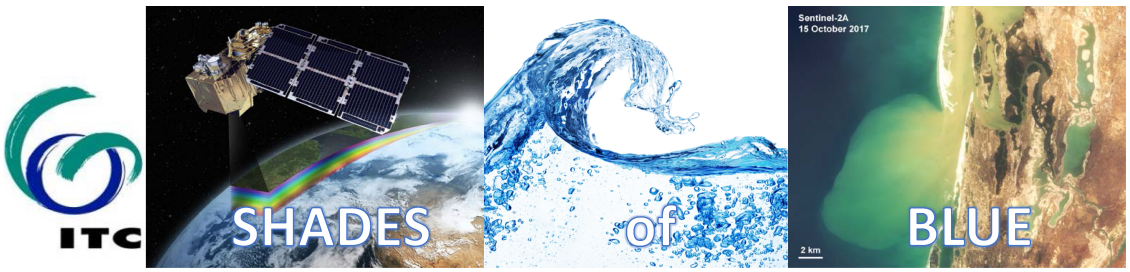
2.1_WQ_IndicesS2

April 21, 2023



UNIVERSITY OF TWENTE.

1 Water quality indices from Sentinel-2 MSI

Author: Suhyb Salama, ITC

1.1 Purpose

This exercise introduces you to working with water quality indices using Sentinel-2 MSI (S2-MSI) level 2A images (i.e. atmospherically corrected images). This is a python script in jupyter notebook and follows from the exercises presented in proceeding weeks.

1.2 Prerequisites

- You should have an account in Google Earth Engine. If not done yet please sign up now.
- You should have worked out the first two exercises of exercise 1.1; and exercise 1.2.

1.3 What you will learn

1. Carry out cloud and shadow masking;
2. Apply different indices for assessing water quality;
   2.1 Normalized Difference Water Index (**NDWI**);
   2.2 Normalized Difference Turbidity Index (**NDTI**);
   2.3 Normalized Difference Chlorophyll Index (**NDCI**);

3. Perform spatial subsets and areal aggregation;

4. visualize the results as maps and time series.

1.4 What will you achieve

At the end of this exercise you will be able to use Google Earth Engine to access satellite data, assess water quality using band ratios and indices from S2-MSI.

1.5 Introduction

Load the required libraries and authenticate your Google account

[2]:
```
# import Google earth engine module
import ee
import geemap
import geemap.colormaps as cm
Map =geemap.Map()
#import pandas as pd
#import matplotlib.pyplot as plt
```

If you are not authenticated (logged in Google Earth) then you should acquire the authentication code, otherwise you can skip this cell

[3]:
```
try:
    ee.Initialize()
except Exception as e:
    ee.Authenticate()
    ee.Initialize()
```

```
<IPython.core.display.HTML object>
```

```
Enter verification code:
4/1AVHEtk54vi1MDLykCAOSDTEAWeYoNyB_HtmvaGG67U6cha7Jvs7Kp4CnXcM


Successfully saved authorization token.
```

1.6 Data access

Request Google Earth Engine to access Sentinel 2 data Level-2A data. Level 2 means that these images are atmospherically corrected. To have a better understanding of data level of S2-MSI you can consult the user guide of S2-MSI.

[4]:
```
s2_boa_col = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
```

1.6.1 Inquire the band names of S2-MSI image collection

- Select one image (the first);
- Inquire the bands names of this image and print.

[5]:
```
first_image = s2_boa_col.first()
Bname = first_image.get('system:band_names')
print('system:band_names', Bname.getInfo())
```

```
system:band_names ['B1', 'B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B8', 'B8A', 'B9',
'B11', 'B12', 'AOT', 'WVP', 'SCL', 'TCI_R', 'TCI_G', 'TCI_B', 'MSK_CLDPRB',
'MSK_SNWPRB', 'QA10', 'QA20', 'QA60']
```

1.7 Filter the S2-MSI image collection to:

- area of interest *AOI*
- time period *s_date* and *e_date*
- cloud percentage *cld_per* and probability *cld_prb*

[6]:
```
aoi = ee.Geometry.Polygon([
    [-2.98, 54.4],
    [-2.98, 54.3],
    [-2.95, 54.3],
    [-2.95, 54.4],
    [-2.98, 54.4]])

s_date = '2018-01-01'
e_date = '2021-12-31'
cld_per = 20
```

- Apply the filter on the Sentinel-2 surface reflectance

[7]:
```
# filter s2-MSI bottom of atmosphere
s2_boa_flt = s2_boa_col\
    .filterBounds(aoi)\
    .filterDate(s_date, e_date)\
    .filter(ee.Filter.lte('CLOUDY_PIXEL_PERCENTAGE', cld_per))
```

1.8 Remove clouds and their shadows

There are two ways: 1. Define a function to mask cloud and their shadows using the **SCL** class table:

Value	Description
1	Saturated or defective
2	Dark Area Pixels
3	Cloud Shadows
4	Vegetation
5	Bare Soils
6	Water
7	Clouds Low Probability / Unclassified
8	Clouds Medium Probability
9	Clouds High Probability
10	Cirrus
11	Snow / Ice

You can build a function as:

[8]:
```
def mask_cloud_shadow(img):
    # Identify good pixels from the SCL band.
    good_pixels = img.select('SCL').eq(6)#.neq(1).neq(3).lt(7)
    #noLand = img.select('B11').lt(50)
    return img.updateMask(good_pixels).divide(10000)
```

2- The second approach is to use the Quality Check band QA at 60 meter resolution.

This approach produce better results than the classification of the ***SCL*** *shown above*

[9]:
```
def s2_cloud_mask(image):
    quality_band = image.select('QA60')

    # using the bit mask for clouds and cirrus clouds respectively
     # Bits 10 and 11 are clouds and cirrus, respectively.
    cloudmask = 1 << 10
    cirrusmask = 1 << 11

    # we only want clear skies
    mask = quality_band.bitwiseAnd(cloudmask).eq(0) and (quality_band.
 ↪bitwiseAnd(cirrusmask).eq(0))

    # we'll divide by 10000 to make interpreting the reflectance values easier
    return image.updateMask(mask).divide(10000).copyProperties(image, ["system:
 ↪time_start"])
```

1.9 Generate a cloud-free composite

- Apply the cloud and shadow mask function **mask_cloud_shadow**

[10]:
```
s2_boa_cloudfree=s2_boa_flt.map(s2_cloud_mask)
```

- Reduce the image collection to to show the mean or median

[11]:
```
means2_sr = s2_boa_cloudfree.reduce(ee.Reducer.mean())
```

- Inquire the bands' names after the **reduce** operation

[12]:
```
Bname = means2_sr.get('system:band_names')
Bnames =  Bname.getInfo()
print(Bnames)
```

```
['B1_mean', 'B2_mean', 'B3_mean', 'B4_mean', 'B5_mean', 'B6_mean', 'B7_mean',
'B8_mean', 'B8A_mean', 'B9_mean', 'B11_mean', 'B12_mean', 'AOT_mean',
'WVP_mean', 'SCL_mean', 'TCI_R_mean', 'TCI_G_mean', 'TCI_B_mean',
'MSK_CLDPRB_mean', 'MSK_SNWPRB_mean', 'QA10_mean', 'QA20_mean', 'QA60_mean']
```

- Display the mean/median in **GEEMAP** using bands 2, 3 and 5

[13]:
```
rgbVis = {
    'min':0.0,
    'max':0.1,
    'bands':[Bnames[4],Bnames[2],Bnames[1]],
}
```

[14]:
```
Map.addLayer(means2_sr, rgbVis, 'SAms2',1)
#show the map
Map
```

```
Map(center=[20, 0], controls=(WidgetControl(options=['position',␣
↪'transparent_bg'], widget=HBox(children=(Togg...
```

2 Generate water quality indices

Here below we list the bands of Sentinel 2, more information is in this link

Sentinel-2 Bands	Name convention in this course	Central Wavelength (μm)	Resolution (m)
Band 1 - Coastal aerosol	Violet	0.443	60
Band 2 - Blue	Blue	0.49	10
Band 3 - Green	Green	0.56	10
Band 4 - Red	Red	0.665	10
Band 5 - Vegetation Red Edge	redEdge1	0.705	20
Band 6 - Vegetation Red Edge	redEdge2	0.74	20
Band 7 - Vegetation Red Edge	redEdge3	0.783	20
Band 8 - NIR	NIR_1	0.842	10
Band 8A - Vegetation Red Edge	NIR_2	0.865	20
Band 9 - Water vapour	NIR_3	0.945	60
Band 10 - SWIR - Cirrus	SWIR1	1.375	60
Band 11 - SWIR	SWIR2	1.61	20
Band 12 - SWIR	SWIR3	2.19	20

2.1 Normalized difference water index (NDWI)

- Normalized difference water index (NDWI) is a satellite based index used for mapping and detecting surface water bodies.

- Water absorbs electromagnetic radiation in visible to infrared spectrum, that is why green and Near Infrared (NIR) bands are used to detect the water bodies.

- For Sentinel-2 we will use the green (Band 3 centered at 560 nm) and NIR_1 (Band 8 centered at 842 nm) bands for generating NDWI. Sentinel-2 has a second NIR bands centered at 865 nm.

$$NDWI = \frac{Band(green) - Band(NIR_1)}{Band(green) + Band(NIR_1)}$$

- NDWI varies between [-1 and +1], positive values are most likely water bodies.

[15]:
```
def get_ndwi(img):
    date = img.date()
    green_band = img.select('B3')
    nir_band   = img.select('B8')
    A = green_band.subtract(nir_band)
    B = green_band.add(nir_band)
    ndwi = A.divide(B).rename('ndwi').copyProperties(img, ["system:time_start"])
    ndwi.set({
            'system:time_start': date,
            'system:time_end': date,
            'year': date.get('year'),
            'month': date.get('month'),
            'date': date})
    return ndwi
```

2.2 Normalized difference turbidity index (NDTI)

- The Normalize Difference Turbidity Index (NDTI) is used to estimate the turbidity in water bodies.
- It uses the phenomenon that reflectance is higher in green than the red for clear water. Hence, with increased turbidity, the reflectance of red band also increases.
- NDTI for Sentinel-2 uses the green (Band 3 centered at 560 nm) and red (Band 4 centered at 665 nm) bands.

$$NDTI = \frac{Band(red) - Band(green)}{Band(red) + Band(green)}$$

[16]:
```
def get_ndti(img):
    date = img.date()
    green_band = img.select('B3')
    red_band   = img.select('B4')
    A = red_band.subtract(green_band)
    B = red_band.add(green_band)
    ndti = A.divide(B).rename('ndti').copyProperties(img, ["system:time_start"])
    ndti.set({
        'system:time_start': date,
        'system:time_end': date,
```

```
        'year': date.get('year'),
        'month': date.get('month'),
        'date': date})
    return ndti
```

2.3 Normalized difference chlorophyll index

Normalized difference chlorophyll index (NDCI) is used to estimate the content of phytoplankton in water bodies.

- This index uses the information from the reflectance peak centered at the red edge band 705 nm (redEdge1 band) which is maximally sensitive to the variations in chlorophyll-a (chl-a) concentration in water. Similarly, a wide spectral absorption peak between 665 nm and 675 nm (red band) is generally assigned to the absorption by chl-a pigments (red band centered at 665 nm).
- Accordingly, in the current exercise, Sentinel-2 red (band 4) and far-red (band 5) bands are used to create the NDCI.

$$NDCI = \frac{Band(redEdge1) - Band(red)}{Band(red) + Band(redEdge1)}$$

[17]:
```
def get_ndci(img):
    date = img.date()
    red_band = img.select('B4')
    redEdge1_band  = img.select('B5')
    A = redEdge1_band.subtract(red_band)
    B = redEdge1_band.add(red_band)
    ndci = A.divide(B).rename('ndci').copyProperties(img, ["system:time_start"])
    ndci.set({
        'system:time_start': date,
        'system:time_end': date,
        'year': date.get('year'),
        'month': date.get('month'),
        'date': date})
    return ndci
```

2.4 Algal bloom detection index (ABDI)

Algal Bloom Detection Index (ABDI) consists of two parts. The first part is mainly used to extract algal blooms, which is expressed as the difference between band 6 reflectance centered at 740 nm (farRed2 band) and a baseline between red and NIR_2 bands. Accordingly, the second part of ABDI was designed to eliminate the impact of turbid water, which is achieved by subtracting half of the green band reflectance from the red band.

$$ABDI = [Band(redEdge2) - Band(red) - 0.375\,(Band(NIR_2) - Band(red))] - [Band(red) - 0.5Band(green)]$$

The constant of 0.375 in the above equation follows for the ratio

$$\frac{\lambda(redEdge2) - \lambda(red)}{\lambda(NIR_2) - \lambda(red)} = \frac{740 - 665}{865 - 665} = 0.375$$

- Note that ABDI applies to bands with different spatial resolution, e.e B3 is at 20 m while B4 is at 10 m. - Therefore a re-sampling is required before being able to apply the ABDI.

2.5 Resampling Sentinel-2 bands

[18]:
```
def get_ABDI(img):
    date = img.date()
    green_band = img.select('B3')
    red_band   = img.select('B4')
    redEdg2_band20   = img.select('B6')
    NIR2_20    = img.select('B8A')
    proj_10m = green_band.projection()#this is an object with all the␣
 ↪information about the projection needed for a 10m resolution

    redEdg2_band10 =  redEdg2_band20.resample('bilinear').reproject(proj_10m)
    NIR2_10 = NIR2_20.resample('bilinear').reproject(proj_10m)

    A =  redEdg2_band10.subtract(red_band)
    B =NIR2_10.subtract(red_band)
    C=A.subtract(B.multiply(0.375))
    D = red_band.subtract(green_band.multiply(0.5))
    abdi = C.subtract(D).rename('abdi').copyProperties(img, ["system:
 ↪time_start"])
    abdi.set({
        'system:time_start': date,
        'system:time_end': date,
        'year': date.get('year'),
        'month': date.get('month'),
        'date': date})
    return abdi
```

[19]:
```
ndwi=s2_boa_cloudfree.map(get_ndwi)
ndti=s2_boa_cloudfree.map(get_ndti)
ndci=s2_boa_cloudfree.map(get_ndci)
abdi=s2_boa_cloudfree.map(get_ABDI)
```

[22]:
```
val= cm.palettes.Spectral
palette=val.default

Vis = {
  'min': -1,
  'max': 1,
  'palette': palette}
zoom_scale = 12
```

```
u_lon= -2.93
u_lat =54.37
# establish a geometry feature from the point and the scale and center the map␣
 ↪to it
geometry = ee.Geometry.Point(u_lon,u_lat)


Map.centerObject(geometry,zoom_scale)
Map.addLayer(ndwi.first(), Vis, 'ndwi')
Map.addLayer(ndti.first(), Vis, 'ndti')
Map.addLayer(ndci.first(), Vis, 'ndci')
Map.addLayer(abdi.first(), Vis, 'abdi')
Map
```

```
Map(bottom=1762.0, center=[54.37, -2.93],␣
 ↪controls=(WidgetControl(options=['position', 'transparent_bg'], widg...
```

2.5.1 Mask out the non-water pixels

- Retrieve water pixels using the Normalized difference water index (NDWI)

- There are two ways to mask the time series of derived water quality indices:
  1- Make an average of "permanent" water pixels, by taking the average or median of NDWI. Then use the resulting "average/median" NDWI to mask non-water pixels.
  2- Mask each water quality image with the corresponding NDWI.

- In this exercise we will apply the second "more difficult" solution.
  a- join the image collection of the NDWI with the image collection of the water quality index, using the **join function**. b- to join two image collection, we define the index as the primary and the NDWI as the secondary source, in addition to the condition of joining (time).
  A join operation returns elements from the primary collection that match any element in the secondary collection according to the match condition in the filter.
  c- **saveFirst** is to add the first match to the collection as an additional property
  d- after joining the two collection, apply the function **s2_land_mask** which uses the NDWI to mask out the non-water pixels.

[104]:
```
join_imgs_coll = ee.ImageCollection(ee.Join.saveFirst('landMask').
 ↪apply(abdi,ndwi,ee.Filter.equals(leftField= 'system:time_start',rightField =␣
 ↪'system:time_start')))
def s2_land_mask(img):
    nowater = ee.Image(img.get('landMask')).gt(0)
    return img.updateMask(nowater)

masked_abdi = join_imgs_coll.map(s2_land_mask)
```

[147]:
```
Map.addLayer(masked_abdi.first().select('abdi'), Vis, 'Masked_abdi')
```

2.6 End of exercise assignment

You have generated four indices for S2-MSI: - use the normalized difference water index (NDWI) to mask the water pixels; - use the exercise of the previous week to compute the monthly and yearly mean in addition monthly climatology - evaluate the seasonality of each index; - evaluate how the indices are related to each other's.

[ ]: