**Introduction**

This project aims at implementing various open-source geospatial libraries on Python such as Pandas, Geopandas, Earthpy, Matplotlib, and Descartes in JupyterLab equipped for spatial analysis over spatial datasets and shapefiles. Additionally, the project demonstrates how to perform spatial analysis such Buffering and Dissolving as well as data visualization like plotting data on map, making histogram for spatial dataset without using software like ArcGIS Pro but only implementing Python codes on JupyterLab. This project also trains us to run the workflow of analyzing spatial data but using our own datasets to be more familiar with Python and spatial analysis.

**Study Area**

The study areas are Antwerp, Belgium (Figure 1) and Seattle, the United States. Antwerp is the largest city in Belgium and the capital of Antwerp Province in the Flemish Region. It’s an urban center both economically and culturally.

地图

描述已自动生成

Figure 1 Study Area of Antwerp

The second area is Seattle, the United States (Figure 2). Seattel is a seaport city located on the West Coast of the U.S. It’s the most populous city in both the state of Washington and the Pacific Northwest region of North America. It’s located between the saltwater Puget Sound to the west and Lake Washington to the east.

地图

描述已自动生成

Figure 2 Study Area of Seattle

**Data and Data Processing**

|  |  |  |
| --- | --- | --- |
| Name | Projected Coordinate System | Description |
| Landuse.shp | EPSG: 31370 | Land use polygon shapefile covers the city of Antwerp |
| Layer\_streets.shp | EPSG: 31370 | Street line shapefile covers the city of Antwerp |
| CRAB\_ subset.shp | EPSG: 31370 | Address point shapefile covers the city of Antwerp |
| DWW\_Storm\_Outfalls.shp | EPSG: 4326 | Outfall shapefile covers the city of Seattle |
| Seattle\_Streets.shp | EPSG: 4326 | Street line shapefile covers the city of Antwerp |
| Zoning\_Detailed.shp | EPSG: 4326 | Bounding area shapefile covers the city of Seattle |

**Methods**

1. **Setting Up Environment**

This project is implemented on JupyterLab by using Anaconda Navigator. After fully setting up the Anaconda Navigator, I launched JupyterLab and checked the environment of working notebook. There were a few geospatial libraries remain uninstalled. I updated index for the environment and encountered errors when typing commands on the terminal window. The version of Anaconda Navigator has conflicts with the version of some geospatial libraries. Finally, I use ‘pip install’ in notebook and then install the open-sourced libraries (Figure 3).

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

Figure 3 Coding to install geospatial libraries.

1. **Read Files and Identify Projected Coordinate System**

After configuring the environment and installed essential libraries for further analysis, I used geopandas.read\_file() to create a GeoDataFrame object converting the original dataset to specific format for plotting and performing geospatial analysis (Figure 4).

图形用户界面, 文本, 应用程序

描述已自动生成

Figure 4 use of function provided by GeoPandas

1. **Select Attributes and check if it needs to be reprojected**

Use the function provided by GeoPandas to get the CRS of the data and print them before dropping the first row. (Figure 5)

Use the function provided by GeoPandas to get the CRS of the data and print them before dropping titles of each column. (Figure 5) ‘data.crs’ is an attribute that we can access in GeoPandas to get information about the CRS of a GeoDataFrame. It returns a CRS object that can tell us the CRS of this dataframe.

图形用户界面

中度可信度描述已自动生成

Figure 5 the visualization of data

1. **Calculating areas**

In geospatial libraries like GeoPandas, ‘geometry.area’ calculates the area of a geometric shape within a GeoDataFrame. And using ‘geometry.area.hist’ can create a histogram for the area (Figure 6). I used these functions and created a histogram of Antwerp.

图表, 直方图

描述已自动生成

Figure 6 Python codes that create a histogram

1. **Set Map Bounds for Plotting**

Before I want to formulate a colored map on JupyterLab, I need to adjust the original data because plotting data on map needs a boundary for each axis (Figure 7).

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

Figure 7 minimum and maximum data values for the boundary

1. **Visualization Python Code**

I imported Descartes for plotting polygons and the exhibited the different land use conditions classified by different colors. Then I used matplotlib for plotting points and created graphs for spatial analysis such as Buffering (Figure 8).

文本

描述已自动生成

Figure 8 using descrates to plot areas for different use

**Results**

Figure 9 represents the land use graph of the city of Antwerp

文本

描述已自动生成

Figure 9 Land Use Graph of the City of Antwerp

The following plot demonstrates the distribution of urban green areas in the city of Antwerp (Figure 10).

**图示

描述已自动生成**

Figure 10 Urban Green Areas

The 300m buffer around the green areas represents the coverage of green area of the city of Antwerp. This graph shows the extremely high coverage of plants in Antwerp that is very untypical for such a modern city nowadays especially a center of developed country both economically and culturally.

图示

中度可信度描述已自动生成

Figure 11 Urban Green Areas with Buffer polygons

After rerun the workflow of these spatial analysis, my dataset shows where the outfalls locate with a small buffer area. An outfall refers to the point where water from drainage exits a pipe and flows out of the established sewer and drainage network, eventually entering a stream, body of water, ditch, or similar watercourse. These outfalls are storm outfalls that are located around the lake and the river in the city, demonstrating the effectiveness of Seattle's sewer system and its ability to withstand storms (Figure 12).

地图

描述已自动生成

Figure 12 Urban outfalls in the city of Seattle

**Discussion**

This project utilizes various open-source geospatial Python libraries, including Pandas, GeoPandas, Earthpy, and Descartes. Additionally, it incorporates several common and useful spatial analysis techniques. The most significant challenge I encountered was that the found dataset couldn't be seamlessly integrated into the original spatial analysis workflow, making it challenging to configure the parameters for coding to achieve an ideal output. My original data has complex features and preparing the dataset for use in JupyterLab proved to be challenging. However, I skipped certain steps like calculating area and instead used Matplotlib and buffering to obtain the final output. Since outfalls don’t need to be classified by different types, setting bounding values for points was unnecessary.

**Reference**

Wertz, Ingrid. *Combined Sewer Overflow (CSO) 59 Outfall Cleaning SEPA Determination of Non-Significance (DNS)*.