// The goal of this script is to create estimates of Land Surface Temperature (LST)

// using Landsat 8 data acquired during the summer of 2023

// Clip to study area

var clipToCol = function(image){

return image.clipToCollection(wor);

};

//cloud mask

function maskL8sr(col) {

// Bits 3 and 5 are cloud shadow and cloud, respectively.

var cloudShadowBitMask = (1 << 3);

var cloudsBitMask = (1 << 5);

// Get the pixel QA band.

var qa = col.select('QA\_PIXEL');

// Both flags should be set to zero, indicating clear conditions.

var mask = qa.bitwiseAnd(cloudShadowBitMask).eq(0)

.and(qa.bitwiseAnd(cloudsBitMask).eq(0));

return col.updateMask(mask);

}

//vis params

var vizParams = {

bands: ['SR\_B5', 'SR\_B6', 'SR\_B4'],

min: 0,

max: 0.4,

gamma: [1, 0.9, 1.1]

};

var vizParams2 = {

bands: ['SR\_B4', 'SR\_B3', 'SR\_B2'],

min: 0,

max: 0.3,

gamma: 1.4,

};

//load the collection:

{

var col = ee.ImageCollection('LANDSAT/LC08/C02/T1\_L2')

.map(maskL8sr)

.filterDate('2023-06-01','2023-9-30')

.filterBounds(geometry)

.map(clipToCol);

}

// Applies scaling factors.

function applyScaleFactors(image) {

var opticalBands = image.select('SR\_B.').multiply(0.0000275).add(-0.2);

var thermalBands = image.select('ST\_B.\*').multiply(0.00341802).add(149.0);

return image.addBands(opticalBands, null, true)

.addBands(thermalBands, null, true);

}

col = col.map(applyScaleFactors);

print(col, 'Summer colection');

//Center the map screen on Worcester

Map.centerObject(wor, 11);

// find image with the least cloud coverage

var leastcloud = col.sort('CLOUD\_COVER').first();

Map.addLayer(leastcloud, vizParams2, 'least cloud RGB');

image = leastcloud // if leastcloud does not have cloud cover, use this as the input

//Center the map screen on Worcester

Map.centerObject(wor, 11);

//image reduction - If Image Collection is over a long range of dates, this will find the median

// value of each pixel between those dates. In this case, our collection

// is the summer of 2023 so our final image will be the median of all imagery

// after it has been cloud masked.

//median

{

var image = col.median(); // if leastcloud has cloud cover, use the median as the input

print(image, 'image');

Map.addLayer(image, vizParams2, 'median');

}

// calculate NDVI

var ndvi = image.normalizedDifference(['SR\_B5',

'SR\_B4']).rename('NDVI');

var ndviParams = {min: -1, max: 1, palette: ['blue', 'white',

'green']};

print(ndvi,'ndvi');

Map.addLayer(ndvi, ndviParams, 'ndvi');

//get surface temperature, convert to degree celcius

var LST = image.select('ST\_B10').subtract(273.15).rename('LST');

Map.addLayer(LST, {min: 15.569706944223423, max:39.328077233404645, palette: [

'040274', '040281', '0502a3', '0502b8', '0502ce', '0502e6',

'0602ff', '235cb1', '307ef3', '269db1', '30c8e2', '32d3ef',

'3be285', '3ff38f', '86e26f', '3ae237', 'b5e22e', 'd6e21f',

'fff705', 'ffd611', 'ffb613', 'ff8b13', 'ff6e08', 'ff500d',

'ff0000', 'de0101', 'c21301', 'a71001', '911003'

]},'LST');

// Export the image to your Google Drive.

Export.image.toDrive({

image: LST,

description: "LST2023",

maxPixels: 1e8,

region: wor,

crs: 'EPSG:26986',

scale: 30

});

//Uncomment this part when you get to question 4. The code below

//generate linear regression plot between SAVI and LST. Modify the

//code to run regression between IBI and LST. Refer to Part 1 code

//for IBI calcualation

//Savi code

var SAVI = image.expression('1.5 \* (NIR - RED) / (NIR + RED + 0.5)', {

'NIR': image.select('SR\_B5'),

'RED': image.select('SR\_B4')

}).rename('SAVI');

//Regression between NDVI and SAVI

//Change the variables to other spectral indices to look at relationships between

//different pairs of indices

var constant = ee.Image(1);

var xvar = SAVI.select(['SAVI']);

var yvar = LST.select(['LST']);

var imgRegress = ee.Image.cat(constant,xvar, yvar).rename(['constant','xvar','yvar']);

print(imgRegress,'imgregress')

// Calculate regression coefficients for the set of pixels intersecting the

// above defined region using reduceRegion. The numX parameter is set as 2

// because the constant and the SWIR1 bands are independent variables and they

// are the first two bands in the stack; numY is set as 1 because there is only

// one dependent variable (SWIR2) and it follows as band three in the stack.

var linearRegression = imgRegress.reduceRegion({

reducer: ee.Reducer.linearRegression({

numX: 2,

numY: 1

}),

geometry: wor,

scale: 30,

});

// Convert the coefficients array to a list.

var coefList = ee.Array(linearRegression.get('coefficients')).toList();

print(coefList,'coef')

// Extract the y-intercept and slope.

var b0 = ee.List(coefList.get(0)).get(0); // y-intercept

var b1 = ee.List(coefList.get(1)).get(0); // slope

// Extract the residuals.

var residuals = ee.Array(linearRegression.get('residuals')).toList().get(0);

var prediction = imgRegress.select('xvar').multiply(ee.Image.constant(b1)).add(ee.Image.constant(b0));

//var prediction = imgRegress.expression('b1\*xvar+b0',{'xvar':imgRegress.select('xvar')})

var palettes = require('users/gena/packages:palettes');

var palette = palettes.colorbrewer.RdYlGn[9]; //https://github.com/gee-community/ee-palettes

var residuals = imgRegress.select('yvar').subtract(prediction)

Map.addLayer(residuals, {min: -5, max: 5, palette: palette}, 'Residuals'); //adjust the min and max values to get the best contrast

//Regression between LST and SAVI

var lstsavi = SAVI.select('SAVI')

.addBands(LST)

.rename(['SAVI', 'LST']);

// sample N points from the 2-band image

var values = lstsavi.sample({ region: wor, scale: 30, numPixels: 1000, geometries: true})

// // plot sampled features as a scatter chart

var chart = ui.Chart.feature.byFeature(values, 'SAVI', 'LST')

.setChartType('ScatterChart')

.setOptions({ pointSize: 2, pointColor: 'red', width: 300, height: 300, titleX: 'SAVI', titleY: 'LST', trendlines: {

0: {

type: 'linear',

showR2: true,

visibleInLegend: true

}

}

})

print(chart)

// Calculate IBI (Impervious Surface Index)

var IBI = image.expression(

'((MIR1 + RED) - (NIR + BLUE)) / ((MIR1 + RED) + (NIR + BLUE))', {

'MIR1': image.select('SR\_B6'),

'RED': image.select('SR\_B4'),

'NIR': image.select('SR\_B5'),

'BLUE': image.select('SR\_B2')

}).rename('IBI');

// Add IBI to the map for visualization

Map.addLayer(IBI, {min: -1, max: 1, palette: ['blue', 'white', 'brown']}, 'IBI');

// Regression between LST and SAVI

var lstSavi = SAVI.select('SAVI')

.addBands(LST)

.rename(['SAVI', 'LST']);

// Regression between LST and IBI

var lstIbi = IBI.select('IBI')

.addBands(LST)

.rename(['IBI', 'LST']);

// Sample 1000 random points for regression analysis

var samplePoints = 1000;

var saviValues = lstSavi.sample({region: wor, scale: 30, numPixels: samplePoints, geometries: true});

var ibiValues = lstIbi.sample({region: wor, scale: 30, numPixels: samplePoints, geometries: true});

// Plot scatter chart for LST vs. SAVI

var saviChart = ui.Chart.feature.byFeature(saviValues, 'SAVI', 'LST')

.setChartType('ScatterChart')

.setOptions({

pointSize: 2, pointColor: 'red', width: 300, height: 300, titleX: 'SAVI', titleY: 'LST',

trendlines: {0: {type: 'linear', showR2: true, visibleInLegend: true}}

});

print(saviChart, 'LST vs. SAVI Regression');

// Plot scatter chart for LST vs. IBI

var ibiChart = ui.Chart.feature.byFeature(ibiValues, 'IBI', 'LST')

.setChartType('ScatterChart')

.setOptions({

pointSize: 2, pointColor: 'blue', width: 300, height: 300, titleX: 'IBI', titleY: 'LST',

trendlines: {0: {type: 'linear', showR2: true, visibleInLegend: true}}

});

print(ibiChart, 'LST vs. IBI Regression');

Export.image.toDrive({

image: residuals,

description: "LST VS IBI",

maxPixels: 1e8,

region: wor,

crs: 'EPSG:26986',

scale: 30

});