

Remote Landfill Atmospheric Gas Monitoring Tools - How to Guide

Nicholas Kinsella
Ulster University

Table of Contents

1. Introduction and justification	3
2. Setup	4
2.1 Anaconda Navigator	4
2.2. Creating a Conda Environment.....	4
2.3 Setting up Jupyter Lab	5
2.4 OpenEO setup using Anaconda Navigator.....	8
2.5 OpenEO setup using PyPi.....	9
2.6 Registering with Copernicus Data Space Ecosystem.	9
2.7 Running the tools in Jupyter-lab.....	11
2.8 Authentication with OpenEO.....	12
3. Data and Dependencies	13
4. Methodology.....	14
4.1 Sentinel-5 Atmospheric Gas Timeseries:	14
4.2 Sentinel-5 Atmospheric Gas Concentration Map:	18
4.3 Sentinel-2 Multi-Band-Multi-Pass CH ₄ Map	22
5. Expected Results and Demo	29
5.1 Sentinel-5 Atmospheric Gas Timeseries:	29
5.2 Sentinel-5 Atmospheric Gas Concentration Map:	30
5.3 Sentinel-2 Multi-Band-Multi-Pass CH ₄ Map:	31
6. Troubleshooting.....	32
6.1 Remote disconnected error (All tools)	32
6.2 Concurrent job error (S5-AGM)	32
6.3 Value error (S2-MBMP)	34
6.4 Internal Server Error (All Tools)	34
7. References	35

1. Introduction and justification

Methane (CH₄), a greenhouse gas with a warming potential 28 times greater than CO₂ over a 100-year period, has seen its atmospheric concentration increase by over 250% since the industrial revolution. Despite CH₄'s shorter atmospheric lifespan due to ultraviolet sunlight interaction (Vigano et al., 2008) it still contributes to at least a quarter of anthropogenic warming (Pandey et al., 2023).

Mismanaged landfills can produce significant CH₄ emissions (Ferronato et al., 2017) and the EU Landfill Directive of 1999 mandates the collection or flaring of CH₄ produced by organic waste decomposition (European Union, 1999; Themelis & Ulloa, 2007). Spain, which sent 11.5 million tonnes of waste to landfills in 2017 (European Environment Agency, 2022), has experienced large emission events despite landfill management on par with other EU countries (Castillo-Giménez et al., 2019; European Space Agency, 2021).

Satellite missions have in recent years improved their CH₄ measurement capabilities, offering advantages over ground-based detectors (Parker et al., 2011). PreZero, a waste management company operating 23 landfills in Spain, currently relies on ground-based detectors in a one detector per hectare grid. The tools outlined in this guide have been created while consulting with them. They expressed interest in other gasses beyond CH₄ so these have been included where possible (Aguasca, 2024; Hidalgo, 2024; Salami, 2024).

Three Python tools were created to display atmospheric gas data from Sentinel-5 and CH₄ data from Sentinel-2 satellites.

- **Sentinel-5 Atmospheric Gas Timeseries (S5-AGT):** A time series for each landfill location showing atmospheric gas concentrations of Carbon monoxide (CO), formaldehyde (HCHO), nitro dioxide (NO₂), ozone (O₃), sulphur dioxide (SO₂) and methane (CH₄).
- **Sentinel-5 Atmospheric Gas Map (S5-AGM):** a map of daily atmospheric gas concentrations of CO, HCHO, NO₂, O₃, SO₂ and CH₄.
- **Sentinel-2 Multi-Band-Multi-Pass CH₄ Map (S2-MBMP):** A high-resolution map that can show CH₄ point sources and clouds for selected landfill.

The tools can be downloaded or cloned from the following Github repository. This guide outlines their installation and use. The files for which can be found at the following URL:

[https://github.com/zelcon01/Landfill Atmospheric Gas Monitor Tools](https://github.com/zelcon01/Landfill_Atmospheric_Gas_Monitor_Tools)

2. Setup

Anaconda Navigator, containing Python and additional tools, includes 'Conda', a package manager for creating shareable environments with necessary packages. This includes Jupyter Lab for running Python code. This will be installed first.

2.1 Anaconda Navigator

To download Anaconda, navigate to <https://docs.anaconda.com/anaconda/install/> and follow the instructions of your associated operating system.

2.2. Creating a Conda Environment

In the Anaconda Navigator side bar, click the 'Environments' tab. You will see the installed packages (fig.1).

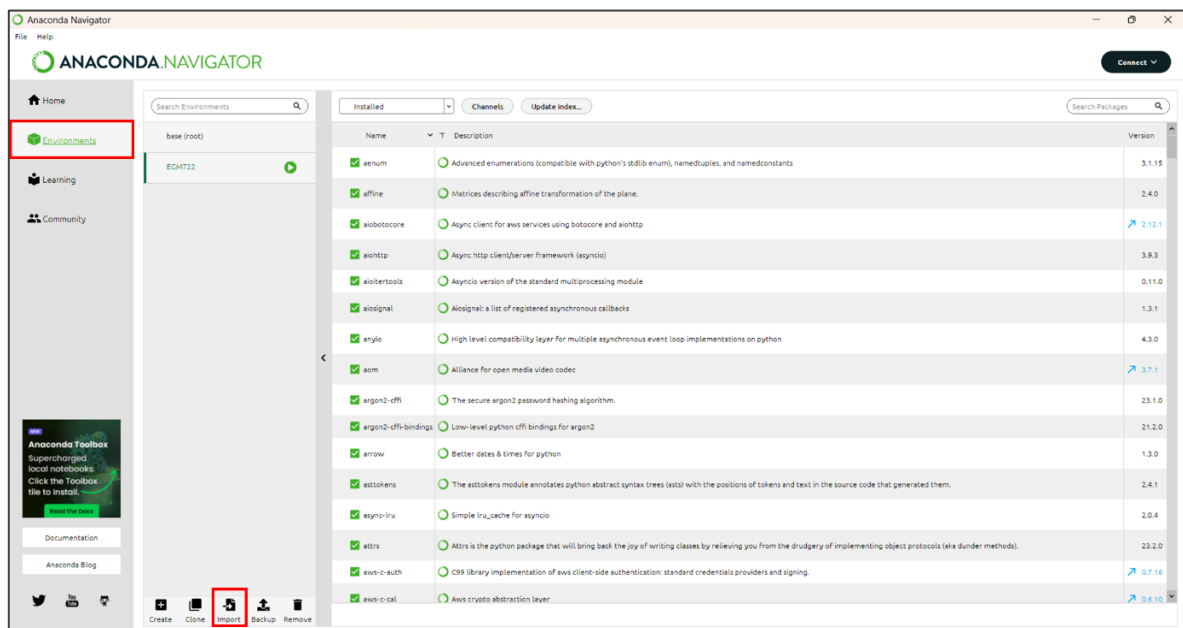


Figure 1: Environments tab of Anaconda Navigator with environments tab and import button highlighted in red.

Next click on the imports tab (fig.1) and select the file 'environment.yml' contained in the .zip file of the tool's download, choosing an appropriate name for the environment (fig.2).

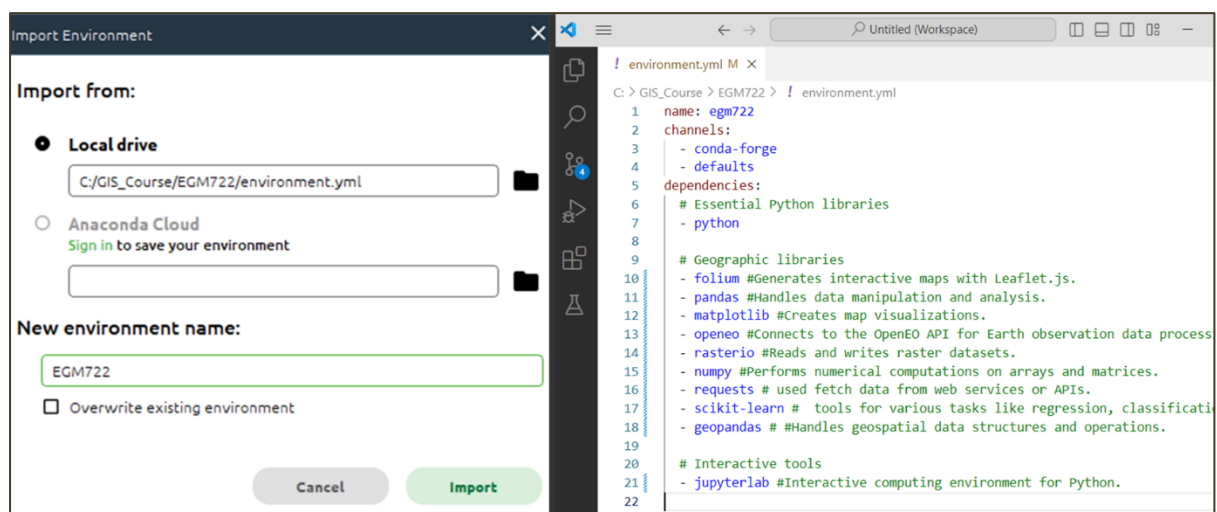


Figure 2: The import config box (left) and the contents of 'environment.yml' (right).

Click Import. Depending on the connection speed of your network, the process of installing all the packages may take several minutes. Once finished you will be returned to the environments tab (fig.1)

Next click on the 'Home' tab in Anaconda Navigator's sidebar (fig.3).

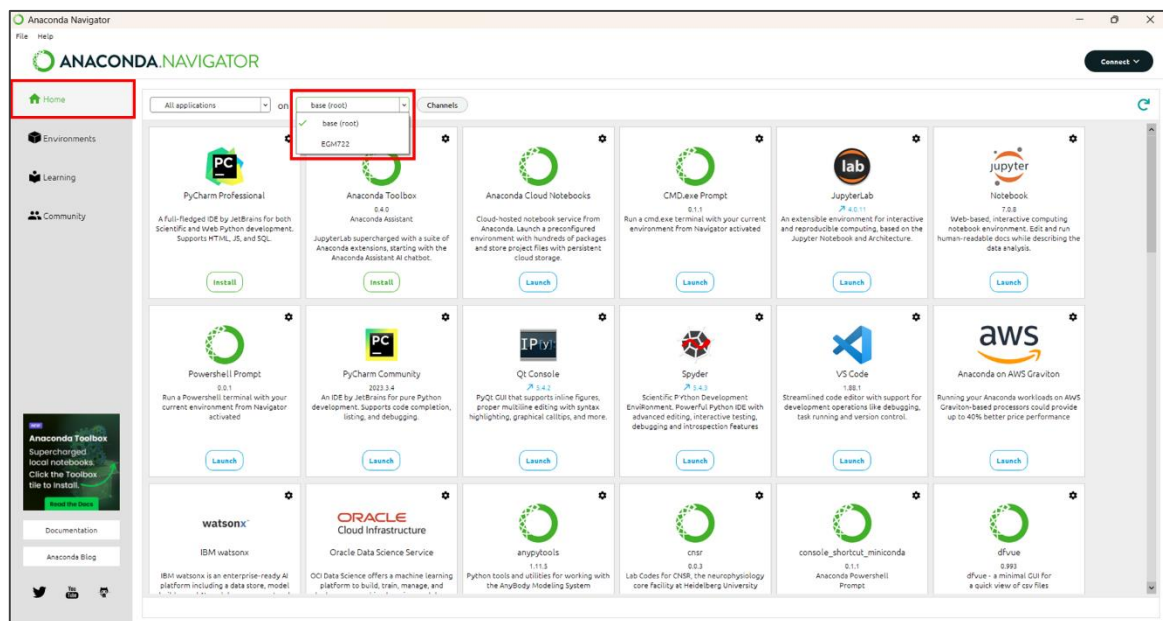


Figure 3: Anaconda Navigator with home tab and environment switching dropdown in red.

The dropdown highlighted in figure 3 should display two options, 'base (root)' and the name of your new environment (in figure 3 this is 'EGM722'). **Ensure your environment name is always selected here or the dependencies installed earlier will not be available to it.**

2.3 Setting up Jupyter Lab

A configuration file ('.config') needs to be created to change the settings used by Jupyter Lab by default. Launch the CMD.exe prompt (fig.4)

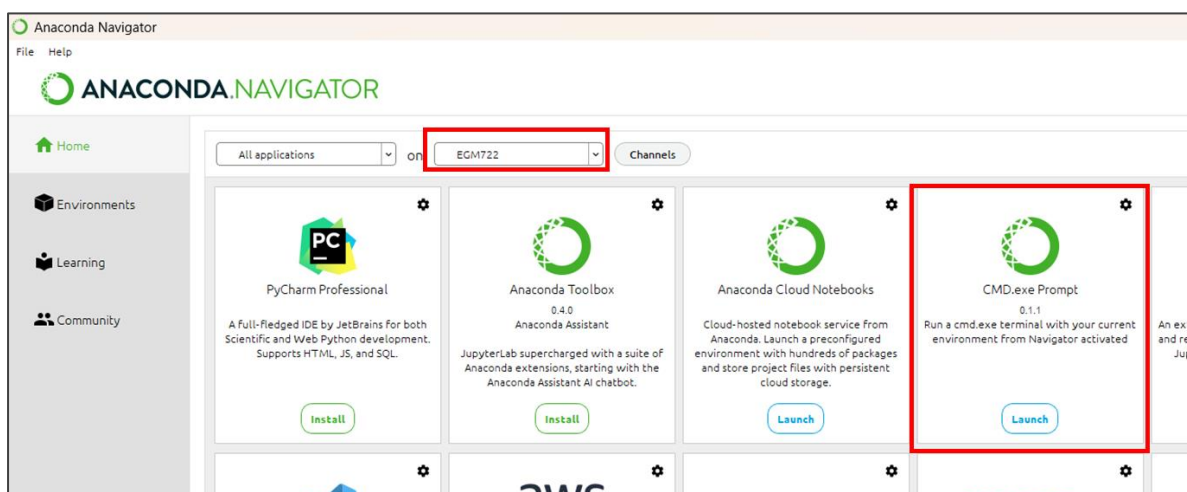


Figure 4: Highlighted locations of selected environment and CMD.exe Prompt

In the command prompt, enter the command:

```
jupyter lab --generate-config
```

This will create a new folder in your user directory called 'jupyter' containing a python script `jupyter_lab_config.py`. On Windows this is usually 'C:\Users\<your_username>'.

Jupyter lab will by default open in your user directory. Due to security restrictions, it is not possible to navigate to the parent directory of the launch location. So if Jupyter launches in 'C:\Users\RockyBalboa', it is not possible to move to 'C:\Users' or 'C:\EGM722'. If the directory you are keeping your data in is outside your user directory, you will need to change the default opening folder to your data directory.

This location is also where you should store the following files and folder:

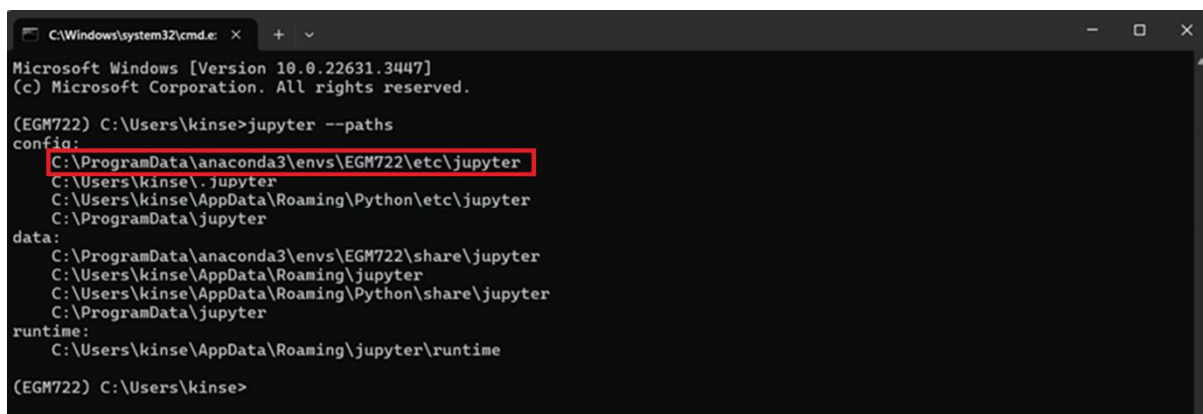
- Sentinel_5P_Atmospheric_Gas_Time_Series.ipynb
- Sentinel_5P_Atmospheric_Gas_Map.ipynb
- Sentinel_2_CH4_Multi-Band-Multi-Pass.ipynb
- The folder "Data"

If your data directory is in your user directory, you should be able to click and navigate there using the interface of Jupyter Lab. If that is not the case, you will need to do the following:

Open an Anaconda Navigator CMD.exe prompt and type the following command:

```
jupyter --paths
```

This will show something like figure 5 although your file paths will be unique to you.



```

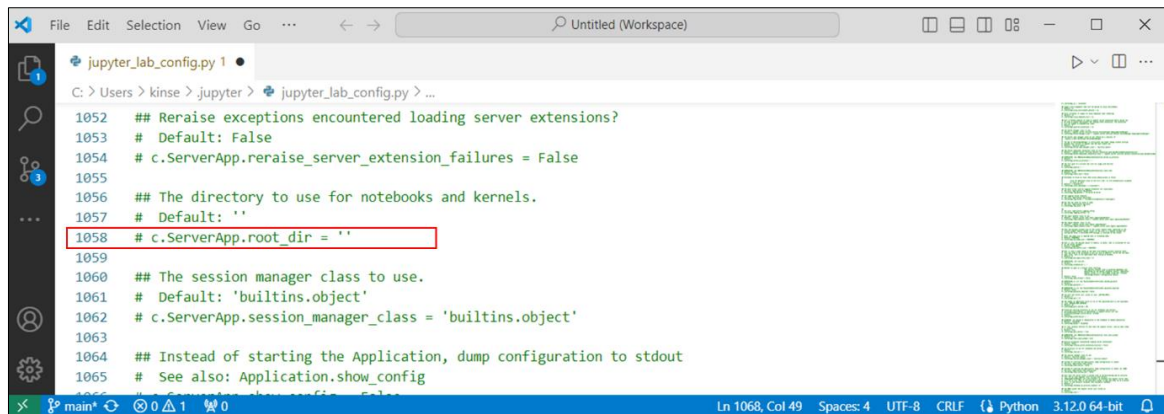
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

(EGM722) C:\Users\kinse>jupyter --paths
config:
  C:\ProgramData\anaconda3\envs\EGM722\etc\jupyter
  C:\Users\kinse\.jupyter
  C:\Users\kinse\AppData\Roaming\Python\etc\jupyter
  C:\ProgramData\jupyter
data:
  C:\ProgramData\anaconda3\envs\EGM722\share\jupyter
  C:\Users\kinse\AppData\Roaming\jupyter
  C:\Users\kinse\AppData\Roaming\Python\share\jupyter
  C:\ProgramData\jupyter
runtime:
  C:\Users\kinse\AppData\Roaming\jupyter\runtime
(EGM722) C:\Users\kinse>
  
```

Figure 5: results of 'jupyter --paths' command showing path used by environment highlighted in red.

The 'jupyter_lab_config.py' file mentioned earlier needs to be copy pasted into that folder.

Once 'jupyter_lab_config.py' file has been moved, open it in Notepad++, Visual Studio Code or if you don't have those, Notepad. Using the shortcut 'CTRL + F' type in the following line: 'c.ServerApp.root_dir' (without quotes) and you should find the section highlighted in figure 6.



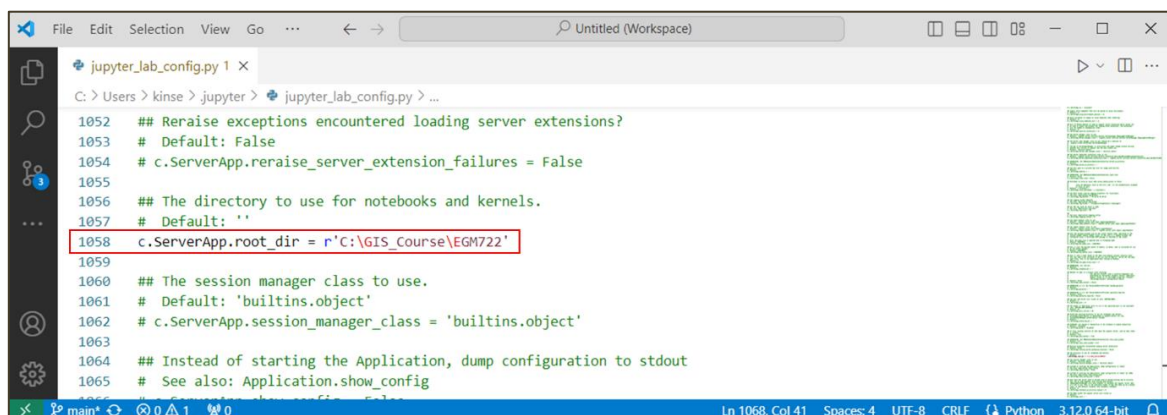
```

1052  ## Reraise exceptions encountered loading server extensions?
1053  # Default: False
1054  # c.ServerApp.reraise_server_extension_failures = False
1055
1056  ## The directory to use for notebooks and kernels.
1057  # Default: ''
1058  # c.ServerApp.root_dir = ''
1059
1060  ## The session manager class to use.
1061  # Default: 'builtins.object'
1062  # c.ServerApp.session_manager_class = 'builtins.object'
1063
1064  ## Instead of starting the Application, dump configuration to stdout
1065  # See also: Application.show_config

```

Figure 6: location of 'c.ServerApp.root_dir' in jupyter_lab_config.py

Remove the '#' and space from the start and add the path used by your environment between the quote marks, adding a 'r' beforehand (fig.7).



```

1052  ## Reraise exceptions encountered loading server extensions?
1053  # Default: False
1054  # c.ServerApp.reraise_server_extension_failures = False
1055
1056  ## The directory to use for notebooks and kernels.
1057  # Default: ''
1058  c.ServerApp.root_dir = r'c:\GIS_Course\EGM722'
1059
1060  ## The session manager class to use.
1061  # Default: 'builtins.object'
1062  # c.ServerApp.session_manager_class = 'builtins.object'
1063
1064  ## Instead of starting the Application, dump configuration to stdout
1065  # See also: Application.show_config

```

Figure 7: path to data directory added to jupyter_lab_config.py

Save and close this file and return to the Anaconda Navigator 'Home' tab. Launch Jupyter Lab and if you have followed the steps correctly, you should see that your data directory is automatically displayed (fig.8).

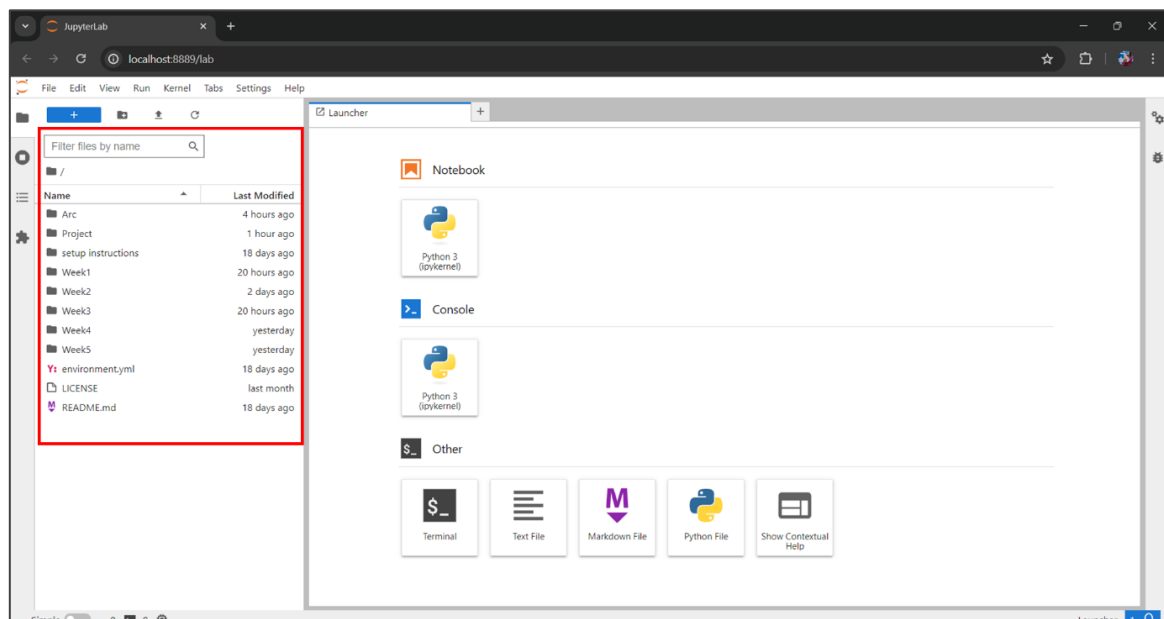


Figure 8: Jupyter Lab showing by default the data directory

2.4 OpenEO setup using Anaconda Navigator

OpenEO is an open-source API that allows access to the earth observation satellite missions run by the Copernicus program. These include the satellites used by this tool.

First search in the Anaconda Navigator environments tab for 'openeo'. Make sure that 'Not installed' is selected (fig.9). If the package appears here, click its tick box and select apply. If you can't see it here, please go to section 2.5.

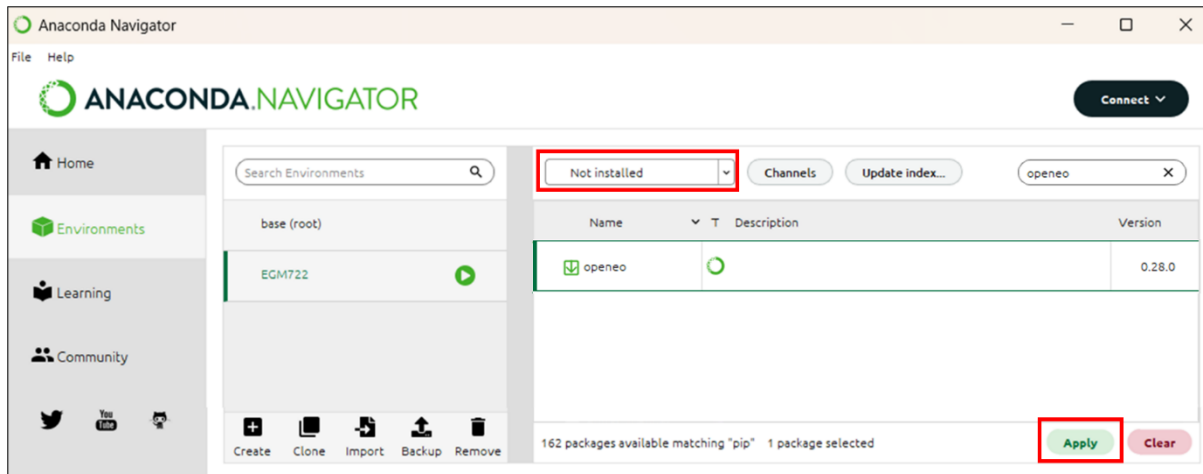


Figure 9: installing OpenEO from Anaconda Navigator.

Next you will be presented with the following screen (fig.10). Once this has finished processing the request. Simply click 'apply' to begin the installation.

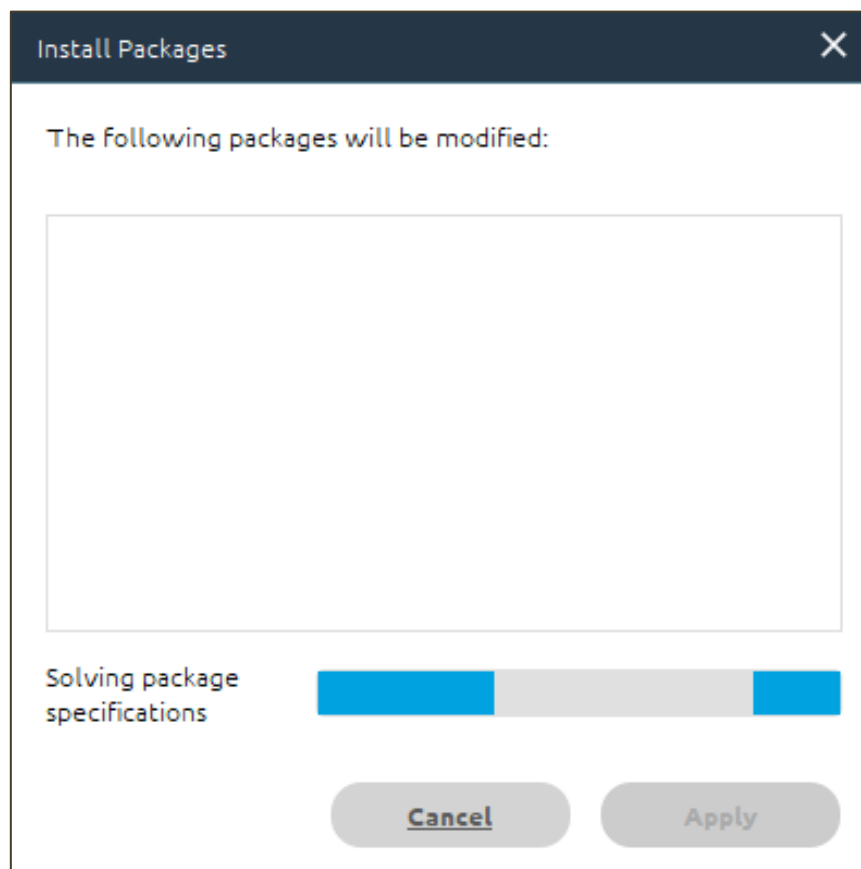


Figure 10: Anaconda Navigator package installer loading screen.

2.5 OpenEO setup using PyPi

If Anaconda Navigator cannot find OpenEO you can use PyPi, the official third-party software library for Python. Search for 'pip', selecting the appropriate tick-box and then clicking apply (fig.11), then clicking apply once the install packages prompt has finished loading (fig.10).

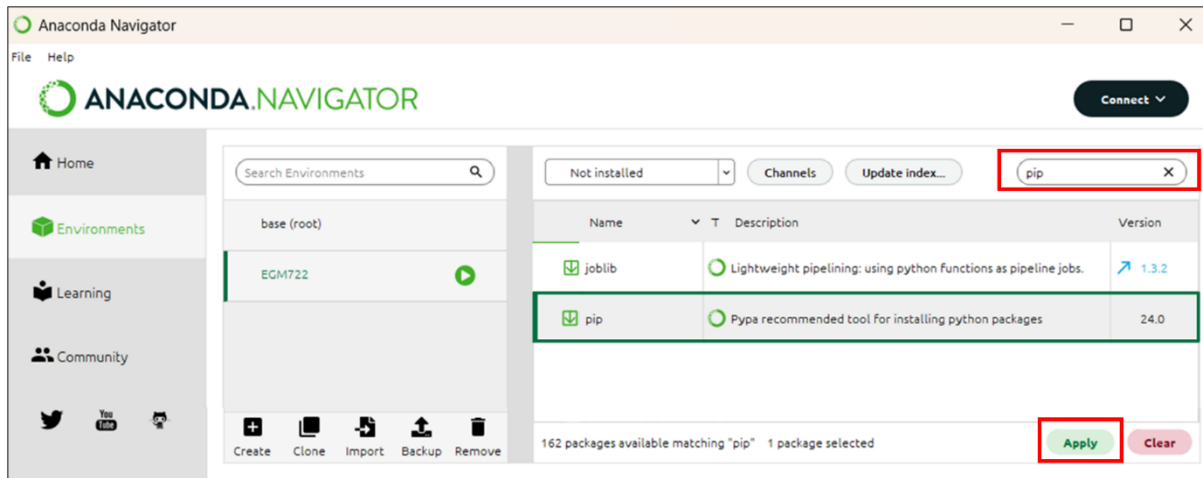


Figure 11: Installing pip via Anaconda Navigator

Open an Anaconda Navigator CMD.exe prompt and type the following command:

```
pip install openeo
```

Once the process has completed, you can close the CMD.exe prompt window.

2.6 Registering with Copernicus Data Space Ecosystem.

Accessing and analysing OpenEO data requires an authentication. To do this, you need to complete a Copernicus Data Space Ecosystem Registration. Go to <https://dataspace.copernicus.eu/> and click the green login button (fig.12)

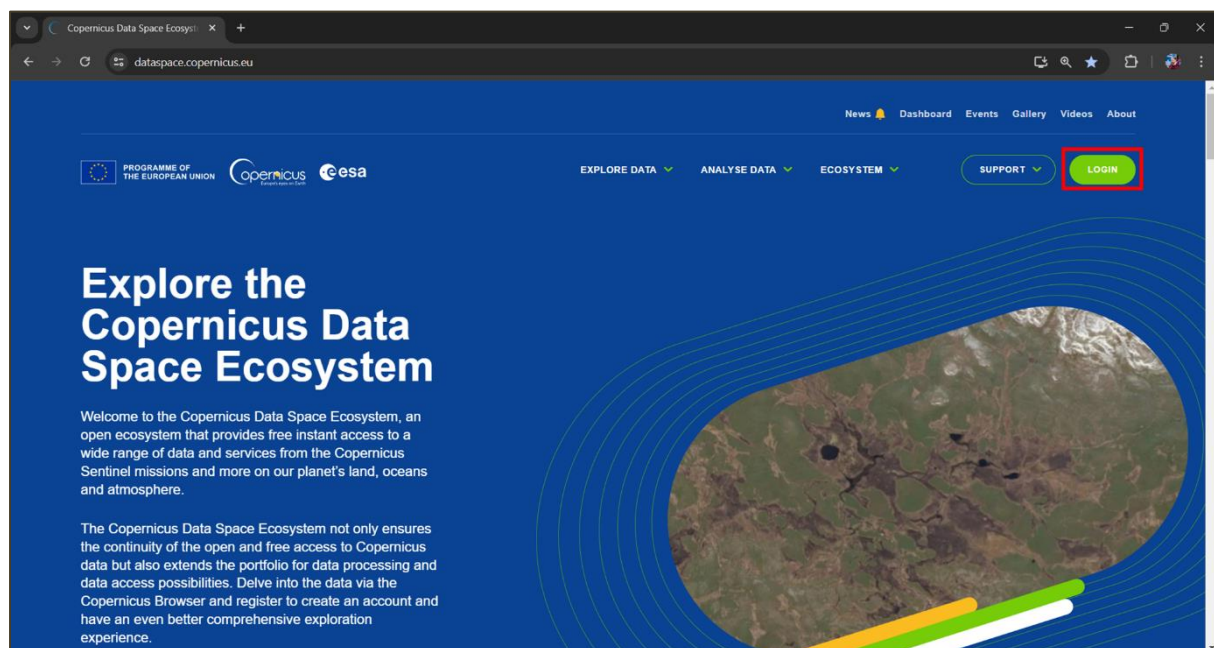


Figure 12: Copernicus Dataspace landing page with login button highlighted in red.

Next click the green 'register' button (fig.13):

Figure 13: Copernicus Dataspace sign in page.

On the following page, fill out the application form and then at the bottom click the green 'register' button (fig.14).

Figure 14: End of Copernicus registration page with register button highlighted in red.

Once registered, you will receive an email asking to verify your address. You can then log-in with your email and chosen password.

For any registration problems, email: help-login@dataspace.copernicus.eu

2.7 Running the tools in Jupyter-lab

Now that (almost) everything has been setup you can launch Jupyter Lab in the Anaconda Navigator (fig.15). Remember as always that your project environment (here 'EGM722') should be selected and not 'base (root)'.

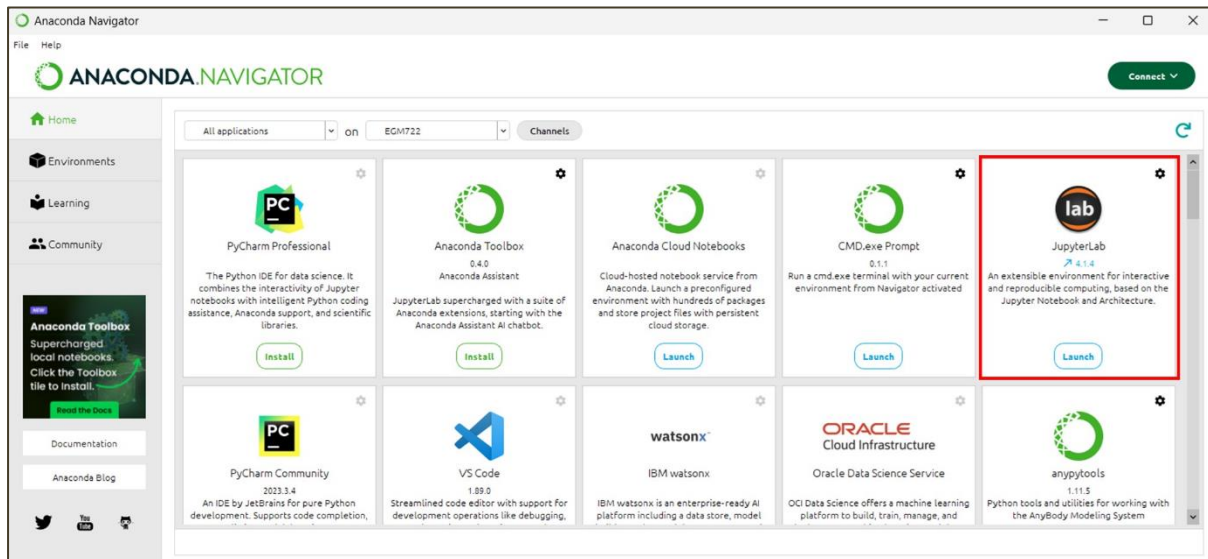


Figure 15: Location of Jupyter Lab in Anaconda Navigator highlighted in red.

Once Jupyter lab opens, you should see the three tools on the left (fig.16).

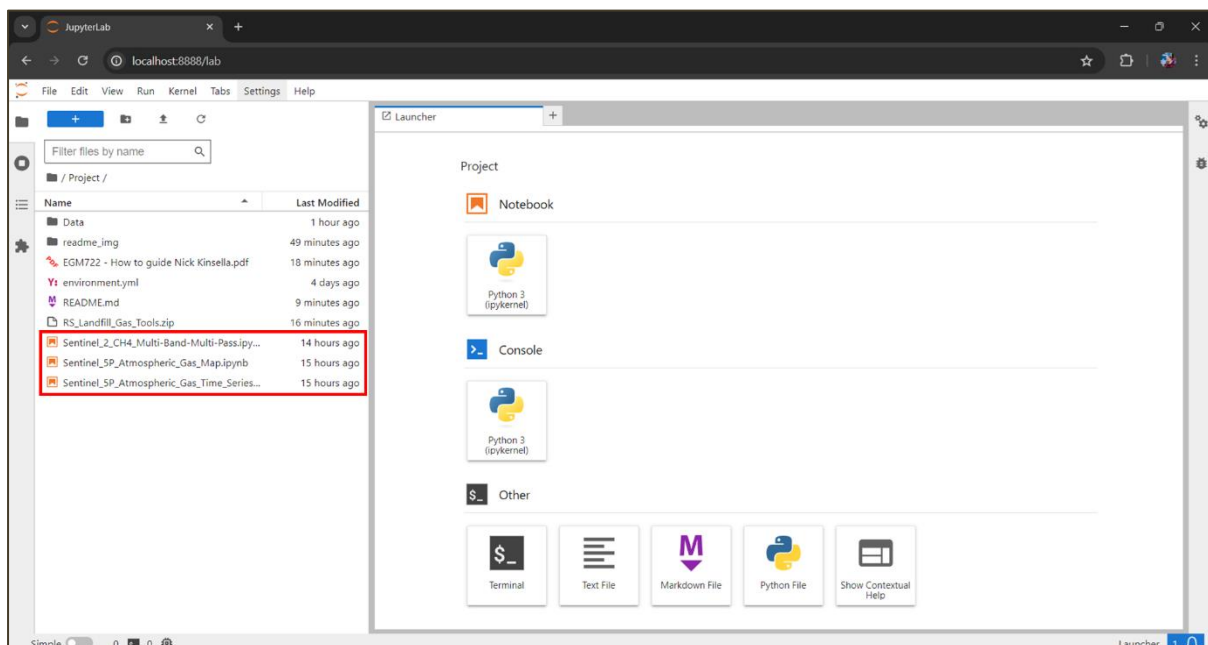


Figure 16: Location of tool scripts in Jupyter lab highlighted in red.

Click on one of the tools to open it. You can then follow the instructions, running the code by clicking the play button (fig.17).

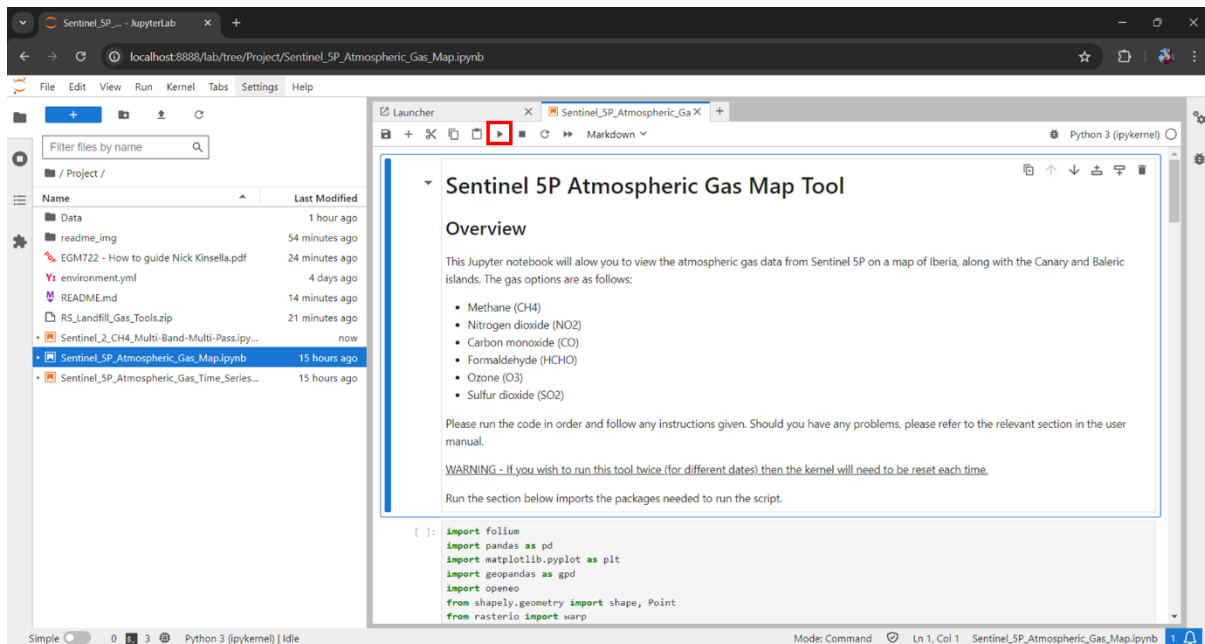


Figure 17: Location of the ‘play’ button which runs each of the code segments of the workbooks.

2.8 Authentication with OpenEO

The very first time one of the tools are run, the following section of code...

```
connection = openeo.connect(url="openeo.dataspace.copernicus.eu")
connection.authenticate_oidc()
```

... will provide you with a URL that will look something like this:

Visit https://auth.example.com/device?user_code=EAXD-RQXV to authenticate.

Copy this into your web browser and login using the Copernicus Data Space. Once this is complete, run tool’s Python script again and it will receive an authentication token, printing the message:

Authorized successfully.

In future you may be prompted with a new URL to create a new authentication token, whereby you should repeat the steps of this section.

3. Data and Dependencies

The datasets used by these tools are outlined in table 1.

Table 1: Datasets used by tools

Description	Data	Source	For use in tool(s)
PreZero Landfill Locations	Vector, point	PreZero International.	S5-AGT, S5-AGM
PreZero Landfill Bounding	Long/Lat .csv file, point	PreZero Landfill Locations pre-processed.	S2-MBMP
Sentinel 5P (TROPOMI) total columns of CO, HCHO, NO ₂ , O ₃ , SO ₂ and CH ₄	Raster 4,518m x 3,552m	Copernicus Dataspace – OpenEO.	S5-AGT, S5-AGM
Sentinel 2 (MSI) bands 2, 3, 4 and 12	Raster 20m ²	Copernicus Dataspace – OpenEO.	S2-MBMP

The dependencies contained in the environment.yml file are outlined in table 2.

Table 2: dependencies required to use the tools.

Name	Description	For use in tool(s)
Python	Programming language to run the code	S5-AGTS, S5-AGM & S2-MBMP
Folium	Generates interactive maps with Leaflet.js.	S5-AGTS & S5-AGM
Pandas	Data manipulation and analysis.	S5-AGTS, S5-AGM & S2-MBMP
Geopandas	Geospatial data manipulation and analysis.	S5-AGTS & S5-AGM
Matplotlib	Creates map visualizations	S5-AGTS, S5-AGM & S2-MBMP
Rasterio	Reads and edits raster datasets.	S5-AGM & S2-MBMP
Numpy	Performs numerical computations on arrays	S5-AGTS, S5-AGM & S2-MBMP
Requests	Used fetch data from web services or APIs.	S2-MBMP
Scikit-learn	Linear regression for brightness correction	S2-MBMP
OpenEO	API for Earth observation data processing	S5-AGT, S5-AGM & S2-MBMP

4. Methodology

4.1 Sentinel-5 Atmospheric Gas Timeseries:

This tool creates a timeseries graph of gas concentrations for selected landfills, identifying high concentration days using average concentration in a 10km buffer. It is based on a Python notebook from Copernicus OpenEO (2024), modified to use Sentinel 5 data and PreZero's landfill locations. Despite Sentinel 5's low spatial resolution, it provides daily measurements of several gases, useful for studying transient atmospheric phenomena like methane emissions (Varon et al., 2021). The processing steps are presented in Figure 18 and the code thereafter.

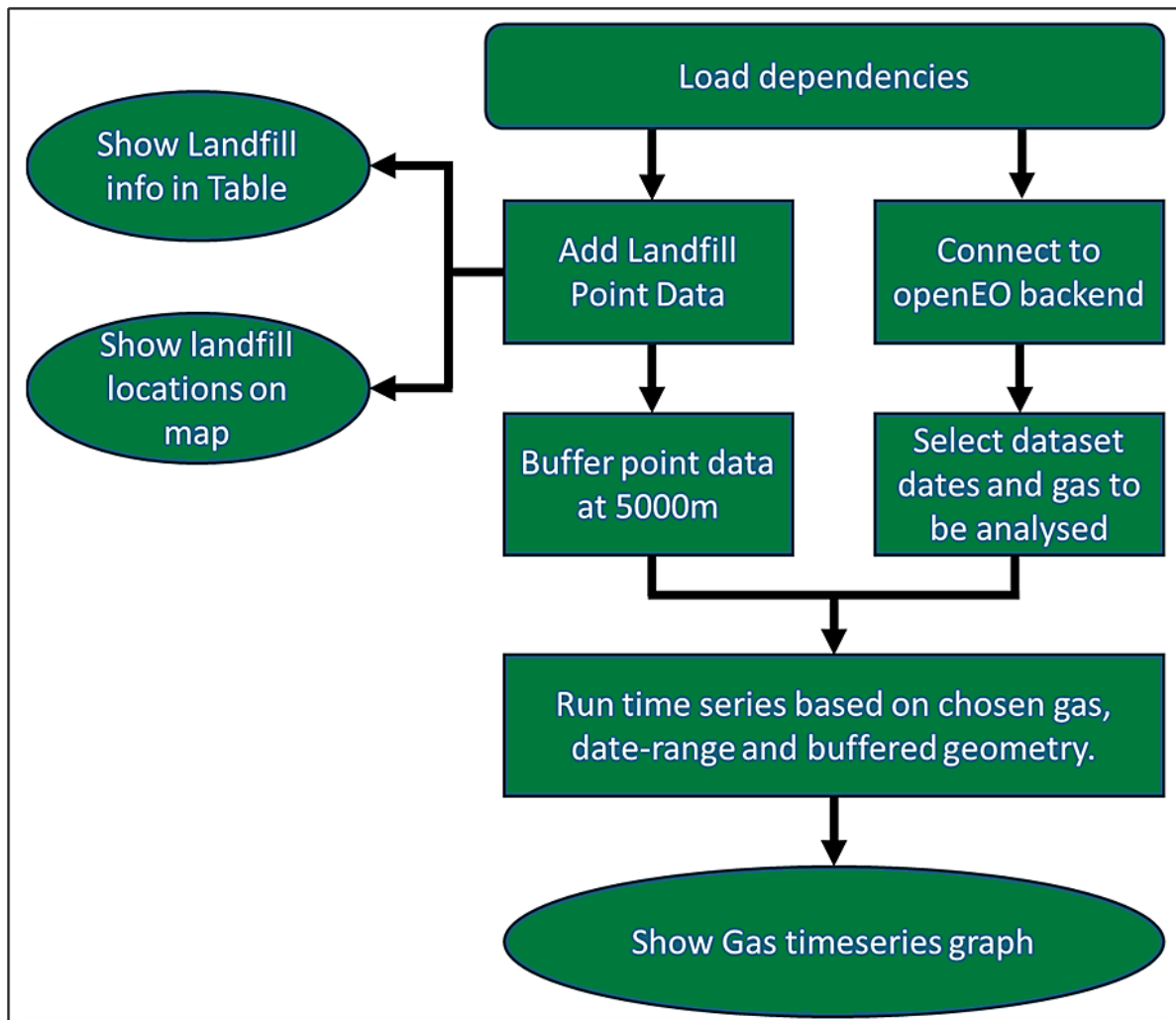


Figure 18: Flowchart of processes used for Gas timeseries analysis.

Code 1 loads the dependencies for the tool to run.

```

import folium
import pandas as pd
import matplotlib.pyplot as plt
import scipy.signal
import numpy as np
import geopandas as gpd
import openeo
from shapely.geometry import shape
  
```

Code 1: Loading of dependencies for the code to run.

Code 2 connects to the OpenEO backend.

```
connection = openeo.connect(url="openeo.dataspace.copernicus.eu")
connection.authenticate_oidc()
```

Code 2: Connecting to OpenEO

Code 3 uses geopandas to read a .geojson file that contains point vector data of the landfill location and then displays it for the end user to see.

```
landfills =
gpd.read_file(r"C:\GIS_Course\EGM722\Project\Data\PZ_landfill_point4326.geoj
son")
landfills
```

Code 3: Displaying the contents of the landfill file for easy reference.

Code 4 shows the locations of the landfills on a map using folium. The user can click on a map pin to view the landfill's address. The map is centred on the loaded geometries by calculating the average x and y location of all the datapoints.

```
# This creates the map and centres it on the dataset's centre point.
centroids = landfills.geometry.centroid
centre = [centroids.y.mean(), centroids.x.mean()]
site_map = folium.Map(location=centre, zoom_start=5)

# Adding the landfill locations to the map
for feature in landfills.iterfeatures():
    # Extract feature number from properties
    feature_number = feature['properties']['Landfill']

    # Extract coordinates of the feature
    coordinates = feature['geometry']['coordinates']

    # Create a marker with label for each feature
    folium.Marker(location=[coordinates[1], coordinates[0]],
                  popup=f"Feature {feature_number}").add_to(site_map)

# Display the map
site_map
```

Code 4: Code for displaying the landfill points on a map for easy reference.

Code 5 adds a buffer of 10,000 metres to the landfill point data, and then formats it as a .geojson, as the OpenEO API requires it in that format.

```
# loading dataframe
landfill_10000m = landfills

# The dataset is projected in EPSG:4326 with its units in degrees. This
needs to be converted to CRS to EPSG:2062, which is in metres.
landfill_10000m = landfill_10000m.to_crs(epsg=2062)

# Now the dataframe is converted, a buffer of 5000m is added to each point
landfill_10000m['geometry'] = landfill_10000m.buffer(10000)
```

```
# Now the buffered data needs to be converted back to EPSG:4326 because the
Sentinel data is projected in EPSG:4326.
landfill_10000m = landfill_10000m.to_crs(eps=4326)

# The time series analysis requires that A GeoJSON format file is used for
the analysed areas, so this bit produces a file suitable for that.
landfill_10000m_geojson = landfill_10000m.__geo_interface__
```

Code 5: Code for adding buffers to landfill point data

Code 6 selects the specific dataset for the time series analysis, including the date range and gas type.

```
s5cube_timeseries = connection.load_collection(
    "SENTINEL_5P_L2",
    temporal_extent=["2021-08-01", "2021-10-31"], # format YYYY-MM-DD
    bands=["CH4"], # Gas options 'CO', 'HCHO', 'NO2', 'O3', 'SO2', 'CH4'
)
```

Code 6: Selecting date for time series and gas to be monitored.

Code 7 runs the time series analysis and saves the results as a .csv file. This process can take several minutes depending on how much time has been selected in the temporal extent.

```
timeseries =
s5cube_timeseries.aggregate_spatial(geometries=landfill_5000m_geojson,
reducer="mean")

#This saves the results as a .CSV file which can be viewed in Microsoft
Excel or a similar package. It will be saved in the indicated location.
job = timeseries.execute_batch(out_format="CSV", title="Gas timeseries")

job.get_results().download_file("Gas_Timeseries_results/Gas_timeseries.csv")

pd.read_csv("Gas_Timeseries_results/Gas_timeseries.csv", index_col=0)
```

Code 7: Running the data collection for the time series.

Code 8 plots the data on a timeseries graph using much of the script taken from the NDVI OpenEO notebook (OpenEO, 2024). Modifications include scaling the value of the Y axis to whatever data is loaded and allowing the user to select which landfills they want to view on the graph.


```

def plot_timeseries(filename, selected_landfill_ids=None, figsize=(15, 10)):
    df = pd.read_csv(filename, index_col=0)
    df.index = pd.to_datetime(df.index)
    df = df.sort_index()

    fig, ax = plt.subplots(figsize=figsize, dpi=90)

    if selected_landfill_ids:
        df_selected = df[df['feature_index'].isin(selected_landfill_ids)]
        for landfill_id, group in df_selected.groupby("feature_index"):
            group["avg(band_0)"].plot(marker="o", label=f"Landfill
{landfill_id}", ax=ax)
    else:
        df.groupby("feature_index")["avg(band_0)"].plot(marker="o", ax=ax)

    ax.set_title(filename.split("/")[-1])
    ax.set_ylabel("Parts per billion for CH4 or mol/m2 for all other gases")

    ymin = df["avg(band_0)"].min()
    ymax = df["avg(band_0)"].max()
    ymin_with_margin = ymin - 0.1 * (ymax - ymin)
    ymax_with_margin = ymax + 0.1 * (ymax - ymin)
    ax.set_ylim(ymin_with_margin, ymax_with_margin)

    ax.legend(title="Landfill id", loc='upper left', bbox_to_anchor=(1.02,
1), ncol=2)
    ax.xaxis.set_major_locator(plt.MaxNLocator(30))
    ax.grid(True)

# Change the selected landfill ids for the ones you want to view.
plot_timeseries("Gas_Timeseries_results/Gas_timeseries.csv",
    selected_landfill_ids=[13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23])

```

Code 8: For plotting the time series data on a graph

4.2 Sentinel-5 Atmospheric Gas Concentration Map:

The tool makes a map of atmospheric gas concentrations over the Iberian peninsula, the Balearic and Canary islands, with marked locations for where PreZero has its landfills. This allows the user to continue their analysis based on the results of the time series. As a high concentration signal could be generally elevated levels in the atmosphere, or it could be a landfill emission. Figure 19 provides an overview of the main steps in processing.

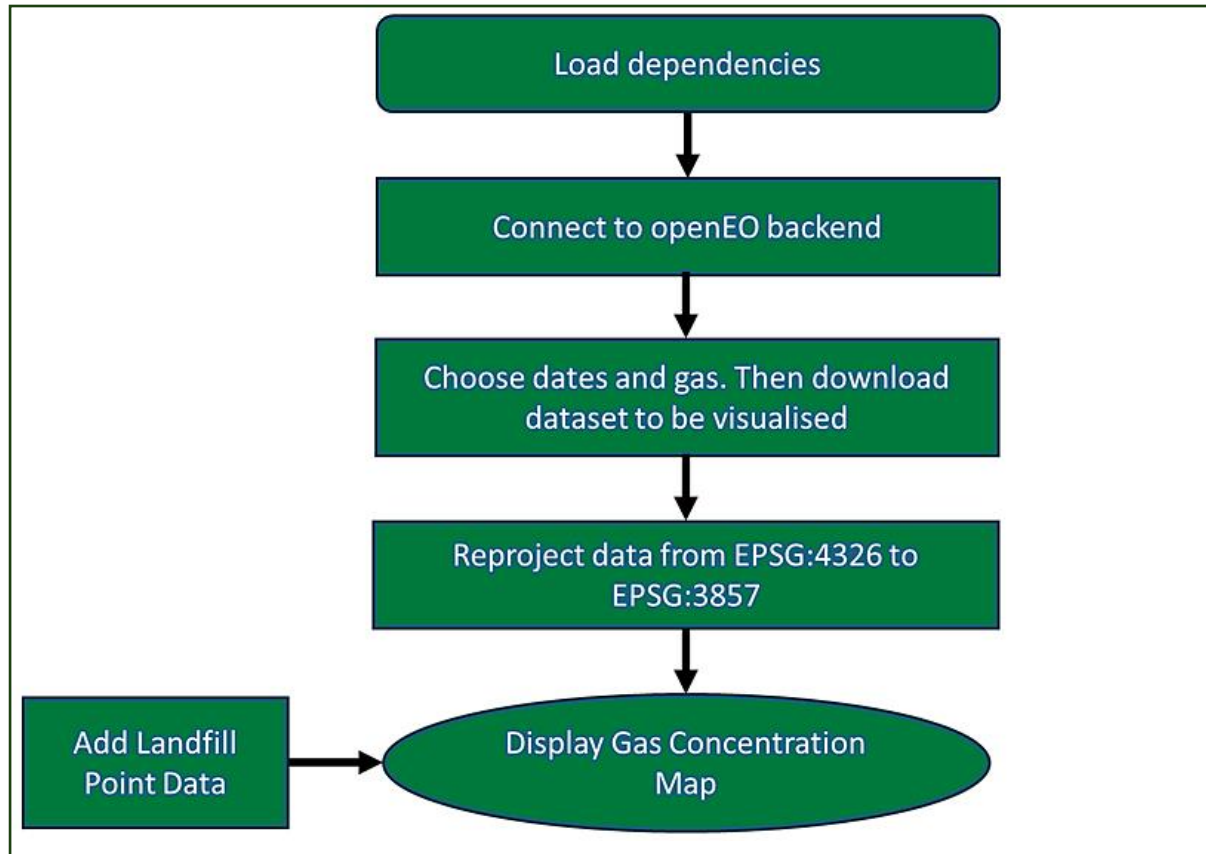


Figure 19: Flowchart of processes used for Gas Concentration Map

Code 9 shows the dependencies that are loaded.

```

import folium
import pandas as pd
import matplotlib.pyplot as plt
import geopandas as gpd
import openeo
from shapely.geometry import shape, Point
from rasterio import warp
from matplotlib import cm
from matplotlib.colors import Normalize
import folium.raster_layers
import rasterio
from rasterio import warp
import numpy as np
  
```

Code 9: Loading of dependencies for the code to run.

Code 10 connects to the OpenEO backend.

```
connection = openeo.connect(url="openeo.dataspace.copernicus.eu")
connection.authenticate_oidc()
```

Code 10: Connecting to OpenEO

Code 11 selects the specific date to be viewed and what gas is to be monitored. It then downloads it as a .GTiff file.

```
cube = connection.load_collection(
    collection_id="SENTINEL_5P_L2",
    temporal_extent=["2023-06-01", "2023-06-01"], # format YYYY-MM-DD. Only
    one date should be selected so the to and from fields should be identical.
    spatial_extent={"west": -19.5, "south": 27.0, "east": 5.0, "north":
    44.5},
    bands=["CH4"], Gas monitoring options: 'CO', 'HCHO', 'NO2', 'O3', 'SO2',
    'CH4'
)
cube.download("Sentinel-5P_Spain.GTiff")
```

Code 11: Code for selecting dates and gas, then downloading the data.

Code 12 takes the Sentinel-5P_Spain.GTiff file and reprojects its CRS from EPSG:4326 to EPSG:3857, the projection of the folium map.

```
dst_crs = 'EPSG:3857'
# Open the gas data file that is in EPSG:4326 and calculate its bounds
with rasterio.open('Sentinel-5P_Spain.GTiff') as src:
    transform, width, height = rasterio.warp.calculate_default_transform(
        src.crs, dst_crs, src.width, src.height, *src.bounds)
    # Copy and update the metadata from the source dataset
    kwargs = src.meta.copy()
    kwargs.update({
        'crs': dst_crs,
        'transform': transform,
        'width': width,
        'height': height
    })
    # Create a new gas data file in EPSG:3857
    with rasterio.open('Sentinel-5P_Spain3857.GTiff', 'w', **kwargs) as dst:
        #if you wish to run this without restarting the kernel, you will need to
        rename this
        # Loop through each band in the source dataset
        for ind in range(1, src.count + 1):
            # Reproject each band and write it to the destination dataset
            rasterio.warp.reproject(
                source=rasterio.band(src, ind),
                destination=rasterio.band(dst, ind),
                src_transform=src.transform,
                src_crs=src.crs,
                dst_transform=transform,
                dst_crs=dst_crs,
```

Code 12: Code for reprojecting raster

Code 13 loads the folium map. The pins are clickable for the landfill information and the map is centred on those features. Because the CH₄ dataset is not a continuous raster, a piece of code ignoring those values has been included. The colourmap is then set using the matplotlib library. The raster bounds are set using a west, south, east, north longitude and latitude. Finally the legend bar is set to scale to whatever atmospheric gas dataset is shown on the map.

```
# This section loads the map

#This is the reprojected gas data
gas_data = r'C:\GIS_Course\EGM722\Project\Sentinel-5P_Spain3857.GTiff'

# Open raster file, load values and prepare them to be displayed.
dataset = rasterio.open(gas_data, 'r')
rasdata = dataset.read()[0]
rasdata_normed = rasdata / rasdata.max() * 10

# When the data is displayed, this says to ignore values of zero.
non_zero_values = rasdata[rasdata != 0]
min_value = non_zero_values.min()
max_value = non_zero_values.max()
normalized_data = (non_zero_values - min_value) / (max_value - min_value)

# Create a colourmap for the non-zero values
colourmap = cm.turbo
colourmap_index = np.zeros_like(rasdata, dtype=np.float64)
colourmap_index[rasdata != 0] = normalized_data

#Loading in the landfill locations
PZ_landfill_Locations =
gpd.read_file(r"C:\GIS_Course\EGM722\Project\Data\PZ_landfill_point4326.geojson")

# This creates the map and centres it on the geometries.
centroids = PZ_landfill_Locations.geometry.centroid
center = [centroids.y.mean(), centroids.x.mean()]
gas_concentration_map = folium.Map(location=center, zoom_start=5,
tiles='CartoDB Positron')

# Adding the PreZero landfill locations to the map and making them clickable
for info
for feature in PZ_landfill_Locations.iterfeatures():
    # Extract feature number from properties
    feature_number = feature['properties']['Landfill']
    # Extract coordinates of the feature
    coordinates = feature['geometry']['coordinates']
    # Create a marker with label for each feature
    folium.Marker(location=[coordinates[1], coordinates[0]],
                    popup=f"Feature
{feature_number}").add_to(gas_concentration_map)
```



```
# Adding the gas concentration dataset to the map
folium.raster_layers.ImageOverlay(
    image=colourmap(colourmap_index),
    name='gas concentration in atmosphere',
    opacity=0.6,
    bounds=[[27.0, -19.5], [44.4, 5.0]], # this should be the same as the
    spatial extent of cube
    interactive=False,
    cross_origin=False,
    zindex=1
).add_to(gas_concentration_map)

# Creating the legend for gas concentration
fig, ax = plt.subplots(figsize=(8, 0.2))
cbar = plt.colorbar(cm.ScalarMappable(norm=Normalize(vmin=min_value,
vmax=max_value),
                                cmap=colourmap),
                    cax=ax, orientation='horizontal')
cbar.set_label('Parts per billion for CH4 or mol/m2 for all other gasses')

# Display the map
gas_concentration_map
```

Code 13: Code for configuring and loading Gas Concentration Map

4.3 Sentinel-2 Multi-Band-Multi-Pass CH₄ Map

Sentinel 5's resolution is very coarse, so to help users locate emission plumes relative to a landfill, Sentinel 2's MSI instrument with a 20m² spatial resolution and return frequency of 3-5 days was employed.

Varon et al. (2021) showed that methane plumes from point sources could be imaged by differencing Sentinel-2's SWIR-1 and SWIR-2 bands. The tool runs an analysis using a similar multi-band-multi-pass retrieval method:

First it calculates a multi-band-single-pass calculation for both active emission and no emission dates, resulting in two datasets which are then used together for a multi-band-multi-pass method.

The multi-band-single-pass equation is as follows:

$$MBSP = cB12 - B11$$

Where:

- B12 is the Sentinel-2 SWIR-2 band.
- B11 is the Sentinel-2 SWIR-1 band.
- c is calculated using the difference in the median of B12 to B11 and then adding the difference to B12.

Once active emission and no emission scenes have been calculated, the following equation is used to calculate the multi-band-multi-pass raster.

$$MBMP = ActiveMBSP - NoMBSP$$

Where:

- ActiveMBSP is the multiband single pass for the active emission scene
- NoMBSP is the multiband single pass for the no emission scene.

Figure 20 provides an overview of the main steps in processing.

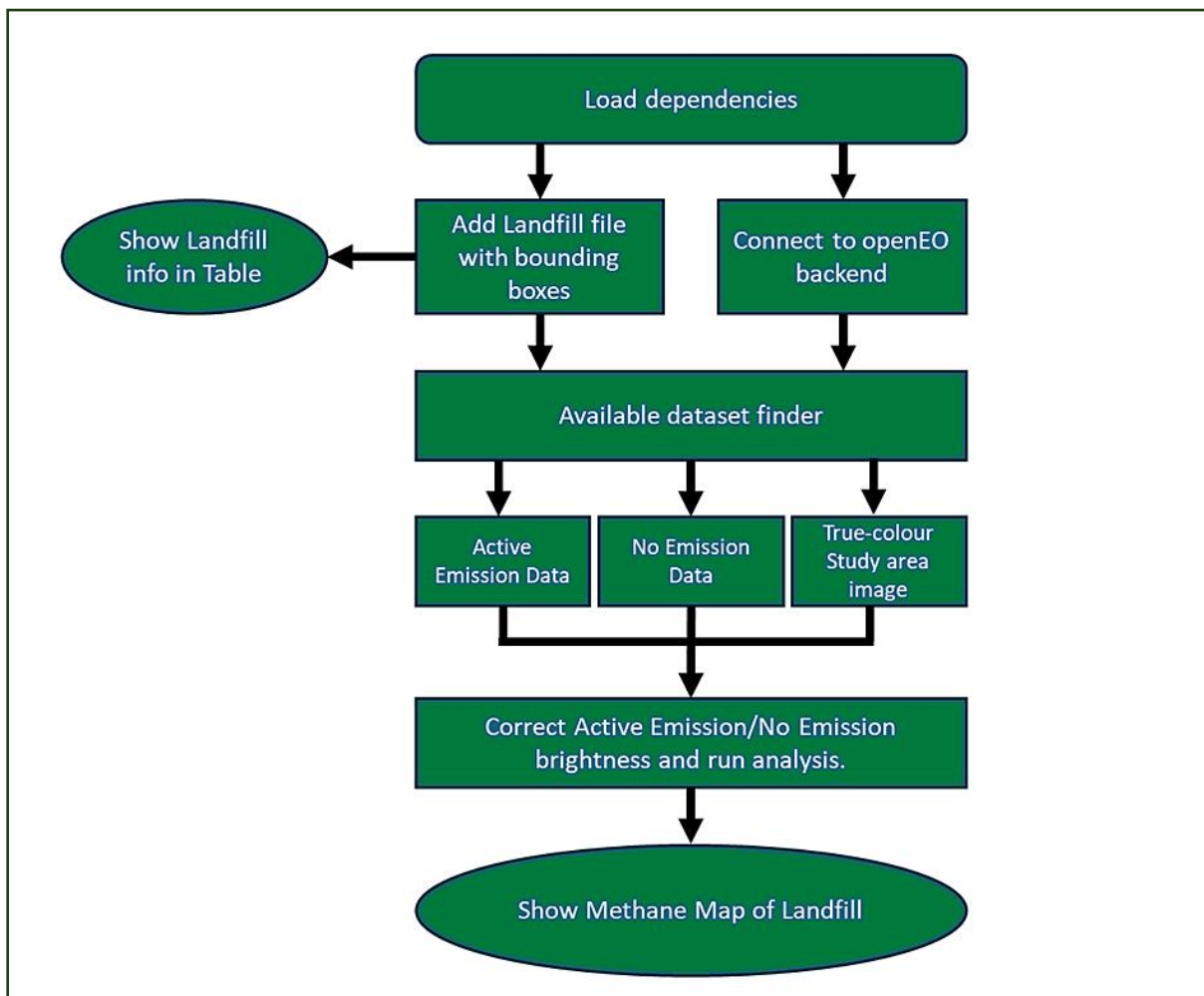


Figure 20: Flowchart of processes used for Sentinel 2 Landfill CH₄ Map

Code 14 shows the dependencies that are loaded.

```
import folium
import pandas as pd
import matplotlib.pyplot as plt
import openeo
import rasterio
import numpy as np
import requests
from sklearn.linear_model import LinearRegression
```

Code 14: Loading of dependencies for the code to run.

Code 15 connects to the OpenEO backend.

```
connection = openeo.connect(url="openeo.dataspace.copernicus.eu")
connection.authenticate_oidc()
```

Code 15: Connecting to OpenEO

Code 16 reads a .geojson file that contains the landfill location bounding boxes, and then displays it for the end user to see. There are options for small and large bounding boxes.

```
landfill_csv =
pd.read_csv(r'C:\GIS_Course\EGM722\Project\Data\PreZero_Landfill_Bounding_small.csv')
```

```
# landfill_csv =
pd.read_csv(r'C:\GIS_Course\EGM722\Project\Data\PreZero_Landfill_Bounding_small.csv')
landfill_csv
```

Code 16: Displaying the contents of the landfill file for easy reference.

Code 17 provides the available datasets for a date range and landfill selected by the user. It also has a cloud filter set at 15% as excessive cloud can interfere with the tool.

```
def get_spatial_extent(landfill_id):
    landfill = landfill_csv[landfill_csv['id'] == landfill_id].iloc[0]
    return {
        "west": landfill['west'],
        "south": landfill['south'],
        "east": landfill['east'],
        "north": landfill['north']
    }

def fetch_available_dates(landfill_id, temporal_extent):
    spatial_extent = get_spatial_extent(landfill_id)
    catalog_url =
f"https://catalogue.dataspace.copernicus.eu/resto/api/collections/Sentinel2/search.json?box={spatial_extent['west']}%2C{spatial_extent['south']}%2C{spatial_extent['east']}%2C{spatial_extent['north']}&sortParam=startDate&sortOrder=ascending&page=1&maxRecords=1000&status=ONLINE&dataset=ESA-DATASET&productType=L2A&startDate={temporal_extent[0]}T00%3A00%3A00Z&completionDate={temporal_extent[1]}T00%3A00%3A00Z&cloudCover=%5B0%2C{cloud_cover}%5D"
    response = requests.get(catalog_url)
    response.raise_for_status()
    catalog = response.json()
    dates = [date.split('T')[0] for date in map(lambda x:
x['properties']['startDate'], catalog['features'])]
    return dates

# Please enter your parameters here.
landfill_id = 24 # Specify the landfill ID.
temporal_extent = ["2019-10-01", "2019-11-30"] # Specify the the date range
you want to check for available data.
cloud_cover = 15

available_dates = fetch_available_dates(landfill_id, temporal_extent)
print("Available dates:", available_dates)
```

Code 17: Code for determining available dates for analysis.

Code 18 is for designating the active emission dataset. Here the user will select the day that the emission was seen with the S5-AGT tool, along with the appropriate landfill id.

```
def active_emission(landfill_id, temporal_extent):
    landfill = landfill_csv[landfill_csv['id'] == landfill_id].iloc[0]
    active_emission = connection.load_collection(
```



```

        "SENTINEL2_L2A",
        temporal_extent=temporal_extent,
        spatial_extent={
            "west": landfill['west'],
            "south": landfill['south'],
            "east": landfill['east'],
            "north": landfill['north']
        },
        bands=["B11", "B12"],
    )
    active_emission.download("Sentinel-2_active_emissionMBMP.GTiff")

# Enter parameters for the active emission day
landfill_id = 24 # Specify the landfill ID
temporal_extent = ["2019-11-20", "2019-11-20"]

active_emission(landfill_id, temporal_extent)

```

Code 18: Code for downloading active emission dataset.

Like the code before it, Code 19 is for designating the no emission dataset that will be used to compare to the active emission dataset.

```

def no_emission(landfill_id, temporal_extent):
    landfill = landfill_csv[landfill_csv['id'] == landfill_id].iloc[0]
    no_emission_collection = connection.load_collection(
        "SENTINEL2_L2A",
        temporal_extent=temporal_extent,
        spatial_extent={
            "west": landfill['west'],
            "south": landfill['south'],
            "east": landfill['east'],
            "north": landfill['north']
        },
        bands=["B11", "B12"],
    )
    no_emission_collection.download("Sentinel-2_no_emissionMBMP.GTiff")

# Enter parameters for the no emission day
landfill_id = 24 # Specify the landfill ID
temporal_extent = ["2019-10-06", "2019-10-06"]

no_emission(landfill_id, temporal_extent)

```

Code 19: Code for downloading no emission dataset.

Finally, as with the two codes before, Code 20 downloads a true colour satellite image to aid in the visualisation of the data.

```

def truecolour_image(landfill_id, temporal_extent):
    landfill = landfill_csv[landfill_csv['id'] == landfill_id].iloc[0]
    truecolour_image_collection = connection.load_collection(
        "SENTINEL2_L2A",

```

```

        temporal_extent=temporal_extent,
        spatial_extent={
            "west": landfill['west'],
            "south": landfill['south'],
            "east": landfill['east'],
            "north": landfill['north']
        },
        bands=["B02", "B03", "B04"],
    )
    truecolour_image_collection.download("Sentinel-2_truecolour.GTiff")

# Enter parameters for the no emission day
landfill_id = 24 # Specify the landfill ID
temporal_extent = ["2019-10-06", "2019-10-06"]

truecolour_image(landfill_id, temporal_extent)

```

Code 20: Code for downloading true colour satellite image for data visualisation

The code 21 runs the analysis as outlined at the beginning of this section (4.3).

```

# Define file paths
Active_Multiband = "Sentinel-2_active_emissionMBMP.GTiff"
No_Multiband = "Sentinel-2_no_emissionMBMP.GTiff"

# Define each band from both datasets
with rasterio.open(Active_Multiband) as Active_img,
rasterio.open(No_Multiband) as No_img:
    Active_B11 = Active_img.read(1)
    Active_B12 = Active_img.read(2)
    No_B11 = No_img.read(1)
    No_B12 = No_img.read(2)
    # Calculate the median difference for Active_B11 and Active_B12
    median_diff_active = np.median(Active_B11) - np.median(Active_B12)
    # Adjust Active_B12
    Corrected_Active_B12 = Active_B12 + median_diff_active
    # Calculate the median difference for No_B11 and No_B12
    median_diff_no = np.median(No_B11) - np.median(No_B12)
    # Adjust No_B12
    Corrected_No_B12 = No_B12 + median_diff_no
    # Calculate the fractional change
    frac_change = (Active_B11 - Corrected_Active_B12) - (No_B11 -
Corrected_No_B12)

```

Code 21: Code running analysis

Code 22 displays the map. It firstly takes the true colour satellite image bands, stacks them and applies a brightness factor to make the image clearer.

```

# Open the and load the truecolour satellite image file and define which
band is which.
fp = 'Sentinel-2_truecolour.GTiff'
img = rasterio.open(fp)

```

```
blue = img.read(1)
green = img.read(2)
red = img.read(3)

# Change this number up or down if the satellite background image is too
dark or bright.
brightness_factor = 0.05
blue = np.clip(blue * brightness_factor, 0, 255)
green = np.clip(green * brightness_factor, 0, 255)
red = np.clip(red * brightness_factor, 0, 255)

# Stack the blue, green and red bands to make a colour image.
rgb = np.dstack((red, green, blue))
rgb = rgb / rgb.max()

# Create a new figure
plt.figure(figsize=(10, 8))

# Display the RGB image
plt.imshow(rgb)

# Calculate mean and standard deviation of frac_change
mean = np.nanmean(frac_change)
std = np.nanstd(frac_change)

# Create a mask where values are not 1.5 SD above the mean
mask = frac_change < (mean + 1.5 * std)

# Apply the mask
masked_frac_change = np.ma.masked_array(frac_change, mask=mask)

vmax = np.nanmax(frac_change)
vmin = 0

# Display the masked_frac_change data on top of the RGB image
plt.imshow(masked_frac_change, cmap='plasma', alpha=1, vmin=vmin)

# Add a colorbar and labels
cbar = plt.colorbar(label='CH4 Levels', shrink=0.7)
cbar.set_ticks([vmin, vmax])
cbar.set_ticklabels(['Lower', 'Higher'])
plt.title('CH4 Concentration')

# Dataset resolution in metres
resolution = 20

# Get the dimensions of the image
height, width, _ = rgb.shape
```

```
# Create arrays representing the x and y coordinates in meters
x = np.arange(0, width * resolution, 1000)
y = np.arange(0, height * resolution, 1000)

# Set the x and y ticks and labels
plt.xticks(x / resolution, x)
plt.yticks(y / resolution, y)
plt.xlabel('X (meters)')
plt.ylabel('Y (meters)')

# Add grid lines
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Please enter the emission date where it says '[emission date here]' if you
# want to download more than one map.
plt.savefig(f'S2HRM[emission_date_here].jpg', format='jpg',
bbox_inches='tight')

# Show the plot
plt.show()
```

Code 22: Code displaying the map

5. Expected Results and Demo

This section will illustrate the functionality of the tools by showcasing documented methane plumes and their corresponding depiction in the results.

5.1 Sentinel-5 Atmospheric Gas Timeseries:

After running the S5-AGT tool a line graph will be displayed (fig.22 & fig.23). dots with a connecting line indicated consecutive days of methane emissions. The tool measures any atmospheric emissions over the landfill and so it isn't necessarily an indication of a methane plume from a landfill.

A methane plume occurred near the PreZero managed Canada Hermosa on the 23rd of April 2023 (Jet Propulsion Laboratory, 2024). The data indicates that the CH₄ emission may have occurred over 9 days from the 15th to the 24th of February 2023 (fig.22). However given the strict laws enforced on Spanish landfills (Castillo-Giménez et al., 2019) what is more likely is that this is a short duration plume happening during a more general pervasiveness of methane in the atmosphere.

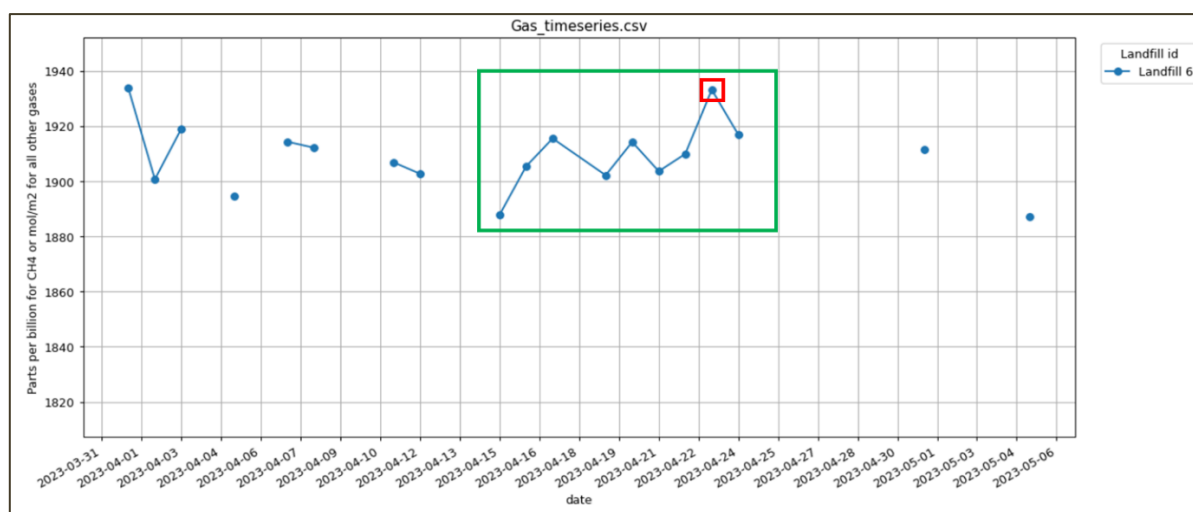


Figure 21: timeseries of landfill 6 'Cañada Hermosa', with the signal relating to the plume on the 23rd of April 2023 highlighted in red and sequential days of emissions highlighted in green.

Better candidates for plumes, which can often be short lived (Varon et al., 2021) are singular high emissions. Figure 23 shows a known emission from the Pinto biomethanization plant near Madrid (The Guardian, 2024) and 2 other isolated high values, which may warrant further investigation.

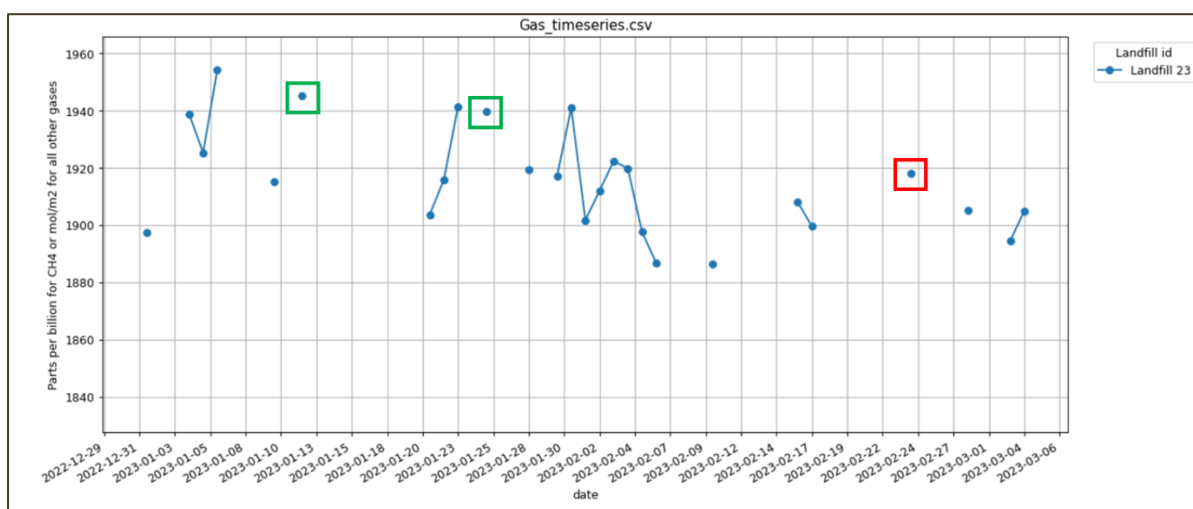


Figure 22: timeseries of landfill 23 'Pinto', with the signal relating to the 24th of February highlighted in red and two isolated high values highlighted in green.

5.2 Sentinel-5 Atmospheric Gas Concentration Map:

Once the S5-AGM tool has been run, it will display the atmospheric gas concentrations. Figure 23 and 24 below show the data for the 24th of February 2023 and the emission from that day can clearly be seen.

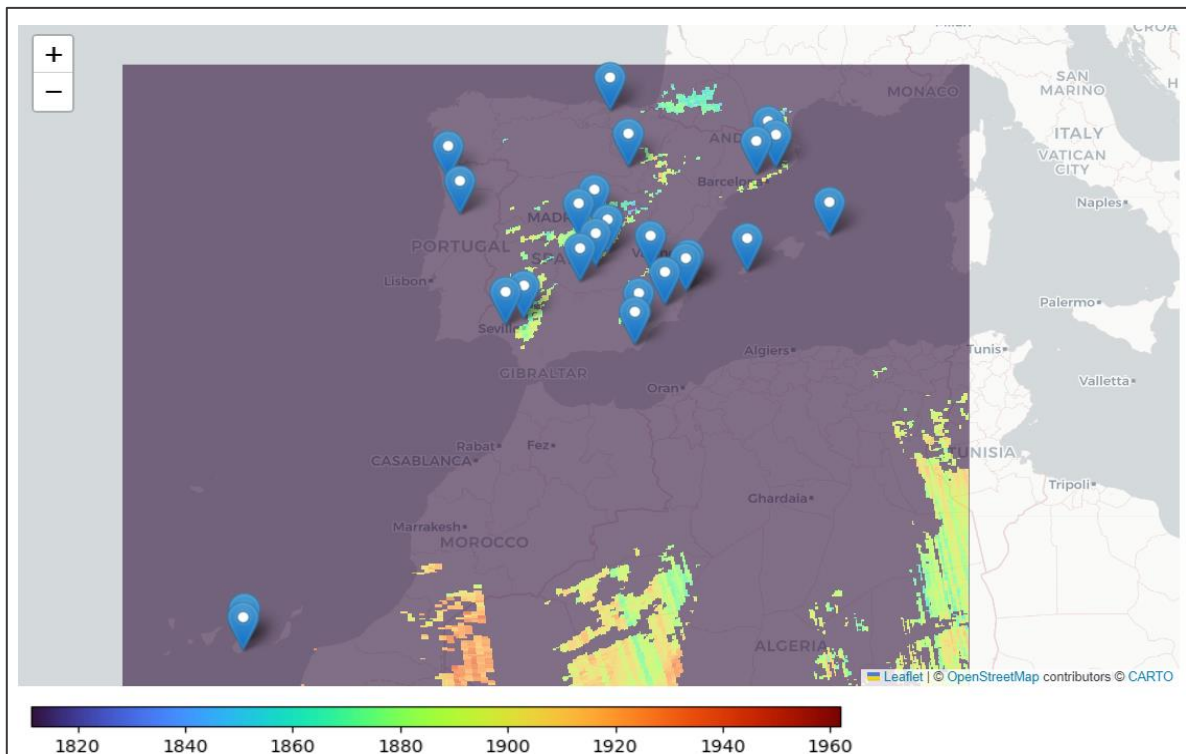


Figure 23: Screenshot of leaflet CH4 concentrations map, over Spain with landfill locations shown as blue pins.

The user can then zoom in to an area of interest. In figure 24 the Pinto Waste Site is shown along with a distinct hotspot of dark orange, characteristic of a CH₄ plume (European Space Agency, 2021).

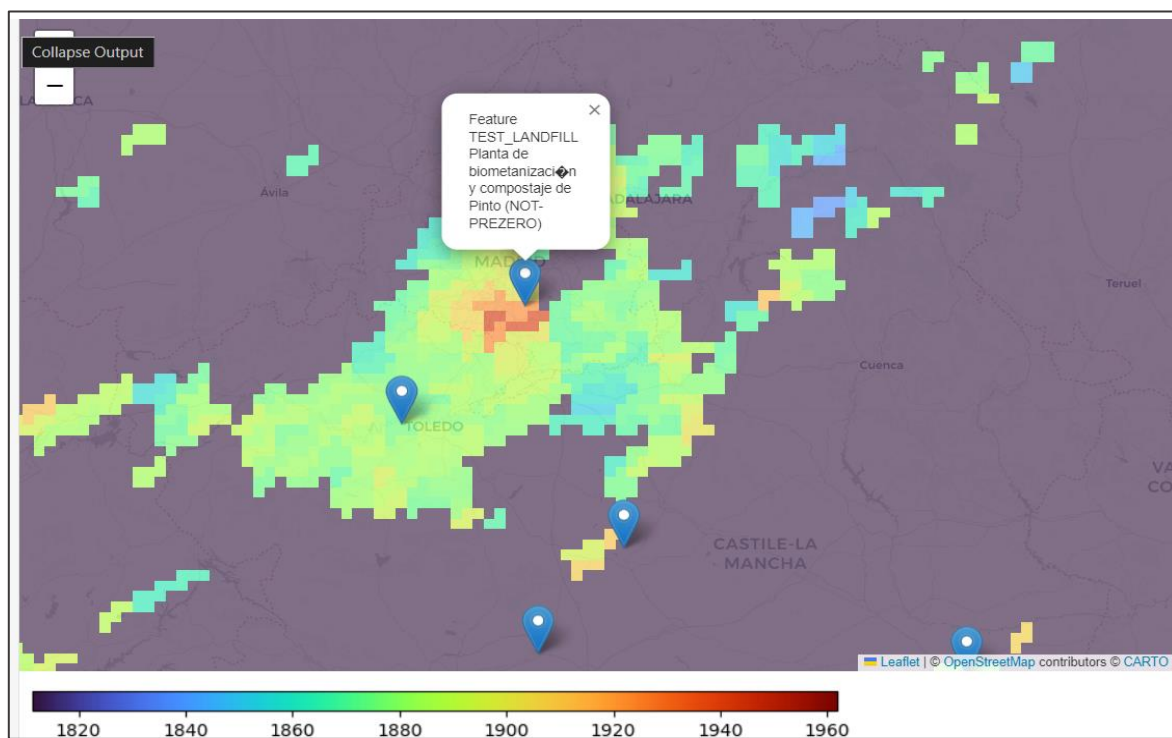


Figure 24: Close-up of Sentinel 5P data showing methane plume from

5.3 Sentinel-2 Multi-Band-Multi-Pass CH₄ Map:

The emission date of 24th Feb 2023 for the Pinto biomethanization plant, wasn't available due to Sentinel 2's return period (Sentinel Hub, 2024). However, if a satellite pass is lucky and catches a methane plume in the act, it provides near definitive information about a plume's origin.

Figure 25 below shows a methane plume from an Algerian oil and gas field in late 2019.

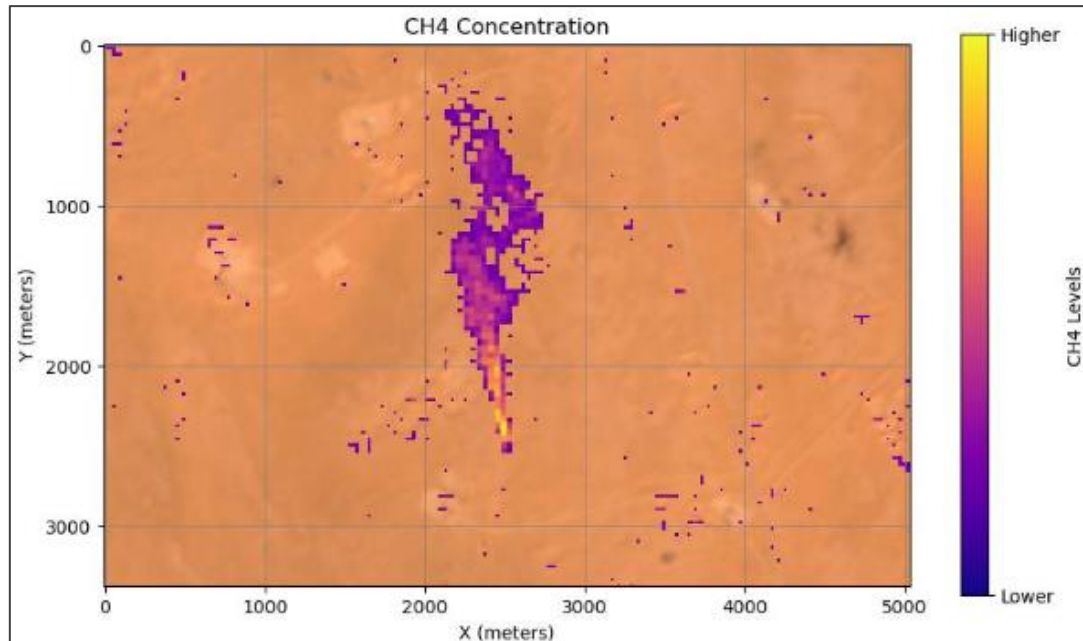


Figure 25: Methane Emission Plume in Algeria on 20th of November 2019 at 31.65849758813171, 5.905299672029146

A search on Google earth reveals this to be a emanating from the end of some kind of pipe, coming out of one of the facilities (fig.26).

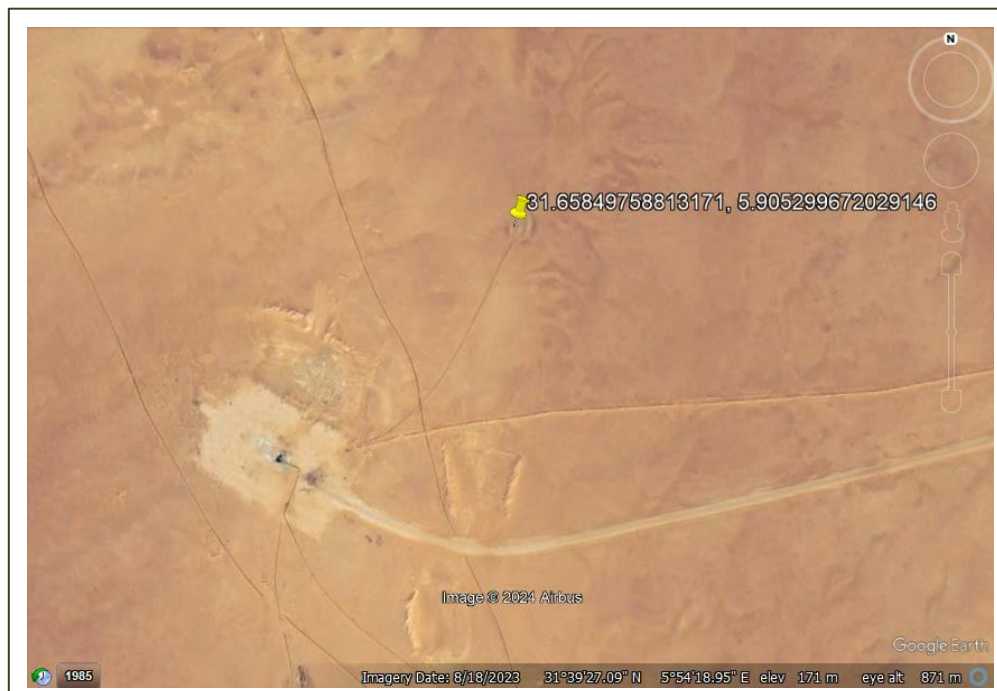


Figure 26: Satellite imagery from Google Earth showing a petrochemical facility with the location of the point source indicated by the yellow pin at the end of what appears to be a pipe. (Google Earth, 2024)

6. Troubleshooting

Input errors have been covered in the accompanying Jupyter notebook, however there are two errors unrelated to a user error which can cause problems. These are detailed below.

6.1 Remote disconnected error (All tools)

Error: Remote disconnected

This can occur when there are issues with the Copernicus network. In the event that you see an error like this you can check page <https://dataspace.copernicus.eu/news> for any downtime messages and you can also contact the Copernicus dataspace team via the form at <https://helpcenter.dataspace.copernicus.eu/hc/en-gb/requests/new>

6.2 Concurrent job error (S5-AGM)

OpenEoApiError: [400] ConcurrentJobLimit: Job was not started because concurrent job limit (2) is reached. (ref: r-240413b5d1b240118da9f9ed90807c58)

This can happen when the tool is run, cancelled and then run again. If this happens the process is still running in the background and needs to be cancelled.

To do this go to the following URL: <https://openeo.dataspace.copernicus.eu/>

You will be presented with the following screen (fig.29). Please click the highlighted login button.

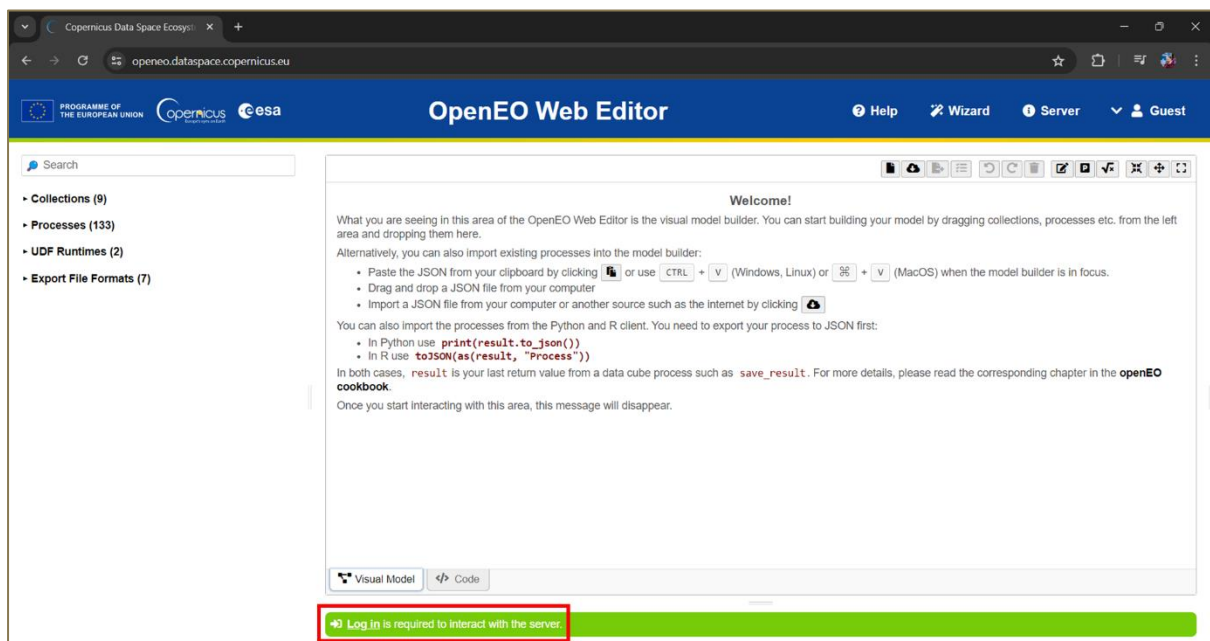


Figure 27: OpenEO Web Editor with login button highlighted in red

You will then see the following screen (fig.30).

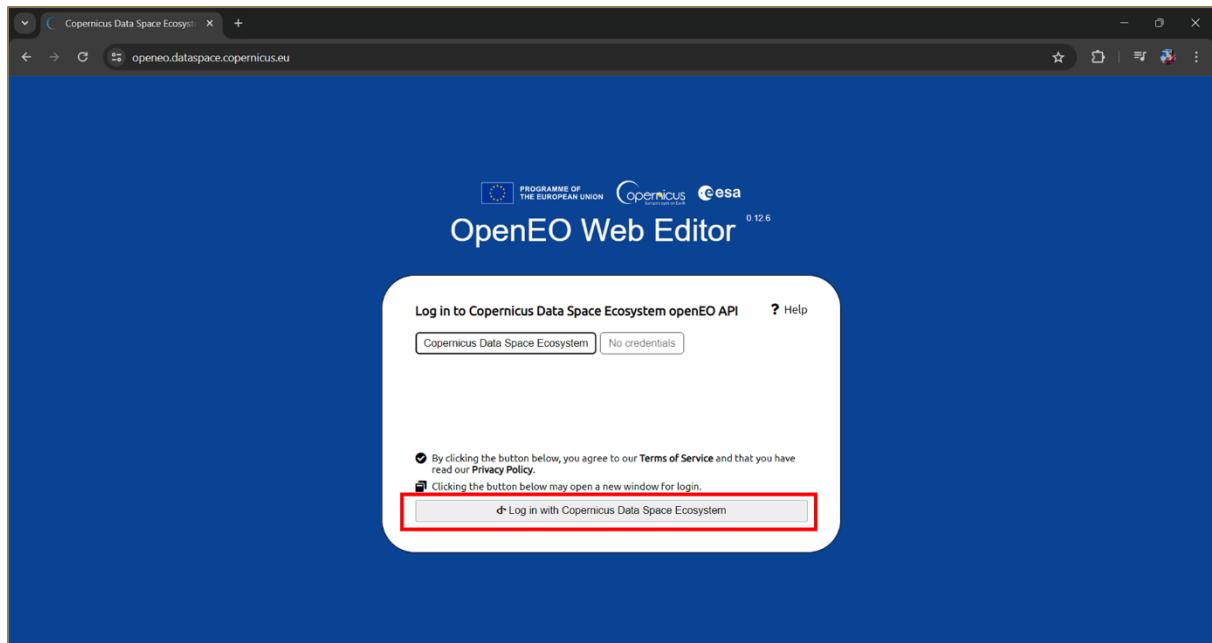


Figure 28: OpenEO Web editor login prompt with login button highlighted in red.

Simply click the highlighted button (fig.31) and follow the process. You should then be returned to the OpenEO web editor but now you will see a list of processes including the active ones.

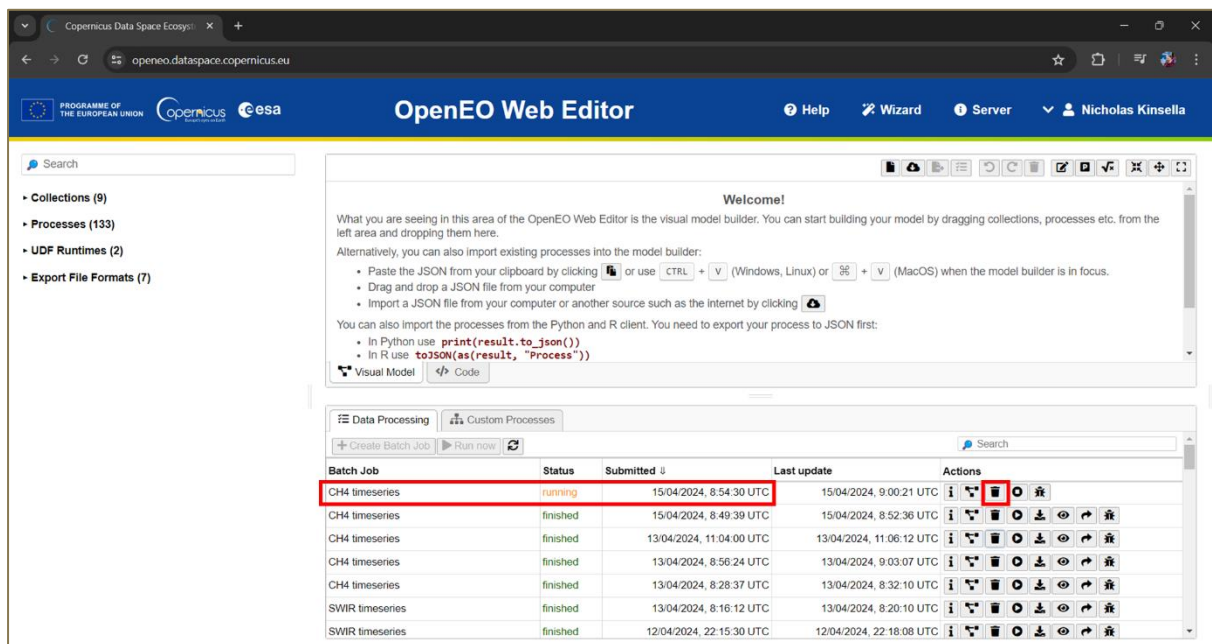


Figure 29: OpenEO Web Editor showing batch job screen, with running job and delete button highlighted.

To stop the process, simply click the highlighted bin button (delete). This should allow the tool to work normally again.

6.3 Value error (S2-MBMP)

```
-----
ValueError                                Traceback (most recent call last)
Cell In[50], line 20
    18 Corrected_No_B12 = No_B12 + median_diff_no
    19 # Calculate the fractional change
--> 20 frac_change = (Active_B11 - Corrected_Active_B12) - (No_B11 - Corrected_No_B12)

ValueError: operands could not be broadcast together with shapes (169,252) (650,913)
```

This error can happen for 2 reasons.

Firstly because the user has accidentally selected different landfills for each section of the tool. This downloads slightly different shaped images and are incompatible.

The second reason is that the tool has been used multiple times and the new download failed to overwrite the old download. If you suspect this is the case, close Jupyter-lab and Anaconda Navigator, then in your data directory, delete the following files.

- Sentinel-2_active_emissionMBMP.GTiff
- Sentinel-2_no_emissionMBMP.GTiff
- Sentinel-2_truecolour.GTiff

Then reopen everything and run the tool again.

6.4 Internal Server Error (All Tools)

```
OpenEoApiError: [500] Internal: Server error: KazooTimeoutError('Connection time-out') (ref: r-240
5108830e742b59ef8ac2f28647fb5)
```

This can occur when there are issues with the Copernicus network. In the event that you see an error like this you can check page <https://dataspace.copernicus.eu/news> for any downtime messages and you can also contact the Copernicus dataspace team via the form at <https://helpcenter.dataspace.copernicus.eu/hc/en-gb/requests/new>

2920 Words

7. References

- Aguasca, N. (2024). Zoom conversation with Nuria Aguasca from PreZero España, 9th of April 2024.
- Castillo-Giménez, J., Montañés, A., & Picazo-Tadeo, A.J. (2019). Performance in the treatment of municipal waste: Are European Union member states so different? *Science of the Total Environment*, 687, 1305-1314.
- European Union. (1999). Council Directive 1999/31/EC of 26th of April 1999 on the landfill of waste. Retrieved 15th of April, 2024, from: <https://eur-lex.europa.eu/eli/dir/1999/31/2018-07-04>
- European Space Agency (2021) Satellites detect large methane emissions from Madrid landfills. Retrieved 14th of April, 2024, from: https://www.esa.int/Applications/Observing_the_Earth/Satellites_detect_large_methane_emissions_from_Madrid_landfills
- European Environment Agency. (2022). Early warning assessment related to the 2025 targets for municipal waste and packaging waste - Spain. Retrieved 5th of May, 2024 from: <https://www.eea.europa.eu/publications/many-eu-member-states/spain>
- Ferronato, N., Torretta, V., Ragazzi, M., & Rada, E.C. (2017). Waste mismanagement in developing countries: A case study of environmental contamination. *UPB Sci. Bull*, 79(2), 185-196.
- Google Earth. (2024). Satellite view of Algerian petrochemical facility. Retrieved 10th of May 2024.
- Hidalgo, M. (2024). Zoom conversation with Marcelino Hidalgo from PreZero España, 10th of April 2024.
- Jet Propulsion Laboratory. (2024). VISIONS: The EMIT Open Data Portal. Retrieved 10th of May 2024 from: <https://tinyurl.com/2s4kkwe5>
- Microsoft Copilot Designer. (2024). Cover artwork. Generated on 5th of May 2024, Available at: <https://tinyurl.com/45rr2h52>
- OpenEO. (2024). NDVI Timeseries. [Online]. Retrieved 22nd of April 2024, from: https://documentation.dataspace.copernicus.eu/notebook-samples/openeo/NDVI_Timeseries.html
- Pandey, S., van Nistelrooij, M., Maasackers, J.D., Sutar, P., Houweling, S., Varon, D.J., Tol, P., Gains, D., Worden, J., & Aben, I. (2023). Daily detection and quantification of methane leaks using Sentinel-3: a tiered satellite observation approach with Sentinel-2 and Sentinel-5p. *Remote Sensing of Environment*, 296, 113716.
- Parker, R., Boesch, H., Cogan, A., Fraser, A., Feng, L., Palmer, P.I., Messerschmidt, J., Deutscher, N., Griffith, D.W., Notholt, J., & Wennberg, P.O. (2011). Methane observations from the Greenhouse Gases Observing SATellite: Comparison to ground based TCCON data and model calculations. *Geophysical Research Letters*, 38(15).
- Salami, C. (2024). Zoom conversation with Carlos Salami from PreZero España, 10th of April 2024.
- Sentinel Hub. (2024) "About Sentinel-2 Data." Retrieved 5th of May 2024, from: <https://docs.sentinel-hub.com/api/latest/data/sentinel-2-l2a/>
- Sentinel Hub. (2024) "About Sentinel-5P Data." Retrieved 23rd April 2024, from: <https://docs.sentinel-hub.com/api/latest/data/sentinel-5p-l2/>
- Themelis, N.J. and Ulloa, P.A. (2007). Methane generation in landfills. *Renewable Energy*, 32(7), 1243-1257.
- Varon, D. J., Jervis, D., McKeever, J., Spence, I., Gains, D., & Jacob, D. J. (2021). High-frequency monitoring of anomalous methane point sources with multispectral Sentinel-2 satellite observations. *Atmospheric Measurement Techniques*, 14, 2771–2785.

Vigano, I., Van Weelden, H., Holzinger, R., Keppler, F., McLeod, A., & Röckmann, T. (2008). Effect of UV radiation and temperature on the emission of methane from plant biomass and structural components. *Biogeosciences*, 5(3), 937-947.