Guidelines to assess trends in NDVI (1999-2020) time series using GEE and R

- ***Version***: 1
- ***Subject***: Ecoinformatics (UGR)
- ***Author***: Curro Bonet-García (fjbonet@gmail.com)

Objectives

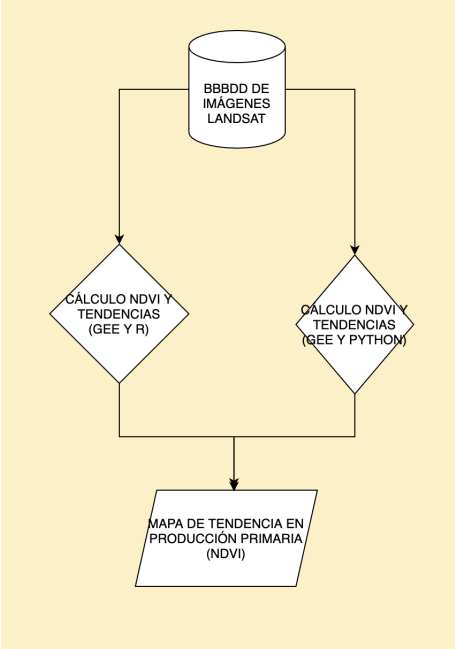
The overarching objective of this section is to create a map that shows the trend of photosynthetic activity of Sierra Nevada vegetation cover. This will be done using remote sensing techinques. More specifically, we will work with NDVI (Normalized Difference Vegetation Index). This index is considered as a good surrogate of primary production.

Besides, we have two types of learning objectives:

- Ecological: To improve our knolwedge on the functional aspects of terrestrial ecosystems. To connect with the importance of long term series to assess the impact of global change
- Informatics:
    - To discover the huge potential of Google Earth Engine to process remote sensing data.
    - To learn two different ways of executing the same workflow. This will elicit the multifunctionality of ecoinformatic techniques.

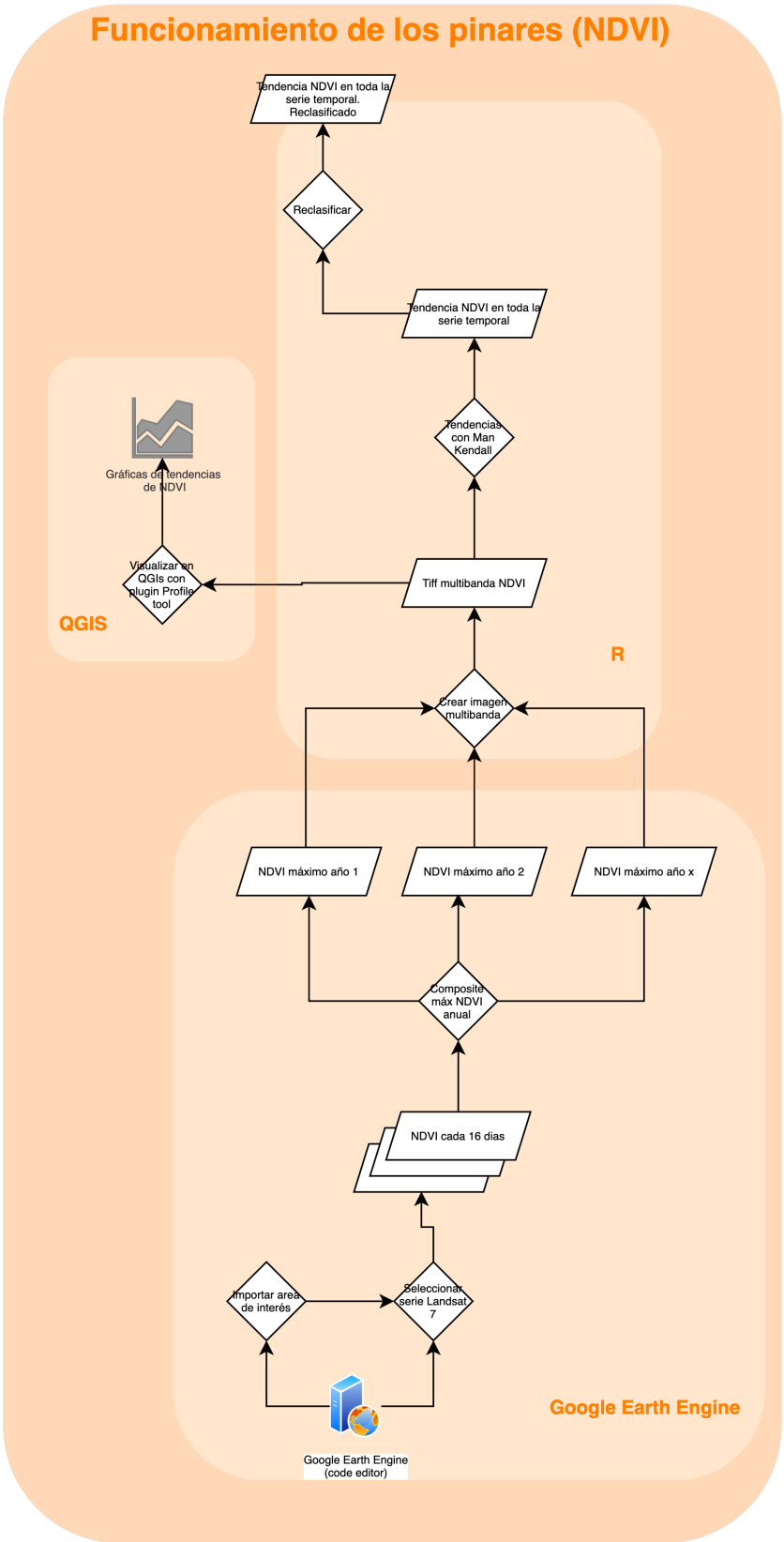
General workflow

The figure below shows the general workflow to achieve our objectives. The two threads depict the two different approaches that will follow to create a map showing the NDVI trend from 1999 to 2020. This document deals with the thread on your left (the one which uses GEE and R).



Besides, the following image, shows the step by step procedure that we will describe in this document.

Funcionamiento de los pinares (NDVI)



Step by step (Google Earth Engine). Load Landsat 7 collection. Create yearly NDVI composite.

- (0) This wokflow is also described in this video (Spanish).
- (1) Open Google Earth Engine code editor and login with your user name.
- (2) Create a new repository: Scripts -> New -> Repository. Name: ecoinformatics.
- (3) Create a new file: New-> file. Name: create_ yearly_NDVI
- (4) Since GEE contains information for the whole planet, we firstly define the area of interest. The example below considers that the area of interest is Sierra Nevada protected area. But, you should use a specific area of that region in order to overcome the limits of our internet connection.
    - (4.1) Import the limit of the area of interest (AOI). Each of you will use a different region. Go to the corresponding folder within our Classroom to find the limit of your AOI. You can also use this shapefile that comprises the borders of Sierra Nevada protected area.

      In order to import any of the above described shapefiles, just follow these steps: Assets-> New -> Table upload -> Shapefiles. Select all files called **pnatural_23030.**. You can also upload a *.zip* file containing all the files belonging to the shapefile.

    - (4.2) Add a name for the newly created asset: **pnatural_23030**
    - (4.3) Press *Upload*. After some seconds, a new asset will be created.
- (5) Add **pnatural_23030** to the script.

  Copy and paste the code below in your script. You have to change the path to your folder within GEE. Run the script to see the results.

```
var zona = ee.FeatureCollection("users/fjbonet/pnatural_23030");
Map.centerObject(zona,11);
```

- We have used the following functions:
    - *ee.FeatureCollection* (https://developers.google.com/earth-engine/feature_collections) can contain geometrical objects. In this case, we have used it to contain the limits of Sierra Nevada protected area.
    - *Map.centerObject* puts the focus on our area of interest. We will see Sierra Nevada when executing this line.

- (6) Load Landsat 7 collection. Filter by "zona" and year.
    - We will use the function *ee.ImageCollection* to access all the time series pertaining Landsat 7. More specifically, we will use the product called LANDSAT/LE07/C01/T1_SR to compute NDVI.
    - All the images existing in the above mentioned datasets will be stored in the variable called *l7collection*.
    - We will also filter all the images by area (using *filterBounds*) and by year (using *filterDate*). Since we will create a yearly NDVI image, we will have to change the year several times (22 times, from 1999 to 2020).

  Copy and paste the code below in your script. Run the script to see the results. Nothing happens? You have not asked GEE to show the results... Just add *print(l7collection);* to the last line of the script. Run again and you will see a list showing the description of all the images.

```
// Load landsat 7 dataset. Filter by "zona" and time.

var l7collection = ee.ImageCollection('LANDSAT/LE07/C01/T1_SR')
          .filterBounds(zona)
          .filterDate('2003-01-01', '2003-12-31');
```

- (7) Calculate NDVI for each image.
    - Since NDVI is a very commonly used index, GEE has implemented a function that calculates it in a very straightfoward way. *normalizedDifference* computes the following operation for bands *B3* (red) and *B4* (infrared): *(B4-B3)/(B4+B3)*
    - The obtained result is stored in a new band called *NDVI*
    - This process is "stored" in a function called *getNDVI*

  Copy and paste the code below in your script. Run the script to see the results. Nothing new happens? We have just create a new function (*getNDVI*), but we have not used it yet.

```
// Calculate NDVI
var getNDVI = function(img){
  return img.addBands(img.normalizedDifference(['B4','B3']).rename('NDVI'));
};
```

- (8) Use a loop to calculate NDVI for each image within the collection.
    - The funciton *map()* is a very simple way to create loops in GEE. It allows to "project" a function (*getNDVI*) over a collection (*l7collection*). The result is stored in a variable called *l7ndvi*
    - When using this function, GEE add a new band to each image called *NDVI*.

  Copy and paste the code below in your script. Check if the new band is created.

```
 // map over image collection. Calculates NDVI for each image in the collection
var l7ndvi = l7collection.map(getNDVI);
print(l7ndvi);
```

- (9) Create a composite (a single image from many others) containing the maximum NDVI value per pixel in the selected year.
    - Compositing is a very common process in remote sensing that refers to the process of combining spatially overlapping images into a single one based on an aggregation function. In this case, we will creates a single image that contains the maximum NDVI value per pixel. I.e. The maximum value of NDVI in the pixel *A* could occurr in the date *x*, while that maximum value in pixel *B* could occurr in date *y*. This means that the date of the image would be a composite of all the combined images.
    - The function *qualityMosaic* is in charge of create the above mentioned composite. It operates over the NDVI collection (*l7ndvi*) and stores the result in a variable called *composite*.

  Copy and paste the code below in your script. Check if the new image is created.

```
var composite = l7ndvi.qualityMosaic('NDVI').clip(zona);
print(composite);
```

- (10) Visualize the obtained image. We will use the function *Map.addLayer* to view the NDVI composite created above.
    - We will first create a variable called *ndviPalette* that contains 17 hexadecimal numbers. Each of them means a specific color. This is a palette: comma delimited list of color strings which are alinarly interplated between the maximum and minimum values in the visualization parameters.
    - Then, we will use the function *Map.addLayer* to add the band *NDVI* of the combined image (*composite*) using the above created palette (*ndviPalette*).

  Copy and paste the code below in your script. You will see a nice image showing the maximum NDVI value per pixel of the selected year.

```
//Visualize NDVI
var ndviPalette = ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718',
                '74A901', '66A000', '529400', '3E8601', '207401', '056201',
                '004C00', '023B01', '012E01', '011D01', '011301'];
Map.addLayer(composite.select('NDVI'),
             {min:0, max: 1, palette: ndviPalette}, 'ndvi');
```

- (11) Export the composite image to file. The last step of this workflow within GEE is to export the image to a *.tif* file.
    - The function *Export.image.toDrive* allows us to export the band *NDVI* from the image *composite* to a *.tif* file called *ndvi-1999*.
    - It is possible to specify the file format, the coordinate system and other properties of the exported image.
- When excecuting this function, a new task will appear in the corresponding tab. We must click on "run". A new box will appear that lets us change the name of the image and the folder of our Drive account where it will be saved.

  Copy and paste the code below and click Run on the script. We are almost done!

```
// Export to image
Export.image.toDrive({
  image: composite.select('NDVI'),
  description: 'ndvi_2003',
  scale: 30,
  region: zona,
  fileFormat: 'GeoTIFF',
  crs: 'EPSG:23030'
});
```

- (12) I am sure that you have realized that this whole process could be improved by creating a loop that interates the years. You are right. Including this looping feature will be one of the improvements that you will do using Python after this session.

Step by step (RStudio). Compute Mann Kendall time series analysis on NDVI yearly maps.

- (0) This wokflow is also described in this video (Spanish).
- (1) We first define the working directory. You just have to change the path to your own one.

```
#establish working directory
setwd("/Users/fjbonet_trabajo/ndvi")

#install and load required packages
if (!require(raster)) install.packages("raster")
if (!require(Kendall)) install.packages("Kendall")

library(raster)
library(Kendall)
```

- (2) Create a stack containing all *.tiff* files created by GEE. A RasterStack is a collection of Raster Layers with the same spatial extent and resolution. It can be created from raster files.
    - We first create a list containing the names of all *.tif* files that will be included in the stack.
    - Then, we use the above mentioned function (*Stack*) to create a multi-layer object.
    - Once we applied *Stack* function, we will use *Brick* function over it. A Brick is a multi-layer raster object. It is similar to *Stack*, but its processing time should be shorter.
    - Finally, we will export the created object to a *.tif* file. We will use this multi-layer file to create trends graphs in Quatum GIS.

```
# Create a list containing the names of all .tif files.
list_images <- list.files(pattern='*.tif', full.names=TRUE)
# Then, we create a stack with all of them
ndvis <- brick(stack(list_images))
plot(ndvis)
# Export the stack to .tif as a multiband file. It will be useful to create
trend graphs.
writeRaster(ndvis, filename="ndvi_1999_2020.tif", format="GTiff",
overwrite=TRUE)
```

- (3) Run Mann Kendall test over all "bands" within the multi-layer object. This test is very useful to analyze data collected over time for consistently increasing or decreasing trends. Since it is a non-parameetric test, it can be used for all distributions.
    - We will "embed" Mann Kendall test within an R function called *fun_k*
    - Then, we will create a new object called *kendal_result* that contains the results of applying Kendall function over the multi-layer object (*ndvi*)

- Mann Kendall test yield another spatial object containing five bands. We will pay attention to the first two ones:
  - *tau*= Kendall's *tau* statistic. This layer contain the trends that Kendall has found in our time series. Negative values means negative trends and viceversa. 0 means that there is no trend.
  - *sl*= Kendall's *sl* statistic. The signification value of the two-sided p-value. It indicates the statistical significance of the trend.
- Finally, we export the *tau* band to a *.tif* image called *tau.tif*

```
# We first create a function containing the MannKendall test
fun_k <-function(x){return(unlist(MannKendall(x)))}
# That function is applied over the above created stack.
kendal_result <-calc(ndvis, fun_k)
# The obtained results has several "bands". We will export "layer.1" which
depicts "tau" (Kendall's tau statistic)
writeRaster(kendal_result$layer.1, filename="tau_p.tif", format="GTiff",
overwrite=TRUE)
plot(kendal_result$layer.1)
```

- (4) It is also interesting to reclassify the *tau* layer, in order to obtain a shorter list of classes. By using the function *reclassify*, as follows:
  - This process requires the creation of a R numeric string (called *m*) containing the reclassify table.
  - This string is converted to a matrix called *rclmat* using the function *matrix*
  - A new raster object is created (*rc _ tau*) when applying the *reclassify* function to the original raster (*kendal_result$tau*)
  - We finally create a new raster with the reclassified information.

```
m <- c(-1, -0.25, -1, -0.25, 0.25, 0, 0.25, 1, 1)
rclmat <- matrix(m, ncol=3, byrow=TRUE)
rc_tau <- reclassify(kendal_result$layer.1, rclmat)
writeRaster(rc_tau, filename="rc_tau.tif", format="GTiff", overwrite=TRUE)
```

From old value	To old value	New value
-1	-0.25	-1
-0.25	0.25	0
0.25	1	1

Browsing results using QGIS

- (1) Add the following layers to an empty QGIS project:
    - *tau.tif*: Raster layer showing the NDVI trend per pixel.
    - *rc_tau.tif*: reclassified version of the above mentioned raster layer.
    - *ndvi_1999_2020.tif*: This a multiband raster image that contains all the NDVI images created in GEE.
    - WMS service showing the most recent ortophoto: http://www.ign.es/wms-inspire/pnoa-ma
- (2) Display *tau.tif* using a *singleband pseudocolor* render type. Adjust the maximum value to 0.99 and click to "*clip out of range values*". Add some transparency.
- (3) We will build a nice graph showing the NDVI trend of any selected pixel. In order to do that, follow these steps within QGIS:
    - Install a plugin called "*Temporal/Spectral profile tool*". Menu *plugins->Manage and install plugins*. The instalation will create a new bottom that depicts a red graph.
    - Select the layer called "*ndvi_1999_2020*" in QGIS.
    - Click on the plugin bottom.
    - Click on the *settings* tab and add the amount of layers that we are using: 1, 2, 3, ...22.
    - Click on any pixel of the selected layer and you will see a graph showing its NDVI trend. See image below.