manuelcampagnolo Create cloud_screening_and_temporal_charts_with_code_editor.md

aa59981 · 2 weeks ago History

Smooth introduction to the GEE code editor with examples

All examples use Sentinel-2 data.

Access image collection (Sentinel-2)

- Access, filter and plot Sentinel-2 image collection
  - Link to script for basic access to Sentinel-2 data: basic_S2_composite.js

The script accesses Sentinel-2, level 2A images and it filters by dates and by bounds: here, the region of interest `geometry` is a single point defined by its coordinates. All Sentinel-2 tiles that *intersect* the geometry are selected. `CLOUDY_PIXEL_PERCENTAGE` is an `Image` property and can be used to sort or filter the `ImageCollection`. Note that sorting the collection by the property `CLOUDY_PIXEL_PERCENTAGE` should be applied last since it is computationally more demanding.

```
// ROI: in this case it is a single point determined by its
longitude and latitude
var geometry = ee.Geometry.Point([-9.18498, 38.70708]);

// access image collection, filter for location and range of
dates
// sort by percentage of clouds (most cloudier first)
var S2 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
                .filterBounds(geometry)
                .filterDate('2024-06-01', '2024-09-30')
                .select(['B8', 'B4', 'B3','B2'])
                .sort('CLOUDY_PIXEL_PERCENTAGE',true);

// center map; 16 is the zoom level; 17 would zoom in further
Map.centerObject(geometry, 16);

// add true color composite layer to the map
Map.addLayer(S2.first(), {bands: ['B4', 'B3', 'B2'], min: 0,
max: 2500}, 'Sentinel-2 level 2A RGB=432');

// print to console
print(S2);

// Add geometry to the map
Map.addLayer(geometry, {color: 'red'}, 'Vinha ISA');
```

If you want to plot a false color composite, you can use instead

```
Map.addLayer(S2.first(), {bands: ['B8', 'B4', 'B3'], min:
[0,0,0], max: [4500, 3500, 3500]}, 'Sentinel-2 level 2A
RGB=843');
```

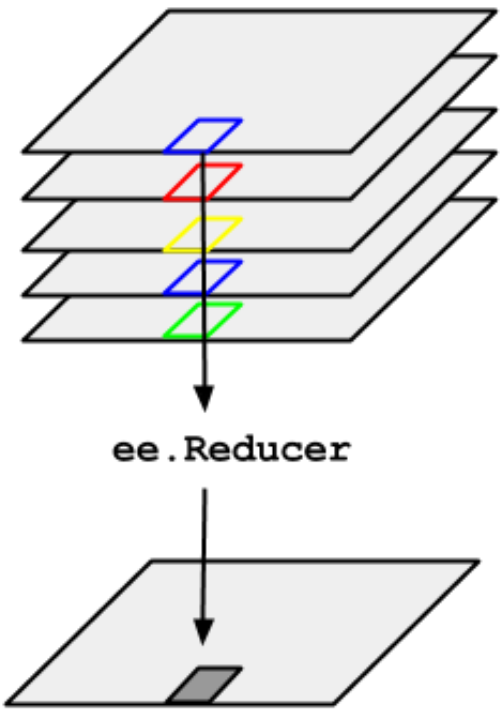
Create simple (median) temporal composite; temporal reducer

▾ Select images with low cloud cover and combine them into a single image

- Link to script for to create basic Sentinel-2 (median) temporal composite: basic_temporal_composite.js

The idea is to filter the Sentinel-2 image collection using the property `CLOUDY_PIXEL_PERCENTAGE`. Only images with less than 10% cloud cover are selected. Then selected images are combined with a *temporal reducer* which can be for instance the `mean` or the `median`.

```
// ROI: in this case it is a single point determined by its
longitude and latitude
var geometry = ee.Geometry.Point([-9.18498, 38.70708]);

// access image collection, select 10 m bands, filter for
location and range of dates
var S2 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
                .select(['B2','B3','B4','B8'])
                .filterBounds(geometry)
                .filterDate('2024-01-01', '2024-03-01')

// filter using property
var filtered =
S2.filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 10));

// reduce image collection to image
var S2clear=filtered.median()

// center map; 13 is the zoom level; 14 would zoom in more
Map.centerObject(geometry, 13);

// simple set of parameters for visualization
var vizParams={bands: ['B8', 'B4', 'B3'], min: 0, max: 3000}

// add layer
Map.addLayer(S2clear, vizParams, 'Sentinel-2 level 2A,
RGB=843, Jan 1-Mar 1, 2024');
```



In the example above, `median` is applied to all values of the image collection for the same pixel. As a result, the date for each pixel of the reduced image can be distinct: for instance for one pixel the median value could correspond to `2022-01-05` while for a neighbor pixel the date could be , say, `2022-02-10'.

Create a new band (e.g. NDVI) and add it to the image

▾ Create an index with normalized difference; add index to image bands
In remote sensing, it is very common to use an operation called *normalized difference* between two bands to compute an index. The most well-known index is the NDVI which measures the *greenness* of the land cover.

We could created those indices with an expression or we can simply use the *normalized difference* operation available in GEE (see https://developers.google.com/earth-engine/apidocs/ee-image-normalizeddifference).

```
// image needs to be defined, and has to have bands names B8
and B4

// create new band NDVI: notice that values are between −1 and
1.
var ndvi = image.normalizedDifference(['B8',
'B4']).rename('NDVI');

// add band to image
image = image.addBands([ndvi])
```

Create a basic temporal chart for NDVI

▾ Function, map and temporal chart

- Link to script for access Sentinel-2 data and create a basic NDVI chart: basic_NDVI_chart.js

The idea is to add the NDVI band to each image of a Sentinel-2 collection, and plot the NDVI values at a certain location along time with `ui.Chart.image.seriesByRegion`: see https://developers.google.com/earth-engine/guides/charts_overview and https://developers.google.com/earth-engine/guides/charts_image_collection for an overview of charts in GEE.

```
// ROI: in this case it is a single point determined by its
longitude and latitude
var geometry = ee.Geometry.Point([-9.18498, 38.70708]);

// access image collection, filter for location and range of
dates
// sort by percentage of clouds (most cloudier first)
var S2 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
      .filterBounds(geometry)
      .filterDate('2022-06-01', '2024-09-30')
      .select(['B8', 'B4'])

// center map; 16 is the zoom level; 17 would zoom in further
Map.centerObject(geometry, 16);

// print to console
print(S2);

// Add geometry to the map
Map.addLayer(geometry, {color: 'red'}, 'Vinha ISA');

// Function that adds an NDVI band to an image with B4 and B8
var add_ndvi_to_s2 = function(image) {
  var ndvi = image.normalizedDifference(['B8',
'B4']).rename('NDVI');
  return image.addBands([ndvi]);
};

// Add NDVI to all the images of the collection
var S2 = S2.map(add_ndvi_to_s2)

// Create chart
var chart =
    ui.Chart.image
        .seriesByRegion({
          imageCollection: S2,
          band: 'NDVI',
          regions: geometry,
          reducer: ee.Reducer.mean(),
          scale: 10,
          xProperty: 'system:time_start'
        });

print(chart);
```

NDVI chart with cloud screening at the pixel level with Sentinel-2 QA band

▾ Cloud screening with Sentinel-2 QA band

- Link to script for access Sentinel-2 data and create a basic NDVI chart with built-in cloud screening: QA_screening_NDVI_chart.js

In this script, we filter clouds using two distinct strategies:

- Using the property `CLOUDY_PIXEL_PERCENTAGE` for the whole tile: we select only tiles that have a cloud cover under a certain threshold we define;
- Using the built-in *band* `QA60` of the Sentinel-2 Surface Reflectance product; this allow us to mask individual pixels within an image independently of the cloud cover.

```
// ROI: in this case it is a single point determined by
longitude and latitude
var geometry = ee.Geometry.Point([-9.18498, 38.70708]);

/**
 * 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
 * https://developers.google.com/earth-
engine/datasets/catalog/COPERNICUS_S2_SR_HARMONIZED
 */
function maskS2clouds(image) {
  var date = image.get('system:time_start'); // otherwise,
this property is lost
  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).set('system:time_start',
date);
}


// access image collection, filter for location and range of
dates
// use built-in cloud screening (tile and pixel level)
var S2 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
      .filterBounds(geometry)
```

```
      .filterDate('2022-06-01', '2024-09-30')
      .select(['B8', 'B4','QA60'])
      // Pre-filter to get less cloudy granules.
      .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',20))
      .map(maskS2clouds);

// center map;
Map.centerObject(geometry, 16);

// print to console
print(S2);

// Add geometry to the map
Map.addLayer(geometry, {color: 'red'}, 'Vinha ISA');

// Function that adds an NDVI band to an image with B4 and B8
var add_ndvi_to_s2 = function(image) {
  var ndvi = image.normalizedDifference(['B8',
'B4']).rename('NDVI');
  return image.addBands([ndvi]);
};

// Add NDVI to all the images of the collection
var S2 = S2.map(add_ndvi_to_s2)

// Create chart
var chart =
    ui.Chart.image
        .seriesByRegion({
          imageCollection: S2,
          band: 'NDVI',
          regions: geometry,
          reducer: ee.Reducer.mean(),
          scale: 10,
          xProperty: 'system:time_start'
        });

print(chart);
```

NDVI chart with cloud screening at the pixel level with Cloud Score+ for Sentinel-2

Cloud Score+ is a Google product that is derived from Sentinel-2 [https://ieeexplore.ieee.org/document/10208818] and that can be combined with Sentinel-2 imagery to mask pixels with cloud score above some given threshold. The code below uses the `linkCollection` method to combine the Sentinel-2 collection with the Cloud Score+ collection. By default, the match is based on the `system:index` image property.

```
// ROI: in this case it is a single point determined by its
longitude and latitude
var geometry = ee.Geometry.Point([-9.18498, 38.70708]);

// Cloud Score+ image collection. Note Cloud Score+ is
produced from Sentinel-2
// Level 1C data and can be applied to either L1C or L2A
collections.
var csPlus =
ee.ImageCollection('GOOGLE/CLOUD_SCORE_PLUS/V1/S2_HARMONIZED');

// Use 'cs' or 'cs_cdf', depending on your use case; see docs
for guidance.
var QA_BAND = 'cs';
// The threshold for masking; values between 0.50 and 0.65
generally work well.
// Higher values will remove thin clouds, haze & cirrus
shadows.
var CLEAR_THRESHOLD = 0.60;

// access image collection, filter for location and range of
dates
// link S2 collection with csPlus and update mask using
QA_band
var S2 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
      .filterBounds(geometry)
      .filterDate('2022-06-01', '2024-09-30')
      .select(['B8', 'B4'])
      .linkCollection(csPlus, [QA_BAND])
      .map(function(img) {
        return
img.updateMask(img.select(QA_BAND).gte(CLEAR_THRESHOLD));
    })

// print to console
print(S2);

// center map; 11 is the zoom level; 12 would zoom in further
Map.centerObject(geometry, 16);

// Add geometry to the map
Map.addLayer(geometry, {color: 'red'}, 'Vinha ISA');
```

```
// Function adds an NDVI band to an image
var add_ndvi_to_s2 = function(image) {
  var ndvi = image.normalizedDifference(['B8',
'B4']).rename('NDVI');
  return image.addBands([ndvi]);
};

// add NDVI band to each image
var S2 = S2.map(add_ndvi_to_s2)

var chart =
    ui.Chart.image
        .seriesByRegion({
          imageCollection: S2,
          band: 'NDVI',
          regions: geometry,
          reducer: ee.Reducer.mean(),
          scale: 10,
          xProperty: 'system:time_start'
        })

print(chart);
```

Create NDVI charts for a set of locations

▾ Multi-point NDVI charts with Cloud Score+ screening

- Link to script for access Sentinel-2 data and create a multi-point NDVI chart with cs-Plus cloud screening: points_cs_charts.js

The Google Code Editor allows us to digitize geometries (points, lines or polygons) and add those geometries to our scripts. This can be used to extract a list of point coordinates. Then, the coordinates can be copied into a list and used to define a feature collection.

```
// ROI: in this case it is a feature collection of points
// Firstly, we obtain a list os points possibly by digitizing
with the code editor interactive tools
var multipoints =[[-9.18511947486878, 38.70673673565854],
         [-9.185698832015996, 38.707121861392295],
         [-9.184983887235997, 38.70708122565936]];

// the following code read each point from the list, and adds
it as a `ee.Geometry.Point` to a feature collection.
// As a result, the variable  `geometry` below is a feature
collection of single part point geometries.
var geometry =
```

```
ee.FeatureCollection(multipoints.map(function(p){
  var point = ee.Feature(ee.Geometry.Point(p), {})
  return point
}))

print(geometry)

// Cloud Score+ image collection. Note Cloud Score+ is
produced from Sentinel-2
// Level 1C data and can be applied to either L1C or L2A
collections.
var csPlus =
ee.ImageCollection('GOOGLE/CLOUD_SCORE_PLUS/V1/S2_HARMONIZED');

// Use 'cs' or 'cs_cdf', depending on your use case; see docs
for guidance.
var QA_BAND = 'cs';
// The threshold for masking; values between 0.50 and 0.65
generally work well.
// Higher values will remove thin clouds, haze & cirrus
shadows.
var CLEAR_THRESHOLD = 0.60;

// access image collection, filter for location and range of
dates
// sort by percentage of clouds (most cloudier first)
var S2 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
      .filterBounds(geometry)
      .filterDate('2022-06-01', '2024-09-30')
      .select(['B8', 'B4'])
      .linkCollection(csPlus, [QA_BAND])
      .map(function(img) {
        return
img.updateMask(img.select(QA_BAND).gte(CLEAR_THRESHOLD));
    })


// center map; 16 is the zoom level; 17 would zoom in further
Map.centerObject(geometry, 16);

// print to console
print(S2);

// Add geometry to the map
Map.addLayer(geometry, {color: 'red'}, 'Vinha ISA');

// Add NDVI to one image
var add_ndvi_to_s2 = function(image) {
  var ndvi = image.normalizedDifference(['B8',
'B4']).rename('NDVI');
  return image.addBands([ndvi]);
```

```
};

// Add NDVI to all images
var S2 = S2.map(add_ndvi_to_s2)

// Create chart with options
var chart =
    ui.Chart.image
        .seriesByRegion({
          imageCollection: S2,
          band: 'NDVI',
          regions: geometry,
          reducer: ee.Reducer.mean(),
          scale: 10,
          xProperty: 'system:time_start'
        })
        .setOptions({
          interpolateNulls: true,
          title: 'NDVI Value by Date',
          hAxis: {title: 'Date', titleTextStyle: {italic:
false, bold: true}},
          vAxis: {
            title: 'NDVI',
            titleTextStyle: {italic: false, bold: true}
          },
          lineWidth: 2,
          colors: ['blue','red','green'], //['blue', 'yellow',
'green','red','brown','purple'],
        });

print(chart);
```

Export an image to Google Drive as a geotiff file

▾ Export.image.toDrive

In this exercise, we creta e cloud masked

```
// ROI: in this case it is a single point determined by its
longitude and latitude
var geometry = ee.Geometry.Point([-9.18498, 38.70708]);

// Cloud Score+ image collection. Note Cloud Score+ is
produced from Sentinel-2
// Level 1C data and can be applied to either L1C or L2A
collections.
var csPlus =
ee.ImageCollection('GOOGLE/CLOUD_SCORE_PLUS/V1/S2_HARMONIZED');
```

```
// Use 'cs' or 'cs_cdf', depending on your use case; see docs
for guidance.
var QA_BAND = 'cs';
// The threshold for masking; values between 0.50 and 0.65
generally work well.
// Higher values will remove thin clouds, haze & cirrus
shadows.
var CLEAR_THRESHOLD = 0.60;

// access image collection, filter for location and range of
dates
// link S2 collection with csPlus and update mask using
QA_band
// at the end, create a single image  by reducing with median
var S2clear =
ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
      .filterBounds(geometry)
      .filterDate('2024-07-30', '2024-09-30')
      .select(['B8', 'B4','B3'])
      .linkCollection(csPlus, [QA_BAND])
      .map(function(img) {
        return
img.updateMask(img.select(QA_BAND).gte(CLEAR_THRESHOLD))
      .median();
    })

// export to drive
// Set the export "scale" and "crs" parameters
// The defined region means that the exported image is going
to be 2000 m wide
Export.image.toDrive({
  image: S2clear,
  description: 'S2_screened_for_clouds', // file name
  folder: 'agricultura_digital',
  region: geometry.buffer(1000),
  scale: 10,
  crs: 'EPSG:3763' // Portuguese official CRS (meters)
});
```

Suggestion: Try exporting geometry to *shapefile* following instructions on https://developers.google.com/earth-engine/guides/exporting_tables.