Case Study 2 (Part 2) - County-level Time-series NDVI

1 Code overview

You need to download the Case Study 2(Part 2).docx code file. Once you've uploaded and imported the shapefile as Case Study 2(Part 1), launch the code editor in your browser, copy and paste the code to your own code editor, and run the code.

2 Procedures

2.1 Part 1 code

We need all the code from part 1 except for the "Clip and visualize CDL and MODIS in Wisconsin" section.

2.2 Calculate NDVI in masked MODIS

Code block 1:

```
/**
* Calculate NDVI
*/
var getNDVI = function(image) {
  var ndvi = image.expression(
    '(nir - red) / (nir + red)', {
    nir: image.select([1]).float(),
    red: image.select([0]).float()
  });
  // Keep the value within the range of (-1, 1)
  return ndvi.updateMask(ndvi.gt(-1)).updateMask(ndvi.lt(1));
};

// Calcualte VI
var calmodVI = maskmodVI.map(getNDVI)
```

ee.Image.expression: doc link

```
Image.expression(expression, map)
'''
Evaluates an arithmetic expression on an image, possibly involving additional images.
The bands of the primary input image are available using the built-in function b(), as b(0) or b('band_name').

Variables in the expression are interpreted as additional image parameters which must be supplied in opt_map. The bands of
    each such image can be accessed like image.band_name or image[0].

Both b() and image[] allow multiple arguments, to specify multiple bands, such as b(1, 'name', 3). Calling b() with no
    arguments, or using a variable by itself, returns all bands of the image.

If the result of an expression is a single band, it can be assigned a name using the '=' operator (e.g.: x = a + b).

Returns the image computed by the provided expression.
```

```
Args:
  this.image: Image
  expression: String
  map: Dictionary, optional
'''
```

2.3 Add band name

```
/**
* Function to add "DOY" as bandname
*/
var addBandname = function(img) {
  // Get string-format date
  var datestring = ee.String(img.get('system:index'));
  var format = 'YYYY_MM_dd';
  // Parse the string-format date
  var eedate = ee.Date.parse(format, datestring);
  // Get day
  var doy = eedate.getRelative('day', 'year').add(1);
  // Get year
  var year = eedate.get('year');
  // Add day and year to band
  var bandname = ee.String(year).cat('_').cat(doy);
  return img.set('bandname',bandname);
};

calmodVI = calmodVI.map(addBandname);
```

ee.Date.parse: doc link

```
ee.Date.parse(format, date, timeZone)
'''

Args:
  format: String
  date: String
  timeZone: String, default: null
'''
```

ee.Date.getRelative: doc link

```
Date.getRelative(unit, inUnit, timeZone)
'''
Returns the specified (0-based) unit of this date relative to a larger unit, e.g. getRelative('day', 'year') returns a
    value between 0 and 365.

Args:
  this.date: Date
  unit: String
  inUnit: String
  timeZone: String, default: null
'''
```

2.4 Convert Imagecollection to an image

Code block 2:

```
/**
```

```
* stackCollection(collection):
* convert a collection to a single image with bands representing each image in the collection
* Note: the band will be in reverse order
*/

var stackCollection = function(collection) {
  return ee.Image(collection.iterate(appendBand));
};

var modVIStack = stackCollection(calmodVI);
```

ee.ImageCollection.iterate: doc link

```
ImageCollection.iterate(algorithm, first)
'''
Applies a user-supplied function to each element of a collection. The user-supplied function is given two arguments: the
    current element, and the value returned by the previous call to iterate() or the first argument, for the first
    iteration. The result is the value returned by the final call to the user-supplied function.
Returns the result of the Collection.iterate() call.

Args:
  this.collection: Collection
  algorithm: Function
  first: Object, optional
'''
```

2.5 Reduce NDVI to 500m scale

Code block 3:

```
var meanNDVI = modVIStack.reduceRegions(counties, ee.Reducer.mean(), 500);
```

ee.Image.reduceRegions: doc link

```
Image.reduceRegions(collection, reducer, scale, crs, crsTransform, tileScale)
'''
Apply a reducer over the area of each feature in the given collection.
The reducer must have the same number of inputs as the input image has bands.

Returns the input features, each augmented with the corresponding reducer outputs.

Args:
  this.image: Image
  collection: FeatureCollection
  reducer: Reducer
  scale: Float, default: null
  crs: Projection, default: null
  crsTransform: List, default: null
  tileScale: Float, default: 1
'''
```

ee.Reducer.mean: doc link

```
ee.Reducer.mean
'''
Returns a Reducer that computes the (weighted) arithmetic mean of its inputs.
'''
```

2.6 Export data

Code block 4:

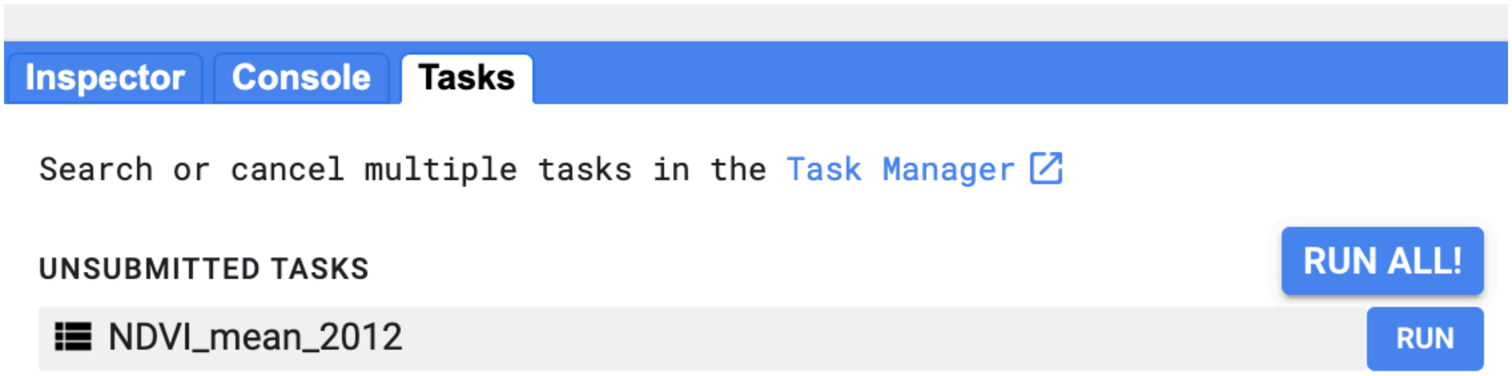
```
/**
* Export ************
*/
var exportVI = function(table, prefix) {
  Export.table.toDrive({
    collection: table.select([".*"], null, false), // The exported data is a list [ndvi], we only need the numbers inside.
    description: prefix,
    folder: 'test',
    fileNamePrefix: prefix
  });
};

exportVI(meanNDVI, 'NDVI_mean_' + year);
```

2.6.1 Tutorial: Exporting Table and Vector Data

2.6.2 Visualization in web

You can see a button ”RUN” if your code runs correctly (Fig. 1).



Search or cancel multiple tasks in the Task Manager

UNSUBMITTED TASKS

NDVI_mean_2012

Fig. 1

2.7 Use for-loop to export the data

link

This is the link to the complete code for downloading data from 2007-2022 using a for loop. Once you’ve uploaded and imported the shapefile, you can click on this link, copy the code to your own code editor in browser, and execute the code.

3 To do task

3.1 Calculate and Export Enhanced Vegetation Index (EVI) Data for a Selected Region in Google Earth Engine

modify the Sec. 2.2 code to calculate the Enhanced Vegetation Index (EVI) for a selected region and time period. After calculating the EVI, export the data to your Google Drive for further analysis. Instead of using the way we calculate NDVI in Case Study 1, you need to use ”ee.Image.expression” to calculate evi now.

Hint: The formula for EVI is:

$EVI = 2.5 \times \frac{(NIR - Red)}{(NIR + 6 \times Red - 7.5 \times Blue + 1)}$