Identifying Future Urban Growth and Environmental Sensitivity

CPLN 670 - Geospatial Software Design
Huiling He
Instructor: Prof. Dana Tomlin

The full Google Earth Engine code and ArcPy scripts are all available on github: https://github.com/huilingh/Final-project-for-CPLN670---Geospatial-Software-Design

Introduction

Over the past few decades, my hometown city – Guangzhou, located in the Pearl River Delta in south China, has been through a drastic urbanization process. Recently the central government announced the Pearl River Delta Greater Bay Area Plan, which aims at strengthening the connections and collaborations among cities in the Pearl River Delta. A larger scale of construction and urban growth is expected to take place with the announcement of the plan. As a student in urban planning, I am interested in finding how cities and human activities changes in the past few years, and how these changes impact our environments.

This project is going to identify areas where future urbanization is likely to take place and areas that are the most sensitive to environmental changes, and explore what factors are associated with future urbanization and environmental sensitivity. I will develop two indexes: the Future Urbanization Index (FUI) and the Environmental Sensitivity Index (ESI). For each of these two indexes, I will have several Decision Factors (DF), to find locations with the highest environmental sensitivity that are also the most likely to be developed as urban land in the future. These locations indicates that future urban growth is more likely to threaten our environment.

I will be using different approaches in Google Earth Engine and ArcPy for data exploration and calculations.

Google Earth Engine

Calculation of FUI and ESI

Google Earth Engine Methodology

Identify the decision factors of FUI that contribute to a higher probability of future urbanization. These factors will be given different weights:

- Areas with the most drastic population growth *0.3
- Distance to the existing urban land *0.3
- Distance to the major transportation assets (railroad) *0.2
- Flatlands (lower slope) – reclassify it as four quantile so that flat lands has the highest value *0.2

Identify the decision factors of ESI that contribute to a higher environmental sensitivity. These factors will be given different weights:

- Distance to water body *0.2
- Elevation *0.2
- Steep slope – reclassify it so that upper slope has the highest value *0.3
- Tree coverage *0.2
- Non-tree vegetation coverage *0.1

Overlay the FUI and ESI and identify areas where environment is more likely to be threatened by future urban growth:

FUI * ESI

Google Earth Engine Data Preparation

a. Identify the study region

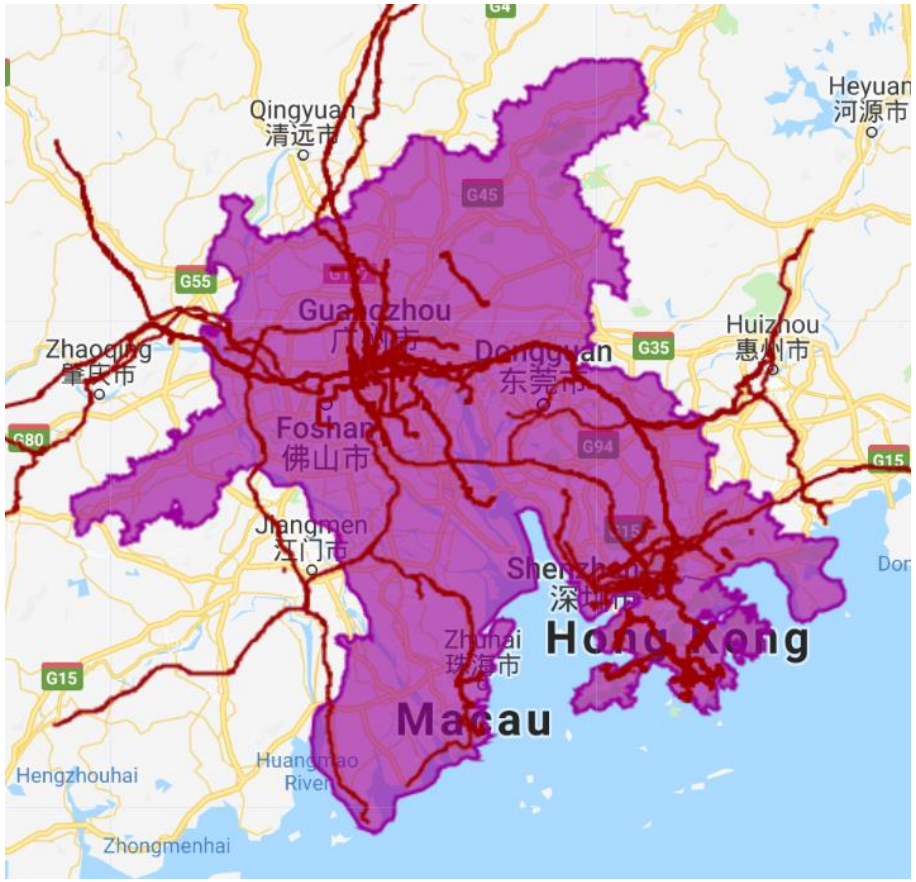
The study region will be consistent with the boundary defined by the Pearl River Delta Greater Bay Area Plan. I create the shapefile of the boundary in ArcGIS using an API from a map product to get the coordinates of the city's boundary. This shapefile is uploaded to Google Fushion table and loaded in EE script.


```
// Boundary of the study area
var boundary =
ee.FeatureCollection('ft:1Lxv4frwdPPBHYv9mJ_FSwaPUzEHgh4xxw6sakUVO');
Map.centerObject(boundary, 8);
Map.addLayer(boundary, {color: '990099'}, 'Study Area')
```

Google Earth Engine Data Preparation

b. Upload the railroad shapefile from OpenStreetMap

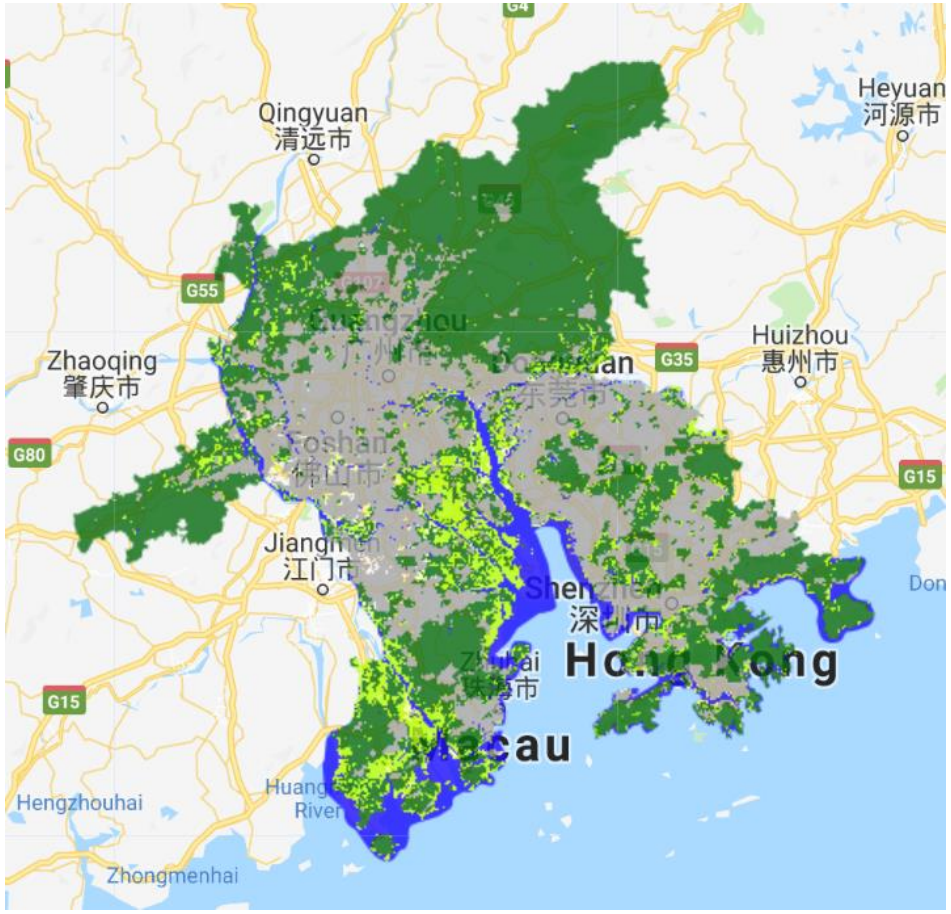
The railroad shapefile was derived from OpenStreetMap, also uploaded to Google Fushion table and loaded in EE script.




```
// Railways
var rail =
ee.FeatureCollection('ft:1M6J0GtaZomf3yDxz48gyCZk3sa7GQLSLk6KmYV_X');
//Map.addLayer(rail, {color: '990000'}, 'rail');
```

Google Earth Engine Data Preparation

c. Import images from Google Earth Engine dataset

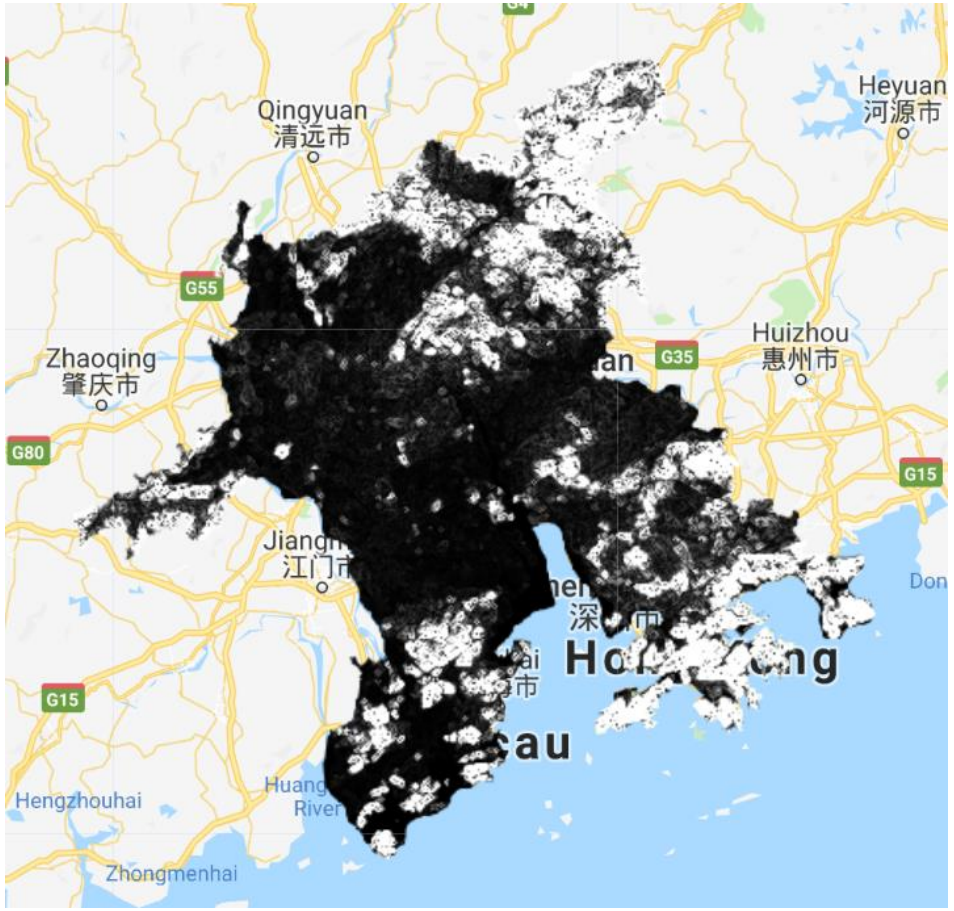



```
// MODIS Land Cover (2001-2016) in 2015
var land = ee.ImageCollection('MODIS/006/MCD12Q1').select(['LC_Type4']).filterDate('2015-01-01', '2015-12-31');
// Compute the mean value in 2015
var land15 = land.reduce(ee.Reducer.mean()).clip(boundary);
Map.addLayer(land15, {min:0, max:8, opacity:0.8,
palette:['1c0dff','05450a','086a10','54a708','78d203','009900','b6ff05','f9ffa4','a5a5a5']}, 'MODIS Land Type (2015)')
```

Google Earth Engine Data Preparation

c. Import images from Google Earth Engine dataset


```
// ALOS global DSM
var DSM =
ee.Image('JAXA/ALOS/AW3D30_V1_1').select(['AVE']).clip(boundary);
                Map.addLayer(DSM, {min:-479, max:500}, 'ALOS DSM')
```




```
// calculate the slope based on the ALOS global DSM data
var slope = ee.Terrain.slope(DSM)
Map.addLayer(slope, {min:0, max:5}, 'ALOS Slope')
```

Google Earth Engine Data Preparation

c. Import images from Google Earth Engine dataset


```
// MODIS Tree (2000-2016) in 2015
var tree =
ee.ImageCollection('MODIS/051/MOD44B').select(['Percent_Tree_Cover']).filterDate('2015-01-
01', '2015-12-31');
// Compute the mean value in 2015
var tree15 = tree.reduce(ee.Reducer.mean()).clip(boundary);
Map.addLayer(tree15, {opacity:0.8}, 'Tree Coverage in 2015')
```

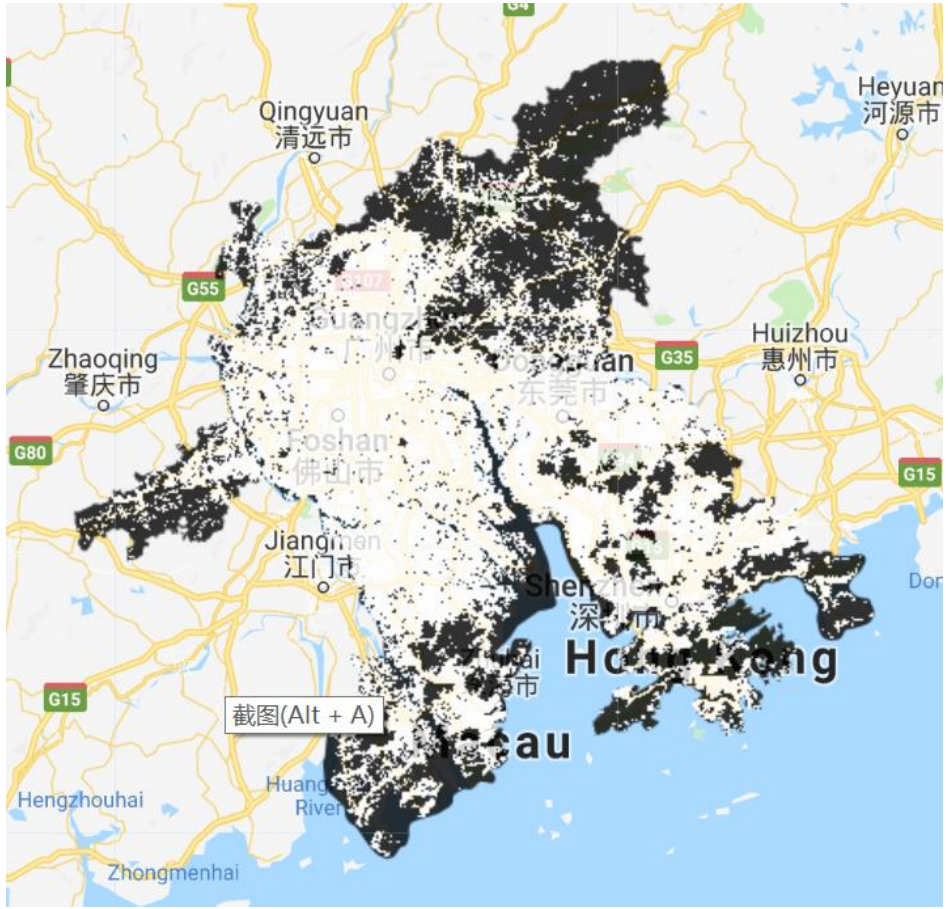
Google Earth Engine Data Preparation

c. Import images from Google Earth Engine dataset

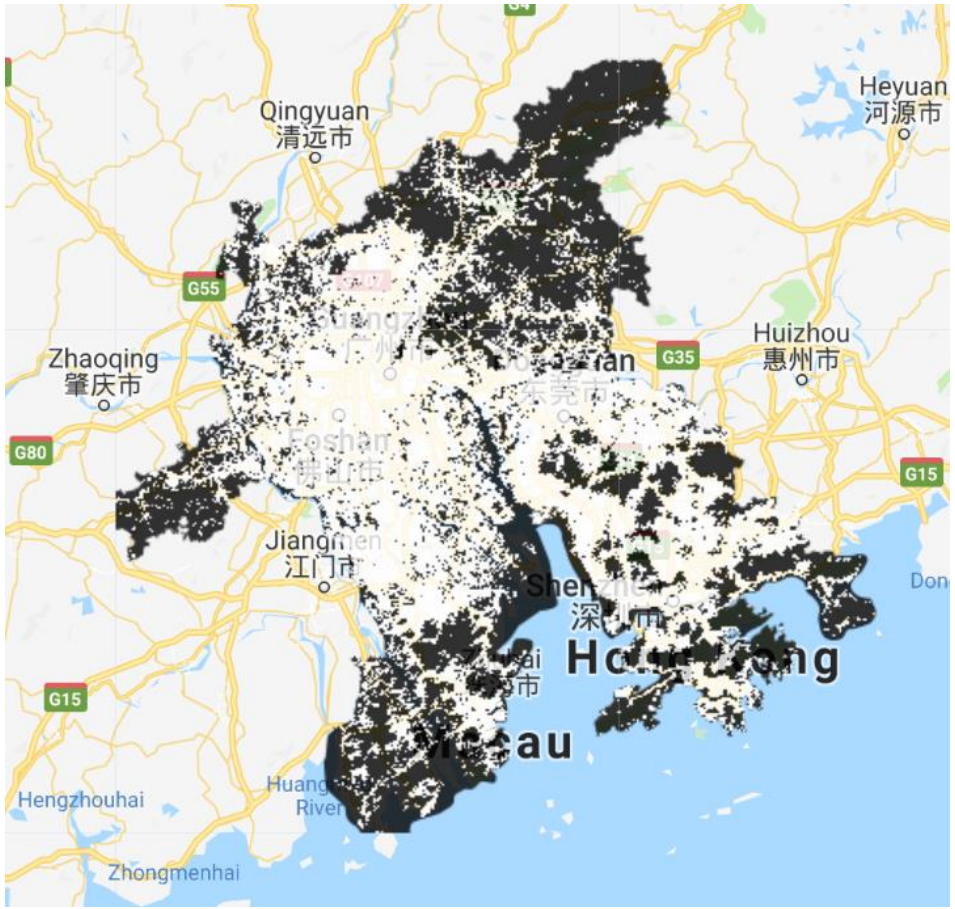

```
// MODIS non-tree vegetation coverage in 2015
var vege =
ee.ImageCollection('MODIS/051/MOD44B').select(['Percent_NonTree_Vegetation']).filterDate('20
15-01-01', '2015-12-31');
// Compute the mean value in 2015
var vege15 = vege.reduce(ee.Reducer.mean()).clip(boundary);
Map.addLayer(vege15, {opacity:0.8}, 'Non-Tree Coverage in 2015')
```

Google Earth Engine Data Preparation

c. Import images from Google Earth Engine dataset




```
// GHSL population grid in 2015
var GHSLpop15 =
ee.Image('JRC/GHSL/P2016/POP_GPW_GLOBE_V1/2015').clip(boundary)
Map.addLayer(GHSLpop15, {opacity:0.8}, 'Population in 2015')
```




```
// GHSL population grid in 2000
var GHSLpop00 =
ee.Image('JRC/GHSL/P2016/POP_GPW_GLOBE_V1/2000').clip(boundary)
Map.addLayer(GHSLpop00, {opacity:0.8}, 'Population in 2000')
```

Google Earth Engine Data Preparation

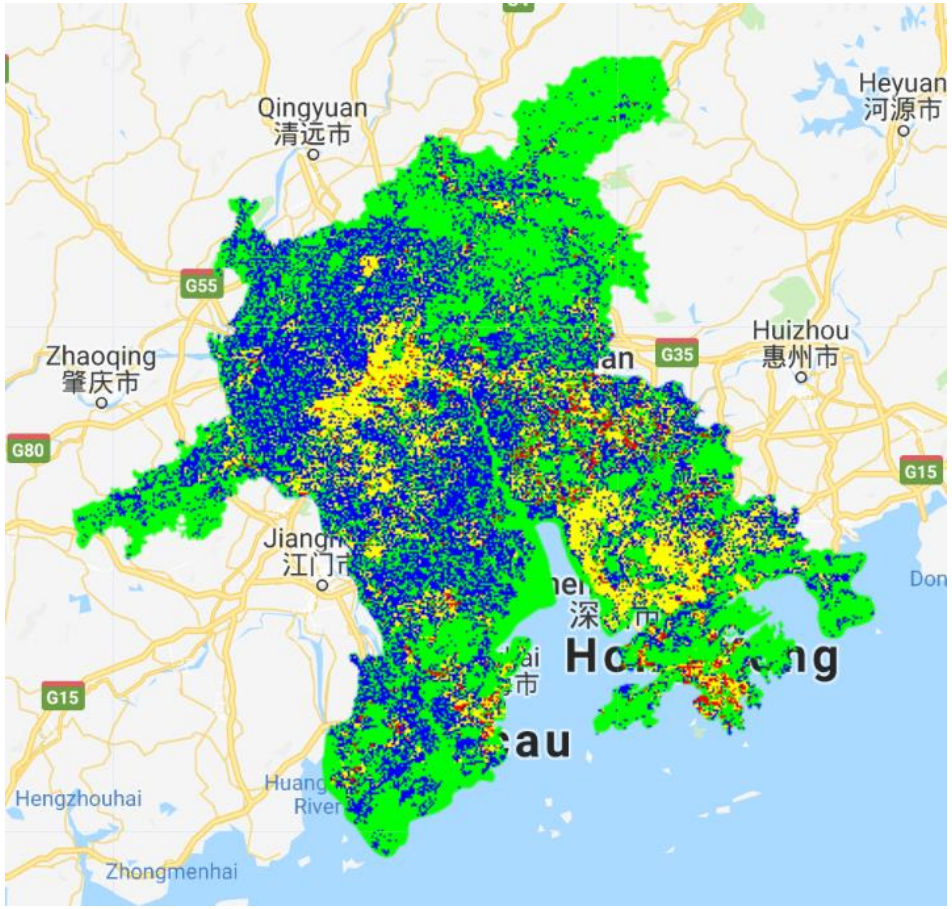
c. Import images from Google Earth Engine dataset


```
// the function to identify water body from MODIS land cover data; water will be reassigned to value 1, and the other types will be value 0
var remap_water = function(oldmap){
  var newmap = oldmap.remap([0], [1], 0, 'LC_Type4_mean')
  return newmap
}
var water = remap_water(land15)
Map.addLayer(water, {min:0, max:1, opacity: 0.5, palette:['000000','0000ff']}, 'Water Body')
```

Google Earth Engine Decision Factors of FUI

a. Calculate and reclassify population change from 2000 to 2015

Larger values will be reassigned to higher value, indicating that urban growth is more likely to occur in areas with large population change

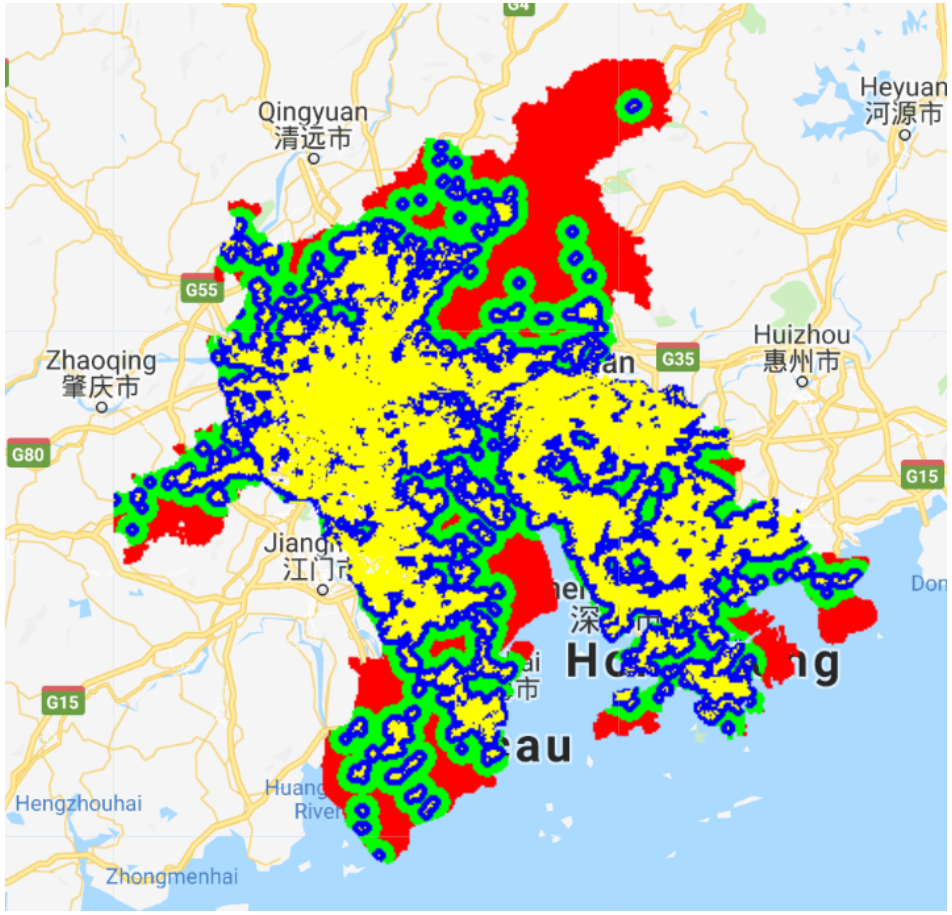



```
// DF1: Population changes
var popChange = GHSLpop15.subtract(GHSLpop00)
Map.addLayer(popChange, {min:-5000, max:5000}, 'Population Change')
print(popChange)

// generate a histogram of population change and see the distribution
var histPop = ui.Chart.image.histogram(popChange, boundary, 50)
print(histPop)

// reclassify population change
var reclassify_Pop = function(oldmap){
  var first = oldmap.lte(-128);
  var second = oldmap.lte(0).and(oldmap.gt(-128));
  var third = oldmap.lte(128).and(oldmap.gt(0));
  var four = oldmap.gte(128);

  var reclass1 = first.remap([1],[1],0)
  var reclass2 = second.remap([1],[2],0);
  var reclass3 = third.remap([1],[3],0);
  var reclass4 = four.remap([1],[4],0);

  var reclass = reclass1.add(reclass2).add(reclass3).add(reclass4);
  return reclass;
}

var reclassPop = reclassify_Pop(popChange);
    Map.addLayer(reclassPop, {min:1, max:4, palette:['ff0000','00ff00','0000ff','ffff00']}, 'Reclassed
                                                    population change')
```

Google Earth Engine Decision Factors of FUI

b. Calculate and reclassify distance to existing urban lands in 2015

Smaller distance will be reassigned to higher value, indicating that urban growth is more likely to occur in areas closer to existing urban land

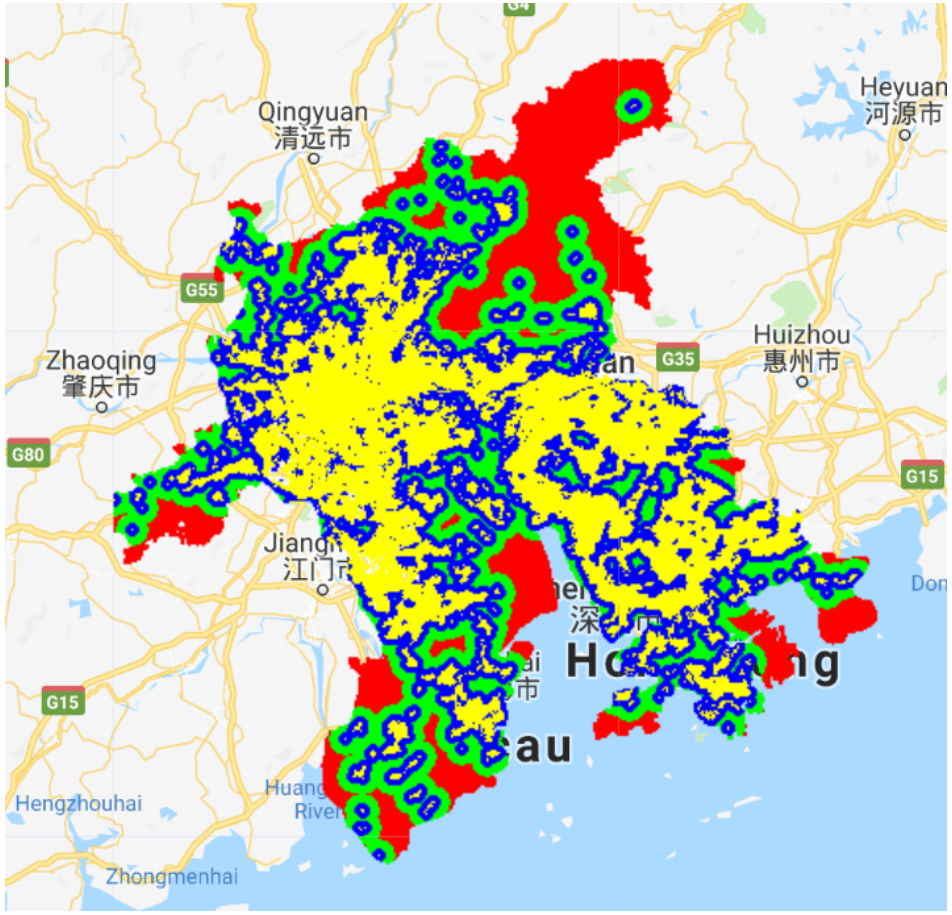



```
// DF2: Distance to existing urban lands in 2015
// the function to identify the urban and built-up lands from MODIS land cover; urban lands will be reassigned to value 1, and the other types will be value 0
var remap_urban = function(oldmap){
  var newmap = oldmap.remap([8], [1], 0, 'LC_Type4_mean')
  return newmap
}
var urban15 = remap_urban(land15)
Map.addLayer(urban15, {min:0, max:1, opacity:0.7}, 'Urban 2015')
print(urban15)

// euclident distance to urban land in 2015
var disUrban = urban15.distance(ee.Kernel.euclidean(50000, 'meters'))
Map.addLayer(disUrban, {min:0, max:5000, opacity:0.7}, 'Distance to Urban')

// generate a histogram of distance to urban land and see the distribution
var histUrban = ui.Chart.image.histogram(disUrban, boundary, 500)
print(histUrban)
```

Google Earth Engine Decision Factors of FUI

b. Calculate and reclassify distance to existing urban lands in 2015

Smaller distance will be reassigned to higher value, indicating that urban growth is more likely to occur in areas closer to existing urban land

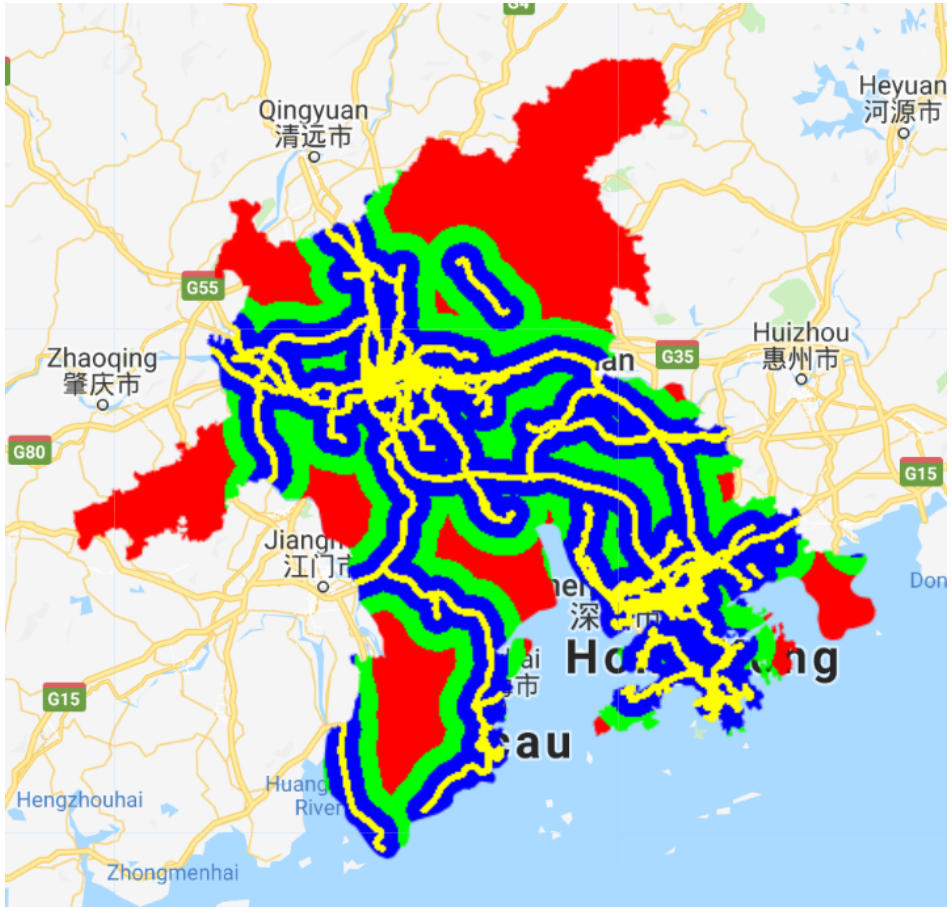



```
 // reclassify distance to urban land
var reclassify_Urb = function(oldmap){
  var first = oldmap.lte(0);
  var second = oldmap.lte(2000).and(oldmap.gt(0));
  var third = oldmap.lte(5000).and(oldmap.gt(2000));
  var four = oldmap.gte(5000);

  var reclass1 = first.remap([1],[4],0)
  var reclass2 = second.remap([1],[3],0);
  var reclass3 = third.remap([1],[2],0);
  var reclass4 = four.remap([1],[1],0);

  var reclass = reclass1.add(reclass2).add(reclass3).add(reclass4);
  return reclass;
}

var reclassUrb = reclassify_Urb(disUrban);
      Map.addLayer(reclassUrb, {min:1, max:4, palette:['ff0000','00ff00','0000ff','ffff00']}, 'Reclassed
                                                    distance to urban')
```

Google Earth Engine Decision Factors of FUI

c. Calculate and reclassify distance to railroad

Smaller distance will be reassigned to higher value, indicating that urban growth is more likely to occur in areas closer to railroad

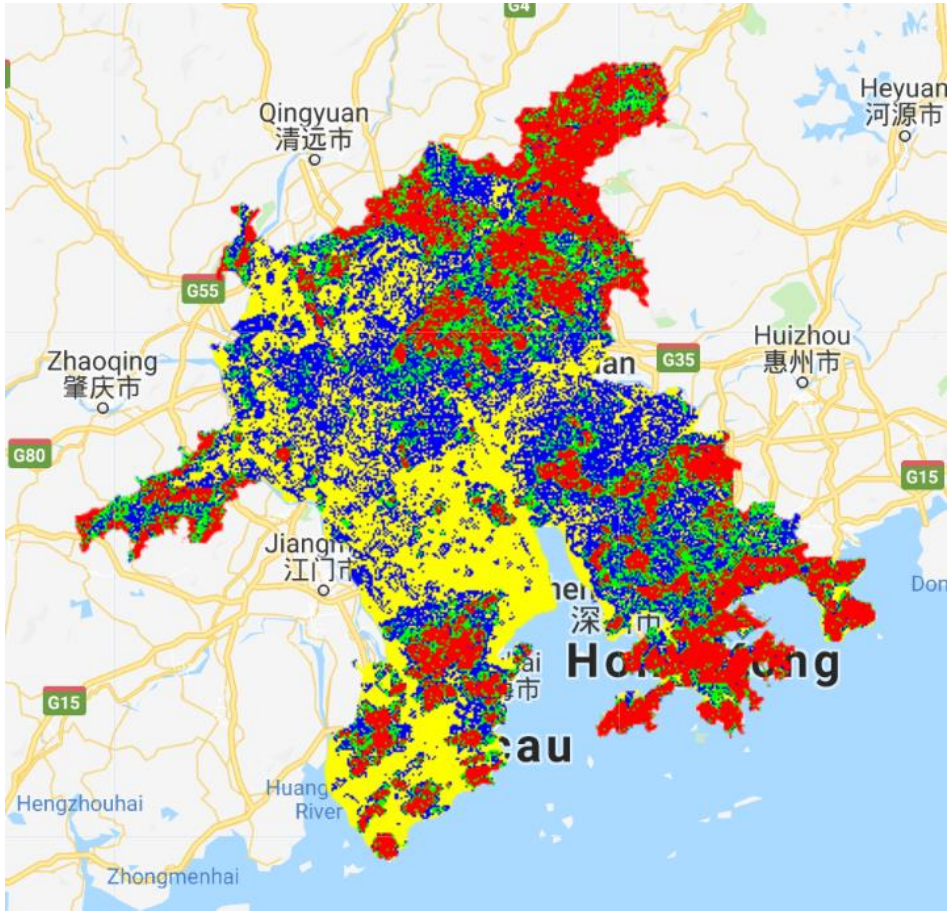



```
// DF3: Distance to railroads
// distance to railroads
var disTrans = rail.distance(100000).clip(boundary)
Map.addLayer(disTrans, {min:0, max:20000, opacity:0.7, palette:['ff0000','ffff00']}, 'Distance to Rail')
print(disTrans)

// generate a histogram of distance to railroads and see the distribution
var histTrans = ui.Chart.image.histogram(disTrans, boundary, 500)
print(histTrans)

// reclassify distance to railroads
var reclassify_Trans = function(oldmap){
  var first = oldmap.lte(1000);
  var second = oldmap.lte(5000).and(oldmap.gt(1000));
  var third = oldmap.lte(10000).and(oldmap.gt(5000));
  var four = oldmap.gte(10000);

  var reclass1 = first.remap([1],[4],0)
  var reclass2 = second.remap([1],[3],0);
  var reclass3 = third.remap([1],[2],0);
  var reclass4 = four.remap([1],[1],0);

  var reclass = reclass1.add(reclass2).add(reclass3).add(reclass4);
  return reclass;
}

var reclassTrans = reclassify_Trans(disTrans);
Map.addLayer(reclassTrans, {min:1, max:4, palette:['ff0000','00ff00','0000ff','ffff00']}, 'Reclassed
distance to rail')
```

Google Earth Engine Decision Factors of FUI

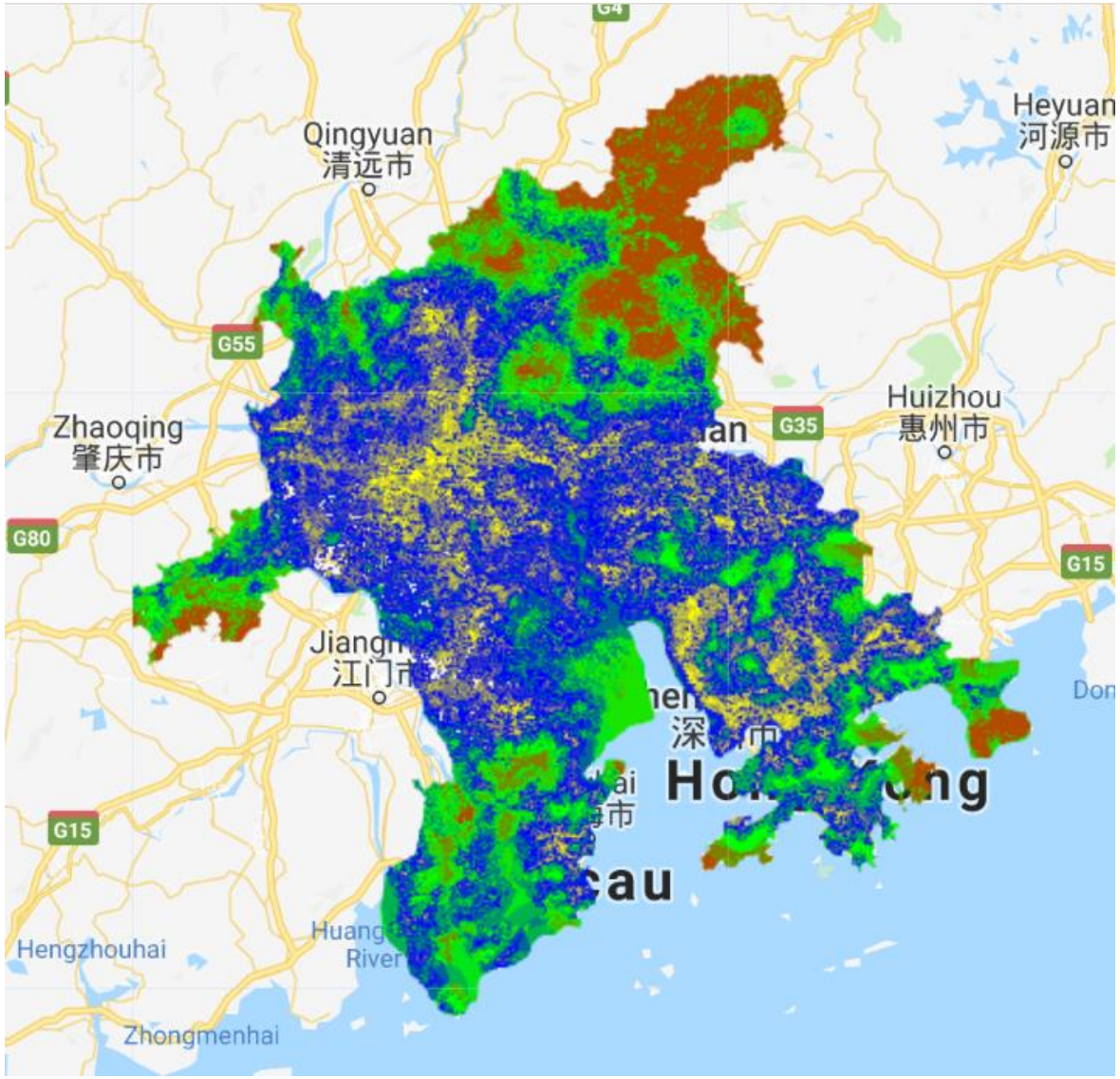
d. Calculate and reclassify slope

Lower slope will be reassigned to higher value, indicating that urban growth is more likely to occur on lower slopes areas




```
// DF4: Slope (lower slopes)
// generate a histogram of slope and see the distribution
var histSlope = ui.Chart.image.histogram(slope, boundary, 500)
print(histSlope)

// the slope image will be reclassified into two new images:
// the first reclassification of slope is for the decision factor of FUI:
// slope with lower values will be reassigned to higher value, indicating that urban growth is more likely
to occur on lower slopes areas
var reclassify_Slope1 = function(oldmap){
  var first = oldmap.lte(0.25);
  var second = oldmap.lte(1).and(oldmap.gt(0.25));
  var third = oldmap.lte(3).and(oldmap.gt(1));
  var four = oldmap.gte(3);

  var reclass1 = first.remap([1],[4],0)
  var reclass2 = second.remap([1],[3],0);
  var reclass3 = third.remap([1],[2],0);
  var reclass4 = four.remap([1],[1],0);

  var reclass = reclass1.add(reclass2).add(reclass3).add(reclass4);
  return reclass;
}

var reclassSlope1 = reclassify_Slope1(slope);
Map.addLayer(reclassSlope1, {min:1, max:4, palette:['ff0000','00ff00','0000ff','ffff00']}, 'Reclassed
slope (FUI)')
```

Google Earth Engine Calculate FUI by giving weights to the Decision Factors




```
// calculate FUI giving weights to the decision factors
var FUI =
reclassPop.multiply(0.3).add(reclassUrb.multiply(0.3)).add(reclassTrans.multipl
y(0.2)).add(reclassSlope1.multiply(0.2))
Map.addLayer(FUI, {min:1, max:4, palette:['ff0000','00ff00','0000ff','ffff00']},
'FUI')
```

Higher value indicates that future urbanization is more likely to occur

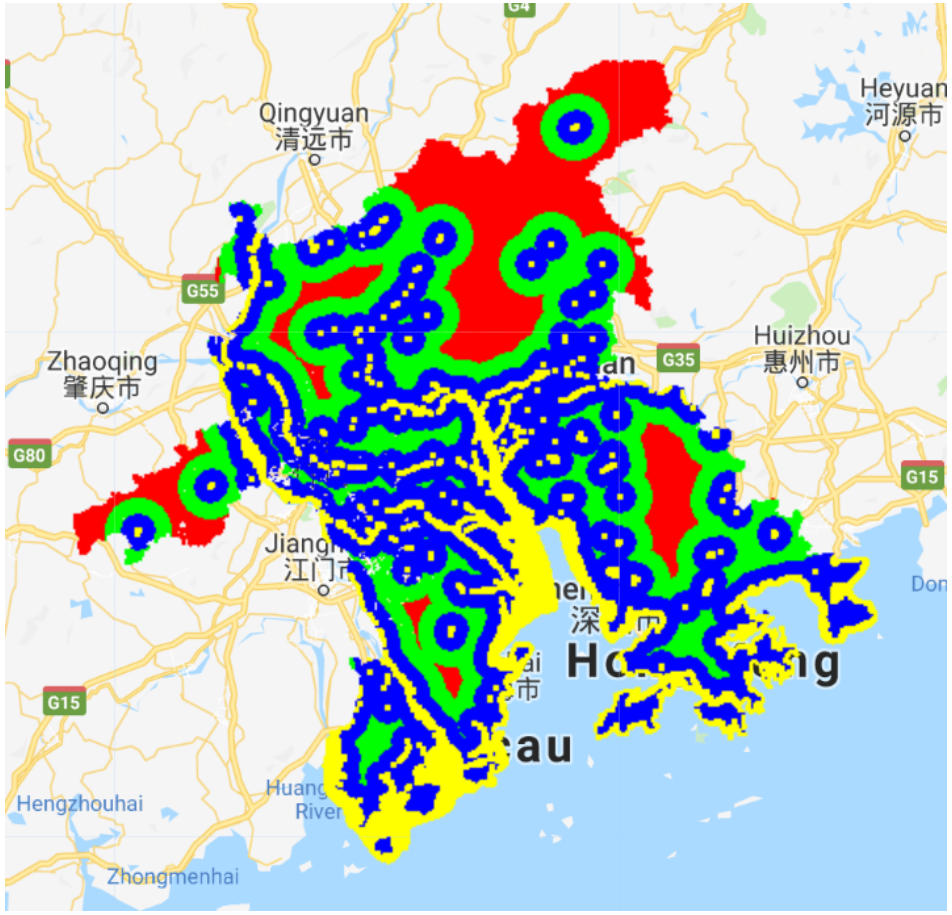
FUI Value

- 1
- 2
- 3
- 4

Google Earth Engine Decision Factors of ESI

a. Calculate and reclassify distance to water body

Smaller distance will be reassigned to higher value, indicating that areas closer to water body are more environmentally sensitive

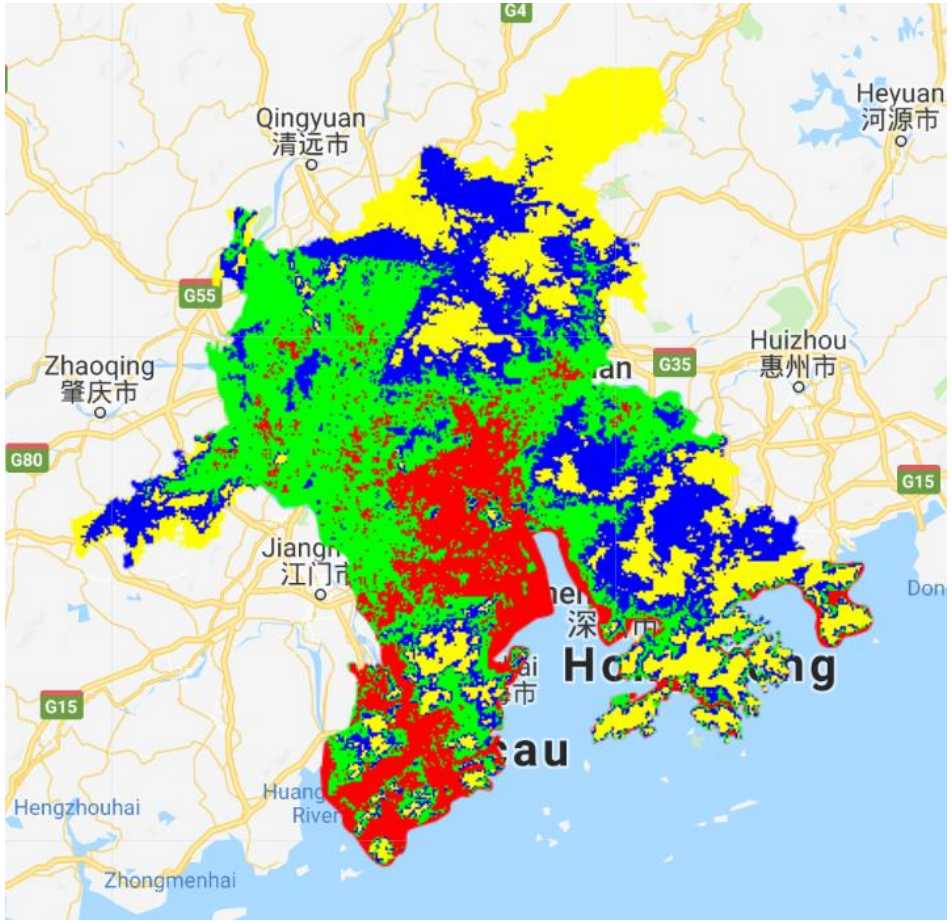



```
// DF1: Distance to water body
// euclident distance to water body in 2015
var disWater = water.distance(ee.Kernel.euclidean(70000, 'meters'))
Map.addLayer(disWater, {min:0, max:10000, opacity:1}, 'Distance to Water')

// generate a histogram of distance to water and see the distribution
var histWater = ui.Chart.image.histogram(disWater, boundary, 800)
print(histWater)

// reclassify distance to water
var reclassify_Water = function(oldmap){
  var first = oldmap.lte(1000);
  var second = oldmap.lte(5000).and(oldmap.gt(1000));
  var third = oldmap.lte(10000).and(oldmap.gt(5000));
  var four = oldmap.gte(10000);

  var reclass1 = first.remap([1],[4],0)
  var reclass2 = second.remap([1],[3],0);
  var reclass3 = third.remap([1],[2],0);
  var reclass4 = four.remap([1],[1],0);

  var reclass = reclass1.add(reclass2).add(reclass3).add(reclass4);
  return reclass;
}

var reclassWater = reclassify_Water(disWater);
Map.addLayer(reclassWater, {min:1, max:4, palette:['ff0000','00ff00','0000ff','ffff00']}, 'Reclassed
distance to water')
```

Google Earth Engine Decision Factors of ESI

b. Calculate and reclassify elevation, so that higher elevation will be reassigned to higher value

Higher elevation will be reassigned to higher value, indicating that areas with higher elevation are more environmentally sensitive

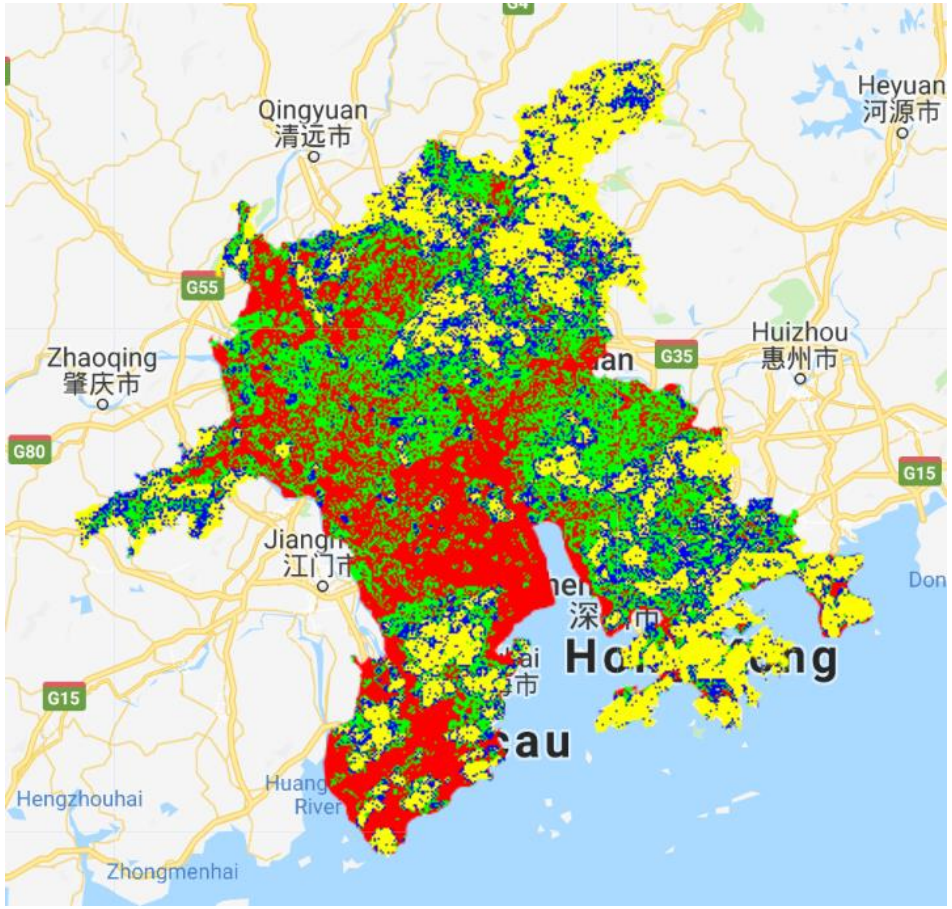



```
// DF2: Elevation
// generate a histogram of DSM elevation and see the distribution
var histDSM = ui.Chart.image.histogram(DSM, boundary, 800)
print(histDSM)

// reclassify elevation
var reclassify_DSM = function(oldmap){
  var first = oldmap.lte(0);
  var second = oldmap.lte(16).and(oldmap.gt(0));
  var third = oldmap.lte(72).and(oldmap.gt(16));
  var four = oldmap.gte(72);

  var reclass1 = first.remap([1],[1],0)
  var reclass2 = second.remap([1],[2],0);
  var reclass3 = third.remap([1],[3],0);
  var reclass4 = four.remap([1],[4],0);

  var reclass = reclass1.add(reclass2).add(reclass3).add(reclass4);
  return reclass;
}

var reclassDSM = reclassify_DSM(DSM);
Map.addLayer(reclassDSM, {min:1, max:4, palette:['ff0000','00ff00','0000ff','ffff00']}, 'Reclassed
elevation')
```

Google Earth Engine Decision Factors of ESI

c. Calculate and reclassify slope, so that higher slope will be reassigned to higher value

Higher slope will be reassigned to higher value, indicating that areas with higher slope are more environmentally sensitive

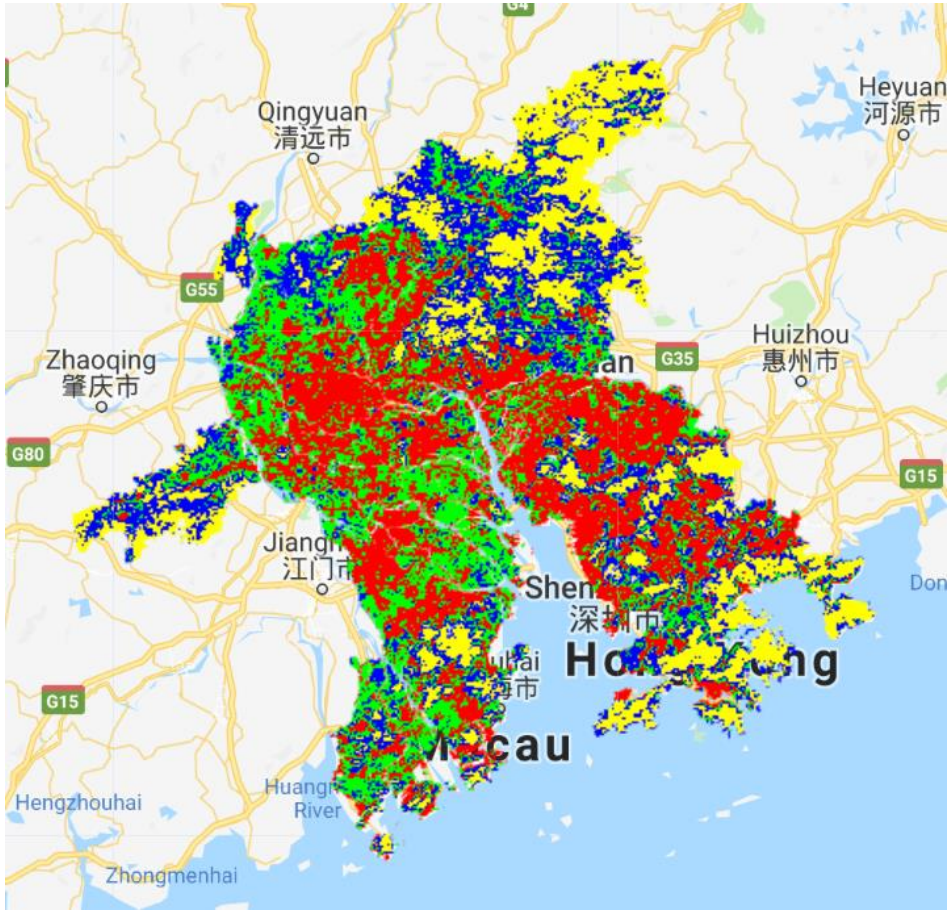



```
// DF3: Slope (higher slopes)
// the second reclassification of slope is for the decision factor of ESI:
// slope with higher values will be reassigned to higher value, indicating that higher slopes areas are more environmentally sensitive
// (the second reclassification is the reverse of the first one)
var reclassify_Slope2 = function(oldmap){
  var first = oldmap.lte(0.25);
  var second = oldmap.lte(1).and(oldmap.gt(0.25));
  var third = oldmap.lte(3).and(oldmap.gt(1));
  var four = oldmap.gte(3);

  var reclass1 = first.remap([1],[1],0)
  var reclass2 = second.remap([1],[2],0);
  var reclass3 = third.remap([1],[3],0);
  var reclass4 = four.remap([1],[4],0);

  var reclass = reclass1.add(reclass2).add(reclass3).add(reclass4);
  return reclass;
}

var reclassSlope2 = reclassify_Slope2(slope);
Map.addLayer(reclassSlope2, {min:1, max:4, palette:['ff0000','00ff00','0000ff','ffff00']}, 'Reclassed slope (ESI)')
```

Google Earth Engine Decision Factors of ESI

d. Calculate and reclassify tree coverage, so that larger tree coverage will be reassigned to higher value

Larger tree coverage will be reassigned to higher value, indicating that areas with more trees are more environmentally sensitive

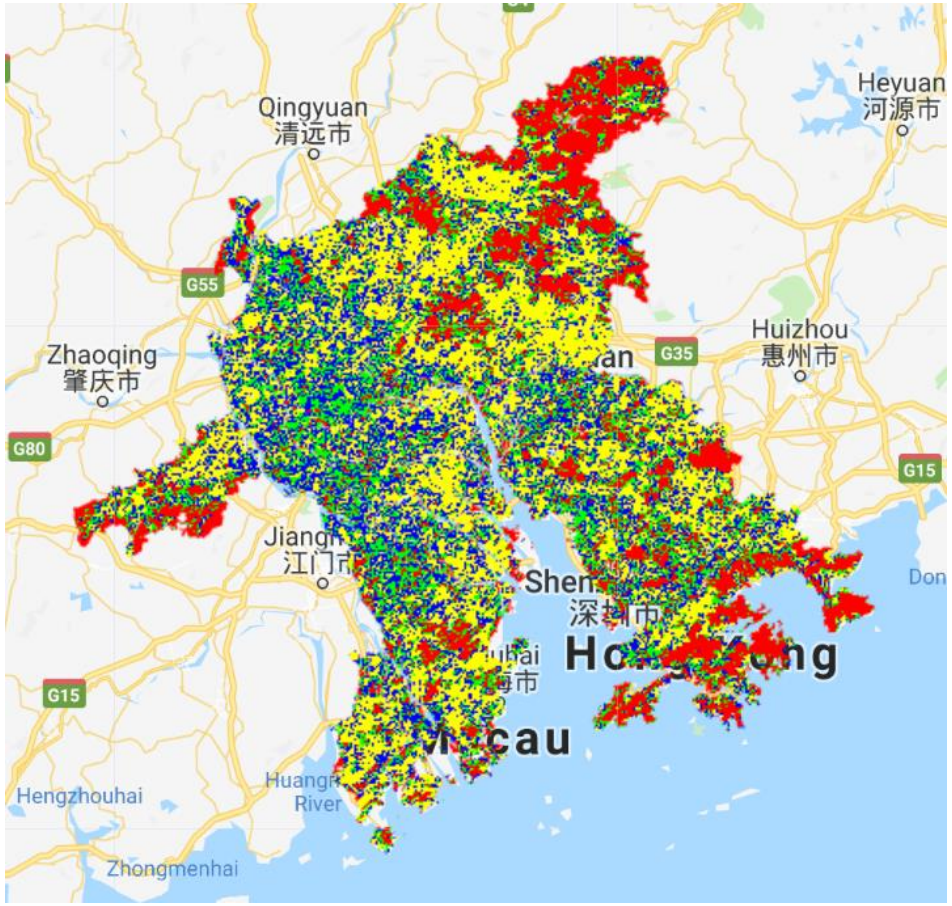



```
// DF4: Tree coverage
// generate a histogram of tree coverage percentage and see the distribution
var histTree = ui.Chart.image.histogram(tree15, boundary, 500)
print(histTree)

// reclassify tree coverage
var reclassify_Tree = function(oldmap){
  var first = oldmap.lte(10);
  var second = oldmap.lte(20).and(oldmap.gt(10));
  var third = oldmap.lte(50).and(oldmap.gt(20));
  var four = oldmap.gte(50);

  var reclass1 = first.remap([1],[1],0)
  var reclass2 = second.remap([1],[2],0);
  var reclass3 = third.remap([1],[3],0);
  var reclass4 = four.remap([1],[4],0);

  var reclass = reclass1.add(reclass2).add(reclass3).add(reclass4);
  return reclass;
}

var reclassTree = reclassify_Tree(tree15);
Map.addLayer(reclassTree, {min:1, max:4, palette:['ff0000','00ff00','0000ff','ffff00']}, 'Reclassed tree
coverage')
```

Google Earth Engine Decision Factors of ESI

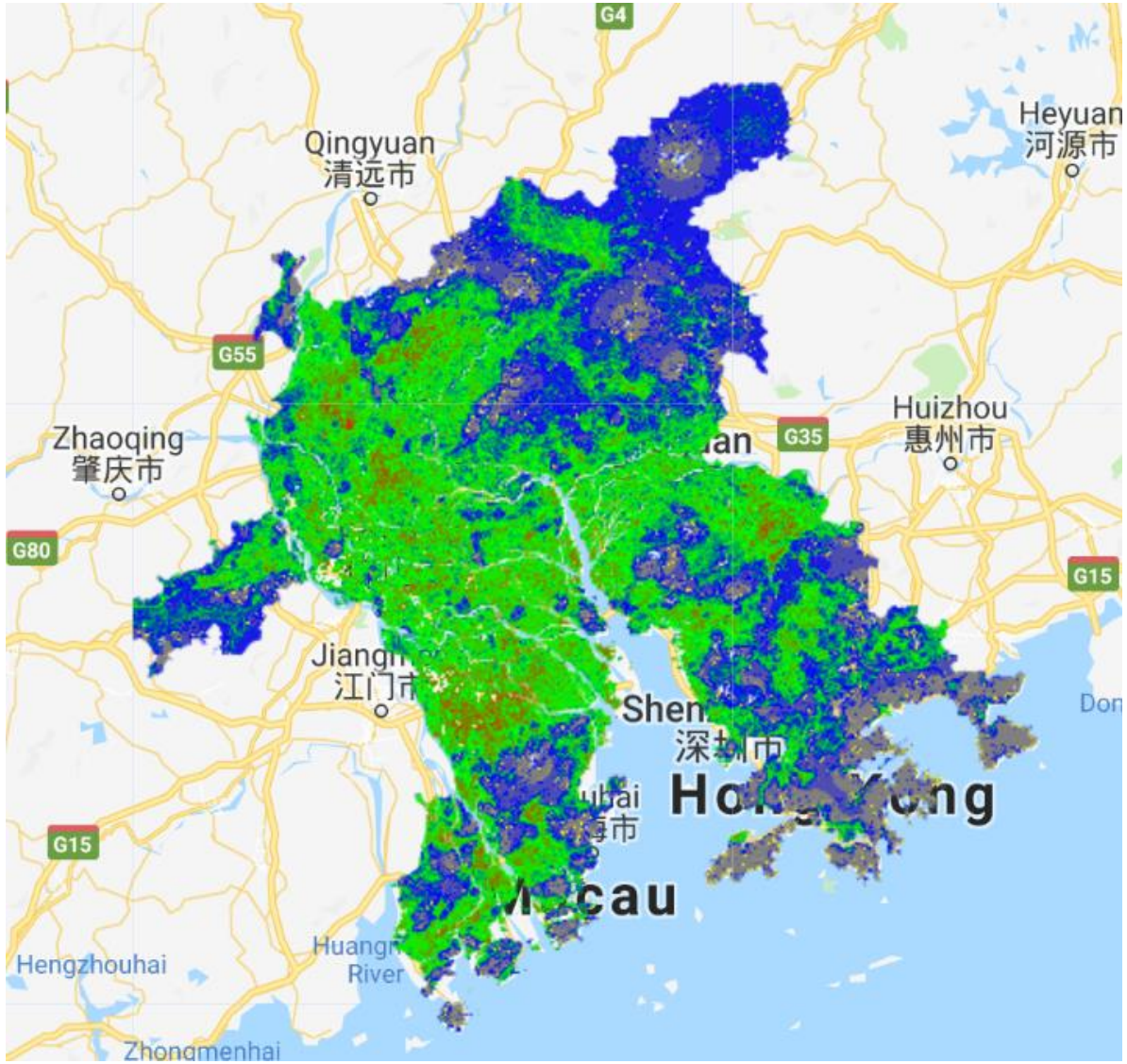
e. Calculate and reclassify non-tree vegetation coverage, so that larger tree coverage will be reassigned to higher value

Larger non-tree vegetation coverage will be reassigned to higher value as well




```
// DF5: Non-tree vegetation coverage
// generate a histogram of non-tree vegetation coverage percentage and see the distribution
var histVege = ui.Chart.image.histogram(vege15, boundary, 500)
print(histVege)

// reclassify non-tree vegetation coverage
var reclassify_Vege = function(oldmap){
  var first = oldmap.lte(40);
  var second = oldmap.lte(50).and(oldmap.gt(40));
  var third = oldmap.lte(60).and(oldmap.gt(50));
  var four = oldmap.gte(60);

  var reclass1 = first.remap([1],[1],0)
  var reclass2 = second.remap([1],[2],0);
  var reclass3 = third.remap([1],[3],0);
  var reclass4 = four.remap([1],[4],0);

  var reclass = reclass1.add(reclass2).add(reclass3).add(reclass4);
  return reclass;
}

var reclassVege = reclassify_Vege(vege15);
Map.addLayer(reclassVege, {min:1, max:4, palette:['ff0000','00ff00','0000ff','ffff00']}, 'Reclassed non-tree vegetation')
```

Google Earth Engine Calculate ESI by giving weights to the Decision Factors



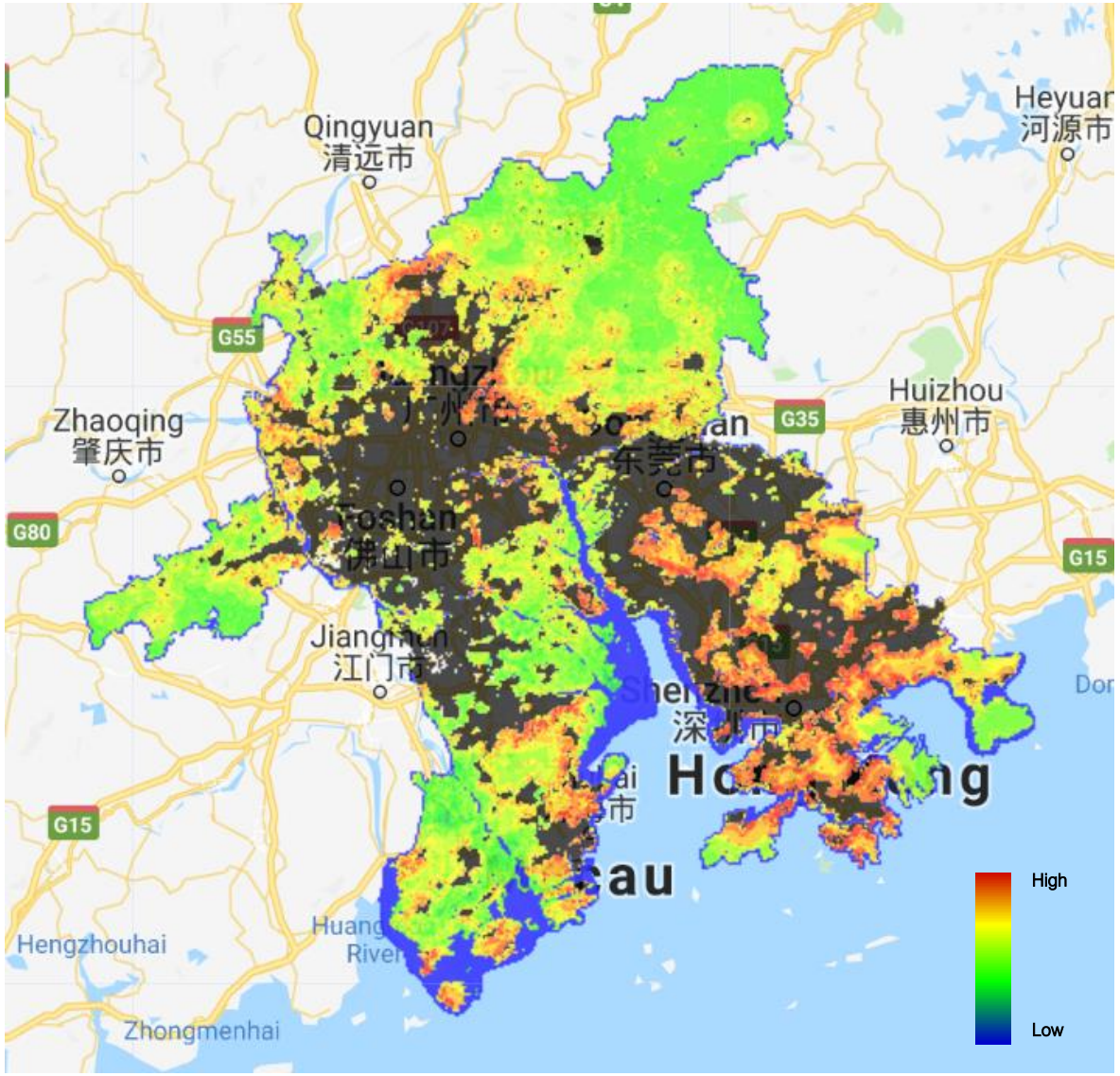

```
// calculate ESI giving weights to the decision factors
var ESI =
reclassWater.multiply(0.2).add(reclassDSM.multiply(0.2)).add(reclassSlope2.multiply(0.3)).add(reclassTree.multiply(0.2)).add(reclassVege.multiply(0.1));
Map.addLayer(ESI, {min:1, max:4, palette:['ff0000','00ff00','0000ff','ffff00']}, 'ESI')
```

Higher value indicates a higher environmental sensitivity

ESI Value

- 1
- 2
- 3
- 4

Google Earth Engine Overlay FUI and ESI



FUI / ESI	1	2	3	4
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12
4	4	8	12	16

Index:
Grid cells with the highest environmental sensitivity that are also the most likely areas to be developed as urban areas, which indicates that urban growth is potentially threatening our environment.

```
// combine FUI and ESI
var COMB = FUI.multiply(ESI)

// create a mask to hide existing urban land
var maskUrban = urban15.neq(1)
Map.addLayer(urban15, {min:0, max:1, palette:['ffffff','000000'], opacity:0.7},
'Urban 2015')
Map.addLayer(COMB.mask(maskUrban), {min:1, max:9,
palette:['0000ff','00ff00','ffff00','ff0000'], opacity:0.7}, 'Combined FUI/ESI')
```

ArcPy

A toolbox of data exploration by extracting information from raster to vector

ArcPy Methodology

This ArcPy toolbox contains four tools:

Tool1: create a fishnet of the study region

Tool2: run GWR and compute FUI

Tool3: run GWR and compute ESI

Tool4: overlay the result of FUI and ESI

ArcPy Methodology

The purpose of building this toolbox is to further explore the image data obtained from Google Earth Engine dataset and try to identify future urban growth and environmental sensitivity through a different approach in ArcPy than in Google Earth Engine. Unlike Google Earth Engine in which I calculate the Future Urbanization Index (FUI) and Environmental Sensitivity Index (ESI) by giving different weights of the decision factors, here I will use one indicator of urban growth (nightlight changes) and one indicator of environmental sensitivity (ALOS topographic diversity) to explore the spatial relationships between the decision factors the indicators. Using ArcPy, I will run two Geographically Weighted Regressions:

The first regression is for FUI: the dependent variable is the nightlight changes from 2000 to 2013, and the predictors are the decision factors for calculating the FUI in Google Earth Engine, including the population change from 2000 to 2015, distance to existing land in 2005, distance to railroad, and slope.

The second regression is for ESI: the dependent variable is the ALOS topographic diversity, and the predictors are the decision factors for calculating the ESI in Google Earth Engine, including the distance to water, elevation, slope, tree coverage and non-tree vegetation coverage in 2015.

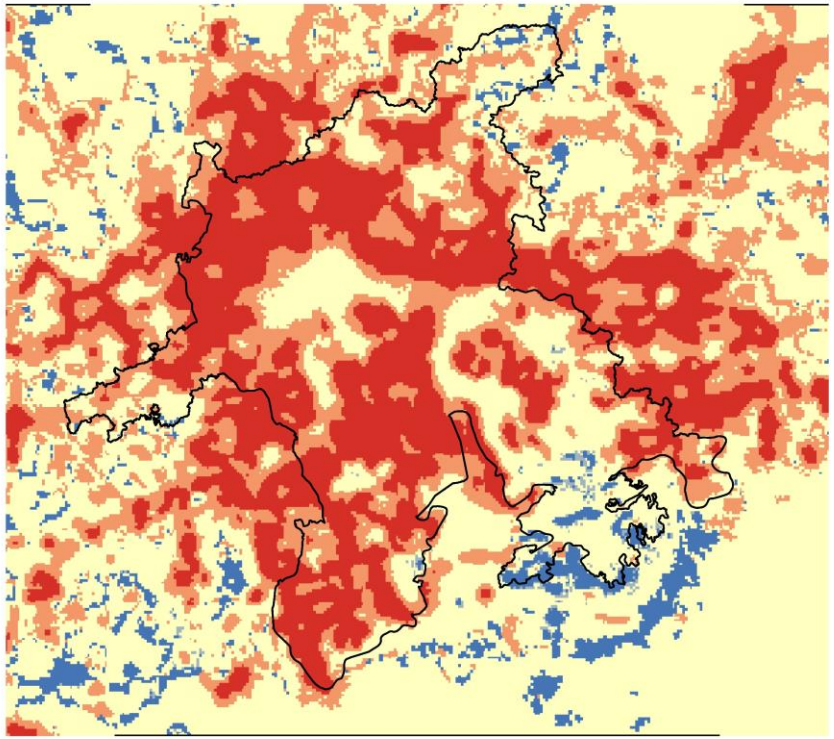
We will see how these decision factors (i.e., predictors) are spatially correlated with the indicators (i.e., dependent variable).

Similar to Google Earth Engine, in ArcPy, I will reclassify the predicted values from the GWR results into values 1 to 4, and multiply the predicted values of FUI (the predicted nightlight change) and ESI (the predicted topographic diversity) to identify where urban growth is potentially threatening our environment.

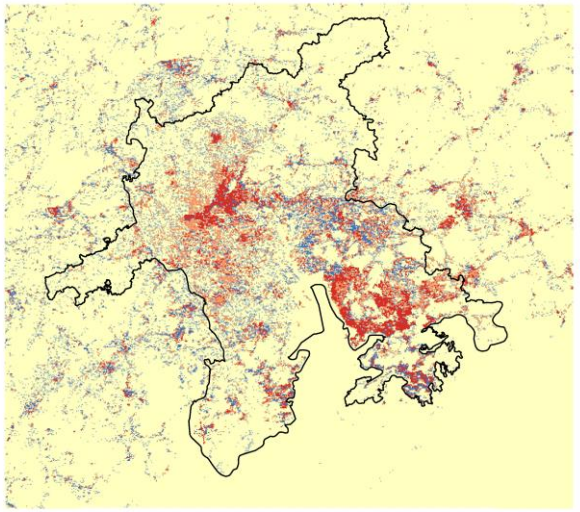
ArcPy Data Preparation

Export images from Google EE as tif and load them in GIS

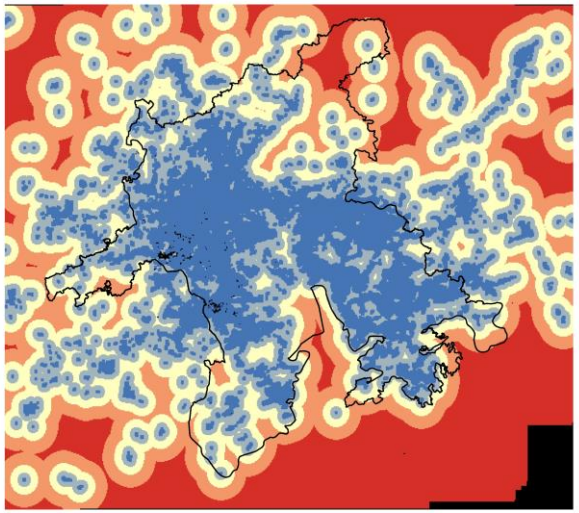
a. Dependent variable (Nightlight change) and predictors of FUI



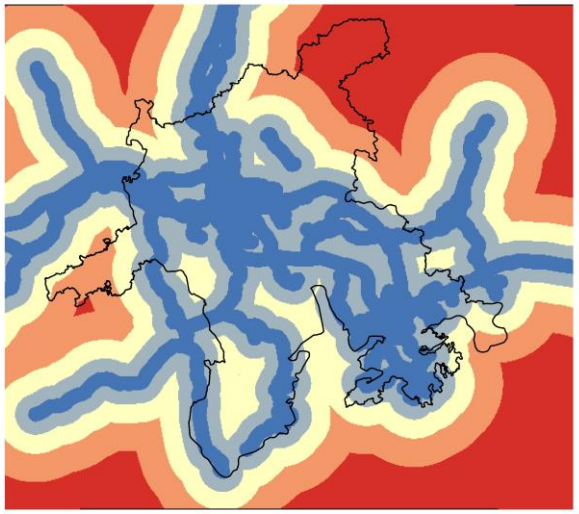
Nightlight change (2000-2013)



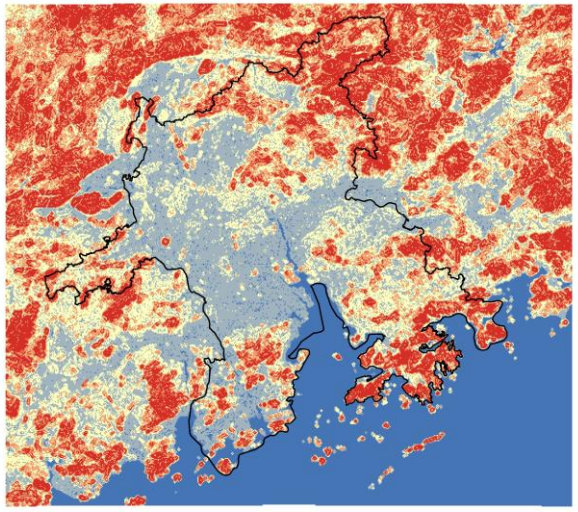
Population change (2000-2015)



Distance to existing urban land in 2005



Distance to railroad

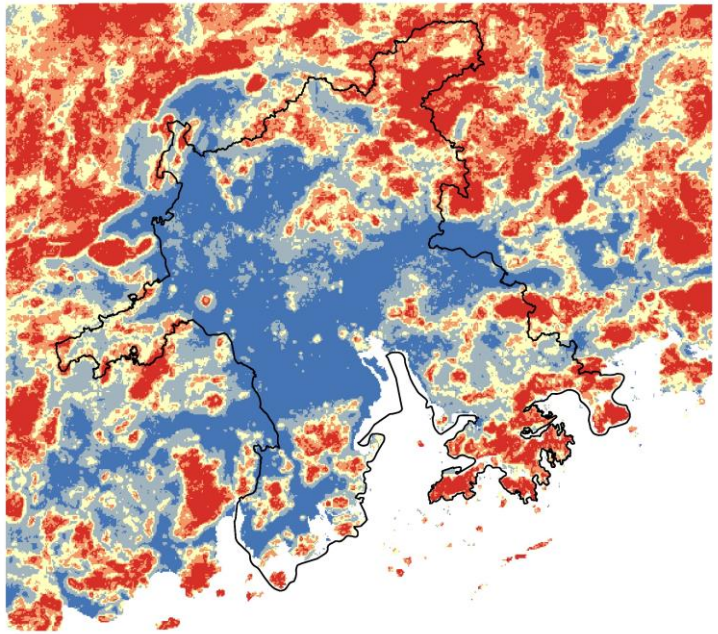


Slope

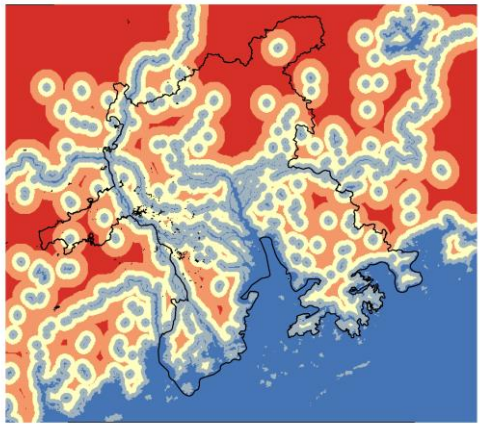
ArcPy Data Preparation

Export images from Google EE as tif and load them in GIS

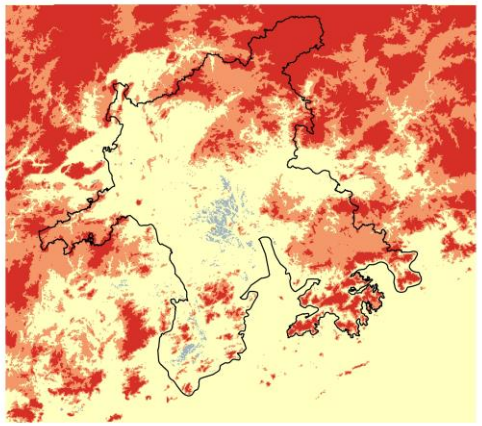
b. Dependent variable (Topographic diversity) and predictors of ESI



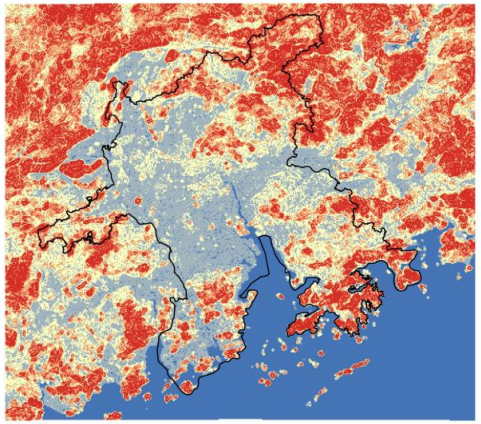
ALOS Topographic diversity



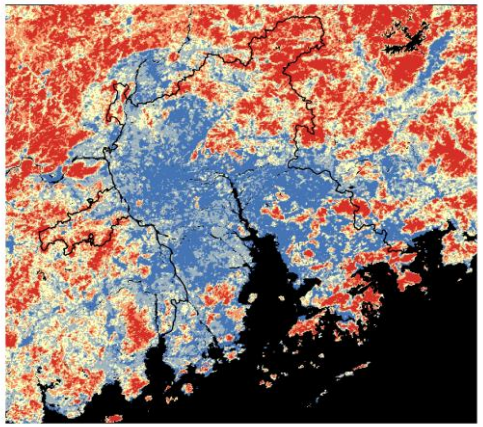
Distance to water



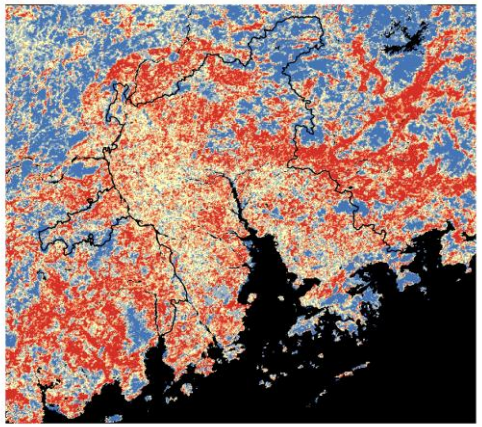
Elevation



Slope

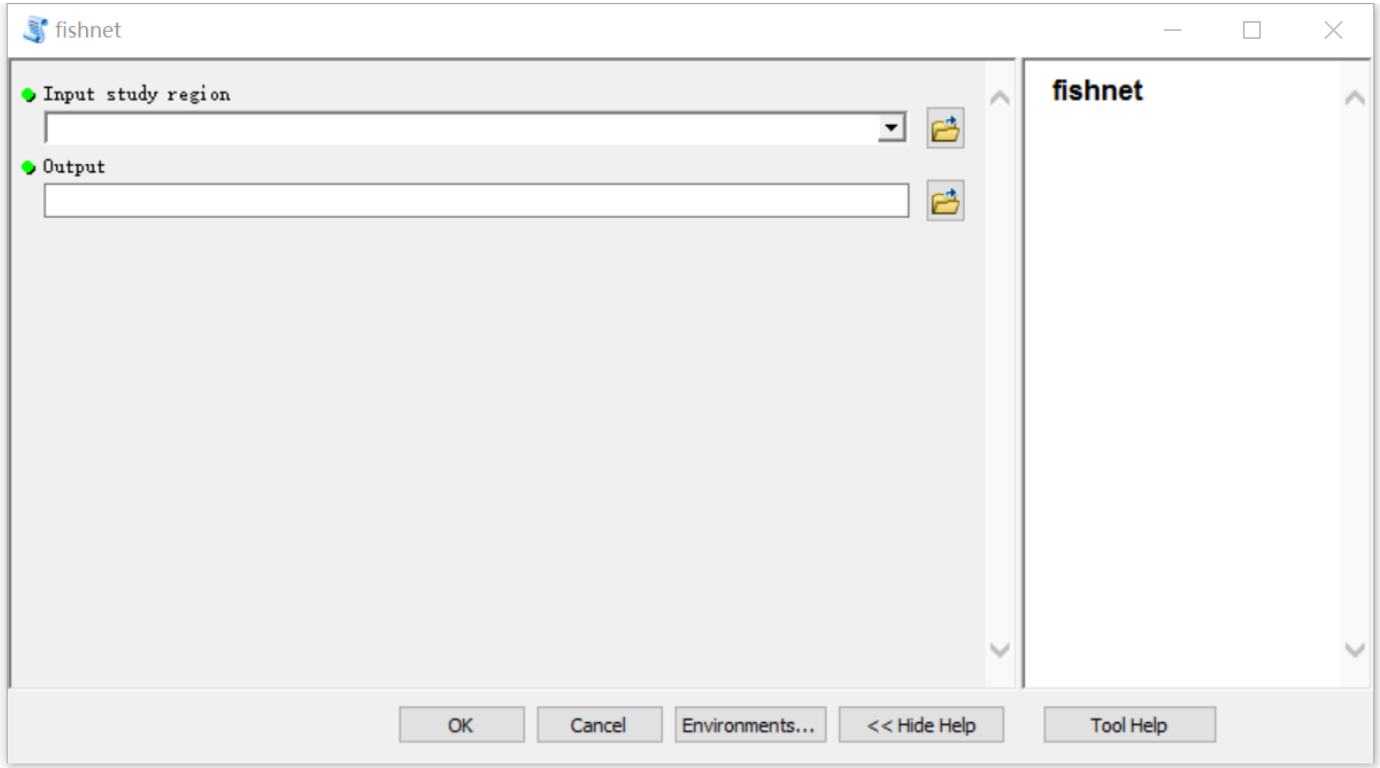


Tree coverage



Non-tree vegetation coverage

ArcPy Tool1: Create a fishnet of the study region



Step1:

Using the tool Create Fishnet in ArcGIS. The output will be a rectangular shapefile with the extend of my study region. In this case, the extend is specified manually in ArcPy script, and the cell size is 2500*2500 meters.

arcpy.CreateFishnet_management

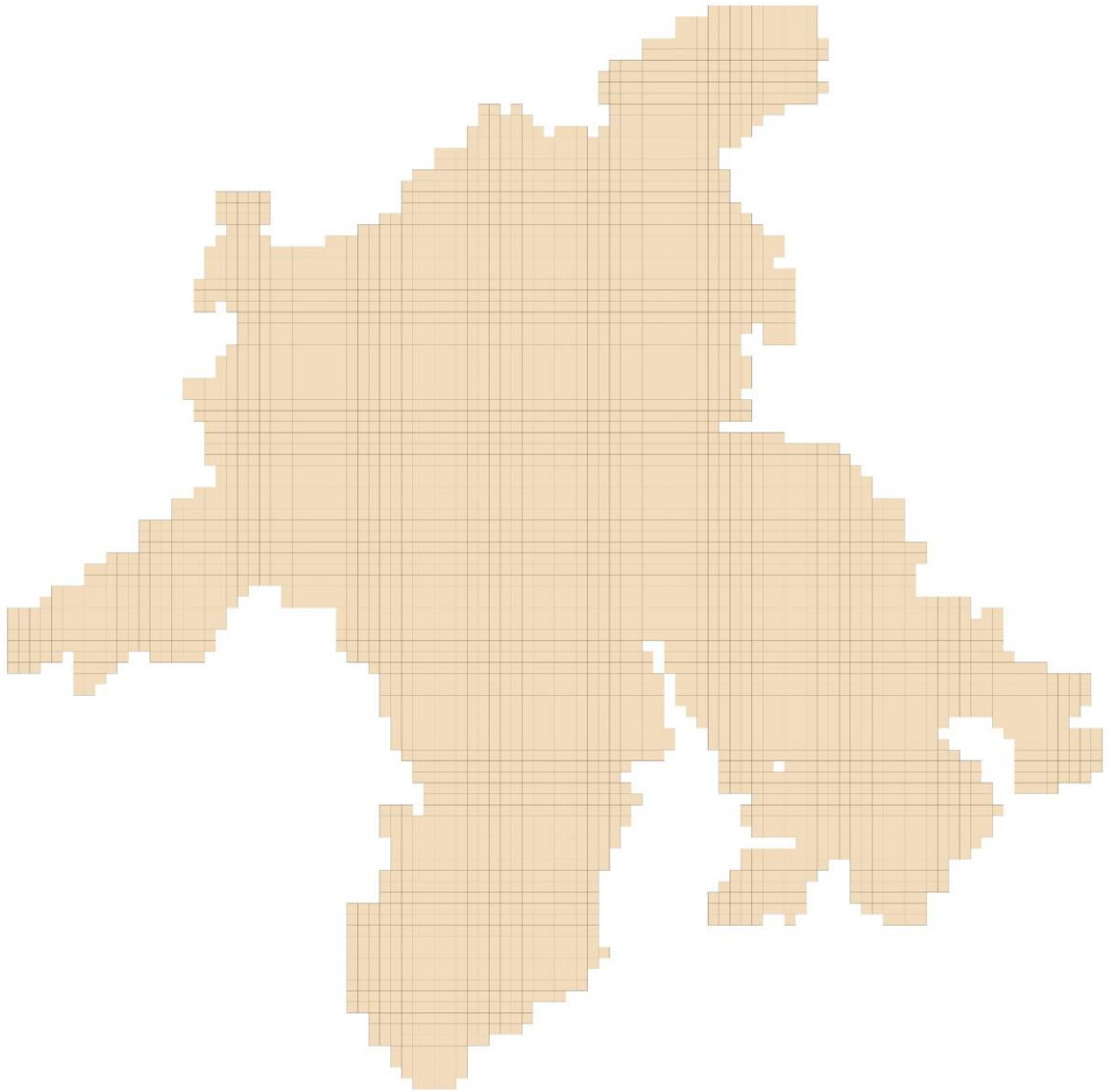
Step2:

Getting the shape of the study region using Select by Location to select the grid cells from the fishnet which fall into the study region.

arcpy.SelectLayerByLocation_management

ArcPy Tool1 Output: Fishnet of the study region

Input

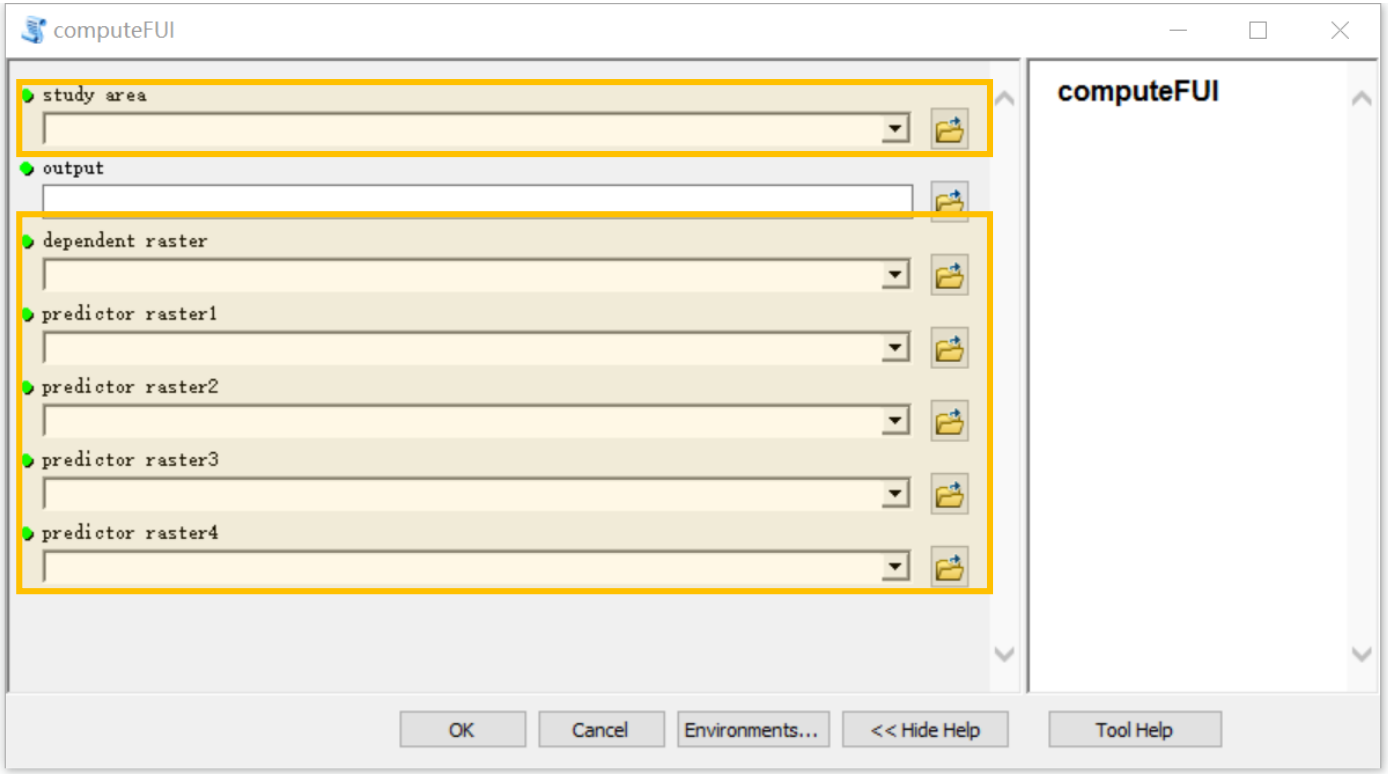


Output

ArcPy Tool1 Code

```
# Import external modules
import sys, os, string, math, arcpy, traceback, numpy
from arcpy import env

# set spatial reference
env.outputCoordinateSystem = arcpy.SpatialReference("WGS 1984 World Mercator")

# Allow output to overwite any existing grid of the same name
arcpy.env.overwriteOutput = True

try:
    # read input
    inputfeature = arcpy.GetParameterAsText(0)
    output = arcpy.GetParameterAsText(1)

    originCoordinate = "12510695.68633178 2480781.535645782"
    yAxisCoordinate = "12510695.68633178 2480791.535645782"
    oppositeCoorner = "12759882.01763127 2728238.175390986"
    root = "E:/Study-18Fall/GeospaticialSoftware/EEProject_Data/shp_arcpy/"
    fishnet = root + "_fishnetboundary.shp"

    arcpy.CreateFishnet_management(fishnet, originCoordinate, yAxisCoordinate, "2500",
"2500", "0", "0", oppositeCoorner, "NO_LABELS", "#", "POLYGON")

    arcpy.MakeFeatureLayer_management(fishnet, "fishnet_lyr")
    arcpy.SelectLayerByLocation_management("fishnet_lyr", "INTERSECT", inputfeature)
    arcpy.CopyFeatures_management("fishnet_lyr", output)

    arcpy.Delete_management(fishnet)


except Exception as e:
    # If unsuccessful, end gracefully by indicating why
    arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
    # ... and where
    exceptionreport = sys.exc_info()[2]
    fullermessage   = traceback.format_tb(exceptionreport)[0]
    arcpy.AddError("at this location: \n\n" + fullermessage + "\n")
```

ArcPy Tool2: Run GWR and compute FUI



Step1:

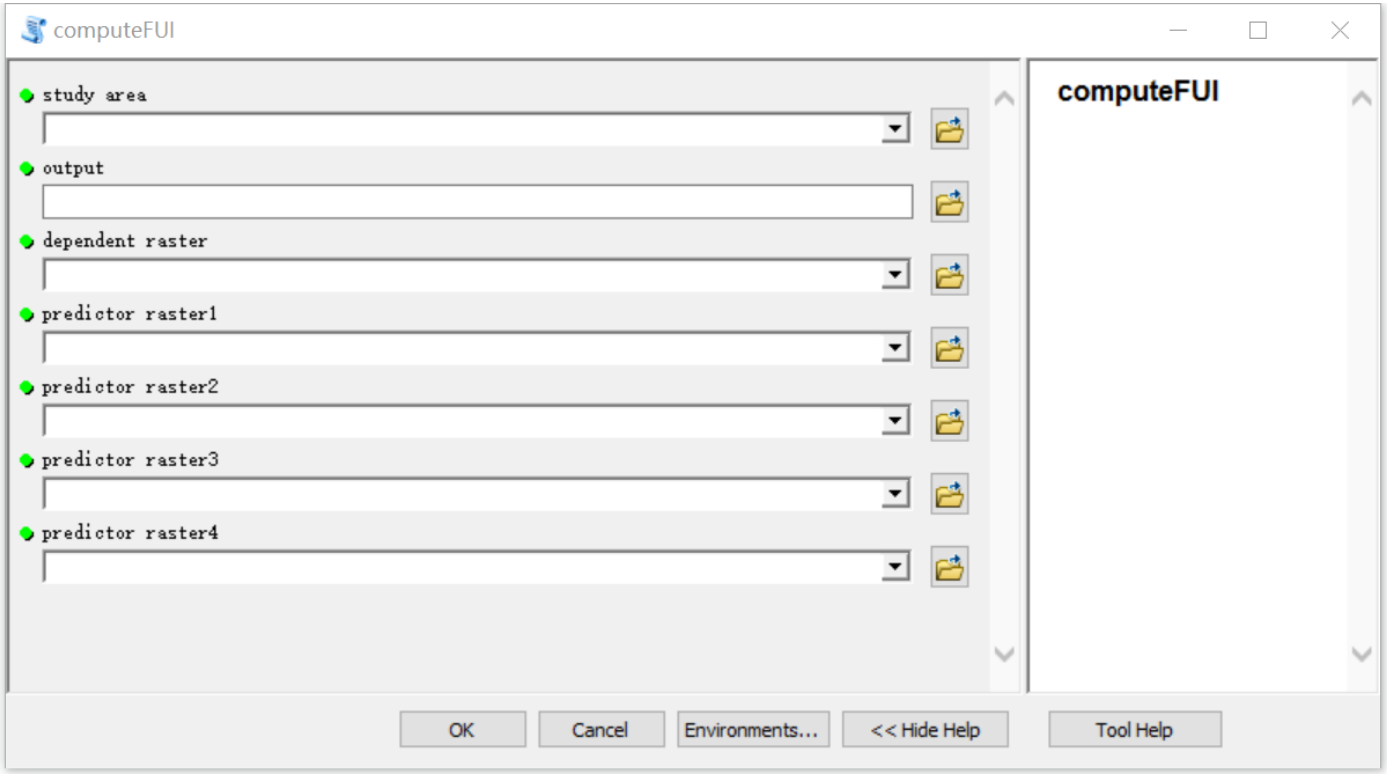
Using the tool Raster to Point to convert values of each grid cell from raster layers into point shapefiles.

The input data will be:

1. Fishnet of the study region obtained from Tool1
2. Raster layer of dependent variable. In this case, since I use the nightlight change image as an indicator of urban growth, this will be my dependent variable
3. Raster layers of the four predictors (i.e. the Decision Factors of FUI)

arcpy.RasterToPoint_conversion

ArcPy Tool2: Run GWR and compute FUI



Step2:

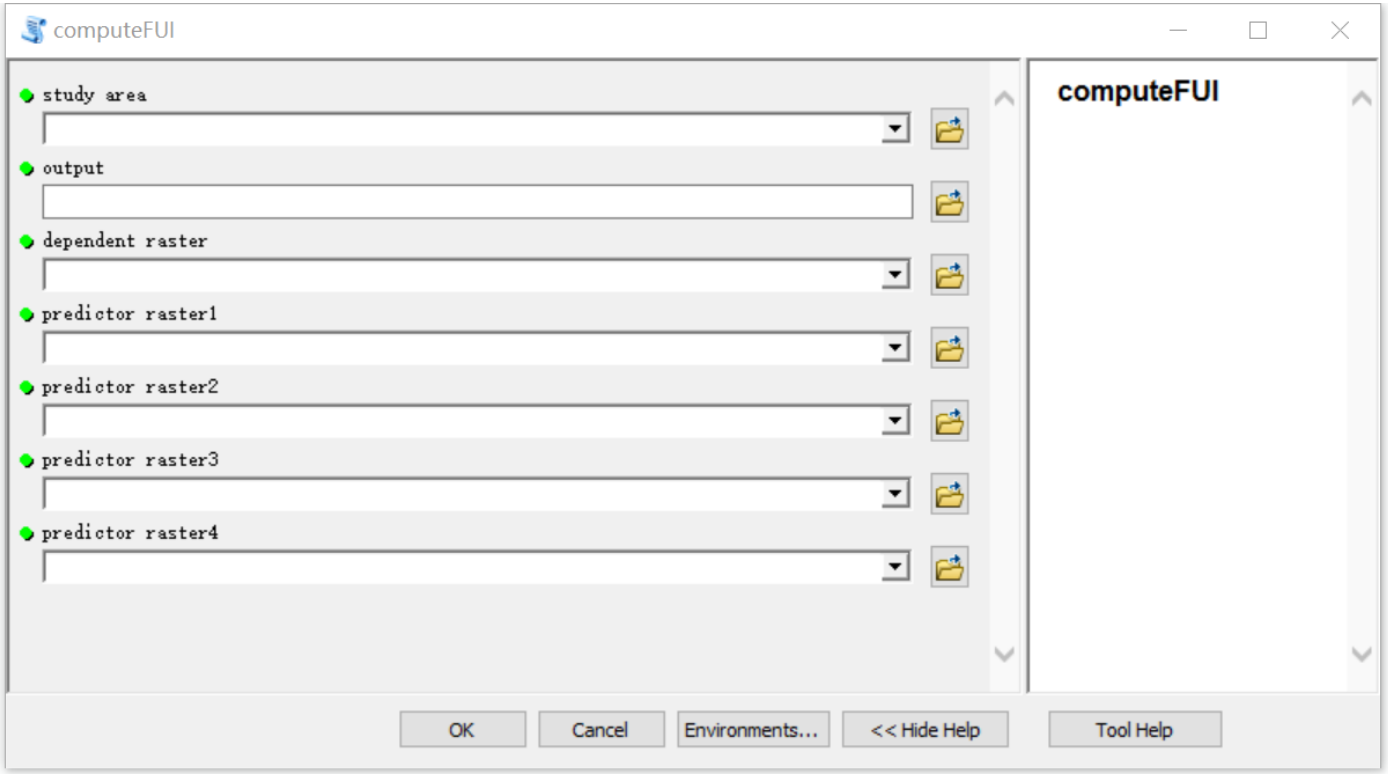
Define a function at the beginning of the script to combine several spatial joins. Each point shapefile (converted from the rasters above) will be joined to the fishnet of study region by averaging the values of points that fall within each grid cell.

This step will generate a final spatial join fishnet with all the average values of the dependent variable and predictors.

def SpatialJoinFUI(shapefile that will be joined to, dependent variable, predictor1, predictor2, predictor3, predictor4, final spatial join result)

arcpy.SpatialJoin_analysis

ArcPy Tool2: Run GWR and compute FUI



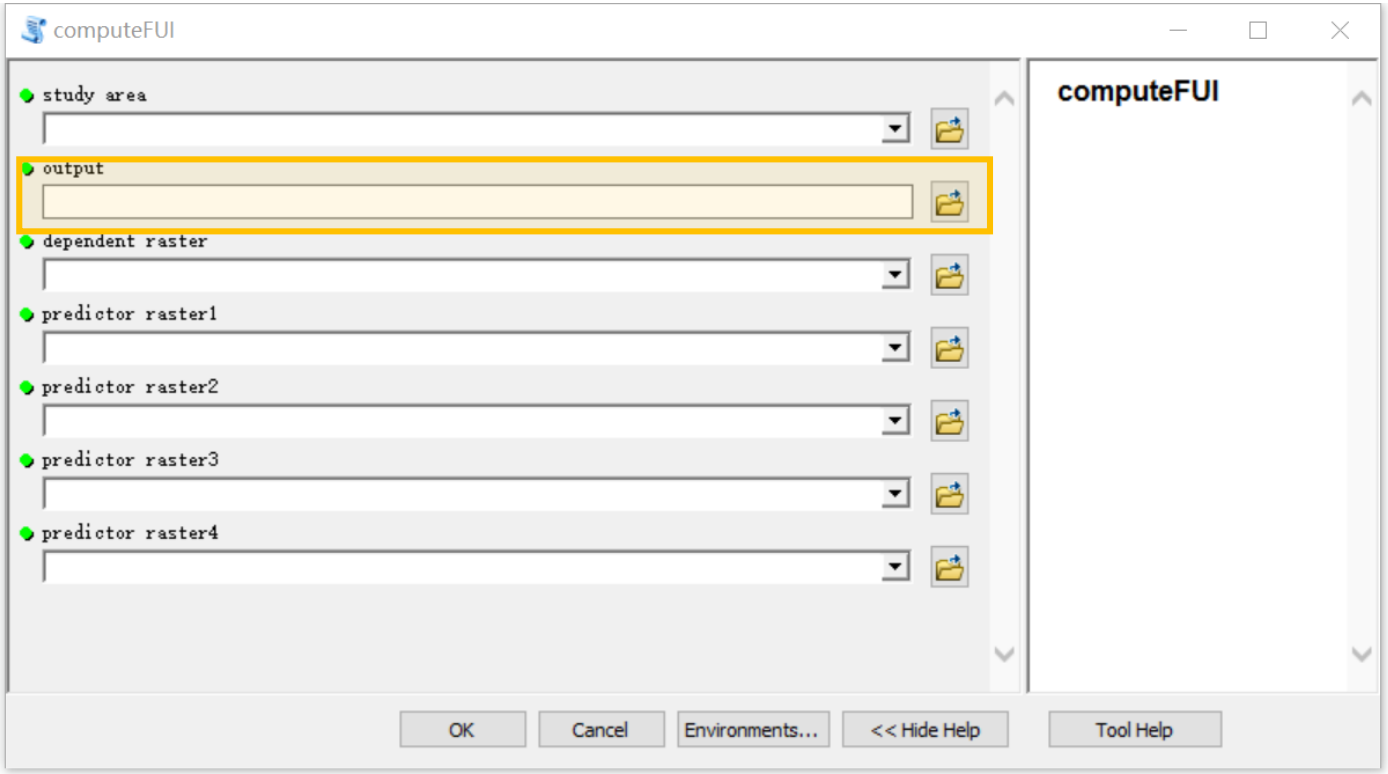
Step3:
Run the GWR on each cell in the fishnet shapefile.

This step will generate a GWR result shapefile that contains the observed and predicted values of the dependent variable, values and coefficients of the predictors, residuals and standardized residuals, etc.

A GWR_supp dbf file will also be created automatically by the GWR tool in ArcGIS, which contains the global statistics of the regression.

arcpy.GeographicallyWeightedRegression_stats

ArcPy Tool2: Run GWR and compute FUI



Step4:
Reclassify the predicted values of nightlight changes into four categories by equal intervals.

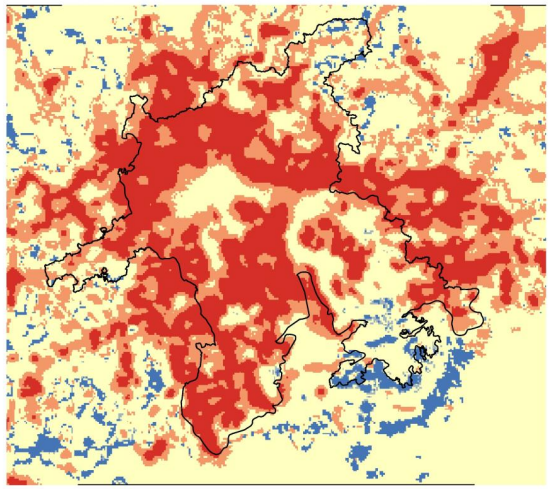
Then delete the unnecessary fields which are included in the GWR result shapefile.

arcpy.UpdateCursor

arcpy.DeleteField_management

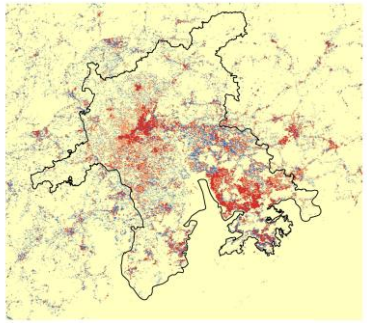
ArcPy Tool2 Output: Spatial join result

Input

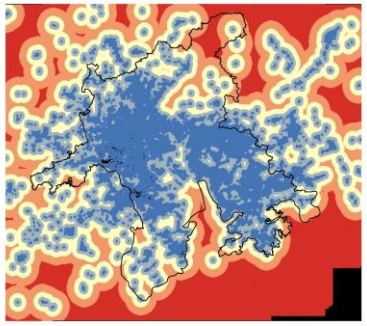


Nightlight change (2000-2013)

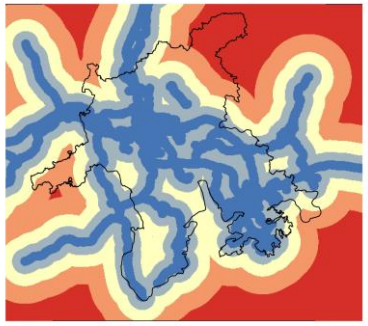
Population change



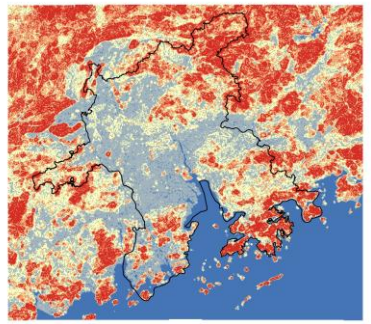
Distance to urban land 2005



Distance to railroad



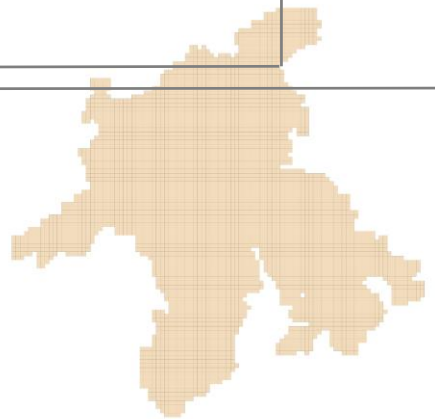
Slope



Table

spaResult_FUI

	Id	pointid	DV	pre1	pre2	pre3	pre4
	0	211439	30. 75	5. 11832	3625. 296103	2961. 066555	5. 31522
	0	211444	10. 3	0	6073. 018842	5029. 415206	. 387583
	0	210144	28. 42	0	3868. 979158	2990. 002904	. 183134
	0	210149	45. 46	1. 01118	1941. 173461	937. 961529	4. 0087
	0	210154	44. 44	-1. 4619	1941. 173461	841. 755083	9. 22927
	0	210159	25. 86	. 146879	3873. 894248	2591. 00741	7. 05791



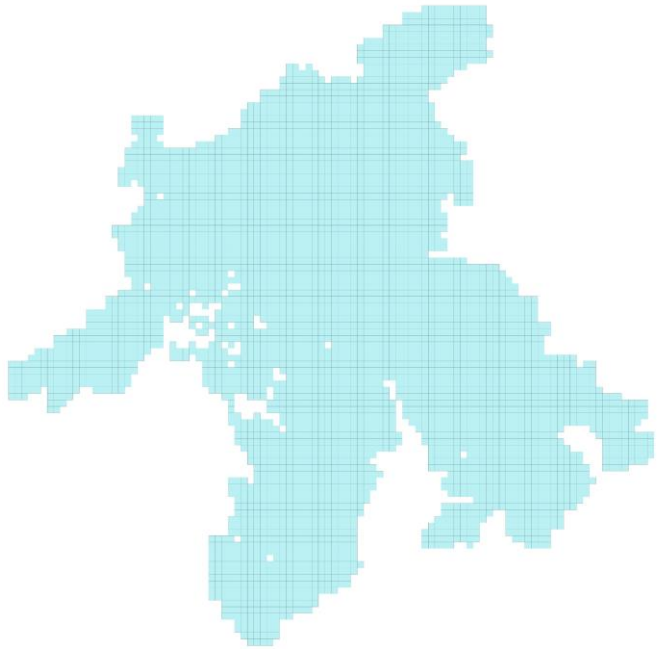
Output

ArcPy Tool2 Output: GWR result of FUI

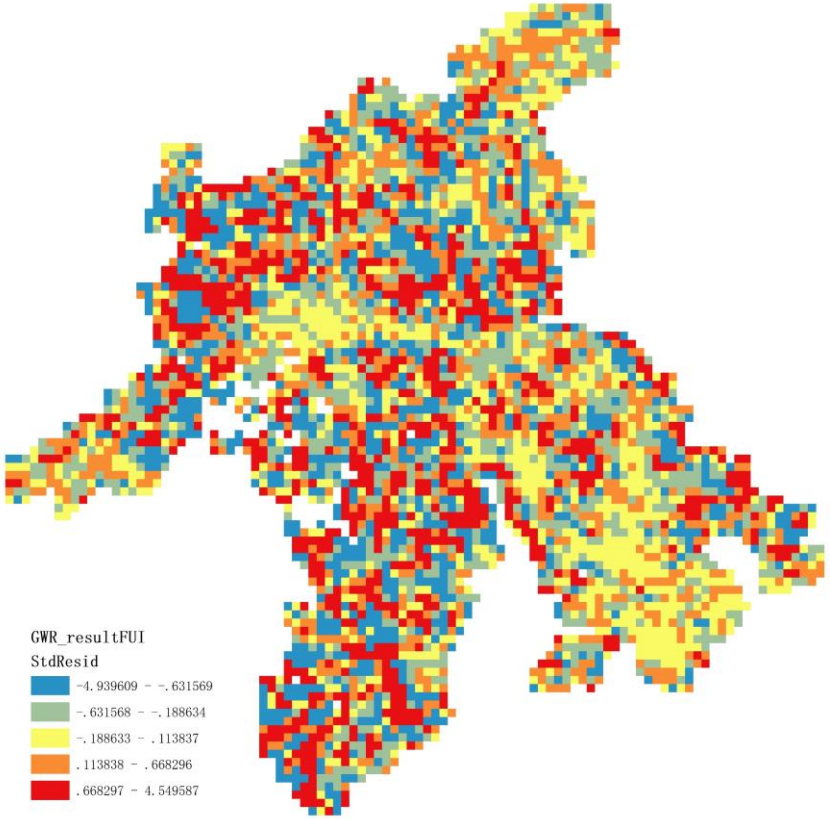
Table

GWR_resultFUI

Observed	Cond	LocalR2	Predicted	Intercept	C1_pre1	C2_pre2	C3_pre3	C4_pre4	Residual	StdError	StdErr_Int	StdErrC1_p	StdErr
32.5	6.576542	.817229	34.888251	53.925808	-.22061	-.001457	-.006839	-.695076	-2.388251	3.287592	2.212726	.301502	
40.25	7.409288	.843549	39.917137	53.463414	-.307759	-.001959	-.005757	-.67478	.332863	3.206894	2.261011	.329459	
30.75	8.588223	.855269	24.696298	53.123743	-.352577	-.002492	-.004818	-.62524	6.053702	2.706696	2.425038	.359992	
10.3	9.88255	.822340	13.532948	50.479455	-.326862	-.002666	-.004097	-.394478	-3.232948	3.640799	2.615674	.380830	
28.42	6.578207	.759355	25.907097	55.322061	-.165916	-.001089	-.008381	-.778019	2.512903	3.624357	2.362475	.263957	
45.46	6.155205	.811162	41.957965	55.133771	-.236586	-.00151	-.007296	-.788957	3.502035	3.481665	2.169865	.286509	
44.44	7.021819	.846629	38.862695	54.486552	-.361	-.002109	-.005833	-.774546	5.577305	3.292698	2.188806	.318776	
25.86	8.588896	.860131	26.200717	54.374135	-.408226	-.002771	-.004671	-.747729	-.340717	3.581906	2.398454	.353643	
7.96	10.191693	.808652	13.415503	50.073864	-.380792	-.00279	-.003867	-.4104	-5.455503	3.752347	2.723753	.380802	
5.64	11.615206	.674387	.078890	35.320388	.076795	-.001782	-.002808	.714038	5.56111	3.388275	3.00469	.376292	
11.34	6.664935	.654236	8.497542	49.616809	.152182	-.000553	-.007751	-.361245	2.842458	2.913964	2.362245	.208838	
31.66	6.60388	.695800	25.068608	53.548541	-.08724	-.000807	-.008329	-.776535	6.591392	3.639996	2.365814	.240099	
45.56	6.117903	.799948	40.257841	56.824568	-.264295	-.001582	-.008013	-.937159	5.302159	1.989269	2.199219	.271987	
31.18	7.038235	.856873	31.408455	56.443688	-.46464	-.002387	-.005969	-.956982	-.228455	3.068914	2.1329	.309475	
9.98	9.511708	.868502	18.398467	57.116938	-.528058	-.003276	-.004475	-.998333	-8.418467	3.360964	2.430131	.348207	
4.74	11.288436	.754617	8.974127	46.892105	-.447598	-.002474	-.003794	-.302157	-4.234127	3.756609	2.854594	.364825	
4.78	10.952963	.578653	.142029	24.49808	.532850	-.000467	-.002755	1.218705	4.637971	3.338038	3.01154	.328197	
14.175	6.653934	.586843	20.086279	45.295986	.181555	-.000268	-.006896	.053716	-5.911279	3.255286	2.476032	.109180	
35.825	6.349378	.607474	31.993711	48.364619	.317660	-.000454	-.00719	-.349799	3.831289	3.516241	2.309701	.213694	
46.85	5.931833	.723208	41.738794	53.604731	.026004	-.001542	-.00693	-.743547	5.111206	3.320623	2.113089	.230227	
31.25	8.246352	.883717	31.956134	60.674691	-.663823	-.003197	-.005679	-1.297435	-.706134	3.078523	2.439145	.318472	
9.325	12.682817	.884218	14.253487	63.447985	-.902346	-.004261	-.004249	-1.522172	-4.928487	3.449836	3.228947	.389987	
4.2	12.1452	.622840	6.249383	36.617018	.044461	-.001299	-.003647	.117351	-2.049383	3.56117	3.165917	.317033	
4.625	10.565534	.536597	3.197605	11.094531	.974354	.000255	-.001423	1.561883	1.427395	3.471629	3.6175	.403993	
6.32	5.508933	.588602	8.340297	41.583675	-.029865	-.00213	-.004553	.876292	-2.020297	3.047826	1.854573	.040572	
13.92	5.893219	.542076	21.049452	45.408872	-.021737	-.002342	-.005235	.319263	-7.129452	3.555348	2.333983	.044078	
18.32	6.241069	.436142	34.288163	43.581392	.007911	-.001315	-.005049	.694236	-15.968163	3.232678	2.301194	.049223	
36.04	5.814102	.466959	40.839381	44.186651	.164831	-.000935	-.004846	.330084	-4.799381	3.435414	2.047101	.077463	
45.36	6.429602	.656844	36.8473	49.639025	.427031	-.001787	-.004912	-.394642	8.5127	3.398005	2.265834	.212751	
22.92	8.835238	.857172	24.163079	61.345585	-.238642	-.004104	-.004028	-1.195057	-1.243079	3.501787	2.745605	.246163	
7.06	12.091912	.836702	7.922962	62.819167	-.41331	-.004493	-.003816	-1.425435	-.862962	3.246687	3.564079	.344412	
3.74	12.012926	.484459	4.413095	25.623609	2.326317	-.000817	-.002355	.412220	-.673095	3.464113	3.69592	.348713	
3.2	9.302056	.485526	5.536899	9.872248	1.05777	-.000489	-.000165	1.388025	-2.336899	3.6961	3.129476	.277238	
2.32	10.937545	.694764	4.686103	3.210668	.196516	-.00121	.001718	1.694313	-2.366103	3.632612	3.091354	.283456	

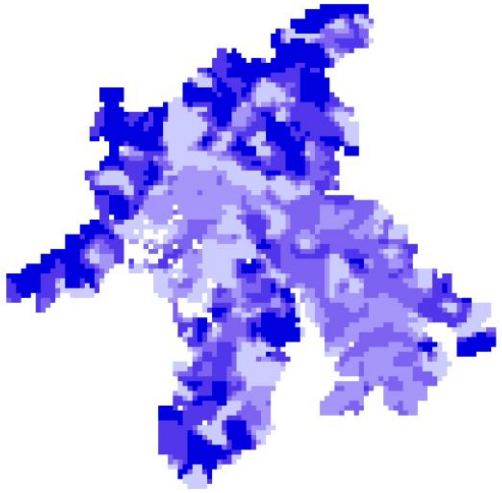


ArcPy Tool2 Output: GWR result of FUI

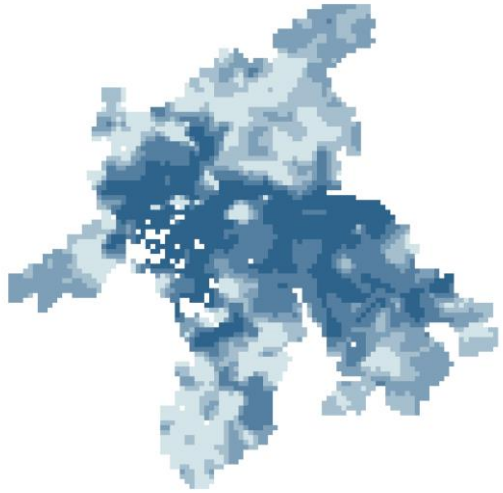


Standardized Residuals

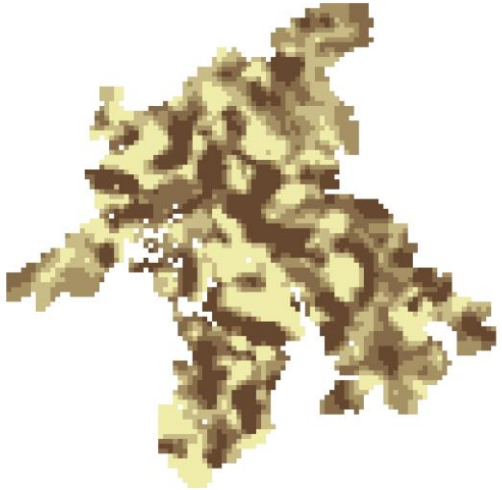
Coefficients of predictors



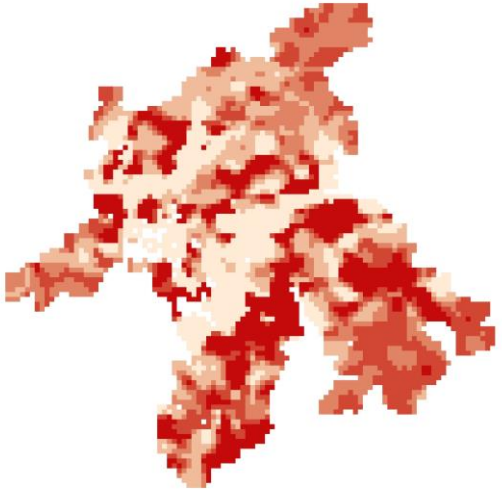
Population change (2000-2015)



Distance to existing urban land in 2005

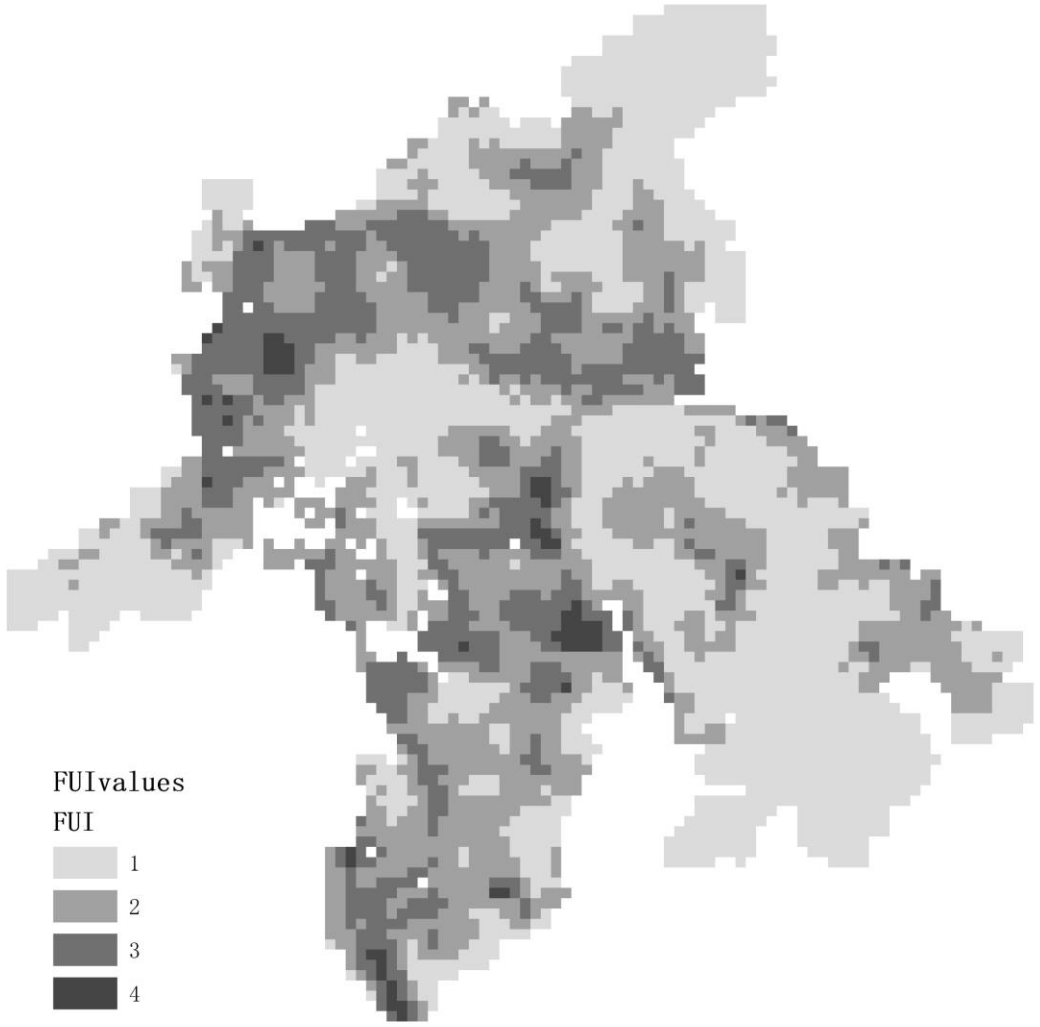


Distance to railroad



Slope

ArcPy Tool2 Output: Reclassified FUI values (predicted nightlight changes)



Table

FUIvalues

FID	Shape *	Source_ID	FUI
0	Polygon	0	3
1	Polygon	1	4
2	Polygon	2	3
3	Polygon	3	2
4	Polygon	4	3
5	Polygon	5	4
6	Polygon	6	4
7	Polygon	7	3
8	Polygon	8	2
9	Polygon	9	1
10	Polygon	10	1
11	Polygon	11	3
12	Polygon	12	4
13	Polygon	13	3
14	Polygon	14	2
15	Polygon	15	1
16	Polygon	16	1
17	Polygon	17	2
18	Polygon	18	3
19	Polygon	19	4
20	Polygon	20	3
21	Polygon	21	2
22	Polygon	22	1
23	Polygon	23	1
24	Polygon	24	1
25	Polygon	25	2
26	Polygon	26	3
27	Polygon	27	4
28	Polygon	28	3
29	Polygon	29	3
30	Polygon	30	1
31	Polygon	31	1
32	Polygon	32	1

1

(0 out of 4220 Selected)

FUIvalues

ArcPy Tool2 Code

```
# Import external modules
import sys, os, string, math, arcpy, traceback, numpy
from arcpy import env

# Allow output to overwite any existing grid of the same name
arcpy.env.overwriteOutput = True


root = "E:/Study-18Fall/GeospaticialSoftware/EEProject_Data/shp_arcpy/"

def SpatialJoinFUI(target, dependent, predictor1, predictor2, predictor3, predictor4, output):
        inter1 = root + "inter1.shp"
        inter2 = root + "inter2.shp"
        inter3 = root + "inter3.shp"
        inter4 = root + "inter4.shp"

        # spatial join dependent variable
        fieldmappings1 = arcpy.FieldMappings()
        fieldmappings1.addTable(target)
        fieldmappings1.addTable(dependent)

        gridValue1 = fieldmappings1.findFieldMapIndex("grid_code")
        fieldmap1 = fieldmappings1.getFieldMap(gridValue1)

        field1 = fieldmap1.outputField
        field1.name = "DV"
        fieldmap1.outputField = field1

        fieldmap1.mergeRule = "mean"
        fieldmappings1.replaceFieldMap(gridValue1, fieldmap1)

        arcpy.SpatialJoin_analysis(target, dependent, inter1, "JOIN_ONE_TO_ONE", "KEEP_ALL", fieldmappings1, "CONTAINS")
```

ArcPy Tool2 Code

```
# spatial join the first predictor
    fieldmappings2 = arcpy.FieldMappings()
    fieldmappings2.addTable(inter1)
    fieldmappings2.addTable(predictor1)

    gridValue2 = fieldmappings2.findFieldMapIndex("grid_code")
    fieldmap2 = fieldmappings2.getFieldMap(gridValue2)

    field2 = fieldmap2.outputField
    field2.name = "pre1"
    fieldmap2.outputField = field2

    fieldmap2.mergeRule = "mean"
    fieldmappings2.replaceFieldMap(gridValue2, fieldmap2)

    arcpy.SpatialJoin_analysis(inter1, predictor1, inter2, "JOIN_ONE_TO_ONE", "KEEP_ALL", fieldmappings2, "CONTAINS")


    # spatial join the second predictor
    fieldmappings3 = arcpy.FieldMappings()
    fieldmappings3.addTable(inter2)
    fieldmappings3.addTable(predictor2)

    gridValue3 = fieldmappings3.findFieldMapIndex("grid_code")
    fieldmap3 = fieldmappings3.getFieldMap(gridValue3)

    field3 = fieldmap3.outputField
    field3.name = "pre2"
    fieldmap3.outputField = field3

    fieldmap3.mergeRule = "mean"
    fieldmappings3.replaceFieldMap(gridValue3, fieldmap3)

    arcpy.SpatialJoin_analysis(inter2, predictor2, inter3, "JOIN_ONE_TO_ONE", "KEEP_ALL", fieldmappings3, "CONTAINS")
```

ArcPy Tool2 Code

```
# spatial join the third predictor
    fieldmappings4 = arcpy.FieldMappings()
    fieldmappings4.addTable(inter3)
    fieldmappings4.addTable(predictor3)

    gridValue4 = fieldmappings4.findFieldMapIndex("grid_code")
    fieldmap4 = fieldmappings4.getFieldMap(gridValue4)

    field4 = fieldmap4.outputField
    field4.name = "pre3"
    fieldmap4.outputField = field4

    fieldmap4.mergeRule = "mean"
    fieldmappings4.replaceFieldMap(gridValue4, fieldmap4)

    arcpy.SpatialJoin_analysis(inter3, predictor3, inter4, "JOIN_ONE_TO_ONE", "KEEP_ALL", fieldmappings4, "CONTAINS")


    # spatial join the fourth predictor
    fieldmappings5 = arcpy.FieldMappings()
    fieldmappings5.addTable(inter4)
    fieldmappings5.addTable(predictor4)

    gridValue5 = fieldmappings5.findFieldMapIndex("grid_code")
    fieldmap5 = fieldmappings5.getFieldMap(gridValue5)

    field5 = fieldmap5.outputField
    field5.name = "pre4"
    fieldmap5.outputField = field5

    fieldmap5.mergeRule = "mean"
    fieldmappings5.replaceFieldMap(gridValue5, fieldmap5)

    arcpy.SpatialJoin_analysis(inter4, predictor4, output, "JOIN_ONE_TO_ONE", "KEEP_ALL", fieldmappings5, "CONTAINS")
```

ArcPy Tool2 Code

```
 # Delete the intermediate shapefile
        arcpy.Delete_management(inter1)
        arcpy.Delete_management(inter2)
        arcpy.Delete_management(inter3)
        arcpy.Delete_management(inter4)

        return output



# If Spatial Analyst license is available, check it out
if arcpy.CheckExtension("spatial") == "Available":
    arcpy.CheckOutExtension("spatial")

    try:
        # read input
        inboundary = arcpy.GetParameterAsText(0)
        output = arcpy.GetParameterAsText(1)
        inraster1 = arcpy.GetParameterAsText(2)
        inraster2 = arcpy.GetParameterAsText(3)
        inraster3 = arcpy.GetParameterAsText(4)
        inraster4 = arcpy.GetParameterAsText(5)
        inraster5 = arcpy.GetParameterAsText(6)

        # convert raster to points
        rasterPoint1 = root + inraster1[:-4] + "_temp.shp"
        rasterPoint2 = root + inraster2[:-4] + "_temp.shp"
        rasterPoint3 = root + inraster3[:-4] + "_temp.shp"
        rasterPoint4 = root + inraster4[:-4] + "_temp.shp"
        rasterPoint5 = root + inraster5[:-4] + "_temp.shp
```

ArcPy Tool2 Code

```
arcpy.RasterToPoint_conversion(inraster1, rasterPoint1, "VALUE")
arcpy.RasterToPoint_conversion(inraster2, rasterPoint2, "VALUE")
arcpy.RasterToPoint_conversion(inraster3, rasterPoint3, "VALUE")
arcpy.RasterToPoint_conversion(inraster4, rasterPoint4, "VALUE")
arcpy.RasterToPoint_conversion(inraster5, rasterPoint5, "VALUE")


# spatial join points to fishnet and get the average values
spatialJoin_result = root + "spaResult_FUI.shp"
SpatialJoinFUI(inboundary, rasterPoint1, rasterPoint2, rasterPoint3, rasterPoint4, rasterPoint5, spatialJoin_result)



# run GWR
GWR_result = root + "GWR_resultFUI.shp"
predictor = "pre1;pre2;pre3;pre4"
arcpy.GeographicallyWeightedRegression_stats(spatialJoin_result, "DV", predictor, GWR_result, "ADAPTIVE", "BANDWIDTH PARAMETER","#", "40", "#", "#", "#", "#", "#", "#")


# reclassify GWR predicted value and get the FUI classification
# Replicate the GWR result
GWR_reclassify = output
arcpy.Copy_management(GWR_result, GWR_reclassify)
arcpy.AddField_management(GWR_reclassify, "FUI", "DOUBLE", 20, 5)

# get the min and max predicted values
GWR_min = GWR_reclassify[:-4] + "_min.shp"
GWR_max = GWR_reclassify[:-4] + "_max.shp"

arcpy.Sort_management(GWR_reclassify,GWR_min,[["Predicted", "ASCENDING"]])
arcpy.Sort_management(GWR_reclassify,GWR_max,[["Predicted", "DESCENDING"]])

enumeration_min = arcpy.SearchCursor(GWR_min)
minValue = enumeration_min.next().getValue("Predicted")
arcpy.AddMessage("Minimum predicted value:  " + str(minValue) + "\n")
```

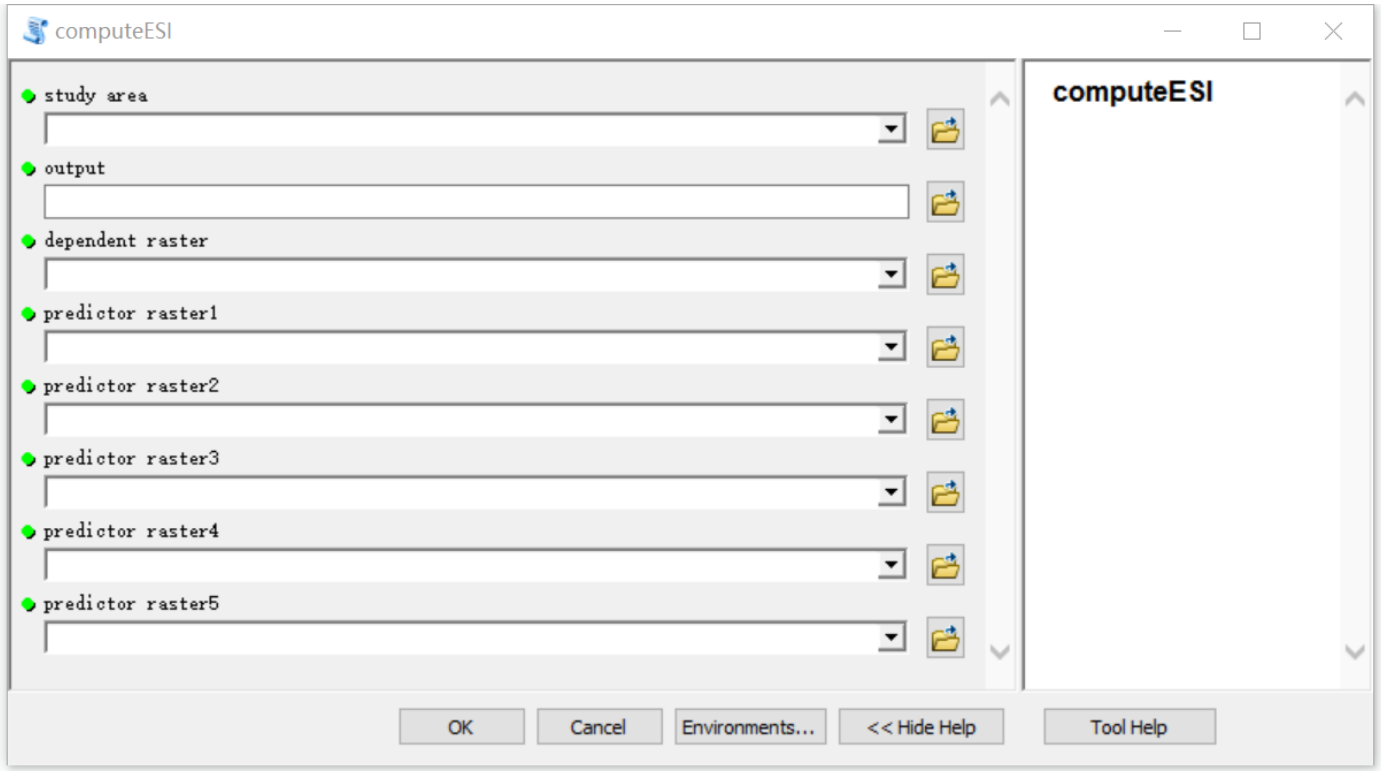
ArcPy Tool2 Code

```
enumeration_max = arcpy.SearchCursor(GWR_max)
maxValue = enumeration_max.next().getValue("Predicted")
arcpy.AddMessage("Maximum predicted value:  " + str(maxValue) + "\n")

increase = (maxValue - minValue)/4
first = increase + minValue
second = increase*2 + minValue
third = increase*3 + minValue

del enumeration_min
del enumeration_max

# Delete the intermediate shapefile
arcpy.Delete_management(GWR_min)
arcpy.Delete_management(GWR_max)

# reclassify
enumeration = arcpy.UpdateCursor(GWR_reclassify)

for row in enumeration:
    predicted = row.getValue("Predicted")
    if predicted < first:
        row.setValue("FUI", 1)
    elif predicted >= first and predicted < second:
        row.setValue("FUI", 2)
    elif predicted >= second and predicted < third:
        row.setValue("FUI", 3)
    else:
        row.setValue("FUI", 4)

    enumeration.updateRow(row)
```

ArcPy Tool2 Code

```
        # delete unnecessary fields
        dropFieldFUI = ["Observed", "Cond", "LocalR2", "Predicted", "Intercept", "C1_pre1", "C2_pre2", "C3_pre3", "C4_pre4",
                        "Residual", "StdError", "StdErr_Int", "StdErrC1_p", "StdErrC2_p", "StdErrC3_p", "StdErrC4_p", "StdResid"]
        arcpy.DeleteField_management(GWR_reclassify, dropFieldFUI)


        del row
        del enumeration

        # Delete the intermediate shapefile
        arcpy.Delete_management(rasterPoint1)
        arcpy.Delete_management(rasterPoint2)
        arcpy.Delete_management(rasterPoint3)
        arcpy.Delete_management(rasterPoint4)
        arcpy.Delete_management(rasterPoint5)


    except Exception as e:
        # If unsuccessful, end gracefully by indicating why
        arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
        # ... and where
        exceptionreport = sys.exc_info()[2]
        fullermessage   = traceback.format_tb(exceptionreport)[0]
        arcpy.AddError("at this location: \n\n" + fullermessage + "\n")

    # Check in Spatial Analyst extension license
    arcpy.CheckInExtension("spatial")
else:
    print "Spatial Analyst license is " + arcpy.CheckExtension("spatial")
```

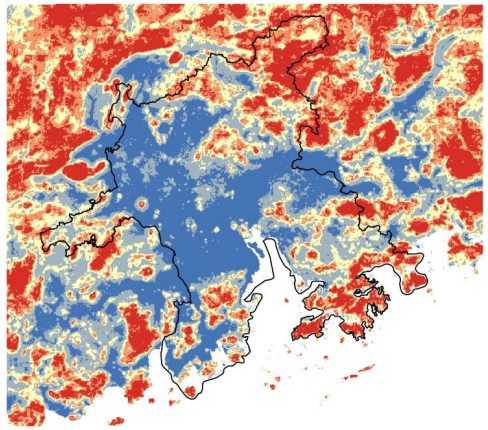
ArcPy Tool3: Run GWR and compute ESI



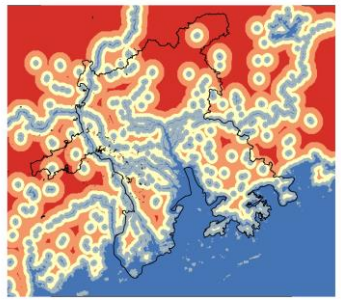
This tool is basically the same as the second tool, except that there is one more predictor included here to run the regression and compute the ESI, since I use five Decision Factors here.

ArcPy Tool3 Output: Spatial join result

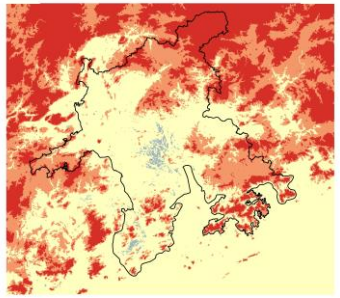
Input



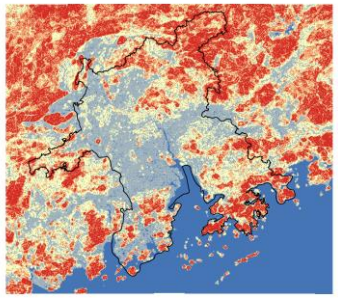
ALOS Topographic diversity



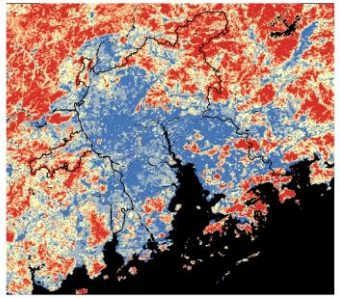
Distance to water



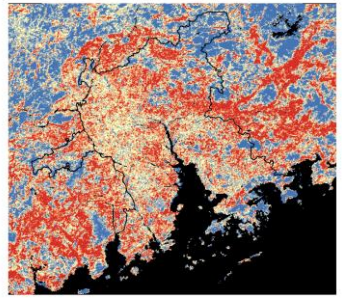
Elevation



Slope



Tree coverage

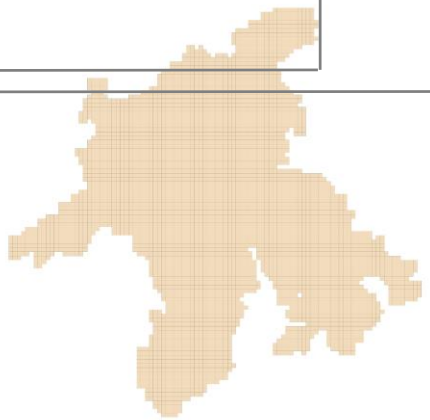


Non-tree vegetation

Table

spaResult_ESI

	Id	pointid	DV	pre1	pre2	pre3	pre4	pre5
▶	0	211429	.617586	<Null>	0	.372030	<Null>	<Null>
	0	211434	.518197	<Null>	14	2.90525	<Null>	<Null>
	0	211439	.622576	235.355339	39	5.31522	<Null>	<Null>
	0	211444	.622226	<Null>	0	.387583	<Null>	<Null>
	0	210144	.462465	<Null>	0	.183134	<Null>	<Null>
	0	210149	.567380	449.574173	34	4.0087	<Null>	<Null>
	0	210154	.657526	1351.845093	176	9.22927	36.44	51.4



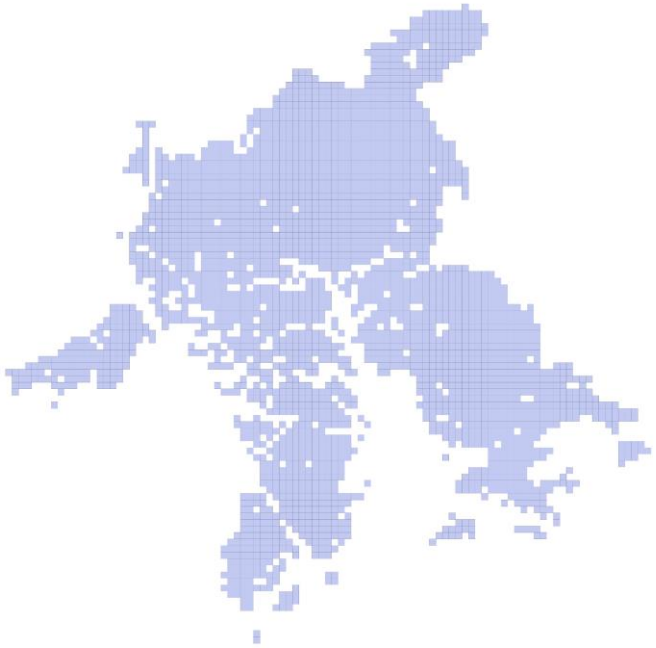
Output

ArcPy Tool3 Output: GWR result of ESI

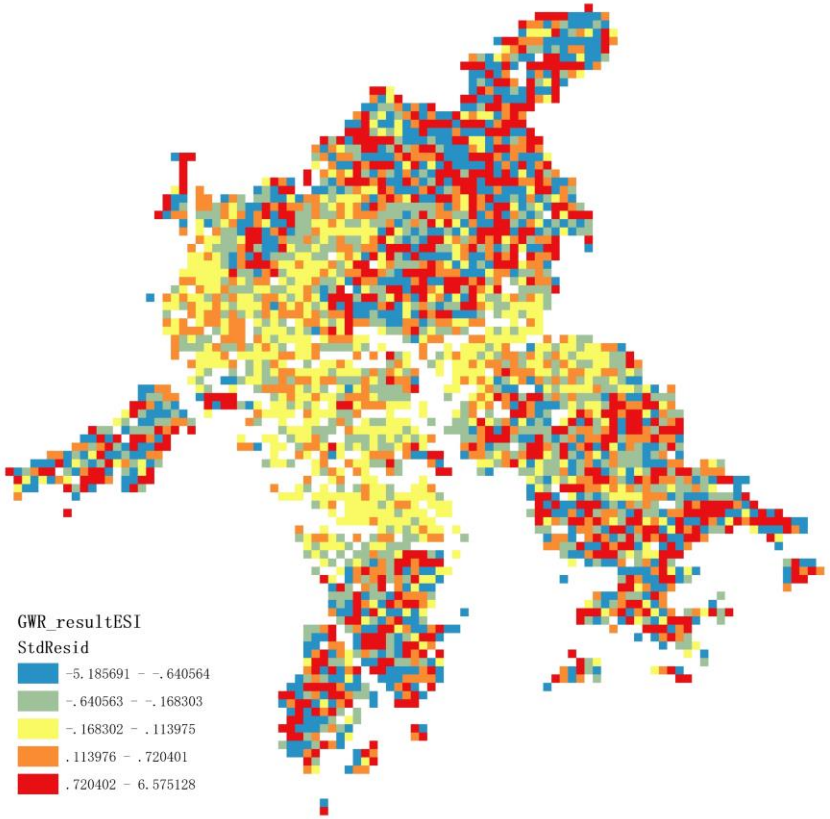
Table

GWR_resultESI

FID	Shape *	Observed	Cond	LocalR2	Predicted	Intercept	C1_pre1	C2_pre2	C3_pre3	C4_pre4	C5_pre5	Residual	StdError	StdErr_1
0	Polygon	.657526	43.540094	.985731	.641616	-.091238	-.000003	-.001167	.089957	-.000753	.002709	.015910	.005702	.0
1	Polygon	.709768	42.440532	.985618	.745006	-.087788	-.000003	-.001189	.090110	-.000709	.002640	-.035238	.016675	.0
2	Polygon	.329226	30.512104	.976212	.315902	-.022936	-.000004	-.001641	.092390	-.000226	.001349	.013324	.021785	.0
3	Polygon	.471345	33.227533	.981086	.470341	-.043883	-.000005	-.00166	.093265	-.000185	.001722	.001004	.013046	.0
4	Polygon	.374250	43.756341	.980476	.406521	-.136358	.000000	-.001874	.099721	-.000157	.003127	-.032271	.021070	.0
5	Polygon	.528108	47.138358	.977914	.522870	-.160308	.000001	-.001993	.101511	-.000038	.003477	.005238	.018404	.0
6	Polygon	.510535	49.783047	.971600	.504257	-.167967	.000000	-.001862	.098835	.000392	.003551	.006278	.021396	.0
7	Polygon	.086289	28.833578	.966241	.060674	-.032087	.000000	-.002529	.098212	.000930	.000976	.025615	.022357	.0
8	Polygon	.233954	31.584931	.976204	.242184	-.049579	-.000002	-.003276	.107698	.001033	.001262	-.00823	.016821	.0
9	Polygon	.547526	45.382012	.978757	.574081	-.138629	-.000001	-.002869	.111594	.000351	.002926	-.026555	.018679	.0
10	Polygon	.547686	47.250092	.972805	.548045	-.144094	-.000002	-.002446	.106051	.000602	.003045	-.000359	.017002	.0
11	Polygon	.399770	47.513031	.963345	.346099	-.136341	-.000004	-.001927	.096915	.001468	.002877	.053671	.023304	.0
12	Polygon	.023145	25.165525	.946361	.061511	-.030506	.000010	-.002413	.088144	.002432	.000246	-.038366	.023090	.0
13	Polygon	.016056	26.349573	.949758	.048813	-.036685	.000007	-.002694	.093182	.002210	.000475	-.032757	.021658	.0
14	Polygon	.045770	28.022153	.954991	.041852	-.046822	.000004	-.002959	.098203	.001958	.000800	.003917	.022058	.0
15	Polygon	.256846	30.15513	.963680	.218393	-.061387	.000001	-.003195	.103066	.001728	.001200	.038453	.022981	.0
16	Polygon	.402924	42.640085	.979335	.415414	-.123587	-.000002	-.003093	.114677	.000408	.002641	-.01249	.015818	.0
17	Polygon	.414907	45.647243	.974682	.366443	-.139766	-.000004	-.002833	.111748	.000598	.002941	.048464	.020387	.0
18	Polygon	.421220	44.154659	.965546	.415113	-.129285	-.000006	-.002093	.099314	.001622	.002717	.006107	.022682	.0
19	Polygon	.034821	23.707172	.940011	.048173	-.022249	.000014	-.001517	.073787	.002745	-.000046	-.013352	.021263	.0
20	Polygon	.030643	24.436077	.942260	.062136	-.031224	.000013	-.00187	.078402	.002883	.000082	-.031493	.020782	.0
21	Polygon	.034516	25.832266	.943907	.061477	-.044493	.000011	-.002062	.081930	.002833	.000381	-.026961	.022051	.0
22	Polygon	.032649	27.591276	.946630	.052845	-.059697	.000008	-.002211	.085515	.002642	.000774	-.020196	.023400	.0
23	Polygon	.058247	29.627234	.954011	.065790	-.080738	.000003	-.002971	.097074	.002359	.001355	-.007543	.021086	.0
24	Polygon	.163031	30.863937	.963427	.144417	-.088579	.000000	-.003119	.101053	.002069	.001633	.018614	.021069	.0
25	Polygon	.175272	32.097761	.976518	.169210	-.07652	-.000001	-.003312	.108236	.001529	.001522	.006062	.017258	.0
26	Polygon	.360347	34.365814	.981395	.370979	-.077115	-.000001	-.003255	.112834	.000879	.001634	-.010632	.015520	.0
27	Polygon	.230515	41.934496	.967138	.204559	-.12785	-.000008	-.0022	.102098	.001484	.002729	.025956	.019074	.0
28	Polygon	.126559	24.453179	.941978	.101184	-.032044	.000017	-.000835	.062408	.002943	.000030	.025375	.023588	.0
29	Polygon	.166742	25.295987	.943654	.132002	-.042823	.000016	-.001158	.065889	.003242	.000126	.034740	.021850	.0
30	Polygon	.070103	27.717441	.943593	.075009	-.070471	.000013	-.001356	.070866	.003162	.000760	-.004906	.022060	.0
31	Polygon	.030862	30.316059	.945858	.023906	-.105006	.000006	-.001749	.079081	.002762	.001679	.006956	.016326	.0
32	Polygon	.163159	32.172749	.973399	.205968	-.067984	.000002	-.002598	.105361	.000515	.001516	-.042809	.020942	.0
33	Polygon	.054543	34.512337	.977843	.057735	-.079577	-.000001	-.002797	.111172	.000105	.001871	-.003192	.019745	.0

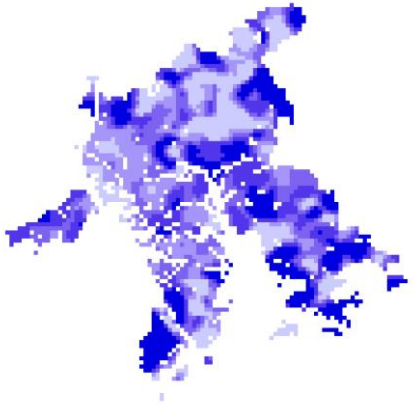


ArcPy Tool3 Output: GWR result of ESI

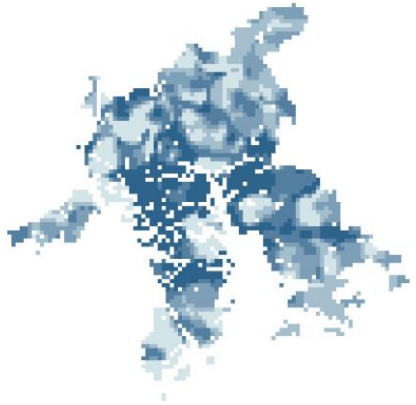


Standardized Residuals

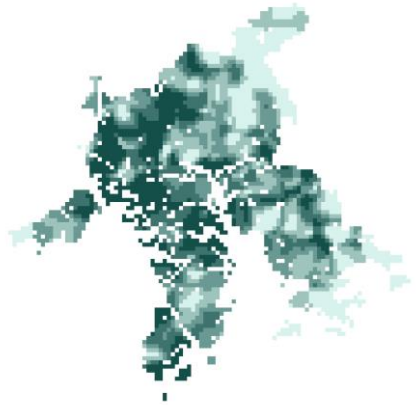
Coefficients of predictors



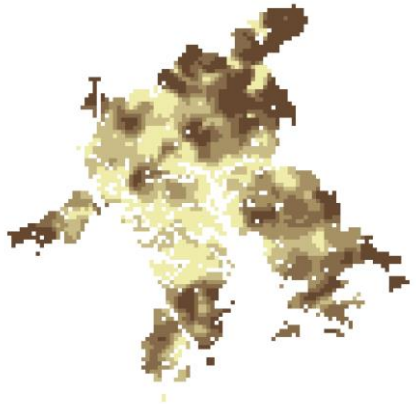
Distance to water



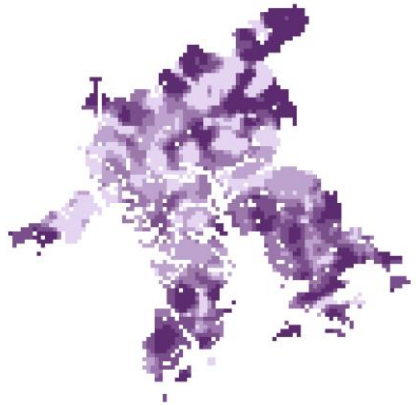
Elevation



Slope

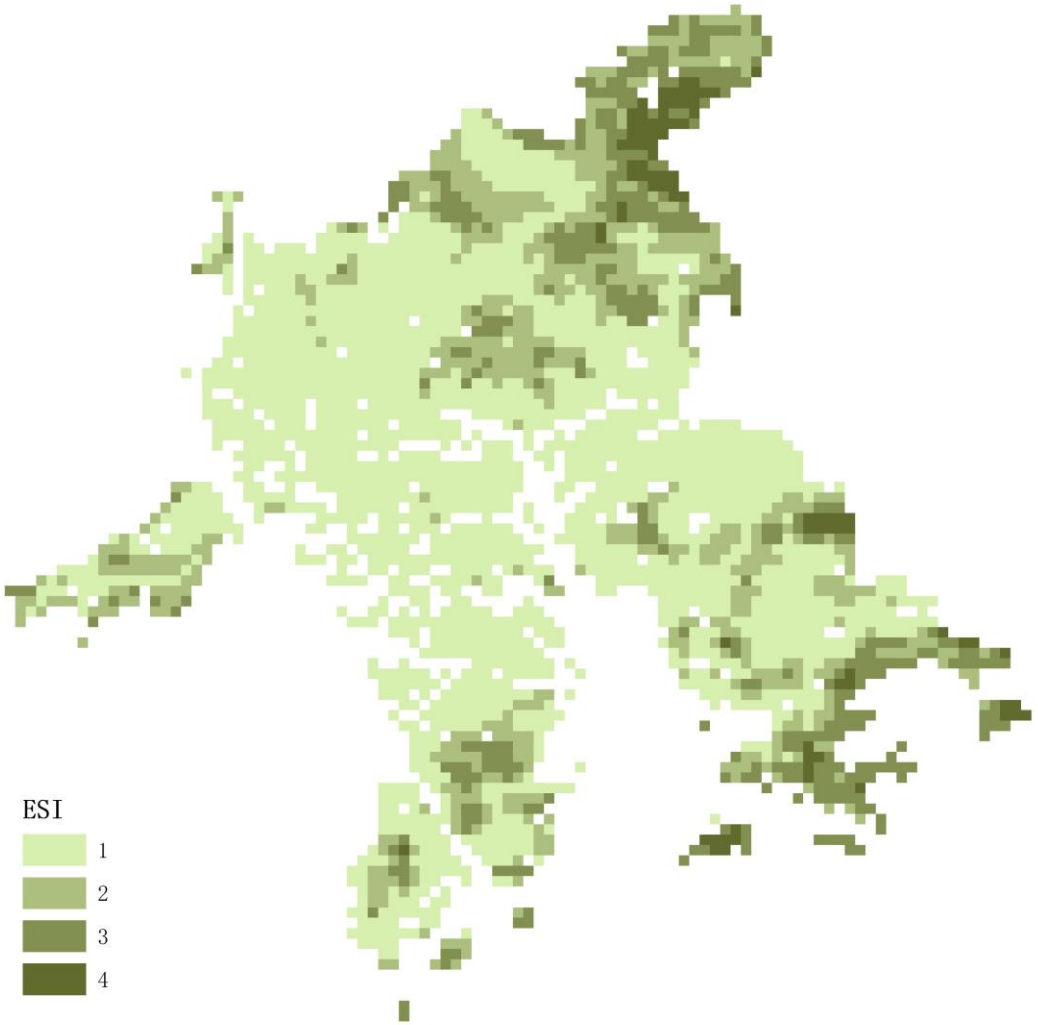


Tree coverage



Non-tree vegetation

ArcPy Tool3 Output: Reclassified ESI values (predicted topographic diversity)



Table

computedESI

FID	Shape *	Source_ID	BSI
0	Polygon	6	3
1	Polygon	13	3
2	Polygon	52	2
3	Polygon	53	2
4	Polygon	57	2
5	Polygon	58	3
6	Polygon	59	2
7	Polygon	69	1
8	Polygon	70	1
9	Polygon	75	3
10	Polygon	76	3
11	Polygon	77	2
12	Polygon	84	1
13	Polygon	85	1
14	Polygon	86	1
15	Polygon	87	1
16	Polygon	92	2
17	Polygon	93	2
18	Polygon	94	2
19	Polygon	103	1
20	Polygon	104	1
21	Polygon	105	1
22	Polygon	106	1
23	Polygon	107	1
24	Polygon	108	1
25	Polygon	109	1
26	Polygon	110	2
27	Polygon	114	1
28	Polygon	126	1
29	Polygon	127	1
30	Polygon	128	1
31	Polygon	129	1
32	Polygon	132	1
33	Polygon	133	1
34	Polygon	141	3

1 (0 out of 2959

computedESI

ArcPy Tool3 Code

```
# Import external modules
import sys, os, string, math, arcpy, traceback, numpy
from arcpy import env

# Allow output to overwite any existing grid of the same name
arcpy.env.overwriteOutput = True


root = "E:/Study-18Fall/GeospaticialSoftware/EEProject_Data/shp_arcpy/"

def SpatialJoinESI(target, dependent, predictor1, predictor2, predictor3, predictor4, predictor5, output):
        inter1 = root + "inter1.shp"
        inter2 = root + "inter2.shp"
        inter3 = root + "inter3.shp"
        inter4 = root + "inter4.shp"
        inter5 = root + "inter5.shp"

        # spatial join dependent variable
        fieldmappings1 = arcpy.FieldMappings()
        fieldmappings1.addTable(target)
        fieldmappings1.addTable(dependent)

        gridValue1 = fieldmappings1.findFieldMapIndex("grid_code")
        fieldmap1 = fieldmappings1.getFieldMap(gridValue1)

        field1 = fieldmap1.outputField
        field1.name = "DV"
        fieldmap1.outputField = field1

        fieldmap1.mergeRule = "mean"
        fieldmappings1.replaceFieldMap(gridValue1, fieldmap1)

        arcpy.SpatialJoin_analysis(target, dependent, inter1, "JOIN_ONE_TO_ONE", "KEEP_ALL", fieldmappings1, "CONTAINS")
```

ArcPy Tool3 Code

```
# spatial join the first predictor
    fieldmappings2 = arcpy.FieldMappings()
    fieldmappings2.addTable(inter1)
    fieldmappings2.addTable(predictor1)

    gridValue2 = fieldmappings2.findFieldMapIndex("grid_code")
    fieldmap2 = fieldmappings2.getFieldMap(gridValue2)

    field2 = fieldmap2.outputField
    field2.name = "pre1"
    fieldmap2.outputField = field2

    fieldmap2.mergeRule = "mean"
    fieldmappings2.replaceFieldMap(gridValue2, fieldmap2)

    arcpy.SpatialJoin_analysis(inter1, predictor1, inter2, "JOIN_ONE_TO_ONE", "KEEP_ALL", fieldmappings2, "CONTAINS")


    # spatial join the second predictor
    fieldmappings3 = arcpy.FieldMappings()
    fieldmappings3.addTable(inter2)
    fieldmappings3.addTable(predictor2)

    gridValue3 = fieldmappings3.findFieldMapIndex("grid_code")
    fieldmap3 = fieldmappings3.getFieldMap(gridValue3)

    field3 = fieldmap3.outputField
    field3.name = "pre2"
    fieldmap3.outputField = field3

    fieldmap3.mergeRule = "mean"
    fieldmappings3.replaceFieldMap(gridValue3, fieldmap3)

    arcpy.SpatialJoin_analysis(inter2, predictor2, inter3, "JOIN_ONE_TO_ONE", "KEEP_ALL", fieldmappings3, "CONTAINS")
```

ArcPy Tool3 Code

```
    # spatial join the third predictor
    fieldmappings4 = arcpy.FieldMappings()
    fieldmappings4.addTable(inter3)
    fieldmappings4.addTable(predictor3)

    gridValue4 = fieldmappings4.findFieldMapIndex("grid_code")
    fieldmap4 = fieldmappings4.getFieldMap(gridValue4)

    field4 = fieldmap4.outputField
    field4.name = "pre3"
    fieldmap4.outputField = field4

    fieldmap4.mergeRule = "mean"
    fieldmappings4.replaceFieldMap(gridValue4, fieldmap4)

    arcpy.SpatialJoin_analysis(inter3, predictor3, inter4, "JOIN_ONE_TO_ONE", "KEEP_ALL", fieldmappings4, "CONTAINS")


# spatial join the fourth predictor
    fieldmappings5 = arcpy.FieldMappings()
    fieldmappings5.addTable(inter4)
    fieldmappings5.addTable(predictor4)

    gridValue5 = fieldmappings5.findFieldMapIndex("grid_code")
    fieldmap5 = fieldmappings5.getFieldMap(gridValue5)

    field5 = fieldmap5.outputField
    field5.name = "pre4"
    fieldmap5.outputField = field5

    fieldmap5.mergeRule = "mean"
    fieldmappings5.replaceFieldMap(gridValue5, fieldmap5)
```

ArcPy Tool3 Code

```
arcpy.SpatialJoin_analysis(inter4, predictor4, inter5, "JOIN_ONE_TO_ONE", "KEEP_ALL", fieldmappings5, "CONTAINS")


# spatial join the fifth predictor
fieldmappings6 = arcpy.FieldMappings()
fieldmappings6.addTable(inter5)
fieldmappings6.addTable(predictor5)

gridValue6 = fieldmappings6.findFieldMapIndex("grid_code")
fieldmap6 = fieldmappings6.getFieldMap(gridValue6)

field6 = fieldmap6.outputField
field6.name = "pre5"
fieldmap6.outputField = field6

fieldmap6.mergeRule = "mean"
fieldmappings6.replaceFieldMap(gridValue6, fieldmap6)

arcpy.SpatialJoin_analysis(inter5, predictor5, output, "JOIN_ONE_TO_ONE", "KEEP_ALL", fieldmappings6, "CONTAINS")

# Delete the intermediate shapefile
arcpy.Delete_management(inter1)
arcpy.Delete_management(inter2)
arcpy.Delete_management(inter3)
arcpy.Delete_management(inter4)
arcpy.Delete_management(inter5)

return output
```

ArcPy Tool3 Code

```
# If Spatial Analyst license is available, check it out
if arcpy.CheckExtension("spatial") == "Available":
    arcpy.CheckOutExtension("spatial")

    try:
        # read input
        inboundary = arcpy.GetParameterAsText(0)
        output = arcpy.GetParameterAsText(1)
        inraster1 = arcpy.GetParameterAsText(2)
        inraster2 = arcpy.GetParameterAsText(3)
        inraster3 = arcpy.GetParameterAsText(4)
        inraster4 = arcpy.GetParameterAsText(5)
        inraster5 = arcpy.GetParameterAsText(6)
        inraster6 = arcpy.GetParameterAsText(7)

        # convert raster to points
        rasterPoint1 = root + inraster1[:-4] + "_temp.shp"
        rasterPoint2 = root + inraster2[:-4] + "_temp.shp"
        rasterPoint3 = root + inraster3[:-4] + "_temp.shp"
        rasterPoint4 = root + inraster4[:-4] + "_temp.shp"
        rasterPoint5 = root + inraster5[:-4] + "_temp.shp"
        rasterPoint6 = root + inraster6[:-4] + "_temp.shp"

        arcpy.RasterToPoint_conversion(inraster1, rasterPoint1, "VALUE")
        arcpy.RasterToPoint_conversion(inraster2, rasterPoint2, "VALUE")
        arcpy.RasterToPoint_conversion(inraster3, rasterPoint3, "VALUE")
        arcpy.RasterToPoint_conversion(inraster4, rasterPoint4, "VALUE")
        arcpy.RasterToPoint_conversion(inraster5, rasterPoint5, "VALUE")
        arcpy.RasterToPoint_conversion(inraster6, rasterPoint6, "VALUE")


        # spatial join points to fishnet and get the average values
        spatialJoin_result = root + "spaResult_ESI.shp"
        SpatialJoinESI(inboundary, rasterPoint1, rasterPoint2, rasterPoint3, rasterPoint4, rasterPoint5, rasterPoint6, spatialJoin_result)
```

ArcPy Tool3 Code

```
# run GWR
    GWR_result = root + "GWR_resultESI.shp"
    predictor = "pre1;pre2;pre3;pre4;pre5"
    arcpy.GeographicallyWeightedRegression_stats(spatialJoin_result, "DV", predictor, GWR_result, "ADAPTIVE", "BANDWIDTH PARAMETER","#", "50", "#", "#", "#", "#", "#", "#")


    # reclassify GWR predicted value and get the FUI classification
    # Replicate the GWR result
    GWR_reclassify = output
    arcpy.Copy_management(GWR_result, GWR_reclassify)
    arcpy.AddField_management(GWR_reclassify, "ESI", "DOUBLE", 20, 5)

    # get the min and max predicted values
    GWR_min = GWR_reclassify[:-4] + "_min.shp"
    GWR_max = GWR_reclassify[:-4] + "_max.shp"

    arcpy.Sort_management(GWR_reclassify,GWR_min,[["Predicted", "ASCENDING"]])
    arcpy.Sort_management(GWR_reclassify,GWR_max,[["Predicted", "DESCENDING"]])

    enumeration_min = arcpy.SearchCursor(GWR_min)
    minValue = enumeration_min.next().getValue("Predicted")
    arcpy.AddMessage("Minimum predicted value:  " + str(minValue) + "\n")

    enumeration_max = arcpy.SearchCursor(GWR_max)
    maxValue = enumeration_max.next().getValue("Predicted")
    arcpy.AddMessage("Maximum predicted value:  " + str(maxValue) + "\n")

    increase = (maxValue - minValue)/4
    first = increase + minValue
    second = increase*2 + minValue
    third = increase*3 + minValue

    del enumeration_min
    del enumeration_max
```

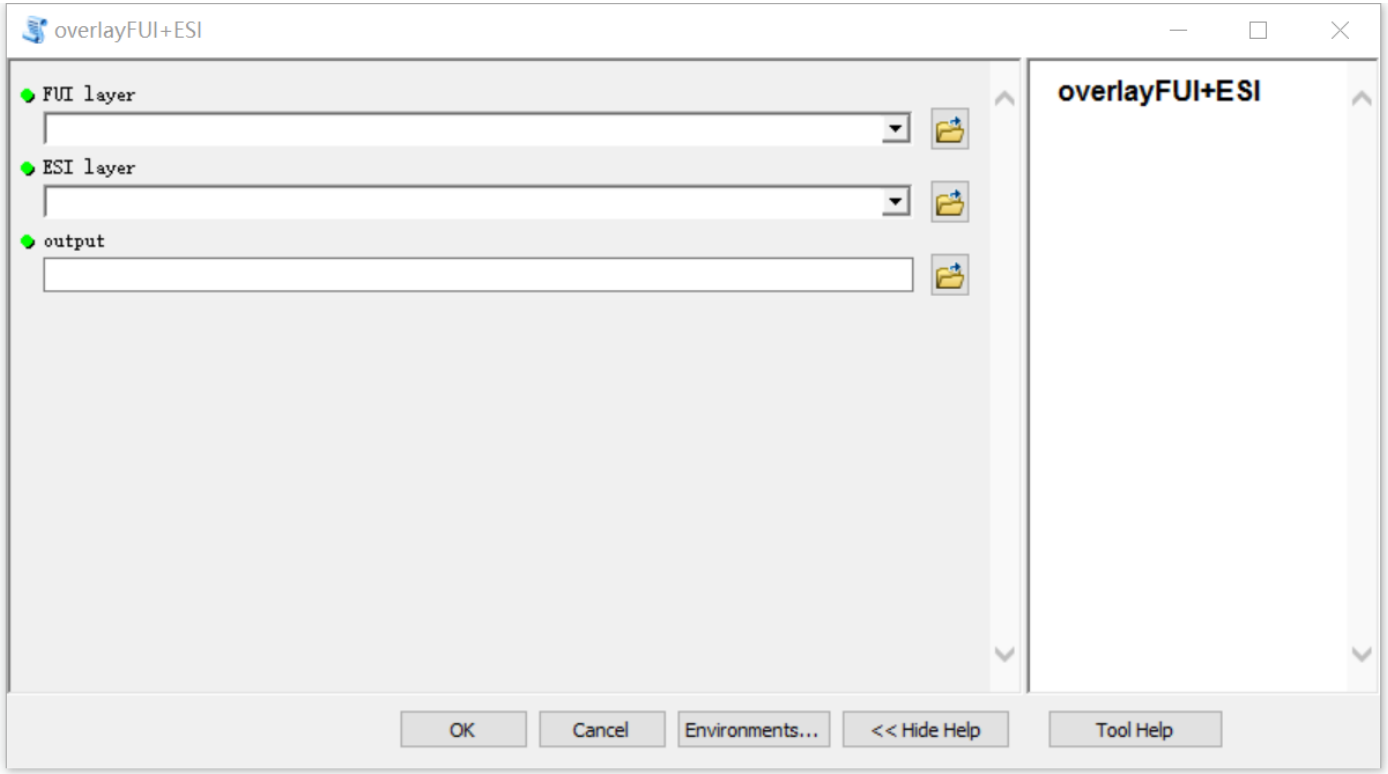
ArcPy Tool3 Code

```
# Delete the intermediate shapefile
 arcpy.Delete_management(GWR_min)
 arcpy.Delete_management(GWR_max)

 # reclassify
 enumeration = arcpy.UpdateCursor(GWR_reclassify)

 for row in enumeration:
    predicted = row.getValue("Predicted")
    if predicted < first:
       row.setValue("ESI", 1)
    elif predicted >= first and predicted < second:
       row.setValue("ESI", 2)
    elif predicted >= second and predicted < third:
       row.setValue("ESI", 3)
    else:
       row.setValue("ESI", 4)

    enumeration.updateRow(row)


 # delete unnecessary fields
 dropFieldESI = ["Observed", "Cond", "LocalR2", "Predicted", "Intercept", "C1_pre1", "C2_pre2", "C3_pre3", "C4_pre4", "C5_pre5",
              "Residual", "StdError", "StdErr_Int", "StdErrC1_p", "StdErrC2_p", "StdErrC3_p", "StdErrC4_p", "StdErrC5_p", "StdResid"]
 arcpy.DeleteField_management(GWR_reclassify, dropFieldESI)



 del row
 del enumeration
```

ArcPy Tool3 Code

```
        # Delete the intermediate shapefile
        arcpy.Delete_management(rasterPoint1)
        arcpy.Delete_management(rasterPoint2)
        arcpy.Delete_management(rasterPoint3)
        arcpy.Delete_management(rasterPoint4)
        arcpy.Delete_management(rasterPoint5)
        arcpy.Delete_management(rasterPoint6)


    except Exception as e:
        # If unsuccessful, end gracefully by indicating why
        arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
        # ... and where
        exceptionreport = sys.exc_info()[2]
        fullermessage   = traceback.format_tb(exceptionreport)[0]
        arcpy.AddError("at this location: \n\n" + fullermessage + "\n")

    # Check in Spatial Analyst extension license
    arcpy.CheckInExtension("spatial")
else:
    print "Spatial Analyst license is " + arcpy.CheckExtension("spatial")
```

ArcPy Tool4: Overlay the result of FUI and ESI



Step1:
Tabular join the table of FUI and ESI by the field “Source_ID”

The input data will be:
1. The reclassified FUI layer calculated from Tool2
2. The reclassified ESI layer calculated from Tool3

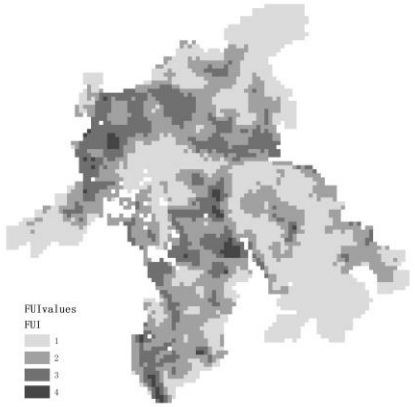
arcpy.AddJoin_management

Step2:
Multiply the reclassified values of FUI and ESI and store the values in a new field called “Index”. Cells with higher value mean that this cell potential urban growth is more likely to threaten the natural environment.

arcpy.UpdateCursor

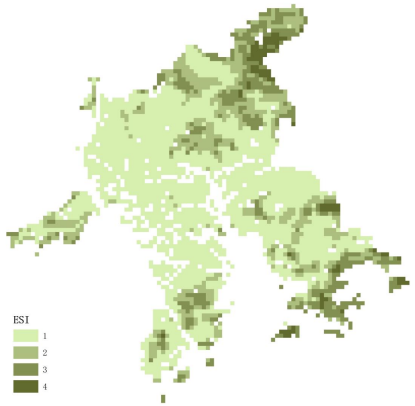
ArcPy Tool4 Output: Overlay FUI and ESI

Input

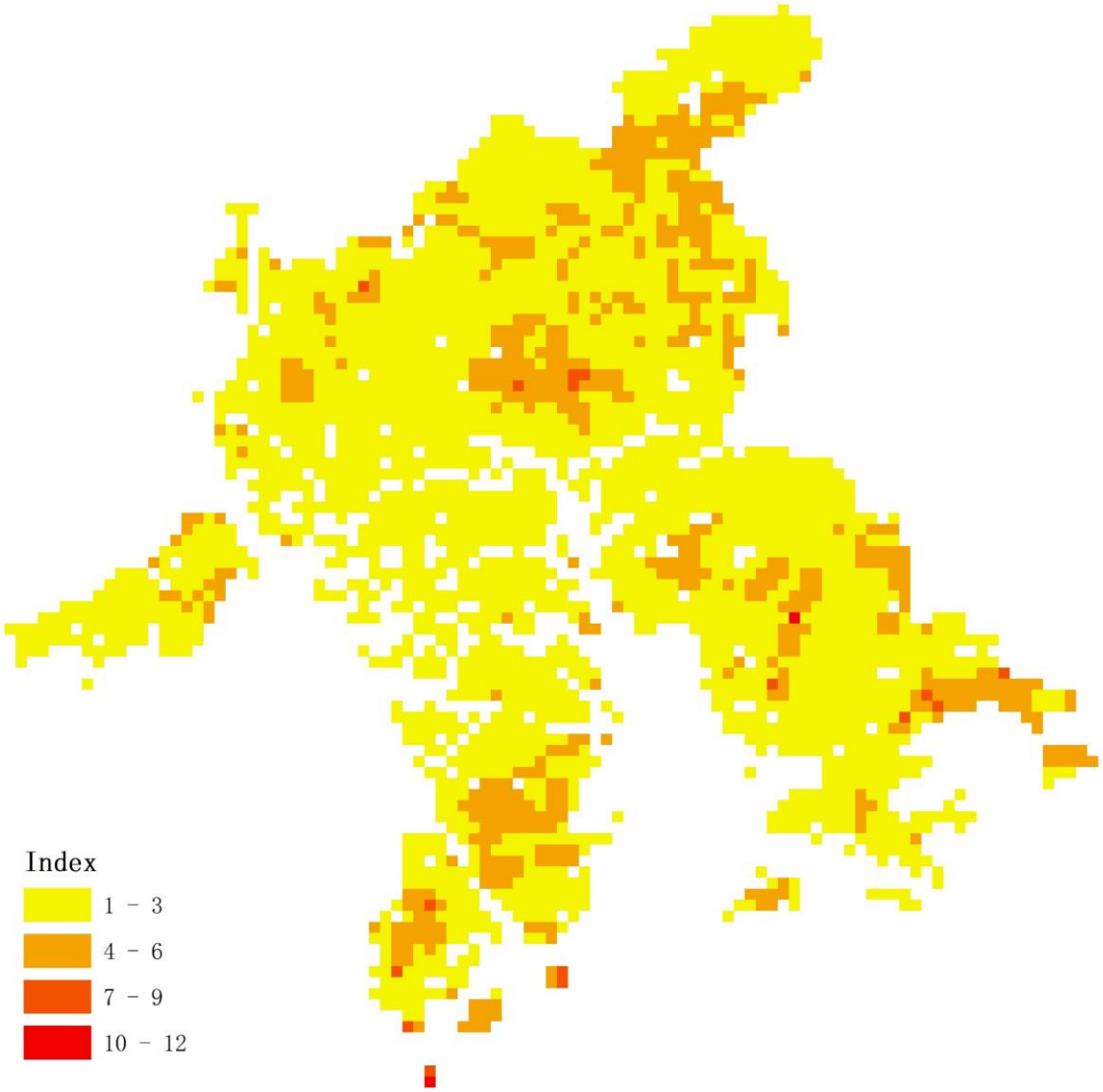


Reclassified FUI

x



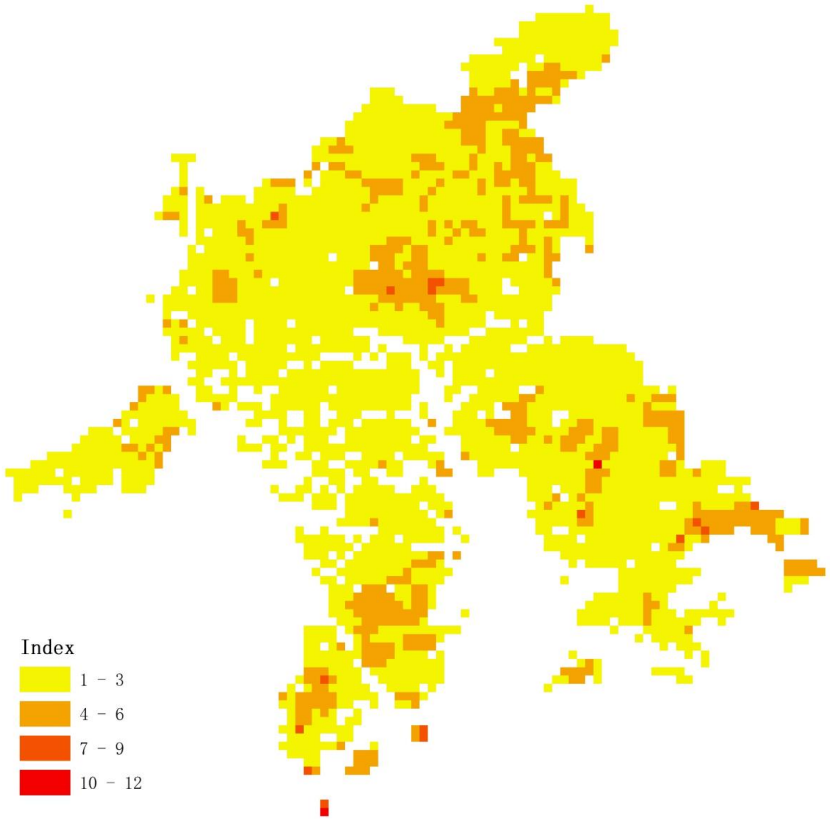
Reclassified ESI



Output

ArcPy Tool4 Output: Overlay FUI and ESI

FUI	BSI	Index
4	3	12
3	3	9
4	2	8
3	2	6
2	2	4
2	3	6
2	2	4
3	1	3
3	1	3
2	3	6
2	3	6
2	2	4
3	1	3
3	1	3
3	1	3
2	1	2
2	2	4
2	2	4
2	2	4
3	1	3
2	1	2
3	1	3
3	1	3
3	1	3
3	1	3
2	1	2
2	2	4
2	1	2
3	1	3
3	1	3
3	1	3
2	1	2
3	1	3
3	1	3
2	3	6



FUI / ESI	1	2	3	4
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12
4	4	8	12	16

Index:
Grid cells with the highest environmental sensitivity that are also the most likely areas to be developed as urban areas.

ArcPy Tool4 Code

```
# Import external modules
import sys, os, string, math, arcpy, traceback, numpy
from arcpy import env

# Allow output to overwite any existing grid of the same name
arcpy.env.overwriteOutput = True

root = "E:/Study-18Fall/GeospaticialSoftware/EEProject_Data/shp_arcpy/"

try:
	# read input
	inputFUI = arcpy.GetParameterAsText(0)
	inputESI = arcpy.GetParameterAsText(1)
	output = arcpy.GetParameterAsText(2)


	# tabular join
	join_table = arcpy.AddJoin_management(inputFUI, "Source_ID", inputESI, "Source_ID", "KEEP_COMMON")
	arcpy.CopyFeatures_management(join_table, output)


	# calculation
	arcpy.AddField_management(output, "FUI", "DOUBLE", 20, 5)
	arcpy.AddField_management(output, "ESI", "DOUBLE", 20, 5)
	arcpy.AddField_management(output, "Index", "DOUBLE", 20, 5)

	fieldnameFUI = inputFUI[:-3] + "_1"
	fieldnameESI = inputESI[:-3] + "_3"

	enumeration = arcpy.UpdateCursor(output)
```

ArcPy Tool4 Code

```
        for row in enumeration:
            FUI = row.getValue(fieldnameFUI)
            ESI = row.getValue(fieldnameESI)
            row.setValue("FUI", FUI)
            row.setValue("ESI", ESI)

            index = FUI*ESI
            arcpy.AddMessage("FUI*ESI =  " + str(index))
            row.setValue("Index", index)

            enumeration.updateRow(row)


        del row
        del enumeration



except Exception as e:
        # If unsuccessful, end gracefully by indicating why
        arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
        # ... and where
        exceptionreport = sys.exc_info()[2]
        fullermessage   = traceback.format_tb(exceptionreport)[0]
        arcpy.AddError("at this location: \n\n" + fullermessage + "\n")
```