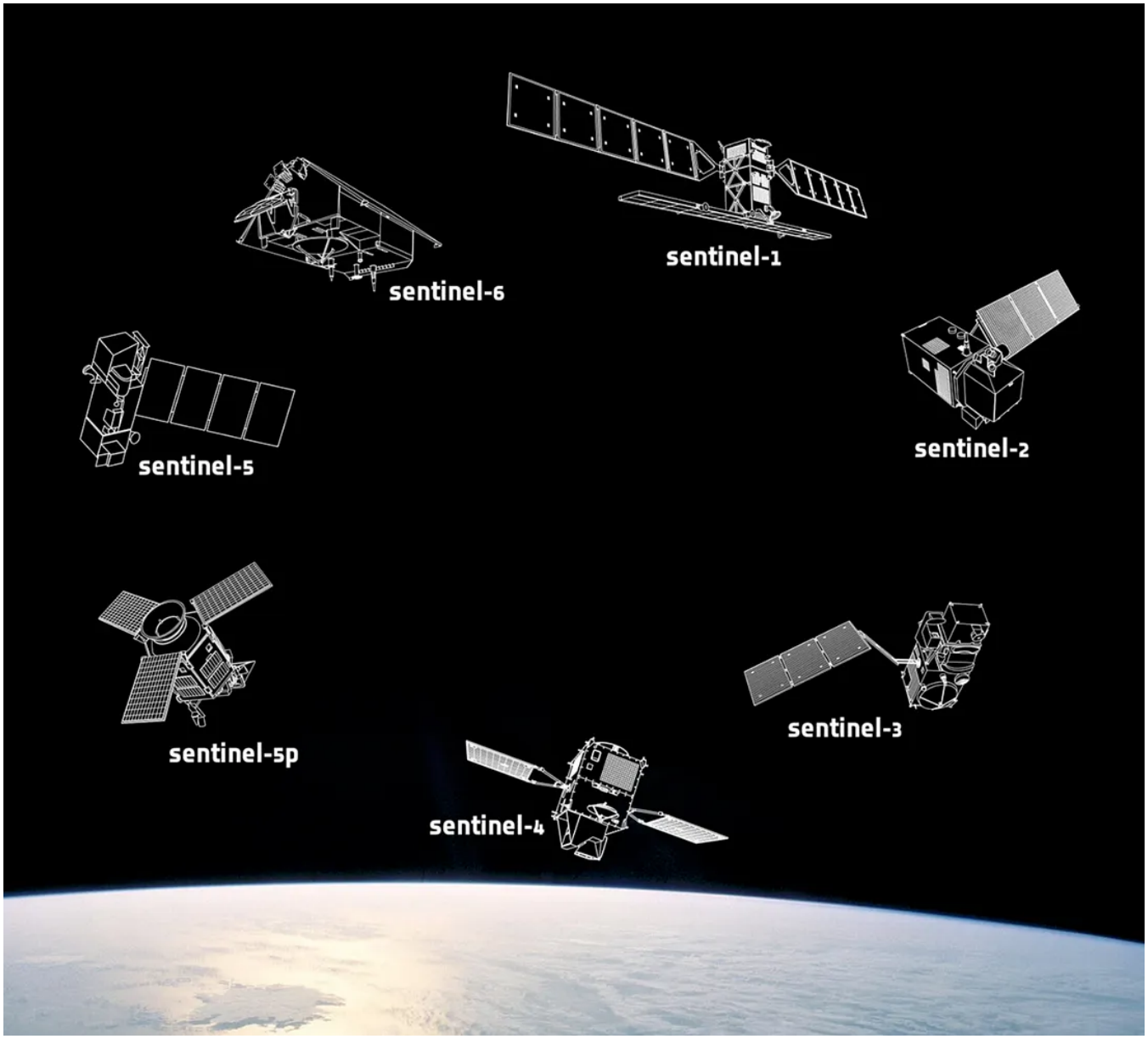
Querying and Downloading Sentinel Satellite Data with Python

8 min read · Jan 4, 2024

Gabriela Reis Follow

Welcome to this tutorial on accessing and downloading Sentinel satellite data using Python. Before we dive into the technical aspects, let's understand what Sentinel satellites are and why they are so important.



Sentinel Family. Source: ESA.

The Sentinel satellites are a fleet of Earth observation missions developed by the European Space Agency (ESA) as part of the Copernicus Programme, a cornerstone of Europe's initiative to monitor the Earth and its environment continuously. These satellites are designed to provide a comprehensive, up-to-date picture of our planet's health, behavior, and resources.

The Sentinels are equipped with a range of technologies, such as radar and multispectral imaging instruments, enabling them to observe Earth in various ways. This capability is critical for a wide range of applications, from monitoring climate

change and managing natural disasters to ensuring sustainable agricultural practices and safeguarding ocean health.

The Sentinel fleet currently includes several active satellites, each serving a unique purpose:

Sentinel-1 is a polar-orbiting, all-weather, day-and-night radar imaging mission for land and ocean services. Sentinel-1A was launched on 3 April 2014 and Sentinel-1B on 25 April 2016. Both were taken into orbit on a Soyuz rocket from Europe's Spaceport in French Guiana. The mission ended for Sentinel-1B in 2022 and plans are in force to launch Sentinel-1C as soon as possible.

Sentinel-2 is a polar-orbiting, multispectral high-resolution imaging mission for land monitoring to provide, for example, imagery of vegetation, soil and water cover, inland waterways and coastal areas. Sentinel-2 can also deliver information for emergency services. Sentinel-2A was launched on 23 June 2015 and Sentinel-2B followed on 7 March 2017.

Sentinel-3 is a multi-instrument mission to measure sea-surface topography, sea- and land-surface temperature, ocean colour and land colour with high-end accuracy and reliability. The mission supports ocean forecasting systems, as well as environmental and climate monitoring. Sentinel-3A was launched on 16 February 2016 and Sentinel-3B joined its twin in orbit on 25 April 2018.

Sentinel-4 is a payload devoted to atmospheric monitoring that will be embarked upon a Meteosat Third Generation-Sounder (MTG-S) satellite in geostationary orbit.

Sentinel-5 Precursor — also known as Sentinel-5P — is the forerunner of Sentinel-5 to provide timely data on a multitude of trace gases and aerosols affecting air quality and climate. It has been developed to reduce data gaps between the Envisat satellite — in particular the Sciamachy instrument — and the launch of Sentinel-5. Sentinel-5P was taken into orbit on 13 October 2017 on a Rockot launcher from the Plesetsk Cosmodrome in northern Russia.

Sentinel-5 is a payload that will monitor the atmosphere from polar orbit aboard a MetOp Second Generation satellite.

Sentinel-6 carries a radar altimeter to measure global sea-surface height, primarily for operational oceanography and for climate studies. The first satellite was launched into orbit on 21 November 2020 on a SpaceX Falcon 9 rocket from the Vandenberg Air Force Base in California, US.

The data collected by these satellites are invaluable for scientists, policymakers, and businesses, helping them make informed decisions to protect the environment and manage resources efficiently. With open access to Sentinel data, anyone with the right tools and knowledge can tap into this wealth of information for research, innovation, or practical applications.

In this tutorial, we will guide you through the process of connecting to the Sentinel API, querying for satellite data, and downloading it for your specific use case. Whether you are a researcher, student, or just a satellite data enthusiast, you will find this tutorial a practical starting point to explore the vast potential of Sentinel satellite data. Let's get started!

1. Installing Sentinelsat package

`!pip install sentinelsat` in a Jupyter notebook cell, it will install the `sentinelsat` package in your current Python environment, making it available for import and use in your notebook. This is necessary because Google Colab does not include the `sentinelsat` package in its default environment.

```
# !: Execute shell command in Jupyter notebook/Google Colab
# pip: Python package manager used for installing packages
# install: Command for pip to install a specific package
# sentinelsat: Name of the package to be installed
!pip install sentinelsat
```

2. Importing necessary modules and functions

```
# from sentinelsat: Import specific functions from the 'sentinelsat' package
# SentinelAPI: Class used to interact with the Copernicus Open Access Hub API
# read_geojson: Function to read a GeoJSON file and convert its contents
# geojson_to_wkt: Function to convert GeoJSON geometry to WKT (Well-Known Text)
from sentinelsat import SentinelAPI, read_geojson, geojson_to_wkt

# from datetime: Import the 'date' class from Python's 'datetime' module
# date: Used for handling dates, allows creating date objects and working with
from datetime import date
```

3. Create a SentinelAPI object for accessing the Sentinel data

Before proceeding to this step, you must create your account on this site. Save your username and password to use in the code.

```
# SentinelAPI: This is a class from the 'sentinelsat' module.
# It's used to create an object that interacts with the Copernicus Open Access
# 'your.username': Your username for accessing the Sentinel data.
# 'N12345678': Your password for accessing the Sentinel data.
# 'https://finhub.nsdc.fmi.fi': The URL of the API endpoint you are accessing.
# This line creates an instance of the SentinelAPI class with your credentials
# allowing you to interact with the Sentinel data through the API.
api = SentinelAPI('your.username', 'N12345678', 'https://finhub.nsdc.fmi.fi')
```

4. Setup and Execute Sentinel Data Query and Download

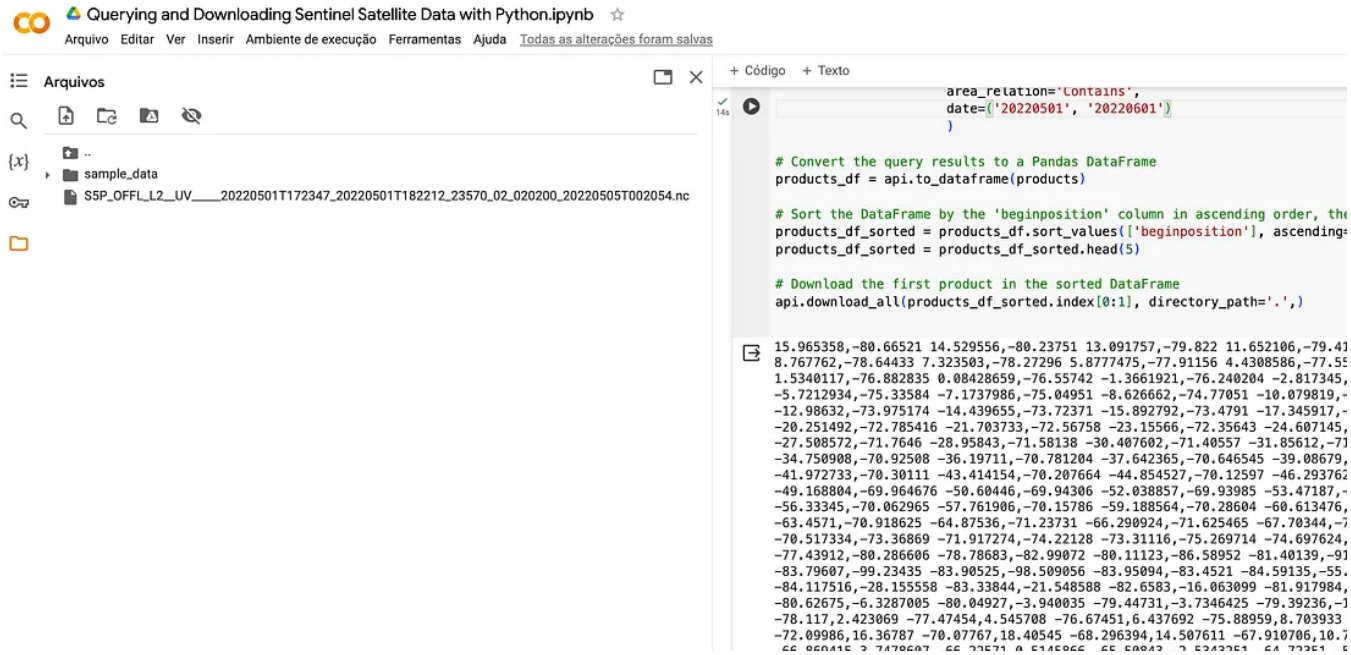
Each comment explains the purpose and action of the corresponding line of code, providing a clear understanding of each step in the process of querying and downloading Sentinel satellite data.

The parameter `producttype='L2__UV____'` in the `api.query()` function specifies the type of Sentinel product you are querying. In this case, it is the Level-2 ultraviolet (UV) data. The exact product type code ('L2__UV____') corresponds to a specific dataset provided by the Sentinel missions. You will need to refer to Sentinel's product specification documents to understand the exact nature of this and other product types.

```
# Define coordinates for the area of interest
longitude   = -54.7  # Central longitude of the area of interest
latitude    = -2.0   # Central latitude of the area of interest

# Define the delta (range) for latitude and longitude
delta_lat   = 0.2  # Latitude range (+/-) to create the bounding box
delta_lon   = 0.2  # Longitude range (+/-) to create the bounding box

# Calculate coordinates for the corners of the bounding box
lat_c1     = latitude + delta_lat  # Top latitude of the bounding box
lat_c2      = latitude - delta_lat  # Bottom latitude of the bounding box
lon_c1     = longitude + delta_lon  # Right longitude of the bounding box
lon_c2      = longitude - delta_lon  # Left longitude of the bounding box

# Define a footprint (area of interest). Currently set as a single point.
footprint = 'POINT (%s %s)' % (longitude, latitude)  # WKT format for a single

# Query the API for Sentinel products within the defined footprint and time ran
products = api.query(footprint,
                     producttype='L2__UV____',  # Specified type of Sentinel p
                     area_relation='Contains',  # Spatial relation between the
                     date=('20220501', '20220601')  # Time range for the query
                     )

# Convert the query results to a Pandas DataFrame
products_df = api.to_dataframe(products)  # Easier manipulation and analysis of
```

```
# Sort the DataFrame by the 'beginposition' column in ascending order, then kee
products_df_sorted = products_df.sort_values(['beginposition'], ascending=[True
products_df_sorted = products_df_sorted.head(5)  # Limit the results to top 5

# Download the first product in the sorted DataFrame
api.download_all(products_df_sorted.index[0:1], directory_path='.')  # Download
```

Running the cell above will download the file or files, depending on the period you set. After a few seconds or minutes, depending on your internet connection, the file(s) will be downloaded to your machine (if you use Spyder, VScode, etc) or to your collab space.



To work with netCDF (`.nc`) data, including data from Sentinel satellites, one can use various software tools and programming libraries. Here's a brief overview:

Programming Libraries (Python, R, etc.):

- Python libraries like `netCDF4` , `xarray` , and `matplotlib` are commonly used for opening, analyzing, and visualizing netCDF data.
- R also has packages like `ncdf4` and `raster` for handling netCDF files.

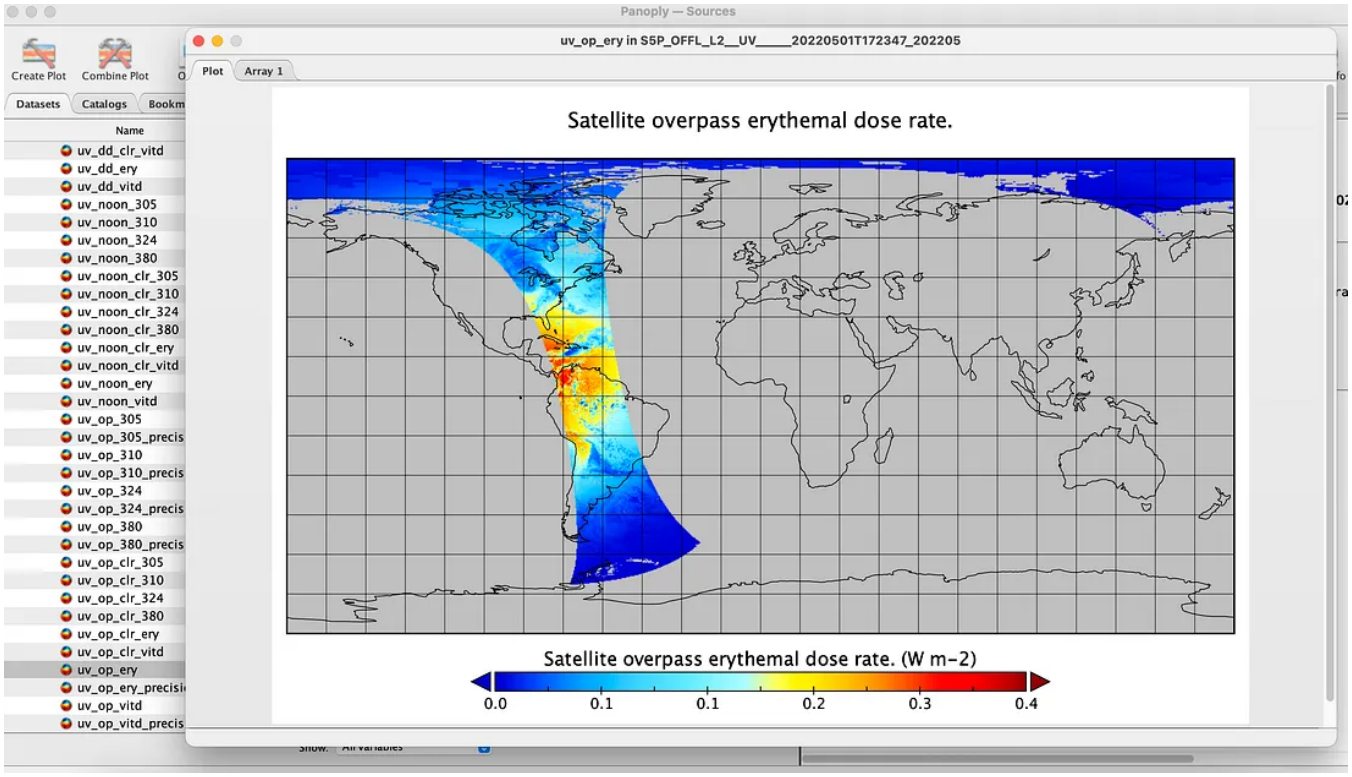
GIS **Software:**

- Geographic Information System (GIS) software like QGIS can open and display netCDF files, allowing for spatial analysis and mapping.

NASA's Panoply:

- Panoply, developed by NASA, is a user-friendly tool that can open netCDF files, allowing users to view data slices, create plots, and export images or data in various formats.
- It doesn't require programming skills and is a good choice for quick visualization

and exploration of data.



The file downloaded in the example of this code opened with Panoply.

Each of these methods offers different capabilities, ranging from simple visualization (Panoply) to complex data analysis (Python, R, GIS software). The choice of tool depends on the user's specific needs and technical expertise.

Conclusions

In this tutorial, we've navigated through the process of accessing and downloading Sentinel satellite data using Python. Starting from establishing a connection with the Sentinel API, we defined a specific geographic area of interest, performed a data query for a certain product type and date range, and sorted and downloaded the relevant data.

By now, you should have a good understanding of how to:

- Connect to the Sentinel API using your credentials.
- Define a precise geographic footprint for data retrieval.
- Query the Sentinel database for specific satellite data products.

- Sort and process the query results using Pandas.
- Download the selected satellite data for your analysis.

This tutorial aimed to empower you with the basic tools and knowledge to leverage the vast repository of Earth observation data provided by the Sentinel satellites. Such skills are invaluable for a wide range of applications, from environmental monitoring and climate research to urban planning and disaster response.

Remember, the Sentinel database is an ever-growing and updating resource, offering new insights into our planet. The techniques you've learned here can be adapted and expanded upon to suit your specific data needs and research goals.

Written by Gabriela Reis

Postdoctoral Researcher | Laboratoire d'Optique Atmosphérique - Université de Lille | Doctor in Physics of the Atmosphere & in Environmental Sciences

Responses (1)

Julio Freitas
Feb 19, 2024

Olá. Ainda não li o artigo, mas preciso entender bem sobre sensoriamento remoto, radares. Sabe onde encontrar material de qualidade? Poderia me responder no e-mail juliofreitas gggguuglllle dot com por favor. Obrigado

More from Gabriela Reis

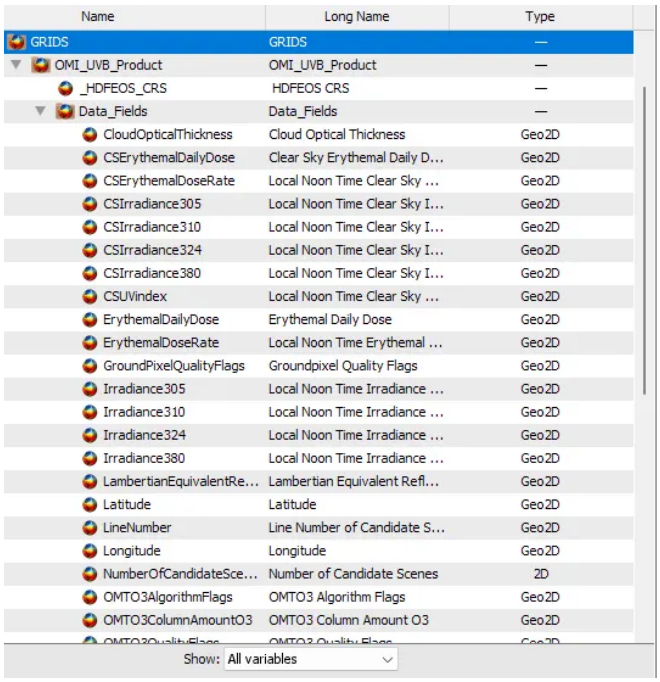
1	2006010101	- SBNT 010100Z 13013KT 9999 7000SE VCSH BKN018...
2	2006010101	- SBNT 010108Z 11012KT 4000 RA SCT007 BKN018 B...
3	2006010101	- SBNT 010130Z 11013KT 2000 +RA SCT008 BKN018 ...
4	2006010102	- SBNT 010200Z 10009KT 9999 VCSH SCT008 SCT017...
...	...	...
152792	2022123020	SBNT 302000Z 11016KT 9999 SCT023 27/22 Q1011
152793	2022123021	SBNT 302100Z 12012KT 9999 FEW021 SCT070 26/22 ...
152794	2022123022	SBNT 302200Z 13010KT 9999 FEW021 26/22 Q1012
152795	2022123023	SBNT 302300Z 12012KT 9999 SCT021 SCT033 26/22 ...

Gabriela Reis

Dealing with cloud cover from METAR DATA using PYTHON

In this tutorial, I will guide you through the steps to open, filter, and perform other necessary actions to access cloud cover patterns...

Dec 21, 2023 53



Group "GRIDS"

In file "OMIAuraL2GOMUVBG2010m0101v0032016m0601t105204.he5"

Group full name: HDFEOS/GRIDS

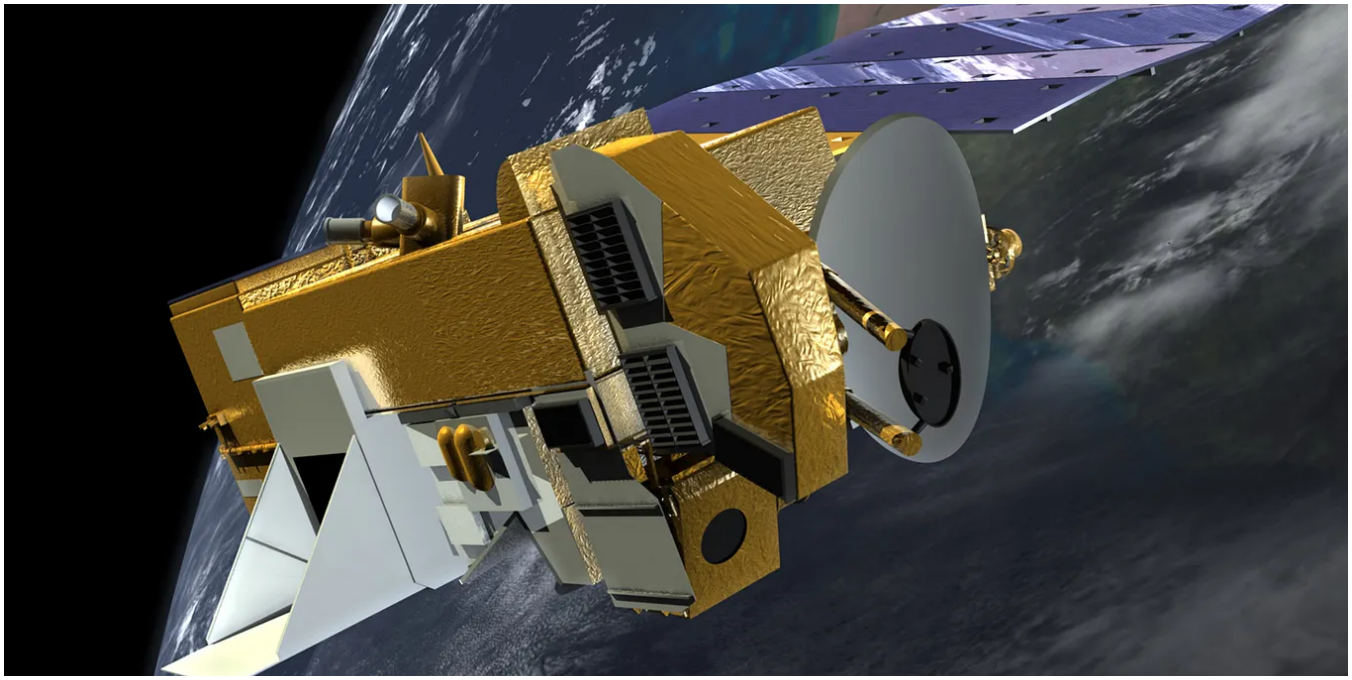
```
group: OMI_UVB_Product {
  dimensions:
    XDim = 1440;
    YDim = 720;
    nCandidate = 15;
  variables:
    short _HDFEOS_CRS;
      :Projection = "HE5_GCTP_GEO";
      :UpperLeftPointMtrs = -1.8E8, -9.0E7; // double
      :LowerRightMtrs = 1.8E8, 9.0E7; // double

  group: Data_Fields {
    variables:
      float CSErythemalDailyDose(nCandidate=15, YDim=720, XDim=1440
        :Offset = 0.0; // double
        :MissingValue = -1.2676506E30f; // float
        :ScaleFactor = 1.0; // double
        :_FillValue = -1.2676506E30f; // float
        :UniqueFieldDefinition = "OMI-Specific";
        :Title = "Clear Sky Erythemal Daily Dose";
        :Units = "J/m2";
        :_ChunkSizes = 1U, 90U, 180U; // uint
        :units = "J/m2";
```

Gabriela Reis

A brief tutorial on how to extract a time series from multiple he5 files

In this post I will show a way to extract time series from multiple he5 files derived from instruments on board NASA's AURA satellite.

Jul 26, 2023 5



Gabriela Reis

Easy Guide to OMI & GOME-2 UV Index Data

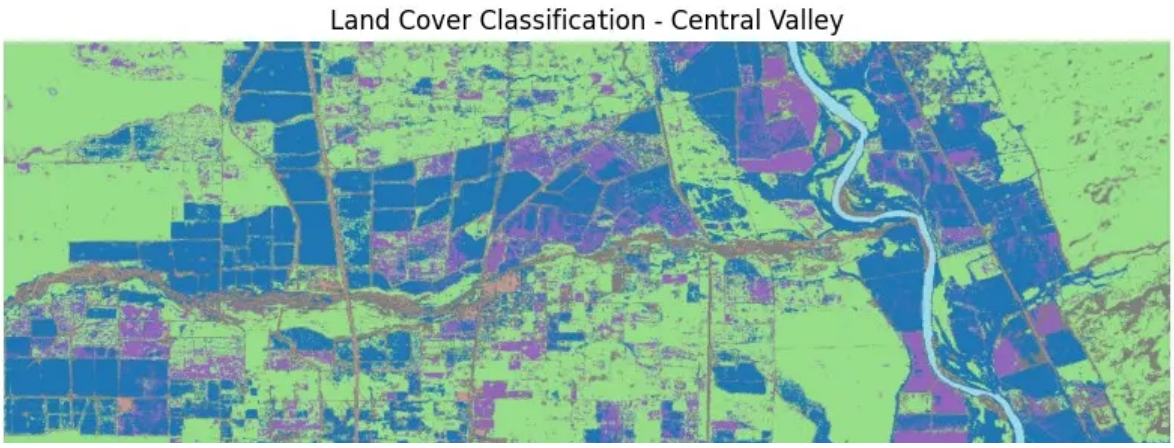
I provide here concise steps to download UV index data from the OMI and GOME-2 satellites instruments. This guide is tailored for ease and...

Dec 20, 2023 50

See all from Gabriela Reis

Recommended from Medium

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 8))
plt.imshow(classified, cmap='tab20')
plt.title('Land Cover Classification - Central Valley')
plt.axis('off')
plt.show()
```



Dr.Preethi Balaji

A Python-Based Workflow for Land Cover Classification Using Geemap, Rasterio, Geopandas, Numpy...

If you've read my previous articles, you'll know I'm a long-time connoisseur of Google Earth Engine (GEE) — especially its JavaScript API...

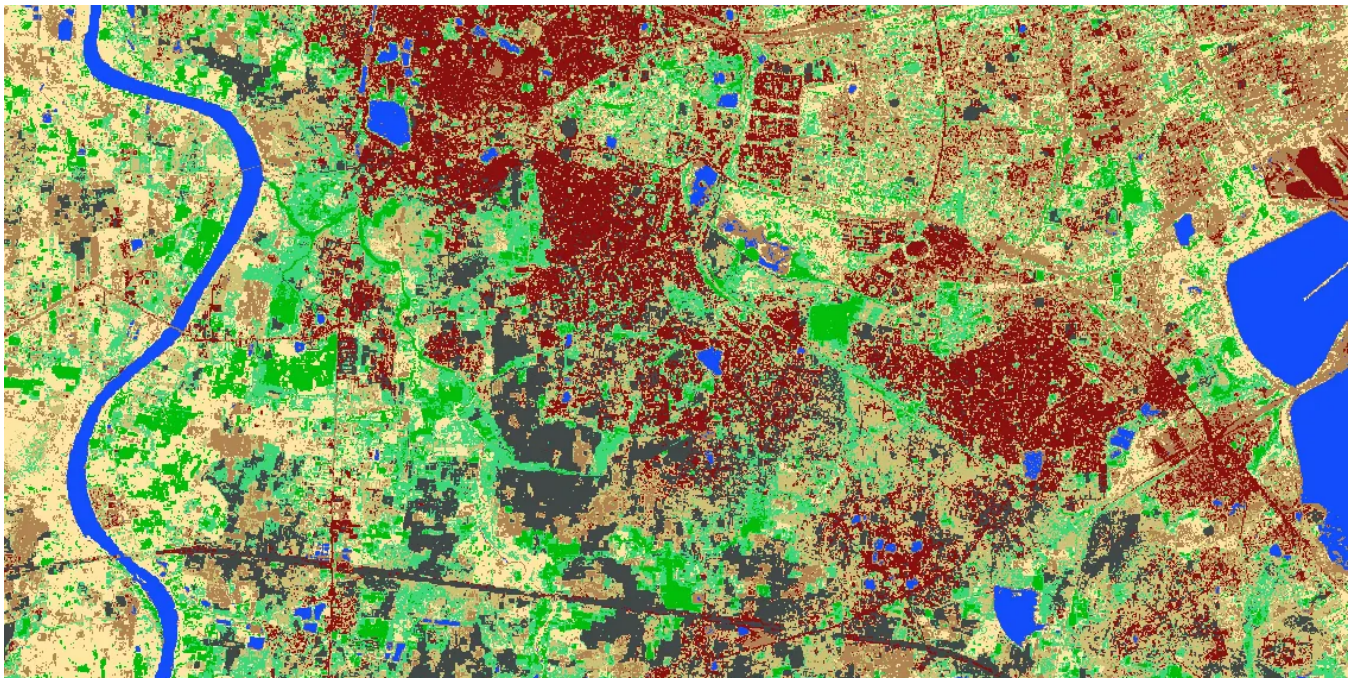
Jul 23, 2025 112 1

Stacy Mwangi

Python Biodiversity Mapping

advanced spatial analytics that identify critical habitats, predict species distributions, and optimize conservation strategies for...

Oct 3, 2025 1 1



Avinash Mehta

Semi-Automatic Land Cover Mapping Using K-Means: A Simple Trick to Save Hours of Labeling

Land use and land cover (LULC) analysis has become increasingly accessible thanks to open Earth Observation data and cloud platforms. Yet...

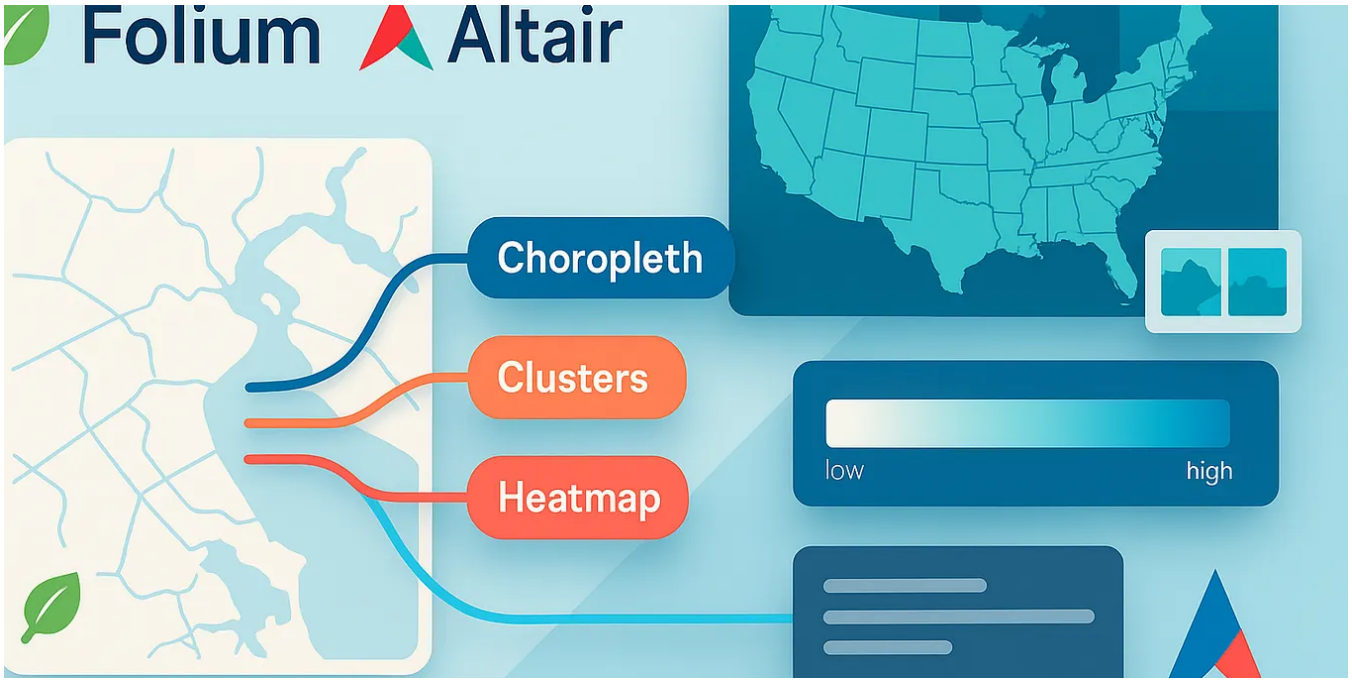
Nov 8, 2025 15 1

PY In Python in Plain English by Tech Brand

Data Cleaning Takes 80% of Time. Here Are My Python Tools

The unsexy work that separates working data scientists from tutorial completers

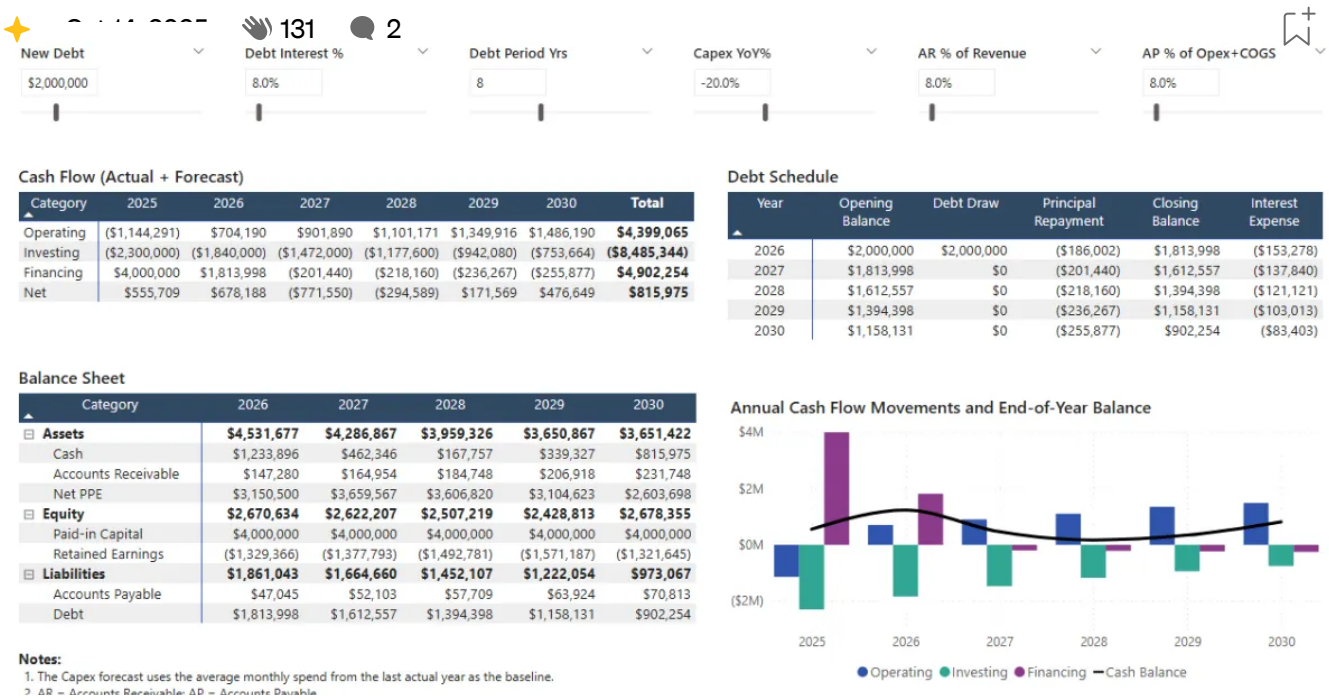
Dec 27, 2025 175 2



Modexa

10 Folium/Altair Map Tricks That Pop in Python

Practical, copy-pasteable patterns to turn raw coordinates into clear, fast, and interactive analytics — without fighting your stack.



Category	2025	2026	2027	2028	2029	2030	Total
Operating	($1,144,291)	$704,190	$901,890	$1,101,171	$1,349,916	$1,486,190	**$4,399,065**
Investing	($2,300,000)	($1,840,000)	($1,472,000)	($1,177,600)	($942,080)	($753,664)	**($8,485,344)**
Financing	$4,000,000	$1,813,998	($201,440)	($218,160)	($236,267)	($255,877)	**$4,902,254**
Net	$555,709	$678,188	($771,550)	($294,589)	$171,569	$476,649	**$815,975**

Year	Opening Balance	Debt Draw	Principal Repayment	Closing Balance	Interest Expense
2026	$2,000,000	$2,000,000	($186,002)	$1,813,998	($153,278)
2027	$1,813,998	$0	($201,440)	$1,612,557	($137,840)
2028	$1,612,557	$0	($218,160)	$1,394,398	($121,121)
2029	$1,394,398	$0	($236,267)	$1,158,131	($103,013)
2030	$1,158,131	$0	($255,877)	$902,254	($83,403)

Category	2026	2027	2028	2029	2030
Assets	**$4,531,677**	**$4,286,867**	**$3,959,326**	**$3,650,867**	**$3,651,422**
Cash	$1,233,896	$462,346	$167,757	$339,327	$815,975
Accounts Receivable	$147,280	$164,954	$184,748	$206,918	$231,748
Net PPE	$3,150,500	$3,659,567	$3,606,820	$3,104,623	$2,603,698
Equity	**$2,670,634**	**$2,622,207**	**$2,507,219**	**$2,428,813**	**$2,678,355**
Paid-in Capital	$4,000,000	$4,000,000	$4,000,000	$4,000,000	$4,000,000
Retained Earnings	($1,329,366)	($1,377,793)	($1,492,781)	($1,571,187)	($1,321,645)
Liabilities	**$1,861,043**	**$1,664,660**	**$1,452,107**	**$1,222,054**	**$973,067**
Accounts Payable	$47,045	$52,103	$57,709	$63,924	$70,813
Debt	$1,813,998	$1,612,557	$1,394,398	$1,158,131	$902,254

In ChallengeJP by Jacek Polewski

Building a Financial Planning and Analysis Dashboard in 8 Steps

Build an end-to-end FP&A dashboard in Microsoft Power BI with integrated P&L, cash flow, balance sheet, and scenario-based forecasting.

6d ago 11

See more recommendations