Introduction and data analysis with

Google Earth Engine

Training Course on
'Fundamentals of Remote Sensing & GIS and Oceanographic Applications'
April 08-12, 2024

Gaurav Khairnar

qb.khairnar-p@incois.gov.in

International Training Centre for operational Oceanography(ITCO),

INCOIS, Hyderabad, India

Index

Working with imageCollection

```
// Assign Geomety point using Longitude and Latitude in WGS 4326 Projection
var point_geometry = ee.Geometry.Point([84.80, 19.31])


 // ###################### Sentinel 2 Data Visualization ################################
// To  convert into remote sensing reflectance
function rrs(image){
  return image.divide(10000); }

var Sentinel2 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')//Sentinel2 image collection
              .filterDate('2020-01-01', '2020-01-30') // required daterange
              .filterBounds(point_geometry) // image of area of interest
              .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',20))// Pre-filter to get less cloudy granules.
              .map(rrs); // Remote Sensing Reflectance

/// visualization parameters for sentinel2
var Sentinel_visualization = {
  min: 0.0, // minimum value in image
  max: 0.3, // maximum value in image
  bands: ['B8', 'B4', 'B3'],// Selecting NIR, Red, Green for False color composit
};
```

```
// ############################ Landsat8 Data Visualization ##############################
// Applies scaling factors.
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 Landsat8 = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2') // Landsat image collection
    .filterDate('2020-01-01', '2020-01-30'); // required daterange
Landsat8 = Landsat8.map(applyScaleFactors); // applying scalling factors


/// visualization parameters fro landsat
var landsat_visualization = {
  bands: ['SR_B4', 'SR_B3', 'SR_B2'], // selecting Red, Green, Blue bands for True color composit
  min: 0.0, // minimum value in image
  max: 0.3, // maximum value in image
};


Map.setCenter(84.80, 19.31,10) // After running code Map center will be that coordinate ex.
Map.setCenter(long,lat,zoom level)


// Adding Layers in Map
 /// Map.addLayer(imagecollection,visualization parameters,name,visibility)
Map.addLayer(Landsat8, landsat_visualization, 'Landsat_RGB');
Map.addLayer(point_geometry,{},'Point',false)
Map.addLayer(Sentinel2,Sentinel_visualization, 'Sentinel2_RGB',false);
```

Composit and Mosaic

```
//var geometry = ee.Geometry.Point([72.25, 20.85])


// Sentinel Data Visualization
function rrs(image){
  return image.divide(10000);}


var Sentinel2 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
                .filterDate('2020-01-03', '2020-01-06').filterBounds(geometry)
                .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',20)).map(rrs);


var Sentinel_visualization = { min: 0.0,
  max: 0.3,
  bands: ['B8', 'B4', 'B5']};


// Selecting first image from imageCollection
var first = ee.Image(Sentinel2.toList(Sentinel2.size()).get(0)) // Selecting first image from image collection
// Selecting second image from imageCollection
var second = ee.Image(Sentinel2.toList(Sentinel2.size()).get(1)) // Selecting second image from image collection


var mosaic = Sentinel2.mosaic() // Combining images horizontally
var RGB = Sentinel2.select(['B4', 'B3', 'B2'])
var composit = RGB.mean() // Combining image vertically


Map.addLayer(first,Sentinel_visualization, 'Sentinel2_1');
Map.addLayer(second,Sentinel_visualization, 'Sentinel2_2');
Map.addLayer(mosaic,Sentinel_visualization,'Mosaic',false)
Map.addLayer(composit,{min: 0, max: 0.1}, 'Composite Image',false);
```

Data management tools

```
//############### Import shapefile geometry ##################
//Import shapefile from Assests tab


// Sentinel Data Visualization
function rrs(image){
  return image.divide(10000);}
var Sentinel2 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
                .filterDate('2020-01-20', '2020-01-25').filterBounds(Mumbai)
                // Pre-filter to get less cloudy granules.
                //.filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',20))
                 .map(rrs);

var Sentinel_visualization = {min: 0.0,
  max: 0.1,
  bands: ['B4', 'B3', 'B2']};

var Mumbai_image = Sentinel2.mosaic().clip(Mumbai)
 ///Clipping Image using Mumbai polygon shapefile
/// Clipping can not be perform on ImageCollection.
```

```
// ################## Indices Calculation ############################

var nir = Mumbai_image.select('B5'); // Selecting NIR band

var red = Mumbai_image.select('B4'); // Selecting Red band

var ndvi = nir.subtract(red).divide(nir.add(red)).rename('NDVI'); // Calculating NDVI =(NIR-Red)/(NIR+Red)


// visualization parameters for NDVI

var ndvi_visualization = { min: -1,

  max: 1,

  palette:['yellow', 'white', 'green']}


///######### Centroid of polygon ################

var Mumbai_centroid = Mumbai.geometry().centroid()

var long = ee.Number(Mumbai_centroid.coordinates().get(0))

var lat = ee.Number(Mumbai_centroid.coordinates().get(1))


//############## Buffer ##########################

var buffer = Mumbai.geometry().buffer(10000) // Units are in meters


//############## Intersection ##########################

var intersection = buffer.intersection(geometry,ee.ErrorMargin(1))


Map.setCenter(long.getInfo(),lat.getInfo(),10)

Map.addLayer(ndvi,ndvi_visualization, 'NDVI',false)

Map.addLayer(Mumbai_image,Sentinel_visualization, 'Mumbai',false)

Map.addLayer(Sentinel2,Sentinel_visualization, 'Sentinel2_RGB',false);

Map.addLayer(Mumbai,{},'Mumbai Boundary',false)

Map.addLayer(buffer,{},'Mumbai buffer',false)

Map.addLayer(intersection,{},'intersection',false)
```

Spectral signature extraction

```
var point1 = ee.Geometry.Point([92.68325,12.54871])

var point2 = ee.Geometry.Point([92.68972,12.54261])


function rrs(image){

  return image.divide(10000);}


var Sentinel2_imageCollection = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')//Sentinel2 image collection

            .filterDate('2023-01-20', '2023-01-26') // required daterange

            .filterBounds(point1) // image of area of interest

            .map(rrs); // Remote Sensing Reflectance


// Selecting spectral bands

var sentinel = Sentinel2_imageCollection.first().select(['B2','B3','B4','B5','B6','B7','B8','B8A','B9'])



// plotting spectral signature curve

var options = {

title: 'Spectral signature ', // Title of chart

hAxis: {title: 'wavelength'}, //horizontal axis name

vAxis: {title: 'reflectance'}, // vertical axis name

lineWidth: 2, // linewidth

pointSize: 4, // point size

series: {

0: {color: '50d643',labelInLegend: 'point1'}, // color and legend of 1st line

1: {color: 'ff0000',labelInLegend: 'point2'}// color and legend of 2nd line


}};
```

```
var chart = ui.Chart.image.regions({
  image:sentinel, // Image data
  regions:[point1,point2], // points of which specral signature required

}).setOptions(options)



print(chart)
Map.setCenter(92.68325,12.54871)
Map.addLayer(point1,{},'Point1')
Map.addLayer(point2,{color:'red'},'Point2')
```

Timeseries analysis

```
var geometry = ee.Geometry.Point([92.68325825556089,12.548719092511044])


// Sentinel Data Visualization
function rrs(image){
  var scaled = image.divide(10000)
  return scaled.copyProperties(image,['system:index','system:time_start']);
}


var Sentinel2 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
                .filterDate('2020-01-01', '2021-01-01').filterBounds(geometry)
                // Pre-filter to get less cloudy granules.
                .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',15))
                .map(rrs);



var Sentinel_visualization = {
  min: 0.0,
  max: 0.1,
  bands: ['B4', 'B3', 'B2'],
};
```

```
// ################## Indices Calculation ##############################
var calculateNDVI = function(image) {
  var ndvi = image.normalizedDifference(['B8', 'B4']).rename('NDVI');
  return image.addBands(ndvi);
};

// Map the NDVI calculation function over the Image Collection
var sentinelCollectionWithNDVI = Sentinel2.map(calculateNDVI);




var options = {
title: 'NDVI Timeseries',
hAxis: {title: 'Months'},
vAxis: {title: 'NDVI'},
lineWidth: 2,
pointSize: 4,
series: {
0: {color: 'ff0404',labelInLegend: 'NDVI'},
}};

var chart = ui.Chart.image.seriesByRegion({
  imageCollection:sentinelCollectionWithNDVI.select(['NDVI']),
  regions: geometry,
  reducer : ee.Reducer.mean()
  }).setOptions(options)

print(chart)
```

Export data

```
// Sentinel Data Visualization
function rrs(image){
  return image.divide(10000);}

var Sentinel2 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
                .filterDate('2020-01-20', '2020-01-25').filterBounds(geometry)
                // Pre-filter to get less cloudy granules.
                //.filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',20)).map(rrs);

var Sentinel_visualization = {
  min: 0.0,
  max: 0.1,
  bands: ['B4', 'B3', 'B2'],
};

var Mumbai_image = Sentinel2.mosaic().clip(geometry) ///Clipping Image using Mumbai polygon shapefile
/// Clipping can not be perform on ImageCollection.


Export.image.toDrive({
  image:Mumbai_image,
  description:'Mumbai_clip',
  scale:10,
  region:geometry
})
```