GEOG683: Advanced GIS
Jeff Yen
821274604

Vegetation Dynamics from NDVI Time Series Analysis at Carmel Valley

1. Project Motivation and Objectives

Remote sensing (RS) imagery has been most commonly used to generate several land use and land cover (LULC) maps and to conduct LULC change analysis because of the ability to observe landscape across regions in a synoptic and timely. Since vegetation is the majority of land covers on earth, understanding vegetation dynamics is important to promote our lives in many perspectives (e.g., urban planning and management).

The normalized difference vegetation index (NDVI) is the most predominant vegetation indices for monitoring vegetation dynamics. However, previous research on vegetation dynamics primarily emphasized the area of vegetation change or vegetation NDVI value change. Therefore, this pilot study aims to provide more comprehensive aspects of vegetation dynamics based on three indicators:

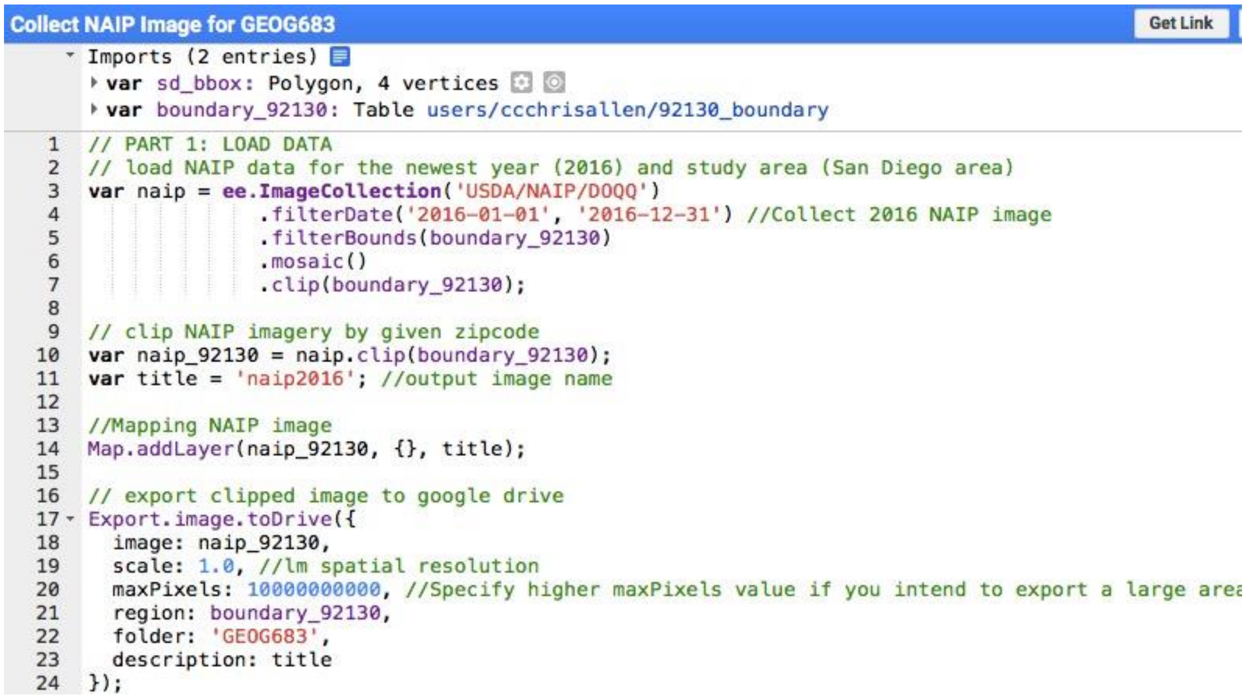
- Vegetation NDVI value change
- Speed of vegetation NDVI value change
- Acceleration of vegetation NDVI value change

Another merit of this study is to automate the entire image processing routines of vegetation dynamics via Python 2.7 and ArcPy. Hopefully, the python script used in this project can be partially reused for other research purposes, specifically in the part of I/O (read/write) image information.

2. Data and Study Area

Four images (2010, 2012, 2014, 2016) were collected from National Agriculture Imagery Program (NAIP) using Google Earth Engine, scripted in JavaScript. The collected NAIP images have the following characteristics:

- Spatial resolution: 1 m
- Spectral resolution: R, G, B, NIR
- Radiometric resolution: 8-bit unsigned (0 – 255)



```
Imports (2 entries)
  var sd_bbox: Polygon, 4 vertices
  var boundary_92130: Table users/ccchrisallen/92130_boundary
```

```javascript
// PART 1: LOAD DATA
// load NAIP data for the newest year (2016) and study area (San Diego area)
var naip = ee.ImageCollection('USDA/NAIP/DOQQ')
              .filterDate('2016-01-01', '2016-12-31') //Collect 2016 NAIP image
              .filterBounds(boundary_92130)
              .mosaic()
              .clip(boundary_92130);

// clip NAIP imagery by given zipcode
var naip_92130 = naip.clip(boundary_92130);
var title = 'naip2016'; //output image name

//Mapping NAIP image
Map.addLayer(naip_92130, {}, title);

// export clipped image to google drive
Export.image.toDrive({
  image: naip_92130,
  scale: 1.0, //1m spatial resolution
  maxPixels: 10000000000, //Specify higher maxPixels value if you intend to export a large are
  region: boundary_92130,
  folder: 'GEOG683',
  description: title
});
```

Figure 1. Collecting NAIP images through code editor of Google Earth Engine

This project chose zip code 92130, Carmel Valley (Figure 2), as the study area where urbanization has contributed to substantial change of vegetation coverage in 6 years (2010 – 2016).

3. Methods

The entire image processing routine is divided into three parts:

- Reading image information from .csv file

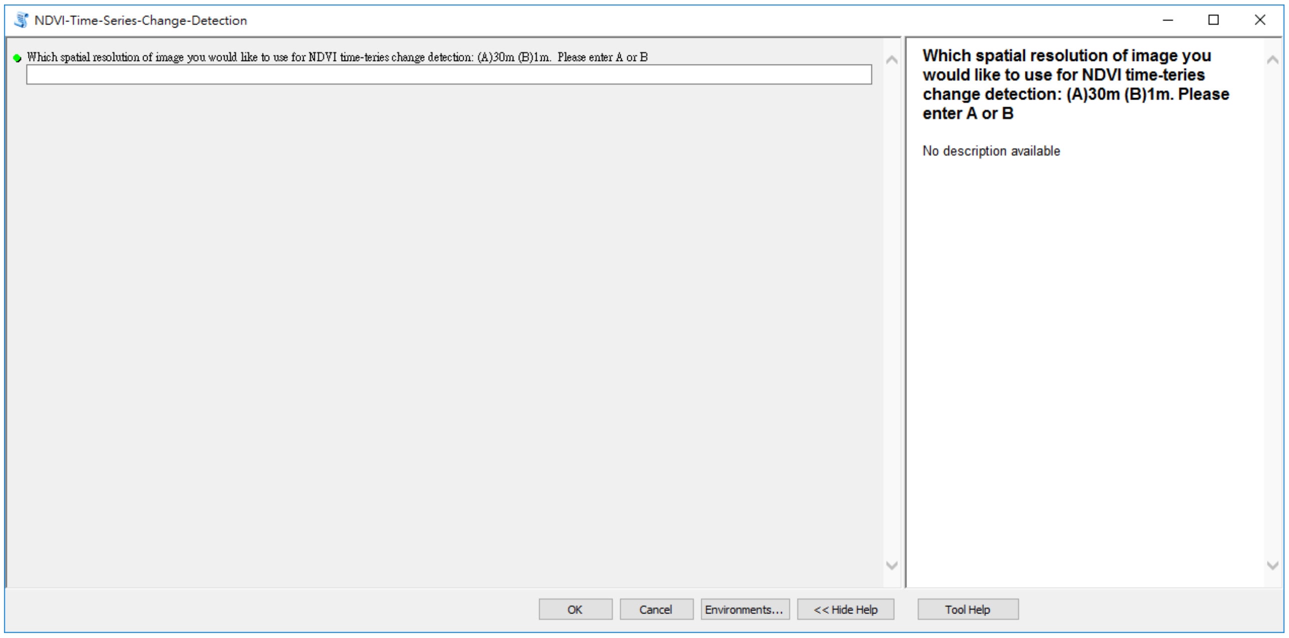
- Classifying vegetation
- Performing vegetation dynamics based on NDVI time-series change detection

The following ArcPy methods were adopted to develop each part of image processing routine of this project:

- arcpy.GetParameterAsText
- arcpy.env.workspace
- arcpy.CheckOutExtension
- arcpy.AddMessage
- arcpy.Raster
- arcpy.GetRasterProperties_management
- arcpy.sa (e.g., Int, Float, SetNull, RemapRange and Reclassify)

3.1. Reading Image Information From CSV File

In the beginning, users need to specify image absolute path, time stamp, and NDVI threshold for each image input in csv file. This design is time-efficient for users since no need to add each image input on ArcMap every time. Moreover, this part of code can be easily customized and reused to read image information for different research purposes. In this project, there are two options of image input for users: either 30m or 1m NAIP images (Figure2). Performing four 1m resolution images for vegetation dynamics generally takes 20 minutes. To quickly look at vegetation dynamics, please select 30m resolution images.


NDVI-Time-Series-Change-Detection

Which spatial resolution of image you would like to use for NDVI time-teries change detection: (A)30m (B)1m. Please enter A or B

Which spatial resolution of image you would like to use for NDVI time-teries change detection: (A)30m (B)1m. Please enter A or B

No description available

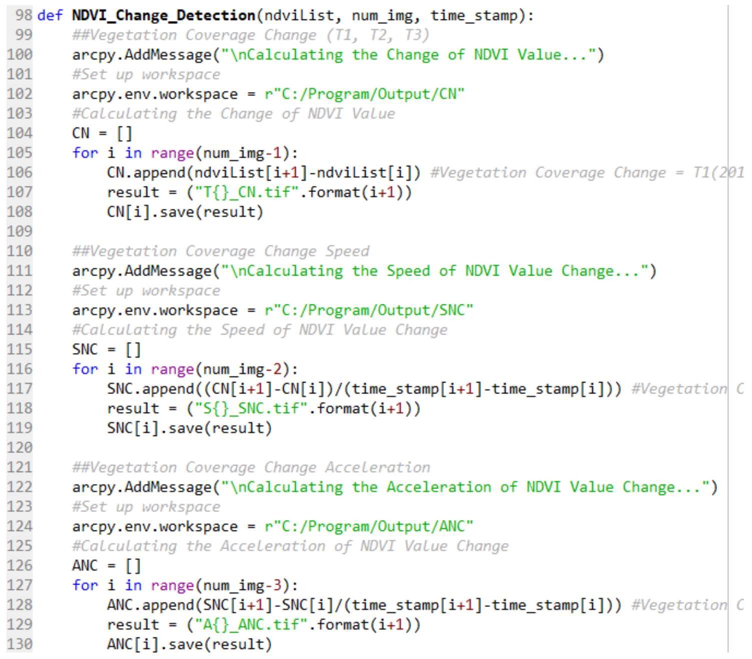
Figure2. A user interface of NDVI time-series change detection tool.

3.2 Classifying Vegetation

Here I created two functions, **NDVI_Calculation** and **Vegetation_Classification**, to iteratively classify multi-temporal vegetation coverage from NDVI time-series. The first function (Figure 3) will generate and output NDVI time-series based on the following algorism, modified from (Jensen, 2017):

$$\left[\left(\frac{NIR - R}{NIR + R}\right) \times 100\right] + 100$$

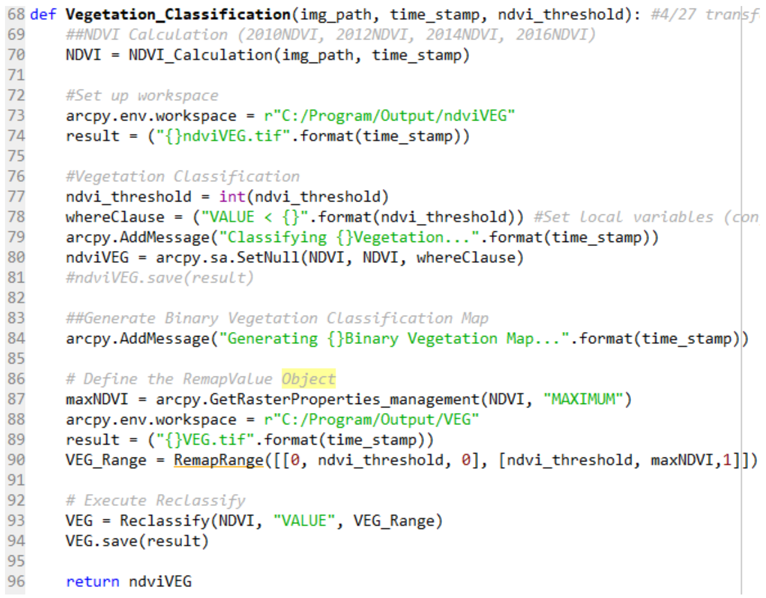
This equation stretches NDVI value from 0 to 200 that avoids negative NDVI value and promotes the efficiency of NDVI threshold interpretation. Next, I visually interpreted the NDVI layers to find the most appropriate NDVI value as the threshold for vegetation classifications. The thresholds will be manually entered on csv file. For example, if the threshold is 122, NDVI value over 122 will be classified as vegetation; in the opposite, it will be classified as non-vegetation.



```python
def NDVI_Change_Detection(ndviList, num_img, time_stamp):
    ##Vegetation Coverage Change (T1, T2, T3)
    arcpy.AddMessage("\nCalculating the Change of NDVI Value...")
    #Set up workspace
    arcpy.env.workspace = r"C:/Program/Output/CN"
    #Calculating the Change of NDVI Value
    CN = []
    for i in range(num_img-1):
        CN.append(ndviList[i+1]-ndviList[i]) #Vegetation Coverage Change = T1(201
        result = ("T{}_CN.tif".format(i+1))
        CN[i].save(result)

    ##Vegetation Coverage Change Speed
    arcpy.AddMessage("\nCalculating the Speed of NDVI Value Change...")
    #Set up workspace
    arcpy.env.workspace = r"C:/Program/Output/SNC"
    #Calculating the Speed of NDVI Value Change
    SNC = []
    for i in range(num_img-2):
        SNC.append((CN[i+1]-CN[i])/(time_stamp[i+1]-time_stamp[i])) #Vegetation C
        result = ("S{}_SNC.tif".format(i+1))
        SNC[i].save(result)

    ##Vegetation Coverage Change Acceleration
    arcpy.AddMessage("\nCalculating the Acceleration of NDVI Value Change...")
    #Set up workspace
    arcpy.env.workspace = r"C:/Program/Output/ANC"
    #Calculating the Acceleration of NDVI Value Change
    ANC = []
    for i in range(num_img-3):
        ANC.append(SNC[i+1]-SNC[i]/(time_stamp[i+1]-time_stamp[i])) #Vegetation C
        result = ("A{}_ANC.tif".format(i+1))
        ANC[i].save(result)
```

Figure 3. Definition of NDVI_Calculation function

The second function (Figure 4) performs the classification based on this rule. The classified NDVI, called ndviVEG, will be returned and appended into a list, named ndviList, for iterative vegetation dynamics calculation. To visually interpret the change of vegetation coverage over time, binary vegetation classifications (0: non-veg; 1: veg) will be generated by reclassifying ndviVEG layers.



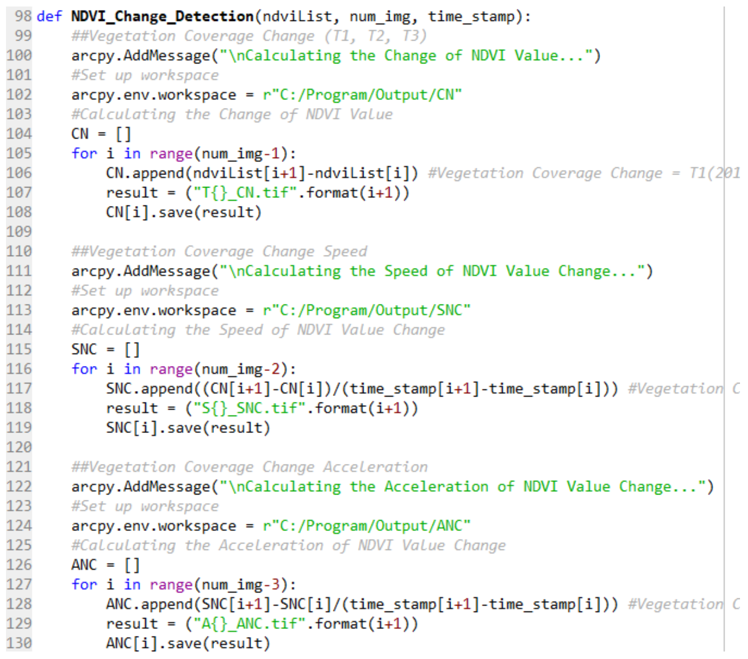
```python
def Vegetation_Classification(img_path, time_stamp, ndvi_threshold): #4/27 transf
    ##NDVI Calculation (2010NDVI, 2012NDVI, 2014NDVI, 2016NDVI)
    NDVI = NDVI_Calculation(img_path, time_stamp)

    #Set up workspace
    arcpy.env.workspace = r"C:/Program/Output/ndviVEG"
    result = ("{}ndviVEG.tif".format(time_stamp))

    #Vegetation Classification
    ndvi_threshold = int(ndvi_threshold)
    whereClause = ("VALUE < {}".format(ndvi_threshold)) #Set local variables (con
    arcpy.AddMessage("Classifying {}Vegetation...".format(time_stamp))
    ndviVEG = arcpy.sa.SetNull(NDVI, NDVI, whereClause)
    #ndviVEG.save(result)

    ##Generate Binary Vegetation Classification Map
    arcpy.AddMessage("Generating {}Binary Vegetation Map...".format(time_stamp))

    # Define the RemapValue Object
    maxNDVI = arcpy.GetRasterProperties_management(NDVI, "MAXIMUM")
    arcpy.env.workspace = r"C:/Program/Output/VEG"
    result = ("{}VEG.tif".format(time_stamp))
    VEG_Range = RemapRange([[0, ndvi_threshold, 0], [ndvi_threshold, maxNDVI,1]])

    # Execute Reclassify
    VEG = Reclassify(NDVI, "VALUE", VEG_Range)
    VEG.save(result)

    return ndviVEG
```

Figure 4. Definition of Vegetation_Classification function

3.3 Performing Vegetation Dynamics Based on NDVI Time-Series Change Detection

This study defined three functions to iteratively generate three indicators for vegetation dynamics (Figure 5):

- Vegetation NDVI value change (CN): T1(2012-2010), T2(2014-2012), T3(2016-2014)
- Speed of vegetation NDVI value change (SNC): S1:(T2-T1)/2, S2:(T3-T2)/2
- Acceleration of vegetation NDVI value change (ANC): A1: (S2-S1)/2



```
def NDVI_Change_Detection(ndviList, num_img, time_stamp):
    ##Vegetation Coverage Change (T1, T2, T3)
    arcpy.AddMessage("\nCalculating the Change of NDVI Value...")
    #Set up workspace
    arcpy.env.workspace = r"C:/Program/Output/CN"
    #Calculating the Change of NDVI Value
    CN = []
    for i in range(num_img-1):
        CN.append(ndviList[i+1]-ndviList[i]) #Vegetation Coverage Change = T1(201
        result = ("T{}_CN.tif".format(i+1))
        CN[i].save(result)

    ##Vegetation Coverage Change Speed
    arcpy.AddMessage("\nCalculating the Speed of NDVI Value Change...")
    #Set up workspace
    arcpy.env.workspace = r"C:/Program/Output/SNC"
    #Calculating the Speed of NDVI Value Change
    SNC = []
    for i in range(num_img-2):
        SNC.append((CN[i+1]-CN[i])/(time_stamp[i+1]-time_stamp[i])) #Vegetation C
        result = ("S{}_SNC.tif".format(i+1))
        SNC[i].save(result)

    ##Vegetation Coverage Change Acceleration
    arcpy.AddMessage("\nCalculating the Acceleration of NDVI Value Change...")
    #Set up workspace
    arcpy.env.workspace = r"C:/Program/Output/ANC"
    #Calculating the Acceleration of NDVI Value Change
    ANC = []
    for i in range(num_img-3):
        ANC.append(SNC[i+1]-SNC[i]/(time_stamp[i+1]-time_stamp[i])) #Vegetation C
        result = ("A{}_ANC.tif".format(i+1))
        ANC[i].save(result)
```

Figure 5. Definition of NDVI_Change_Detection function

4. Results (Take 1m NAIP as an example)

4.1. Binary Vegetation Classifications

According to Table 1, vegetation coverage and percentage were decreasing from 2010 to 2016 at Carmel Valley. The following maps (Figure 6) show the spatial distribution of vegetation coverage over time. The east-southern Carmel Valley might have the large amount decrease of vegetation coverage.

Table 1. The change of vegetation coverage and percentage from 2010 to 2016

Year	**Vegetation Coverage (m^2)**	**Percentage of Vegetation Coverage (%)**
2010	45423760	16.1
2012	32251308	11.4
2014	20784488	7.4
2016	14123916	5.0

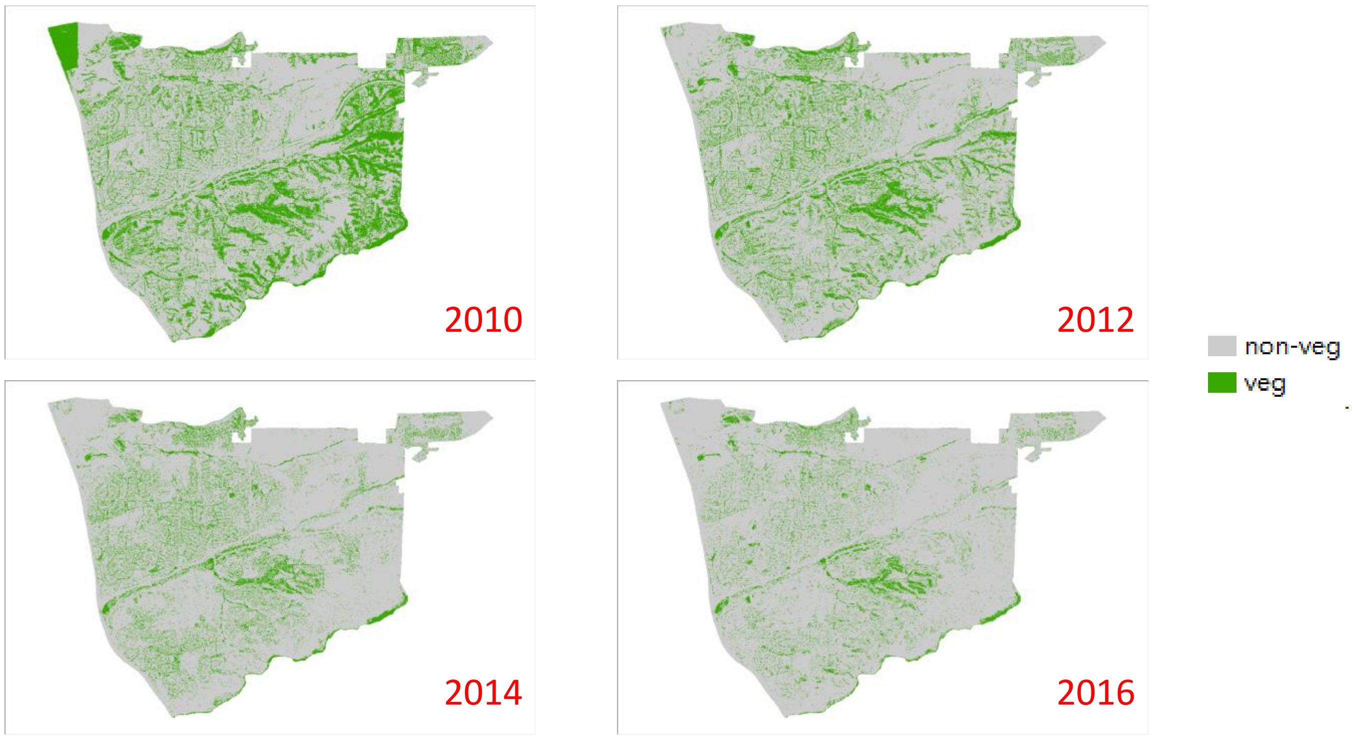


Figure 6. Binary vegetation classifications from 2010 to 2016

4.2. The Change of Vegetation NDVI Value

Table 2 indicates the average vegetation NDVI value is decreasing since 2010. The standard deviation vegetation NDVI value slightly increased in period of 2010 to 2014 but decreased from 2012 to 2016 eventually. Similarly, the east-southern Carmel Valley might have the large amount decrease of vegetation NDVI value change (Figure 7).

Table 2. The change of vegetation NDVI value from 2010 to 2016

Time	Avg. VEG_NDVI Value Change	Std. VEG_NDVI Value Change
T1 (2012 - 2010)	13.55	10.68
T2 (2014 - 2012)	7.54	11.73
T3 (2016 - 2014)	-20.08	9.86

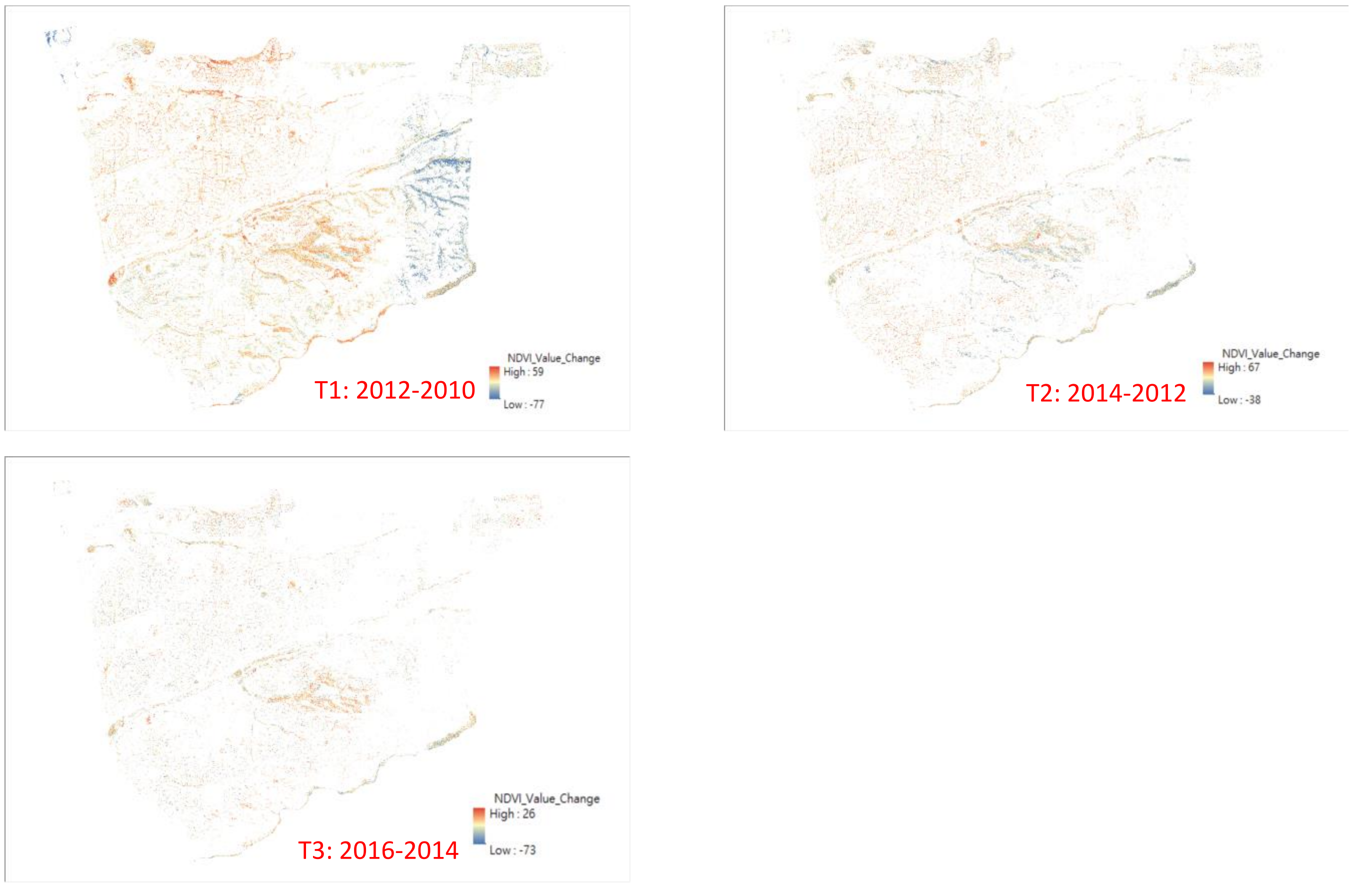


Figure 7. The change of vegetation NDVI Value from 2010 to 2016

4.3. The Speed of Vegetation NDVI Value Change

Apparently, Table 2 shows the identical temporal trend of vegetation dynamics with Table 1(The change of vegetation coverage and percentage). The average speed of vegetation NDVI value change is slow down but the standard deviation increases. However, it is not easy to tell the spatial distribution of the speed change from Figure 8 because of per-pixel-based classification and very small minimum mapping unit (1m).

Table 3. The speed of vegetation NDVI value from 2010 to 2016

Time	**Avg. Speed of VEG_NDVI Value Change**	**Std. Speed of VEG_NDVI Value Change**
S1:(T2-T1)/2	-5.73	8.93
S2:(T3-T2)/2	-13.41	9.69

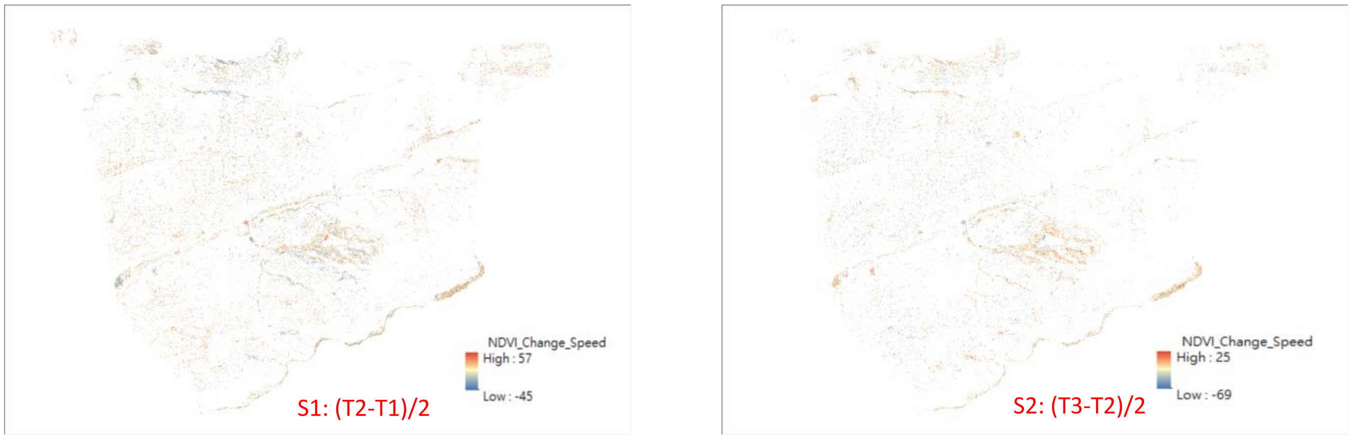


Figure 8. The speed of vegetation NDVI value from 2010 to 2016

4.4. The Acceleration of Vegetation NDVI Value Change

Table 4 demonstrates the average acceleration of vegetation NDVI value is -8.94 and the standard deviation is 13.03. Likewise, it is not easy to tell the spatial distribution of the speed change from Figure 9 because of per-pixel-based classification and very small minimum mapping unit (1m). However, the maximum and minimum acceleration is 44 and -83, respectively.

Table 4. The acceleration of vegetation NDVI value from 2010 to 2016

Time	**Avg. Acceleration of VEG_NDVI Value Change**	**Std. Acceleration of VEG_NDVI Value Change**
A1:(S2-S1)/2	-8.94	13.03

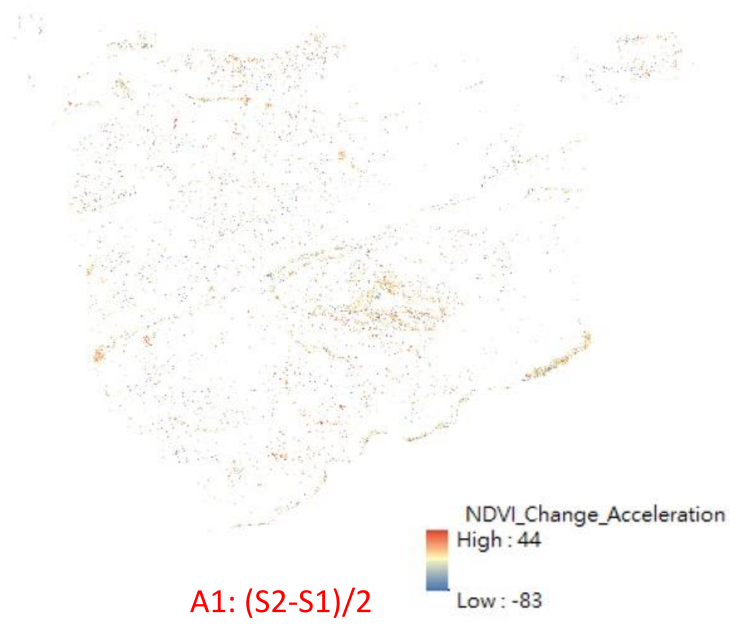


Figure 9. The acceleration of vegetation NDVI value from 2010 to 2016

5. Technical Limitations, Challenges and Problems

Subjective NDVI threshold selection is the primary limitation of this project. Namely, it affects the results of vegetation dynamics based on NDVI time-series. The major technical challenge of this project is the difficulty in appropriately visualizing the results, particularly the speed and acceleration of vegetation NDVI value change. I tried to use brighter color and darker background to highlight the pixels, but it did not work. Grouping pixels into a larger spatial unit for visualization purpose might be worth to try.

6. Lessons Learned

The major lesson I learned from this class is how to use ArcPy to develop various geospatial projects. Specifically, the lab assignments are really helpful exercises. Most of the functions I used in here have been introduced in the class or the lab assignments. I am glad that I am able to automate geospatial workflow through coding Python.

7. Reference

Jensen, J. R. (2017). *Introductory Digital Image Processing: A Remote Sensing Perspective* (4th ed.): Prentice Hall Press.