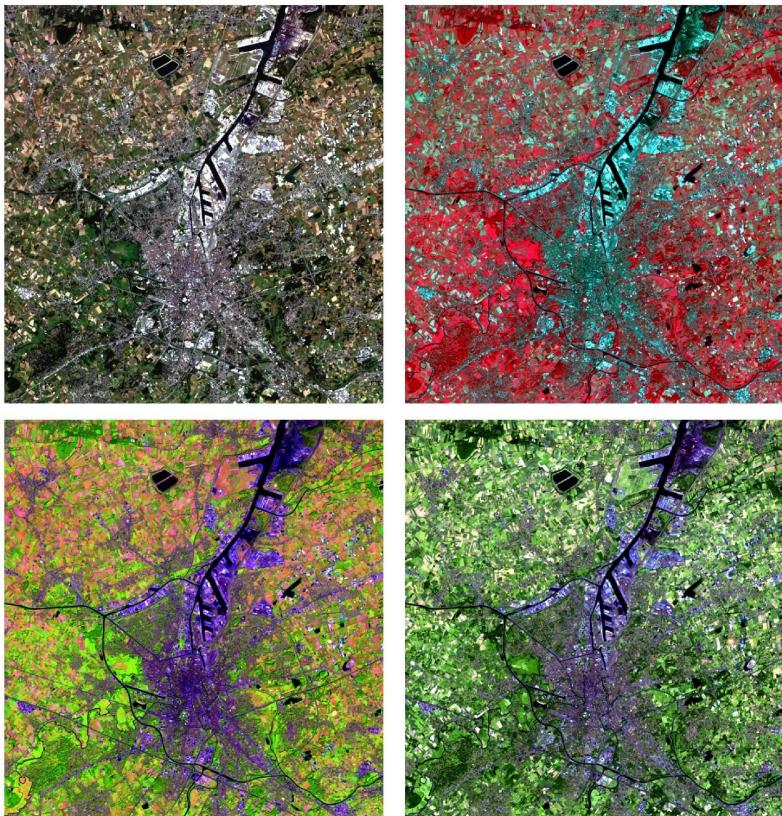


Table of contents

1. Home	2
1.1 Over deze cursussite	2
2. P2 - Beeldvoorbewerking	4
2.1 Practicum 2: Image download & preprocessing	4
2.2 Sentinel-2 intro and download	5
2.3 Introduction to SNAP	8
2.4 RGB colour composites	10
2.5 Image preprocessing in SNAP	12
2.6 RGB colour composites 2	16
3. P3- Intro tot Google Earth Engine	17
3.1 Doel van het practicum	17
3.2 EarthEngine data catalog	18
3.3 Introductie tot Earth Engine	19
3.4 Satellietdata oproepen, filteren en visualiseren	21
3.5 Reducing ImageCollections	24
3.6 Cloud masking	25
3.7 (Extra) Oefeningen	27
4. P4 - Feature Extraction	28
4.1 Intro	28
4.2 Spectrale indices	29
4.3 Beeldstatistieken	33
4.4 Textuuranalyse	35
4.5 Principale componenten analyse	36
4.6 Oefeningen	39
5. P5 - Beeldclassificatie	41
5.1 Practicum 5: Beeldclassificatie	41
5.2 CASE - Mangrove monitoring in Suriname	42
5.3 Niet-gesuperviseerde classificatie	45
5.4 Gesuperviseerde classificatie	47
5.5 Verbeteren van de classificatie	53
5.6 Oppervlaktebepaling	54
5.7 Spectrale responsiecurves	55
5.8 Oefeningen	57

1. Home

1.1 Over deze cursussite



Over deze cursus

Welkom bij de cursussite voor de practica van Teledetectie 2021. Deze cursus is ontworpen om studenten te basis aan te leren voor het verkrijgen, voorverwerken, analyseren en exporteren van remote sensing data.

Volgende topics komen doorheen de 5 practica's aan bod:

- **Practicum 1:** Beeldinterpretatie (zie Ufora)
- **Practicum 2:** Downloaden en voorverwerking van een enkel Sentinel-2 beeld in SNAP
- **Practicum 3:** Toegang en leren werken met Google Earth Engine. Basis beeldvisualisatie en
- **Practicum 4:** Feature extractie in Google Earth Engine
- **Practicum 5:** Beeldclassificatie
- **Practicum 6:** Tijdserieanalyse

Deze cursussite laat toe om snel stukken scripts te kopiëren, om zo vlotte gang van zaken mogelijk te maken. In een 'tutorial' leer je bepaalde functionaliteit kennen door het copy-pasten van de voorbeeldscripts. In de hierop volgende oefeningen zul je zelf de gebruikte tools moeten toepassen.

Doorheen de practica wordt deze site aangevuld met nieuwe documentatie, extra informatie en FAQ's.

Afdruckbare PDF's

Afdruckbare pdf's per practicum worden voorzien, voor wie hieraan de voorkeur geeft. De PDF-versie van de volledige practicasite is te vinden via [deze link](#).



[Remote Sensing | Spatial Analysis lab \(REMOSA\)](#)



2. P2 - Beeldvoorbewerking

2.1 Practicum 2: Image download & preprocessing

Doeel van het practicum

In dit practicum zien we enkele tools voor het downloaden van losse Sentinel-2 beelden, het maken van verschillende beeldcomposieten en enkele 'preprocessing'-technieken. Hiervoor wordt gebruik gemaakt van SNAP. Na deze 'introductie'-sessie wordt gebruik gemaakt van Google Earth Engine voor verdere beeldverwerkingoefeningen.

Inhoud:

- Downloaden van remote sensing data:
 - via ESA sentinel hub
 - via andere bronnen

- Introductie tot ESA SNAP:

- Inlezen van RS beelden
- Basisfunctionaliteiten
- Aanmaken van beeldcomposieten
- SNAP vs andere software

- Beeldvoorbewerking in SNAP (Sentinel-2):

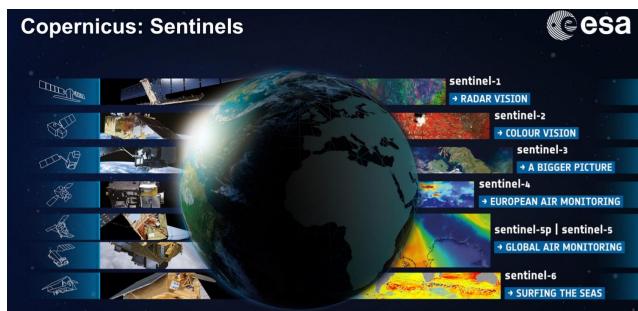
- Radiometrische/atmosferische correctie
- Resampling
- Subsetting
- Mosaicing

2.2 Sentinel-2 intro and download

The ESA Copernicus programme

Copernicus is the EU's Earth Observation Programme, looking at our planet and its environment for the ultimate benefit of all European citizens. The overall goal is achieving a global, continuous, autonomous, high quality, wide range Earth observation capacity.

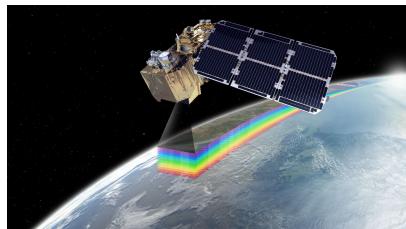
Under the copernicus programme, ESA is developing a series of next-generation Earth observation missions under the name of 'Sentinel' programme. This Sentinel Programme, consists of multiple satellites, each focussing on a different aspect of Earth observation: atmospheric, Oceanic and Land monitoring:



Current Sentinel satellites, with their main goal. (Source: ESA)

In this practical will focus on the multispectral imagery taken by Sentinel 2 satellites.

The Sentinel-2 mission



Sentinel-2 is the copernicus Earth observation mission by ESA with the goal to perform terrestrial observations in support of services such as forest monitoring, land cover changes detection, and natural disaster management. It consists of two identical satellites, Sentinel-2A and Sentinel-2B. An interesting infograph about the Sentinel-2 mission can be found [here](#).

The Sentinel-2 mission has the following capabilities:

- Multi-spectral data with 13 bands in the visible, near infrared, and short wave infrared part of the spectrum
- Systematic global coverage of land surfaces from 56° S to 84° N, coastal waters, and all of the Mediterranean Sea
- Revisiting every 5 days under the same viewing angles.
- Spatial resolution of 10 m, 20 m and 60 m
- 290 km field of view
- Free and open data policy

To achieve frequent revisits and high mission availability, the two identical Sentinel-2 satellites (Sentinel-2A and Sentinel-2B) operate simultaneously. The orbit is Sun synchronous at 786 km (488 mi) altitude.

Sentinel 2 data download

All data captured by the ESA copernicus Sentinel program are completely freely available to the public. The most convinient way to download Sentinel data is through the Copernicus Open Access Hub, a platform dedicate to provide easy acces to the user. For this, an user account is required.

To register go to [registration page](#). To acces the data hub, go to <https://scihub.copernicus.eu/>.

Ex 2.1 - Downloading a Sentinel 2 Level 1C image

In the first exercise, you will download an image from the Copernicus Open Access Hub.

- Go to <https://scihub.copernicus.eu/>
- Click 'Open hub' to access the Interactive Graphical User Interface
- Log in (or create an account)



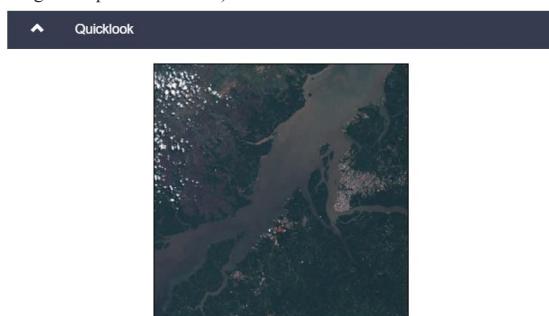
- Zoom to Belém, a city in the north of Brazil, close to the gateway of the Amazon river
- Switch the 'Open street' view to 'sentinel-2 cloudless + Overlay' view
- Switch to 'navigation mode'
- Draw a rectangle around Belém:



- At the button 'Insert search criteria': go for 'advanced search'
- Look for a 2021 image (sensing period), Sentinel-2A, level 1C (product type) with a cloud cover of maximum 10%. Then click on the search button:

<input checked="" type="checkbox"/> Mission: Sentinel-2	Satellite Platform	Product Type
	<input type="button" value="▼"/>	<input type="button" value="▼"/>
	Relative Orbit Number (from 1 to 143)	Cloud Cover % (e.g.[0 TO 9.4])
	<input type="button" value=""/>	<input type="button" value="[0 TO 10]"/>

- Click on the search button
- Search for an image that contains the major part of the city (inspect the image in a quick look view)



- Download this image to a folder on your computer.

Sentinel file naming convention

The naming of the Sentinel products follows the Compact Naming Convention:

MMM_MSIXXX_YYYYMMDDHHMMSS_Nxxyy_ROOO_Txxxxx_"Product Discriminator".SAFE

Where:

MMM: is the mission ID (S2A/S2B)

MSIXXX: MSIL1C denotes the Level-1C product level/ MSIL2A denotes the Level-2A product level (see 'radiometric correction').

YYYYMMDDTHHMMSS: the datatake sensing start time

Nxxyy: the Processing Baseline number (e.g. N0204)

ROOO: Relative Orbit number (R001 - R143)

Txxxxx: Tile Number field

.SAFE: Product Format (Standard Archive Format for Europe)

The products contain two dates. The first date (YYYYMMDDHHMMSS) is the datatake sensing time. The second date is the "Product Discriminator" field, which is 15 characters in length, and is used to distinguish between different end user products from the same datatake. Depending on the instance, the time in this field can be earlier or slightly later than the datatake sensing time.

Thus, the following filename

*S2A_MSIL1C_20170105T013442_N0204_R031_T53NMJ_20170105T013443.SAFE

identifies a Level-1C product acquired by Sentinel-2A on the 5th of January, 2017 at 1:34:42 AM. It was acquired over Tile 53NMJ(2) during Relative Orbit 031, and processed with PDGS Processing Baseline 02.04.

Ex 3.2 - naming convention

- Explain the different components of the name:

S2A_MSIL1C_20180812T143751_N0206_R096_T19KGA_20180812T143751.N0206.R096.T19KGA.20180812T143751.SAFE
(example)

Other useful RS data sources

Earth Explorer

ESA has Sentinel-2, NASA has Landsat. However Landsat has a lower spatial resolution of 30m compared to the 10m of Sentinel-2 and Sentinel 2 has more spectral bands, Landsat imagery is probably the most used EO-data in science. This is because the Landsat program is the longest-running Earth Observation program of the entire Earth. Landsat-1 was already launched on July 23, 1972 resulting. Due to this difference, Landsat is on this moment more useful for historic land-change assessments than Sentinel-2 (launched in 2015).

Landsat data is also freely available to the public. For this, the United States Geological Survey has created a data portal with extensive collections of EO data, with Landsat satellite imagery, Radar data, UAS data, digital line graphs, digital elevation model data, aerial photos, Sentinel satellite data, ...

Link: earthexplorer.usgs.gov

Other data sources

Following website contains a nice overview of online free EO data sources:

<https://www.geoawesomeness.com/list-of-top-10-sources-of-free-remote-sensing-data/>

2.3 Introduction to SNAP

About SNAP



SNAP, the SeNtinel Application Platform is developed by the ESA specifically to process Sentinel-imagery, however also other remotey sensed images can be read. The current version is 8.0.0. SNAP is a relatively new software especially designed for the analysis of Sentinel products (Sentinel 2A was launched in 2015) and hence still contains some bugs (especially for mac-users, might try the older version 7.0.0). Not all applications are supported that you will find in classic Image Processing programs such as ENVI, but it is very user friendly and ideal to introduce you to satellite image processing. Also, it is free!

Overview

Exercise: Opening a Sentinel-2 image in snap

- Open the sentinel image that you have downloaded (you do not need to unzip it). You can do this in several ways:
 - a. Drag and drop the zip folder in the Products explorer
 - b. Click file > Open Products and browse to your zip-folder
 - c. Click and browse to your zip-folder.
- Unfold the image folder. Explore the files included. Open the Blue, Green, Red and NIR image.
- Test the tile buttons. Make sure you can see the four images simultaneously:
- Explore the navigation panel.

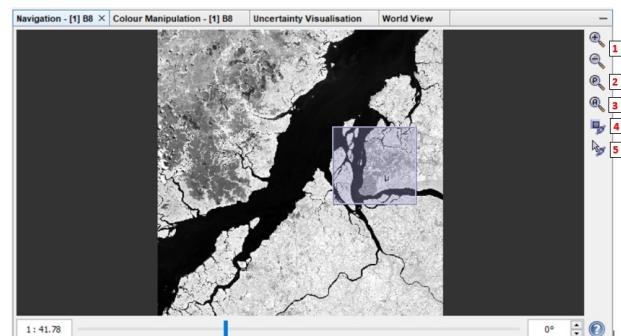
Sentinel 2 Bands

Let's have a quick look at the specifications of a Sentinel-2 image. There are 13 Sentinel 2 bands in total, with a resolution of 10, 20 or 60m:

Sentinel-2 Bands	Central Wavelength (μm)	Resolution (m)
Band 1 - Coastal aerosol	0.443	60
Band 2 - Blue	0.490	10
Band 3 - Green	0.560	10
Band 4 - Red	0.665	10
Band 5 - Vegetation Red Edge	0.705	20
Band 6 - Vegetation Red Edge	0.740	20
Band 7 - Vegetation Red Edge	0.783	20
Band 8 - NIR	0.842	10
Band 8A - Vegetation Red Edge	0.865	20
Band 9 - Water vapour	0.945	60
Band 10 - SWIR - Cirrus	1.375	60
Band 11 - SWIR	1.610	20
Band 12 - SWIR	2.190	20

The navigation window

The **Navigation Window** is used to move the viewport of an Image View, to zoom in and out of it and to rotate the image in steps of 5 degrees using the spinner control below the image preview. The current viewport is depicted by a semi-transparent rectangle, which can be dragged in order to move the viewport to another location.



The navigation window

In the bottom left, you will find the zoom factor: zoom is relative to the drawing extents.

A scale factor of:

- 1 shows a part of the image
- 2 shows entities twice as large
- 0.5 shows entities half as large

The text box at the left side of slider can be used to adjust the zoom factor manually. The Navigation window additionally provides the following features via its tool buttons (top right):

1. **Zoom In:** Zooms in by a factor of 1.2.
- Zoom Out:** Zooms out by a factor of 1/1.2.
2. **Zoom Actual Pixel:** Sets the zoom factor to the default value so that the size of an image pixel has the same size of a display pixel.
3. **Zoom All:** Adjusts the viewport to cover the entire image.
4. **Synchronise Views:** Synchronises the viewports of all compatible image views.
5. **Synchronise Cursor:** Displays a synchronised cursor on all opened image views.

You can also zoom the images by scrolling on the image, or by clicking



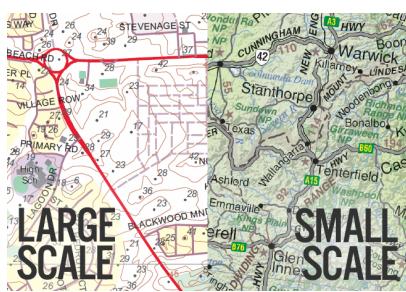
in the toolbar.

Zoom factor vs Representative Fraction

The **zoom factor** is not the same as a **Representative Fraction (RF)**, which is often used to indicate the scale of a map. The RF indicates the ratio between the number of units on the map to the number of units on the ground.

The RF factor 1:100000 e.g. implies that one cm on map is equal to 1 km on land. Maps are described as either large-scale or small-scale. Large-scale maps show a smaller amount of area with a greater amount of detail. The geographic extent shown on a large-scale map is small. A large scaled map expressed as a representative scale would have a smaller number to the right of the ratio.

For example, a large-scale map could have a RF scale of 1: 1,000. Large-scale maps are typically used to show neighbourhoods, a localize area, small towns, etc. Small-scale maps show a larger geographic area with few details on them. The RF scale of a small-scale map would have a much larger number to the right of the colon such as 1: 1,000,000. Small-scale maps are used to show the extent of an entire country, region, or continent.

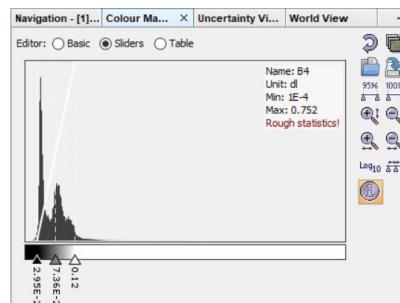


- Zoom to the Airport
- Explore the **World View panel**. The red rectangle indicates the position of the image on the globe.



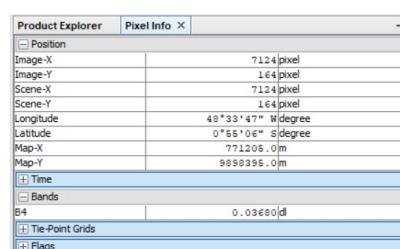
The Colour Manipulation tool

The colour manipulation tool window window is used to modify the colours used in the image. If you are opening an Image View of a data product's band, the Sentinel Toolbox either loads image settings from the product itself (BEAM-DIMAP format only) or uses default colour settings. In the Colour manipulation panel, explore the histogram. On the image, zoom to the airport and adjust the contrast. Restore the contrast afterwards.



Pixel info view

If you click on the tab 'Pixel View' (right to the product explorer), pixel information will be displayed while you move the mouse over the band image view.

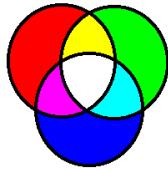


2.4 RGB colour composites

About colour composites

Multispectral imagery, such as Sentinel-2, consists of several bands of data. As seen during in previous chapter, these bands can be displayed individually as a grey scale image (black = low reflectance, white = high reflectance), but they can also be displayed as a combination of three bands: a **colour composite** (NL: kleurcomposit).

When creating a colour composite: the three primary colours are used: red, green and blue. When they are combined in various proportions, different colours are produced per pixel. When 3 spectral bands (both visible as non-visible bands) are assigned to a primary colour, a colour composite is formed.



By combining different proportions of the three primary colours Red, Green and Blue, various colours are created

Two "famous" colour composites

True Colour Composite

The most straightforward colour composite is the **true colour composite** (also **natural colour composite**), where the three visual primary colour bands of a multispectral image are assigned to their corresponding colour.

For Sentinel-2, this composite is created as: Red: B4, Green: B3, Blue: B2.



Sentinel-2 Normal Composite of Ghent.

False Colour Composite

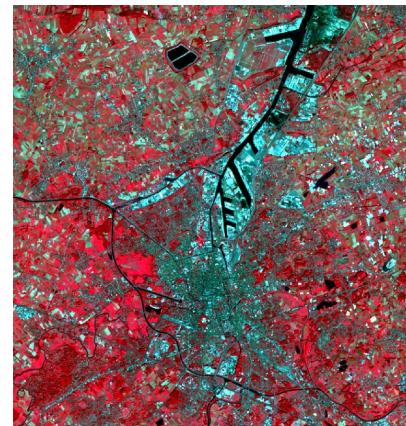
Beside the 'normal' colour composites, any band of a multispectral satellite image can be assigned to the primary colour bands in a composite. In all

those other cases, the colour of a target object on the image, will have a different colour compared to its actual colour.

The most famous of these is the **False Colour Composite**, where the NIR-band is assigned to the red colour, the red band to the green colour and the green band to the blue colour. It is very suitable to detect vegetation, since vegetation has a high reflectance in the NIR band.

Clear water will appear dark-bluish, while turbid water (with a lot of sediments) will be cyan. Bare soils, roads and buildings may appear in various shades of blue, yellow or grey, depending on their composition.

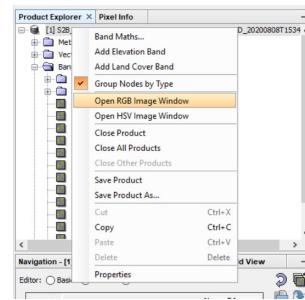
For Sentinel-2, this composite is created as: Red: B8, Green: B4, Blue: B3.



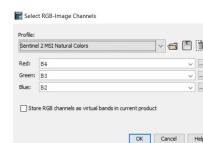
Sentinel-2 False Colour Composite of Ghent.

Opening a RGB image in SNAP

Let's create our own image composites in SNAP! This is actually very easy to do. Just right-click on the image folder and click on 'Open RGB Image window':



A window will appear with some possible S2 band combinations, but you can also create your own.



Some typical S2 band combinations have their own name, such as (Red, Green, Blue):

- Natural Colour: **4 3 2 ***
- False colour Infrared: **8 4 3 ***
- False colour Urban: **12 11 4**
- Agriculture: **11 8 2**
- Atmospheric penetration: **12 11 8a ***
- Healthy vegetation: **8 11 2**
- Land/Water: **8 11 4**
- Natural Colours with Atmospheric Removal: **12 8 3**
- Shortwave Infrared: **12 8 4**
- Vegetation Analysis: **11 8 4**

With the current Sentinel-2 Level 1C-product open, only the band combinations with a * can now be displayed. Why is that?

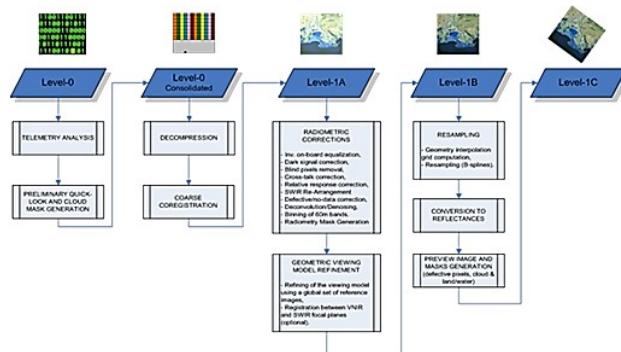
Excercise: open band composites

- Open the image as a natural colour composite
- Open the image as a false colour infrared composite
- Tile the images evenly and explore the difference in colour (for example in the areas with green vegetation).

2.5 Image preprocessing in SNAP

Radiometric & atmospheric correction

Satellite images obtained by the sensing device are not directly usable. They need to go through a series of pre-processing before they are ready to use. The scheme below illustrates the pre-processing steps that Sentinel-images undergo before they are made available for the user. This includes **geometric correction**, some **radiometric correction** (noise reduction, defective pixels identification) the computation of cloud masks, etc. The outcome is a level 1C product, which is Top-Of-the-Atmosphere (TOA).



TOA reflectances are subjected to radiometric bias caused by different lighting conditions, atmospheric interactions and viewing geometry. In order to relate reflectances to physical field properties, TOA reflectance values are converted to BOA (Bottom Of Atmosphere) corrected reflectance values. This radiometric correction is an essential part in image processing. BOA, Sentinel-2 processing level 2A, is available for the user (except for recent images) or can be created by the user itself, using the Sen2Cor freeware.



Figure: A true color comparison of the surface reflectance product (top) and a top of atmosphere reflectance image (bottom) in adjacent scenes captured by the same satellite (Planet.com)

In Snap, the conversion of level 1C TOA-reflectance to level 2A BOA-reflectance can be done through Sen2Cor (plug-in or stand-alone). Sen2Cor corrects the reflectance values based on (among others) ‘look-up tables’, these are tables that relate physical parameters to model coefficients. Parameters such as inclination and product type are sensor dependent

(different for Landsat as for Sentinel or Spot). On board, optical satellites have some meteorological sensors that measure atmosphere features such as the air thickness and the amount of aerosols among others. This information is available as a ‘header file’ for each image.

Since December 2018, users can download Level-2A processed products directly. In case of this exercise, we downloaded a Level 1C product. Thus, let's perform an atmospheric correction!

Exercise: atmospheric correction with Sen2Cor

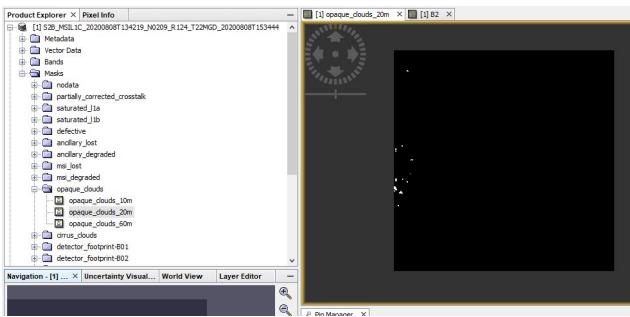
- In the folder where you have saved the image, unzip the Sentinel-image.
- Go to ‘Optical’ > ‘Thematic Land Processing’ > ‘Sen2Cor processor’ > ‘Sen2Cor-280’
- When you choose the source product, click on the ‘...’, browse to the image and navigate to the ‘MTD_MSIL1C.xml’ product.
- In the tab ‘processing parameters’, set the resolution to ‘ALL’
- The other processing parameters are by default taken from a combination of the image metadata (header file) and look-up tables. This is why you will normally use the default processing parameters. However, if you want to adjust these parameters, you can do that manually.
- Run Sen2Cor (!be patient, it will take a while to process the entire image.)
- Explore the outcome image (RGB). What differences do you see according to the original image?

Installing 'Sen2Cor' plugin

Possibly sen2cor isn't installed yet. To do this, go to ‘Tools’ > ‘Plugins’. During the first run, you'll get an error, after which an extra bundle will be installed.).

Intermezzo: Cloud Masks

The image contain clouds. This means that there are some blind pixels, which lack information on the reflectance of the earth's surface at the sensing time. This phenomenon is very common in tropical areas with a rainy season. It is possible that over the whole period of the rainy season, you will not be able to obtain images with a cloud cover of less than 90%. In such cases, Radar imaging can be useful, but are complexer. An introduction to radar imaging will be given later in these practicals.



Included in a Sentinel-2 image folder you can find some cloud masks at a resolution of 10m, 20m and 60m. These cloud masks enable the user to identify cloudy and cloud-free pixels. The masks include both dense clouds (opaque clouds) and cirrus clouds. These cloud masks are computed by a threshold algorithm. Below, the methods are described that identify the cloud pixels (for your information).

Identification of dense clouds

Dense clouds, also called opaque clouds, are characterised by a high reflectance in the blue spectral region (B2). The method used to identify dense cloud pixels is based on B2 reflectance threshold. To avoid false detection, mainly due to snow/cloud confusion, SWIR reflectance in B11 and B12 are also used. Snow and clouds both have a high reflectance in the blue. Cloud reflectance is high in the SWIR, whereas snow presents a low reflectance. Additional criteria based on B10 reflectance are added to avoid high altitude ice cloud and snow confusion (both having a low reflectance in the SWIR bands B11 and B12). At B10, there is a high atmospheric absorption band and only high altitude clouds are detected. However, this last criterion is only applied after a first detection of cloud pixel in the blue band where cirrus is transparent.

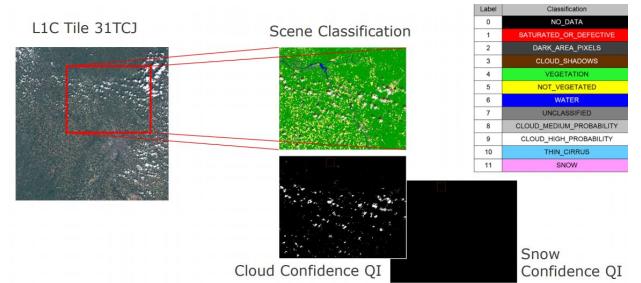
Identification of cirrus clouds

Cirrus clouds are thin, transparent or semi-transparent clouds, forming at high altitudes, approximately 6-7 km above the Earth's surface. The method of identifying cirrus cloud pixels from dense cloud pixel is based on two spectral criteria: (1) B10 corresponds to a high atmospheric absorption band: only high altitude clouds can be detected, (2) cirrus clouds, being semi-transparent, cannot be detected in the B2 blue band. A pixel with low reflectance in the B2 band and high reflectance in the B10 band has a good probability of being cirrus cloud but this is not a certainty. Some opaque clouds have a low reflectance in the blue and can be identified as cirrus cloud. To limit false detections (due to high reflectance in the blue or due to the fact that clouds are not spectrally registered), a filter using morphology-based operations is applied on both dense and cirrus cloud masks: (1) erosion, to remove isolated pixels, (2) dilatation, to fill the gap and extend clouds. If after morphology operations, a pixel is both dense and cirrus, the dense cloud mask prevails.

Sen2Cor scene classification

The Sen2Cor processor you've runned for the atmospheric correction from the level 1C to the level 2A product also contains a scene classification algorithm. This algorithm creates a scene classification, where pixels are classified in some broad classes:

Here, clouds are classified into 'cloud probability masks', which are in general more precise than the level 1C cloud masks.



Exercise: Visualize cloud masks

- Visualize the cloud masks.
- If you look at the cloud masks, you will see that these are not very precise. These cloud masks are useful for rough estimations. Later we will see alternative ways to identify cloud pixels more precisely.

Resampling

In order to display the other band combinations, some geometrical pre-processing is necessary. The bands have to be resampled to an equal resolution. The goal is to resample the image bands to 10m (you can take B2, B3, B4 or B8 as a reference band). This means that all other bands will be upsampled.

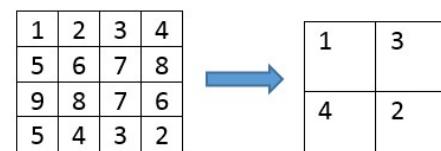
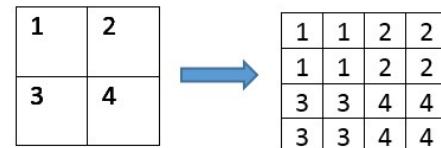


Image resampling scheme. Top: upsampling (nearest neighbor). Bottom: Downsampling (minimum).

Exercise: resampling

- In the product explorer, select the outcome image of Sen2Cor. Go to *Raster > Geometric operations > Resampling*.
- Select the 'Save as... BEAM-DIMAP' box. Browse to your directory. Choose a logical name for the target product.
- Resampling Parameters: Choose a reference band that has a resolution of 10m, or choose for a pixel resolution of 10m. Use an upsampling method of your choice (Read the help for more details on the different algorithms).
- Run *resampling*.
- Saving the images takes a lot of time. Again, be patient!

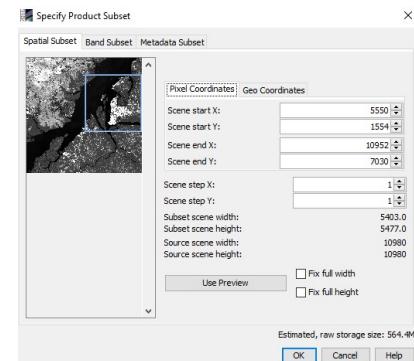
Image Subsetting

Processing an entire Sentinel-2 image takes a lot of processing capacity and time (as you probably have noted already). Therefore, you will now learn how to only process a small part of the image. You can choose to reduce the spatial extent of the image, or you can choose to reduce the amount of bands in the image, or a combination of both.

An important aspect is that creating a subset is only possible for bands that have the same size. Thus, this will only be possible **after resampling**.

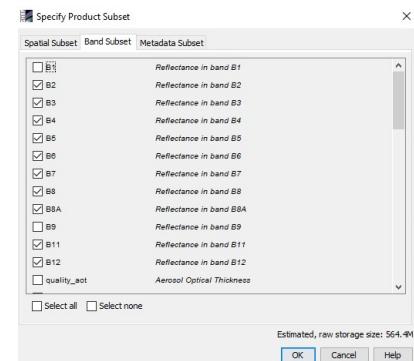
Exercise: subsetting an image

- Select the resampled image in the product explorer. Go to *Raster > Subset*.
- Select a spatial subset by choice (by adjusting the scene start and end). Make sure your spatial extent is substantially smaller than the original image.



Snap Subsetting screen.

- Select only following bands: [B2, B3, B4, B5, B6, B7, B8, B8A, B11, B12]
- You can see an estimation of the new required storage space.



Snap Subsetting screen.

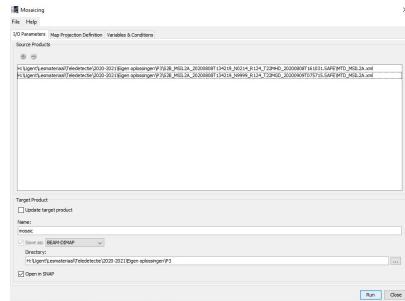
- Click OK
- Another option to make a subset is 'Spatial subset from view'. Zoom in on your image. Rightclick and select 'Spatial subset from view'.
- FYI: it is also possible to take a subset of an image, based on a vector layer.

Mosaicing

Mosaicing is the merging of several arbitrarily shaped images and often used to merge two neighbouring satellite images.

Exercise: mosaicing

- Download an image that is located next to the image you are already working with, dating from the same time as the original image was taken.
- You can download it directly in Level 2A, thus skipping the sen2cor atmospheric correction.
- Resample the image.
- Go to raster > Geometric operations > Mosaicing

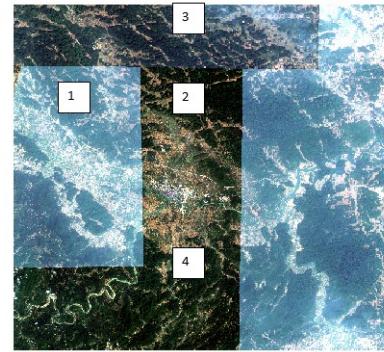


Snap mosaicing screen.

- Add the two source products.
- Choose the directory in which you want to save the mosaic image.
- In the Map Projection Definition you can choose the Coordinate Reference System (CRS). Choose for UTM/WGS84 (automatic)
- Choose for a resolution of 10m.
- The input products don't need to be orthorectified (because they already are).
- In the tab 'Variables and Conditions', click the - symbol.
- Select Band 2,3,4 and 8
- Run Mosaicing.
- Open the RGB-image of the product. Compare it to the two original images.

Extra: Examine the example of Landsat satellite image after merging below.

- What went wrong when mosaicing images 1 and 2?
- Why is there a colour difference in 2 and 3?
- Why is there no observable colour difference in 2 and 4?
- Have you any idea how to eliminate the colour difference between 2 and 3, given that neighbouring satellite images always partly overlap?



Landsat images mosaic

2.6 RGB colour composites 2

Displaying more band combinations

When you have performed an image resampling, open again the RGB image window in SNAP. You will notice that the list with possible band combinations is larger. Test some of the following band combinations and explore the colour differences. Which features are most clear on the following band combinations?

- Natural Colours: **4 3 2**
- False colour Infrared: **8 4 3**
- False colour Urban: **12 11 4**
- Agriculture: **11 8 2**
- Atmospheric penetration: **12 11 8a**
- Healthy vegetation: **8 11 2**
- Land/Water: **8 11 4**
- Natural Colours with Atmospheric Removal: **12 8 3**
- Shortwave Infrared: **12 8 4**
- Vegetation Analysis: **11 8 4**

3. P3- Intro tot Google Earth Engine

3.1 Doel van het practicum

Doel van dit practicum

In dit practicum maken we kennis met het Google Earth Engine. We behandelen hoe data kan opgezocht en gevisualiseerd worden en hoe data over een bepaalde tijdsperiode te aggregeren. Hiervoor maken we in enkele voorbeeldgebruik van zowel Sentinel-2 multispectrale beelden als Landsat-8 beelden.

Voorbereiding

Voor dit practicum heb je enkel een laptop/PC nodig waarop - bij voorkeur - Google Chrome op is geïnstalleerd.

Verder heb je een Google Earth Engine account nodig. Deze kun je gratis aanmaken via https://earthengine.google.com/new_signup/. Indien je dit nog niet hebt gedaan, dien je dit eerst te doen.

3.2 EarthEngine data catalog

De Earth Engine Data Catalog

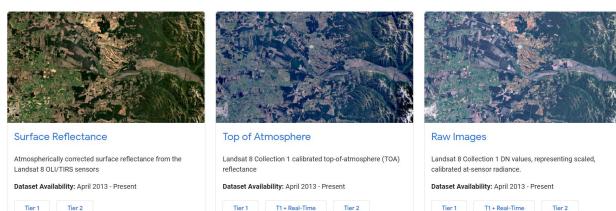
Om het aanbod aan aardobservatie-data in Google Earth Engine te bekijken en te doorzoeken, kan gebruik gemaakt worden van de Earth Engine Data Catalog: <https://developers.google.com/earth-engine/datasets>. Via deze catalogus kun je eenvoudig rasterdata allerhande opzoeken: satellietdata, weerdata, terreindata, populatiedata,... Via de catalogus vindt je ook de noodzakelijke code om de beeldsets in je script te importeren vinden.

In de komende voorbeeldoefening maken we gebruik van Landsat data. In Earth Engine zijn Landsatbeelden in eerste instantie opgedeeld op basis van Collecties:

- **Collection 2:** omvat de volledige Landsat collectie volgens de recent (April 2021) geoptimaliseerde preprocessing-keten en opslagstructuur van de USGS. Je kiest dus voor nieuwe scripts/oefeningen voor deze collectie.
- **Collection 1:** omvat de collectie van Landsatbeelden volgens de oude procedures. Deze collectie is nog steeds beschikbaar als overgingmaatregeling, maar wordt vanaf 1 januari 2022 volledig vervangen.

Een tweede opdeling van de landsatcollectie gebeurd op basis van de uitgevoerde correcties. Elk beeld is dus beschikbaar in 3 vormen:

- 'Surface reflectance': atmosferisch gecorrigeerde beelden: 'Bottom of Atmosphere'.
- 'Top-Of-Atmosphere': niet atmosferisch gecorrigerd, wel radiometrisch gecalibreerd.
- 'Raw Images': niet radiometrisch gecalibreerd.

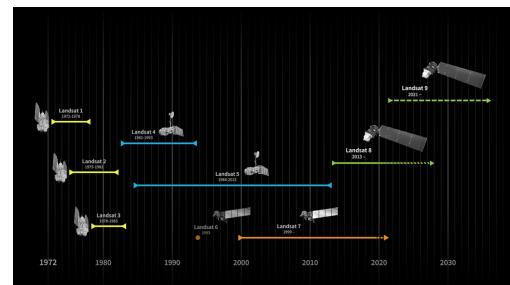


Een derde opdeling gebeurt ten slotte op basis van de beeldkwaliteit:

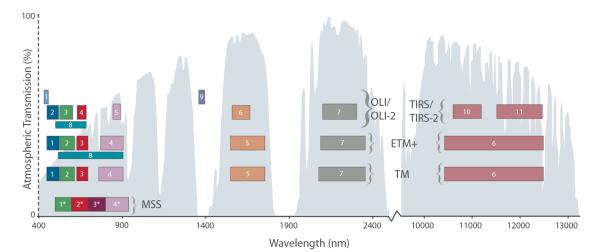
- **Tier 1:** De meest kwalitatieve beelden, geschikt voor tijdserie-analyse. De beelden zijn zowel geometrisch als radiometrisch kwalitatief goed bevonden volgens opgestelde standaarden.
- **Tier 2:** De beelden zijn geometrisch en/of radiometrisch minder kwalitatief bevonden, maar zijn wel nog inzetbaar voor bepaalde doeleinden.
- **Tier 1 + Real-Time:** De Tier-1 database uitgebreid met de meest recente data die nog niet kwalitatief gekeurd zijn en bijgevolg dus nog "fouten" kunnen bevatten. (Enkel als TOA beschikbaar)

Het Landsat programma

Landsat is het langst lopende aardobservatie satellietprogramma en is sinds 1972 continue operationeel. Het is een samenwerking tussen de *United States Geological Survey (USGS)* en de *NASA*. Op 27 september 2021 werd de Landsat 9 gelanceerd en is daarmee de meest recente Landsat-satelliet.

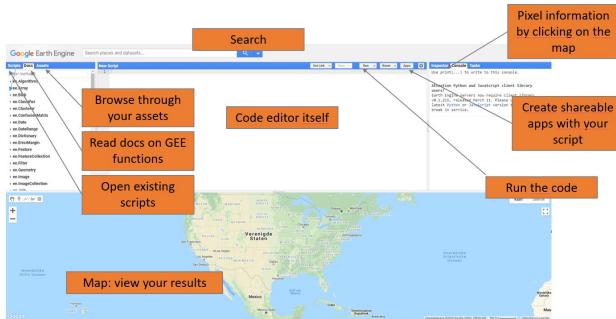


Onderstaande grafiek geeft een overzicht van de bandverdeling van de Landsatsatellieten. Huidig actieve Landsat-satellieten zijn Landsat 9 (OLI-2/TIRS-2) 8 (OLI/TIRS) en Landsat 7 (ETM+) (Bron: [NASA](#))



3.3 Introductie tot Earth Engine

De Google Earth Engine Interface



The Google Earth Engine code editor interface.

De interface van de Earth Engine code editor is op zich vrij simpel. Er kunnen 5 grote blokken onderscheiden worden:

1. Het linkerpaneel, met 3 tabs:
 - a. **Scripts**: je eigen bibliotheek met scripts, onder te verdelen in repositories, folders en scripts. Ook de scripts waar je schrijf- en leesrechten hebt kun je hierin terugvinden.
 - b. **Docs**: Bevat informatie over de functies die beschikbaar zijn in Earth Engine. Hier kun je snel de functionaliteiten en beschrijving van de input- en outputparameters terugvinden.
 - c. **Assets**: oplijsting met de 'assets' die je opgeladen/aangemaakt hebt in Earth Engine. Assets kunnen rasters of vectoren (met bijvoorbeeld trainingsdata of studiegebied).
2. De **code editor** zelf in het middenpaneel, waar je scripts kunt aanmaken/ bewerken, delen en opslaan.
3. Het rechterpaneel, met 3 tabs:
 - a. De **Console**: waar eventuele output of foutmeldingen naar geschreven worden. de 'print()'functie wordt steeds gebruikt om hier informatie te bekijken.
 - b. De **Inspector**: hiermee kun je op specifieke pixels in de 'map view' klikken, waarna de overeenkomstige pixelinformatie wordt gevisualiseerd.
 - c. De **Tasks**: bevat een oplijsting van de 'exports' die in het script werden aangemaakt (als je bijvoorbeeld een satellietbeeld naar je Google Drive wenst te sturen). Een export moet hier steeds nog manueel gestart worden.
4. De **Map View**: waar het beeldmateriaal wordt gevisualiseerd.
5. De **zoekfunctie** waarmee beeldmateriaal beschikbaar binnen de Google Cloud kan worden opgezocht.

Earth Engine code: Javascript 101

Google Earth Engine maakt voor zijn code-editor gebruik van Javascript als programmeertaal, maar om vertrouwd te geraken met GEE hoeft je geen Javascript-expert te worden. GEE gebruikt namelijk hoofdzakelijk eigen

'classes' en functionaliteiten, waardoor je slechts een basiskennis javascript nodig hebt.

Daarom starten we eerst met een spoedcursus Javascript, waarop we onze verdere 'Earth Engine'-magie kunnen bouwen.

"Hello World"

DE 'PRINT'-FUNCTIE

Zoals gebruikelijk is bij het leren van een programmeertaal, groeten we de wereld met ons eerste lijntje code.

- Open <https://code.earthengine.google.com/>, en voeg volgend lijntje toe aan het nieuwe script.

```
//Printen van Hello World
print('Hello World')
```

Klik daarna op 'Run'. Proficiat! Het eerste scriptje is geschreven. Hiermee heb je onmiddellijk ook een eerste uitermate handige functie gezien. De 'print'-functie kun je gebruiken om bepaalde informatie naar de Console te schrijven, zoals metadata,

Verder valt hieruit ook op te merken dat een dubbele voorwaartse **slash //** gebruikt wordt om notities te nemen binnen de code.

STRINGS

Proficiat! Het eerste scriptje is geschreven. Laat ons deze string nu onderbrengen in een variabele. In Javascript dient een variabele altijd geïnformeerd te worden met `var` statement. Indien je dit zou weglaten, zal je op een 'error' stoten.

```
//Aanmaken van de variabele 'aString'
var aString = 'Hello World'
print(aString)
```

Om het datatype van de variabele `aString` na te gaan, kun je dit oproepen met de functie 'typeof()'-statement:

```
// Type van de variabele aString naar de Console schrijven
print(typeof(aString))
```

FUNCTIES

Een functie in Google Earth Engine ziet er uit volgens volgende opbouw:

```
var functienaam = function(inputvariabelen) {
    //Hier de functie-bewerkingen
    //output = a + b
    return output
};
```

Bijvoorbeeld: een functie waarbij je een string naar keuze kunt groeten.

```
//Hello Function:
var hello_function = function(String) {
    var goeindag = 'Hello ' + String
    return goeindag
};

//Functie uitvoeren:
var hallo = hello_function('Boerekot');
print(hallo)
```

```
//Variabelen aangemaakt binnen de functie worden enkel daar gebruikt.  
// Ze gebruiken buiten de functie levert dus foutmeldingen op:  
print(goeindag)
```

LIJSTEN

Een lijst in Javascript wordt steeds opgegeven met [en]. Een lijstindex begint steeds vanaf '0', waarbij de eerste waarde dus op positie 0 staat.

```
var lievelingsnummers = [8, 6, 3, 27]  
print('Eerste lievelingsnummer in de lijst = ', lievelingsnummers[0])  
  
//Lijstelementen aanpassen  
var automerken = ["BMW", "Volkswagen", "Minerva"]  
automerken[2] = ["Opel"]  
  
print(automerken)
```

OBJECTEN

Een Object wordt aangegeven met '{ en '}'. Aan een object hangen steeds enkele variabelen die tot het object behoren.

```
//object  
var beelden = {  
  Sensor: "Sentinel 2",  
  Regios: ["Belgium", "France", "Vaticano"],  
  Aantalbeelden: 2,  
  1: "Ja"  
}
```

Om een eigenschap van een object op te roepen, wordt steeds een puntje '.' gebruikt: object.eigenschap. Indien we bijvoorbeeld de sensor van ons aangemaakte beeldmateriaal willen nagaan:

```
// Sensor bekijken  
print(beelden.Sensor)  
  
// Andere methode via haakjes []  
print('Regios: ', beelden['Regios'])
```

Javascript referenties

Het spreekt voor zich dat we hier slechts de basic-syntax van Javascript hebben aangehaald. Het is zeker niet nodig om eerst Javascript onder de knie te krijgen om in Google Earth Engine te kunnen werken. Aangezien Google Earth Engine slechts specifieke codes gebruikt, zul je doorheen de practica het nodige leren.

Specifieke Earth Engine objecten

`ee.Image`

Een `Image` is rasterdata bestaande uit één of meerdere banden, waarvan elke band een eigen naam, datatype, resolutie en projectie heeft. Een enkel Sentinel-2 beeld zoals in Practicum 3 gedownload werd, zal als één `Image` kunnen worden opgeslagen.

Om een `Image` in te laden en Earth Engine wordt gebruik gemaakt van `ee.Image`. In volgend hoofdstuk wordt dit geïllustreerd.

`ee.ImageCollection`

Een `ImageCollection` is een collectie van meerdere `Image`'s. De [Sentinel-2 MSI Level-2A collectie](#) bevat bijvoorbeeld het volledige aanbod aan atmosferisch gecorrigeerde Sentinel-2 beelden. Elke collectie bevat verdere informatie in de Data Catalog. Een collectie is steeds in een bepaalde volgorde gesorteerd zijn. Standaard is dit o.b.v. datum, maar aangepaste sorteringen zijn eveneens mogelijk, zoals we in een komende oefening gaan zien.

3.4 Satellietdata oproepen, filteren en visualiseren

Visualisatie van een enkelvoudig satellietbeeld

Laten we simpel starten met het afbeelden van een enkel rasterbeeld. In Practicum 2 gingen we te werk met een Sentinel-2 beeld van de Braziliaanse stad Belém uit 2021. Aangezien de volledige [Sentinel-bibliotheek](#) beschikbaar is binnen Earth Engine, kan dit beeld eenvoudig worden ingeladen. Bekijk hiervoor eerst de naam nog eens van je gedownload S2-bestand, bijvoorbeeld:

```
S2B_MSIL1C_20200808T134219_N0209_R124_T22MGD_20200808T153444.SAF
```

In Earth Engine is het vette gedeelte van de filenaam belangrijk. Dit wordt als volgt in earth-engine ingeladen, via 'ee.Image':

```
//Voorbeeld: Sentinel-2 beeld van voorig practicum
var S2_Belem = ee.Image('COPERNICUS/S2_SR/
20200808T134219_20200808T134214_T22MGD')
print(S2_Belem)

// Zoom in de Map-view in naar het beeld, met Zoom-factor 9
Map.centerObject(S2_Belem, 9);
```

Hiermee werd slechts een variabele aangemaakt die het beeld omvat. Om het beeld te visualiseren wordt gebruik gemaakt van de functie

`Map.addLayer()`:

```
//Visualiseren van het satellietbeeld
Map.addLayer(S2_Belem);
```

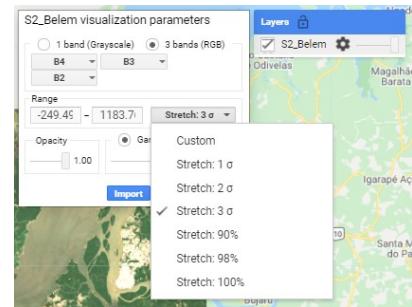
Bij het uitvoeren van bovenstaande code bekomen we een zwart vlak, niet bepaald de visualisatie die we wensen.



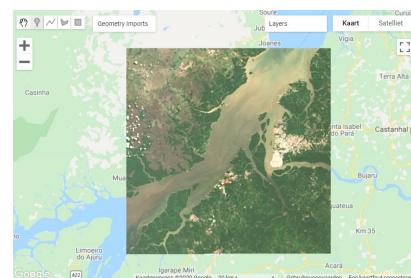
Bij het uitvoeren van bovenstaande code bekomen we een zwart vlak, niet bepaald de visualisatie die we wensen. Dit komt omdat we nog geen visualisatieparameters hebben aangegeven, waardoor de eerste 3 banden naar de rode, groene en blauwe band respectievelijk worden toegekend en de pixelrange zo groot is dat alle pixels een zwarte kleur krijgen. Om dit manueel aan te passen, zoek je je toegevoegde laag in 'Layers' in de Map-view. Klik op het tandwieltje. Een visualisatie-schermpje springt open. Pas de parameters aan, zodat je een normale kleurencomposit verkrijgt, met een stretch van 3 sigma en druk op 'Apply'. Een visueel beter resultaat wordt verkregen.

Image stretching

Verschillende stretch opties laten toe de histogrammen van het beeld te strechten om een betere visualisatie te krijgen. De stretch wordt uitgevoerd op basis van de huidige map view: ben je bijvoorbeeld ingezoomd om een stuk homogeen bos, wordt de stretch hierbinnen uitgevoerd.



Instellen van de visualisatieparameters kan via 'Layers' in de Map view.



Het is echter niet handig om steeds opnieuw de visualisatie handmatig in te stellen. Gelukkig kan deze ook als code geïmporteerd worden in GEE (klik op 'Import'). De visualisatieparameters worden toegevoegd in de Imports. Deze kunnen dan in de `Map.addLayer()`-functie worden meegeven tijdens het visualiseren.

```
var imagevisparams = B4, B3 and B2 from -363.1772660390337 to 1387.920064392943
bands: ["B4", "B3", "B2"]
gamma: 1
max: 1387.920064392943
min: -363.1772660390337
opacity: 1
```

In de code-editor zelf kunnen de visualisatieparameters eveneens gedefinieerd worden als een Object.

```
// Aanmaken van visualisatieparameters
var visualization = {
  min: 0,
  max: 3000,
  bands: ['B4', 'B3', 'B2'],
};

Map.centerObject(S2_Belem, 9);
Map.addLayer(S2_Belem, visualization, 'Belém_met_Vis');
```

Beeldcollecties zoeken en filteren

In voorgaande paragraaf visualiseerden we een Sentinel-2 beeld die we reeds hadden opgezocht waarvan wisten dat de kwaliteit goed zat én waarvan we de bestandsnaam reeds kenden. Het is natuurlijk niet handig om steeds een filenaam te moeten kennen om verder te kunnen werken in Earth

Engine. Daarmee zouden we ook de geweldige kracht van het programma om doorheen vele petabytes aan aardobservatiedata te zoeken onbenut laten.

In wat volgt gaan we op basis van een locatie op zoek gaan naar geschikte satellietbeelden, door het filteren van gehele beeldcollecties.

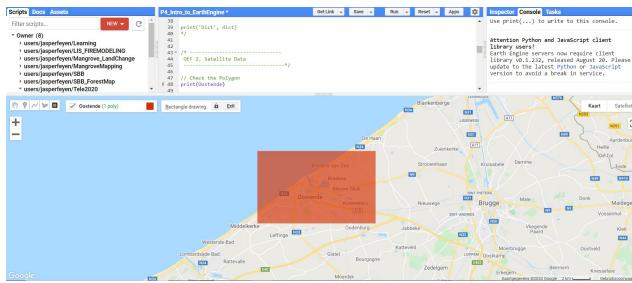
Region of Interest (ROI)

Starten doen we met het intekenen van een gewenste **Region Of Interest** (ROI) in de Map View. Een ROI is niets anders dan de afbakening van het studiegebied, waarbinnen we onze data wensen te verkrijgen.

Er kan rechtstreeks gezocht worden naar een locatie via de zoekbalk bovenaan of door het scrollen met de muis. Teken vervolgs een gewenste gebied in door gebruik te maken van de toolknoppen in de "Map View":



In dit voorbeeld kiezen we voor de Konigin der badsteden, Oostende, als studiegebied:



Automatisch wordt een nieuwe variabele aangemaakt onder de naam 'geometry', welke eenvoudig hernoemd kan worden naar een eenvoudig te gebruiken variabelenaam:

```
P4_Intro_to_EarthEngine *
  * Imports (1 entry) *
    * var Oostende: Polygon, 4 vertices
      type: Polygon
      coordinates: List (1 element)
      geodesic: false
```

Bekijk de eigenschappen van de polygoon door het naar de console te printen:

```
//Polygoon-informatie naar de console schrijven:
print(Oostende)
```

Inlezen en filteren van een ImageCollection

Voor deze oefening maken we als afwisseling gebruik van Landsat-8 beelden (zie ook het stukje omtrent de [Earth Engine data catalog](#)). De importeer-code kan gekopieerd worden uit de data catalog en ziet er als volgt uit:

```
var L8 = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2')
print('Grootte van de L8-collectie:', L8.size())
```

Hiermee verwijst de variabele 'L8' naar de volledige Landsat-8 collectie (surface reflectance). De '.size()'functie berekent het aantal beelden dat in deze collectie zijn begrepen. Een hele hoop, sinds de collectie alle L8-beelden van de volledige aarde omvat. Om hier verder mee te werken dient

de verzameling bijgevolg gefilterd te worden. Filteren kan op basis van de metadata:

```
//Filteren o.b.v. datum, locatie:
var L8 = L8.filterDate('2020-01-01', '2020-12-31') //Op basis van datum
.filterBounds(Oostende) //op basis van locatie (de AOI);

//Printen van de nieuwe grootte
print('L8 size na filtering', L8.size())

// Printen van de collectie voor inspectie
print('Filtered collection: ', L8)
```

De beelden in de collectie zijn standaard gesorteerd op datum, indien we dus het bovenste beeld eruit halen, zal dit het eerste Landsat-8 beeld zijn gemaakt in 2021. Met de functie `.first()`, halen we deze eruit. Print deze naar de console en bekijk het verschil met de de Imagecollectie.

```
// Krijg het eerste (standaard oudste) beeld uit de collectie:
var L8_first = L8.first()
print('Eerste Beeld:', L8_first)
```

In een volgende stap kunnen we dit beeld ook gaan visualiseren, met `javascript Map.addLayer()`. Ook nu kunnen we dit als een echte kleurencomposit visualiseren (voor Landsat 8 betekent dit dus B2 (blauw), B3 (groen) en B4 (rood)). De bandnamen voor Landsat-8 Surface Reflectance beelden in google earth engine werden aangepast naar SR_B*.

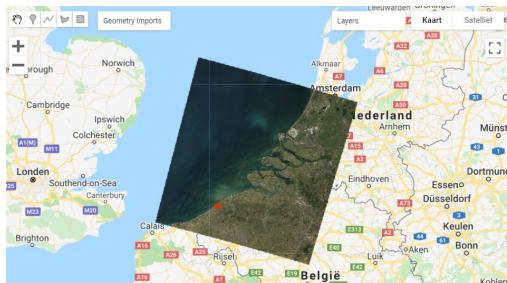
```
// Landsat-8 visualisatie instellen.
var trueColor = {
  bands: ['SR_B4', 'SR_B3', 'SR_B2'],
  min: 8000,
  max: 15000,
  gamma: 1.4,
};
Map.addLayer(L8_first, trueColor, 'L8_TrueColorComposite')
```



Eerste Landsat 8 beeld binnen de gefilterde collectie

Mogelijk is dit eerste beeld niet het meest ideale wat betreft de wolkbedekking, waardoor er weinig te zien valt. Laten we nu op zoek gaan naar het beeld met de laagste wolkbedekking binnen de collectie. Dit doen we in eerste instantie door de collectie te sorteren volgens het percentage cloudcover, wat standaard tot de metadata van een Landsatbeeld behoort. Bekijk het beeld. Wat valt je op? Wordt het volledige gebied bedekt?

```
//Sorteren van de collectie obv cloud cover
var L8_sortedCC = L8.sort('CLOUD_COVER',true);
Map.addLayer(L8_sortedCC.first(), trueColor, 'Least Cloud cover 2020')
```



Landsat 8-beeld met laagste wolkbedekking binnen de gefilterde collectie

Bekijk op welke dag de sensor dit beeld heeft genomen. Gebruik hiervoor de ‘inspector’ om de beeld eigenschappen verder te bekijken.

```
Inspector Console Tasks
> Point (2.8513, 51.1869) at 611m/px
> Pixels
-> Objects
  > Layer 1: Image LANDSAT/LC08/C01/T1_SR/LC08_199024_20200128 (12 ...
  > L8_TrueColorComposite: Image LANDSAT/LC08/C01/T1_SR/LC08_199024...
  > Least Cloud cover 2020: Image LANDSAT/LC08/C01/T1_SR/LC08_19902...
    type: Image
    id: LANDSAT/LC08/C01/T1_SR/LC08_199024_20200807
    version: 1598689035815422
    bands: List (12 elements)
    properties: Object (24 properties)
```

De inspector

Opdracht 3.1 - Valse kleurencomposiet voor Gent

Visualiseer in een nieuw script een valse kleurencomposit van een Sentinel-2 beeld (Tier 1, Surface Reflectance). Neem hierbij Gent als ROI, met een beeld uit 2019 met de laagste wolkbedekking.

Voor het sorteren van de wolkenbedekking, zoek je de gepaste eigenschap om op te sorteren. Deze kun je [hier](#) vinden.

Bewaar je script.

Oplossing

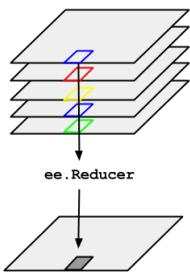
GEE script: <https://code.earthengine.google.com/0861ba83272bc305848ee2d113c4d3ef>

3.5 Reducing ImageCollections

Over Reducing

'Reducing' een beeld- of datacollectie in Google Earth Engine is het proces waarbij de beeldcollectie wordt geaggregeerd over tijd, ruimte, banden,

In dit proces wordt een beeldcomposiet aangemaakt van de beschikbare beelden in de collectie, waarbij per pixel een bepaalde vooropgestelde waarde wordt gekozen, zoals het min, max, gemiddelde, mediaan,... De collectie wordt als het ware 'gereduceerd' tot één enkel visualiseerbaar beeld.



Reducing an ImageCollection: principe.

Een voorbeeld: neem de eerste pixel van de gefilterde en gesorteerde Landsat 8 collectie `L8_sortedCC`. Visualiseer het resultaat. Wat valt je op? Is ditmaal het volledige gebied bedekt?

```
// Reducer over de L8_sortedCC collectie, waarbij steeds de eerste
pixel genomen wordt.
var L8_first_red = L8_sortedCC.reduce(ee.Reducer.first());

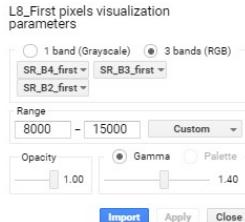
// Bekijk de eigenschappen van het gereduceerd beeld
print(L8_first_red)

var visParams_first = {
  bands: ['SR_B4_first', 'SR_B3_first', 'SR_B2_first'],
  min: 800,
  max: 15000,
  gamma: 1.4,
};

// Visualiseren als een normale kleurencomposit
Map.addLayer(L8_first_red, visParams_first, 'L8_First pixels')
```

Bandbenaming na reducing

Let ook, bij het aanroepen van `ee.reducer`, worden ook de banden hernoemd. Houd hier rekening mee bij het visualiseren. Het eventueel hernoemen van banden kan via de functie `.rename()`



Bandnamen bij de 'First'-gereduceerde collectie.

ee.Reducer.first() VS .first()

In een eerdere oefening namen we reeds het eerste beeld uit een hele collectie met de `.first()` functie. Dit is dus niet hetzelfde, gezien een reducer zicht niet beperkt tot één enkel beeld, maar de volledige collectie gaat reduceren.

Shortcut syntax

Bepaalde – veel gebruikte – reducers hebben ook een zogenaamde 'shortcut' syntax in Earth engine: `mean()`, `median()`, `min()` en `sum()` kunnen rechtstreeks gebruikt worden. Deze shortcut syntax zorgt ervoor dat een collectie eenvoudiger te reduceren is, zonder de hele `.reduce((ee.Reducer.mean()))` syntax te moeten gebruiken. Een voorbeeld:

```
// Een Median() Reducer over de Landsat-8 collectie
var L8_median = L8.reduce(ee.Reducer.median());

// Of via de short-syntax (geeft zelfde resultaat)
var L8_median = L8.median();

// Visualiseren
Map.addLayer(L8_median, trueColor, 'L8_median')
```



Voorbeeld mediane reducer over de L8_sortedCC-collectie

Opdracht 3.2

Probeer enkele van de Reducers uit op je Sentinel-2 collectie van Gent. Bewaar je script.

3.6 Cloud masking

Cloud Masking

Wolkbedekking is een grote barrière tijdens het analyseren en processen van (spectrale) satellietbeelden. Recente satellietdata komen veelal ook met automatische classificaties van de wolkbedekking, waardoor deze relatief eenvoudig uit het beeld verwijderd kunnen worden (zie ook P2: [cloud masking](#)).

Earth engine bevat naast deze standaard ‘cloud masks’ ook algoritmes om de wolken en wolkschaduw te verwijderen uit het beeld. Een keten van filter, masker en reduce strategiën kan de aanwezigheid van wolken minimaliseren. Volgende 3 stappen kunnen onderdeel zijn van deze keten:

1. Filteren op maximaal wolkenpercentage
2. Cloudmasking algoritme toepassen om wolken te verwijderen
3. Reduceren met een mediane reducer van de resterende collectie

1. Filteren van de ImageCollection op wolkbedekking

Een eerste optie is om een beeldcollectie te filteren (zie [voorgaand](#)) op wolkbedekking, waardoor enkel de beelden binnen een paalde range van wolkenpercentages worden weerhouden:

```
// Filteren van de Landsat 8 collectie tot beelden met maximaal 40%
wolkbedekking
L8 = L8.filterMetadata('CLOUD_COVER', 'less_than', 40)
```

FilterMetaData

Bij de voorgaande filters gebruikten we de redelijk eenvoudige functies `.filterBounds()` en `.filterDate()`, twee standaardfilters om op respectievelijk locatie (van een geometrie) en datum te filter.

De functie `.filterMetadata()` wordt gebruikt om te filteren op eerder welke Metadata-eigenschap dat een beeld bevat. Gebruik de [Docs](#) om het gebruik van deze functie verder te bekijken.

Beschikbare metadata kan steeds in de Earth Engine Catalog worden geraadpleegd. Voor Landsat 8: https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C02_T1_L2#image-properties.

2. Cloud Masks

Als 2e stap kunnen de overgebleven wolken/wolkschaduwen per beeld worden ‘geknip’ (cloudmask) door deze pixels naar een waarde 0 om te

zetten. Een standaard algoritme is bij de meeste beeldcollecties reeds gegeven als voorbeeld onderaan in de catalogus:

Voor Landsat 8 : Op basis van het FMASK-algoritme, waarbij pixels worden ingedeeld in verschillende wolken-klassen. https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C01_T1_SR !Momenteel nog niet in de recente Collectie 2. Een aangepaste cloudmaskfunctie kun je hieronder terugvinden.

Voor Sentinel 2 : https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2_SR).

Toepassen van de cloudmask voor Landsat 8:

```
// 1. Voeg een extra filter in obv wolkbedekking ('Cloud_cover')
L8 = L8.filterMetadata('CLOUD_COVER', 'less_than', 40)

// Definieren van CloudMask-functie (gegeven)

function maskL8sr(image) {
  // Gebaseerd op de QA-waarde, wat de uitkomst is van het FMASK
  // algoritme
  // QA-waarde 4 komt overeen met wolken
  var cloudShadowBitMask = (1 << 3);
  var cloudsBitMask = (1 << 4);
  // Get the pixel QA band.
  var qa = image.select('QA_PIXEL');

  // Both flags should be set to zero, indicating clear conditions.
  var mask = qa.bitwiseAnd(cloudShadowBitMask).eq(0)
    .and(qa.bitwiseAnd(cloudsBitMask).eq(0));
  return image.updateMask(mask);
}

// Pas de functie over elk beeld binnen de collectie toe met de .map-
// functie;
var L8_masked = L8.map(maskL8sr);

// Eerste beeld uit collectie zonder Cloudmask:
Map.addLayer(L8.first(), trueColor, 'L8 - 1e beeld - No Cloudmask')

// Eerste beeld uit collectie mét Cloudmask:
Map.addLayer(L8_masked.first(), trueColor, 'L8 - 1e beeld - met
Cloudmask')
```

Resulterend is een ImageCollectie met dezelfde beelden, maar waaruit de wolken gemaskerd zijn (mask toegepast). Echter kunnen wel sommige wolkenranden nog zichtbaar zijn, die de mask-functies hebben gemist.

OPDRACHT 3.3

Maak de L8_masked collectie aan, en neem hiervan een `.median()` reducer. Visualiseer dit beeld. Merk je een verbetering in vergelijking met de voorgaande `.median()`-gereduceerde beelden, zonder de `cloudmask`?

De `.map()`-functie

In bovenstaand voorbeeld werd de cloudmask-functie toegepast door gebruik te maken van `.map()`. D `.map()` wordt steeds gebruikt om een functie (die op afzonderlijke beelden dient toegepast te worden, zoals `maskL8sr`) toe te passen over elk beeld binnen een ImageCollection afzonderlijk. Het is als het ware een veel efficiënte manier dan de aangemaakte functie te itereren via een for-loop.

EXTRA: Sentinel-2 Cloud Masking optie 2 met S2Cloudless

Onderstaande Sentinel-2 cloudmask-procedure is ter aanvulling van bovenstaande principes. Het betreft een relatief rest open-source cloudmask-algoritme dat gebruik maakt van een externe detector. Resultaten zijn doorgaans accurater dan de "standaard" cloud-masking functie gegeven bij de Sentinel-2 collectie. Er kan voor komende oefeningen/ het praktisch examen met beide procedures gewerkt worden, dus zoals wat in [Cloud Masking met Cloudmasks](#) werd gebruikt.

Deze Sentinel-2 cloudmasking functie steunt op een afzonderlijke collectie: [Sentinel-2 Cloud Probability](#). Het wordt opgeroepen op basis van 2 functies:

`getS2_SR_CLOUD_PROBABILITY` : dat zowel de Sentinel-2 Surface Reflectance collectie oproept als de 'S2 cloud probability' collectie. Deze functie behoeft dus geen parameters en geeft een ImageCollectie als resultaat waarbij beide collecties met elkaar gemerged zijn per beeld.

`maskClouds` : dat op basis van de cloudprobability een cloudmask aanmaakt en toepast.

Om dus tot een collectie te komen waarbij de 'cloudmask' is toegepast, gebruik je dus onderstaande code:

```
var getS2_SR_CLOUD_PROBABILITY = function () {
  var innerJoined = ee.Join.inner().apply({
    primary: ee.ImageCollection("COPERNICUS/S2_SR"),
    secondary: ee.ImageCollection("COPERNICUS/S2_CLOUD_PROBABILITY"),
    condition: ee.Filter.equals({
      leftField: 'system:index',
      rightField: 'system:index'
    })
  });
  var mergeImageBands = function (joinResult) {
    return ee.Image(joinResult.get('primary'))
      .addBands(joinResult.get('secondary'));
  };
  var newCollection = innerJoined.map(mergeImageBands);
  return ee.ImageCollection(newCollection);
};

// Mask out clouds
var maskClouds = function(image) {
  var cloudProbabilityThreshold = 40;
  var cloudMask =
  image.select('probability').lt(cloudProbabilityThreshold);
  return image.updateMask(cloudMask);
};

var S2_coll = getS2_SR_CLOUD_PROBABILITY()
  .filterMetadata('CLOUDY_PIXEL_PERCENTAGE', 'less_than', 50) //Voorselectie obv wolken
  .filterDate('2019-08-01', '2019-10-30')
  .map(maskClouds)
```

Vervolgens kun je met `S2_coll` verder werken.

3.7 (Extra) Oefeningen

Onderstaande oefeningen kunnen gebruikt worden om P4 verder in te oefenen.

Oefening .1 - Area 51



Niveau: gemakkelijk

Stappen:

1. Open een nieuw script
2. Laad een Landsat-8 collectie in. Ga voor de beelden met de hoogste kwaliteit. Welke collectie kies je dan?
3. Filter je collectie op basis van volgende gevens:

- **Periode:** september 2019

- **Locatie:** Area 51. Hiervoor kun je volgende punt-locatie in je script gebruiken:

```
var Area51 = ee.Geometry.Point([-115.81441562461978,
37.2386297535804]);
```

- **Wolkbedekking:** minder of gelijk aan 10%

4. Hoeveel beelden blijven er nog over die aan bovenstaande criteria voldoen?
5. Van de resterende beeldencollectie neem je het eerste beeld (niet gesorteerd). Bekijk de metadata. Van welke datum is dit beeld afkomstig?
6. Visualiseer het beeld als een Normale Kleuren composiet, een Valse Kleurencomposiet. Je kunt zelf je visualisatieparameters definiëren door ze handmatig aan in te stellen.
7. Analyseer het volledige door jezelf volgende vragen te stellen:
 - Waar is er vegetatie te vinden? Waar is dit natuurlijk, waar onnatuurlijk?
 - Welke features herken je thv Area 51?
8. Het gebied geeft ook een ideale aanleiding om de 'Geology'-composiet eens uit te testen. De combinaties is als volgt: **RGB = SWIR2-SWIR1-BLUE**. Ga na welke Landsat-8 banden hiervoor benodigd zijn. (Maak gebruik van de [Landsat 8 bandentabel](#)). Deze composiet maakt visuele inspectie van grote structurele eigenschappen van gesteenten (zoals plooien en breuken) gemakkelijker.

Oplossing

Script: <https://code.earthengine.google.com/724bbe7796ebd4ff9657dc5f0c1baaa>

Oefening 4.2

Fires are raging in Bolivia, hitting particularly hard the Chiquitano dry forests of the country's southern Santa Cruz region. The fires are largely the result of intentional burning to convert forest to farmland. Sources say this practice has recently intensified after Bolivian president Evo Morales signed a decree in July expanding land demarcated for livestock production and the agribusiness sector to include Permanent Forest Production Lands in the regions of Beni and Santa Cruz. Satellite data indicate 2019 may be a banner year for forest loss, with tree cover loss alerts spiking in late August to levels more than double the average of previous years. Human communities are suffering due to the fires, with reports of smoke-caused illnesses and drinking water shortages. Meanwhile, biologists are worried about the plants and animals of the Chiquitano dry forests, many of which are unique, isolated and found nowhere else in the world.

GEGEVEN: Startscript met aanduiding van het studiegebied (ROI) <https://code.earthengine.google.com/2eba6fb83f6efb634a8286b52fd89bbc>

GEVRAAGD: Maak een 2 Sentinel-2 beelden aan, met volgende specificaties:

- Gebruik van de Sentinel-2 Top-Of-Atmosphere (TOA) collectie.
- Gefilterd op het studiegebied (ROI)
- Gefilterd op maximale wolkbedekking van 30%
- Wolken dienen worden te verwijderd uit de beelden.
- Gefilterd op de periodes:

```
- Beeld 1: het jaar 2017 (1 juni t.e.m 30 oktober)
- Beeld 2: het jaar 2019 (1 juni t.e.m 30 oktober))
```

- Voor elke pixel de mediane waarde van beide overgebleven collecties

SUBVRAAG 1.1 - Hoeveel beelden resteren er nog in beide collecties na filtering? **SUBVRAAG 1.2** - Visualiseer een Normale Kleurencomposiet voor beide jaren.

=====

Oefening 4.2

Coming Soon

4. P4 - Feature Extraction

4.1 Intro

Met de beelden die we uit de ``ImageCollections`` halen, gaan we statistieken halen, indices bereken en handelingen uitvoeren om informatie te benadrukken.

Als slot behandelen we methodieken om een Principale Componentenanalyse (PCA) uit te voeren op de (multidimensionale) beelden.

4.2 Spectrale indices

1. Spectral indices

Spectral indices zijn combinaties van 2 of meerdere spectrale banden die gebruikt worden om bepaalde features extra in de verf te zetten of ze te herberekken naar een relatieve schaal.

NDVI

De meest gebruikte index is de **Normalized Difference Vegetation Index (NDVI)**, en wordt berekend als:

$$[NDVI = \{ NIR - RED \} / (NIR + RED)]$$

Waarbij:

NIR = reflectie in het nabij-infrarode gebied van het spectrum (oftwel Near-Infrared)

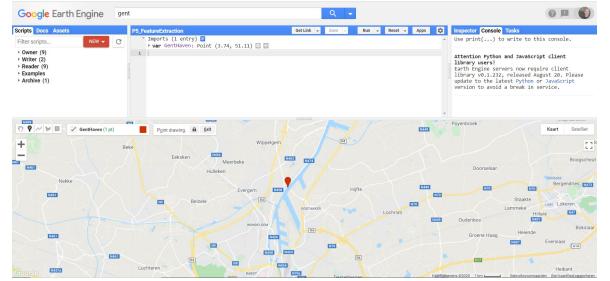
RED = reflectie in het rode gebied van het spectrum

De resulterende index krijgt waarden binnen tussen -1 en 1.

Volgens deze formule is de densiteit van vegetatie (NDVI) op een gegeven plaats in het beeld gelijk aan de verschillen in intensiteit van het gereflecteerde licht in het rood en infrarode deel van het spectrum, gedeeld door de soms van deze intensiteiten. Vegetatie absorbeert immers een groot deel van het zichtbare licht ten behoeve van de fotosynthese (dus lage Rood-reflectie), maar weerkaatst vrijwel al het infrarode licht (hoge IR-reflectiewaarde), waardoor de ndvi stijgt. Hoe denser de vegetatie, hoe hoger de ndvi. Andere lichamen, zoals water, observeren IR dan weer beter tot zeer goed, waardoor de ndvi daalt.

In Earth Engine kan de NDVI (en andere indices) op verschillende manieren berekend worden. We starten met de ‘meest conventionele’:

- We starten deze oefening door het aanmaken van een Sentinel-2 beeld in Gent. Maak een punt-geometrie aan ergens ter hoogte van de Gentse haven te Evergem.



- **Stap 1 - Importeren en visualisatie Sentinel-2 beeld:** Importeer de Sentinel-2 Surface Reflectance collection en zoek naar het beeld met de laagste wolkbedekking uit 2019 in de periode mei-juni (= de late lente). Bekijk op welke datum het beeld werd genomen. Visualiseer als een valse kleurencomposit.

- **Stap 2 -** We beschikken tevens over de grenzen van Gent in een vectorfile (een polygoon). Voeg deze toe aan je script. De vectorfile bevat alle gemeenten in België, waaruit we Gent op basis van de 'Name'-eigenschap filteren.

```
var Gent = ee.FeatureCollection('projects/ee-teledetectie-2021/assets/P5-FeatureExtraction/Belgium_municipalities').filter(ee.Filter.eq('Name', 'Gent'))
```

Je kunt je beeld nu verder begrenzen a.d.h.v. deze vectorfile. Gebruik hiervoor de functie `.clip(Gent)` die je toepast op je verkregen Sentinel-2 beeld.

Oplossing

```
//1. Importeren van de Sentinel-2 collectie.
var S2 = ee.ImageCollection("COPERNICUS/S2_SR");

//Filteren op basis van datum (lente 2019) + beeld met laagste
wolkenpercentage selecteren
var S2_Gent_Lente19 = S2.filterBounds(HavenGent)
  .filterDate('2019-03-20', '2019-06-30')
  .sort('CLOUDY_PIXEL_PERCENTAGE')
  .first();

print('Gent_Lente19:', S2_Gent_Lente19)

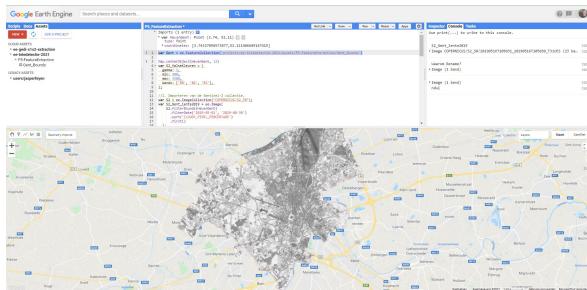
//Clippen naar Grenzen Gent
var Gent = ee.FeatureCollection('projects/ee-teledetectie-2021/assets/P5-FeatureExtraction/Gent_Bounds')
var S2_Gent_Lente19 = S2_Gent_Lente19.clip(Gent);

//Visualisatieparameters (of handmatig instellen)
var S2_ValseKleuren = {
  gamma: 2,
  min: 275,
  max: 2088,
  bands: ['B8', 'B4', 'B3'],
};

//Toevoegen aan Map
Map.addLayer(S2_Gent_Lente19, S2_ValseKleuren, 'Valse Kleuren lente
2019')
```

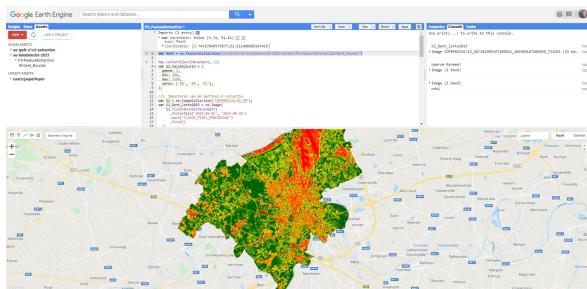
- Stap 3 - Toevoegen van de NDVI:** Een eerste methode om een NDVI aan te maken is via de ingebouwde `.normalizedDifference()` functie. Ga na welke Sentinel-2 banden je nodig hebt om de ndvi te berekenen. (Maak eventueel gebruik van de ‘Docs’-tab.)

```
//2. Aanmaken NDVI via NormalDifference()-functie. Vul de '?' in
var ndvi = S2_Gent_Lente19.normalizedDifference(['?', '?']).rename('NDVI');
Map.addLayer(ndvi, {}, 'ndvi_lente_2019') //Zonder
visualisatieparametres
```



- Een ndvi wordt meestal afgebeeld met een kleurenschema, zoals onderstaand voorbeeld:

```
// Met visualisatie
var ndviParams = {min: -1, max: 1, palette:
  ['red','yellow','darkgreen']>;
Map.addLayer(ndvi, ndviParams, 'ndvi_2019_vis');
```



Oefening: Connecteer de ndvi-waarden met de gepaste landbedekkingsklasse.

NDVI waarde	Landbedekking (Lente)
Negative values (< 0)	rocks, bare soil, clouds
Small values (0.1 or less)	shrubs and meadows
Moderate values (0.2 to 0.3)	temperate and tropical forests
Large values (0.6 to 0.8)	(clouds,) water and snow

Antwoord

NDVI waarde	Landbedekking (Lente)
Negative values (< 0)	clouds, water and snow
Small values (0.1 or less)	rocks and bare soil
Moderate values (0.2 to 0.3)	shrubs and meadows
Large values (0.6 to 0.8)	temperate and tropical forests

2. Bandbewerkingen

Bandbewerkingen kunnen worden gebruikt om een nieuw beeld aan te maken van de reeds bestaande banden. Het berekenen van indices zoals de NDVI, is al een treffend voorbeeld hiervan. Andere mogelijkheden zijn ratio's, het verschil van 2 beelden op 2 verschillende tijdstippen om mogelijke veranderingen visueel te benadrukken, ...

Er zijn 2 manieren om in Earth Engine een bewerking uit te voeren.

Optie 1 - Bewerkingen via operatoren

De basisoperators maken gebruik van 2 inputs: ofwel 2 beelden, ofwel beeld en 1 constante. De bewerkingen worden steeds per pixel en per band uitgevoerd. Voorbeeld van operatoren zijn `add()`, `subtract()` en `divide()`.

```
//NDVI berekenen aan de hand van bandwerkenden met operatoren
//Lange uitwerking: noodzakelijke banden eerst selecteren en
onderbrengen in een nieuwe variabele
var nir = S2_Gent_Lente19.select('B8');
var red = S2_Gent_Lente19.select('B4');
var ndvi2 = nir.subtract(red).divide(nir.add(red)).rename('NDVI');
Map.addLayer(ndvi2, ndviParams, 'ndvi via operatoren');
```

Het resultaat is logischerwijs identiek als de voorgaande ndvi-berekening.

Optie 2 - Bewerkingen via expressies

Het spreekt voor zich dat bovenstaande methode voor complex wiskundige bewerkingen niet handig is. Voor dergelijke bewerkingen wordt aangeraden om gebruik te maken van `image.expression()`, gezien de inputvariabelen hier afzonderlijk worden aangegeven, waardoor de

bewerking gemakkelijker wordt weergegeven en het coderen zo vereenvoudigd wordt. De expressie aanvaardt tevens ook constanten. Variabelen die binnen de expressie worden gebruikt, moeten steeds worden aangegeven, zoals in onderstaande NDVI-berekening;

```
//NDVI aan de hand van een expressie
var ndvi3 = SS2_Gent_Lente19.expression(
  '(NIR - RED) / (NIR + RED)', {
    'NIR': S2_Gent_Lente19.select('B8'),
    'RED': S2_Gent_Lente19.select('B4')
});
```

Ook hier is het resultaat hetzelfde als de vorige ndvi-berekeningen.

Gebruikte operators binnen expressies

Onderstaande tabel geeft de binnen de expressies gehanteerde operators weer (bron: [Earth Engine guide](#))

Operators for expression()		
Arithmetic	+ - * / % **	Add, Subtract, Multiply, Divide, Modulus, Exponent
Comparison	== != < > <= >=	Equal, Not Equal, Less Than, Greater than, etc.
Logical	&& ! ^	And, Or, Not, Xor
Ternary	? :	If then else

Andere indices

Naast de NDVI bestaan er nog een heleboel andere indices, elk met een eigen toepassing.

De Normalized Difference Water Index (NDWI)

Er bestaan 2 indices met de naam 'NDWI', beiden gerelateerd aan water:

1. De *NDWI* ontwikkeld door [Gao \(1996\)](#), als index voor het watergehalte van vegetatie. De Index is gebaseerd op de NIR (gevoelig voor vegetatie) en SWIR (gevoelig voor water) banden:

$$\text{[NDWI} = \{ \text{NIR} - \text{SWIR} \text{ over } \text{NIR} + \text{SWIR} \}.\text{]}$$

1. De *NDWI* ontwikkeld door [McFeeters \(1996\)](#), als index voor het verscherpen van verschillen in waterlichamen;

$$\text{[NDWI} = \{ \text{GREEN} - \text{NIR} \text{ over } \text{GREEN} + \text{NIR} \}.\text{]}$$

Opdracht NDWI

Test beide NDWI-indices uit op het Sentinel-beeld van de Gentse Haven en omstreken. Bekijk de verschillen. Kijk hiervoor zeker naar naburige natuurgebieden en waterplassen.

Opdrachten

Oef 4.1 - NDVI per seizoen

1. Maak een wolkenvrije beeldencollectie aan (gebruik een maximaal aan wolkbedekking van 30%) aan van de regio Durbuy . Gebruik onderstaande cloudmask-functie voor Sentinel-2:

```
function masks2clouds(image) {
  var qa = image.select('QA60');

  // Bits 10 and 11 are clouds and cirrus, respectively.
  var cloudBitMask = 1 << 10;
  var cirrusBitMask = 1 << 11;

  // Both flags should be set to zero, indicating clear conditions.
  var mask = qa.bitwiseAnd(cloudBitMask).eq(0)
    .and(qa.bitwiseAnd(cirrusBitMask).eq(0));

  return image.updateMask(mask);
}
```

2. Maak een functie aan om de NDVI te berekenen. Laat de functie dan los op de Imagecollectie via `.map()`.

- Maak aan de hand van de collectie 3 beelden aan met een `median()`-reducer, binnen volgende periodes:
 - Jan-Februari (Winter)
 - April-Mei (Lente)
 - Juli-Augustus (Zomer)

2. Visualiseer voor elk seizoen een Normale Kleurencomposit en een NDVI-beeld. Gebruik onderstaande visualisatieparameters bij het plotten:

```
//Visualisatieparameters instellen
var NormaleKleuren = {
  min: 0,
  max: 1500,
  bands: ['B4', 'B3', 'B2'],
};

var ndviParams = {
  min: 0,
  max: 1,
  bands: ['NDVI'],
  palette: ['red','yellow','darkgreen']
};
```

Oplossing Oefening NDVI per seizoen

Via deze link: <https://code.earthengine.google.com/138d65e9dd498c4d78df6fa879bdbbb7?noload=true>

EXTRA: Toevoegen van een legende

Om een overzichtelijke legende toe te voegen aan je kaart, kun je onderstaande code gebruiken. Hiermee voeg je een legendepaneel toe voor continue ndvi-data.

```
/*
 * LEGENDE TOEVOEGEN */
-----

// set position of panel
var legend = ui.Panel({
  style: {
    position: 'bottom-left',
    padding: '8px 15px'
  }
});

// Create legend title
var legendTitle = ui.Label({
  value: 'ndvi',
  style: {
    fontWeight: 'bold',
    fontSize: '18px',
    margin: '0 0 4px 0',
    padding: '0'
  }
});
```

```

// Add the title to the panel
legend.add(legendTitle);

// create the legend image
var lon = ee.Image.pixelLonLat().select('latitude');
var gradient = lon.multiply((ndviParams.max-ndviParams.min)/
100.0).add(ndviParams.min);
var legendImage = gradient.visualize(ndviParams);

// create text on top of legend
var panel = ui.Panel({
  widgets: [
    ui.Label(ndviParams['max'])
  ],
});

legend.add(panel);

// create thumbnail from the image
var thumbnail = ui.Thumbnail({
  image: legendImage,
  params: {bbox:'0,0,100', dimensions:'10x200'},
  style: {padding: '1px', position: 'bottom-center'}
});

// add the thumbnail to the legend
legend.add(thumbnail);

// create text on top of legend
var panel = ui.Panel({
  widgets: [
    ui.Label(ndviParams['min'])
  ],
});

legend.add(panel);

Map.add(legend);

```

4.3 Beeldstatistieken

Histogram

Een histogram is, binnen de remote sensing, een grafische weergave van de statistische frequentie van de pixelwaarden binnen een satellietbeeld. Deze pixelwaarden verspreiden zich tussen de waarden 0 en 255. In een histogram worden deze waarden op de x-as geplot, terwijl de overeenkomstige frequentie voor elke waarde binnen het beeld op de Y-as wordt geplot.

In wat volgt maken we een histogram aan van het aangemaakte ndvi-beeld. Hier voor is steeds een regio (dus polygoon) noodzakelijk, waarvoor een histogram wordt aangemaakt. In een eerste fase doen we dit voor het volledige weerhouden satellietbeeld. Op de geometrie van dit beeld naar earth engine te vertalen naar een polygoon maken we gebruik van de functie `.geometry()`.

```
var ROI = ndvi.geometry();
```

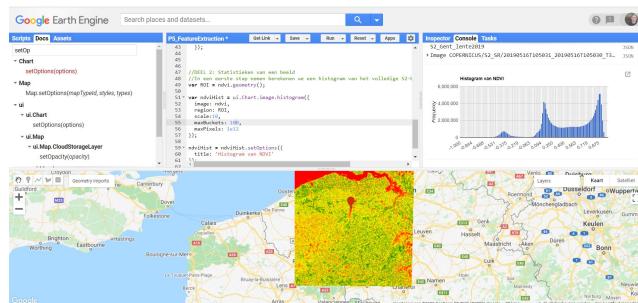
Bekijk de resulterende ROI eventueel door deze te mappen met `Map.addLayer(ROI)`. Uiteraard kun je ook handmatig een polygoon intekenen.

Om een histogram aan te maken wordt gebruik gemaakt van de `ui.Chart.image.histogram()`-functie binnen Earth Engine. Deze functie neemt volgende elementen aan (ook te checken via de 'Docs'): het beeld, de ROI, de schaal waarover de histogram wordt berekend, het aantal te plotten histogrambalkjes. Layout-opties worden afzonderlijk toegekend via `.setOptions()`.

```
//Initialiseren van een histogram via de ui.Chart functie
var ndviHist = ui.Chart.image.histogram({
  image: ndvi,
  region: ROI,
  scale: 10,
  maxBuckets: 50,
  maxPixels: 1e12
});

//Histogram updaten met stijlopties
ndviHist = ndviHist.setOptions({
  title: 'Histogram van NDVI in de Gentse Haven'
});

//Histogram schrijven naar de console
print(ndviHist)
```



Parameters meegeven aan een functie

Als je parameters meegeeft aan een functie in Javascript, kun je dit op 2 manieren doen:

1. De (noodzakelijke) parameters meegeven in volgorde aan de functie.

Bijvoorbeeld: `ui.Chart.image.histogram(ndvi, ROI, 10, 50)`. Hierbij is het noodzakelijk dat de parameters in juist volgorde worden meegegeven en er geen parameters worden overgeslagen.

2. Opstellen van een `dictionary`, waarbij de parameters explicet worden toegekend, zoals in het voorbeeld hierboven. Dit zorgt voor wat extra overzicht.

```
var ndviHist = ui.Chart.image.histogram({
  image: ndvi,
  region: ROI,
  scale: 10,
  maxBuckets: 50,
  maxPixels: 1e12
});
```

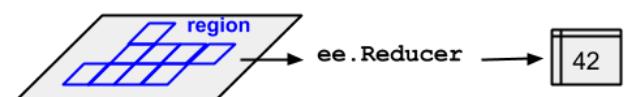
NDVI Histogram van Haven Gent/Vlaanderen

Als je kijkt naar het resulterend histogram, dan kun je enkele pieken opmerken: rond -0.3, rond 0.1 en rond 0.8. Verklaar de oorsprong van deze pieken.

Bandstatistieken

Om beeldstatistieken binnen een bepaalde ROI te berekenen binnen Earth Engine wordt gebruik gemaakt van `image.reduceRegion()`, of binnen meerdere regio's `image.reduceRegions()`. Het principe van deze 'beeldreducer' is hetzelfde als een reducer van een `ImageCollection`, met dat verschil dat de pixels binnen een regio van eenzelfde beeld worden gereduceerd, zoals in onderstaande figuur wordt geïllustreerd. Hiermee kan m.a.w. - binnen een bepaalde ROI - bepaalde statistieken berekend worden, zoals het minimum, gemiddelde pixelwaarde, maximum, mediane waarde, ...

Voorbeeld 1 - ReduceRegion



Illustratie van een Reducer toegepast op een beeld (Image) binnen een bepaalde regio.

```
//Statistieken van NDVI: Toepassen van een Reducer
var ndvi_mean = ndvi.reduceRegion({
  reducer: ee.Reducer.mean(),
  geometry: ROI,
  scale: 10,
  maxPixels: 1e12
});

print('Gemiddelde NDVI-waarde', ndvi_mean)
```

Voorbeeld 2 - ReduceRegions

In dit voorbeeld werken we met de [WorldClim-data](#). We berekenen per provincie de totale maandelijkse gemiddelde neerslag. In welke provincie valt er het meeste regen in de maand Januari?

- Start met het visualizeren van het beeld.
- Filter het beeld op basis van de Maand-eigenschap tot de maand Januari
- Gebruik de ReduceRegions-functie, gelijkaardig aan de reduceRegion(), maar ditmaal geef je aan volgens welke collectie (= featurecollectie) je de reductie wilt uitvoeren. Geef daarnaast ook de gebruiken reducer aan (Sum)

Oplossingen

```
var dataset = ee.ImageCollection('WORLDCLIM/V1/MONTHLY');
var meanPrec = dataset.select('prec');
var meanPrecVis = {
  min: 0.0,
  max: 100.0,
  palette: ['blue', 'purple', 'cyan', 'green', 'yellow', 'red'],
};

var meanPrecJan = meanPrec.filter(ee.Filter.eq('month',1)).first()
print(meanPrec)
// Clippen op basis van de provincies
var provinces = ee.FeatureCollection('projects/ee-teledetectie-2021/assets/PS-FeatureExtraction/Belgium_provinces')
var prec_Belgium = meanPrecJan.clip(provinces)
Map.addLayer(prec_Belgium, meanPrecVis, 'Mean Precipitation');

//Map.addLayer(provinces)
Map.centerObject(provinces)

//REDUCE REGIONS
var reduced = prec_Belgium.reduceRegions({
  'collection': provinces,
  'reducer':ee.Reducer.sum(),
  //'scale':927,
})

print(reduced)
```

4.4 Textuuranalyse

Textuuranalyse

Beeldtextuur kan worden gedefinieerd als '*de set van metrieken die berekend worden om bepaalde waargenomen textuureigenschappen van het beeld te kwantificeren*'. Het geeft informatie over de spatiale verspreiding van kleuren of intensiteiten binnen het beeld of een bepaalde regio. De spectrale reflectie van wolken en ijs kan bijvoorbeeld zeer gelijkend zijn, maar de textuur zeer verschillend. Textuur toevoegen als een extra inputband kan dus helpen bij het optimaliseren van een beeldclassificatie.

Textuur wordt steeds berekend binnen een bepaalde zone, ook wel *neighborhood* genoemd.

In volgende paragraaf, zullen we enkele textuur indices berekenen. We keren hiervoor even terug naar ons satellietbeeld uit Belém, gezien dit beeld geschikt is ter illustratie van textuur. Hier nog even de code om tot het beeld te komen:

```
//Inladen van het gekende Belém Sentinel-2 beeld
var S2_Belem = ee.Image('COPERNICUS/S2_SR/
20200808T134219_20200808T134214_T22MGD')

// Zoom in de Map-view in naar het beeld, met Zoom-factor 9
Map.centerObject(S2_Belem, 9);
Map.addLayer(S2_Belem, {min: 0, max: 3000, bands: ['B4', 'B3', 'B2']});
```

Standaard deviatie

De **Standaard Deviatie** (SD) berekent de spreiding van de pixelwaarde-distributie binnen de neighborhood.

```
//1) Standaard deviatie
// Compute standard deviation (SD) as texture van B8
var stdev = S2_Belem.select('B8').reduceNeighborhood({
  reducer: ee.Reducer.stdDev(),
  kernel: ee.Kernel.circle(7),
});
Map.addLayer(stdev, {min: 0, max: 2000}, 'SD of NDVI');
```

Grey Level Co-occurrence matrix

Voor het beschrijven van de textuur wordt gebruik gemaakt van de grey level co-occurrence matrix (GLCM) van Haralick et al. (1973). Het is een matrix dat weergeeft hoeveel verschillende combinaties van pixelgrijswaarden voorkomen in een bepaalde *neighborhood*. Aan de hand van de GLCM kunnen vervolgens verschillende textuurattributen berekend worden om de textuur te beschrijven.

Entropie

Een eerste voorbeeld is de entropie. De entropie van een beeld vertaalt de *randomness* van de intensiteit naar een index. Het kan worden gezien als een maat voor 'pixeldiversiteit' binnen een bepaalde zone (*neighborhood*). In dit voorbeeld wordt een kernel aangemaakt met radius 7m, waardoor de texturberekening dus steeds binnen een radius van 7m wordt berekend. Gezien een entropy-berekening enkel kan worden uitgevoerd op discrete waarden, dienen we het pixeltype eerst om te zetten naar een integer.

```
// Compute the gray-level co-occurrence matrix (GLCM), get contrast
var glcm = S2_Belem.select('B8').glcmTexture({size: 4});
var contrast = glcm.select('B8_contrast');

//Mappen van contrast: speel met de min, max waarden (via stretching)
Map.addLayer(contrast,
  {},
  'contrast');
var variance = glcm.select('B8_var')
Map.addLayer(variance,
  {},
  'variance');
var ent = glcm.select('B8_ent');
Map.addLayer(ent,
  {},
  'Entropy');
```

GLCM-textuur in Earth Engine

Je merkt wellicht dat de textuurafbeelding van verschillende textuurmaten uit de GLCM varieert naarmate je in- en uitzoomt. Dit is ligt aan het feit dat Earth Engine dit steeds per 'view window' uitrekent, waardoor de resultaten zo verschillend lijken. Echter, als je het beeld exporteert uit Earth Engine en bijvoorbeeld in SNAP openst, bekom je wel een visueel begrijpbaar resultaat.

De berekende banden kunnen in Earth Engine echter wel verder gebruikt worden voor classificatie.

EXTRA: Export-script

Hieronder kun je een voorbeeldscriptje vinden om een raster naar je Google Drive te exporteren. Na een 10-tal minuten kun je dit terug vinden op je persoonlijke Google Drive en downloaden naar je desktop.

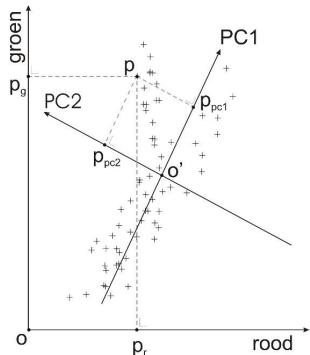
```
Export.image.toDrive({
  image: entropy,
  description: 'B8_Entropy',
  scale: 10, //RESOLUTIE
  folder: 'Sentinel_2_export',
  maxPixels: 1e12 //Vergroot de exportcapaciteit
});
```

4.5 Principale componenten analyse

De Principale componentenanalyse

Een **principale componentenanalyse (PCA)** is een lineaire transformatie waarbij de banden worden gezocht die het meeste informatie bevatten. Veelal wordt deze gebruikt om processingtijd van grote datasets te beperken, zoals vaak bij hyperspectrale beelden het geval is.

Het achterliggende principe is dat de verschillende spectrale banden van een sensor vaak informatie dragen die (gedeeltelijk) gecorreleerd zijn aan elkaar, waardoor er sprake is van onnodige of dubbele informatie. PCA zorgt voor een transformatie van de multispectrale data zodat nieuwe variabelen niet meer of amper met elkaar gecorreleerd zijn.



Principe van PCA voor een dataset met 2 banden (rood en groen) die met elkaar gecorreleerd zijn. Een PCA werd uitgevoerd, bestaande uit een translatie (van o naar o') en een rotatie zodoende dat de er 2 nieuwe variabelen ontstaan (PC1 en PC2).

De eerste principale component wordt dus gevonden door de richting in de data waar het meeste variatie te vinden is en bijgevolg de meeste data. De 2e principale component komt hier loodrecht op te staan. Het spreekt voor zich dit dit in een 2-dimensionale ruimte zoals in bovenstaande figuur eenvoudig is, maar dit kan eveneens worden toegepast in een meer-dimensionale dataset (zoals bijvoorbeeld een Sentinel-2 beeld met 12 banden).

Mathematisch gezien kan een PCA worden gezien als een zoektocht naar een translatie en rotatie van de multidimensionale dataset, waardoor de covariantiematrix van de dataset een diagonale matrix wordt na transformatie. In andere woorden, elke nieuwe variabele is gecorreleerd met zichzelf en niet meer met de andere variabele.

De eigenwaarden en eigenvectoren van de originele covariantiematrix worden dus berekend. Elke eigenwaarde met de geassocieerde eigenvector beschrijven dan de nieuwe principale component, waarbij de eigenvector de richting geeft van de nieuwe component en de eigenwaarde als een proxy dient voor de hoeveel informatie dat de component bevat. (voor de specifieke wiskundige details kan worden verwezen naar de cursus van Wiskunde 1 (Lineaire algebra) of naar [deze 5-minuten durende oprissings..](#)

Defening

In Earth Engine is het relatief eenvoudig om de PCA-berekeningen door te voeren.

1. Open een Nieuw Script: 'PC5-PCA'

2. We nemen opnieuw het Sentinel-2 beeld uit Belem. Aangezien enkel de banden B2, B3, B4, B5, B6, B8, B8A, B11 en B12 relevant zijn voor verdere analyse, weerhouden we enkel deze banden met de functie

```
.select()
```

```
// Importeren van het Sentinel-2 beeld van Belem
var S2_Belem = ee.Image('COPERNICUS/S2_SR/
20200808T134219_20200808T134214_T22MGD')

// Enkel banden relevant voor de PCA weerhouden in de Image
var bands = ['B2','B3','B4','B5','B6','B7','B8','B8A','B11','B12'];
var S2_Belem = S2_Belem.select(bands)
```

3. Opstellen van de PCA-functie (met bijhorende 'helper'-functie)

```
// Code adapted from https://developers.google.com/earth-engine/guides/arrays_eigen_analysis
var getPrincipalComponents = function(centered, scale, region) {
  // Collapse the bands of the image into a 1D array per pixel.
  var arrays = centered.toArray();

  // Berekenen van de covariantie a.d.h.v. een reduceRegion
  var covar = arrays.reduceRegion({
    reducer: ee.Reducer.centeredCovariance(),
    geometry: region,
    scale: scale,
    maxPixels: 1e9
  });
  print(covar, 'covariance')
  // Get the 'array' covariance result and cast to an array.
  // This represents the band-to-band covariance within the region.
  var covarArray = ee.Array(covar.get('array'));

  // Perform an eigen analysis and slice apart the values and vectors.
  var eigens = covarArray.eigen();
  print(eigens)
  // eigenValues is a P-length vector of Eigenvalues.
  var eigenValues = eigens.slice(1, 0, 1);

  print(eigenValues,'EigenValues')
  // This is a PxP matrix with eigenvectors in rows.
  var eigenVectors = eigens.slice(1, 1);
  print(eigenVectors, 'Eigenvectors')
  // Convert the array image to 2D arrays for matrix computations.
  var arrayImage = arrays.toArray(1);

  // Left multiply the image array by the matrix of eigenvectors.
  var principalComponents =
    ee.Image(eigenVectors).matrixMultiply(arrayImage);

  // Turn the square roots of the Eigenvalues into a P-band image.
  var sdImage = ee.Image(eigenValues.sqrt())
    .arrayProject([0]).arrayFlatten([getNewBandNames('sd')]);

  // Turn the PCs into a P-band image, normalized by SD.
  return principalComponents
    // Throw out an unneeded dimension, [] -> [].
    .arrayProject([0])
    // Make the one band array image a multi-band image, [] -> image.
    .arrayFlatten([getNewBandNames('pc')])
    // Normalize the PCs by their SDs.
    .divide(sdImage);
};

// De PCAfunctie heeft nog aan een helper-functie, dat een lijst met nieuwe bandnamen samenstelt

var getNewBandNames = function(prefix) {
  var seq = ee.List.sequence(1, bandNames.length());
  return seq.map(function(b) {
    return ee.String(prefix).cat(ee.Number(b).int());
  });
};
```

4. Toepassen van de PCA-functie. Hiervoor dienen eerst enkele inputparameters met informatie worden aangemaakt

```
/* DE PCA-functie is opgesteld. In wat volgt kunnen we deze
toepassen */
// We zetten enkele parameters klaar, die we later zullen gebruiken.
```

```

var scale = 30; // (30m om processingtijd wat te verminderen =>
sentinel2 kan tot 10m)
var bandNames = S2_Belem.bandNames(); // De bandnamen overnemen
naar een lijst
var region = S2_Belem.geometry(); // We nemen de regio van het
volledige beeld

```

5. De gebruikte PCA-functie heeft een '*mean centered*' beeld nodig. Dit betekent dat per band het gemiddelde de nieuwe '0-waarde' wordt, waarbij elke pixel een nieuwe waarde krijgt relatief aan deze 0-waarde. Dit zorgt voor een snellere covariantie-berekening.

```

// Mean center the data to enable a faster covariance reducer
// and an SD stretch of the principal components.

var meanDict = S2_Belem.reduceRegion({
  reducer: ee.Reducer.mean(),
  geometry: region,
  scale: scale,
  maxPixels: 1e12
});

var means = ee.Image.constant(meanDict.values(bandNames));
print(means, 'Gemiddeldes per band')
var centered = S2_Belem.subtract(means);
print(centered, 'Mean centered Image')

```

6. Vervolgens kunnen we de PCA-functie toepassen

```

// Uitvoeren van de PCA-functie => resultaat is beeld met PC's als
nieuwe banden
var pcaImage = getPrincipalComponents(centered, scale, region);

//Bekijken van pcaImage
//print(pcaImage)

```

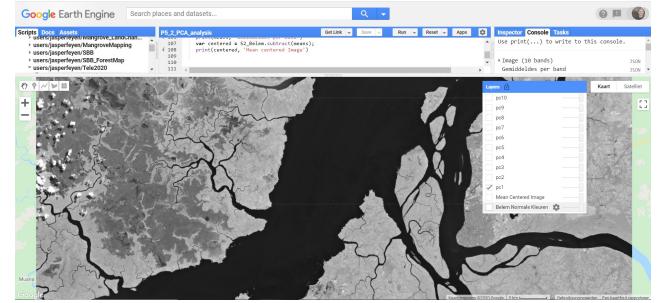
7. De resulterende `Image`, `pcaImage` is een beeld met als banden de berekende principale componenten. Via een `for`-lus kunnen we elk deze banden ook in 1x plotten naar de mapview.

```

// Plot each PC as a new layer
for (var i = 0; i < bandNames.length().getInfo(); i++) {
  var band = pcaImage.bandNames().get(i).getInfo();
  Map.addLayer(pcaImage.select([band]), {min: -2, max: 2}, band);
}

```

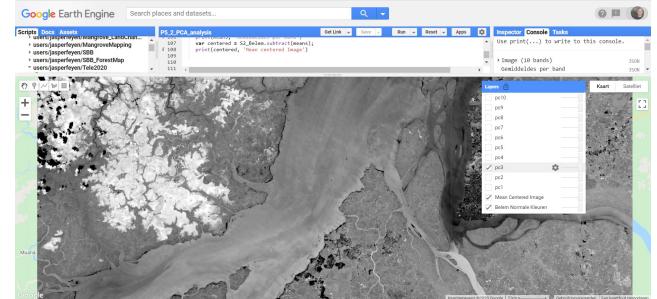
Resultaten:



Principale Component 1



Principale Component 2



Principale Component 3

4.6 Oefeningen

Onderstaande oefeningen kunnen gebruikt worden om de principes uit Practicum 4 verder in te oefenen.

Oefening 5.1 - Brand-index

Niveau: gemiddeld

In volgende oefening bekijken we een index die de gebieden aangestast door bosbranden belicht: de **Normalized Burn Ratio (NBR)**. Door de NBI te berekenen voor en na natuurbranden, kan een spatiale inschatting worden gemaakt van de ernst van beschadiging aan de omgeving.

De NBR wordt op een vergelijkbare manier als de NDVI berekend, waarbij het de rode band vervangen werd door een SWIR-band:

$\text{NBR} = \{\text{NIR} - \text{SWIR}\} / (\text{NIR} + \text{SWIR})$

In volgende oefening maken we een analyse van een grote bosbrand (genaamd 'Camp Fire') die op 8 november nabij Paradise in Californië woedde (info [hier](#)). Dit doen we aan de hand van Landsat-8 beelden.

Stappenplan

- Start met een nieuwe script. Kopieer en plak alvast de Cloud-mask functie voor Landsat-8 (zie vorig practicum):

```
function maskL8sr(image) {
  // Gebaseerd op de QA-waarde, wat de uitkomst is van het FMASK algoritme
  // QA-waarde 4 komt overeen met wolken
  var cloudShadowBitMask = (1 << 3);
  var cloudsBitMask = (1 << 4);
  // Get the pixel QA band.
  var qa = image.select('QA_PIXEL');

  // Both flags should be set to zero, indicating clear conditions.
  var mask = qa.bitwiseAnd(cloudShadowBitMask).eq(0)
    .and(qa.bitwiseAnd(cloudsBitMask).eq(0));
  return image.updateMask(mask);
}
```

- Filter de L8 (Surface Reflectance, Tier 1) collectie op basis van volgende ROI:

```
var ROI = ee.Geometry.Polygon(
  [[[-121.63966294798034, 39.877127888100304],
  [-121.63966294798034, 39.71622101257041],
  [-121.35470506223815, 39.71622101257041],
  [-121.35470506223815, 39.877127888100304]]], null, false);
```

- Maak een functie aan waarin de NBR-index wordt berekend **voor de Landsat 8 banden**. Gebruik de SWIR-2 band. ([tip](#)). Voeg de index toe aan de collectie met de `.map()`-functie.
- Filter de collectie verder met een maximale 'CLOUD_COVER' van 30% en pas de cloud mask toe.
- Maak nu 2 beelden aan binnen dezelfde periode van het jaar. Clip naar de ROI (met `.clip()`) en voeg ze toe als Normale Kleuren:
 - 1 Beeld voor de brand: een mediaan binnen de periode 1 mei 2018 - 30 juni 2018
 - 1 Beeld na de brand: een mediaan binnen de periode 1 mei 2019 - 30 juni 2019

- Visualiseer nu ook de NBR's voor beide beelden (gebruik de range van [0,1])

- Om een indicatie te krijgen van de brandernst, dient het verschil tussen beide NBR's te worden berekend. Voer hiervoor de som `NBR_voor - NBR_na`. Vermenigvuldig het resultaat met 1000 om een betere schaling te verkrijgen. Geef het de naam "NBR_verschil". Voeg de index ook als zwart-wild beeld toe aan het beeld, met een stretch tussen [-200, 1000]
- Visueel vallen er alvast heel wat zaken af te leiden. Echter voor verdere toepassing is het beter om de resulterende klassen op te delen, zoals in onderstaande tabel, opgesteld door de United States Geological Survey (USGS).

Severity Level	dNBR Range (scaled by 10^3)	dNBR Range (not scaled)
Enhanced Regrowth, high (post-fire)	-500 to -251	-0.500 to -0.251
Enhanced Regrowth, low (post-fire)	-250 to -101	-0.250 to -0.101
Unburned	-100 to +99	-0.100 to +0.99
Low Severity	+100 to +269	+0.100 to +0.269
Moderate-low Severity	+270 to +439	+0.270 to +0.439
Moderate-high Severity	+440 to +659	+0.440 to +0.659
High Severity	+660 to +1300	+0.660 to +1.300

Burn severity klassen, met bijhorende grenzen volgens het verschil in NBR-index (dNBR). Opgesteld door de USGS. (Bron: [UN-spider](#))

Om het NBR-verschil beeld om te zetten naar deze discrete klassen, dienen we het beeld te *reclassificeren*. In Earth Engine doen we dit volgens een `.where()`-functie. Pas onderstaande code aan in je script, om dit toe te passen. Eveneens werd de bijhorende palette meegegeven.

```
// Remap values. gt= greater than (>), lte = less than or equal (<=)
var Burn_severity = ee.Image(1).clip(ROI) //Initialiseert een leeg beeld
  .where(NBR_verschil.gt(-500).and(NBR_verschil.lte(-251)), 1)
  .where(NBR_verschil.gt(-250).and(NBR_verschil.lte(-101)), 2)
  .where(NBR_verschil.gt(-100).and(NBR_verschil.lte(99)), 3)
  .where(NBR_verschil.gt(100).and(NBR_verschil.lte(269)), 4)
  .where(NBR_verschil.gt(270).and(NBR_verschil.lte(439)), 5)
  .where(NBR_verschil.gt(440).and(NBR_verschil.lte(659)), 6)
  .where(NBR_verschil.gt(660).and(NBR_verschil.lte(1300)), 7)

var BurnSeverity_VIS = {bands:[ "constant"], palette:
  ["#b930c", "#98c825", "#00ff37", "#fff708", "#fbf716", "#ff7310", "#c20665"]}
Map.addLayer(Burn_severity,BurnSeverity_VIS, 'Burn Severity classes')
```

Opplossing

Script met volledige uitwerking: <https://code.earthengine.google.com/77ed985d48b3ed27bbb531f1e8a4f19f>

Oefening 5.2 - Herbebossing Regenwoud in Aimores, Brazilië

OPGELET!: deze oefening werd opgesteld met de Collectie-1 Landsat beelden: [Landsat-8](#), [Landsat-7](#). Dus maak deze best adhv deze collecties (er worden geen nieuwe beelden meer in deze collectie toegevoegd, de historische beelden zullen beschikbaar blijven).

Info: Examenopdracht 2019-2020.

Tip: Gebruik van NDVI als indicator van ontbossing

Context

Voor deze opdracht trekken we naar Aimores, in de Braziliaanse provincie Minas Gerais. Dit gebied bestond honderden jaren geleden uit uitgestrekt tropisch bos, het Atlantische woud, en bevat een buitengewoon grote biodiversiteit. In de 20e eeuw werd het leeuwendeel van dit gebied echter ontbost, waardoor naar schatting slechts 15% van het Atlantische woud is overgebleven.

In 1999 besloot een koppel om het heft in eigen handen te nemen door het starten van een herbebossingsproject in het gebied, met groot succes.

Gegeven:

- Afbakening van het projectgebied (als [Shape-file \(Studiegebied\)](#)).
- Tijdstip 1: 2000. Hiervoor dien je gebruik te maken van een [Landsat-7 beeld](#). OPMERKING: je kunt in dit geval nog gebruik maken van de Collectie 1 (zie link), ondanks deze als 'deprecated' gelabeld is.
- Tijdstip 2: 2020. Hiervoor maak je gebruik van een [Landsat-8 beeld](#), [Collectie 1](#).
- Cloudmask-functie voor L457 (Collectie 1):

```
var cloudMaskL457 = function(image) {
  var qa = image.select('pixel_qa');
  // If the cloud bit (5) is set and the cloud confidence (7) is high
  // or the cloud shadow bit is set (3), then it's a bad pixel.
  var cloud = qa.bitwiseAnd(1 << 5)
    .and(qa.bitwiseAnd(1 << 7))
    .or(qa.bitwiseAnd(1 << 3));
  // Remove edge pixels that don't occur in all bands
  var mask2 = image.mask().reduce(ee.Reducer.min());
  return image.updateMask(cloud.not()).updateMask(mask2);
};
```

- Cloudmask-functie voor L8 (Collectie 1):

```
function maskL8sr(image) {
  // Bits 3 and 5 are cloud shadow and cloud, respectively.
  var cloudShadowBitMask = (1 << 3);
  var cloudsBitMask = (1 << 5);
  // Get the pixel QA band.
  var qa = image.select('pixel_qa');
  // Both flags should be set to zero, indicating clear conditions.
  var mask = qa.bitwiseAnd(cloudShadowBitMask).eq(0)
    .and(qa.bitwiseAnd(cloudsBitMask).eq(0));
  return image.updateMask(mask);
}
```

Gevraagd:

Maak een beeld aan, waar voor elke pixel te zien is of er vegetatie is bijgekomen of verdwenen tussen 2000 en 2020 binnen het projectgebied.

Tips:

- Maak zelf een beeld aan per jaar aan waar de wolkbedekking ontbreekt of gemaskeerd is.
- Gebruik een gepaste index.
- De bandverdeling van Landsat 7/8 is verschillend! Houd hier rekening mee.

Oplossing

Script: <https://code.earthengine.google.com/2a3fec22e58b9d8cc2f508606b151726>

Oefening 5.3 - De Enhanced Vegetation Index (EVI)

De EVI index

De EVI is gelijkaardig aan de NDVI daar het gebruikt wordt om de aanwezigheid (of ‘greenness’) van vegetatie a.d.h.v. satellietbeelden te kwantificeren. Het werd ontwikkeld om aan enkele “limitaties” van de NDVI te voldoen:

- EVI is gevoeliger voor gebieden met hogere biomassa
- EVI reduceert de invloed van de atmosferische condities
- EVI corrigeert de ‘canopy background noise’, die bij NDVI voorkomt

De EVI is het meest gebruikte alternatief voor de NDVI.

De EVI wordt berekend als volgt:

$$\text{EVI} = G * \frac{\text{NIR} - \text{R}}{\text{NIR} + C1 * \text{RED} - C2 * \text{BLUE} + L}$$

(waarbij G : een versterkende constante, $C1, C2$: coëfficiënten en L : een ‘canopy background adjustment factor’)

Voor Sentinel 2, wordt deze formule: $\text{EVI}_{\text{S2}} = 2.5 * \frac{\text{B8} - \text{B4}}{\text{B8} + 6 * \text{B4} - 7.5 * \text{B2} + 1}$

Gevraagd: vergelijk de EVI met de NDVI voor het S2-beeld in Belém.

5. P5 - Beeldclassificatie

5.1 Practicum 5: Beeldclassificatie

Doelstelling

In dit practicum verdiepen we ons in het proces van de beeldclassificatie, waarbij manieren om remote sensing beelden te vertalen in landgebruikskaarten. Als voorbeeld nemen we een classificatie van mangroves.

Beeldclassificatie: introductie

Beeldclassificatie (Engels: '*Image Classification*') is de (complex) procedure waarbij een multiband rasterbeeld wordt ingedeeld in klassen om hieruit spatiale informatie te ontrekken. Het is wellicht de meest belangrijke handeling bij digitale beeldanalyse. Het resulterende beeld kan gebruikt worden om thematische kaarten aan te maken.

Gedurende de classificatie worden pixels ingedeeld in groepen, op basis van hun gemeenschappelijke karakteristieken. Er bestaan twee grote groepen van classificatiemethoden: *Supervised* of gesuperviseerde en *Unsupervised* of niet-gesuperviseerde classificatie.

In een **niet-gesuperviseerde** classificatie deelt de computer pixels in die spectraal gezien gelijkend zijn aan elkaar in een aantal klassen. Dit aantal kan op voorhand worden vastgelegd, of in de classificatieprocedure worden bepaald. Belangrijk is dat de analyse wordt uitgevoerd zonder dat de gebruiker hiervoor voorbeeldklassen of trainingsdata als input moet voorzien. De enige noodzakelijke input door de gebruiker betreft de keuze van het algoritme en eventueel het aantal gewenste resulterende klassen.

In een **gesuperviseerde** classificatie wordt de classifier *getrained* op basis van enkele voorbeeldklassen of *training data*. Nieuwe, ongeklasseerde data wordt dan aan het algoritme voorgelegd, die dit dan op basis van het getrainde algoritme gaat indelen. In een finale stap wordt dan een andere portie van gekende data, de *validatiedata*, gebruikt om de accuraatheid

(*accuracy*) van de classificatie kwantitatief te bepalen. In dit type classificatie is het aantal resulterende klassen eveneens bepaald op basis van het aantal klassen meegegeven tijdens het trainen van de data.

Supergeviseerde vs niet-gesuperviseerde classifiers

Niet-gesuperviseerde classificatie	Gesuperviseerde classificatie
+	+
Er is geen voorkennis nodig van het gebied.	De trainingsklassen komen overeen met de eigenlijke landbekking
Minimale input van menselijke fouten	Trainingdata is herbruikbaar, tenzij de landbedekking verandert.
Gemakkelijk uit te voeren	Resultaat is meteen bruikbaar, gezien de klassen gekend zijn.
-	-
De resulterende klassen komen niet per sé overeen met gewenste landbekkingklassen	De opgegeven klassen komen niet altijd overeen met de spectrale klassen
Spatiale relaties of spatiale context wordt niet meegenomen in de data	Zekerheid/consistentie is niet over alle klassen gelijk
Interpretatie kan moeilijk zijn	Verzamelen van (voldoende) trainingdata kost moeite, geld en tijd

5.2 CASE - Mangrove monitoring in Suriname

Over Mangroves

Defenitie en verspreiding

Het **Mangrove-ecosysteem** dat kan gezien worden als een soort moerasecosysteem, gedomineerd door mangrovebomen/bossen. Het Mangrove-ecosysteem is de zone die zich doorgaans bevindt het intergetijdenzone tussen zee en land in zowel tropische als subtropische gebieden. Binnen dit Mangrove-ecosysteem bevinden zich de Mangrovebomen zelf ('*True Mangroves*') en de geassocieerde soorten (*Mangrove associates*).

Hoewel er geen vastgelegde defenitie bestaat van mangrovebossen, kan deze van Tomilson (2016) worden gezien als de meest geaccepteerde:

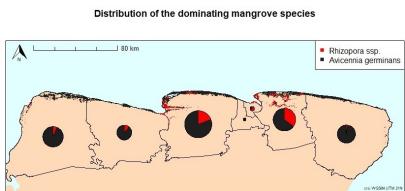
True Mangroves are plant species that:

1. Occur only in mangrove forests and are not found in terrestrial communities
2. Play a major role in the structure of the mangrove community, sometimes forming pure stands.
3. Have morphological specialisations to the mangrove environment
4. Have some mechanism for salt exclusion

In totaal bestaan er zo'n 55 mangrovesoorten die aan bovenstaande defenitie voldoen, waarvan de hoogste soortendiversiteit zich voordeut rond het westelijk deel van de Stille Oceaan. Globale inschattingen van dit ecosysteem variëren tussen 110.000 tot 240.000 km² (Giri, 2016).

Mangroves in Suriname

In Suriname bestaan er "slechts" 5 mangrovesoorten waarvan 2 soorten de grootste dominante vertonen: de zwarte mangrove of 'Parwa' (*Avicennia germinans*) die hoofdzakelijk langs de kustlijnen groeit in homogene bestanden en de rode mangrove of 'Mangro' (*Rhizophora mangle*), die landinwaarts en langs rivieroever terug te vinden is.



Verspreiding van de 2 dominerende Mangrovesoorten in 2018 (SBB, 2019).



Mangrove mapping

LU/LC klassen

In de voorbeeld voor beeldclassificatie gaan we zelf een LU/LC-kaart maken voor de kustzone ter hoogte van Paramaribo, de hoofdstad van Suriname. Hierin gaan we onderscheid maken tussen volgende klassen, zodat we Mangroves kunnen onderscheiden:

1. Mangrove
2. Ander bos
3. Water
4. Crops/grasvegetatie
5. Urban (stedelijke omgeving)
6. Naakte Bodem (soil)

Onderscheiden van Mangroves op satellietbeelden

Als laatste voorbereidende stap gaan we na op welke manier we Mangroves (visueel) kunnen onderscheiden van de andere klassen (met in het bijzonder aangrenzend inlands bos).

Healthy Vegetation Composite

Het aanmaken van verschillende beeldcomposieten helpt om visueel de verschillende landbedekkingsklassen te onderscheiden. Naas de Normale Kleurencomposit (gemakkelijkst interpreerbaar voor het menselijk oog) en de Valse Kleurencomposit (benadrukking verschil vegetatie - andere klassen) kan ook de *healthy vegetation composite* nuttig zijn.

Deze compositiet is gemaakt met volgende banden: RGB = NIR, SWIR en Blauw, wat met de resepctievelijke Sentinel-2 banden *B8, B11, B2* kan worden aangemaakt.

In deze compositiet kleurt 'gezonnde vegetatie' met toetsen van rood, oranje en geel. Doordat de SWIR-band wordt gebruikt, kunnen verschillende fasen van plantengroei en/of stress worden nagegaan. Mangrovebossen, kan op deze compositiet worden onderscheiden van andere bossen door hun specifieke watergebonden milieu.

Aanmaken van Beeldcomposieten

Maak een 'Healthy Vegetation'-compositiet aan van de Surinaamse kustlijn. Bekijk met welke stretch de mangrovezone het best tot uiting komt.

Mangrove Vegetation Index (MVI)

Doorheen de jaren werden ook enkele specifieke Mangroves indices ontwikkeld, om de detectie van Mangroves te vergemakkelijken. Een goed voorbeeld is de '**Mangrove Vegetation Index**' of **MVI** ([Baloley, 2020](#)), dat recentelijk werd ontwikkeld op basis van de Sentinel-2 banden B2, B8, B11, die de specifieke 'groenheid' en vochtomstandigheid die de mangrovebossen typeert in de verf zet, om zo een onderscheid te kunnen maken met de naburige terrestrische bossen en vegetaties.

De MVI wordt als volgt geformuleerd:

$$\text{MVI} = \{ \text{NIR} - \text{Green} \over \text{SWIR1} - \text{Green} \}$$

Wat zicht vertaald naar Sentinel-2 banden als:

$$\text{MVI} = \{ \text{B8} - \text{B3} \over \text{B11} - \text{B3} \}$$

5.3 Niet-gesuperviseerde classificatie

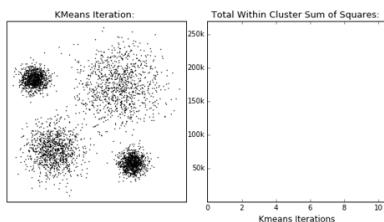
Niet-gesuperviseerde classificatie is de procedure waarbij pixel worden ingedeeld volgens spectraal gelijkende klassen zonder dat hierbij trainingsklassen nodig zijn. De resulterende klassen worden dan ook *spectrale klassen* genoemd.

Het is aan de gebruiker om de resulterende klassen te interpreteren en te labelen.

Er bestaan verschillende niet-gesuperviseerde algoritmen, maar de meest gekende groep is deze van de *clustering*. In een clusteranalyse worden pixels met gelijkende spectrale kenmerken tot dezelfde klasse gerekend.

K-Means Clustering

Een van de meest gebruikte cluster-algoritmen is de "K-means" clustering, waarbij de gebruiker op voorhand een aantal beginclusters opgeeft waarmee het algoritme arbitrair dat aantal clusters in de multi-dimensionale ruimte. Elke pixel wordt dan in een eerste fase toegekend tot de cluster waar de pixels zich gemiddeld het dichtst bij bevindt. Na deze eerste '*run*' worden de clusters herberekend, waarbij de variantie binnen elke cluster wordt geminimaliseerd. Hierna worden de pixels opnieuw toegekend tot de 'best passende' cluster. Deze procedure wordt herhaald (iteraties) totdat er zich geen significante verplaatsing van de clustercentra meer voordoet en de variantie binnen elke cluster dus ook niet meer significant daalt.



Principe van de K-means clustering in een 2-dimensionaal vlak. (Bron: dashee87.github.io)

Classificatie van de Surinaamse kustzone

In Earth Engine zit de clustering ook vervat in `ee.Clusterer`. We maken in volgend voorbeeld gebruik van de weka k-means cluster.

- Maak een nieuw script aan: P6_UnsupervisedClass.
- Zoals steeds filteren en reducen we een satellietbeeld, om met een wolkenvrije `image` verder te kunnen werken. We focussen ons in dit voorbeeld op de kustlijn van Suriname, ter hoogte van de hoofdstad: Paramaribo. We focussen hierbij op maanden binnen de grote droge tijd (Augustus - December), aangezien de wolkbedekking dan beperkter zou moeten zijn in vergelijking met de natte tijden.

Maak hiervoor eerst een polygoon aan met de locatie van Paramaribo.

Eventueel kun je hiervoor onderstaande code gebruiken:

```
var Paramaribo =
/* color: #d63000 */
/* shown: false */
/* displayProperties: [
{
  "type": "rectangle"
}
] */
ee.Geometry.Polygon(
  [[[-55.31615692285674, 6.000339363352038],
  [-55.31615692285674, 5.8043169248564865],
```

```
[-54.91446930078643, 5.8043169248564865],
[-54.91446930078643, 6.000339363352038]]], null, false);
```

Start Daarna met het aanmaken van het S2-beeld

```
//
// STAP 1 - Inladen en klaarzetten van S2-beeld. Met extra cloud-
masking
// -----
//Cloudprobability functie:
// Functie die nieuwe CloudProbability collectie samenvoegt met S2
(sen2cloudless)
// meer info: https://medium.com/sentinel-hub/cloud-masks-at-your-service-6e5b2cb2ce8a
var getS2_SR_CLOUD_PROBABILITY = function () {
  var innerJoined = ee.Join.inner().apply({
    primary: ee.ImageCollection("COPERNICUS/S2_SR"),
    secondary: ee.ImageCollection("COPERNICUS/
S2_CLOUD_PROBABILITY"),
    condition: ee.Filter.equals({
      leftField: 'system:index',
      rightField: 'system:index'
    })
  });
  var mergeImageBands = function (joinResult) {
    return ee.Image(joinResult.get('primary'))
      .addBands(joinResult.get('secondary'));
  };
  var newCollection = innerJoined.map(mergeImageBands);
  return ee.ImageCollection(newCollection);
}

// Mask out clouds
var maskClouds = function(image) {
  var cloudProbabilityThreshold = 40;
  var cloudMask =
image.select('probability').lt(cloudProbabilityThreshold);
  return image.updateMask(cloudMask);
};

//Aanmaken van een ImageCollection ter hoogte van de kustlijn met
mangroves en de hoofdstad Paramaribo, Suriname

var S2_coll = getS2_SR_CLOUD_PROBABILITY()
  .filterDate('2019-08-01','2019-10-30')// Filteren voor het
jaar 2020, droge tijd
  .filterMetadata('CLOUDY_PIXEL_PERCENTAGE','less_than',50) ////
Voorselectie obv wolken
  .map(maskClouds) //toepassen van de cloudmaskfunctie
  .filterBounds(Paramaribo); //collectie filteren obv de
Kustzonegeometrie

//Omzetten collectie naar een Image, door de.median() reducer toe te
passen. Hierna clippen we ook tot onze ROI
var S2_im = S2_coll.median()
  .clip(Paramaribo) //Bekijk de .clip-eigenschappen
in de Docs
```

De `.clip()` functie

De `.clip()`-functie wordt toegepast om het resulterende beeld bij te snijden naar de exacte grenzen van de aangemaakte polygoon (ROI). `.clip()` is enkel toepasbaar op beelden van het `image`-type, maar kan niet worden toegepast op een `"ImageCollection"`, gezien dit een te grote rekencapaciteit zou vergen. Daarom wordt de functie `.FilterBounds()` gebruikt, waarbij enkel gefilterd wordt op basis van een spatiale feature (punt, lijn of polygoon) maar geen beelden worden bijgesneden.

- De Earth Engine clusterer volgt het algoritme van [Weka](#), een open-source machine learning softwarepakket. In dit algoritme worden eerst pixels uit het beeld 'gesampled', waarop het k-means algoritme wordt losgelaten. Eenmaal het algoritme op punt is, wordt het toegepast op de rest van het beeld. M.a.w. wordt er een niet-gesuperviseerd model getrained op een willekeurige sample van pixels, die representatief wordt geacht voor de rest van het beeld. Het aantal te sampelen pixels moet dus voldoende groot gekozen worden, maar indien het te groot wordt, zal earth engine een foutmelding geven: `clusters: Layer error: Computed value is too large.` De maximale capaciteit van earth engine ligt bij ca. 1 miljoen pixels.

```
// Aanmaken "training" dataset.
var training = S2_im.sample({
  region: Paramaribo,
  scale: 10,
  numPixels: 5000
});
```

- Eenmaal de pixels geselecteerd zijn die gebruikt gaan worden voor het aanmaken van de k clusters, kan het 'K-means cluster-algoritme' worden getraind. Deze bevat enkele parameters die de gebruiker nog kan instellen:
 - nClusters:** het eerste en het enige verplicht aan te geven argument dat het aantal gewenste clusters aangeeft.
 - init:** de initialisatiemethode. Hiermee kan de manier waarop de initiële clustercentra worden gekozen. De *default*-instelling kiest ad random de beginpunten. Deze zullen we in deze oefening gebruiken.
 - distanceFuntion:** welke afstandsberekening moet worden toegepast: *Euclidisch* of *Manhattan*. De euclidische afstand is *default*.
 - maxIterations:** het maximale aantal iteraties, indien opgegeven. De clusteranalyse stopt na dit aantal iteraties.

```
// Clusterer opstellen en trainen. Opgeven van 5 klassen, de andere parameters laten we op default.
var Kmeans_cl = ee.Clusterer.wekaKMeans(5).train(training);

// Laat de cluster los op het volledige beeld
var classified = S2_im.cluster(Kmeans_cl);
```

* Finaal visualiseren we ook het resultaat. Om de bekomen klassen snel een afzonderlijke kleur te geven, maken we gebruik van `.randomVisualizer()`.

```
// Display the clusters with random colors.
Map.addLayer(classified.randomVisualizer(), {}, 'clusters');
```



Voorbeeldresultaat van de K-means clustering.

Opdrachten

1. Interpreteer de bekomen klassen en tracht ze te linken aan landbedekkingsklassen. Gebruik hiervoor ook een Normale Kleuren, Valse Kleuren en 'Healthy Vegetation' (RGB = B8,B11,B2) composiet.
2. Herhaal bovenstaande clustering enkele keren, met onderstaande parameters. Vergelijk de resultaten ook steeds met elkaar:
 - Je het aantal clusters optrekt naar 10.
 - Je de parameter `maxIterations` binen de `ee.ClustererwekaKMeans()` functie toevoegd. Test de waarden 1, 10 en 30
 - Je de factor `numPixels` vergroot (naar bv 10000).

5.4 Gesuperviseerde classificatie

Intro

Achtergrond

Gesuperviseerde classificatie is gebaseerd op een door de gebruiker opgestelde trainingdataset. Deze trainingsdata zijn representatief voor de specifieke gewenste klassen. Het gesuperviseerde algoritme gebruikt dan deze trainingsdata als referentie, om onbekende pixels te classificeren.

Een belangrijk element hierbij is dus voorkennis van het desbetreffende gebied. De *trainingsamples* die worden opgesteld worden ook wel *ground truth data* genoemd en komen overeen met gekende landbekking op locaties.

Er zijn verschillende opties om aan gronddata te komen:

1. Door middel van veldbezoek, waarbij **GPS-data** ter plaatse worden verzameld. Deze methode zorgt voor de meest kwalitatieve data en zekerheid, maar is vanzelfsprekend duur en arbeidintensief om uit te voeren.
2. Door gebruik van **referentiekaarten** of oudere beschikbaar kaartmateriaal of **beeldmateriaal van hogere resolutie**, bijvoorbeeld beelden bekomen gedurende een drone-campagne.
3. Door manuele **interpretatie van de satellietbeelden**, waarbij de gebruiker beschikt over enige expertkennis en voorgaande ervaring.

Een ander aspect is dat de trainingdata zo dicht mogelijk opgenomen wordt bij het tijdstip van opname van het gebruikte remote sensing beeld. Landbedekking kan immers snel veranderen: stadsontwikkeling, ontbossing, seizoенale impact, agrarische veranderingen, kusterosie, ...

In deze oefening van gesuperviseerde classificatie zul je zelf trainingsamples moeten aanmaken, want er zijn geen GPS-punten beschikbaar. Wel beschik je over een beknopt verslag van een veldcampagne, dat je een idee kan geven van de aanwezige landbekkingklassen en aangezien we ons beperken tot enkele brede klassen, zul je tevens gemakkelijk visueel trainingsamples kunnen aanmaken. Hiervoor is het aanmaken en analyseren van verschillende composieten aangewezen.

Beeldcomposieten aanmaken

- Start opnieuw met het aanmaken van een wolkenvrije dataset, dewelke kan overgenomen worden van vorige oefening. Belangrijk hier is ook om de gewenste banden te selecteren, die we tijdens de classificatie gaan aanmaken.

```
// -----
// STAP 1 - Inladen en klaarzetten van S2-beeld. Met extra cloud-
masking
// -----
//Cloudprobability functie:
// Functie die nieuwe CloudProbability collectie samenvoegt met S2
(sen2cloudless)
// meer info: https://medium.com/sentinel-hub/cloud-masks-at-your-
service-6e5b2cb2ce8a
```

```
var getS2_SR_CLOUD_PROBABILITY = function () {
  var innerJoined = ee.Join.inner().apply({
    primary: ee.ImageCollection("COPERNICUS/S2_SR"),
    secondary: ee.ImageCollection("COPERNICUS/
S2_CLOUD_PROBABILITY"),
    condition: ee.Filter.equals({
      leftField: 'system:index',
      rightField: 'system:index'
    })
  });
  var mergeImageBands = function (joinResult) {
    return ee.Image(joinResult.get('primary'))
      .addBands(joinResult.get('secondary'));
  };
  var newCollection = innerJoined.map(mergeImageBands);
  return ee.ImageCollection(newCollection);
}

// Mask out clouds
var maskClouds = function(image) {
  var cloudProbabilityThreshold = 40;
  var cloudMask =
image.select('probability').lt(cloudProbabilityThreshold);
  return image.updateMask(cloudMask);
};

//Aanmaken van een ImageCollection ter hoogte van Mangroves
Paramaribo, Suriname
var S2_coll = getS2_SR_CLOUD_PROBABILITY()
  .filterDate('2019-08-01','2019-10-30')// Filteren voor het
jaar 2020, droge tijd
  .filterMetadata('CLOUDY_PIXEL_PERCENTAGE','less_than', 50) //Voorselectie obv wolken
  .map(maskClouds) //toepassen van de cloudmaskfunctie
  .filterBounds(ROI); //collectie filteren obv de
Kustzonegeometrie
print(S2_coll)

//Omzetten collectie naar een Image, door .median() te nemen. Hierna
clippen we ook tot onze ROI
//Ook selecteren we de banden waarmee we verder willen werken
var bands = ['B2','B3','B4','B5','B6','B7','B8','B8A','B11','B12'];

var S2_im = S2_coll.median()
  .select(bands)
  .clip(ROI) //Bekijk de .clip-eigenschappen in de
Docs

Map.centerObject(S2_im, 11)
Map.addLayer(S2_im,{min:50,max:
1800,bands:'B4,B3,B2'},'NormaleKleuren_2020',0)
Map.addLayer(S2_im,{min:700,max:
4500,bands:'B8,B4,B3'},'FalseKleuren_2020',0)
Map.addLayer(S2_im,{min:500,max:
4000,bands:'B8,B11,B2'},'Healthy_Vegetation_2020',0)
```

Trainingsamples aanmaken

"Whereas the actual classification of multispectral image data is a highly automated process, assembling the training data needed for classification is anything but automatic. In many ways, the training effort required in supervised classification is both an art and a science" Lillesand & Kiefer (pg 544)

- Nadat het wolkenvrije Sentinel-2 beeld is ingeladen, kunnen we deze gebruiken om enkele representatieve *samples* te verzamelen van enkele landbedekkingklassen waar we in geïnteresseerd zijn. Er zijn 2 manieren om trainingsdata in Earth Engine op te laden:
 - a. Door ze in te tekenen als polygonen binnen per klasse, zoals we in komend voorbeeld zullen toepassen.
 - b. Door eerder ingetekende trainingssamples of GPS-punten op te laden als een 'Asset'. Dit kunnen *shapefiles*, of *.csv*-bestanden zijn.
- Hover met je muis over de '*Geometry Imports*' box, dat zich naast de geometrietools bevindt. Klik op '*'+new layer'.
- Elke gewenste landbedekkingklasse dient als een afzonderlijke laag te worden aangemaakt. Laat ons bijvoorbeeld starten met de eenvoudigste klasse 'water'. Zoom in op het beeld en teken polygonen in over oppervlaktes waar je zeker van bent dat het waterlichamen betreft. Het is goed hierin te variëren binnen verschillende types van zowel zee als rivieren en andere waterlichamen. Teken ca. 10 polygonen in per klasse. Neem hiervoor zeker je tijd, gezien het belangrijk is dit zeer precies te doen. De kwaliteit van de inputdata bepaalt tevens de kwaliteit van de classificatie, oftewel *Garbage in = Garbage out*. Als je een fout gemaakt heb, kun je even op 'exit' duwen en de laats ingetekende polygoon verwijderen.

```
| class | Landbekkingsklasse | :----:|-----:| 1 | Mangrove | 2 |
OtherForest | 3 | Water | 4 | Crop | 5 | Urban | 6 | BareSoil |
```



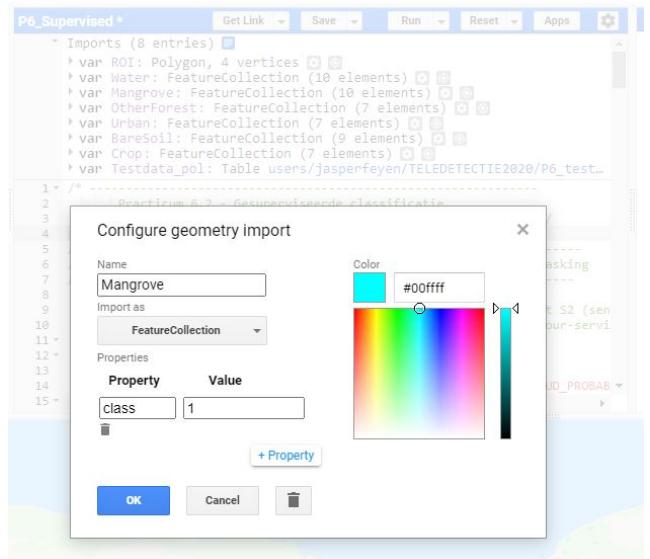
Polygonen vs puntdata als traingdata

Trainingdata kan bestaan uit puntdata of trainingdata. Beiden hebben hun voor- en nadeel.

Puntdata gebruiken als inputdata kan bijvoorbeeld als je beschikt over een grote set GPS-veldpunten van locaties waar je exact weet tot welke landbedekkingsklasse een pixel hoort. Deze pixel wordt dan als referentie aanschouwt. Deze zekerheid van de trainingsdata is dus groot. Een nadeel bij pixels is dat het minder de variëteit van de pixels binnen de klasse opneemt en de totale set aan trainingspixels beperkt blijft.

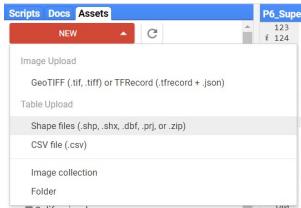
Polygonen worden gebruikt om gebieden in te tekenen voor een bepaalde klasse. In earth engine wordt elke pixel binnen deze polygoon dan gebruikt als inputdata. Dit zorgt ervoor dat de trainingsset groter wordt en de variatie binnen een bepaalde klasse beter wordt omvat. Een nadeel is echter dat zo ook foute pixels kunnen worden meegenomen door het onzorgvuldig intekenen van de polygoon.

- Eenmaal je klaar bent met het intekenen van een klasse, kun je deze import configureren. Klink hiervoor op het tandwiel naast de klasse. Geef het een gepaste naam. Daarna verander je de 'Import as' van *geometry* naar type *FeatureCollection*. Voeg daarna een *property* toe met de naam 'class', door te klikken op 'Add property'. De eerste klasse geef je waarde 1, de 2e 2, Zorg er wel voor dat je goed weet welke waarde je aan welke klasse geeft. Neem hierbij eventueel de *class*-nummering over van bovenstaande tabel.



- Herhaal dit voor elke klasse. Uiteindelijk verschijnt elke klasse als een 'FeatureCollection' bij de imports-lijst in je script:





Voorbeeld trainingsamples voor Mangrove

- Nu de 5 klassen aangemaakt zijn, dienen we ze samen te vatten als een complete trainingsset in earth engine; een gezamenlijke `FeatureCollection`, waarbij de 'class'-*property* wordt overgenomen.

```
//2. Trainingssamples samenvoegen tot 1 ``FeatureCollection``

var classNames =
Mangrove.merge(OtherForest).merge(Water).merge(Crop).merge(Urban).merge(BareSoil);
print(classNames)
```



De trainingsdata aanmaken

Nu hebben we reeds een `FeatureCollection` met trainingspolygone, maar deze zeggen nog niks over de trainingspixels. In een volgende stap, extraheren we de trainingspixels per band uit het Sentinel-2 beeld op basis van de aangemaakte polygonen. Dit doen we met de `sampleRegions()` functie. Deze functie extraheert alle pixels binnen opgegeven polygonen, en schrijft elke pixel afzonderlijk naar een nieuw `Feature` binnen een `FeatureCollection`, diewelke ook de class-*property* mee krijgen. Afhankelijk van de grootte van de polygonen, kan deze dataset dus zeer lijvig worden.

```
//Trainingspixels extraheren naar featurecollection = trainingsdata
var traindata = S2_im.sampleRegions({
  collection: classNames, //De trainingspolygone
  properties: ['class'], //Dit neemt de gewenste eigenschappen van de
  collection over
  scale: 10,
  tileScale: 4
});
print('Aantal trainingspixels: ',traindata.size());
//print(traindata) //Niet doen
print(traindata.first()) //Eerste waarde bekijken,
```

Layer error: Computed value is too Large

Mogelijk heb je bij het gebruik van `sampleRegions()` al onderstaande foutmelding gehad:

Layer error: Computed value is too large.

Dit krijg je dus als je berekening de maximum-memorycapaciteit overschrijdt. Dit is eenvoudig op te lossen door het toevoegen van een extra parameter `tileScale` aan de `sampleRegions()`-functie, zoals in bovenstaand voorbeeld werd gedaan. Door het verhogen van de `tileScale` zal Earth Engine de opdracht in meerdere stukjes indelen, waardoor de berekening minder gemakkelijk 'out of memory' zal lopen. De bewerking zal hierdoor wel meer tijd in beslag nemen, dus verhoog `tileScale` slechts gelijdelijk.

Zie ook de 'Docs' van `sampleRegions()`

De classifier trainen

In een volgende stap maken we een classificatiemodel aan en trainen we deze op basis van de traindata. Er bestaan verschillende mogelijke classifiers en 'machine learning'-algoritmen. In Google Earth Engine zitten deze beschikbaar in de `ee.Classifier`-groep. We gebruiken er 3, waarna we kijken welke tot de meest accurate classificatie leidt o.b.v. de validatiedata.

Minimum Distance Classifier

In een eerste instantie dienen we de classifier te trainen, op basis van de opgestelde trainingsdata. We dienen ook aan te geven welke van de `properties` binnen de trainingssamples de 'class'-bevat, en welke `properties` gebruikt moeten worden om mee te classificeren (de banden).

```
//4. De classifiers trainen en toepassen
// A. Minimum Distance classifier (gebruik van default-waarde
//euclidische afstand)
var MinDist = ee.Classifier.minimumDistance().train({
  features: traindata,
  classProperty: 'class',
  inputProperties: bands //verwijzing naar de eerder aangemaakte
bands-lijst
});
```

CART classifier

```
// B. CART classifier
var Cart = ee.Classifier.smileCart().train({
  features: traindata,
```

```

    classProperty: 'class',
    inputProperties: bands //verwijzing naar de eerder aangemaakte
bands-lijst
  );
}

```

Random Forest classifier

```

// C. Random Forest
var RandomForest = ee.Classifier.smileRandomForest({
  numberTress: 60
}).train({
  features: traindata,
  classProperty: 'class',
  inputProperties: bands //verwijzing naar de eerder aangemaakte
bands-lijst
});

```

Beeld classificeren en visualiseren

Eenmaal de classifier(s) opgesteld zijn, kunnen ze worden toegepast op het volledige S2-beeld. Elke pixel wordt dus toegekend tot een klasse, op basis van de kennis opgedaan uit de trainingsdata.

```

// 5. Classifiers toepassen

//MinimumDistance
var classified_MD = S2_im.classify(MinDist)
var classified_CART = S2_im.classify(Cart)
var classified_RF = S2_im.classify(RandomForest)

```

Bij het visualiseren willen we een visueel overzichtelijk resultaat krijgen. Aangezien we eindigen met discrete klassen, stellen we hiervoor een palette op, dat per klasse een kleur aangeeft.

```

var palette = [
  'FF0000', // mangrove (1) // rood
  '7FC00', // ander bos (2) // lichtgroen
  '1E900F', //water (3) // blauw
  'FFFFD10', //crop (4) //geel
  '000000', //stad // zwart
  '876829', //BareSoil // bruin
];

var classvis = {min:1, max:6, palette: palette}

Map.addLayer(classified_MD,classvis,'MinimumDistance')
Map.addLayer(classified_CART,classvis,'CART')
Map.addLayer(classified_RF,classvis,'RandomForest')

```

Accuraatheidsbepaling (Accuracy assessment)

In de *accuracy assessment* toetsen we de bruikbaarheid van het classificatiemodel: welke foutenmarge is er aanwezig? Dit doen we op basis van een foutenmatrix (**error matrix**). Om deze op te stellen hebben we nood aan een set van (onafhankelijke) testdata. We willen met andere woorden weten hoe goed onbekende pixels worden geclasseerd door de classifier.

Training- Validation en Testdata

Een veel gebruikte methode bij het opstellen van modellen, is het opsplitsen van de *trainingset*-dataset in training- en validatiedata. Hierbij wordt de *trainingdata* at random gesplits in meestal een 80/20-verhouding. Daarnaast wordt er vaak nog gebruik gemaakt van een derde, volledig afzonderlijke dataset: de **testdataset**:



Traindata= de data gebruikt om het model te trainen en dus te *fitten*. Het model bekijkt en leert van deze data.

Validatiedata= Het deel van de data dat gebruikt zal worden om na te gaan hoe goed het model werkt op onbekende data. Dit deel zal dus niet gebruikt worden om het model te trainen. Hierdoor kunnen verschillende classificatiemodellen en parameters binnen het model tegenover elkaar worden afgewogen en het model zo worden geperfected. Dit wordt ook wel *parameter tuning* genoemd. Validatiedata wordt dus gebruikt tijdens de ontwikkeling en het zoeken van het beste model. Bij spatiale data echter, dient hier voorzichtig mee te worden omgegaan door het fenomeen van *spatiale autocorrelatie*. Hierbij zijn de validatiepixels veleel buurpixels van de trainpixels. Dit is het geval wanneer er bijvoorbeeld gebruik wordt van polygonen als inputdata.

Testdata = Deze afzonderlijke dataset wordt gebruikt om bij een finaal model accuraatheidsmaten van de bekomen classificatie te berekenen. Testdatasets worden meestal ook zeer goed verzorgd en zijn goed verzamelde (veld)datapunten. De *spatiale autocorrelatie* vervalt hier.

De Error Matrix: interpretatie

Zie ook de foutenmatrix handboek pagina 577

De Error Matrix wordt opgesteld door het vergelijken van de geclasseerde testdata-waarden en de referentiedata. Onderstaande matrix geeft een voorbeeld van dergelijke matrix:

Classification	Ground reference data						User's accuracy
	Bareland/Constr	Dead Mangrove	Mangrove	Other Forest	Other Vegetation	Water	
Bareland Constr	40	3	0	0	4	3	50 80.00%
Dead Mangrove	0	44	1	0	1	4	50 88.00%
Mangrove	0	0	125	1	0	1	127 98.43%
Other Forest	1	1	8	97	2	0	109 88.99%
Other Vegetation	2	2	7	12	87	1	111 78.38%
Water	0	2	2	0	1	154	159 96.86%
Total	43	52	143	110	95	163	606 84.37%
Producer's accuracy	93.02%	84.62%	87.41%	88.18%	91.58%	94.48%	
Overall accuracy	90.26%						
Kappa index							

- **Rijen:** resultaat van de classificatie
- **Kolommen:** validatie data
- **Diagonaal:** pixels die goed geclasseerd zijn (validatie data = classificatie)
- **Niet-diagonaal:**
 - **Omissie:** de niet diagonale kolom-elementen. Deze pixels behoren tot een klasse, maar werden ingedeeld in een verkeerde klasse. In het voorbeeld: 18 pixels moesten Mangrove zijn, maar werden ingedeeld onder andere klassen: 1 als 'Dead Mangrove', 8 als *Other Forest*, 7 als *Other Vegetation* en 2 als *Water.
 - **Commissie:** Deze pixels geven aan welke pixels verkeerd werden ingedeeld in deze klasse. Voor *Water* zijn dit er bijvoorbeeld 2 (moesten 'Dead Mangrove') + 2 (moest *Mangrove* zijn + 1 (moest *Other Vegetation* zijn) = 5 pixels verkeerd als '*Water*' ingedeeld.

Op basis van de error matrix kunnen er enkele **accuraatheidsmaten** worden berekend.

- **Overall accuracy:** Dit is de som van de diagonale elementen, gedeeld door het totaal aantal pixels (= "juist ingedeeld"/totaal).
- **Producer accuracy:** het aantal correct ingedeeld pixels in elke klasse, gedeeld door het aantal validatiepixels van die klassen. Het geeft een indicatie hoe goed de validatiepixels geclasseerd werden (Bijvoorbeeld Mangrove = $(125/143 = 87,41\%)$ werd goed geclasseerd). Deze maat geeft de probabilitet weer dat een pixel die in een bepaalde klasse werd gestopt in werkelijkheid ook tot die klasse behoort.
- **User/consumer accuracy:** Het aantal correct geclasseerde pixels in elke klasse (diagonalelementen), gedeeld door het aantal pixels ingedeeld in die klasse (rijtotaal).
- **Kappa (KHAT) index:** de kappa index is een gecorigeerde accuraatheidsmaat, die intra- en interobserver agreement in rekening houdt. Het houdt m.a.w. rekening met pixels die per toeval juist geclasseerd zijn. Onderstaande tabel geeft een interpretatie weer van de kappa-waarde.

Values of kappa	Interpretation
< 0	No agreement
0-0.19	Poor agreement
0.20-0.39	Fair agreement
0.40-0.59	Moderate agreement
0.60-0.79	Substantial agreement
0.80-1.00	Almost perfect agreement

De Error matrix in Earth Engine

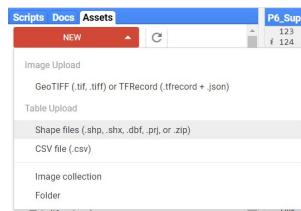
INLEZEN VAN DE TESTDATASET

In voorliggend voorbeeld maken we gebruik van een extra testdataset voor het opstellen van de error matrix. Gezien we geen optimalisatie van parameters gaan doorvoeren, maken we geen gebruik van validatedata en wordt de volledige trainingscollectie ook trainingsdata.

De gebruikte testdataset bestaat uit kleine polygonen met een diameter van 25m en representeren GPS-punten genomen op veldbezoek. De 25m-buffer rond de GPS-punten werd genomen om voldoende testpixels te weerhouden voor de accuracy assessment.

Je kunt de shape-file hier downloaden: [P6testdata_poly.zip](#)

Om deze toe te voegen aan Earth Engine, laad je deze op via de 'Asset'-tab. Daarna kun je het importeren als een `FeatureCollection` in je script. Noem dit 'Testdata_pol'. Bekijk ook even deze polygonen. Onder welke `property` zitten de klassen hier opgeslagen?



Vervolgens extraheren we de pixelwaarden op basis van deze testpolygons, net zoals we dit gedaan hebben bij de trainingspixels:

```
//Na inlezen van validatedata: maak een testdatacollectie, zoals bij
het opmaken van traindata
print('Testpolygons',Testdata_pol) //bekijk de properties

//Testpixels extraheren naar featurecollection
var testdata = S2_im.sampleRegions({
  collection: Testdata_pol, //De trainingspolygons
  properties: ['val'], //Dit neemt de gewenste eigenschappen van de
  collection over
  scale: 10
});
print('Aantal testpixels: ',traindata.size());
```

Nu we de testpixelwaarden geëxtraheerd hebben, kunnen we deze vergelijken met de geclasseerde pixels in een **error matrix**. Tevens staat Earth Engine een rechtstreekse berekening van de verschillende accuraatheidsmaten toe. In onderstaand stukje code staat het voorbeeld voor de Minimum Distance classifier. Pas dit toe voor alle classifiers. Welke classifier heeft de grootste algemene accuraatheid (*Overall accuracy*)?

```
// Validatie met de testdata
var val_MinDist = testdata.classify(MinDist);
var ErrorMatrix_MinDist = val_MinDist.errorMatrix('val',
'classification')

print('MinDist Validation error matrix: ',
ErrorMatrix_MinDist.array().transpose());
print('MinDist Validation overall accuracy: ',
ErrorMatrix_MinDist.accuracy());
print('MinDist Producer Accuracy: ',
ErrorMatrix_MinDist.producersAccuracy());
print('MinDist User/Consumer Accuracy: ',
ErrorMatrix_MinDist.consumersAccuracy());
print('Kappa index: ', ErrorMatrix_MinDist.kappa());

//Omvatten naar een Feature + transpose
var ErrorMatrix_MinDist = ee.Feature(null, {matrix:
ErrorMatrix_MinDist.array().transpose()});
```

De Error matrix in Earth Engine

In Earth Engine wordt de error matrix opgeroepen met de `ee.ConfusionMatrix()`-functie. Resulterend is een lijst met 7 elementen (de rijen), waarbij elke rij op zijn beurt bestaat uit 7 elementen (de kolommen). In Earth Engine corresponderen de rijen met de referentiedata en de kolommen met de geclasseerdeerde data. Met de `array().transpose()`-functie kunnen we deze matrix transponeren, zodat deze overeenkomst met de matrix in het voorbeeld (kolommen: referentie, rijen: geclasseerd), wat de standaard weergave is.

```
errorMatrix(actual, predicted, order)
Computes a 2D error matrix for a collection by comparing two columns of a collection: one containing the actual values, and one containing predicted values. The values are expected to be small contiguous integers, starting from 0. Axis 0 (the rows) of the matrix correspond to the actual values, and Axis 1 (the columns) to the predicted values.
```

Daarnaast wordt ook klasse '0' meegerekend in de berekening. Gezien deze in ons voorbeeld niet bestaat, zal de 1e rij/kolom enkel 0-waarden bevatten.

Voorbeeld: interpretatie error matrix

In ons voorbeeldje werd onderstaande ErrorMatrix verkregen voor de MinDist classificer. Door de `.transpose()`-functie komt de referentiedata terecht in de kolommen, terwijl de classificatiedata zich in de rijen bevindt.

```
MinDist Validation error matrix:
List (7 elements)
0: [0,0,0,0,0,0,0]
1: [0,179,71,0,0,0,0]
2: [0,59,257,0,15,22,0]
3: [0,3,0,438,0,0,0]
4: [0,0,12,0,147,75,41]
5: [0,0,0,0,0,193,18]
6: [0,0,0,0,0,171,19]
```

Na wat opschoning in excel, ziet de error matrix van dit voorbeeld er uit als volgt:

Classified data	Reference data						Consumer accuracy
	Mangrove	Otherforest	Water	Crop	Urban	Soil	
Mangrove	179	71	0	15	22	0	250 71.60%
OtherForest	59	257	0	0	0	0	353 72.80%
Water	3	0	438	0	0	0	441 99.32%
Crop	0	12	0	147	75	41	275 53.45%
Urban	0	0	0	0	193	18	211 91.47%
Soil	0	0	0	0	171	19	190 10.00%
	241	340	438	162	461	78	1720
Producer's accuracy	74.27%	75.59%	100.00%	90.74%	41.87%	24.36%	
Overall accuracy	71.69%						
Kappa index	65.50%						

Welke klassen scoren goed? Welke zijn minder accuraat? Aan de *overall accuracy* en de *Kappa index* kan worden afgeleid dat de classificatie reeds een goede indicatie heeft, maar nog voor verbetering vatbaar is. Hoofdzakelijk de klassen 'water' en 'crop' scoren goed, terwijl 'Soil' moeilijker te onderscheiden valt van 'crop' en 'urban'. Betere trainingsdata is hier dus de boodschap!

Extra: Exporteren van de Error Matrix

In Google Earth Engine is de weergave van de error matrix niet zo handig. Om verdere accuraatheidsmaten uit te rekenen en een betere interpretatie te kunnen uitvoeren kan het handig zijn om de error matrix te exporteren als een .csv-bestand, dewelke in andere software (zoals excel) geopend kan worden.

Met de `Export.table.toDrive()`-functie kunnen we de matrix exporteren naar onze Google Drive. Hiervoor dienen we dit eerst om te zetten naar een *feature*.

```
//Omwettzen naar een Feature
var ErrorMatrix_MinDist = ee.Feature(null, {matrix: ErrorMatrix_MinDist.array()});

//Exporteren van de errormatrix
Export.table.toDrive({
  collection: ee.FeatureCollection(ErrorMatrix_MinDist),
  description: 'P6_Errormatrix',
  fileFormat: 'CSV',
  folder: 'TELEDETECTIE_2021'
});
```

Volledig script

Via deze link: <https://code.earthengine.google.com/665cbaa3a887d685d4d07c6081902f19>

5.5 Verbeteren van de classificatie

In de gesuperviseerde classificaties testten we reeds 3 'classifiers', waarvan de *Random Forest* classifier leidde tot een iets algemene accuraatheid. Desondanks ze al tot acceptabele percentages leidden, bestaan er enkele opties om de classificatie nog te verbeteren.

Aanpassen van het aantal trainingssamples en aanpassen *sampling*-strategie

De opgemaakte trainingssamples kunnen worden verbeterd door het aantal samples omhoog te trekken, of door de *sampling*-strategie aan te passen. In dit voorbeeld tekenden we zelf polygonen in op basis van visuele inspectie, waardoor we enkel de pixels werden aangeduid waar we relatief zeker waren van de desbetreffende klasse.

Een betere techniek zou erin bestaan om op voorhand willekeurige locaties door de computer aan te laten duiden, eventueel gestratificeerd per landbedekkingsklasse ('*stratified sampling*').

Toevoegen van Indices

Ook door de input-banden aan te passen of extra banden toe te voegen, kan er voor zorgen dat de onderscheiding van de klassen wordt verbeterd. Zo kan:

- er hoogtedata (zgn. *Digital Elevation Model* of DEM) worden toegevoegd, waaruit bv de hellingsgraad van het terrein kan worden berekend. Zo kunnen land*features* met een specifieke topografie zoals basins, kanalen, toppen, dalen, hellingen gemakkelijker worden geïdentificeerd.
- er informatie uit berekende indices helpen voor versterking van verschillen tussen klassen, zoals NDVI, MNDWI of in de context van mangrove-classificatie de MVI.

Opdracht

Tracht je classificatie te verbeteren door:

- enkele indices toe aan het Sentinel-2 beeld van vorige oefening, zoals de MVI, NDVI en NDWI.
- Textuurmatrices toe te voegen. Zie hiervoor [P5 - Textuur](#) Gebruik deze keer enkel de Random Forest classifier. Bekijk de bekomen accuraatheid. Merk je verbeteringen op?

5.6 Oppervlaktebepaling

Oppervlaktebepaling

Nu de classificatie is uitgevoerd, kunnen we per finale landbedekkingsklasse ook de oppervlakte bepalen. Gezien het bekomen classificatieleresultaat een rasterbeeld met pixels is, zal de oppervlakte een klasse gelijk zijn aan de som van de oppervlakte van elke pixel binnen de klasse.

- `.area()` : wordt gebruikt om de oppervlakte van `Features` te bepalen.
- `ee.Image.pixelArea()` : wordt gebruikt om de oppervlakte van pixels binnen een rasterbeeld te bepalen. Het resultaat wordt weergegeven in m^2 .

In Google Earth Engine zijn hier verschillende methodes voor de oppervlakteberekening.

- **Stap 1** - afzonderen van de klasse met een `.eq()`-functie. Het resultaat is een binair beeld, waarbij de '1'waarde de geselecteerde klasse(n) voorsteld, waarde '0' de overige pixels.

```
// Mangroveklasse afzonderen uit het geclasseerd beeld
var Mangrove_class = classified_RF.eq(1)

// Visualiseren van het beeld
Map.addLayer(Mangrove_class, {min:0, max:1, palette: ['white', 'green']}, 'Mangrove Cover')
```

- **Stap 2** - pixelwaarde vervangen door de oppervlakte van de pixel.

```
//Pixelwaarden (=1) vermenigvuldigen met de pixeloppervlakte.
var areaImage = Mangrove_class.multiply(ee.Image.pixelArea())
```

- **Stap 3** - Nu elke pixel van de Mangrove-klasse een waarde heeft gelijk aan de oppervlakte, kunnen we de som nemen van alle pixels in de regio om de totale oppervlakte te verkrijgen. Dit doen we met een Reducer (zie ook P4)

```
var area_Mangrove = areaImage.reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: ROI,
  scale: 10, //minimumpixelgrootte zorgt voor meest accurate waarde.
  maxPixels: 1e10 //Vergroten rekencapaciteit
})
```

- **Stap 4** - Het resultaat is een dictionary, waarbij de enige 'waarde' de oppervlakte is (check dit door het resultaat naar de Console te prullen). Print nu de oppervlakte naar de console. Deze is momenteel uitgedrukt in m^2

```
var MangroveAreaHa =
ee.Number(area.get('classification')).divide(1e4).round()
print('Oppervlakte Mangrove: ', MangroveAreaHa)
```

Opdracht

Bereken en print ook de oppervlakte van de klasse 'Ander bos' binnen het studiegebied.

EXTRA: Oppervlaktehistogram

Hierboven werd een standaardmethode behandeld om de oppervlakte per klasse in een geclasseerd beeld afzonderlijk te berekenen. Met onderstaande code kun je meteen de oppervlakte per klasse gezamenlijk berekenen en via de `ui.Chart.image.byClass()` functie een histogram visualiseren die de oppervlaktes weergeeft.

```
var areaImageHa =
ee.Image.pixelArea().divide(1e4).addBands(classified_RF);

// Berekenen oppervlakte per klasse met een reduceRegion()-functie
// We voeren m.a.w. een reductie uit, waarbij ons
// classificatieleresultaat als regio's gelden, waarbinnen de soms wordt
// berekend. De .group()-functie maakt dit mogelijk
var areas = areaImageHa.reduceRegion({
  reducer: ee.Reducer.sum().group({
    groupField: 1,
    groupName: 'classification', //De bandnaam van het
    //geclasseerd beeld is standaard 'classification'
  }),
  geometry: ROI,
  scale: 10,
  tileSize: 6,
  maxPixels: 1e10
});

var classAreas = ee.List(areas.get('groups'))
print(classAreas)

var areaChart = ui.Chart.image.byClass({
  image: areaImageHa,
  classBand: 'classification',
  region: ROI,
  scale: 20, //Op 20m pixelgrootte berekenen voor beperken
  rekencapaciteit
  reducer: ee.Reducer.sum(),
  classLabels: ['', 'Mangrove', 'Other Forest', 'Water',
  'Crop', 'Urban', 'Bare Soil'], // ''0' is een lege klasse
}).setOptions({
  hAxis: {title: 'Classes'},
  vAxis: {title: 'Area Km2'},
  title: 'Area by class',
  series: [
    0: {color: 'red'},
    1: {color: 'green'},
    2: {color: 'blue'},
    3: {color: 'yellow'},
    4: {color: 'black'},
    5: {color: 'brown'}
  ]
});
print(areaChart);
```

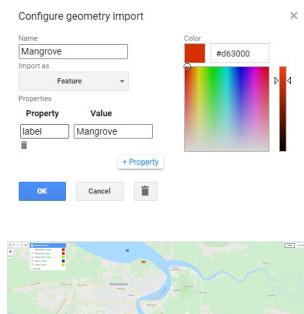
5.7 Spectrale responsiecurves

Als laatste onderdeel van dit practicum, kijken we even naar de spectrale signaturen van onze klassen. Uit deze curves kunnen we volgende zaken afleiden (zie ook Practicum 1):

- Welke banden/indices zorgen voor onderscheid tussen klassen?
- Welk spectrale curves heeft elk van onze klassen? Kunnen we deze curves ook verklaren?

Aanmaken voorbeeldFeatures

Om spectrale curves te maken, hebben we eveneens voorbeeld-samples nodig van elk van de klassen. Dit kan door opnieuw nieuwe *features* in te tekenen (bv één punt voor 1 klasse). We doen dit bijvoorbeeld voor de klassen Mangrove, OtherForest, Water en Urban. Zoek één representatief punt voor elk van de klassen. Maak deze aan als het type *Feature* en geef het een *property* 'label', met een bijpassende naam



Eenmaal de VoorbeeldFeatures zijn aangemaakt kun je ze samenvoegen in het script:

```
var vbPol = ee.FeatureCollection([Mangrove,OtherForest,Water,Urban]);
```

S2-beeld aanmaken

Maak ook opnieuw het Sentinel-2 beeld uit vorige oefening aan, met de indices:

```
//
-----
// STAP 1 - Inladen en klaarzetten van S2-beeld. Met extra cloud-
masking
// -----
//Cloudprobability functie:
// Functie die nieuwe CloudProbability collectie samenvoegt met S2
// (sen2cloudless)
// meer info: https://medium.com/sentinel-hub/cloud-masks-at-your-
// service-6e5b2cb2ce8a
var getS2_SR_CLOUD_PROBABILITY = function () {
    var innerJoined = ee.Join.inner().apply({
        primary: ee.ImageCollection("COOPERNICUS/S2_SR"),
        secondary: ee.ImageCollection("COOPERNICUS/
S2_CLOUD_PROBABILITY"),
        condition: ee.Filter.equals({
            leftField: 'system:index',
            rightField: 'system:index'
        })
    });
    var mergeImageBands = function (joinResult) {
        return ee.Image(joinResult.get('primary'))
            .addBands(joinResult.get('secondary'));
    };
    var newCollection = innerJoined.map(mergeImageBands);
    return ee.ImageCollection(newCollection);
};

// Mask out clouds
```

```
var maskClouds = function(image) {
    var cloudProbabilityThreshold = 40;
    var cloudMask =
    image.select('probability').lt(cloudProbabilityThreshold);
    return image.updateMask(cloudMask);
};

//Aanmaken van een ImageCollection ter hoogte van Mangroves
//Paramaribo, Suriname
var S2_coll = getS2_SR_CLOUD_PROBABILITY()
    .filterDate('2019-08-01','2019-10-30')// Filteren voor het
jaar 2020, droge tijd
    .filterMetadata('CLOUDY_PIXEL_PERCENTAGE','less_than',50) // /
Voorselectie obv wolken
    .map(maskClouds) //toepassen van de cloudmaskfunctie
    .filterBounds(ROI); //collectie filteren obv de
Kustzonegeometrie

// -----
// TOEVOEGEN INDICES
// -----
var addIndices = function (image) {
    var mvi = image.expression('(B8-B3)/(B11-B3)', {
        'B8': image.select('B8'),
        'B3': image.select('B3'),
        'B11': image.select('B11')
    }).float().rename('MVI');
    var ndvi = image.normalizedDifference(['B8',
'B4']).rename('NDVI');
    var ndwi = image.normalizedDifference(['B3',
'B12']).rename('NDWI');
    return image.addBands(mvi).addBands(ndvi).addBands(ndwi);
};

//Toepassen indices + medianreducer + clippen
var S2_im = S2_coll.map(addIndices).median().clip(Paramaribo);
```

Cloud mask methode

In voorgaande code wordt opnieuw gebruik gemaakt van de extra S2-cloudmask methode. Je kunt evengoed gebruik maken van de andere strategiën, zoals gezien in "Cloud Masking" van Practicum 4.

Spectrale responsiecurve aanmaken

Vervolgens kunnen we de Chart aanmaken. Tevens linken we de overeenkomstige golflengtes aan de banden, om zo een spectrum te krijgen met de golflengte in de X-as.

```
// Banden S2 aangeven (hier: volledig, inclusief B1 en B9 (worden
weggelaten in classificatie wegens onbruikbaar)
var wavelengths
=[ 'B1','B2','B3','B4','B5','B6','B7','B8','B8A','B9','B11','B12']

//De overeenkomstige golflengte per band aangeven (zie bandenverdeling
//Sentinel-2).
var wavelengths =[443.9,496.6,559,664.5,703.9,740.2,782.5,835.1,864.8,
945,1613.7,2202.4]

//Aanmaken Chart
var Chart = ui.Chart.image.regions({
    image: S2_im.select(bands),
    regions: vbPol,
```

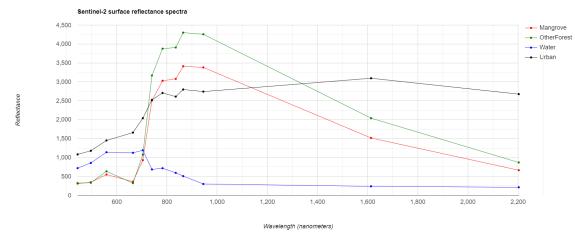
```

reducer: ee.Reducer.mean(),
scale: 10,
seriesProperty: 'label',
xLabels: wavelengths
})

var plotOptions = {
  title: 'Sentinel-2 surface reflectance spectra',
  hAxis: {title: 'Wavelength (nanometers')},
  vAxis: {title: 'Reflectance'},
  lineWidth: 1,
  pointSize: 4,
  series: [
    0: {color: 'red'}, // Mangrove
    1: {color: 'green'}, // Forest
    2: {color: 'blue'}, // water
    3: {color: 'black'}, //crops/grass
  ]};

```

```
print(Chart.setOptions(plotOptions));
```



5.8 Oefeningen

OEF 5.1 - Classificatie van België

Gegeven

In volgende oefening maken we een landclassificatie van België op basis van een Landsat-8 beeldcomposit. Om de grenzen van België te bekomen, maken we gebruik van volgende dataset: https://developers.google.com/earth-engine/datasets/catalog/USDOS_LSIB_SIMPLE_2017

Om hieruit België te filteren, maak je gebruik van onderstaande code:

```
var countries = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017');
var bel = countries.filterMetadata('country_na', 'equals', 'Belgium');
```

Opdracht

- Maak één wolkenvrij Landsat-8 beeld aan voor 2019. Weerhoud enkel de bruikbare banden voor classificatie: voor Landsat-8 dit zijn volgende banden: blauw, groen, rood, nir, swirl1, swirl2.
- Clip je resulterende beelden op basis van de Belgische grens.
- Voer een classificatie uit met volgende klassen: *bos, water, grasland, landbouw, urban*. Maak hiervoor je eigen trainingssamples aan. Vergeet ze niet samen te voegen tot één `FeatureCollection`.
- Kies 2 classifiers naar keuze.
- Visualiseer je resultaat naar eigen keuze. Evaluere het visueel: is je classificatie geslaagd? Welke classifier lijkt beter?
- Bereken de *Overall Accuracy* van je classificatie. Maak gebruik van deze validatieset: [P6_Oef1_validation.zip](#). Hierbij maak je gebruik van de eigenschap 'val', waarbij volgend schema geldt:

val	Landbekkingsklasse
1	Bos
2	Water
3	Grasland
4	Landbouw
5	Urbaan

Oplossing Oef 5.1

Voor deze oefening kun je zowel Landsat Collectie 2 (de allernieuwste) of Landsat Collectie 1 (toekomstig niet meer gebruikt, maar momenteel nog in een overgangsfase) gebruiken. Het verschil is verwaarloosbaar.

Oplossing Landsat 8 Collectie 1: <https://code.earthengine.google.com/b80d9ab8e0be2e309dbd64fe3f7a0f81> Oplossing Landsat 8 Collectie 2: <https://code.earthengine.google.com/021d9c2592813d2779aff69722edf08a>

Opgelet: het spreekt voor zich dat de gebruikte trainingsvectoren in dit voorbeeld zeer rudimentair zijn, met een ruwe classificatie tot gevolg. Daarbij werden de testpunten ook gehaald uit een bestaande landcover classificatie van België, maar met een resolutie van 100m, waardoor de kwaliteit van de testdata niet perfect is. Verder is een verbetering van het classificatieresultaat ook mogelijk door het gebruik maken van Multitemporale beelden: hiermee wordt het sezonaal karakter van de o.a. de landbouwvelden mee in rekening gebracht, waardoor een beter onderscheid tussen o.a. grasland en landbouw (sterker temporeel karakter) behaald kan worden.

OEF 5.2 - Eyjafjallajökull



De gletsjer Eyjafjallajökull is een van de kleinere gletsjers op IJsland en heeft een oppervlakte van ongeveer 100 km². De Eyjafjallajökull ligt ten noorden van het plaatsje Skógar. Op de oostflank van de vulkaan, nabij de bergpas Fimmvörðuháls, vond op 20 maart 2010 nieuwe vulkanische activiteit plaats. Een tweede explosieve uitbarsting in de hoofdkrater van de Eyjafjallajökull, begon op 14 april 2010. In grote delen van Europa werd het vliegverkeer dagenlang volledig stilgelegd vanwege de aswolken die de vliegtuigen kunnen beschadigen.

Gegeven

Maak gebruik van volgend Sentinel-2 beeld en ROI, genomen in 2019:

```
var S2 = ee.Image('COPERNICUS/S2_SR/20190810T125311_20190810T125306_T27VWL');
var ROI = ee.Geometry.Polygon(
  [[[[-19.967455239503007, 63.845568279400595],
    [-19.967455239503007, 63.398959439658746],
    [-18.824877114503007, 63.398959439658746],
    [-18.824877114503007, 63.845568279400595]]], null, false);
```

```
var S2 = ee.Image('COPERNICUS/S2_SR/
20190810T125311_20190810T125306_T27VWL').clip(ROI)
```

- In deze oefening gaan we geen cloud mask toevoegen, maar de wolken en wolkenschaduwen opnemen in de classificatie.
- Training data: Chinese experten digitaliseerden verschillende polygonen in het gebied rond de vulkaan. Hierbij werd onderscheid gemaakt in 5 klassen:
 - Gletsjer
 - Schaduw
 - Bodem
 - Vegetatie
 - Water
 - Wolken

De Trainingfiles werden reeds ondergebracht in een `FeatureCollection` en kunnen via deze link worden ingelezen:

```
var traindata = ee.FeatureCollection("users/jasperfeyen/
TELEDETECTIE2020/P6_oef2_training");
```

- **Referentie data:** tijdens een veldcampagne in 2019 werd het gebied rond de vulkaan intensief bemonsterd. Honderden pixels werden op het terrein bezocht en de landbedekking werd geregistreerd.

Je kunt de shape-file hier downloaden: [P6_oef2_val.zip](#)

Gevraagd

Classificeer het 2019 beeld met behulp van 2 supervised classifiers naar keuze. Voor de classificatie transformeer je de data, zodat je slechts 3 getransformeerde banden overhoudt die de meeste informatie bevatten.

Oplossing

Via volgend script: <https://code.earthengine.google.com/38fd0ac0f35c8a4175afb7273d8eae4b>