**var** segmentation **=** **ee.FeatureCollection**("users/esposito1976938/spatial\_join"),

classi **=** **ee.FeatureCollection**("users/esposito1976938/iCOLT2023\_vec\_HAgt05\_clean"),

flood **=** **ee.FeatureCollection**("users/esposito1976938/flood\_area\_emiliaromagna"),

s2Sr **=** **ee.ImageCollection**("COPERNICUS/S2\_SR\_HARMONIZED"),

s2Clouds **=** **ee.ImageCollection**("COPERNICUS/S2\_CLOUD\_PROBABILITY"),

geometry **=**

/\* color: #98ff00 \*/

/\* shown: false \*/

**ee.Geometry.Polygon**(

[[[11.29679239988353, 44.47742383573328],

[11.61539591550853, 44.099936971230925],

[12.46408976316478, 44.135429419672484],

[12.48880900144603, 44.204381482428616],

[12.33225382566478, 44.57239474544317],

[11.616769206524156, 44.65158171960682]]]),

CHIRPS **=** **ee.ImageCollection**("UCSB-CHG/CHIRPS/PENTAD");

var selectedFeatures = segmentation.filter(ee.Filter.eq('DN', 163204));

print('selectedFeatures (campi da considerare)', selectedFeatures)

print('selectedFeatures size', selectedFeatures.size())

var part1 = selectedFeatures; //in this case the geometry can be drawn in the GEE window

print(part1);

var startDate = '2022-08-01';

var endDate = '2023-10-31';

startDate = ee.Date(startDate);

endDate = ee.Date(endDate);

print(startDate);

print(endDate);

print('startDate:', startDate.format('yyyy-MM-dd').getInfo() + " " + endDate.format('yyyy-MM-dd').getInfo());

var numero\_orbita = 117

var sentinel1 = ee.ImageCollection('COPERNICUS/S1\_GRD')

.filterDate(startDate,endDate)

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

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

.filter(ee.Filter.eq('instrumentMode', 'IW'))

.filter(ee.Filter.eq('relativeOrbitNumber\_start',numero\_orbita)) //asc

//.filter(ee.Filter.eq('relativeOrbitNumber\_start', 146)) //asc

//.filter(ee.Filter.eq('relativeOrbitNumber\_start', 124)) //desc

.filterBounds(part1)

;

print (sentinel1,'Sentinel1');

// Image count function for SAR

function calc\_RVI(img) {

var bandsAscVV = img.select('VV');

var bandsAscVH = img.select('VH');

var RVI = (bandsAscVH.multiply(4)).divide(bandsAscVV.add(bandsAscVH)).rename('RVI')

.copyProperties(img).set('system:time\_start', (img.get('system:time\_start')));

return RVI

}

var RVI = sentinel1.map(calc\_RVI);

print (RVI, 'rvi')

var dependent = 'RVI';

// The number of cycles per year to model.

var harmonics = 3;

// Make a list of harmonic frequencies to model.

// These also serve as band name suffixes.

var harmonicFrequencies = ee.List.sequence(1, harmonics);

// Function to get a sequence of band names for harmonic terms.

var getNames = function(base, list) {

return ee.List(list).map(function(i) {

return ee.String(base).cat(ee.Number(i).int());

});

};

// Construct lists of names for the harmonic terms.

var cosNames = getNames('cos\_', harmonicFrequencies);

var sinNames = getNames('sin\_', harmonicFrequencies);

// Independent variables.

var independents = ee.List(['constant', 't']).cat(cosNames).cat(sinNames);

// Function to add a constant band.

var addConstant = function(image) {

return image.addBands(ee.Image(1));

};

// Function to add a time band.

var addTime = function(image) {

// Compute time in fractional years since the epoch.

var date = image.date();

var years = date.difference(ee.Date('1970-01-01'), 'year');

var timeRadians = ee.Image(years.multiply(2 \* Math.PI));

return image.addBands(timeRadians.rename('t').float());

};

var addHarmonics = function(freqs) {

return function(image) {

// Make an image of frequencies.

var frequencies = ee.Image.constant(freqs);

// This band should represent time in radians.

var time = ee.Image(image).select('t');

// Get the cosine terms.

var cosines = time.multiply(frequencies).cos().rename(cosNames);

// Get the sin terms.

var sines = time.multiply(frequencies).sin().rename(sinNames);

return image.addBands(cosines).addBands(sines);

};

};

var harmonicLandsat = RVI

.map(addConstant)

.map(addTime)

.map(addHarmonics(harmonicFrequencies));

print ('harmoniclandasat', harmonicLandsat)

// The output of the regression reduction is a 4x1 array image.

var harmonicTrend = harmonicLandsat

.select(independents.add(dependent))

.reduce(ee.Reducer.linearRegression(independents.length(), 1));

var harmonicTrendCoefficients = harmonicTrend.select('coefficients')

.arrayProject([0])

.arrayFlatten([independents]);

var fittedHarmonic = harmonicLandsat.map(function(image) {

return image.addBands(

image.select(independents)

.multiply(harmonicTrendCoefficients)

.reduce('sum')

.rename('fitted\_RVI'));

});

print(ui.Chart.image.series(

fittedHarmonic.select(['fitted\_RVI', 'RVI']), part1, ee.Reducer.median(), 10, 'system:time\_start')

.setOptions({

title: 'Ascending RVI orbit'+numero\_orbita+ ' - vigneti in flood area',

lineWidth: 1,

pointSize: 3,

curveType: 'function'

})

);

var period = 'days';

var perido\_lenght = 10;

// Create list of dates for time series

var n\_periods = endDate.difference(startDate,period).round();

var dates\_indexes = ee.List.sequence(0,ee.Number(n\_periods).subtract(1),perido\_lenght);

print('numero\_di\_periods (days se usiamo days)',n\_periods);

print('dates\_indexes', dates\_indexes);

var make\_datelist = function(n) {

return startDate.advance(n,period);

};

var dates = dates\_indexes.map(make\_datelist);

print('Start dates for the aggregation iteration', dates);

var MAX\_CLOUD\_PROBABILITY = 50;

function maskClouds(img) {

var clouds = ee.Image(img.get('cloud\_mask')).select('probability');

var isNotCloud = clouds.lt(MAX\_CLOUD\_PROBABILITY);

return img.updateMask(isNotCloud).copyProperties(img, ['system:time\_start','system:index']);

}

// The masks for the 10m bands sometimes do not exclude bad data at

// scene edges, so we apply masks from the 20m and 60m bands as well.

// Example asset that needs this operation:

// COPERNICUS/S2\_CLOUD\_PROBABILITY/20190301T000239\_20190301T000238\_T55GDP

function maskEdges(s2\_img) {

return s2\_img.updateMask(

s2\_img.select('B8A').mask().updateMask(s2\_img.select('B9').mask())).copyProperties(s2\_img, ['system:time\_start','system:index']);

}

//SPACECRAFT\_NAME: Sentinel-2B

// Filter input collections by desired data range and region.

// .filter(ee.Filter.eq('SPACECRAFT\_NAME', 'Sentinel-2B'))

//.filter(ee.Filter.eq('MGRS\_TILE', '32TQQ'))

var criteria = ee.Filter.and(

ee.Filter.bounds(geometry), ee.Filter.date(startDate, endDate));

s2Sr = s2Sr.filter(criteria).map(maskEdges);

s2Clouds = s2Clouds.filter(criteria);

print(s2Clouds)

// Join S2 SR with cloud probability dataset to add cloud mask.

var s2SrWithCloudMask = ee.Join.saveFirst('cloud\_mask').apply({

primary: s2Sr,

secondary: s2Clouds,

condition:

ee.Filter.equals({leftField: 'system:index', rightField: 'system:index'})

});

var s2CloudMasked\_collection =

ee.ImageCollection(s2SrWithCloudMask).map(maskClouds);

print(s2CloudMasked\_collection);

//Map.addLayer(s2CloudMasked\_collection);

function apply\_radiometric\_scaling\_factor(image)

{

return image.multiply(0.0001).copyProperties(image, ['system:time\_start','system:index']);

}

var filtered\_and\_radiometrically\_scaled\_S2 = s2CloudMasked\_collection.map(apply\_radiometric\_scaling\_factor);

print('filtered\_and\_radiometrically\_scaled\_S2 size', filtered\_and\_radiometrically\_scaled\_S2.size());

function add\_NDVI(img)

{

var ndvi = img.normalizedDifference(['B8', 'B4']).rename('NDVI');

img = img.addBands(ndvi);

return img.copyProperties(img, ['system:time\_start','system:index']);

}

var s2CloudMasked\_collection\_with\_NDVI = filtered\_and\_radiometrically\_scaled\_S2.map(add\_NDVI)

var funzione\_di\_aggregazione\_S2 = function(d1) {

var start = ee.Date(d1);

var end = ee.Date(d1).advance(perido\_lenght,period);

var delta\_giorni = ee.Number(end.difference(start, 'days'));

var meta\_delta\_giorni = delta\_giorni.divide(2);

var data\_centrale = start.advance(meta\_delta\_giorni, 'days');

var date\_range = ee.DateRange(start,end);

// dobbiamo usare la versione harmonized, altrimenti

// ci sono alcune immagini con 23 bande e altre con 21

// e questo crea problemi al reducer

var S2\_sub\_period =s2CloudMasked\_collection\_with\_NDVI.filterDate(date\_range);

var number\_of\_images\_aggregated = S2\_sub\_period.size();

// con set imposto 4 nuove poprietà

// la prima è start\_date che conterrà la

// data di start di ciascun sotto periodo

// per ogni immagine aggregata

// la seconda è end\_date che conterrà la

// data di end di ciascun sotto periodo

// per ogni immagine aggregata

// la terza è n\_images

// che conterrà il numero di immagini che sono

// state aggregate nel sotto-periodo considerato

return(S2\_sub\_period.max().set('start\_date', start).set('end\_date', end).set('system:time\_start', data\_centrale.millis()).set('n\_images', number\_of\_images\_aggregated).clip(geometry).copyProperties(S2\_sub\_period, ['system:index'])); //.clip(geometry

};

var list\_of\_images\_S2\_aggregated = dates.map(funzione\_di\_aggregazione\_S2);

print('list\_of\_images\_S2\_aggregated', list\_of\_images\_S2\_aggregated);

// Attenzione che l'aggregazione fa perdere la scala!!!!!

var S2\_filtered\_aggregated = ee.ImageCollection(list\_of\_images\_S2\_aggregated);

print('S2\_filtered\_aggregated', S2\_filtered\_aggregated);

Map.centerObject(geometry, 10);

// Filtro la collezione aggregata per escludere i sub-periodi per cui non ho le immagini

S2\_filtered\_aggregated = S2\_filtered\_aggregated.filter(ee.Filter.gt('n\_images',0));

print('S2\_filtered\_aggregated post rimozione', S2\_filtered\_aggregated);

var count = S2\_filtered\_aggregated.size();

// casting the image collection in a list of ee.Images

var S2\_L2A\_imagesList = S2\_filtered\_aggregated.toList(count);

// The dependent variable we are modeling.

var dependent\_1 = 'NDVI';

var harmonic\_Sentinel\_2 = S2\_filtered\_aggregated

.map(addConstant)

.map(addTime)

.map(addHarmonics(harmonicFrequencies));

print ('harmonicsentinel2', harmonic\_Sentinel\_2);

// The output of the regression reduction is a 4x1 array image.

var harmonicTrend\_Sentinel\_2 = harmonic\_Sentinel\_2

.select(independents.add(dependent\_1))

.reduce(ee.Reducer.linearRegression(independents.length(), 1));

// Turn the array image into a multi-band image of coefficients.

var harmonicTrendCoefficients\_Sentinel\_2 = harmonicTrend\_Sentinel\_2.select('coefficients')

.arrayProject([0])

.arrayFlatten([independents]);

// Compute fitted values.

var fittedHarmonic\_Sentinel\_2 = harmonic\_Sentinel\_2.map(function(image) {

return image.addBands(

image.select(independents)

.multiply(harmonicTrendCoefficients\_Sentinel\_2)

.reduce('sum')

.rename('fitted\_NDVI'));

});

// Plot the fitted model and the original data at the ROI.

print(ui.Chart.image.series(

//fittedHarmonic.select(['fitted', 'NDVI']), roi, ee.Reducer.mean(), 30)

fittedHarmonic\_Sentinel\_2.select(['fitted\_NDVI', 'NDVI']), part1, ee.Reducer.median(), 10, 'system:time\_start')

.setOptions({

title: 'NDVI - vigneti in flood area',

lineWidth: 1,

pointSize: 3

})

);

// Display the ROI and NDVI composite on the map.

Map.centerObject(geometry, 11);

//Map.addLayer(harmonicLandsat,

// {bands: 'NDVI', min: 0.1, max: 0.9, palette: ['white', 'green']},

// 'NDVI Mosaic');

//Map.addLayer(roi, {color: 'yellow'}, 'ROI');

var IC\_ALL = fittedHarmonic.select('RVI', 'fitted\_RVI').merge(fittedHarmonic\_Sentinel\_2.select('NDVI', 'fitted\_NDVI'));

print ('ic\_all', IC\_ALL)

var chart\_RVI\_NDVI =

ui.Chart.image.series({

imageCollection: IC\_ALL,

region: part1,

reducer: ee.Reducer.median(),

scale: 10,

xProperty: 'system:time\_start',

})

.setOptions({

title: 'Ascending RVI orbit' +numero\_orbita+ '- NDVI vigneti in flood area',

hAxis: {title: 'Date', titleTextStyle: {italic: false, bold: true}},

vAxis: {

title: 'RVI-NDVI',

titleTextStyle: {italic: false, bold: true}

},

lineWidth: 3,

colors: ['1d6b99', '1d6b99','red' ,'red'],

curveType: ['function', 'function'],

interpolateNulls: true // mettere a false per vedere la serie bucata

});

print (chart\_RVI\_NDVI);

Map.addLayer(flood, {}, 'flood')

Map.addLayer(classi, {}, 'classi')

Map.addLayer(segmentation, {}, 'segmentation')

Map.addLayer(selectedFeatures, {}, 'selectedFeatures')

var selectedFeatures\_1 = segmentation.filter(ee.Filter.eq('DN', 128623));

var part2 = selectedFeatures\_1

print(ui.Chart.image.series(

fittedHarmonic.select(['fitted\_RVI', 'RVI']), part2, ee.Reducer.median(), 10, 'system:time\_start')

.setOptions({

title: 'Ascending RVI orbit'+numero\_orbita+ ' - vigneti',

lineWidth: 1,

pointSize: 3,

curveType: 'function'

})

);

print(ui.Chart.image.series(

//fittedHarmonic.select(['fitted', 'NDVI']), roi, ee.Reducer.mean(), 30)

fittedHarmonic\_Sentinel\_2.select(['fitted\_NDVI', 'NDVI']), part2, ee.Reducer.median(), 10, 'system:time\_start')

.setOptions({

title: 'NDVI - vigneti',

lineWidth: 1,

pointSize: 3

})

);

var chart\_RVI\_NDVI\_1 =

ui.Chart.image.series({

imageCollection: IC\_ALL,

region: part2,

reducer: ee.Reducer.median(),

scale: 10,

xProperty: 'system:time\_start',

})

.setOptions({

title: 'Ascending RVI orbit' +numero\_orbita+ '- NDVI colture vigneti ',

hAxis: {title: 'Date', titleTextStyle: {italic: false, bold: true}},

vAxis: {

title: 'RVI-NDVI',

titleTextStyle: {italic: false, bold: true}

},

lineWidth: 3,

colors: ['1d6b99', '1d6b99','red' ,'red'],

curveType: ['function', 'function'],

interpolateNulls: true // mettere a false per vedere la serie bucata

});

print (chart\_RVI\_NDVI\_1)

Map.addLayer(selectedFeatures\_1, {}, 'selectedFeatures\_1')

var filtered\_CHIRPS = CHIRPS.filter(ee.Filter.date(startDate, endDate));

print(filtered\_CHIRPS)

var CHIRPS\_GSD = filtered\_CHIRPS.first().projection().nominalScale()

function add\_median\_on\_ROI\_as\_property(image)

{

// ispirazione: https://developers.google.com/earth-engine/guides/debugging

var median = image.reduceRegion(

{reducer: ee.Reducer.median(), geometry: geometry, scale: CHIRPS\_GSD}).get('precipitation');

return image.set('median', median);

}

var filtered\_CHIRPS\_with\_median = filtered\_CHIRPS.select(['precipitation']).map(add\_median\_on\_ROI\_as\_property)

print('filtered\_CHIRPS\_with\_median',filtered\_CHIRPS\_with\_median)

var medianList\_y = ee.Array(filtered\_CHIRPS\_with\_median.aggregate\_array('median'));

var time\_start\_x = ee.Array(filtered\_CHIRPS\_with\_median.aggregate\_array('system:time\_start'));

print('medianList\_y',medianList\_y)

function normalize\_array (array)

{

var min\_y = medianList\_y.reduce(ee.Reducer.min(), [0]).get([0])

var max\_y = medianList\_y.reduce(ee.Reducer.max(), [0]).get([0])

var count\_y = medianList\_y.reduce(ee.Reducer.count(), [0]).get([0])

//x normalized = (x – x minimum) / (x maximum – x minimum)

var normalized\_array = medianList\_y.subtract(min\_y).divide(max\_y.subtract(min\_y))

return normalized\_array

}

var min\_y = medianList\_y.reduce(ee.Reducer.min(), [0]).get([0])

var max\_y = medianList\_y.reduce(ee.Reducer.max(), [0]).get([0])

print('min\_y',min\_y)

print('max\_y', max\_y)

var medianList\_y\_normalized = normalize\_array(medianList\_y\_normalized)

//print('min\_y', ee.Array(ee.List.repeat(min\_y, count\_y)))

print('medianList\_y\_normalized', medianList\_y\_normalized)

// Define the chart and print it to the console.

var chart\_original = ui.Chart.array.values({array: medianList\_y, axis: 0, xLabels: time\_start\_x}).setOptions({

title: 'Precipitation [mm/pentad]',

colors: ['blue'],

hAxis: {

title: 'Timestart',

titleTextStyle: {italic: false, bold: true}

},

vAxis: {

title: 'Precipitation [mm/pentad]',

titleTextStyle: {italic: false, bold: true}

},

pointSize: 4,

dataOpacity: 0.4,

lineWidth: 1,

legend: {position: 'none'},

});

print(chart\_original);

// Define the chart and print it to the console.

var chart\_normalized = ui.Chart.array.values({array: medianList\_y\_normalized, axis: 0, xLabels: time\_start\_x}).setOptions({

title: 'Precipitation normalized between 0 and 1',

colors: ['blue'],

hAxis: {

title: 'Timestart',

titleTextStyle: {italic: false, bold: true}

},

vAxis: {

title: 'Normalized precipitation',

titleTextStyle: {italic: false, bold: true}

},

pointSize: 4,

dataOpacity: 0.4,

lineWidth: 1,

legend: {position: 'none'},

});

print(chart\_normalized);

function add\_median\_normalize\_on\_ROI\_as\_property(image)

{

// ispirazione: https://developers.google.com/earth-engine/guides/debugging

var median = image.reduceRegion(

{reducer: ee.Reducer.median(), geometry: geometry, scale: CHIRPS\_GSD}).get('precipitation');

// medianList\_y.subtract(min\_y).divide(max\_y.subtract(min\_y))

var normalized\_median = ee.Number(image.get('median')).subtract(min\_y).divide(max\_y.subtract(min\_y))

// aggiugno 1 per non avere troppi grafici centrati sull'asse delle x

var normalized\_median\_band = ee.Image(normalized\_median).add(1).rename(['normalized\_median\_precipitation\_on\_ROI']);

image = image.addBands(normalized\_median\_band)

return image.set('median\_normalized',normalized\_median);

}

var filtered\_CHIRPS\_with\_normalized\_median = filtered\_CHIRPS\_with\_median.select(['precipitation']).map(add\_median\_normalize\_on\_ROI\_as\_property)

print('filtered\_CHIRPS\_with\_normalized\_median',filtered\_CHIRPS\_with\_normalized\_median)

var IC\_ALL\_bis = fittedHarmonic.select('RVI', 'fitted\_RVI').merge(fittedHarmonic\_Sentinel\_2.select('NDVI', 'fitted\_NDVI')).merge(filtered\_CHIRPS\_with\_normalized\_median.select('normalized\_median\_precipitation\_on\_ROI'));

var chart\_RVI\_NDVI\_precipitation =

ui.Chart.image.series({

imageCollection: IC\_ALL\_bis,

region: part1,

reducer: ee.Reducer.median(),

scale: 10,

xProperty: 'system:time\_start',

})

.setOptions({

title: 'Ascending RVI orbit' +numero\_orbita+ '- NDVI timeseries vigneti with precipitation in flood area',

hAxis: {title: 'Date', titleTextStyle: {italic: false, bold: true}},

vAxis: {

title: 'RVI-NDVI-normalized precipitation in flood area',

titleTextStyle: {italic: false, bold: true}

},

lineWidth: 1,

colors: ['green', 'black','red' ,'red', 'blue'],

//curveType: ['function', 'function'],

type: "line",

pointSize: 4,

dataOpacity: 0.4,

interpolateNulls: true // mettere a false per vedere la serie bucata

});

print (chart\_RVI\_NDVI\_precipitation);

var chart\_RVI\_NDVI\_precipitation\_1 =

ui.Chart.image.series({

imageCollection: IC\_ALL\_bis,

region: part2,

reducer: ee.Reducer.median(),

scale: 10,

xProperty: 'system:time\_start',

})

.setOptions({

title: 'Ascending RVI orbit' +numero\_orbita+ '- NDVI timeseries vigneti with precipitation',

hAxis: {title: 'Date', titleTextStyle: {italic: false, bold: true}},

vAxis: {

title: 'RVI-NDVI-normalized precipitation',

titleTextStyle: {italic: false, bold: true}

},

lineWidth: 1,

colors: ['green', 'black','red' ,'red', 'blue'],

//curveType: ['function', 'function'],

type: "line",

pointSize: 4,

dataOpacity: 0.4,

interpolateNulls: true // mettere a false per vedere la serie bucata

});

print (chart\_RVI\_NDVI\_precipitation\_1)