# INTRODUCTION TO GOOGLE EARTH ENGINE (GEE) FOR SATELLITE IMAGE TIME SERIES (SITS) ANALYSIS

Based on MSc research by Alejandro Fonseca [LINK](https://essay.utwente.nl/88728/1/fonsecagomez.pdf)

In the analysis of satellite image time series (SITS) uncovers long-term land surface dynamics (e.g., climate change, weather extremes, human-induced disturbances). Novel techniques have arisen to analyze SITS as image archives have become publicly available and new open-source cloud-computing tools enable us to retrieve meaningful information quickly from SITS. This offers an outstanding opportunity to explore the full potential of time series analysis for a wide range of applications, including ecosystem resilience, dynamic land cover and use mapping, space-time anomaly detection, phenological agricultural metrics extraction, and climate change modeling.

## Learning outcome of the practical

|  |
| --- |
| The main objective of this practical is to introduce you to GEE for processing Landsat time series data. During the practical, you will:   * Acquire and make Landsat time series data analysis-ready in GEE   GEE is the most widely used cloud computing environment for geospatial data processing and analysis. Landsat offers the longest publicly available record of the Earth’s surface from space. |

## Overview topics

The case study should take you approximately 6 hours.

### Before the practical

1. Create a GEE account
2. Explanation of the interface
3. Common data types in JavaScript in GEE
4. Writing a Function
5. Saving your outputs

### Main elements of the practical

1. Image visualization
2. Filtering image collections
3. Creating mosaics and composites from image collections and subsets using Clip
4. Cloud Masking
5. Calculating Spectral Indices
6. Plot band histograms and spectral profiles
7. Short assignment

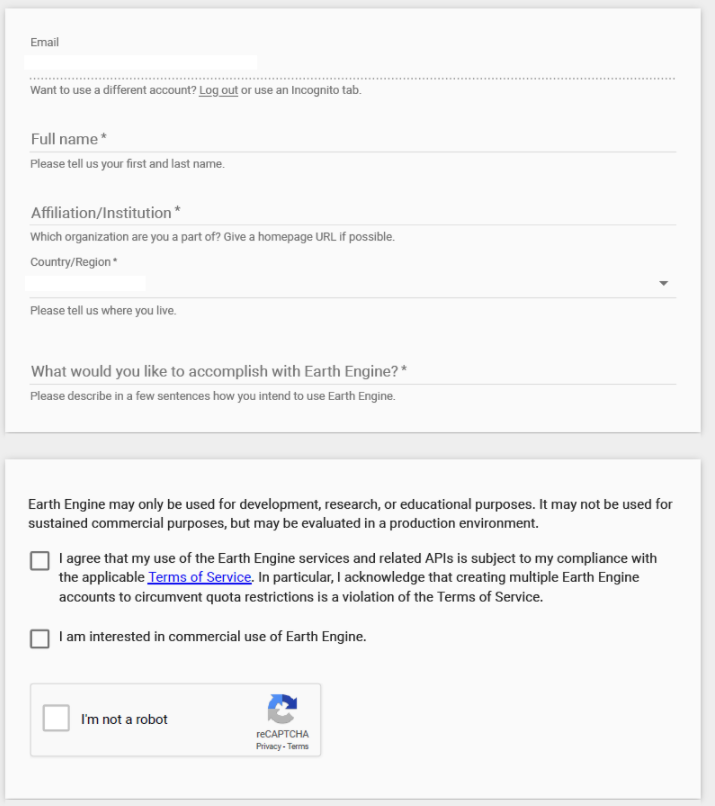
## Before the practical

### Create a GEE account.

You can skip this section if:

* You already have a GEE account
* You have some background in coding

Go to [https://signup.earthengine.google.com/](https://signup.earthengine.google.com/#!/) and sign-up for your GEE. You will need this account to manipulate scripts, docs and assets. You can use your existing Gmail account to sign-up. It usually takes 1-2 days for approval. Hence do this step as soon as possible. To create an account, you must fill out the online form. They will request your email address, name, institution and location, and a brief explanation of how the platform will be used. You will receive authorization of your GEE account via email.

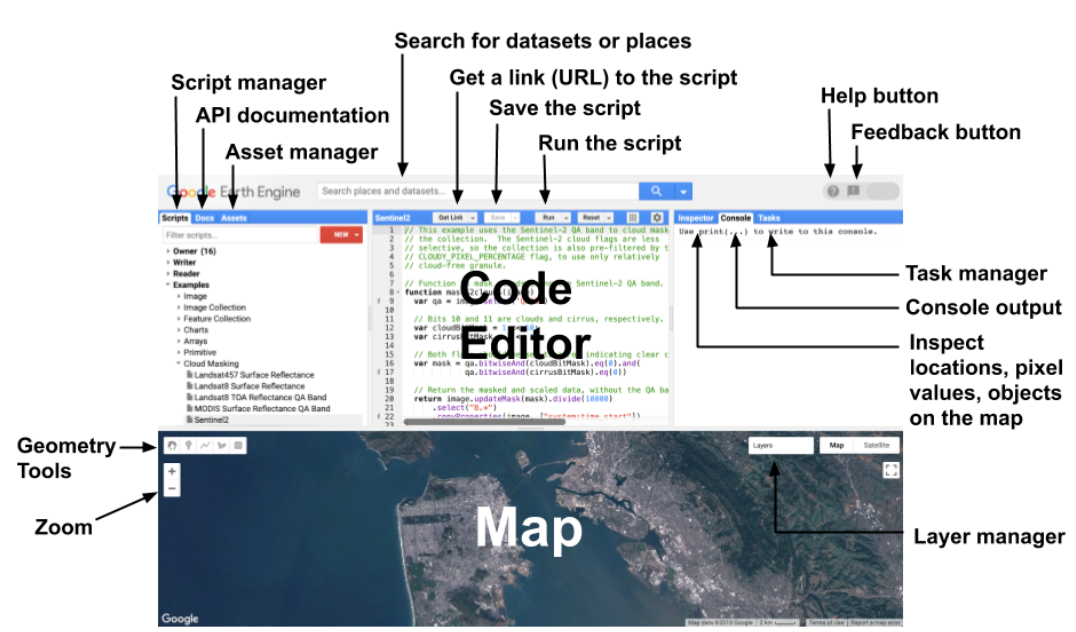


Here there are some tips for creating your account:

* Use Google Chrome browser
* Ensure you are logged into only a Google account when signing in to GEE. This is the account that will be associated with GEE.
* If you use your @student.utwent.nl, the process takes a couple of hours. However, after completing your MSc, you cannot get access to that account and therefore can no longer access GEE.

### Interface Explanation

Once your account is fully activated, you can access your GEE account through this [link](https://code.earthengine.google.com/3bf818e88952e4f84c4fac90c3ea3672). The following guide describes the main elements of the JavaScript GEE interface. Please use the guide to explore the interface yourself. Visit <https://developers.google.com/earth-engine/cloud> for further details.



GEE interface environment (<https://developers.google.com/earth-engine/guides/playground?hl=en>)

* The code editor: It is an interactive environment for developing applications on GEE. There are buttons to save the current script, execute it, and restart the map. The *Get Link* button creates a unique link for the script in the address bar.
* The map contains the layers executed in the script.
* The search bar is at the top and allows you to search for addresses and data sets.
* The Script’ Catalog is in the upper left. It contains documentation and data added by the user,
* Inspector allows you to explore data on the map
* The console shows the results printed in the script (images, collections, values, tables, figures).
* Task Manager allows long-running managing processes (data ingestion processes, export of layers and images, among others).

### A brief and short introduction to JavaScript in GEE

This script at the following link introduces the basic JavaScript syntax: <https://developers.google.com/earth-engine/tutorials/tutorial_js_01>. The *Code Editor* is an Integrated Development Environment for the Earth Engine JavaScript API. It offers an easy way to type, debug, run and manage code in the editor dialog.

* Open the following script [LINK](https://code.earthengine.google.com/6f5985bdacbee00bcaaf92497ec8fd04) and click *Run* to execute it and see the output in the MAP window. This is a simple example of the power of GEE. It shows how easily you can access information with just a few lines of code. Please click on the *Inspector* tab and click anywhere on the map pane for information about the image. Select different locations. In this example you do not need to understand the entire process. It is something you will learn through this practical.

Tip: You can use the keyboard shortcut *CTRL Enter* to run the code in the Code Editor instead of the **RUN** button (Yes, like in R 😊)

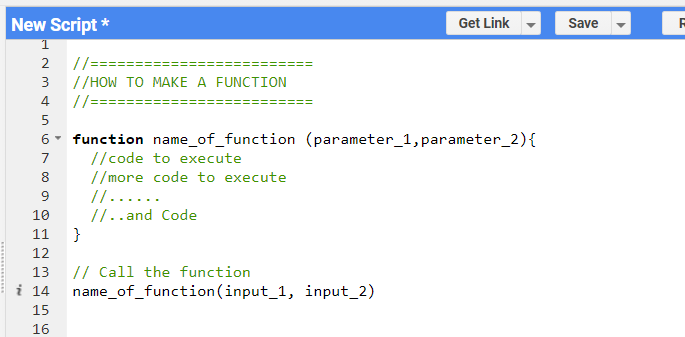
### Common data types on JavaScript in GEE

Data types include strings, numbers, arrays, lists, dictionaries, functions. Please click the following [LINK](https://code.earthengine.google.com/adb5977b03554b6ac7bb69367f9ad80d) and view your code editor for further details. Symbol (//) is used for comments between executable scripts.

### Writing a Function in GEE

Functions are handy when you have a series of statements that you would like to repeat many times in the code. For example, perhaps you want to calculate the pixelwise sum of your image data. We can create a function to call sum each time we want to execute it in our code, rather than re-write the code each time.

The skeleton of a function is below:



* Copy and paste the following example into your code editor. Execute it. What does the code do? Change the *in\_values*, number of *in\_values*, and operator (e.g., *Subtract*).

Functions are particularly useful for calculating vegetation indices, such as the normalized difference vegetation index (NDVI): (NIR – red) / (NIR + red).

* Try to change the function, so that it calculates NDVI. Supply the input\_values NIR = 0.7 and red = 0.1. Did the function return an NDVI of 0.75?

//========

//EXAMPLE

//========

function calculate\_Sum (in\_value1, in\_value2) {

var Sum = ee.Number(in\_value1).add(ee.Number(in\_value2)); // code that calculates the sum

print(Sum);

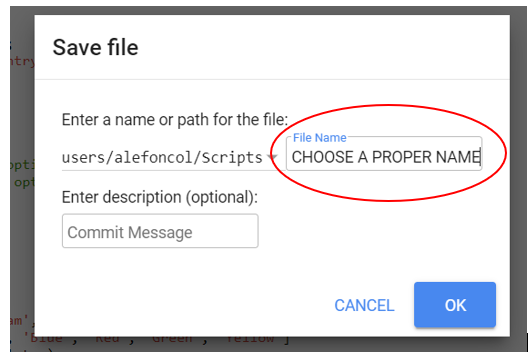
}

// call the function

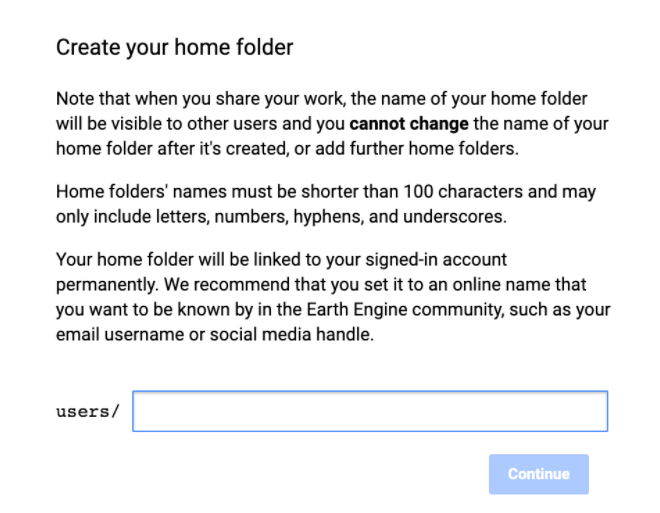
calculate\_Sum(75, 82);

### Saving your Work

While you’re working on your scripts, you may want to save a copy for yourself. If you try to click the *Save* button, you will get the following message:



If this is the first time you are saving a script in GEE, you will be prompted to choose the name of your *home folder*. Choose the name carefully, as it cannot be changed once created.



### Main elements of the practical

### Image visualization

Viewing satellite images or other geoinformation data layers is one of the basic functions of a geospatial analysis tool. With a small script it is possible to display a layer in the lower panel of GEE.

The steps to display an image are as follows:

* Identify the layer or image that you want to display. This can be done using the search bar at the top of the panel.
* Define a variable to load the layer.
* Select one or more bands of the layer or image.
* Define the display parameters (maximum and minimum values, color palette, etc.).
* Add the layer to the map (variable, display parameters and name).
* Please click the following [LINK](https://code.earthengine.google.com/76eb5b2254977cd7e99cfda9768ff8e0) to see an example of a layer loaded with a defining a variable. Click on the *Layers* tab in the map pane. You can use it to turn the layer on and off, as well as change the transparency with the slider bar. This tab also indicates processing status. WorldClim is a gridded climatology of the globe. It consists of nineteen bioclimatic variables, which are used for species distribution modeling among other applications (<https://www.worldclim.org/data/bioclim.html>).

At this moment, you might wonder how you can get the information about the datasets and bands and how to analyze images over a region of interest.

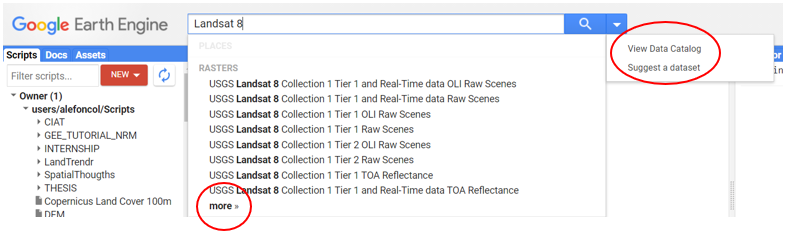
Most datasets in Earth Engine come as an *ImageCollection.* An I*mageCollection* is a dataset consisting of images taken at different times and locations - usually from the same satellite or data provider. You can load a collection by searching the  [Earth Engine Data Catalog](https://developers.google.com/earth-engine/datasets) for the *ImageCollection ID*. Go to the *View all Dataset* tab or use the search pane to look for the *Sentinel-2 Level 1C* dataset. You will find its id from the catalog (COPERNICUS/S2\_SR). This is the id you can assign to a variable to load the image collection. Visit the  [Sentinel-2, Level 1C page](https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2) and see *Explore in the Earth Engine* section to find the code snippet to load and visualize the collection. This snippet is a great starting point for your work with the dataset.

* Click the **Copy Code Sample** button highlighted in red. Paste the code into the code editor. Click *Run*, and you will see the image tiles load in the map.

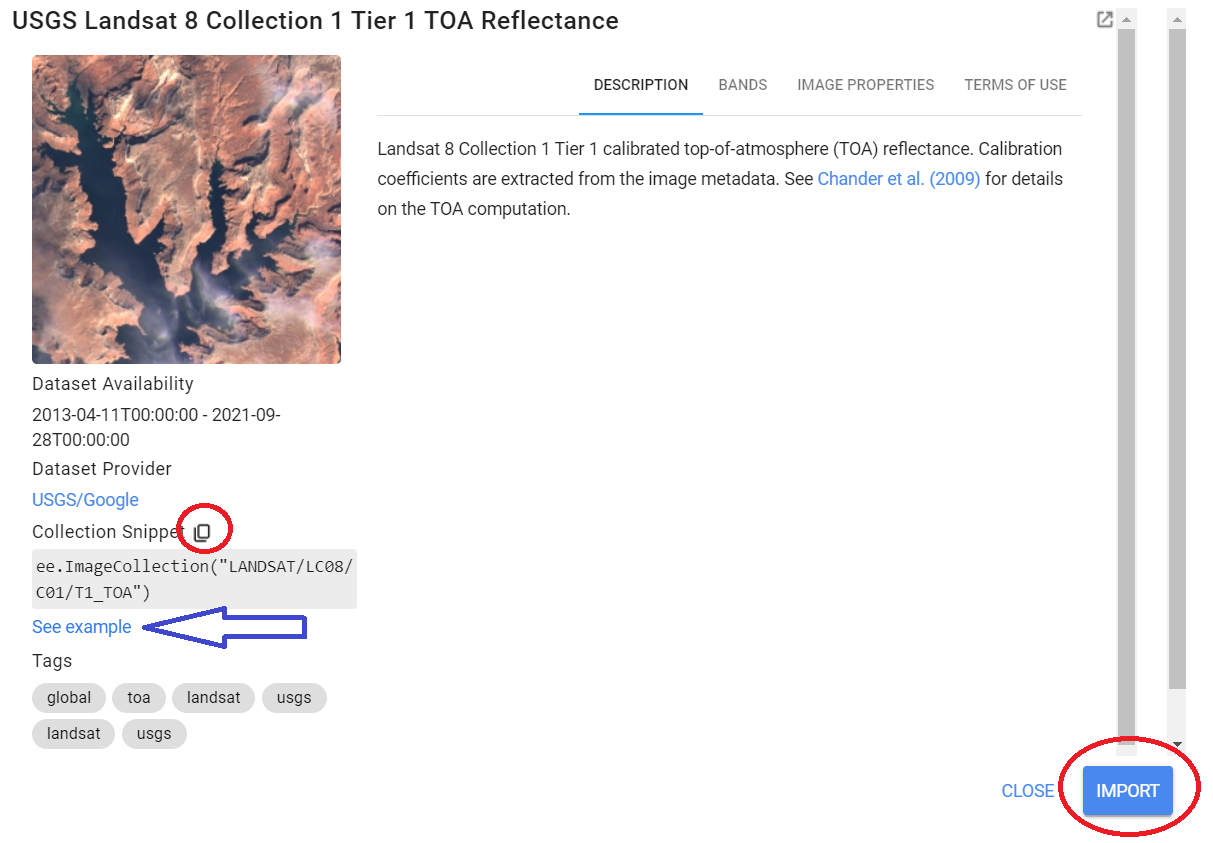


Alternatively, you can use the search bar at the top of the interface and search for a dataset.

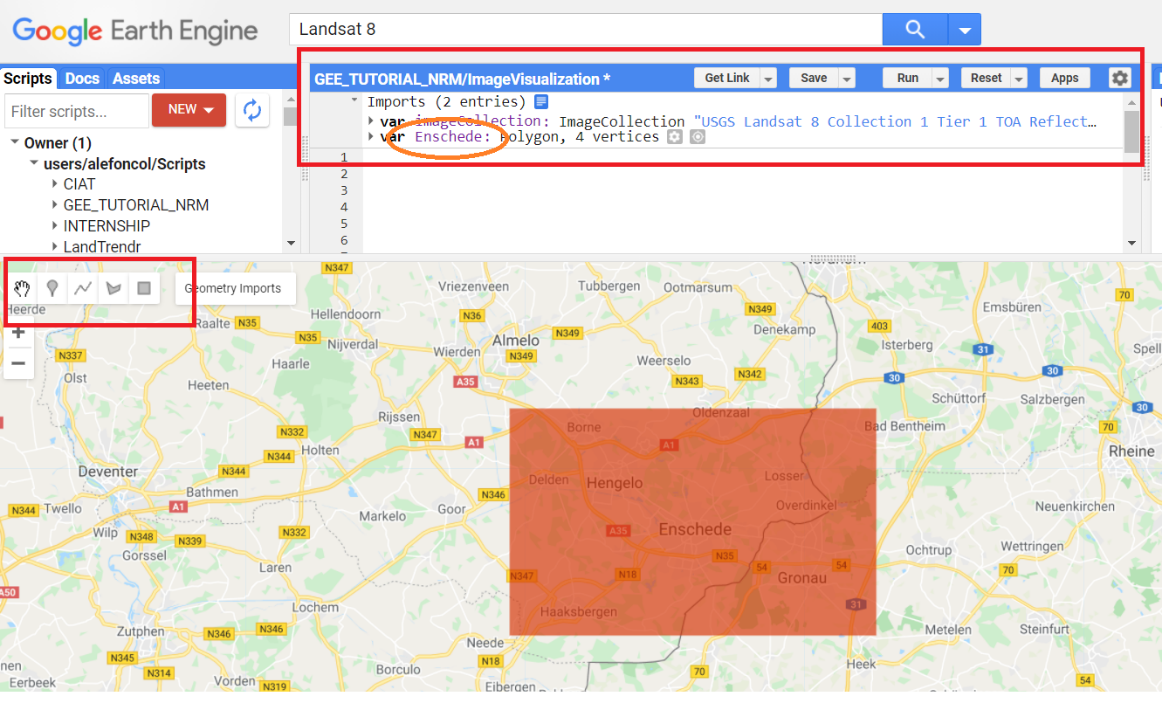
* Search for “Landsat 8”. A series of results under *RASTERS* is displayed. Scroll down and select “Landsat 8 Collection 1 Tier 1 TOA Reflectance.” This is the radiometrically corrected (but not atmospherically corrected) Landsat-8 OLI product.



The following pop-up should appear. Go through the tabs at the upper righthand side for information on the bands, image properties and other useful information. At this step, you have two options to import the imagery. You can click on the *IMPORT* button or click on *Collection Snippet square.* The *See example* link takes you to example code to manipulate the image collection.



* Using the option import, you will see that the data set is directly imported as a variable in the *Imports space* with the name of *imageCollection*. Now use the geometry tool to draw a polygon over Enschede to constrain the visualization to a specific area of interest. Be aware that in the *Imports space,* GEE also stores the geometry features. Please change the name of the variable to Enschede.



Let us try to add an image for Enschede.

* Please copy and paste the following code. You can see how to perform a true color (RGB) visualization selecting Landsat bands 4, 3, 2, with some visualization parameters, map center (defined latitude/longitude), and polygon from the *Map Space.*

var year\_2020 = imageCollection.filterDate('2020-05-01' , '2020-05-30');

var RGB = year\_2020.select(['B4', 'B3', 'B2']);

var RGBVis = {

min: 0.0,

max: 0.4,

};

Map.setCenter(6.893929 , 52.22083, 12 ); // Coordinates map center and image zoom

Map.addLayer(RGB, RGBVis, 'True Color (Enschede)');

You can find the result in this [LINK](https://code.earthengine.google.com/4d42663b9e9e9cbe2fc6a333a0691609).

* + Try other dates and other visualization parameters. You can also try other band combinations: Color Infrared (5, 4, 3), Short-Wave Infrared (7, 6 4), and Agriculture (6, 5, 2).
  + Find the Sentinel-2 Level 2A product using the search bar. This is the Sentinel-2 surface reflectance (atmospherically corrected) product. Click on the example tab and run the script. Change the map center coordinates to your home town. You can also change the collection dates and the cloud thresholding filter. Finally, explore different band combinations. A description of Sentinel-2 bands can be found here: <https://en.wikipedia.org/wiki/Sentinel-2>.

### Filtering image collections

You now know how to access images from different sensors and center the map in an area of interest. Let us try to use more parameters to subset the Imagecollection considering that image collections contain all imagery ever collected by the sensor, which is generally not practical for analysis. GEE uses **filters** to select the appropriate images. There are many types of filters. Look at the *ee.Filter* module (image below) to see all available filters. Select a filter and then run the filter() function with the filter parameters.

* Please click in this [LINK](https://code.earthengine.google.com/ed634832c69f12bca4b5c888c8e9d4aa) or copy/paste in a new editor window the following snippet:

// Full ImageCollection (Bulk)

var s2 = ee.ImageCollection("COPERNICUS/S2");

Map.centerObject(Enschede, 12) // Enschede Coordinates: 6.894690692093399,52.22102131859881

//Filter by Medata (Cloud Percentage)

var filtered = s2.filter(ee.Filter.lt('CLOUDY\_PIXEL\_PERCENTAGE', 30))

// Filter by date

var filtered = s2.filter(ee.Filter.date('2020-01-01', '2020-30-31'))

// Filter by location

var filtered = s2.filter(ee.Filter.bounds(Enschede))

// we can apply filters together by using " . " after every sentence.

// and filter the image in one row.

var filtered = s2.filter(ee.Filter.lt('CLOUDY\_PIXEL\_PERCENTAGE', 5))

.filter(ee.Filter.date('2019-01-01', '2020-12-30'))

.filter(ee.Filter.bounds(Enschede))

print('filtered size : ' , filtered.size())

var rgbVis = { min: 0.0, max: 2000, bands: ['B4', 'B3', 'B2'], };

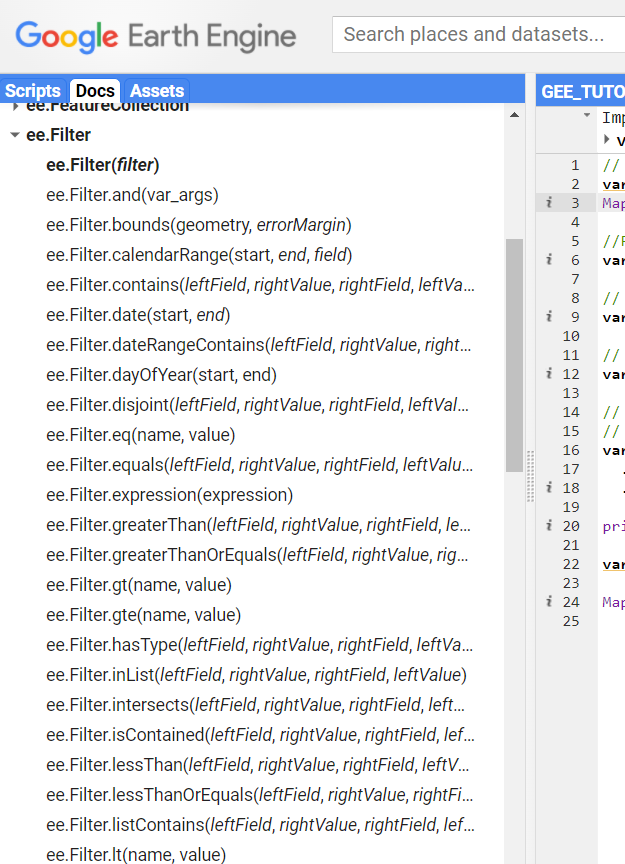
Map.addLayer(filtered, rgbVis, 'Filtered')

We will now elaborate on the three main types of filtering techniques

* **Filter by metadata**: You can apply a filter on the image metadata using filters such as *ee.Filter.eq()*, *ee.Filter.lt()*, etc. You can filter by PATH/ROW values, Orbit number, Cloud cover etc.
* **Filter by date**: You can select images in a particular date range using filters such as *ee.Filter.date()*.
* **Filter by location**: You can select the subset of images with a bounding box, location or geometry using the *ee.Filter.bounds()*. You can also use the drawing tools to draw a geometry for filtering.

After applying the filters, you can use the **size ()** function to check how many images match the filters.

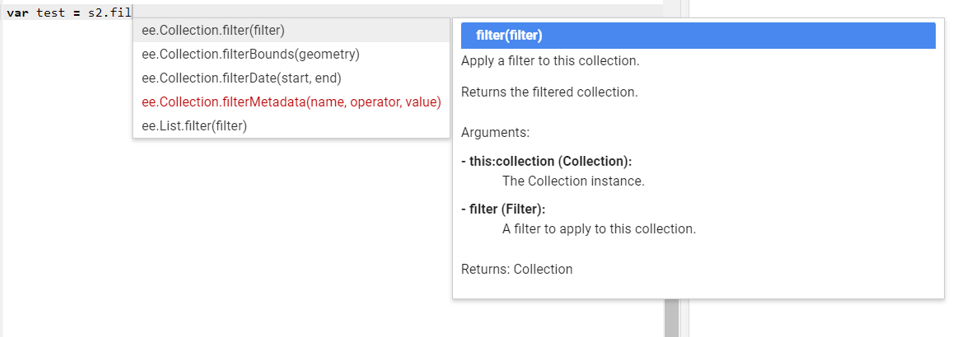
Add and explore more imagecollection filters (*ee.filter*) with the s2 ImageCollection previously added. Also, explore other filters with other sensors. For more filter options, see the chart below.



Tip: Use one of the useful shortcuts in GEE for completing structured code sentences (correct coding structure using Java Script). The JavaScript syntax might not be intuitive when you start to work with the GEE. Therefore, you can use the key CTRL + SPACE BAR to complete sentences. Let’s see an example:

In the current code editor (filtering Image Collection) write:

**var test = s2.fil** and then press: Ctrl + space bar.

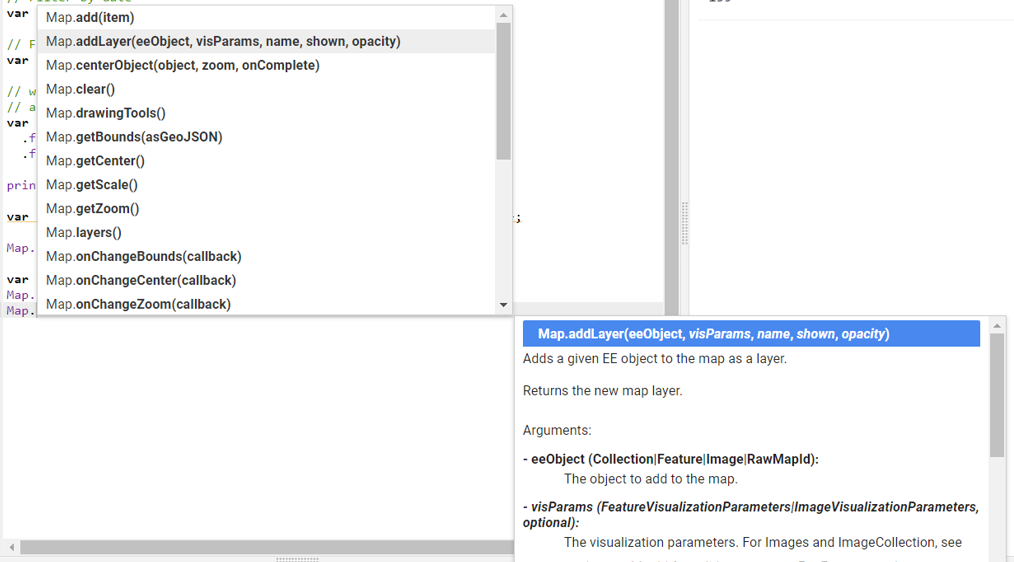


Scroll down (with your mouse) and choose one of the options using the arrows of your keyboard and the TAB key. You will see that the sentence has been partially complete. If you press Ctrl + space bar once more you can see the arguments required.



Tip (1): Try with *filterDate* and then press Ctrl + bar space to get a better of understanding of the functionality.

Tip (2): Try writing in the code editor: **Map**, press Ctrl + bar space, select *Map.addLayer* and press Ctrl + bar space again (check line 24 of the current code editor).



Try to replicate the same process yourself, in order to understand how to filter image collections. It is important you understand the process and its structure. Later, we are going to apply other functions such as cloud mask, image compositing or reductions, and harmonizing multiple Landsat missions to create an extended time series.

### Creating Mosaics & Composites from Image Collections

Joining several contiguous images to create an image that covers a larger area is known as a mosaic. So, displaying the collection in GEE implicitly creates a mosaic with the last pixels on top in the overlapping area. The Earth Engine *Mosaic* feature is used to create mosaics of multiple images in a collection. To improve the map's appearance, we can cut the mosaic created using a predetermined geometry (it can be a shapefile or a polygon) and the *clip* function.

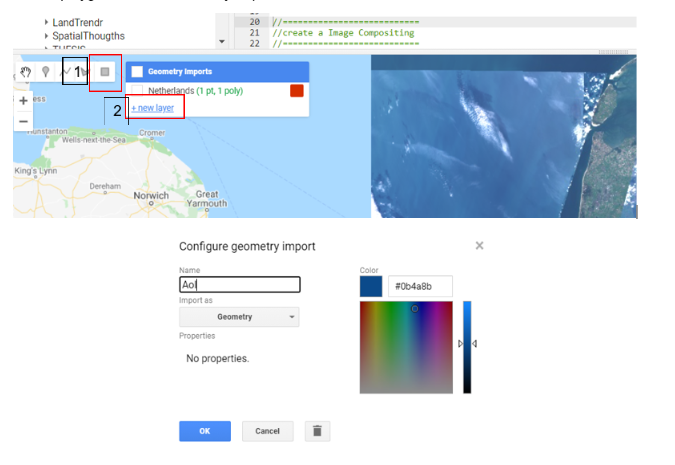
Image compositing on the other hand is the process of joining spatially overlapping images into a single image with an aggregation function (median, mean, average, latest, newest, greenest, medoid). This is performed on a pixel basis for images stacked in time.

Moreover, we can create a composite image by applying selection criteria. Here we are going to use the *median ()* function to create a composite where each pixel value is the median of all corresponding pixels in the stack. There are other functions such as medoid, average, first, last, greenest, and so on. More details on these functions can be in Azzari & Lobell, (2017):

First, we are going to create the mosaic over the whole Twente region. Afterwards, we will make an image composite with *median ()*. Finally, we will clip the image using the FAO administrative boundaries.

Please open the following [LINK](https://code.earthengine.google.com/f92fa4cb28ae02a08ec2e550b7126bb8?noload=true), which already has an image collection filtered for an area of the Netherlands.

* Zoom in and out with the + and – tabs to see the way the image quality changes.
* Create an area of interest polygon using the geometry tools and name it as you prefer (e.g., AOI). You will need to add a new layer to do this. You can also toggle off The Netherlands polygon.





* Let’s create the mosaic: Add the following code to the link above and run it. Be sure to change the polygon name from AOI. Toggle off the filtered collection to properly see your mosaic.

var mosaic = filtered.mosaic().clip(AoI)  
Map.addLayer(mosaic, rgbVis, 'Mosaic');

* Create an image composite for the mosaic and add the images to the map by adding the following code to the link above.

var medianComposite = filtered.median(). clip (AoI);   
var maxComposite = filtered.max();

var minComposite = filtered.min();

var meanComposite = filtered.mean();

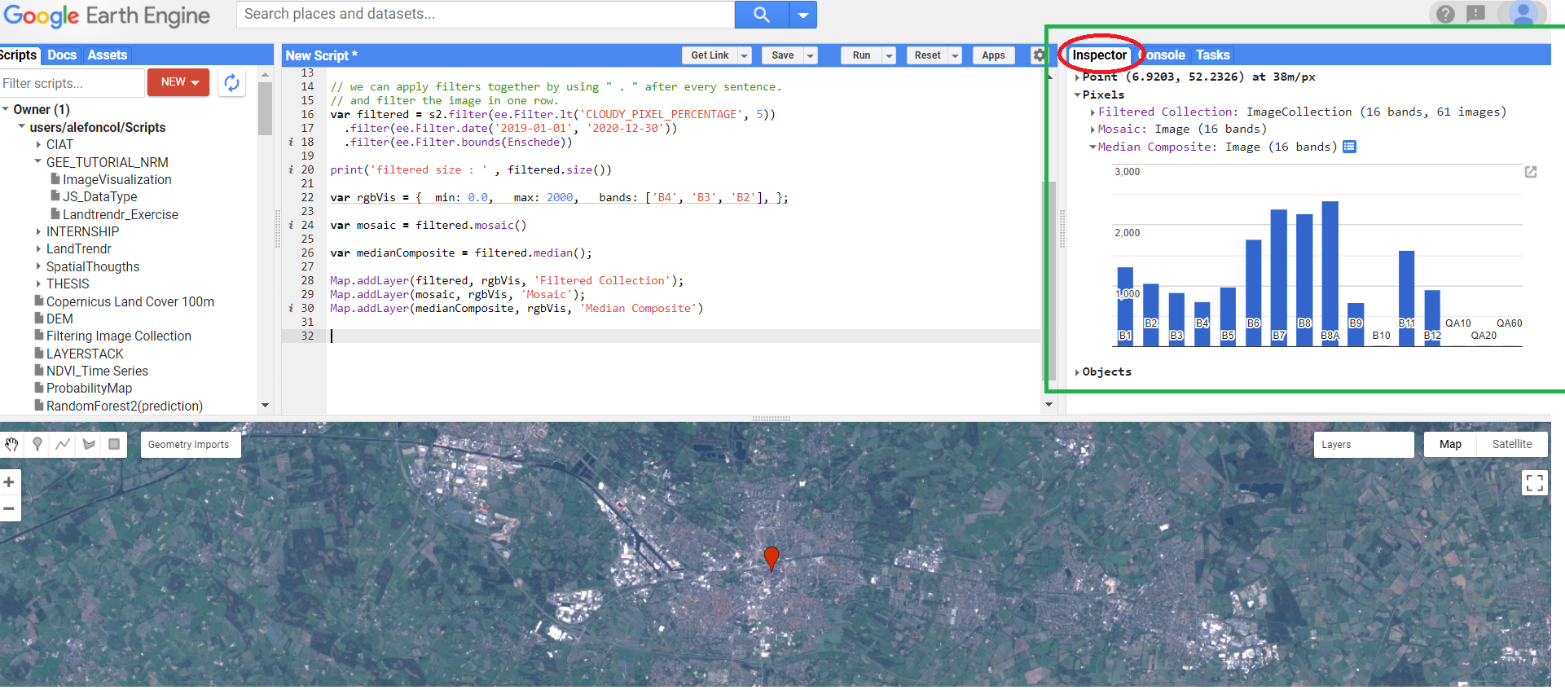
Map.addLayer(medianComposite, rgbVis, 'medianComposite');

Map.addLayer(maxComposite, rgbVis, 'maxComposite');

Map.addLayer(meanComposite, rgbVis, 'meanComposite');

Map.addLayer(minComposite, rgbVis, 'minComposite');

Use the *Inspector*tabon the righthand side of the screen and click over the area that you want to retrieve information from. Use inspector to interpret the different composites with the histograms that appear. Do this for different land cover types.

****

* At the end of this step, you must have the following result [LINK](https://code.earthengine.google.com/1b73106337f60ffd34d4f96118eca22a?noload=true). Please compare the properties of the different composites using the visualization properties and inspector tab

Tip: You can use the following function to create a circular clip.

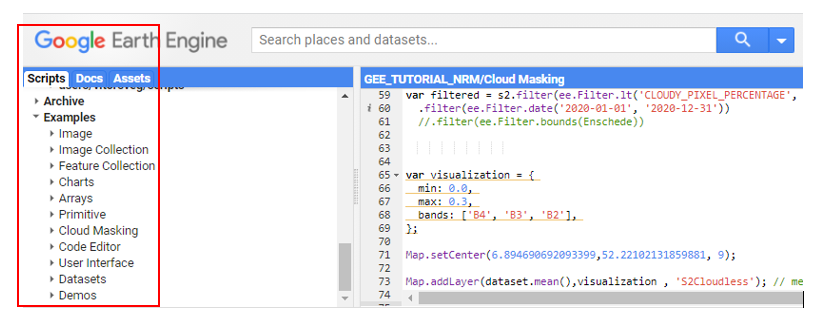
// Create a circle by drawing a 20000 meter buffer around a point.  
var roi = ee.Geometry.Point([ x , y ]).buffer(20000); //x,y are coordinates

* Create a median composite for another year and load into the map on your own. Try with your hometown and another sensor like Landsat.

### Cloud masking

At this point, it's worth stepping back and considering what's been done to make the median composite. Earth Engine has loaded the entire Sentinel 2 collection for all of The Netherlands and calculated the median on a pixel-by-pixel basis. That is a lot of data! Of course, you could compute medians by first filtering the collection to reduce the data load as you have already done. In either case, it would be a huge undertaking to download all the imagery to make the composite if you did not have cloud computing. With GEE, you get a result in seconds! Masking pixels in an image makes those pixels transparent and excludes them from analysis. ***Each pixel in each band of an image has a mask.*** Those with a mask value of 0 or below will be transparent. Those with a mask of any value above 0 will be displayed. The mask of an image is set using a call like *image1.mask(image2)*. This call takes the values of *image2* and makes them the mask of *image1*. Any pixels in *image2* that have the value 0 will be made transparent in image1.

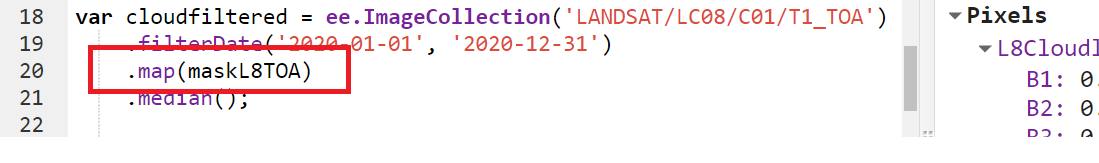
Let us illustrate the concept by applying a cloud mask. The first thing we will do is to identify the pixels affected by clouds. Sentinel and Landsat have quality assessment (QA) bands that enable you to identify certain characteristics of the pixels: *Landsat 8 BQA* and *Sentinel QA60*. You can call different code editor functions to identify undesirable characteristics. The GEE contains pre-defined functions for masking clouds for popular datasets. Visit Scripts Tab 🡪 Examples 🡪 Cloud Masking for further details.



Please click on the following script tailor-made for you [LIN K](https://code.earthengine.google.com/653d12e5fd386757c7088b040eaf3a6a) where you can better grasp *CloudMasking* functions and previous concepts such as *Image Compositing* in Earth Engine. Some important points:

* The cloud masking function depends on the type of data you want to use, e.g., Top of Atmosphere (TOA) or Surface Reflectance. The difference between these data types is explained here: <https://www.l3harrisgeospatial.com/Learn/Blogs/Blog-Details/ArtMID/10198/ArticleID/16278/Digital-Number-Radiance-and-Reflectance>).
* In the example provided, we use a function for the Landsat 8 TOA collection and another for Sentinel 2 surface reflectance. Please check documentation before applying these functions beforehand.
* The example shows two image compositing methods (mean, median). In reality, you can apply any compositing method that you consider the most appropriate for your analysis.
* The example shows functions as applied over an Image collection and a single image.
* What is the purpose of .map(maskclouds)?

**RTA:** The Earth Engine structure does handle loops well- they are generally bulky and inefficient. With the map () operation you specify a function that can be independently applied to each element of the image collection without loops. You can use the operation for any kind of function. In other words is the way to do “loop of GEE”.



* Apply another cloud mask for Landsat 8 surface reflectance or other product of your choosing. Get familiar with how to retrieve data from the documentation provided and assess the cloud mask in your area of interest.

### Calculating vegetation indices

At this stage it is important to recall some concepts for a better understanding of vegetation indices:

Digital Number (DN) is the generic term used for pixel values. These values correlate with the brightness energy that is observed and measured at the sensor. Although the DNs are related to the surface reflectance values, they are not the same. Depending on the purpose of the study, comparing DNs acquired by a satellite, without correcting for atmospheric effects and other sources of radiometric errors, can lead to problems - even if the Landsat DNs have been calibrated and processed to some degree.

Radiance is directly measured by remote sensing instruments. Radiance includes the radiation reflected from the surface, in addition to the radiation that comes in from neighboring pixels, and the radiation reflected from clouds. Radiance is dependent on not only the illumination (both its intensity and direction), but the orientation and position of the target. In other words, the path of light through the atmosphere can change as the light travels down to the earth through the atmosphere, suffering wavelength-dependent scattering and absorption. The light then diffusively reflects off of the Earth's surface, travelling back up through the Earth's atmosphere, enduring further scattering effects as it does so. Some of these factors are accounted for in the TOA data. The Bottom of Atmosphere (BOA) or surface reflectance is satellite derived Top of Atmosphere (TOA) reflectance corrected for the scattering and absorption effects of atmospheric gases and aerosols. It is the most widely used product for time series analysis (Chen et al., 2018)

Spectral indices are central to many applications of Earth observation. In such cases, you will need to compute a pixel-wise ratio of 2 or more spectral bands in Earth Engine. A large (but not exhaustive) list of indices can be found at <https://www.l3harrisgeospatial.com/docs/vegetationindices.html>. The most common format is the normalized difference between two bands. This format is widely used because one spectral channel in the combination normalizes according to different soil backgrounds, atmospheric constituents and topography. Earth Engine provides a helper function *normalizedDifference ()*to calculate these indices. The most widely used is the Normalized Difference Vegetation Index (NDVI). Alternatively, you can use the *expression ()* function calculate spectral indices.

* In the following script, you can see four different methods for calculating spectral indices in Earth Engine: [LINK](https://code.earthengine.google.com/39c332159a4c4256640b303a9e374100?noload=true). Compare and contrast the different methods.

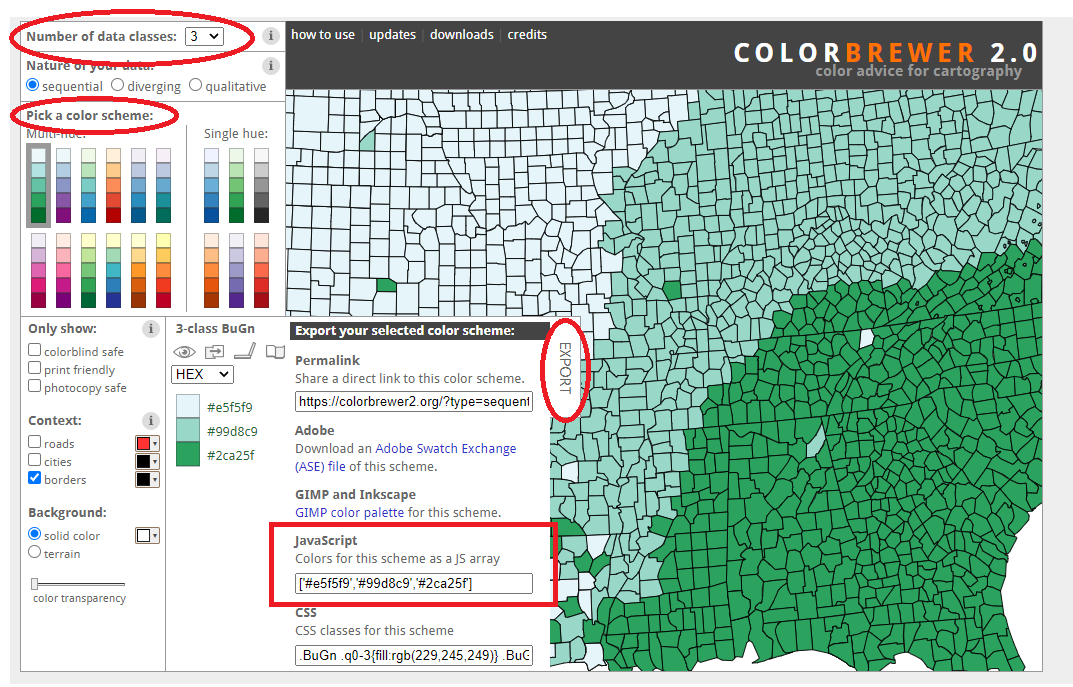
In this script you also see vector data. This is derived from the FAO Global Administrative Unit Layers 2015 (<https://developers.google.com/earth-engine/datasets/catalog/FAO_GAUL_2015_level2>).

You can also change the color palette of your choosing. In this case, we are given options from color brewer 2.0 (<https://colorbrewer2.org/>). Color brewer provides a number of publication-quality palettes to visualize either categorical or continuous data. An image collection of Landsat 8 surface reflectance is used for this purpose. Please see the general description of sensor specifications to better understand the bands.

(<https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C02_T1_L2?hl=en#bands>)

To personalize your color palette in GEE using the color brewer 2.0 follow these steps:

* Link to the website
* Define the number of data classes
* Pick a suitable color scheme
* Click on the “EXPORT” tab
* Copy the colors as the JavaScript array and past it when you define a palette in visualization options.



* Generate three spectral indices from the link provided and apply over the "region" collection. Explore different color palettes to visualize your data.

### Time series and temporal profiles

This exercise will introduce you to making a 2-D time-series figure from spectral information in an area of interest.

For this exercise, it is important for you to understand two concepts:

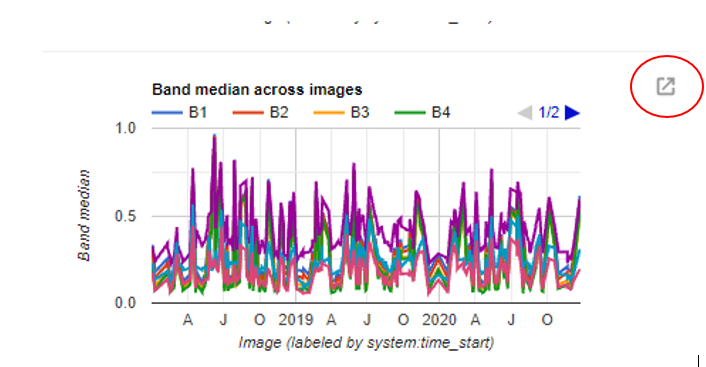
**Reducer**: Its operation allows you to compute statistics on a large number of inputs. In GEE, you need to run reduction operations when creating composites, calculating statistics, doing regression analysis, etc. GEE includes a large number of built-in reducer functions such as *ee.Reducer.sum()*, *ee.Reducer.histogram()*, and *ee.Reducer.linearFit()*. They can perform a variety of statistical operations on input data without loops. You can run reducers using the *reduce ()* function as well. GEE supports running reducers on all data structures that can contain multiple values, such as Images (reducers run on different bands), ImageCollections, FeatureCollections, Lists, Dictionaries, etc. A result is always a number. For more information, please see: (<https://developers.google.com/earth-engine/guides/reducers_intro?hl=en> and <https://developers.google.com/earth-engine/guides/reducers_reduce_region>.

**Greenest** **Pixel**: The greenest pixel is the pixel with the highest NDVI value over a given compositing period This type of compositing is useful to identify forest disturbances and monitor vegetation health. The greenest pixel on an annual basis reduces seasonality, radiometric inconsistencies, and other sources of variation. More information on the greenest pixel can be found here:

<https://developers.google.com/earth-engine/tutorials/tutorial_api_06?hl=en#make-a-greenest-pixel-composite>

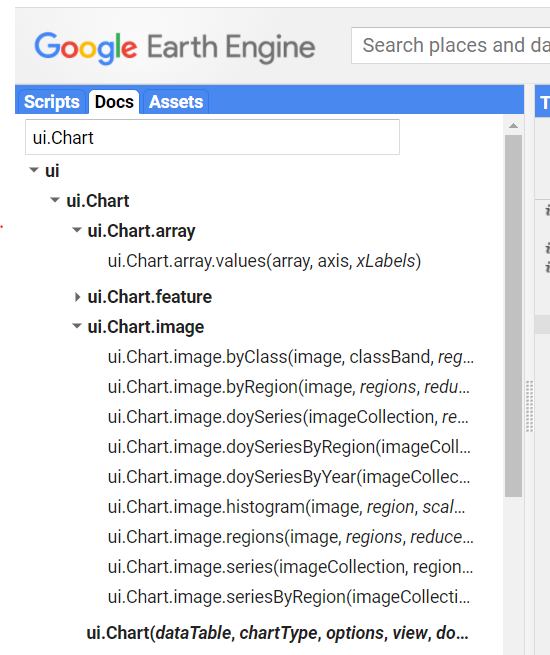
* Please click [here](https://code.earthengine.google.com/f26c5c35d63a9b2ed27bb480e96857c4) to open the code with the time series

Feel free to change the Area of Interest (AOI = red square) and check the temporal spectral profile for the NDVI and Landsat bands. You can expand the chart by clicking on the pop-out icon in the upper right corner of the chart. You can also hover over points in the chart to retrieve the “x” and “y” values.



GEE has a lot of functionality with plotting time series. You can find more information in the “Docs” tab at the up-right side of the window by typing *UI.Chart.Image* (see image below).

In the following example you will plot a “smooth” time series based on the the MODIS Terra 16-day vegetation indices at 250m spatial resolution using different charting options.



* Time Series by Region: Comparing two regions (AOI) through time
* Time Series by Year: Comparing different spectral profiles
* Time series Image Collection: Using a reducer value to display temporal changes
* Please click here [LINK](https://code.earthengine.google.com/20c8601b4dc868f3e6fad954466f029a) to display and run the code. Notice the structure of every chart in the code editor and try to replicate them with your previous collections. Change the geometries and try with polygons instead of points and compare (Hint: consider the spatial resolution of the MODIS images).

### Harmonization

Harmonization is the process of making comparable sensor bands and spectral values across sensors of the same family to ensure continuity in long-term time series. There are significant differences between the spectral characteristics of Landsat ETM+, TM and OLI that require normalization in this way to ensure changes detected are due to land cover changes and not sensor differences. Some of the reasons you might want to harmonize Landsat data include: producing a long time series that spans Landsat TM, ETM+, and OLI, generating near-date intra-annual composites to reduce the effects of missing observations from SLC-off gaps in Landsat ETM+, cloud/shadow masking, or increase the observation frequency within a time series.

The harmonization you will perform is a pixel-wise linear transformation of ETM+ to make it compatible with OLI spectral bands. The table below shows the coefficients and slope proposed in the Roy et al. (2016) method to make the linear transformation.

|  |  |  |
| --- | --- | --- |
| HARMONIZED BAND | Regression Type | Transformation functions using RMA & OLS regression coefficients |
| Blue l (~ 0.48 µm) | RMA | OLI = -0.095 + 0.9785 ETM |
|  | OLS | OLI = 0.0003 + 0.84474 ETM+ |
|  | OLS | ETM+ = 0.0183 + 0.8850 OLI |
| Green l (~ 0.56 µm) | RMA | OLI = -0.0016 + 0.9542 ETM |
|  | OLS | OLI = 0.0088 + 0.88483 ETM+ |
|  | OLS | ETM+ = 0.0123 + 0.9317 OLI |
| Red l (~ 0.66 µm) | RMA | OLI = -0.0022 + 0.9825 ETM |
|  | OLS | OLI = 0.0061 + 0.9047 ETM+ |
|  | OLS | ETM+ = 0.0123 + 0.9371 OLI |
| NIR l (~ 0.85 µm) | RMA | OLI = -0.0021 + 1.0073 ETM |
|  | OLS | OLI = 0.0412 + 0.8462 ETM+ |
|  | OLS | ETM+ = 0.0448 + 0.8339 OLI |
| SWIR\_1 l (~ 1.61 µm) | RMA | OLI = -0.0030 + 1.0171 ETM |
|  | OLS | OLI= 0.0254 + 0.8937 ETM+ |
|  | OLS | ETM+ = 0.0306 + 0.8639 OLI |
| SWIR\_2 l (~ 2.21 µm) | RMA | OLI = 0.0029 + 0.9949 ETM |
|  | OLS | OLI = 0.0172 + 0.9071 ETM+ |
|  | OLS | ETM+ = 0.0116 + 0.0165 OLI |

Every band in the table contains three linear expressions. The first two expressions convert ETM+ to OLI using RMA (Reduced Major Axis) and OLS (ordinary least squares) regression. In the following [LINK](https://code.earthengine.google.com/a3feae2e22b4e3799327a49b826a49cc), you can find a script that introduces you to the concept of harmonization for a long-term time series. Note that the coefficients for transforming ETM+ to OLI apply to TM according to Roy et al. (2016). Therefore, you can harmonize the three missions using just the harmonization functions from OLI to ETM+ and vice versa.

We do not do Landsat/Setinnel-2 harmonization in this practical, but you may be interested in it for your own work. Please see a couple of recent illustrations at the following links: <https://www.sciencedirect.com/science/article/pii/S0034425720300924> and <https://www.mdpi.com/2072-4292/12/2/281>.

* The image Collection from 1985 to 2020 chart represents the images per sensor used in the harmonization of NDVI values. The intra-annual median composite represents the NDVI median values for the long time series from 1985 to 2020. As you can see, the chart is smooth and continuous compared with the previous one. Try to display the same chart using one of the harmonized bands such as NIR or RED.
* Use the script to adjust the coefficients for the Landsat TOA collection and apply to different spectral vegetation indices, such as EVI in another area of interest.

## References

Azzari, G., & Lobell, D. B. (2017). Landsat-based classification in the cloud: An opportunity for a paradigm shift in land cover monitoring. *Remote Sensing of Environment*, *202*, 64–74. https://doi.org/10.1016/J.RSE.2017.05.025

Chen, J., Li, Y., Ma, Q., Shen, X., Zhao, A., & Li, J. (2018). Preliminary evaluation of sentinel-2 bottom of atmosphere reflectance using the 6SV code in Beijing area. *International Geoscience and Remote Sensing Symposium (IGARSS)*, *2018*-*July*, 7760–7763. https://doi.org/10.1109/IGARSS.2018.8517598

Kennedy, R. E., Yang, Z., & Cohen, W. B. (2010). Detecting trends in forest disturbance and recovery using yearly Landsat time series: 1. LandTrendr - Temporal segmentation algorithms. *Remote Sensing of Environment*, *114*(12), 2897–2910. https://doi.org/10.1016/j.rse.2010.07.008

Roy, D. P., Kovalskyy, V., Zhang, H. K., Vermote, E. F., Yan, L., Kumar, S. S., & Egorov, A. (2016). Characterization of Landsat-7 to Landsat-8 reflective wavelength and normalized difference vegetation index continuity. *Remote Sensing of Environment*, *185*, 57–70. https://doi.org/10.1016/j.rse.2015.12.024