Google Earth Engine

Time Series Analysis in Earth Engine

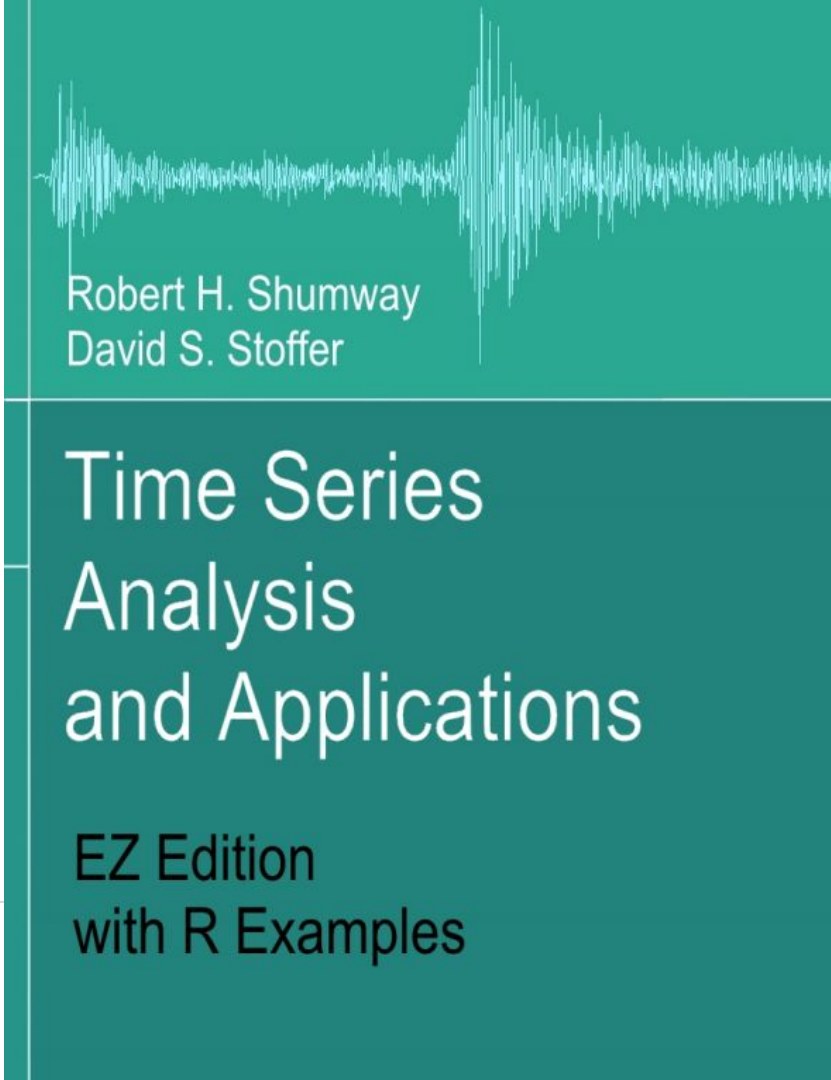
Nick Clinton, nclinton@google.com
Earth Engine Developer Relations

https://goo.gl/lMwd2Y

Disclaimer: this is not a statistics course!

(It's about how to do things in Earth Engine)

See this text for background:

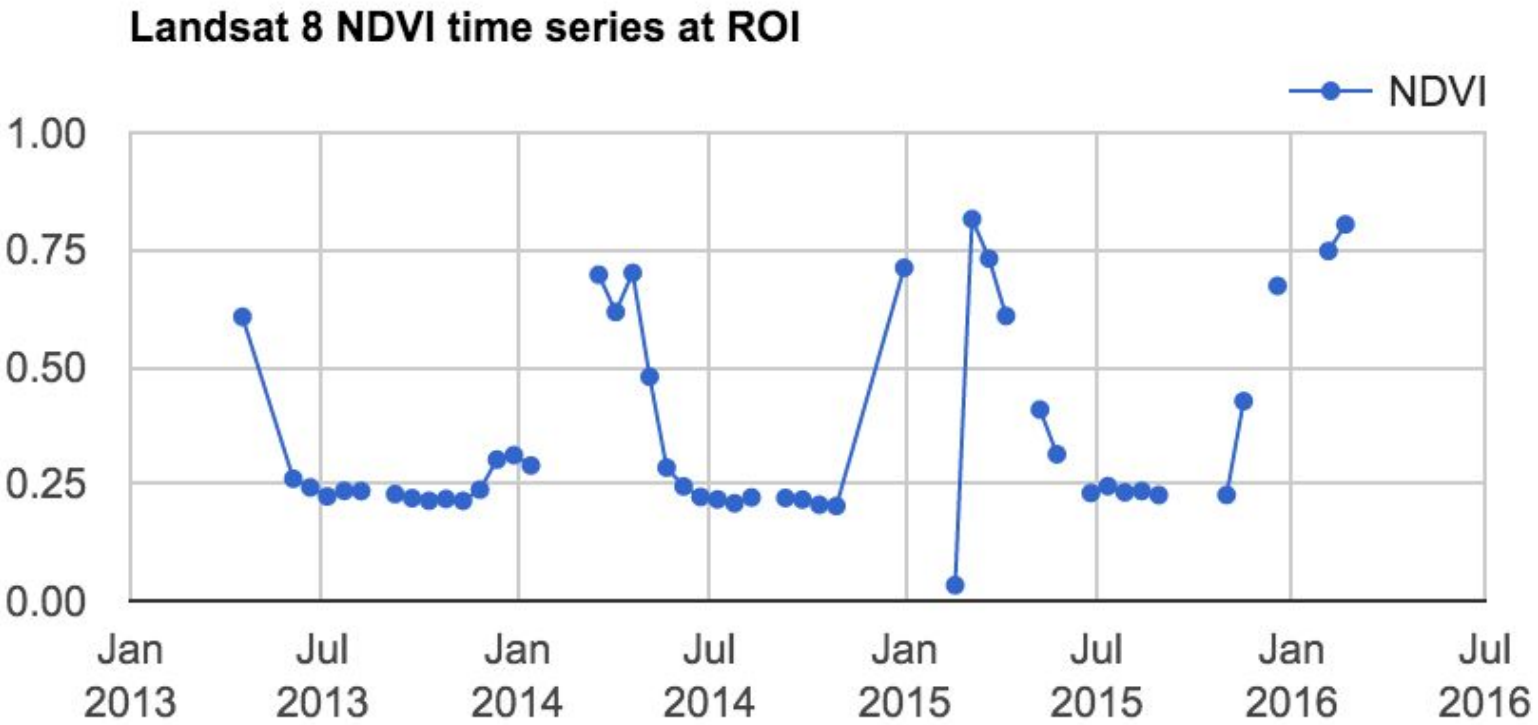


What this course IS about:

- Our time series: Landsat 8 NDVI, about 64 images (and growing...)
- A framework for time series analysis in Earth Engine using **join(), map(), reduce()**
- Earth Engine examples:
  - Linear modeling
  - Harmonic modeling
  - Auto-covariance and auto-correlation
  - Cross-covariance and cross-correlation
  - Auto-regressive models (time permitting)

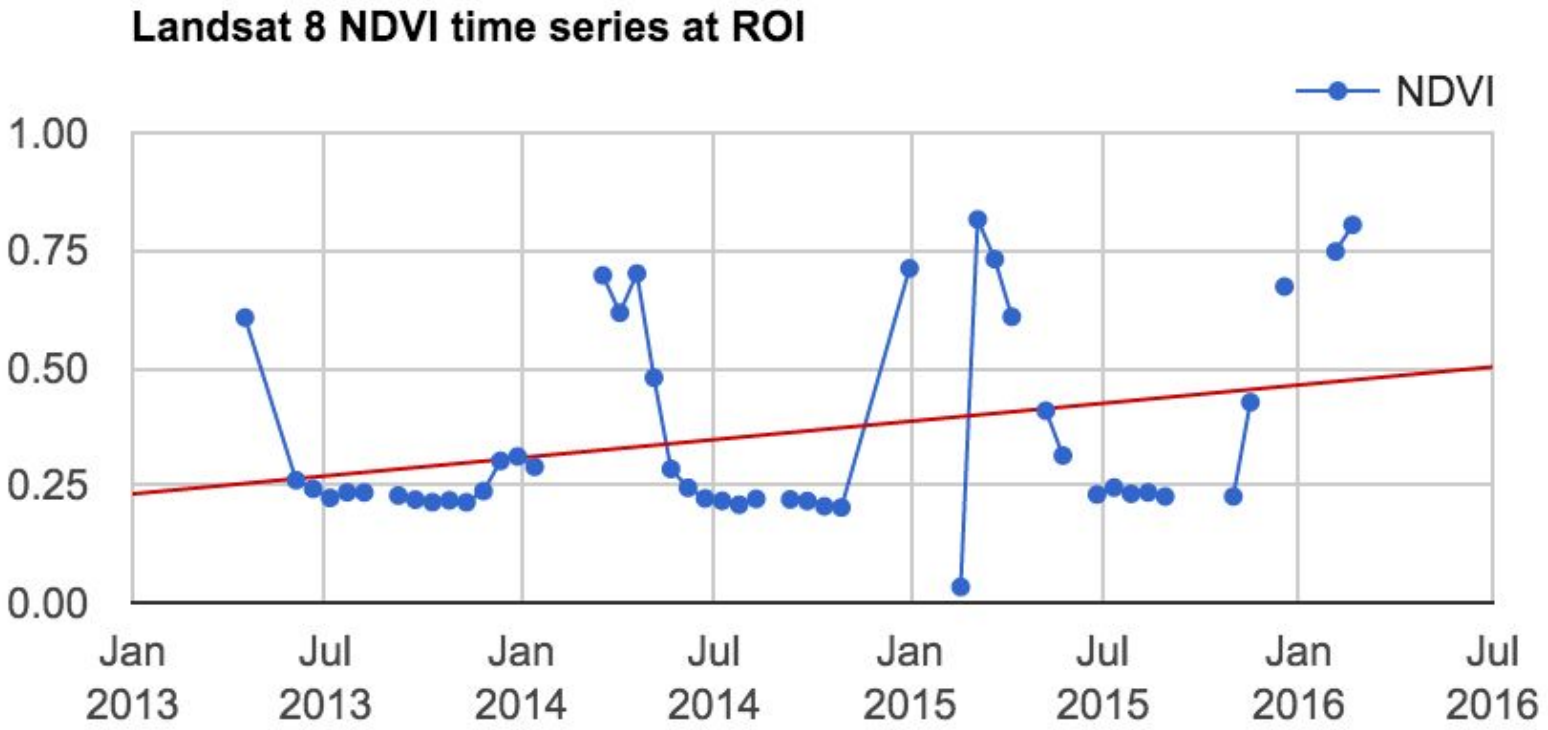
Linear model of time

`ImageCollection`: irregularly spaced points, missing data, noisy data



https://code.earthengine.google.com/0827ce3bbe401c769090e1d192d47b74

Red line: linear trend



https://code.earthengine.google.com/0827ce3bbe401c769090e1d192d47b74

Linear model of time (trend):

$$p_t = \beta_0 + \beta_1 t + e_t$$

p_t = *NDVI* at time t
t = time
e_t = random error

We want β's for every pixel...

To give each image bands representing independent variables, `map()` a function over the collection:

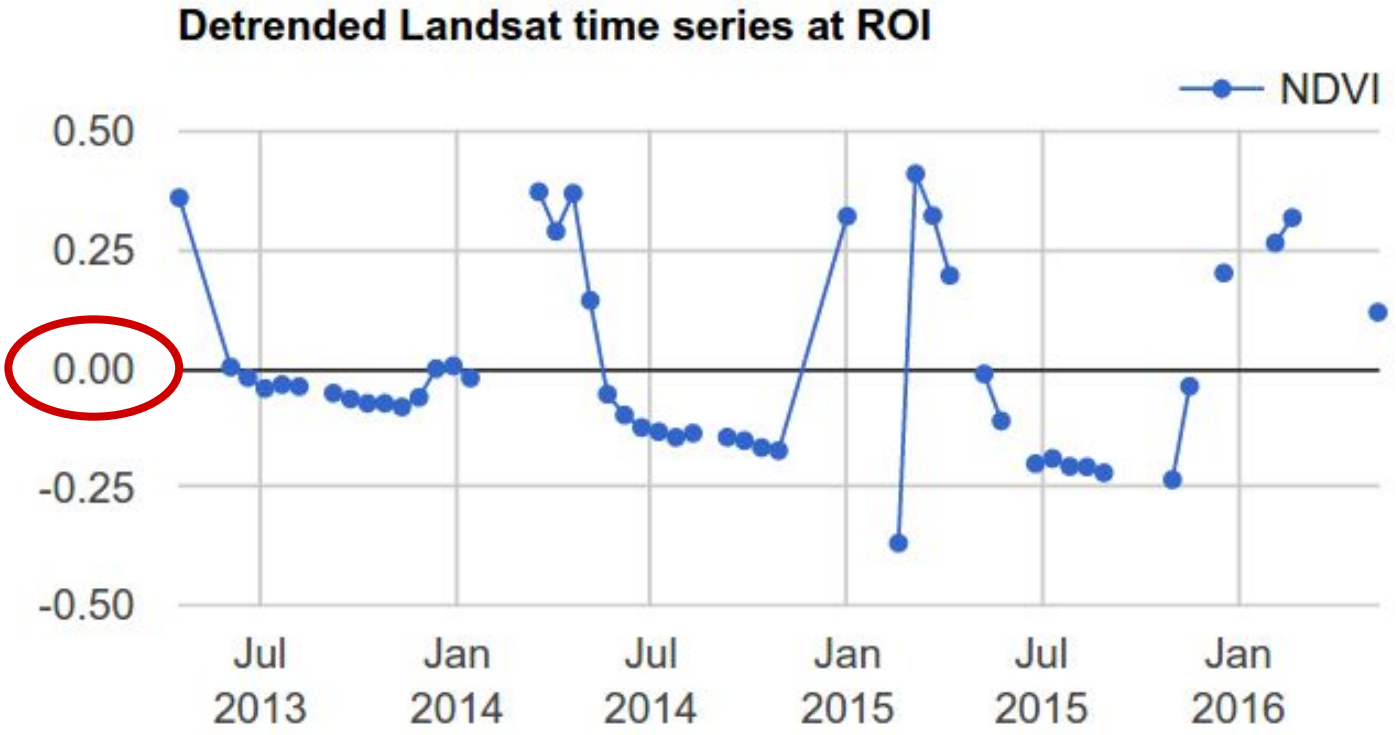
```
var addVariables = function(image) {
  var date = ee.Date(image.get(timeField));
  var years = date.difference(ee.Date('1970-01-01'), 'year');
  return image
    .addBands(image.normalizedDifference(['B5', 'B4']).rename('NDVI')).float()
    .addBands(ee.Image(years).rename('t').float())
    .addBands(ee.Image.constant(1));
};
```

Compute the β's using the `linearRegression` reducer:

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

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

var trend = filteredLandsat.select(independents.add(dependent))
    .reduce(ee.Reducer.linearRegression(independents.length(), 1));

var coefficients = trend.select('coefficients')
     .arrayProject([0])
     .arrayFlatten([independents]);
```


https://code.earthengine.google.com/6cc3ea3444b0032fbf362e2b6227f36a

"Detrend" (compute p_t - $\hat{p}_t$):



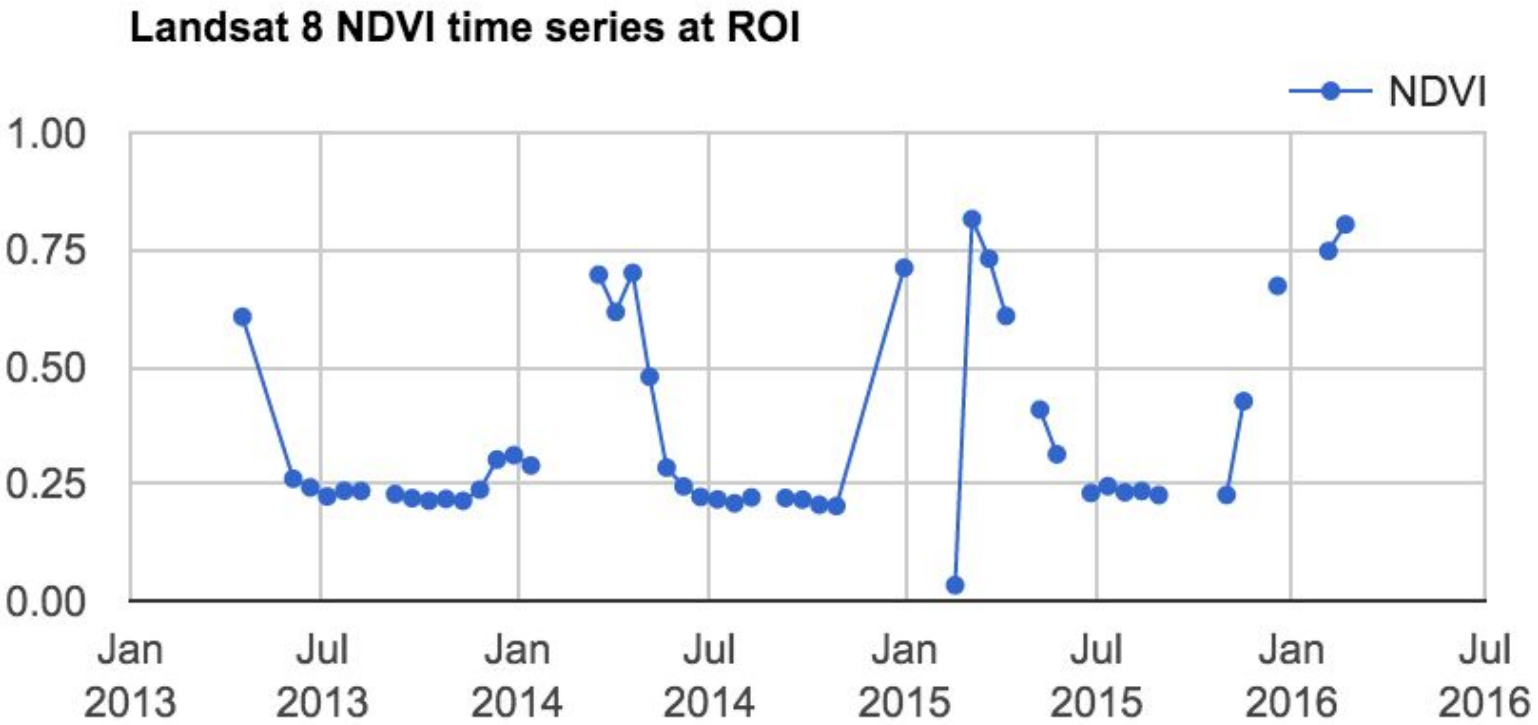
To detrend, map() a function over the collection:

```
var detrended = filteredLandsat.map(function(image) {
  return image.select(dependent).subtract(
          image.select(independents).multiply(coefficients).reduce('sum'))
          .rename(dependent)
          .copyProperties(image, [timeField]);
});
```

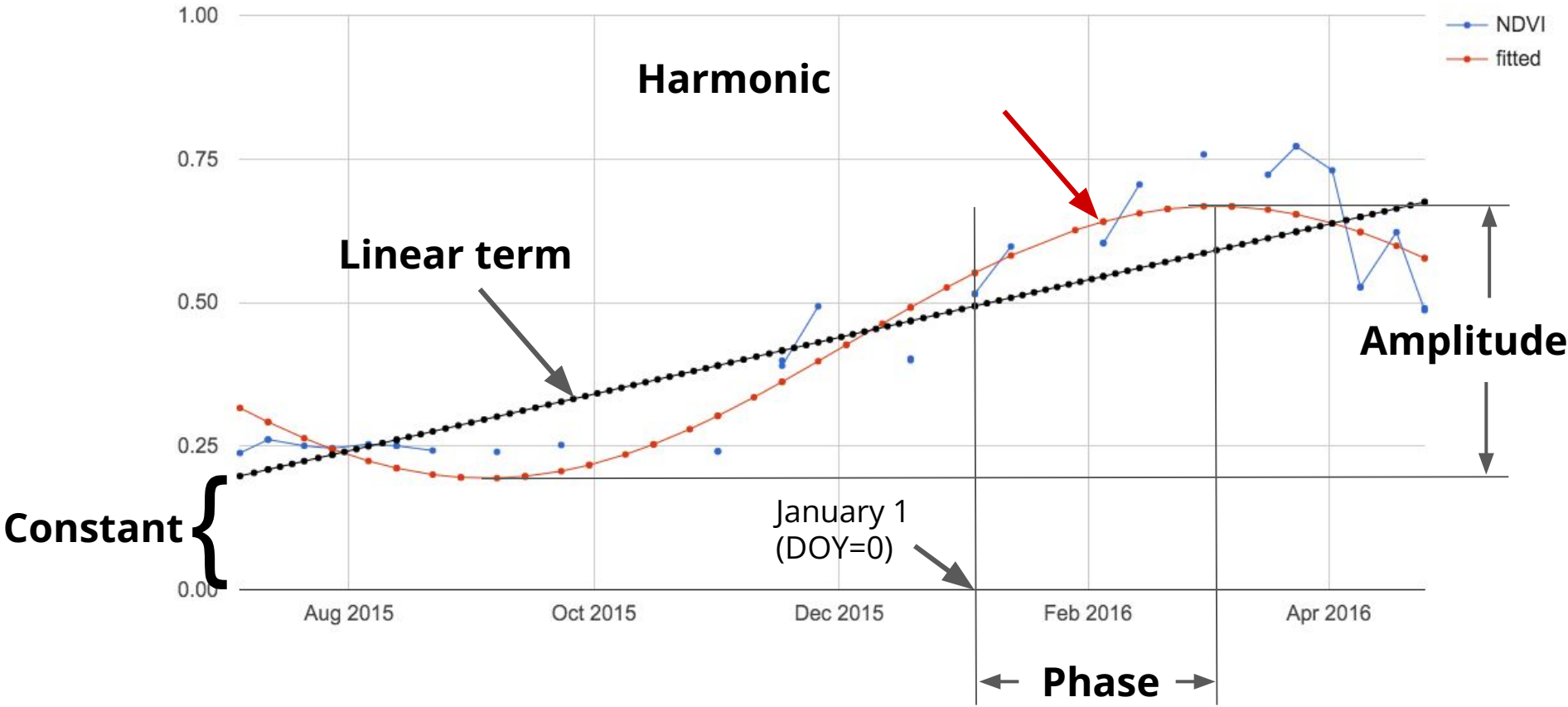

https://code.earthengine.google.com/25baa7d588882c7f86b2e3c0334272fb

Harmonic model of time

Original data: a time series with a regular cycle?



https://code.earthengine.google.com/0827ce3bbe401c769090e1d192d47b74


1.00
0.75
0.50
0.25
0.00
NDVI
fitted
Harmonic
Linear term
Amplitude
Constant
January 1
(DOY=0)
Phase
Aug 2015
Oct 2015
Dec 2015
Feb 2016
Apr 2016

Shumway and Stoffer (2017), equations 4.1-4.2:

$$A\cos(2\pi\omega t - \varphi) = \beta_2\cos(2\pi\omega t) + \beta_3\sin(2\pi\omega t)$$

$\beta_2 = A\cos(\varphi)$
$\beta_3 = A\sin(\varphi)$
A = **amplitude** = $(\beta_2^{\,2} + \beta_3^{\,2})^{½}$
φ = **phase** = $\text{atan}(\beta_3/\beta_2)$
ω = angular frequency

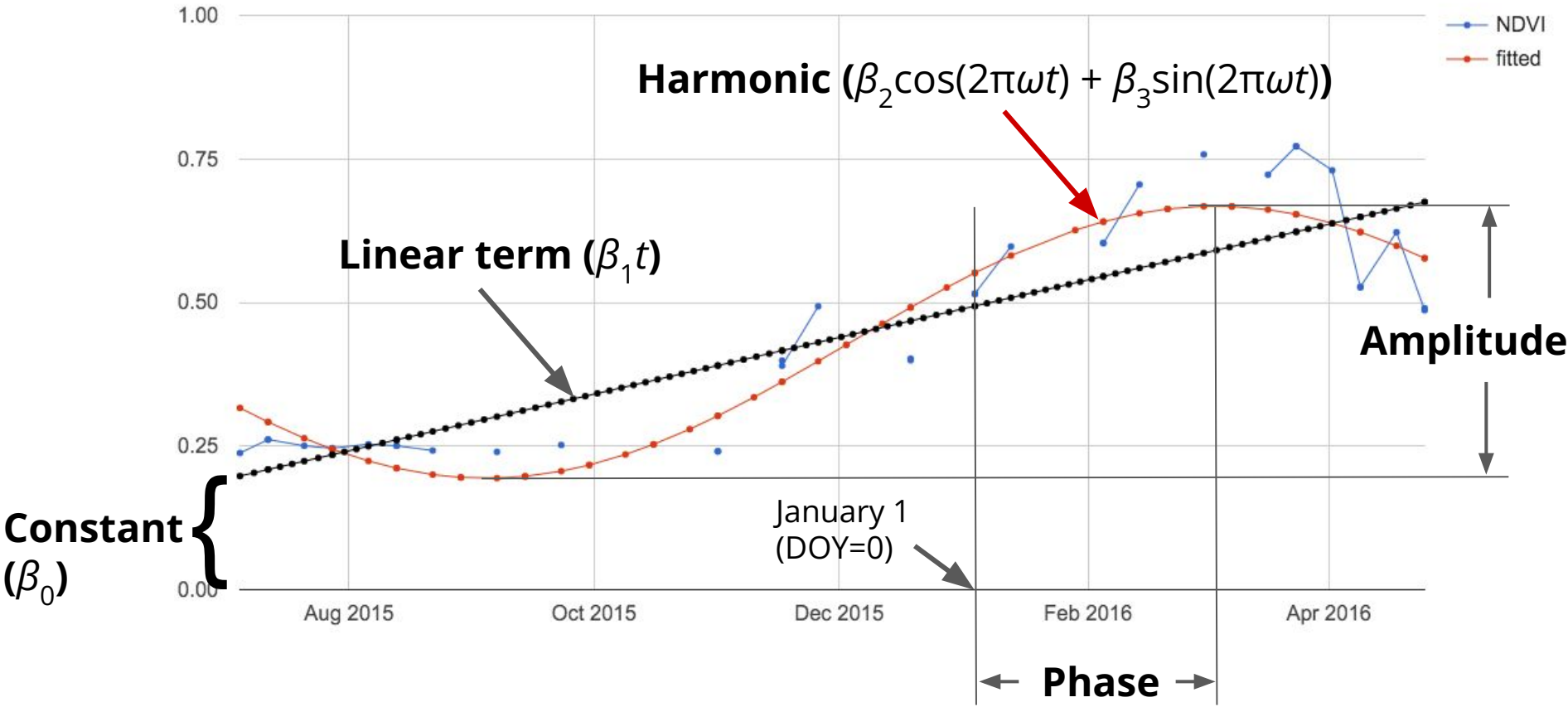
Use this to linearize the harmonic function of phase and amplitude...

Fit the following linear model with ($\omega = 1$):

$$p_t = NDVI_t = \beta_0 + \beta_1 t + A\cos(2\pi\omega t - \varphi) + e_t$$

$$= \beta_0 + \beta_1 t + \beta_2 \cos(2\pi\omega t) + \beta_3 \sin(2\pi\omega t) + e_t$$

$$NDVI_t = \beta_0 + \beta_1 t + \beta_2 \cos(2\pi\omega t) + \beta_3 \sin(2\pi\omega t) + \ldots$$



Compute the β's using the `linearRegression` reducer:

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

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'));
});

var harmonicTrend = harmonicLandsat
  .select(harmonicIndependents.add(dependent))
  .reduce(ee.Reducer.linearRegression(harmonicIndependents.length(), 1));
```

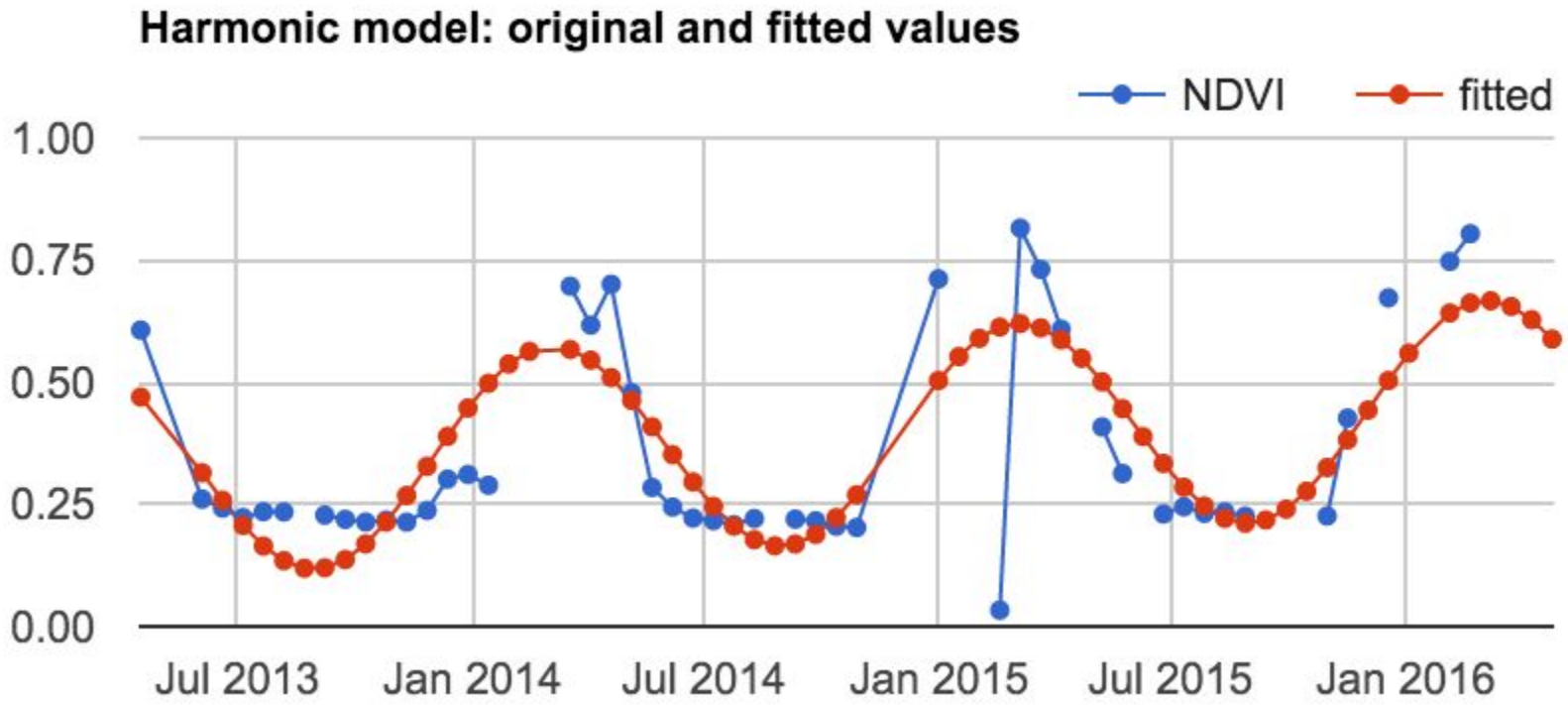

https://code.earthengine.google.com/e509dc5852e56f46b5805b5bbd514725

Plug the coefficients into the linear model:

```
var harmonicTrendCoefficients = harmonicTrend.select('coefficients')
  .arrayProject([0])
  .arrayFlatten([harmonicIndependents]);

var fittedHarmonic = harmonicLandsat.map(function(image) {
  return image.addBands(
    image.select(harmonicIndependents)
      .multiply(harmonicTrendCoefficients)
      .reduce('sum')
      .rename('fitted'));
});
```


https://code.earthengine.google.com/7aeaecf64629c3b5221305a6b7775c2c

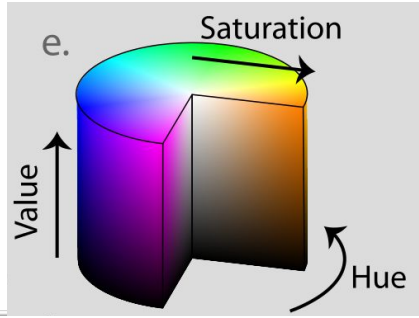
Plot the results:



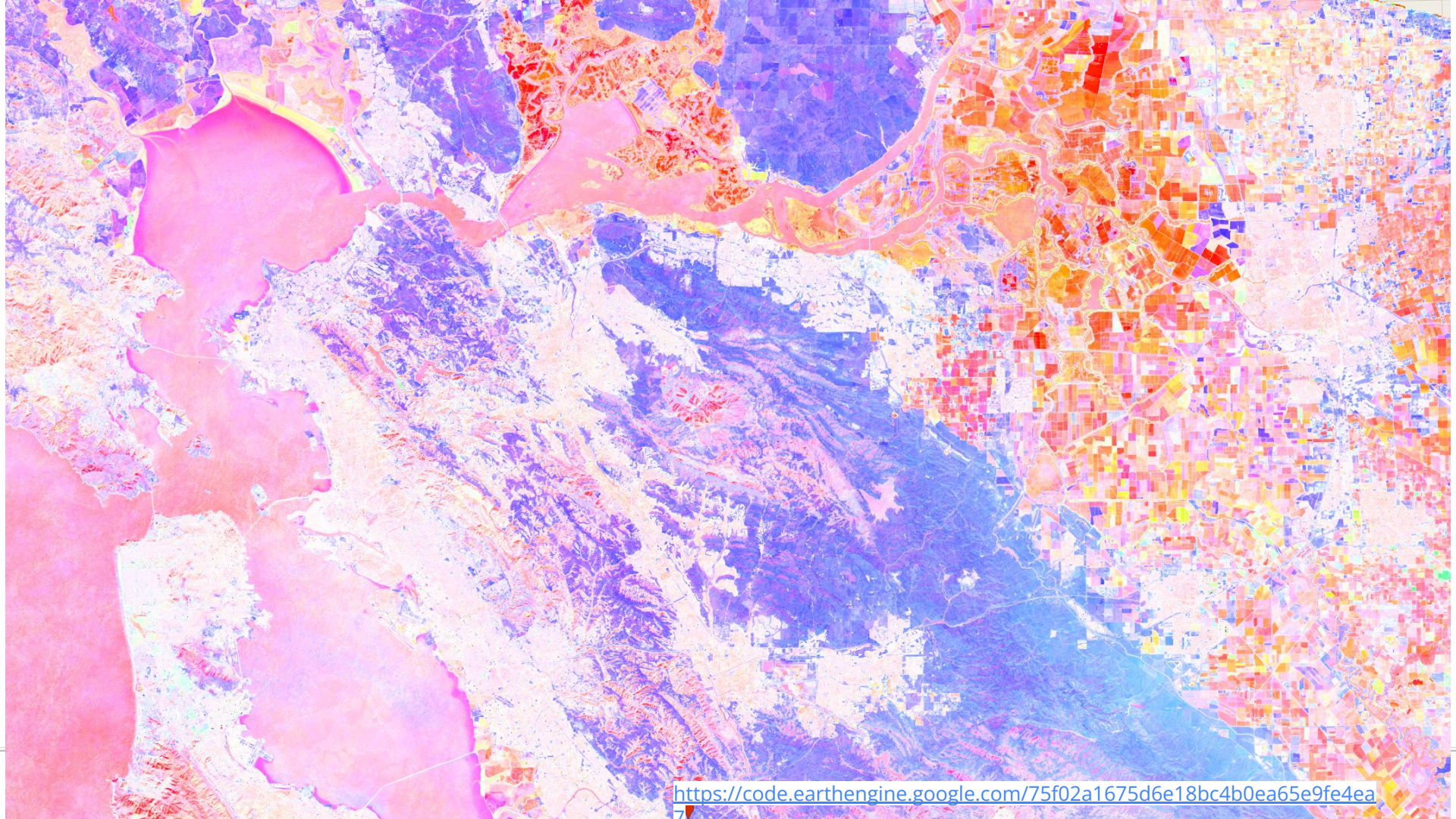
https://code.earthengine.google.com/7aeaecf64629c3b5221305a6b7775c2c

Map the results:

```
var phase = harmonicTrendCoefficients.select('cos').atan2(
            harmonicTrendCoefficients.select('sin'));

var amplitude = harmonicTrendCoefficients.select('cos').hypot(
                harmonicTrendCoefficients.select('sin'));

var rgb = phase.unitScale(-Math.PI, Math.PI).addBands(   // hue
          amplitude.multiply(2.5)).addBands(             // saturation
          ee.Image(1)).hsvToRgb();                       // value

Map.addLayer(rgb, {}, 'phase (hue), amplitude (saturation)');
```



https://code.earthengine.google.com/9701bfb171d2911884a4306488ea7c3d


https://code.earthengine.google.com/75f02a1675d6e18bc4b0ea65e9fe4ea

Auto-covariance and Auto-correlation

Shumway and Stoffer (2016), equations 1.27, 1.26

(assume stationarity):

$$\text{AutoCovariance} = \text{Cov}(p_t, p_{t-l}) = E[(p_t - E[p_t])(p_{t-l} - E[p_{t-l}])]$$

$$\text{AutoCorrelation} = \text{Corr}(p_t, p_{t-l})$$
$$= \text{Cov}(p_t, p_{t-l}) \,/\, (\text{SD}(p_t)\text{SD}(p_{t-l}))$$

t = time

$p_t = NDVI_t$ = pixel value at time t

$p_{t-l} = NDVI_{t-l}$ = pixel value at time t-l

Get p_t and p_{t-l} into the same collection with a join:

```
var lag = function(leftCollection, rightCollection, lagDays) {
  var filter = ee.Filter.and(
    ee.Filter.maxDifference({
      difference: 1000 * 60 * 60 * 24 * lagDays,
      leftField: timeField,
      rightField: timeField
    }),
    ee.Filter.greaterThan({
      leftField: timeField,
      rightField: timeField
  }));

  return ee.Join.saveAll({
    matchesKey: 'images',
    measureKey: 'delta_t',
    ordering: timeField,
    ascending: false, // Sort reverse chronologically
  }).apply({
    primary: leftCollection,
    secondary: rightCollection,
    condition: filter
  });
};
```


https://code.earthengine.google.com/6b69374329dd43110cdd4c80d8d65784

Join and add the lagged images as bands:

```
var lagged17 = lag(detrended, detrended, 17);

var merge = function(image) {
  var merger = function(current, previous) {
    return ee.Image(previous).addBands(current);
  };
  return ee.ImageCollection.fromImages(image.get('images')).iterate(merger, image);
};

var merged17 = ee.ImageCollection(lagged17.map(merge));
```

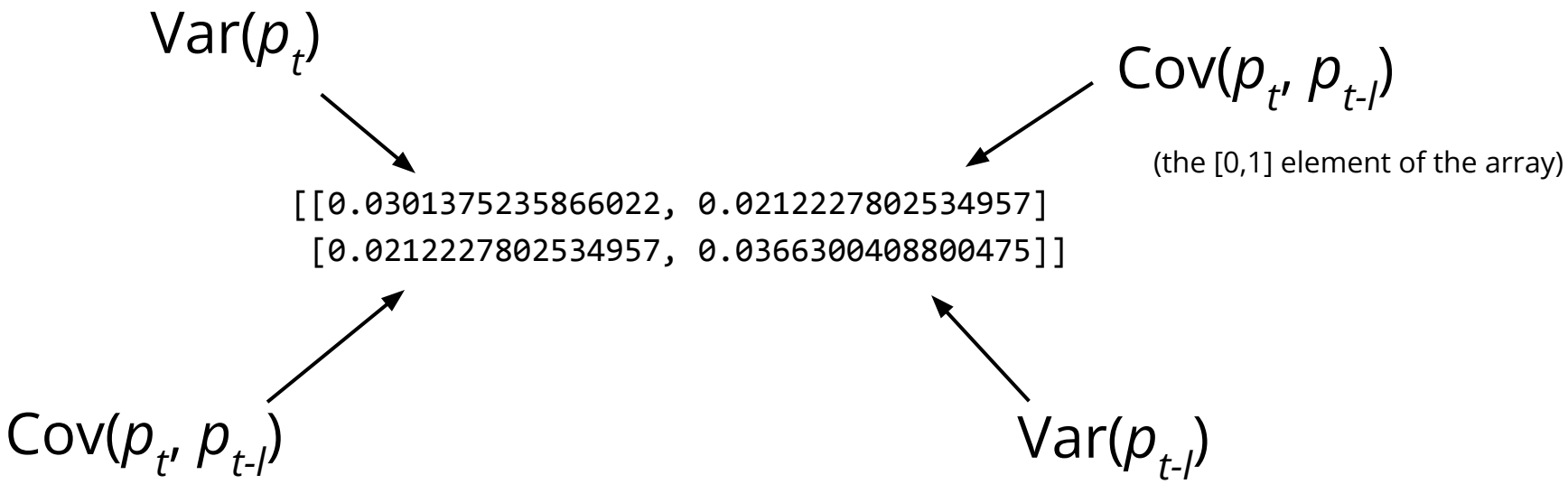

https://code.earthengine.google.com/f78114ea2dbd973700855f6ce8b711fb

Compute and display covariance:

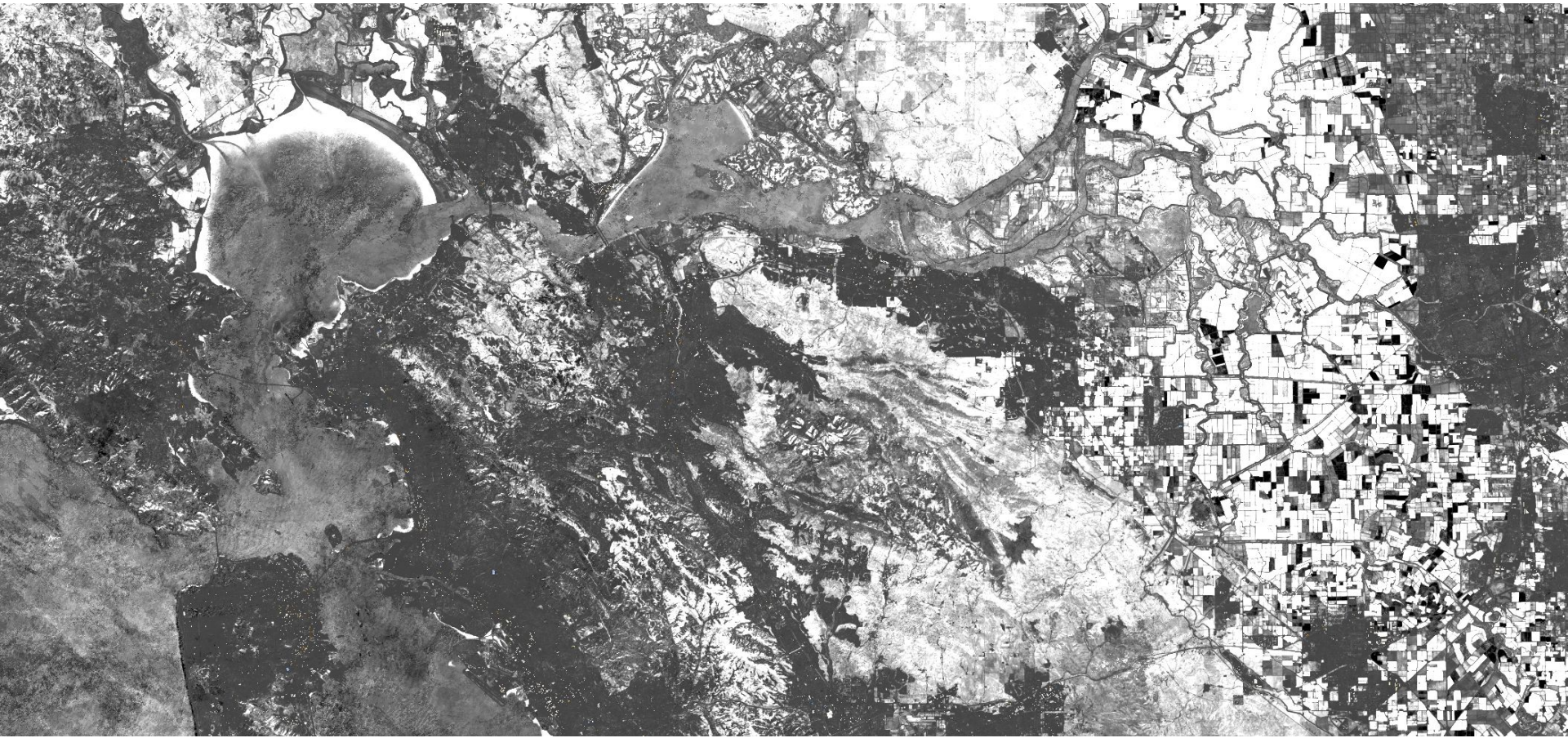
```
var covariance = function(mergedCollection, band, lagBand) {
  return mergedCollection.select([band, lagBand]).map(function(image) {
    return image.toArray();
  }).reduce(ee.Reducer.covariance(), 8);
};

var lagBand = dependent.cat('_1');

var covariance17 = ee.Image(covariance(merged17, dependent, lagBand));

Map.addLayer(covariance17.arrayGet([0, 1]), {}, 'covariance (lag = 17 days)');
```


https://code.earthengine.google.com/62d4cb7632fb73335d0130620b0e2d32

Variance-Covariance matrix in a pixel (the ROI):



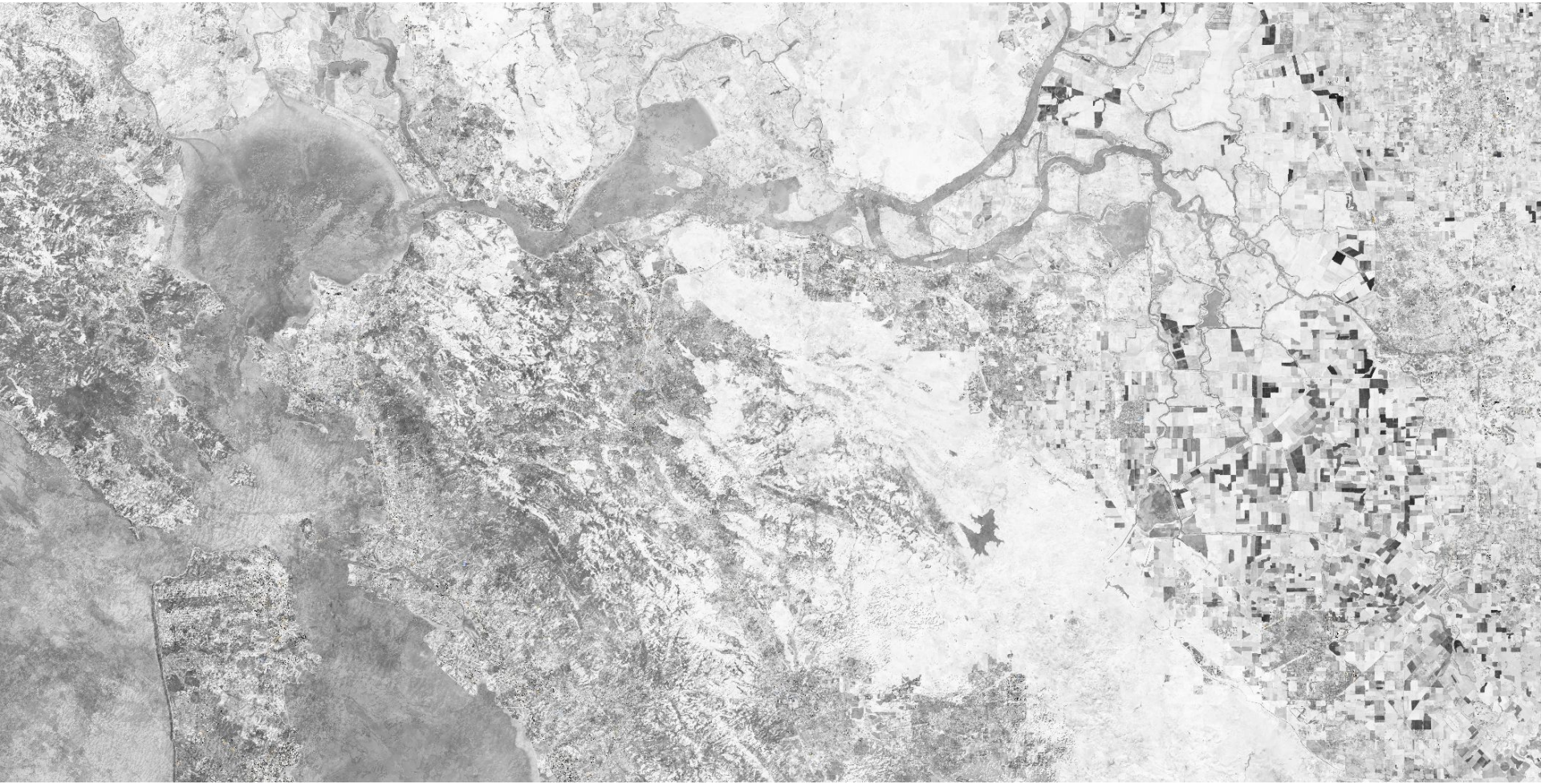

```
print(covariance17.reduceRegion('first', roi, 30));
```



Auto-correlation:

```
var correlation = function(vcArrayImage) {
  var covariance = ee.Image(vcArrayImage).arrayGet([0, 1]);
  var sd0 = ee.Image(vcArrayImage).arrayGet([0, 0]).sqrt();
  var sd1 = ee.Image(vcArrayImage).arrayGet([1, 1]).sqrt();
  return covariance.divide(sd0).divide(sd1).rename('correlation');
};

var correlation17 = correlation(covariance17);

Map.addLayer(correlation17, {min: -1, max: 1}, 'correlation (lag = 17 days)');
```


https://code.earthengine.google.com/8555fe060c685a048c13b2c8d0b634a9



Google Earth Engine

https://code.earthengine.google.com/8555fe060c685a048c13b2c8d0b634a9

Cross-correlation

Google Earth Engine

Shumway and Stoffer (2017), equations 1.30-1.31:

$$\text{CrossCovariance} = \text{Cov}(p_t, c_{t-l}) = \text{E}[(p_t - \text{E}[p_t])(c_{t-l} - \text{E}[c_{t-l}])]$$

$$\text{CrossCorrelation} = \text{Corr}(p_t, c_{t-l})$$
$$= \text{Cov}(p_t, c_{t-l}) \,/\, (\text{SD}(p_t)\text{SD}(c_{t-l}))$$

t = time

$p_t = NDVI_t$ = pixel value at time t

c_{t-l} = covariate value at time $t-l$

Cross-covariance and cross-correlation:

```
// Precipitation (covariate)
var chirps = ee.ImageCollection('UCSB-CHG/CHIRPS/PENTAD');

// Join the t-1 (l = 1 pentad) precipitation images to the Landsat.
var lag1PrecipNDVI = lag(filteredLandsat, chirps, 5);

var merged1PrecipNDVI = ee.ImageCollection(lag1PrecipNDVI.map(merge));

var cov1PrecipNDVI = covariance(merged1PrecipNDVI, 'NDVI', 'precipitation');
Map.addLayer(cov1PrecipNDVI.arrayGet([0, 1]), {}, 'NDVI - PRECIP cov (lag = 5)');

var corr1PrecipNDVI = correlation(cov1PrecipNDVI);
Map.addLayer(corr1PrecipNDVI, {min: -0.5, max: 0.5}, 'NDVI - PRECIP corr (lag = 5)');
```


https://code.earthengine.google.com/5270e4c7ab8ce495180c235fcf9da9f0

Cross-covariance and cross-correlation, part 2:

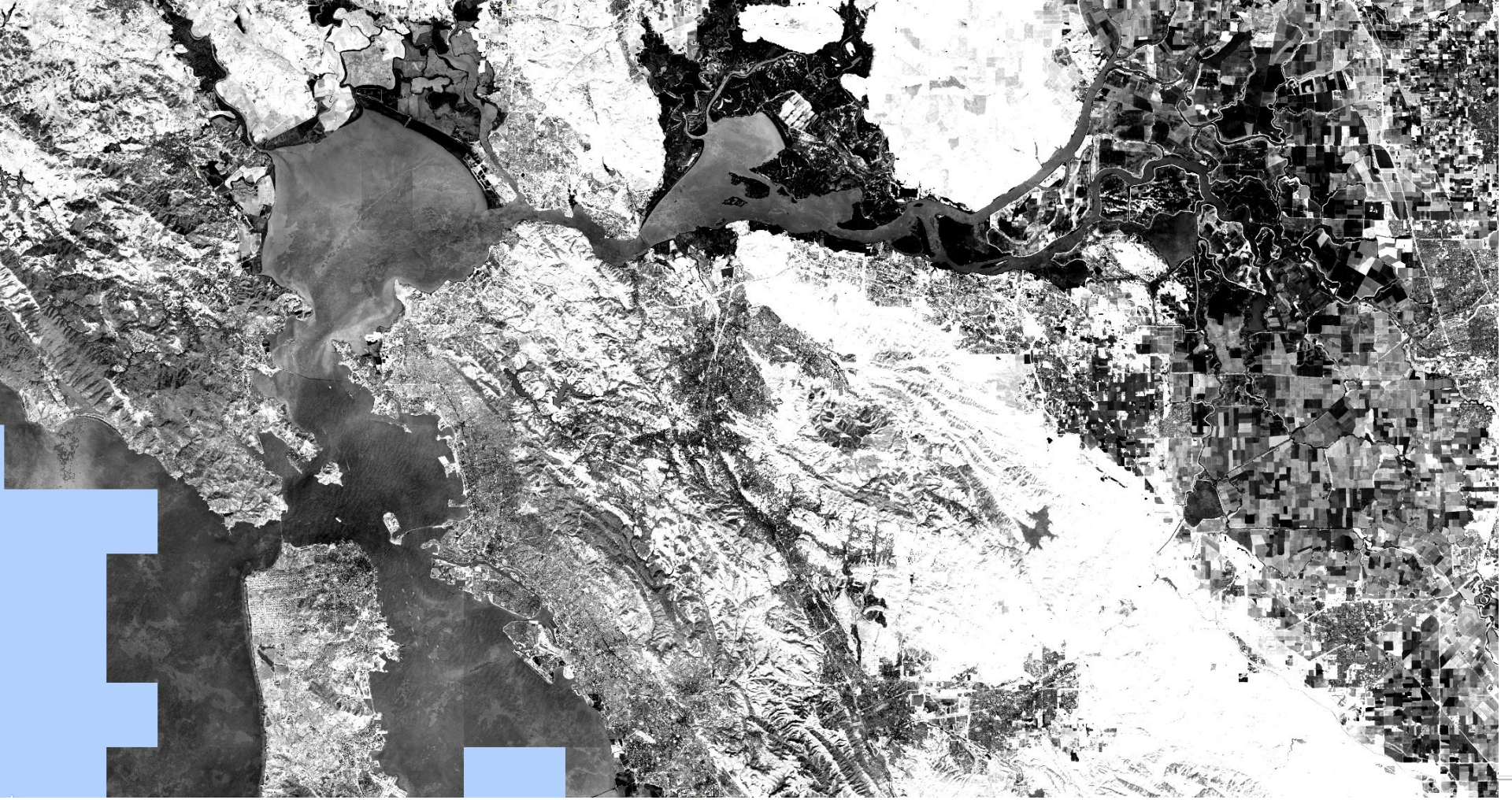
```
// Join the precipitation images from the previous month
var lag30PrecipNDVI = lag(filteredLandsat, chirps, 30);

var sum30PrecipNDVI = ee.ImageCollection(lag30PrecipNDVI.map(function(image) {
  var laggedImages = ee.ImageCollection.fromImages(image.get('images'));
  return ee.Image(image).addBands(laggedImages.sum().rename('sum'));
}));

var cov30PrecipNDVI = covariance(sum30PrecipNDVI, 'NDVI', 'sum');
Map.addLayer(cov1PrecipNDVI.arrayGet([0, 1]), {}, 'NDVI - sum cov (lag = 30)');

var corr30PrecipNDVI = correlation(cov30PrecipNDVI);
Map.addLayer(corr30PrecipNDVI, {min: -0.5, max: 0.5}, 'NDVI - sum corr (lag = 30)');
```


https://code.earthengine.google.com/b741cd9406890a2075c47474da85625f

Warning:

Shumway and Stoffer (2017), Example 1.26:

"...it is worthwhile to discuss the idea of **prewhitening** a series **prior to** a **cross-correlation** analysis. The basic idea is simple... at least one of the series must be white noise. If this is not the case, there is no simple way to tell if a cross-correlation estimate is significantly different from zero."



Google Earth Engine

https://code.earthengine.google.com/b741cd9406890a2075c47474da85625f

Auto-regressive models

Auto-regressive (AR) model:

$$p_t = \beta_0 + \beta_1 p_{t-1} + \beta_2 p_{t-2} + e_t$$

p_t = *NDVI* at time t
t = time
e_t = random error

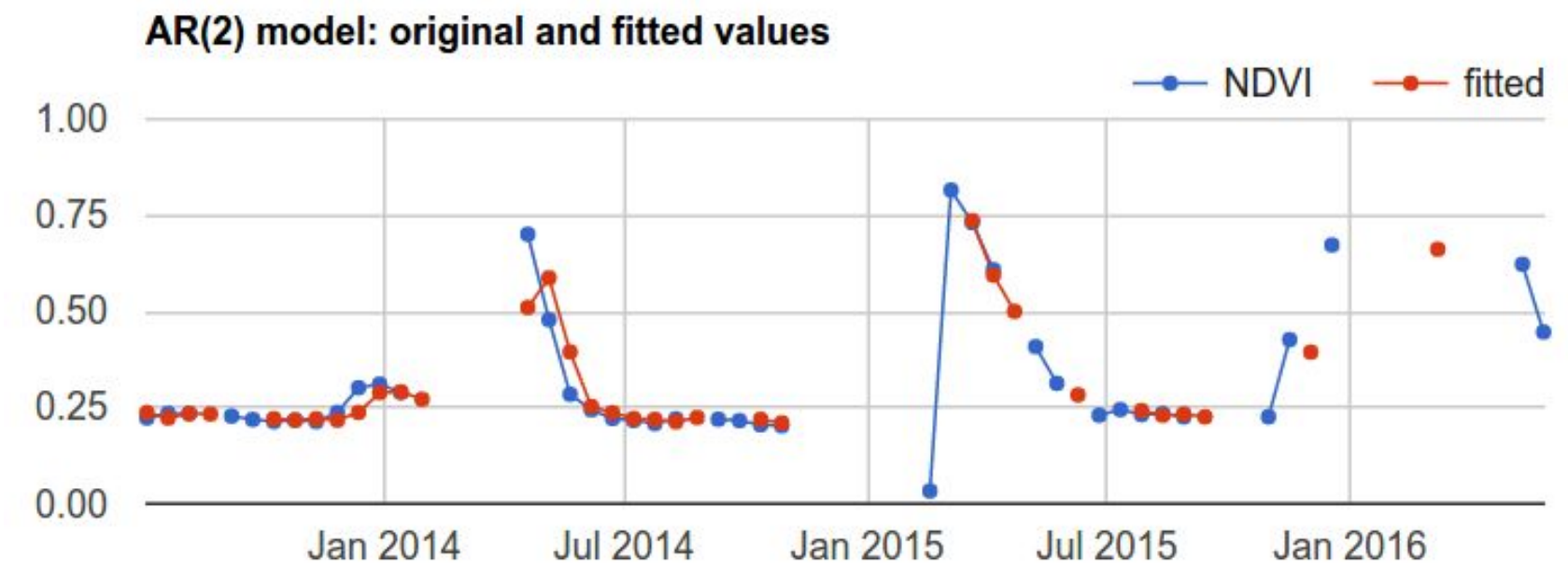
We want β's for every pixel...

AR model:

There's a problem here...

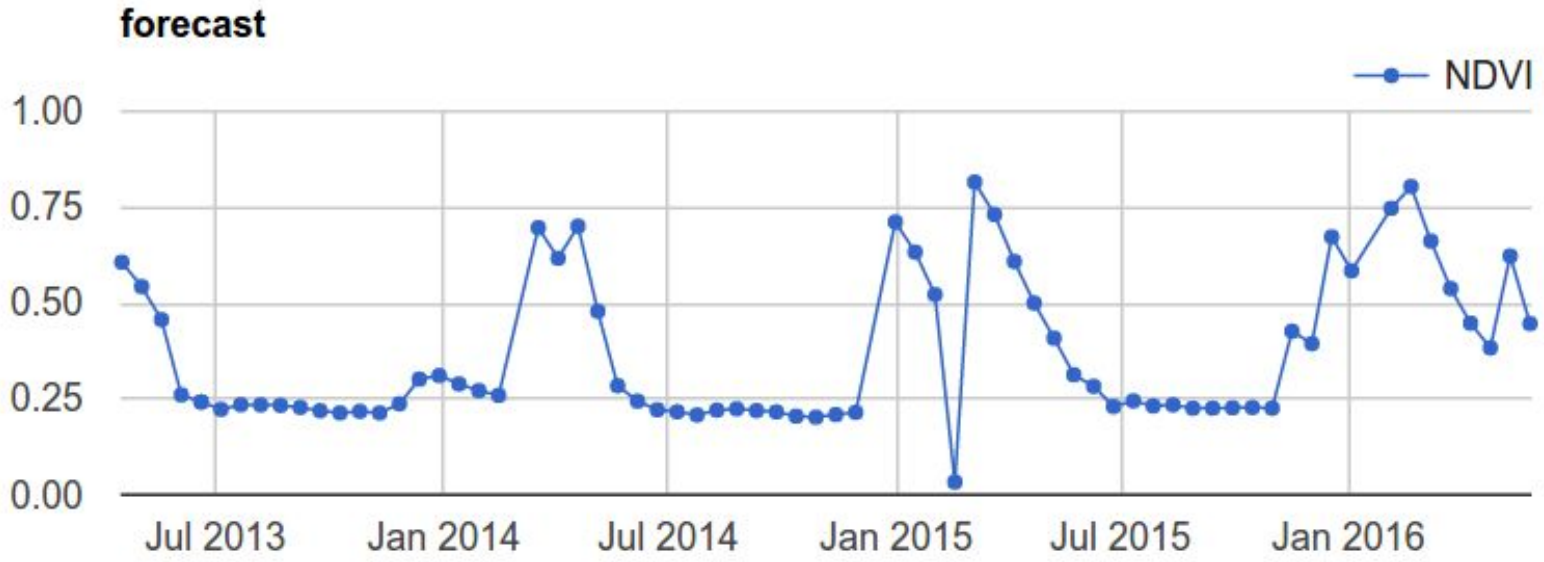
```
var lagged34 = ee.ImageCollection(lag(filteredLandsat, filteredLandsat, 34));

var merged34 = lagged34.map(merge).map(function(image) {
  return image.set('n', ee.List(image.get('images')).length());
}).filter(ee.Filter.gt('n', 1));

var arIndependents = ee.List(['constant', 'NDVI_1', 'NDVI_2']);

var ar2 = merged34
  .select(arIndependents.add(dependent))
  .reduce(ee.Reducer.linearRegression(arIndependents.length(), 1));

var arCoefficients = ar2.select('coefficients')
  .arrayProject([0])
  .arrayFlatten([arIndependents]);

var fittedAR = merged34.map(function(image) {
  return image.addBands(
    image.expression('beta0 + beta1 * p1 + beta2 * p2', {
      p1: image.select('NDVI_1'),
      p2: image.select('NDVI_2'),
      beta0: arCoefficients.select('constant'),
      beta1: arCoefficients.select('NDVI_1'),
      beta2: arCoefficients.select('NDVI_2')
    }).rename('fitted'));
});
```

Missing data:



https://code.earthengine.google.com/2785ea1a318ef249610310ad1e224602

Filling in missing data and forecasting with an AR model.



Warning! Here be dragons:

https://code.earthengine.google.com/c5fbadc5136d94e80ba428376b05d148

Pop Quiz!

How would you implement smoothing?

https://code.earthengine.google.com/37bbd8aa75040a810f972a58c09755bb

Smoothing

```
var join = ee.Join.saveAll({
  matchesKey: 'images'
});

var diffFilter = ee.Filter.maxDifference({
  difference: 1000 * 60 * 60 * 24 * 17,
  leftField: timeField,
  rightField: timeField
});

var threeNeighborJoin = join.apply({
  primary: modeled,
  secondary: modeled,
  condition: diffFilter
});

var smoothed = ee.ImageCollection(threeNeighborJoin.map(function(image) {
  var collection = ee.ImageCollection.fromImages(image.get('images'));
  return ee.Image(image).addBands(collection.mean().rename('mean'));
}));
```


https://code.earthengine.google.com/37bbd8aa75040a810f972a58c09755bb

What will you create with Time Series?

code.earthengine.google.com