Lecture 06
GEE Image Manipulation:
band arithmetic, thresholds, masks, reducers

2024-04-08

Sébastien Valade

VNIVERSIDAD NACIONAL
AVTONOMA DE
MEXICO

Previous lecture:

GEE introduction:

⇒ setup, datasets, image visualization, image collection filtering

Today:

GEE image manipulation:

⇒ band arithmetic (spectral indices), thresholds, masks, reducers

Previous lecture:

GEE introduction:

⇒ setup, datasets, image visualization, image collection filtering

Today:

GEE image manipulation:

⇒ band arithmetic (spectral indices), thresholds, masks, reducers

Table of Contents

1. Band arithmetic
    1. Remote sensing principles
    2. Spectral Indices using band arithmetic

2. Thresholding & Masking
    1. Thresholding
    2. Masking

3. Analyzing
    1. Apply reducers
    2. Export GEE objects

1. Band arithmetic
    1. Remote sensing principles
    2. Spectral Indices using band arithmetic

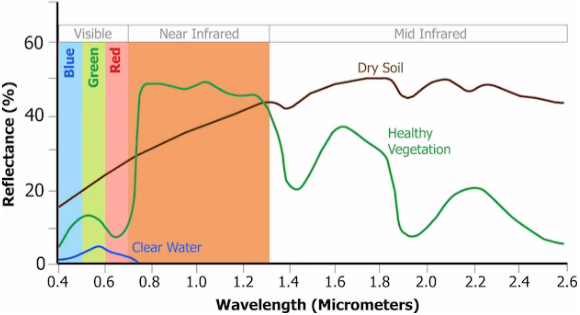
2. Thresholding & Masking
    1. Thresholding
    2. Masking

3. Analyzing
    1. Apply reducers
    2. Export GEE objects

Remote Sensing basic principle:

⇒ the amount of light reflected by the Earth surface (= **reflectance**) varies depending on both the *surface type* and the *wavelength* of the incident light

⇒ each land cover has a unique **spectral signature**



→ vegetation:

- in the *Visible (VIS)* range: reflects green light & absorbs blue and red light ⇒ appears green to our eye
- in the *Near Infrared (NIR)* range: reflectance increases dramatically ⇒ useful to detect vegetation

Spectral Indices:

⇒ **Spectral Indices** combine multiple bands (often with simple operations of subtraction and division) to help to distinguish particular land covers/use in an image

⇒ **Band arithmetic** is the process of adding, subtracting, multiplying, or dividing two or more bands from an image, and is the basis of many remote sensing analyses

⇒ Common spectral indices (ref):

- **NDVI** (Normalized Difference Vegetation Index)

$$NDVI = \frac{NIR - red}{NIR + red}$$

- **NDSI** (Normalized Difference Snow Index)

$$NDWI = \frac{green - SWIR}{green + SWIR}$$

- **NBRI** (Normalized Burned Ratio Index)

$$NDWI = \frac{NIR - SWIR}{NIR + SWIR}$$

- **EVI** (Enhanced Vegetation Index)

$$EVI = 2.5 \times \frac{\text{NIR} - \text{red}}{\text{NIR} + 6 \times \text{red} - 7.5 \times \text{blue} + 1}$$

Spectral Indices:

⇒ **Spectral Indices** combine multiple bands (often with simple operations of subtraction and division) to help to distinguish particular land covers/use in an image

⇒ **Band arithmetic** is the process of adding, subtracting, multiplying, or dividing two or more bands from an image, and is the basis of many remote sensing analyses

⇒ Common spectral indices (ref):

- **NDVI** (Normalized Difference Vegetation Index)

$$NDVI = \frac{NIR - red}{NIR + red}$$

- **NDSI** (Normalized Difference Snow Index)

$$NDWI = \frac{green - SWIR}{green + SWIR}$$

- **NBRI** (Normalized Burned Ratio Index)

$$NDWI = \frac{NIR - SWIR}{NIR + SWIR}$$

- **EVI** (Enhanced Vegetation Index)

$$EVI = 2.5 \times \frac{\text{NIR} - \text{red}}{\text{NIR} + 6 \times \text{red} - 7.5 \times \text{blue} + 1}$$

1.2. Spectral Indices using band arithmetic

Spectral Indices:

⇒ **Spectral Indices** combine multiple bands (often with simple operations of subtraction and division) to help to distinguish particular land covers/use in an image

⇒ **Band arithmetic** is the process of adding, subtracting, multiplying, or dividing two or more bands from an image, and is the basis of many remote sensing analyses

⇒ Common spectral indices (ref):

- **NDVI** (Normalized Difference Vegetation Index)

$$NDVI = \frac{NIR - red}{NIR + red}$$

- **NDSI** (Normalized Difference Snow Index)

$$NDWI = \frac{green - SWIR}{green + SWIR}$$

- **NBRI** (Normalized Burned Ratio Index)

$$NDWI = \frac{NIR - SWIR}{NIR + SWIR}$$

- **EVI** (Enhanced Vegetation Index)

$$EVI = 2.5 \times \frac{\mathrm{NIR} - \mathrm{red}}{\mathrm{NIR} + 6 \times \mathrm{red} - 7.5 \times \mathrm{blue} + 1}$$

Spectral Indices:

⇒ **Spectral Indices** combine multiple bands (often with simple operations of subtraction and division) to help to distinguish particular land covers/use in an image

⇒ **Band arithmetic** is the process of adding, subtracting, multiplying, or dividing two or more bands from an image, and is the basis of many remote sensing analyses

⇒ Common spectral indices (ref):

- **NDVI** (Normalized Difference Vegetation Index)

$$NDVI = \frac{NIR - red}{NIR + red}$$

- **NDSI** (Normalized Difference Snow Index)

$$NDWI = \frac{green - SWIR}{green + SWIR}$$

- **NBRI** (Normalized Burned Ratio Index)

$$NDWI = \frac{NIR - SWIR}{NIR + SWIR}$$

- **EVI** (Enhanced Vegetation Index)

$$EVI = 2.5 \times \frac{\mathrm{NIR} - \mathrm{red}}{\mathrm{NIR} + 6 \times \mathrm{red} - 7.5 \times \mathrm{blue} + 1}$$

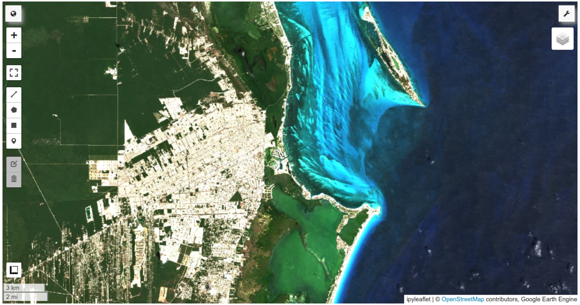
EX 1: NDVI (Normalized Difference Vegetation Index)

⇒ NDVI is a measure of the **greenness** of vegetation

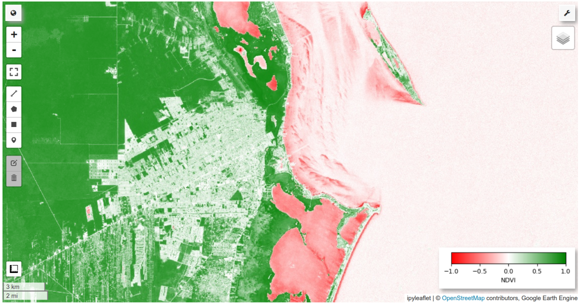
⇒ NDVI values range from -1 to 1:

- **low** values (≤ 0): water, bare soil, urban areas
- **high** values (≥ 0.5): vegetation

Natural color image


3 km
2 mi
ipyleaflet | © OpenStreetMap contributors, Google Earth Engine

NDVI image


3 km
2 mi
−1.0
−0.5
0.0
0.5
1.0
NDVI
ipyleaflet | © OpenStreetMap contributors, Google Earth Engine

EX 1: NDVI (Normalized Difference Vegetation Index)

⇒ NDVI is a measure of the **greenness** of vegetation

⇒ NDVI values range from -1 to 1

⇒ implementation in GEE:

```
# NDVI calculation from Image object in GEE (Sentinel-2 image)

# - using basic math operators
nir = image.select('B8')                 # Sentinel-2 nir band
red = image.select('B4')                 # Sentinel-2 red band
numerator = nir.subtract(red)            # band arithmetic: nir - red
denominator = nir.add(red)               # band arithmetic: nir + red
ndvi = numerator.divide(denominator)     # band arithmetic: numerator / denominator

# - using normalizedDifference method
ndvi = image.normalizedDifference(['B8', 'B4'])
```

EX 2: EVI (Enhanced Vegetation Index)

⇒ EVI is similar to the NDVI, it is used to quantify the **greenness** of vegetation

⇒ EVI however corrects for some *atmospheric conditions* and *canopy background noise* and is more sensitive in areas with dense vegetation (incorporates an “L” value to adjust for canopy background, “C” values as coefficients for atmospheric resistance, and values from the Blue band)

⇒ implementation in GEE:

```
# EVI calculation from Image object in GEE (Sentinel-2 image)
nir = image.select('B8')
red = image.select('B4')
blue = image.select('B2')

evi = image.expression(
    '2.5 * ((NIR - RED) / (NIR + 6 * RED - 7.5 * BLUE + 1))',
    {
        'NIR': nir,
        'RED': red,
        'BLUE': blue
    })
```

1. Band arithmetic
    1. Remote sensing principles
    2. Spectral Indices using band arithmetic

2. Thresholding & Masking
    1. Thresholding
    2. Masking

3. Analyzing
    1. Apply reducers
    2. Export GEE objects

Thresholding

⇒ **Thresholding** is a technique which uses a number (the *threshold value*) and *logical operators* to create a categorized image (pixels are partitioned into categories)

EX 1: thresholding an NDVI image into 2 classes (vegetation vs. non-vegetation):

1. select a *threshold value* above which areas are vegetated, e.g. 0.5
2. use a *logical operator* to binarize the NDVI pixels:

$$\text{NDVI} > 0.5 \Rightarrow 1 \text{ (vegetation)}$$
$$\text{NDVI} \leq 0.5 \Rightarrow 0 \text{ (non-vegetation)}$$

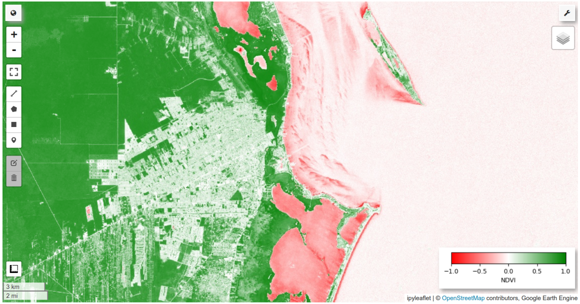
```
# NDVI binary thresholding in GEE
threshold = 0.5
img_thresh = ndvi.gt(threshold)  # logical operator "greater than" (gt) on ndvi image
```

Thresholding

⇒ **Thresholding** is a technique which uses a number (the *threshold value*) and *logical operators* to create a categorized image (pixels are partitioned into categories)

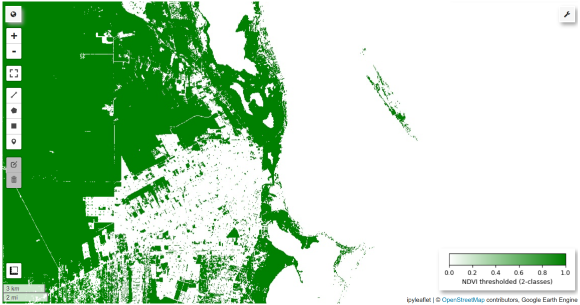
EX 1: thresholding an NDVI image into 2 classes (vegetation vs. non-vegetation):

NDVI image



ipyleaflet | © OpenStreetMap contributors, Google Earth Engine

NDVI image thresholded (2 classes)



ipyleaflet | © OpenStreetMap contributors, Google Earth Engine

Thresholding

⇒ **Thresholding** is a technique which uses a number (the *threshold value*) and *logical operators* to create a categorized image (pixels are partitioned into categories)

EX 2: thresholding an NDVI image into 3 classes (e.g., vegetation / non-vegetation / water):
⇒ implementation in GEE:

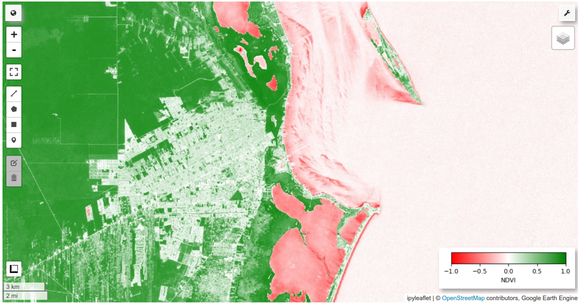
```
# NDVI advanced thresholding in GEE
threshold_1 = -0.1       # set water threshold
threshold_2 = 0.5        # set vegetation threshold

img_thresh = ee.Image(1)                                 # Initialize new thresholded image with all values = 1
img_thresh = img_thresh.clip(ndvi.geometry())           # Use clip to constrain size of the ndvi image
img_thresh = img_thresh.where(ndvi.lte(threshold_1), 0)         # Make all NDVI values <= threshold_1 equal 0
img_thresh = img_thresh.where(ndvi.gte(threshold_2), 2)   # Make all NDVI values >= threshold_2 equal 2
```

Thresholding

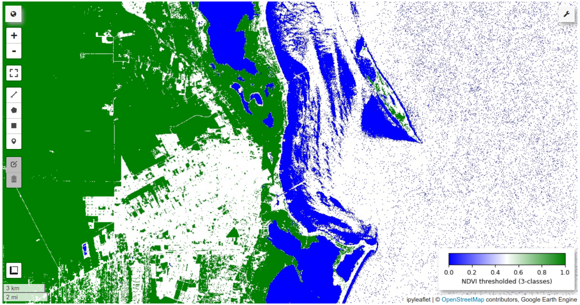
⇒ **Thresholding** is a technique which uses a number (the *threshold value*) and *logical operators* to create a categorized image (pixels are partitioned into categories)

EX 2: thresholding an NDVI image into 3 classes (e.g., vegetation / non-vegetation / water):

NDVI image


3 km
2 mi
−1.0
−0.5
0.0
0.5
1.0
NDVI
ipyleaflet | © OpenStreetMap contributors, Google Earth Engine

NDVI image thresholded (3 classes)

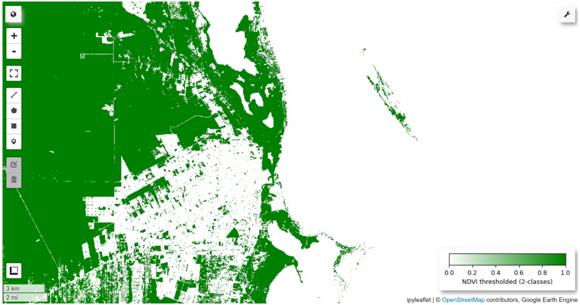

3 km
2 mi
0.0
0.2
0.4
0.6
0.8
1.0
NDVI thresholded (3-classes)
ipyleaflet | © OpenStreetMap contributors, Google Earth Engine

Masking

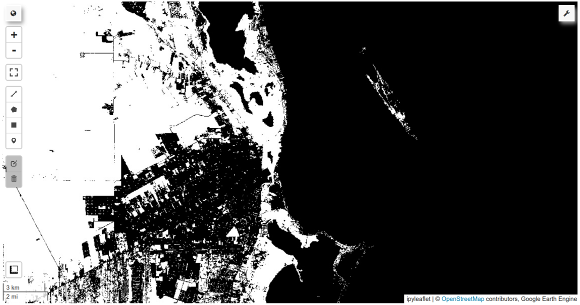
⇒ **Masking** an image is a technique that *removes specific areas* of an image (those covered by the mask) from being displayed or analyzed

EX: mask non-forest regions of thresholded NDVI image:

NDVI image thresholded (2 classes)


0.0
0.2
0.4
0.6
0.8
1.0
NDVI thresholded (2-classes)
3 km
2 mi
ipyleaflet | © OpenStreetMap contributors, Google Earth Engine

mask (white=keep, black=discard)

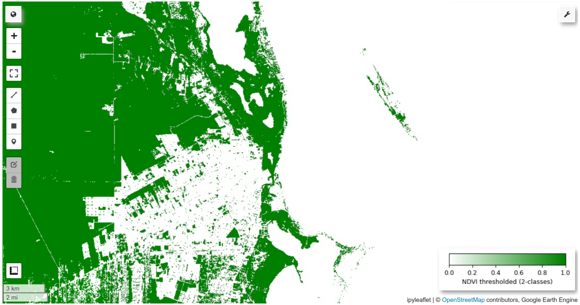

3 km
2 mi
ipyleaflet | © OpenStreetMap contributors, Google Earth Engine

Masking

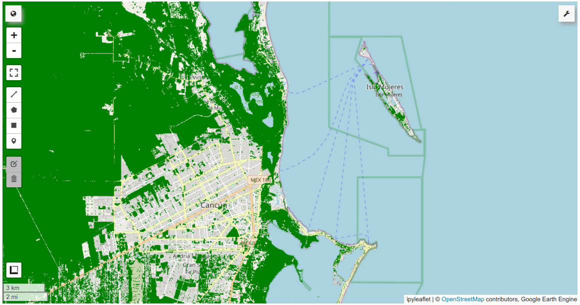
⇒ **Masking** an image is a technique that *removes specific areas* of an image (those covered by the mask) from being displayed or analyzed

EX: mask non-forest regions of thresholded NDVI image:

NDVI image thresholded (2 classes)


3 km
2 mi
0.0
0.2
0.4
0.6
0.8
1.0
NDVI thresholded (2-classes)
ipyleaflet | © OpenStreetMap contributors, Google Earth Engine

NDVI image masked (white=keep, black=discard)


Isla Mujeres
Cancún
MEX 180
3 km
2 mi
ipyleaflet | © OpenStreetMap contributors, Google Earth Engine

Masking

⇒ **Masking** an image is a technique that *removes specific areas* of an image (those covered by the mask) from being displayed or analyzed

EX: mask non-forest regions of thresholded NDVI image:
⇒ implementation in GEE:

```
# NDVI masking in GEE

mask = img_thresh.eq(1)                     # Create a binary mask of non-forest
img_masked = img_thresh.updateMask(mask)    # Update the img_thresh mask with the non-forest mask
mask_final = img_masked.mask()              # Updated mask

# Visualize masked image
Map.addLayer(img_masked, {'min': 0, 'max': 1, 'palette': ['green']}, 'Masked Forest Layer')

# Visualize updated mask
Map.addLayer(mask_final, {}, 'img_masked Mask')
```

1. Band arithmetic
   1. Remote sensing principles
   2. Spectral Indices using band arithmetic
2. Thresholding & Masking
   1. Thresholding
   2. Masking
3. Analyzing
   1. Apply reducers
   2. Export GEE objects

Reducers

⇒ **Reducers** are the way to aggregate data over *time*, *space*, *bands*, *arrays* and other data structures in Earth Engine:

- **time**: `imageCollection.reduce()`
- **space**: `image.reduceRegion()`, `image.reduceNeighborhood()`
- **bands**: `image.reduce()`
- **attributes** of FeatureCollections: `featureCollection.reduceColumns()`

⇒ how data is aggregated will be defined by `ee.Reducer` class: ee.Reducer.min, ee.Reducer.max, etc.

Reducers

⇒ **Reducers** are the way to aggregate data over *time*, *space*, *bands*, *arrays* and other data structures in Earth Engine:

- **time**: `imageCollection.reduce()`
- **space**: `image.reduceRegion()`, `image.reduceNeighborhood()`
- **bands**: `image.reduce()`
- **attributes** of FeatureCollections: `featureCollection.reduceColumns()`

⇒ how data is aggregated will be defined by `ee.Reducer` class: ee.Reducer.min, ee.Reducer.max, etc.

Reducers

⇒ **Reducers** are the way to aggregate data over *time*, *space*, *bands*, *arrays* and other data structures in Earth Engine:

- **time**: `imageCollection.reduce()`
- **space**: `image.reduceRegion()`, `image.reduceNeighborhood()`
- **bands**: `image.reduce()`
- **attributes** of FeatureCollections: `featureCollection.reduceColumns()`

⇒ how data is aggregated will be defined by `ee.Reducer` class: ee.Reducer.min, ee.Reducer.max, etc.

Reducers

EX 1: apply **morphological** operations (e.g., *erosion*, *dilation*) to the thresholded image to remove small isolated patches of vegetation/non-vegetation

⇒ use the image.reduceNeighborhood() method to apply a min/max operation to the neighborhood (*kernel*) around each pixel:

```
# Morphological operations in GEE

# Define a square, uniform kernel.
kernel = ee.Kernel.square(radius=5)

# Dilate by taking the max in kernel neighborhood
dilated = img_thresh.reduceNeighborhood(
    reducer=ee.Reducer.max(),
    kernel=kernel
)

# Erode by taking the min in kernel neighborhood
eroded = img_thresh.reduceNeighborhood(
    reducer=ee.Reducer.min(),
    kernel=kernel
);
```

Reducers

EX 2: calculate the *area* covered by the vegetation

⇒ use the `image.reduceRegion()` method to sum all pixel marked as vegetation:

```
# Create a pixel area image in which pixel value = pixel area in m2
img_pixArea = ee.Image.pixelArea()
mask_area = img_pixArea.updateMask(mask)

# Sum the area of vegetation pixels
crs = mask.projection()  # get coordinate reference system
crsTransform = mask.projection().getInfo()['transform']  # get coordinate reference system transform
geometry = mask.geometry()  # get region where to compute area

area = mask_area.reduceRegion(
    reducer=ee.Reducer.sum(),
    geometry=geometry,
    crs=crs,
    crsTransform=crsTransform,
    maxPixels=1e10,
)

# Fetch summed area property
square_meters = area.getNumber('area').round()
square_kilometers = square_meters.divide(1e6).round()
```

Reducers

EX 3: get the *mean NDVI* index from the MODIS Vegetation Indices collection (16-Day Global 500m since Feb-2000)

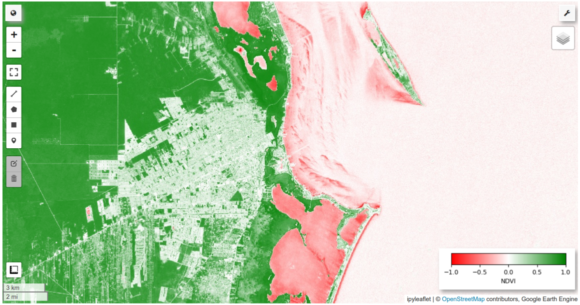
⇒ use the `imageCollection.reduce()` method

```
collection = ee.ImageCollection('MODIS/061/MOD13A1')  # Get the MODIS Terra Vegetation collection
collection_ndvi = collection.select('NDVI');  # Select the NDVI band
ndvi_mean = collection_ndvi.reduce(ee.Reducer.mean()) # Reduce the image collection to get the mean NDVI

# Visualize
mean_vis_params = {'min': 0, 'max': 9000, 'palette': ['ffffff', 'ce7e45', 'df923d', 'f1b555', 'fcd163',
 '99b718', '74a901', '66a000', '529400', '3e8601', '207401', '056201', '004c00', '023b01', '012e01',
  '011d01', '011301']}
Map.addLayer(ndvi_mean, mean_vis_params, 'Mean NDVI')
```

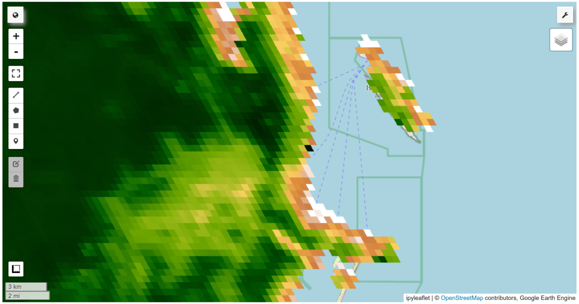
Reducers

EX 3: get the *mean NDVI* index from the MODIS Vegetation Indices collection (16-Day Global 500m since Feb-2000)

NDVI image (Sentinel-2)


−1.0
−0.5
0.0
0.5
1.0
NDVI
3 km
2 mi
ipyleaflet | © OpenStreetMap contributors, Google Earth Engine

NDVI image - mean since Feb-2000 (MODIS)


3 km
2 mi
ipyleaflet | © OpenStreetMap contributors, Google Earth Engine

Exporting

⇒ Exporting Earth Engine objects can be useful to either **save** the results of an analysis, or to **import** them into another software for further processing

⇒ The `geemap` library provides several methods to export GEE objects to various formats
For example:

- ee_to_numpy: extracts a rectangular region of pixels from an image into a Numpy array
- `ee_export_image`: download GeoTiff from a URL link
- `ee_export_image_to_drive`: save GeoTiff to Google Drive
- etc.

Exporting

EX: export a *region* of the image as a Numpy Array, and analyze it

```
# Export GEE object to Numpy array

# - Select region to extract
# region = ee.Geometry.Point(-86.85, 21.17).buffer(50*1000)  # Circle with radius in meters around a point
region = Map.draw_last_feature.geometry()   # Draw feature on Map an use it

# - Export numpy array
img_np = geemap.ee_to_numpy(
    image,                       # GEE image object
    region=region,               # Region to extract
    bands=['B4', 'B3', 'B2'],    # RGB bands
    scale=10                     # resolution in meters
    )
```