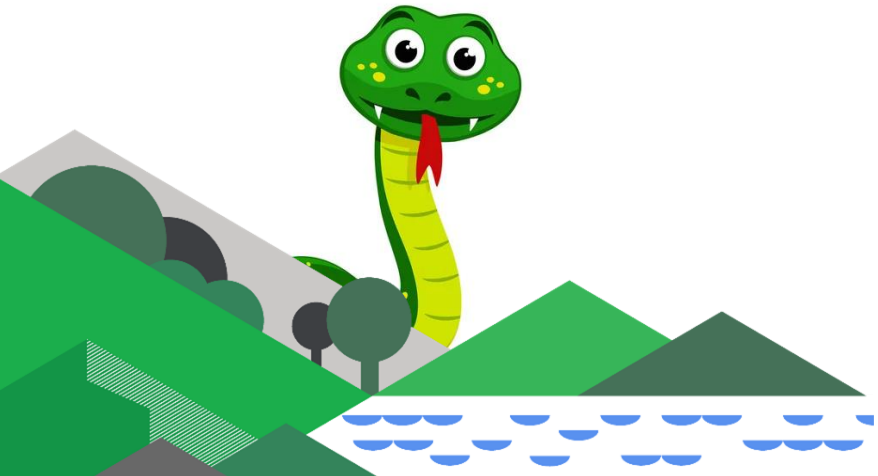
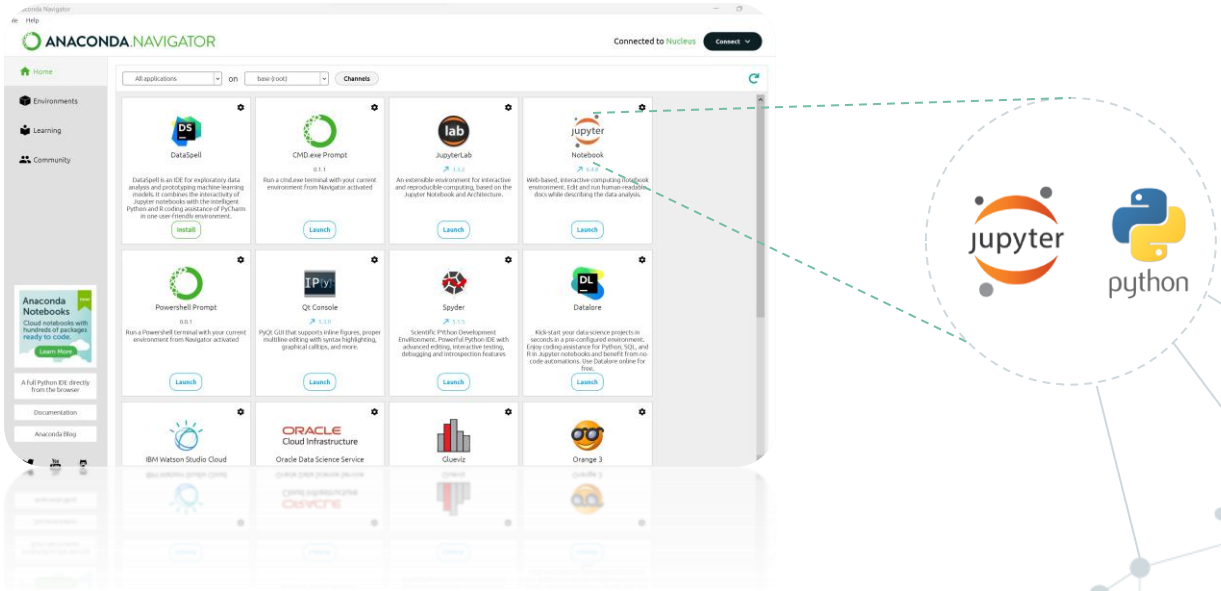
Python for computational sciences

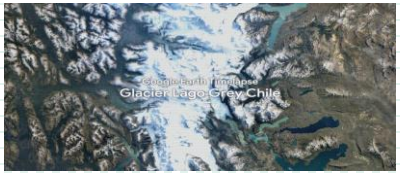


Objective of the project

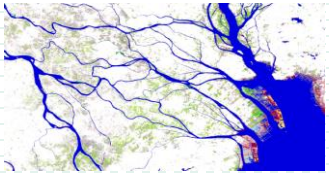
Download, process and analyze satellite images through Python language



Deforestation



Climate change



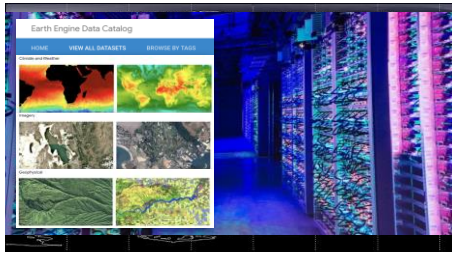
Access to water

Decision-makers need tools

Using these datasets requires lots of computational power

What is Google Earth Engine?

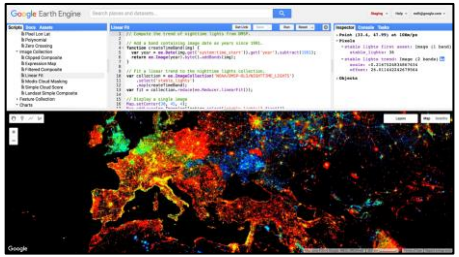
Data Catalog



More than 40 years of historical imagery and scientific data sets

70PB and 800+ curated geospatial datasets, including near-real-time satellite imagery.

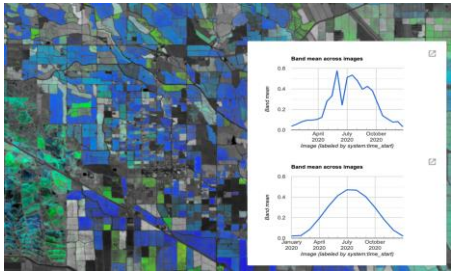
Computation Platform



A powerful tool to analyze and visualize Earth data at scale.

Parallel processing for speed and scale, with machine learning built in.

Visualization



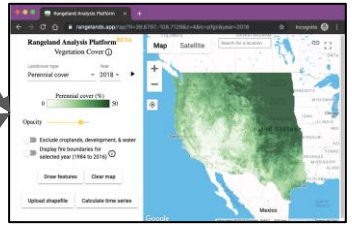
- Computations on images (per pixel)
- Machine learning
- Time series analysis

Earth Engine Backend Servers

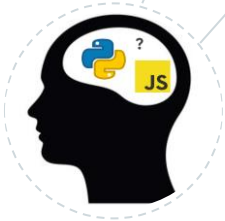
Javascript Client Library

Earth Engine Code Editor

Web (REST) API



Web Application



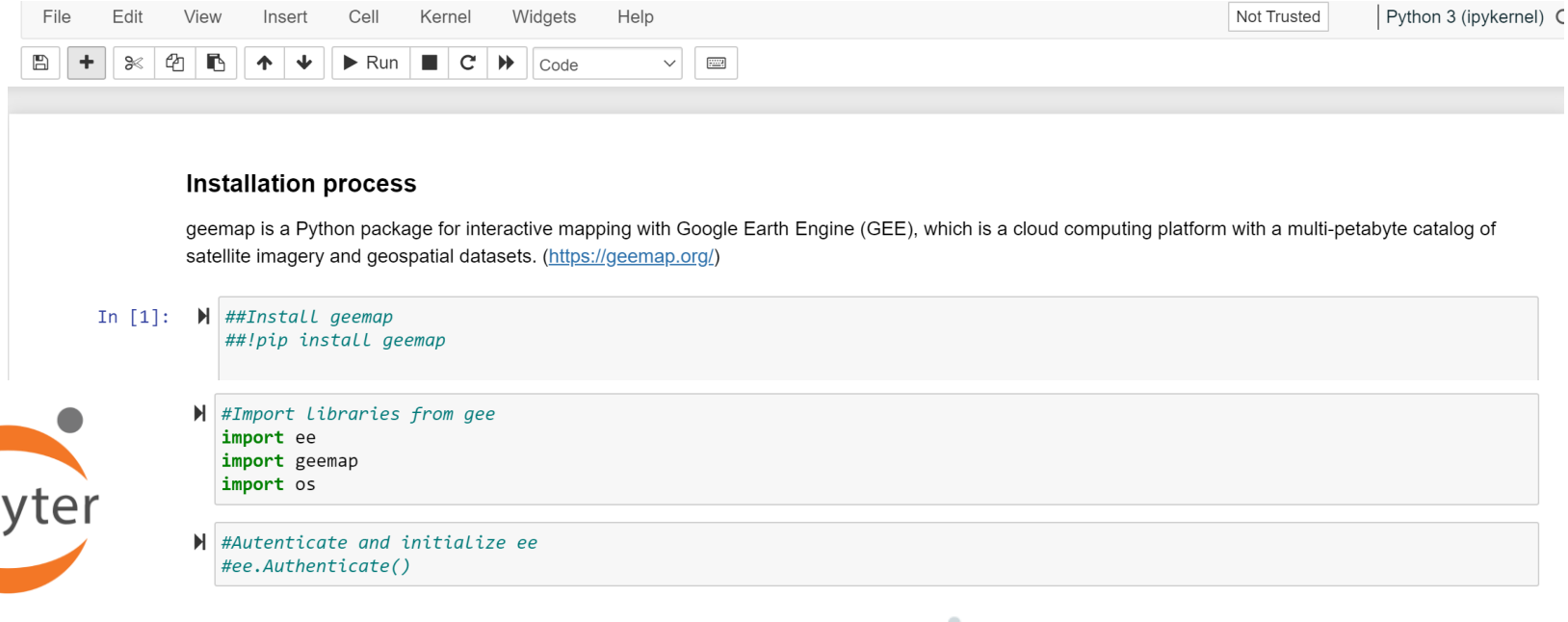
Python Client Library

Jupyter Notebooks

jupyter

geemap: A Python package for interactive mapping with Google Earth Engine

https://doi.org/10.21105/joss.02305



Importing data from the GEE catalog

Earth Engine's public data catalog includes a variety of standard Earth science raster datasets - https://developers.google.com/earth-engine/datasets/catalog

In [4]:
```
Map = geemap.Map()

#Importing country boundaries
countries = ee.FeatureCollection("USDOS/LSIB/2017")
roi = countries.filter(ee.Filter.eq("COUNTRY_NA", "Italy"))

Map.addLayer(roi,{}, "Italy")
Map.centerObject(roi,6)

Map
```

Importing data from the GEE catalog

Earth Engine's public data catalog includes a variety of standard Earth science raster datasets

In [6]:
```
##Set centre
Map2.setCenter(11.924308,42.389864, 15)
```

Importing satellite imagery

In [7]:
```
##Location Point
point = ee.Geometry.Point(11.924308,42.389864)
```

In [8]:
```
##Import satellite images
data = ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED").filterBounds(point);
```

In [9]:
```
##Filtering image
image1 = ee.Image(data.filterDate ("2021-01-01", "2022-08-30").sort("CLOUD_COVERAGE_ASSESSMENT").first())
```

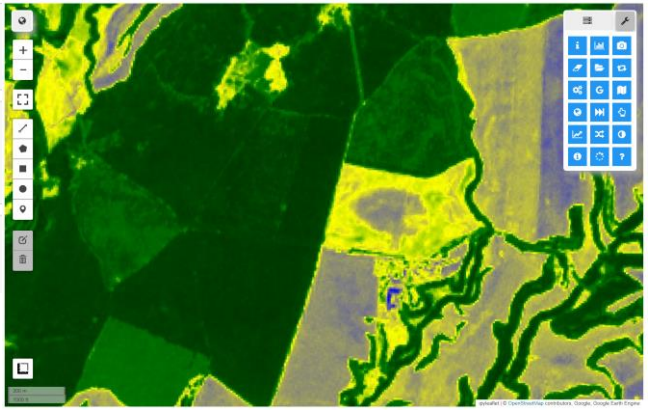
In [10]:
```
vis_params = {'bands': ['B4', 'B3', 'B2'], 'min': 0.0, 'max': 2200, 'opacity': 1.0, 'gamma': 1.0}
```

In [11]:
```
Map2.addLayer(image1,vis_params, "Image")
```

[12]:
```
##Create function of vegetation indices
def getNDVI(image):
    return image.normalizedDifference(["B8", "B4"])

ndvi1 = getNDVI(image1)
```

[13]:
```
##Create params of visualization for spectral indices
visualization = {
    "min":0,
    "max":1,
    "palette":["blue","yellow", "green", "darkgreen", "black"]
}
```

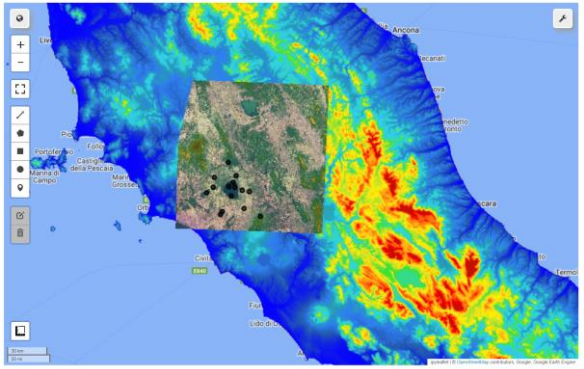
[14]:
```
#Plot Map
Map2.addLayer(ndvi1,visualization, "NDVI")

Map2
```



NDVI helps to differentiate vegetation from other (artificial) land cover types and determine its general condition. It also allows to define and visualize vegetated areas on the map, as well as to detect abnormal changes in the growth process.

Extract pixel values from a satellite image

In [17]:
```
Map3 = geemap.Map()

#Add Dataset from Google Earth Engine Catalog

DEM = ee.Image("CGIAR/SRTM90_V4") ##Digital Elevation Model

#Visualization parameters
vis_params = {
    "min": 0,
    "max": 2000,
    "palette": ['0602ff', '235cb1', '307ef3', '269db1', '30c8e2', '32d3ef', '3ae237',
    'b5e22e', 'd6e21f', 'fff705', 'ffd611', 'ffb613', 'ff8b13', 'ff6e08',
    'ff500d', 'ff0000', 'de0101', 'c21301']
}

Map3.addLayer(DEM,vis_params, "DEM")
Map3.addLayer(S2Bands,{
    "bands":["B4", "B3", "B2"],
    "min": 0,
    "max": 2200}, "Sentinel2")

#Map3.centerObject(roi,6)
Map3.centerObject(point,10)

Map3
```



In [18]:
```
work_dir = os.path.expanduser("~/Downloads")
in_shp = os.path.join(work_dir, "Plots/plots.shp")
```

In [19]:
```
in_fc = geemap.shp_to_ee(in_shp)
Map3.addLayer(in_fc, {}, "plots")
```

In [ ]:
```
out_shp = os.path.join(work_dir, "dem.shp")
geemap.extract_values_to_points(in_fc, DEM, out_shp)
```

In [27]:
```
out_csv = os.path.join(work_dir, "Sentinel2bands.csv")
geemap.extract_values_to_points(in_fc, S2Bands, out_csv)
```

```
Generating URL ...
Downloading data from https://earthengine.googleapis.com/v1alpha/projects/earthengine-legacy/tables/0844f17e03d96a2d0a69da6f69a37b51-937b994ceb1b48f2a459ec56657843bf:getFeatures
Please wait ...
Data downloaded to C:\Users\Alexander\Downloads\Sentinel2bands.csv
```

	A	B	C	D	E	F	G	H	I	J	K	L	M
	B2	B3	B4	B5	B6	B7	B8	B8A	B11	B12	system:index	id	clase
1	384	322	167	163	153	166	140	142	60	54	0	1	water
2	385	310	146	134	144	143	139	133	55	40	1	2	water
3	406	342	186	175	179	178	154	164	77	65	2	3	water
4	216	188	81	68	53	60	55	48	121	96	3	4	water
5	400	339	160	165	179	191	154	165	70	63	4	5	water
6	346	582	368	1012	3075	3769	3792	4073	2025	848	5	6	forest
7	337	606	357	1024	3185	3943	4201	4311	2174	885	6	7	forest
8	372	794	376	1208	4512	5711	5704	6060	2505	1006	7	8	forest
9	338	564	345	980	3007	3606	3664	3883	2028	869	8	9	forest
10	346	636	364	1073	3323	4023	3839	4245	2143	916	9	10	forest
11	1652	2298	3288	3739	3848	4057	4091	4333	4227	2909	10	11	croop
12	1798	2578	3700	4312	4402	4677	4652	5083	4846	3266	11	12	croop
13	896	1268	1686	2193	2782	3024	3152	3235	2558	1776	12	13	croop
14	825	1166	1682	1878	1972	2049	2097	2252	2641	2231	13	14	croop
15	1550	2220	3304	3652	3834	4093	4340	4459	4563	3097	14	15	croop

CSV data import and graph plotting

In [20]:
```
import pandas as pd
import seaborn as sns
```

In [21]:
```
csvdata = pd.read_csv("Sentinel2bands.csv")
csvdata.head()
```

Out[21]:

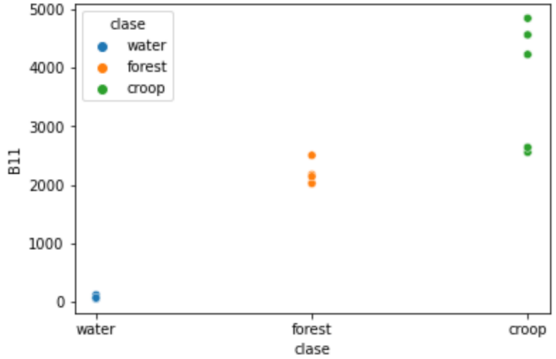
	B2	B3	B4	B5	B6	B7	B8	B8A	B11	B12	system:index	id	clase
0	384	322	167	163	153	166	140	142	60	54	0	1	water
1	385	310	146	134	144	143	139	133	55	40	1	2	water
2	406	342	186	175	179	178	154	164	77	65	2	3	water
3	216	188	81	68	53	60	55	48	121	96	3	4	water
4	400	339	160	165	179	191	154	165	70	63	4	5	water

In [23]:
```
first_column = csvdata.pop('clase')

csvdata.insert(0, 'clase', first_column) ## Reordering the columns

dataplot = csvdata.iloc[0: , :11] #Selecting a range of columns

dataplot.head()

#display(dataplot)
```

Out[23]:

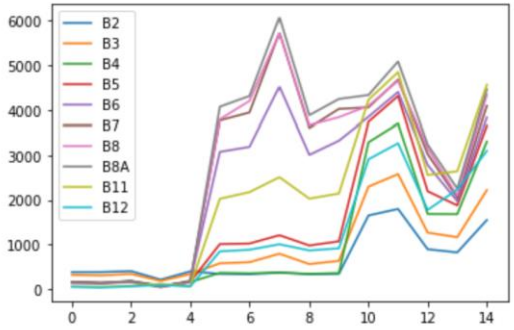
	clase	B2	B3	B4	B5	B6	B7	B8	B8A	B11	B12
0	water	384	322	167	163	153	166	140	142	60	54
1	water	385	310	146	134	144	143	139	133	55	40
2	water	406	342	186	175	179	178	154	164	77	65
3	water	216	188	81	68	53	60	55	48	121	96
4	water	400	339	160	165	179	191	154	165	70	63

In [24]:
```
p=sns.scatterplot(data=dataplot,x="clase",y = "B11", hue="clase")
```



In [28]:
```
dataplot.plot()
```

Out[28]: <AxesSubplot:>



Thanks!