},
 'Outdoor': {'url': 'https://api.os.uk/maps/raster/v1/zxy/Outdoor_3857/{z}/{x}/{y}.png?key=
  'html_attribution': 'Contains OS data &copy Crown copyright and database right 2023',
  'attribution': 'Contains OS data (C) Crown copyright and database right 2023',
  'key': '<insert your valid OS MapsAPI Key. Get a free key here - https://osdatahub.os.uk/
  'min_zoom': 7,
  'max_zoom': 16,
  'max_zoom_premium': 20,
```

```
 'bounds': [[49.766807, -9.496386], [61.465189, 3.634745]],
 'name': 'OrdnanceSurvey.Outdoor'},
'Outdoor_27700': {'url': 'https://api.os.uk/maps/raster/v1/zxy/Outdoor_27700/{z}/{x}/{y}.pn
 'html_attribution': 'Contains OS data &copy Crown copyright and database right 2023',
 'attribution': 'Contains OS data (C) Crown copyright and database right 2023',
 'key': '<insert your valid OS MapsAPI Key. Get a free key here - https://osdatahub.os.uk/
 'crs': 'EPSG:27700',
 'min_zoom': 0,
 'max_zoom': 9,
 'max_zoom_premium': 13,
 'bounds': [[0, 0], [700000, 1300000]],
 'name': 'OrdnanceSurvey.Outdoor_27700'},
'Light': {'url': 'https://api.os.uk/maps/raster/v1/zxy/Light_3857/{z}/{x}/{y}.png?key={key}
 'html_attribution': 'Contains OS data &copy Crown copyright and database right 2023',
 'attribution': 'Contains OS data (C) Crown copyright and database right 2023',
 'key': '<insert your valid OS MapsAPI Key. Get a free key here - https://osdatahub.os.uk/
 'min_zoom': 7,
 'max_zoom': 16,
 'max_zoom_premium': 20,
 'bounds': [[49.766807, -9.496386], [61.465189, 3.634745]],
 'name': 'OrdnanceSurvey.Light'},
'Light_27700': {'url': 'https://api.os.uk/maps/raster/v1/zxy/Light_27700/{z}/{x}/{y}.png?ke
 'html_attribution': 'Contains OS data &copy Crown copyright and database right 2023',
 'attribution': 'Contains OS data (C) Crown copyright and database right 2023',
 'key': '<insert your valid OS MapsAPI Key. Get a free key here - https://osdatahub.os.uk/
 'crs': 'EPSG:27700',
 'min_zoom': 0,
 'max_zoom': 9,
 'max_zoom_premium': 13,
 'bounds': [[0, 0], [700000, 1300000]],
 'name': 'OrdnanceSurvey.Light_27700'},
'Leisure_27700': {'url': 'https://api.os.uk/maps/raster/v1/zxy/Leisure_27700/{z}/{x}/{y}.pn
 'html_attribution': 'Contains OS data &copy Crown copyright and database right 2023',
 'attribution': 'Contains OS data (C) Crown copyright and database right 2023',
 'key': '<insert your valid OS MapsAPI Key. Get a free key here - https://osdatahub.os.uk/
 'crs': 'EPSG:27700',
 'min_zoom': 0,
 'max_zoom': 5,
 'max_zoom_premium': 9,
 'bounds': [[0, 0], [700000, 1300000]],
 'name': 'OrdnanceSurvey.Leisure_27700'}}}
```

In Folium, you can pass one of the custom `xyzservices.TileProvider` listed above or a

Leaflet-style URL to the tiles parameter: `http://{s}.yourtiles.com/{z}/{x {y}.png`. You can find a list of free tile providers here: http://leaflet-extras.github.io/leaflet-providers/preview/. Be sure to check their terms and conditions and to provide attribution (see below)

```
map = folium.Map(location=location, zoom_start=13, tiles="CartoDB.DarkMatterNoLabels") # foli
map
```

```
<folium.folium.Map at 0x148eedd9b90>
```

Or we can point to a custom *tileset URL*:

```
map = folium.Map(location=location, zoom_start=12, tiles="https://mt1.google.com/vt/lyrs=r&x=
map
```

```
<folium.folium.Map at 0x148eedd92d0>
```

Also, the *Mapbox Static Tiles API* serves raster tiles generated from Mapbox Studio styles. Raster tiles can be used in traditional web mapping libraries like Mapbox.js, Leaflet, OpenLayers, and others to create interactive slippy maps. The Static Tiles API is well-suited for maps with limited interactivity or use on devices that do not support WebGL.

Usage of the Mapbox APIs is governed by the Mapbox Terms of Service. Please visit https://www.mapbox.com/legal/tos/ for more information.

```
# Your Mapbox access token
mapbox_access_token = 'YOUR_MAPBOX_ACCESS_TOKEN_HERE'

# Mapbox tile URL
tile_url = f'https://api.mapbox.com/styles/v1/mapbox/light-v9/tiles/{{z}}/{{x}}/{{y}}?access_

# Create a folium map
map = folium.Map(location=[53.406872, -2.973286], zoom_start=13, tiles = tile_url, attr='Mapb
map
```

```
<folium.folium.Map at 0x148ed575050>
```

5.1.2 Add a point marker

To add a single marker to a *Folium* map, create a `folium.Marker`. Supply a `folium.Icon` as the argument of the parameter `icon` to influence how the marker is styled, and set `tooltip` to display a text when the mouse pointer hovers over it.

```
center = folium.Marker(location=location, tooltip="City Centroid", icon=folium.Icon(color="g
center.add_to(map)
map
```

```
<folium.folium.Map at 0x148ed575050>
```

```
# Similarly..
popup = ["Tom", "Kendall", "Sean", "Zachary", "Karla"]

# Longitude and Latitude coordinates
lng = [-3.2031323, -0.2416811, -3.4924087, -4.3725404, -2.6607571]
lat = [53.4118332, 51.5285582, 55.940874, 55.8553807, 51.4684681]

# Creating a folium map
map = folium.Map(location=[53.4118332, -3.2031323], zoom_start=5, tiles = "NASAGIBS.ViirsEar

# Adding markers with popups
for i in range(len(popup)):
    folium.Marker([lat[i], lng[i]], popup=popup[i]).add_to(map)

map
```

```
<folium.folium.Map at 0x148ec79ae50>
```

5.1.3 Add a layer of points

Folium also supports to add entire layers, for instance, as `geopandas.GeoDataFrames`. Folium implements *Leaflet*'s `geoJSON` layers in its `folium.features.GeoJson` class. We can initialise such a class (and layer) with a GeoDataFrame, and add it to a map. In the example below, we work again with the GeoDataFrame of terrorist attacks in Germany.

```
wgs = 'EPSG:4326'
gdf = gpd.read_file("../data/germany.shp")
gdf = gdf.set_crs(wgs)
```

Let's explore the data around Berlin

```
berlin = (52.5200, 13.4050)
map = folium.Map(location= berlin, zoom_start=10)

attacks_layer = folium.features.GeoJson(gdf, name="Terrorist attacks")
attacks_layer.add_to(map)
map
```

```
<folium.folium.Map at 0x148eed77810>
```

We can also add a pop-up window to our map which would show some details when clicking on the location of the attacks:

```
popup = folium.GeoJsonPopup(fields=["year", "gname", "nwound"], aliases =["Year", "Terrorist
                            style="background-color: yellow;")
map = folium.Map(location= berlin, zoom_start=10)
attacks_layer = folium.features.GeoJson(gdf[gdf.nwound.notna()], name="Terrorist attacks", po
attacks_layer.add_to(map)

map
```

```
<folium.folium.Map at 0x148ef2d7810>
```

5.1.4 Add a polygon layer

We can do the same with polygons. Let's load municipalities in Serbia.

```
serbia = gpd.read_file("../data/serbia_data.shp")
serbia.head()
```

	name	id	poverty	name_sr	arr_dom	arr_int	pop	male	female	net	g
0	Aranđelovac	1	23.3	Aranđelovac	4042.0	958.0	39456.0	19166.0	20290.0	72348.0	1
1	Batočina	2	36.1	Batočina	11.0	0.0	10085.0	4995.0	5090.0	63243.0	8
2	Knić	3	40.1	Knić	753.0	55.0	12533.0	6312.0	6221.0	65463.0	9
3	Kragujevac	4	23.8	Kragujevac	2244.0	2417.0	154290.0	74515.0	79775.0	79415.0	1
4	Lapovo	5	23.9	Lapovo	39.0	36.0	6785.0	3294.0	3491.0	65350.0	9

Important:

We need to project the GeoDataFrame back to the WGS crs; web-maps, because of their interactivity and because users can pan to different areas, work with the WGS projection and cannot handle planar crs.

```
wgs = 'EPSG:4326'
serbia.to_crs(wgs, inplace = True)
```

We will use the `folium.Choropleth` to display arrivals of tourists in July 2023, per municipality. Choropleth maps are more than simply geometries in Folium, which could be displayed as a `folium.features.GeoJson` layer. Rather, this class takes care of categorising data, adding a legend, and a few more small tasks to quickly create beautiful thematic maps.

The class expects an input data set that has an explicit, `str`-type, index column, as it treats the geospatial input and the thematic input as separate data sets that need to be joined (see also, below, how we specify both `geo_data` and `data`). A good approach to create such a column is to copy the data frame's index into a new column, for instance `id_str`.

```
serbia["id_str"] = serbia.index.astype(str)
```

Now we can create the polygon choropleth layer, and add it to a map object. Due to the slightly complex architecture of *Folium*, we have to supply a number of parameters: - `geo_data` and `data`, the geospatial and thematic input data sets, respectively. Can be the same `GeoDataFrame`. - `columns`: a tuple of the names of relevant columns in `data`: a unique index column, and the column containing thematic data - `key_on`: which column in `geo_data` to use for joining `data` (this is basically identical to `columns`, except it's only the first value)

```
# getting the location
centroid = serbia.unary_union.centroid
location = (centroid.coords[0][1], centroid.coords[0][0])
```

Let's look at the number of arrivals of tourists across the municipalities. - `arr_dom` indicates the number of domestic tourists (from other areas in Serbia) in july 2023. - `arr_int` indicates the number of international tourists in july 2023.

```
serbia[['arr_dom', 'arr_int', 'Province', 'name']].head()
```

	arr_dom	arr_int	Province	name
0	4042.0	958.0	Šumadijski	Aranđelovac
1	11.0	0.0	Šumadijski	Batočina
2	753.0	55.0	Šumadijski	Knić

	arr_dom	arr_int	Province	name
3	2244.0	2417.0	Šumadijski	Kragujevac
4	39.0	36.0	Šumadijski	Lapovo

```
map = folium.Map(location=location, zoom_start=7)
arrivals_choro = folium.Choropleth(geo_data=serbia, data=serbia, columns=("id_str", "arr_int"
                                   fill_color="Oranges", line_weight=0.2, line_color='black
                                   legend_name="Nr of International Tourists, July 2023",
                                   highlight = True) ## look at what this does!
arrivals_choro.add_to(map)
map
```

```
<folium.folium.Map at 0x148ef2e8610>
```

In such an interactive map, it would be nice to display some property of each polygon when hovering with the mouse over it. `folium.Choropleth` does not support this. Yet, we can add a "transparent" layer using `folium.features.GeoJson`, and configure it to display `GeoJsonTooltip`. We can keep the `map` we created above, and simply add the layer to it.

```
# folium GeoJson layers expect a styling function, that receives each of the map's feature ar
# It can, however, also return a static style:
def style_function(feature):
    return {
        "color": "transparent",
        "fillColor": "transparent",
    }

tooltip = folium.features.GeoJsonTooltip(fields=("name",), aliases=("Municipality:",))
tooltip_layer = folium.features.GeoJson(serbia, style_function=style_function, tooltip=toolti
tooltip_layer.add_to(map)
map
```

```
<folium.folium.Map at 0x148ef2e8610>
```

The problem now is that we cannot highlight the polygons and fully figure out the boundaries of the areas we are hovering on. However, we can also tackle that. First we create again our map.

```
# base map
map = folium.Map(location=[serbia.geometry.centroid.y.mean(), serbia.geometry.centroid.x.mean
```

```
C:\Users\gfilo\AppData\Local\Temp\ipykernel_13356\3770209863.py:2: UserWarning: Geometry is i

  map = folium.Map(location=[serbia.geometry.centroid.y.mean(), serbia.geometry.centroid.x.me
```

Then, the choropleth object as before. This time we also set `use_jenks` as `True`. The number of desired classes needs to be passed by using the parameter `bins`.

```
# Add the choropleth layer
arrivals_choro = folium.Choropleth(geo_data=serbia, data = serbia, columns=['id_str', 'arr_in
                                   use_jenks = True, bins = 7, fill_color='Oranges',line_wei
                                   fill_opacity=0.7, legend_name='Arrivals Intensity').add_t
                                   ## no highlight here
```

Then, below, we define two functions that always take a feature as a parameter (each geographic element on the map, the municipality polygons).

1. **`style_function`**:
    - Purpose: Defines the default style for each polygon on the map.
    - Sets the fill color, border color, border weight, and fill opacity for polygons. The provided colors and values are applied to all polygons by default.
2. **`highlight_function`**:
    - Purpose: Changes the style of a polygon when it is hovered over.
    - Alters the fill color and opacity when the mouse hovers over a polygon, making it stand out.

```
# Define style function
def style_function(feature):
    return {
        'fillColor': arrivals_choro.color_scale(feature['properties']['arr_int']),  # fill co
        'color': 'black',  # Border color
        'weight': 0.5, # Border weight
        'fillOpacity': 0.7 # Fill color opacity
    }

# Define highlight function
def highlight_function(feature):
```

```
    return {
        'fillColor': '#ff0000',  # Fill color when highlighted
        'color': 'black',  # Border color when highlighted
        'weight': 0.5, # Border weight when highlighted
        'fillOpacity': 1.0 # Fill color opacity
    }
```

Essentially, here, we use the `choropleth`, `arrivals_choro` just to get the colours assigned to the feature, on the basis of the `bins` that we created with the scheme. Then we pass these colors to the same features in the `folium.GeoJson` class below. In this case the layer is not transparent. Finally, as before, we add a `tooltip` and a `popup`.

The popup appears when the polygon is clicked but this aspect can be personalised too.

```
# Add popups and pass styling options
folium.GeoJson(serbia, style_function = style_function,
               highlight_function=highlight_function,
               tooltip= folium.features.GeoJsonTooltip(fields=['name']),
               popup=folium.features.GeoJsonPopup(fields=['arr_dom', 'arr_int'], aliases = [
    ).add_to(map)

map
```

```
<folium.folium.Map at 0x148ef30c7d0>
```

5.2 Interactive Maps and API

Let's go back to the Bike Points example we starting looking at in the Web's architecture session.

```
import requests
import pandas as pd
import geopandas as gpd
from shapely.geometry import Point

# Request to Transport for London API for BikePoint data
response = requests.get("https://api.tfl.gov.uk/BikePoint/")
bike_stations = pd.DataFrame(response.json())
```

We extract the numeric part from the `id` column

```
bike_stations['stationID'] = [id.split('_')[1] if '_' in id else None for id in bike_stations
bike_stations['stationID'] = bike_stations['stationID'].astype('int64')
```

and we create a `GeoDataFrame`

```
bike_stations_gdf = gpd.GeoDataFrame(bike_stations, geometry=[Point(xy) for xy in zip(bike_st
bike_stations_gdf.set_crs('EPSG:4326', inplace = True)  # Set CRS
bike_stations_gdf.head()
```

	$type	id	url	commonName
0	Tfl.Api.Presentation.Entities.Place, Tfl.Api.P...	BikePoints_1	/Place/BikePoints_1	River Street , Cle
1	Tfl.Api.Presentation.Entities.Place, Tfl.Api.P...	BikePoints_2	/Place/BikePoints_2	Phillimore Garde
2	Tfl.Api.Presentation.Entities.Place, Tfl.Api.P...	BikePoints_3	/Place/BikePoints_3	Christopher Stree
3	Tfl.Api.Presentation.Entities.Place, Tfl.Api.P...	BikePoints_4	/Place/BikePoints_4	St. Chad's Street
4	Tfl.Api.Presentation.Entities.Place, Tfl.Api.P...	BikePoints_5	/Place/BikePoints_5	Sedding Street, S

Now let's add some data about trips made by hire bikes. We need to use the station IDs for the beginning and end of the trips. Transport for London publishes online all trips made by hire bikes along many other datasets related to bike usage in London. The files are published weekly. They have information on starting and ending stations, exact time of the trips.

We can download the files for August 2018 and do some cleaning to map the most used routes in London. We first need to filter for completed trips and select trips with different origins/destinations. The next step is to aggregate the trips by pairs of origin and destination stations. The results should be how many trips have originated and ended from a specific pair in August 2018.

```
import os

# URLs of hire bikes data in August 2018
urls = [
    "https://cycling.data.tfl.gov.uk/usage-stats/121JourneyDataExtract01Aug2018-07Aug2018.csv
    "https://cycling.data.tfl.gov.uk/usage-stats/122JourneyDataExtract08Aug2018-14Aug2018.csv
    "https://cycling.data.tfl.gov.uk/usage-stats/123JourneyDataExtract15Aug2018-21Aug2018.csv
    "https://cycling.data.tfl.gov.uk/usage-stats/124JourneyDataExtract22Aug2018-28Aug2018.csv
    ]

# Destination folder
folder = "../data/tfl"
```

```
# Create the destination folder if it doesn't exist
os.makedirs(folder, exist_ok=True)

# Function to download and save a file
def download_file(url, folder):
    response = requests.get(url)
    filename = url.split('/')[-1]
    filepath = os.path.join(folder, filename)
    with open(filepath, 'wb') as file:
        file.write(response.content)

# Download each file
for url in urls:
    download_file(url, folder)
```

The code above performs the following actions:

- Define URLs: A list of URLs is created, each pointing to a CSV file containing data on hire bikes for different weeks in August 2018.
- Destination Folder: It sets a destination folder where the downloaded files will be stored and creates the directory if it doesn't exist
- Define Download Function: `download_file` is a function defined to download a file from a given URL and save it to the specified destination folder. - It uses `requests.get` to fetch the file and writes it to the destination folder.
- Download: A loop goes through each URL in the urls list and calls `download_file` to download and save each file in the defined destination folder.

Now let's load them with `glob`, a library that loads all the files from a directory, and then create the dataframe of journeys.

```
import glob

# Reading all CSV files in the directory
files = glob.glob(f'{folder}/*JourneyDataExtract*.csv')

# Concatenating the CSV files into a single DataFrame
journeys = pd.concat((pd.read_csv(file) for file in files))
journeys.rename(columns = {"Duration" : "duration", "StartStation Id": "fromStation_id", "End
journeys = journeys[journeys['fromStation_id'] != journeys['toStation_id']] # same station j
journeys.head()
```

	Rental Id	duration	Bike Id	End Date	toStation_id	EndStation Name	S
0	78826223	780	13775	02/08/2018 19:13	291	Claverton Street, Pimlico	0
1	78986704	660	3498	06/08/2018 17:45	742	Blenheim Crescent, Ladbroke Grove	0
2	78990055	1620	922	06/08/2018 18:36	609	Sugden Road, Clapham	0
3	78940313	5340	12074	05/08/2018 17:27	441	Sail Street, Vauxhall	0
4	79004458	1440	13220	07/08/2018 07:32	309	Embankment (Savoy), Strand	0

Finally, let's get a summary dataframe thanks to `groupby` and other `pandas` functions.

```
# Filter and aggregate the data
journeys_agg = (
    journeys.dropna(subset=['toStation_id', 'fromStation_id'])
    .query("`fromStation_id` in @bike_stations['stationID'] and `toStation_id` in @bike_stati
    .query("duration > 0 and duration <= 180*60")
    .groupby(['fromStation_id', 'toStation_id'])
    .agg(journeys=('fromStation_id', 'count'), mean_duration=('duration', 'mean'))
    .reset_index()
    .assign(share_trips=lambda x: (100 * x.journeys / x.journeys.sum()))
)
```

What we did above is combining different pandas functions:

- `.dropna(subset=['toStation_id', 'fromStation_id'])`: Removes rows where either the start or end station ID is missing.
- `.query("fromStation_idin @bike_stations['stationID'] and toStation_id in @bike_stations['stationID']")`: Further filters the data to include only those journeys where both the start and end stations are listed in bike_stations.
- `.query("duration > 0 and duration <= 180*60")`: Keeps only those journeys with a duration greater than 0 and less than or equal to 180 minutes.
- `.groupby(['fromStation_id', 'toStation_id'])`: Groups the data by the combination of start and end station IDs.
- `.agg(journeys=('fromStation_id', 'size'), mean_duration=('duration', 'mean'))`: Aggregates each group to calculate the total number (`count`) and average duration (`mean`) of journeys.
- `.reset_index()`: Resets the DataFrame's index, turning the grouped columns back into regular columns.
- `.assign(share_trips=lambda x: 100 * x.journeys / x.journeys.sum())`: Creates a new column `share_trips`, calculating each route's share of total journeys as a percentage.

The `.query()` method in `pandas` allows you to filter a DataFrame using a query string. It's a more concise and readable way to perform filtering compared to traditional Boolean indexing. It takes the following syntax: `.query('expression')` where 'expression' is a string that describes the condition to filter by. Inside the expression, you can refer to DataFrame columns directly by their names and use logical operators (like ==, !=, >, <, in, and, or) to define the conditions. **Example:** `.query('age > 30 and city == "New York"')` filters the DataFrame to include only rows where the 'age' column is greater than 30 and the 'city' column equals "New York"

No need to remember this!

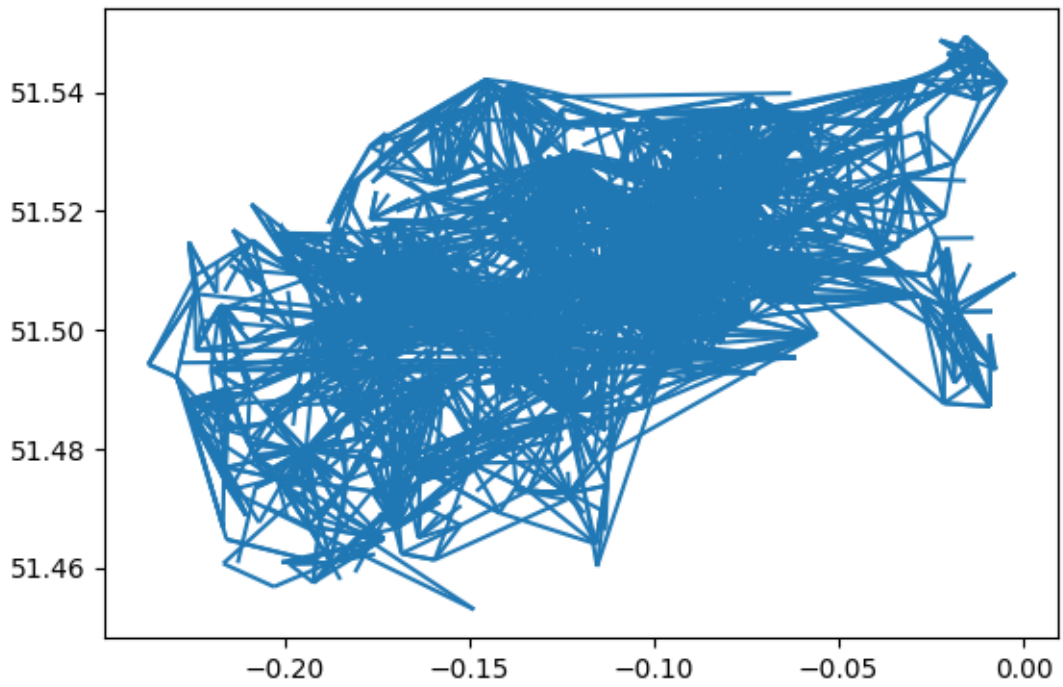
```
journeys_agg[['journeys', 'mean_duration', 'share_trips']].describe()
```

	journeys	mean_duration	share_trips
count	178345.000000	178345.000000	178345.000000
mean	4.859817	1296.806824	0.000561
std	9.243570	927.149791	0.001066
min	1.000000	60.000000	0.000115
25%	1.000000	780.000000	0.000115
50%	2.000000	1120.000000	0.000231
75%	5.000000	1530.000000	0.000577
max	679.000000	10800.000000	0.078341

Most origin/destination pairs have an average of 4.80 trips during the period. The average duration is around 21 min (1297/60). We can then filter our journeys to the top 2 percentiles in terms of counts between two stations. Most pairs do not have any trips (none goes from the furthest station in Hackney down to Oval station).

```
import numpy as np
from shapely.geometry import LineString

# Calculate the 98th percentile value for the number of journeys
percentile_value = np.percentile(journeys_agg['journeys'], 98)

# Filter out the top 2% of journeys based on the number of journeys
top_journeys = journeys_agg[journeys_agg['journeys'] >= percentile_value]

# Continuing with the merging and creation of desire lines as before
top_journeys = top_journeys.merge(bike_stations, left_on='fromStation_id', right_on='station
top_journeys = top_journeys.merge(bike_stations, left_on='toStation_id', right_on='stationID
```

```
geometry = [LineString([(row['lon_from'], row['lat_from']), (row['lon_to'], row['lat_to'])])
desire_lines = gpd.GeoDataFrame(top_journeys, geometry=geometry, crs = 'EPSG:4326')

desire_lines.plot()
```



Plotting all lines is quite messy but we can use `folium` again :) Unfortunately, we can't use `Choropleth` for `LineString` geometries. We have to carry out some manual work.

```
import json
import jenkspy
import matplotlib
import matplotlib.pyplot as plt
# Convert the desire_lines GeoDataFrame to GeoJSON
desire_lines_geojson = desire_lines.to_json()

# breaks with fisher-jenks
breaks = jenkspy.jenks_breaks(desire_lines['journeys'], n_classes=5)

# Use a Matplotlib colormap
cmap = plt.get_cmap('Oranges', len(breaks))  # 'viridis' can be replaced with any Matplotlib
```

```
# Generate colors for each break
colors = [matplotlib.colors.rgb2hex(cmap(i)) for i in range(cmap.N)]
cmap
```

Let's define again a styling function for our lines.

```
# Define the style function using the generated colors
def assign_color(value):
    for i, break_point in enumerate(breaks[:-1]):
        if value < breaks[i+1]:
            return colors[i]
    return colors[-1]

style_function = lambda feature: {
    'color': assign_color(feature['properties']['journeys']),
    'weight': feature['properties']['share_trips']*150,
    'opacity': max(feature['properties']['share_trips']*25, 0.05)
}
```

```
# Create and display the Folium map
map = folium.Map(location=[51.5074, -0.1278], zoom_start=12, tiles="CartoDB.DarkMatterNoLabel
folium.GeoJson(desire_lines_geojson, style_function=style_function).add_to(map)
map
```

```
<folium.folium.Map at 0x14882935e10>
```

Exercise: Look at the documentation for one of the APIs that we discussed last week. - Think of 2 ideas of web maps you could construct from your chosen API/data. - Think about the basemap and the data you would plot on top.

Folium is just one of many packages that provide an easy way to create interactive maps using data stored in (geo-)pandas `DataFrames`. Other interesting libraries include:

- GeoViews.
- Bokeh.
- kepler.gl.
- pydeck.

Nevertheless, Folium allows for a huge degree of personalisation and it's worth exploring further its documentation.

6 Data Architectures and Tiles

The **Lecture slides** can be found here.

This **lab**'s notebook can be downloaded from here.

In this lab, we will explore and familiarise with some of the most common data formats for web mapping: GeoJSON and Mbtiles.

6.1 GeoJSON

To get familiar with the format, we will start by creating a GeoJSON file from scratch. Head over to the following website:

https://geojson.io/

In there, we will create together a small example to better understand the building blocks of this file format.

We will pay special attention to the following aspects:

- Readability.
- Coordinate system.
- Ability to add non-spatial information attached to each record.
- How to save it as a file.

Excercise:

Create a GeoJSON file for the following data and save them to separate files:

1. Your five favourite spots in Liverpool
2. A polygon of what you consider to be the boundary of the neighbourhood where you live and the city centre of Liverpool. Name each.
3. A route that captures one of your favourite walks around the Liverpool region

If you are comfortable, upload the files to Microsoft Teams to share them with peers.

6.1.1 GeoJSON in Python

With the files from the exercise at hand, we will learn how to open them in a Python environment. Then, let's begin by importing the necessary libraries; `geojson` is used for handling GeoJSON files.

```
import geopandas as gpd
```

Now, place the geojson files you have created in the data folder used in these sessions. As always, the data folder should be stored in the directory where the notebook is running from. For this example, we will assume that the file is called `map.geojson`. We can read the file as:

```
liverpool = gpd.read_file("../data/map.geojson")
liverpool.head()
```

We can also plot and explore the content of the GeoDataFrame with `Folium`. Folium, which we will see more in detail later on, helps create interactive maps from data stored in `geopandas.GeoDataFrame`.

```
import folium

liverpool_centroid = (53.41058, -2.97794)
# Create a Folium map centered around this point
map = folium.Map(location=liverpool_centroid, zoom_start=13, tiles="CartoDB.DarkMatterNoLabel

# Add the liverpool data to the map, this will plot each geometry in the GeoDataFrame
folium.GeoJson(liverpool).add_to(map)
map
```

Once read, the geojson behaves exactly like any GeoDataFrame we have seen so far. We can therefore operate on it and tap into the functionality from `pandas` and `geopandas`. For example, we can and reproject the layer to the to British National Grid.

```
liverpool_bng = liverpool.to_crs(crs = "EPSG: 27700")
```

When we inspected our geojson, we noted that the spatial data is stored in the following format `POINT (-2.977367 53.40753)`. This is called "well known text" (`wkt`) and is a representation that spatial databases like PostGIS use as well. Another way to store spatial data as text for storage or transfer, less (human) readable but more efficient is the "well known blurb" (`wkb`). We can use the `shapely` library to handle the WKT representation of the geometry and then convert it to WKB format.

```
from shapely import wkt
from shapely.geometry import Point
import shapely.wkb

# Load the WKT representation of the point
wkt_string = "POINT (-2.977367 53.40753)"

# Convert the WKT representation into a Shapely Point object
point = wkt.loads(wkt_string)

# Convert the Point object into WKB format
wkb_data = shapely.wkb.dumps(point)
wkb_data.hex()
```

Excercise: - Read the `GeoJSON` created for your favorite walks in Liverpool and calculate their length

Once you are happy with the data as we will hypothetically need it, you can write it out to any other file format supported in `geopandas`. For example, we can create a Geopackge file with the same information. For this, we can use the function `to_file`. See an example below:

```
# Write 'liverpool_bng' to a GeoPackage file
liverpool_bng.to_file("../data/liverpool_bng.gpkg", layer="liverpool_bng", driver="GPKG")
```

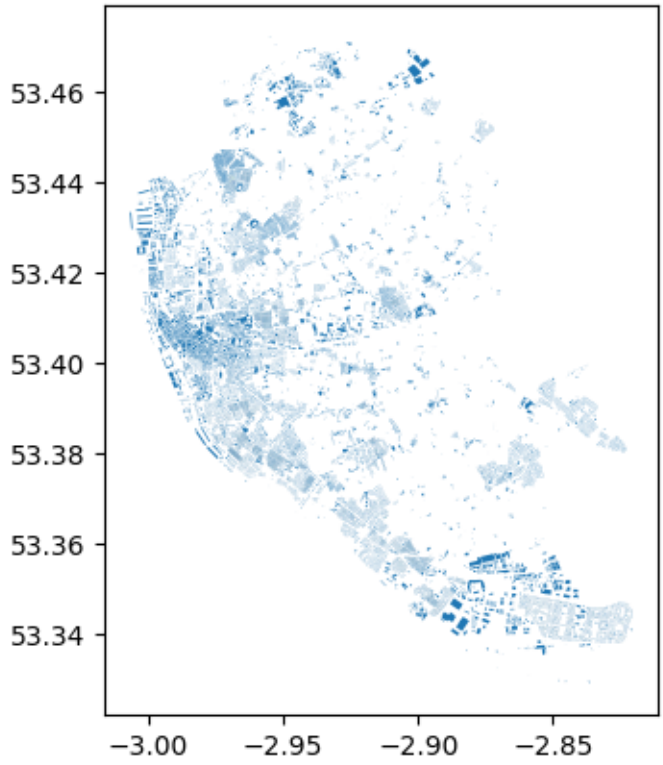
6.2 Tilesets and `Mbtiles`

In this section we will dive into the concept of tiles to understand why they have been so transformative in the world of web mapping. We have already seen the usage of tilesets above with `folium` and with `contextily` (although within a static context). We will see that `folium`, integrates different tileset options already.

For this section, let's start by getting the building footprints from OpenStreetMap with `osmnx`

```
import osmnx as ox
tags = {"building": True} #OSM tags
buildings = ox.features_from_place("Liverpool, UK", tags = tags)
buildings = buildings.reset_index()
 # sometimes building footprints are represented by Points, let's disregard them
buildings = buildings[(buildings.geometry.geom_type == 'Polygon') | (buildings.geometry.geom
```

```
buildings.plot()
```



Let's save the GeoDataFrame a geojson and call it `buildings_liverpool.geojson`.

```
buildings[['osmid','geometry']].to_file('data/buildings_liverpool.geojson',  driver='GeoJSON
```

Then, register for a MapBox account here.

6.2.1 Optional: Generating .mbtiles in Python

In Python, you can use `togeojsontiles` to make dynamic `.mbtiles` files. This is useful to visualize large data appropriately at any zoom level. **This step cannot be ran on University Machines** It requires the installation of `tippecanoe` on your machine. Follow the corresponding instructions for Windows and for MAC

```
pip install togeojsontiles
from togeojsontiles import geojson_to_mbtiles
```

```
TIPPECANOE_DIR = '/usr/local/bin/'

# Convert GeoJSON to .mbtiles
togeojsontiles.geojson_to_mbtiles(
    filepaths=['/data/buildings_liverpool.geojson'],
    tippecanoe_dir=TIPPECANOE_DIR,
    mbtiles_file='liverpool.mbtiles',
    maxzoom=14)
```

You can also try **uploading** it directly to Mapbox. This part of the code has not been tested so it should serve as guidance.

```
import requests
import os

# Define your Mapbox token
mapbox_token = ""

# Endpoint for Mapbox Tiling Service uploads
url = 'https://api.mapbox.com/uploads/v1/mapbox'

# Path to your .mbtiles file
mbtiles_file_path = "../data/liverpool.mbtiles'

# Prepare the headers
headers = {
    'Authorization': f'Bearer {MAPBOX_ACCESS_TOKEN}',
    'Content-Type': 'application/json'
}

# Prepare the data for the POST request
with open(mbtiles_file_path, 'rb') as file:
    files = {'file': file}
    response = requests.post(url, headers=headers, files=files)

# Check the response
if response.status_code == 200:
    print("Upload initiated successfully.")
    print(response.json())
else:
    print("Failed to initiate upload.")
    print(response.text)
```

The optional steps end here

6.2.2 Uploading to Mapbox Studio:

After creating the `.mbtiles` file, you can upload it *manually* to Mapbox Studio (unless you managed to make the cell above work):

- Navigate to Mapbox Studio.
- Start a New Style and chose a template (Monochrome, blank etc.)
- Upload the `liverpool.mbtiles` (if you created it) or the `liverpool.geojson`. Press the + symbol, `Custom Layer` or `Data Visualisation`, `Upload Data`, and choose your file
- Once uploaded, you should see your layer in the existing sources.
- Style it according to your requirements: It should look like a nicer version of this:

Or through *scripting* with Mapbox's Uploads API (see above)

6.2.3 Visualizing the Tiles with `Folium`:

First, we need to get the `id` of your Mapbox style by copying its url:

url example: mapbox://styles/gabrif/clrpby9lc00a501pecacz8b7h

the `style_id` is the last section of the URL, in this case `clrpby9lc00a501pecacz8b7h`

Then, we can use `folium`, which we will more in detail later on, and use our tileset. Folium essentially allows us to create interactive maps, usually starting from data stored in a `GeoDataFrame`. While folium provides a series of built-in tilesets, here it is demonstrated how to employ one's own.

```
# Create a Folium map centered at a specific location
import folium

mapbox_user = 'gabrif'
mapbox_token = ''
style_id = 'clrpby9lc00a501pecacz8b7h'

tiles = 'https://api.mapbox.com/styles/v1/'+mapbox_user+'/'+style_id+'/tiles/256/{z}/{x}/{y}'
map = folium.Map(location=[53.406872, -2.973286], zoom_start=14,
                 tiles=tiles,
                 attr='Mapbox')
```

```
# Display the map
map
```

```
<folium.folium.Map at 0x2b2b49b31d0>
```

7 Retrieving Data From OpenStreetMap

Gabriele Filomena has readapted parts of this notebook for preparing this notebook. Copyright (c) Michael Szell. Original sources include:

- OSMnx examples: https://github.com/gboeing/osmnx-examples
- pyrosm examples: https://pyrosm.readthedocs.io/en/latest/basics.html#read-street-networks

The **Lecture slides** can be found here.

This **lab**'s notebook can be downloaded from here.

7.1 What is OpenStreetMap?

OpenStreetMap is a free and open map service. It is a collaborative global effort to collect free and open geodata. *Source: wiki.openstreetmap.org*. OpenStreetMap (OSM) is a global collaborative (crowd-sourced) database and project that aims at creating a free editable map of the world containing of information about our environment. It contains data about streets, buildings, different services, and landuse, to mention just a few.

OSM has more than 8 million registered users who contribute around 4 million changes daily. Its database contains data that is described by more than 7 billion nodes (that make up lines, polygons and other objects). While the most well-known side of OpenStreetMap is the map itself, the project is much more than that. OSM's data can be used for many other purposes such as **routing**, **geocoding**, **education**, and **research**. OSM is also widely used for humanitarian response, e.g., in crisis areas (e.g. after natural disasters) and for fostering economic development. Read more about humanitarian projects that use OSM data from the Humanitarian OpenStreetMap Team (HOTOSM) website.

7.1.1 Main tools in this lesson

7.1.1.1 OSMnx

This week we will explore a Python package called `OSMnx` that can be used to retrieve street networks from OpenStreetMap, and construct, analyse, and visualise them. `OSMnx` can also

fetch most of the other data stored in OSM, such as building footprints, transport networks, parks, Points of Interest, etc..`OSMNx` also includes tools to find routes on a network downloaded from OpenStreetMap, and implements algorithms for finding shortest connections for walking, cycling, or driving.

To get an overview of the capabilities of the package, please refer to the following scientific article describing the package:

> Boeing, G. 2017. "OSMnx: New Methods for Acquiring, Constructing, Analyzing, > and Visualizing Complex Street > Networks." Computers, Environment and Urban Systems 65, 126-139. doi:10.1016/j.compenvurbsys.2017.05.004

7.1.1.2 NetworkX

We will also use `NetworkX` to manipulate and analyse the street network data retrieved from OpenStreetMap. NetworkX is a Python package that can be used to create, manipulate, and study the structure, dynamics, and functions of complex networks. `OSMNx` is built on top `NetworkX` and `GeoPandas`.

```
import geopandas as gpd
import osmnx as ox
import numpy as np
import networkx as nx
import pandas as pd
import matplotlib.pyplot as plt
```

7.2 Download, manipulate, and visualise OpenStreetMap data with OSMnx

A useful feature of `OSMNx` is its easy-to-use tools to download OpenStreetMap data via the project's OverPass API. In this section, we will learn how to download and visualise the street network and additional data from OpenStreetMap covering different areas of interest.

7.2.1 Boundaries from OpenStreetMap

`OSMnx` lets you download place boundary geometries from OpenStreetMap, project them, and plot them. For a more in-depth demonstration of querying by place, see this notebook.

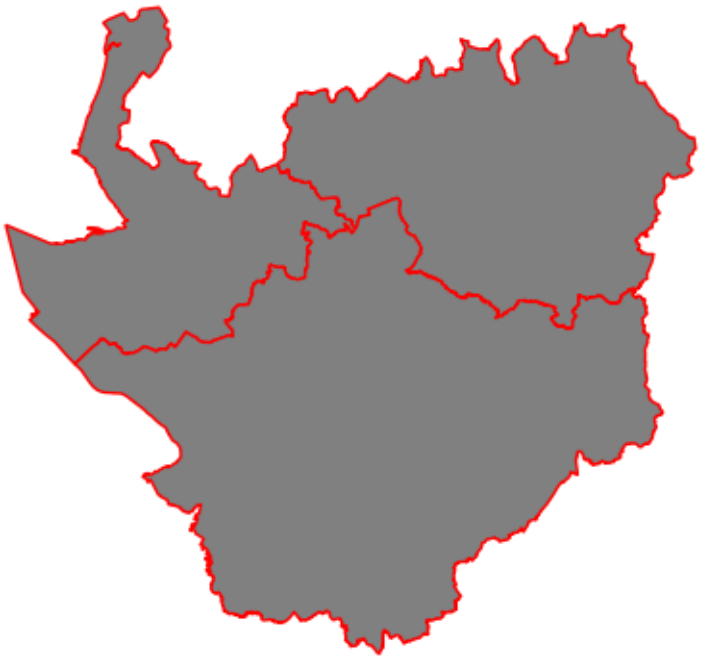
Important:

Data downloaded from OSM are always in the WGS crs. You need to convert it for any type of spatial operation or computation to the appropriate coordinate reference system.

```
# get the boundary polygon for manhattan, project it, and plot it
city = ox.geocode_to_gdf("Manhattan")
ax = city.plot(fc="gray", ec="none")
ax.axis("off")
```

```
bulgaria = ox.geocode_to_gdf("Bulgaria")
ax = bulgaria.plot(fc="gray", ec="none")
ax.axis("off")
```

```
# get boundary polygons for several authorities in the UK, and plot
place_names = ["Merseyside", "Greater Manchester", "Cheshire"]
places = ox.geocode_to_gdf(place_names)
ax = places.plot(fc="gray", ec="red")
_ = ax.axis("off")
```



7.2.2 Download and model Street Networks

The `osmnx.graph`module downloads data to construct a routable road network graph, based on an user-defined area of interest. This area of interest can be specified, for instance, using a place name, a bounding box, or a polygon. In the place name query, OSMnx uses the Nominatim Geocoding API. This means that place names should exist in the OpenStreetMap database (run a test search at openstreetmap.org or nominatim.openstreetmap.org). `OSMnx`, amongst its several functionalities, lets you analyse, plot, and export the network in different file formats. The downloaded street networks are by default **directed** and preserve one-way directionality. For a more in-depth demonstration of creating street networks, see this notebook.

You can download a street network by providing OSMnx any of the following (demonstrated in the examples below): - a bounding box - a lat-lon point plus a distance - an address plus a distance - a place name or list of place names (to automatically geocode and get the boundary of) - a polygon of the desired street network's boundaries - a .osm formatted xml file

You can also specify several different network types: - `drive` - get drivable public streets (but not service roads) - `drive_service` - get drivable streets, including service roads - `walk` - get all streets and paths that pedestrians can use (this network type ignores one-way directionality) - `bike` - get all streets and paths that cyclists can use - `all` - download all non-private

OSM streets and paths (this is the default network type unless you specify a different one) - `all_private` - download all OSM streets and paths, including private-access ones

7.2.2.1 Method #1: Passing a bounding box

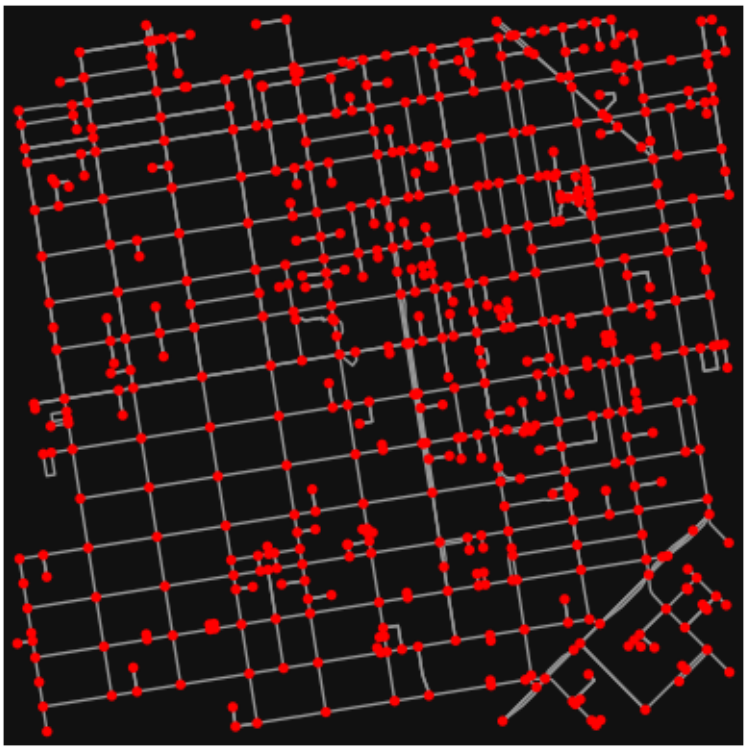
This constructs the network from all the OSM nodes and ways within the bounding box. The plot function is built on `Matplotlib`, thus it follows the procedures and methods discussed in session II.

```
# define a bounding box in San Francisco
north, south, east, west = 37.79, 37.78, -122.41, -122.43
# define a bounding box around ITU
north, south, east, west = 55.6646, 55.6540, 12.5767, 12.6077

# create network from that bounding box
G = ox.graph_from_bbox(north, south, east, west, network_type="drive_service")
ox.plot_graph(G, node_color="r", figsize=(8, 8))
```

7.2.2.2 Method #2: Passing a lat-lon point and bounding box distance in meters

This creates a bounding box n meters North, South, East, and West of the point, then constructs the network from all the OSM nodes and ways within the bounding box. Here's a useful tool for defining bboxes

```
# define a point at the corner of California St and Mason St in SF
location_point = (37.791427, -122.410018)
# create bikeable network from point, inside bounding box of N, S, E, W each 750m from point
G = ox.graph_from_point(location_point, dist=750, dist_type="bbox", network_type="bike")
ox.plot_graph(G, node_color="r", figsize=(5,5))
```



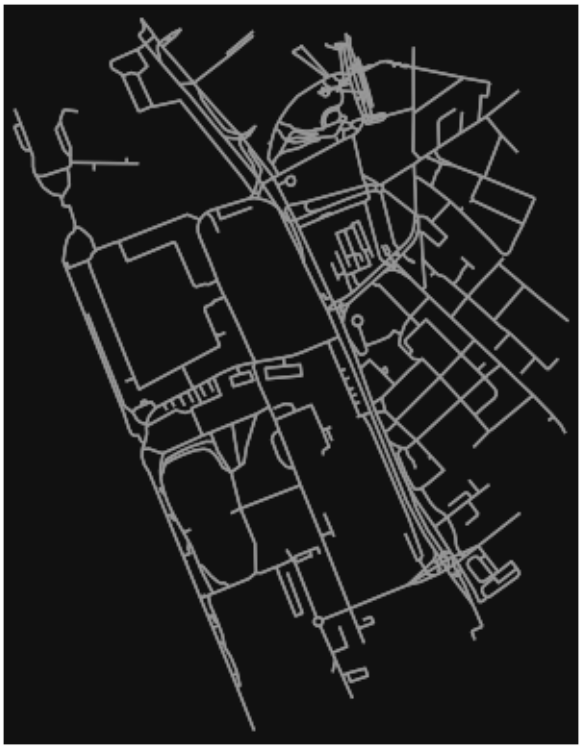
7.2.2.3 Method #3: Passing a lat-lon point and *network* distance in meters

This creates a bounding box n meters North, South, East, and West of the point, then constructs the network from all the OSM nodes and ways within the bounding box. Then it truncates the network by removing all nodes further than n meters from the point along the network.

```
location_point = (53.40, -2.99)
# create network only of nodes within 750m along the network from point
G1 = ox.graph_from_point(location_point, dist=1000, dist_type="network")
ox.plot_graph(G1, node_color="none", figsize=(5,5))
```

Note the plot above shows the network within 750m (traveling distance along the network) from the `location_point`. By default, the `network_type` parameter value is `all`, meaning that we do not filter out paths that restrict certain types of traffic. This also means that one-way streets are honored as one-way and you cannot travel the wrong direction down them. Thus, the 750m takes into account only those nodes you can reach within 500m while only traveling in the allowed direction of the street. Instead (below), we can specify `network_type='walk'` to build a street network only of paths that walking is allowed on. This also makes every path bi-directional in the directed network, because you can walk in either direction on the sidewalk of a one-way street. The 750m now takes into account those nodes you can reach within 750m while traveling in either direction (even if it's a one-way street).

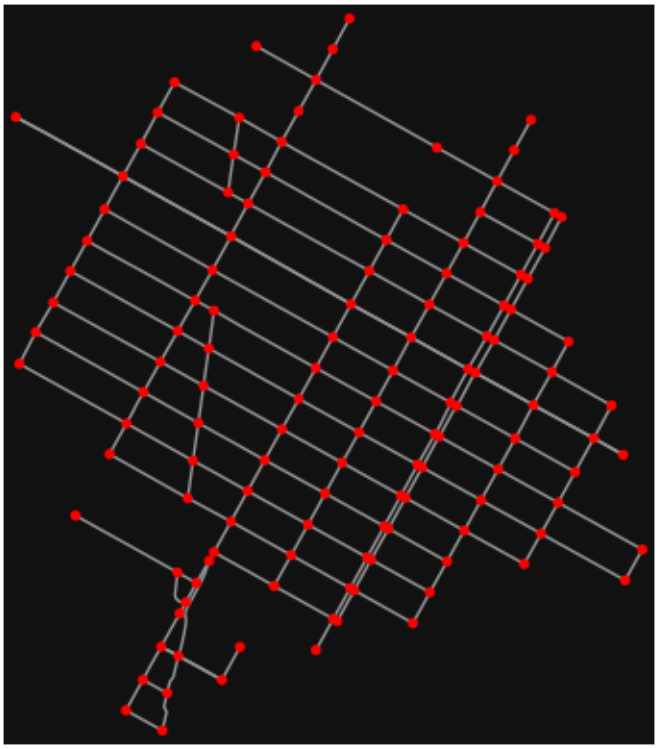
```
# create network only of nodes within 750m walking along the network from point, only walkabl
G2 = ox.graph_from_point(location_point, dist=750, dist_type="network", network_type="walk")
ox.plot_graph(G2, node_color="none", figsize=(5,5))
```



7.2.2.4 Method #4, Passing an address and distance (*bounding box* or *network*) in meters

This geocodes the address, creates a bounding box, downloads the network, then truncates it by network distance (if distance_type='network').

```
# network from address, including only nodes within 1km along the network from the address
G = ox.graph_from_address(address="350 5th Ave, New York, NY", dist=1000, dist_type="network"
ox.plot_graph(G, node_color="r", figsize=(5,5))
```



7.2.2.5 Method #5: Passing a place name

This geocodes the place name, gets the place's boundary shape polygon and bounding box, downloads the network within the bounding box, then truncates it to the place's boundary polygon.

```
G = ox.graph_from_place("Bastia, Corsica", network_type="drive")
ox.plot_graph(G, node_color="none") # without plotting nodes
```

Be aware that this is a `MultiDiGraph`, i.e. a multigraph (parallel edges are possible) that is directed. Because there can be multiple links between a pair of nodes, each link is identified with a triple: (node1id, node2id, counter)

```
list(G.edges)[:10]
```

```
[(60370565, 338870652, 0),
 (60370565, 338870615, 0),
```

```
 (60370586, 338870618, 0),
 (60370599, 338870618, 0),
 (60370599, 60370602, 0),
 (60370602, 60370612, 0),
 (60370602, 60370599, 0),
 (60370612, 721266231, 0),
 (60370612, 60370602, 0),
 (60370622, 2358639572, 0)]
```

These values above correspond to `u`, `v` and `key`. More on that later, but, essentially, `u` is the id of the "from-node" and `v` the id of the "to-node"

7.2.2.6 Method #6 Passing a Poylgon in the WGS crs

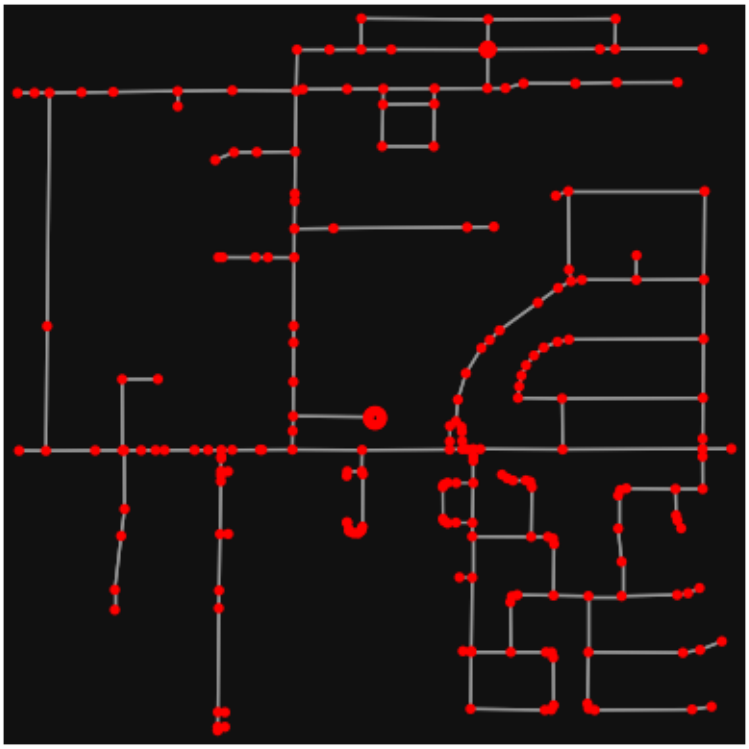
```
# get the boundary polygon for Sarajevo, and plot it
city = ox.geocode_to_gdf("Sarajevo, Bosnia")
city_proj = ox.project_gdf(city)
ax = city_proj.plot(fc="gray", ec="none")
ax.axis("off");
```

```
sarajevo = ox.geocode_to_gdf("Sarajevo, Bosnia")
polygon = sarajevo["geometry"].iloc[0]

G = ox.graph_from_polygon(polygon, network_type="drive_service")
ox.plot_graph(G, node_size=0, edge_color="w", edge_linewidth=0.3);
```

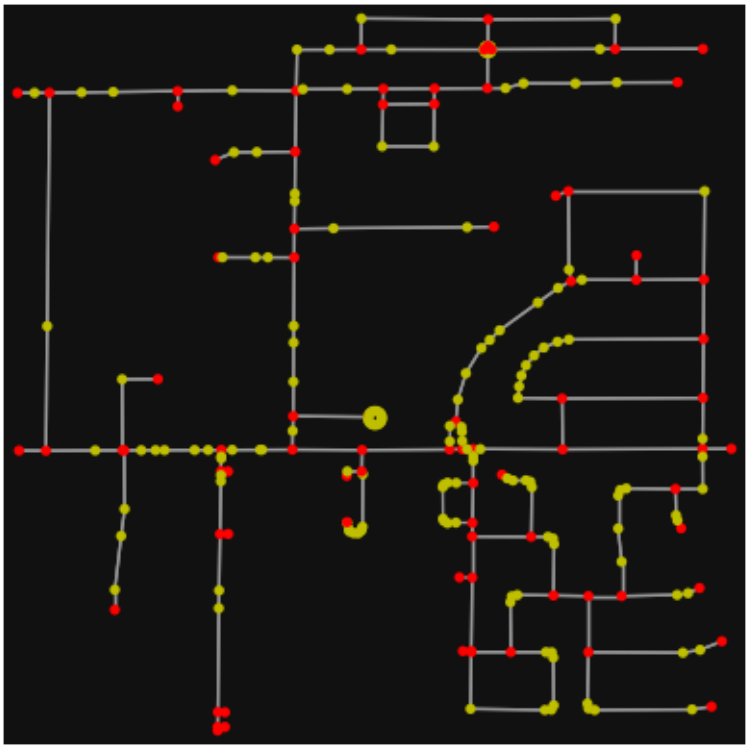
7.2.3 Simplifying and Cleaning the Street Network Topology

Simplification is normally done by `OSMnx` automatically under the hood, but we can break it out to see how it works. OpenStreetMap nodes are weird. They include intersections, but they also include all the points along a single block where the street curves. The latter are not nodes in the graph theory sense, so we remove them algorithmically and consolidate the set of edges between "true" network nodes into a single edge. There are two simplification modes, strict and non-strict. The main difference is that unlike strict mode, non-strict mode allows simplification to an "expansion graph" (i.e., if the graph were undirected, nodes with degree 2 as long as the incident edges have different OSM IDs).

```
# create a network around some (lat, lng) point but do not simplify it yet
location_point = (33.299896, -111.831638)
G = ox.graph_from_point(location_point, network_type="drive_service", dist=500, simplify=Fals
ox.plot_graph(G, node_color="r", figsize=(5,5))
```

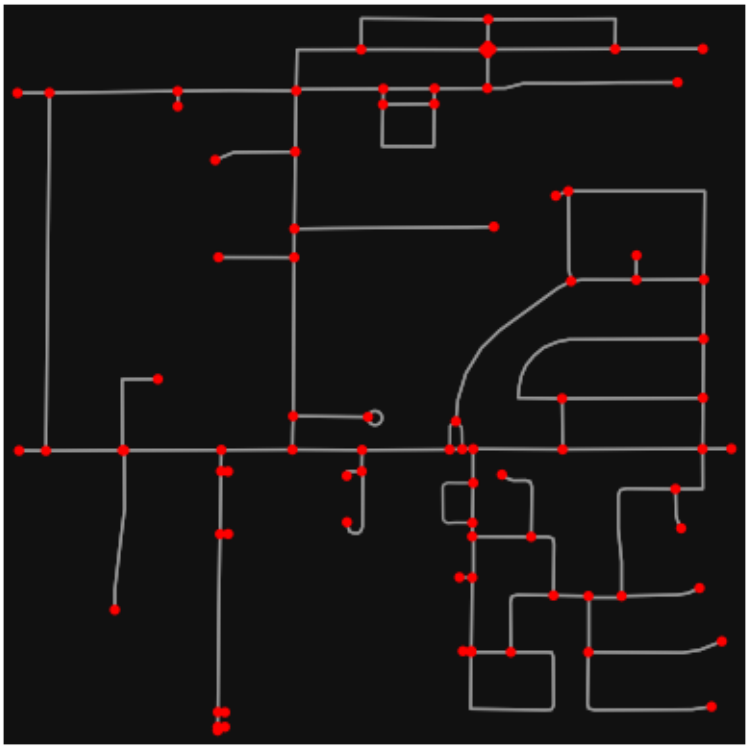


```
# turn off strict mode and see what nodes we'd remove, in yellow
colors = ["r" if ox.simplification._is_endpoint(G, node) else "y" for node in G.nodes()]
fig, ax = ox.plot_graph(G, node_color=colors, figsize=(5,5))
```



The yellow markers above are OSM nodes. We'll remove the nodes in yellow as they're not real network nodes (intersections/dead-ends).

```
# simplify the network
G = ox.simplify_graph(G)
ox.plot_graph(G, node_color="r", figsize=(5,5))
```



7.2.3.1 Optional: Complex intersection consolidation

Many real-world street networks feature complex intersections and traffic circles, resulting in a cluster of graph nodes where there is really just one true intersection. Similarly, divided roads are often represented by separate centerline edges: the intersection of two divided roads thus creates 4 nodes, representing where each edge intersects a perpendicular edge, but these 4 nodes represent a single intersection in the real world. Traffic circles similarly create a cluster of nodes where each street's edge intersects the roundabout.

`OSMnx` can consolidate nearby intersections and optionally rebuild the graph's topology.

```
# get a street network and plot it with all edge intersections
point = 55.667708, 12.596266
G = ox.graph_from_point(point, network_type="drive", dist=500)
ox.plot_graph(G, node_color="r", figsize=(5,5))
```

Notice the complex intersections creating clusters of nodes.

We'll specify that any nodes with 15 meter buffers of each other in this network are part of the same intersection. Adjust this tolerance based on the street design standards in the community you are examining, and use a projected graph to work in meaningful units like meters. We'll also specify that we do not want dead-ends returned in our list of consolidated intersections.

```
# get a GeoSeries of consolidated intersections
G_proj = ox.project_graph(G)
intersections = ox.consolidate_intersections(G_proj, rebuild_graph=False, tolerance=15, dead
# compare to number of nodes in original graph
print(len(intersections), "vs", len(G))
```

```
73 vs 122
```

Note that these cleaned up intersections give us more accurate intersection counts and densities, but do not alter or integrate with the network's topology. To do that, we need to **rebuild the graph**.

```
# consolidate intersections and rebuild graph topology
# this reconnects edge geometries to the new consolidated nodes
cleaned = ox.consolidate_intersections(G_proj, rebuild_graph=True, tolerance=15, dead_ends=Fa
ox.plot_graph(cleaned, node_color="r", figsize=(5,5))
```

Notice how many nodes are merged into a new single centroid node, with edge geometries extended to connect to it. Similar consolidation occurs at the intersection of the divided roads.

Note

Running `consolidate_intersections` with `rebuild_graph=True` may yield somewhat (but not very) different intersection counts/densities compared to `rebuild_graph=False`. The difference lies in that the latter just merges buffered node points that overlap, whereas the former checks the topology of the overlapping node buffers before merging them. This prevents topologically remote but spatially proximate nodes from being merged. For example: - A street intersection may lie directly below a freeway overpass's intersection with an on-ramp. We would not want to merge these together and connect their edges: they are distinct junctions in the system of roads. - In a residential neighbourhood, a bollarded street may create a dead-end immediately next to an intersection or traffic circle. We would not want to merge this dead-end with the intersection and connect their edges.

These examples illustrate (two-dimensional) geometric proximity, but topological remoteness. Accordingly, in some situations we may expect higher intersection counts when using `rebuild_graph=True` because it is more cautious with merging in these cases. The trade-off is that it has higher time complexity than `rebuild_graph=False`.

7.2.4 From OSMNX to NetworkX and Geopandas Classes

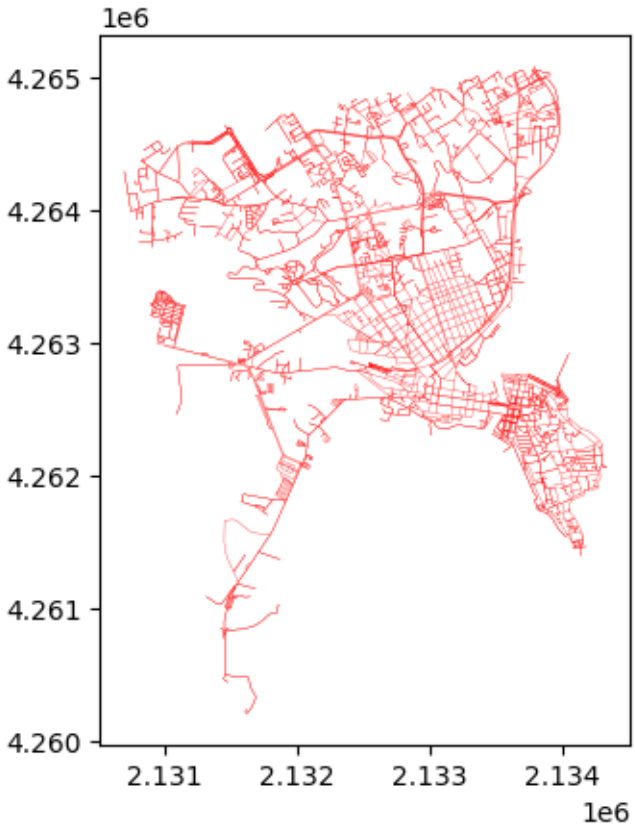
Now, we have seen how to use some basic `osmnx` functions to obtain a graph representation of the street network. While this is simple and straightforward, I advise working on the street network represented as `geopandas.GeoDataFrame`s and `networkx` graphs. While at the beginning this may be more tedious, it also gives us much more control, both in terms of visualisation and analysis depth. Before proceeding, it is important to keep in mind what type of graphs can be modelled through OSMNx and NetworkX, and what that means for the infrastructure we are modelling.

Have a look at NetworkX documentation for details.

By default, the graphs obtained through OSMNX are 'MultiDiGraph'

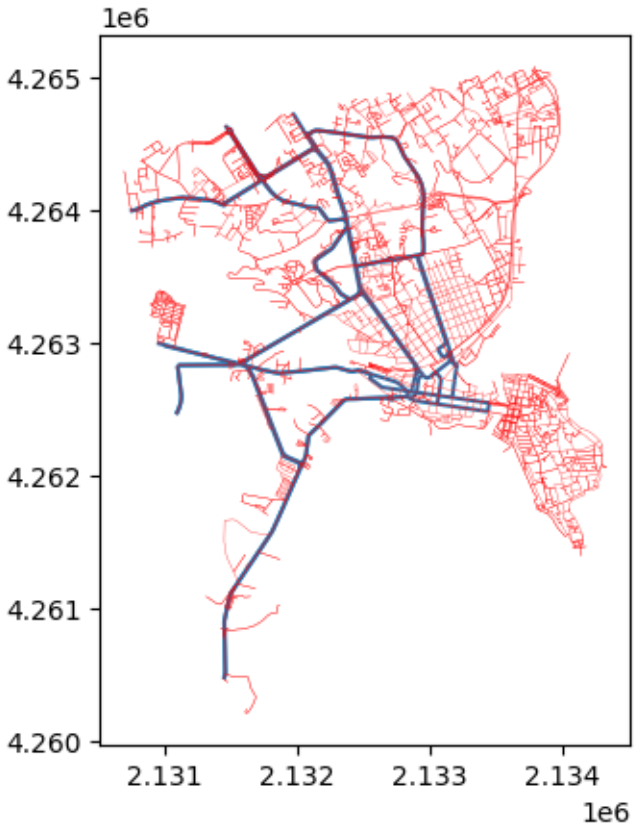
```
siracusa_graph = ox.graph_from_address("Largo XXV Luglio, Siracusa", network_type="all", dist
sicily_epsg = 'EPSG:23030'
nodes = ox.graph_to_gdfs(siracusa_graph, nodes=True, node_geometry=True, edges = False).to_cr
edges = ox.graph_to_gdfs(siracusa_graph, nodes=False, edges =  True, fill_edge_geometry=True)
```

```
edges.plot(lw = 0.2, color = 'red')
```



We can visualise major roads on top of the street network:

```
ax = edges[edges.highway.isin(['primary', 'secondary'])].plot()
edges.plot(ax = ax, lw = 0.2, color = 'red')
```




```
edges.head()
```

u	v	k
33566408	297074052	0
	9130394527	0
	7876226633	0
33566411	8185067296	0
	33566412	0

```
nodes.head()
```

osmid	y	x	street_count	highway	geometry
33566408	37.068451	15.280819	3	NaN	POINT (2132529.446 4262765.911)
33566411	37.065793	15.287216	4	NaN	POINT (2133165.090 4262581.928)
33566412	37.065258	15.286985	3	NaN	POINT (2133156.199 4262517.746)
33566413	37.064692	15.286909	3	NaN	POINT (2133162.078 4262452.755)
33566854	37.060586	15.293191	3	NaN	POINT (2133819.897 4262104.005)

As you can see an edge's index is a `MultiIndex` given by `u`, `v`, and `key`. `u` and `v` in `networkx` language stand for "from-node" and "to-node". `key`, on the other hand, indicates whether more than one edge links `u` and `v`. When a pair of `u` and `v` nodes have more then one edge, every other edge will have an incremental `key` value (the second one 1, the third one 2, etc., very rare that there are more than two edges per pair of nodes). This means that we are dealing with a `MultiGraph` representation of the street network. In other words, we have two street segments connecting the same two nodes in different directions. This could, for example, indicate that the directionality is accounted for (one-way segments). Based on our objective and analysis, such an aspect might be relevant, or not.

7.2.4.1 (Re)converting to a `MultiGraph/MultiDiGraph`

First, we get rid of the `pandas` `MultiIndex` and we verify whether we are working with bidirectional edges.

```
edges.reset_index(inplace = True)
edges['key'].sum()
```

```
96
```

```
import networkx as nx
Mg = nx.MultiGraph()
Mg.add_nodes_from(nodes.index)
```

Copying the nodes' attributes (this step is not particularly essential unless we need some information associated with the nodes that we want to use when executing `networkx` methods). Anything else we can rely on the information that is in the `GeoDataFrame`.

```
# add the nodes' attributes
attributes = nodes.to_dict()

for attribute_name in nodes.columns:
    if nodes[attribute_name].apply(lambda x: type(x) == list).any():
        continue
    # only add this attribute to nodes which have a non-null value for it
    attribute_values = {k:v for k, v in attributes[attribute_name].items() if pd.notnull(v)}
    nx.set_node_attributes(Mg, name=attribute_name, values=attribute_values)
```

We do something similar for edges. In this case, it is more likely that there is information that we want to copy, e.g. the length of the edges, into the `MultiGraph`.

```
edges['length'] = edges.geometry.length

# add the edges and attributes that are not u, v, key, null, or of type list
## u, v, and key are added directly as you can see from the last line
for row in edges.itertuples():
    attrs = {label: value for label, value in row._asdict().items() if (label not in ['u', 'v
             (isinstance(value, list) or pd.notnull(value))}
    Mg.add_edge(row.u, row.v, key=row.key, **attrs)
```

7.2.4.2 Converting to a `Graph/DiGraph`

In this case, we need to get rid of the "duplicated" edges. For the sake of simplicity, we remove al the edges with `key` equal to 1. This is a very rough approach.

```
edges_copy = edges[edges.key == 0].copy()
G = nx.Graph()
G.add_nodes_from(nodes.index)
```

Then, we copy the attributes with the same approach used before, although, this time, without adding the `key` value to the edges. The function `add_edge` of the class the `graph` does not take the `key` argument as there could be only one edge between two nodes.

```
# ignore fields containing values of type list
attributes = nodes.to_dict()

for attribute_name in nodes.columns:
    if nodes[attribute_name].apply(lambda x: type(x) == list).any():
```

```
        continue
    # only add this attribute to nodes which have a non-null value for it
    attribute_values = {k: v for k, v in attributes[attribute_name].items() if pd.notnull(v)}
    nx.set_node_attributes(G, name=attribute_name, values=attribute_values)

# add the edges and attributes that are not u, v, null, or of type list
## u, and v are added directly as you can see from the last line
for row in edges_copy.itertuples():
    attrs = {label: value for label, value in row._asdict().items() if (label not in ['u', 'v
    G.add_edge(row.u, row.v, **attrs)
```

And just to be sure:

```
len(list(G.nodes())) == len(nodes)
```

```
True
```

7.3 Routing with NetworkX

Both with `osmnx` and `networkx` we can compute paths across the network, of different kinds, with different weights and methods (see here. Shortest paths, while they seem to allude to an explicit reference to distance or time, may be computed in relation to other weights or costs, based on the purposes of the user.

```
import random
# Get a list of all nodes in the graph
nodes = list(G.nodes())

# Select a random node
origin_node = random.choice(nodes)

while(True):
    destination_node = random.choice(nodes)
    if destination_node != origin_node:
        break

path_nodes = nx.shortest_path(G, source=origin_node, target=destination_node, weight='length
```

`shortest_path`, the simplest function for computing shortest paths in `networkx`, returns the list of nodes traversed by the path, including the origin and the destination nodes. We have to transform the sequence of nodes into a sequence of edges to get the corresponding route.

```
path_edges = [(path_nodes[i], path_nodes[i + 1]) for i in range(len(path_nodes) - 1)]
path_edges[0]
```

```
(8127132944, 607017640)
```

When we look at the elements of the edges sequence, we can see that this is a tuple containing the index of the `u` and `v` nodes that the edge links together. This is the `networkx` representations of an edge inside a `graph`.

```
path_edges[0]
```

```
(8127132944, 607017640)
```

However, for that edge, we want to get the attribute that refers to the index in the `edges_copy` `GeoDataFrame`. For that, We use a list comprehension.

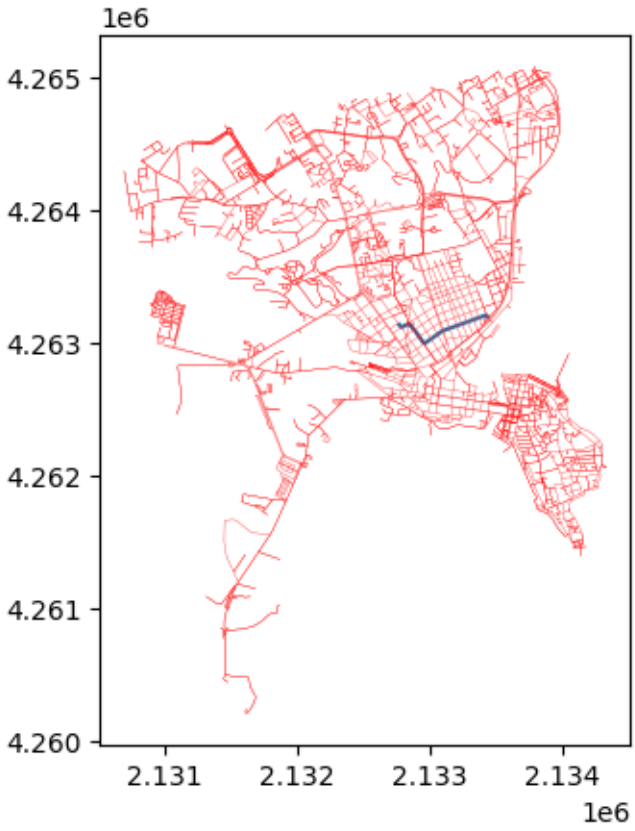
```
G[33566408][297074052]
```

```
{'Index': 429,
 'key': 0,
 'osmid': 828152472,
 'highway': 'secondary',
 'maxspeed': '50',
 'oneway': False,
 'reversed': True,
 'length': 65.9190093168531,
 'geometry': <LINESTRING (2132466.835 4262786.532, 2132529.446 4262765.911)>}
```

```
path_edges = [G.edges[edge]['Index'] for edge in path_edges]
```

And then we can plot the route.

```
ax = edges_copy[edges_copy.index.isin(path_edges)].plot()
edges_copy.plot(ax = ax, lw = 0.2, color = 'red')
```



Exercise:

Compute different shortest paths between a node (1) of your choice and a set of 20 other random nodes, for a driver and a walker. Visualise and compare the results. In order to differentiate the users, you are expected to consider their travelling speeds, in relation to the speed limit of the edges. Consider that, at least for the driver, you may want to do that in a `MultiDiGraph`.

You can use as case-study one of the graphs used above for Siracusa (Italy) or Sarajevo (Bosnia). Otherwise, just pick a city you like.

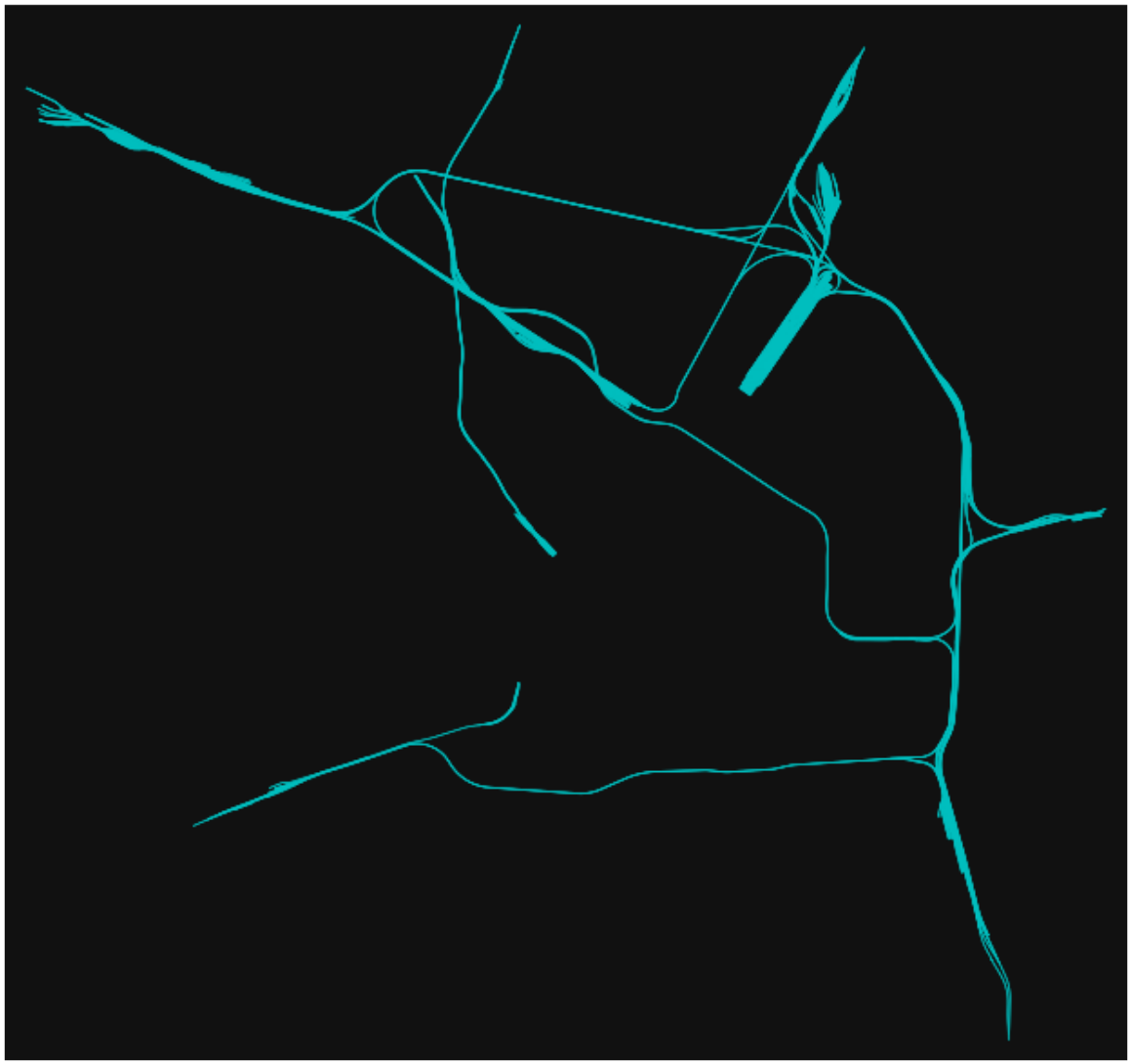
7.4 Fetching other networks with OSMNX

When downloading graphs, one can also pass a `custom_filter` to specify what OSM ways/routes/links they want represented in their graph. This would override the fact that, by default, graph functions in `osmnx` fetch street networks.

From `osmnx` documentation: `custom_filter` (string) – a custom ways filter to be used instead of the network_type presets e.g., `["power"~"line"]` or `["highway"~"motorway|trunk"]`.

This method does not work with bus services, since they are not stored in OpenStreetMap Data as Links but rather relations. See here for more info.

```
# railway infrastructures
place = "Milan, Italy"
G = ox.graph_from_place(place, custom_filter='["railway"~"rail"]')
ox.plot_graph(G, edge_color="c", edge_linewidth=0.5, node_size=0);
```



```
# network of the canals of amsterdam
place = "Amsterdam, Netherlands"
G = ox.graph_from_place(place, custom_filter='["waterway"~"canal"]')
ox.plot_graph(G, edge_color="c", edge_linewidth=1, node_size=0);
```

7.5 Retrieving Other OSM data

OSMNx power also lies on the fact that it allows obtaining any other element represented in OpenStreetMap data. You can use the same methods described above for graphs. Instead of using `graph_from_place`, you would use, for example, `features_from_place`, `features_from_address`, etc.

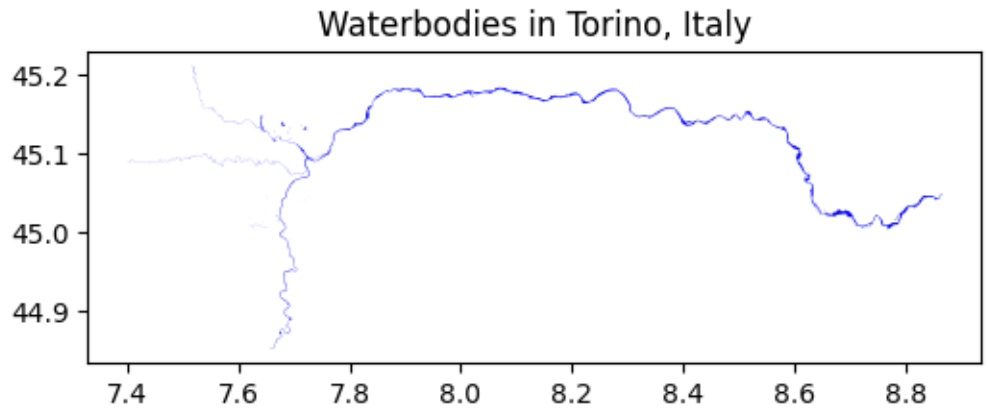
7.5.0.1 Fetching Building Footprints

```
# Specify the location and the data type
place = 'Torino, Italy'
tags = {'building': True}

# Fetch building footprints
buildings = ox.features_from_place(place, tags=tags)
buildings = buildings[buildings.geometry.geom_type == 'Polygon']

# Plot the building footprints
ax = buildings.plot(figsize=(6, 6), color = 'orange')
ax.set_title('Building Footprints in Torino, Italy')
```

7.5.0.2 Water bodies

```
# Tags for waterbodies
water_tags = {'natural': ['water']}

# Fetch water bodies
waterbodies = ox.features_from_place(place, tags=water_tags)

# Plot water bodies
ax = waterbodies.plot(figsize=(6, 6), color='blue')
ax.set_title('Waterbodies in Torino, Italy')
```

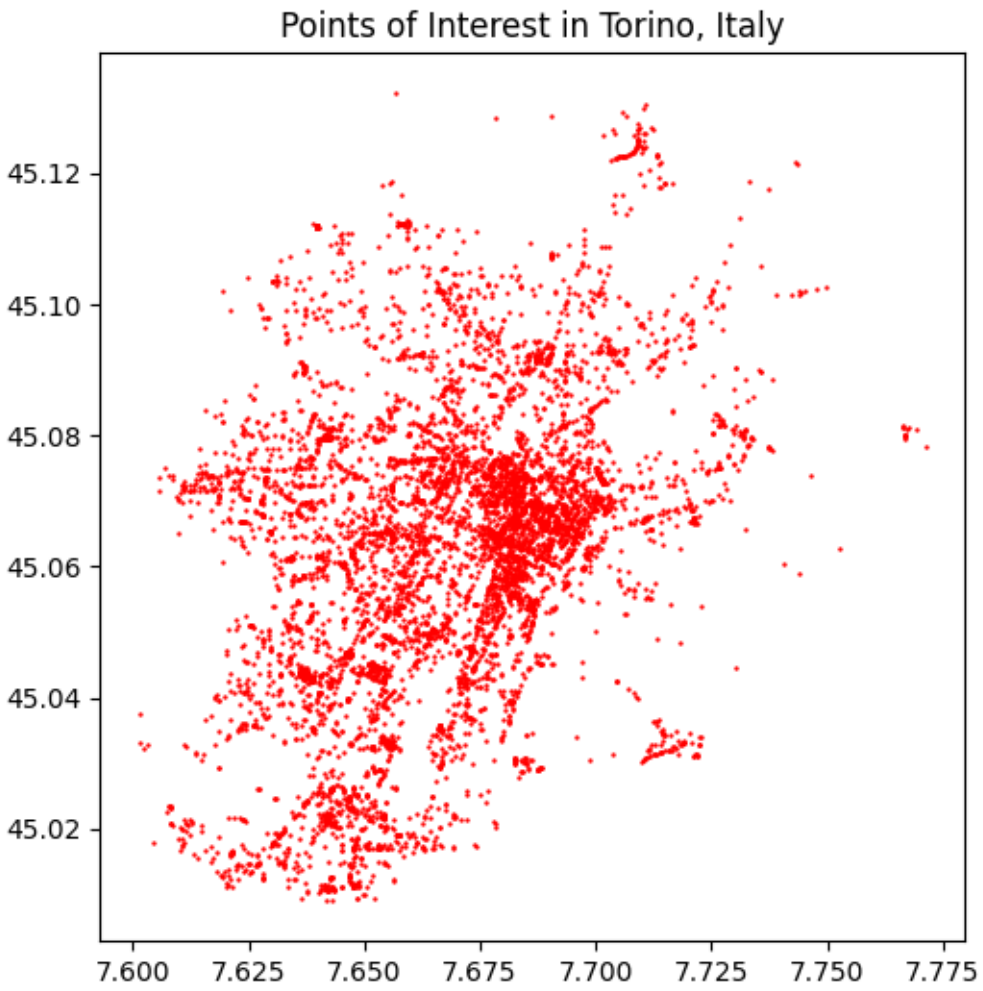
```
Text(0.5, 1.0, 'Waterbodies in Torino, Italy')
```



7.5.0.3 POIs

```
# Tags for POIs
poi_tags = {'amenity': True}

# Fetch POIs
pois = ox.features_from_place(place, tags=poi_tags)
pois = pois[pois.geometry.geom_type == 'Point']
# Plot POIs
ax = pois.plot(figsize=(6, 6), color='red', markersize = 0.5)
ax.set_title('Points of Interest in Torino, Italy')
```

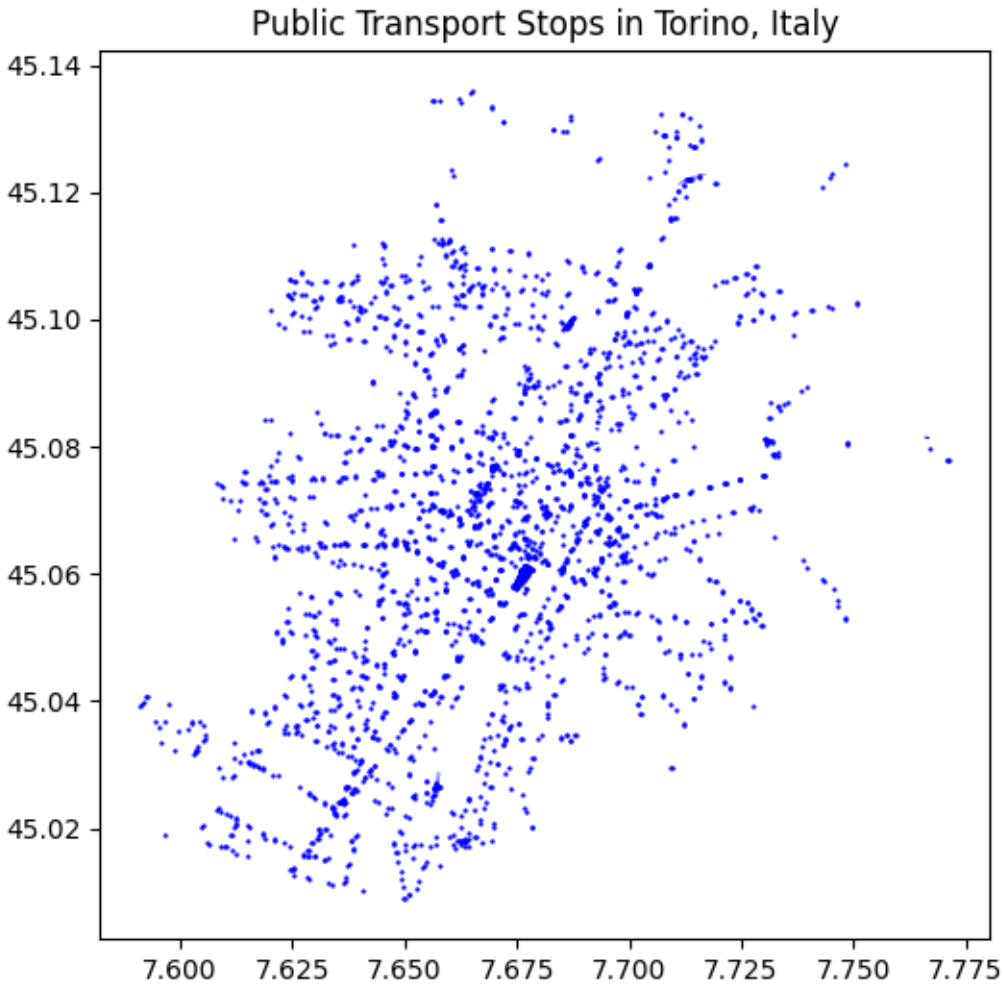
```
Text(0.5, 1.0, 'Points of Interest in Torino, Italy')
```



7.5.0.4 Public Transport Bus Stops

```
# Tags for public transport
tags = {'public_transport': True}

# Fetch public transport routes
pt_stops = ox.features_from_place(place, tags=tags)

# Plot bus routes
ax = pt_stops.plot(figsize=(6, 6), color='blue', markersize = 0.5)
ax.set_title('Public Transport Stops in Torino, Italy')
```

```
Text(0.5, 1.0, 'Public Transport Stops in Torino, Italy')
```



8 Dashboards

The **Lecture slides** can be found here.

This **lab**'s notebook can be downloaded from here.

```
## Run only if necessary
# !pip install panel
# !pip install hvpolt
```

```
# Standard library imports
import datetime as dt

# Third-party imports
import geopandas as gpd
import hvplot.pandas
import numpy as np
import pandas as pd
import panel as pn
import plotly.express as px

# Initialize Panel with extensions
pn.extension('plotly', design='material')
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): application/vnd.holoviews_exec.v0+json, text/html
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_
```

```
Unable to display output for mime type(s): text/html
```

Importing the Data

```
df = pd.read_csv('../data/GTD_2022.csv', low_memory=False)
```

```
## visualise main columns
df[['gname', 'year', 'date', 'country_txt', 'nkill', 'nwound','weaptype1_txt']].head()
```

	gname	year	date	country_txt	nkill	nwound	we
0	Unknown	2005	28/05/2005	Pakistan	1.0	0.0	Fi
1	Ansar al-Sunna	2005	29/05/2005	Iraq	1.0	0.0	Fi
2	Al-Qaida in Iraq	2007	08/06/2007	Iraq	15.0	0.0	Fi
3	Taliban	2010	15/06/2010	Afghanistan	1.0	0.0	Fi
4	Communist Party of India - Maoist (CPI-Maoist)	2011	06/01/2011	India	1.0	0.0	Fi

8.1 Creating a Basic Dashboard: Map + Date Slider

We are going to use `Panel` a Python library for creating interactive and dynamic web-based dashboards and applications. It allows data scientists, analysts, and so forth, to turn data sets into interactive dashboards using a wide array of widgets, plots, and layouts without requiring deep web development skills. Panel supports various plotting libraries like Matplotlib, Bokeh, and Plotly, as well as our friend Folium, and allows working with Pandas, NumPy, and others. It's flexible enough to serve either as a standalone app or embedded in existing web applications, and it can be deployed easily. Have a look at this article for some insights on how to exploit `Panel` for nice data visualisations.

```
df['date']= pd.to_datetime(df['date'], dayfirst=True)
date_slider = pn.widgets.DateSlider(
    name='Date',
    start=df['date'].min(),
    end=df['date'].max(),
    value=df['date'].min()  # Single date value
)

df['date']= pd.to_datetime(df['date'], dayfirst=True).dt.date # this is not very handy but o
```

And here's the slider

```
date_slider
```

```
DateSlider(design=<class 'panel.theme.materi..., end=Timestamp('2020-12-31 0..., name='Date'
```

Now, we create a function to update the map, on the basis of the slider.

```
def update_map_date(selected_date):
    # Filter the DataFrame based on the selected date
    filtered_df = df[df['date'] == selected_date].copy()
    # Create the scatter geo plot
    fig = px.scatter_geo(filtered_df, lat='latitude', lon='longitude')
    return fig
```

```
# Bind the update_map function to the date_slider, vertical layout
interactive_panel = pn.Column(date_slider, pn.bind(update_map_date, date_slider))

# Serve the Panel dashboard
interactive_panel.servable()
```

```
Column(design=<class 'panel.theme.materi...)
    [0] DateSlider(design=<class 'panel.theme.materi..., end=Timestamp('2020-12-31 0..., name
    [1] ParamFunction(function, _pane=Plotly, defer_load=False, design=<class 'panel.theme.ma
```

`dashboard.servable()` makes the dashboard "servable," meaning it can be run as a standalone app or served via a notebook, depending on how you're using `Panel`. When you call `.servable()` on a Panel object, you're essentially telling Panel that this is the object you want to display when the dashboard is run. If you're working in a Jupyter notebook, this allows the dashboard to be rendered inline.

8.1.1 The `bind` function

In `Panel`, `pn.bind` is a function that creates a dynamic link between widget values and a function, enabling interactive and reactive applications without the need for explicit callbacks or event handlers. When you use `pn.bind`, you're essentially telling Panel to watch certain parameters (like the value of a widget) and call a specified function whenever those parameters change, automatically passing the new values to the function.

Here's a breakdown of how `pn.bind` works:

- `pn.bind` links a function to one or more parameters (often widget values). This function will be called whenever the linked parameters change.

- When the linked parameters change, pn.bind automatically calls the specified function with the new values as arguments. This eliminates the need for manual event handling or value extraction within the function.
- This binding creates a reactive link, meaning the output of the function will automatically update in the UI when the input parameters change. This is fundamental for creating interactive and dynamic dashboards.

Usage with Layouts: When you use `pn.bind` within a Panel layout (like `pn.Column` or `pn.Row`), the return value of the bound function is dynamically inserted into the layout. If the function returns a plot, a table, or any other visual component, that component will update in the UI whenever the function is triggered by a change in the bound parameters. Here's a simple example to illustrate:

```
# Define a slider
slider = pn.widgets.IntSlider(name='Slider', start=0, end=10, value=5)

# Define a function that takes a parameter and returns a value based on that parameter
def multiply_by_two(x):
    return x * 2

# Bind the function to the slider's value
bound_function = pn.bind(multiply_by_two, slider.param.value)

# Create a layout that displays the slider and the output of the bound function
layout = pn.Column(slider, bound_function)
layout.servable()
```

```
Column(design=<class 'panel.theme.materi...)
    [0] IntSlider(design=<class 'panel.theme.materi..., end=10, name='Slider', value=5)
    [1] ParamFunction(function, _pane=Str, defer_load=False, design=<class 'panel.theme.mater
```

`pn.Column(...)` creates a vertical layout (column) containing the row with certain elements/widgets/panes. `pn.Column` is used when you want to stack components vertically. Contrarily, `pn.Row()`: creates a horizontal layout (row). `pn.Row` is used when you want to place components side by side horizontally. In both cases you can add one or more elements.

8.2 Some More Widgets

`Panel` offers a range of widgets for interactive user inputs. These widgets can be used to receive input from users and update the dashboards/panels accordingly. Have a look here for a list of other widgets and usage examples.

```
# Select widget for choosing a city
group_selector = pn.widgets.Select(name='Group', options=list(df.gname.unique()))

# RangeSlider for selecting a numeric range
year_slider = pn.widgets.IntSlider(name='Year', start=df.year.min(), end=df.year.max(), step=

# RangeSlider for selecting a numeric range
range_slider = pn.widgets.RangeSlider(name='Nr of People Killed', start=df.nkill.min(), end=

# CheckBox for a boolean choice
check_box = pn.widgets.Checkbox(name='Check Me')

# TextInput for freeform input
text_input = pn.widgets.TextInput(name='Enter Text')

# RadioButtons for exclusive selection
radio_button = pn.widgets.RadioButtonGroup(name='Options', options=['Option 1', 'Option 2',
```

```
# Layout these widgets in a column
widgets_column = pn.Column(group_selector, year_slider, check_box, text_input, radio_button)
widgets_column
```

```
Column(design=<class 'panel.theme.materi...)
    [0] Select(design=<class 'panel.theme.materi..., name='Group', options=['Unknown', 'Ansa
    [1] IntSlider(design=<class 'panel.theme.materi..., end=2020, name='Year', start=1970, va
    [2] Checkbox(design=<class 'panel.theme.materi..., name='Check Me')
    [3] TextInput(design=<class 'panel.theme.materi..., name='Enter Text')
    [4] RadioButtonGroup(design=<class 'panel.theme.materi..., name='Options', options=['Opti
```

8.2.1 Subsetting the Dataframe on the basis of categorical variables

```
# Create a dataset just for Iraq (feel free to change it)
iraq_df = df[df.country_txt == 'Iraq'].copy()
# Calculate the centroid of the filtered points for centering the map
center_lat = iraq_df['latitude'].mean()
center_lon = iraq_df['longitude'].mean()
```

```
# Create a Select widget for the 'group' column
group_selector = pn.widgets.Select(name='Group', options=iraq_df['gname'].unique().tolist())
```

```
# Define a function to update the map based on the selected group
@pn.depends(group_selector.param.value)
def update_map_iraq(selected_group):
    # Filter the DataFrame based on the selected group
    filtered_df = iraq_df[iraq_df['gname'] == selected_group].copy()

    # Generate the map
    fig = px.scatter_geo(filtered_df, lat='latitude', lon='longitude', title=f"Attacks carri
                         center={"lat": center_lat, "lon": center_lon})

    # Adjusting the map's view to a 'closer' zoom
    fig.update_geos(projection_type="natural earth", lataxis_range=[center_lat-10, center_la
    # Return the figure
    return fig

# Create a Panel layout to display the widget and the map
dashboard = pn.Column(group_selector, update_map_iraq)

# Display the dashboard
dashboard.servable()
```

```
Column(design=<class 'panel.theme.materi...)
    [0] Select(design=<class 'panel.theme.materi..., name='Group', options=['Ansar al-Sunna'
    [1] ParamFunction(function, _pane=Plotly, defer_load=False, design=<class 'panel.theme.ma
```

`Plotly Express` is here used to generate a geographical scatter plot.

px.scatter_geo: This creates a scatter plot on a geographic map. The arguments `lat='latitude'` and `lon='longitude'` specify the DataFrame columns that contain the latitude and longitude coordinates for the points to be plotted. The title argument sets the title of the map, and center specifies the central point of the map view, ensuring the map is centered around the points of interest. `fig.update_geos` updates the geographic layout of the figure. It's used here to adjust the map's projection and zoom level.

- `projection_type="natural earth"`: Sets the map's projection type to "natural earth," which is a visually appealing and commonly used projection for world maps.
- `lataxis_range=[center_lat-10, center_lat+10]`: Defines the range of latitude to be displayed on the map. This setting zooms in on the region by limiting the latitude range to 10 degrees above and below the center latitude.
- `lonaxis_range=[center_lon-20, center_lon+20]`: Defines the range of longitude to be displayed on the map. Similar to lataxis_range, this limits the longitude range to

20 degrees on either side of the center longitude, effectively zooming in on the area of interest.

We will be switching to `folium` in a bit, also for the sake of continuity and because it's more powerful, so don't worry to much about it.

```
columns = ['gname', 'year', 'date', 'country_txt', 'nkill', 'nwound','weaptype1_txt']

df = df.dropna(subset=['latitude', 'longitude'])
# Widget to select a country
countries = sorted(df['country_txt'].unique().tolist())
country_selector = pn.widgets.Select(name='Country', options=countries)
```

8.3 Complex Layouts

Layouts in Panel are used to organize widgets and plots in a structured manner. We are mainly working with `pn.Row()` and `pn.Column()` but familiarise yourself with other possible layouts (see here).

```
# still using the country selector here

new_country_selector = pn.widgets.Select(name='Country', options=countries)
```

We can add a scatter plot to the dashboard

```
# Plot
def update_scatter(data, width, height, title):
        return data.hvplot.scatter(
        x='year',
        y=['nkill', 'nwound'],  # Ensure these column names match your DataFrame
        title=title,
        width=width,
        height=height
    )
```

And a function that updates several panels based on the country selector. See what happens when you select another country from the list.

```
@pn.depends(new_country_selector.param.value)
def update_dashboard(country):
    data = df[df['country_txt'] == country].copy()

    # Creating a plot for number of attacks over time
    plot = update_scatter(data, 800,300,
                          title = f'Number of People Killed and Wounded Over Time in {country

    # Creating a summary table
    table = pn.widgets.DataFrame(data[columns], show_index=False, width=600)

    return pn.Column(plot, table)

# Layout the dashboard
dashboard = pn.Column(
    pn.Row(country_selector),
    update_dashboard
)

dashboard.servable()
```

```
Column(design=<class 'panel.theme.materi...)
    [0] Row(design=<class 'panel.theme.materi...)
        [0] Select(design=<class 'panel.theme.materi..., name='Country', options=['Afghanista
    [1] ParamFunction(function, _pane=Column, defer_load=False, design=<class 'panel.theme.ma
```

8.3.1 Using Folium Maps

As mentioned, `Panel` allows us to incoporate `folium` maps.

```
# let's reset variables to avoid interactions between different functions/dashboards
%reset -f -s

# remporting again
import geopandas as gpd
import hvplot.pandas
import numpy as np
import pandas as pd
import panel as pn
import plotly.express as px
```

```
# Initialize Panel with extensions
pn.extension('plotly', design='material')
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): application/vnd.holoviews_exec.v0+json, text/html
```

```
# reload the df
df = pd.read_csv('../data/GTD_2022.csv', low_memory=False)
df = df.dropna(subset=['latitude', 'longitude'])
countries = sorted(df['country_txt'].unique().tolist())
country_selector = pn.widgets.Select(name='Country', options=countries)
```

```
# Function to create a map centered on the selected country
import folium
from folium.plugins import MarkerCluster

def create_foliumMap(data):

    # Calculate the mean latitude and longitude to center the map
    center_lat = data['latitude'].mean()
    center_lon = data['longitude'].mean()

    # Create a Folium map centered on the average location
    folium_map = folium.Map(location=[center_lat, center_lon], zoom_start=6)

    # Use a MarkerCluster to add markers for each event
    marker_cluster = MarkerCluster().add_to(folium_map)

    # Add a marker for each event
    for idx, row in data.iterrows():
        folium.Marker(
            location=[row['latitude'], row['longitude']],
            popup=f"Date: {row['date']}<br>Deaths: {row['nkill']}",
        ).add_to(marker_cluster)
```

```
    # Return the Folium map object
    return folium_map
```

This function communicates with the one above for updating the attributes used to create the Folium map.

```
# Panel doesn't directly render Folium maps, so we need to render it as HTML
def update_map_country(df, country, width, height):
    # Filter the DataFrame for the selected country
    data = df[df['country_txt'] == country].copy()
    folium_map = create_foliumMap(data)
    # Panel doesn't directly render Folium maps, so we need to render it as HTML
    return pn.pane.HTML(folium_map._repr_html_(), width=width, height=height)
```

Then we bind the function to the widget and pass the DataFrame, along with the size attributes.

```
width = 700
height = 500
map_pane = pn.bind(update_map_country, df,  country_selector.param.value, width, height)

# Layout the dashboard
dashboard = pn.Column(
    pn.Row(country_selector),
    map_pane
)

dashboard.servable()
```

```
Column(design=<class 'panel.theme.materi...)
    [0] Row(design=<class 'panel.theme.materi...)
        [0] Select(design=<class 'panel.theme.materi..., name='Country', options=['Afghanista
    [1] ParamFunction(function, _pane=HTML, defer_load=False, design=<class 'panel.theme.mate
```

You may have noticed the use of the `@pn.depends` decorator. In `Panel` this is used to create reactive functions, namely functions that automatically update their output when an input parameter changes. This is a core concept in creating interactive and dynamic dashboards with `Panel`.

Exercise:

Now, try to create a layout where you put together the map, the dataframe panel and the scatter plot, using a Column or a Row Layout.

8.4 Grid Layouts

`GridSpec` in `Panel` is a layout object that allows you to arrange your visual components in a grid-like structure, specifying the position and size of each component by defining rows and columns. You can access and assign components to specific areas of the grid using slicing syntax, e.g., `grid[0, 0] = component` places a component in the first row and first column of the grid.

Now, let's try to combine the map, the scatterplot, and the dataframe panel in a unique dashboard. We are going to create a `GridSpecLayout`, see https://panel.holoviz.org/reference/layouts/GridSpec.html

```
# let's reset variables to avoid interactions between different functions/dashboards
%reset -f -s

# remporting again
import geopandas as gpd
import hvplot.pandas
import numpy as np
import pandas as pd
import panel as pn
import plotly.express as px

# Initialize Panel with extensions
pn.extension('plotly', design='material')
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_

Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_

Unable to display output for mime type(s): text/html

Unable to display output for mime type(s): application/vnd.holoviews_exec.v0+json, text/html
```

```
# reload the df
df = pd.read_csv('../data/GTD_2022.csv', low_memory=False)
df = df.dropna(subset=['latitude', 'longitude'])
countries = sorted(df['country_txt'].unique().tolist())
country_selector = pn.widgets.Select(name='Country', options=countries)
columns = ['gname', 'year', 'date', 'country_txt', 'nkill', 'nwound','weaptype1_txt']
```

Making a smaller dataset

```
df = df[df.year > 2000]# only 2000s
```

```
# Function to create a map centered on the selected country
import folium
from folium.plugins import MarkerCluster

def create_foliumMap(data):

    # Calculate the mean latitude and longitude to center the map
    center_lat = data['latitude'].mean()
    center_lon = data['longitude'].mean()

    # Create a Folium map centered on the average location
    folium_map = folium.Map(location=[center_lat, center_lon], zoom_start=6)

    # Use a MarkerCluster to add markers for each event
    marker_cluster = MarkerCluster().add_to(folium_map)

    # Add a marker for each event
    for idx, row in data.iterrows():
        folium.Marker(
            location=[row['latitude'], row['longitude']],
            popup=f"Date: {row['date']}<br>Deaths: {row['nkill']}",
        ).add_to(marker_cluster)

    # Return the Folium map object
    return folium_map

# Panel doesn't directly render Folium maps, so we need to render it as HTML
def update_map_country(df, country):
    # Filter the DataFrame for the selected country
    data = df[df['country_txt'] == country].copy()
    folium_map = create_foliumMap(data)
    # Panel doesn't directly render Folium maps, so we need to render it as HTML
    return pn.pane.HTML(folium_map._repr_html_())

# Plot
def update_scatter(data, title):
        return data.hvplot.scatter(
        x='year',
        y=['nkill', 'nwound'],  # Ensure these column names match your DataFrame
        title=title,
```

```
    )
```

```
countries = sorted(df['country_txt'].unique().tolist())
country_selector = pn.widgets.Select(name='Country', options=countries)

@pn.depends(country_selector.param.value)
def update_components(country):
    # Update the map
    map_pane = update_map_country(df, country)

    # Filter the data for the DataFrame pane
    data = df[df['country_txt'] == country]

    # DataFrame pane
    df_pane = pn.widgets.DataFrame(data[columns])

    # Scatter plot
    plot = update_scatter(data, title=f'Number of People Killed and Wounded Over Time in {cou

    # Use GridSpec for layouts
    grid = pn.GridSpec(width=1400, height=750)
    grid[0, 0:3] = country_selector
    grid[1:6, :3] = map_pane
    grid[1:3, 3:6] = plot
    grid[3:5, 3:6] = df_pane

    return grid

# Layout the dashboard
dashboard = pn.Column(update_components)
dashboard.servable()
```

```
Column(design=<class 'panel.theme.materi...)
    [0] ParamFunction(function, _pane=GridSpec, defer_load=False, design=<class 'panel.theme
```

Think about including other widgets to improve the visualisation. For some countries it's impossible to look at the scatter plot at first, because the amount of records associated with it.

Also consider using `Spacer`s to organise your layout, see: https://panel.holoviz.org/reference/layouts/GridSpec.h

8.5 Alternative Widgets

Let's play with another widget that we briefly mentioned above, the `RadioButtonGroup` widget.

```
# let's reset variables to avoid interactions between different functions/dashboards
%reset -f -s

# remporting again
import geopandas as gpd
import hvplot.pandas
import numpy as np
import pandas as pd
import panel as pn
import plotly.express as px
import folium

# Initialize Panel with extensions
pn.extension('plotly', design='material')
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_

Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_

Unable to display output for mime type(s): text/html

Unable to display output for mime type(s): application/vnd.holoviews_exec.v0+json, text/html
```

```
df = pd.read_csv('../data/GTD_2022.csv', low_memory=False)
df = df.dropna(subset=['latitude', 'longitude'])
countries = sorted(df['country_txt'].unique().tolist())
country_selector = pn.widgets.Select(name='Country', options=countries)
```

```
# Create a dataset just for Iraq (feel free to change it)
iraq_df = df[df.country_txt == 'Iraq'].copy()
# Calculate the centroid of the filtered points for centering the map
center_lat = iraq_df['latitude'].mean()
center_lon = iraq_df['longitude'].mean()

# Function to create a Folium map based on the option selected (nkills or nwound)
```

```
def create_foliumMap(option):
    m = folium.Map(location=[center_lat, center_lon], zoom_start=6)  # Centered on Iraq

    for idx, row in iraq_df.iterrows():
        if option == 'Killed':
            folium.Circle(
                location=[row['latitude'], row['longitude']],
                radius=row['nkill'] * 100,  # Adjust size factor as needed
                color='red',
                fill=True,
                fill_color='red'
            ).add_to(m)
        elif option == 'Wound':
            folium.Circle(
                location=[row['latitude'], row['longitude']],
                radius=row['nwound'] * 100,  # Adjust size factor as needed
                color='blue',
                fill=True,
                fill_color='blue'
            ).add_to(m)

    return m

# Panel widgets for options
option_selector = pn.widgets.RadioButtonGroup(
    name='Select Option',
    options=['Killed', 'Wound'],
    button_type='success'
)
```

```
# Function to update the map based on the selected option
@pn.depends(option_selector.param.value)
def update_map(option):
    folium_map = create_foliumMap(option)
    return pn.pane.HTML(folium_map._repr_html_(), width=700, height=500)

# Dashboard layout
dashboard = pn.Column(
    option_selector,
    update_map
)
```

```
dashboard.servable()
```

```
Column(design=<class 'panel.theme.materi...)
    [0] RadioButtonGroup(button_type='success', design=<class 'panel.theme.materi..., name='
    [1] ParamFunction(function, _pane=HTML, defer_load=False, design=<class 'panel.theme.mat
```

This map is a bit slow in terms of reaction. Think about possible improvements or how you could tweak it (e.g. recreating the map everytime the button is pressed, etc.).

Also, consider using `Panel` in combination with `Bokeh` or `Geoviews`. For nice and smoother (although more pre-designed) for geospatial data visualizations, you can use Panel to host `Geoviews` plots, enabling the assembly of various visualization components into a cohesive dashboard. You can use Geoviews plotting tools to render maps with different geometries and overlay them on tile sources. See for example here, or here

GeoViews is built on top of Bokeh and HoloViews. It simplifies the process of creating maps, allowing you to work directly with **GeoDataFrames** and use a variety of geometries (points, lines, polygons) for your visualizations. Check: https://geoviews.orgs.

Exercise:

Pick the dataset you used for Assignment I and spend around 30 minutes trying to create a dashboard that is effective at communicating some aspects your consider relevant of the dataset. As discussed in the lecture, start by thinking exactly what you want to communicate, and build from there.

Think about how to improve the dashboards above, include a widget whose use was not demonstrated above.

Presentation

You will then have 30 seconds to present your dashboard and hit the following points:

- What the dashboard shows
- What interactivity/analysis element(s) you have used
- One thing you think is really effective about it

Remember, 30 seconds. Short and sweet. Make them count

9 GPS Traces and Animations

The **Lecture slides** can be found here.

This **lab**'s notebook can be downloaded from here.

```
## check if those are installed
pip install geoviews
```

```
import pandas as pd, geopandas as gpd
import movingpandas as mpd

import os
from shapely.geometry import Point, LineString, Polygon
from matplotlib import pyplot as plt
import movingpandas as mpd

import folium
import requests
from io import BytesIO

import warnings
warnings.filterwarnings('ignore')
```

9.1 Getting GPS trajectories from OpenStreetMap Data

The OSM API provides access to GPS traces contributed by the OpenStreetMap community. Users can query and retrieve traces based on parameters like geographical bounding box, time range, and user ID. The API supports output formats such as GPX for easy integration into applications or analysis. You can have a look at the last GPS traces uploaded here. Each trace entry likely includes:

- Trace ID (unique identifier for the trace)
- Trace Name (optional name provided by the user)
- Uploader Username
- Upload Date

- Trace Summary (e.g., distance, duration)

It's not possible to download traces from OSM using `OSMnx`. One has to script their own code

Parts of this section of the notebook have been readapted from this notebook created by Anita Graser.

9.1.1 Downaloding GPS traces from the OpenStreetMap

We can try to get OSM traces around the Uni of Liverpool Campus. Feel free to change the area.

```
import osmnx as ox
# Define the place name
place_name = "University of Liverpool, UK"

# Get the latitude and longitude
latitude, longitude = ox.geocoder.geocode(place_name)

# Create the bbox
bbox = ox.utils_geo.bbox_from_point((latitude, longitude), dist = 1500)
bbox
```

```
(53.420732655032396,
 53.39375304496761,
 -2.9432044446907044,
 -2.988462877347038)
```

We need the `bbox_to_osm_format` function below to convert our bounding box coordinates into the format required by the OpenStreetMap (OSM) API. The input tuple contains coordinates in the order (north, south, east, west). The function rearranges these values into the string "west,south,east,north", which is the format expected by OSM for API queries.

```
def bbox_to_osm_format(bbox):
    """
    Convert bounding box coordinates to OSM API format.

    Parameters
    ----------
    bbox: tuple
        A tuple containing the bounding box coordinates in the order (north, south, east, wes
```

```
    Returns
    -------
    bbox_str: str
        A string representing the bounding box in the format "west,south,east,north".
    """
    north, south, east, west = bbox
    bbox = f"{west},{south},{east},{north}"
    return bbox
```

```
bbox = bbox_to_osm_format(bbox) # needs to be {west},{south},{east},{north} for OSM Api
bbox
```

```
'-2.988462877347038,53.39375304496761,-2.9432044446907044,53.420732655032396'
```

The `get_osm_traces` function below retrieves GPS traces from OpenStreetMap within a specified bounding box, processing up to a user-defined maximum number of pages. It uses a while loop to fetch and parse GPS data into GeoDataFrames, querying the OSM API by incrementally updating the page number until no more data is available or the maximum page limit is reached.

Upon fetching the data, the function concatenates these individual GeoDataFrames into a single comprehensive GeoDataFrame. Before returning the final GeoDataFrame, it cleans the dataset by dropping a predefined list of potentially irrelevant or empty columns.

```
def get_osm_traces(max_pages = 2,  bbox='16.18, 48.09, 16.61, 48.32'):
    """
    Retrieve OpenStreetMap GPS traces within a specified bounding box.

    Parameters
    ----------
    max_pages: int, optional
        The maximum number of pages to retrieve. Defaults to 2.
    bbox: str, optional
        The bounding box coordinates in the format 'west, south, east, north'. Defaults to '

    Returns
    -------
    final_gdf: GeoDataFrame
        A GeoDataFrame containing the retrieved GPS traces.
    """
```

```
    all_data = []
    page = 0

    while (True) and (page <= max_pages):
        # Constructing the URL to query OpenStreetMap API for GPS trackpoints within a speci
        url = f'https://api.openstreetmap.org/api/0.6/trackpoints?bbox={bbox}&page={page}'

        # Sending a GET request to the constructed URL
        response = requests.get(url)

        # Checking if the response status code is not 200 (indicating success) or if the resp
        # If either condition is met, the loop breaks, indicating no more data to retrieve
        if response.status_code != 200 or not response.content:
            break

        # Reading the content of the response, which contains GPS trackpoints data, into a Ge
        # The 'layer' parameter specifies the layer within the GeoDataFrame where trackpoints
        gdf = gpd.read_file(BytesIO(response.content), layer='track_points')

        if gdf.empty:
            break

        all_data.append(gdf)
        page += 1

    # Concatenate all GeoDataFrames into one
    final_gdf = gpd.GeoDataFrame(pd.concat(all_data, ignore_index=True))

    # dropping empty columns
    columns_to_drop = ['ele', 'course', 'speed', 'magvar', 'geoidheight', 'name', 'cmt', 'des
                       'src', 'url', 'urlname', 'sym', 'type', 'fix', 'sat', 'hdop', 'vdop',
                       'pdop', 'ageofdgpsdata', 'dgpsid']

    final_gdf = final_gdf.drop(columns=[col for col in columns_to_drop if col in final_gdf.co
    return final_gdf
```

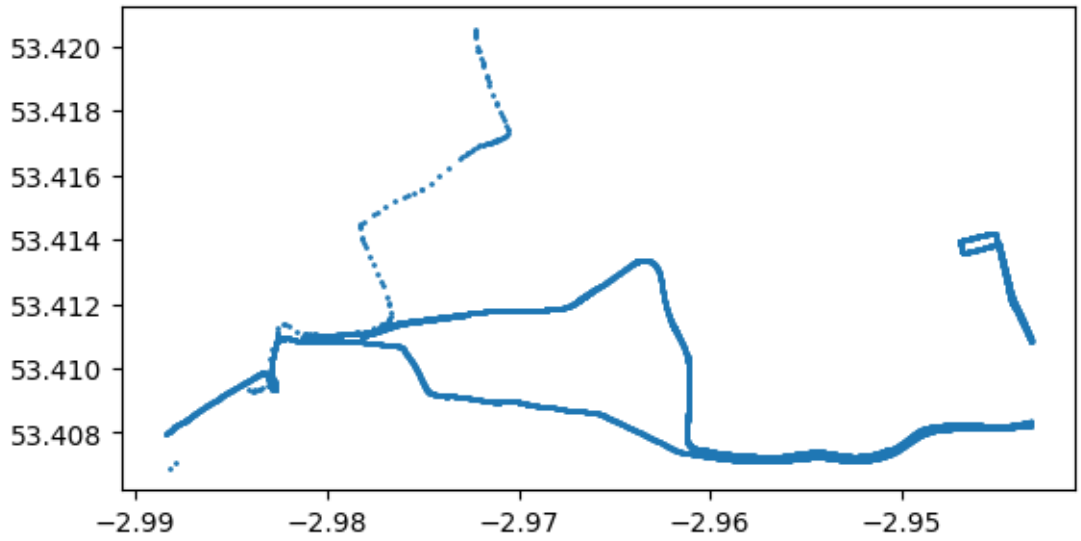
We initially download 2 pages of GSP traces. We can try with higher numbers to get more data. More recent traces are downloaded first.

```
max_pages = 2 # pages of data,
gps_track_points = get_osm_traces(max_pages, bbox)
```

```
gps_track_points.head()
```

	track_fid	track_seg_id	track_seg_point_id	time	geometry
0	0	0	0	2023-09-24 10:41:39+00:00	POINT (-2.98825 53.40690)
1	0	0	1	2023-09-24 10:41:46+00:00	POINT (-2.98793 53.40708)
2	0	0	2	2023-09-24 10:41:47+00:00	POINT (-2.98408 53.40931)
3	0	0	3	2023-09-24 10:41:49+00:00	POINT (-2.98391 53.40927)
4	0	0	4	2023-09-24 10:41:50+00:00	POINT (-2.98378 53.40933)

```
gps_track_points.plot(markersize = 0.5) # still are trackpoints still
```



9.1.2 Transform the `Point` GeoDataFrame in a `LineString` GeoDataFrame with `MovingPandas`

`MovingPandas` is a Python library designed for analyzing movement data using `Pandas`. It extends `Pandas`' capabilities to handle temporal and spatial data jointly. It is the ideal library for trajectory data analysis, visualization, and exploration. MovingPandas provides functionalities for processing and analyzing movement data, including trajectory segmentation, trajectory aggregation, and trajectory generalization.

See examples and documentation at https://movingpandas.org

The following code creates a `TrajectoryCollection` object by providing the GPS track points data, specifying the column that identifies trajectories, and indicating the column containing timestamps.

```
osm_traces = mpd.TrajectoryCollection(gps_track_points, 'track_fid', t='time')
print(f'The OSM traces download contains {len(osm_traces)} tracks')
```

```
The OSM traces download contains 14 tracks
```

```
for track in osm_traces:
    print(f'Track {track.id}: length={track.get_length(units="km"):.2f} km')
```

```
Track 0: length=9.96 km
Track 1: length=3.02 km
Track 2: length=2.84 km
Track 3: length=2.72 km
Track 4: length=2.92 km
Track 5: length=3.29 km
Track 6: length=4.95 km
Track 7: length=4.12 km
Track 8: length=4.87 km
Track 9: length=1.21 km
Track 10: length=0.03 km
Track 11: length=0.00 km
Track 12: length=0.70 km
Track 13: length=0.36 km
```

Below we set up some visualization options for `holoviews`. Holoviews is a library for creating interactive visualizations of complex datasets with ease. Holoviews provides a declarative approach to building visualizations, allowing you to express your visualization intent concisely. Holoviews combines the usage of Matplotlib, Bokeh, and Plotly and supports creating interactive visualizations. You can easily add interactive elements like hover tooltips, zooming, panning, and selection widgets to your plots. Compared to `folium`, however, it does not allow for as much freedom and development outside the box. Folium is primarily designed for creating web-based maps with features like tile layers and GeoJSON overlays, offering a high degree of customization for map-based visualizations. Holoviews, on the other hand, focuses more on general-purpose interactive visualizations beyond just maps, offering a broader range of visualization options.

See examples and documentation at https://holoviews.org.

```
from holoviews import opts, dim
import hvplot.pandas

plot_defaults = {'linewidth':5, 'capstyle':'round', 'figsize':(9,3), 'legend':True}
opts.defaults(opts.Overlay(active_tools=['wheel_zoom'], frame_width=500, frame_height=400))
hvplot_defaults = {'tiles':None, 'cmap':'Viridis', 'colorbar':True}
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_

Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_

Unable to display output for mime type(s): text/html

Unable to display output for mime type(s): application/vnd.holoviews_exec.v0+json, text/html
```

Traces generalisation

The `MinTimeDeltaGeneralizer` class from MovingPandas is applied to the `osm_traces` object. This class is responsible for simplifying or generalizing the GPS traces based on a minimum time delta. In this case, traces are being generalized to a tolerance of one minute.

```
from datetime import datetime, timedelta
osm_traces = mpd.MinTimeDeltaGeneralizer(osm_traces).generalize(tolerance=timedelta(minutes=1
osm_traces.hvplot(title='OSM Traces', line_width=3, width=700, height=400)
```

```
:Overlay
   .WMTS.I      :WMTS   [Longitude,Latitude]
   .Path.I      :Path   [Longitude,Latitude]
   .Path.II     :Path   [Longitude,Latitude]
   .Path.III    :Path   [Longitude,Latitude]
   .Path.IV     :Path   [Longitude,Latitude]
   .Path.V      :Path   [Longitude,Latitude]
   .Path.VI     :Path   [Longitude,Latitude]
   .Path.VII    :Path   [Longitude,Latitude]
   .Path.VIII   :Path   [Longitude,Latitude]
   .Path.IX     :Path   [Longitude,Latitude]
   .Path.X      :Path   [Longitude,Latitude]
   .Path.XI     :Path   [Longitude,Latitude]
   .Path.XII    :Path   [Longitude,Latitude]
   .Path.XIII   :Path   [Longitude,Latitude]
```

```
   .Path.XIV    :Path   [Longitude,Latitude]
   .Points.I    :Points   [Longitude,Latitude]   (triangle_angle)
   .Points.II   :Points   [Longitude,Latitude]   (triangle_angle)
   .Points.III  :Points   [Longitude,Latitude]   (triangle_angle)
   .Points.IV   :Points   [Longitude,Latitude]   (triangle_angle)
   .Points.V    :Points   [Longitude,Latitude]   (triangle_angle)
   .Points.VI   :Points   [Longitude,Latitude]   (triangle_angle)
   .Points.VII  :Points   [Longitude,Latitude]   (triangle_angle)
   .Points.VIII :Points   [Longitude,Latitude]   (triangle_angle)
   .Points.IX   :Points   [Longitude,Latitude]   (triangle_angle)
   .Points.X    :Points   [Longitude,Latitude]   (triangle_angle)
   .Points.XI   :Points   [Longitude,Latitude]   (triangle_angle)
   .Points.XII  :Points   [Longitude,Latitude]   (triangle_angle)
   .Points.XIII :Points   [Longitude,Latitude]   (triangle_angle)
   .Points.XIV  :Points   [Longitude,Latitude]   (triangle_angle)
```

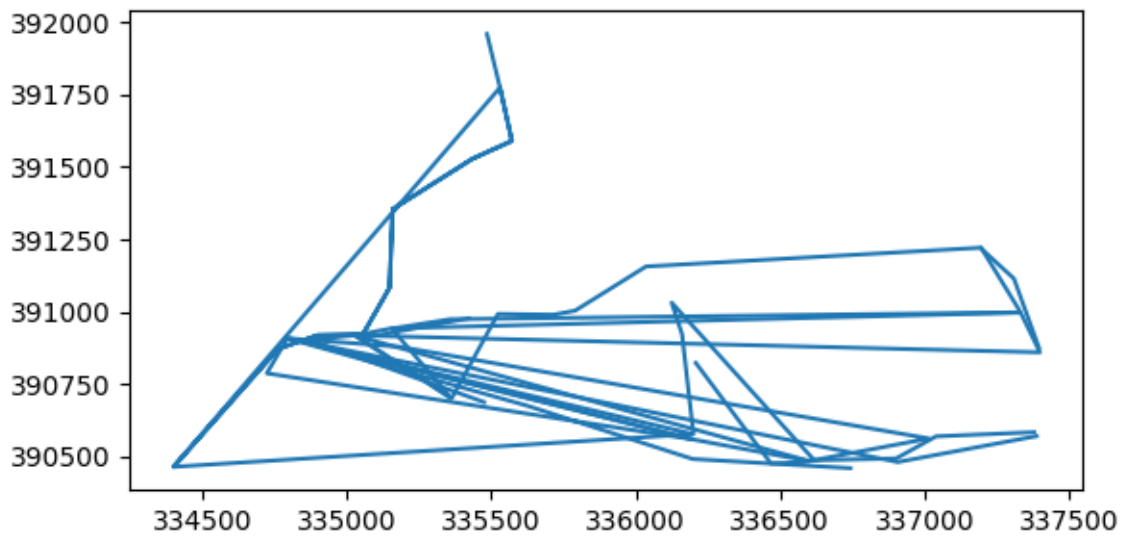
Visualising by speed

```
osm_traces.get_trajectory(0).add_speed(overwrite=True, units=("km","h"))
osm_traces.get_trajectory(0).hvplot(
    title='Speed (km/h) along track', c='speed', cmap='RdYlBu',
    line_width=3, width=700, height=400, tiles='CartoLight', colorbar=True)
```

```
:Overlay
   .WMTS.I   :WMTS   [Longitude,Latitude]
   .Path.I   :Path   [Longitude,Latitude]   (speed)
   .Points.I :Points   [Longitude,Latitude]   (speed,triangle_angle)
```

One can also convert the `TrajectoryCollection` (MovingPandas' object) to a LineString `GeoDataFrame`. This would allow us to take advantage of what we learnt across the previous sessions. For example, we can plot through `GeoPandas` and `matplotlib` after having projected the layer.

```
crs = 'EPSG:27700'
gps_tracks = osm_traces.to_traj_gdf()
gps_tracks = gps_tracks.set_crs("EPSG:4326").to_crs(crs)
```



As you can see, from the map above, OSM traces may include also odd traces that jump from different locations of the city.

Exercise: Explore the `gps_tracks` dataset and try to compute the average speed of each of the tracks. Try to understand, also using the `gps_track_points` (do not forget to project it, in case), if you can find a way to filter out traces that follow very odd routes.

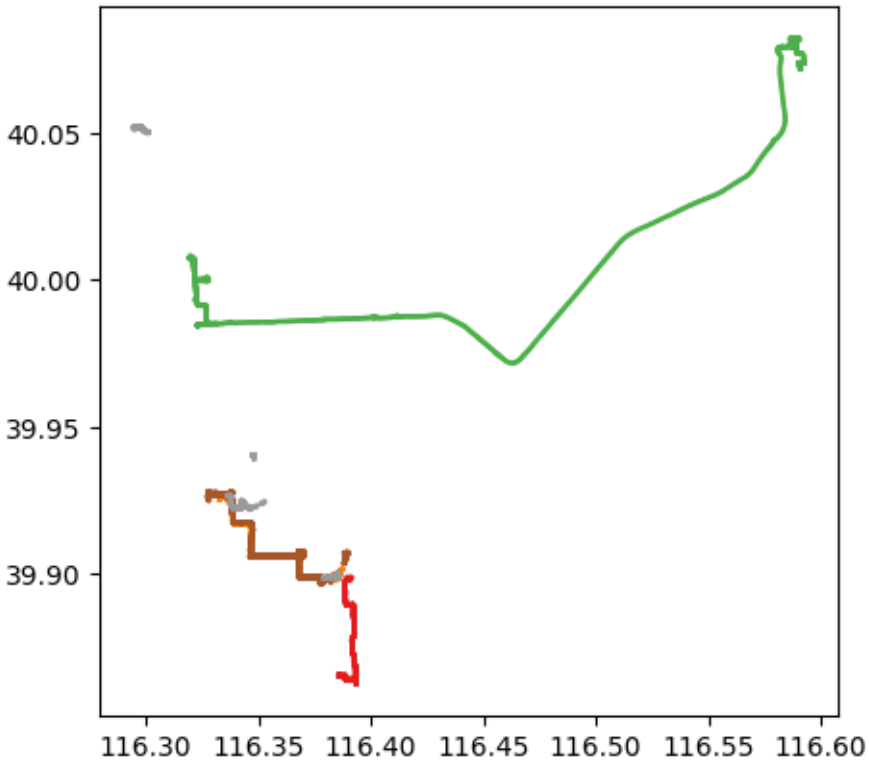
9.2 Animations in Folium

9.2.1 Animating GPS tracks

```
from folium.plugins import TimestampedGeoJson
```

Download this file in your data folder.

```
gdf = gpd.read_file('../data/geolife_small.gpkg')
gdf['t'] = pd.to_datetime(gdf['t'])
gdf = gdf.set_index('t').tz_localize(None)
gdf.index.name = None
gdf.plot(column = 'trajectory_id', markersize = 1, cmap = 'Set1')
```



- `gdf['t'] = pd.to_datetime(gdf['t'])`: This converts the values in the 't' column of `gdf` into datetime objects using `pd.to_datetime()`. This is useful for time-series data where 't' likely represents time information.
- `gdf = df.set_index('t').tz_localize(None)`: This sets the index to the column `t` and removes the timezone information using `tz_localize(None)`

```
def geo_df_to_timestamped_geojson(geo_df, colour="blue", marker_type="LineString"):
    """
    Convert a GeoDataFrame to a list of timestamped GeoJSON features.

    Parameters
    ----------
        geo_df: GeoDataFrame
            The GeoDataFrame containing the geographic data.
        colour:str
            The color of the line or marker. Defaults to "blue".
        marker_type: str
            The type of marker to use, e.g., "LineString" or "Point". Defaults to "LineStrin

    Returns
```

```
    -------
        feature_list: list
            A list of GeoJSON features, each representing a timestamped marker or line.
    """

    df = geo_df.copy()
    x_coords = df["geometry"].x.to_list()
    y_coords = df["geometry"].y.to_list()
    coordinate_list = [[x, y] for x, y in zip(x_coords, y_coords)]
    times_list = [i.isoformat() for i in df.index.to_list()]

    feature_list = [
       {
           "type": "Feature",
           "geometry": {
               "type": marker_type,
               "coordinates": coordinate_list,
           },
           "properties": {
               "times": times_list,
               "style": {
                   "color": colour,
                   "weight": 3,
               },
           },
       }
    ]

    return feature_list
```

Obtaining features from the `GeoDataFrame` so that we can plot them in `folium` directly.

```
features = geo_df_to_timestamped_geojson(gdf, colour="blue", marker_type="LineString")
```

This `TimestampedGeoJson` plugin allows us to draw data on an interactive folium map and animate it depending on a time variable (year, month, day, hour, etc) and supports any kind of geometry (points, line strings, etc). The only thing you have to do is to make sure to "feed" it with a geojson in the right format.

```
xy = gdf.unary_union.centroid
map = folium.Map(location=[xy.y, xy.x], zoom_start=10)
```

```
TimestampedGeoJson({"type": "FeatureCollection", "features": features},
    period="PT1S",
    add_last_point=True,
    transition_time=10
).add_to(map)

map
```

```
<folium.folium.Map at 0x1f06e0bf1d0>
```

The arguments of the `TimestampedGeoJson` plugin are:

- The geojson with the data.
- `period`: It is the time step that the animation will take starting from the first value. For example: "P1M" 1/month, "P1D" 1/day, "PT1H" 1/hour, and 'PT1M' 1/minute.
- `duration`: Period of time which the features will be shown on the map after their time has passed. If None, all previous times will be shown.

9.2.2 Other Animations with Folium

This part of the notebook has been readapted from this post.

We are going to use the data of New York's Citi Bike-sharing service. This is publicly available and can be downloaded from here. In this notebook, we used the data for the month of March 2024. Place it in your data folder, you can download it from here.

```
# Import the file as a Pandas DataFrame
nyc = pd.read_csv("../data/NY_march_bikeData.csv")
nyc.head()
```

	ride_id	rideable_type	started_at	ended_at	start_station_name
0	9F5EEFB7CF78C6EA	electric_bike	2024-03-21 13:32:42	2024-03-21 13:35:42	Stevens - River Ter &
1	01E2E1F218F9D303	electric_bike	2024-03-16 08:26:15	2024-03-16 08:47:53	Pershing Field
2	C2914D662B33AA55	electric_bike	2024-03-15 15:17:36	2024-03-15 15:32:12	Pershing Field
3	6723F7CE51888493	electric_bike	2024-03-13 08:34:26	2024-03-13 08:50:08	Pershing Field
4	EAB7E29AAB546941	electric_bike	2024-03-15 15:37:58	2024-03-15 16:19:32	Pershing Field

For every trip made we have:

- `started_at`: start date and time of the trip.

- `ended_at`: end date and time the trip.
- `start_station_id`, `start_station_name`, `start_lat`, and `start_lng`: all the useful info to identify and locate the start station.
- `end_station_id`, `end_station_name`, `end_lat`, and `end_lng`: all the useful info to identify and locate the end station of the trip.

```
type(nyc.loc[0, 'started_at'])
```

```
str
```

Transformation

The variables `started_at` and `ended_at` were imported as text (string). we are going to do the following:

- Change the type of these variables to datetime.
- Compute the duration of the trip in seconds.
- Define `started_at` as the data frame's index. This will allow us to pivot (group) values easier by time of day.
- Create a new column called `ty` to help us in the pivoting.

```
# Setting the right format for started_at and ended_at
nyc['started_at'] = pd.to_datetime(nyc['started_at'])
nyc['ended_at'] = pd.to_datetime(nyc['ended_at'])
nyc['duration'] = (nyc['ended_at'] - nyc['started_at']).dt.total_seconds()

# Define the startime as index and create a new column
nyc = nyc.set_index('started_at')
nyc['ty'] = 'station'
nyc.head(1)
```

started_at	ride_id	rideable_type	ended_at	start_station_name
2024-03-21 13:32:42	9F5EEFB7CF78C6EA	electric_bike	2024-03-21 13:35:42	Stevens - River Ter & 6 S

Now, we need to feed TimestampedGeoJson plugin with the right data format. To do so, we will transform the data frame. First, we count the number of trips per hour that start at each station

```
# Aggregate number of trips for each start station by hour of the day
start = nyc.pivot_table('duration',
                        index = ['start_station_id',
                                 'start_lat',
                                 'start_lng',
                                 nyc.index.hour],
                        columns = 'ty',
                        aggfunc='count').reset_index()

start.head()
```

ty	start_station_id	start_lat	start_lng	started_at	station
0	6786.02	40.714002	-74.093255	12	1
1	6839.10	40.714002	-74.093255	17	1
2	6948.10	40.714002	-74.093255	14	1
3	6955.05	40.714002	-74.093255	12	1
4	7141.07	40.714002	-74.093255	16	1

In the code above we use the pandas `pivot_table()` to group the data. Note that the last index we use is "nyc.index.hour". This takes the index of the data frame and, since it is of the datetime format, we can get the hour of that value like shown above. Similarly, we could get the day or month. Thus, to get a daily average we will divide by the numbers of days.

```
days = nyc.index.day.max()
start['station'] = start['station']/days
```

Now, we will change the name of the columns and define the color we want the points to have on the map.

```
# Rename the columns
start.columns = ['station_id', 'lat', 'lon', 'hour', 'count']

# Define the color
start['fillColor'] = '#53c688'

# The stops where less than one daily trip
# will have a different color
start.loc[start['count']<1, 'fillColor'] = '#586065'
start.head(1)
```

	station_id	lat	lon	hour	count	fillColor
0	6786.02	40.714002	-74.093255	12	0.032258	#586065

After transforming the data frame to the format we need, we have to define a function that takes the values from it and creates the geojson with the right properties to use in the plugin.

```
def create_geojson_features(df):
    features = []

    for _, row in df.iterrows():
        feature = {
            'type': 'Feature',
            'geometry': {
                'type':'Point',
                'coordinates':[row['lon'],row['lat']]
            },
            'properties': {
                'time': pd.to_datetime(row['hour'], unit='h').__str__(),
                'style': {'color' : ''},
                'icon': 'circle',
                'iconstyle':{
                    'fillColor': row['fillColor'],
                    'fillOpacity': 0.8,
                    'stroke': 'true',
                    'radius': row['count'] + 5
                }
            }
        }
        features.append(feature)

    return features
```

The function:

- Takes a data frame and iterates through its rows.
- Creates a feature and defines the geometry as a point, taking the lat and lon variables from our data frame.
- Defines the rest of the properties taking the other variables:

    1. Takes the hour variable to create a time property. ** This one is the most important step to animate our data. **
    2. Takes `fillColor` to create the fillColor property.

3. Defines the radius of the point as a function of the count variable.

Once the function is defined, we can use it on our data frame and get the geojson.

```
start_geojson = create_geojson_features(start)
start_geojson[0]
```

```
{'type': 'Feature',
 'geometry': {'type': 'Point',
  'coordinates': [-74.0932553, 40.714001716666665]},
 'properties': {'time': '1970-01-01 12:00:00',
  'style': {'color': ''},
  'icon': 'circle',
  'iconstyle': {'fillColor': '#586065',
   'fillOpacity': 0.8,
   'stroke': 'true',
   'radius': 5.032258064516129}}}
```

With this, we can now create the animated map.

```
nyc_map = folium.Map(location = [40.71958611647166, -74.0431174635887],
                     tiles = "CartoDB Positron",
                     zoom_start = 14)

TimestampedGeoJson(start_geojson,
                   period = 'PT1H',
                   duration = 'PT1M',
                   transition_time = 1000,
                   auto_play = True).add_to(nyc_map)

nyc_map
```

```
<folium.folium.Map at 0x1f06e08a210>
```

It might be a good idea to analyze where trips end at different times of the day. Even more, it could be interesting to see both maps side by side and try to identify a pattern.

The DualMap plugin can help us achieve this. To use it, we will first run a similar script to the one we did above to get the number of trips that end at each station by time of the day.

```
nyc1 = nyc.reset_index().set_index('ended_at')
end = nyc1.pivot_table('duration',
                       index = ['end_station_id',
                                'end_lat',
                                'end_lng',
                                nyc1.index.hour],
                       columns = 'ty',
                       aggfunc='count').reset_index()

end['station'] = end['station']/days

end.columns = ['station_id', 'lat', 'lon', 'hour', 'count']
end['fillColor'] = '#e64c4e'
end.loc[end['count']<1, 'fillColor'] = '#586065'
end_geojson = create_geojson_features(end)
```

We first create the dual map and then we add start_geojson to the left map (m1) and the end_geojson to the right map (m2).

```
from folium.plugins import DualMap

dualmap = DualMap(location = [40.71958611647166, -74.0431174635887],
                  tiles = 'cartodbpositron',
                  zoom_start = 14)

TimestampedGeoJson(start_geojson,
                   period = 'PT1H',
                   duration = 'PT1M',
                   transition_time = 1000,
                   auto_play = True).add_to(dualmap.m1)

TimestampedGeoJson(end_geojson,
                   period = 'PT1H',
                   duration = 'PT1M',
                   transition_time = 1000,
                   auto_play = True).add_to(dualmap.m2)

dualmap
```

```
<folium.plugins.dual_map.DualMap at 0x1f06f4caa50>
```

Another interesting plugin, `HeatmapWithTime`, allows animating heatmaps. According to the documentation, the input needs to be a list of lists:

"data (*list of lists of points of the form [lat, lng] or [lat, lng, weight]*) – The points you want to plot. The outer list corresponds to the various time steps in sequential order. (weight is in (0, 1] range and defaults to 1 if not specified for a point)"

Therefore, we need to create this list of points for each hour of the day and then put all those lists in a new one.

```
# Create an empty list
df_hour_list = []

# Create a series with the different hours of the day
hours = pd.Series(nyc.index.hour.unique().sort_values())

# Create a list of points for each hour of the day
def create_list(hour):
    df_hour_list.append(nyc.loc[nyc.index.hour == hour,
                                ['start_lat',
                                 'start_lng']].
                        groupby(['start_lat',
                                 'start_lng']).sum().reset_index().values.tolist())
hours.apply(create_list);
```

```
from folium.plugins import HeatMapWithTime

location = [40.71958611647166, -74.0431174635887]
# Add trip events to the map
map_time = folium.Map(location=location,
                      tiles="CartoDB Positron",
                      zoom_start=12)

HeatMapWithTime(df_hour_list,
                auto_play=True,
                max_opacity=0.5,
                gradient = {0.2: '#FBD973',
                            0.4: '#fa782f',
                            0.75: '#F16578',
                            1: '#782890'}).add_to(map_time)
map_time
```

```
<folium.folium.Map at 0x1f06e05f890>
```

Exercise:

Explore the other `folium` plugins and think which of them could be useful for your last assignment. See: https://python-visualization.github.io/folium/latest/user_guide/plugins.html

If you have time also have a look at PyDeck.

Pydeck is a Python library used for creating interactive maps and data visualizations. It can be exploited to generate appealing maps with features like scatter plots, line plots, choropleths, and, more importantly, **3D visualisations**. Pydeck is built on top of Deck.gl, a WebGL-powered library for rendering large-scale data visualizations. It offers various customization options for styling and configuring the visualizations and supports integration with `Pandas` and `GeoPandas`.

Pydeck installation is not trivial and can be result in conflicts with other libraries/versions, see: https://deckgl.readthedocs.io/en/latest/installation.html

Its usage for the final assignment is optional.

10 Assignment II Example

```
import geopandas as gpd
import hvplot.pandas
import numpy as np
import pandas as pd
import panel as pn
import plotly.express as px
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): application/vnd.holoviews_exec.v0+json, text/html
```

This notebook demonstrates how to export a `.ipynb` notebook file to a `.HTML` file (easy) without losing the functionalities of dashboards created with `panel` (i.e. interactivity). This is a basic example, but should work even for more complex cases

10.1 Saving Static HTML with Panel-based dashboards

10.1.1 Initial steps (case-specific), just to get some data

Need the code below to visualise the dashboard

```
# Initialize Panel with/ without extensions
pn.extension()
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.holoviews_

Unable to display output for mime type(s): text/html

Unable to display output for mime type(s): application/vnd.holoviews_exec.v0+json, text/html
```

```
# load the df
df = pd.read_csv('../data/GTD_2022.csv', low_memory=False)
df = df.dropna(subset=['latitude', 'longitude'])
df = df[df.region_txt == 'Middle East & North Africa']
countries = sorted(df['country_txt'].unique().tolist())
country_selector = pn.widgets.Select(name='Country', options=countries)
```

```
# Function to create a map centered on the selected country
import folium
from folium.plugins import MarkerCluster

def create_foliumMap(data):

    # Calculate the mean latitude and longitude to center the map
    center_lat = data['latitude'].mean()
    center_lon = data['longitude'].mean()

    # Create a Folium map centered on the average location
    folium_map = folium.Map(location=[center_lat, center_lon], zoom_start=6)

    # Use a MarkerCluster to add markers for each event
    marker_cluster = MarkerCluster().add_to(folium_map)

    # Add a marker for each event
    for idx, row in data.iterrows():
        folium.Marker(
            location=[row['latitude'], row['longitude']],
            popup=f"Date: {row['date']}<br>Deaths: {row['nkill']}",
        ).add_to(marker_cluster)

    # Return the Folium map object
    return folium_map
```

This function communicates with the one above for updating the attributes used to create the Folium map.

```
# Panel doesn't directly render Folium maps, so we need to render it as HTML
def update_map_country(df, country, width, height):
    # Filter the DataFrame for the selected country
    data = df[df['country_txt'] == country].copy()
    folium_map = create_foliumMap(data)
    # Panel doesn't directly render Folium maps, so we need to render it as HTML
    return pn.pane.HTML(folium_map._repr_html_(), width=width, height=height)
```

Then we bind the function to the widget and pass the DataFrame, along with the size attributes.

```
width = 700
height = 500
map_pane = pn.bind(update_map_country, df,  country_selector.param.value, width, height)

# Layout the dashboard
dashboard = pn.Column(
    pn.Row(country_selector),
    map_pane
)
```

10.1.2 Important Steps for making the dashboard interactive in the static html:

10.1.2.1 Export the dashboard locally, on your own machine.

```
from bokeh.resources import INLINE
dashboard.save('M:\dashboard\dashboard.html', embed=True, resources=INLINE)
```

```
WARNING:bokeh.core.validation.check:W-1005 (FIXED_SIZING_MODE): 'fixed' sizing mode requires
```

10.1.3 Then incorporate the dashboard as an HTML object

```
from IPython.display import HTML

# Assuming 'example.html' is your HTML file
with open('M:\dashboard\dashboard.html', 'r') as file:
    html_content = file.read()

# Display the HTML content in the notebook
HTML(html_content)
```

10.1.3.1 You can still show the "original layour" (if it differs). This won't be interactive in your submission but will give the reader an idea.

```
dashboard.servable()
```

```
Column(design=<class 'panel.theme.materi...)
    [0] Row(design=<class 'panel.theme.materi...)
        [0] Select(design=<class 'panel.theme.materi..., name='Country', options=['Algeria',
    [1] ParamFunction(function, _pane=HTML, defer_load=False, design=<class 'panel.theme.mat
```

10.1.3.2 Now export the jupyter notebook file to a static htlm (follow instructions here).

If these steps do not work, ask on Teams, before the 1st of May, or in class.