Changes and trends in remotely-sensed Ecosystem Functional Attributes: A pilot assessment with the Google Earth Engine platform

Bárbara Ferreira

Supervised by: João Gonçalves[1], Eduardo Marques[2] and João Honrado[1]

[1]Faculdade de Ciências da U. Porto / CIBIO/InBIO – Centro de Investigação em Biodiversidade e Recursos Genéticos da Universidade do Porto (ECOCHANGE group) [2]Faculdade de Ciências da U. Porto / Departamento de Ciência de Computadores – FCUP / CRACS / INESC-TEC.

Abstract

Motivation: Geospatial cloud computing platforms such as Google Earth Engine (GEE) are providing exciting new opportunities to handle, process and model large volumes of Earth Observation (EO) data at speeds much higher than those of desktop-based environment. Despite these advantages, platforms such as GEE are missing required processing algorithms such as a time-series smoothing algorithms (e.g. Whitaker-Anderson). Such routines are important for improving the signal-to-noise ratio in EO-based spectral vegetation indices (VI) time-series used to proxy quantify ecosystem's primary production dynamics and its spatiotemporal change.

Results: The Moving Average Smoother (MAS) preserves more the time-series while the Whittaker Smoother (WS) is less affected by extreme values. We also observed greatest anomalies in the centre-left of the Iberian Peninsula and positive trends in the mountain areas of the Iberian Peninsula.

Supplementary information: Supplementary data are available at the end of the report (Appendix).

1 Introduction

Monitoring vegetation and/or land surface dynamics is crucial to assess global changes and predict future events (such as droughts and wildfires). Remote Sensing (RS)/Earth Observation (EO) can be used for this purpose. Remote sensing consists in acquiring information about the Earth's surface without actually being in contact with it. This is done by sensing and recording reflected or emitted light and processing this data (Joseph, 2005). Data products of RS are available at multiple spatial and temporal resolutions thus supporting multiple applications in environmental and ecological sciences.

For instance, RS can provide biophysical descriptors of ecosystem function, known as Ecosystem Functional Attributes (EFAs) (Alcaraz-Segura, et al., 2017). EFAs describe the exchanges of matter and energy between the biota and the physical environment, in particular, indicators of productivity, seasonality and phenology of carbon gains (Alcaraz-Segura, et al., 2017). The use of EFAs derived from satellite data as predictor variables for monitoring and modelling has numerous advantages. First, they can be easily monitored though RS at different spatial scales and over large extents, using a common protocol (Foley, et al., 2007). Second, EFAs show a rapid response to environmental changes than structural or compositional attributes (for example land-cover and species richness) (Mouillot, et al., 2013), possibly allowing to anticipate and predict future events. Third, EFAs offer an integrative response to environmental changes, so species changes can be linked to pressures on the functioning of the ecosystem.

Some spectral indexes derived from satellite data are related to functional variables of ecosystems, such as primary production, surface temperature, and albedo (Alcaraz-Segura, et al., 2009). Currently, several spectral vegetation indices exist and perhaps the most widely used one is the Normalized Difference Vegetation Index (NDVI) (Gonçalves, et al., 2016). This spectral index is a linear estimator of the fraction of absorbed photosynthetically active radiation intercepted by vegetation (Alcaraz-Segura, et al., 2009). The main control of carbon gains is through radiation interception, so the NDVI has been widely used to describe regional patterns of primary net productivity, which is the most integrative indicator of ecosystem functioning (McNaughton, et al., 1989; Virginia and Wall, 2001). The NDVI, as a descriptor of ecosystem functions, has been shown to greatly contribute for application in conservation biology, ecosystem management and in analyzing ecological responses to environmental changes (Alcaraz-Segura, et al., 2009). In addition, NDVI time series obtained from RS sources are valuable for exploring trends in land-surface phenology as well as seasonality (Alcaraz-Segura, et al., 2009).

Complementarily, geospatial cloud computing platforms such as Google Earth Engine (GEE) are providing exciting new opportunities to handle, process and model large volumes of Earth Observation (EO) data at speeds much higher than those of desktop-based environments (Gorelick, et al., 2017). Despite these advantages, platforms such as GEE are challenging due to their steep learning curve and, also, because many required processing algorithms are still lacking from its core API. An example is the inexistence of time-series smoothing algorithms (e.g., Whitaker-Anderson). Such routines are important for improving the signal-to-noise ratio in EO-based spectral vegetation indices (VI) time-series used to proxy quantify ecosystem's primary production dynamics and its change. Google Earth Engine is a cloud-based platform for planetary-scale environmental data analysis. This platform is available through two client libraries (Javascript and Python) that provide wrappers around the web API. Both APIs do not run the code directly on the Earth Engine servers at Google. Instead, the client library encodes the script into a set of JSON objects,

sends the objects to Google and waits for a response. No processing is done on the server until that result is explicitly requested i.e., 'lazy evaluation' (Gorelick, et al., 2017).

The main objective of this project was to evaluate the potential of GEE to calculate measures of Ecosystem Functional Attributes (EFAs, mainly related to the annual dynamics of primary productivity and energy-matter flows), based on NDVI time series available in the MODIS product MOD13Q1. In order to achieve this overarching objective, we devised three more specific goals (Fig. 1):

i. The development of computational solutions which use the GEE to calculate annual measures related to EFAs based on time-series of NDVI currently available on this platform;
ii. To implement time-series smoothing to improve image data and reduce noise;
iii. Use the calculated annual EFA indicators to evaluate changes, trends and dynamics in ecosystems and land surface.

This report is structured as follows. Section 2 consists in the description of the routines used in this project. The section 3 comprises the results obtained and section 4 the discussion of these results. The last section contains the main conclusion from the conducted work.

2 Methods

2.1 Proposed workflow

Bellow, we present the proposed workflow for this project (Fig.1.)

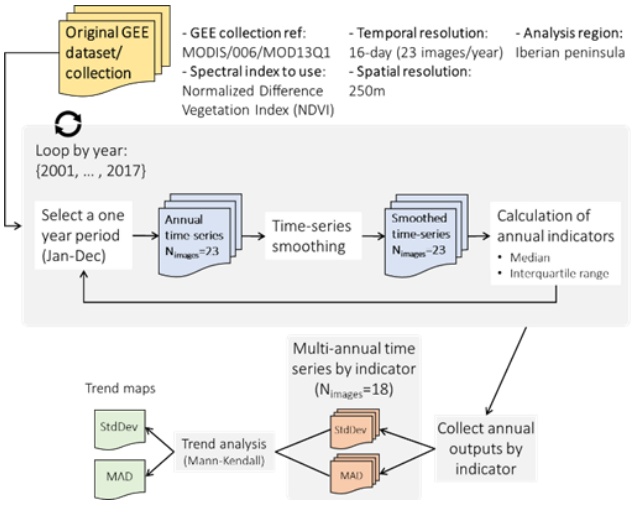

Fig. 1. Proposed workflow

2.2 Study area description

The study area chosen was the Iberian Peninsula and it is bounded by 10.151 °W – 4.307 °E longitude and 35.496 °N – 44.151 °N. This area is a very heterogeneous region in terms of biogeography, climate, orography, geology, and soil types. As several studies have mentioned, this region has many desirable properties for RS studies (LLoyd, 1989; Lobo, et al., 1997). Regardless of the small size (581 000 km2), it has an enormous landscape diversity because of its climate, geological features and biogeographical location (Alcaraz, et al., 2006).

2.3 Spectral vegetation indices and MODIS products

Satellite data are often used in the form of spectral indices. These indices are mathematical combinations of two or more spectral bands selected to describe the parameters of interest (Gonçalves, et al., 2016). As mentioned above, the NDVI is one of the most widely used vegetation indices (Gonçalves, et al., 2016) and is a linearly related to the fraction of absorbed photosynthetically active radiation intercepted by vegetation (Alcaraz-Segura, et al., 2009). The NDVI is calculated from the reflectance in the red (R) (580 – 680 nm) and near-infrared (NIR) (725 – 1100 nm) wavelengths as follows:

$$NDVI = (NIR - R)/(NIR + R) \quad (1)$$

The index varies between -1.0 and 1.0, with mid to high positive values for healthy vegetation, low positive values for sparsely vegetated soils and very low positive and negative values for water bodies or urban areas (IEAGHG). Despite its usefulness, NDVI has got some limitations related to, for example:

- Measurements can be affected by moisture and aerosols;
- Occurrence of saturation when there are high levels of biomass or high leaf area index;
- Interference of the clouds with the measurement can occur if they cover the whole area or if they create shadows, leading to misinterpretations of the NDVI values (IEAGHG).

In this study, we used image data from the Moderate Resolution Imaging Spectroradiometer (MODIS), namely the MODIS Terra Vegetation Indices 16-Day L3 Global 250m version-6 MOD13Q1 product, available in Google Earth Engine platform. This data set consists of 16-day composites with a spatial resolution of 250m. For this study, we used the NDVI band for the period of time from 2001 to 2018 (however, only complete years currently available from MODIS archive were used to calculate annual metrics). More information on the MODIS MOD13Q1 product is available in Table 1.

Table 1. Main characteristic of the bands of the MODIS MOD13Q1 product

Description of band	Units	Valid Range	Scale Factor
16-day NDVI average	NDVI	-2000 to 10000	0.0001
16-day EVI average	EVI	-2000 to 10000	0.0001
VI quality indicators	Bit field	0 to 65534	N/A
Surface Reflectance Band 1	Reflectance	0 to 10000	0.0001
Surface Reflectance Band 2	Reflectance	0 to 10000	0.0001
Surface Reflectance Band 3	Reflectance	0 to 10000	0.0001
Surface Reflectance Band 7	Reflectance	0 to 10000	0.0001
View zenith angle of VI Pixel	Degree	0 to 18000	0.01
Sun zenith angle of VI pixel	Degree	0 to 18000	0.01
Relative azimuth angle of VI pixel	Degree	-18000 to 18000	0.01
Day of year VI pixel	Julian day	1 to 366	N/A
Quality reliability of VI pixel	Rank	0 to 3	N/A

2.4 Implementing analysis/processing routines in Google Earth Engine

As mentioned above, the GEE is a computing platform that allows users to perform geospatial analysis either through the Javascript or the Python API. To start working with GEE, we first select the product we want to analyse (that will be an ImageCollection). Then, we select the band we want to analyse (in this example the NDVI), then we filter the collection by date and region of interest (the complete 2017 year and the Iberian Peninsula). To filter by bounds, there are several ee.Geometry types available at GEE. After selecting the ImageCollection, we can extract annual metrics

(in this example median), through the use of a reducer (ee.Reducer). This process is illustrated bellow (Code Snippet 1):

```
bounds = ee.Geometry.Polygon(
  [[[-10.1513671875, 44.11914151643737],
  [-10.1513671875, 35.496456056584165],
  [1.669921875, 36.06686213257888],
  [4.306640625, 42.90816007196054],
  [-1.7138671875, 44.15068115978094]]]);

collection = ee.ImageCollection('MODIS/006/MOD13Q1')
  .select('NDVI').filterDate('2017-01-01', '2017-12-31')
  .filterBounds(bounds);

median = collection.median();
```

Code Snippet 1. Getting MODIS MOD13Q1 collection in GEE. Example to show how to get the product to analyze, select bands, filter by date and bounds and the use of a reducer.

In GEE, the reducer is used to obtain an individual image (Fig. 2) from a collection of images. The reducers are very useful for the extraction of annual metrics, calculation of trends and other statistics. In this project, we also needed to use the ee.Array type available in the GEE, to help the implementation of the Whittaker algorithm, such as creation the auxiliary matrixes and to solve the system of equations. The GEE represents 1-D vectors, 2-D matrices, 3-D cubes, and higher dimensional hypercubes with the ee.Array type. The dimension of an array is the number of axes along which the data varies. For instance, 0-D arrays are scalar numbers, 1-D arrays are vectors, 2-D arrays are matrices, 3-D arrays are cubes, and >3-D arrays are hypercubes.

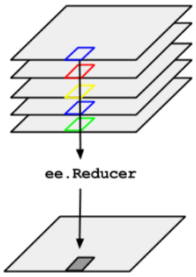

Fig. 2. Illustration of a reducer applied to an ImageCollection (Adapted from https://developers.google.com/earth-engine/reducers_image_collection)

When working with the GEE, some processes can be automated through functions, so we created an ancillary library with some of the functions we used the most (see Appendix A). As mentioned above, we first need to select a collection and then filter it by date and the required spatial bounds. In order to do this, we defined a function (getCollection) for retrieving and filtering an ImageCollection (Code snippet 2).

```
def getCollection(roi, startDate, endDate,
                  product='MODIS/006/MOD13Q1',
                  bands=['NDVI']):
    return ee.ImageCollection(product).select(bands)
                                      .filterDate(startDate,endDate)
                                      .filterBounds(roi)
```

Code snippet 2. Getting MODIS MOD13Q1 collection in GEE.

Then if the collection has missing values, we can replace them with the function setValueForMaskedPixels. This function creates a mask in the selected band in order to replace the missing values (Code Snippet 3):

```
def setValueForMaskedPixels(coll, band, value):
    def auxProc(img):
      return img.addBands(img.select(band).unmask(value), overwrite=True)
    return coll.map(auxProc)
```

Code Snippet 3. Function setValueForMaskedPixels for replacing missing values

After this pre-processing stage, we will test and compare two smoothing algorithms: (i) the moving average smoothing (MAS) and (ii) Whittaker smoothing (WS) (Eilers, 2003). The MAS algorithm had already a proposed solution using the GEE Javascript API (https://gis.stackexchange.com/questions/250426/moving-averages-and-seasonality-filters), so we only had to adapt for the Python API and generalize some of its input parameters. The WS algorithm was not implemented in the GEE API, so we implemented it, using the GEE Python API. In order to validate our GEE implementation of the WS algorithm we compared it against other implementation that we also developed using only Python (and without GEE processing). Finally, we extracted two annual metrics from the NDVI time series, the median (related to productivity) and the interquantile range (related to the annual seasonal variation). Addicionally, we also annalysed the trends through the Kendall's Tau-b rank correlation.

2.4.1 Moving average smoothing

The MAS is a simple algorithm that creates a new data set where each value is the average of the n neighbours of a given focal point in time. In this study and considering that our data has a frequency of 16-days, we tested two and four nearest neighbours (i.e., either 17 (equation (2)) or 33 days).

$$\bar{p}_i = \frac{p_{n-1} + p_i + p_{n+1}}{3} \qquad (2)$$

There was already a proposed solution to implement this algorithm in the GEE Javascript API, so we only adapted to the GEE Python API and created a function with this solution (Code Snippet 4).

```
def movingAverageSmoother(coll, days=17, bandName='MAS'):
    diffFilter = ee.Filter.maxDifference(difference=1000*60*60*24*days,
                                         leftField = timeField,
                                         rightField= timeField);
    threeNeighborJoin = join.apply(primary=coll,
                                   secondary=coll,
                                   condition= diffFilter)

    def smooth(image):
      collection = ee.ImageCollection.fromImages(image.get('images'))
      return ee.Image(image).addBands(collection.mean().rename(bandName))

    smoothed = ee.ImageCollection(threeNeighborJoin.map(smooth))
    return smoothed
```

Code snippet 4. GEE implementation of the Moving Average Smoother (MAS).

2.4.2 Whittaker smoothing

In the beginning, we have a time series *y*, of length *m* and we want to fit a series *z* to *y*. To do this, we have to balance two points: first, the fidelity to the original data and, second, the roughness of *z*. The smoother the *z* is, the more it will deviate from the original data. The fidelity to the original data can be measured as the sum of squares of differences: $S = \sum_i (y_i - z_i)^2$. For measuring the roughness of *z*, we can use differences ($\Delta z_i = z_i - z_{i-1}$). Squaring and summing these differences gives us a simple and effective measure of the roughness of *z*: $R = \sum_i (\Delta z_i)^2$. A balanced combination these two points is the sum $Q = S + \lambda R$, where λ is a number chosen by us. The goal of the penalized least squares is to find the series *z* that minimizes *Q*. The larger the λ, the stronger the influence of *R* on the *Q* is, making *z* smoother. This can make the fit of the data worse (Eilers, 2003). To make the calculations easier, we will use matrices and vectors:

$$Q = |y - z|^2 + \lambda |Dz|^2 \qquad (3)$$

Where $|a|^2 = \sum_i {a^2}_i$ indicates the quadratic norm of any vector *a*, and *D* is a matrix such as $Dz = \Delta z$. To find the vector of partial derivatives, we use results from matrix calculus:

$$\frac{\partial Q}{\partial z^T} = -2(y - z) + 2\lambda D^T D z \qquad (4)$$

And when we solve the equation to 0, we get the following linear system of equations:

$$(I + \lambda D^T D) z = y \qquad (5)$$

Where *I* is the identity matrix.

In order to be able to handle missing data, the missing elements of *y* are simply set to an arbitrary value, and a vector *w* of weights is introduced,

with $w_i = 0$ for missing observations and $w_i = 1$ otherwise. Then, we get the following final system of equations that allow us to calculate z:

$$(W + \lambda D^T D)z = Wy \quad (6)$$

Where $W = diag(w)$, a diagonal matrix with w on its diagonal (Eilers, 2003). This last approach was not implemented in this project.

In order to do this using the GEE, we created a function (whittakerSmoother) that creates the matrixes I, D and converts the ImageCollection into an array. Then we solve the previous equations for finding z (Code Snippet 5).

```
def whittakerSmoother(coll, plambda, d=1):
    plambda = ee.Number(plambda)
    m = coll.size().getInfo()
    I = ee.Array.identity(m)
    D =  diffArray(I, d)
    A_arr = (D.matrixMultiply(D.transpose()).multiply(plambda)).add(I)
    A_img = ee.Image(A_arr)
    z_img = A_img.matrixSolve(coll.toArray())
    z_coll = ee.ImageCollection(z_img)
    return z_coll
```

Code snippet 5. GEE implementation of the Whittaker smoother.

3 Results

3.1 Smoothing methods

3.1.1 Validation

Numerical comparison (Table 2) between the GEE implementation of Whittaker's smoothing algorithm (Code Snippet 5) and a version of the algorithm implemented using the numpy Python library (Appendix A, whittakerSmoothNumpy method).

Table 2. Max absolute error between GEE implementation of Whittaker and the Python version

Type of region	Max Absolute Error
Urban	1.8×10^{-12}
Agriculture	4.5×10^{-12}
Broadleaf	3.6×10^{-12}
Coniferous	3.6×10^{-12}

3.1.2 Comparison between smoothers

As we can see from the plots of Fig. 3, the MAS when d = 17 is more sensitive to negative outliers, which are related to clouds, but it has a better performance on the positive peaks (typically with much less measurement uncertainty and related to the maximum vigour of vegetation during the annual cycle). However, we can see that for certain types of soil (Broadleaf) the d = 33 is better because it is less sensitive to extreme negative outliers (the time series is 'smoother').

When we compare the $\lambda = 3$ and the $\lambda = 5$ in the WS, we can see that the results are very similar, but, as we expected, the $\lambda = 5$ has a higher smoothing effect on the outliers. In general, we can see a higher smoothing in the agriculture and urban classes.

Regarding the comparison between smoothers, the difference is higher. In particular, the MAS preserves better the annual peaks of NDVI. This will lead to higher intra-annual values, and also, higher differences between the calculated metrics, such as the interquartile range. In particular, as we can see from the plots in Figure 3, the agriculture zones might be the most affected by the difference between the algorithms.

3.2 Extraction of annual metrics from time-series

Regarding the median, we can see in the maps (Figure 4) that the highest values of NDVI are located in the north area of the Iberian Peninsula and the lowest values are located in the highest altitude of the Pyrenees Mountains and some agriculture regions. In the maps, we can also see that in 2005 and 2010 and increasing in vegetation (higher NDVI value) in the centre-left region of the Iberian Peninsula.

Regarding the interquartile range, the NDVI highest values are located the Pyrenees Mountains and the Cantabrian Mountains. The lowest values of NDVI are located in the northwest and the southeast coastline of the Iberian Peninsula. In the maps, we can also see that in 2005 and 2010 and increasing in vegetation (higher NDVI value) in the centre-left region of the Iberian Peninsula for both metrics.

3.3 Changes and trends in Earth Surface in the Iberian Peninsula

In the map from Fig. 6, we can see that the greatest anomalies occur in the center-left of the Iberian Peninsula. In Portugal, we can clearly see this difference in the Arga Mountain Range.

As we can see from the map with the Kendall's Tau-b rank (Fig. 7), in general, we can see positive trends, namely in the regions with mountains, the negative trends are located in to the region to the South of Valencia and the region to the Southeast of the Cabañeros National Park.

4 Discussion

4.1 Smoothing algorithms

The lack of time-series smoothing algorithms in the GEE platform provide a challenge in the analysis of data. As we can see from the results, both algorithms tested are valid and correctly implemented. In terms of performance the MAS algorithm is faster than the WS algorithm because the first one only calculates the average and the second involves solving systems of equations. In terms of extraction of annual measures of Ecosystem Functioning, both algorithms affect the interquartile range (although in the WS the effects are higher), in the way that the smoother the time-series, the smaller the interquartile range will be. However, the smoothing algorithms can be helpful in reducing the influence of extreme values. Also, the performance will vary depending on the target land cover class and the distribution of clouds across the Iberian Peninsula.

The WS implementation in this project has a limitation of not being able to convert the obtained ee.Array with the smoothed time-series in an ImageCollection with the timestamps needed for the construction of the map.

4.2 Annual metrics

As mentioned above, the median of NDVI is higher in the Eurosiberian region, as expected because of the presence of broadleaf deciduous or marcescent forests. In contrast, the median NDVI is generally lower in agriculture areas but this may vary depending on crop type and cycle (for example annual crops will have bare soil in some parts of the annual cycle leading to a strong decrease in NDVI). In the north and northwest of the Iberian Peninsula with temperate climate, we can see high levels of vegetation activity in comparison with the other Iberian Peninsula regions with mediterranean climate. This difference might be due to the availability of water supplies throughout the year. Regarding the interquartile range, the highest values of NDVI are observed at mountains with seasonal snow coverage, at forests with deciduous trees and, agricultural regions with

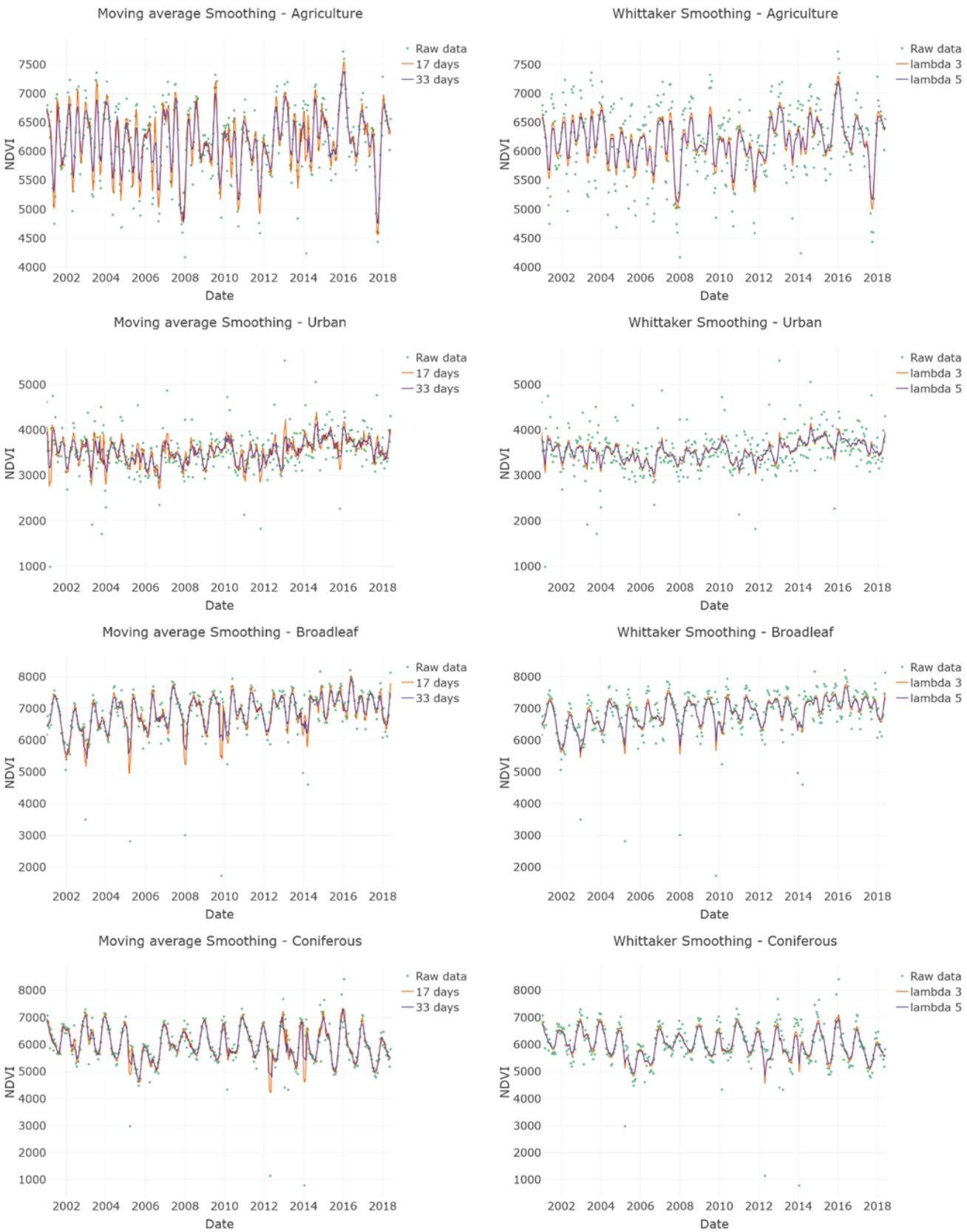

Fig.3. Plots representing the Moving Average Smoother (MAS) and the Whittaker Smoother (WS) on different areas. The MAS was tested with two different days (17 and 32) and the WS was tested with two different λ (3 and 5). The regions are located at: Agriculture – 41.715 °N, 8.786 °W; Urban - 41.691 °N, 8.832 °W; Broadleaf – 41.818 °N, 8.284 °W; Coniferous – 41.696 °N, 6.979 °W

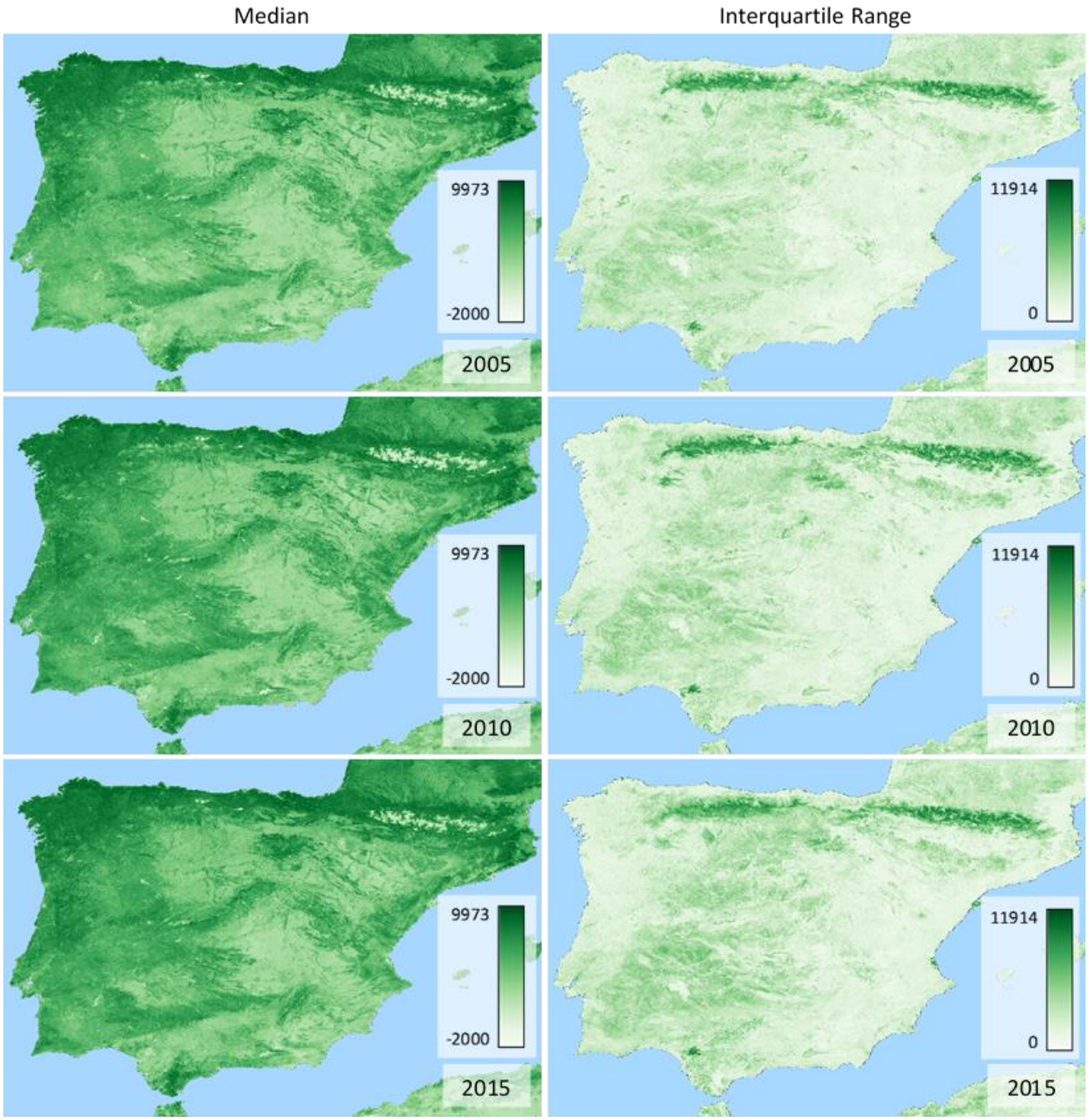

Fig. 4. Maps of annual metrics (median and interquartile range) for different years (2005, 2010, 2015).

annual crops or with evergreen plantations without irrigation. If the agriculture region has annual crops, it is when the highest values of the interquartile range of NDVI occur. This might be due to the fact that we observe extreme values of NDVI, for example when the plantation is starting we observe the lowest value of NDVI and before harvesting we observe the highest NDVI values. This leads to a higher interquartile range.

Regarding the anomalies observed (Fig.6.), we can clearly see in the region zoomed (Arga Mountain Range) that we have a high deviation from the median, this is expected since there was a major wildfire in the region in 2004. Overall, areas affected by disturbances (such as forest fires or droughts) or by the change of the land usage (such as cutting forest areas will present lower stability/ higher fluctuations (thus higher MAD values).

The Kendall's Tau-b rank express the variation or trends in the long term. The observed positive trends (Mountains regions) are in accordance with those found in the literature (Alcaraz-Segura, et al., 2008; Gonzalez-Alonso, et al., 2004). The negative trends observed might be due to a change in land usage or a change in the management of vegetation (Alcaraz-Segura, et al., 2008). In general, a positive trend in the median means that there was a cumulative increase in the biomass and vegetal coverage. The negative trends observed might suggest progressive changes of loss of vegetal coverage (artificialization of the land).

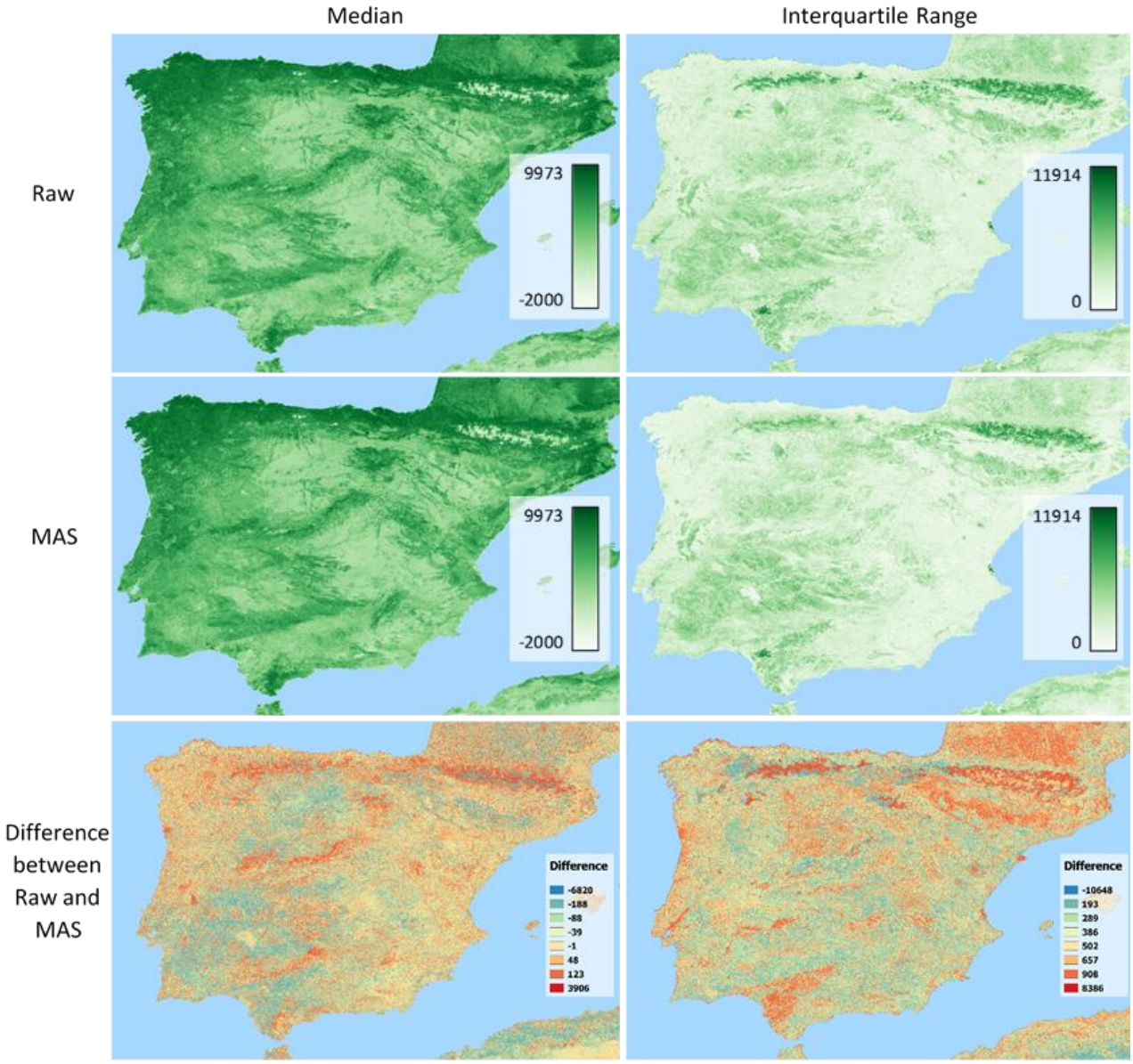

Fig. 5. Maps with the original data, after the moving average smoothing and the difference between the original data and the smoothing for two different metrics: median and interquartile range. The extreme values of differences are outliers.

5 Conclusion

In this project, we showed an application for the GEE platform in pre-processing the data and extracting metrics for quantifying several Ecosystem Functioning Attributes. In turn, these allow to evaluate several important characteristics regarding ecosystem dynamics both regarding their seasonal variation but also across multiple years thus portraying changes and trends in Earth's surface. This platform presents a great opportunity for managing large volumes of data since the computation is made on the GEE servers and therefore it does not require a high computational power. However, there is a steep learning curve since it requires some programming skills.

RS variables also present some challenges in analysis due to missing values coming from various sources (for example clouds). This is why it is important to implement smoothing algorithms, and, in this project, we were able to test two algorithms. However, the WS implemented in this project has a limitation of not being able to extract a map for better visualization (at least in this first version of the implementation).

This project is a starting point to start using the GEE for ecosystem studies. Future perspectives for the continuation of this study, include the implementation of the WS to be able to extract maps from the GEE, calculate other metrics (average, minimum and maximum) and use other vegetation indexes available at the GEE. Additionally, it would be interesting to scale this work for the global scale.

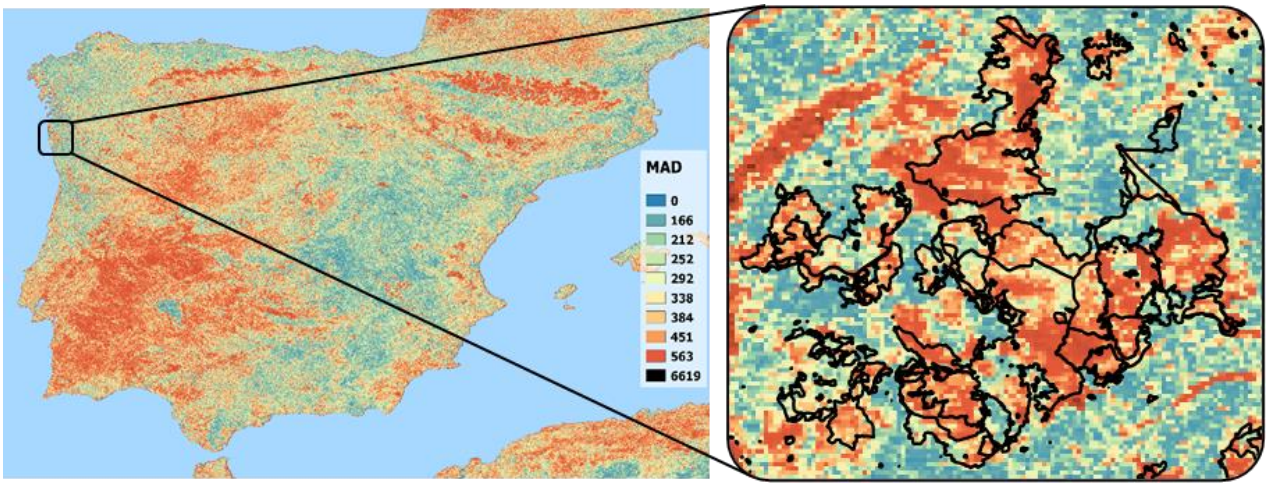

Fig.6. Map with the median absolute deviation (MAD) for the years between 2001 and 2017. Detailed region of the Arga Mountain Range. The extreme values of differences are outliers.

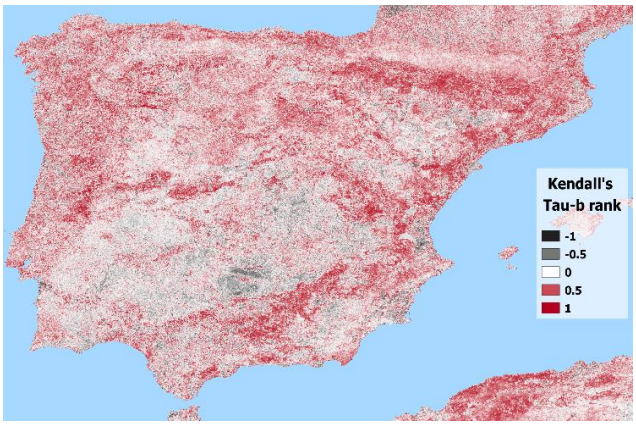

Fig.7. Map with the Kendall's Tau-b rank over the median of NDVI. Positive trends are colored red and negative trends colored grey/black.

References

Alcaraz-Segura, D., et al. Trends in the surface vegetation dynamics of the National Parks of Spain as observed by satellite sensors. Applied Vegetation Science 2008;11(4):431-440.

Alcaraz-Segura, D., et al. Use of descriptors of ecosystem functioning for monitoring a national park network: a remote sensing approach. Environmental Management 2009;43(1):38-48.

Alcaraz-Segura, D., et al. Potential of satellite-derived ecosystem functional attributes to anticipate species range shifts. International journal of applied earth observation and geoinformation 2017;57:86-92.

Alcaraz, D., Paruelo, J. and Cabello, J. Identification of current ecosystem functional types in the Iberian Peninsula. Global Ecology and Biogeography 2006;15(2):200-212.

Eilers, P.H. A perfect smoother. Analytical chemistry 2003;75(14):3631-3636.

Foley, J., et al. Amazonia revealed: forest degradation and loss of ecosystem goods and services in the Amazon Basin. Frontiers in Ecology and the Environment 2007;5(1):25-32.

Gonçalves, J., et al. Exploring the spatiotemporal dynamics of habitat suitability to improve conservation management of a vulnerable plant species. Biodiversity and Conservation 2016;25(14):2867-2888.

Gonzalez-Alonso, F., et al. Spanish vegetation monitoring during the period 1987–2001 using NOAA-AVHRR images. International Journal of Remote Sensing 2004;25(1):3-6.

Gorelick, N., et al. Google Earth Engine: Planetary-scale geospatial analysis for everyone. Remote Sensing of Environment 2017;202:18-27.

IEAGHG. Quantification Techniques For CO2 Leakage - 2012/02 January, 2012. In.; 2012.

Joseph, G. Fundamentals of Remote Sensing. Universities Press; 2005.

LLoyd, D. A phenological description of Iberian vegetation using short wave vegetation index imagery. International Journal of Remote Sensing 1989;10(4-5):827-833.

Lobo, A., Marti, J.I. and Gimenez-Cassina, C.C. Regional scale hierarchical classification of temporal series of AVHRR vegetation index. International Journal of Remote Sensing 1997;18(15):3167-3193.

McNaughton, S.J., et al. Ecosystem-level patterns of primary productivity and herbivory in terrestrial habitats. Nature 1989;341:142.

Mouillot, D., et al. A functional approach reveals community responses to disturbances. Trends in Ecology & Evolution 2013;28(3):167-177.

Virginia, R. and Wall, D. Ecosystem function, principles of. Encyclopedia of Biodiversity. 2001.

Appendix A

```
"""
BBC project GEE library.
@author: Bárbara Ferreira, João Gonçalves, Eduardo Marques
"""

# Imports
import ee
import plotly.offline as py
from plotly.graph_objs import *
import pandas as pd
import plotly.graph_objs as go
import numpy as np

# Library functions
def initialize():
    '''
    Perform initialisation related to GEE or other libraries/resources.
    '''
    ee.Initialize()

def getCollection(roi,startDate, endDate, product='MODIS/006/MOD13Q1', bands=['NDVI']):
    '''
    Get image collection from GEE.

    Parameters
    ----------
    roi:
        Region of interest.
    startDate:
        Start date (YYYY-MM-DD format).
    endDate:
        End date (YYYY-MM-DD format).
    product:
        Product of interest; the default is 'MODIS/006/MOD13Q1'.
    bands:
        List of bands; the defaults is ['NDVI'].

    Returns
    -------
    Image collection for given product, period, and bands
    '''
    return ee.ImageCollection(product).select(bands).filterDate(startDate,endDate)\
                                      .filterBounds(roi)
```

```
# Smoothing algorithms
def setValueForMaskedPixels(coll, band, value):
    '''Set value for masked pixels in a band.

    Parameters
    ----------
    coll:
        Image collection.
    band:
        Band name.
    value:
        Value to use for masked pixels.

    Returns
    -------
    Image collection reflecting the transformation.
    '''
    def auxProc(img):
      return img.addBands(img.select(band).unmask(value), overwrite=True)
    return coll.map(auxProc)

def annualMedian(sYear, eYear, collection):
    '''Calculate the selected metric by year iteratively.

    Parameters
    ----------
    sYear:
        Start year.
    eYear:
        End year.
    collection:
        ImageCollection filtered by area and band.

    Returns
    -------
    Image collection with the calculated metrics by year.
    '''
    yrs = ee.List.sequence(sYear, eYear)
    coll = collection
    def aux(yr):
        return coll.filter(ee.Filter.calendarRange(yr, yr, 'year')).median().set('year', yr)
    metric = ee.ImageCollection.fromImages(yrs.map(aux))
    return metric

def movingAverageSmoother(coll, days=17, bandName='MAS'):
    '''
    Smooth image collection values using a moving average.
    Parameters
    ----------
    coll:
        Image collection.
    days:
        Number of days to consider for moving average; the default is 17.
    bandName:
        Moving average band name; the default value is 'MAS'.

    Returns
    -------
    Collection with moving average band added.
    '''
    join = ee.Join.saveAll(matchesKey = 'images')
    timeField = 'system:time_start'
    diffFilter = ee.Filter.maxDifference(difference=1000 * 60 * 60 * 24 * days,
                                         leftField = timeField,
                                         rightField= timeField);
    threeNeighborJoin = join.apply(primary=coll,
                                   secondary=coll,
                                   condition= diffFilter)
    def smooth(image):
      collection = ee.ImageCollection.fromImages(image.get('images'))
      return ee.Image(image).addBands(collection.mean().rename(bandName))
    smoothed = ee.ImageCollection(threeNeighborJoin.map(smooth))
    return smoothed

def diffArray(array, d):
    '''
    Ancillary function to calculate D.

    Parameters
    ----------
    array:
        Identity matrix.
    d:
        Order parameter; the default value is 1.

    Returns
    -------
    Transpose difference matrix.
    '''
    # Confirm / force input as array
    array = ee.Array(array)
    # slice(0, 1)     - remove first line
    # slice(0, 0, -1) - remove last line
    diff = array.slice(0, 1).subtract(array.slice(0, 0, -1))
    if (d > 1) :
        diff = diffArray(diff, d - 1)
    return diff.transpose()

def whittakerSmoother(coll, plambda, d=1):
    '''
    Smooth image collection using Whittaker's algorithm.

    Parameters
    ----------
    coll:
        Image collection.
    plambda:
        Value for lambda parameter (cannot be named 'lambda' in Python!).
    d:
        Order parameter; the default value is 1.

    Returns
    -------
    Collection with smoothed values (single image;does not retain timestamps!).
    '''
    plambda = ee.Number(plambda)
    m = coll.size().getInfo()
    I = ee.Array.identity(m)
    D =  diffArray(I, d)
    A_arr = (D.matrixMultiply(D.transpose()).multiply(plambda)).add(I)
    A_img = ee.Image(A_arr)
    z_img = A_img.matrixSolve(coll.toArray())
    z_coll = ee.ImageCollection(z_img)
    return z_coll

def medianAbsoluteDeviation(coll, band='NDVI', madBand='MAD', scaleFactor=1.4826):
    '''Compute median absolute deviation image.

    Parameters
    ----------
    coll:
        Image collection.
    band:
        Band name.
    madBand:
        Name for MAD band.
    scaleFactor:
        Scale factor.

    Returns
    -------
    Image for (scaled) MAD.
    '''
    mv = coll.select(band).median()
    def auxProc(img):
        return img.select(band).subtract(mv).abs().rename(madBand)
    return coll.map(auxProc).select(madBand).median().multiply(scaleFactor)

# Areas (commonly used ones)
def roiIberianPeninsula():
    '''
    Obtain Iberian peninsula as a ee.Geometry.Polygon object.

    Returns
    -------
        GEE area corresponding to Iberian Peninsula (ee.Geometry.Polygon instance).
    '''
    return  ee.Geometry.Polygon([[[-10.1513671875, 44.11914151643737],
                                  [-10.1513671875, 35.496456056584165],
                                  [1.669921875, 36.06686213257888],
                                  [4.306640625, 42.90816007196054],
                                  [-1.7138671875, 44.15068115978094]]])
def roiPortugal():
    '''
    Obtains a Bounding box for Portugal as a ee.Geometry.Polygon object.

    Returns
    -------
    GEE area corresponding to Iberian Peninsula (ee.Geometry.Polygon instance).
    '''
    return ee.Geometry.Polygon([[[-9.42626953125, 42.27730877423709],
                                 [-10.01953125, 36.721273880045004],
                                 [-6.943359375, 36.79169061907076],
                                 [-5.86669921875, 42.27730877423709]]])

def poiPortugal(number):
    '''
    Obtains a point of interest for Portugal as a ee.Geometry.Point object.

    Parameters
    ----------
        number:
            index of the point chosen from the list.
    Returns
    -------
    GEE area corresponding to the point chosen (ee.Geometry.Point instance).
    '''
    points = ee.Geometry.MultiPoint([

    [-8.611118,  41.147610], # 0 Urban
    [-8.832963,  41.690937], # 1 Urban
    [-9.137294,  38.709498], # 2 Urban
    [-8.936192,  38.915126], # 3 Agriculture, rice fields
    [-8.799194,  39.080529], # 4 Agriculture, irrigated land
    [-8.660068,  41.473792], # 5 Agriculture, annual crops
    [-8.786169,  41.714940], # 6 Agriculture, annual crops
    [-8.040545,  37.740470], # 7 Non-irrigated arable land
    [-7.869313,  41.155305], # 8 Vineyards
    [-7.786793,  41.182720], # 9 Vineyards
    [-8.645273,  41.481805], # 10 Mixed forest, pine and oak
    [-6.979213,  41.695547], # 11 Coniferous forest
    [-8.849557,  41.724666], # 12 Coniferous forest, maritime pine
    [-9.014289,  39.751502], # 13 Coniferous forest
    [-8.713733,  38.374178], # 14 Coniferous forest, Setúbal
    [-8.284018,  41.818183], # 15 Broadleaf forest, oak
    [-8.151166,  41.789738], # 16 Broadleaf forest
    [-7.917131,  41.885765], # 17 Broadleaf forest
    [-7.061342,  41.449262], # 18 Broadleaf forest
    [-8.821137,  41.748027], # 19 Sparsely vegetated areas, shrubland
    [-8.840177,  41.752333], # 20 Sparsely vegetated areas, shrubland
    [-6.899133,  41.049832], # 21 Transitional woodland-shrubland
    [-7.410011,  38.282250], # 22 Water bodies, Alqueva
    [-8.194189,  41.775663], # 23 Water bodies, Barragem V. Furnas
    [-8.701132,  40.673974], # 24 Salt marshes, Ria de Aveiro
    [-8.632161,  40.692142], # 25 Salt marshes
    [-8.963463,  39.792758], # 26 Incendios 2017 - Pinhal Leiria
    [-8.946331,  39.903403], # 27 Incendios 2017 - Pinhal Leiria
    [-9.015335,  39.738506], # 28 Incendios 2017 - Pinhal Leiria
    [-8.295746,  39.968640], # 29 Incendios 2017 - Pinhal Leiria
    [-8.113004,  40.022892], # 30 Incendios 2017 - Pinhal Leiria
    [-7.914538,  40.172341], # 31 Incendios 2017 - Teixeira
    [-8.108416,  40.288649], # 32 Incendios 2017 - S Martinho da Cortiça
    [-8.179800,  40.189663], # 33 Incendios 2017 - S Martinho da Cortiça
    [-8.179800,  40.189663], # 34 Incendios 2017 - S Martinho da Cortiça
    [-8.846288,  41.716167], # 35 Incendios 2005 - V Castelo
    [-8.847773,  41.732842], # 36 Incendios 2005 - V Castelo
    [-8.829270,  41.713540], # 37 Incendios 2005 - V Castelo
    [-8.802372,  41.722677], # 38 Incendios 2005 - V Castelo
    [-8.802372,  41.722677], # 39 Incendios 2005 - France, Caminha
    [-8.592608,  39.816111], # 40 Incendios 2005 - Leiria/Pombal
    ],'EPSG:4326')
    return points.geometries().get(number)
```

```
# Image processing
def getImageThumbnail(image, roi, bands='NDVI', Min=-2000, Max=10000,
                      palette = 'b3ffb3,001a00'):
    '''
    Get thumbnail URL for given image and ROI.

    Parameters
    ----------
     image:
           Image object
     roi:
           region of interest
     bands:
           comma-separated list of bands, defaults to 'NDVI'
     min:
           min value to consider for plotting, defaults to -2000
     max: int
           max value to consider for plotting, defaults to 10000

    Returns
    -------
    str
    String identifying the GEE thumbnail URL.
    '''
    return image.getThumbUrl({'min': Min,
                              'max': Max,
                              'bands': bands,
                              'region': roi.bounds().getInfo()['coordinates'],
                              'palette': palette})

def exportTiffToDrive(coll, roi, fileName, scale = 250):
    '''
    Export image to Google Drive.

    Parameters
    ----------
     coll:
          Image collection after aplying a reducer.
     roi:
          Region of interest.
     scale:
          Scale of the pixel from the product we are using.
     fileName:
          Name of the file to be exported.

    Returns
    -------
    str
    String identifying the GEE thumbnail URL.
    '''
    task_config = {'description': 'imageToDriveExample',
                   'scale': scale,
                   'region': roi.bounds().getInfo()['coordinates']}
    task = ee.batch.Export.image(coll, fileName, task_config)
    task.start()
    return task

def exportVideoToDrive(coll, roi, description='GEE video', dimensions=600):
    '''
    Export video for an image collection to Google Drive.

    It is assumed that the collection defines a single band that is converted
    by this procedure to three RGB bands using ee.Image.visualize().

    Parameters
    ----------
     coll:
          Image collection.
     roi:
          Region of interest.
     description:
          Description (ID) for video.
     dimensions:
          Dimensions in pixel (video will have a resolution of dimensions x
          dimensions pixels).

    Returns
    -------
    str
    String identifying the GEE thumbnail URL.
    '''
    def auxProc(image):
        return image.visualize(min=-2000,
                               max=10000,
                               forceRgbOutput=True)
    vcoll = coll.map(auxProc)
    task = ee.batch.Export.video.toDrive(collection=vcoll,
                                         description=description,
                                         dimensions=dimensions,
                                         framesPerSecond=1,
                                         region=roi.bounds().getInfo()['coordinates'])
    task.start()
    return task
def smoothedMASCollectionsToDF(collection1, collection2):
    '''
    Merge two collections from the movingAverageSmoother function into a single dataframe

    Parameters
    ----------
     collection1 and collection2:
        Image collection of the movingAverageSmoother function

    Returns
    -------
    data frame
    '''
    coll1 = np.array(collection1[1:])
    coll2 = np.array(collection2[1:])
    df_m = pd.DataFrame(coll1)
    df_m.columns = ['id', 'longitude', 'latitude', 'time', 'NDVI', 'MAS']
    df_m.id = df_m.id.str.replace('_', '-')
    df_m['WS'] = coll2[:,5]
    return df_m
```

```
def smoothedWHITCollectionsToDF(collection1, collection2, df_m):
    '''
    Merge two collections from the whittakerSmoother into a single dataframe

    Parameters
    ----------
     collection1 and collection2:
        Image collection of the whittakerSmoother function
     df_m:
        Data frame of the smoothedMASCollectionsToDF function
    Returns
    -------
    data frame
    '''
    coll1_w = np.array(collection1[1][4])
    coll2_w = np.array(collection2[1][4])
    df_w = df_m.loc[:,('id', 'longitude', 'latitude', 'time', 'NDVI')]
    df_w['MAS'] = coll1_w
    df_w['WS'] = coll2_w
    return df_w

def exportPlot(df, fileName, typeFile, fileNamepng, startDate, endDate,
               plotTitle = "Whittaker vs. Moving average", name1 = "Raw data",
               name2 = "Moving average smoothing",name3 = "Whittaker smoothing",
               xtitle = "Date", ytitle = "NDVI", color1 = '#2ca25f',
               color2 = '#e66101', color3 = '#5e3c99'):
    '''
    Contruct plot of NDVI values of raw data, and with two smoothing algorithms.

    Parameters
    ----------
     df:
         Data frame with the time series to plot
     fileName: string
         Name of the output file
     typeFile:
         Name of the extension of the output file
     startDate:
        Start date (YYYY-MM-DD format).
     endDate:
        End date (YYYY-MM-DD format).
     plotTitle:
         A string with the plot title
     xtitle and y-title:
        A string with the x-axis and y-axis name
     name1, name2, name3:
        A string with the name of the time series
     color1, color2, color3:
         A string with the hexadecimal color of the time series.

    Returns
    -------
    Saves an html file of the plot
    '''
    trace_raw = go.Scatter(
                x = df.id,
                y = df.NDVI,
                name = name1,
                line = dict(color = color1),
                mode = 'markers',
                opacity = 0.7)
    trace_mas = go.Scatter(
                x = df.id,
                y = df.MAS,
                name = name2,
                line = dict(color = color2),
                opacity = 1)
    trace_ws = go.Scatter(
                x = df.id,
                y = df.WS,
                name = name3,
                line = dict(color = color3),
                opacity = 1)
    data1 = [trace_raw,trace_mas,trace_ws]
    layout = dict(title = plotTitle, titlefont = dict(size=32),
                  xaxis=dict(title=xtitle,
                             titlefont=dict(size=30),
                             range = [startDate, endDate],
                             tickfont=dict(size=27)),
                  yaxis=dict(title=ytitle,
                             titlefont=dict(size=30),
                             tickfont=dict(size=27)),
                  legend=dict(font=dict(size=27)),
                  margin = dict(l = 150))
    fig = dict(data=data1, layout=layout)
    # Make plot
    py.plot(fig, filename = fileName + '.html', image = typeFile,
            image_filename = fileNamepng, image_height = 750, image_width = 1200)


def wAux(m, plambda):
    I = np.mat(np.identity(m))
    D = np.diff(I)
    A = I + plambda * ( D * np.transpose(D))
    return A

def whittakerSmoothNumpy(y_np, plambda):
    m = len(y_np)
    A_np = wAux(m, plambda)
    z_np = np.linalg.solve(A_np, y_np)
    return z_np
```