

# Monitoring Gold Mining Activity Using SAR

## (A1.8)

---

### Authors

Lucio Villa, Sidney Novoa, Milagros Becerra, Andréa Puzzi Nicolau, Karen Dyson, Karis Tenneson, John Dilger

---

### Overview

The expansion of gold mining has had a large impact on the rainforests of the Amazon over the last decades. To take just one example, it has affected both the biodiversity and the lives of local people in the Madre de Dios region of southeastern Peru. In this chapter, we will review a methodology developed to generate early warnings of deforestation based on the use of synthetic aperture radar (SAR) images. First, we will identify the Sentinel-1 images suitable for the construction of a time series of preprocessed datasets. Second, we will run a change detection analysis based on a statistical analysis of the Sentinel-1 images (Canty et al. 2019). Finally, we will show the steps to follow in the post-processing stage by filtering information with forest / non-forest and bodies of water datasets.

### Learning Outcomes

- Selecting and creating a multitemporal SAR mosaic.
- Generating SAR change detection based on a statistical analysis of Sentinel-1 images.
- Post-processing of alerts generated by filtering maximum area patches and information of forest/non-forest and water bodies.

### Helps if you know how to:

- Import, filter, and visualize images (Part F1).
- Work with time-series data in Earth Engine (Part F4).
- Write a function and `map` it over an `ImageCollection` (Chap. F4.0).
- Use the `require` function to load code from existing modules (Chap. F6.1).
- Export results to assets (Chap. F6.2).

## **Introduction to Theory**

Accelerated demand for natural resources has transformed the Amazon rainforest into a new economic frontier that generates commodities such as agricultural products, livestock, and more recently, minerals, especially gold (RAISG 2020). In Peru, illegal gold mining is a serious problem that affects local populations in the southeastern region of Madre de Dios (Yard et al. 2012, Asner and Tupayachi 2017, Alvarez-Berrios et al. 2021). According to Caballero et al. (2018), it has led to the deforestation of about 1,000 km<sup>2</sup> of rainforest, affecting protected areas, Indigenous communities, and sustainable management areas. Gold mining is carried out throughout the year, even during the rainy season.

Optical Earth-observation satellites have played a very important role in monitoring gold-mining deforestation in recent years. Madre de Dios is one of the few tropical regions in the world for which there is well-documented information on annual forest loss due to this deforestation driver (Asner et al. 2013, Asner and Tupayachi 2017, Caballero et al. 2018, Nicolau et al. 2019, Csillick and Asner 2020, Alarcón Aguirre et al. 2021). However, these resources do not reveal the progress of illegal mining during the rainy season and at other times when the optical images are obscured by clouds. This reduces the window of opportunity to know the dynamics of the activity and guide actions to control the advance of deforestation.

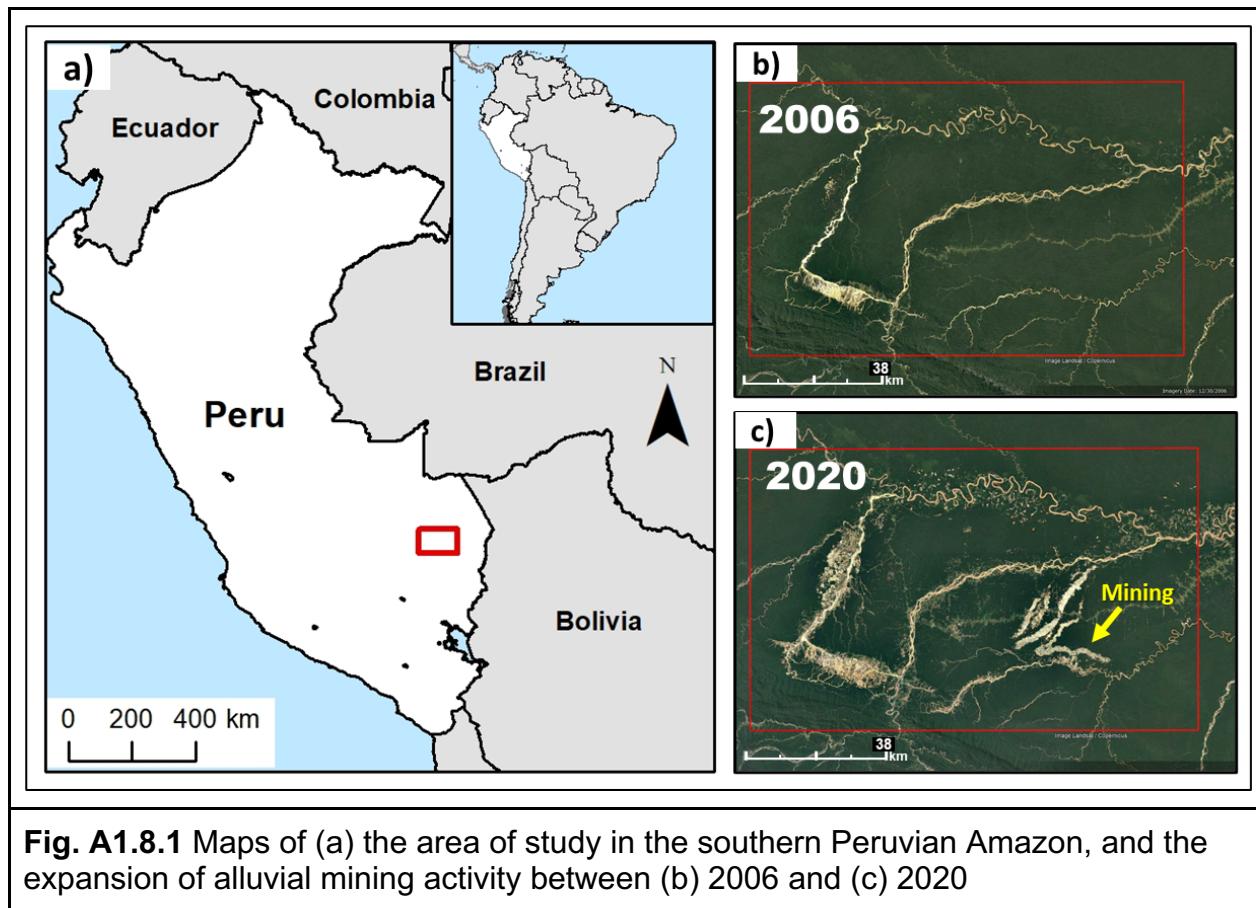
Satellite-based SAR sensors obtain information throughout the entire year thanks to the ability of microwaves to penetrate cloud cover (Ballere et al. 2021, Nicolau et al. 2021). Therefore, we are able to capture changes due to gold mining expansion with SAR data from the European Space Agency (ESA) Sentinel-1 C-band satellite. In this chapter, we will look at a successful example of how to process radar images to generate early warnings of gold-mining deforestation.

## **Practicum**

SAR sensors transmit microwave signals at an oblique angle and measure the backscattered portion of the signal in order to analyze features on the surface. Unlike optical sensors, which are passive, SAR is an active instrument with its own source of illumination, and it is one of the few sensing instruments that allows full control of the signal polarization on both the transmit and receive paths. The majority of today's SAR sensors are linearly polarized, and transmit horizontally and/or vertically polarized wave forms (i.e., SAR bands can be VV, VH, HV, or HH). While interpreting optical imagery is similar to interpreting a photograph, interpreting SAR data requires a different way of thinking, in that the signal is instead responsive in complex ways to surface

characteristics such as structure and moisture. More information about the theory and concepts behind SAR is available in the SAR Handbook (Flores-Anderson et al. 2019).

The area of study for this chapter is the region surrounding a mining corridor located in Madre de Dios in the southern Peruvian Amazon (Fig. A1.8.1).



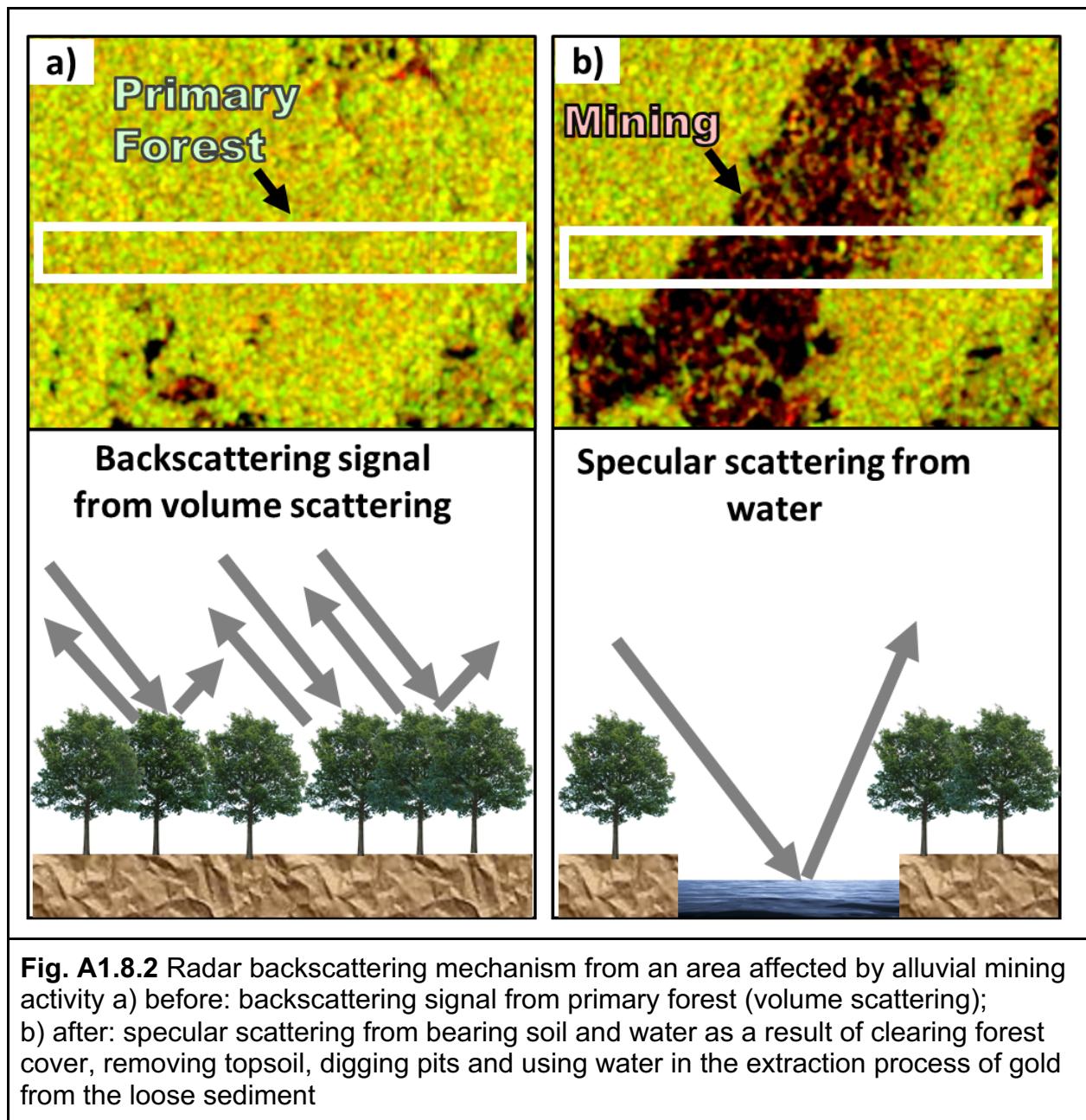
**Fig. A1.8.1** Maps of (a) the area of study in the southern Peruvian Amazon, and the expansion of alluvial mining activity between (b) 2006 and (c) 2020

Alluvial gold mining exploitation generates changes in the radar backscattering mechanism over forested areas, since it involves deforestation, removal of topsoil, excavation, and the use of water for the extraction of gold from the loose sediment. Fig. A1.8.2 shows the changes in SAR Sentinel-1 images and a radar backscattering mechanism before (Fig. A1.8.2a) and after (Fig. A1.8.2b) the impact of alluvial gold mining activity.

Therefore, you will learn how to use SAR Sentinel-1 time series to detect changes generated by alluvial gold mining activities in forested areas.

The first step is to create a SAR mosaic for a given period of time and orbit pass. Then, we will apply the omnibus Q-test algorithm to generate change alerts from the SAR

Sentinel-1 time series. Finally, we will filter and eliminate potential false positive alerts coming from other activities with the same temporal pattern as the mining activity (e.g., natural forest loss by river expansion or water over bare soil during the rainy season).



### Section 1. Creating a Single SAR Mosaic

If you have not already done so, you can add the book's code repository to the Code Editor by entering

[https://code.earthengine.google.com/?accept\\_repo=projects/gee-edu/book](https://code.earthengine.google.com/?accept_repo=projects/gee-edu/book) (or the short URL [bit.ly/EEFA-repo](https://bit.ly/EEFA-repo)) into your browser. The book's scripts will then be available in the script manager panel to view, run, or modify. If you have trouble finding the repo, you can visit [bit.ly/EEFA-repo-help](https://bit.ly/EEFA-repo-help) for help.

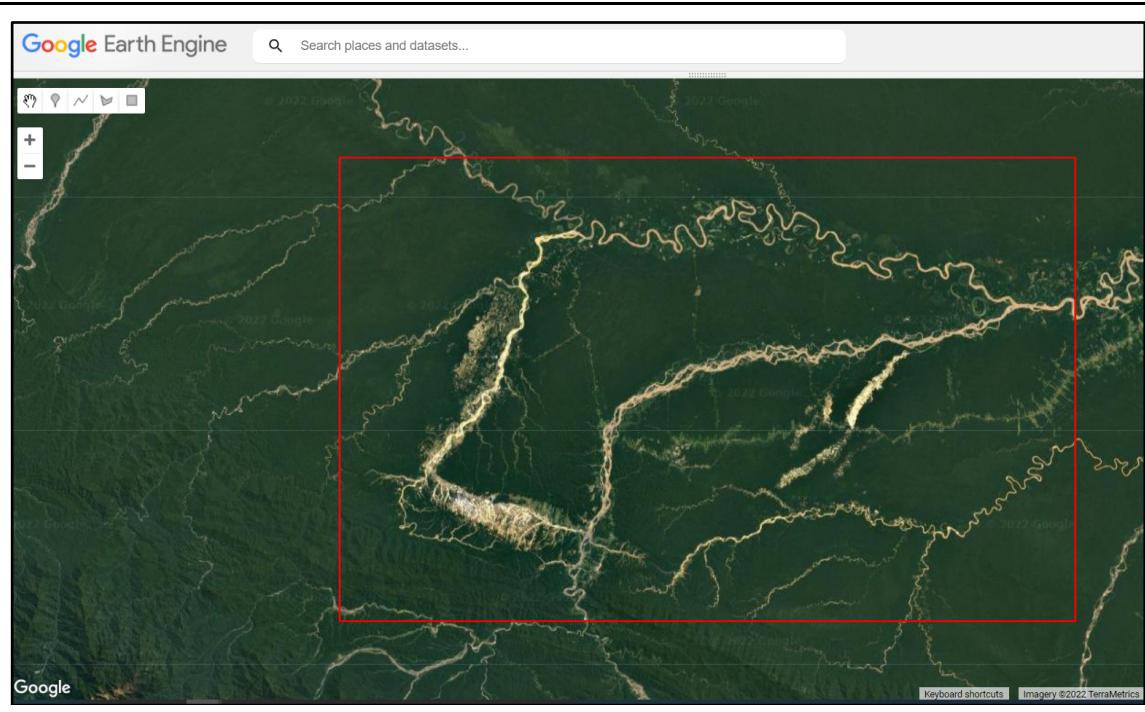
We will use ESA's Sentinel-1 dual-polarized (VV+VH) in descending orbit pass to create a single SAR mosaic over the area of study. Sentinel-1 SAR Ground Range Detected (GRD) data (Fig. A1.8.3) is stored in Earth Engine in two formats: logarithmic scale (dB) and the original values (power scale named as FLOAT). We will use the latter since mathematical operations should not be applied into data on a logarithmic scale. The Sentinel-1 dataset is composed of Level-1 SAR amplitude multi-look images preprocessed according to the following steps: orbit metadata update, removal of border and thermal noise, radiometric calibration and terrain correction.

The screenshot shows the Earth Engine data catalog interface. At the top, there is a navigation bar with links to Home, View all datasets, Browse by tags, Landsat, MODIS, Sentinel, and API Docs. Below the navigation bar, the title "Sentinel-1 SAR GRD: C-band Synthetic Aperture Radar Ground Range Detected, log scaling" is displayed, along with a small thumbnail image of the radar imagery. To the right of the title, there is a "Dataset Availability" section showing the time range from 2014-10-03T00:00:00Z to 2021-10-13T00:00:00. Below that is a "Dataset Provider" section indicating "European Union/ESA/Copernicus". Underneath these sections is an "Earth Engine Snippet" containing the code "ee.ImageCollection('COPERNICUS/S1\_GRD')". Further down is a "Tags" section with several tags: backscattering, copernicus, esa, eu, polarization, radar, sar, and sentinel. At the bottom of the main content area, there are links for Description, Bands, Image Properties, and Terms of Use. A detailed description of the dataset follows, stating that the Sentinel-1 mission provides data from a dual-polarization C-band Synthetic Aperture Radar (SAR) instrument at 5.405GHz (C band). This collection includes the S1 Ground Range Detected (GRD) scenes, processed using the Sentinel-1 Toolbox to generate a calibrated, ortho-corrected product. The collection is updated daily. New assets are ingested within two days after they become available. Below this description is a caption "Fig. A1.8.3 Sentinel-1 imagery available in the Earth Engine data catalog".

Copy and paste the code below to define the area of study (Fig. A1.8.1), convert this vector to a boundary image, and add it to the map (Fig. A1.8.4).

```
//////////  
/// Section One  
//////////  
  
// Define the area of study.  
var aoi = ee.FeatureCollection('projects/gee-book/assets/A1-8/mdd');
```

```
// Center the map at the aoi.  
Map.centerObject(aoi, 9);  
  
// Create an empty image.  
var empty = ee.Image().byte();  
  
// Convert the area of study to an EE image object  
// so we can visualize only the boundary.  
var aoiOutline = empty.paint({  
    featureCollection: aoi,  
    color: 1,  
    width: 2  
});  
  
// Select the satellite basemap view.  
Map.setOptions('SATELLITE');  
  
// Add the area of study boundary to the map.  
Map.addLayer(aoiOutline, {  
    palette: 'red'  
}, 'Area of Study');
```



**Fig. A1.8.4** The area of study around the mining corridor located in the Madre de Dios region

Next, copy and paste the code below to define two functions, `maskAngle` and `getCollection`. The first function masks sections of SAR images acquired at an incidence angle less than 31 or greater than 45 degrees. The second function filters the Sentinel-1 imagery to a specific period of time, region of interest, and orbit pass. Note that the Sentinel-1 GRD dataset is imported inside the second function.

```
// Function to mask the SAR images acquired with an incidence angle
// lower or equal to 31 and greater or equal to 45 degrees.
function maskAngle(image) {
  var angleMask = image.select('angle');
  return image.updateMask(angleMask.gte(31).and(angleMask.lte(45)));
}

// Function to get the SAR Collection.
function getCollection(dates, roi, orbitPass0) {
  var sarCollFloat = ee.ImageCollection('COPERNICUS/S1_GRD_FLOAT')
    .filterBounds(roi)
    .filterDate(dates[0], dates[1])
    .filter(ee.Filter.eq('orbitProperties_pass', orbitPass0));
```

```
    return sarCollFloat.map(maskAngle).select(['VV', 'VH']);  
}
```

Copy and paste the code below to import the Sentinel-1 collection, define time variables (a list of dates) and the orbit pass variable, apply the functions, create a mosaic by using the `mosaic` function, and clip the mosaic to the study area.

```
// Define variables: the period of time and the orbitpass.  
var listOfDates = ['2021-08-01', '2021-08-12'];  
var orbitPass = 'DESCENDING';  
  
// Apply the function to get the SAR mosaic.  
var sarImageColl = getCollection(listOfDates, aoi, orbitPass)  
  .mosaic()  
  .clip(aoi);  
print('SAR Image Mosaic', sarImageColl);
```

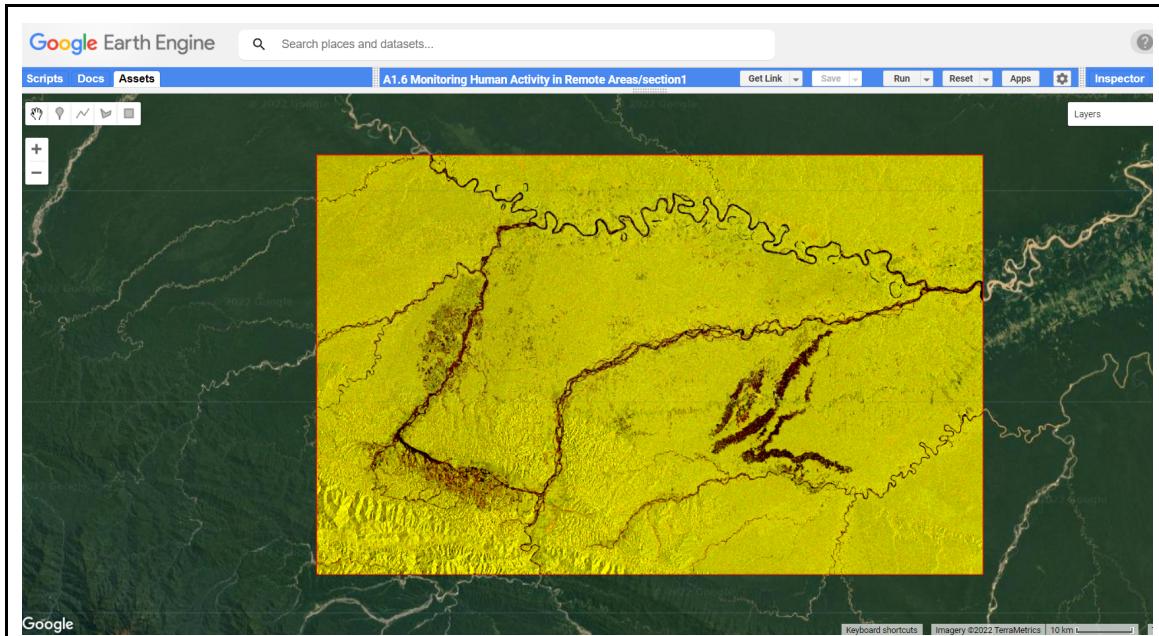
Before adding the mosaic to the map, it's important to scale the values to a logarithmic scale (`log10().multiply(10.0)`). The parameters of visualization (`sarVis`) should be taken between 3 and -23 decibels (dB). Copy and paste the code below to do so and to add the mosaic to the map (Fig. A1.8.5). The code creates a scaled image and draws it using visualization parameters set through trial and error.

```
// Apply logarithmic scale.  
var sarImageScaled = sarImageColl.log10().multiply(10.0);  
  
// Visualize results.  
var sarVis = {  
  bands: ['VV', 'VH', 'VV'],  
  min: [-18, -23, 3],  
  max: [-4, -11, 15]  
};  
Map.addLayer(sarImageScaled, sarVis, 'Sentinel-1 / SAR Mosaic');
```

**Code Checkpoint A18a.** The book's repository contains a script that shows what your code should look like at this point.

**Question 1.** How many bands (polarizations) does Sentinel-1 have over the area of study?

**Question 2.** Using the **Inspector** tool, explore the values from the VV and VH bands over different land covers. Which one do you think is better able to detect forested areas?



**Fig. A1.8.5** SAR Sentinel-1 mosaic generated in previous section

## Section 2. Creating a SAR Mosaic Time Series

We will reuse most of the code from Section 1, only changing the period of time and not applying the mosaic function just yet. Start this section by opening the following code checkpoint.

**Code Checkpoint A18b.** The book's repository contains a script to use to begin this section. You will need to start with that script and paste code below into it.

Expand the SAR `ImageCollection` in the **Console** and note that it is composed of 30 elements. To create a time series of mosaics, we need to define two additional functions (`getDates` and `mosaicSAR`). The first function converts the format of the date from milliseconds to format '`'YYYY-MM-dd'`'. The second function filters the SAR `ImageCollection` using the list of dates and generates a mosaic. The result is an `ImageCollection` of mosaics per date. Copy and paste the code below to define these two functions.

```

// Function to get dates in 'YYYY-MM-dd' format.
function getDates(dd) {
  return ee.Date(dd).format('YYYY-MM-dd');
}

// Function to get a SAR Mosaic clipped to the study area.
function mosaicSAR(dates1) {
  dates1 = ee.Date(dates1);
  var imageFilt = sarImageColl
    .filterDate(dates1, dates1.advance(1, 'day'));
  return imageFilt.mosaic()
    .clip(aoi)
    .set({
      'system:time_start': dates1.millis(),
      'dateYMD': dates1.format('YYYY-MM-dd')
    });
}

```

Now, copy and paste the code below to generate a list of dates without duplicate elements (i.e., where there are images from the same dates in the collection, we only keep one). We avoid duplicates by using the `ee.List.distinct` and the `getDates` functions, and output an `ImageCollection` of mosaics per date.

```

// Function to get a SAR Collection of mosaics by date.
var datesMosaic = ee.List(sarImageColl
  .aggregate_array('system:time_start'))
  .map(getDates)
  .distinct();

// Get a SAR List and Image Collection of mosaics by date.
var getMosaicList = datesMosaic.map(mosaicSAR);
var getMosaicColl = ee.ImageCollection(getMosaicList);
print('get Mosaic Collection', getMosaicColl);

```

Finally, copy and paste the code below to set the visualization parameter (`sarVis`) and add two SAR mosaics filtered by the date of acquisition as an example (one from 2021-01-04 and the other from 2021-12-18; Fig. A1.8.6).

```

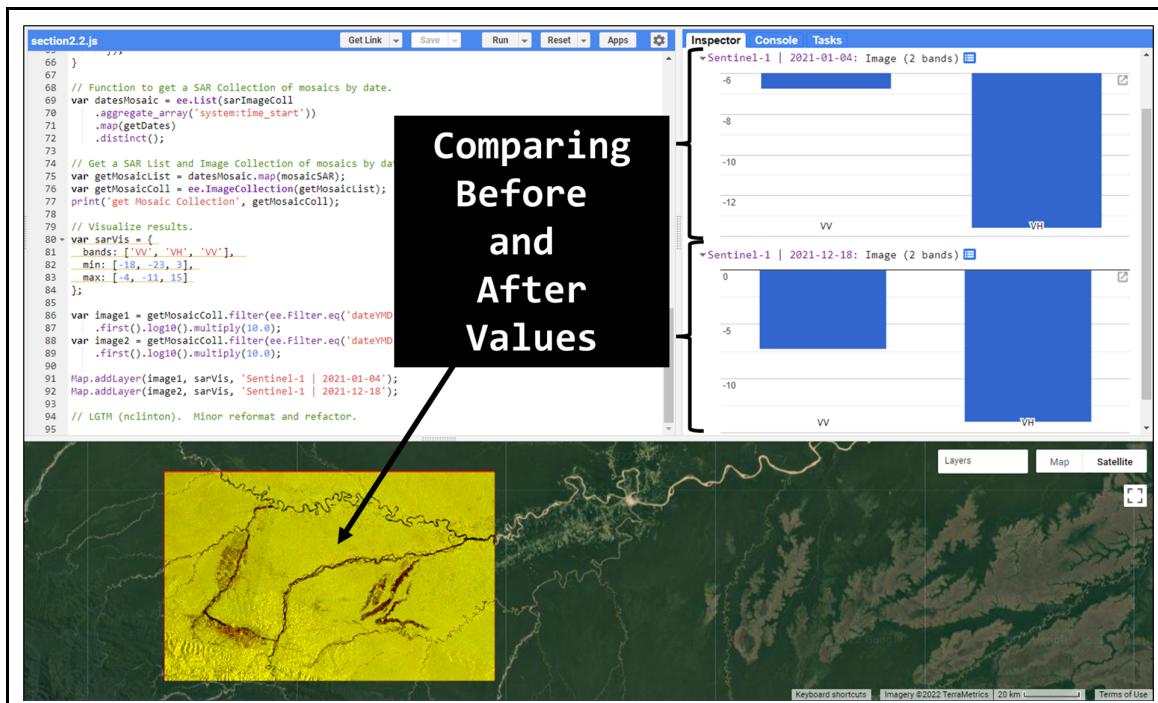
// Visualize results.

var sarVis = {
  bands: ['VV', 'VH', 'VV'],
  min: [-18, -23, 3],
  max: [-4, -11, 15]
};

var image1 = getMosaicColl
  .filter(ee.Filter.eq('dateYMD', '2021-01-04'))
  .first().log10().multiply(10.0);
var image2 = getMosaicColl
  .filter(ee.Filter.eq('dateYMD', '2021-12-18'))
  .first().log10().multiply(10.0);

Map.addLayer(image1, sarVis, 'Sentinel-1 | 2021-01-04');
Map.addLayer(image2, sarVis, 'Sentinel-1 | 2021-12-18');

```



**Fig. A1.8.6** SAR Sentinel-1 mosaics generated in Section 2. The **Inspector** tool shows a comparison between before and after values of VV and VH SAR bands.

Note that we applied the logarithmic scale for visualization purposes. Zoom in and switch between the layers to note the differences between the images.

**Code Checkpoint A18c.** The book's repository contains a script that shows what your code should look like at this point.

**Question 3.** How many images were taken by Sentinel-1 over the area of study between 2018-01-01 and 2020-01-01?

**Question 4.** Using the **Inspector** tool, explore the temporal changes of the values from VV and VH bands over new mining areas.

### **Section 3. Generate SAR Change Detection**

There are different methods for detecting changes using SAR data. In this case, we will use a SAR change detection method based on Canty et al. (2020).

This methodology allows us to identify changes for a series of “ $k$ ” uncorrelated SAR images using a pixel-based omnibus likelihood ratio test statistic Q for covariance matrices ( $\Sigma_i$ ,  $i = 1 \dots k$ ), based on a Wishart distribution. The Q test is defined by

$$\ln Q = n \left( pk \ln k + \sum_{i=1}^k \ln |X_i| - k \ln \left| \sum_{i=1}^k X_i \right| \right)$$

Where  $X_i = n\Sigma^*$  (with  $\Sigma^*$  being the maximum likelihood estimate of the covariance matrices  $\Sigma_i$ ), “n” the number of looks and “p” the dimensionality of the covariance matrices (with p = 2 for dual polarization SAR data). The  $| \cdot |$  denotes the determinant.

The Q test is an omnibus test statistic because it evaluates the equality of several covariance matrices simultaneously. Thus, this test statistic tests the null hypothesis (no change,  $H_0$ ) against alternative hypothesis (change,  $H_1$ ) using SAR time series pixel-based data and the level of significance (probability that the null hypothesis  $H_0$  is true, also known as  $p$ -value) estimated in each iteration.

In the first iteration, the first two first images in the time series are tested (null hypothesis of no change against the alternative of change):

1)  $H_0: \Sigma_1 = \Sigma_2$  against  $H_1: \Sigma_1 \neq \Sigma_2 \rightarrow$  Null hypothesis rejected?

If the Null hypothesis ( $H_0$ ) is not rejected, then the test continues including the values of the next image in the series:

2)  $H_0: \Sigma_1 = \Sigma_2 = \Sigma_3$  against  $H_1: \Sigma_1 = \Sigma_2 \neq \Sigma_3 \rightarrow$  Null hypothesis rejected?

However, if the null hypothesis is rejected in this iteration, then the interval of time for this change is labeled and the test is restarted from there.

The explanation of the Omnibus likelihood ratio test statistic is beyond the scope of this chapter but more details can be found in Carty et al. (2020).

The next steps correspond to obtaining a single change detection output image using the time series of mosaics generated in Section 2.

To do so, we will use modules adapted from the original JavaScript libraries by Carty et al (2020). Beginning from the last code checkpoint from Section 2, copy and paste the code below to import the adapted modules and define a variable that stores the number of images in our collection. This number will be used later on for visualization purposes.

```
// Libraries of SAR Change Detection (version modified).
// The original version can be found in:
// users/mortcarty/changedetection
var omb = require(
  'projects/gee-edu/book:Part A - Applications/A1 - Human
  Applications/A1.8 Monitoring Gold Mining Activity Using
  SAR/modules/omnibusTest_v1.1'
);
var util = require(
  'projects/gee-edu/book:Part A - Applications/A1 - Human
  Applications/A1.8 Monitoring Gold Mining Activity Using
  SAR/modules/utilities_v1.1'
);

// Count the length of the list of dates of the time-series.
var countDates = datesMosaic.size().getInfo();
```

Before applying the SAR change algorithm, we need to define the input parameters such as the `significance` and the reducer to be applied (`median` in this case). The result is an `ee.Dictionary` that contains several images: among these are `cmap`, `smap`, `fmap`, `bmap`. The `cmap` image shows the occurrence of the most recent significant change, `smap` shows the first significant change, `fmap` shows the frequency of significant changes, and `bmap` shows the interval in which each significant change occurred.

Copy and paste the code below to define such variables, apply the algorithm to the list of SAR mosaics (`getMosaicList`), and extract the results.

```
// Run the algorithm and print the results.  
var significance = 0.0001;  
var median = true;  
var result = ee.Dictionary(omb.omnibus(getMosaicList, significance,  
    median));  
print('result', result);  
  
// Change maps generated (cmap, smap, fmap and bmap)  
// are detailed in the next commented lines.  
  
// cmap: the interval in which the most recent significant change  
occurred (single-band).  
// smap: the interval in which the first significant change occurred  
(single-band).  
// fmap: the frequency of significant changes (single-band).  
// bmap: the interval in which each significant change occurred ((k - 1)-band).  
  
// Extract and print the images result  
// (cmap, smap, fmap and bmap) from the ee.Dictionary.  
var cmap = ee.Image(result.get('cmap')).byte();  
var smap = ee.Image(result.get('smap')).byte();  
var fmap = ee.Image(result.get('fmap')).byte();  
var bmap = ee.Image(result.get('bmap')).byte();
```

The values for `cmap`, `smap`, and `bmap` are numbers that correspond to dates. These are the dates that are stored in the `datesMosaic` list. For example, the pixel value of 0 corresponds to the first date (2021-01-04), the pixel value of 1 corresponds to the second date (2021-01-16), and so on (Fig. A1.8.7).

If we want to export the resulting images we need to also export the list of dates. To do so, we need to create a `FeatureCollection` since we can't currently export lists directly

in Earth Engine. Copy and paste the code below to create a `FeatureCollection` where each feature contains the date information as a property. We are also printing the dates in order to visualize the pixel-date association (expand the list on the **Console** to see it.)

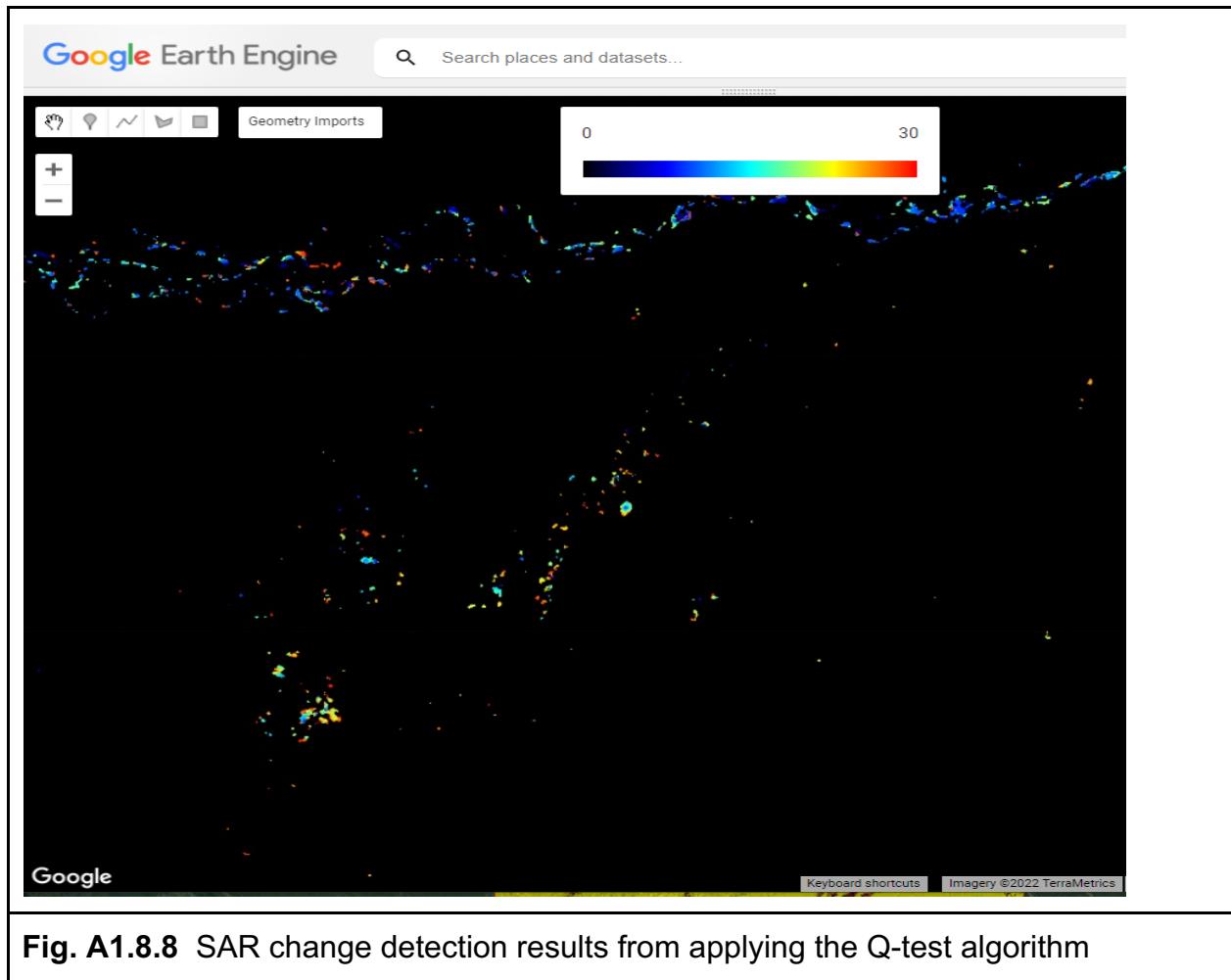
```
// Build a Feature Collection from Dates.  
var fCollectionDates = ee.FeatureCollection(datesMosaic  
    .map(function(element) {  
        return ee.Feature(null, {  
            prop: element  
       });  
    }));  
print('Dates', datesMosaic);
```

system:index	prop
0	2021-01-04
1	2021-01-16
2	2021-01-28
3	2021-02-09
4	2021-02-21
5	2021-03-05
6	2021-03-17
7	2021-03-29
8	2021-04-10
9	2021-04-22
10	2021-05-04
11	2021-05-16
12	2021-05-28
13	2021-06-09
14	2021-06-21
15	2021-07-03
16	2021-07-15
17	2021-07-27
18	2021-08-08
19	2021-08-20
20	2021-09-01
21	2021-09-13
22	2021-09-25
23	2021-10-07
24	2021-10-19
25	2021-10-31
26	2021-11-12
27	2021-11-24
28	2021-12-06
29	2021-12-18

**Fig. A1.8.7** List of dates to be exported as a CSV file. Each value (index) is associated with a specific date of change of the raster file (smap).

Now we can add the results to the map. Copy and paste the code below to define visualizations parameters, make a legend that associates date numbers with colors, and add the resulting images (Fig. A1.8.8). To load results faster, change the `Map.centerObject` function at the top of the script to `Map.setCenter(-70.003, -12.849, 12)`—we are zooming in to a specific area—and leave only the `smap` layer checked under the **Layers** panel.

```
// Visualization parameters.  
var jet = ['black', 'blue', 'cyan', 'yellow', 'red'];  
var vis = {  
    min: 0,  
    max: countDates,  
    palette: jet  
};  
  
// Add resulting images and legend to the map.  
Map.add(util.makeLegend(vis));  
Map.addLayer(cmap, vis, 'cmap - recent change (unfiltered)');  
Map.addLayer(smap, vis, 'smap - first change (unfiltered)');  
Map.addLayer(fmap.multiply(2), vis, 'fmap*2 - frequency of changes');
```



**Fig. A1.8.8** SAR change detection results from applying the Q-test algorithm

Now, copy and paste the code below to export two items to Google Drive: `fCollectionDates`, the dates of SAR images processed (Fig. A1.8.7); and `smap`, the image of the first significant change (Fig. A1.8.8).

```
// Export the Feature Collection with the dates of change.
var exportDates = Export.table.toDrive({
  collection: fCollectionDates,
  folder: 'datesChangesDN',
  description: 'dates',
  fileFormat: 'CSV'
});
// Export the image of the first significant changes.
var exportImgChanges = Export.image.toAsset({
  image: smap,
  description: 'smap',
```

```
assetId: 'your_asset_path_here/' + 'smap',
region: aoi,
scale: 10,
maxPixels: 1e13
});
```

**Code Checkpoint A18d.** The book's repository contains a script that shows what your code should look like at this point.

**Question 5.** What is the difference between the `smap` and `cmap` images?

**Question 6.** How does the `FeatureCollection` `fCollectionDate` relate to the change maps `smap`, `cmap`, and `fmap`?

#### Section 4. Filtering and Postprocessing Alerts

As explained earlier in the Practicum, the `smap` results need to be filtered in order to eliminate possible false positives. The false positives are associated with the forest loss due to river morphology and the presence of muddy water bodies (Fig. A1.8.2).

In this section, we will explore different options to filter false positives and, therefore, postprocess the results generated.

Like we did in previous sections, copy and paste the code below into a new script to import the study area and the exported `smap` image. Remember that our analysis covered 30 Sentinel-1 images from distinct dates.

```
///////////
/// Section Four
///////////

// Define the area of study.
var aoi = ee.FeatureCollection('projects/gee-book/assets/A1-8/mdd');

// Center the map.
Map.centerObject(aoi, 10);

// Create an empty image.
var empty = ee.Image().byte();
```

```

// Convert the area of study to an EE image object so we can visualize
// only the boundary.
var aoiOutline = empty.paint({
  featureCollection: aoi,
  color: 1,
  width: 2
});

// Select the satellite basemap view.
Map.setOptions('SATELLITE');

// Add the area of study boundary to the map.
Map.addLayer(aoiOutline, {
  palette: 'red'
}, 'Area of Study');

// Import the smap result from section 3.
var changeDetect = ee.Image('projects/gee-book/assets/A1-8/smap');

// Visualization parameters.
var countDates = 30;
var jet = ['black', 'blue', 'cyan', 'yellow', 'red'];
var vis = {
  min: 0,
  max: countDates,
  palette: jet
};

// Add results to the map.
Map.addLayer(changeDetect, vis, 'Change Map Unfiltered');

```

Next, copy and paste the code below to import from the Earth Engine data catalog and add to the map all the sources of information for filtering false positives: Shuttle Radar Topography Mission (SRTM) digital elevation data, Hansen Global Forest Change data, and JRC Global Surface Water data (Fig. A1.8.9).

```

// Digital Elevation Model SRTM.
// https://developers.google.com/earth-
engine/datasets/catalog/USGS_SRTMGL1_003

```

```

var srtm = ee.Image('USGS/SRTMGL1_003').clip(aoi);
var slope = ee.Terrain.slope(srtm);
var srtmVis = {
  min: 0,
  max: 1000,
  palette: ['black', 'blue', 'cyan', 'yellow', 'red']
};
Map.addLayer(srtm, srtmVis, 'SRTM Elevation');
var slopeVis = {
  min: 0,
  max: 15,
  palette: ['black', 'blue', 'cyan', 'yellow', 'red']
};
Map.addLayer(slope, slopeVis, 'SRTM Slope');

// Hansen Global Forest Change v1.8 (2000-2020)
// https://developers.google.com/earth-
engine/datasets/catalog/UMD_hansen_global_forest_change_2020_v1_8
var gfc = ee.Image('UMD/hansen/global_forest_change_2020_v1_8').clip(
  aoi);
var forest2020 = gfc.select('treecover2000')
  .gt(0)
  .updateMask(gfc.select('loss')
    .neq(1))
  .selfMask();
Map.addLayer(forest2020,
{
  min: 0,
  max: 1,
  palette: ['black', 'green']
},
'Forest cover 2020');

// JRC Yearly Water Classification History, v1.3 (Updated until Dec
2020).
// https://developers.google.com/earth-
engine/datasets/catalog/JRC_GSW1_3_GlobalSurfaceWater
var waterJRC = ee.Image('JRC/GSW1_3/GlobalSurfaceWater').select(
  'max_extent');
var waterVis = {

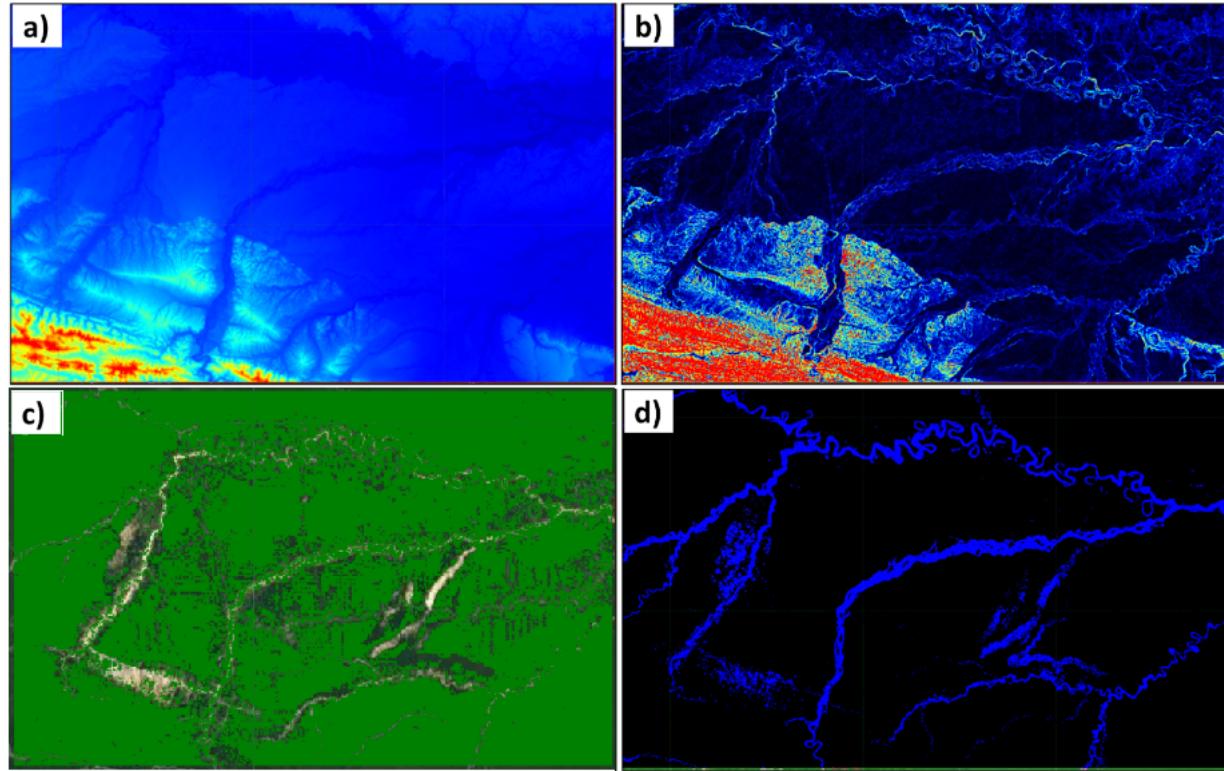
```

```

    min: 0,
    max: 1,
    palette: ['blue', 'black']
};

Map.addLayer(waterJRC.eq(0), waterVis, 'Water Bodies until 2020');

```



**Fig. A1.8.9** Layers used to filter false positives alerts: a) SRTM elevation, red color shows areas over 1000 meters above sea level; b) SRTM slope, red color shows areas with slope over 15 degrees; c) Hansen Global Forest Change, green color shows forested areas updated to 2020; d) JRC Yearly Water Classification History, blue color shows the maximum extent of water surface detected from 1984 to 2020.

You can toggle these layers on and off, zoom in and out, and inspect pixel values to analyze them.

The SRTM elevation layer ('`SRTM Elevation`') and the slope ('`SRTM Slope`') derived from it with `ee.Terrain.slope` show red areas that correspond to an altitude over 1000 meters above sea level and a slope over 15 degrees, compared to blue areas, which are closer to sea level or to flat terrain. We chose these options because mining activity in

this region is located in lowlands. Furthermore, the SAR Sentinel-1 data provided by Earth Engine are not radiometric-terrain corrected. So, steep slopes ( $>15$  degrees) generate distortions in SAR images, and therefore, potential false changes between two or more images taken at different times.

The Hansen Global Forest Change data are composed of forested areas in 2000 (the '`'treecover2000'` band) and the forest loss between 2001 and 2020. In this sense, the binary layer '`'Forest cover 2020'`' previously generated and added shows in green the forested areas updated until 2020, with all the non-forested and the forest loss between 2001-2020 areas masked.

The JRC Yearly Water Classification History data shows surface water extent and change between 1984 and 2020. The binary layer '`'Water Bodies until 2020'`' previously added shows in blue the water bodies' maximum extent between 1984 and 2020. Non-water bodies are shown in black.

Note that alluvial mining expansion pattern in the study area is associated with primary forest loss, and the appearance of new surface water patches (Fig. A1.8.2).

Now, we will filter the false positives based on thresholds. We will mask any pixel in `smap` marked as change over areas greater than 1000 meters above sea level, slope greater than 15 degrees according to the SRTM data (classified as forest until 2020 by the Hansen data), and that are not classified as water bodies by the JRC dataset. Copy and paste below to add the filtered results to the map.

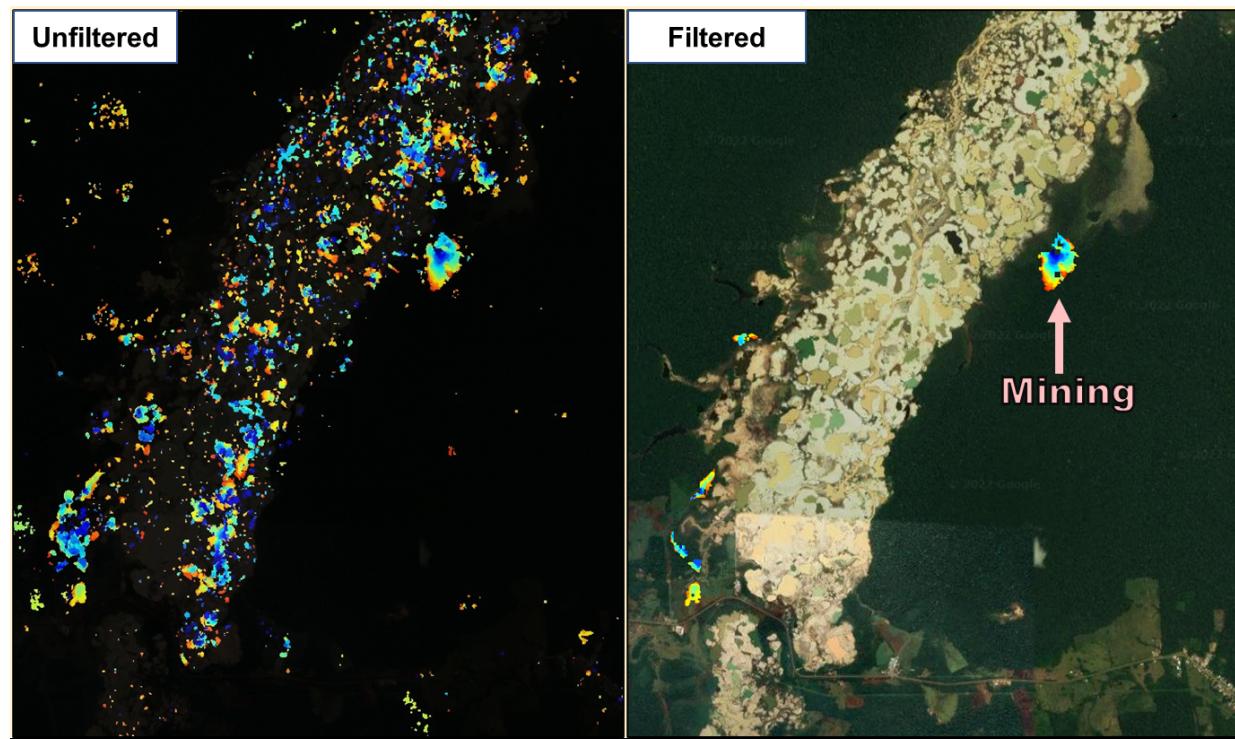
```
// Apply filters through masks.  
var alertsFiltered = changeDetect  
  .updateMask(srtm.lt(1000))  
  .updateMask(slope.lt(15))  
  .updateMask(forest2020.eq(1))  
  .updateMask(waterJRC.eq(0))  
  .selfMask();  
  
// Add filtered results to the map.  
Map.addLayer(alertsFiltered,  
{  
  min: 0,  
  max: countDates,  
  palette: jet  
},
```

```
'Change Map Filtered',  
1);
```

We can still improve the results a bit more. Copy and paste the code below to define and apply a function that eliminates small pixel patches and isolated pixels. We do this because we know that in this area, mining activities occur in areas of at least 0.5 hectares.

```
// Function to filter small patches and isolated pixels.  
function filterMinPatchs(alerts0, minArea0, maxSize0) {  
    var pixelCount = alerts0.gt(0).connectedPixelCount(maxSize0);  
    var minPixelCount = ee.Image(minArea0).divide(ee.Image  
        .pixelArea());  
    return alerts0.updateMask(pixelCount.gte(minPixelCount));  
}  
  
// Apply the function and visualize the filtered results.  
var alertsFiltMinPatchs = filterMinPatchs(alertsFiltered, 10000, 200);  
  
Map.addLayer(alertsFiltMinPatchs, vis,  
    'Alerts Filtered - Minimum Patches');
```

Turn off all the other layers to visualize the filtered result. By analyzing the results without the filters and with the filters, we can see that we eliminated most of the false positives (Fig. A1.8.10).



**Fig. A1.8.10** A comparison of results, after ('Alerts Filtered - Minimum Patches') and before filtering false positives ('Change Map Unfiltered').

Finally, we can export the outcome. Copy and paste the code below to export to the Drive.

```
// Export filtered results to the Drive.
Export.image.toDrive({
  image: alertsFiltMinPatchs,
  description: 'alertsFiltered',
  folder: 'alertsFiltered',
  region: aoi,
  scale: 10,
});
```

**Code Checkpoint A18e.** The book's repository contains a script that shows what your code should look like at this point.

## Synthesis

In this chapter, we mapped the changes generated by the alluvial mining activity over a forested area and between a period of time using Sentinel-1 SAR time series. For this, we separated the methodology into three steps. First, we were able to build a time series from Sentinel-1 mosaics. Second, we estimate all the changes based on the omnibus Q-test change detection algorithm. Finally, we filter the detected changes based on existing forest/non-forest, water bodies, and elevation data, and a minimum mapping unit in order to retrieve the changes generated by the alluvial mining activity only.

Now, it's your turn to explore the use of the methodology.

**Assignment 1.** In this chapter we applied the methodology for a given SAR orbit. Describe how we could identify the different SAR orbits over a specific area of study.

**Assignment 2.** Describe whether these alerts, which are generated by a change detection algorithm, are different for the ascending or descending orbit over our area of study.

## Conclusion

In this chapter, you have learned how to analyze the changes generated by alluvial mining activity over forested areas based on the application of a SAR change detection methodology. This is possible because of the significant impact generated by this activity over the environment (that is, the deforestation and the use of water in the alluvial gold wash machine) that is reflected in the backscatter signal of SAR data. This methodology can be applied to other study cases since a good understanding of the principles of change detection has been achieved in this chapter and complemented by chapters in part F4.

## Feedback

To review this chapter and make suggestions or note any problems, please go now to [bit.ly/EEFA-review](https://bit.ly/EEFA-review). You can find summary statistics from past reviews at [bit.ly/EEFA-reviews-stats](https://bit.ly/EEFA-reviews-stats).

## References

Aguirre GA, Robles RRC, Duarez FMG, et al (2021) Dinámica de la pérdida de bosques en el sureste de la Amazonía peruana: Un estudio de caso en Madre de Dios. Ecosistemas 30:2175. <https://doi.org/10.7818/ECOS.2175>

Álvarez-Berrios N, L'Roe J, Naughton-Treves L (2021) Does formalizing artisanal gold mining mitigate environmental impacts? Deforestation evidence from the Peruvian Amazon. *Environ Res Lett* 16:64052. <https://doi.org/10.1088/1748-9326/abede9>

Asner GP, Llactayo W, Tupayachi R, Luna ER (2013) Elevated rates of gold mining in the Amazon revealed through high-resolution monitoring. *Proc Natl Acad Sci USA* 110:18454–18459. <https://doi.org/10.1073/pnas.1318271110>

Asner GP, Tupayachi R (2017) Accelerated losses of protected forests from gold mining in the Peruvian Amazon. *Environ Res Lett* 12:94004. <https://doi.org/10.1088/1748-9326/aa7dab>

Ballère M, Bouvet A, Mermoz S, et al (2021) SAR data for tropical forest disturbance alerts in French Guiana: Benefit over optical imagery. *Remote Sens Environ* 252:112159. <https://doi.org/10.1016/j.rse.2020.112159>

Bourbigot M, Johnsen H, Piantanida R, Hajduch G (2021) Sentinel-1 Product Specification for products generated with IPF 3.4.0. CLS S-1 Mission Perform. Cent. pp. 186

Caballero Espejo J, Messinger M, Román-Dañobeytia F, et al (2018) Deforestation and forest degradation due to gold mining in the Peruvian Amazon: A 34-year perspective. *Remote Sens* 10:1903. <https://doi.org/10.20944/preprints201811.0113.v1>

Canty MJ, Nielsen AA, Conradsen K, Skriver H (2020) Statistical analysis of changes in Sentinel-1 time series on the Google Earth Engine. *Remote Sens* 12:46. <https://doi.org/10.3390/rs12010046>

Csillik O, Asner GP (2020) Near-real time aboveground carbon emissions in Peru. *PLoS One* 15:e0241418. <https://doi.org/10.1371/journal.pone.0241418>

Flores-Anderson AI, Herndon KE, Thapa RB, Cherrington E (2019) The SAR Handbook: Comprehensive Methodologies for Forest Monitoring and Biomass Estimation

Fröjse L (2014) Unsupervised change detection using multi-temporal SAR data: A case study of Arctic sea ice. KTH: Royal Institute of Technology

Gorelick N, Hancher M, Dixon M, et al (2017) Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sens Environ* 202:18–27. <https://doi.org/10.1016/j.rse.2017.06.031>

Kellndorfer J, Flores-Anderson AI, Herndon KE, Thapa RB (2019) Using SAR data for mapping deforestation and forest degradation. *SAR Handbook Compr Methodol For*

Monit Biomass Estim ServirGlobal Hunstville, AL, USA 65–79.  
<https://doi.org/10.25966/68c9-gw82>

Nicolau AP, Flores-Anderson A, Griffin R, et al (2021) Assessing SAR C-band data to effectively distinguish modified land uses in a heavily disturbed Amazon forest. *Int J Appl Earth Obs Geoinf* 94:102214. <https://doi.org/10.1016/j.jag.2020.102214>

Nicolau AP, Herndon K, Flores-Anderson A, Griffin R (2019) A spatial pattern analysis of forest loss in the Madre de Dios region, Peru. *Environ Res Lett* 14:124045.  
<https://doi.org/10.1088/1748-9326/ab57c3>

Proisy C, Mougin E, Fromard F, Rudant JP (1998) Télédétection radar des mangroves de Guyane Française. Séminaire Télédétection Végétation, Montpellier, Fr 1996-11-26, Photo interprétation 36:26

RAISG (2020) Amazonia Under Pressure.  
<https://atlas2020.amazoniasocioambiental.org/en>. Accessed 25 Feb 2022

Richards JA (2009) Remote Sensing with Imaging Radar. Springer

Rignot EJM, van Zyl JJ (1993) Change detection techniques for ERS-1 SAR data. *IEEE Trans Geosci Remote Sens* 31:896–906. <https://doi.org/10.1109/36.239913>

Yard EE, Horton J, Schier JG, et al (2012) Mercury exposure among artisanal gold miners in Madre de Dios, Peru: A cross-sectional study. *J Med Toxicol* 8:441–448.  
<https://doi.org/10.1007/s13181-012-0252-0>