Guideline

High-Resolution Soil Moisture Retrieval Using C-Band Radar Sentinel-1 Data

Introduction

This tutorial consists of blocks of code for step by step building Soil moisture retrieval using c-Band Radar Sentinel 1 with the support of the cosmic-ray neutron sensor CRNS. We will use Google Earth Engine (GEE) which is a free, cloud-based analysis of satellite data analysis with a catalogue of satellite imagery and geospatial datasets. All satellite data used in this study were processed in this cloud computing platform. A JavaScript application program interface (API) was used to analyze satellite data. However, Python and R versions can also be used. The first step is to either load the shapefiles of the study area or manually enter the coordinates of the CRNS installation site. We will use GPS coordinates to define the study area in this example. In summary, the steps are the following:

1. localization of the CRNS GPS coordinate
2. build a buffer around the localization of the CRNS (make sure not to select buildings or roads)
3. define your temporal window, it is recommended to select all the available images with a single Mode, Ascending or Descending
4. map over the collection and apply a boxcar filter and select de VV or VH radar backscattered signal and calculate the Radar vegetation Index
5. Export the result as a CSV file and build the regression equation
6. put in the map of the collection the parameters estimated with the calibration equation

Steps 1 to 5

```
/*
```

```
  spatio-temporal soil moisture estimation based on sentinel-1
and in-situ cosmic-ray neutron sensor CRNS
  In this example, The CRNS installed in Petzenkirchen, Vienna
Austria, will be used.
- (Step-1) Preprocessing;
    Map Visualization
    Cosmic-Ray Neutron Sensor GPS (Lon, Lat)
    Buffer 500m radius size around the CRNS
    define Date and later date range
    Center the to the localization of the CRNS
    Visualisation of the map
*/

Map.setOptions('SATELLITE')
var crnsPosition=ee.Geometry.Point([15.14841,48.15467]);
var region=crnsPosition.buffer(500)
var now=ee.Date(Date.now())
var boxcar = ee.Kernel.circle(10);
Map.centerObject(crnsPosition,15)
Map.addLayer(region,{},'Foot Print')

/*
  Function to calculate radar vegetation indices,  in this
example 4 indices were estimated and rename bandes.
*/

var radarVegetationIndex=function(image){
  var sigma1=image.expression('vv/vh',
{'vv':image.select('VV'),'vh':image.select('VH')}).rename('Sigma1
')
  var sigma2=image.expression('(vv-vh)/(vv +vh)',
{'vv':image.select('VV'),'vh':image.select('VH')}).rename('Sigma2
')
  var sigma3=image.expression('(vv-vh)/vv',
{'vv':image.select('VV'),'vh':image.select('VH')}).rename('Sigma3
')
  var sigma4=image.expression('(vv-vh)/vh',
{'vv':image.select('VV'),'vh':image.select('VH')}).rename('Sigma4
')
  return
image.addBands([sigma1,sigma2,sigma3,sigma4]).copyProperties(imag
e,['system:time_start'])
}
```

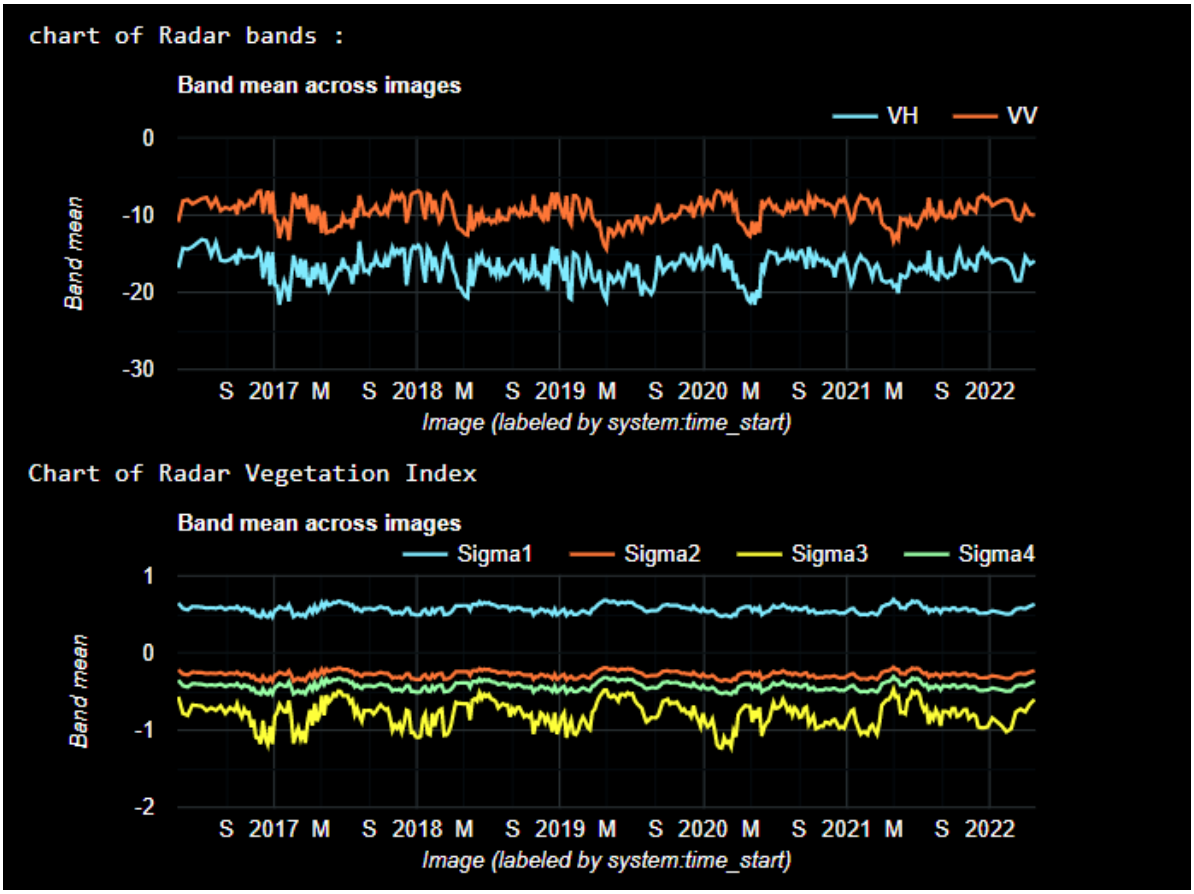
```
/*
Application of sentinel1 preprocessing technique
application of boxcar filtering
application of the radar vegetation index
*/

var collection = ee.ImageCollection("COPERNICUS/S1_GRD")
                  .filterBounds(region)

.filter(ee.Filter.listContains('transmitterReceiverPolarisation',
'VV'))

.filter(ee.Filter.listContains('transmitterReceiverPolarisation',
'VH'))
                  .filter(ee.Filter.eq('orbitProperties_pass',
'ASCENDING'))
                  .filterDate(now.advance(-6,'year'),now)
                  .map(function(image){return
image.clip(region).select(['VV','VH']).convolve(boxcar)})
                  .map(radarVegetationIndex)

var bands=['Sigma1','Sigma2','Sigma3','Sigma4']
var chart_VVVH =
ui.Chart.image.series(collection.select(['VV','VH']), region,
ee.Reducer.mean(), 10)
var chart_RVI = ui.Chart.image.series(collection.select(bands),
region, ee.Reducer.mean(), 10)
print(
      'chart of Radar bands : ',chart_VVVH,
      'Chart of Radar Vegetation Index', chart_RVI)
```



Step 6

Application of model to convert Radar signal into Soil moisture

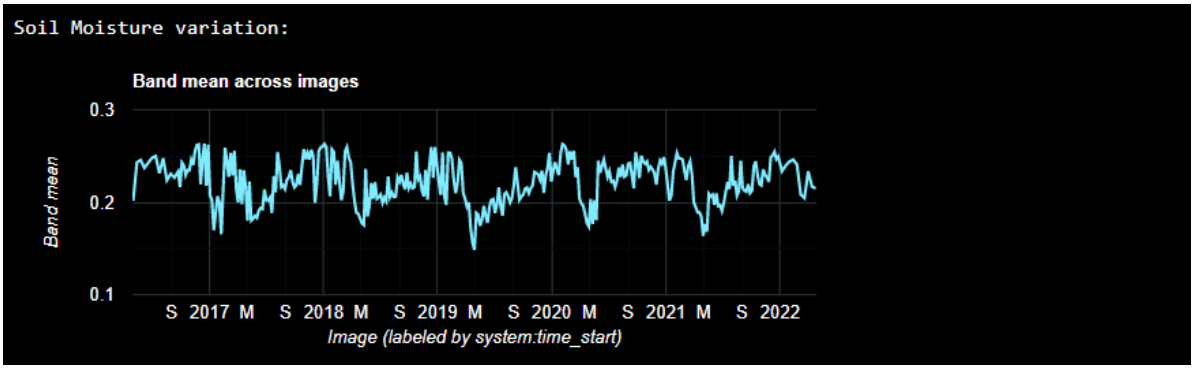
lm(formula = y ~ x)

Coefficients:

(Intercept) x

0.36759 0.01524

```
var soilmoisture = collection
                    .select(['VV'])
                    .map(function(image){
                      return image.multiply(0.01524)
                                  .add(0.36759)
                                  .select('VV')
                                  .rename('SSM')
                                  .copyProperties(image,
['system:time_start'])
                    })
var chart_SM= ui.Chart.image.series(soilmoisture.select('SSM'),
region, ee.Reducer.mean(), 10)
print('Soil Moisture variation: ', chart_SM)
```



Visualization of the map

```
var vis = {min: 0.1, max: 0.3, palette: 'red,orange,green, blue',
opacity:0.75};
Map.addLayer(soilmoisture.mean(),vis ,'Soil Moisture',true)

/**********************************************************************
*****************************************
 * Legend

**********************************************************************
*****************************************/

function makeColorBarParams(palette) {
  return {
    bbox: [0, 0, 1, 0.1],
    dimensions: '100x40',
    format: 'png',
    min: 0,
    max: 1,
    palette: palette,
  };
}

// Create the color bar for the legend.
var colorBar = ui.Thumbnail({
  image: ee.Image.pixelLonLat().select(0),
  params: makeColorBarParams(vis.palette),
  style: {stretch: 'horizontal', margin: '0px 8px', maxHeight:
'30px'},
});

// Create a panel with three numbers for the legend.
```

```
var legendLabels = ui.Panel({
  widgets: [
    ui.Label(vis.min, {margin: '4px 8px'}),
    ui.Label(
        ((vis.max-vis.min) / 2+vis.min),
        {margin: '8px 8px', textAlign: 'center', stretch:
'horizontal'}),
    ui.Label(vis.max, {margin: '4px 8px'})
  ],
  layout: ui.Panel.Layout.flow('horizontal')
});

var legendTitle = ui.Label({
  value: 'top Surface Soil Moisture cm3/cm3',
  style: {fontWeight: 'bold'}
});

var legendPanel = ui.Panel([legendTitle, colorBar,
legendLabels]);
Map.add(legendPanel);
```