// This code is taken from DEMO example of GEE platform

// Load a collection of Landsat TOA reflectance images.

var landsatCollection = ee.ImageCollection('LANDSAT/LC08/C01/T1\_TOA');

// Set the region of interest to a point.

var roi = ee.Geometry.Point([-121.14, 37.98]);

// The dependent variable we are modeling.

var dependent = 'NDVI';

// The number of cycles per year to model.

var harmonics = 1;

// 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 constructBandNames = 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 = constructBandNames('cos\_', harmonicFrequencies);

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

// Independent variables.

var independents = ee.List(['constant', 't'])

.cat(cosNames).cat(sinNames);

// Function to mask clouds in Landsat 8 imagery.

var maskClouds = function(image) {

var score = ee.Algorithms.Landsat.simpleCloudScore(image).select('cloud');

var mask = score.lt(10);

return image.updateMask(mask);

};

// Function to add an NDVI band, the dependent variable.

var addNDVI = function(image) {

return image

.addBands(image.normalizedDifference(['B5', 'B4'])

.rename('NDVI'))

.float();

};

// Function to add a time band.

var addDependents = function(image) {

// Compute time in fractional years since the epoch.

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

var timeRadians = ee.Image(years.multiply(2 \* Math.PI)).rename('t');

var constant = ee.Image(1);

return image.addBands(constant).addBands(timeRadians.float());

};

// Function to compute the specified number of harmonics

// and add them as bands. Assumes the time band is present.

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);

};

};

// Filter to the area of interest, mask clouds, add variables.

var harmonicLandsat = landsatCollection

.filterBounds(roi)

.map(maskClouds)

.map(addNDVI)

.map(addDependents)

.map(addHarmonics(harmonicFrequencies));

// 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));

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

var harmonicTrendCoefficients = harmonicTrend.select('coefficients')

.arrayProject([0])

.arrayFlatten([independents]);

// Compute fitted values.

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

return image.addBands(

image.select(independents)

.multiply(harmonicTrendCoefficients)

.reduce('sum')

.rename('fitted'));

});

// 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)

.setOptions({

title: 'Harmonic model: original and fitted values',

lineWidth: 1,

pointSize: 3,

}));

// Pull out the three bands we're going to visualize.

var sin = harmonicTrendCoefficients.select('sin\_1');

var cos = harmonicTrendCoefficients.select('cos\_1');

// Do some math to turn the first-order Fourier model into

// hue, saturation, and value in the range[0,1].

var magnitude = cos.hypot(sin).multiply(5);

var phase = sin.atan2(cos).unitScale(-Math.PI, Math.PI);

var val = harmonicLandsat.select('NDVI').reduce('mean');

// Turn the HSV data into an RGB image and add it to the map.

var seasonality = ee.Image.cat(phase, magnitude, val).hsvToRgb();

Map.centerObject(roi, 11);

Map.addLayer(seasonality, {}, 'Seasonality');

Map.addLayer(roi, {}, 'ROI');