Landfill Methane Emission Tool How to guide.

Name: Nicholas Kinsella, Student number: B00893122

Table of Contents

[1. Introduction and justification 1](#_Toc164721012)

[2. Setup 2](#_Toc164721013)

[2.1 Anaconda Navigator 2](#_Toc164721014)

[2.2. Creating a Conda Environment 2](#_Toc164721015)

[2.3 Setting up Jupyter Lab 4](#_Toc164721016)

[2.4 openEO setup using Anaconda Navigator 6](#_Toc164721017)

[2.5 openEO setup using PyPi 7](#_Toc164721018)

[2.6 Registering with Copernicus Data Space Ecosystem. 8](#_Toc164721019)

[2.7 Authentication with openEO 9](#_Toc164721020)

[3. Methodology 10](#_Toc164721021)

[3.1 Gas Timeseries: 10](#_Toc164721022)

[3.3 Gas Concentration Map: 14](#_Toc164721023)

[4. Expected results 19](#_Toc164721024)

[5. Troubleshooting 20](#_Toc164721025)

[5.1 Remote disconnected error 20](#_Toc164721026)

[5.2 Concurrent job error 21](#_Toc164721027)

[6. References 22](#_Toc164721028)

# 1. Introduction and justification

Methane (CH4) is the second most impactful greenhouse gas after carbon dioxide. Its warming potential is around 28 times more than CO2 over a 100-year period, although it is far less persistent in the atmosphere due to it being broken down by interaction with ultraviolet sunlight (Vigano et al., 2008). Despite this attrition, its concentration in the earth’s atmosphere has increased more than 250% since the industrial revolution and it is estimated to be responsible for at least a quarter of anthropogenic warming (Pandey et al., 2023).

Landfills are a significant contributor to global methane emissions du The 1999 EU Landfill Directive requires that methane produced by the anaerobic decomposition of organic waste is either collected for sale on the energy market, or is burnt off by flaring (European Union, 1999; Themelis & Ulloa, 2007). Despite this, methane plumes can still be produced if landfill conditions are mismanaged (Ferronato et al., 2017).

Spain sends more waste to landfills than any other country in the European Union at 12 million tonnes per year in 2017 (Grupo SPR, 2020) although on a per capita basis, it ranks 23rd in the block (Eurostat, 2022). Although the management of landfills in Spain is on par with countries like the UK, Ireland, Sweeden and Italy (Castillo-Giménez et al., 2019) unexpected emission events can still occur. In August and October 2021 two very large plumes of methane were detected by earth observation satellites from a landfill near Madrid. Had this gas been captured, it is estimated to have been enough to heat as many as 350,000 homes for a year (European Space Agency, 2021).

PreZero, a multinational waste management company, operates 23 landfill active and closed landfills across Spain. The author interviewed several of the staff regarding their practices and they currently do not use remote sensing platforms to monitor any gas emissions from their facilities, instead relying on ground-based detectors in a one detector per hectare grid (citation https://library-guides.ucl.ac.uk/harvard/personal-communication). Such an approach might miss emission events from unforeseen locations, such compromised as closed cells, or other areas away from the open face of the landfill. The staff also expressed an interest in seeing data for other gasses, particularly those associated with smells that can lead to complaints.

Two tools for displaying atmospheric CH4 data along with other gasses of interest from S5P were created in the programming language Python.

1. **Gas Timeseries:** The first tool displays a time series of gas values for each landfill location over a set of dates that can be specified by the user. The gasses which can be monitored are CO, HCHO, NO2, O3, SO2, CH4 If any emission events are detected, the tool can indicate how many days the emission persisted for.
2. **Gas Concentration Map:** A second tool was developed to display a map of the dataset for a specific day to allow visual assessment.

This guide outlines the installation and use of this tool. The most up-to-date version of it can be downloaded from the following Github repository: <https://github.com/zelcon01/egm722>.

# 2. Setup

Anaconda Navigator is an application that contains the Python programming language along with the additional tools, libraries and utilities. Within Anaconda lies the package manager ‘Conda’ which allows the end user to create a shareable development environment containing the necessary packages to run a particular script, including Jupyter Lab, where the Python code itself will be ran. We will be installing these in this section.

## 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’. You will see the installed packages (fig.1).

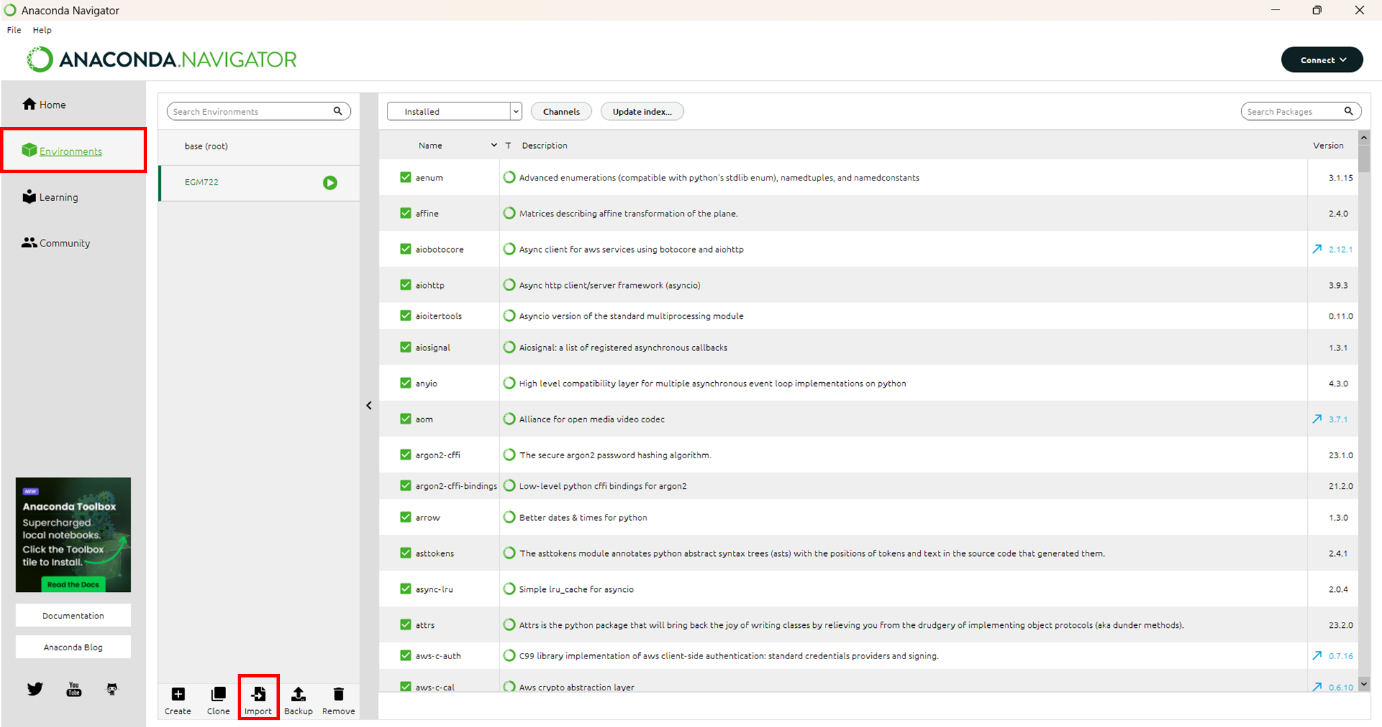


Figure 1: Environments tab of Anaconda Navigator with environments tab and import button highlighted in red.

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

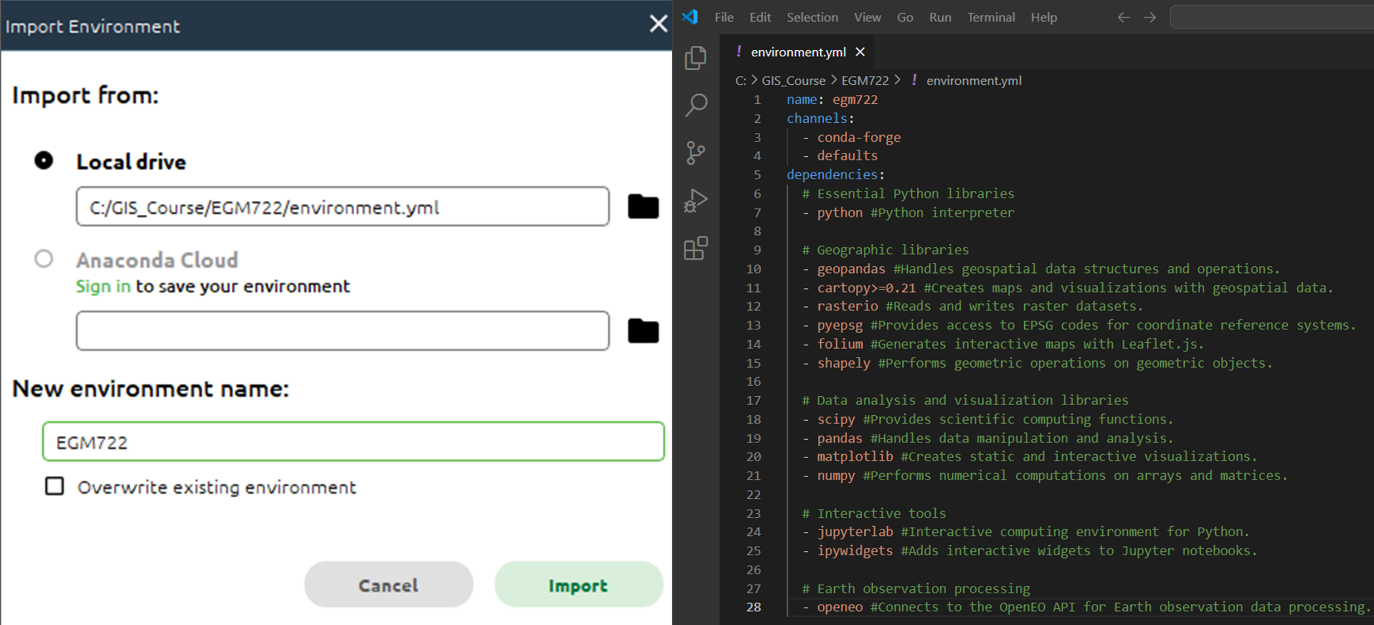


Figure 2:The import config box (left) and the contents of ‘environment.ymal’ (right).

Click Import. Depending on the connection speed of your network, the process of installing all the packages may take several minutes. Once the installation is finished you will be returned to the environments tab (fig.1) and you should see that over 160 packages have been installed.

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

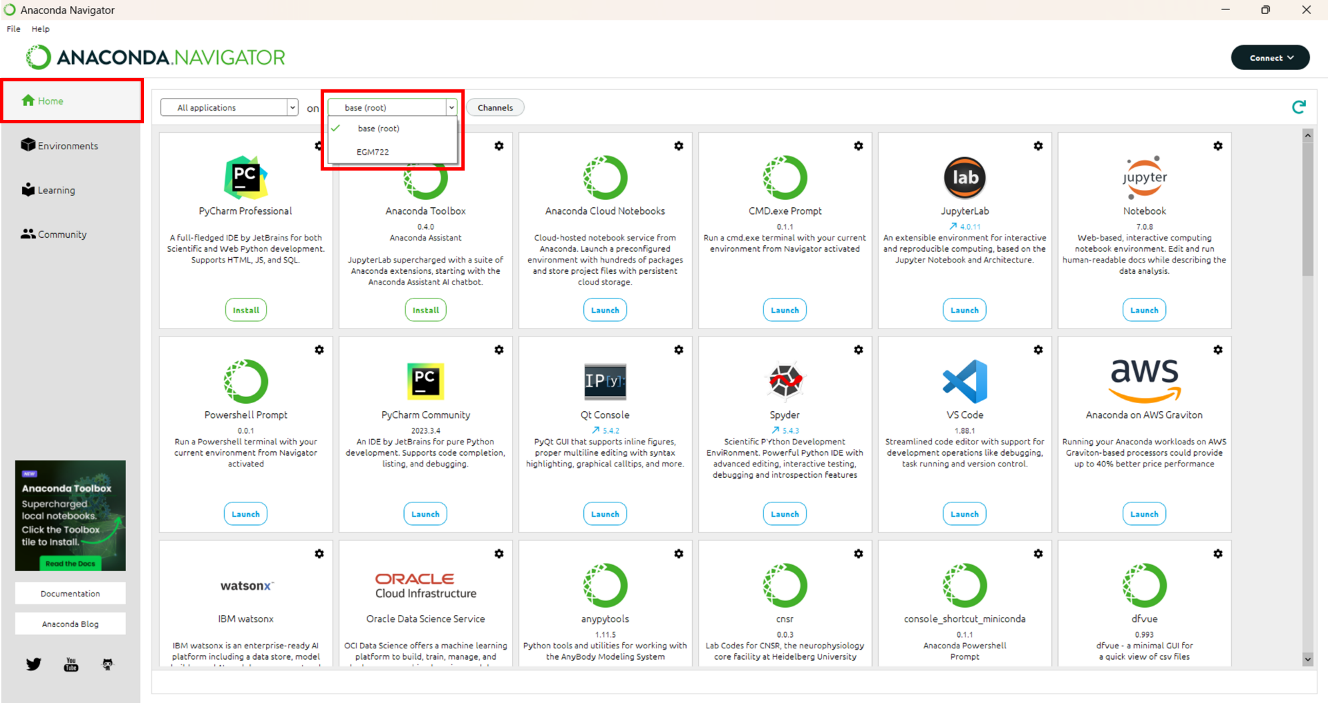


Figure 3: Anaconda Navigator with home tab and environment switching dropdown in red.

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). **When running the startup process it is essential that your environment name is selected here or the dependencies we 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. To do this, launch the CMD.exe Prompt, ensuring that your new environment is selected (fig.4)

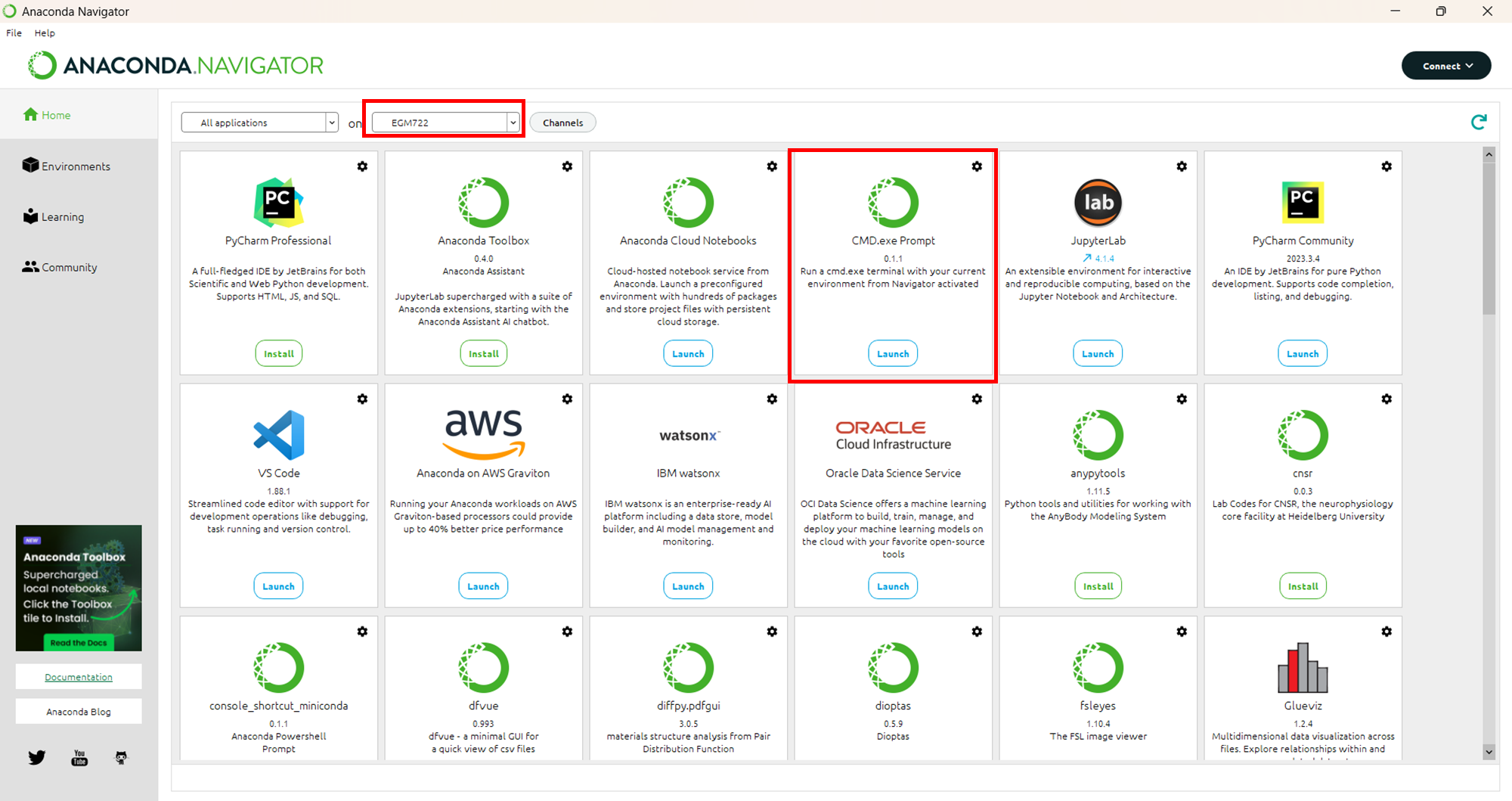


Figure 4: Highlighted locations of selected environment and CMD.exe Prompt

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 juptyer\_lab\_config.py. On Windows this is usually C\Users\<your\_username>.

Jupyter lab will by default open in your user directory. Unfortunately due to security restrictions it isn’t possible to navigate to the parent directory of the launch location. So if Jupyter launches in ‘C:\Users\RockyBalboa, it isnt’ possible to move to ‘C:\Users’ or, ‘C:\EGM722’ for example. 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.

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 isn’t the case, you will need to do the following:

Open an Anaconda Navigator CMD.exe prompt (remembering of course to activate your new environment before launching it) and type the following command:

|  |
| --- |
| jupyter --paths |

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

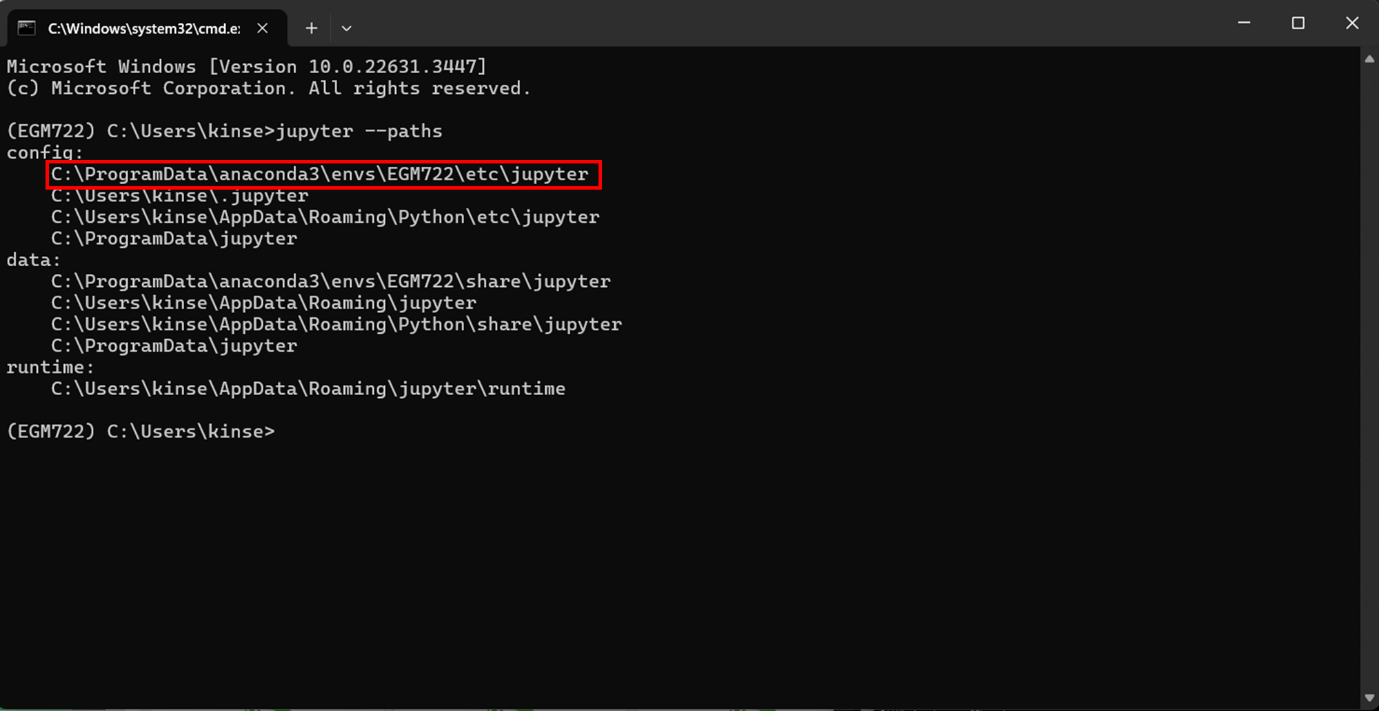


Figure 5: results of ‘jupyter –paths’ command showing path used by environment highlighted in red.

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 will do. Using the shortcut ‘CTRL + F’ type in the following line of code: ‘c.ServerApp.root\_dir’ and you should find the section highlighted in figure 6.

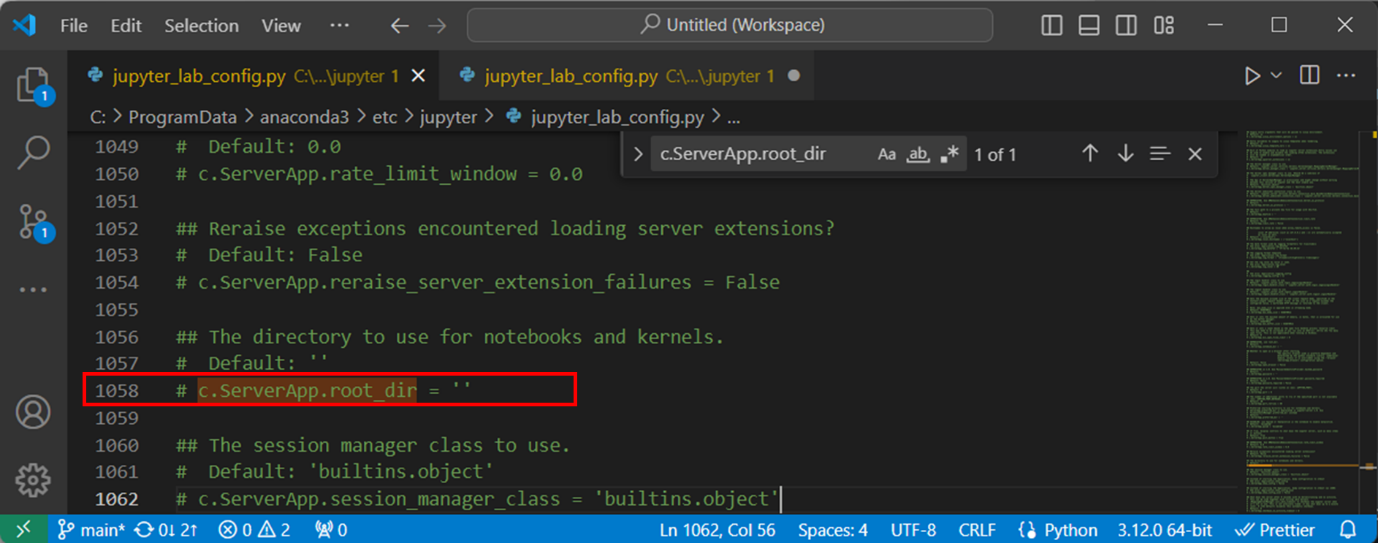


Figure 6: location of 'c.ServerApp.root\_dir' in jupyter\_lab\_config.py

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).

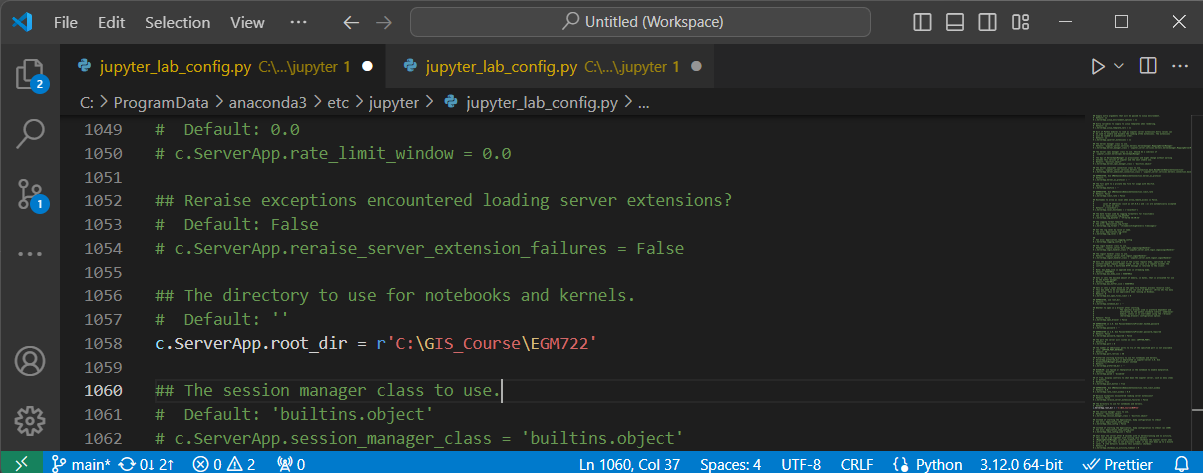


Figure 7: path to data directory added to jupyter\_lab\_config.py

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 (figure 8).

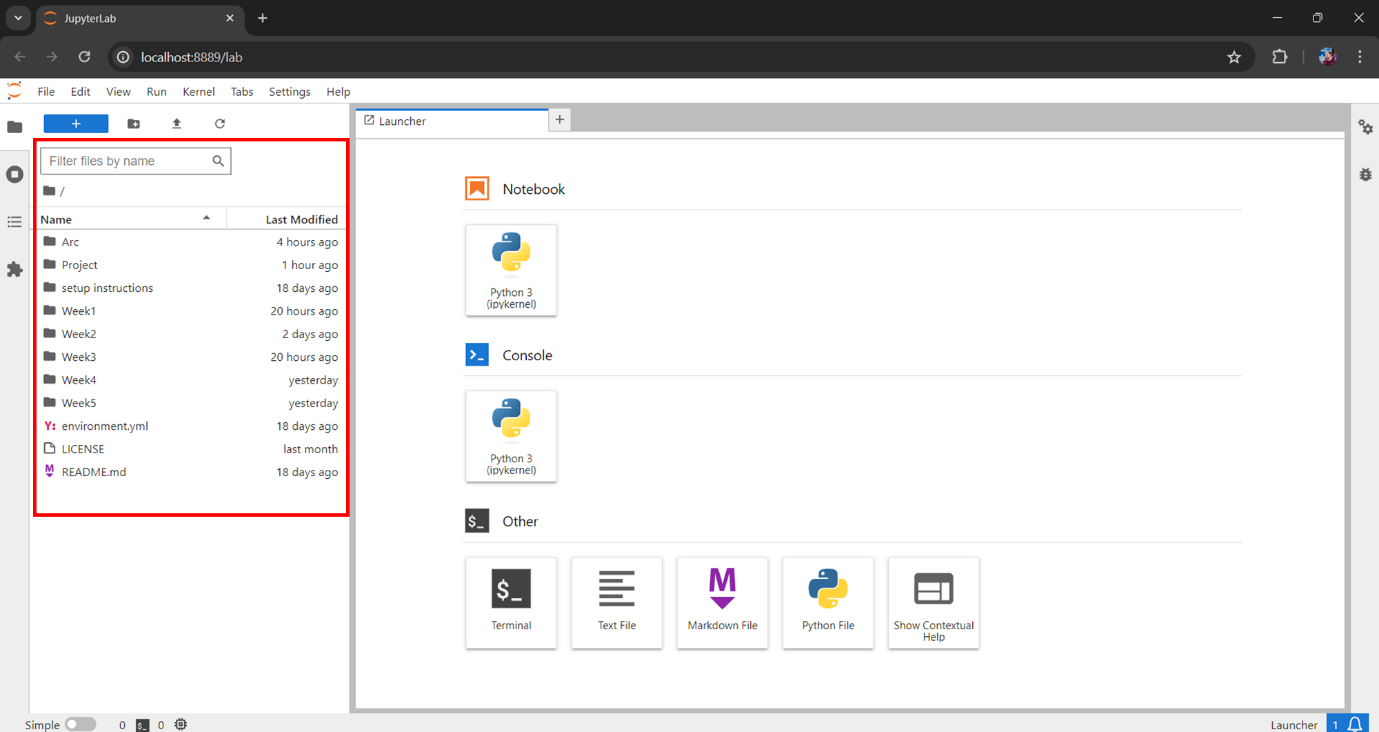


Figure 8: Jupyter Lab showing by default the data directory

### 2.4 openEO setup using Anaconda Navigator

openEO is an open-source API that allows access to the earth observation satellite missions provided by the Copernicus program. These include Sentinel series of satellites used by this tool.

To install openEO you can 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.

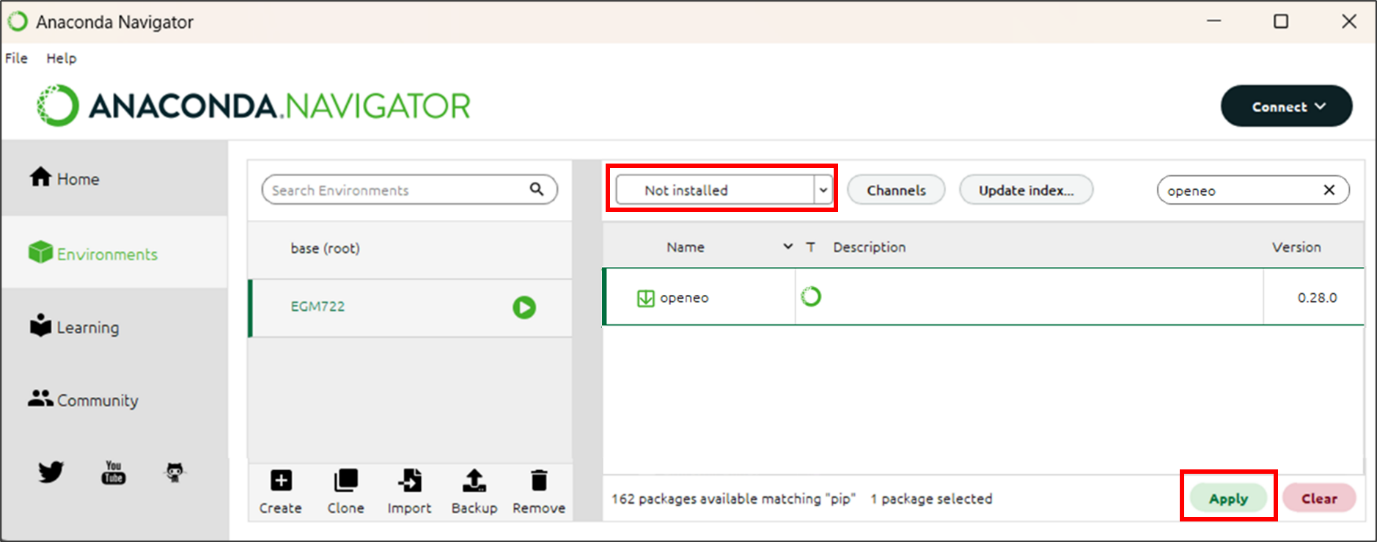


Figure 9: installing openEO from Anaconda Navigator.

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

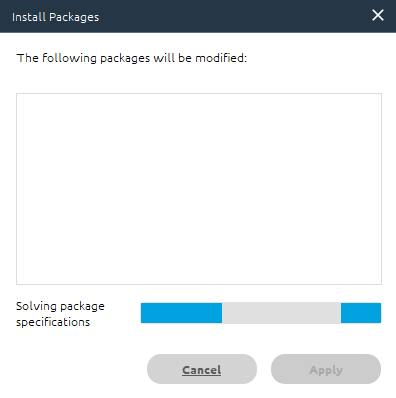


Figure 10: Anaconda Navigator package installer loading screen.

## 2.5 openEO setup using PyPi

In the event that Anaconda Navigator cannot find openEO you can use PyPi, the official third-party software library for Python.

Using the search function in the Anaconda Navigator Environments tab, search for ‘pip’, selecting the appropriate tick-box and then clicking apply (fig.11), then clicking apply once the install packages prompt (fig.10) has finished loading.

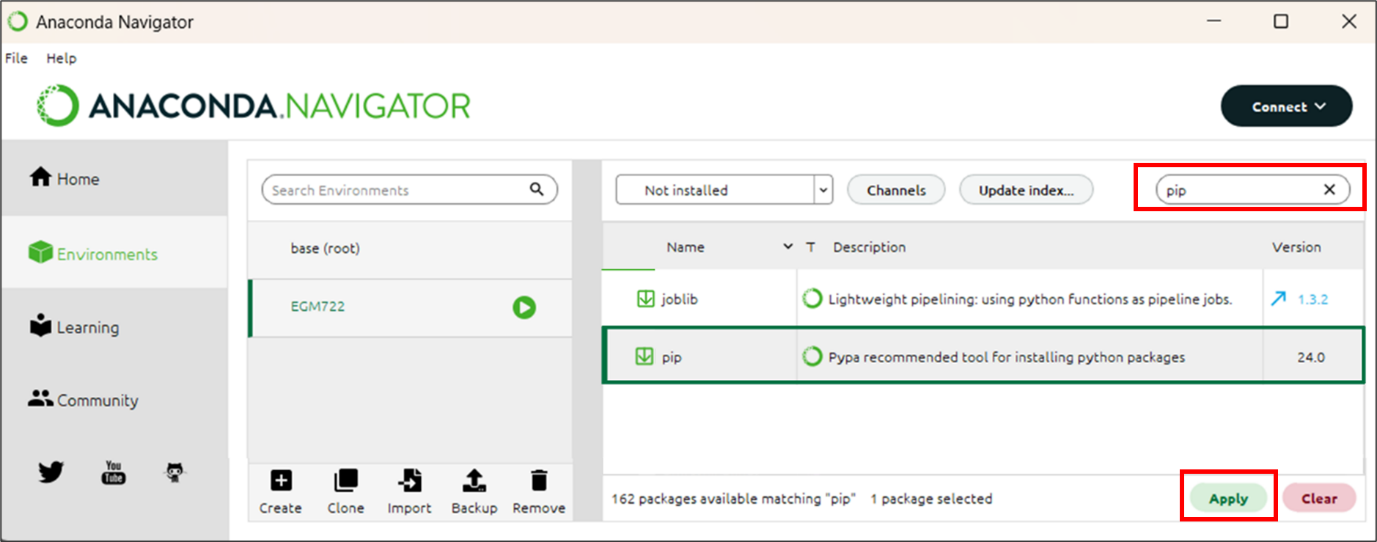


Figure 11: Installing pip via Anaconda Navigator

Open an Anaconda Navigator CMD.exe prompt (remembering as always activate your new environment before launching it) and type the following command.

|  |
| --- |
| pip install openeo |

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

## 2.6 Registering with Copernicus Data Space Ecosystem.

Although it is possible to browse openEO metadata without being logged in. 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)

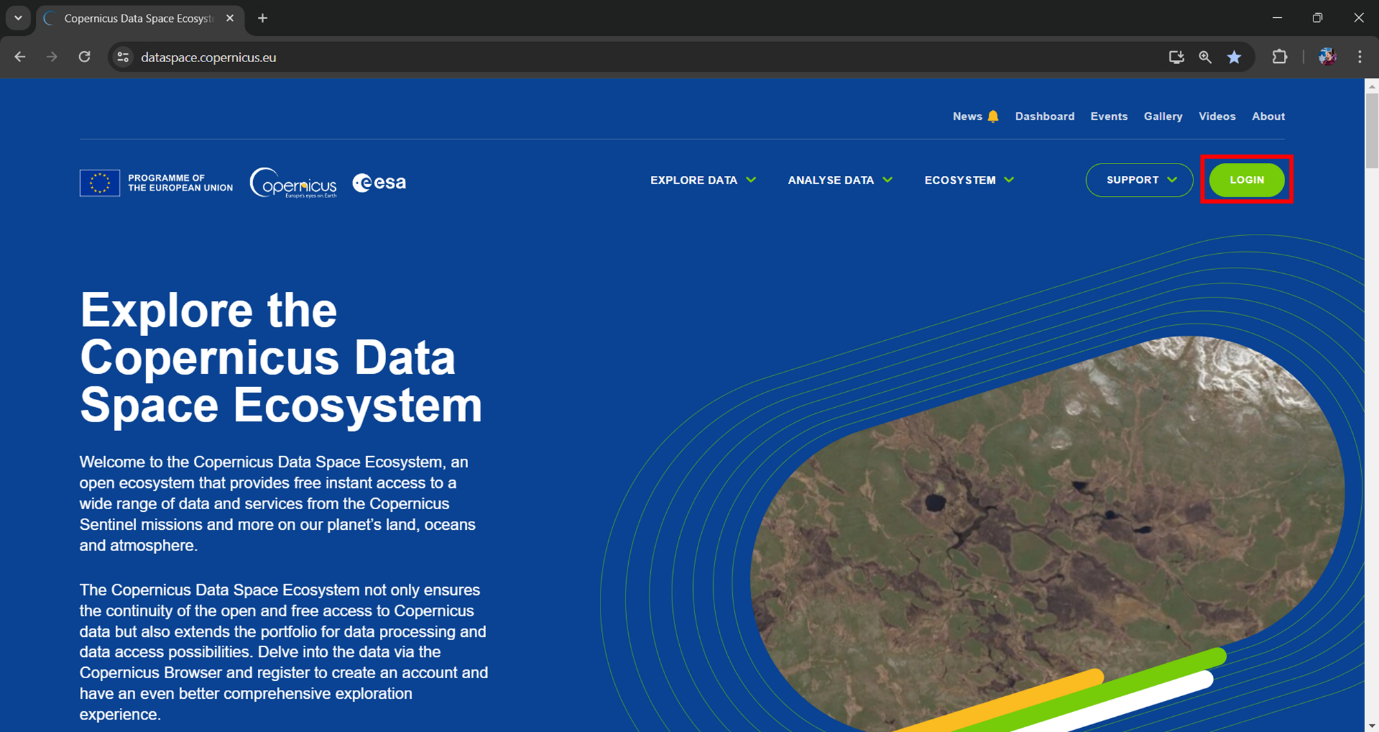


Figure 12: Copernicus Dataspace landing page with login button highlighted in red.

Next click the green ‘register’ button:

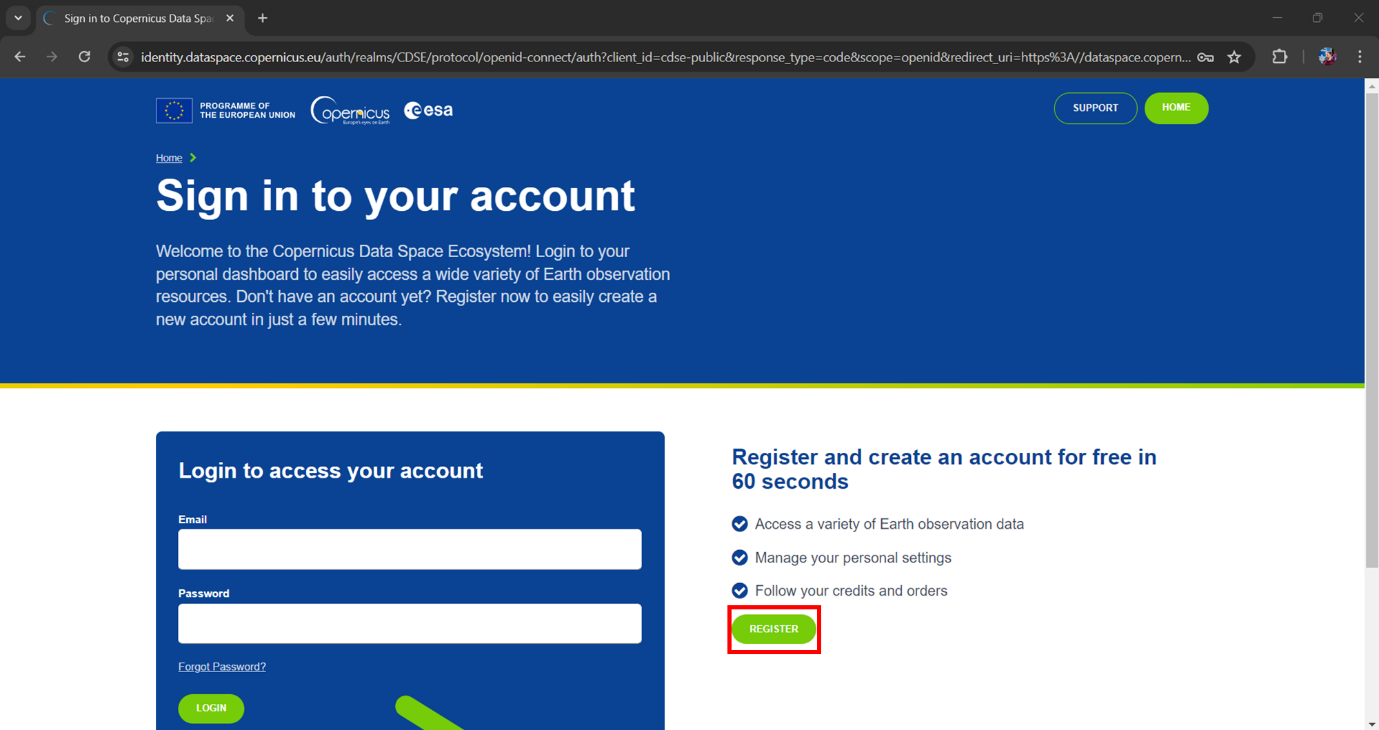


Figure 13: Copernicus Dataspace sign in page.

On the following page, fill out the application form and then at the bottom, complete the prompt to show that you are not a robot and then the green ‘register’ button (fig.14).

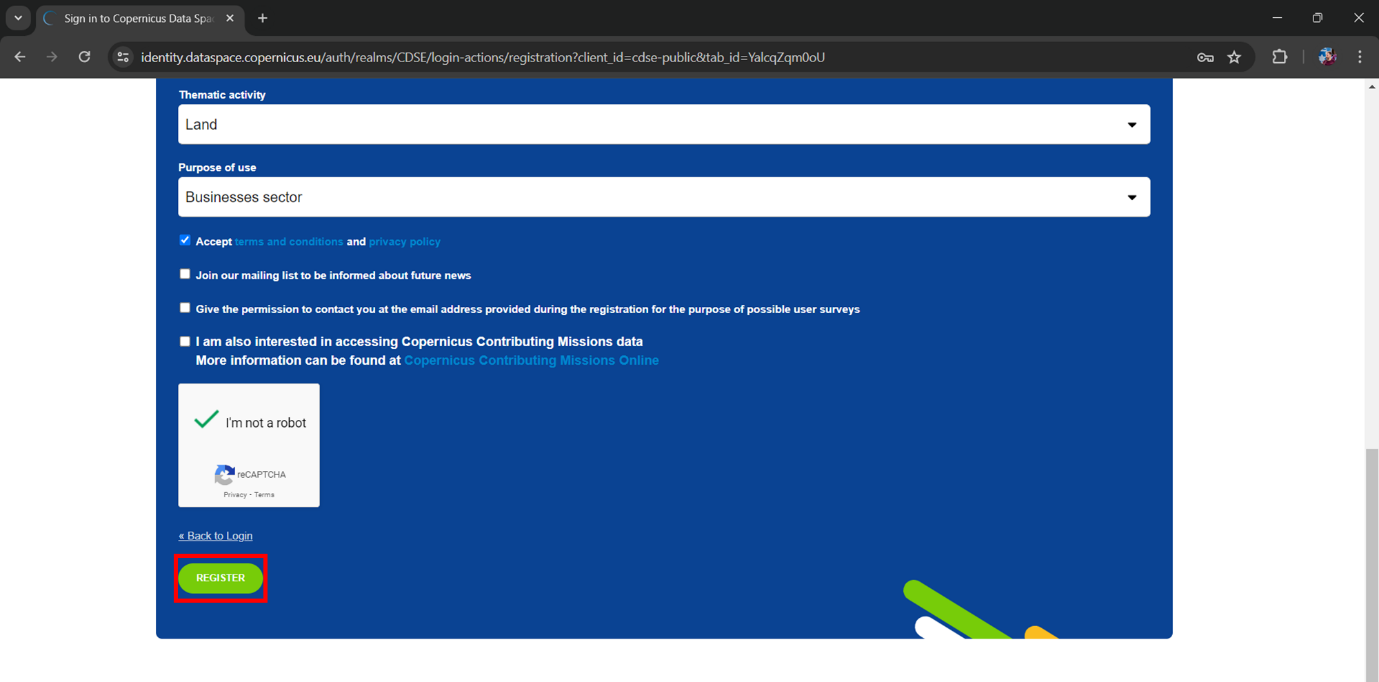


Figure 14: End of Copernicus registration page with register button highlighted in red.

Once registered, you will receive an email asking to verify your address. Click the ‘verify email address’ button as soon as you receive it. The registration process will then be complete and you can log-in with your email and chosen password.

Should you have any problems with registration, you can email help-login@dataspace.copernicus.eu

## 2.7 Authentication with openEO

The very first time 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. |

You need to copy this URL into your web browser and login using the Copernicus Data Space Ecosystem email and password. Once this is complete, run tool’s Pyhton 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 (2.7).

# 3. Methodology

(A methods section that clearly explains what your code does – if you’re performing a certain kind of analysis, this should explain the steps of the analysis and the theory behind it. This section should be written in the style of a methods section for a journal article or technical report.)

## 3.1 Gas Timeseries:

The timeseries functionality used here was based on a Normalised Difference Vegetation Index time series using Sentinel 2 data developed by the team at openEO (2024), although it has been significantly modified to draw Sentinel 5P data for a selection of gas types, using geometries based on a file of point data provided by PreZero. Despite its low spatial resolution of 5.5 x 3.5km, Sentinel 5p was chosen as it provides measurements of gases in dedicated level 2 datasets and its temporal resolution is very high, providing daily data (Sentinel Hub, n.d.), which is essential when investigating transient gases.

Figure 15 provides an overview of the main steps in processing and presenting the data and the code itself is presented thereafter.

The tool takes its primary data from the Copernicus Sentinel 5P CH4 dataset which became available in 2021. This dataset provides daily methane measurements at a spatial resolution of 5.5 x 3.5km

The tool makes a timeseries of methane emissions for all 23 sites, displaying these on a chart allowing the user to see how many days an emission event lasts.

The tool then allows the end user to see a map of the methane data for a specific date over a specific landfill. Spatial statistics are then provided to estimate the peak level of methane in parts per billion as well as the average of the plume.

When asked which of the gasses they would be interested in, PreZero staff said that measuring CH4 was the most important, but that the other gasses mentioned were also of interest, particularly HCHO and SO2 for things like unpleasant odours which can lead to complaints from local residents.

The Copernicus Sentinel 5P (S5P) satellite CH4 dataset became available in 2021. Alowing daily readings of atmospheric methane concentrations globally and this was used in conjunction with the commercial GHGSat to detect the methane plumes near Madrid (European Space Agency, 2021).

A methods section that clearly explains what your code does – if you’re performing a certain kind of analysis, this should explain the steps of the analysis and the theory behind it. This section should be written in the style of a methods section for a journal article or technical report.

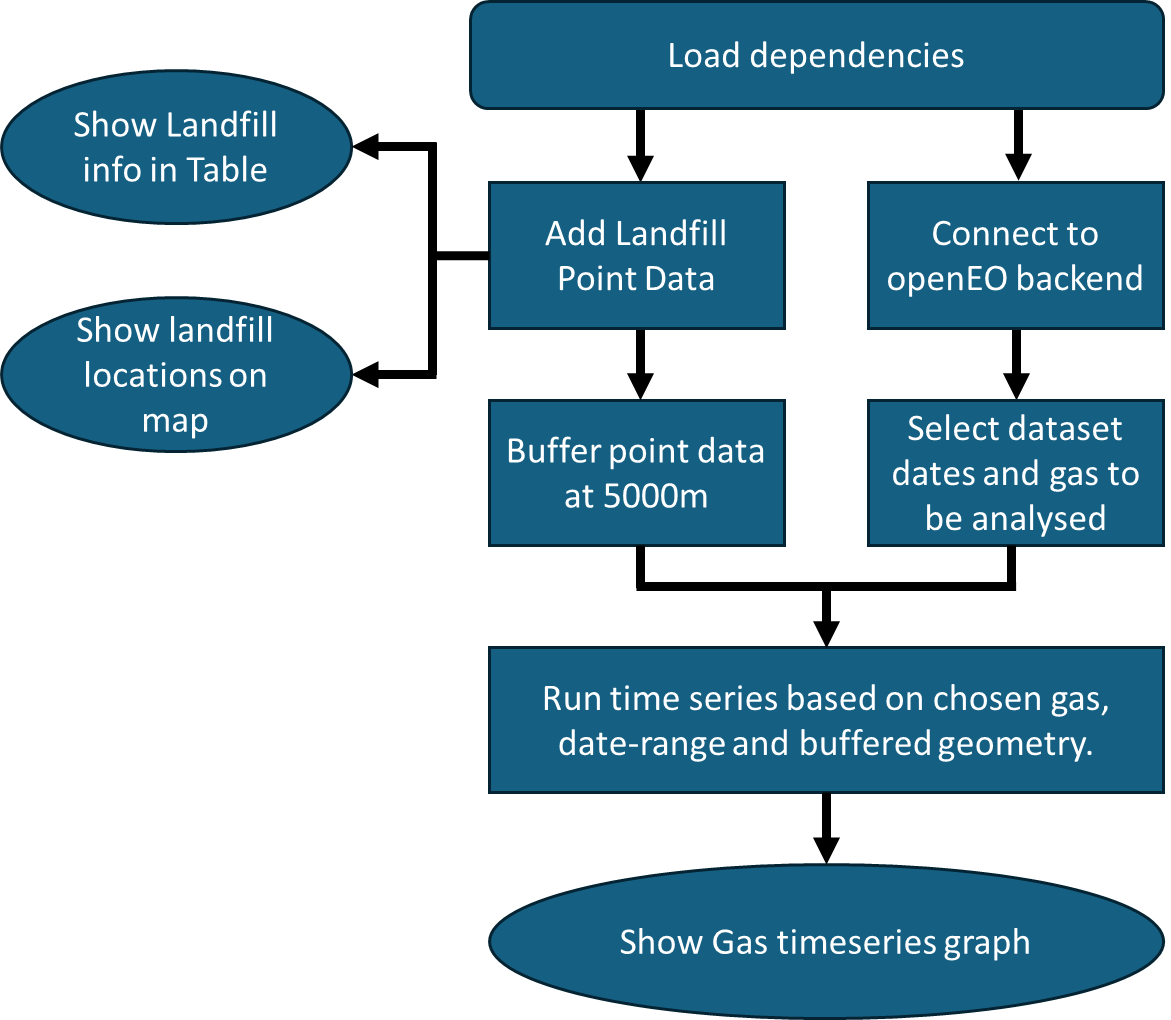


Figure 15: Flowchart of processes used for Gas timeseries analysis.

Code 1 shows the dependencies that are loaded, each of their uses are as follows:

* **Folium:** Used to visualize geospatial data on an interactive map.
* **Pandas:** For data analysis and manipulation.
* **Matplotlib:** for plotting.
* **Scipy.signal:** for signal processing.
* **Numpy:** for numerical computing.
* **Geopandas:** for working with geographic data.
* **OpenEo:** for working with the OpenEO API for Earth Observation data processing
* **Shapely.geometry (shape):** for geometric operations

|  |
| --- |
| 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.

|  |
| --- |
| connection = openeo.connect(url="openeo.dataspace.copernicus.eu") |
| connection.authenticate\_oidc() |

Code 2: Connecting to openEO

|  |
| --- |
| landfills = gpd.read\_file(r"C:\GIS\_Course\EGM722\Project\Data\test\_landfill.geojson") |
| landfills |

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

|  |
| --- |
| # This creates the map and centres it on the geometries. |
| centroids = landfills.geometry.centroid |
| center = [centroids.y.mean(), centroids.x.mean()] |
| site\_map = folium.Map(location=center, 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.

|  |
| --- |
| # This section applies a buffer of 5000m to each landfill for the analysis. |
|  |
| # loading dataframe |
| landfill\_5000m = gpd.read\_file(r"C:\GIS\_Course\EGM722\Project\Data\test\_landfill.geojson") |
|  |
| # 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\_5000m = landfill\_5000m.to\_crs(epsg=2062) |
|  |
| # Now the dataframe is converted, a buffer of 5000m is added to each point |
| landfill\_5000m['geometry'] = landfill\_5000m.buffer(5000) |
|  |
| # Now the buffered data needs to be converted back to EPSG:4326 because the Sentinel data is projected in EPSG:4326. |
| landfill\_5000m = landfill\_5000m.to\_crs(epsg=4326) |
|  |
| # The time series analysis requires that A GeoJSON format file is used for the analised areas, so this bit produces a file suitable for that. |
| landfill\_5000m\_geojson = landfill\_5000m.\_\_geo\_interface\_\_ |

Code 5: Code for adding buffers to landfill point data

|  |
| --- |
| # This selects the specific dataset for the time series analysis. |
|  |
| 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.

|  |
| --- |
| 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.

|  |
| --- |
| def plot\_timeseries(filename, figsize=(15, 10)): #here you can specify how big the graph is on the screen |
| 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) |
| 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 gasses") |
|  |
| # Calculate the minimum and maximum values |
| ymin = df["avg(band\_0)"].min() |
| ymax = df["avg(band\_0)"].max() |
|  |
| # Add 10% to the minimum and maximum values |
| 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="parcel id", loc='upper left', bbox\_to\_anchor=(1.02, 1), ncol=2) |
| ax.xaxis.set\_major\_locator(plt.MaxNLocator(30)) |
| ax.grid(True) |
|  |
| plot\_timeseries("Gas\_Timeseries\_results/Gas\_timeseries.csv") |

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

## 3.3 Gas Concentration Map:

Figure 16 provides an overview of the main steps in processing and presenting the data and the code itself is presented thereafter.

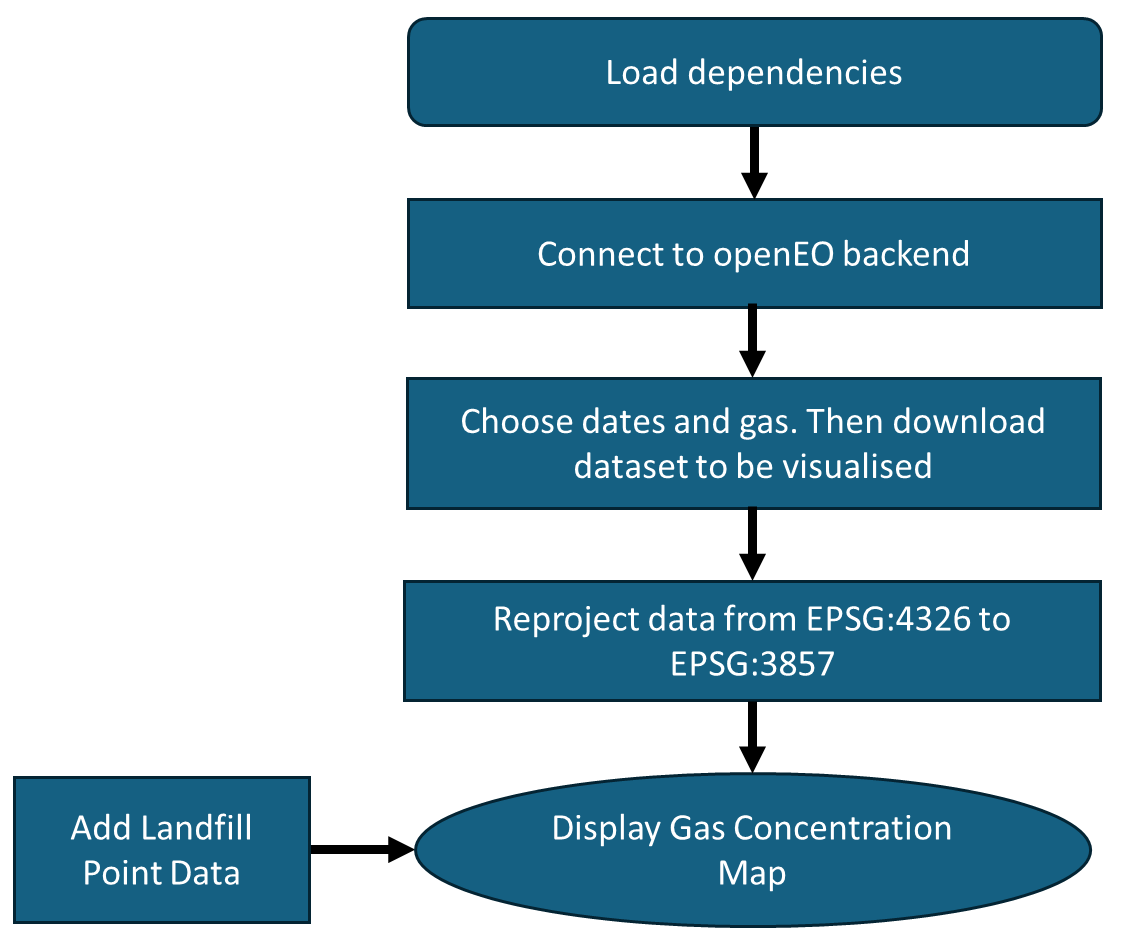


Figure 16: Flowchart of processes used for Gas Concentration Map

|  |
| --- |
| 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.

|  |
| --- |
| connection = openeo.connect(url="openeo.dataspace.copernicus.eu") |
| connection.authenticate\_oidc() |

Code 10: Connecting to openEO

|  |
| --- |
| 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.

|  |
| --- |
| dst\_crs = 'EPSG:3857' |
|  |
| # Open the gas data file that is in ESPG: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, |
| resampling=rasterio.warp.Resampling.nearest) |

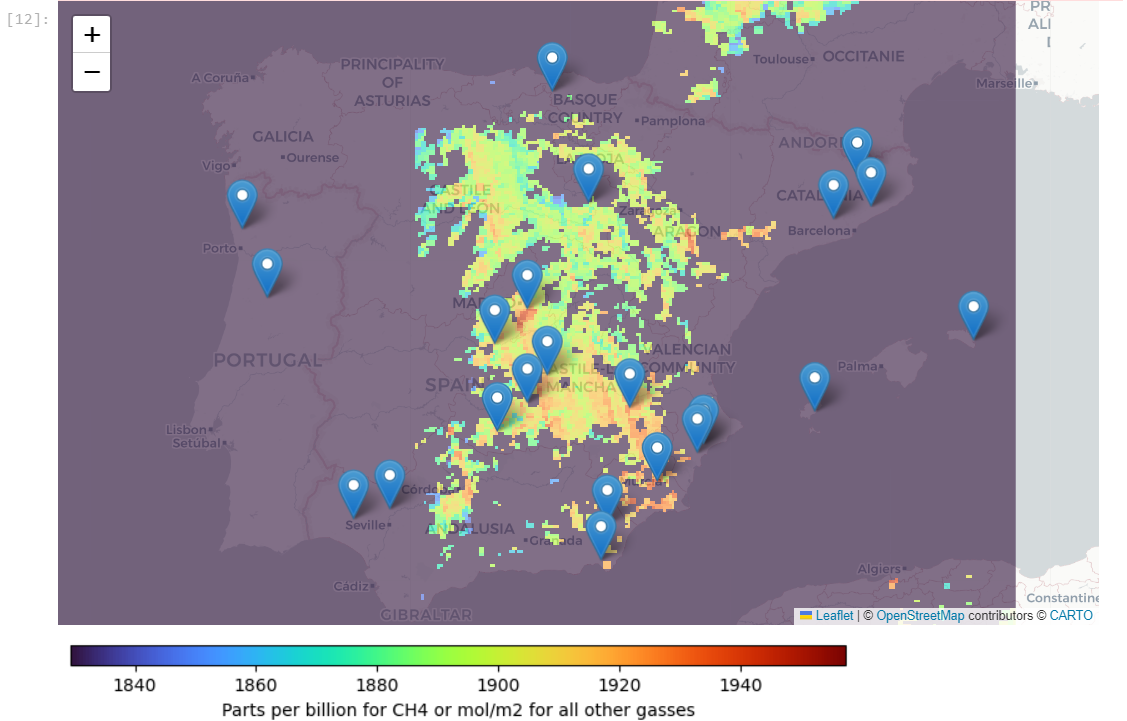
Code 12: Code for reprojecting raster

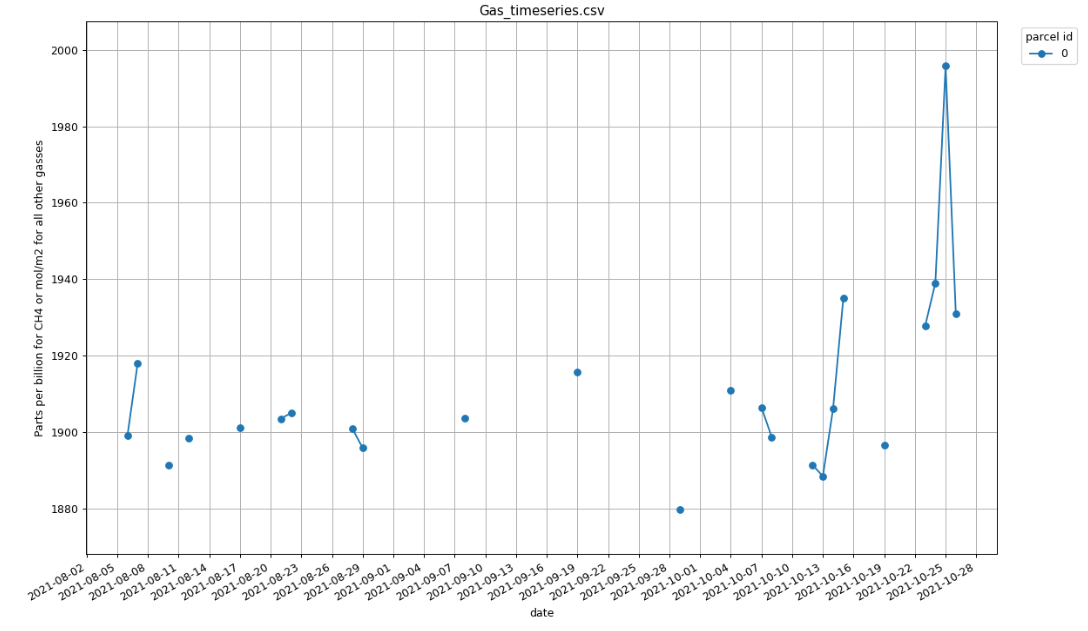
|  |
| --- |
| # This section loads the map |
|  |
| #This is the reprojected gas data |
| gas\_data = r'C:\GIS\_Course\EGM722\Project\Sentinel-5P\_Spain3857.GTiff' |
|  |
| # Open raster file, load values and prepare them to be displayed. |
| dataset = rasterio.open(gas\_data, 'r') |
| rasdata = dataset.read()[0] |
| rasdata\_normed = rasdata / rasdata.max() \* 10 |
|  |
| # set bounds using EPSG:3857 projection |
| dest\_crs = 'EPSG:3857' |
| left, bottom, right, top = [i for i in dataset.bounds] |
| bounds\_4326 = warp.transform\_bounds(src\_crs=dataset.crs, dst\_crs=dest\_crs, left=left, |
| bottom=bottom, right=right, top=top) |
|  |
| # Loading colourmap for gas\_data |
| colormap = cm.turbo |
|  |
| # 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 colormap for the non-zero values |
| colormap\_index = np.zeros\_like(rasdata, dtype=np.float64) |
| colormap\_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") |
| Test\_landfill\_Location = gpd.read\_file(r"C:\GIS\_Course\EGM722\Project\Data\test\_landfill.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 Test landfill location to the map and making it clickable for info |
| for feature in Test\_landfill\_Location.iterfeatures(): |
| # Extract feature number from properties |
| feature\_number = feature['properties']['DIRECCION'] |
| # 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=colormap(colormap\_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=colormap), |
| 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. Expected results

A section that explains the expected result of running your code.





# 5. Troubleshooting

A section that provides some troubleshooting advice in case things go wrong.

## 5.1 Remote disconnected error

|  |
| --- |
| 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>

## 5.2 Concurrent job error

|  |
| --- |
| 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.?)



Figure 17: OpenEO Web Editor with login button highlighted in red

You will then see the following screen.

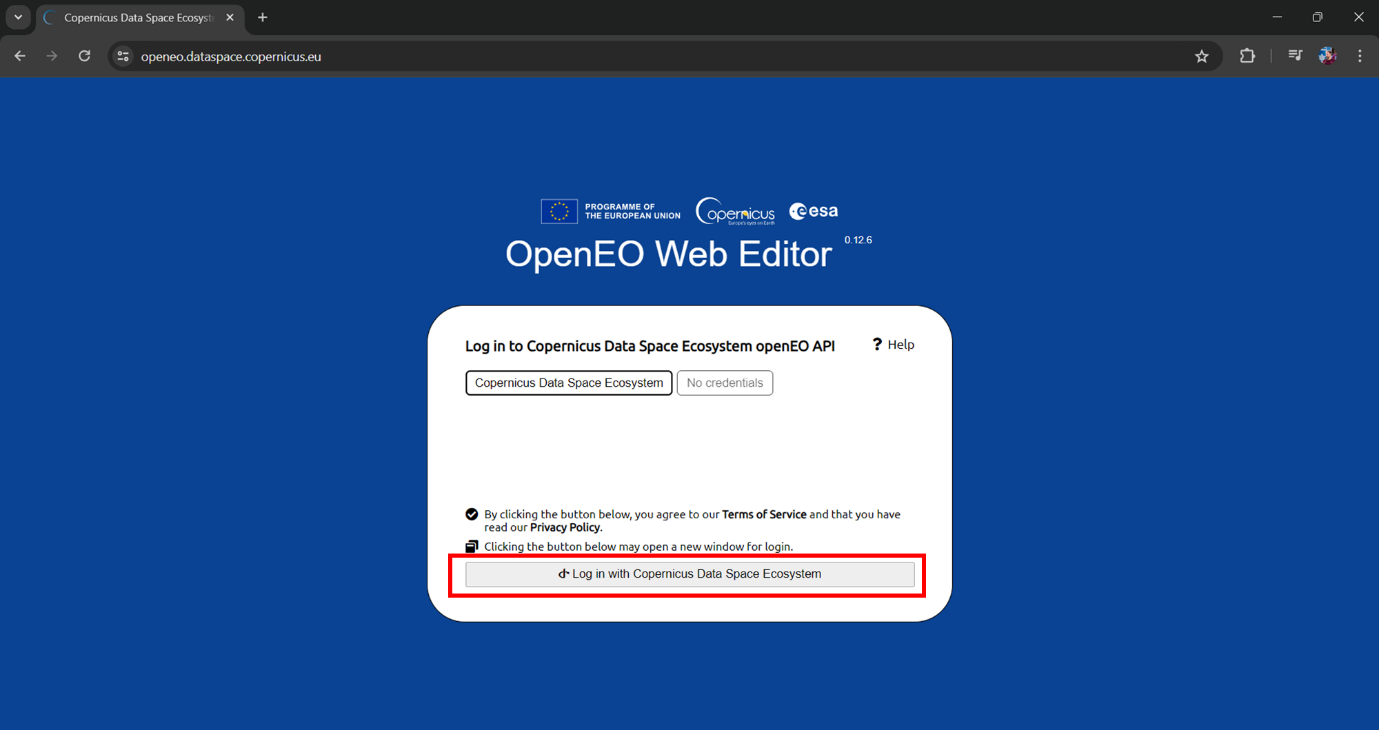


Figure 18: OpenEO Web editor login prompt with login button highlighted in red.

Simply click the highlighted button 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.

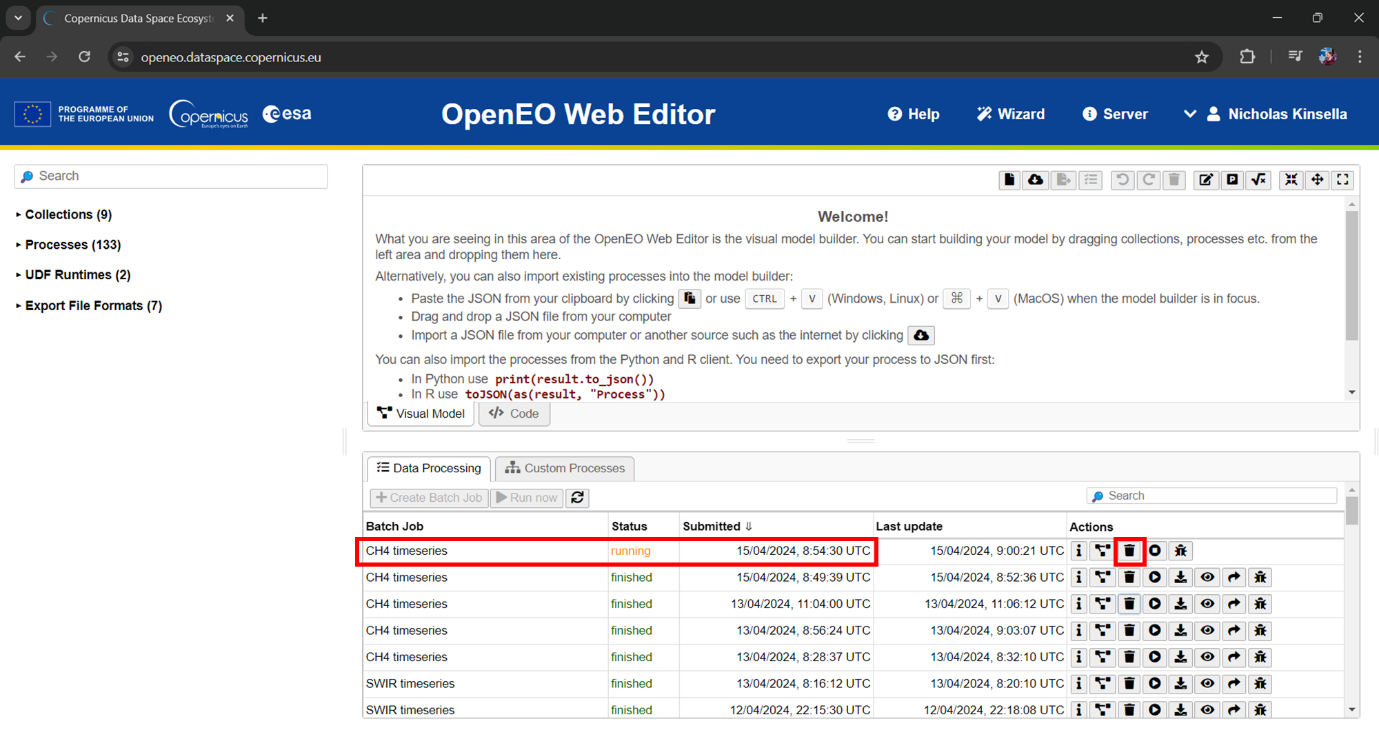


Figure 19: OpenEO Web Editor showing batch job screen, with running job and delete button highlighted.

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

# 6. References

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 26 April 1999 on the landfill of waste. Retrieved April 14, 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. Available at: https://www.esa.int/Applications/Observing\_the\_Earth/Satellites\_detect\_large\_methane\_emissions\_from\_Madrid\_landfills (Accessed: April 11, 2024).

Eurostat. (2022). Municipal waste statistics. Retrieved April 14, 2024, from: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Municipal\_waste\_statistics

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.

Grupo SPR (2020) Spain is the European country that throws the most waste into dumps. Available at: https://www.grupo-spr.com/en/spain-waste-dumps/ (Accessed: April 11, 2024).

openEO. (2024). NDVI Timeseries. [Online]. Available at: https://documentation.dataspace.copernicus.eu/notebook-samples/openeo/NDVI\_Timeseries.html (Accessed: April 22, 2024).

Pandey, S., van Nistelrooij, M., Maasakkers, 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.

Sentinel Hub. (n.d.) "About Sentinel-5P Data." [Online] Available at: https://docs.sentinel-hub.com/api/latest/data/sentinel-5p-l2/ [Accessed 23 April 2024].

Themelis, N.J. and Ulloa, P.A. (2007). Methane generation in landfills. Renewable Energy, 32(7), 1243-1257.

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.