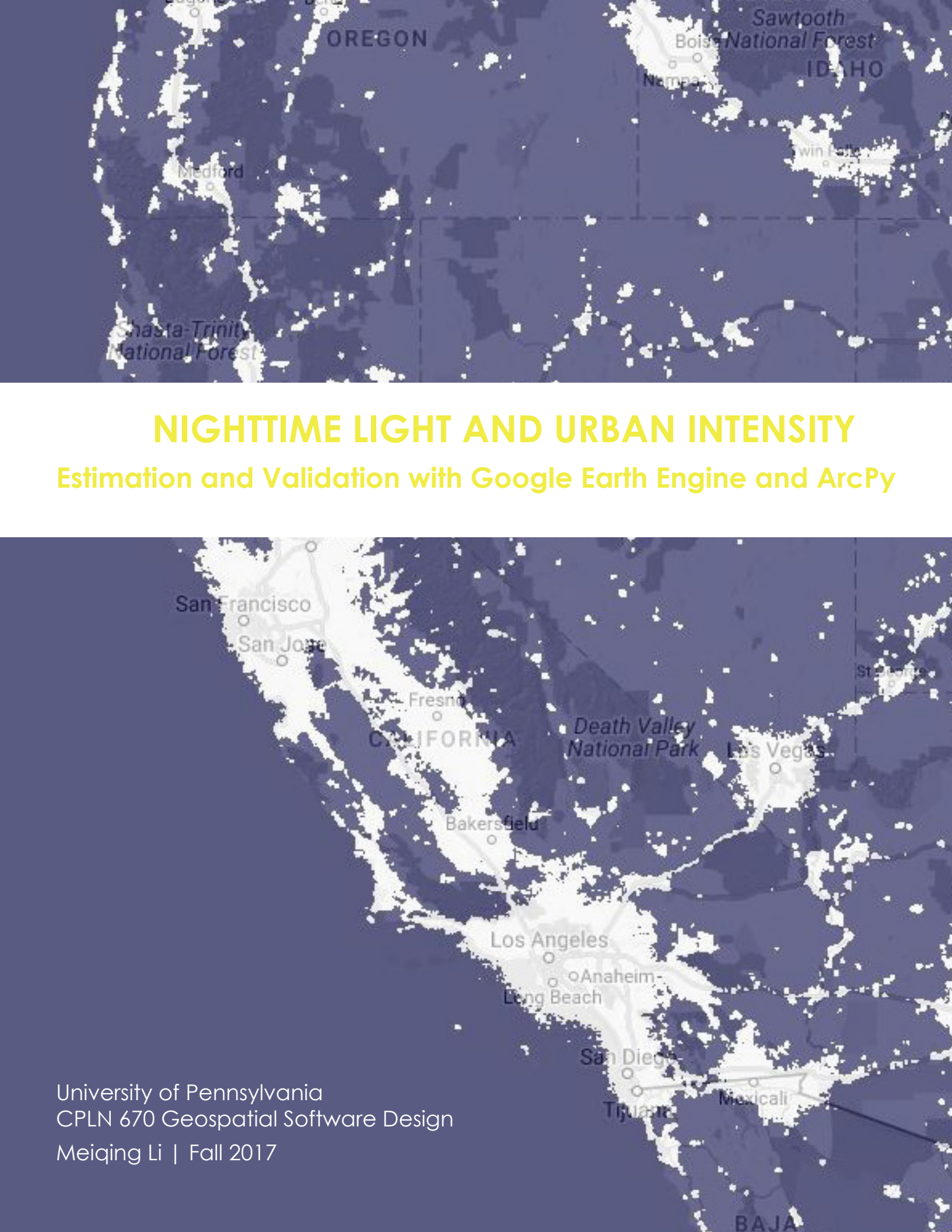


NIGHTTIME LIGHT AND URBAN INTENSITY

Estimation and Validation with Google Earth Engine and ArcPy

University of Pennsylvania
CPLN 670 Geospatial Software Design
Meiqing Li | Fall 2017

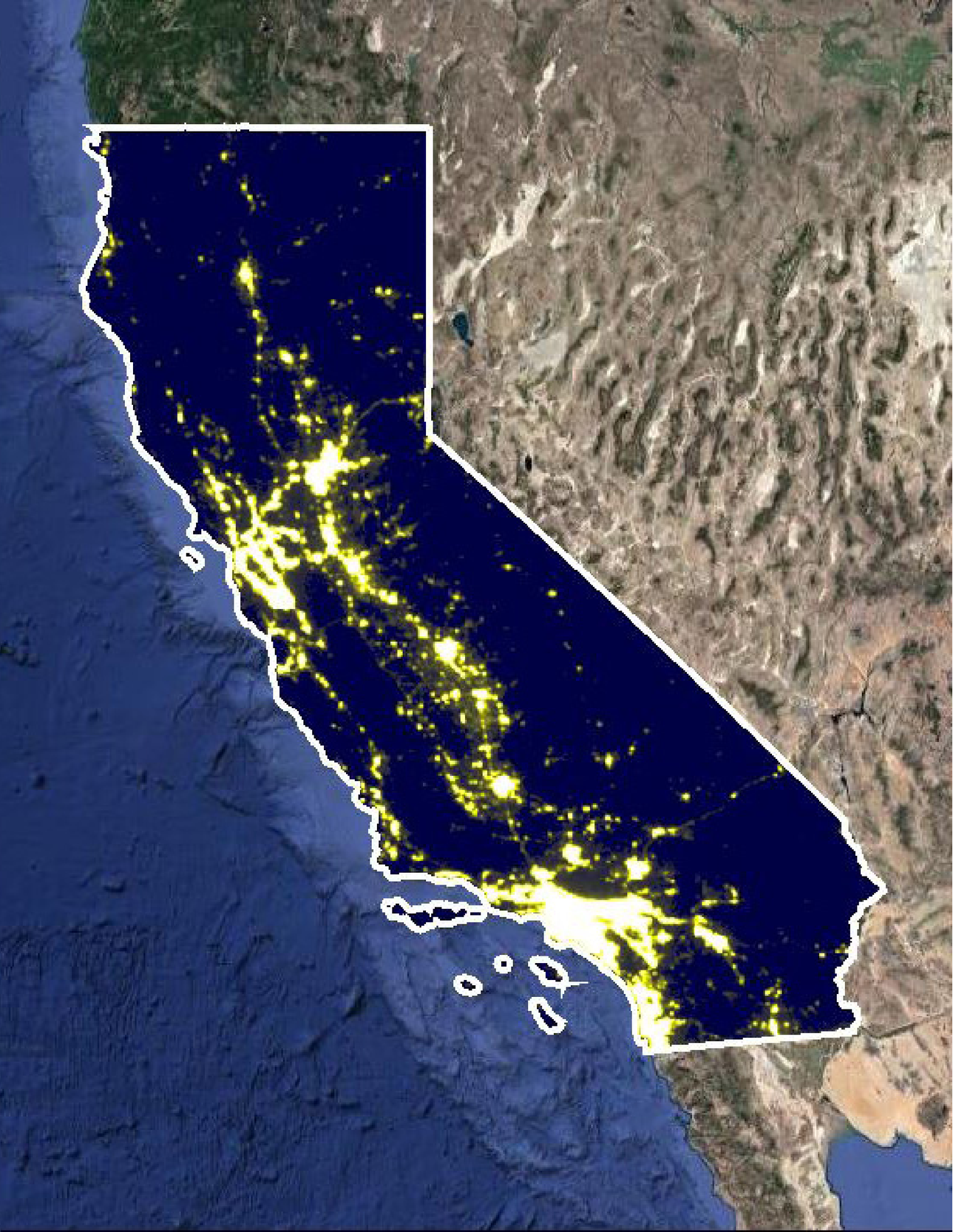
Thanks Professor Dana Tomlin for continuous feedback and support.

This is the final project for CPLN 670 Geospatial Software Design at University fo Pennsylvania School of Design. It consists of two parts that apply different geospatial software and techniques:

1) The **Google Earth Engine** project utilizes the capability of raster processing working with satellite imagery,

2) and the **ArcPy** project focuses on relational data management and calculation.

TABLE OF CONTENT

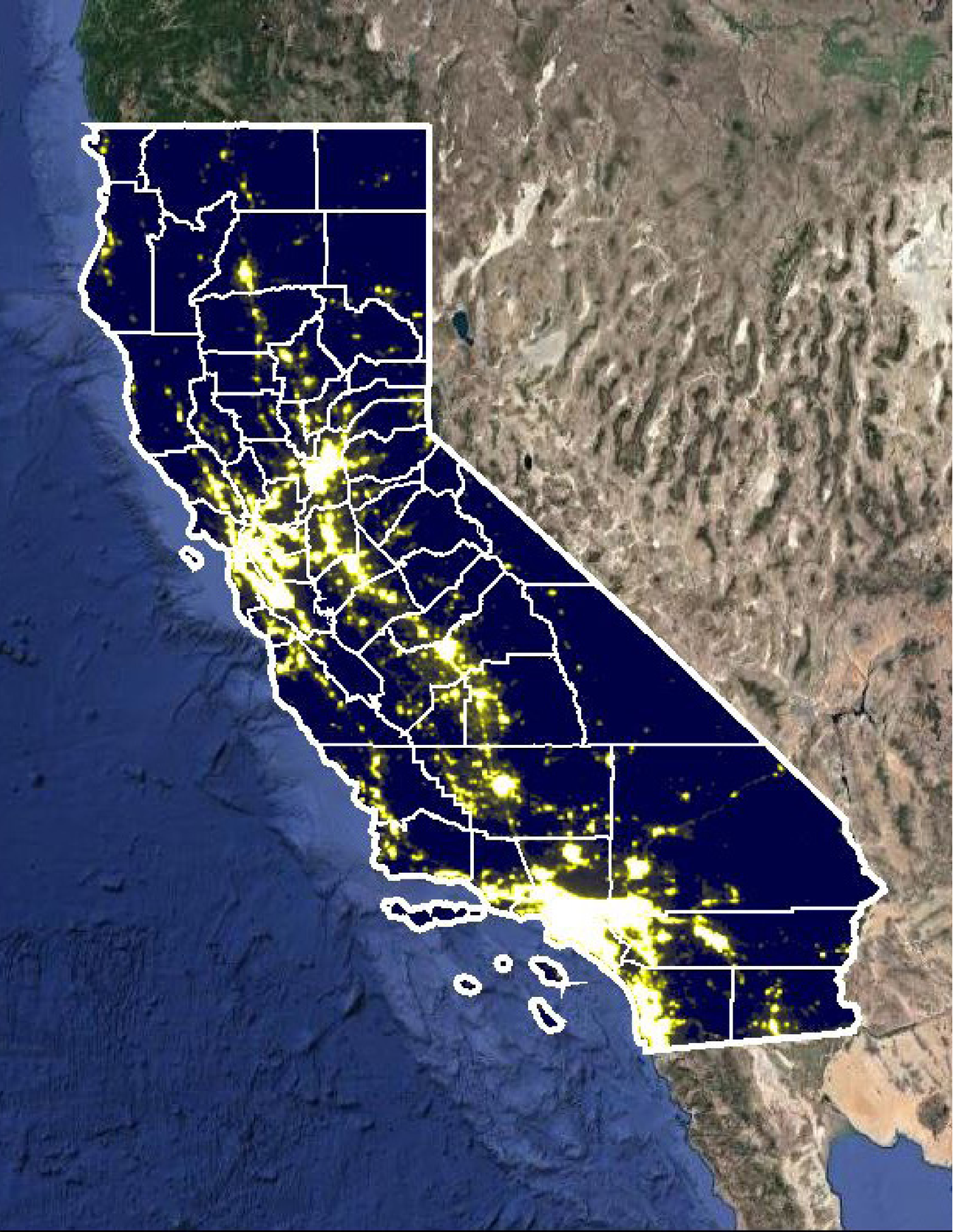


DMSP/OLS NTL of California (2013)

INTRODUCTION

This project stems from an idea to use remote sensing imagery to explore the unrevealed social and economic conditions of cities and regions where such data are scarce or with limited access. Both daytime and nighttime remote sensing imagery can shade light on spatial patterns of human activities and settlements. While nighttime light (NTL) imagery is a good proxy of urban intensity, daytime land cover data well indicate the built-up areas. Combining these images of different spatial resolutions can enhance their predictive power in terms of statistical significance and scale. This can be especially useful for studying urban growth patterns of cities in developing countries that are experiencing rapid urban changes, while at the same time lack effective monitoring tools to ensure sustainable growth.

The ideas for this project were inspired by several researches validating the relationship between NTL and urban activities applying variations of DMSP/OLS, Landsat and MODIS images as well as in different contexts. For example, Mellander et al. (2015) explore the correlation between nighttime light (NTL) and socio-economic dynamics using fine-grained statistical data from Sweden. Akiyama (2012) conducted statistical tests and regression on light intensity and urban spatial factors including road and building density, as well as dynamic population distribution. Some researchers attempt to improve the robustness of estimation through calibration by combinations of satellite imagery. Zhang et al. (2013, 2015) introduce two different measures of urban intensity integrating DMSP/OLS NTL, Landsat and MODIS NDVI composites.



DMSP/OLS NTL of California counties (2013)

Goals & Objectives

As an integrate part of the final project for CPLN 670 Geospatial Software Design, the Google Earth Engine project aims to develop a generalizable tool/method that can be easily applied to different contexts and scales for estimation purpose. Using the Google Earth Engine platform, the tool standardizes the procedure of filtering, classifying, resampling, visualizing and computing satellite imagery. It outputs estimates of urban intensity within a specific geographic area and time period, which can be validated by actual socio-economic data in ArcGIS (the ArcPy project for this class).

The ArcPy application for second part of this project continues the exploration of NTL data by taking the output from raster processing in Google Earth Engine, and compare the estimation with socio-economic factors indicated by census and spatial data. It outputs a series of tables that contain comprehensive information for more rigorous statistical analysis.

Data

Remote Sensing Imagery

There are three major sources of remote sensing imagery used in this project: DMSP/OLS NTL, Landsat-7 and MODIS composite.

DMSP/OLS NTL

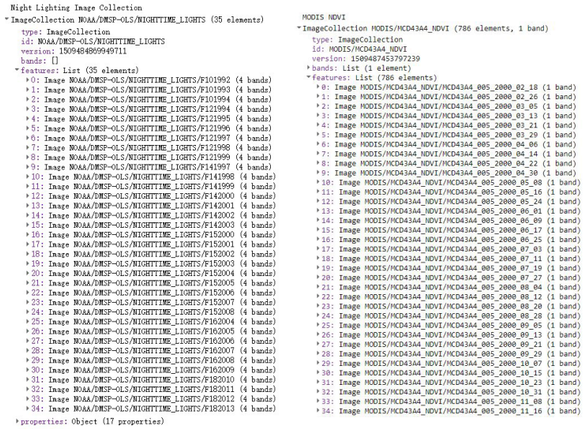
Prepared by NOAA, the DMSP-OLS Nighttime Lights Time Series Version 4 provides cloud free composite making use of all available data from each calendar year from 1992 through 2013. It has a 'stable lights' band which is a very good source to indicate persisting lighting in urban areas. However, pixels in urban areas are often saturated and suffer from overglow effects (Zhang, et al., 2015). With 30 arc second (about 1km) grids, its resolution is coarse. Therefore, it would be ideal to have other data complementing NTL. It is also a question to bear in mind that what could be the finest areas represented by this dataset.

Landsat-7 NDVI Composite

Landsat-7 imagery from USGS is a fine-grained dataset with 30m resolution. The annual NDVI composite could thus be an ideal source to accompany NTL. However, there has been ETM+ scan line corrector (SLC) failure since May 2003, resulting in 22% missing pixels. Without calibration, the Landsat-7 NDVI Composite is not considered a good source to indicate land cover conditions.

MODIS NDVI Composite

The MODIS imagery has a medium resolution at the range of 250m to 1km. Its 16-day NDVI composite can thus be a substitute when good-quality Landsat-7 imagery is not available or well-calibrated. This imagery source has a time-series collection since 2000. Although this tool can be applied to any geographic areas and scale as long as capacity allows, this project takes California as study area, and conducts county or MSA level analysis.

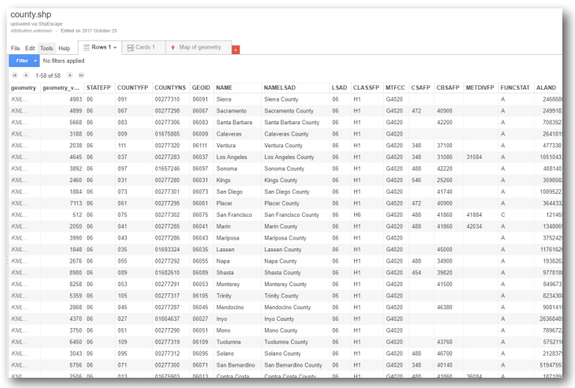


DMSP/OLS and MODIS Image Collections

Socio-Economic

TIGER/Line® with Selected Demographic and Economic Data

The 2010 Census Estimates with selected demographic and economic data are obtained from United States Census Bureau. These data are available at geodatabase and shapefile format, and stored in a series of database tables. The shapefiles associated with them serve as the boundary shapefiles of our geographic unit of analysis (i.e. county, MSA). These shapefiles need to be uploaded as Google Fusion Tables before processing in Google Earth Engine.



County shapefiles converted to Google Fusion Table

Scope of California OSM shapefile

Urban Form

Open Street Map (OSM)

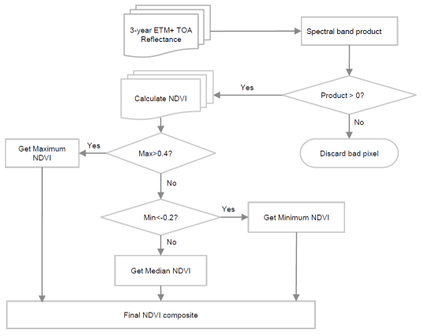
The OSM shapefiles from an online source (http://download.geofabrik.de/north-america.html) provide the spatial distribution of roads and buildings in details. The roads and buildings are represented in the form of line and polygon features, whose length and area can be calculated and spatially joined to different geographic units of analysis.

METHODS

Algorithms

Normalized Difference Urban Index (NDUI)

The calculation of NDUI is based on the method from Zhang (et al., 2015). It is a normalized difference index analogous to NDVI (Normalized Difference Vegetation Index), a commonly used index to determine the density of greenness on a patch of land. The NDUI composite combines the coarse-resolution NTL and fine-grained Landsat NDVI composite to maximize the separation between urban areas and easily confused barren lands. However due to the SLC failure problem of Landsat-7 images, they should first go through a calibration process before being used for NDUI composition. The diagram below illustrates the proposed procedure to generate Landsat NDVI by Zhang et al. (2015).



The procedure to generate NDVI composites from multi-year Landsat-7/ETM+ imagery with the proposed Mixed NDVI method (Zhang et al., 2015)

Once the Landsat NDVI is successfully calibrated and composited, Despite the difference sources of NDVI composite, we can apply the same equation to derive a NDUI composite combing NTL and NDVI:

$$NDUI = (NTL - NDVI)/(NTL + NDVI)$$

The NTL value ranges from 0 to 63, and NDVI has a value between -1 and 1. The calculation of NDUI follows the rationale that there is an inverse relationship between vegetation density and luminosity. Therefore, when both NTL and NDVI are normalized, their normalized difference between -1 and 1 enhances the robustness of the estimated urban intensity. As a result, in urban core areas with sparse vegetation cover, NDUI will have a high value close to 1, while unlit rural areas with more vegetation will have lower values close to 1.

In this project, I use a simplified version of this calibration algorithm to identify bad pixels from the product of selected bands, and discard the zero pixels with a mask. The raw image for calibration is also processed by filtering only images from 2013 (the latest available year of NTL data), and taking the mean. Since none of the postprocessing pixels in the study area are good ones, I get NDVI from MODIS instead.

Pearson Correlation

The time series NDUI values generated from satellite imageries are validated through comparing with actual socio-economic and urban form data. For simple validation, the ArcPy project matches data from multiple sources to the same unit of analysis, and calculates the Pearson correlation between NDUI and other urban intensity indicators (i.e. population density, and road density). The equation for calculating Pearson correlation is as follows (assume population and sample have the same distribution):

$$\rho_{x,y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - [E(X)]^2}\sqrt{E(Y^2) - [E(Y)]^2}}$$

If NDUI has a high correlation close to 1 with some actual socio-economic indicators, it is likely a good estimator for urban intensity.

Tools

Google Earth Engine

The processing of three types of remote sensing imagery makes use of the following tools and functions (following the sequence appearing in code):

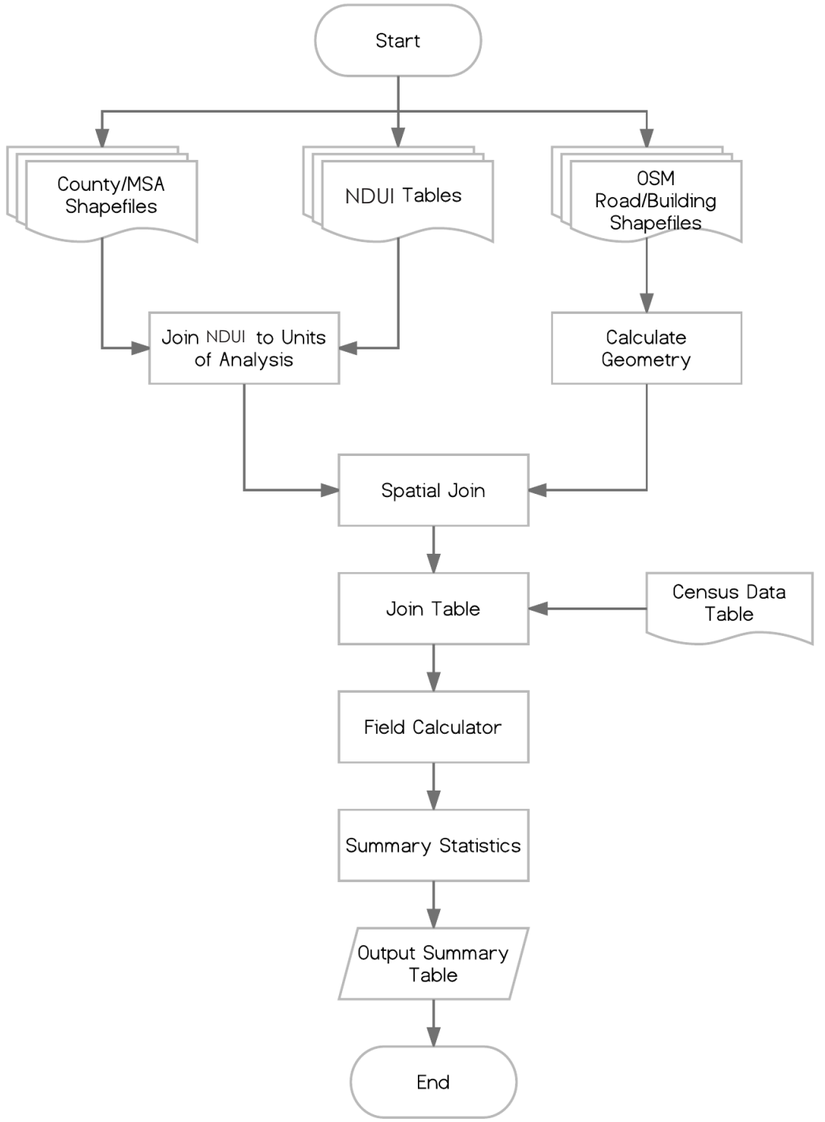
Map.setOptions, print, FeatureCollection.filterMetadata, Image.paint, Map.addLayer, ImageCollection, Image.expression, Image.clamp, Image.clip, Image.eq, Image.gt, Image.mask, Image.select, Image.normalizedDifference, ImageCollection.filterDate, ImageCollection.mean, Image.reduceRegions, Reducer.mean, Join.inner.apply, Filter.equals, Number, Feature.get, Number.subtract/add/divide, FeatureCollection.map, Chart.feature.histogram, Chart.setSeriesNames, Chart.setOptions, Export.table.toDrive

ArcPy

The Arcpy project utilizes the following tools (following the sequence appearing in code):

CheckExtension, CheckoutExtension, GetParameterAsText, Copy_management, JoinField_management, AddError, CheckInextension, Describe.shapeType, AddField_management, CalculateField_management, FieldMappings.addTable, FieldMappings.findFieldMapIndex, FieldMapping.getFiedlMap, FieldMappings.replaceFieldMap, SpatialJoin_analysis, Delete_management, Addmessage, UpdateCursor.getValue, UpdateCursor.setValue, UpdateCursor.updateRow, Statis-tics_analysis

Flowchart of Google Earth Engine Project

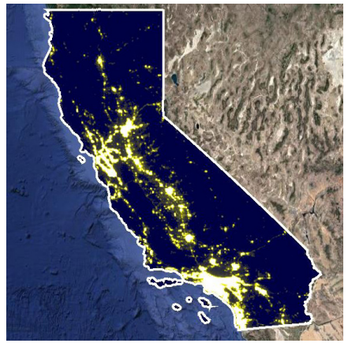


Flowchart of ArcPy Project

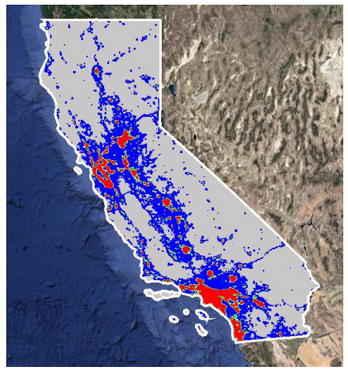
Google Earth Engine

Estimating Urban Intensity through
Daytime and Nighttime Satellite Imagery

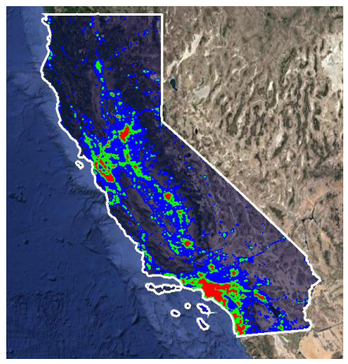
DMSP/OLS NTL



DMSP/OSL NTL of California in 2013



Reclassified light intensity in 2013



Zonal classification of urban areas

```
Stable Light Band 2013
Image NOAA/DMSP-OLS/NIGHTTIME_LIGHTS/F182013 (1 band)
  type: Image
  id: NOAA/DMSP-OLS/NIGHTTIME_LIGHTS/F182013
  version: 1504844073441627
  bands: List (1 element)
    0: "stable_lights", unsigned int8, EPSG:4326, 43201x16801 px
      id: stable_lights
      crs: EPSG:4326
      crs_transform: List (6 elements)
      data_type: unsigned int8
        type: PixelType
        max: 255
        min: 0
        precision: int
      dimensions: [43201,16801]
  properties: Object (4 properties)
```

```
//extract stable light band
var NTL = ee.Image('NOAA/DMSP-OLS/NIGHTTIME_LIGHTS/F182013').expression('b(1)');
print('Stable Light Band 2013', NTL);
//since the original band value has a range of 0 to 255,
//need to resample it to NTL's record range from 0 to 63
var NTL = NTL.clamp(0,63);
print(NTL,'NTL');
```

The exploratory analysis of DMSP/OLS takes the most recent year NTL data of California, and color codes the pixels based on values. By examining the data of 'stable_lights' band, the values need conversion from [0, 255] to [0, 63]. It is then normalized to a range between 0 and 1, making possible the NDUI calculation combing MODIS/Landsat NDVI.

```
//classify urban, suburban and rural zones by NTL values
//create zonal masks
var rural = NTL_CA.gt(0);
var suburban = NTL_CA.gt(15);
var urban = NTL_CA.gt(62);

//visualize zonal classification
var CA_rural = ee.Image(1);
var CA_rural = CA_rural.mask(rural);
Map.addLayer(CA_rural, {min: 0, max: 1,palette: '#0000FF'},'Rural');

var CA_suburban = ee.Image(1);
var CA_suburban = CA_suburban.mask(suburban);
Map.addLayer(CA_suburban, {min: 0, max: 1,palette: '#00FF00'},'Suburban');

var CA_urban = ee.Image(1);
var CA_urban = CA_urban.mask(urban);
Map.addLayer(CA_urban, {min: 0, max: 1,palette: '#FF0000'},'Urban');

//normalize NTL values for urban index calculation
var NTL_CA = NTL.expression('b(0)/63');
print(NTL_CA,'NTL_CA');
```


Landsat-7 NDVI Composite

Landsat-7 NDVI annual compostie (2013)

Landsat-7 NDVI annual compostie (2002)

Bad pixel mask of Landsat-7 (2013)

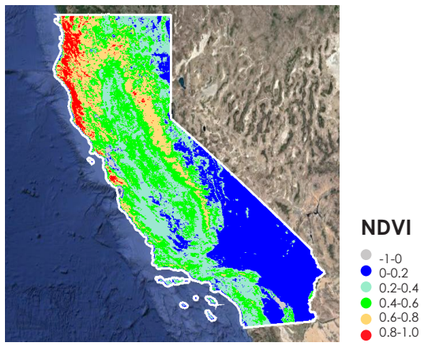
This step examines whether the Landsat NDVI or calibrated Landsat TOD data are reliable enough to be incorporated into the NDUI calculation. The algorithm as mentioned in the methodology section, is based on Zhang et al. (2015), which applies a bad pixel mask to filter out those with at least one zero-value bands among Band 1-5 and 7. Because the equipment failure happened in May 2003, the annual NDVI composite of year 2002 is selected as a baseline to compare. As illustrated by the NDVI maps, there is noticeable changes in terms of vegetation density between year 2002 and 2013. However it is still unclear whether this is due to the SLC failure or there is actually significant expansion of urban land coverage along the coastal areas.

```
//There has been a ETM+ scan line corrector (SLC) failure since May 2003.
//extract an image before 2003 and compare
var LandsatNDVI1 = ee.Image('LANDSAT/LE7_L1T_ANNUAL_NDVI/2002'); //Landsat-7 NDVI composite in 2002
//clip images to study area (California)
var LandsatNDVI1_CA = LandsatNDVI1.clip(CAstate);
//add Landsat-7 NDVI image from 2002 to the map
Map.addLayer(LandsatNDVI1_CA, {min: 0, max: 1, palette: palette}, 'Landsat-7 NDVI (2002)');

//calibrate Landsat-7 TOA Reflectance imagery and compare
//import Landsat-7 TOA Reflectance image collection; take the average of annual values
var LandsatTOA = ee.ImageCollection('LANDSAT/LE7_L1T_TOA').filterDate('2013-01-01','2013-12-31').mean();
print(LandsatTOA, 'LANDSAT/LE7_L1T_TOA');

//generate a bad pixel mask by examining the product of Bands 1-5 and Band 7
//if the product equals 0, the pixel should be eliminated
var mask = LandsatTOA.expression(
    'Blue * Green * Red * NIR * SWI1 * SWI2', {
      'Blue': LandsatTOA.select('B1'),
      'Green': LandsatTOA.select('B2'),
      'Red': LandsatTOA.select('B3'),
      'NIR': LandsatTOA.select('B4'),
      'SWI1': LandsatTOA.select('B5'),
      'SWI2': LandsatTOA.select('B7'),
});
print(mask, 'mask');
var mask_CA = mask.clip(CAstate);
Map.addLayer(mask_CA, {min: 1, max: 0, palette: '111111'}, 'mask');

//apply mask to the LandsatTOA image
var LandsatTOA = LandsatTOA.mask(mask);
print(LandsatTOA, 'Landsat TOA (calibrated)');
//calculate NDVI of the calibrated LandsatTOA image
var LandsatNDVI2 = LandsatTOA.normalizedDifference(['B4', 'B3']);
print(LandsatNDVI2, 'Landsat NDVI (calibrated)');
//clip images to study area (California)
var LandsatNDVI2_CA = LandsatNDVI2.clip(CAstate);
//add Landsat-7 TOA image from 2013 to the map
Map.addLayer(LandsatNDVI2_CA, {min: 0, max: 1, palette: palette}, 'Landsat-7 TOA (calibrated)');
/**
 * Here since the whole area is masked, no data from Landsat-7 is applicable in this case.
**/
```

Unfortunately, the whole study area is under the coverage of bad pixel mask, implying the Landsat-7 dataset is not suitable for this study.

MODIS NDVI Compostie

Average MODIS NDVI 16-day compostie (2013)

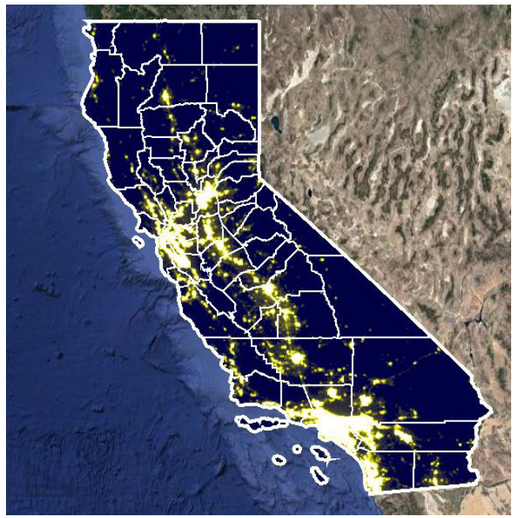


Average MODIS NDVI 16-day composite reclassified (2013)

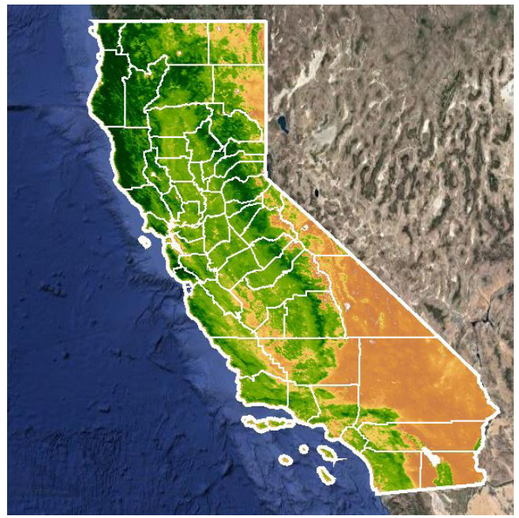
Despite the relatively lower resolution compared with Landsat, MODIS NDVI composite is an alternative source to supplement the urban intensity estimation by NTL. A collection of all MODIS 16-day NDVI images from 2013 are filtered by date, and reduced to a single NDVI composite which contains the average values of each pixel through our the year. Ideally though, to enhance the statistical robustness, NDVI values below zero (which are typically water) should be excluded in calculation.

```
//visualize light intensity classification
var c1 = MODISNDVI_CA.lt(0);
var c2 = MODISNDVI_CA.gt(0);
var c3 = MODISNDVI_CA.gt(0.2);
var c4 = MODISNDVI_CA.gt(0.4);
var c5 = MODISNDVI_CA.gt(0.6);
var c6 = MODISNDVI_CA.gt(0.8);

var CA_c1 = ee.Image(1);
var CA_c1 = CA_c1.mask(c1);
Map.addLayer(CA_c1, {min: 0, max: 1,palette: '#c9c7c7'},'c1');

var CA_c2 = ee.Image(1);
var CA_c2 = CA_c2.mask(c2);
Map.addLayer(CA_c2, {min: 0, max: 1,palette: '#0000FF'},'c2');

var CA_c3 = ee.Image(1);
var CA_c3 = CA_c3.mask(c3);
Map.addLayer(CA_c3, {min: 0, max: 1,palette: '#9aedca'},'c3');

var CA_c4 = ee.Image(1);
var CA_c4 = CA_c4.mask(c4);
Map.addLayer(CA_c4, {min: 0, max: 1,palette: '#00ff00'},'c4');

var CA_c5 = ee.Image(1);
var CA_c5 = CA_c5.mask(c5);
Map.addLayer(CA_c5, {min: 0, max: 1,palette: '#ffd670'},'c5');

var CA_c6 = ee.Image(1);
var CA_c6 = CA_c6.mask(c6);
Map.addLayer(CA_c6, {min: 0, max: 1,palette: '#ff0000'},'c6');
```


County/MSA-Level NDUI Compostie



Aggregate NTL values into California counties

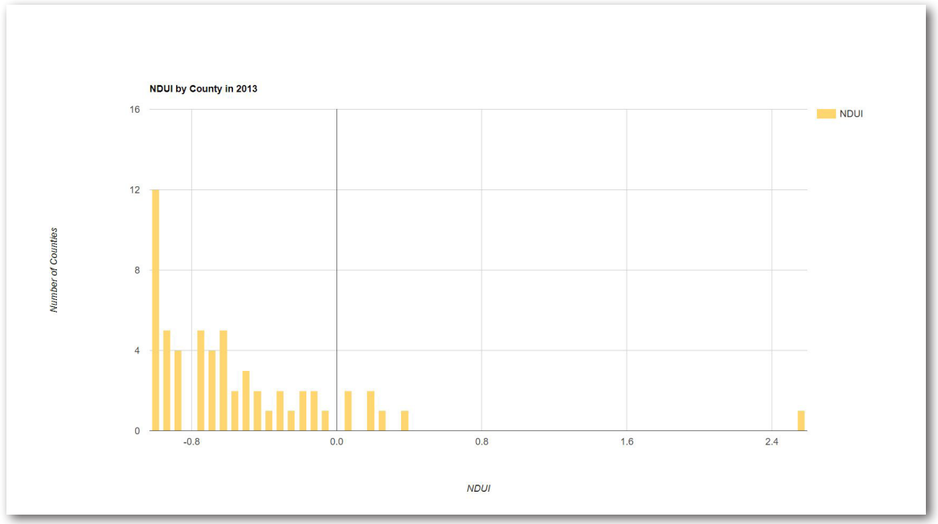


Aggregate MODIS NDVI values into California counties

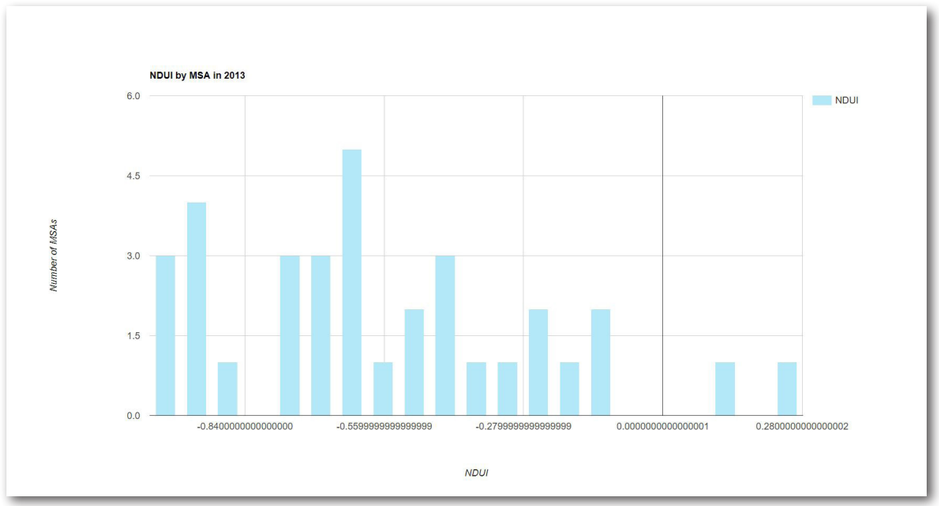
```
//aggreagte NTL and NDVI values into units of analysis

//get the mean of NTL values in each CA counties
var CACounty_NTL = NTL_CA.reduceRegions({
  collection: CACounty,
  reducer: ee.Reducer.mean(),
  scale: 1000,
});
// Print the first feature, to illustrate the result.
print(ee.Feature(CACounty_NTL.first()),'County NTL');

//get the mean of NDVI values in each CA counties
var CACounty_NDVI = MODISNDVI_CA.reduceRegions({
  collection: CACounty,
  reducer: ee.Reducer.mean(),
  scale: 1000,
});
// Print the first feature, to illustrate the result.
print(ee.Feature(CACounty_NDVI.first()),'County NDVI');

//join NTL and NDVI features of CA counties
//define an inner join.
var innerJoin = ee.Join.inner();

//specify an equals filter
var filter1 = ee.Filter.equals({
  leftField: 'COUNTYFP',
  rightField: 'COUNTYFP'
});

//apply the join
var innerJoined1 = innerJoin.apply(CACounty_NTL, CACounty_NDVI, filter1);
//display the join result
print('Inner join output:', innerJoined1);
```

```
//calculate Normalized Difference Urban Index (NDUI)
//apply a function to all objects in a feature collection
var field = function NDUI(feature) {
var thePrimaryFeature   = ee.Feature(feature.get('primary'));
var thePrimaryMean      = ee.Number(thePrimaryFeature.get('mean'));
var theSecondaryFeature = ee.Feature(feature.get('secondary'));
var theSecondaryMean    = ee.Number(theSecondaryFeature.get('mean'));
var theNumerator        = thePrimaryMean.subtract(theSecondaryMean);
var theDenominator      = thePrimaryMean.add(theSecondaryMean);
var CANDUI              = theNumerator.divide(theDenominator);
return feature.set({'NDUI': CANDUI});};

var innerJoined1 = ee.FeatureCollection(innerJoined1.map(field));
print('California Counties NDUI', innerJoined1.getInfo());

//visualize the results in histogram by counties
var TheCHART    = Chart.feature.histogram( innerJoined1, 'NDUI', 50 );
var TheCHART    = TheCHART.setSeriesNames( ['NDUI'] );
var TheCHART    = TheCHART.setOptions( { title:  'NDUI by County in 2013',
                                         colors: ['#ffd670'],
                                         hAxis:  {title:'NDUI'},
                                         vAxis:  {title:'Number of Counties'}});
print(TheCHART);
```

NDUI = (NTL – NDVI)/(NTL + NDVI)



Histogram of California NDUI by county (2013)



Histogram of California NDUI by MSA (2013)

County/MSA-level NDUI composite are visualized in chart on Google Earth Engine platform. The distribution is sparse while a majority of counties and MSAs have negative NDUI values.

Time-Series Analysis


```
California Counties NDUI
▾FeatureCollection (58 elements, 3 columns)
  type: FeatureCollection
  ▸columns: Object (3 properties)
  ▾features: List (58 elements)
    ▾0: Feature 1_1
        type: Feature
        id: 1_1
        geometry: null
      ▾properties: Object (3 properties)
          NDUI: -0.961047315006428
        ▾primary: Feature 1 (Polygon, 30 properties)
            type: Feature
            id: 1
          ▾geometry: Polygon, 4983 vertices
              type: Polygon
            ▸coordinates: List (1 element)
              geodesic: true
          ▸properties: Object (30 properties)
        ▾secondary: Feature 1 (Polygon, 30 properties)
            type: Feature
            id: 1
          ▸geometry: Polygon, 4983 vertices
          ▾properties: Object (30 properties)
              ALAND: 2468686374
              AWATER: 23299110
              CBSAFP:
              CLASSFP: H1
```


Finally the resulted NDUI values for each county and MSA are exported to csv tables using the Export.table.toDrive function. This makes it easier to gather data for time-series analysis by changing very few parameters. Due to the problem of using function within a loop, this script should be executive multiple times rather than fully automated.

```
//export the FeatureCollection to a CSV file
Export.table.toDrive({
  collection: innerJoined1,
                /** User-defined input**/
  description:'County_NDUI_2012',
  fileFormat: 'CSV'
});

Export.table.toDrive({
  collection: innerJoined2,
                /** User-defined input**/
  description: 'MSA_NDUI_2012',
  fileFormat: 'CSV'
});
```

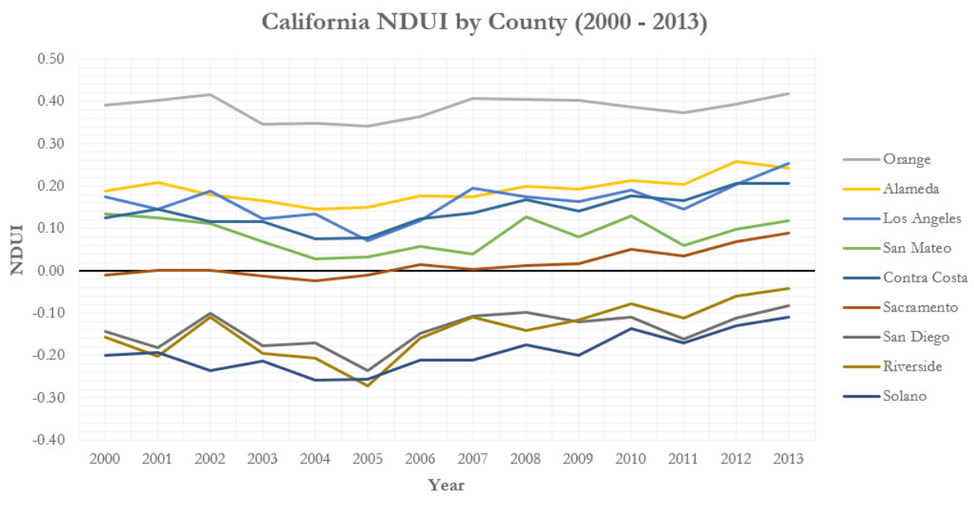
Noted in the following code, user is required to define two input image parameters each execution for time series outputs.

```
//import DMSP-OLS imagery, extract stable light band
                                  /** User-defined input**/
var NTL = ee.Image('NOAA/DMSP-OLS/NIGHTTIME_LIGHTS/F182012').expression('b(1)'); //select stable lights
//since the original band value has a range of 0 to 255, need to resample it to NTL's record range fro
var NTL = NTL.clamp(0,63);
//normalize NTL values
var NTL = NTL.expression('b(0)/63');

//import MODIS imagery
//create a new image based on the mean of NDVI of images from specific year
                                                    /** User-defined input**/
var MODISNDVI = ee.ImageCollection('MODIS/MCD43A4_NDVI').filterDate('2012-01-01','2012-12-31').mean();
```

Figures below demonstrate the time-series outputs at county and MSA level. The results coincide with intuition: 1) there is a steadying growing trend of average urban intensity for each county in California between 2000 and 2013, indicated by the increasing NDUI; 2) San Francisco has an exceptionally high NDUI, which might be due to its significantly small size; 3) counties with large cities exhibit higher NDUI values.

County	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
San Francisco	2.35685	3.11530	2.91144	2.14287	2.03696	2.14089	2.22508	2.53114	2.69513	2.13113	1.57132	2.57829	1.94477	2.57575
Orange	0.39178	0.40158	0.41664	0.34502	0.34900	0.34065	0.36457	0.40598	0.40476	0.40178	0.38686	0.37219	0.39407	0.41905
Alameda	0.18912	0.20818	0.18000	0.16499	0.14593	0.14910	0.17575	0.17471	0.20005	0.19297	0.21284	0.20377	0.25813	0.24171
Los Angeles	0.17527	0.14430	0.18886	0.12171	0.13399	0.07148	0.11714	0.19454	0.17497	0.16359	0.19005	0.14477	0.20383	0.25325
San Mateo	0.13432	0.12514	0.11027	0.06759	0.02870	0.03190	0.05811	0.03823	0.12661	0.08074	0.12989	0.05955	0.09870	0.11724
Contra Costa	0.12455	0.14405	0.11693	0.11692	0.07625	0.07861	0.12366	0.13662	0.16851	0.14130	0.17638	0.16498	0.20683	0.20515
Sacramento	-0.01047	-0.00028	0.00097	-0.01194	-0.02384	-0.00947	0.01489	0.00404	0.01282	0.01553	0.05147	0.03390	0.06749	0.08937
San Diego	-0.14457	-0.18136	-0.10115	-0.17669	-0.17056	-0.23578	-0.14784	-0.10817	-0.09764	-0.12023	-0.10859	-0.16090	-0.11300	-0.08169
Riverside	-0.15754	-0.20218	-0.10952	-0.19499	-0.20669	-0.27248	-0.15907	-0.10979	-0.14081	-0.11548	-0.07913	-0.11127	-0.06054	-0.04091
Solano	-0.19946	-0.19266	-0.23659	-0.21292	-0.25941	-0.25656	-0.21137	-0.21128	-0.17413	-0.20033	-0.13635	-0.17151	-0.13044	-0.10878

California NDUI by County (2000 - 2013)

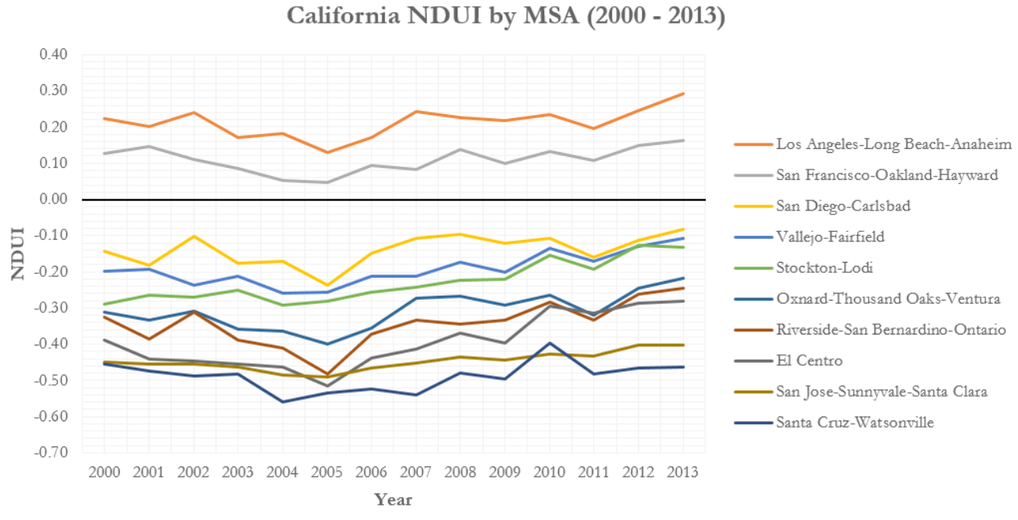


California NDUI by County (2000 - 2013)
Note: San Francisco is excluded from this chart due to exceptionally high value

The following table shows the annual average NDUI of California's top 10 MSAs between year 2000 and 2013. It shows similar patterns and results with those at county levels, while it is very intuitive that large metropolitan areas for example Los Angeles and San Francisco have significantly higher NDUI than other cities.

MSA	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
Los Angeles-Long Beach-Anaheim	0.22319	0.20019	0.24074	0.17135	0.18223	0.12996	0.17152	0.24152	0.22687	0.21769	0.23416	0.19589	0.24672	0.29168
San Francisco-Oakland-Hayward	0.12618	0.14532	0.10909	0.08526	0.05205	0.04810	0.09332	0.08240	0.13833	0.09990	0.13306	0.10640	0.14998	0.16295
San Diego-Carlsbad	-0.14458	-0.18137	-0.10115	-0.17670	-0.17057	-0.23579	-0.14784	-0.10817	-0.09765	-0.12024	-0.10859	-0.16091	-0.11301	-0.08169
Vallejo-Fairfield	-0.19946	-0.19266	-0.23659	-0.21292	-0.25941	-0.25656	-0.21137	-0.21128	-0.17413	-0.20033	-0.13635	-0.17151	-0.13044	-0.10878
Stockton-Lodi	-0.28830	-0.26380	-0.26912	-0.25210	-0.29323	-0.28082	-0.25528	-0.24178	-0.22236	-0.21979	-0.15544	-0.19320	-0.12648	-0.13217
Oxnard-Thousand Oaks-Ventura	-0.31179	-0.33322	-0.30929	-0.35837	-0.36296	-0.39922	-0.35452	-0.27356	-0.26813	-0.29340	-0.26448	-0.31928	-0.24654	-0.21644
Riverside-San Bernardino-Ontario	-0.32570	-0.38709	-0.31254	-0.38934	-0.41186	-0.48228	-0.37141	-0.33211	-0.34414	-0.33437	-0.28328	-0.33224	-0.26113	-0.24556
El Centro	-0.38854	-0.43969	-0.44775	-0.45426	-0.46307	-0.51572	-0.43740	-0.41390	-0.37019	-0.39761	-0.29419	-0.31469	-0.28707	-0.28153
San Jose-Sunnyvale-Santa Clara	-0.45017	-0.45449	-0.45527	-0.46401	-0.48588	-0.48982	-0.46551	-0.45228	-0.43538	-0.44347	-0.42751	-0.43323	-0.40228	-0.40166
Santa Cruz-Watsonville	-0.45454	-0.47267	-0.48729	-0.48152	-0.56027	-0.53402	-0.52240	-0.54127	-0.48036	-0.49614	-0.39713	-0.48118	-0.46555	-0.46258

NDUI of top 10 Califronia MSAs (2000 - 2013)

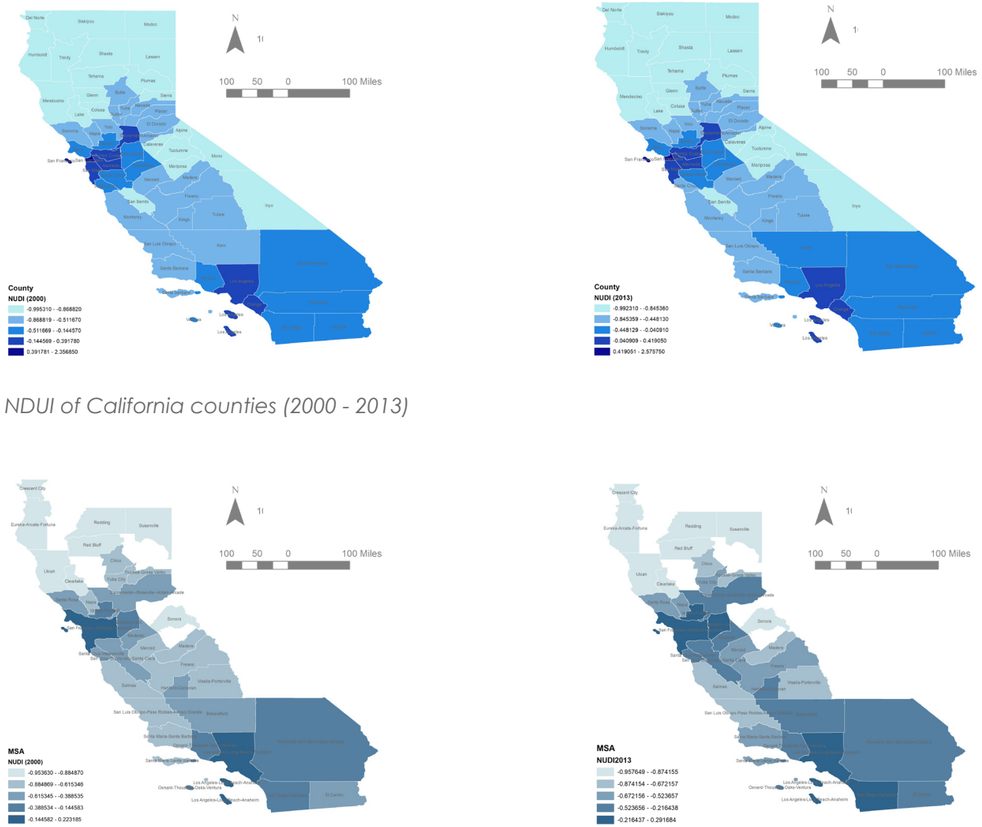


NDUI of top 10 Califronia MSAs (2000 - 2013)

ArcPY

Validating Estimated Urban Intensity
through Census and Urban Form Data

County/MSA-Level NDUI Compostie



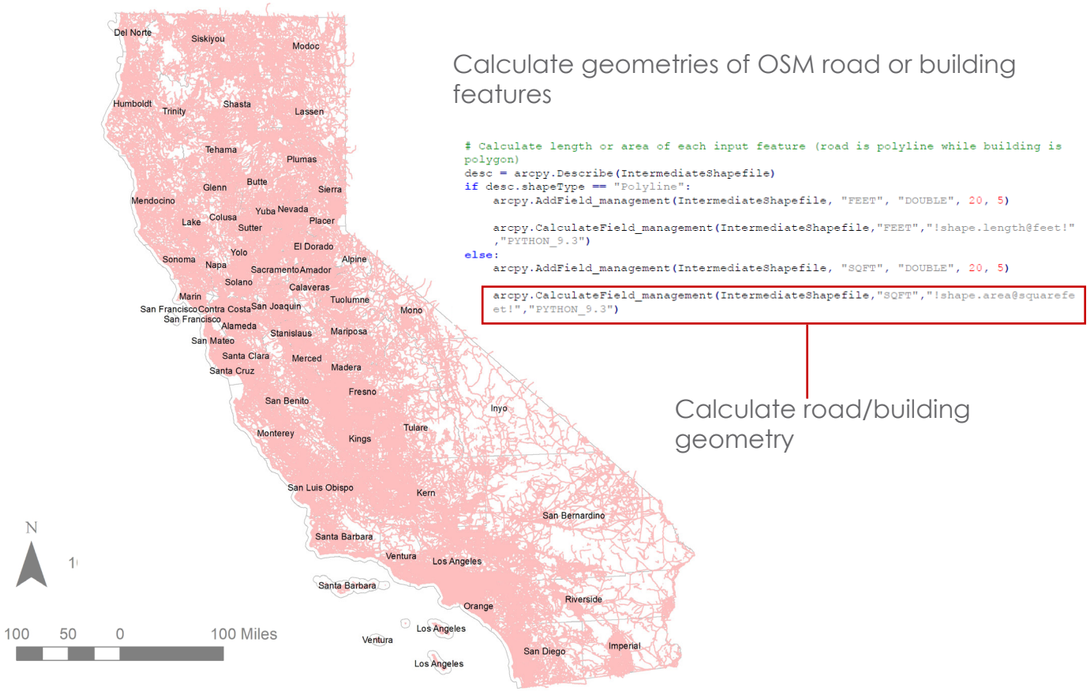
NDUI of California counties (2000 - 2013)

NDUI of Califronia MSAs (2000 - 2013)

NDUI outputs from Google Earth Engine are joined and illustrated in ArcGIS. As indicated by increasing average NDUI between 2000 and 2013, significant urban intensity growth happens in several MSAs such as Bakersfield, Merced, and Stockton-Lodi.

```
# Replicate the input shapefile
arcpy.Copy_management(InputShapefile, OutputShapefile)

# Join user-specified NUDI table to corresponding shapefile
arcpy.JoinField_management(in_data=OutputShapefile, in_field=InputField,
join table=JoinTable, join field=JoinField)
```


Spatial Join OSM Urban Form Features



Aggregate OSM road features to California county/MSA

Calculate geometries of OSM road or building features

```
# Calculate length or area of each input feature (road is polyline while building is
polygon)
desc = arcpy.Describe(IntermediateShapefile)
if desc.shapeType == "Polyline":
    arcpy.AddField_management(IntermediateShapefile, "FEET", "DOUBLE", 20, 5)

    arcpy.CalculateField_management(IntermediateShapefile,"FEET","!shape.length@feet!"
    ,"PYTHON_9.3")
else:
    arcpy.AddField_management(IntermediateShapefile, "SQFT", "DOUBLE", 20, 5)

    arcpy.CalculateField_management(IntermediateShapefile,"SQFT","!shape.area@squarefe
    et!","PYTHON_9.3")
```

Calculate road/building geometry

Spatial join calculated OSM road/ building featured to each unit of analysis (i.e. county/MSA)

```
# Spatial join and summarize road polyline or building polygon features to each unit
of analysis
targetFeatures  = County_MSA
joinFeatures    = IntermediateShapefile # The road/building features with calculated
length or area
outputFeatures  = Output

# create a list of fields to sum
fieldNamesToSum = ['FEET', 'SQFT']

# create the field mapping object
fieldMappings = arcpy.FieldMappings()

# populate the field mapping object with the fields from both feature classes
fieldMappings.addTable(targetFeatures)
fieldMappings.addTable(joinFeatures)

# loop through the field names to sum
for fieldName in fieldNamesToSum:

    # get the field map index of this field and get the field map
    fieldIndex = fieldMappings.findFieldMapIndex(fieldName)
    fieldMap = fieldMappings.getFieldMap(fieldIndex)

    # update the field map with the new merge rule (by default the merge rule is
    'First')
    fieldMap.mergeRule = 'Sum'

    # replace with the updated field map
    fieldMappings.replaceFieldMap(fieldIndex, fieldMap)

arcpy.SpatialJoin_analysis(targetFeatures, joinFeatures, outputFeatures,
"JOIN_ONE_TO_ONE", "KEEP_ALL", fieldMappings, "INTERSECT", "", "")
```

Calculate Socio-Economic Indicators

Total road length in each county

FEET	B00001e1	B00002e1	B01001e1	Area	Pop_den	Road_den
~~13628365.1945~~	310	203	3021	~~249136.5484~~	~~0.012123~~	~~54.688782~~
50916680.9032	108249	41943	1465832	257525.7431	5.691982	197.714917
37014091.9773	32851	11943	435850	981374.4485	0.444122	37.716584
16422794.8729	2592	1679	44767	268562.8767	0.166691	61.150651
39812251.3464	63802	22455	840833	571932.9182	1.47016	69.610002
205485425.664	783732	278300	10038388	1230537.2772	8.157728	166.988379
35675500.499	36086	15617	495078	457894.7125	1.081205	77.912017
23815567.9038	13408	3406	150998	360405.1011	0.418967	66.079997
115999413.023	226057	84459	3223096	1172151.0032	2.749728	98.962858
37656281.3974	26886	12076	366280	389072.285	0.941419	96.784795
8915284.37369	58086	25790	840763	60059.1091	13.998926	148.441835
16268020.4969	18520	7812	258349	214501.4226	1.204416	75.841084
16821130.5688	1447	734	17789	378869.3789	0.046953	44.398232
42241090.7273	4776	1426	32645	1222505.9506	0.026703	34.552871
10984474.4673	12822	5165	140295	204241.6974	0.686907	53.781743
46748977.0952	15250	6773	178942	996471.874	0.179576	46.914497
52505671.7174	33346	11202	428441	976747.8586	0.43864	53.755605
26649960.22	1297	1049	13373	830771.6131	0.016097	32.078564
46920423.9316	6931	3189	87544	1004436.9071	0.087157	46.713162
28059977.3721	2649	1322	18373	2648757.5325	0.006936	10.593638
20298789.366	707	674	14146	811141.7808	0.01744	25.024958
24237264.1541	4087	1940	54079	589080.0376	0.091802	41.144263
28942170.1957	31618	11673	425753	234702.7178	1.81401	123.314167
151090953.128	131284	46563	2094769	5207139.8312	0.402288	29.016112
43483340.3571	73510	27752	1096068	208175.0024	5.265128	208.878779
4803969.85588	290	325	1131	192480.0425	0.005876	24.958275
32167389.2193	12589	6378	182093	462679.8857	0.393562	69.524071
17910211.8201	14927	5476	207320	265098.6145	0.782049	67.560564
12635542.436	5258	1950	73437	166796.9873	0.440278	75.754021
17404243.7665	5131	1852	57557	360129.6869	0.159823	48.327712
45203733.9228	11826	5598	135034	1049530.6815	0.128661	43.070426
107501380.283	166561	64176	2298032	1891537.1763	1.214902	56.832814
128541222.184	58416	19310	865736	2114121.2251	0.409502	60.801254
11368854.299	1996	820	21396	299495.3986	0.07144	37.96003
11962212.7397	2656	928	27788	318504.3705	0.087245	37.557452
37464566.9894	914	510	9184	1088677.0366	0.008436	34.41293

```
# Calculate, record, and report the shape's t
population density (ppl/hectare)
TotalArea    = (landArea + waterArea) / 10000
PopDen       = Population / TotalArea
RoadDen      = Road / TotalArea
nextRecord.setValue(Field1,TotalArea)
nextRecord.setValue(Field2,PopDen)
nextRecord.setValue(Field3,RoadDen)
enumerationOfRecords.updateRow(nextRecord)
```

```
# Calculate average NTL across the year
NTL    =
(field2+field3+field4+field5+field6+field7+
ield13+field14+field15) / 14
NTL_popden = NTL * popden
NTL_roadde = NTL * roadden
nextRecord.setValue(Field4, NTL)
nextRecord.setValue(Field5, NTL_popden)
nextRecord.setValue(Field6, NTL_roadde)
enumerationOfRecords.updateRow(nextRecord)
```

After aggregating the OSM road/building features to each county/MSA, ArcGIS' field calculation tool assists the final computation of a few socio-economic and urban form indicators, for example population density, and road density.

Pearson Correlation

	NTL		NTL_popden		NTL_roadde
	-0.959856		-0.011636		-52.493379
	0.016749		0.095333		3.311442
	-0.685214		-0.304319		-25.843942
	-0.871585		-0.145285		-53.297991
	-0.307943		-0.452725		-21.435903
	0.162696		1.327233		27.168413
	-0.617677		-0.667836		-48.124472
	-0.553561		-0.231924		-36.579337
	-0.139155		-0.382638		-13.771176
	-0.553856		-0.52141		-53.604812
	2.354066		32.954391		349.441835
	-0.355696		-0.428406		-26.976349
	-0.970316		-0.045559		-43.080302
	-0.939864		-0.025098		-32.47501
	-0.630675		-0.433215		-33.918801
	-0.897986		-0.161256		-42.120582
	-0.746382		-0.327393		-40.122223
	-0.994209		-0.016004		-31.892806
	-0.959786		-0.083652		-44.834659
	-0.961922		-0.006672		-10.190255
	-0.941519		-0.01642		-23.561481

NDUI * Pop_den

NDUI * Road_den

NDUI for each county

$$\rho_{x,y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - [E(X)]^2}\sqrt{E(Y^2) - [E(Y)]^2}}$$

```
# Summary statistics
# I want to calculate the correlation between NTL and population density, as well as
NTL and road density, using Pearson's correlation.
arcpy.Statistics_analysis(in_table=Output, out_table=OutputTable,
statistics_fields="Pop_den MEAN;Pop_den STD;Road_den MEAN;Road_den STD;NTL MEAN;NTL
STD;NTL popden MEAN;NTL roadde MEAN")
```

Basically this table calculates all value inputs for the Pearson correlation equation including X,Y and XY, which are later summarize by the summary statistics table tool to mean and standard deviation.

Table

correlation

Rowid	FID	FREQUENCY	MEAN_NTL_POPDEN	MEAN_NTL_ROADDE	MEAN_NTL	STD_NTL	MEAN_POP_DEN	STD_POP_DEN	MEAN_ROAD_DEN	STD_ROAD_DEN
1	0	58	0.499933	-22.329595	-0.558898	0.529809	1.517792	2.908375	76.206301	58.065292

Finally, ArcGIS outputs a summary statistics table with values necessary for the calculation of Pearson correlations between NDUI and population density, as well as between NDUI and road density. By calculation, the correlation between NDUI and county population density is 0.875 while that between NDUI and road density is 0.659, which supports the hypothesis that NDUI is a good indicator for urban intensity. It is positively correlated with demographic factors such as population density, and urban spatial factors such as road density.

Although this demonstration only validates the correlations between NDUI and two other estimators, the same process can be applied to other variables. It can also examine such correlation at other geographic unit of analysis.

Discussion

Limitations

Despite the interesting results that indicates NTL supplemented by MODIS NDVI can produce quite reliable estimation of urban intensity, there are several limitations related to the application of Google Earth Engine and ArcPy tools used in this project. Firstly, image resolution is a primary factor that determines the quality of analysis. As discovered in the earlier section of this report, in order to conduct analysis at a finer scale, alternative sources such as Landsat (available at 30m spatial resolution) are identified. However, equipment failure makes it necessary to calibrate the data before applying to analysis. Some other factors that affect the input dataset include saturation and overglow of NTL in urban areas, weather and cloud that distract land cover detection. Moreover, the annual average of NDVI composite oversimplifies the seasonal patterns of vegetation density. While Pearson correlation is a reasonable estimate of the relationship among indicators, it does not imply any causality, thus requiring more rigorous statistical tests.

Next Steps

This exploratory study guides several potential directions to dig into, given time and resources availability. Since this study takes single data sources of one specific study area, it is necessary to taken a further step to look into more potential imagery sources (e.g. some readily calibrated DMSP/OLS dataset), and expand the spatial and temporal coverage of study area. Potentially, the unit of analysis could be those other than county or MSA. It will be interesting to see how the results changes across geography and time. Instead of taking the simple average of a series of available images throughout the year, future studies could take into consideration the seasonal patterns of vegetation growth as well as topography. Additional socio-economic data such as LEHD job-work flow could be a valuable source to incorporate so as to study the population dynamics, a crucial factor for urban dynamics. Also, it should keep track of the changes in satellite numbers to ensure consistency. Right now, this study only considers the stable light band, but other bands should be examined as well. In additional to the analysis based on geographic boundaries, a finer-grained method is to aggregate both raster and vector data to grid cells, and examine correlation at pixel-level. Both Google Earth Engine and ArcGIS will enable this method. One possible workflow is to export a geotiff image from selected region and resample the image to difference sizes of pixels in ArcGIS depending on the spatial resolution of original dataset. This raises the potential to develop a user interface that customize the selection of image source, study area, and scale of analysis. Due to the problem of using function within a loop in Google Earth Engine, extract effort would be worthwhile to develop an alternative method to automate exporting of time-series images.

References

1. Akiyama, Y. (2012). "Analysis of Light Intensity Data by the DMSP/OLS Satellite Image Using Existing Spatial Data for Monitoring Human Activity in Japan", ISPRS Annals of the Photogrammetry, *Remote Sensing and Spatial Information Sciences*, Volume I-2, XXII ISPRS Congress, 25 August – 01 September 2012, Melbourne, Australia.
2. Mellander C, Lobo J, Stolarick K, Matheson Z (2015). "Night-Time Light Data: A Good Proxy Measure for Economic Activity?", *PLoS ONE* 10(10).
3. Zhang, Qingling, Bin Li, David Thau, and Rebecca Moore (2015). "Building a Better Urban Picture: Combining Day and Night Remote Sensing Imagery", *Remote Sensing* 2015(7).
4. Zhang, Qingling, Christal Schaaf, Karen C. Seto (2013). "The Vegetation Adjusted NTL Urban Index: A new approach to reduce saturation and increase variation in nighttime luminosity", *Remote Sensing of Environment* 129(2013).

Data Sources

1. Google, Landsat 7 Annual NDVI Composite, 2002, 2013: *https://code.earthengine.google.com/dataset/LANDSAT/LE7_L1T_ANNUAL_NDVI*
2. Google, MODIS Combined 16-Day NDVI, 2000-2013: *https://code.earthengine.google.com/dataset/MODIS/MCD43A4_NDVI*
3. NOAA, DMSP-OLS Nighttime Lights Time Series Version 4, 2000-2013: *https://code.earthengine.google.com/dataset/NOAA/DMSP-OLS/NIGHTTIME_LIGHTS*
4. USGS, USGS Landsat 7 TOA Reflectance (Orthorectified), 2013: *https://code.earthengine.google.com/dataset/LANDSAT/LE7_L1T_TOA*
5. US Census Bureau, Cartographic Boundary Shapefiles: *https://www.census.gov/geo/maps-data/data/tiger-cart-boundary.html*

Complete Code

```
/**
 * Meiqing Li, meiqing@design.upenn.edu
 * CPLN670 Geospatical Software Design
 * Professor Dana Tomlin
 * December 20, 2017
 *
 * Estimation of Urban Intensity in California Through Day and Night Remote Sensing Imagery
 * link to code: https://code.earthengine.google.com/4713547bc3a71d6fbec07023eab4fa89
 *
 * This script explores and processes day and night remote sensing imageries from multiple
sources (i.e. DMSP-OLS,
 * Landsat-7, and MODIS), and develops a method to calculate urban intensity of different
aggregated geographic unit of analysis.
 * It generates an urban index for each area, and export the outputs in csv format, which
are ready for analysis
 * combining social and economic data in the same geographic units.
 * Taking California as study area, the script also identifies and visualizes urban and
vegetation distribution
 * within the state.
 *
 * In addition to identifying approperiate statellite imageries, data prepocessing includes
 * converting shapefile to the following Google fusion tables :
 * US states: 1nEYqvsp5Rz-Bcb65ciRK7oFJ8SHbghh3HxueptKv
 * CA counties: 1v0rngDKfy2cuBsedRlvtwjmc_BAK44XpPXUtSDVx
 * CA MSAs: 1hh6__GDkTvOrxPS0oUPZwswiAjWazFiamr6x8FxG
**/
```

Links to the code:

https://code.earthengine.google.com/291961fe8406ca4c75ecea4ef3f33435
https://code.earthengine.google.com/1ec1d414a4c5b03bbb0a7d6c19b5a748

```
//customize basemap
Map.setOptions('SATELLITE');

//color palette of NDVI
var palette = ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718',
               '74A901', '66A000', '529400', '3E8601', '207401', '056201',
               '004C00', '023B01', '012E01', '011D01', '011301'];

//collect the US state features
var USstate = ee.FeatureCollection('ft:1nEYqvsp5Rz-Bcb65ciRK7oFJ8SHbghh3HxueptKv'); //US
States
var CAstate = USstate.filterMetadata('NAME','equals','California'); //California State
//center to the study area
Map.centerObject(CAstate, 6);
//import CA county features
var CACounty = ee.FeatureCollection('ft:1v0rngDKfy2cuBsedRlvtwjmc_BAK44XpPXUtSDVx'); //CA
Counties
//import CA MSA features
var CAMSA = ee.FeatureCollection('ft:1hh6__GDkTvOrxPS0oUPZwswiAjWazFiamr6x8FxG'); //CA MSAs

//paint the state, county and MSA outlines

// Create an empty image into which to paint the features, cast to byte.
var empty = ee.Image().byte();

var outlines = empty.paint({
  featureCollection: CAstate,
  color: 1,
  width: 4
});
Map.addLayer(outlines, {palette: 'FFFFFF'}, 'California');

var empty = ee.Image().byte();
var outlines = empty.paint({
  featureCollection: CACounty,
  color: 1,
  width: 2
});
Map.addLayer(outlines, {palette: '111111'}, 'Counties');

var empty = ee.Image().byte();
var outlines = empty.paint({
  featureCollection: CAMSA,
  color: 1,
  width: 2
});
Map.addLayer(outlines, {palette: 'FFFFFF'}, 'MSAs');

//import DMSP-OLS imagery
//check DMSP-OLS image collection
var NTL = ee.ImageCollection('NOAA/DMSP-OLS/NIGHTTIME_LIGHTS'); //all night Lighting
print('Night Lighting Images', NTL);
//extract NTL image from 2013 (latest)
var NTL = ee.Image('NOAA/DMSP-OLS/NIGHTTIME_LIGHTS/F182013'); //night Lighting in 2013
print('Night Lighting Image 2013', NTL);
//extract stable light band
var NTL = ee.Image('NOAA/DMSP-OLS/NIGHTTIME_LIGHTS/F182013').expression('b(1)'); //select
stable lights
print('Stable Light Band 2013', NTL);
```

```
//since the original band value has a range of 0 to 255, need to resample it to NTL's record
range from 0 to 63
var NTL = NTL.clamp(0,63);
print(NTL,'NTL');
//clip images to study area (California)
var NTL_CA = NTL.clip(CAstate);

//add NTL image of California to the map
Map.addLayer(NTL_CA, {min: 0, max: 1, palette: ['000044','ffff00','ffffff'], opacity:
0.5},'Stable Light');

//classify light intensity by NTL values
//create masks
var c1 = NTL_CA.eq(0);
var c2 = NTL_CA.gt(0);
var c3 = NTL_CA.gt(20);
var c4 = NTL_CA.gt(30);
var c5 = NTL_CA.gt(40);
var c6 = NTL_CA.gt(50);

//visualize NTL classification
var CA_c1 = ee.Image(1);
var CA_c1 = CA_c1.mask(c1);
Map.addLayer(CA_c1, {min: 0, max: 1,palette: '#c9c7c7'},'c1');

var CA_c2 = ee.Image(1);
var CA_c2 = CA_c2.mask(c2);
Map.addLayer(CA_c2, {min: 0, max: 1,palette: '#0000FF'},'c2');

var CA_c3 = ee.Image(1);
var CA_c3 = CA_c3.mask(c3);
Map.addLayer(CA_c3, {min: 0, max: 1,palette: '#9aedca'},'c3');

var CA_c4 = ee.Image(1);
var CA_c4 = CA_c4.mask(c4);
Map.addLayer(CA_c4, {min: 0, max: 1,palette: '#00ff00'},'c4');

var CA_c5 = ee.Image(1);
var CA_c5 = CA_c5.mask(c5);
Map.addLayer(CA_c5, {min: 0, max: 1,palette: '#ffd670'},'c5');

var CA_c6 = ee.Image(1);
var CA_c6 = CA_c6.mask(c6);
Map.addLayer(CA_c6, {min: 0, max: 1,palette: '#ff0000'},'c6');


//classify urban, suburban and rural zones by NTL values
//create zonal masks
var rural = NTL_CA.gt(0);
var suburban = NTL CA.gt(15);
var urban = NTL_CA.gt(62);

//visualize zonal classification
var CA_rural = ee.Image(1);
var CA_rural = CA_rural.mask(rural);
Map.addLayer(CA_rural, {min: 0, max: 1,palette: '#0000FF'},'Rural');
```

```
var CA_suburban = ee.Image(1);
var CA_suburban = CA_suburban.mask(suburban);
Map.addLayer(CA_suburban, {min: 0, max: 1,palette: '#00FF00'},'Suburban');

var CA_urban = ee.Image(1);
var CA_urban = CA_urban.mask(urban);
Map.addLayer(CA_urban, {min: 0, max: 1,palette: '#FF0000'},'Urban');


//normalize NTL values for urban index calculation
var NTL_CA = NTL.expression('b(0)/63');
print(NTL_CA,'NTL_CA');


//import Landsat-7 imagery

//check Landsat-7 NDVI composite image collection
var LandsatNDVI = ee.ImageCollection('LANDSAT/LE7_L1T_ANNUAL_NDVI'); //all Landsat-7 NDVI
composite
print('LANDSAT/LE7_L1T_ANNUAL_NDVI', LandsatNDVI);
//extract 2013 Landsat-7 NVDI annual composite
var LandsatNDVI = ee.Image('LANDSAT/LE7_L1T_ANNUAL_NDVI/2013'); //Landsat-7 NDVI composite
in 2013
print('LANDSAT/LE7_L1T_ANNUAL_NDVI', LandsatNDVI);
//clip images to study area (California)
var LandsatNDVI_CA = LandsatNDVI.clip(CAstate);
//add Landsat-7 NDVI image from 2013 to the map
Map.addLayer(LandsatNDVI_CA, {min: 0, max: 1, palette: palette}, 'Landsat-7 NDVI (2013)');

//There has been a ETM+ scan line corrector (SLC) failure since May 2003.
//extract an image before 2003 and compare
var LandsatNDVI1 = ee.Image('LANDSAT/LE7_L1T_ANNUAL_NDVI/2002'); //Landsat-7 NDVI composite
in 2002
//clip images to study area (California)
var LandsatNDVI1_CA = LandsatNDVI1.clip(CAstate);
//add Landsat-7 NDVI image from 2002 to the map
Map.addLayer(LandsatNDVI1_CA, {min: 0, max: 1, palette: palette}, 'Landsat-7 NDVI (2002)');

//calibrate Landsat-7 TOA Reflectance imagery and compare
//import Landsat-7 TOA Reflectance image collection; take the average of annual values
var LandsatTOA =
ee.ImageCollection('LANDSAT/LE7_L1T_TOA').filterDate('2013-01-01','2013-12-31').mean();
print(LandsatTOA, 'LANDSAT/LE7_L1T_TOA');

//generate a bad pixel mask by examining the product of Bands 1-5 and Band 7
//if the product equals 0, the pixel should be eliminated
var mask = LandsatTOA.expression(
    'Blue * Green * Red * NIR * SWI1 * SWI2', {
      'Blue': LandsatTOA.select('B1'),
      'Green': LandsatTOA.select('B2'),
      'Red': LandsatTOA.select('B3'),
      'NIR': LandsatTOA.select('B4'),
      'SWI1': LandsatTOA.select('B5'),
      'SWI2': LandsatTOA.select('B7'),
});
```

```
print(mask, 'mask');
var mask_CA = mask.clip(CAstate);
Map.addLayer(mask_CA, {min: 1, max: 0, palette: '111111'}, 'mask');

//apply mask to the LandsatTOA image
var LandsatTOA = LandsatTOA.mask(mask);
print(LandsatTOA, 'Landsat TOA (calibrated)');
//calculate NDVI of the calibrated LandsatTOA image
var LandsatNDVI2 = LandsatTOA.normalizedDifference(['B4', 'B3']);
print(LandsatNDVI2, 'Landsat NDVI (calibrated)');
//clip images to study area (California)
var LandsatNDVI2_CA = LandsatNDVI2.clip(CAstate);
//add Landsat-7 TOA image from 2013 to the map
Map.addLayer(LandsatNDVI2_CA, {min: 0, max: 1, palette: palette}, 'Landsat-7 TOA
(calibrated)');
/**
 * Here since the whole area is masked, no data from Landsat-7 is applicable in this case.
**/

//import MODIS imagery
//check MODIS image collection
var MODISNDVI = ee.ImageCollection('MODIS/MCD43A4_NDVI'); //all MODIS combined 16-day NDVI
print('MODIS NDVI', MODISNDVI);
//create a new image based on the mean of NDVI of images from 2013
var MODISNDVI = MODISNDVI.filterDate('2013-01-01','2013-12-31').mean(); //take the average
of annual values
print('MODIS NDVI 2013', MODISNDVI);
//clip images to study area (California)
var MODISNDVI_CA = MODISNDVI.clip(CAstate);
//add MODIS NDVI image to the map
Map.addLayer(MODISNDVI_CA, {min: 0, max: 1, palette: palette}, 'MODIS NDVI');


//aggreagte NTL and NDVI values into units of analysis

//get the mean of NTL values in each CA counties
var CACounty_NTL = NTL_CA.reduceRegions({
  collection: CACounty,
  reducer: ee.Reducer.mean(),
  scale: 1000,
});
// Print the first feature, to illustrate the result.
print(ee.Feature(CACounty_NTL.first()),'County NTL');

//get the mean of NDVI values in each CA counties
var CACounty_NDVI = MODISNDVI_CA.reduceRegions({
  collection: CACounty,
  reducer: ee.Reducer.mean(),
  scale: 1000,
});
// Print the first feature, to illustrate the result.
print(ee.Feature(CACounty_NDVI.first()),'County NDVI');

//join NTL and NDVI features of CA counties
//define an inner join.
var innerJoin = ee.Join.inner();
```

```
//specify an equals filter
var filter1 = ee.Filter.equals({
  leftField: 'COUNTYFP',
  rightField: 'COUNTYFP'
});

//apply the join
var innerJoined1 = innerJoin.apply(CACounty_NTL, CACounty_NDVI, filter1);
//display the join result
print('Inner join output:', innerJoined1);

//calculate Normalized Difference Urban Index (NDUI)
//apply a function to all objects in a feature collection
var field = function NDUI(feature) {
var thePrimaryFeature   = ee.Feature(feature.get('primary'));
var thePrimaryMean      = ee.Number(thePrimaryFeature.get('mean'));
var theSecondaryFeature = ee.Feature(feature.get('secondary'));
var theSecondaryMean    = ee.Number(theSecondaryFeature.get('mean'));
var theNumerator        = thePrimaryMean.subtract(theSecondaryMean);
var theDenominator      = thePrimaryMean.add(theSecondaryMean);
var CANDUI              = theNumerator.divide(theDenominator);
return feature.set({'NDUI': CANDUI});};

var innerJoined1 = ee.FeatureCollection(innerJoined1.map(field));
print('California Counties NDUI', innerJoined1.getInfo());

//visualize the results in histogram by counties
var TheCHART    = Chart.feature.histogram( innerJoined1, 'NDUI', 50 );
var TheCHART    = TheCHART.setSeriesNames( ['NDUI'] );
var TheCHART    = TheCHART.setOptions( { title:  'NDUI by County in 2013',
                                         colors: ['#ffd670'],
                                         hAxis:  {title:'NDUI'},
                                         vAxis:  {title:'Number of Counties'}});
print(TheCHART);


//add reducer output to MSA features in the collection
var CAMSA_NTL = NTL_CA.reduceRegions({
  collection: CAMSA,
  reducer: ee.Reducer.mean(),
  scale: 1000,
});
print(ee.FeatureCollection(CAMSA_NTL),'MSA NTL');

var CAMSA_NDVI = MODISNDVI_CA.reduceRegions({
  collection: CAMSA,
  reducer: ee.Reducer.mean(),
  scale: 1000,
});
print(ee.FeatureCollection(CAMSA_NDVI),'MSA NDVI');

//join NTL and NDVI features of CA MSAs
//specify an equals filter
var filter2 = ee.Filter.equals({
  leftField: 'CBSAFP',
  rightField: 'CBSAFP'
});

//apply the join
var innerJoined2 = innerJoin.apply(CAMSA_NTL, CAMSA_NDVI, filter2);
//display the join result
print('Inner join output:', innerJoined2);

var innerJoined2 = ee.FeatureCollection(innerJoined2.map(field));
print('California MSAs NDUI', innerJoined2.getInfo());

//visualize the results in histogram by counties
var TheCHART    = Chart.feature.histogram( innerJoined2, 'NDUI', 30 );
var TheCHART    = TheCHART.setSeriesNames( ['NDUI'] );
var TheCHART    = TheCHART.setOptions( { title:  'NDUI by MSA in 2013',
                                         colors: ['#b2e8f7'],
                                         hAxis:  {title:'NDUI'},
                                         vAxis:  {title:'Number of MSAs'}});
print(TheCHART);

//export the FeatureCollection to CSV files
Export.table.toDrive({
  collection: innerJoined1,
  description:'County_NDUI',
  fileFormat: 'CSV'
});

Export.table.toDrive({
  collection: innerJoined2,
  description: 'MSA_NDUI',
  fileFormat: 'CSV'
});
```

```
/**
 * Following is a cleaned up script for exporting time series data.
 * User-defined inputs are highlighted.
**/

//Time Series Output

//collect the US state features
var USstate = ee.FeatureCollection('ft:1nEYqvsp5Rz-Bcb65ciRK7oFJ8SHbghh3HxueptKv'); //US
States
var CAstate = USstate.filterMetadata('NAME','equals','California'); //California State

//import CA county features
var CACounty = ee.FeatureCollection('ft:1v0rngDKfy2cuBsedRlvtwjmc_BAK44XpPXUtSDVx'); //CA
Counties

//import CA MSA features
var CAMSA = ee.FeatureCollection('ft:1hh6__GDkTvOrxPS0oUPZwswiAjWazFiamr6x8FxG'); //CA MSAs

//import DMSP-OLS imagery, extract stable light band
var NTL = ee.Image('NOAA/DMSP-OLS/NIGHTTIME_LIGHTS/F182012').expression('b(1)'); //select
stable lights;
//since the original band value has a range of 0 to 255, need to resample it to NTL's recorc
range from 0 to 63
var NTL = NTL.clamp(0,63);
//normalize NTL values
var NTL = NTL.expression('b(0)/63');

//import MODIS imagery
//create a new image based on the mean of NDVI of images from specific year
var MODISNDVI =
ee.ImageCollection('MODIS/MCD43A4_NDVI').filterDate('2012-01-01','2012-12-31').mean();
//take the average of annual values

//clip images to study area (California)
var NTL_CA = NTL.clip(CAstate);
var MODISNDVI_CA = MODISNDVI.clip(CAstate);

//aggreagte NTL and NDVI values into units of analysis

//get the mean of NTL values in each CA counties
var CACounty_NTL = NTL_CA.reduceRegions({
  collection: CACounty,
  reducer: ee.Reducer.mean(),
  scale: 1000,
});

//get the mean of NDVI values in each CA counties
var CACounty_NDVI = MODISNDVI_CA.reduceRegions({
  collection: CACounty,
  reducer: ee.Reducer.mean(),
  scale: 1000,
});

//join NTL and NDVI features of CA counties
//define an inner join.
var innerJoin = ee.Join.inner();

//specify an equals filter
var filter1 = ee.Filter.equals({
  leftField: 'COUNTYFP',
  rightField: 'COUNTYFP'
});
```



```
//apply the join
var innerJoined1 = innerJoin.apply(CACounty_NTL, CACounty_NDVI, filter1);
//display the join result

//calculate Normalized Difference Urban Index (NDUI)
var field = function NDUI(feature) {
var thePrimaryFeature = ee.Feature(feature.get('primary'));
var thePrimaryMean    = ee.Number(thePrimaryFeature.get('mean'));
var theSecondaryFeature = ee.Feature(feature.get('secondary'));
var theSecondaryMean    = ee.Number(theSecondaryFeature.get('mean'));
var theNumerator      = thePrimaryMean.subtract(theSecondaryMean);
var theDenominator   = thePrimaryMean.add(theSecondaryMean);
var CANDUI               = theNumerator.divide(theDenominator);
return feature.set({'NDUI': CANDUI});};

var innerJoined1 = ee.FeatureCollection(innerJoined1.map(field));

//add reducer output to MSA features in the collection
var CAMSA_NTL = NTL_CA.reduceRegions({
  collection: CAMSA,
  reducer: ee.Reducer.mean(),
  scale: 1000,
});

var CAMSA_NDVI = MODISNDVI_CA.reduceRegions({
  collection: CAMSA,
  reducer: ee.Reducer.mean(),
  scale: 1000,
});

//join NTL and NDVI features of CA MSAs
//specify an equals filter
var filter2 = ee.Filter.equals({
  leftField: 'CBSAFP',
  rightField: 'CBSAFP'
});

//apply the join
var innerJoined2 = innerJoin.apply(CAMSA_NTL, CAMSA_NDVI, filter2);
//display the join result

var innerJoined2 = ee.FeatureCollection(innerJoined2.map(field));

//export the FeatureCollection to a CSV file
Export.table.toDrive({
  collection: innerJoined1,
  description:'County_NDUI_2012',
  fileFormat: 'CSV'
});

Export.table.toDrive({
  collection: innerJoined2,
  description: 'MSA_NDUI_2012',
  fileFormat: 'CSV'
});
```

```
#Validation of Normalized Difference Urban Index (NDUI) in California with Socio-Economic Data

"""
This script first joins the output csv tables containing the normalized urban difference
index (NUDI) calculated in Google Earth Engine from the earlier part of this project.
It then validates index by aggregating socio-economic and spatial data from ACS census and OSM
to the same geographic units of analysis (i.e. county, MSA) with the previous output from
raster calculation. Finally it outputs a standardized table for more robust statistical
analysis.

The tool consists of three parts:
1) Join NUDI tables to their corresponding Count/MSA shapefiles;
2) Calculate road length (or building area) of OSM features, spatially join OSM
raod/building features to their correponding geographic unit of analysis, and summarize;
3) Join socio-economic tables to the output shapefiles from previous steps, and calculate
Pearson Correlations.

It requires the following inputs:
1. NUDI tables by county and MSA from Google Earth Engine project;
2. County and MSA boundary shapefiles;
3. OSM road and building shapefiles;
4. Tables ACS 5-Year Census.

Data sources:
Normalized Urban Different Index (NUDI) for each geographic unit of analysis is calculated
based on day and night remote sensing imagery (see Google Earth Engine project).
County and MSA shapefiles are from US Census Bureau, Cartographic Boundary Shapefiles:
https://www.census.gov/geo/maps-data/data/tiger-cart-boundary.html
OSM shapefiles of California are downloaded from:
http://download.geofabrik.de/north-america.html
Other data preprocessing includes setting standardized projection of input shapefiles, final
calculation of the Pearson Correlation based on output of summary statistics.
"""



#       ***The project consists of three parts. This is the first part of the script.***

"""
Following is the instruction for applying the tool to join NDUI table (input should be in
dbf format) to the input shapefiles based on County/MSA name,
and export a new shapefile:

To create an ArcToolbox tool with which to execute this script, do the following.
1   In  ArcMap > Catalog > Toolboxes > My Toolboxes, either select an existing toolbox
    or right-click on My Toolboxes and use New > Toolbox to create (then rename) a new one.
2   Drag (or use ArcToolbox > Add Toolbox to add) this toolbox to ArcToolbox.
3   Right-click on the toolbox in ArcToolbox, and use Add > Script to open a dialog box.
4   In this Add Script dialog box, use Label to name the tool being created, and press Next.
5   In a new dialog box, browse to the .py file to be invoked by this tool, and press Next.
6   In the next dialog box, specify the following inputs (using dropdown menus wherever
possible)
    before pressing OK or Finish.
        DISPLAY NAME        DATA TYPE       PROPERTY>DIRECTION>VALUE
        Input Shapefile     Shapefile       Input
        NUDI Table          Data Table      Input
        Join Field 1        String          Input
        Join Field 2        String          Input
        Output Shapefile    Shapefile       Output

7   The PROPERTIES > VALIDATION TAB should then be specified as follows.

import arcpy
class ToolValidator():

  def __init__(self):
    import arcpy
    self.params = arcpy.GetParameterInfo()
    return

  def initializeParameters(self):
    # SET DEFAULT TITLE
    self.params[1].value = "??"    #Self refers to the dialog box
    return

  def updateMessages(self):
    if self.params[1].value[-1] != "f":
      self.params[1].setWarningMessage("Please specify a dbf table.")
    return

8   To later revise any of this, right-click to the tool's name and select
Properties.
"""
```



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

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

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

    try:
        # Request user input of data type = Feature Layer and direction = Input
        InputShapefile      = arcpy.GetParameterAsText(0)
        JoinTable           = arcpy.GetParameterAsText(1)
        InputField          = arcpy.GetParameterAsText(2)
        JoinField           = arcpy.GetParameterAsText(3)
        OutputShapefile     = arcpy.GetParameterAsText(4)

        # Replicate the input shapefile
        arcpy.Copy_management(InputShapefile, OutputShapefile)

        # Join user-specified NUDI table to corresponding shapefile
        arcpy.JoinField_management(in_data=OutputShapefile, in_field=InputField,
        join_table=JoinTable, join_field=JoinField)

    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")
```

```
#       ***The project consists of three parts. This is the second part of the script.***

"""
Following is the instruction for applying the tool to calculate the length of OSM road line
shapefile, or area of building polygon shapefile,
spatially join these features to the county and MSA shapefiles with NUDI joined from
previous steps, and output a new shapefile with the summary
of road length and building area in each unit of analysis.

Input OSM shapefiles are required to be projected in the same state plane projection with
the county/MSA shapefile.
In this example, make sure input shapefiles are projected to
'NAD_1983_2011_StatePlane_California_III_FIPS_0403_Ft_US'.

To create an ArcToolbox tool with which to execute this script, do the following.
1   In  ArcMap > Catalog > Toolboxes > My Toolboxes, either select an existing toolbox
    or right-click on My Toolboxes and use New > Toolbox to create (then rename) a new one.
2   Drag (or use ArcToolbox > Add Toolbox to add) this toolbox to ArcToolbox.
3   Right-click on the toolbox in ArcToolbox, and use Add > Script to open a dialog box.
4   In this Add Script dialog box, use Label to name the tool being created, and press Next.
5   In a new dialog box, browse to the .py file to be invoked by this tool, and press Next.
6   In the next dialog box, specify the following inputs (using dropdown menus wherever
possible)
    before pressing OK or Finish.
        DISPLAY NAME        DATA TYPE       PROPERTY>DIRECTION>VALUE
        Road / Building     Shapefile       Input
        County / MSA        Shapefile       Input
        Output              Shapefile       Output

7   To later revise any of this, right-click to the tool's name and select
Properties.
"""
```



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

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

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

    try:
        # Request user input of data type = Feature Layer and direction = Input
        Road_Building       = arcpy.GetParameterAsText(0)
        County_MSA          = arcpy.GetParameterAsText(1)
        Output              = arcpy.GetParameterAsText(2)

        # Define intermediate Shapefile to store temporary outputs
        IntermediateShapefile = Output[:-4] + "_temp" + ".shp"

        # Replicate the input shapefile
        arcpy.Copy_management(Road_Building, IntermediateShapefile)

        # Calculate length or area of each input feature (road is polyline while building is
        polygon)
        desc = arcpy.Describe(IntermediateShapefile)
        if desc.shapeType == "Polyline":
            arcpy.AddField_management(IntermediateShapefile, "FEET", "DOUBLE", 20, 5)
            arcpy.CalculateField_management(IntermediateShapefile,"FEET","!shape.length@feet!"
            ,"PYTHON_9.3")
        else:
            arcpy.AddField_management(IntermediateShapefile, "SQFT", "DOUBLE", 20, 5)

            arcpy.CalculateField_management(IntermediateShapefile,"SQFT","!shape.area@squarefe
            et!","PYTHON_9.3")


        # Spatial join and summarize road polyline or building polygon features to each unit
        of analysis
        targetFeatures  = County_MSA
        joinFeatures    = IntermediateShapefile # The road/building features with calculated
        length or area
        outputFeatures  = Output

        # create a list of fields to sum
        fieldNamesToSum = ['FEET', 'SQFT']

        # create the field mapping object
        fieldMappings = arcpy.FieldMappings()

        # populate the field mapping object with the fields from both feature classes
        fieldMappings.addTable(targetFeatures)
        fieldMappings.addTable(joinFeatures)

        # loop through the field names to sum
        for fieldName in fieldNamesToSum:

            # get the field map index of this field and get the field map
            fieldIndex = fieldMappings.findFieldMapIndex(fieldName)
            fieldMap = fieldMappings.getFieldMap(fieldIndex)

            # update the field map with the new merge rule (by default the merge rule is
            'First')
            fieldMap.mergeRule = 'Sum'

            # replace with the updated field map
            fieldMappings.replaceFieldMap(fieldIndex, fieldMap)

        arcpy.SpatialJoin_analysis(targetFeatures, joinFeatures, outputFeatures,
        "JOIN_ONE_TO_ONE", "KEEP_ALL", fieldMappings, "INTERSECT", "", "")

        # Since each shapefile-generating method (unlike grid-generating methods)
        immediately writes its output to disk,
        # final results need not be explicitly saved, but intermediate results must be
        explicitly deleted.
        arcpy.Delete_management(IntermediateShapefile)

    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")
```

```
#       ***The project consists of three parts. This is the third part of the script.***

"""
Following is the instruction for applying the tool to join socio-economic data from censu
(input should be in dbf format) to the input shapefiles based on GEOID,
and export a new shapefile. It then does a series of calculation based on field values tc
get the Pearson Correlation
between NTL urban index (NUDI) and population and road density:

To create an ArcToolbox tool with which to execute this script, do the following.
1   In  ArcMap > Catalog > Toolboxes > My Toolboxes, either select an existing toolbox
    or right-click on My Toolboxes and use New > Toolbox to create (then rename) a new on
2   Drag (or use ArcToolbox > Add Toolbox to add) this toolbox to ArcToolbox.
3   Right-click on the toolbox in ArcToolbox, and use Add > Script to open a dialog box.
4   In this Add Script dialog box, use Label to name the tool being created, and press Ne
5   In a new dialog box, browse to the .py file to be invoked by this tool, and press Nex
6   In the next dialog box, specify the following inputs (using dropdown menus wherever
possible)
    before pressing OK or Finish.
        DISPLAY NAME        DATA TYPE       PROPERTY>DIRECTION>VALUE
        Input Shapefile     Shapefile       Input
        Join Table          Data Table      Input
        Join Field 1        String          Input
        Join Field 2        String          Input
        Output Shapefile    Shapefile       Output
        Output Table        Table           Output

7   To later revise any of this, right-click to the tool's name and select
Properties.
"""
```



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

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

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

    try:
        # Request user input of data type = Feature Layer and direction = Input
        InputShapefile      = arcpy.GetParameterAsText(0)
        JoinTable           = arcpy.GetParameterAsText(1)
        InputField          = arcpy.GetParameterAsText(2)
        JoinField           = arcpy.GetParameterAsText(3)
        OutputShapefile     = arcpy.GetParameterAsText(4)
        OutputTable         = arcpy.GetParameterAsText(5)

        # Replicate the input shapefile
        arcpy.Copy_management(InputShapefile, OutputShapefile)

        # First delete the irrelevant fields from previous output
        # Drop certain fields from the attribute table of the input shapefile
        arcpy.DeleteField_management(in_table=OutputShapefile,
        drop_field="osm_id;code;fclass;name_1;ref;oneway;maxspeed;layer;bridge;tunnel")

        # Join user-specified table to the shapefile
        arcpy.JoinField_management(in_data=OutputShapefile, in_field=InputField,
        join_table=JoinTable, join_field=JoinField)
        # Here I joined the population of each county by GEOID.
```

```
        # Calculate population and road density in each county
        # Request user input of data type = String and direction = Input
        Field1 = "Area" # In hectares
        arcpy.AddMessage("The name of the field to be added is " + Field1 + "\n")

        Field2 = "Pop_den"
        arcpy.AddMessage("The name of the field to be added is " + Field2 + "\n")

        Field3 = "Road_den"
        arcpy.AddMessage("The name of the field to be added is " + Field3 + "\n")

        # Add new fields to the output shapefile
        arcpy.AddField_management(OutputShapefile, Field1, "DOUBLE", 20, 5)
        arcpy.AddField_management(OutputShapefile, Field2, "DOUBLE", 20, 5)
        arcpy.AddField_management(OutputShapefile, Field3, "DOUBLE", 20, 5)

        # Create an enumeration of updatable records from the shapefile's attribute table
        enumerationOfRecords = arcpy.UpdateCursor(OutputShapefile)

        # Loop through enumeration of records, calculating each geographic unit's total a
        (in hectares) and population density (ppl/hectare)
        for nextRecord in enumerationOfRecords:
            nextRecord.setValue(Field6, NTL_roadde)
            enumerationOfRecords.updateRow(nextRecord)

        # Delete row and update cursor objects to avoid locking attribute table
        del nextRecord
        del enumerationOfRecords

        # Summary statistics
            # I want to calculate the correlation between NTL and population density, as well as
            NTL and road density, using Pearson's correlation.
            arcpy.Statistics_analysis(in_table=Output, out_table=OutputTable,
            statistics_fields="Pop_den MEAN;Pop_den STD;Road_den MEAN;Road_den STD;NTL MEAN;NTL
            STD;NTL_popden MEAN;NTL_roadde MEAN")

        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")

                # Retrieve attribute values (land areas and water areas) from the next record's
                Shape field
                landArea    = nextRecord.getValue("ALAND") # the original values are in square
                meters
                waterArea   = nextRecord.getValue("AWATER")# the original values are in square
                meters
                Population = nextRecord.getValue("B01001e1")
                Road       = nextRecord.getValue("FEET")
```

```
    # Calculate, record, and report the shape's total area (in hectares) and
    population density (ppl/hectare)
    TotalArea   = (landArea + waterArea) / 10000
    PopDen      = Population / TotalArea
    RoadDen     = Road / TotalArea
    nextRecord.setValue(Field1,TotalArea)
    nextRecord.setValue(Field2,PopDen)
    nextRecord.setValue(Field3,RoadDen)
    enumerationOfRecords.updateRow(nextRecord)

# Delete row and update cursor objects to avoid locking attribute table
del nextRecord
del enumerationOfRecords

# Calculate average NTL/NUDI across the years
# Calculate Pearson Correlation between NTL and population and road density
Field4 = "NTL" # average across the years
arcpy.AddMessage("The name of the field to be added is " + Field4 + "\n")

Field5 = "NTL_popden"
arcpy.AddMessage("The name of the field to be added is " + Field5 + "\n")

Field6 = "NTL_roadde"
arcpy.AddMessage("The name of the field to be added is " + Field6 + "\n")

# Add new fields to the output shapefile
arcpy.AddField_management(OutputShapefile, Field4, "DOUBLE", 20, 5)
arcpy.AddField_management(OutputShapefile, Field5, "DOUBLE", 20, 5)
arcpy.AddField_management(OutputShapefile, Field6, "DOUBLE", 20, 5)

# Create an enumeration of updatable records from the shapefile's attribute table
enumerationOfRecords = arcpy.UpdateCursor(OutputShapefile)

for nextRecord in enumerationOfRecords:
    field2   = nextRecord.getValue("Field2")
    field3   = nextRecord.getValue("Field3")
    field4   = nextRecord.getValue("Field4")
    field5   = nextRecord.getValue("Field5")
    field6   = nextRecord.getValue("Field6")
    field7   = nextRecord.getValue("Field7")
    field8   = nextRecord.getValue("Field8")
    field9   = nextRecord.getValue("Field9")
    field10   = nextRecord.getValue("Field10")
    field11   = nextRecord.getValue("Field11")
    field12   = nextRecord.getValue("Field12")
    field13   = nextRecord.getValue("Field13")
    field14   = nextRecord.getValue("Field14")
    field15   = nextRecord.getValue("Field15")
    popden    = nextRecord.getValue("Pop_den")
    roadden   = nextRecord.getValue("Road_den")

    # Calculate average NTL across the year
    NTL   =
    (field2+field3+field4+field5+field6+field7+field8+field9+field10+field11+field12+f
    ield13+field14+field15) / 14
    NTL_popden = NTL * popden
    NTL_roadde = NTL * roadden
    nextRecord.setValue(Field4, NTL)
    nextRecord.setValue(Field5, NTL_popden)
        nextRecord.setValue(Field6, NTL_roadde)
        enumerationOfRecords.updateRow(nextRecord)

    # Delete row and update cursor objects to avoid locking attribute table
    del nextRecord
    del enumerationOfRecords

    # Summary statistics
    # I want to calculate the correlation between NTL and population density, as well as
    NTL and road density, using Pearson's correlation.
    arcpy.Statistics_analysis(in_table=Output, out_table=OutputTable,
    statistics_fields="Pop_den MEAN;Pop_den STD;Road_den MEAN;Road_den STD;NTL MEAN;NTL
    STD;NTL_popden MEAN;NTL_roadde MEAN")

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")
```

University of Pennsylvania
CPLN 670 Geospatial Software Design
Meiqing Li | Fall 2017