Department of Geography

CCST9083 Earth as Seen by Satellite

**Laboratory X: Quantifying the health and density of vegetation - NDVI**

In this lab exercise, you will use the web-based code editor to explore landscape changes over time using the Normalized Difference Vegetation Index (NDVI). To access the Code Editor, simply browse for <https://code.earthengine.google.com/>. Satellite images from Landsat 8 *(USGS Landsat 8 Collection 2 Tier 1 TOA Reflectance)* will be used for this study.

The NDVI is a vegetation index that uses the reflectance properties of healthy vegetation to distinguish vegetated areas from non-vegetated areas (e.g., urbanized areas), as well as estimate how healthy the present vegetation is. This is done by calculating the normalized difference of the Near Infrared (NIR) and the Red (R) band, as healthy vegetation highly reflects Near Infrared light, but barely reflects Red light:

The result is a range of values from -1 to 1, with high positive values indicating an abundance of healthy vegetation, values around zero low to none vegetation and high negative values most likely water.

1. **Area of interest**

Our study focuses on one of China's major cities – Shenzhen, and access our boundary-defining geometry using the GAUL-Level2 layer

|  |
| --- |
| var roi = ee.FeatureCollection("FAO/GAUL/2015/level2")  .filter(ee.Filter.eq('ADM0\_NAME', 'China'))  .filter(ee.Filter.eq('ADM2\_NAME', 'Shenzhen'))  Map.addLayer(roi, {color: 'green', width: 1}, 'ROI');  Map.centerObject(roi,10) |

1. **Filter the Landsat 8 Data by Area & Time**

Load USGS Landsat 8 Collection through the code below:

|  |
| --- |
| //Period: 2014-2017  var l8\_20142017=ee.ImageCollection('LANDSAT/LC08/C01/T1\_TOA')  .filterDate('2014-01-01','2017-12-31')  .filter(ee.Filter.lt('CLOUD\_COVER', 15))  .filterBounds(roi);  var l8\_20142017\_median=l8\_20142017.median()  .clip(roi);  print(l8\_20142017)  //Period: 2018-2021  var l8\_20182021 = ee.ImageCollection('LANDSAT/LC08/C01/T1\_TOA')  .filterDate('2018-01-01','2021-12-31')  .filter(ee.Filter.lt('CLOUD\_COVER', 15))  .filterBounds(roi);  var l8\_20182021\_median = l8\_20182021.median()  .clip(roi);  print(l8\_20182021) |

Click **Run** button in the top menu. The right window will show the results for the period of 2014 – 2017 and 2018 – 2021 in selected study area, respectively.

1. **Calculate NDVI and display results**

Method 1: using the built in function “.normalizedDifference(NIR, RED)”

|  |
| --- |
| var ndvi\_20142017=l8\_20142017\_median.normalizedDifference(['B5','B4']);  var ndviParams = {min: -1, max: 1, palette: ['red', 'yellow', 'green']};  Map.addLayer(ndvi\_20142017, ndviParams, "2014-2017 NDVI");  var ndvi\_20182021 = l8\_20182021\_median.normalizedDifference(['B5','B4']);  var ndviParams = {min: -1, max: 1, palette: ['red', 'yellow', 'green']};  Map.addLayer(ndvi\_20182021, ndviParams, "2018-2021 NDVI"); |

Method 2: using the simple band operations

|  |
| --- |
| var nir = l8\_20142017\_median.select('B5');  var red = l8\_20142017\_median.select('B4');  var ndvi\_20142017 = nir.subtract(red).divide(nir.add(red)).rename('NDVI');  var ndviParams = {min: -1, max: 1, palette: ['red', 'yellow', 'green']};  Map.addLayer(ndvi\_20142017, ndviParams, "2014-2017 NDVI");  var nir = l8\_20182021\_median.select('B5');  var red = l8\_20182021\_median.select('B4');  var ndvi\_20182021 = nir.subtract(red).divide(nir.add(red)).rename('NDVI');  var ndviParams = {min: -1, max: 1, palette: ['red', 'yellow', 'green']};  Map.addLayer(ndvi\_20182021, ndviParams, "2018-2021 NDVI"); |

Click the **Run** button in the top menu. The map window will display the NDVI results. Layer manager enables you to check different layers.

1. **Change detection**

A simple subtraction is used to calculate the NDVI difference between 2014 – 2017 and 2018 – 2021:

|  |
| --- |
| var change\_ndvi=ndvi\_20182021.subtract(ndvi\_20142017)  var ndvidiff = {min: -1, max: 1, palette: ['blue', 'white', 'green']};  Map.addLayer(change\_ndvi,ndvidiff,"NDVI\_change"); |

1. **Pixel value filtering**

|  |
| --- |
| var low\_ndvi= change\_ndvi.lt(0);  var lowNDVI = change\_ndvi.updateMask(low\_ndvi);  Map.addLayer(lowNDVI,ndvidiff,"Low NDVI");  var high\_ndvi=change\_ndvi.gt(0);  var highNDVI = change\_ndvi.updateMask(high\_ndvi);  Map.addLayer(highNDVI,ndvidiff,"High NDVI"); |

1. **Statistics of NDVI**

Calculate the basic statistics within the study area:

|  |
| --- |
| var MeansOfFeatures\_20142017 = ndvi\_20142017.reduceRegions({  collection: roi,  reducer: ee.Reducer.mean(),  scale: 250,  });  var MedianOfFeatures\_20142017 = ndvi\_20142017.reduceRegions({  collection: roi,  reducer: ee.Reducer.median(),  scale: 250,  });  var MinOfFeatures\_20142017 = ndvi\_20142017.reduceRegions({  collection: roi,  reducer: ee.Reducer.min(),  scale: 250,  });  var MaxOfFeatures\_20142017 = ndvi\_20142017.reduceRegions({  collection: roi,  reducer: ee.Reducer.max(),  scale: 250,  });  var CountOfFeatures\_20142017 = ndvi\_20142017.reduceRegions({  collection: roi,  reducer: ee.Reducer.count(),  scale: 250,  });  print(ee.Feature(MeansOfFeatures\_20142017.first()))  print(ee.Feature(MedianOfFeatures\_20142017.first()))  print(ee.Feature(MinOfFeatures\_20142017.first()))  print(ee.Feature(MaxOfFeatures\_20142017.first()))  print(ee.Feature(CountOfFeatures\_20142017.first()))  var MeansOfFeatures\_20182021 = ndvi\_20182021.reduceRegions({  collection: roi,  reducer: ee.Reducer.mean(),  scale: 250,  });  var MedianOfFeatures\_20182021 = ndvi\_20182021.reduceRegions({  collection: roi,  reducer: ee.Reducer.median(),  scale: 250,  });  var MinOfFeatures\_20182021 = ndvi\_20182021.reduceRegions({  collection: roi,  reducer: ee.Reducer.min(),  scale: 250,  });  var MaxOfFeatures\_20182021 = ndvi\_20182021.reduceRegions({  collection: roi,  reducer: ee.Reducer.max(),  scale: 250,  });  var CountOfFeatures\_20182021 = ndvi\_20182021.reduceRegions({  collection: roi,  reducer: ee.Reducer.count(),  scale: 250,  });  print(ee.Feature(MeansOfFeatures\_20182021.first()))  print(ee.Feature(MedianOfFeatures\_20182021.first()))  print(ee.Feature(MinOfFeatures\_20182021.first()))  print(ee.Feature(MaxOfFeatures\_20182021.first()))  print(ee.Feature(CountOfFeatures\_20182021.first())) |

< End >