LARP 743 Final Project

Electric Vehicle & Environmental Change

Yian Wang

Instructor: Dana Tomlin

Introduction

The serious energy and environmental crisis caused by this situation is universal in all countries of the world. Therefore, people nowadays are attaching more and more importance to the development of electric vehicles (EV) to effectively alleviate the energy crisis and promote social and economic development.

There are many reasons why one might consider making the switch to an EV. Electric cars are higher efficiency than gas-powered cars, can reduce your dependence on fossil fuels and require less maintenance than most cars. In addition to that, if your EV run solely on electric power produced by 100% on sustainable, renewable energy resources like wind, solar, geothermal, biomass, and hydropower, it will have better environmental impacts than traditional vehicles.

However, the exact type and intensity of environmental impacts varies on the specific technology used, the geographic location and other factors. In this project, we would like to detect environmental changes by green land and air quality in U.S. states where has the most rapid and advanced growth in electric vehicle usage.

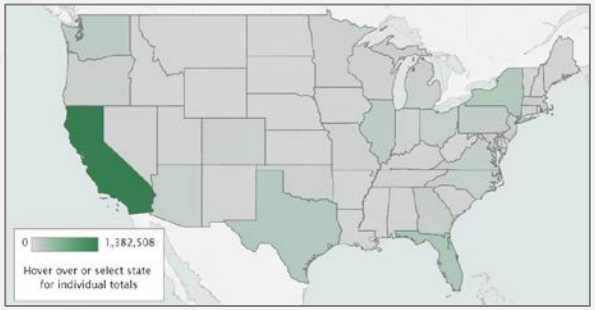


Fig 1 Sales of Electric Vehicle by State

Workflow

State Selection:
- Join Tables
- Calculate sales change
- Calculate generation
- Sort and intersect

Land Change Visualization:
- A Split Panel

Green Change Calculation:
- Remap and subtract
- Frequency Histogram
- Filter and Select
- Reducer Histogram
- Further Analysis

Air Quality Difference:
- Data Transform
- Spatial Join Analysis
- Extract and Compare
- Further Analysis

Analysis Tool

- Google Earth Engine
- ArcPy

Data Source

- Alliance: Advanced Technology Vehicle Sales Dashboard
- NASA LP DAAC: MODIS Land Cover Type Yearly Global 500m
- EIA: Generation of Electricity by Source
- EPA: Air Quality Data Collected at Outdoor Monitors

Code & Result - GEE

Part ONE – State Selection by Sales Change and Energy Source

```
// import all data needed
var state = ee.FeatureCollection("TIGER/2016/States");
var rawdata = ee.FeatureCollection("users/annisann666/data16to18");
var land18 = ee.Image("MODIS/006/MCD12Q1/2018_01_01");
var land16 = ee.Image("MODIS/006/MCD12Q1/2016_01_01");

// setup and some palettes
Map.centerObject(state, 4);
var palette1 = ['ffff55','ff0000'];
var palette2 = ['A8DFFE','3F2BFF'];

// join raw data with state features
var joinfilter = ee.Filter.equals('NAME', null, 'State', null );
var thejoin = ee.Join.inner();
var join = thejoin.apply(state, rawdata, joinfilter);
var datastate = join.map(function(featurepair) {
  var feature1 = ee.Feature(featurepair.get('primary'));
  var feature2 = ee.Feature(featurepair.get('secondary'));
  return feature1.set(feature2.toDictionary());
});

// add sales change from 2016 to 2018
var salechange = function(feature){
  var sale16 = ee.Number(feature.get('EV Sales 2016'));
  var sale18 = ee.Number(feature.get('EV Sales 2018'));
  return feature.set({'sale change from 16 to
18':sale18.subtract(sale16).divide(sale16).multiply(100).log()});};
var salestate = datastate.map(salechange);

// select and map top sale change states
var topchange = salestate.sort('sale change from 16 to 18', false).limit(10);
var topchangelist = topchange.aggregate_array('NAME');
print('Top EV Sales Change:', topchangelist);
var empty = ee.Image().byte();
var maptopsale = empty.paint({featureCollection: topchange, color: 'sale change from 16 to 18'});
Map.addLayer(maptopsale, {palette:palette2, min:5, max:7, opacity:0.7}, 'Top EV Sales change from 16
to 18');
```

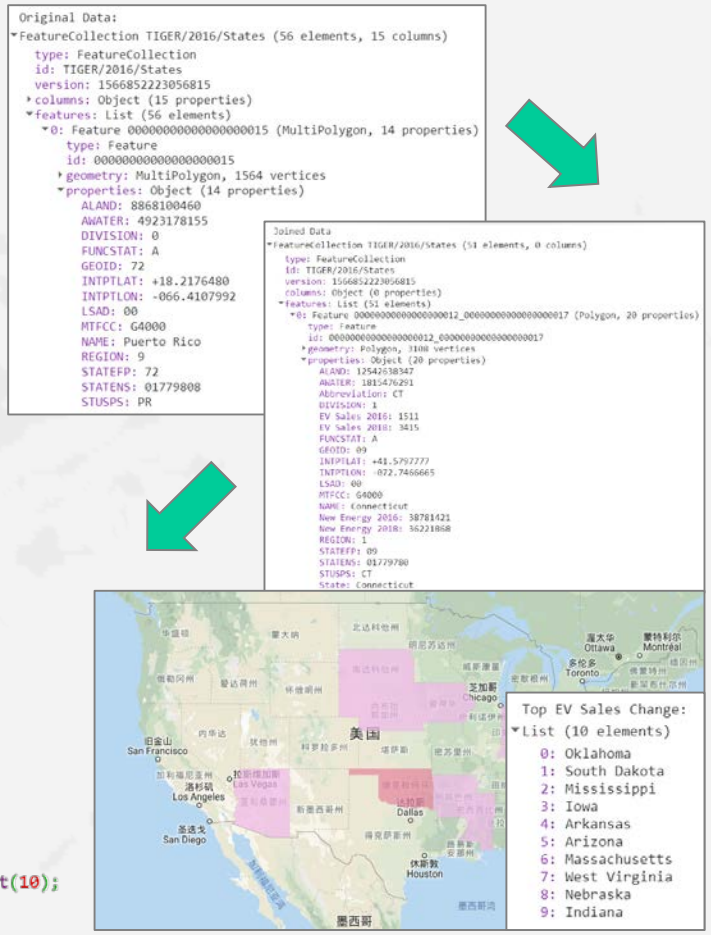


Fig 2
Top 10 Electric Vehicle Sales Change by Statec

Code & Result - GEE

Part ONE – State Selection by Sales Change and Energy Source

```
// add generation change from 2016 to 2018
var genechange = function(feature){
  var gene16 = ee.Number(feature.get('New Energy 2016'));
  var gene18 = ee.Number(feature.get('New Energy 2018'));
  return feature.set({'% generation change from 16 to 18':gene18.subtract(gene16).divide(gene16).multiply(100)});
};
var genestate = datastate.map(genechange);

// select and map top generation source change states
var topgene = genestate.sort('% generation change from 16 to 18', false).limit(10);
var topgenelist = topgene.aggregate_array('NAME');
print('Top Generations Change:', topgenelist);
var empty = ee.Image().byte();
var maptopgene = empty.paint({featureCollection: topgene, color: '% generation change from 16 to 18'});
Map.addLayer(maptopgene, {palette:palette1, max:100, opacity:0.7} , 'Top Generation Change from 16 to 18');

// intersect to get two states: Nebraska & Massachusetts
var topchangeft = ee.Feature(topchange.union().first());
var topgeneft = ee.Feature(topgene.union().first());
var intersect = topgeneft.intersection(topchangeft);
Map.addLayer(intersect, {color:'7100FF', opacity:0.7} , 'Progressed New Energy Usage');
```

Even though electric cars don't need fossil fuels to function, they need electricity which, in most places, is produced using fossil fuels. In order to find both rapid and sustainable electric vehicles development from 2016 to 2018, we define our criteria as:

- states with the fastest EV sales increase
- states with the fastest change that electricity is generated from renewable energy

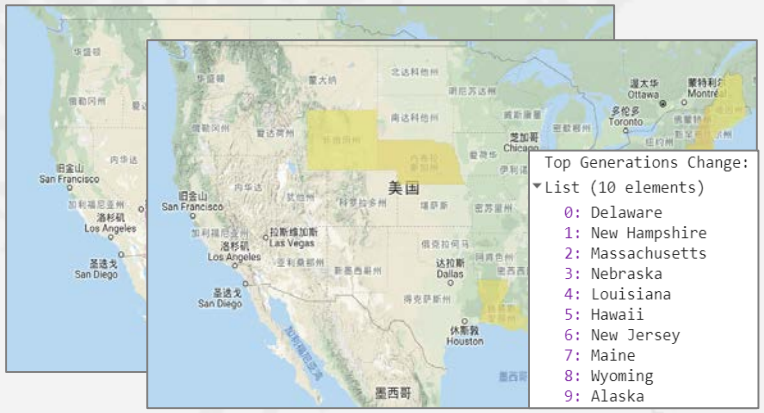
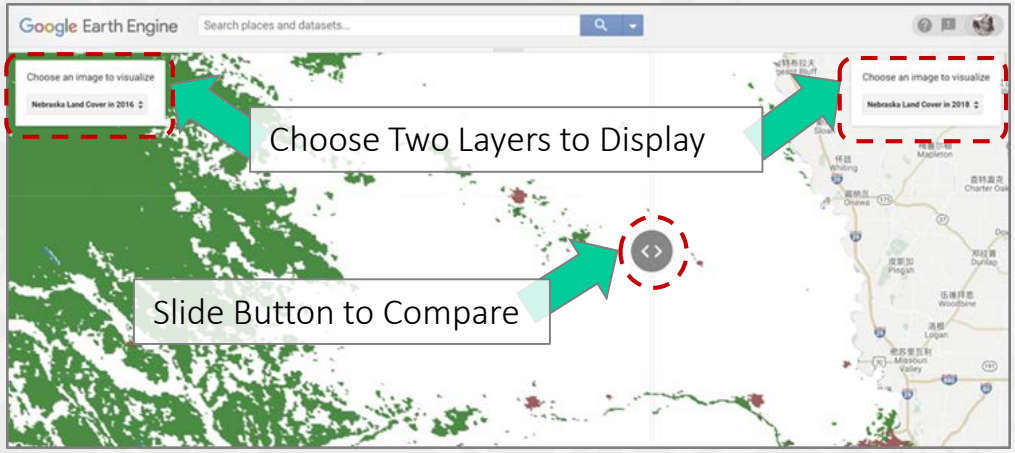


Fig 3 The Intersection of Two Criteria: NE and MA

Code & Result - GEE

Part TWO – Land Change Visualization: A Split Panel



In this part, we focus on two states where both have the most rapid growth in alternative energy usages: **Nebraska** and **Massachusetts**. A split panel is built to better visualize green land change from 2016 to 2018. However, the study area is too large to find more detailed results.

```
// import all data, pre-process, and mapset
var land18 = ee.Image("MODIS/006/MCD12Q1/2018_01_01");
var land16 = ee.Image("MODIS/006/MCD12Q1/2016_01_01");
var state = ee.FeatureCollection("TIGER/2016/States");

var nebraska = state.filterMetadata('NAME','contains','Nebraska');
var neland16 = land16.select('LC_Type1').clip(nebraska);
var neland18 = land18.select('LC_Type1').clip(nebraska);
var massachusetts = state.filterMetadata('NAME','contains','Massachusetts');
var maland16 = land16.select('LC_Type1').clip(massachusetts);
var maland18 = land18.select('LC_Type1').clip(massachusetts);

var palette = ['26BB14', // 01 = Evergreen Needleleaf Forest
               '26BB14', // 02 = Evergreen Broadleaf Forest
               '26BB14', // 03 = Deciduous Needleleaf Forest
               '26BB14', // 04 = Deciduous Broadleaf Forest
               '26BB14', // 05 = Mixed Deciduous Forest
               '26BB14', // 06 = Closed Shrubland
               '26BB14', // 07 = Open Shrubland
               '26BB14', // 08 = Woody Savanna
               '26BB14', // 09 = Savanna
               '26BB14', // 10 = Grassland
               '26BB14', // 11 = Permanent Wetland
               'FFFFFF', // 12 = Cropland
               'CC3333', // 13 = Urban
               'FFFFFF', // 14 = Crops & Natural Vegetation
               'FFFFFF', // 15 = Permanent Snow & Ice 'DDCCAA',
               'FFFFFF', // 16 = Barren / Desert
               '06AFE8'  // 17 = Water
              ].join(',');
var mapset = { min:1, max:17, opacity:0.7, palette:palette};
```

```
// choose the images
var images = {
  "Nebraska Land Cover in 2016": neland16.visualize(mapset),
  "Nebraska Land Cover in 2018": neland18.visualize(mapset),
  "Massachusetts Land Cover in 2016": maland16.visualize(mapset),
  "Massachusetts Land Cover in 2018": maland18.visualize(mapset),
};

// create the left and right map, and have them display layer 0 and 1
var leftMap = ui.Map();
leftMap.setControlVisibility(false);
var leftSelector = addLayerSelector(leftMap, 0, 'top-left');
var rightMap = ui.Map();
rightMap.setControlVisibility(false);
var rightSelector = addLayerSelector(rightMap, 1, 'top-right');

// add a layer selection widget to allow users to change
function addLayerSelector(mapToChange, defaultValue, position) {
  var label = ui.Label("Choose an image to visualize");
  function updateMap(selection){mapToChange.layers().set(0, ui.Map.Layer(images[selection]))}
  var select = ui.Select({items: Object.keys(images), onChange: updateMap});
  select.setValue(Object.keys(images)[defaultValue], true);
  var controlPanel = ui.Panel({widgets: [label, select], style: {position: position}});
  mapToChange.add(controlPanel);
}
// create a split panel to hold the adjacent, linked maps
var splitPanel = ui.SplitPanel({firstPanel: leftMap, secondPanel: rightMap, wipe: true, style:
{stretch: "both"}});

// set the split panel as the only thing in the UI root
ui.root.widgets().reset([splitPanel]);
var linker = ui.Map.Linker([leftMap, rightMap]);
```

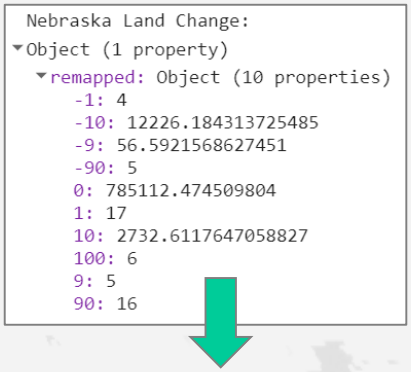
Code & Result - GEE

Part THREE – Green Change Calculation

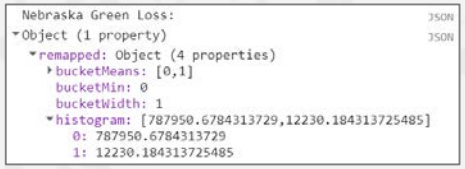
In this part, we aim to get more specific green land change:

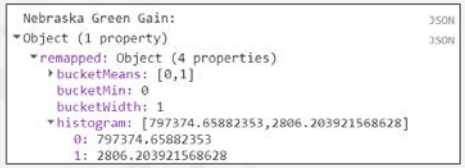
```
var newneland16 = neland16.remap([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
[1,1,1,1,1,10,10,10,10,10,10,0,0,0,0,0,100], 0, 'LC_Type1');
var newneland18 = neland18.remap([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
[1,1,1,1,1,10,10,10,10,10,10,0,0,0,0,0,100], 0, 'LC_Type1');
var nelandchange = newneland18.subtract(newneland16);
var countnelandchange = nelandchange.reduceRegion({reducer: ee.Reducer.frequencyHistogram(),
geometry: nebraska, scale: 500});
print('Nebraska Land Change:',countnelandchange);

// then calculate green loss and gain
var negreenloss = nelandchange.eq(-10).or(nelandchange.eq(9).or(nelandchange.eq(-1)));
Map.addLayer(negreenloss,{palette:'FFFFFF,DC1010',opacity:0.7}, 'Nebraska Green Loss');
var countneloss = negreenloss.reduceRegion({reducer: ee.Reducer.histogram(), geometry: neb
scale: 500});
print('Nebraska Green Loss:',countneloss);
var negreengain = nelandchange.eq(-9).or(nelandchange.eq(1).or(nelandchange.eq(10)));
var negain_remap = negreengain.remap( [1], [1], null, 'remapped');
Map.addLayer(negain_remap,{palette:'ffffff,336633',opacity:0.7}, 'Nebraska Green Gain');
var countnegain = negreengain.reduceRegion({reducer: ee.Reducer.histogram()
scale: 500});
print('Nebraska Green Gain:',countnegain);
```



Value	Change	Gain/Loss	# Pixel
-1	Forest to Others	Loss	4
-10	Grass to Others	Loss	12226
-9	Grass to Fprest	Gain	56
-90	Water to Forest	-	5
-99	Water to Forest	Gain	0
0	Unchanged	-	785112
1	Others to Forest	Gain	17
10	Others to Grass	Gain	2732
100	Others to Water	-	6
9	Forest to Grass	Loss	5
90	Grass to Water	-	16
99	Forest to Water	-	0





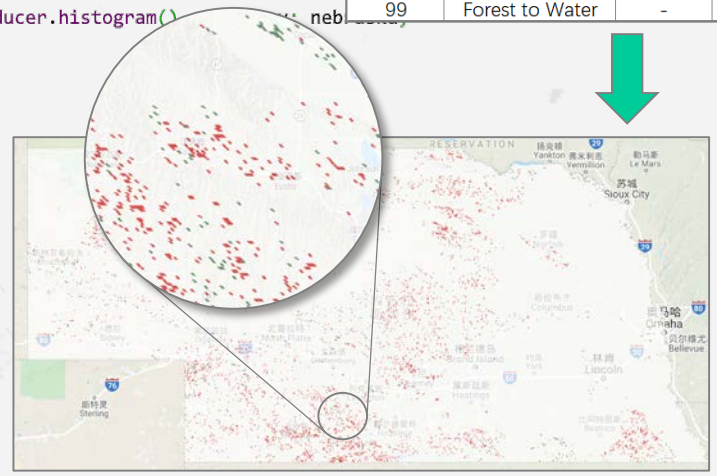


Fig 4 Green Land Gain and Loss in Nebraska

Code & Result - GEE

Part THREE – Green Change Calculation

To find more detailed results about green land change in Nebraska (NE) and Massachusetts (MA), we reclassify 17 types of land use to four categories:

- Forest: Evergreen Needleleaf Forest, Evergreen Broadleaf Forest, Deciduous Needleleaf Forest, Deciduous Broadleaf Forest, Mixed Deciduous Forest
- Grass: Closed Shrubland, Open Shrubland, Woody Savanna, Savanna, Grassland, Permanent Wetland
- Other: Cropland, Urban, Crops & Natural Vegetation, Permanent Snow & Ice, Barren / Desert
- Water: Water

We calculate two states' area of green gain and loss as well:
in NE, loss >> gain; in MA, loss < gain.

```
Massachusetts Land Change:
▾Object (1 property)
  ▾remapped: Object (10 properties)
      -10: 98.7764705882353
      -9: 1316.662745098039
      -90: 4
      -99: 3
      0: 106644.2941176466
      10: 34
      100: 6
      9: 1056.1294117647058
      90: 2
      99: 3
```

Value	Change	Gain/Loss	# Pixel
-1	Forest to Others	Loss	0
-10	Grass to Others	Loss	99
-9	Grass to Fprest	Gain	1317
-90	Water to Forest	-	4
-99	Water to Forest	Gain	3
0	Unchanged	-	106644
1	Others to Forest	Gain	0
10	Others to Grass	Gain	34
100	Others to Water	-	6
9	Forest to Grass	Loss	1056
90	Grass to Water	-	2
99	Forest to Water	-	3

```
Massachusetts Green Loss:
▾Object (1 property)
  ▾remapped: Object (4 properties)
    ▸bucketMeans: [0,1]
      bucketMin: 0
      bucketWidth: 1
    ▾histogram: [108012.95686274461,1154.9058823529408]
        0: 108012.95686274461
        1: 1154.9058823529408
```

```
Massachusetts Green Gain:
▾Object (1 property)
  ▾remapped: Object (4 properties)
    ▸bucketMeans: [0,1]
      bucketMin: 0
      bucketWidth: 1
    ▾histogram: [107817.19999999956,1350.662745098039]
        0: 107817.19999999956
        1: 1350.662745098039
```

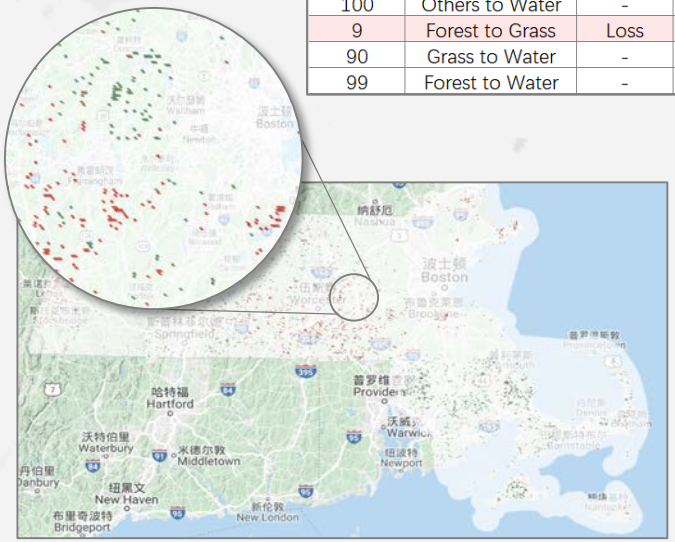


Fig 5 Green Land Gain and Loss in Massachusetts

Code & Result - ArcPy

Part FOUR – Air Quality Difference

The second keyword of environmental impacts is air quality change, and we compiled our data parameters from EPA Ambient Air Quality Monitoring Program to four related parameters: 1)1-hour NO2, 2) 8-hour Ozone, 3) annual PM2.5, 4) annual SO2.

The following code is to transform .csv into .lyr by longitude (X) and latitude (Y).

```
# Import necessary modules
import sys, os, string, math, arcpy, traceback
arcpy.env.overwriteOutput = True

try:
    # read and write the name of the output shapefile
    nameOfInputTable = arcpy.GetParameterAsText(0)
    nameOfXField = arcpy.GetParameterAsText(1)
    nameOfYField = arcpy.GetParameterAsText(2)
    nameOfOutputShapefile = arcpy.GetParameterAsText(3)

    # set the local variables
    in_Table = nameOfInputTable
    x_coords = nameOfXField
    y_coords = nameOfYField
    out_Layer = nameOfOutputShapefile
    saved_Layer = r"C:/Users/USER/Desktop/upenn/LARP743/My Final Project/ArcPy part/output.lyr"

    # make the new layer
    arcpy.MakeXYEventLayer_management(in_Table, x_coords, y_coords, out_

    # print the total rows and save to a layer file
    print(arcpy.GetCount_management(out_Layer))
    arcpy.SaveToLayerFile_management(out_Layer, saved_Layer)

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")
```

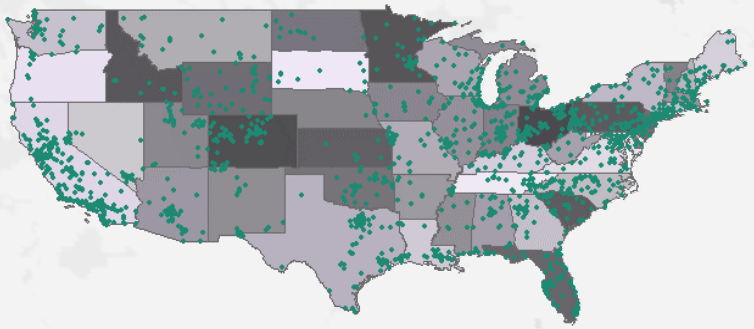
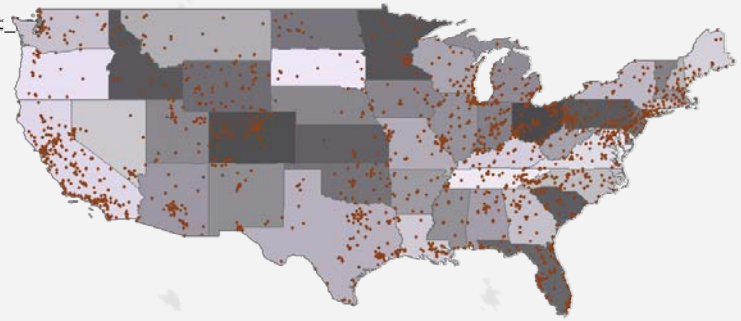


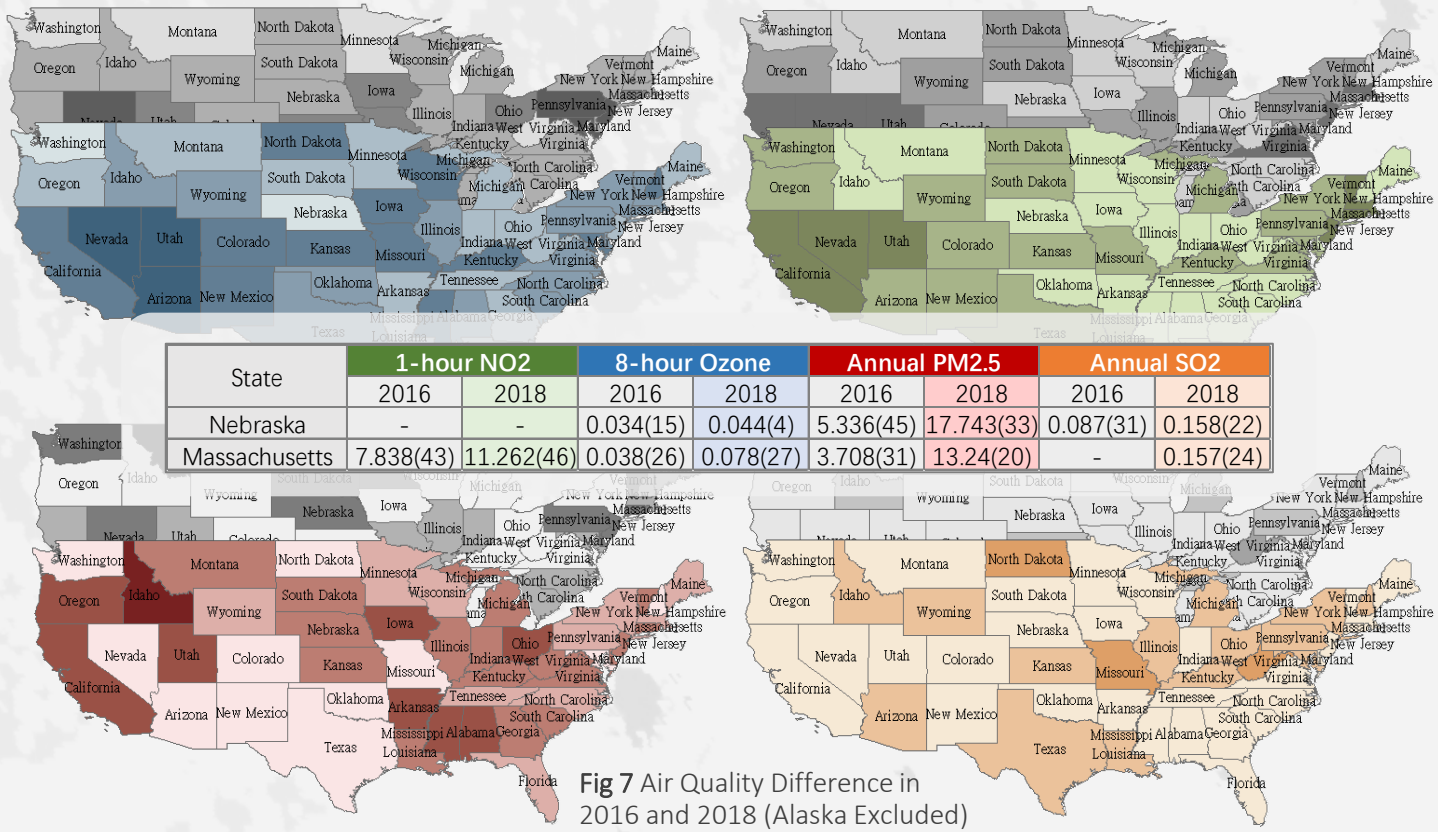
Fig 6 Location of Monitors in 2016 and 2018 (Alaska Excluded)



Code & Result - ArcPy

Part FOUR – Air Quality Difference

The next step is to calculate mean value for four parameters by state. Note that since outliers of data have been removed, null value equals to zero in this function. It can be concluded that the air quality in Nebraska (NE) and Massachusetts (MA) has not improved by development of electric vehicles. However, their air quality rankings (in parentheses) have progressed in terms of PM2.5, Ozone and SO2 parameters.



State	1-hour NO2		8-hour Ozone		Annual PM2.5		Annual SO2	
	2016	2018	2016	2018	2016	2018	2016	2018
Nebraska	-	-	0.034(15)	0.044(4)	5.336(45)	17.743(33)	0.087(31)	0.158(22)
Massachusetts	7.838(43)	11.262(46)	0.038(26)	0.078(27)	3.708(31)	13.24(20)	-	0.157(24)

Fig 7 Air Quality Difference in 2016 and 2018 (Alaska Excluded)

Conclusion

- The real development of electric vehicles (EV) is not only reflected on the sales increase, but also need to take energy structure of electricity into consideration. It can be calculated that Nebraska (NE) and Massachusetts (MA) states have the healthiest structure of EV usage, which will effectively alleviate the energy crisis and promote social and economic development.
- However, the intensity of environmental impacts varies on the specific technology used, the geographic location and other factors. In Nebraska (NE), green land had shrunk 2356 km^2 from 2016 to 2018; while in MA, green gain and loss were flat in that period. This phenomenon may result from their different roles in economy and geographical conditions: NE has the third-highest number of industrial electricity customers of any state, and a significant share of Nebraska's industrial consumption is seasonal demand from farms where electricity is used to run irrigation systems.
- The absolute air quality measurements in NE and MA did not improve during the great progress in EV. Whereas, their comparative rankings in air quality has changed to a great extent, suggesting a potentially positive environmental impact of EV usage. More scientific proof should be given to draw a solid correlation.
- To put it into a nutshell, wide and sustainable usages of EV will help to tackle climate change and meet global goals in the long haul.

Appendix – Full Code

Google Earth Engine - Script ONE

```
// import all data needed
var state = ee.FeatureCollection("TIGER/2016/States");
var rawdata = ee.FeatureCollection("users/annisann666/data16to18");
var land18 = ee.Image("MODIS/006/MCD12Q1/2018_01_01");
var land16 = ee.Image("MODIS/006/MCD12Q1/2016_01_01");

// setup and some palettes
Map.centerObject(state, 4);
var palette1 = ['ffff55','ff0000'];
var palette2 = ['FEA8F2','FF2B2B'];

// join raw data with state features
var joinfilter = ee.Filter.equals('NAME', null, 'State', null );
var thejoin = ee.Join.inner();
var join = thejoin.apply(state, rawdata, joinfilter);
var datastate = join.map(function(featurepair) {
  var feature1 = ee.Feature(featurepair.get('primary'));
  var feature2 = ee.Feature(featurepair.get('secondary'));
  return feature1.set(feature2.toDictionary());
});

// show joined results
print('Original Data:', state);
print('Joined Data', datastate);

// add sales change from 2016 to 2018
var salechange = function(feature){
  var sale16 = ee.Number(feature.get('EV Sales 2016'));
  var sale18 = ee.Number(feature.get('EV Sales 2018'));
  return feature.set({'sale change from 16 to 18':sale18.subtract(sale16).divide(sale16).multiply(100).log()});};
var salestate = datastate.map(salechange);

// select and map top sale change states
var topchange = salestate.sort('sale change from 16 to 18', false).limit(10);
var topchangelist = topchange.aggregate_array('NAME');
print("Top EV Sales Change:", topchangelist);
var empty = ee.Image().byte();
var maptopsale = empty.paint({featureCollection: topchange, color: 'sale change from 16 to 18'});
Map.addLayer(maptopsale, {palette:palette2, min:5, max:7, opacity:0.7}, 'Top EV Sales change from 16 to 18');

// add generation change from 2016 to 2018
var genechange = function(feature){
  var gene16 = ee.Number(feature.get('New Energy 2016'));
  var gene18 = ee.Number(feature.get('New Energy 2018'));
  return feature.set({'% generation change from 16 to 18':gene18.subtract(gene16).divide(gene16).multiply(100)});
};
var genestate = datastate.map(genechange);

// select and map top generation source change states
var topgene = genestate.sort('% generation change from 16 to 18', false).limit(10);
var topgenelist = topgene.aggregate_array('NAME');
print("Top Generations Changes", topgenelist);
var empty = ee.Image().byte();
var maptopgene = empty.paint({featureCollection: topgene, color: '% generation change from 16 to 18'});
Map.addLayer(maptopgene, {palette:palette1, max:100, opacity:0.7} , 'Top Generation Change from 16 to 18');

// intersect to get two states: Nebraska & Massachusetts
var topchangeft = ee.Feature(topchange.union().first());
var topgeneft = ee.Feature(topgene.union().first());
var intersect = topgeneft.intersection(topchangeft);
Map.addLayer(intersect, {color:'7100FF', opacity:0.7} , 'Progressed New Energy Usage');

// detect Nebraska land cover change
// first show the land cover from 16 to 18
var nebraska = datastate.filterMetadata('NAME','contains','Nebraska');
var neland16 = land16.select('LC_Type1').clip(nebraska);
var neland18 = land18.select('LC_Type1').clip(nebraska);
var palette3 = ['268814', // 01 = Evergreen Needleleaf Forest
                '268814', // 02 = Evergreen Broadleaf Forest
                '268814', // 03 = Deciduous Needleleaf Forest
                '268814', // 04 = Deciduous Broadleaf Forest
                '268814', // 05 = Mixed Deciduous Forest
                '7FD687', // 06 = Closed Shrubland
                '7FD687', // 07 = Open Shrubland
                '7FD687', // 08 = Woody Savanna
                '7FD687', // 09 = Savanna
                '7FD687', // 10 = Grassland
                '00C7B2', // 11 = Permanent Wetland
                'FFFFFF', // 12 = Cropland
                'CC3333', // 13 = Urban
                'FFFFFF', // 14 = Crops & Natural Vegetation
                'FFFFFF', // 15 = Permanent Snow & Ice 'DDCCAA',
                'FFFFFF', // 16 = Barren / Desert
                '0CAFE8'  // 17 = Water
                ].join(",");
var napset = { min:1, max:17, opacity:0.7, palette:palette3};
Map.addLayer(neland16, napset, 'Nebraska Land Cover in 2016');
Map.addLayer(neland18, napset, 'Nebraska Land Cover in 2018');


var newneland16 = neland16.remap([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
[1,1,1,1,1,10,10,10,10,10,10,0,0,0,0,0,100], 0, 'LC_Type1');
var newneland18 = neland18.remap([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
[1,1,1,1,1,10,10,10,10,10,10,0,0,0,0,0,100], 0, 'LC_Type1');
var nelandchange = newneland18.subtract(newneland16);
var countnelandchange = nelandchange.reduceRegion({reducer: ee.Reducer.frequencyHistogram(),
geometry: nebraska, scale: 500});
print("Nebraska Land Change:",countnelandchange);

// then calculate green loss and gain
var negreenloss = nelandchange.eq(-10).or(nelandchange.eq(9).or(nelandchange.eq(-1)));
Map.addLayer(negreenloss,{palette:'FFFFFF,DC1010',opacity:0.7}, 'Nebraska Green Loss');
var countneloss = negreenloss.reduceRegion({reducer: ee.Reducer.histogram(), geometry: nebraska,
scale: 500});
print("Nebraska Green Loss:",countneloss);
var negreengain = nelandchange.eq(-9).or(nelandchange.eq(1).or(nelandchange.eq(10)));
var negain_remap = negreengain.remap( [1], [1], null, 'remapped');
Map.addLayer(negain_remap,{palette:'ffffff,336633',opacity:0.7}, 'Nebraska Green Gain');
var countnegain = negreengain.reduceRegion({reducer: ee.Reducer.histogram(), geometry: nebraska,
scale: 500});
print("Nebraska Green Gain:",countnegain);
```

Appendix – Full Code

```
// detect Massachusetts land cover change
// first show the land cover from 16 to 18
var massachusetts = datastate.filterMetadata('NAME','contains','Massachusetts');
var maland16 = land16.select('LC_Type1').clip(massachusetts);
var maland18 = land18.select('LC_Type1').clip(massachusetts);
Map.addLayer(maland16, mapset, 'Massachusetts Land Cover in 2016');
Map.addLayer(maland18, mapset, 'Massachusetts Land Cover in 2018');
var newmaland16 = maland16.remap([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
[1,1,1,1,1,10,10,10,10,10,0,0,0,0,0,100], 0, 'LC_Type1');
var newmaland18 = maland18.remap([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
[1,1,1,1,1,10,10,10,10,10,0,0,0,0,0,100], 0, 'LC_Type1');
var malandchange = newmaland18.subtract(newmaland16);
var countmalandchange = malandchange.reduceRegion({reducer: ee.Reducer.frequencyHistogram(),
geometry: massachusetts, scale: 500});
print('Massachusetts Land Change:',countmalandchange);
```

```
// then calculate green loss and gain
var magreenloss = malandchange.eq(-10).or(malandchange.eq(9)).or(malandchange.eq(-1));
Map.addLayer(magreenloss,{palette:'FFFFFF,DC1010',opacity:0.7}, 'Massachusetts Green Loss');
var countmaloss = magreenloss.reduceRegion({reducer: ee.Reducer.histogram(), geometry: massachusetts,
scale: 500});
print('Massachusetts Green Loss:',countmaloss);
var magreengain = malandchange.eq(-9).or(malandchange.eq(1)).or(malandchange.eq(10));
var magain_remap = magreengain.remap( [1], [1], null, 'remapped');
Map.addLayer(magain_remap,{palette:'ffffff,136611',opacity:0.7}, 'Massachusetts Green Gain');
var countmagain = magreengain.reduceRegion({reducer: ee.Reducer.histogram(), geometry: massachusetts,
scale: 500});
print('Massachusetts Green Gain:',countmagain);

// detect air pollution change *see ArcPy part*
```

Google Earth Engine - Script TWO

```
// another method: compare 2016/2018 images with a split panel
// import all data, pre-process, and mapset
var land18 = ee.Image("MODIS/006/MCD12Q1/2018_01_01");
var land16 = ee.Image("MODIS/006/MCD12Q1/2016_01_01");
var state = ee.FeatureCollection("TIGER/2016/States");

var nebraska = state.filterMetadata('NAME','contains','Nebraska');
var neland16 = land16.select('LC_Type1').clip(nebraska);
var neland18 = land18.select('LC_Type1').clip(nebraska);
var massachusetts = state.filterMetadata('NAME','contains','Massachusetts');
var maland16 = land16.select('LC_Type1').clip(massachusetts);
var maland18 = land18.select('LC_Type1').clip(massachusetts);

var palette = ['26BB14', // 01 = Evergreen Needleleaf Forest
               '26BB14', // 02 = Evergreen Broadleaf Forest
               '26BB14', // 03 = Deciduous Needleleaf Forest
               '26BB14', // 04 = Deciduous Broadleaf Forest
               '26BB14', // 05 = Mixed Deciduous Forest
               '26BB14', // 06 = Closed Shrubland
               '26BB14', // 07 = Open Shrubland
               '26BB14', // 08 = Woody Savanna
               '26BB14', // 09 = Savanna
               '26BB14', // 10 = Grassland
               '26BB14', // 11 = Permanent Wetland
               'FFFFFF', // 12 = Cropland
               'CC3333', // 13 = Urban
               'FFFFFF', // 14 = Crops & Natural Vegetation
               'FFFFFF', // 15 = Permanent Snow & Ice 'DDCCAA',
               'FFFFFF', // 16 = Barren / Desert
               '06AFE8'  // 17 = Water
              ].join(',');
var mapset = { min:1, max:17, opacity:0.7, palette:palette};
```

```
// choose the images
var images = {
  'Nebraska Land Cover in 2016': neland16.visualize(mapset),
  'Nebraska Land Cover in 2018': neland18.visualize(mapset),
  'Massachusetts Land Cover in 2016': maland16.visualize(mapset),
  'Massachusetts Land Cover in 2018': maland18.visualize(mapset),
};

// create the left and right map, and have them display layer 0 and 1
var leftMap = ui.Map();
leftMap.setControlVisibility(false);
var leftSelector = addLayerSelector(leftMap, 0, 'top-left');
var rightMap = ui.Map();
rightMap.setControlVisibility(false);
var rightSelector = addLayerSelector(rightMap, 1, 'top-right');

// add a layer selection widget to allow users to change
function addLayerSelector(mapToChange, defaultValue, position) {
  var label = ui.Label('Choose an image to visualize');
  function updateMap(selection){mapToChange.layers().set(0, ui.Map.Layer(images[selection]))}
  var select = ui.Select({items: Object.keys(images), onChange: updateMap});
  select.setValue(Object.keys(images)[defaultValue], true);
  var controlPanel = ui.Panel({widgets: [label, select], style: {position: position}});
  mapToChange.add(controlPanel);
}

// create a split panel to hold the adjacent, linked maps
var splitPanel = ui.SplitPanel({firstPanel: leftMap, secondPanel: rightMap, wipe: true, style:
{stretch: 'both'}});

// set the split panel as the only thing in the UI root
ui.root.widgets().reset([splitPanel]);
var linker = ui.Map.Linker([leftMap, rightMap]);
```

Appendix – Full Code

ArcPy- Script THREE

```python
# Import necessary modules
import sys, os, string, math, arcpy, traceback
arcpy.env.overwriteOutput = True

try:
    # read and write the name of the output shapefile
    nameOfInputTable = arcpy.GetParameterAsText(0)
    nameOfXField = arcpy.GetParameterAsText(1)
    nameOfYField = arcpy.GetParameterAsText(2)
    nameOfOutputShapefile = arcpy.GetParameterAsText(3)

    # set the local variables
    in_Table = nameOfInputTable
    x_coords = nameOfXField
    y_coords = nameOfYField
    out_Layer = nameOfOutputShapefile
    saved_Layer = r"C:/Users/USER/Desktop/upenn/LARP743/My Final Project/ArcPy part/output.lyr"

    # make the new layer
    arcpy.MakeXYEventLayer_management(in_Table, x_coords, y_coords, out_Layer)

    # print the total rows and save to a layer file
    print(arcpy.GetCount_management(out_Layer))
    arcpy.SaveToLayerFile_management(out_Layer, saved_Layer)

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- Script FOUR

```python
import sys, os, string, math, arcpy, traceback
arcpy.env.overwriteOutput = True

try:
    nameOfStateInput = arcpy.GetParameterAsText(0)
    nameOfMonitorInput = arcpy.GetParameterAsText(1)
    nameOfFieldInput = arcpy.GetParameterAsText(2)
    nameOfMeanOutput = arcpy.GetParameterAsText(3)

    # Want to join field value to states and calculate the mean value for each state
    targetFeatures = nameOfStateInput
    joinFeatures = nameOfMonitorInput
    fieldName = nameOfFieldInput

    # Create a new fieldmappings and add the two input feature classes.
    fieldmappings = arcpy.FieldMappings()
    fieldmappings.addTable(targetFeatures)
    fieldmappings.addTable(joinFeatures)

    # First get the Monitor fieldmap.
    # The output will have the states with the attributes of the specific field.
    # Setting the field's merge rule to mean will aggregate the values
    # for all of the prices for each state into an average value.
    # The field is also renamed to be more appropriate for the output.
    fieldIndex = fieldmappings.findFieldMapIndex(fieldName)
    fieldmap = fieldmappings.getFieldMap(fieldIndex)
    field = fieldmap.outputField

    # Rename the field and pass the updated field object back into the field map
    newField = "mean_" + fieldName[:3]
    field.name = newField
    field.aliasName = newField
    fieldmap.outputField = field

    # Set the merge rule to mean and then replace the old fieldmap in the mappings object
    # with the updated one
    fieldmap.mergeRule = "mean"
    fieldmappings.replaceFieldMap(fieldIndex, fieldmap)

    # spatial join, add new field
    newName = "new" + fieldName[:3]
    arcpy.SpatialJoin_analysis(targetFeatures, joinFeatures, nameOfMeanOutput, "#","#", fieldmappings)
    arcpy.AddField_management(nameOfMeanOutput, newName, "DOUBLE", 10, 4)
    enumerationOfRecords = arcpy.UpdateCursor(nameOfMeanOutput)

    # Loop through that enumeration, set mean sale value for each record
    for nextRecord in enumerationOfRecords:
        meanValue = nextRecord.getValue(newField)
        nextRecord.setValue(newName, meanValue)
        enumerationOfRecords.updateRow(nextRecord)

    # Delete row and update cursor objects
    del nextRecord
    del enumerationOfRecords

except Exception as e:
    arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
    exceptionreport = sys.exc_info()[2]
    fullermessage   = traceback.format_tb(exceptionreport)[0]
    arcpy.AddError("at this location: \n\n" + fullermessage + "\n")
```