

# Hands-on exercises

## Introduction and requirements

---

The purpose of these exercises is to learn the student the basic usage of the available remote sensing cloud platforms to understand, analyse and process SAR-satellites time series over forest-dominated landscapes. Specifically, we aim to produce small-scale mapping workflows which can be easily scaled to regional or even country-level mapping.

Initially we will review the main characteristics of the platform of choice for this training (Google Earth Engine, GEE). Then we will introduce the students to GEE's programming basics through a series of exercises of increasing difficulty.

After that we will use a GEE's application to explore the SAR response of a variety of environments, and we will extract time series of areas affected by natural or man-made hazards.

After this introductory block we explore the possibilities offered by the GEE platform to perform custom analysis of time-series to produce quick maps of changes in the canopy.

We will examine the results and correct them until having a satisfactory image that we will export to our preferred GIS package.

The students are expected to have basics skill on programming and the use of GIS software. As most of the training will use cloud resources, no special configuration is required for the computers to be used. Nevertheless, a good internet connection is mandatory.

## Preparation

---

Every student should have an active Google Earth Engine account.

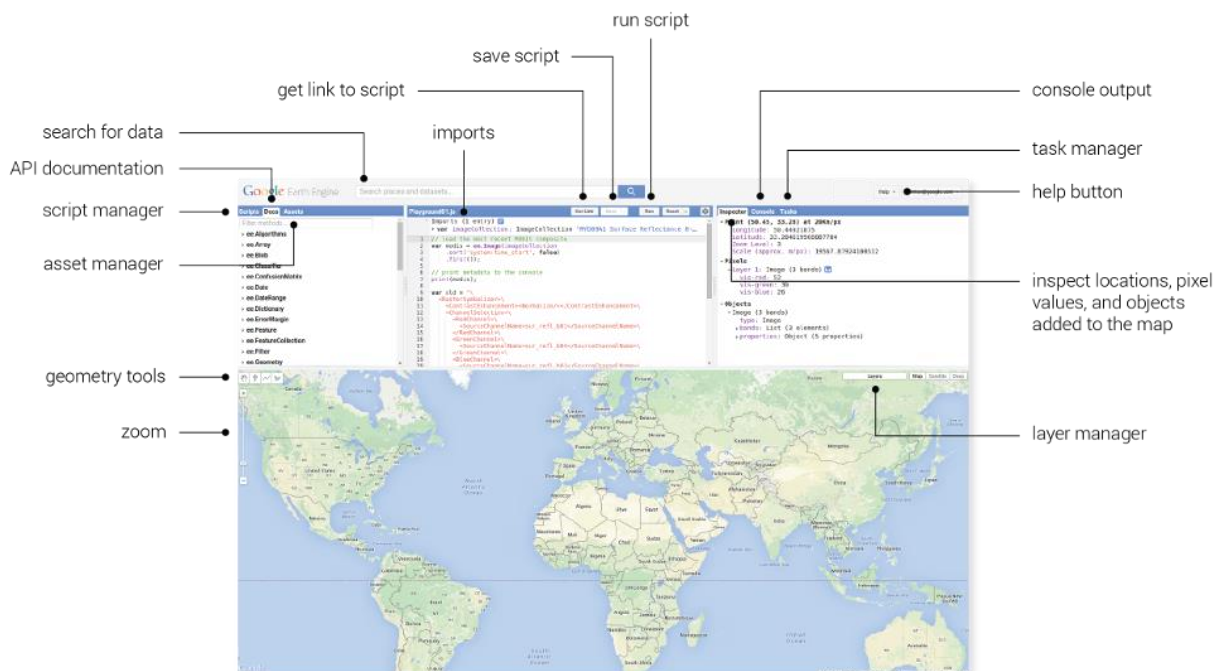
If you don't have a GEE account please go to <https://code.earthengine.google.com/register>, it will take a few minutes to have an email account (which does not need to be gmail) to be authorized.

# 1. Introduction to the Google Earth Engine platform

Google Earth Engine (GEE) is an advanced cloud-based platform designed for environmental data analysis and geospatial processing. It leverages massive computing power to enable scientists, researchers, and developers to run data analysis on a global scale, utilizing a multi-petabyte catalogue of satellite imagery and geospatial datasets. The platform facilitates the detection of changes, mapping of trends, and quantification of differences on the Earth's surface, making it a powerful tool for understanding and addressing environmental challenges such as climate change, deforestation, and water management.

One of the key features of Google Earth Engine is its extensive data archive, which includes historical satellite imagery going back over three decades, as well as real-time data from recent observations. This comprehensive data collection allows users to perform temporal analyses and monitor changes over time across the globe. The platform supports a wide range of datasets, including weather and climate models, elevation data, and land cover maps, making it highly versatile for various applications in environmental science, agriculture, and disaster response.

During this workshop we will use GEE's **Code Editor**, which is an IDE (Interactive Development Environment) made for ease the interaction with GEE routines and resources. Please take a look at the main components of GEE's Code Editor:



We will briefly describe every part:

- 1) Editor Panel: The Editor Panel is where you write and edit your Javascript code. Note that the Run button executes the code.
- 2) Right Panel, composed of:
  - a) Console tab for printing output
  - b) Inspector tab for querying map results (Click on the map and note that *there is a scale in meters associated with the zoom level.*)
  - c) Tasks tab for managing long-running tasks.
- 3) Left Panel, composed of:
  - a) Scripts tab for managing your programming scripts (These are git repos hosted at Google)

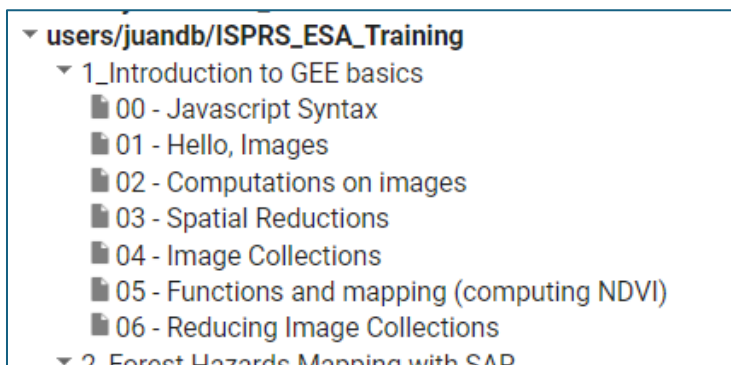
- b) Docs tab for accessing documentation of Earth Engine objects and methods, as well as a few specific to the Code Editor application. This is the definitive API reference and is populated by the server.
  - c) Assets tab for managing assets that you upload. You get 250 GB free.
  - d) Interactive Map: For visualizing map layer output (Note layer controls and the geometry tools)
- 4) Search Bar, used for finding datasets and places of interest.
  - 5) Get Link button: A static snapshot of the Code Editor at the time the button is clicked is produced and placed in the address box of your browser. If you change the code, get a new link. You can email these around for easy collaboration.
  - 6) Help Menu: Gives access to the following options:
    - a) User guide - reference documentation
    - b) Help forum - Google group for discussing Earth Engine
    - c) Shortcuts - Keyboard shortcuts for the Code Editor
    - d) Feature Tour - overview of the Code Editor
    - e) Feedback - for sending feedback on the Code Editor
    - f) Suggest a dataset.

Next, we will perform a walkthrough the main features of google earth engine, using a set of exercises built with the purpose of ease the understanding of GEE's particularities and capabilities.

To start working on those exercises, please go to this url:

[https://code.earthengine.google.com/?accept\\_repo=users/juandb/ISPRS\\_ESA\\_Training](https://code.earthengine.google.com/?accept_repo=users/juandb/ISPRS_ESA_Training)

If everything goes well, you will have access to the complete set of scripts which we will be using during this workshop. The scripts should appear under the 'Writer' folder of the scripts tab:



Please follow your trainer as it guides you through the 'Introduction to GEE basics' set of scripts.

If you want to deepen your knowledge on GEE, there's a wealth of resources in the web. We suggest beginning with the GEE's user guide, a comprehensive guide to every aspect of the platform:

<https://developers.google.com/earth-engine/guides>

## 2- SAR time series using SAR-Watcher

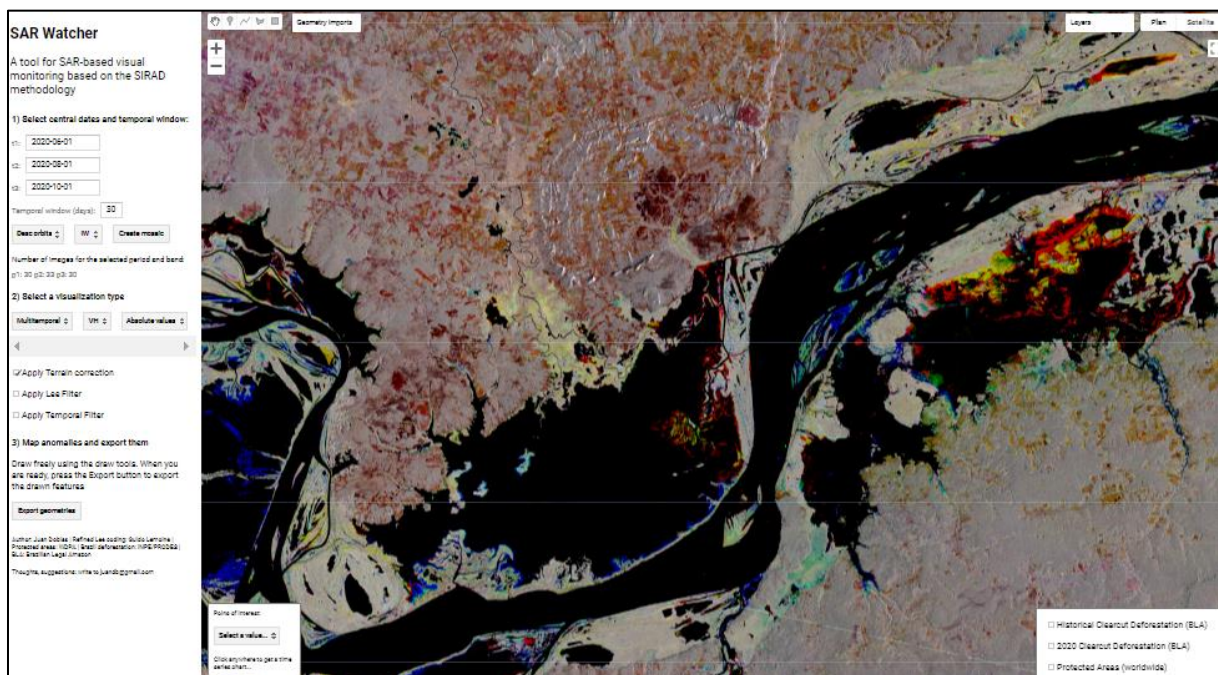
Today, we will use a web app call **SAR Watcher**, which uses the Google Earth Engine platform, to explore and recognize the different types of SAR response on a variety of environments.

We will use this app to check the dynamics of the SAR backscattering, by looking at the Sentinel-1 time series on areas subject to change, such as flooding, deforestation, tidal oscillations and other man-made or natural phenomena.

To access SAR-Watcher, please open this link:

<https://juandb.users.earthengine.app/view/sar-watcher>

You should see something like that:

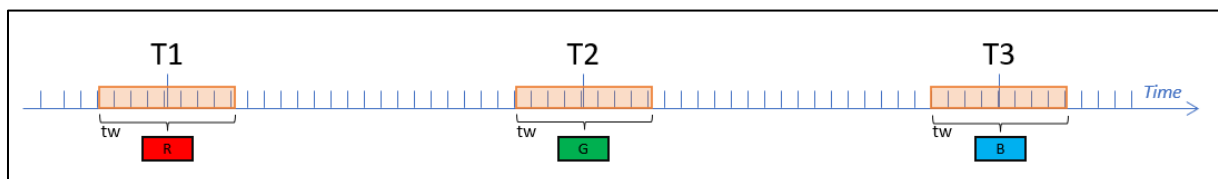


As you can see, the main window of the app shows a raster image. This image is not a usual, optical image, but a *multitemporal SAR mosaic*. We will explain this concept now.

### 1. Multitemporal mosaics: an explainer










The image that you are seeing is the result of the composition of three channels: red (R), green (G) and blue (B), like every raster image that you are used to see and work with. The main difference here is that, instead on placing an optical image `_band_` in each RGB channel, we will place a Sentinel-1 backscattering image corresponding to a certain **time window**.

Please take a look into this figure:



In this figure, every Sentinel-1 available image is represented by a blue tick in the **time** axis. As you can see in the figure, by defining **T1**, **T2** and **T3** and a time window **tw**, you are defining which images will be averaged to compose every one of the color channels (R, G and B).

As this, this kind of image will show the **dynamics** of the radar backscattering during the defined time period. Every color represents a different type of backscatter evolution through time, as showed by the next figure:

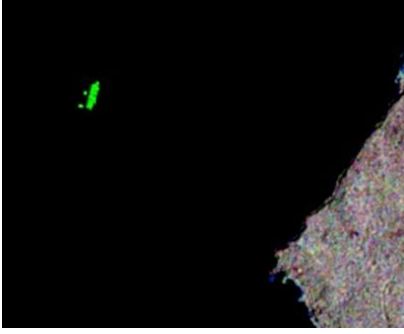
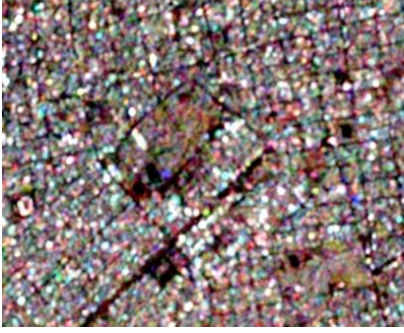

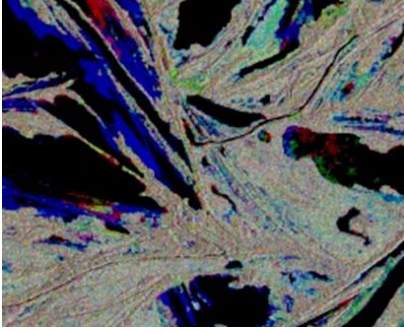
| <u>Color</u> | <u>Sequence</u>   | <u>Interpretation</u>           |
|--------------|---|---------------------------------|
| Blue         |  | Rise in T3                      |
| Green        |  | Rise in T2, <u>decrease T3</u>  |
| Red          |  | <u>Decrease T2</u>              |
| Yellow       |  | <u>Decrease T3</u>              |
| Magenta      |  | <u>Decrease T2, increase T3</u> |
| Cyan         |  | <u>Increase T2</u>              |
| Black        |  | No signal                       |
| White        |  | No variation                    |
| Grey         |  | No variation                    |




## 2. Use SAR-Watcher to recognize different types of reflections

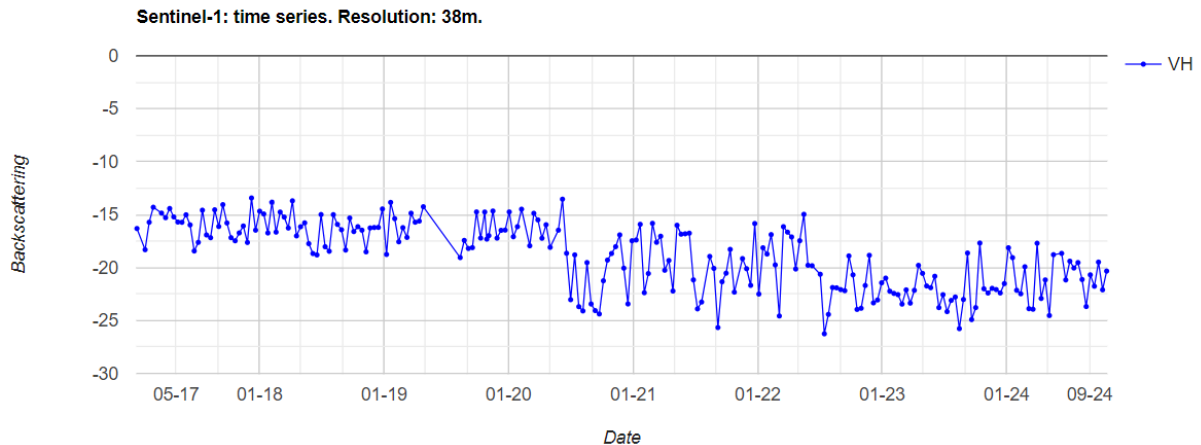
Please use the Sar Watcher application to look for different kinds of land cover and investigate the type of backscattering response associated with them.

We will look for different kinds of land cover, such as:

| Land Cover      | Example   | Question                      |
|-----------------|---|-------------------------------|
| Water           |    | What's that green thing?      |
| Urban           |   | Is there a park in this city? |
| Forest          |  | What's that yellow shape?     |
| Seasonal Floods |  | How do we know is seasonal?   |

### 3. Time series browsing

If you want to investigate with more detail the dynamics of a certain region you can just click on SAR-Watcher in and wait for a chart to appear. By clicking in the  button, you can pop-up the time series chart to better visualize it:



You can also export the corresponding values by clicking on the 'Download CSV' button.

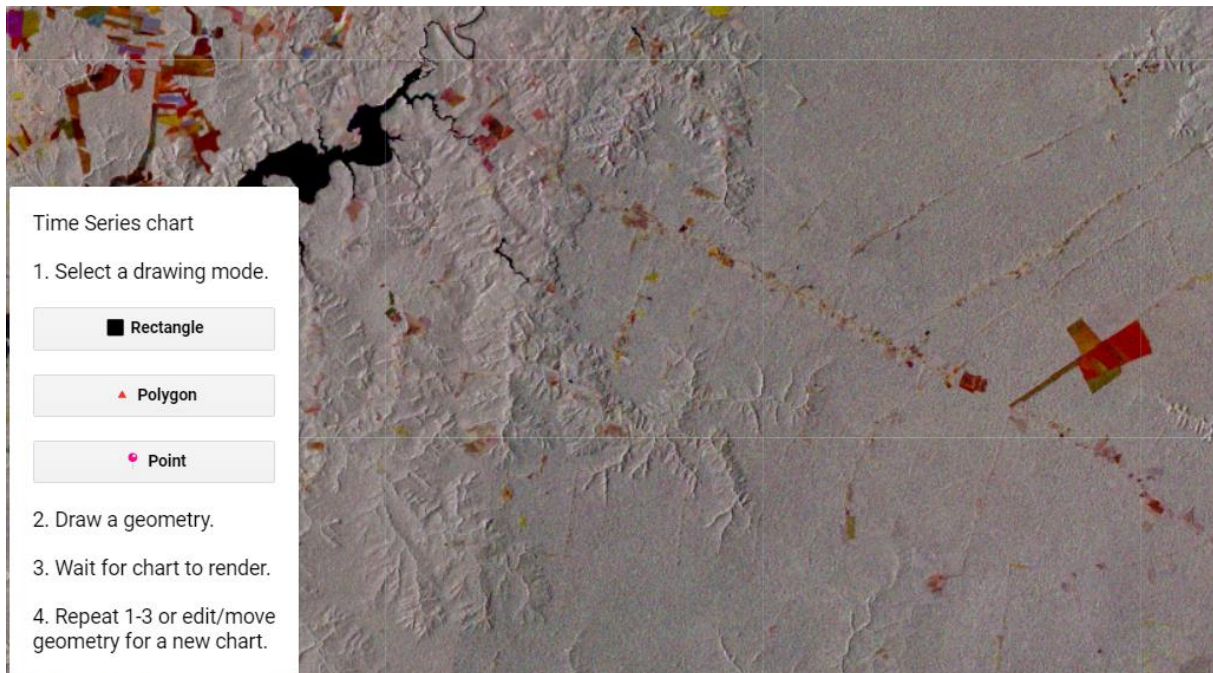
### 4. Explore differences on time series between antropized areas and forest areas in C band.

There is an advanced version of SAR-Watcher that allows to get the mean time series inside an user-specified area, and even to you can give it a try: <https://juandb.users.earthengine.app/view/sar-ts-watcher-mb>.

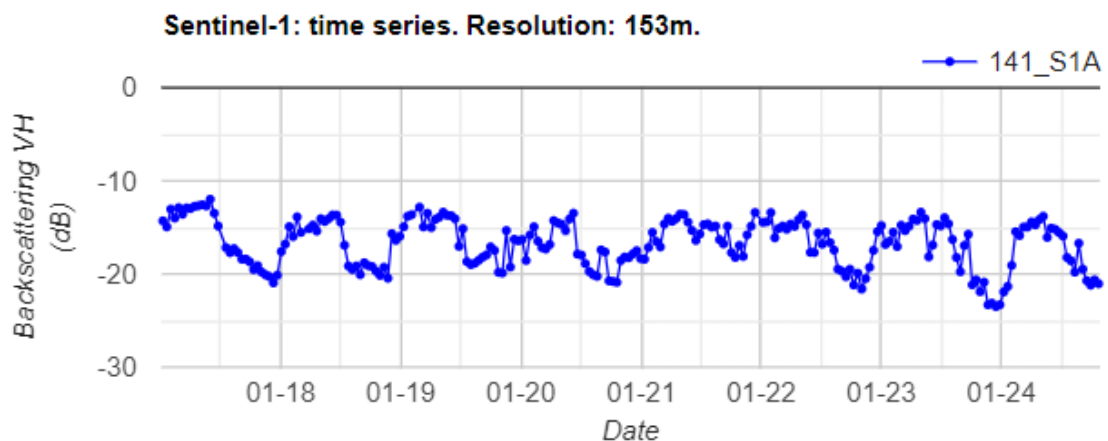
This application allows one to draw a geometry in the multitemporal mosaic and to obtain the time series corresponding to the mean value of the C-band SAR backscattering, all through the selected time span.

In order to obtain these data, please follow the instructions below:

- 1) Open the SAR-Watcher mean time series and focus in an area of your interests. We suggest to start going to an area SE of the city of Santarem, as shown in the figure below (use [this link](#) if you cannot find it):



- 2) Draw an area over a forest area using the tools provided on the box in the left side of the map.
- 3) Wait until a chart with the corresponding time series is drawn.
- 4) Compare the results with the time series of an antropized area (it should look more or less like that:



Please look for forest areas suffering different impacts, examine and explain the differences between the corresponding responses.

Some places to go (click on the link to directly go to each area):

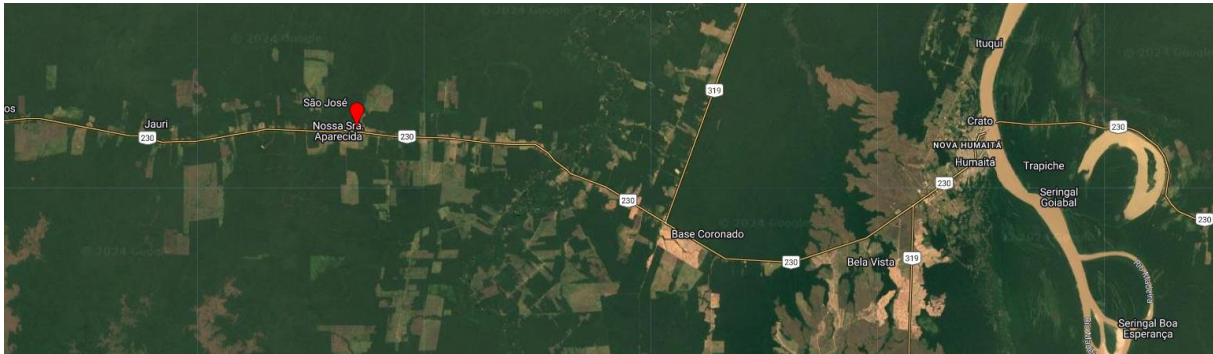
- [Recently deforested area](#) (can you determine the date of deforestation?)
- [Selective logging area](#)
- [Burned area](#)
- [Illegal mining area](#)



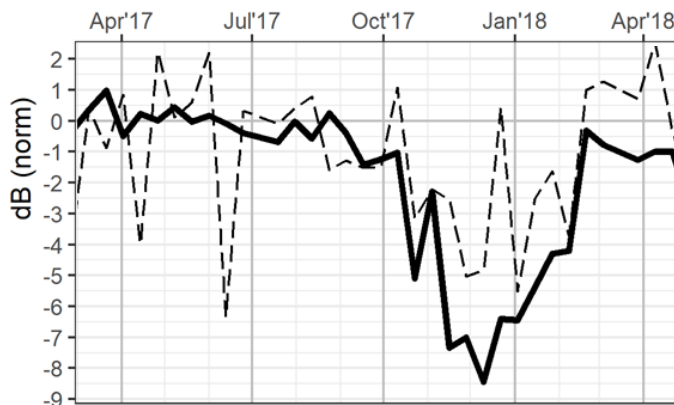
## 3- Automatic deforestation detection using SAR

As a practical exercise on SAR automatic processing, we will be using GEE to get a map of deforestation on an interest area.

We will be working on a frontier area in the Amazon, near the city of Humaitá (Brazil), which has seen a huge increase of deforestation linked to the recent announcement of the paving of a major road between the SW of the amazon and the megacity of Manaus (BR-319):



The main hypothesis behind our job today is that deforestation causes a drop in backscattering. Although not completely correct, this assumption drives all the deforestation detection systems based of SAR. One must know that, if very frequent, this drop in backscattering is not permanent. After a period of some months, as the new cover (pasture, crops) grows, the backscattering signal will recover a level close to the original one. See this example taken from Doblas, 2022:



We will be using the scripts which are in the *2\_Forest Hazards Mapping with SAR* folder (you can have access to the repository by clicking here:

[https://code.earthengine.google.com/?accept\\_repo=users/juandb/ISPRS\\_ESA\\_Training](https://code.earthengine.google.com/?accept_repo=users/juandb/ISPRS_ESA_Training)). They are organized in a growing level of difficulty, from the basic loading and displaying of one single SAR image to complete deforestation mapping.

Please follow your instructor through the exercises associated with every script:

## 1. Load S1 images

This script will show you the basic displaying of Sentinel-1 images, and how to know the main attributes (metadata) of the collection, such as the number of images, the orbits, and the orbits' mode.

## 2. Compute mean value of collection

This script will teach you the first hard lesson of a SAR specialist: you should NEVER use images in decibel scale to average your images.

## 3. Convert time series to Gamma Naught

As you may have learned these days, the magnitude of the SAR backscatter depends on the angle of incidence of the radar wave on the terrain. The higher the angle, the lower the backscattering. One simple way to compensate for this distortion is to divide original backscattering (sigma-naught) by the cosines of the incidence angle, following this equation:

$$\gamma^0 = \frac{\sigma^0}{\cos \theta}$$

This script will guide you through this process, showing the differences between the original and the corrected images.

## 4. Speckle filtering

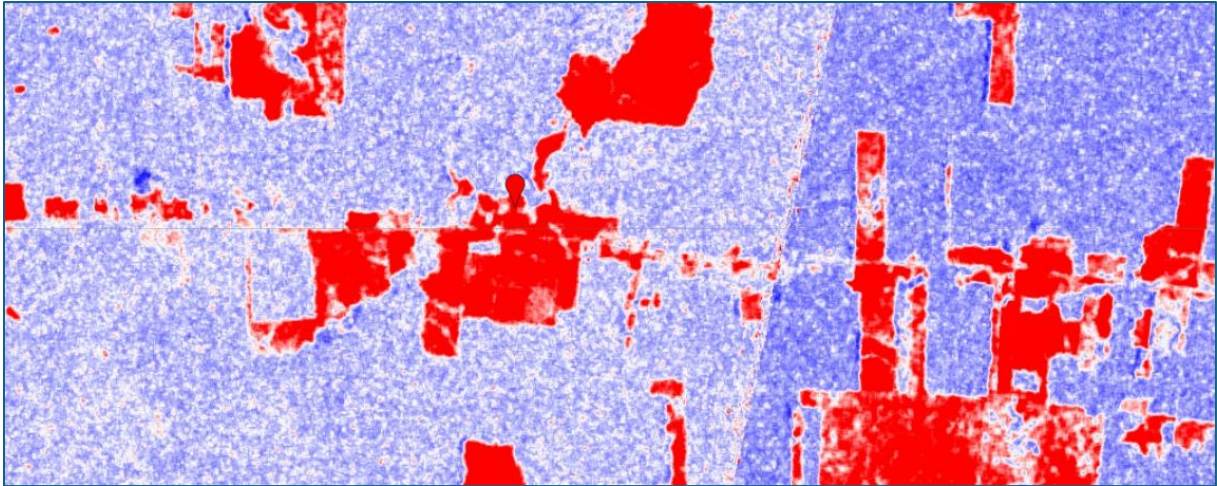
Speckle is a phenomenon inherent to radar images. Although it may contain interesting information, generally SAR users try to remove it in order to reduce the noise associated with their workflows. Speckle reduction is an important discipline in imaging sciences, and many different filters are proposed. Here we will use maybe the simplest of them, a box-car filter. This filter is very effective in reducing noise but can hamper the definition of features on our images. A good balance between noise reduction and image definition should be made by the user.

## 5. First map of change

Now we will finally build our first deforestation map. We will be applying the following procedure:

1. Image selection: We will select all the Sentinel-1 images over the AOI
2. Preprocess data:
  - Convert to gamma nought
  - Speckle filtering: we will apply simple filter (boxcar) or a more complex one (Quegan-Yu temporal filter).
3. Compute background:
  - Select the images acquired **before** the period we are investigating.
  - Compute the mean value of those images and convert it to decibels.
4. Compute backscattering level of the period of interest
  - Select the images acquired **during** the period we are investigating.
  - Compute the mean value of those images and convert it to decibels.
5. Compute the change ratio (difference between mean images in dB)

If everything works correctly, you should get an image like this one:



## 6. Map of deforestation

By applying the hypothesis explained in the introduction of this section, we will look for a threshold value that will retain the areas suffering a significant negative change. These areas were potentially deforested during the period we are investigating.

Please carefully examine the change layer, by using the ‘inspect’ tool, to determine an optimal threshold value which will separate changed from non-changed areas.

Additionally, we will be using an ancillary forest mask (JRC’s 2020 forest type layer) in order to avoid detecting changes in non-forest areas such as pastures, urban areas or water-bodies. Other forest masks, such as INPE’s (<https://terrabrasilis.dpi.inpe.br/en/home-page/>) or Hansen’s yearly forest loss (available in GEE).

## 7. Export results

Finally, we will export the results of our procedure as a raster binary layer, and as polygons. Please feel free to compare your results with Planet basemaps or other ancillary layer of information

## Bonus: Alternatives to Google Earth Mapping

---

Although incredibly powerful, Google Earth Engine has some limitations. Its source code is not open, converting the platform in a kind of ultra performant 'black box'. Also, it is not free and can become very expensive for those users switching from the academy to private enterprises.

In this section we will be talking about the main protocols, libraries and techniques used to process SAR data in an efficient, cloud-based manner, as an alternative to the GEE platform.

**PANGEO:** PANGEO is a community-driven project that provides a platform integrating data, computing, and scientific research to enable big data geoscience research. The initiative leverages open-source tools such as Xarray, Dask, and Jupyter, facilitating the analysis and visualization of large multidimensional datasets. PANGEO aims to promote collaboration among scientists, researchers, and technologists by providing an accessible, scalable, and sustainable environment for data-intensive earth science research, making it easier to work with complex and voluminous datasets typically used in climate and oceanography studies.

**STAC (SpatioTemporal Asset Catalog):** STAC is an open specification designed to standardize the way geospatial data is indexed and discovered. It aims to make geospatial information more accessible and interoperable by providing a common language for describing a wide range of geospatial assets. By organizing data into catalogs, collections, and items, STAC facilitates the integration and sharing of geospatial datasets across different platforms and applications. This promotes easier access to satellite imagery, elevation data, and other spatial resources, benefiting various applications in earth observation, environmental monitoring, and beyond.

**Xarray:** Xarray is an open-source project and Python library designed to bring the labeled data power of pandas to the physical sciences by providing N-dimensional variants of core pandas data structures. It facilitates the handling, analysis, and visualization of multidimensional datasets, especially those used in meteorology, oceanography, and climate science. Xarray integrates tightly with other scientific computing libraries such as Dask for parallel computing and Matplotlib for plotting, making it a pivotal tool in the analysis of large and complex datasets.

**Dask:** Dask is an open-source Python library that provides advanced parallel computing capabilities, designed to scale Python analytics to large datasets and compute clusters. It enables the performance of computations on big data using blocked algorithms and task scheduling, without needing to fit the data into memory. Dask integrates seamlessly with existing Python data analytics tools like NumPy, Pandas, and Scikit-Learn, allowing users to scale their data processing workflows efficiently and flexibly, from laptops to large clusters.

**Copernicus Data Space:** The Copernicus Data Space is part of the European Union's Copernicus Earth observation program, which provides a comprehensive range of Earth observation data and services. The Copernicus Data Space offers a vast repository of environmental data collected from satellite observations, in-situ sensors, and other sources. Additionally, there are a number of resources freely offered which allows any user to deploy Jupyter Notebooks and run APIs to recover and process Copernicus Data on a cloud-based basis.