ΔΠΜΣ Επιστήμη Δεδομένων και Μηχανική Μάθηση

Ανάλυση Γεωχωρικών Δεδομένων

Εργαστήριο 1

Ονοματεπώνυμο: Ευάγγελος Τσόγκας

Αριθμός Μητρώου: 03400120

1. Περιγραφή Πληροφοριακών Συστημάτων

Πριν προχωρήσουμε στην ανάλυση δεδομένων παρατήρησης γης εξερευνούμε σχετικά πληροφοριακά συστήματα με σκοπό να τα αξιολογήσουμε και να τα συγκρίνουμε. Αυτά τα συστήματα είναι το Google Earth Engine, το CREODIAS, το sobloo και το WEkEO.

Google Earth Engine

Το Google Earth Engine αποτελεί ένα αποτελεσματικό εργαλείο για την ανάλυση γεωχωρικών δεδομένων, καθώς παρέχει μεγάλη ποικιλία και ποσότητα διαθέσιμων δεδομένων τα οποία μπορούμε να οπτικοποιήσουμε μέσω του Explorer και να αναλύσουμε χρησιμοποιώντας τον Code Editor. Ο Explorer έχει πολύ καλή και εύχρηστη διεπαφή χρήστη, ενώ ο Code Editor ως ένα web-based περιβάλλον καθιστά εύκολη και γρήγορη την ανάλυση δεδομένων με χρήση κώδικα όπως JavaScript ή Python. Μάλιστα, υπάρχουν αρκετά παραδείγματα χρήσης του και είναι σχετικά φιλικό περιβάλλον ακόμα και για χρήστες χωρίς σημαντικό γνωστικό υπόβαθρο σε γεωχωρικά δεδομένα. Τέλος, παρέχει ακόμα και υπηρεσίες μηχανικής μάθησης με εξειδικευμένους αλγορίθμους, όπως για δεδομένα αισθητήρων.

CREODIAS

Το CREODIAS προσφέρει υπολογιστικές υπηρεσίες μέσω Virtual Machines, dedicated servers κλπ., αλλά παρέχει και περιβάλλοντα για προβολή και αποθήκευση δεδομένων με χρήση των EO Browser και Finder. Οι διεπαφές του αν και ωραίες με χρήσιμα εργαλεία, όπως οπτικοποίηση NDVI, είναι πιο δύσχρηστες σε σχέση με το Google Earth Engine και επίσης αν και παρέχει παραδείγματα περιπτώσεων χρήσης δεν υπάρχουν παραδείγματα ανάλυσης δεδομένων.

sobloo

Το sobloo όπως και το CREODIAS βασίζεται κυρίως σε Virtual Machines για τις υπολογιστικές υπηρεσίες και δεν παρέχει web-based IDE όπως το Google Earth Engine. Έχει ωραία, αλλά σχετικά δύσχρηστη διεπαφή για προβολή δεδομένων στον χάρτη και τα datasets είναι σχετικά περιορισμένα σε αριθμό. Γενικά, οι υπηρεσίες που προσφέρει απευθύνονται περισσότερο σε εμπορικές επιχειρήσεις και όχι τόσο σε ερευνητές.

WEkEO

Το WEkEO έχει αρκετά datasets και περιβάλλον για προβολή τους με εύκολο τρόπο επιλογής πολλών layers, αλλά με περιορισμένες δυνατότητες ανάλυσης και όχι τόσο όμορφη διεπαφή. Και αυτό βασίζεται περισσότερο στη χρήση virtual station ή desktop application για τη χρήση κώδικα και περισσότερων δυνατοτήτων. Τα guides που διαθέτει αφορούν κυρίως τη χρήση των VM και όχι την ανάλυση και επεξεργασία των dataset.

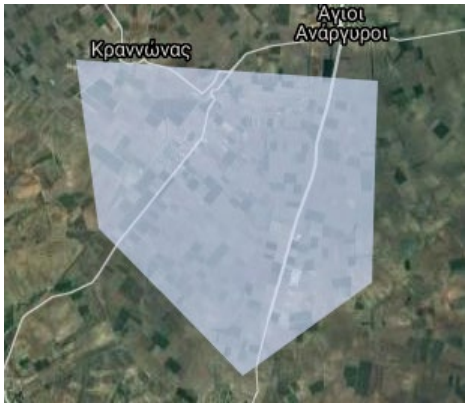
Η επιλογή μας, λοιπόν, για τους σκοπούς της εργασίας με βάση τα παραπάνω είναι το Google Earth Engine.

2. Σύνθετα και NDVI το 2019

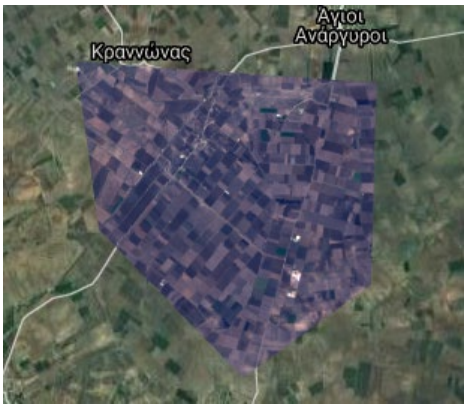
Για τους σκοπούς της μελέτης μας επιλέξαμε μια περιοχή με καλλιέργειες γύρω από τον κάμπο Λαρίσης στην οποία σχεδιάσαμε ένα πολύγωνο (Εικόνα 1). Για το συγκεκριμένο πολύγωνο παίρνουμε τις εικόνες από το δορυφόρο Landsat 8 και τις φιλτράρουμε μόνο για το 2019 και για έως 20% νεφοκάλυψη. Οι εικόνες αυτές είναι 21 και η μικρότερη νεφοκάλυψη είναι 0.03%.

Στην Εικόνα 2 φαίνεται το φυσικό έγχρωμο σύνθετο RGB(432) για το πολύγωνο για τη μέρα με τη μικρότερη νεφοκάλυψη, και στην εικόνα 3 το ψευδέγχρωμο RGB(543). Ο συνδυασμός μπαντών 543 του Landsat 8 κάνει τη βλάστηση να εμφανίζεται ως αποχρώσεις του κόκκινου και όσο πιο σκούρο είναι τόσο πιο πλούσια βλάστηση υπάρχει σε εκείνη την περιοχή.

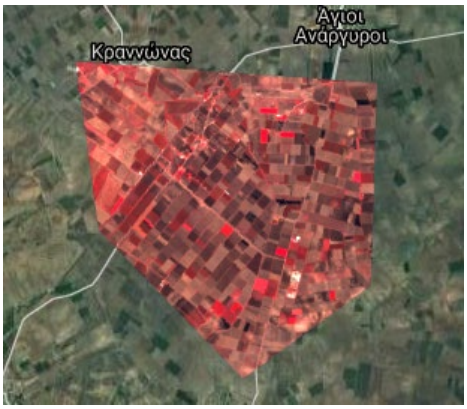
Επίσης, στην εικόνα 4 φαίνεται η απεικόνιση του δείκτη βλάστησης NDVI, όπου πιο έντονο πράσινο αντιστοιχεί σε πιο πλούσια βλάστηση.



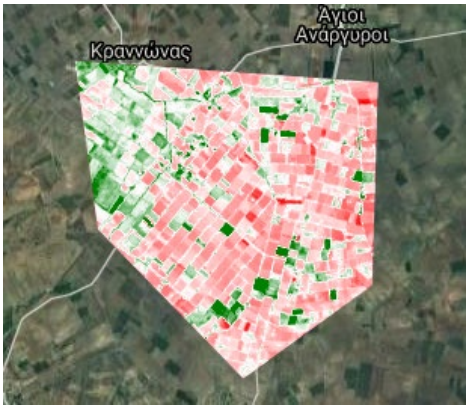
Εικόνα 1: Επιλεγμένο πολύγωνο με καλλιέργειες.



Εικόνα 2: Φυσικό έγχρωμο σύνθετο.



Εικόνα 3: Ψευδέγχρωμο σύνθετο.

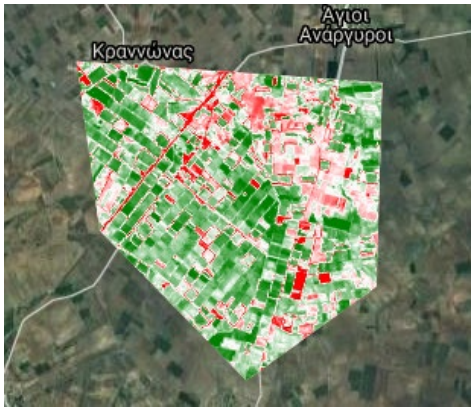


Εικόνα 4: Δείκτης βλάστησης NDVI.

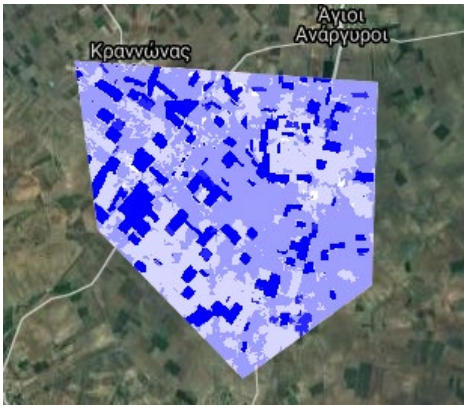
3. Μέγιστες τιμές NDVI το 2018 και 2019

Στην εικόνα 5 φαίνονται οι μέγιστες τιμές NDVI των pixel του πολυγώνου για όλο το 2019 και στην εικόνα 6 οι αντίστοιχες ημερομηνίες (DOY) για τις οποίες είχαμε αυτές τις τιμές. Πιο σκούρο μπλε χρώμα αντιστοιχεί σε μεταγενέστερες ημέρες.

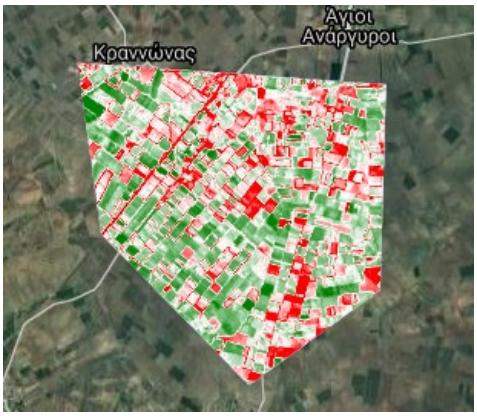
Αντίστοιχα, στις εικόνες 7 και 8 φαίνονται οι τιμές αυτές για το 2018. Παρατηρούμε πως στο 2018 έχουμε μόνο 12 εικόνες διαθέσιμες, οπότε δεν έχουμε μεγάλο εύρος τιμών για καλή οπτικοποίηση.



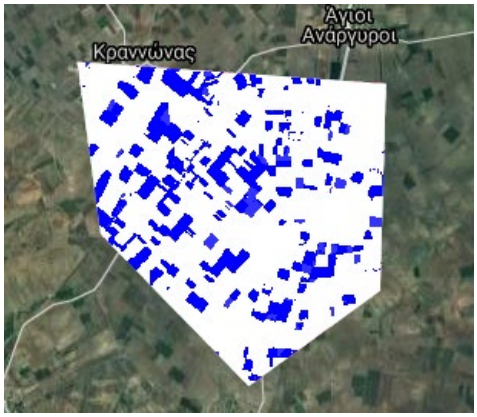
Εικόνα 5: Μέγιστες τιμές NDVI για το 2019.



Εικόνα 6: DOY μεγίστων τιμών NDVI για το 2019.



Εικόνα 7: Μέγιστες τιμές NDVI για το 2018.



Εικόνα 8: DOY μεγίστων τιμών NDVI για το 2018.

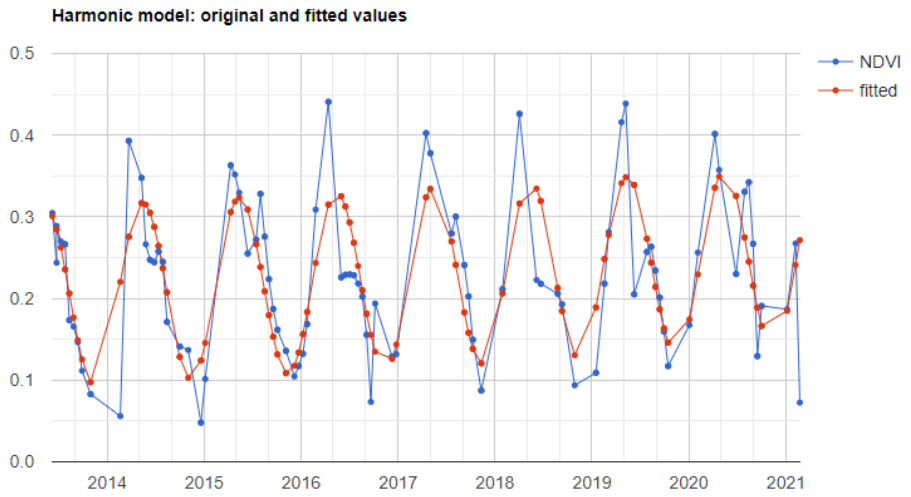
4. Προσαρμογή αρμονικών καμπυλών

Για 4 μικρότερα πολύγωνα στην ίδια περιοχή προσαρμόζουμε αρμονικές καμπύλες στις χρονοσειρές του NDVI δημιουργώντας νέες συνθετικές χρονοσειρές. Για το σκοπό αυτό υπολογίζουμε με γραμμική παλινδρόμηση τους συντελεστές του εξής γραμμικού μοντέλου:

$$NDVI_t = \alpha_0 + \beta_0 t + \sum_{i=1}^{m} a_i \cos(i2\pi\omega t) + \beta_i \sin(i2\pi\omega t)$$

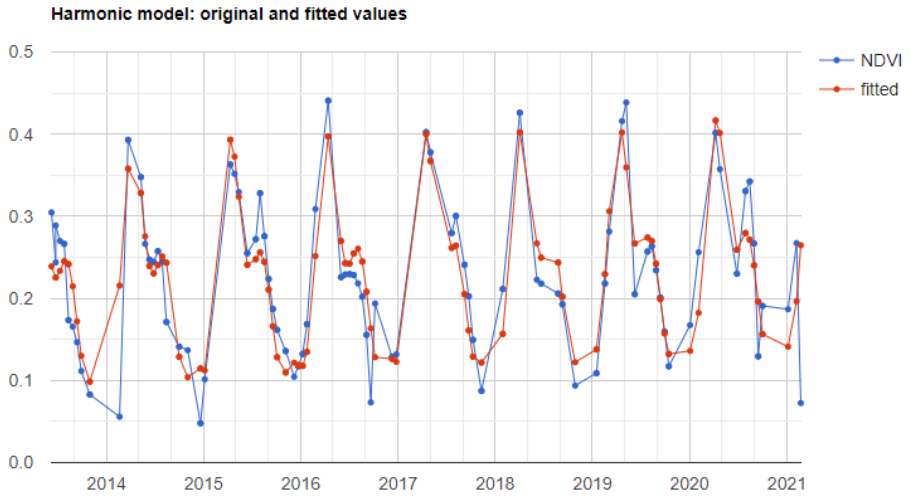
όπου το t είναι το timestamp και m είναι ο βαθμός του πολυωνύμου και αντιστοιχεί στον αριθμό των αρμονικών που θα προσαρμοστούν. Για $\omega = 1$ δοκιμάσαμε πολυώνυμα βαθμών (m) από 1 έως 4 και επιλέξαμε βαθμό 3, καθώς με 4 η βελτίωση προσαρμογής ήταν πολύ μικρή και προτιμούμε το πιο απλό μοντέλο.

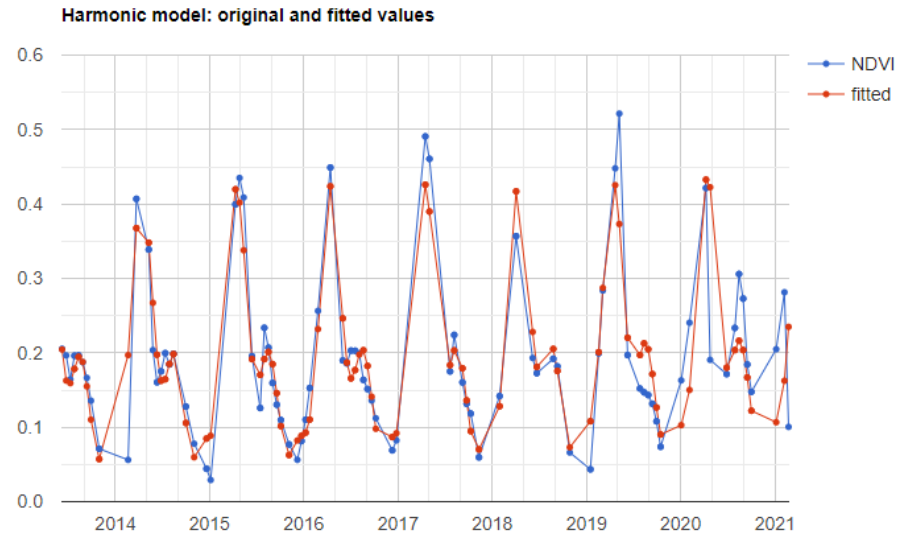
Στην εικόνα 9 φαίνεται ως σημείο αναφοράς η προσαρμογή μιας αρμονικής καμπύλης για το πρώτο πολύγωνο, ενώ στη συνέχεια φαίνονται οι προσαρμογές τριών καμπυλών για κάθε ένα από τα 4 πολύγωνα. Παρατηρούμε πως μια καμπύλη δεν είναι αρκετή, αφού έχει καταφέρει να προσαρμοστεί καλά μόνο στην περίοδο της χρονοσειράς, ενώ με τη χρήση τριών η προσαρμογή είναι αρκετά ικανοποιητική για όλα τα πολύγωνα.

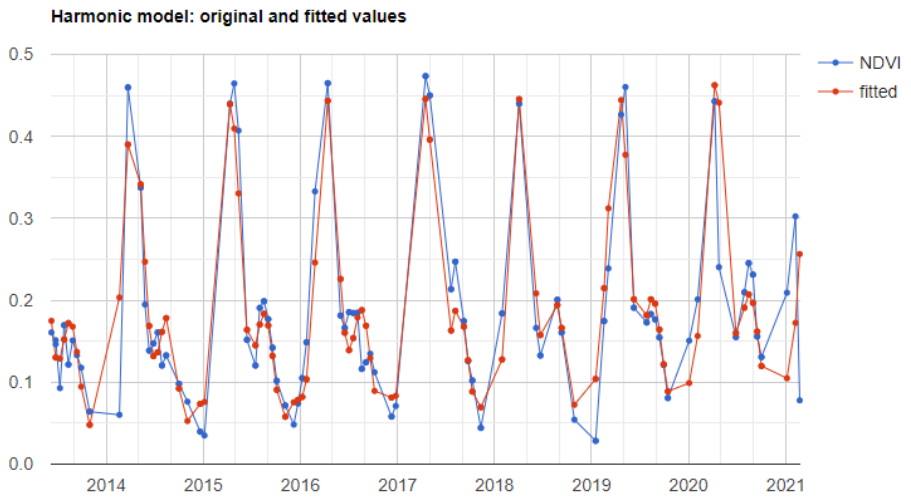


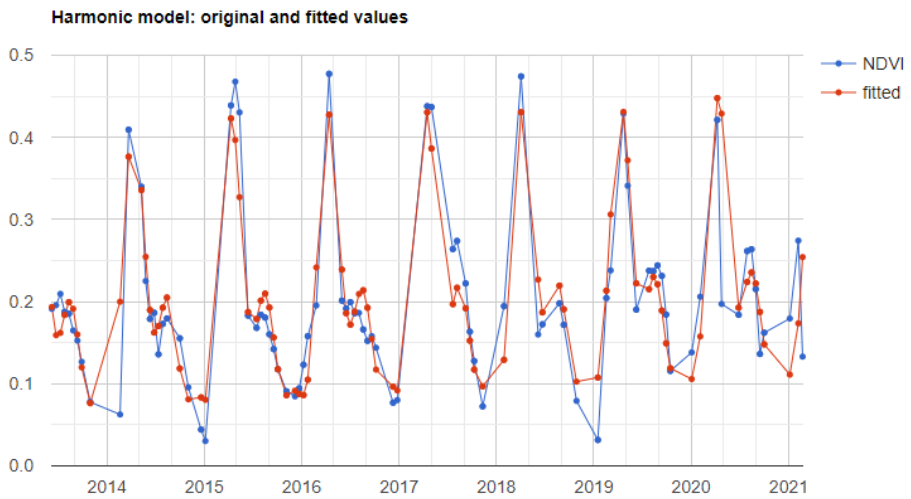
Εικόνα 9: Προσαρμογή μιας αρμονικής καμπύλης στο πρώτο πολύγωνο.

Διαγράμματα προσαρμογής τριών αρμονικών καμπυλών σε κάθε ένα από τα πολύγωνα




Harmonic model: original and fitted values
NDVI
fitted
0.6
0.5
0.4
0.3
0.2
0.1
0.0
2014
2015
2016
2017
2018
2019
2020
2021


Harmonic model: original and fitted values
NDVI
fitted
0.5
0.4
0.3
0.2
0.1
0.0
2014
2015
2016
2017
2018
2019
2020
2021


Harmonic model: original and fitted values
NDVI
fitted
0.5
0.4
0.3
0.2
0.1
0.0
2014
2015
2016
2017
2018
2019
2020
2021

5. Κώδικας

https://code.earthengine.google.com/bff0fbb7e2a8fe78423874843e78e1e9

```
/* Harmonic regression code taken from
https://docs.google.com/presentation/d/1J1rUtf-bkfaJwYJY-
tU17kzKI4U8FnF7Q2_VWqWdaak/edit#slide=id.g1444802fc5_0_1
 and modified to accept trigonometric polynomial order greater than
one */

Map.centerObject(polygon, 12); // focus on our polygon
var timeField = 'system:time_start'; // This field contains UNIX time
in milliseconds.


////////////////////////////////// FUNCTIONS
////////////////////////////////

// Clip image collection given geometry
var clipImages = function(images, geom){
  return images.map(function(image){
    return ee.Image(image).clip(geom)});
};


// Add NDVI band to an image
var addNDVI = function(image) {
  var ndvi = image.normalizedDifference(['B5', 'B4']).rename('NDVI');
  return image.addBands(ndvi);
};


// Add a DOY (day of year) band to an image
var addDOY = function(image){
  var doy = image.date().getRelative('day', 'year'); // this is the
DOY
  doy = ee.Image.constant(doy).uint16().rename('DOY'); // must be
constant
  doy = doy.updateMask(image.select('B8').mask()); // needed for
clipping image
  return image.addBands(doy);
};


// Use this function to add variables for NDVI, time and a constant
// to Landsat 8 imagery.
var addVariables = function(image) {
  var date = ee.Date(image.get(timeField));  // Compute time in
fractional years since the epoch.
  var years = date.difference(ee.Date('1970-01-01'), 'year');
  return image  // Return the image with the added bands.
    .addBands(image.normalizedDifference(['B5',
'B4']).rename('NDVI')).float()  // Add an NDVI band.
    .addBands(ee.Image(years).rename('t').float())  // Add a time
band.
    .addBands(ee.Image.constant(1));  // Add a constant band.
};
```

```
// harmonic analysis of NDVI time series for a given geometry and
polynomial order
var harmonicRegression = function(geometry, order) {
  var dependent = ee.String('NDVI');  // Name of the dependent
variable.
  var harmonicIndependents = ee.List(['constant', 't']);  // Use
these independent variables in the harmonic regression.
  // add cos and sin variables according to polynomial order
  var i;
  for (i = 1; i <= order; i++) {
    harmonicIndependents = harmonicIndependents.add('cos' +
i).add('sin' + i);
  }

  // Add harmonic terms as new image bands.
  var harmonicLandsat = landsat_all.map(function(image) {
    var i;
    for (i = 1; i <= order; i++) {
      var timeRadians = image.select('t').multiply(i * 2 * Math.PI);
      image = image.addBands(timeRadians.cos().rename('cos' + i))
          .addBands(timeRadians.sin().rename('sin' + i));
    }
    return image;
  });

  // The output of the regression reduction
  var harmonicTrend = harmonicLandsat
    .select(harmonicIndependents.add(dependent))

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

  // 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'));
  });
  return fittedHarmonic;  // return the fitted harmonic model
};


// plot NDVI series and fitted values of a given harmonic model at a
given geometry
var plotSeries = function(model, geometry) {
  print(ui.Chart.image.series(
    model.select(['fitted','NDVI']), geometry, ee.Reducer.mean(), 30)
      .setSeriesNames(['NDVI', 'fitted'])
      .setOptions({
        title: 'Harmonic model: original and fitted values',
        lineWidth: 1,
        pointSize: 3,
```

```
  }));
};


//////////////////////////////// DATA
////////////////////////////////

// 2019 collection for the selected polygon
var landsat2019 = ee.ImageCollection(landsat8)
  .filterBounds(polygon) // filter location
  .filterDate('2019-01-01', '2019-12-31') // filter date (2019)
  .filter(ee.Filter.lte('CLOUD_COVER', 20)); // keep images with max
20% cloud cover
landsat2019 = clipImages(landsat2019, polygon); // clip all images in
collection

// 2018 collection for the selected polygon
var landsat2018 = ee.ImageCollection(landsat8)
  .filterBounds(polygon) // filter location
  .filterDate('2018-01-01', '2018-12-31')
  .filter(ee.Filter.lte('CLOUD_COVER', 20));
landsat2018 = clipImages(landsat2018, polygon);

// all time data for the selected polygon
var landsat_all = ee.ImageCollection(landsat8)
  .filterBounds(polygon) // filter location
  .filter(ee.Filter.lte('CLOUD_COVER', 20));


//////////////////////////////// Question 2
////////////////////////////////

print('Collection 2019: ', landsat2019); // print collection

// sort by increasing cloudiness and get the less cloudy (the first
one)
var less_cloudy_2019 = landsat2019.sort('CLOUD_COVER').first();
print('Less cloudy image: ', less_cloudy_2019);

// show true color composite
Map.addLayer(less_cloudy_2019, {bands: ['B4', 'B3', 'B2'], min:6500,
max:12500}, 'true color composite');

// show false color composite
Map.addLayer(less_cloudy_2019, {bands: ['B5', 'B4', 'B3'], min:6500,
max:12500}, 'false color composite');

// compute and show NDVI
var ndvi_image = less_cloudy_2019.normalizedDifference(['B5', 'B4']);
Map.addLayer(ndvi_image, {min:0, max:0.2, palette: ['red', 'white',
'green']}, 'NDVI');


//////////////////////////////// Question 3
////////////////////////////////

// add NDVI and DOY band to every image of 2019 collection
var landsat2019NDVI = landsat2019.map(addNDVI).map(addDOY);

// calculate max NDVI for every pixel of the series for 2019
```

```
var maxNDVI = landsat2019NDVI.select(['NDVI',
'DOY']).reduce(ee.Reducer.max(2)); // max input is 2 in order to
ouput the corresponding values of DOY
print('Max NDVI Image: ', maxNDVI);

// add max NDVI layer
Map.addLayer(maxNDVI.select('max'), {min:0.3, max:0.6, palette:
['red', 'white', 'green']}, 'Max NDVI 2019');

// add DOY of max NDVI
Map.addLayer(maxNDVI.select('max1'), {min:100, max:170, palette:
['white', 'blue']}, 'DOY of max NDVI 2019');

// same work for the 2018 collection
print('Collection 2018: ', landsat2018);
var landsat2018NDVI = landsat2018.map(addNDVI).map(addDOY);
var maxNDVI = landsat2018NDVI.select(['NDVI',
'DOY']).reduce(ee.Reducer.max(2));
Map.addLayer(maxNDVI.select('max'), {min:0.3, max:0.6, palette:
['red', 'white', 'green']}, 'Max NDVI 2018');
Map.addLayer(maxNDVI.select('max1'), {min:100, max:170, palette:
['white', 'blue']}, 'DOY of max NDVI 2018');


//////////////////////////////// Question 4
////////////////////////////////

// add NDVI, time and constant to images
landsat_all = landsat_all.map(addVariables);

// fit harmonic regression models for the four polygons
var order = 3; // polynomial order

var fitted_harmonic1 = harmonicRegression(poly1, order);
var fitted_harmonic2 = harmonicRegression(poly2, order);
var fitted_harmonic3 = harmonicRegression(poly3, order);
var fitted_harmonic4 = harmonicRegression(poly4, order);

// plot NDVI time series and fitted harmonics for the four polygons
plotSeries(fitted_harmonic1, poly1);
plotSeries(fitted_harmonic2, poly2);
plotSeries(fitted_harmonic3, poly3);
plotSeries(fitted_harmonic4, poly4);
```