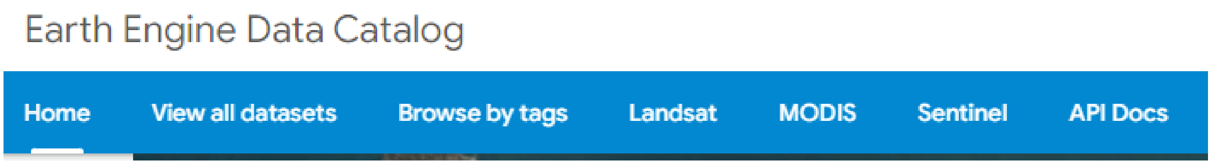
Department of Geography

CCST9083 Earth as Seen by Satellite

Laboratory 1: Introduction to GEE

The Google Earth Engine (GEE) is a cloud-based computing and analysis tool for (geo-)spatial data, powered by the Google Cloud Platform. It can freely be used by everyone to view, compute and evaluate satellite data. One of the most appealing advantages of using the Google Earth Engine is the vast amount of different data available for processing and analysis:



1. Landsat Collections

The Landsat program is the longest-running enterprise for acquisition of satellite imagery of Earth. It is a joint NASA / USGS program. It has been observing Earth continuously since 1972. The spectral and thermal data provided by Landsat sensors are an essential component of many Earth monitoring and research projects.

	Landsat 1-5 MSS	Landsat 4 TM	Landsat 5 TM	Landsat 7 ETM+	Landsat 8 OLI/TIRS	Landsat 9 OLI-2/TIRS-2
Data availability	1972–1999	1982–1993	1984–2012	1999–2021	2013–Present	2021–Present

2. Sentinel Collections

The Sentinels are a constellation of satellites developed by ESA to operationalize the Copernicus program, which include

- All-weather radar images from Sentinel-1 (2014 – Present)
- High-resolution optical images from Sentinel-2 (2015 – Present)
- Ocean and land data suitable for environmental and climate monitoring from Sentinel-3 (2016 – Present)
- Air quality data from Sentinel-5 (2018 – Present)

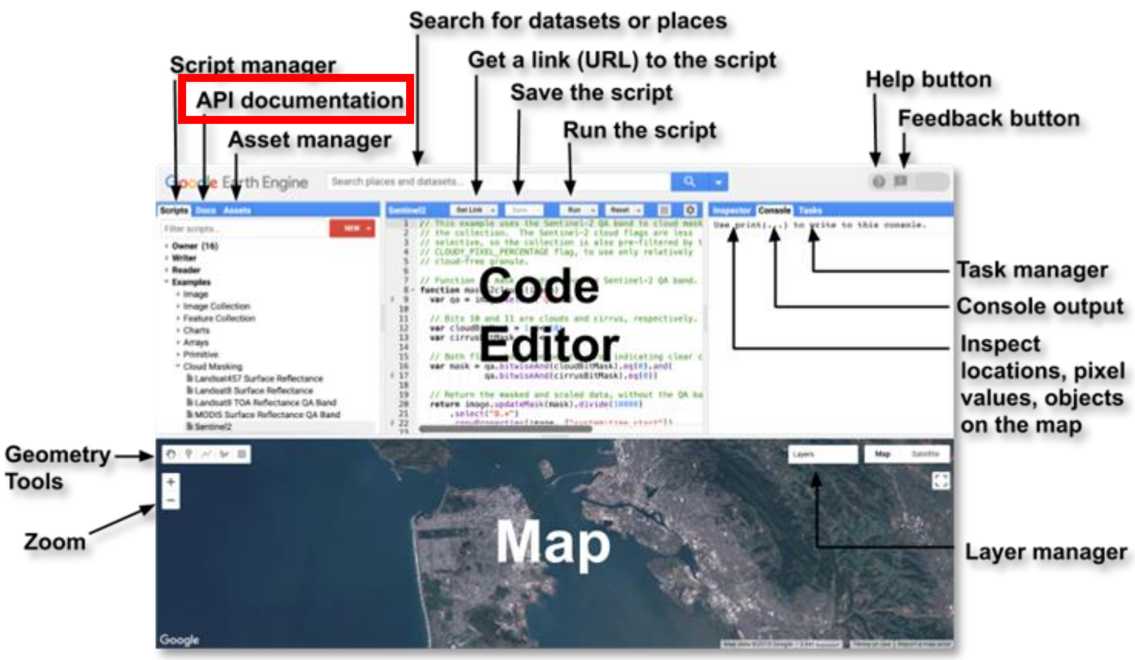
3. MODIS Collections

The Moderate Resolution Imaging Spectroradiometer (MODIS) is a satellite-based sensor used for earth and climate measurements. It is designed to provide measurements in large-scale global dynamics including changes in Earth's cloud cover, radiation budget and processes occurring in the oceans, on land, and in the lower atmosphere.

Get started using Earth Engine

The Google Earth Engine runs in a web-based programming environment, which means that it is not necessary to download any additional software to get programming. To access the so called 'Code Editor', simply browse for https://earthengine.google.com/ and on the upper right side, navigate through the tabs 'Get Started'. Then select an account that you would like to use and finally press the button 'Click here for the signup form'.

(Important note: To use the Google Earth Engine, you will need to register a Google-Account and verify it for the Google Earth Engine.)



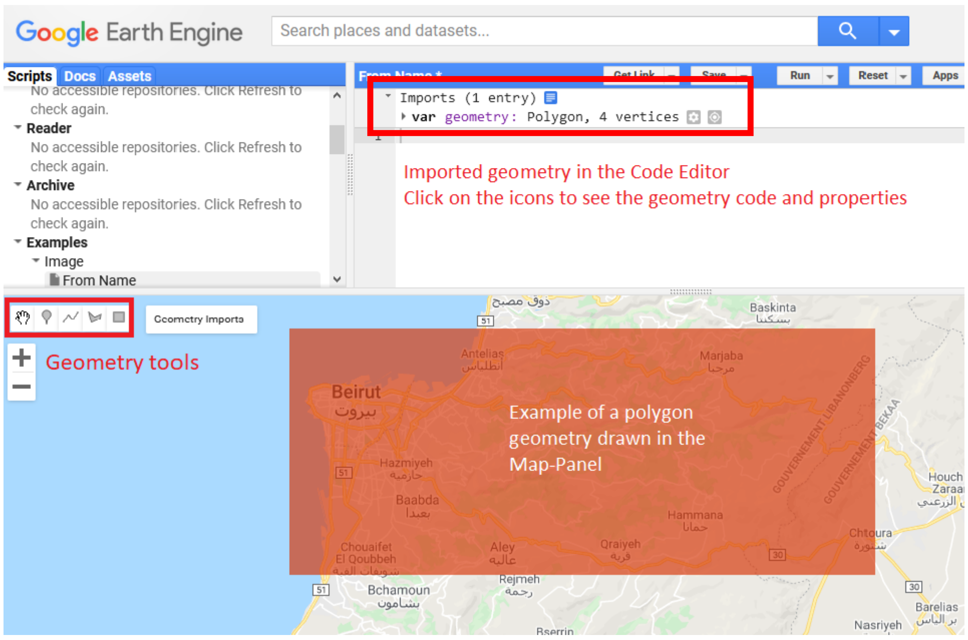
Data Structures

The Google Earth Engine works with two main data structures: Image and Feature data, which are basically raster and vector data types:

- Image data consists of a pixel raster, which contains a specific value for each pixel. Stacks of images, for example when working with time series, are handled by a so called ImageCollection.
- Feature data, on the other hand, consists of a boundary-defining geometry as well as a dictionary of properties, which corresponds to a classic attribute table. Analogous to ImageCollections, a collection of Features is handled by a so called FeatureCollection.

Creating geometries

The GEE allows to conveniently create geometries like points, lines, and polygons without having to write it out as code. Those so-called Features can be used for digitizing objects, creating extents for filtering, building up training and test data for classifications and many more applications. The drawn geometry will be directly imported into the Code Editor where you can check and edit its properties as well as have a look at its code.


Google Earth Engine
Search places and datasets...
Scripts
Docs
Assets
Reader
Archive
Examples
Image
From Name
Get Link
Save
Run
Reset
Apps
Geometry Imports
Geometry tools
Imported geometry in the Code Editor
Click on the icons to see the geometry code and properties
Example of a polygon geometry drawn in the Map-Panel
Beirut
Hazmiyeh
Baabda
Aley
Hammana
Chtoura
Bchamoun

```
Imports (1 entry)
  var geometry: Polygon, 4 vertices
```

Basic JavaScript in GEE

The Code Editor integrated into the GEE works with JavaScript, a common programming language. **Don't be scared though if you have never been in contact with programming before**, as it is really fun and rather easy to pick up for our geospatial applications. Just as with spoken languages, the basic patterns and principles of JavaScript will become familiar very soon just by working with it:

- **Spatially defining the research area**
  There are multiple approaches to define your Area of Interest in the GEE. The most basic and fastest method you can always fall back on is to simply draw a point or polygon shape with the geometries-tools and use its extents as a boundary filter. Another method is to use one of the GEE Data Catalogs numerous vector datasets and filter it for the geometry of your needs.

- **Filter ImageCollections**
  Filtering data allows us to only access the data that is of relevance for our research questions. Filters can easily be added to your desired image collection by adding .filter().

  .filterBounds() //filters for images that lie within your Area of Interest
  .filterDate() //filters for images within the selected range of dates (YYYY-MM-DD)

  ee.filter.eq('ImageProperty', 'Name') //filters 'equals' for Metadata
  ee.filter.lt('ImageProperty', 'Name') //filters 'less than' for Metadata
  ee.filter.gt('ImageProperty', 'Name') //filters 'greater than' for metadata

- **Clip**
  To remove the unnecessary parts of the ImageCollection, we can clip its extents to the extent of our defining geometry.
  Attention: .clip() only works on Images, not on ImageCollections! To generate an Image of your ImageCollection that does not modify your data, you can use .mosaic()

to create a single layerstack You are advised to clip your raster to your geometry, it is best advised to do so at the end of your analysis.

- **Reduce**
  Another way to easily transform your ImageCollections to a single image is to aggregate your data using so called Reducers. In the Docs tab, browse to ee.Reducer to find a comprehensive list of all Reducers available as well as a documentation on how to correctly implement them into your script. The functionality of the Reducers can range from simple statistics like min/max, mean, median or standard deviation to advanced aggregations of the input data like histograms, linear regressions or lists.

- **Assets: import & export**
  You may easily import your own image- or table data to use it in the Google Earth Engine. To do so, navigate to 'Assets' in the top left Panel and click on the button 'NEW'. Then, choose either 'Image Upload' if you want to upload a raster image or 'Table Upload' if you want to upload Shapefiles or CSV-tables. To import your asset into your script, you can either:
    - ✓ Access it the same way you access data from the GEE by adressing its root directory.
    - ✓ Click on the assets entry, then 'Import' in the popup-window.
    - ✓ Hover over the assets entry, then click the third icon on the right 'Import into script'.

  You may also export data you created to your Assets. To do so, you can simply use the Export.image or Export.table, depending on your type of data. Check the category Export in the Docs tab to find out more about its individual configuration parameters. Don't forget to refresh your Assets, as it may not be listed automatically!

<u>Example script for Landsat 8 data: Case Study - Surface Reflectance</u>

//Step 1: Access your boundary-defining geometry

//load the GAUL-Level2 layer to see administrative units.

```
var area = ee.FeatureCollection("FAO/GAUL/2015/level2")
        .filter(ee.Filter.eq('ADM0_NAME', 'China'))
//      .filter(ee.Filter.eq('ADM1_NAME', ''))
        .filter(ee.Filter.eq('ADM2_NAME', 'Guangzhou'))
Map.addLayer(area, {color: 'green', width: 1}, 'area');
```

//Step 2: Choose a Landsat 8 Catalog for your needs and filter it. In this example, we use Tier 1 Surface Reflectance data. Make sure to check the documentation to learn more about its bands and image properties!

```
var ls8 = ee.ImageCollection("LANDSAT/LC08/C02/T1_L2")
        .filterBounds(area)
        .filterDate('2015-01-01', '2021-01-01')
        .filter(ee.Filter.lt('CLOUD_COVER', 10));
print(ls8, 'Image Collection 2015-2021 <10% clouds');
```

//Step 3: Define the visualizing parameters.

//bands 2, 3 and 4 are to be visualized; minimum value is 5000 and maximum value is 12500 for all three bands

// Visualize the ImageCollection in the Map panel and assign a name to it; Hide the layer by default, as it takes long to render

// Center the map panel on the ImageCollection; zoom level is 7

```
var visParams = {
  bands: ['SR_B2', 'SR_B3', 'SR_B4'],
  min: [5000, 5000, 5000],
  max: [12500, 12500, 12500],
};
Map.addLayer(ls8, visParams ,'Landsat 8', false);
Map.centerObject(ls8,7);
```

//Step 4: Further process your ImageCollection

//Turn the ImageCollection to a single layerstack; this is essential for being able to clip it

//Clip the layerstack to the extent of the geometry

//Add the result to the map

```
var ls8_image = ls8.mosaic();
print(ls8_image, 'Layerstack Image 2015-2021, < 10% Cloud Cover');

var ls8_clip = ls8_image.clip(area);
print(ls8_clip, 'Layerstack Image 2015-2021 clipped, < 10% Cloud Cover');

Map.addLayer(ls8_clip, visParams ,'Landsat8 T2 2015-2021 clipped');
```

//Step 5: Use Reducers to have a look at image statistics via the print tab or the Inspector

//The result can be clipped, as it is an Image!

//the band names have changed, so we will have to define a new visParams

```
var ls8_mean = ls8.reduce(ee.Reducer.mean())
              .clip(area);
print(ls8_mean, '2015-2021 reduce mean');

var visParams_mean = {
  bands: ['SR_B2_mean', 'SR_B3_mean', 'SR_B4_mean'],
  min: [5000, 5000, 5000],
  max: [12500, 12500, 12500],
};
Map.addLayer(ls8_mean, visParams_mean, '2015-2021 reduce mean', false);
```

```
var ls8_minMax = ls8.reduce(ee.Reducer.minMax())
              .clip(area);
print(ls8_minMax, '2015-2021 reduce minMax');

var visParams_min = {
  bands: ['SR_B2_min', 'SR_B3_min', 'SR_B4_min'],
  min: [3000, 3000, 3000],
  max: [10000, 10000, 10000],
};
var visParams_max = {
  bands: ['SR_B2_max', 'SR_B3_max', 'SR_B4_max'],
  min: [7500, 7500, 7500],
  max: [17500, 17500, 17500],
};
Map.addLayer(ls8_minMax, visParams_min, '2015-2021 reduce min', false);
Map.addLayer(ls8_minMax, visParams_max, '2015-2021 reduce max', false);
```

//Step 6: Have a look at various charts for your data

//Add a time series chart of the mean surface reflectance values of bands 2, 3 and 4

```
var chart = ui.Chart.image
  .series({
    imageCollection: ls8.select('SR_B2','SR_B3','SR_B4'),
    region: area,
    reducer: ee.Reducer.mean(),
    scale: 200
     })
  .setOptions({
        title: 'Mean Surface Reflectance Value by Date',
        hAxis: {title: 'Date', titleTextStyle: {italic: false, bold: true}},
        vAxis: {title: 'Surface Reflectance',titleTextStyle: {italic: false, bold: true}},
  });

print(chart);
```

//Add a day-of-year-chart of the mean surface reflectance values of bands 2, 3 and 4

```
var doychart = ui.Chart.image
  .doySeries({
    imageCollection: ls8.select('SR_B2','SR_B3','SR_B4'),
    region: area,
    regionReducer: ee.Reducer.mean(),
    scale: 200,
    yearReducer: ee.Reducer.mean(),
    startDay: 1,
    endDay: 365
  })
  .setOptions({
          title: 'Mean Surface Reflectance Value by Date',
          hAxis: {title: 'Day of year', titleTextStyle: {italic: false, bold: true}},
          vAxis: {title: 'Surface Reflectance',titleTextStyle: {italic: false, bold: true}},
  });

print(doychart);
```

<End>