//Original code link : <https://docs.google.com/document/d/1mNIRB90jwLuASO1JYas1kuOXCLbOoy1Z4NlV1qIXM10/edit>

## Data preparation and preprocessing

The first step in analysis of time series data is to import data of interest and plot it at an interesting location.

1. **Load a time series of Landsat data**

1. Search for Landsat 8 surface reflectance and import the "USGS Landsat 8 Surface Reflectance Tier 1" collection.  Name it l8sr.

1. Make a single point geometry with the geometry drawing tools and position the point in a location of interest.  (How about an annual grassland or a deciduous forest?)  Name the import roi.

1. **Filtering, masking and preparing bands of interest**

1. preprocess the Landsat imagery by filtering it to the location of interest, masking clouds, and adding the variables in the model:

// This field contains UNIX time in milliseconds.

var timeField = 'system:time\_start';

// Function to cloud mask from the pixel\_qa band of Landsat 8 SR data.

// (From the Code Editor Examples > Cloud Masking)

function maskL8sr(image) {

  // 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 = image.select('pixel\_qa');

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

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

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

  // Return the masked image, scaled to reflectance, without the QA bands.

  return image.updateMask(mask).divide(10000)

      .select('B[0-9]\*')

      .copyProperties(image, ['system:time\_start']);

}

// Use this function to add variables for NDVI, time and a constant

// to Landsat 8 imagery.

var addVariables = function(image) {

  // Compute time in fractional years since the epoch.

  var date = ee.Date(image.get(timeField));

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

  // Return the image with the added bands.

  return image

    // Add an NDVI band.

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

    // Add a time band.

    .addBands(ee.Image(years).rename('t'))

    .float()

    // Add a constant band.

    .addBands(ee.Image.constant(1));

};

// Remove clouds, add variables and filter to the area of interest.

var filteredLandsat = l8sr

  .filterBounds(roi)

  .map(maskL8sr)

  .map(addVariables);

1. **Plot the time series at the location of interest**

1. To visualize the data, make a chart at the location of interest.  Add a linear trend line for reference (you'll compute that line soon):

// Plot a time series of NDVI at a single location.

var l8Chart = ui.Chart.image.series(filteredLandsat.select('NDVI'), roi)

    .setChartType('ScatterChart')

    .setOptions({

      title: 'Landsat 8 NDVI time series at ROI',

      trendlines: {0: {

        color: 'CC0000'

      }},

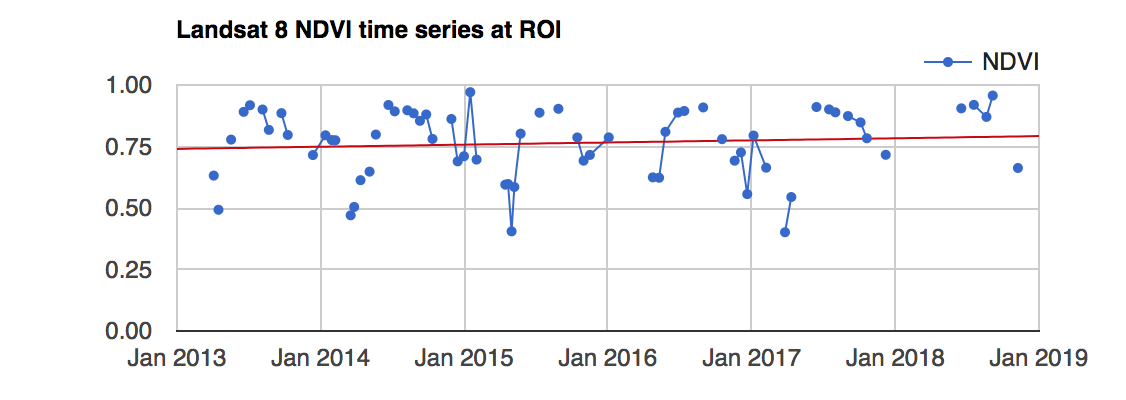
      lineWidth: 1,

      pointSize: 3,

    });

print(l8Chart);

You should see something like this:



## Linear modeling of time

Lots of interesting analyses can be done to time series by harnessing the [linearRegression() reducer](https://developers.google.com/earth-engine/api_docs#eereducerlinearregression).  For example,

1. **Estimate linear trend over time**

Consider the following linear model, where *et* is a random error:

*pt* = *β*0 + *β*1*t + et* (1)

This is the model behind the trendline added to the chart you just created.  This model is useful for detrending data and reducing [stationarity](https://en.wikipedia.org/wiki/Stationary_process) in the time series ([Shumway and Stoffer 2017](http://www.stat.pitt.edu/stoffer/tsa4/tsaEZ.pdf)).  For now, the goal is to discover the values of the *β*'s in each pixel.

1. To fit this trend model to the Landsat-based NDVI series using ordinary least squares (see Lab 5), use the linearRegression() reducer:

// List of the independent variable names

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

// Name of the dependent variable.

var dependent = ee.String('NDVI');

// Compute a linear trend.  This will have two bands: 'residuals' and

// a 2x1 band called coefficients (columns are for dependent variables).

var trend = filteredLandsat.select(independents.add(dependent))

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

// Map.addLayer(trend, {}, 'trend array image');

// Flatten the coefficients into a 2-band image

var coefficients = trend.select('coefficients')

  .arrayProject([0])

  .arrayFlatten([independents]);

The coefficients image is a two band image in which each pixel contains values for *β0* and *β1*.

1. Use the model to "detrend" the original NDVI time series:

// Compute a de-trended series.

var detrended = filteredLandsat.map(function(image) {

  return image.select(dependent).subtract(

          image.select(independents).multiply(coefficients).reduce('sum'))

          .rename(dependent)

          .copyProperties(image, [timeField]);

});

// Plot the detrended results.

var detrendedChart = ui.Chart.image.series(detrended, roi, null, 30)

    .setOptions({

      title: 'Detrended Landsat time series at ROI',

      lineWidth: 1,

      pointSize: 3,

    });

print(detrendedChart);

1. **Estimate seasonality with a harmonic model**

Consider the following linear model, where *et* is a random error, *A* is amplitude, *ω* is frequency, and *φ* is phase:

*pt* = *β*0 + *β*1*t + A*cos(2π*ωt - φ*) + *et*

    = *β*0 + *β*1*t + β*2cos(2π*ωt*) + *β*3sin(2π*ωt*) + *et* (2)

Note that *β*2 = *A*cos(*φ*) and *β*3 = *A*sin(*φ*), implying *A* = (*β*22 + *β*32)½ and *φ* = atan(*β*3/*β*2).  (See [Shumway and Stoffer (2017)](http://www.stat.pitt.edu/stoffer/tsa4/tsaEZ.pdf) equations 4.1 - 4.2).  To fit this model to the time series, set *ω*=1 (one cycle per unit time) and use ordinary least squares regression.

1. The setup for fitting the model is to first add the harmonic variables (the third and fourth terms of equation 2) to the image collection.

// Use these independent variables in the harmonic regression.

var harmonicIndependents = ee.List(['constant', 't', 'cos', 'sin']);

// Add harmonic terms as new image bands.

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

  var timeRadians = image.select('t').multiply(2 \* Math.PI);

  return image

    .addBands(timeRadians.cos().rename('cos'))

    .addBands(timeRadians.sin().rename('sin'));

});

1. Fit the model as with the linear trend, using the linearRegression() reducer:

var harmonicTrend = harmonicLandsat

  .select(harmonicIndependents.add(dependent))

  // The output of this reducer is a 4x1 array image.

  .reduce(ee.Reducer.linearRegression({

    numX: harmonicIndependents.length(),

    numY: 1

  }));

1. Plug the coefficients in to equation 2 in order to get a time series of fitted values:

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

var harmonicTrendCoefficients = harmonicTrend.select('coefficients')

  .arrayProject([0])

  .arrayFlatten([harmonicIndependents]);

// Compute fitted values.

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

  return image.addBands(

    image.select(harmonicIndependents)

      .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)

    .setSeriesNames(['NDVI', 'fitted'])

    .setOptions({

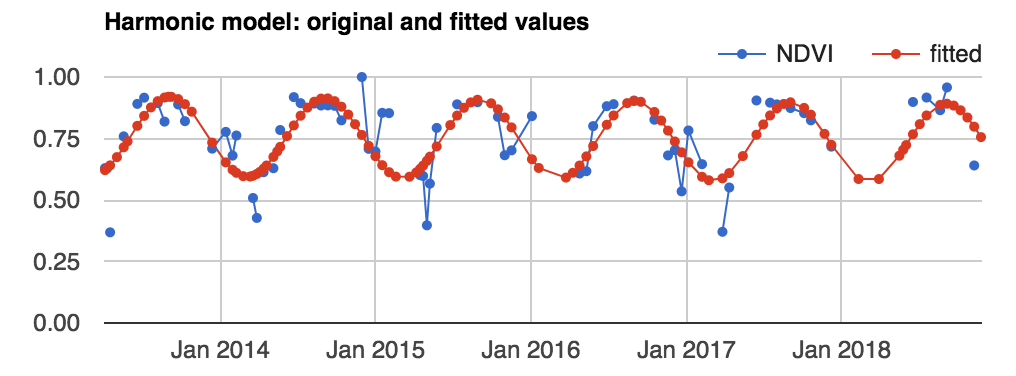
      title: 'Harmonic model: original and fitted values',

      lineWidth: 1,

      pointSize: 3,

}));

You should see something like:



1. Although any coefficients can be mapped directly, it is useful and interesting to map the phase and amplitude of the estimated harmonic model.  First, compute phase and amplitude from the coefficients, then map:

// Compute phase and amplitude.

var phase = harmonicTrendCoefficients.select('sin')

    .atan2(harmonicTrendCoefficients.select('cos'))

    // Scale to [0, 1] from radians.

    .unitScale(-Math.PI, Math.PI);

var amplitude = harmonicTrendCoefficients.select('sin')

    .hypot(harmonicTrendCoefficients.select('cos'))

    // Add a scale factor for visualization.

    .multiply(5);

// Compute the mean NDVI.

var meanNdvi= filteredLandsat.select('NDVI').mean();

// Use the HSV to RGB transform to display phase and amplitude.

var rgb = ee.Image.cat([

  phase,      // hue

  amplitude,  // saturation (difference from white)

  meanNdvi    // value (difference from black)

]).hsvToRgb();

Map.addLayer(rgb, {}, 'phase (hue), amplitude (sat), ndvi (val');