Exercise: EO-Browser

<u>Code: Normalised Difference Snow Index, NDSI-Visualized, Sentinel 2</u>

General description of the script

The Sentinel-2 normalised difference snow index can be used to differentiate between cloud and snow cover as snow absorbs in the short-wave infrared light, but reflects the visible light, whereas cloud is generally reflective in both wavelengths. Snow cover is represented in bright vivid blue.
https://github.com/sentinel-hub/custom-scripts/tree/master/sentinel-2/ndsi-visualized

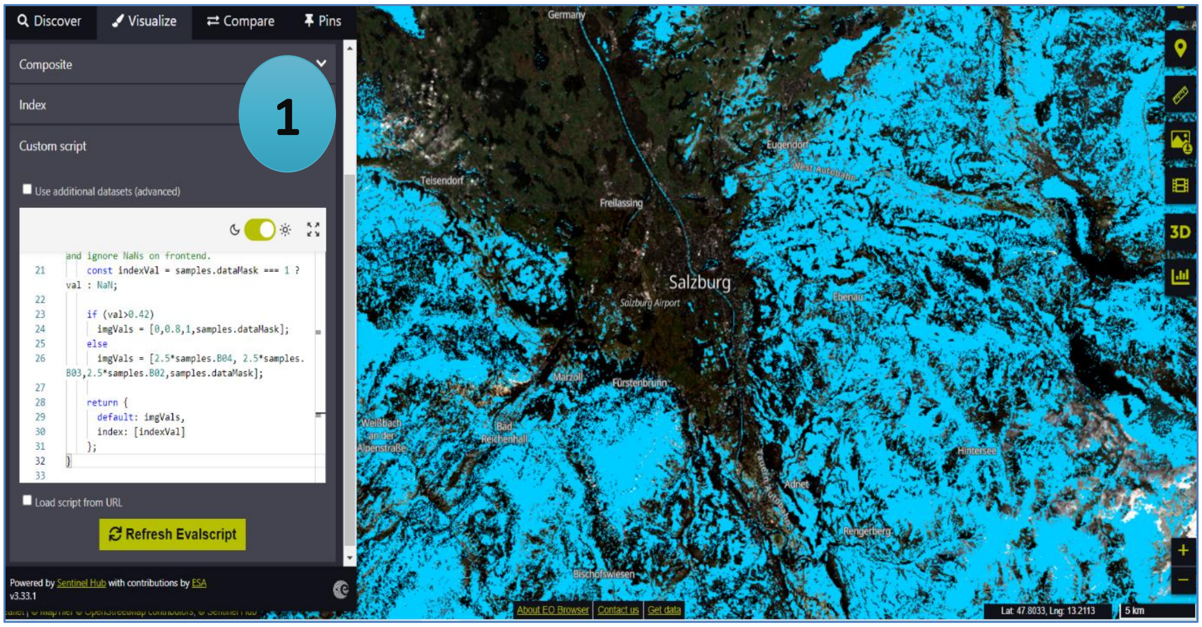
Z

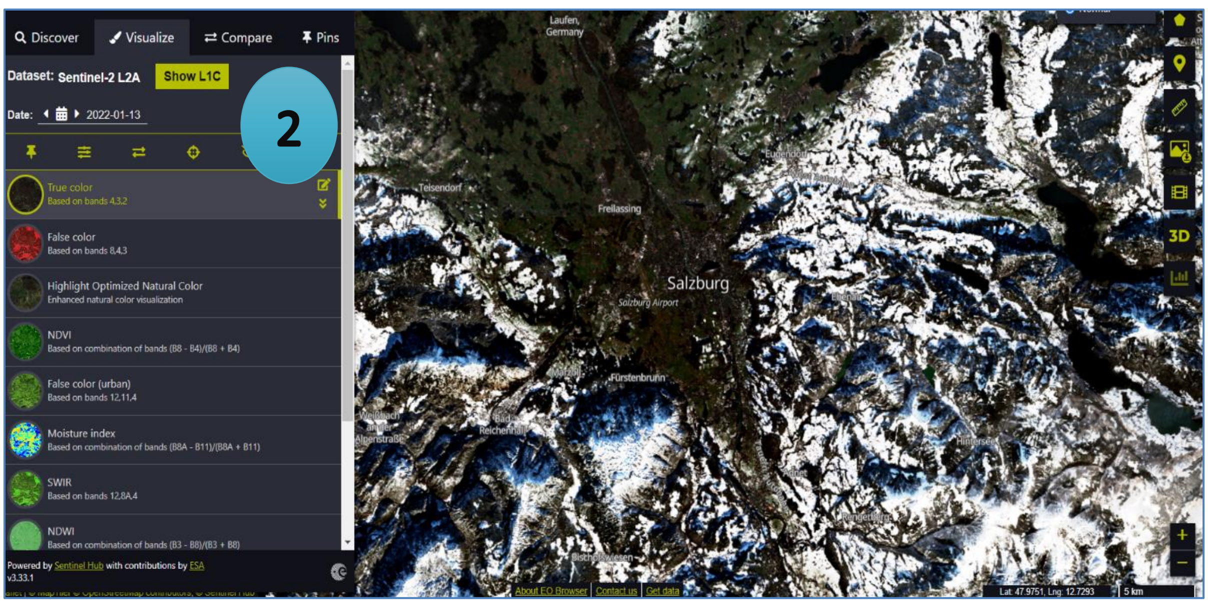
```
//VERSION=3
//Reference: https://earth.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-2a/algorithm

let viz = new Identity();

function setup() {
  return {
    input: ["B03", "B11","B04","B02","dataMask"],
     output: [
       { id: "default", bands: 4 },
       { id: "index", bands: 1, sampleType: "FLOAT32" }
     ]
  };
}

function evaluatePixel(samples) {
    let val = index(samples.B03, samples.B11);
        let imgVals = null;
    // The library for tiffs works well only if there is only one channel returned.
    // So we encode the "no data" as NaN here and ignore NaNs on frontend.
    const indexVal = samples.dataMask === 1 ? val : NaN;

    if (val>0.42)
      imgVals = [0,0.8,1,samples.dataMask];
    else
      imgVals = [2.5*samples.B04, 2.5*samples.B03,2.5*samples.B02,samples.dataMask];

        return {
      default: imgVals,
      index: [indexVal]
    };
}
```

The script returns the values of the four bands red, green, blue and data mask which will return value 0 for "no data" pixels and 1 elsewhere. Results from this script will present snow cover in bright vivid blue which derived from a ratio of two bands, one in the VIR (Band 3) and another one in the SWIR (Band 11). The NDSI output values range from -1 to 1 with values greater than 0.42 considered snow

Here is an NDSI formular: NDSI = $\frac{BAND3-BAND11}{BAND3+BAND11}$

Run the script at the region of Salzburg





The picture number 1 represents NDSI in Salzburg, Austria on 13th January 2022 when was the winter season, so there is high snow extent which is represented by bright blue color. When we look at picture number 2, the true color is related to the NDSI showing in the first picture, the areas covered with snow are shown blue or the NDSI value in the first image.

Script explanation

```
1  //VERSION=3
2  //Reference: https://earth.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-2a/algorithm
3
4  let viz = new Identity();
5
6  function setup() {
7    return {
8      input: ["B03", "B11","B04","B02","dataMask"],
9       output: [
10        { id: "default", bands: 4 },
11        { id: "index", bands: 1, sampleType: "FLOAT32" }
12       ]
13   };
14 }
```

1

This code is Evalscript V3 so we need to specify two functions; `setup` and `evaluatePixel`. The script part 1 above is about declaring variables and setting up input-output. We use `let` to declare variables that we can change later of our programme. For `function setup`, the script returns input and output.

Input are `BAND3`, `BAND11`, `BAND4`, `BAND2` and `dataMask` which `BAND3` and `BAND11` are for calculation NDSI Index. The natural color band combination uses `BAND4`(red), `BAND3`(green), `BAND2`(blue) channels. The last input is `dataMask` using for no data pixels that have value 0 and for 1 elsewhere. In this script, we use value from dataMask as the transparency band for all pixels laying outside of the requested image.

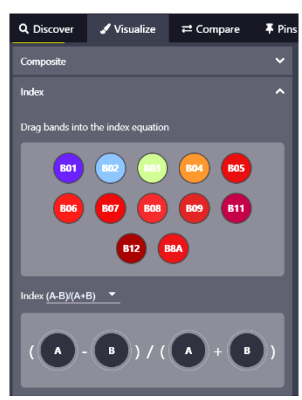
There are 2 output, the first output is `default` that consists of 4 bands and the second output is `index` that will return 1 band. The `sampleType` defines the returned raster sample type, FLOAT32 is a signed 32-bit floating point number.

```
15
16 function evaluatePixel(samples) {
17     let val = index(samples.B03, samples.B11);
18         let imgVals = null;
19     // The library for tiffs works well only if there is only one channel returned.
20     // So we encode the "no data" as NaN here and ignore NaNs on frontend.
21     const indexVal = samples.dataMask === 1 ? val : NaN;
```

2

This part the script will calculate the output values for each pixel by function `evaluatePixel`. In the line 17, the script declares variables `val` which is number of the input value to be mapped. `Index` will calculate difference divided by sum(you can check this index's equation in Visualize tab on EO Browser), for this script returns NDSI value from index (BAND3 – BAND11) / (BAND3 + BAND11).

In line 21, The value from this line cannot be changed because using `const` variable. There is a statement allowing you to check expressions of `dataMask` value (0 or 1).



So this means if a value of dataMask is 1, `indexVal` will return variables `val` (NDSI value), but if dataMask does not equal 1, `indexVal` will return `NaN` (no data). Therefore, if a pixel that has no data or no NDSI value, it will be transparent for that area but only snow index will be shown. The value from this line cannot be changed because using const variable.

```
    if (val>0.42)
      imgVals = [0,0.8,1,samples.dataMask];
    else
      imgVals = [2.5*samples.B04, 2.5*samples.B03,2.5*samples.B02,samples.dataMask];

        return {
      default: imgVals,
      index: [indexVal]
    };
}
```

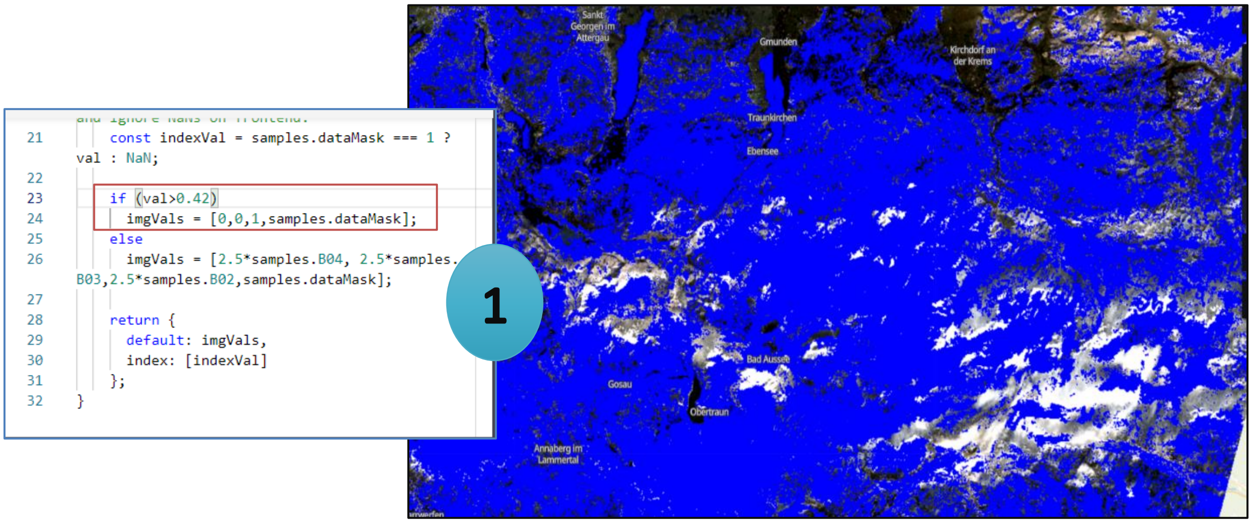
3

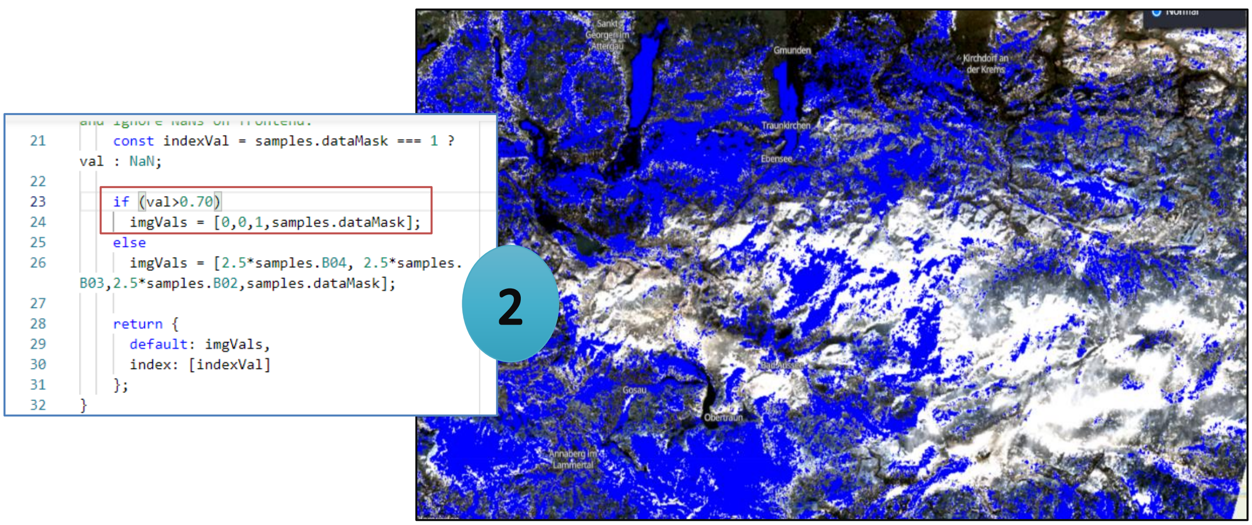
The last part of the script, if a `val` value is more than 0.42 which considered as snow, then `imgVals` returns 0(red), 0.8(green), 1(blue) and dataMask. But if the `val` value is something else, returns natural color which are BAND4,3,2 and dataMask.

As I mentioned above about the 2 outputs, `imgVals` is in the `default` variable which is defined in the output 4 bands(BAND2,3,4,11). `IndexVal` is in the `index` variable that will return 1 band(dataMask)

Apply the scripts

1. Change NDSI threshold value and color



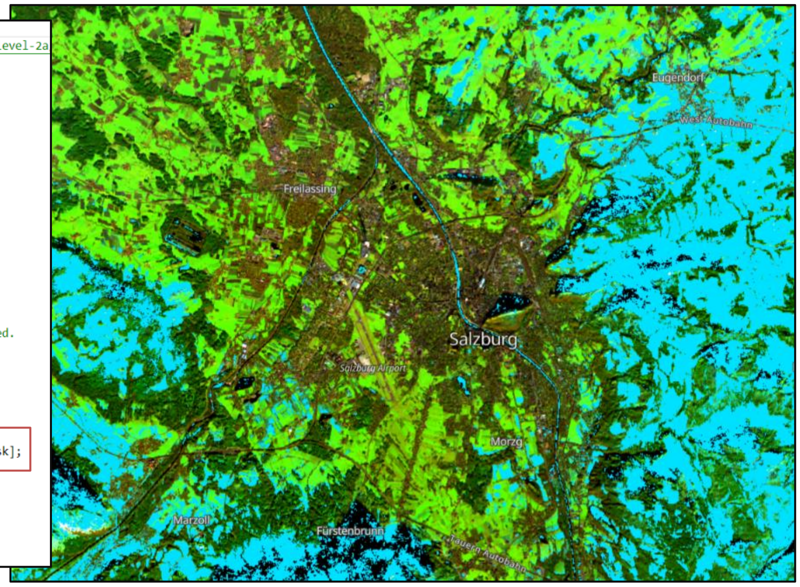


According to NDSI threshold value is commonly assumed to be 0.40(Dozier, 1989; Hall and Riggs, 2007; Sankey et al., 2015) but some literature is 0.7(Maher et al., 2012). I would like to see the difference between these 2 values so I change val value from 0.42(picture no.1) to 0.70(picture no.2). We can see the result that snow areas are reduced from value 0.7 because the number of pixels value 0.7 less than 0.4, so we see less snow in the picture no.2.

Adjusting colors can be done by changing value on RGB values. In this case, I would like to visualize snow to Dark Blue so I set Red/Green band to 0 and Blue Band to 1.

2. Adjust Band Combinations

```
//VERSION=3
//Reference: https://earth.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-2a

let viz = new Identity();

function setup() {
  return {
    input: ["B03", "B11","B08","B02","dataMask"],
     output: [
      { id: "default", bands: 4 },
      { id: "index", bands: 1, sampleType: "FLOAT32" }
     ]
  };
}

function evaluatePixel(samples) {
    let val = index(samples.B03, samples.B11);
    let imgVals = null;
    // The library for tiffs works well only if there is only one channel returned.
    // So we encode the "no data" as NaN here and ignore NaNs on frontend.
    const indexVal = samples.dataMask === 1 ? val : NaN;

    if (val>0.42)
      imgVals = [0,0.8,1,samples.dataMask];
    else
      imgVals = [2.5*samples.B11, 2.5*samples.B08,2.5*samples.B02,samples.dataMask];

    return {
      default: imgVals,
      index: [indexVal]
    };
}
```



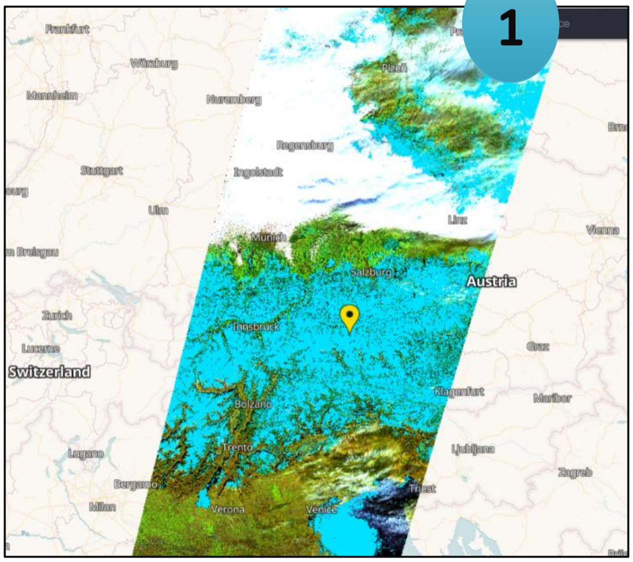
In this part I would like to visualize the areas where values are not in the range of NDSI, so I change from natural color (Band4,3,2) to agriculture band combination(Band11,8,2) because I want to explore the health of crops and these bands are particularly good at highlighting dense vegetation.

The first step is changing Band in the input, Band 8 is the new input so I replace band8 to band4. Next, reordering bands in the statement `imgVals = [2.5*samples.B11, 2.5*samples.B08,2.5*samples.B02,samples.dataMask].`

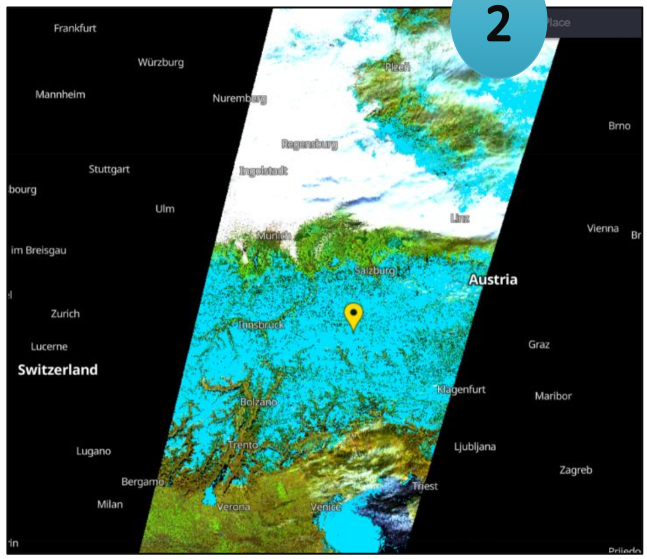
The image will display green color where dense vegetation will be dark green.

3. Delete Data Mask

1



2




```
//VERSION=3
//Reference: https://earth.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-2a/algorithm

let viz = new Identity();

function setup() {
  return {
    input: ["B03", "B11","B08","B02"],
     output: [
       { id: "default", bands: 3 },
     ]
  };
}

function evaluatePixel(samples) {
    let val = index(samples.B03, samples.B11);
    let imgVals = null;
    // The library for tiffs works well only if there is only one channel returned.
    // So we encode the "no data" as NaN here and ignore NaNs on frontend.

    if (val>0.42)
      imgVals = [0,0.8,1];
    else
      imgVals = [2.5*samples.B11, 2.5*samples.B08,2.5*samples.B02];

    return {
      default: imgVals,
    };
}
```

In this part, I want to compare the difference between having dataMask and not having it. So, I modified the code and removed the related to dataMask part.

From the result of Figure 2, areas outside the boundaries of the satellite image are solid black, unlike the figure1 that contains dataMask where all pixels laying outside of the image are transparent so we still can see locations nearby from the based map.