Analyzing the Amazon Deforestation with Machine Learning and the Google Earth Engine
Part I

Teachers: Maria Antonia Brovelli and Vasil Yordanov
maria.brovelli@polimi.it vasil.yordanov@polimi.it

Assistants: Ahmed Abdalgade Ahmed Eisaa, Alessando Austoni, Juan Francisco Amieva, Julia Anna Leonardi, Ahmed Omer Ahmed Mukhtar, Nikolina Zallemi, Martina Giovanna Esposito and Ahmed Mohamed Eltahir Yassin

Politecnico di Milano – DICA | GEOLab

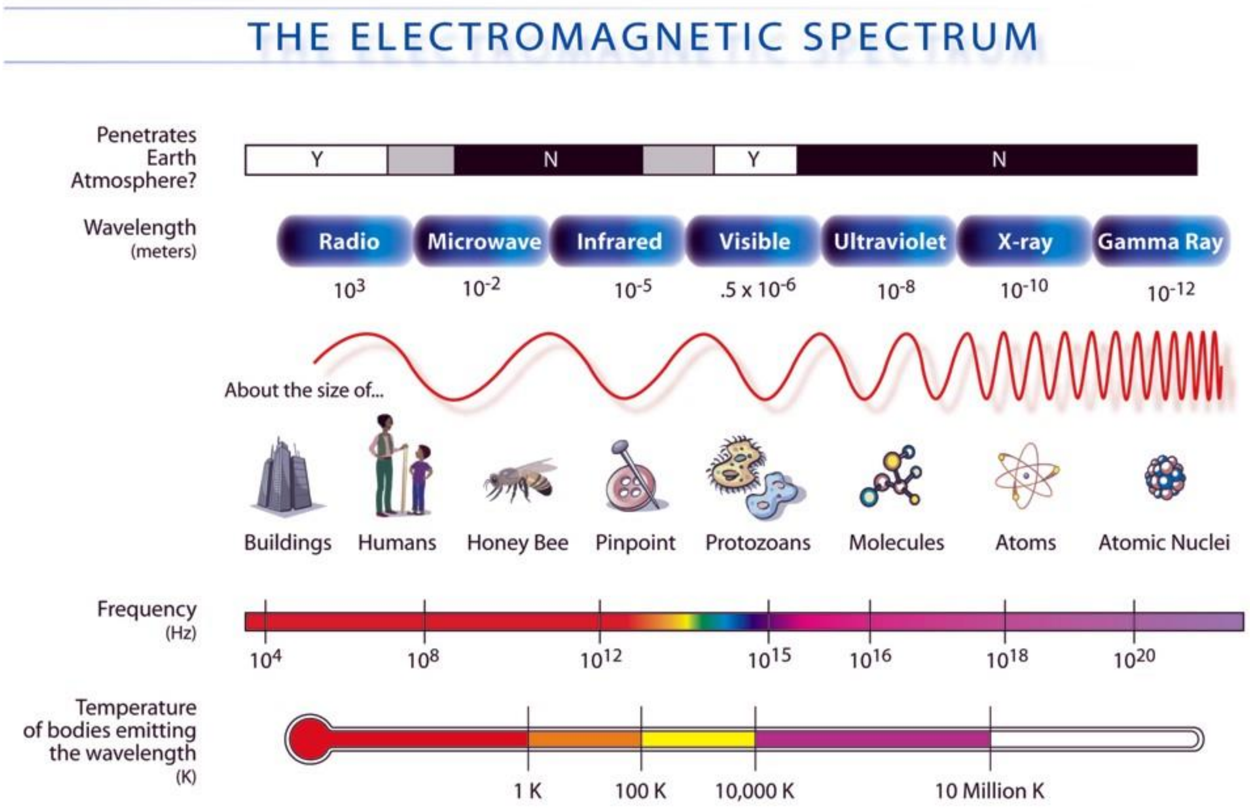
15 March 2022


21 March
International Day of Forests
UNEP/ Will Baxter

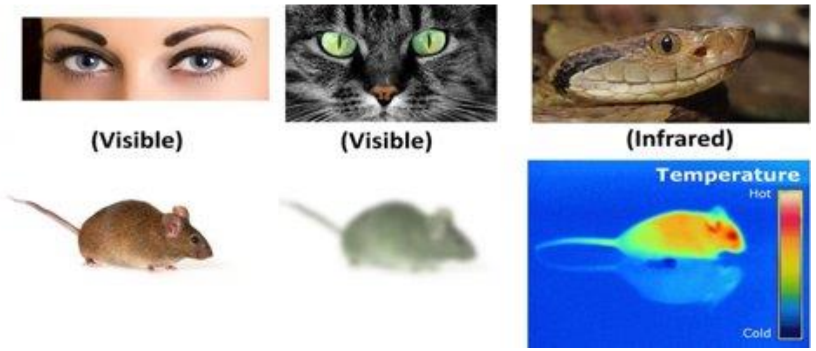
01

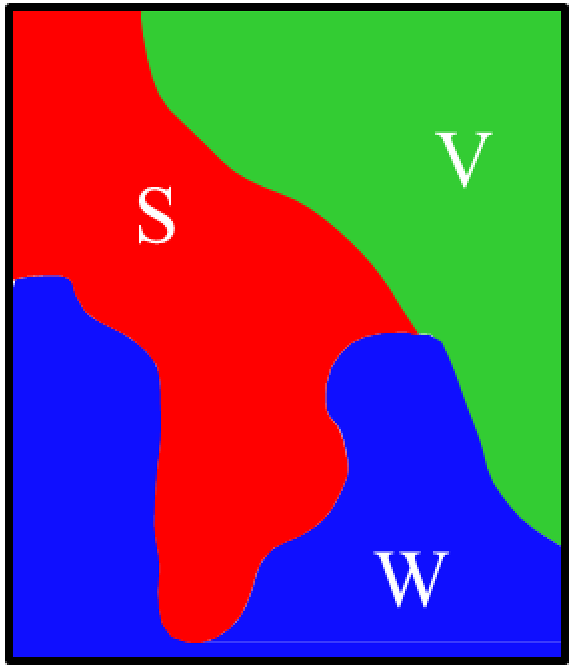
Brief introduction to multispectral images

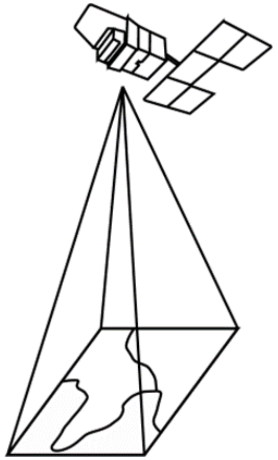
Multi-spectral images in a nutshell



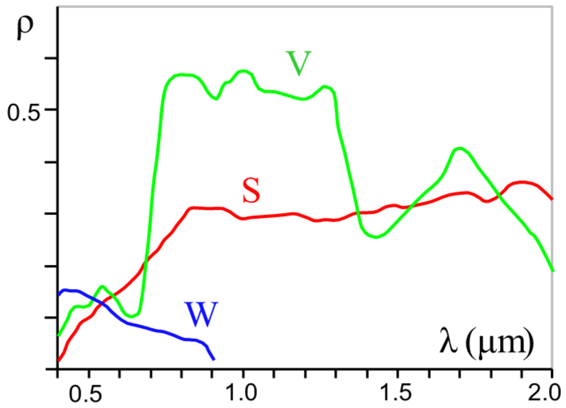
Multi-spectral images in a nutshell







Spectral signature

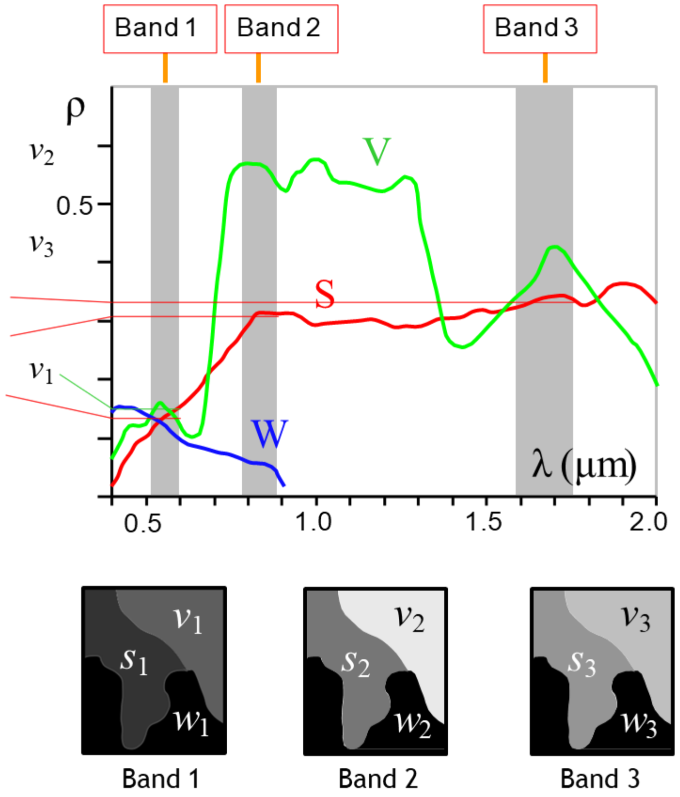


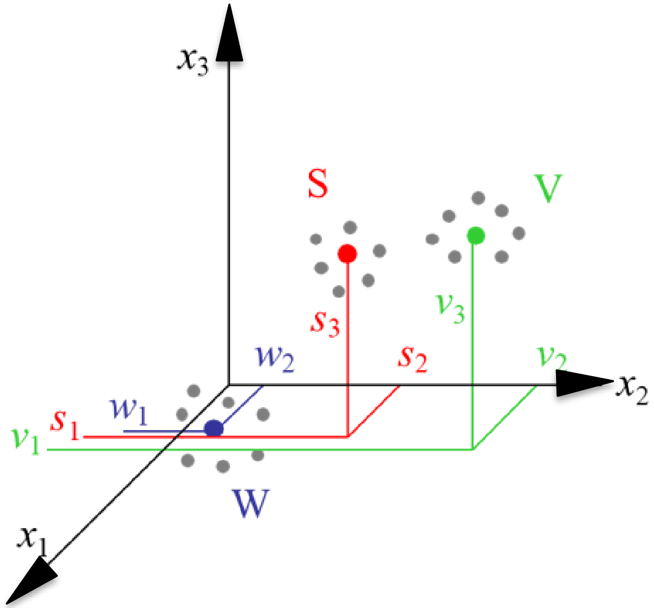
Multi-spectral images in a nutshell

Sentinel-2 Bands	Central Wavelength [micrometers]	Res[m]
Band 1 - Coastal aerosol	0.443	60
Band 2 - Blue	0.49	10
Band 3 - Green	0.56	10
Band 4 - Red	0.665	10
Band 5 - Vegetation Red Edge	0.705	20
Band 6 - Vegetation Red Edge	0.74	20
Band 7 - Vegetation Red Edge	0.783	20
Band 8 - NIR	0.842	10
Band 8A - Vegetation Red Edge	0.865	20
Band 9 - Water vapor	0.945	60
Band 10 - SWIR - Cirrus	1.375	60
Band 11 - SWIR	1.61	20
Band 12 - SWIR	2.19	20

Landsat 8 Bands	Central Wavelength [micrometers]	Res[m]
Band 1 - Coastal aerosol	0.43-0.45	30
Band 2 - Blue	0.45-0.51	30
Band 3 - Green	0.53-0.59	30
Band 4 - Red	0.64-0.67	30
Band 5 – Near Infrared (NIR)	0.85-0.88	30
Band 6 – SWIR 1	1.57-1.65	30
Band 7 – SWIR 2	2.11-2.29	30
Band 8 – Panchromatic	0.50-0.68	15
Band 9 – Cirrus	1.36-1.38	30
Band 10 – Thermal Infrared (TIRS) 1	10.6-11.19	100
Band 11 – Thermal Infrared (TIRS) 2	11.50-12.51	100

Multi-spectral images in a nutshell



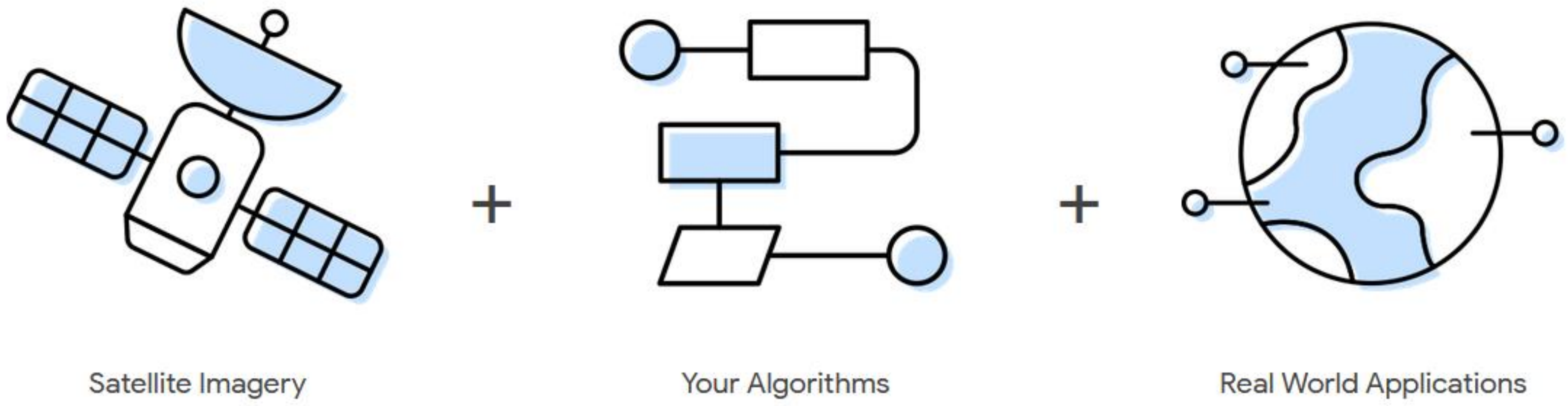


02

Introduction to Google Earth Engine

Meet Earth Engine

Google Earth Engine combines a multi-petabyte catalog of satellite imagery and geospatial datasets with planetary-scale analysis capabilities. Scientists, researchers, and developers use Earth Engine to detect changes, map trends, and quantify differences on the Earth's surface. Earth Engine is now available for commercial use, and remains free for academic and research use.



The Earth Engine Data Catalog

Landsat & Sentinel
10-30m, weekly

MODIS
250m daily

Vector Data
WDPA, Tiger

Terrain & Land Cover

Weather & Climate
NOAA NCEP, OMI, …

… and upload your own vectors and rasters

> 200 public datasets

> 4000 new images every day

> 5 million images

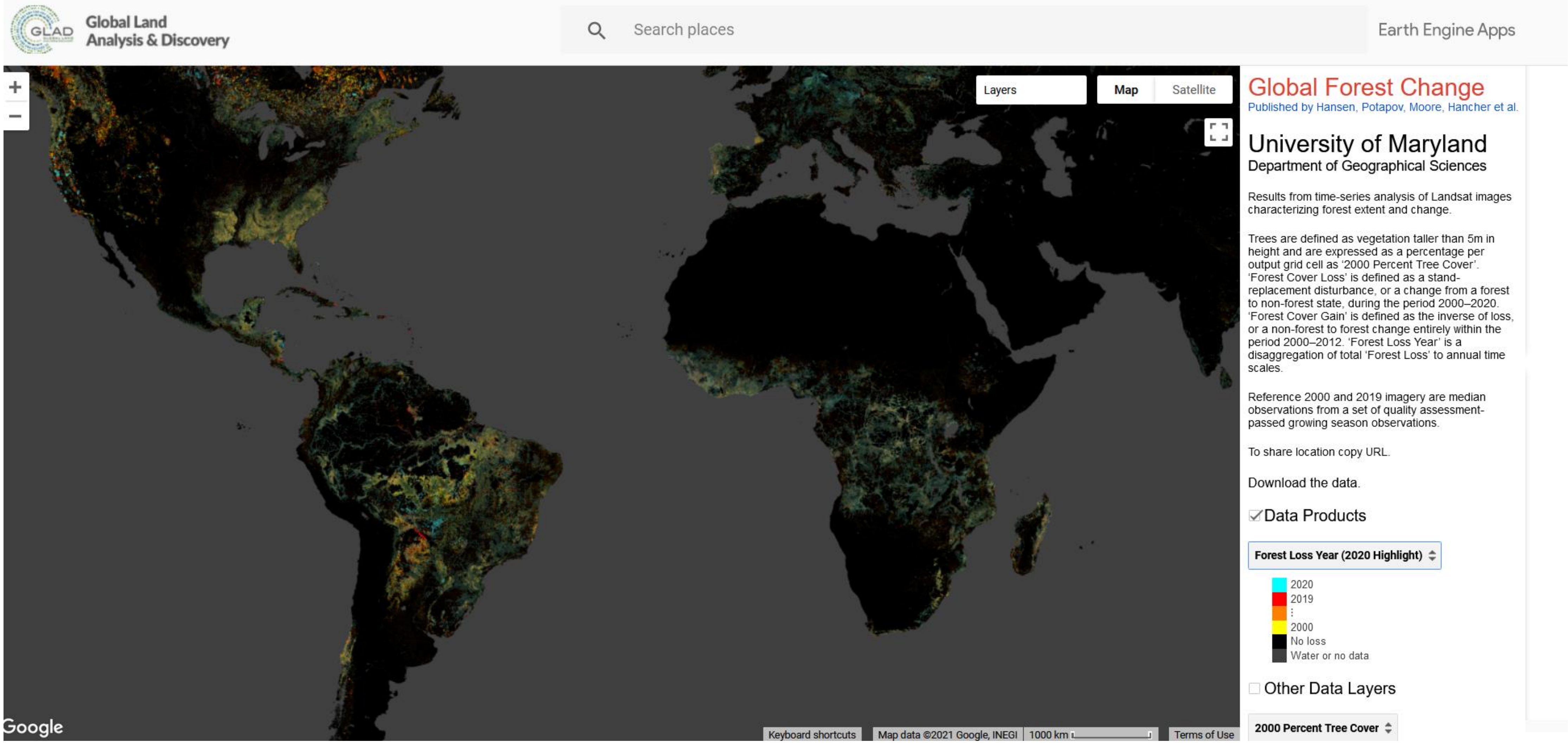
> 7 petabytes of data

Google Infrastructure

Infrastructure by Sumit Saengthong from NounProject.com

Your geospatial workflow in JavaScript or Python

Google Earth Engine



https://glad.earthengine.app/view/global-forest-change

03

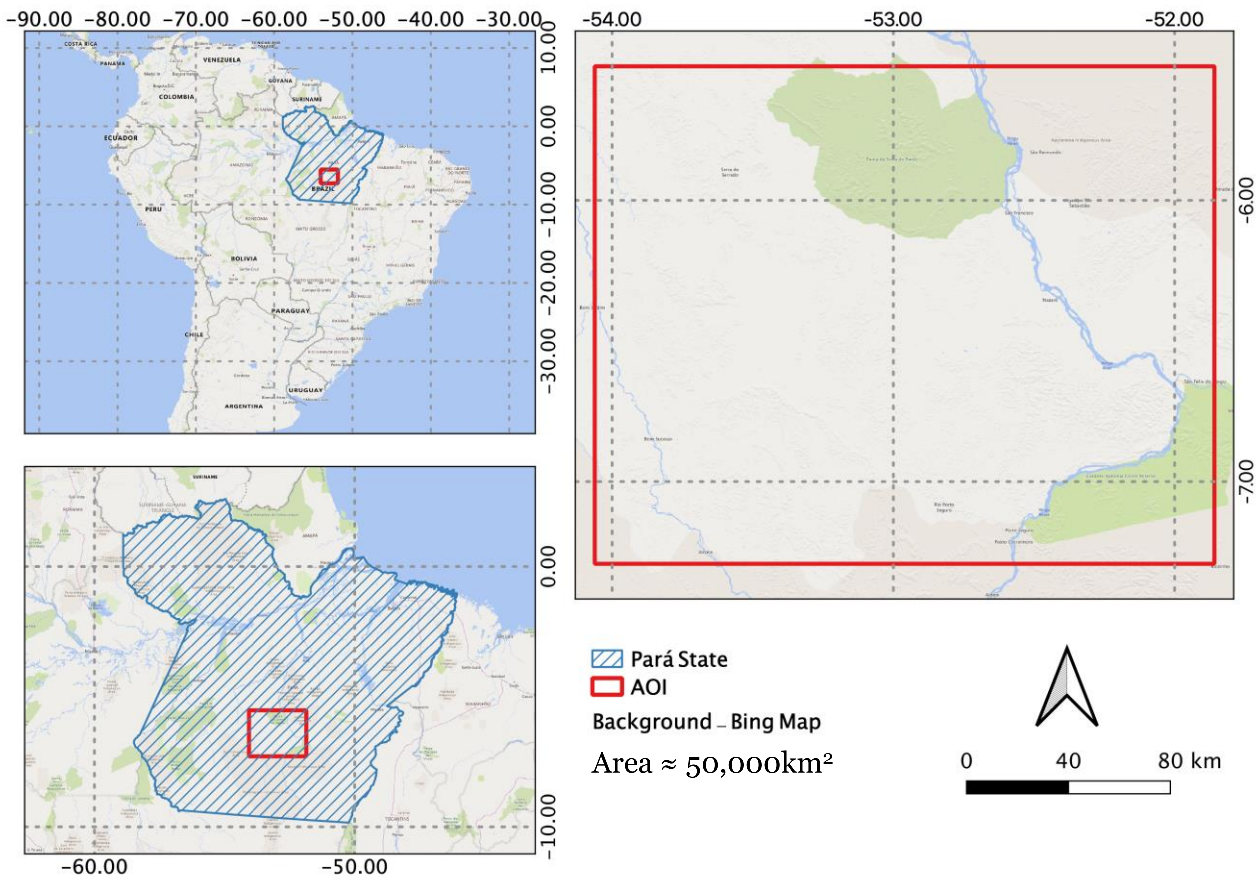
What, where and how

What

Deforestation between the years 2015 - 2019

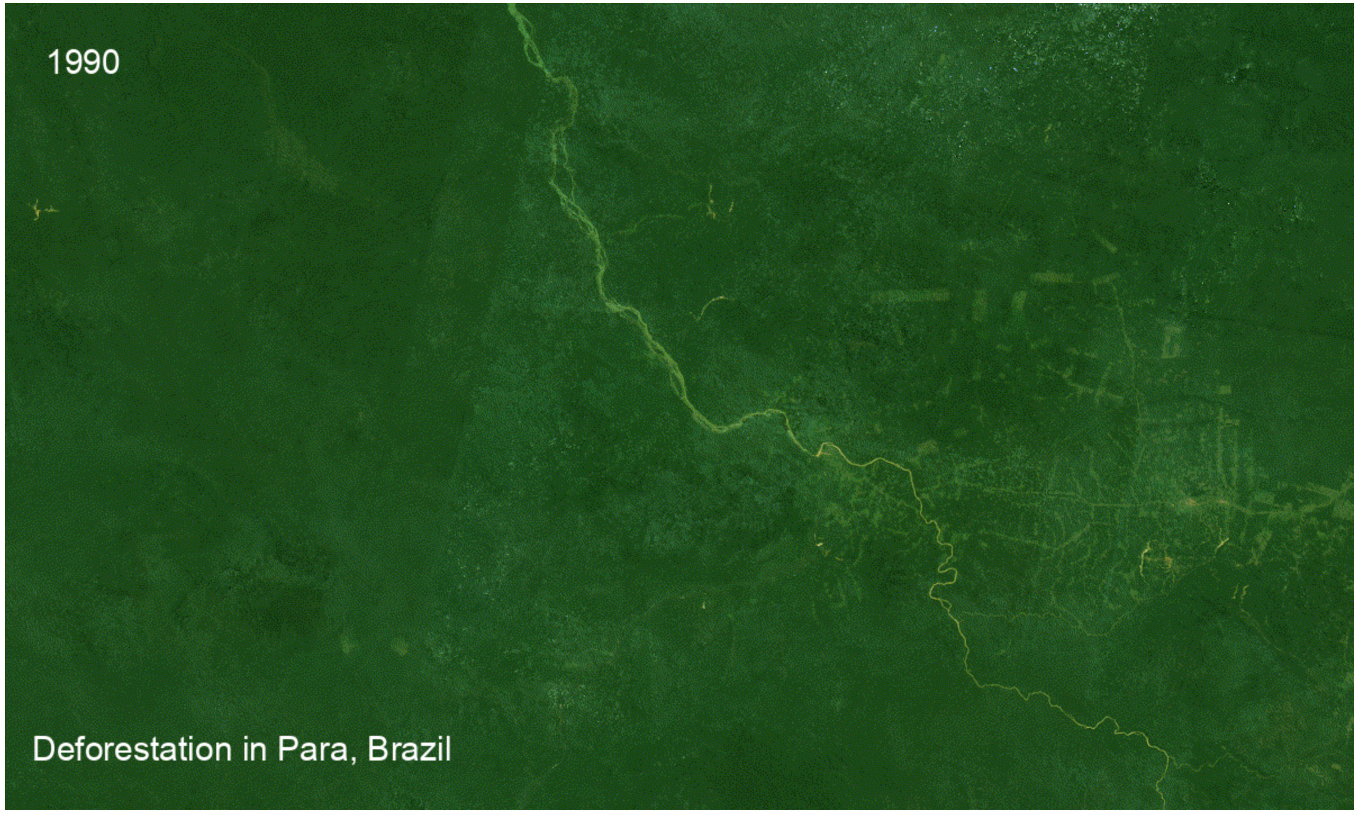
Case study and aim

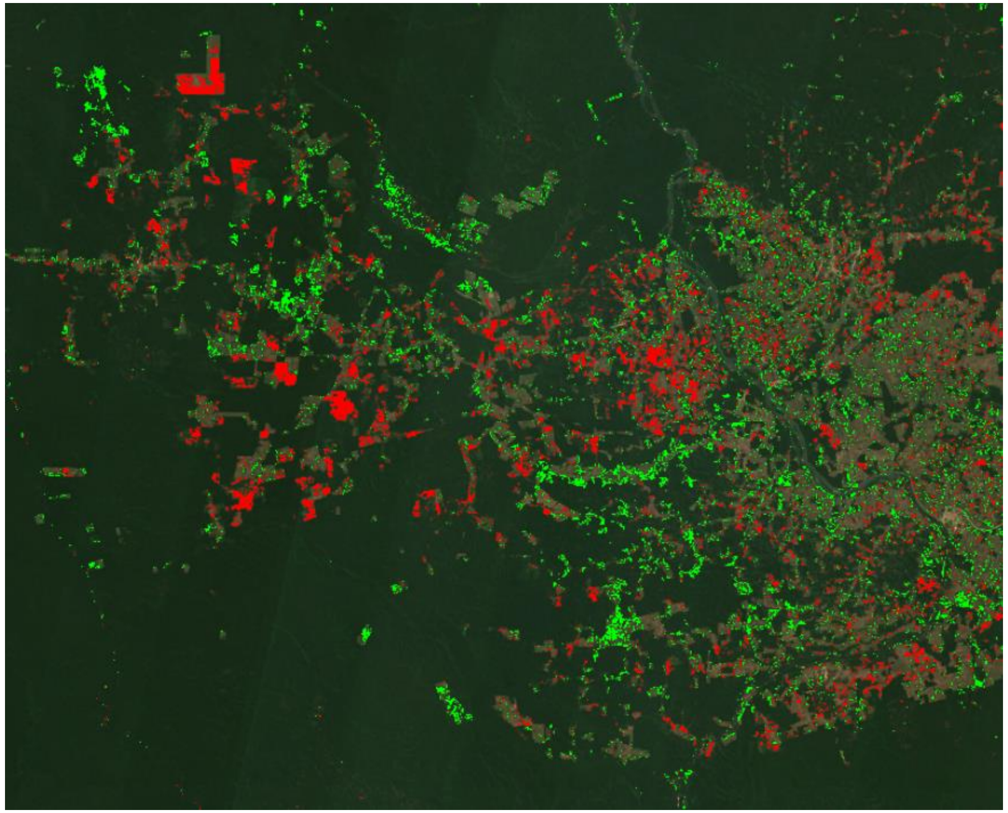
Area of interest – Parà state, Brazil



Case study and aim

Aim – map forest loss and gain in the period 2015 - 2019





Forest loss Forest gain

Workflow

Google Earth Engine Introduction

1. Data Preparation

2. Classification

3. Model Evaluation

4. Forest Loss/Gain Computation

Part I (15 March 2022) → today

Part II (29 March 2022)

Workflow

1. Data Preparation

2. Classification

3. Model Evaluation

4. Forest Loss/Gain Computation

- Define Area of Interest (AOI)
- Create cloud mask
- Define two periods of the analysis
- Define data to be used
- Apply filters to the data

Workflow

1. Data Preparation

2. Classification

3. Model Evaluation

4. Forest Loss/Gain Computation

Define areas that will be used for model training and testing

↓

Train and apply classifier for both periods

Workflow

1. Data Preparation

2. Classification

3. **Model Evaluation**

4. Forest Loss/Gain Computation

Compute error matrix

↓

Compute evaluation metrics
(overall accuracy, Cohen's Kappa)

Workflow

1. Data Preparation

2. Classification

3. Model Evaluation

4. Forest Loss/Gain Computation

Compute the difference between the two classified images

↓

Mask the forest loss/gain areas

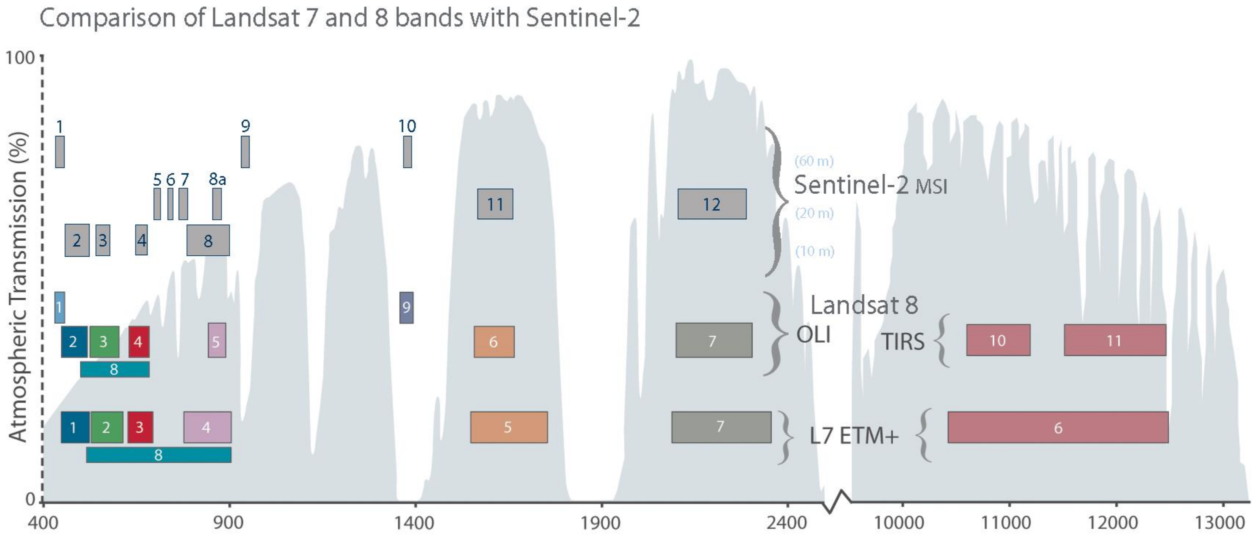
↓

Compute the total and absolute areas

Data

Source: esa.int

	Landsat-8	Sentinel-2
Active years	2013 – present	2015 – present
Spatial resolutions and used bands	30 m/pix (R,G,B, NIR)	10 m/pix (R,G,B, NIR)
Years of interest	2015	2019



Source: usgs.gov

Methods

Classification scheme:

```
var training = image.sample(region, scale)

var classifier = ee.Classifier.cart().train(training)

var result = image.classify(classifier)

var confusionMatrix = classifier.confusionMatrix()
var accuracy = confusionMatrix.accuracy()
```

From Noel Gorelick's Classification and Clustering presentation

Classifiers' mode:

classifier.setOutputMode(mode)

- *Classification* - Discrete input/output classes
- *Regression* - Continuous valued output
- *Probability* - binary classifiers only (for now)

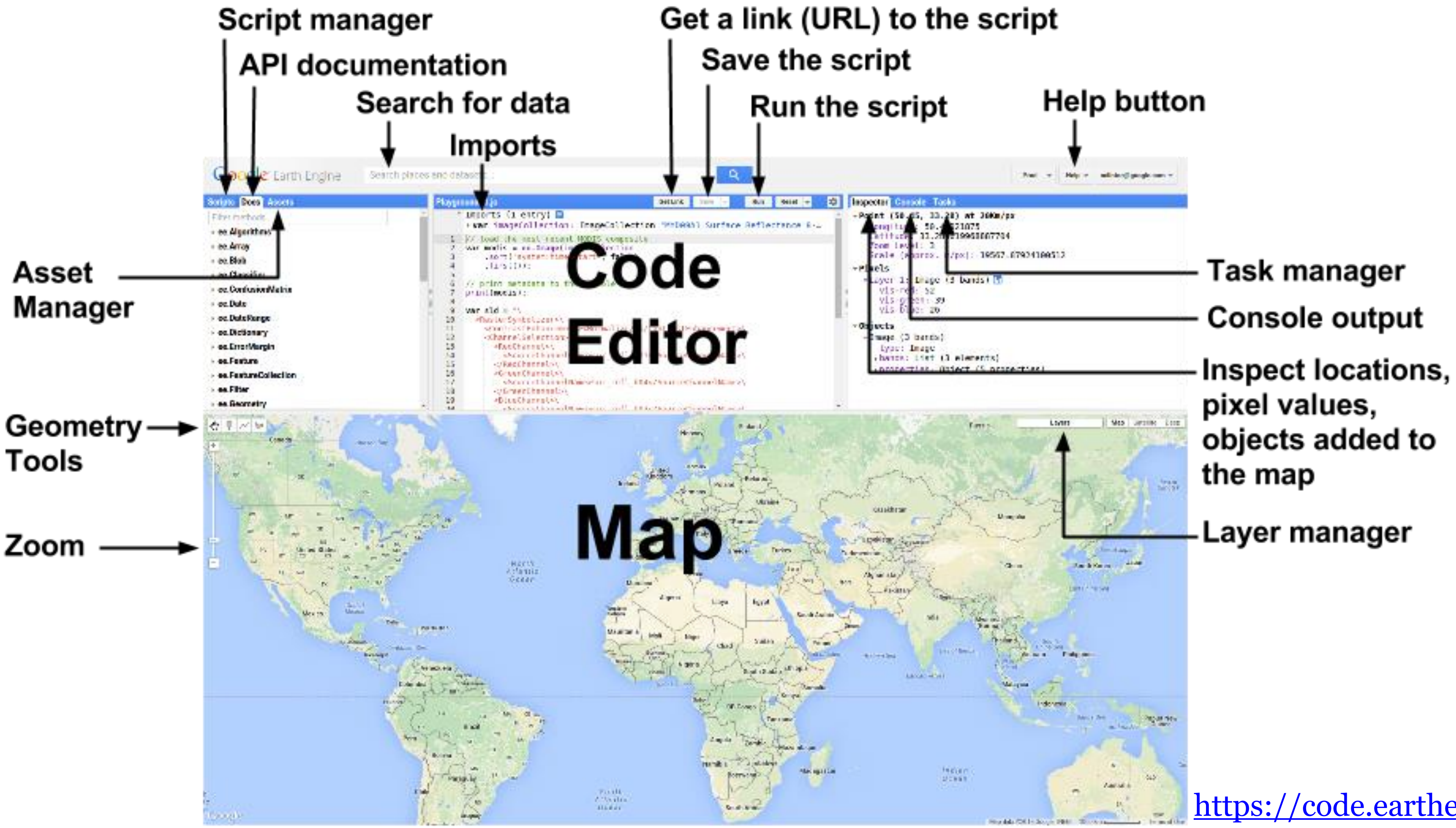
Support Regression: Cart, RF, SVM
Support Probability: Cart, RF, NaiveBayes, SVM

Unsupervised clustering is also available through ee.Clusterer

04

Let’s start with Google Earth Engine

Google Earth Engine



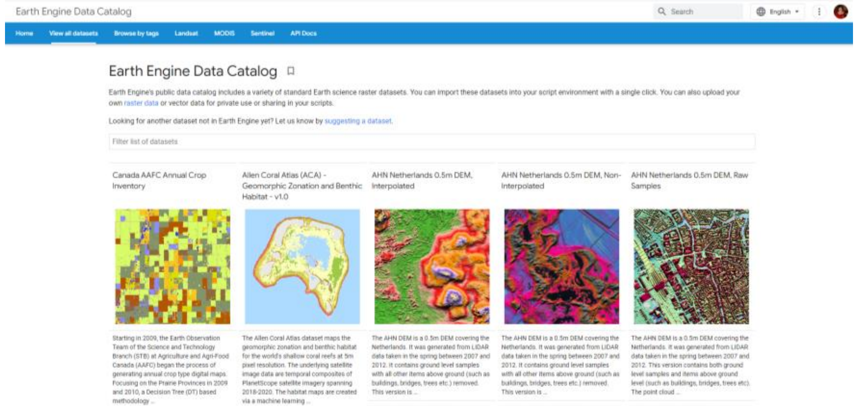
https://code.earthengine.google.com/

05

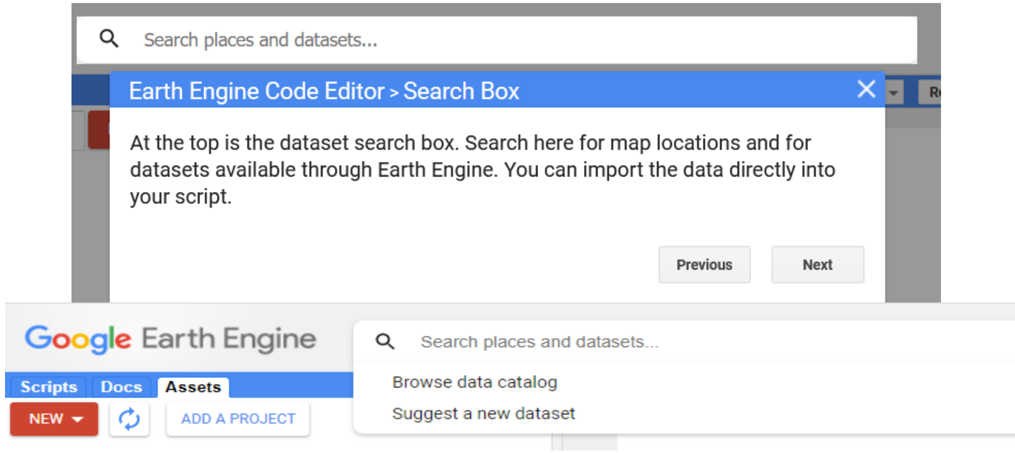
Data Catalogue

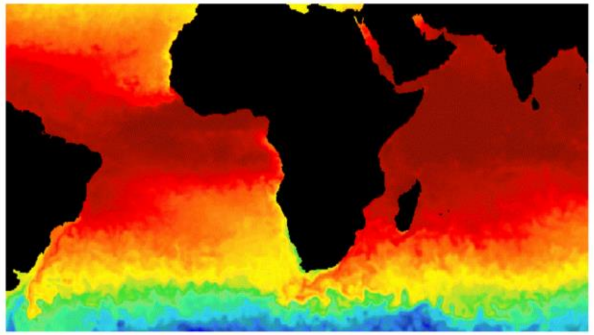
Data Catalogue

Access it through this [webpage]



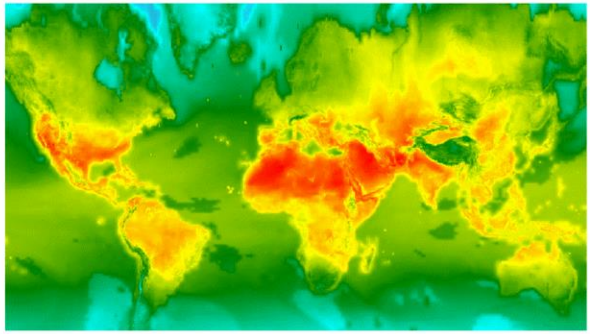
Access it through the Search box at the top of the Code Editor:





Surface Temperature

Thermal satellite sensors can provide surface temperature and emissivity information. The Earth Engine data catalog includes both land and sea surface temperature products derived from several spacecraft sensors, including MODIS, ASTER, and AVHRR, in addition to raw Landsat thermal data.



Climate

Climate models generate both long-term climate predictions and historical interpolations of surface variables. The Earth Engine catalog includes historical reanalysis data from NCEP/NCAR, gridded meteorological datasets like NLDAS-2, and GridMET, and climate model outputs like the University of Idaho MACAv2-METDATA and the NASA Earth Exchange's Downscaled Climate Projections.

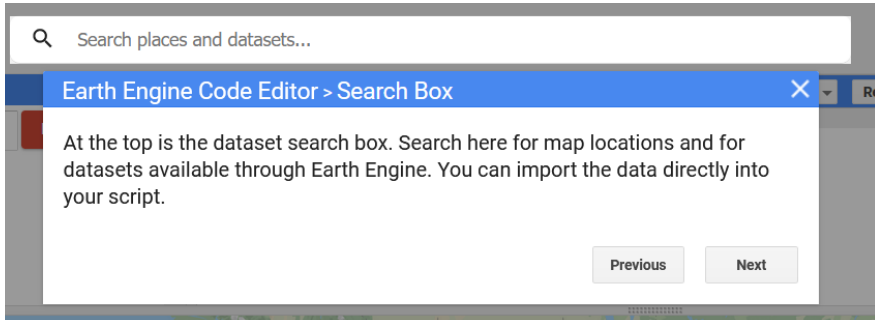
Landsat

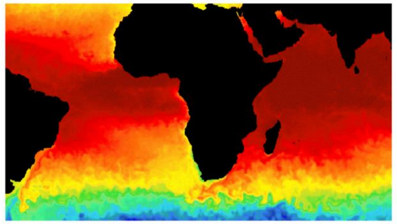
Landsat, a joint program of the USGS and NASA, has been observing the Earth continuously from 1972 through the present day. Today the Landsat satellites image the entire Earth's surface at a 30-meter resolution about once every two weeks, including multispectral and thermal data.

Sentinel

The Copernicus Program is an ambitious initiative headed by the European Commission in partnership with the European Space Agency (ESA). The Sentinels include all-weather radar images from Sentinel-1A and -1B, high-resolution optical images from Sentinel 2A and 2B, as well as ocean and land data suitable for environmental and climate monitoring from Sentinel 3.

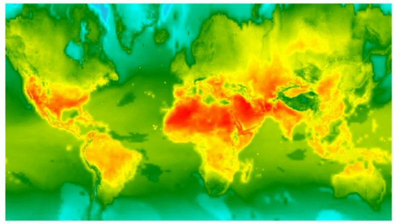
Data Catalogue





Surface Temperature

Thermal satellite sensors can provide surface temperature and emissivity information. The Earth Engine data catalog includes both land and sea surface temperature products derived from several spacecraft sensors, including MODIS, ASTER, and AVHRR, in addition to raw Landsat thermal data.



Climate

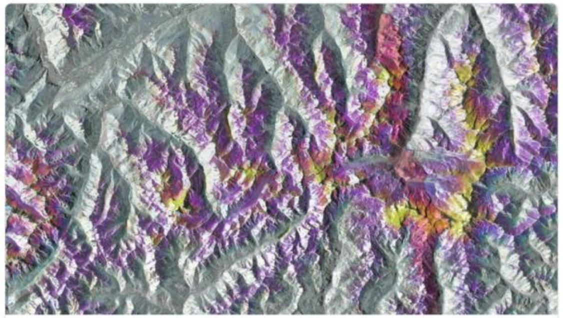
Climate models generate both long-term climate predictions and historical interpolations of surface variables. The Earth Engine catalog includes historical reanalysis data from NCEP/NCAR, gridded meteorological datasets like NLDAS-2, and GridMET, and climate model outputs like the University of Idaho MACAv2-METDATA and the NASA Earth Exchange's Downscaled Climate Projections.

Landsat

Landsat, a joint program of the USGS and NASA, has been observing the Earth continuously from 1972 through the present day. Today the Landsat satellites image the entire Earth's surface at a 30-meter resolution about once every two weeks, including multispectral and thermal data.

Sentinel

The Copernicus Program is an ambitious initiative headed by the European Commission in partnership with the European Space Agency (ESA). The Sentinels include all-weather radar images from Sentinel-1A and -1B, high-resolution optical images from Sentinel 2A and 2B, as well as ocean and land data suitable for environmental and climate monitoring from Sentinel 3.



Sentinel-1 SAR GRD: C-band Synthetic Aperture Radar

Data availability: 2014 – Present

The Sentinel-1 mission provides data from a dual-polarization C-band Synthetic Aperture Radar (SAR) instrument. SAR instruments are capable of acquiring meaningful data in all weather conditions (even clouds) during daytime and nighttime. Sentinel-1 data is used across many domains, including maritime activity, sea-ice mapping, humanitarian aid, crisis response, and forest management.

Sentinel-2 MSI: Multispectral Instrument

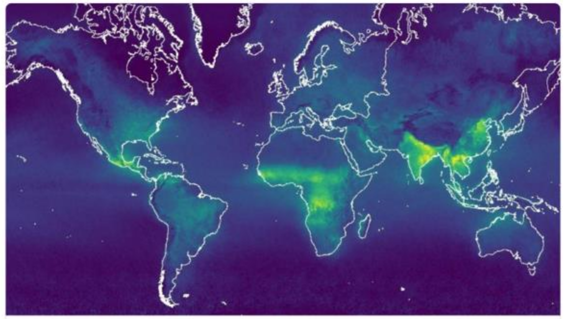
Data availability: 2015 – Present

The Sentinel-2 mission collects high-resolution multispectral imagery useful for a broad range of applications, including monitoring of vegetation, soil and water cover, land cover change, as well as humanitarian and disaster risk.

Sentinel-3 OLCI EFR: Ocean and Land Color Instrument

Data availability: 2016 – Present

The Sentinel-3 instrument provides systematic measurements of the planet's oceans, land, ice, and atmosphere, including the temperature, color and height of the sea surface as well as the thickness of sea ice.



Sentinel-5P TROPOMI: TROPOspheric Monitoring Instrument

Data availability: 2018 – Present

The Sentinel-5 Precursor mission collects data useful for assessing air quality, including concentrations of: ozone, methane, formaldehyde, aerosol, carbon monoxide, nitrogen oxide, and sulphur dioxide.

Data Catalogue

Sentinel-1 SAR GRD: C-band Synthetic Aperture Radar

Data availability: 2014 – Present

The Sentinel-1 mission provides data from a dual-polarization C-band Synthetic Aperture Radar (SAR) instrument. SAR instruments are capable of acquiring meaningful data in all weather conditions (even clouds) during daytime and nighttime. Sentinel-1 data is used across many domains, including maritime activity, sea-ice mapping, humanitarian aid, crisis response, and forest management.

Sentinel-2 MSI: Multispectral Instrument

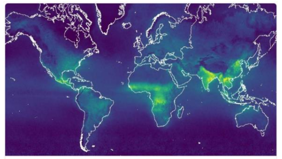
Data availability: 2015 – Present

The Sentinel-2 mission collects high-resolution multispectral imagery useful for a broad range of applications, including monitoring of vegetation, soil and water cover, land cover change, as well as humanitarian and disaster risk.

Sentinel-3 OLCI EFR: Ocean and Land Color Instrument

Data availability: 2016 – Present

The Sentinel-3 instrument provides systematic measurements of the planet's oceans, land, ice, and atmosphere, including the temperature, color and height of the sea surface as well as the thickness of sea ice.



Sentinel-5P TROPOMI: TROPOspheric Monitoring Instrument

Data availability: 2018 – Present

The Sentinel-5 Precursor mission collects data useful for assessing air quality, including concentrations of: ozone, methane, formaldehyde, aerosol, carbon monoxide, nitrogen oxide, and sulphur dioxide.

Surface Reflectance

Level-2A orthorectified atmospherically corrected surface reflectance.

Dataset availability: 2017-03-28 – Present

Top-of-Atmosphere Reflectance

Level-1C orthorectified top-of-atmosphere reflectance.

Dataset availability: 2015-06-23 – Present

Sentinel-2 MSI: MultiSpectral Instrument, Level-2A

Dataset Availability

2017-03-28T00:00:00Z - 2022-02-23T00:00:00

Dataset Provider

European Union/ESA/Copernicus

Earth Engine Snippet

`ee.ImageCollection("COPERNICUS/S2_SR")`

Tags

copernicus esa eu msi reflectance sentinel sr

Description | Bands | Image Properties | Terms of Use

Image Properties

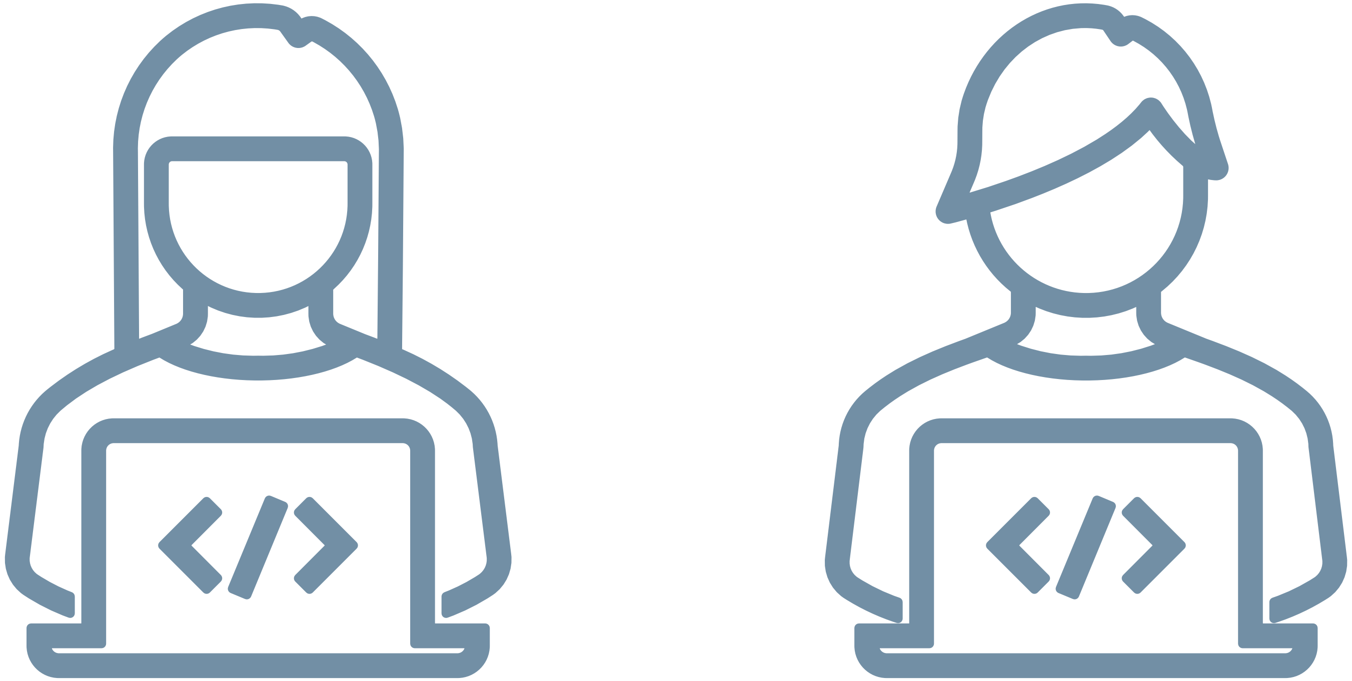
Name	Type	Description
AOT_RETRIEVAL_ACCURACY	DOUBLE	Accuracy of Aerosol Optical thickness model
CLOUDY_PIXEL_PERCENTAGE	DOUBLE	Granule-specific cloudy pixel percentage taken from the original metadata
CLOUD_COVERAGE_ASSESSMENT	DOUBLE	Cloudy pixel percentage for the whole archive that contains this granule. Taken from the original metadata
CLOUDY_SHADOW_PERCENTAGE	DOUBLE	Percentage of pixels classified as cloud shadow
DARK_FEATURES_PERCENTAGE	DOUBLE	Percentage of pixels classified as dark features or shadows
DATASTRIP_ID	STRING	Unique identifier of the datastrip Product Data Item (PDI)
DATATAKE_IDENTIFIER	STRING	Uniquely identifies a given Datatake. The ID contains the Sentinel-2 satellite, start date and time, absolute orbit number, and processing baseline.
DATATAKE_TYPE	STRING	MSI operation mode
DEGRADED_MSI_DATA_PERCENTAGE	DOUBLE	Percentage of degraded MSI and ancillary data
FORMAT_CORRECTNESS	STRING	Synthesis of the On-Line Quality Control (OLQC) checks performed at granule (Product_Syntax) and datastrip (Product Syntax and DS_Consistency) levels
GENERAL_QUALITY	STRING	Synthesis of the OLQC checks performed at the datastrip level (Relative_Orbit_Number)
GENERATION_TIME	DOUBLE	Product generation time
GEOMETRIC_QUALITY	STRING	Synthesis of the OLQC checks performed at the datastrip level (Attitude_Quality_Indicator)
GRANULE_ID	STRING	Unique identifier of the granule PDI (PDI_ID)

Explore in Earth Engine

```
/**
 * Function to mask clouds using the Sentinel-2 QA band
 * @param {ee.Image} image Sentinel-2 image
 * @return {ee.Image} cloud masked Sentinel-2 image
 */
function maskS2clouds(image) {
  var qa = image.select('QA60');

  // Bits 10 and 11 are clouds and cirrus, respectively.
  var cloudBitMask = 1 << 10;
  var cirrusBitMask = 1 << 11;

  // Both flags should be set to zero, indicating clear conditions.
  var mask = qa.bitwiseAnd(cloudBitMask).eq(0)
      .and(qa.bitwiseAnd(cirrusBitMask).eq(0));

  return image.updateMask(mask).divide(10000);
}

var dataset = ee.ImageCollection('COPERNICUS/S2_SR')
                  .filterDate('2020-01-01', '2020-01-30')
                  // Pre-filter to get less cloudy granules.
                  .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',20))
                  .map(maskS2clouds);

var visualization = {
  min: 0.0,
  max: 0.3,
  bands: ['B4', 'B3', 'B2'],
};

Map.setCenter(83.277, 17.7009, 12);

Map.addLayer(dataset.mean(), visualization, 'RGB');
```

Open in Code Editor

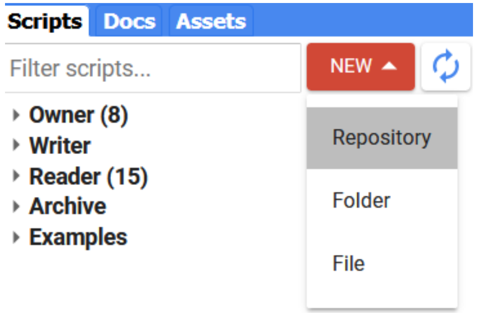
Explore Sentinel-2 MSI on your country

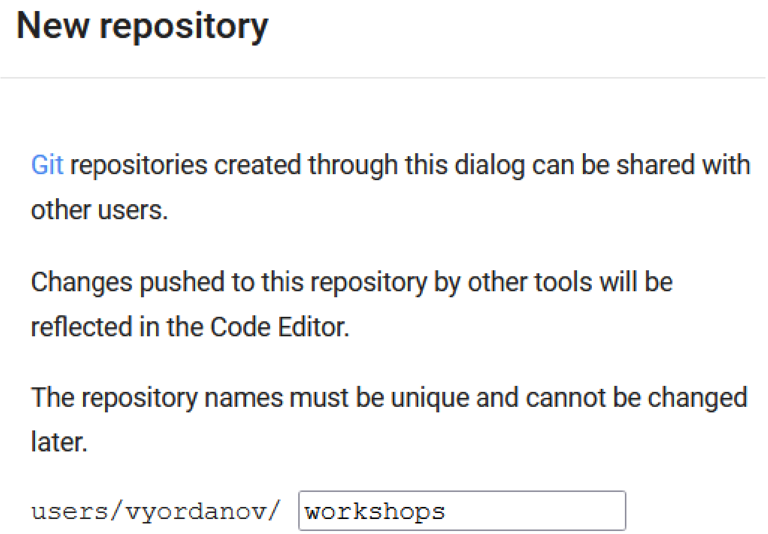


Google Earth Engine

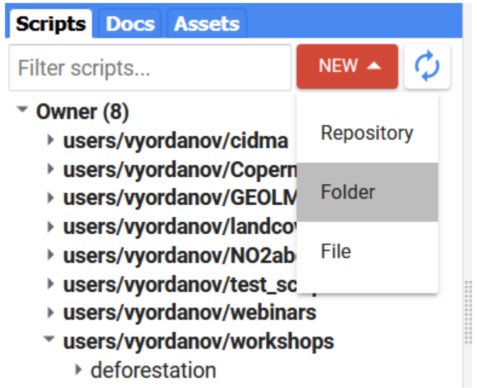
Setup a new workspace

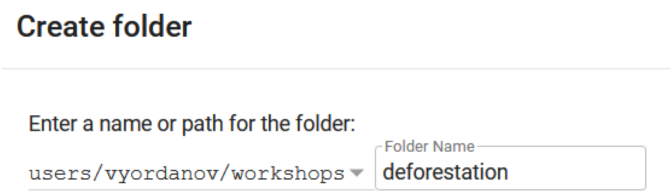
Create new repository:



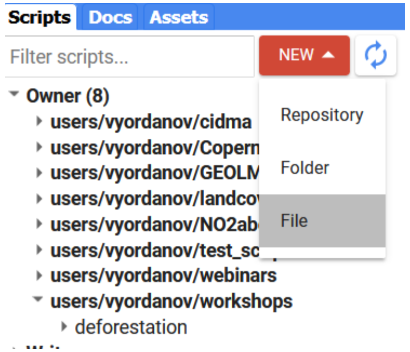


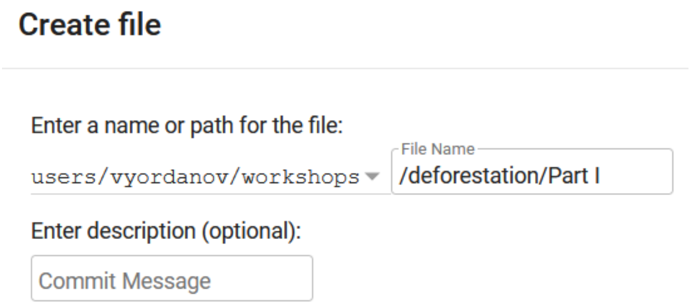
Create new folder:





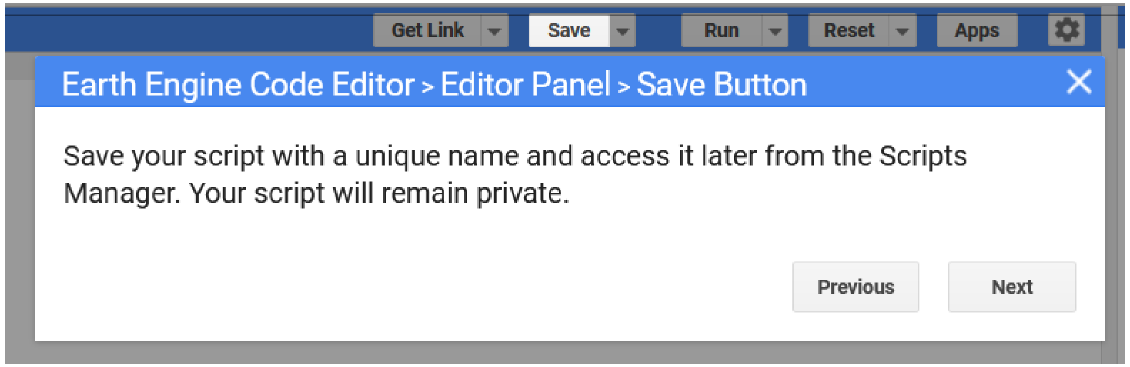
Create new script:

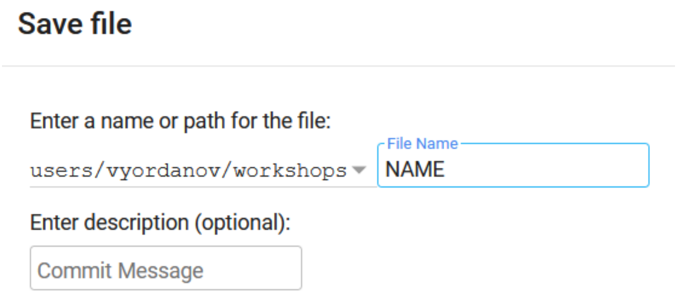




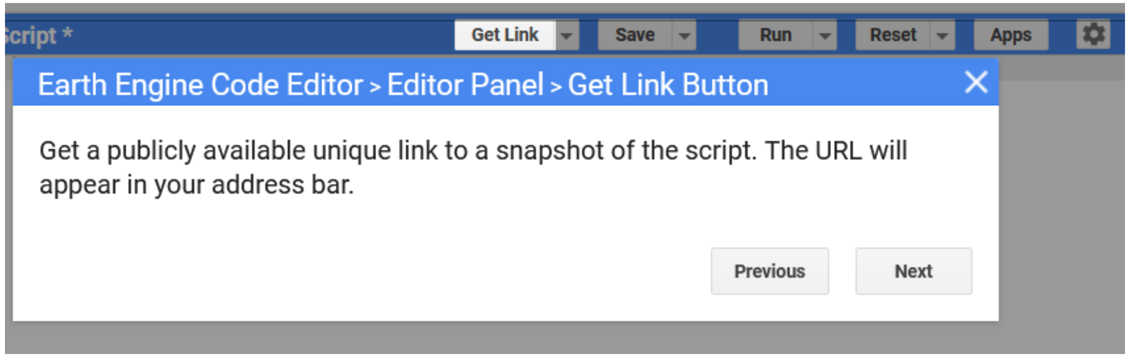
Save your script and share it

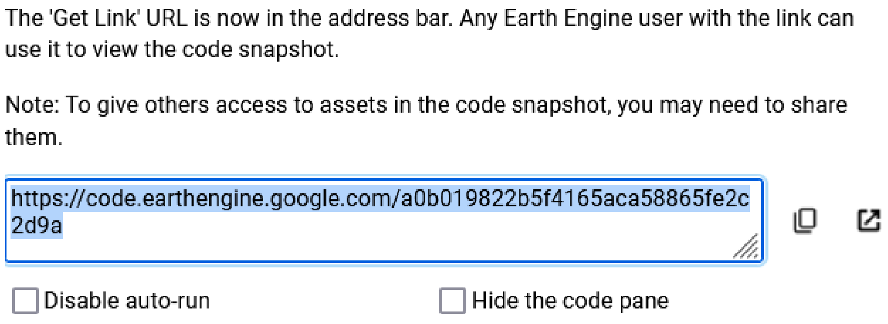
Save the script:



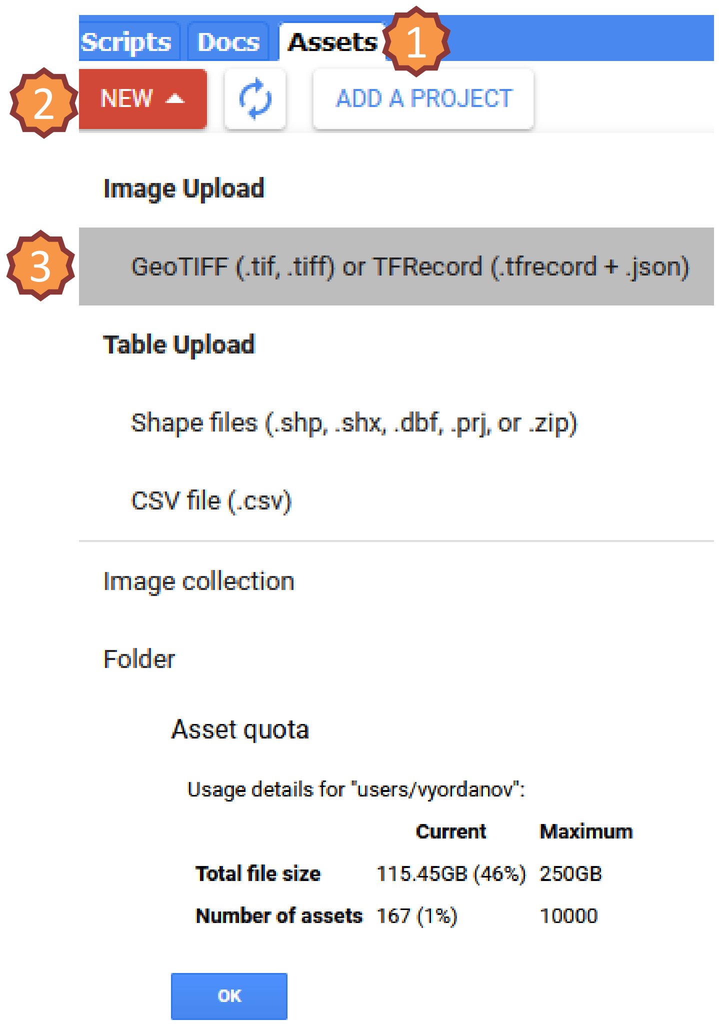


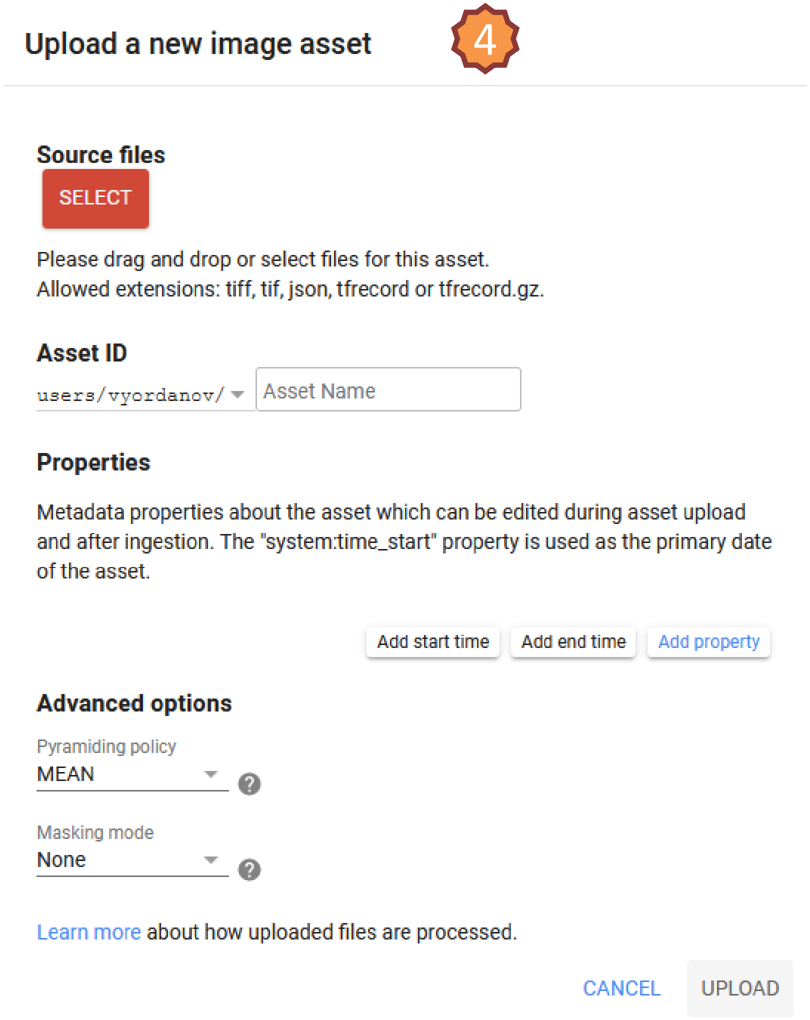
Share the script:



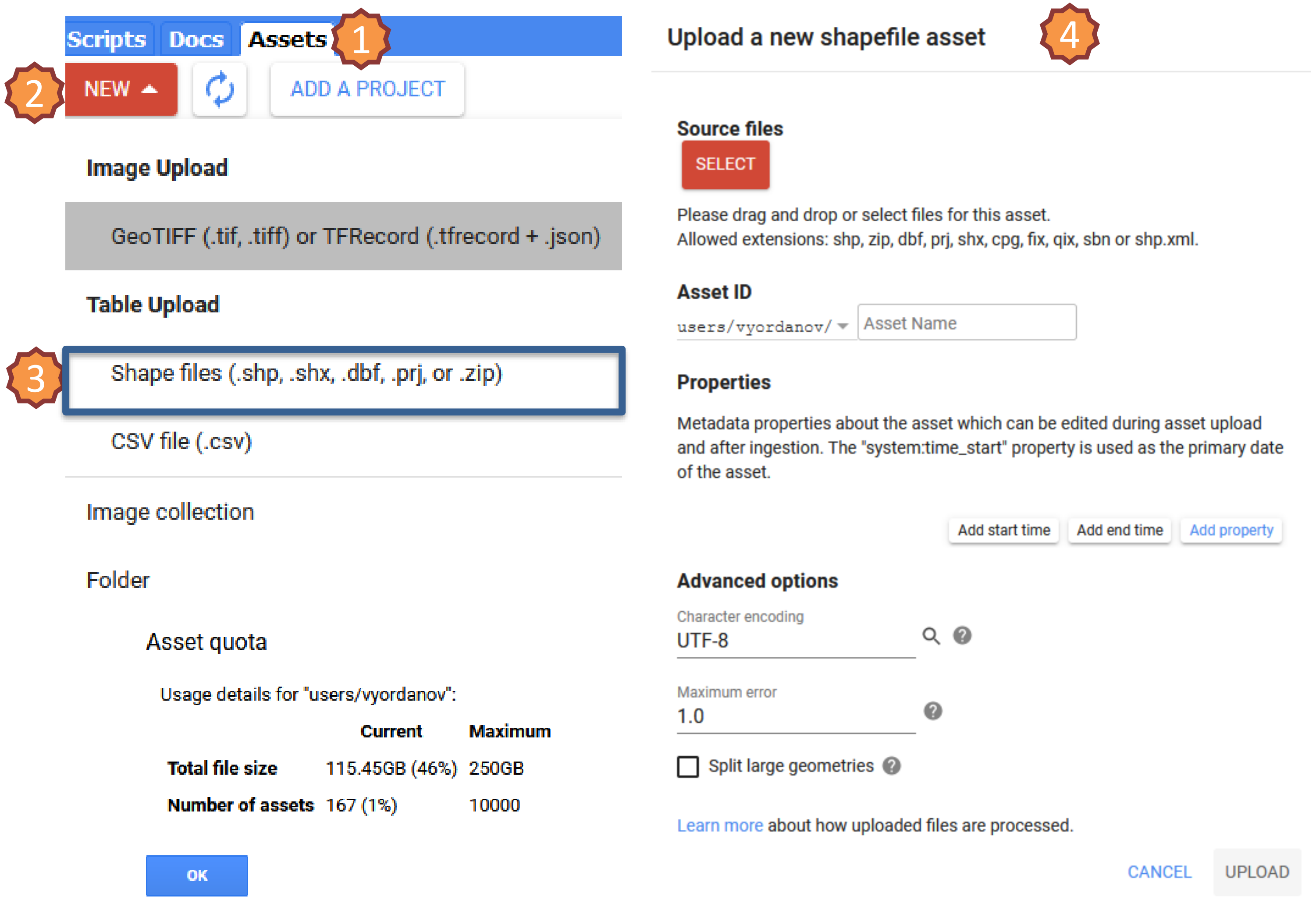


Upload your own data

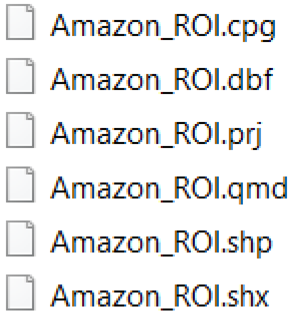




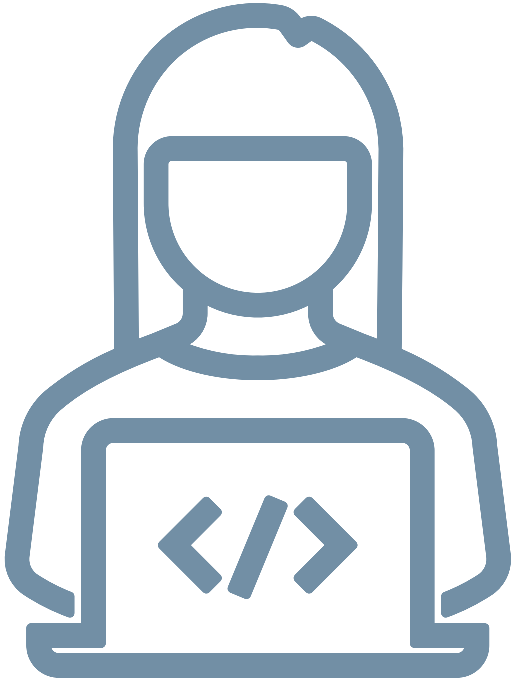
Upload your own data

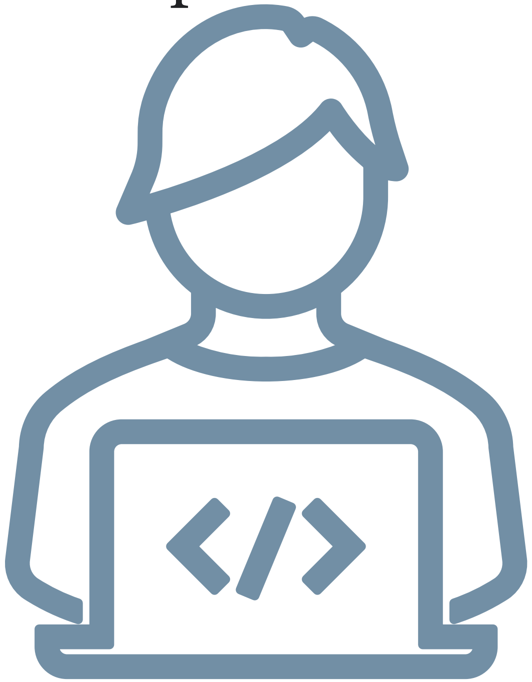


Please download the [Amazon_ROI.zip]() which contains a shape file of the Area of Interest that we are going to work on



Upload the shape file.
Save and share the script





Google Earth Engine

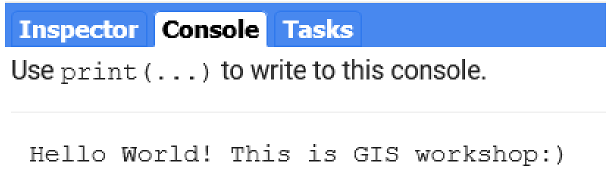
06

Very brief introduction to JavaScript

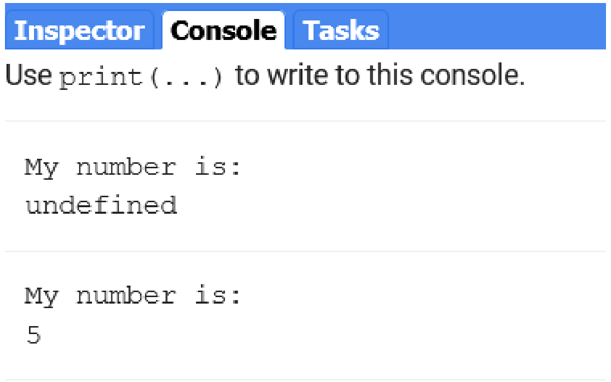
JavaScript – Very brief introduction

**You can copy and paste the code lines*

```
//Print first programming line
print('Hello World! This is GIS workshop:)');

//Double backslash for Single-line comments

/*
Opening /* and closing */ for Multi-line comments
*/

//Selecting multi lines and pressing Ctrl + /  (for English keyboards)
```




```
//Declare a variable.
var myNumber;
print('My number is: ',myNumber);

//Declare and initialize a new variable.
var myNumber = 5;
print('My number is: ',myNumber)
```

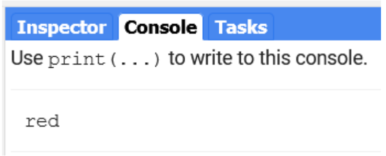


JavaScript – Very brief introduction

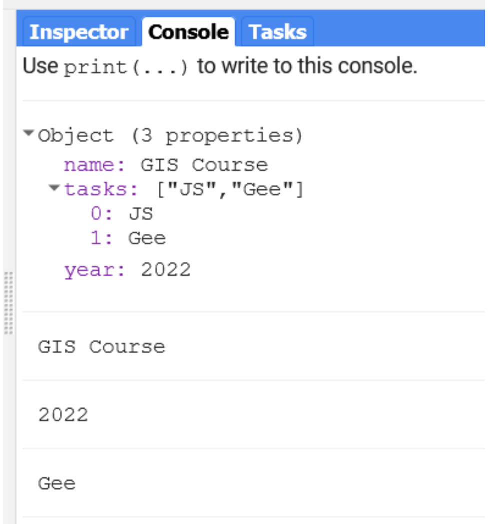
**You can copy and paste the code lines*

```
//Variable;
var myVariable;
//String (“chain of characters”)
var myString = 'Hello, GIS';
//Number
var myNumber = 2022;
var myPi = 3.1416;
```

```
//Array
//Arrays can be composed by different types of objects.
var myNumberArray = [1,2,3,4,5];
var myStringArray = ['red', 'green', 'blue'];
print(myStringArray[0])
```




```
//Dictionary or Objects
//Dictionaries are composed by keys (properties) and values pairs.
 var myDictionary = {
  name: 'GIS Course',
  year: 2022,
  tasks:['JS','Gee']
};
print(myDictionary)
print(myDictionary['name'])
print(myDictionary.year)
print(myDictionary.tasks[1])
```



JavaScript – Very brief introduction

**You can copy and paste the code lines*

```
// This generates a list of numbers from 1 to 10.
var myList = ee.List.sequence(1, 10);
print(myList)
//Define a function that can be applied to the input.

var computeSquares = function(number) {
  // We define the operation using the EE API.
  return ee.Number(number).pow(2);
};

// The map() operation takes a function that works on each element independently
// and returns a value.
// Apply your function to each item in the list by using the map() function.

var squares = myList.map(computeSquares);
print(squares);  // [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

N.B.:

- To ensure that your function is fully usable throughout your code, place its definition in the beginning of your script.
- The use of for-loops is discouraged in Earth Engine. Instead of for-loops, the `map()` function is applied to collections!

Some sources

[Introduction to JavaScript and Google Earth Engine]

[Earth Engine 101 - Introduction to the API]

[Google Earth Engine Developers]

Google Earth Engine

Some of the most relevant object classes on Earth Engine are:

ee.Image: an object to represent an image

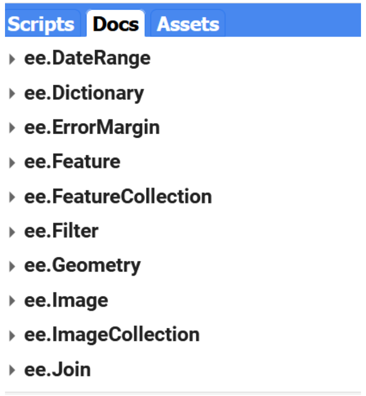
ee.ImageCollection: a stack or time-series of images

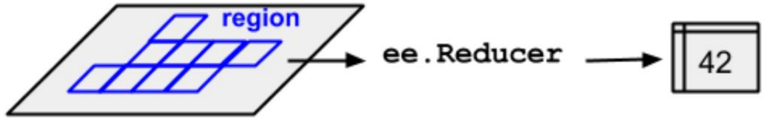
ee.Geometry: an object to represent a vector

ee.Feature: a Geometry with attributes

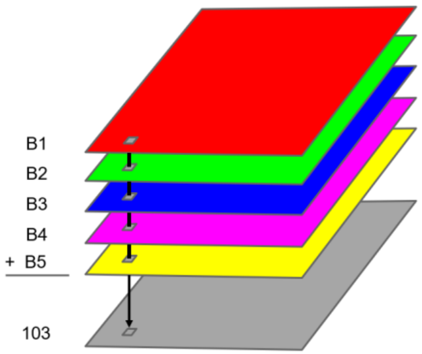
ee.FeatureCollection: set of Features

ee.Reducer: a way to aggregate data over time, space, bands, arrays and other data structures

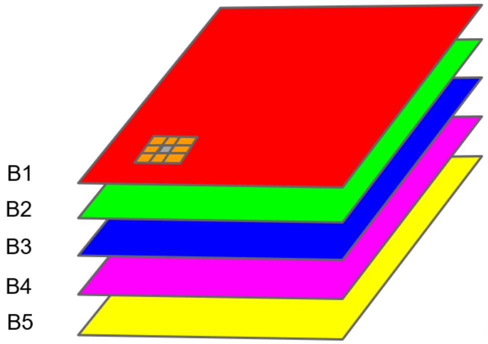




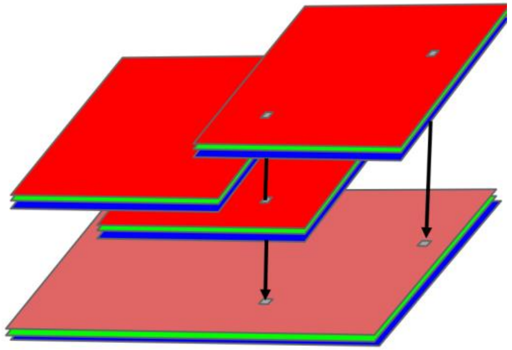
Reduce bands



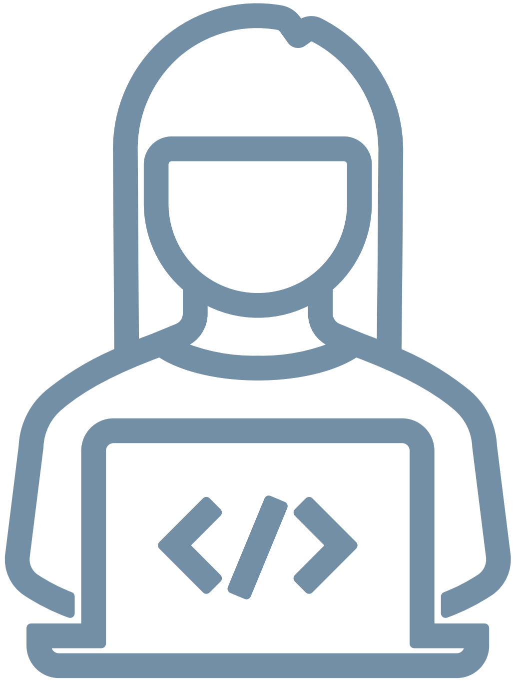
Reduce neighborhood

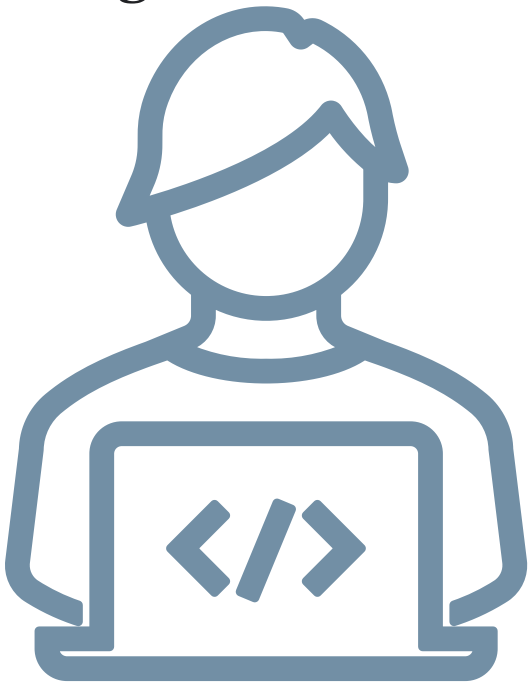


Reduce Image Collection



Compute the median for each band
and each pixel of an image collection





Google Earth Engine

07

Let’s start

Define the region of interest

```
//Define a region of interest;
var pointBrazil = ee.Geometry.Point(-52.97, -6.40);
//they are WGS84 coordinates
 //Center the map with respect to this point
Map.centerObject(pointBrazil,9)
//Add the layer point to the map
Map.addLayer(pointBrazil, {}, "Center of the region")

// Add polygon for the Area of interest
var AOI = ee.FeatureCollection('users/vyordanov/AmazonROI')
Map.centerObject(AOI,9)
Map.addLayer(AOI, {}, "Area of Interest")

// Compute and print the area of our AOI in km²:
var aoiArea=AOI.geometry().area().divide(1000000);
print('AOI Area: ',aoiArea,"[km²]")
```

```
AOI Area:                    JSON
48551.96934273438
[km²]                        JSON
```

GEE image visualization and image collection filtering

```
// Function to apply scaling factors for Landsat images.
function applyScaleFactors(image) {
  var opticalBands = image.select('SR_B.').multiply(0.0000275).add(-0.2);
  var thermalBands = image.select('ST_B.*').multiply(0.00341802).add(149.0);
  return image.addBands(opticalBands, null, true)
              .addBands(thermalBands, null, true);
}

var landsat2015 = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2')
              .filterDate('2015-05-01', '2015-08-30')
              .filterBounds(AOI)
              .filter(ee.Filter.lt('CLOUD_COVER', 10))
              .aside(print)
              .map(applyScaleFactors)
              .median()
              .clip(AOI)

var visParLandsat = {
  bands: ['SR_B4', 'SR_B3', 'SR_B2'],
  min: 0.0,
  max: 0.3,
};
Map.addLayer(landsat2015,visParLandsat,'Landsat 2015')
```

Converts the data values into actual surface reflectance.

Image Collection

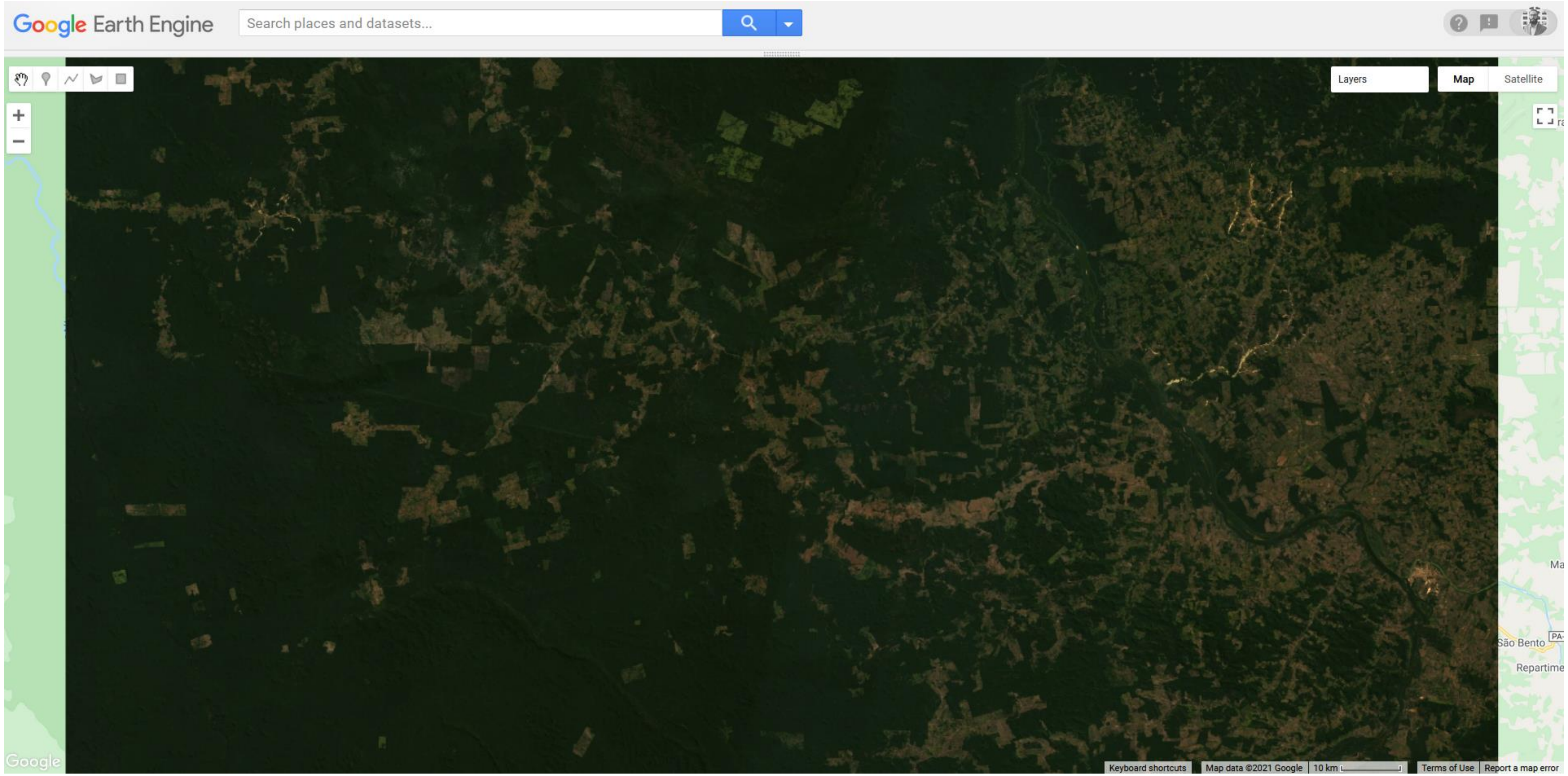
Image

- Calling the Landsat-8 Surface Reflectance Collection
- Applying a temporal filter
- Filtering to show images only in the AOI
- Filtering images that have cloud cover less than 10%
- Prints in the console the image collection
- Applying the scaling function
- Applying a median reducer*
- Clipping the image using the AOI boundaries

Defines the color palette for layer visualization

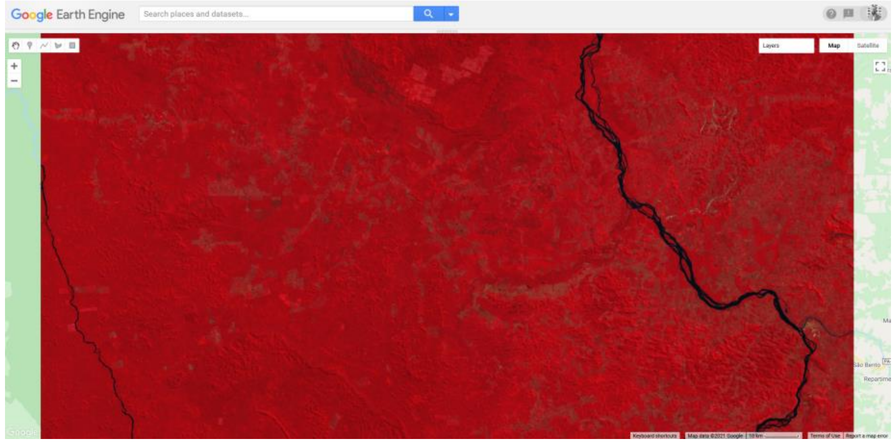
*Reduces an image collection by calculating the median of all values at each pixel across the stack of all matching bands.

GEE image visualization and image collection filtering



GEE image visualization – False Color

```
// We can make another visualization, for example, a False Color. It will use the Near-Infrared Band,
//Red and Green. In this way the vegetation will ‘highlighted’ in red, because a plant with more
//chlorophyll will reflect more near-infrared energy than an unhealthy plant. This is due to the fact
// that healthy vegetation absorbs blue- and red-light energy to fuel photosynthesis and create
//chlorophyll.

var visParLandsatFalseColor = {
  bands: ['SR_B5', 'SR_B4', 'SR_B3'],
  min: 0.0,
  max: 0.5,
};

Map.addLayer(landsat2015, visParLandsatFalseColor,'Landsat 2015 False Color')
```



NDVI (Normalized Differnce Vegetation Index)

$$NDVI = (NIR-RED)/(NIR+RED)$$

The value range of the NDVI is -1 to 1.

Negative values of NDVI (values approaching -1) correspond to water.

Values close to zero (-0.1 to 0.1) generally correspond to barren areas of rock, sand, or snow.

Low, positive values represent shrub and grassland (approximately 0.2 to 0.4), while high values indicate temperate and tropical rainforests (values approaching 1).

https://custom-scripts.sentinel-hub.com/sentinel-2/ndvi/

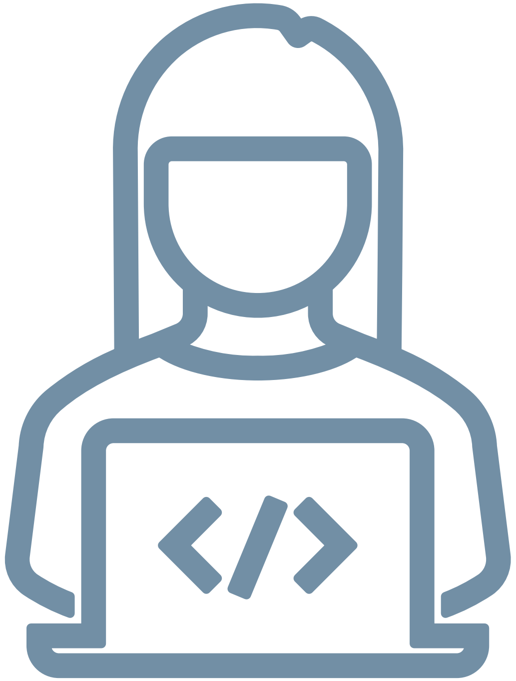
GEE image visualization – NDVI

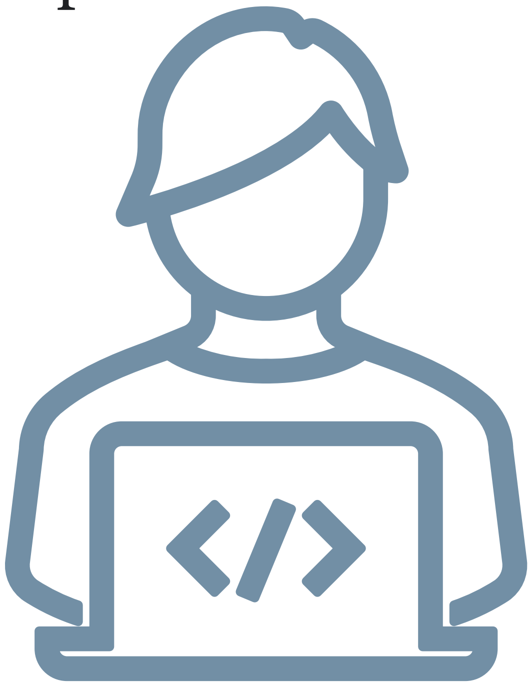
```
//Using NDVI(Normalized Difference Vegetation Index)
//computed using the following formula: (NIR - R) / (NIR + R).
//Consider the three alternatives (comment the other two when you are using one of them)

// 1. Band selection and manual computation:
var nir = landsat2015.select('SR_B5');
var red = landsat2015.select('SR_B4');
var ndvi = nir.subtract(red).divide(nir.add(red)).rename('NDVI');
Map.addLayer(ndvi,{min: -1, max: 1, palette: ['blue', 'white', 'green']},'NDVI2015')

// 2. Using GEE built-in functions:
var ndvi2 = landsat2015.normalizedDifference(['SR_B5', 'SR_B4']).rename('NDVI');
Map.addLayer(ndvi2,{min: -1, max: 1, palette: ['blue', 'white', 'green']},'NDVI2015 2')

// 3. Make it as a function and directly apply it during the Image Collection filtering – the // // // resulted
image will have directly the NDVI band included:
function ndviLS(image){
  var ndvi = image.normalizedDifference(['SR_B5', 'SR_B4']).rename('NDVI');
  return image.addBands(ndvi);
}
// NB: We will use the approach 3 for the rest of webinar. Put the definition of functions at the top of the
script. Do not forget to .map(ndviLS) the ndvi function in the filtering
Map.addLayer(landsat2015.select('NDVI'),{min: -1, max: 1, palette: ['blue', 'white', 'green']},'NDVI2015')
```

GEE image visualization and image collection filtering - Recap

```
function applyScaleFactors(image) {
  var opticalBands = image.select('SR_B.').multiply(0.0000275).add(-0.2);
  var thermalBands = image.select('ST_B.*').multiply(0.00341802).add(149.0);
  return image.addBands(opticalBands, null, true)
              .addBands(thermalBands, null, true);
}

function ndviLS(image){
  var ndvi = image.normalizedDifference(['SR_B5', 'SR_B4']).rename('NDVI');
  return image.addBands(ndvi);
}

var landsat2015 = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2')
                  .filterDate('2015-05-01', '2015-08-30')
                  .filterBounds(AOI)
                  .filter(ee.Filter.lt('CLOUD_COVER', 10))
                  .aside(print)
                  .map(applyScaleFactors)
                  .map(ndviLS)
                  .median()
                  .clip(AOI)

var visParLandsat = {
  bands: ['SR_B4', 'SR_B3', 'SR_B2'],
  min: 0.0,
  max: 0.3,
};
Map.addLayer(landsat2015,visParLandsat,'Landsat 2015')
Map.addLayer(landsat2015.select('NDVI'),{min: -1, max: 1, palette: ['blue', 'white', 'green']},'NDVI2015')
```

Add Landsat 2015 data to your script





Google Earth Engine

08

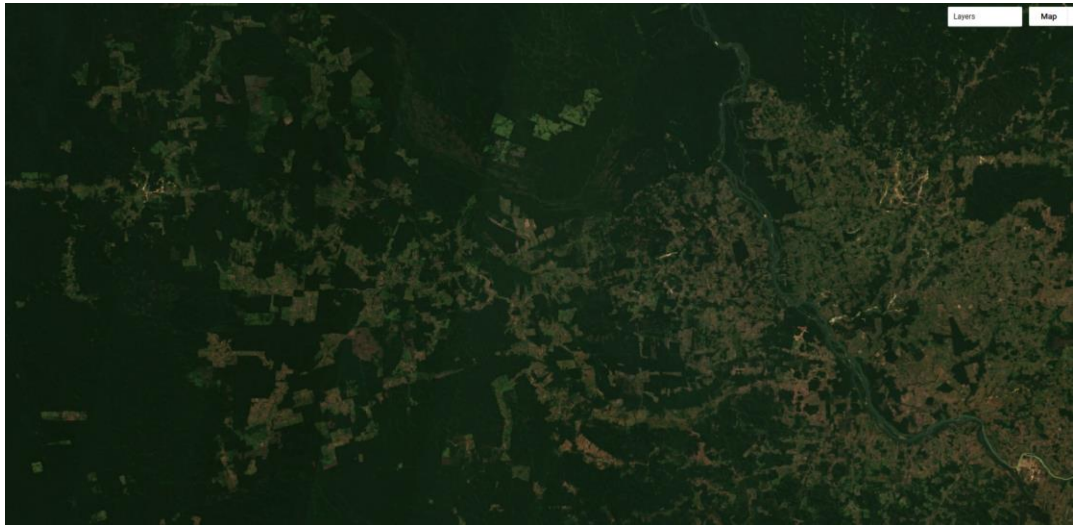
Let’s add also Sentinel-2 image 2019

GEE image visualization – adding an image from a second period

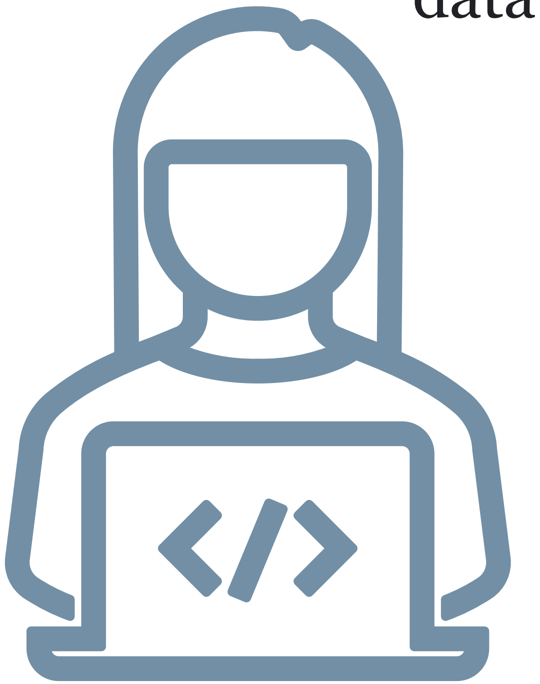
```
// To understand the change in the forest between 2015 and 2019 we need one image of 2019 over
// the same period. We use Sentinel-2 MSI image for 2019 to see the differences in the processing.
// First, it is better to include another suggested (for Sentinel-2) function which allows us to mask
// existing clouds.
// The function to mask the clouds on Sentinel-2 images using QA60 band is:
function maskS2clouds(image) {
  var qa = image.select('QA60');

  // Bits 10 and 11 are clouds and cirrus, respectively. Value 1 means presence of cloud and cirrus
  var cloudBitMask = 1 << 10;
  var cirrusBitMask = 1 << 11;

  // Both flags should be set to zero, indicating clear conditions.
  var mask = qa.bitwiseAnd(cloudBitMask).eq(0)
      .and(qa.bitwiseAnd(cirrusBitMask).eq(0));
  return image.updateMask(mask).divide(10000);
}

// Then we also add the ndvi function for Sentinel-2 (different bands with respect to Landsat)
function ndviSE(image){
  var ndvi = image.normalizedDifference(['B8', 'B4']).rename('NDVI');
  return image.addBands(ndvi);
}
```

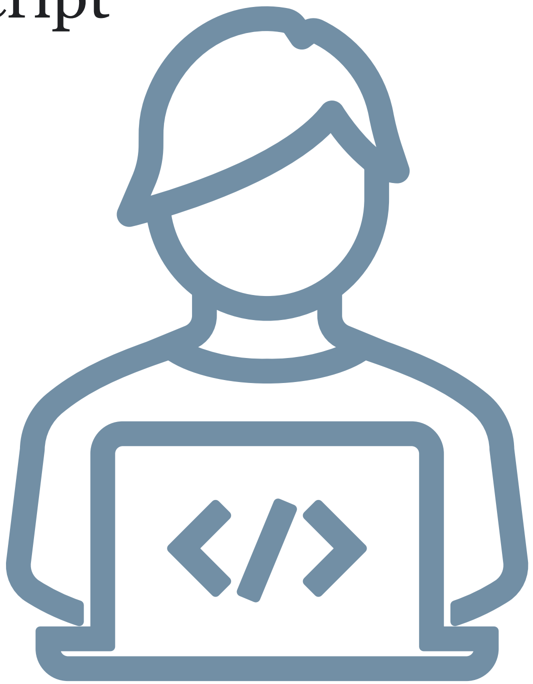
GEE image visualization – adding an image from a second period

```
var sentinel2019 = ee.ImageCollection('COPERNICUS/S2_SR')
                  .filterDate('2019-05-01', '2019-08-30')
                  .filterBounds(AOI)
                  .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 10)) //note the different property
                  .aside(print)
                  .map(maskS2clouds)
                  .map(ndviSE)
                  .median()
                  .clip(AOI);

var visParSentinel = {
  min: 0.0,
  max: 0.3,
  bands: ['B4', 'B3', 'B2'],
};

Map.addLayer(sentinel2019, visParSentinel, 'Sentinel 2019');
Map.addLayer(sentinel2019.select('NDVI'),{min: -1, max: 1, palette: ['blue', 'white',
'green']},'NDVI2019')
```



Add Sentinel 2019 data to your script





Google Earth Engine

09

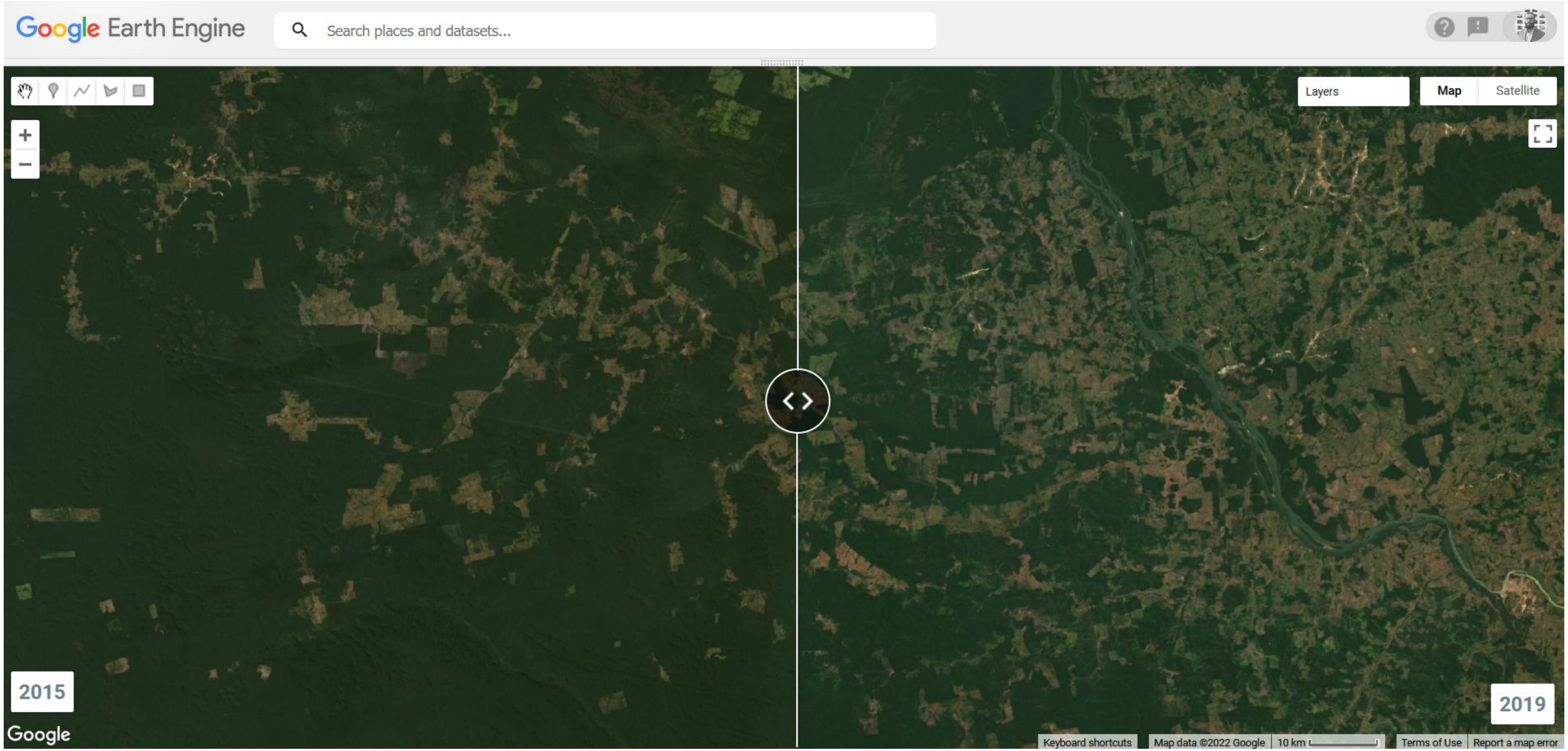
Let’s compare 2015 and 2019 images

GEE image visualization – Split screen*

```
var linkedMap = ui.Map();
linkedMap.addLayer(sentinel2019, visParSentinel, 'Sentinel 2019');
linkedMap.centerObject(AOI,9);
linkedMap.setControlVisibility({all: true, zoomControl: true, mapTypeControl: true})
var linker = ui.Map.Linker([ui.root.widgets().get(0), linkedMap]);

//Add title labels to the maps and create the split panel comprising the two maps
var title_during= Map.add(ui.Label(
'2015', {fontWeight: 'bold', fontSize: '20px', position: 'bottom-left', color: 'slateGrey'}));

var title_after= linkedMap.add(ui.Label(
'2019', {fontWeight: 'bold', fontSize: '20px', position: 'bottom-right', color: 'slateGrey'}));

var splitPanel = ui.SplitPanel({
firstPanel: linker.get(0),
secondPanel: linker.get(1),
orientation: 'horizontal',
wipe: true,
style: {stretch: 'both'}
});
ui.root.widgets().reset([splitPanel]);
```

* It is not relevant for the further analyses

GEE image visualization – Split screen*



Here you can find the full script of the session

Thank you for the attention!

See you in the next part (29 March 2022)

maria.brovelli@polimi.it vasil.yordanov@polimi.it