Google_Earth_Engine_Tutorial-Cloud-Free_Composites

July 23, 2019

1 Introduction to Google Earth Engine

_____ ## Creating cloud-less mosaics

In the following page we will walk through the basics for using google earth engine to do remote sensing at large scale with minimal coding experience.

The objectives of this tutorial are as follows:

- Browse and find relevant data
- Create cloud-free mosaics
- Apply new code examples for Sentinel
- Download data
- Apply techniques to other data

---

2 1 Browse and find relevant data

First we need to be able to find relevant data. Google Earth engine hosts a variety of data

Included themes

- Weather and Climate
  - Temperatures
  - Precipitation
  - Drought
- Land Use
  - Agriculture
  - Urban extent
  - Population estimates
  - Deforestation

  - Water bodies

- Pollution
  - Aerosols
  - Sulphur Dioxide
  - Ozone
- Raw remote sensed data
  - True color images
  - Other bands (thermal, IR etc)
- Topography
  - Elevation slope etc
- Policy
  - Daytime fishing hours
  - Wildfire

Let's go to `https://developers.google.com/earth-engine/datasets/` Click on view all datasets, then search for a topic of interest

2.0.1 *Activity*

With your neighbor do the following:

1) Find one dataset of interest
2) Read the description of it, time period available, coverage etc
3) Discuss potential use case
4) Discuss potential limitations of the data

---

3 2 View satellite data

Let's start by searching for

`Sentinel-2`

then click on "Sentinel-2 MSI:... Level 1-c"

Sentinel-2 is a wide-swath, high-resolution, multi-spectral imaging mission supporting Copernicus Land Monitoring studies, including the monitoring of vegetation, soil and water cover, as well as observation of inland waterways and coastal areas.

Now click on the "Open in Editor" button, then login using your google credentials.

Dataset Availability

2015-06-23T00:00:00 - Present

Dataset Provider

European Union/ESA/Copernicus

Earth Engine Snippet

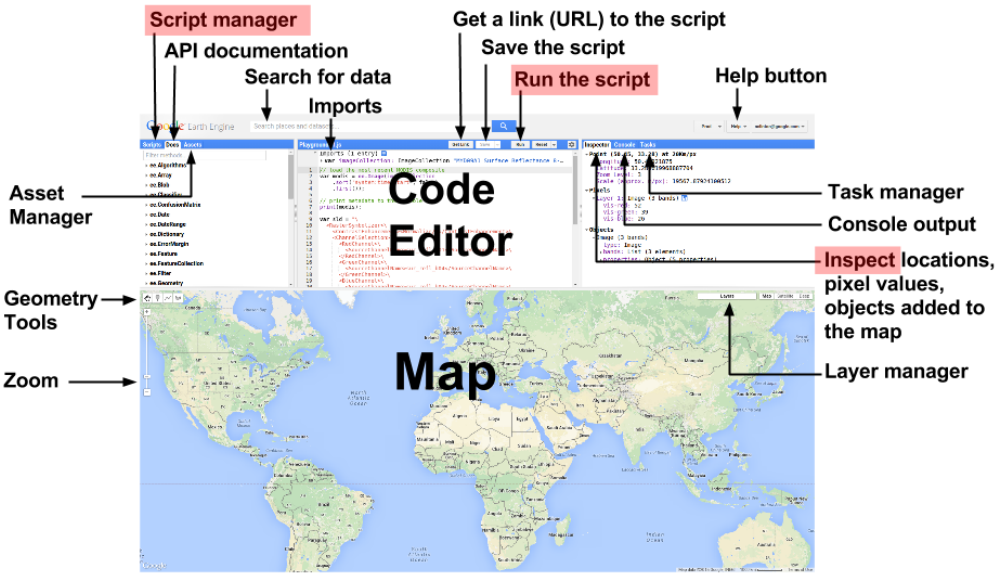
```
ee.ImageCollection("COPERNICUS/S2")
```

Tags

3.0.1 Intro to the code editor

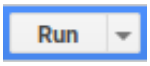
We are going to be running our code on google's servers, so they built a coding environment for us to use.

The parts we will use the most are the - Code editor - Map - Script manager - the Inspector tab - the Tasks tab



3.0.2 *Activity*

With your neighbor do the following:

Click "run" 

TASKS

1) Figure out how to toggle on and off the sentinel image 'layer'

2) Figure out how to save your script, name it Belize_Sentinel , *and then figure out where it is saved.*

3) Reading through the code, figure out what the following is:

```
// Map the function over one year of data and take the median.
// Load Sentinel-2 TOA reflectance data.
var dataset = ee.ImageCollection('COPERNICUS/S2')
                  .filterDate('2018-01-01', '2018-06-30')
                  // Pre-filter to get less cloudy granules.
                  .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
                  .map(maskS2clouds);
```

4) Using the above code, figure out how to change the date range from Jan 1 2017 - Dec 30 2017. Then press the "run" button.

5) Find the line of code that "centers" the map at a specific longitude and latitude. Update it to the center of Belize, then zoom out to see the country, then press "run".

Lon Lat Zoom

```
Map.setCenter(-9.1695, 38.6917, 12);
```

There are two other main components to the code one function called **mask2cloud** this uses data stored on an image called QA60 to help filter out clouds. It uses a "mask" to indicate which pixels have a cloud by assigning them the value of 0 or 1.

```
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);
}
```

The final chunk of code does two things, it tells the computer to take the **median** image and show it on our screen. Why median? Remember we are grabbing all images from Sentinel 2 (comes once ~ every 2 days), and filtering out a few cloudy images. So we have close to 180 images for this region. We need to summarize this data on a pixel by pixel basis, but which pixel value should we choose? Probably median or mean because the represent the middle of the distribution.

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

Map.setCenter(-9.1695, 38.6917, 12);
Map.addLayer(dataset.median(), rgbVis, 'RGB');
```

3.0.3 *Activity*

With your neighbor do the following:

TASKS

1) Play around the the function used to summarize a year's worth of data.

```
Map.addLayer(dataset.median(), rgbVis, 'RGB');
```

Try replacing `median()` with `min()`. Then press 'run'. What is happening?

2) Try `max()`. Then press 'run'. What in the world is happening now?

---

4 3 Apply code examples for Sentinel

4.1 A. Variables

In this next section we are going to learn a few more basics. In particular we are going to learn how to store data in a new 'variable', then we want to clip the images to just covering Belize, and then download that data to google drive.

4.1.1 What are variables?

Variables are simply storage containers like a Tupperware container with a name on it. You can use a variable to store anything: numbers, rasters, data collections, shapefiles etc.

You create a variable by assigning a value to a name

```
var weekly_income = 154```

You can then use that variable to make calculations:

```javascript
var annual_income = 154 * 52
print(annual_income)
8,008
```

4.1.2 Using variables

In this case we might want to store the minimum and median sentinel-2 data for 2017. Let's create two variables by inserting the following into your code:

```
var median_image = dataset.median()
var min_image = dataset.min() ```

Now let's add them as layers to make sure it worked:

```javascript
Map.addLayer(median_image, rgbVis, 'median');
Map.addLayer(min_image, rgbVis, 'min');
```

Press "Run" to see the results.

4.2 B. Crop images to Belize

It is often helpful to limit your data to a particular boundary (country, district, etc). In this example you will see how to clip your images using polygons.

Ok first let's import a shapefile of Belize and add it to the map. Please all the following to the bottom of your code:

```
// Get the boundary of Belize from Google's Fusion Table
// Change Country name for different location

var country_boundary = ee.FeatureCollection('USDOS/LSIB/2013')
    .filter(ee.Filter.eq('name', 'BELIZE'));

Map.addLayer(country_boundary, {color: 'FF0000'}, 'Belize');
```

Press "Run" and see if the polygon was added.

Now we need to see if we can clip the image to the boundary. Luckily, there is a 'clip' function that makes this easy!

Insert and run the following code:

```
var median_image_clip = median_image.clip(country_boundary)
var min_image_clip = min_image.clip(country_boundary)

Map.addLayer(median_image_clip, rgbVis, 'median_clip');
Map.addLayer(min_image_clip, rgbVis, 'min_clip');
```

Try toggling on and off the other layers to see if this worked.

4.3 C. Download data

The final step then is to download our two new layers.

This is done in two steps

1) adding the download code,
2) starting the download task

4.3.1 Inserting the download code

Please insert the following at the bottom of your code:

```
   javascript  // Export image   Export.image.toDrive({     description:"Belize_median_Sentinel
//export file name     image:median_image_clip,                    //data to
export     region:country_boundary.bounds,            //always add a boundary
fordownload     folder:"GEE",                              //destination on
google drive     crs: "EPSG:4326",                          //contert to lat lon
WGS1984     skipEmptyTiles: true,                        //dont download anything
that is empty     scale: 30,                                  //should be set to
resolution of pixel     maxPixels:9000000000                        // just do
this every time   })
```

Note in `country_boundary.bounds`, the `.bounds` converts the complex multi polygon into a simple box that covers the full extent (or bounds) of the country shapefile. `.bounds` will always need to be appended to the name of any polygon you might want to use here.

For more info on exporting data go here.

4.3.2 *Activity*

With your neighbor do the following:

TASKS

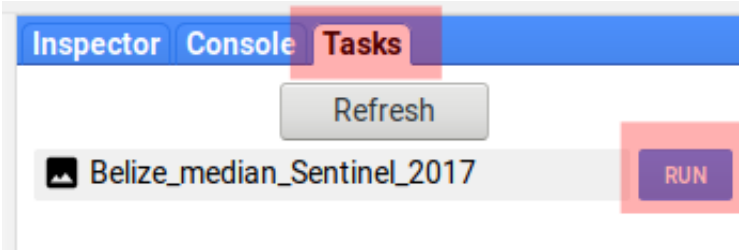
1) Update the `description` and `image` values to download and name the `min_image_clip` variable

Now press "Run"... what happens? Nothing yet.

4.3.3 Starting a download task

We need to tell google earth engine that we want to start a download task.

In order to start the download 1) Click on the "task" tab 2) Click on "RUN" for both tasks



- **Save your script!**

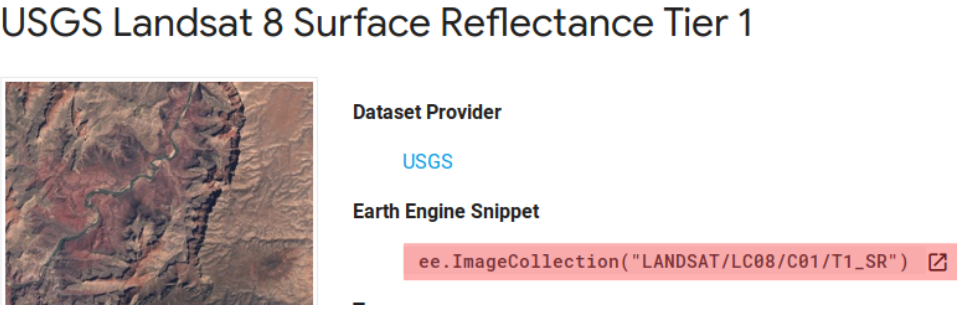
---

5 Apply what you learned

In the following task we are going to see if we can do something similar with another dataset. This time we will be using LandSat.

Let's follow the link here: LandSat 8 Tier 1 Surface Reflectance

Open up the example script here:



5.0.1 *Activity*

With your neighbor do the following:

TASKS

1) Center the image over belize with a zoom level of about 10
2) Change the date range to cover all of 2017
3) Create a new variable to store the median() image
4) Add the Belize shapefile
5) Clip the median image to Belize
6) Download the median image
7) Save your script!