Generating a cloud-free, homogeneous Landsat-8 mosaic of Switzerland using Google Earth Engine

Authors: Sophie C. Stuhler, Reik Leiterer, Philip C. Joerg, Hendrik Wulf, Michael E. Schaepman

Remote Sensing Laboratories
Geographisches Institut
Universität Zürich
Winterthurerstr. 190
8057 Zürich

www.npoc.ch
npoc@geo.uzh.ch
April 2016

1 Introduction

The Swiss National Point of Contact for Satellite Images (NPOC) offers information on data access and processing as well as support and consulting towards their use to Swiss stakeholders. The derivation of prototypes and the process documentation is one of the main services. The here presented report on the automatic mosaicking of Landsat-8 data over Switzerland with the web tool '*Google Earth Engine*' (GEE) gives an overview of preprocessing steps as well as used algorithms and their functionality. In addition, it explains the goals, prospects and limitations of the software and used methods.

Cloud-free optical satellite images are often needed for either representation or derivation of land cover information. The goal of this project was to automatically derive a cloud-free, homogeneous, up-to-date mosaic of Switzerland and its surroundings. The mosaicking algorithm is implemented as a script in GEE, which can be distributed to stakeholders for further modification.

2 Google Earth Engine (GEE)

GEE is a free and web-based cloud computing platform for processing Earth observation data. GEE provides currently over 11 PB of earth observation data and many powerful servers to run existing and own scripts of the GEE community. GEE has a user-friendly interface (Fig. 1) consisting of a code editor panel (middle panel), one panel giving an overview about datasets, algorithms and scripts (left panel), and one panel for detailed information, such as pixel values, printing and open tasks (right panel). In addition, there is a map panel where layers can be displayed over Google Maps (meaning everything needs to be georeferenced). The EarthEngine with Code Editor can be called via: https://code.earthengine.google.com/. A detailed guide to the Code Editor can be found here: https://developers.google.com/earth-engine/playground.

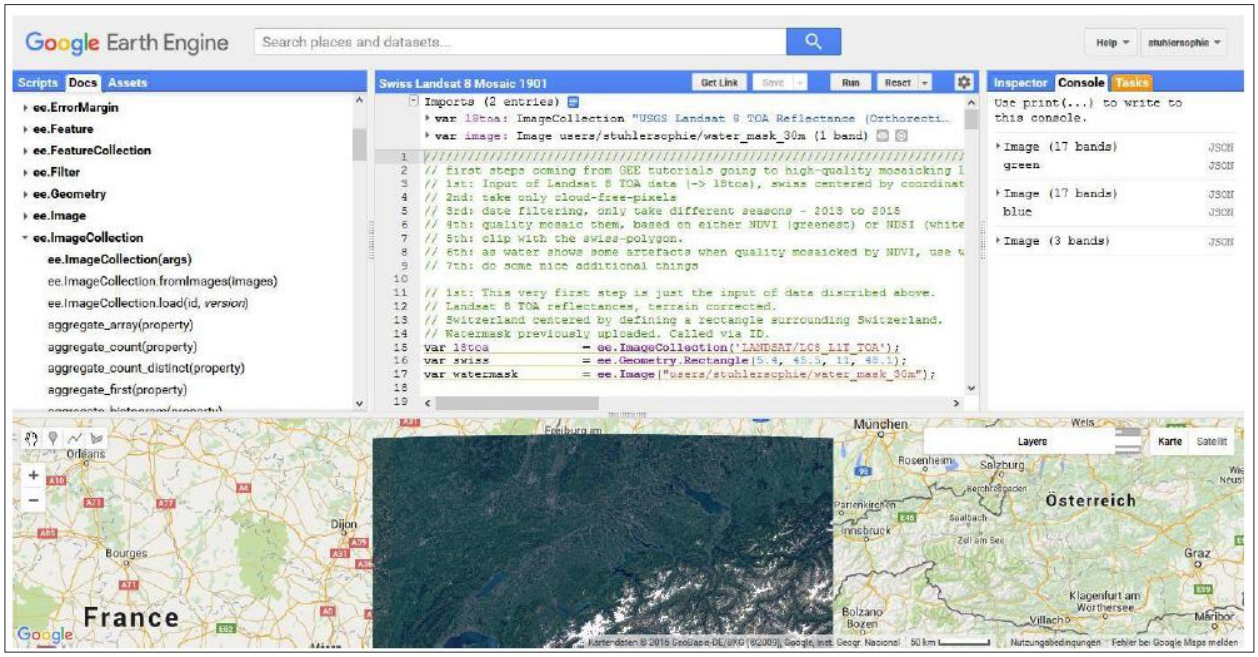

Figure 1: GEE interface with the main working panels on the top and the mapping panel on the bottom.

Working with GEE requires basic programming skills, as it is based on JavaScript. The main principles of GEE are defining variables, function and expressions, which can be defined by "var *variable*". New variables need to start with a minuscule, but may contain numbers and majuscules as well. Earth observation data can be searched via the upper panel that resembles the Google-Maps-Search and then called via their ID.

Defining a variable can look like this:

var *l8toa* = ee.ImageCollection('LANDSAT/LC8_L1T_TOA')

Strings can either be called via single or double quotes ("" or ''). Algorithms can be called via their full address:

var *merged* = ee.ImageCollection(*collection1*).merge(*collection2*)

Or as a direct call after the variable:

var *merged* = *collection1*.merge(*collection2*)

Functions can either be applied as:

var *funcapplied* = function (*imagecollection*)

or as:

var *funcapplied* = ee.ImageCollection(*imagecollection*).map(function).

Comments can be written with double slashes (//). Functions can be called via:

var *added* = function (*image*) {return *image*.addBands(…)}

expressions are called with:

var *dvi* = *image*.expression ('nir – red');

where the used variables also need to be defined.

For more detailed tutorials see: https://developers.google.com/earth-engine/tutorial_api_01.

3 Data

To generate a recent satellite mosaic on GEE, Landsat-8 data were chosen. Sentinel-2 data would have been even more adequate for European/Swiss purposes but was not available on GEE at the time of this project. As most of the implemented algorithms in GEE require *Top of the Atmosphere Reflectance* (TOA) data format, the TOA Landsat-8 data collection was used.

Because open water bodies were subject to artifacts when using the predefined mosaic algorithms that target land areas, we used an additonal water mask. This water mask was derived from an image collection of mostly cloud-free and haze-poor images, which were mosaicked as a most-recent mosaic. The *Normalized Difference Water Index* (NDWI) was added and a threshold of 0.3 was found to discriminate well between water and land. Based on this approach, a binary water mask was created and loaded into the script for the Switzerland mosaic.

4 GEE script

The overall idea was to use different quality bands for mosaicking, i.e. that the pixel with the highest value in the quality band will be used in the resulting mosaic. This procedure was implemented for each season (winter, spring, summer and autumn) during the years 2013-2015, respectively.

The first step was to load the Landsat-8 TOA collection for the defined area of interest (Switzerland).

```
var l8toa = ee.ImageCollection ('LANDSAT/LC8_L1T_TOA')

var swiss = ee.Geometry.Rectangle (4.6, 45.3, 11.9, 48.3)
```

The previously generated watermask was called via an ID.

```
var watermask = ee.Image ("users/stuhlersophie/water_mask_30m")
```

The next step was the development of a cloud-masking function by assigning a cloud score to each pixel in the Landsat-8 collection. We used an internal function implemented in GEE which assigned this cloud score band based on different cloud criteria (‘ee.Algorithms.Landsat.simpleCloudScore’). The criterias were the following: brightness in blue band, brightness in all visible bands, brightness in all infrared bands and low temperatures in thermal band.

To discriminate clouds from snow the *Normalized Difference Snow Index* (NDSI) was used as an additional criterion in the function. In this case, pixels showing a higher cloud score than 25 were masked out, leading to a masking of the inner cores of clouds. Haze and thin clouds could – in some cases – still pass. The lower the cloud score threshold was put, the more single pixels are masked out, which led to some NA-values in the resulting image. The cloud score threshold of 25 was therefore found to work best, letting some single outliers still pass and though masking clouds well enough.

```
var maskClouds = function(l8toa) {
        var scored = ee.Algorithms.Landsat.simpleCloudScore(l8toa)
        return l8toa.mask(scored.select(['cloud']).lt(25))
}
```

The next part was to create a function, adding the quality bands to the images as described above. The quality band was needed to put the best (qualitatively best) as highest value. For this, the *Normalized Difference Vegetation Index* (NDVI) was selected by using the internal function ‘normalizedDifference’. The acquisition time was further added as quality band, which could be used for creating a “most-recent mosaic”. This information was taken out of the Landsat-8 metadata. Also based on the metadata was the information about cloudiness.

```
var addQualityBands = function(l8toa) {
        var one = ee.Image(1)
        return maskClouds(l8toa)
                .addBands (l8toa.normalizedDifference (['B5', 'B4']) .rename ('ndvi'))
                .addBands (l8toa.metadata ('system:time_start'))
                .addBands (one.divide (l8toa.metadata ('CLOUD_COVER')) .rename ('inv_cloudiness'))
}
```

In addition, the NDSI and the *Normalized Difference Water Index* (NDWI) were added as aditional bands to the individual L8 scences. *(This could have been done later - after the mosaicking itself - but if it done at this point the user could try out to modify the script to take these indices as quality bands)*

```
var addNDSI = function(l8toa) {
        return maskClouds(l8toa)
        .addBands(l8toa.normalizedDifference(['B3', 'B6']).rename('ndsi'))
}

var addNDWI = function(l8toa) {
        return maskClouds(l8toa)
        .addBands(l8toa.normalizedDifference(['B3', 'B5']).rename('ndwi'))
}
```

Before applying the functions created above, the Landsat-8 collection was filtered season-wise and year-wise and afterwards merged together, so that four seasonal collections were built (winter, spring, summer, autumn) to which the functions were then applied.

```
var l8toawinter2013 = l8toa.filterDate('2013-11-21', '2014-02-20')

var l8toawinter2014 = l8toa.filterDate('2014-11-21', '2015-02-20')

var l8toawinter2015 = l8toa.filterDate('2015-11-21', '2016-01-11')

var l8toawinter =
        ee.ImageCollection (l8toawinter2013.merge(l8toawinter2014).merge(l8toawinter2015))
        .map(addQualityBands)
        .map(addNDSI)
        .map(addNDWI)

var l8toaspring2013 = l8toa.filterDate('2013-04-07', '2013-06-20')

var l8toaspring2014 = l8toa.filterDate('2014-02-21', '2014-06-20')

var l8toaspring2015 = l8toa.filterDate('2015-02-21', '2015-06-20')

var l8toaspring =
        ee.ImageCollection (l8toaspring2013.merge(l8toaspring2014).merge(l8toaspring2015))
        .map(addQualityBands)
        .map(addNDSI)
        .map(addNDWI)

var l8toasummer2013 = l8toa.filterDate('2013-06-21', '2013-09-20')

var l8toasummer2014 = l8toa.filterDate('2014-06-21', '2014-09-20')

var l8toasummer2015 = l8toa.filterDate('2015-06-21', '2015-09-20')

var l8toasummer =
        ee.ImageCollection (l8toasummer2013.merge(l8toasummer2014).merge(l8toasummer2015))
        .map(addQualityBands)
        .map(addNDSI)
        .map(addNDWI)

var l8toaautumn2013 = l8toa.filterDate('2013-09-21', '2013-11-20')

var l8toaautumn2014 = l8toa.filterDate('2014-09-21', '2014-11-20')
```

```
var l8toaautumn2015 = l8toa.filterDate('2015-09-21', '2015-11-20')

var l8toaautumn =
        ee.ImageCollection (l8toaautumn2013.merge(l8toaautumn2014).merge(l8toaautumn2015))
        .map(addQualityBands)
        .map(addNDSI)
        .map(addNDWI)
```

Because water bodies had different characteristics in all quality bands, a different decision rule was applied for water pixels: The seasonal filtering approach was slightly modified and different quality bands were added. Finally, scenes with a cloud cover of more than 45% were also filtered to obtain a homogenous, still cloud-free mosaic over regions where the NDVI as a quality band worked unsufficiently well, such as for water, bare soil and rock. The cloud cover threshold was estimated based on a try-and-error approach and can therefore not be transferred to other areas directly. However, a cloud cover threshold should be used in any case to reduce differences in reflectance between scene boundaries over lakes.

```
var l8toawater2013 = l8toa.filterDate('2013-05-07', '2013-10-20')

var l8toawater2014 = l8toa.filterDate('2014-04-21', '2014-09-20')

var l8toawater2015 = l8toa.filterDate('2015-05-21', '2015-11-20')

var l8toawater =
        ee.ImageCollection (l8toawater2013.merge(l8toawater2014).merge(l8toawater2015))
        .map(addQualityBands)
        .map(addNDSI)
        .map(addNDWI)
        .filterMetadata('CLOUD_COVER', 'greater_than', 0.45)
```

Based on the algorithm “qualityMosaic” the added quality bands were then used to create a spatially homogenious mosaic. For all seasons, the NDVI was used for the generation of the primary mosaic. In areas where the NDVI worked insufficiently well, we used the previously described band for cloudiness.

```
var greenestwinter = l8toawinter.qualityMosaic ('ndvi')

var greenestspring = l8toaspring.qualityMosaic ('ndvi')

var greenestsummer = l8toasummer.qualityMosaic ('ndvi')

var greenestautumn = l8toaautumn.qualityMosaic ('ndvi')

var bluestrec = l8toawater.qualityMosaic ('inv_cloudiness')
```

In addition, the NDSI was used to create a “whitest” pixel composite for winter.

```
var whitestwinter = l8toawinter.qualityMosaic ('ndsi')
```

To reduce storage and to focus on Switzerland, the created mosaics were afterwards clipped by the area of Switzerland. The area of Switzerland and its surrounding was represented by a rectangle defined by tie points in Geographic Coordinates. This target area (45.3°N - 48.3°N, 4.6°E - 11.9°E) included all larger swiss lakes, such as Lake Geneva, Lake Constance and Lago Maggiore.

(The function ".clip" only works on images, not on image collections and was therefore applied right after the quality mosaicking)

```
var greenestCompositewinter = greenestwinter.clip (swiss)
var greenestCompositespring = greenestspring.clip (swiss)
var greenestCompositesummer = greenestsummer.clip (swiss)
var greenestCompositeautumn = greenestautumn.clip (swiss)
var whitestCompositewinter = whitestwinter.clip (swiss)
var bluestComposite = bluestrec.clip (swiss)
```

As indicated above water bodies needed to be mosaiced differently as compared to land areas. Using the NDVI as a quality band resulted in the more likely depiction of clouds pixels rather than water, because the NDVI of clouds is higher (around zero) as the NDVI of water (usually negative).

Though clouds were masked out in the beginning, haze and the thinner parts of clouds could still pass. It was not possible to define the cloud score low enough to leave only really cloud-free pixels in the end, as there are always some atmospherical particles that contribute to a higher NDVI that resulted in artifacts. Therefore we opted for a different decision rule, i.e. another quality band, over water areas. A similar problem also occurred over rocks and bare soil. Despite the use of a water mask for water bodies, we used an additional NDVI threshold to further mask rock/soil areas. For these masked areas the previously described mosaic band based on the cloudiness was used. It was implemented with a "where"-operator. The NDVI-threshold was set to 0.3 for spring and summer and to 0.5 for autumn. Due to snow presence the above described water mask was used for winter mosaics. Where NDVI fell below the given threshold, the mosaic with the lowest cloudiness was taken. This mosaic worked well over water and land.

```
var ndvispring = greenestCompositespring.select ('ndvi')
var ndvisummer = greenestCompositesummer.select ('ndvi')
var ndviautumn = greenestCompositeautumn.select ('ndvi')
var ndviwintergreen = greenestCompositewinter.select ('ndvi')
var ndviwinterwhite = whitestCompositewinter.select ('ndvi')
var springmosaic = greenestCompositespring.where(ndvispring.lt(0.3),bluestComposite)
var summermosaic= greenestCompositesummer.where(ndvisummer.lt(0.3),bluestComposite)
var autumnmosaic = greenestCompositeautumn.where(ndviautumn.lt(0.5),bluestComposite)
var greenwintermosaic= greenestCompositewinter.where(watermask.eq(1),bluestComposite)
var whitewintermosaic= whitestCompositewinter.where(watermask.eq(1),bluestComposite)
```

As an example, you can see in Figure 2 the improvement for the area of Lake Zürich.

Figure 2. Spring mosaic of Lake Zurich based on NDVI (top) only and on NDVI and least-cloudiness for pixels with NDVI < 0.3 (bottom).

Based on the various mosaics detailed above, further calculations and analysis can be done. In the following, we describe a simple pan-sharpening approach and how the *Enhanced Vegetation Index* (EVI) was calculated.

The EVI was calculated only for spring, but could be easily calculated for any other season. In this case the red, near infrared- and blue band were used for the calculation.

```
var evispring = springmosaic.expression(
        '2.5 * (nir - red) / (nir + 6 * red - 7.5 * blue + 1)'
        {
        red: springmosaic.select('B4'), // 620-670nm, RED
        nir: springmosaic.select('B5'), // 841-876nm, NIR
        blue: springmosaic.select('B2') // 459-479nm, BLUE
        })
    )
```

Pan-sharpening is an image enhancement method based on three RGB bands with lower spatial resolution (here 30 m) and one panchromatic band with a higher spatial resolution (here 15 m). In the basic pan-sharpening approach the bands red, green and blue are transformed from RGB color space into HSV color space (hue, saturation, value), where the value band gets swapped with the panchromatic band. Afterwards the hue-saturation-panchromatic color space is retransformed to RGB. This was implemented in a function, that was afterwards applied to each mosaic. *(As pan-sharpening can lead to an incorrect spectral information, this approach is mostly used for visualization purposes. Further spectral analyses are not advised.)*

```
var pansharp = function (img) {
        var rgb = img.select(['B2', 'B3', 'B4']).float()
        var gray = img.select('B8').float()
        var huesat = rgb.rgbToHsv().select('hue', 'saturation')
        return ee.Image.cat(huesat, gray).hsvToRgb()
        };
var panspring = pansharp(springmosaic);
var pansummer = pansharp(summermosaic);
var panautumn = pansharp(autumnmosaic);
var pangreenwinter = pansharp(greenwintermosaic);
var panwhitewinter = pansharp(whitewintermosaic);
```

5 Mosaic and additional products

In this chapter, the mosaic-derived products are presented. The final, cloud-free summer mosaic for the area of interest is shown in Figure 3, a „whitest“ mosaic (i.e. maximum snow coverage) in Figure 4.

Figure 3. Cloud-free summer mosaic of Switzerland (RGB-visualization).

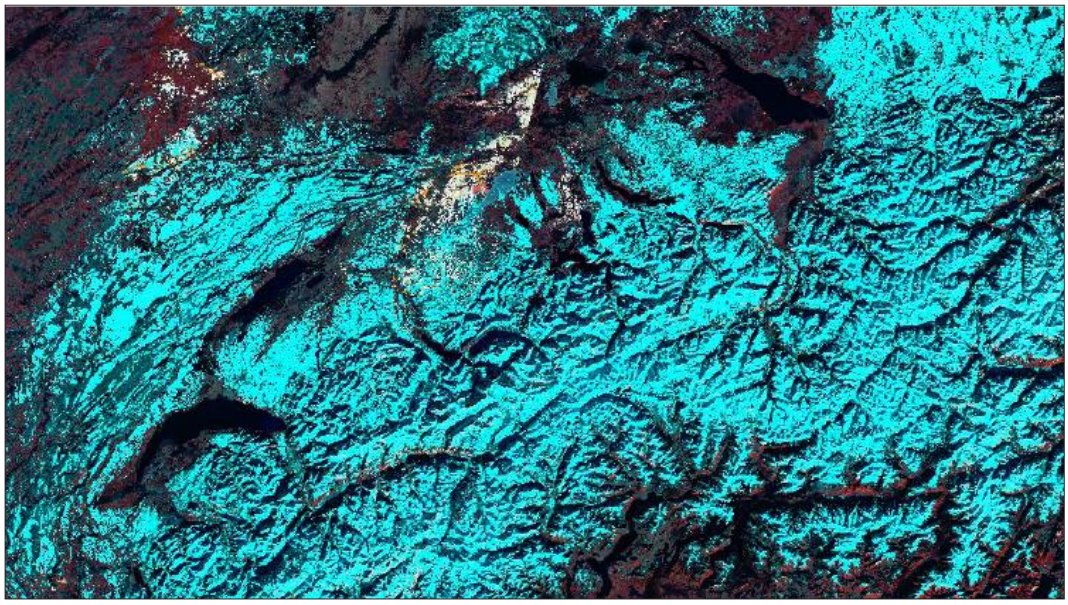

Figure 4: Maximum snow cover mosaicked with NDSI as quality band (CIR-visualization: B6, B4, B3).

The EVI based on the spring mosaic is shown in Figure 5.

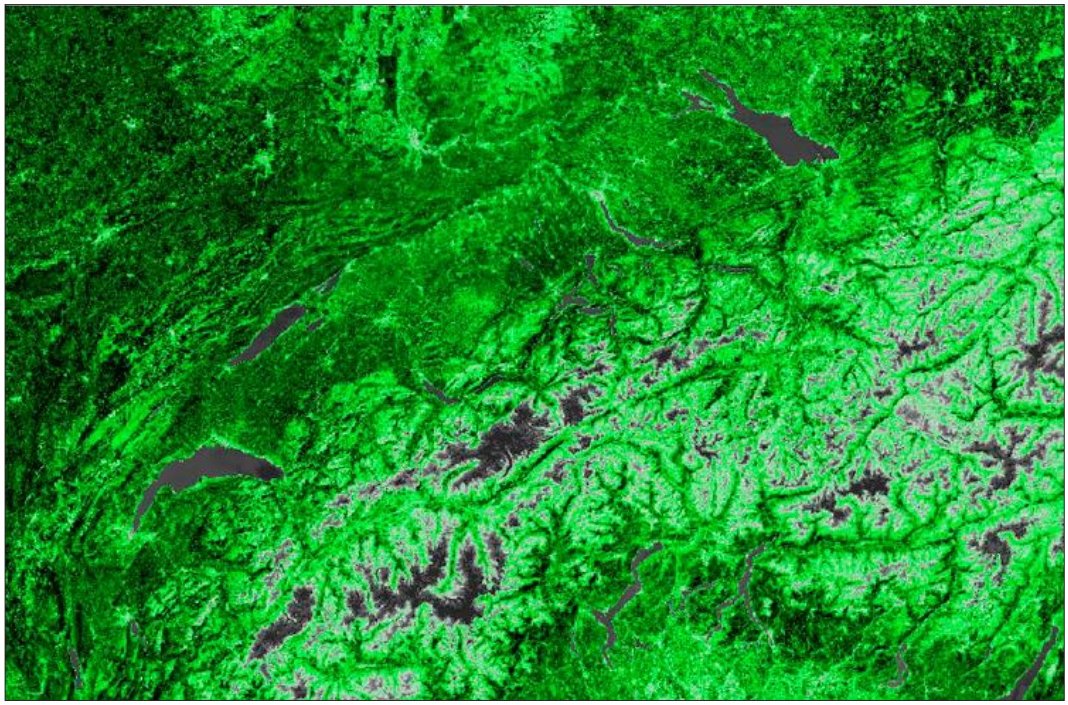

Figure 5: EVI calculated on the spring mosaic, colors range from gray (-1) to dark green (1).

6 Additional informations

For helpful information it is suggested to follow the discussions in the GEE developers forum (https://groups.google.com/forum/#!forum/google-earth-engine-developers) where not only the users but also the engineers from Google reply frequently. Furthermore, we recommend using the online tutorials (https://developers.google.com/earth-engine/) which present basic codes and explain basic functions. As GEE engineers actively work and improve the GEE-API available functions can change over time.

Though most functions could be opened in the scripts panel, some of them appear to be black-box-systems, as their code cannot be read and the functionality is insufficiently explained.

Exporting works via Google Drive (free until 15 GB), which is long winded, as it needs to be loaded to Google Drive first and then downloaded, both can be time consuming.

Script

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// first steps coming from GEE tutorials going to high-quality mosaicking LS8 for Switzerland: one dataset,
summertime (high vegetation), origin of pixels may be either 2013, 2014 or 2015
// 1st: Input of Landsat 8 TOA data (-> l8toa), swiss centered by coordinates (-> swiss) -> rectangle
// 2nd: take only cloud-free-pixels
// 3rd: date filtering, only take different seasons - 2013 to 2015
// 4th: quality mosaic them, based on either NDVI (greenest) or NDSI (whitest, nice example for winter)
// 5th: clip with the swiss-polygon.
// 6th: as water shows some artefacts when quality mosaicked by NDVI, use watermask (water)/use
threshold in NDVI, wherever water is masked use the inverse cloudiness-band as quality feature
// 7th: do some nice additional things

// 1st: This very first step is just the input of data discribed above ---------------------------------------------------------
// Landsat 8 TOA reflectances, terrain corrected.
// Switzerland centered by defining a rectangle surrounding Switzerland.
var l8toa            = ee.ImageCollection('LANDSAT/LC8_L1T_TOA');
var swiss             = ee.Geometry.Rectangle(5.4, 45.5, 11, 48.1);

// Creating the watermask
// This function masks clouds and adds quality bands to Landsat 8 images.
var addQualityBands = function(l8toa) {
  return maskClouds(l8toa)
    // NDWI
    .addBands(l8toa.normalizedDifference(['B3', 'B5']).rename('ndwi'))
    // time in days
    .addBands(l8toa.metadata('system:time_start'))
};

// spring. please adjust to data availability.
var l8toaspring2013   = l8toa.filterDate('2013-04-11', '2013-09-20');
var l8toaspring2014   = l8toa.filterDate('2014-04-21', '2014-09-20');
var l8toaspring2015   = l8toa.filterDate('2015-04-21', '2015-09-20');
var l8toaspring       = ee.ImageCollection(l8toaspring2013.merge(l8toaspring2014).merge(l8toaspring2015))
    .map(addQualityBands);

// Create a greenest pixel composite for each season.
var recentspring  = l8toaspring.qualityMosaic('ndwi');

// Create a water mask based on NDWI values greater than 0.1 .
var ndwi      = recentspring.select(['ndwi']);
function maskWater(ndwi)
{
  var thresh = 0.22;
  return ee.Image(1)
        .and(ndwi.gte(thresh));
}

var watermask = maskWater(ndwi);
```

```
// 2nd: cloud-masking ----------------------------------------------------------------------------------------------------------
// This function masks clouds in Landsat 8 imagery.
// Pixels showing a Cloud Score greater than 25 will be masked out.
// Cloud Score Algorithm scores cloud criteria, such as:
// brightness in blue band, brightness in all visible bands, brightness in all Infrared-bands, low temperatures
in thermal band.
// furthermore Cloud Score discriminates clouds from snow by using a Snow Index (NDSI).
var maskClouds = function(l8toa) {
  var scored = ee.Algorithms.Landsat.simpleCloudScore(l8toa);
  return l8toa.mask(scored.select(['cloud']).lt(25));
};

// This function will add quality bands to the cloud-free images (maskClouds function from above), which will
be used for quality mosaicking;
// Quality bands:
// Normalized Difference Vegetation Index (NDVI), sensitive to photosynthesis activity;
// Acquisition time (system:time_start) from metadata;
// Inverse Cloud Cover (inv_cloudiness)from metadata.
var addQualityBands = function(l8toa) {
  var one       = ee.Image(1);
  return maskClouds(l8toa)
    // NDVI
    .addBands(l8toa.normalizedDifference(['B5', 'B4']).rename('ndvi'))
    .addBands(l8toa.metadata('system:time_start'))
    .addBands(one.divide(l8toa.metadata('CLOUD_COVER')).rename('inv_cloudiness'))
};

// Could also be done later, but if you're interested you can also try quality mosaciking based on NDWI or
NDSI. See what it looks like.
// This function masks clouds and adds a NDSI band to Landsat 8 images.
var addNDSI = function(l8toa) {
  return maskClouds(l8toa)
    // NDSI
    .addBands(l8toa.normalizedDifference(['B3', 'B6']).rename('ndsi'))
};

// This function masks clouds and adds a NDWI band to Landsat 8 images.
var addNDWI = function(l8toa) {
  return maskClouds(l8toa)
    // NDSI
    .addBands(l8toa.normalizedDifference(['B3', 'B5']).rename('ndwi'))
};
```

```
// 3rd: date filtering ----------------------------------------------------------------------------------------------------------
// Create a collection of seasonal images from 2013-2015.
// 3 years, but one mosaic for the season.
// i.e. one l8toawinter actually contains winter from 3 years.
// filter yearwise, merge afterwards into 1 collection per season.

// winter, data availability for winter :-/ adjust after winter 2015/16
var l8toawinter2013   = l8toa.filterDate('2013-11-21', '2014-02-20');
var l8toawinter2014   = l8toa.filterDate('2014-11-21', '2015-02-20');
var l8toawinter2015   = l8toa.filterDate('2015-11-21', '2016-01-19');   // please adjust to data availability.
var l8toawinter       = ee.ImageCollection(l8toawinter2013.merge(l8toawinter2014).merge(l8toawinter2015))
   .map(addQualityBands)
   .map(addNDSI)
   .map(addNDWI);

// spring.
var l8toaspring2013   = l8toa.filterDate('2013-04-07', '2013-06-20');  // L8 data currently available from
04/07/2013
var l8toaspring2014   = l8toa.filterDate('2014-02-21', '2014-06-20');
var l8toaspring2015   = l8toa.filterDate('2015-02-21', '2015-06-20');
var l8toaspring       = ee.ImageCollection(l8toaspring2013.merge(l8toaspring2014).merge(l8toaspring2015))
   .map(addQualityBands)
   .map(addNDSI)
   .map(addNDWI);

// summer.
var l8toasummer2013   = l8toa.filterDate('2013-04-21', '2013-10-20');
var l8toasummer2014   = l8toa.filterDate('2014-04-21', '2014-10-20');
var l8toasummer2015   = l8toa.filterDate('2015-04-21', '2015-10-20');
var l8toasummer       =
ee.ImageCollection(l8toasummer2013.merge(l8toasummer2014).merge(l8toasummer2015))
   .map(addQualityBands)
   .map(addNDSI)
   .map(addNDWI);

// autumn.
var l8toaautumn2013   = l8toa.filterDate('2013-09-21', '2013-11-20');
var l8toaautumn2014   = l8toa.filterDate('2014-09-21', '2014-11-20');
var l8toaautumn2015   = l8toa.filterDate('2015-09-21', '2015-11-20');
var l8toaautumn       =
ee.ImageCollection(l8toaautumn2013.merge(l8toaautumn2014).merge(l8toaautumn2015))
   .map(addQualityBands)
   .map(addNDSI)
   .map(addNDWI);


// filtered for water mosaic. wider time range, but water will look the same in al images
var l8toawater2013    = l8toa.filterDate('2013-05-07', '2013-10-20');
var l8toawater2014    = l8toa.filterDate('2014-04-21', '2014-09-20');
var l8toawater2015    = l8toa.filterDate('2015-05-21', '2015-11-20');
var l8toawater        = ee.ImageCollection(l8toawater2013.merge(l8toawater2014).merge(l8toawater2015))
```

```
    .map(addQualityBands)
    .map(addNDSI)
    .map(addNDWI)
    .filterMetadata('CLOUD_COVER', 'greater_than', 0.45);


// 4th: quality mosaicking ----------------------------------------------------------------------------------------------------------
// Take one band as criterion for which pixel should be taken for the Mosaic (from Image Collection to single
Image)
// e.g. quality band NDVI: the pixel with the highest NDVI-value out of all used scenes will be taken
// Create a greenest pixel composite for each season.
var greenestwinter  = l8toawinter.qualityMosaic('ndvi');
var greenestspring  = l8toaspring.qualityMosaic('ndvi');
var greenestsummer  = l8toasummer.qualityMosaic('ndvi');
var greenestautumn  = l8toaautumn.qualityMosaic('ndvi');
var bluestrec       = l8toawater.qualityMosaic('inv_cloudiness');

// Create a "whitest" pixel composite for winter, based on highest snow presence -> NDSI.
var whitestwinter   = l8toawinter.qualityMosaic('ndsi');

// 5th clipping ---------------------------------------------------------------------------------------------------------------------
// clip the Mosaics to some wide region around Switzerland.
var greenestCompositewinter = greenestwinter.clip(swiss);
var greenestCompositespring = greenestspring.clip(swiss);
var greenestCompositesummer = greenestsummer.clip(swiss);
var greenestCompositeautumn = greenestautumn.clip(swiss);
var whitestCompositewinter  = whitestwinter.clip(swiss);
var bluestComposite         = bluestrec.clip(swiss);

// give out some information about the created variables/images to the Console (see it on the right side).
print(greenestCompositespring, 'green');
print(bluestComposite, 'blue');

var ndvispring        = greenestCompositespring.select('ndvi');
var ndvisummer        = greenestCompositesummer.select('ndvi');
var ndviautumn        = greenestCompositeautumn.select('ndvi');
var ndviwintergreen   = greenestCompositewinter.select('ndvi');
var ndviwinterwhite   = whitestCompositewinter.select('ndvi');

// 6th ------------------------------------------------------------------------------------------------------------------------------
// Composite out of greenest and bluest. bluest over water, greenest over land
var springmosaic      = greenestCompositespring.where(ndvispring.lt(0.4), bluestComposite);
var summermosaic      = greenestCompositesummer.where(ndvisummer.lt(0.4), bluestComposite);
var autumnmosaic      = greenestCompositeautumn.where(ndviautumn.lt(0.5), bluestComposite);
var greenwintermosaic = greenestCompositewinter.where(watermask.eq(1), bluestComposite);
var whitewintermosaic = whitestCompositewinter.where(watermask.eq(1), bluestComposite);



// 7th: do more stuff, e.g. calculate other indices like EVI, or pansharpen ----------------------------------------------
```

```
var evispring = springmosaic.expression(
   '2.5 * (nir - red) / (nir + 6 * red - 7.5 * blue + 1)',
   {
      red:  springmosaic.select('B4'),    // 620-670nm, RED
      nir:  springmosaic.select('B5'),    // 841-876nm, NIR
      blue: springmosaic.select('B2')    // 459-479nm, BLUE
   });


// Pan Sharpening, based on panchromatic band, which has lower spectral but higher spatial resolution.
// Panchromatic bands receive the whole energy of a wider spectral window (usually blue until red light).
// As the incoming energy in a panchromatic band is higher, more effort can be done to increase the spatial
resolution.
// For pan-sharpening the RGB-bands are transformed into HSV (Hue, Saturation, Value).
// As the value (blackness) band resembles well to the panchromatic band, they can be swapped.
// Then the combination of Hue, Saturation and Panchromatic are retransformed into RGB, resulting in a
higher spatial and spectral resolution.
// Do this for nice visualization, better not further calculate on it ;-)
// This function will apply pansharpening to an image (img).
var pansharp  = function (img)  {
   var rgb          = img.select(['B2', 'B3', 'B4']).float();
   var gray         = img.select('B8').float();
// Convert to HSV, swap in the pan band, and convert back to RGB.
   var huesat       = rgb.rgbToHsv().select('hue', 'saturation');
   return  ee.Image.cat(huesat, gray).hsvToRgb();
};

// Here you apply pansharpening to the mosaics.
var panspring = pansharp(springmosaic);
var pansummer = pansharp(summermosaic);
var panautumn = pansharp(autumnmosaic);
var pangreenwinter = pansharp(greenwintermosaic);
var panwhitewinter = pansharp(whitewintermosaic);
print (panspring);

// There usually appear some NA/masked pixels. insert the unsharpened pixels where this is the case.
var panspr    = springmosaic.select(['B2', 'B3', 'B4']).rename(['blue', 'green',
'red']).where(panspring.select('red').gte(0), panspring);
var pansum    = summermosaic.select(['B2', 'B3', 'B4']).rename(['blue', 'green',
'red']).where(pansummer.select('red').gte(0), pansummer);
var panaut    = autumnmosaic.select(['B2', 'B3', 'B4']).rename(['blue', 'green',
'red']).where(panautumn.select('red').gte(0), panautumn);
var pangwin   = greenwintermosaic.select(['B2', 'B3', 'B4']).rename(['blue', 'green',
'red']).where(pangreenwinter.select('red').gte(0), pangreenwinter);
var panwwin   = whitewintermosaic.select(['B2', 'B3', 'B4']).rename(['blue', 'green',
'red']).where(panwhitewinter.select('red').gte(0), panwhitewinter);



// Use Spectral Unmixing as a simple preparation of Land Cover Classification.
// Define spectral endmembers by assigning bands values.
```

```
// Feel free to add other bands, such as snow or rock for example, by exploring the pan-sharpened image's values.
// But if you do so, add the bands in the variable fractions as well (var fractions).
var urbanbare = [0.09, 0.07, 0.07];
var veg       = [0.075, 0.09, 0.04];
var water     = [0.057, 0.02, 0.015];

// Unmix the image.
// Use the "Inspector" to the right side, click on the image and inspect the bands.
// band_0: value resembles to probability of urban/bare land cover.
// band_1: value resembles to probability of vegetated land cover.
// band_2: value resembles to probability of water land cover.
// pan-sharpened image is only used for illustration issues.
// for more reliable results, better take the unsharpened Landsat 8 image. But then adjust the values above!
var fractions = pansum.unmix([urbanbare, veg, water]);
Map.addLayer(fractions, {}, 'unmixed');

// Some ideas for visualization.
Map.setCenter(8.5, 46.8, 8); // Switzerland centered.
var vizParams  = {bands: ['B6', 'B4', 'B3'], min: 0, max: 0.4, gamma: 0.9};
var vizParams2 = {bands: ['B4', 'B3', 'B2'], min: 0, max: 0.24, gamma: 0.9};
var vizParamsPan = {bands: ['red', 'green', 'blue'], min: 0, max: 0.24, gamma: 1.1};
var ndviViz = {bands: ['ndvi'], min: -0.5, max: 1, palette: ['505050', '505050', 'E8E8E8', '00FF33', '003300']};
var ndsiViz = {bands: ['ndsi'], min: -1, max: 1, palette: ['505050', 'E8E8E8', '00FF33', '003300']};
var ndwiViz = {bands: ['ndwi'], min: -1, max: 1, palette: ['505050', 'E8E8E8', '00FF33', '003300']};

// Display the results.
Map.addLayer (springmosaic, vizParams2, 'True-Color Spring');
Map.addLayer (springmosaic, ndsiViz, 'NDSI spring');
Map.addLayer (whitewintermosaic, vizParams, 'CIR white winter');
Map.addLayer (evispring, {min: -0.5, max: 1, palette: ['505050', '505050', 'E8E8E8', '00FF33', '003300']},
'EVI');
Map.addLayer (panspr, vizParamsPan, 'True-Color Pan Spring');
Map.addLayer (pangwin, vizParamsPan, 'True-Color Pan Wintergreen');
Map.addLayer (summermosaic, vizParams2, 'True-Color Summer ');
Map.addLayer (pansum, vizParamsPan, 'True-Color Pan Summer');

// Export
Export.image (pansum, 'Pansharpened_Summer', {region: swiss, maxPixels: 1206250905});
```