

3.2 EarthEngine data catalog

De Earth Engine Data Catalog

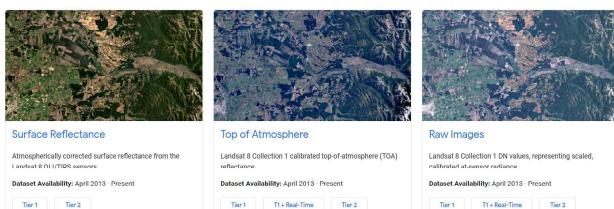
Om het aanbod aan aardobservatie-data in Google Earth Engine te bekijken en te doorzoeken, kan gebruik gemaakt worden van de Earth Engine Data Catalog: <https://developers.google.com/earth-engine/datasets>. Via deze catalogus kun je eenvoudig rasterdata allerhande opzoeken: satellietdata, weerdata, terreindata,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

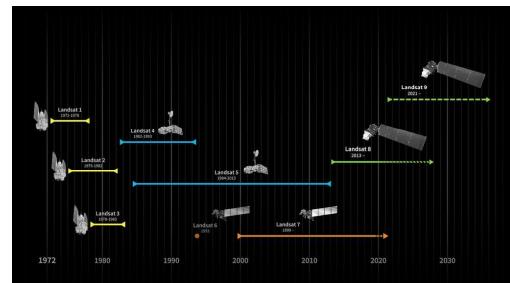


Een derde opdeling gebeurt ten slotte op basis van de beeldkwaliteit:

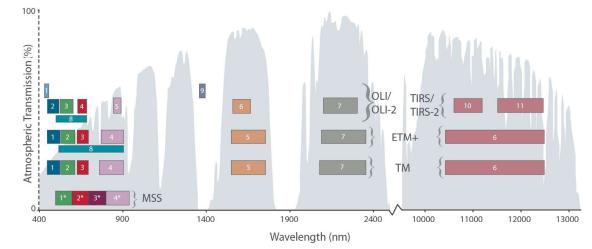
- **Tier 1:** De meest kwalitatieve beelden, geschikt voor tijdserie-analyse. De beelden zijn zowel geometrisch als radiometrisch kwalitatief goed bevonden volgens opgestelde standaarden.
- **Tier 2:** De beelden zijn geometrisch en/of radiometrisch minder kwalitatief bevonden, maar zijn wel nog inzetbaar 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.

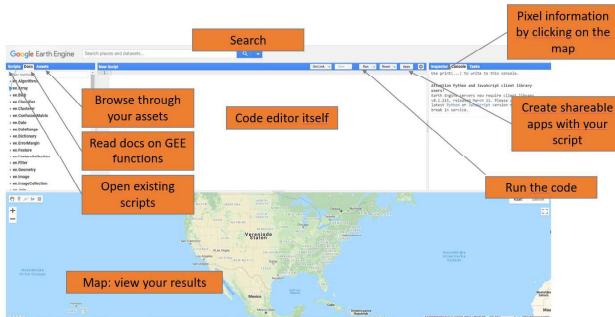


Onderstaande grafiek geeft een overzicht van de bandverdeling van de Landsatsatellieten. Huidig actieve Landsat-satellieten zijn Landsat 9 (OLI-2/TIRS-2) 8 (OLI/TIRS) en Landsat 7 (ETM+) (Bron: [NASA](#))



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 linkerscherm, 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 rechterscherm, 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 worden 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 gecorregende 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 filennaam 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.

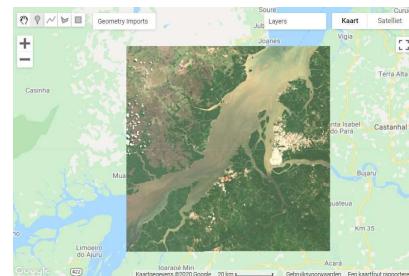


Bij het uitvoeren van bovenstaande code bekomen we een zwart vlak, niet bepaald de visualisatie die we wensen. Dit komt omdat we nog geen visualisatieparameters hebben aangegeven, waardoor de eerste 3 banden naar de rode, groene en blauwe band respectievelijk worden toegekend en de pixelrange zo groot 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-scherm 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 imagevisParam: 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 filennaam 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.

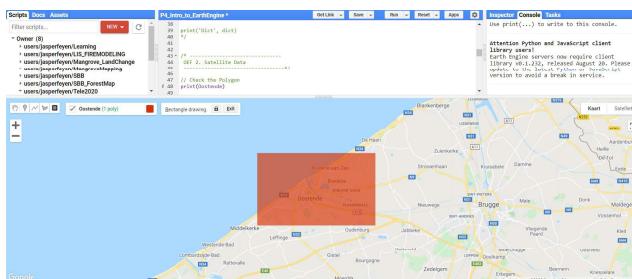
Region of Interest (ROI)

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 wolkenbedekking 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 beeldeigenschappen 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/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 kleurencomposiet 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

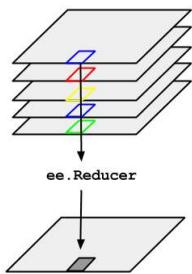
GEE script: <https://code.earthengine.google.com/0861ba83272bc305848ee2d113c4d3ef>

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());

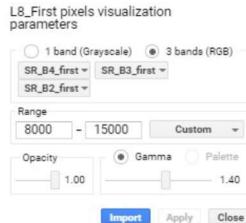
// Bekijk de eigenschappen van het gereduceerd beeld
print(L8_first_red)

var visParams_first = {
  bands: ['SR_B4_first', 'SR_B3_first', 'SR_B2_first'],
  min: 8000,
  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 strategieën 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 - le beeld - No Cloudmask')

// Eerste beeld uit collectie mét Cloudmask:
Map.addLayer(L8_masked.first(), trueColor, 'L8 - le 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()`. De `.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 4.1 - 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

Coming Soon