



LARP 743 Final Project

Electric Vehicle & Environmental Change

Yian Wang

Instructor: Dana Tomlin

Introduction

Introduction

Workflow

Code & Result

Conclusion

Appendix

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.

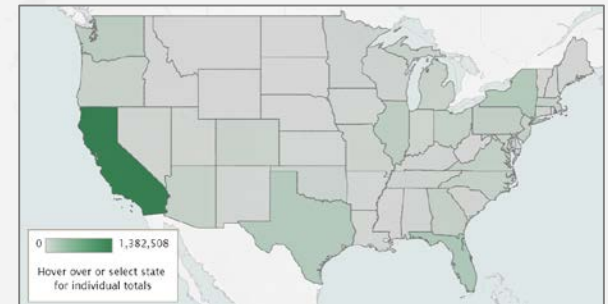


Fig 1 Sales of Electric Vehicle by State



Workflow

Introduction

Workflow

Code & Result

Conclusion

Appendix

State Selection:

- Join Tables
- Calculate sales change
- Calculate generation
- Sort and intersect

Air Quality Difference:

- Data Transform
- Spatial Join Analysis
- Extract and Compare
- Further Analysis

Land Change Visualization:

- A Split Panel

Green Change Calculation:

- Remap and subtract
- Frequency Histogram
- Filter and Select
- Reducer Histogram
- 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

Introduction

Workflow

Code & Result

Conclusion

Appendix

```
1. // import all data needed
2. var state = ee.FeatureCollection("TIGER/2016/States");
3. var rawdata = ee.FeatureCollection("users/annisann666/data16to18");
4. var land18 = ee.Image("MODIS/006/MCD12Q1/2018_01_01");
5. var land16 = ee.Image("MODIS/006/MCD12Q1/2016_01_01");
6.
7. // setup and some palettes
8. Map.centerObject(state, 4);
9. var palette1 = ['ffff55', 'ff0000'];
10. var palette2 = ['A8DFFE', '3F2BFF'];
11.
12. // join raw data with state features
13. var joinfilter = ee.Filter.equals('NAME', null, 'State', null );
14. var thejoin = ee.Join.inner();
15. var join = thejoin.apply(state, rawdata, joinfilter);
16. var datastate = join.map(function(featurepair) {
17.   var feature1 = ee.Feature(featurepair.get('primary'));
18.   var feature2 = ee.Feature(featurepair.get('secondary'));
19.   return feature1.set(feature2.toDictionary());
20. });
21.
22. // add sales change from 2016 to 2018
23. var salechange = function(feature){
24.   var sale16 = ee.Number(feature.get('EV Sales 2016'));
25.   var sale18 = ee.Number(feature.get('EV Sales 2018'));
26.   return feature.set({'sale change from 16 to
27.   18':sale18.subtract(sale16).divide(sale16).multiply(100).log()});
28. }
29. var salestate = datastate.map(salechange);
30.
31. // select and map top sale change states
32. var topchange = salestate.sort('sale change from 16 to 18', false).limit(10);
33. var topchangelist = topchange.aggregate_array('NAME');
34. print('Top EV Sales Change:', topchangelist);
35. var empty = ee.Image().byte();
36. var maptopsale = empty.paint({featureCollection: topchange, color: 'sale change from 16 to 18'});
37. Map.addLayer(maptopsale, {palette:palette2, min:5, max:7, opacity:0.7}, 'Top EV Sales change from 16 to 18');
```

Original Data:
*FeatureCollection TIGER/2016/States (56 elements, 15 columns)
 type: FeatureCollection
 id: TIGER/2016/States
 version: 1566852223056815
 *columns: Object (15 properties)
 *features: List (56 elements)
 *0: Feature 00000000000000000000000015 (MultiPolygon, 14 properties)
 type: Feature
 id: 00000000000000000000000015
 *geometry: MultiPolygon, 1564 vertices
 *properties: Object (14 properties)
 ALAND: 8868100450
 AWATER: 4923178155
 DIVISION: 0
 FUNCSTAT: A
 GEOTID: 72
 INTPTLAT: +18.2176480
 INTPTLON: -066.4107992
 LSAD: 00
 MTFCC: G4000
 NAME: Puerto Rico
 REGION: 9
 STATEFP: 72
 STATENS: 01779808
 STUSPS: PR

Joined Data
*FeatureCollection TIGER/2016/States (51 elements, 0 columns)
 type: FeatureCollection
 id: TIGER/2016/States
 version: 1566852223056815
 *columns: Object (0 properties)
 *features: List (51 elements)
 *0: Feature 00000000000000000000000012_000000000000000000000017 (Polygon, 20 properties)
 type: Feature
 id: 000000000000000000000012_000000000000000000000017
 *geometry: Polygon, 3108 vertices
 *properties: Object (20 properties)
 ALAND: 12542538347
 AWATER: 1885476291
 Abbreviation: CT
 DIVISION: 1
 EV Sales 2016: 1511
 EV Sales 2018: 3415
 FUNCSTAT: A
 GEOTID: 09
 INTPTLAT: +41.5797777
 INTPTLON: -072.7466665
 LSAD: 00
 MTFCC: G4000
 NAME: Connecticut
 New Energy 2016: 38781421
 New Energy 2018: 56231808
 REGION: 1
 STATEFP: 09
 STATENS: 01779780
 STUSPS: CT
 State: Connecticut

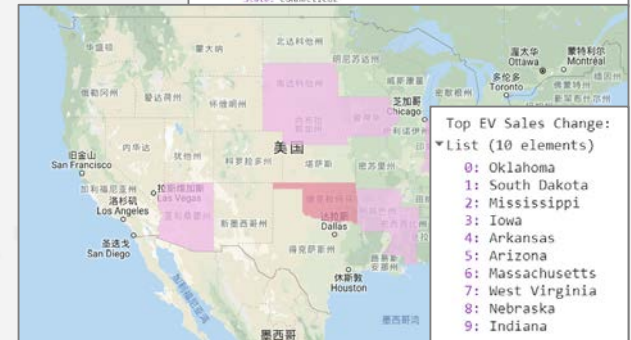


Fig 2
Top 10 Electric Vehicle Sales Change by State

Code & Result - GEE

Part ONE – State Selection by Sales Change and Energy Source

Introduction

Workflow

Code & Result

Conclusion

Appendix

```
37. // add generation change from 2016 to 2018
38. var genechange = function(feature){
39.   var gene16 = ee.Number(feature.get('New Energy 2016'));
40.   var gene18 = ee.Number(feature.get('New Energy 2018'));
41.   return feature.set({'% generation change from 16 to
18':gene18.subtract(gene16).divide(gene16).multiply(100)});
42. };
43. var genestate = datastate.map(genechange);
44.
45. // select and map top generation source change states
46. var topgene = genestate.sort('% generation change from 16 to 18', false).limit(10);
47. var topgenelist = topgene.aggregate_array('NAME');
48. print('Top Generations Change:', topgenelist);
49. var empty = ee.Image().byte();
50. var maptopgene = empty.paint({featureCollection: topgene, color: '% generation change from 16 to
18'});
51. Map.addLayer(maptopgene, {palette:palette1, max:100, opacity:0.7}, 'Top Generation Change from 16 to
18');

53. // intersect to get two states: Nebraska & Massachusetts
54. var topchangeft = ee.Feature(topchange.union().first());
55. var topgeneft = ee.Feature(topgene.union().first());
56. var intersect = topgeneft.intersection(topchangeft);
57. 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

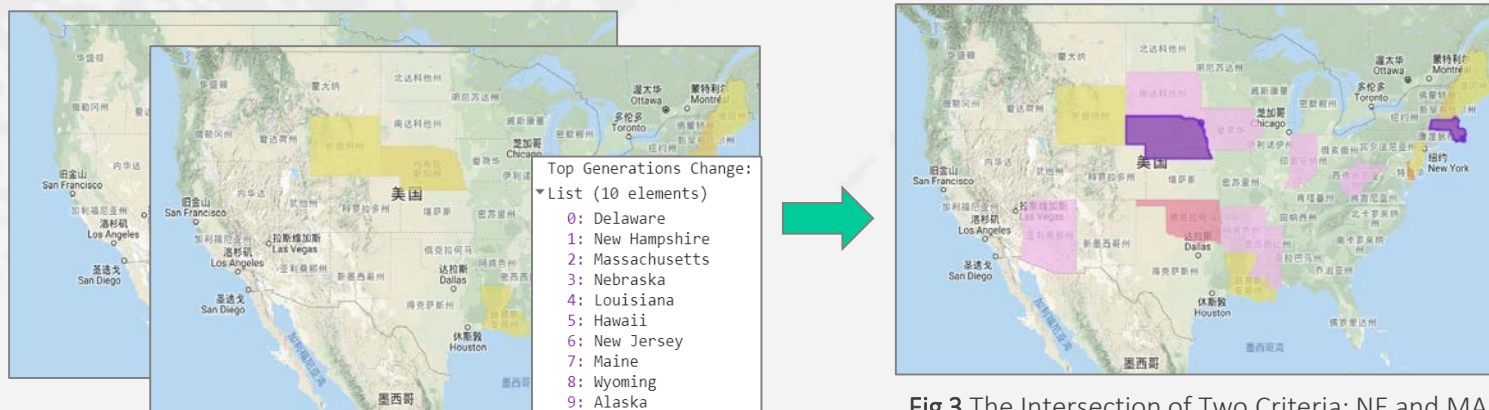
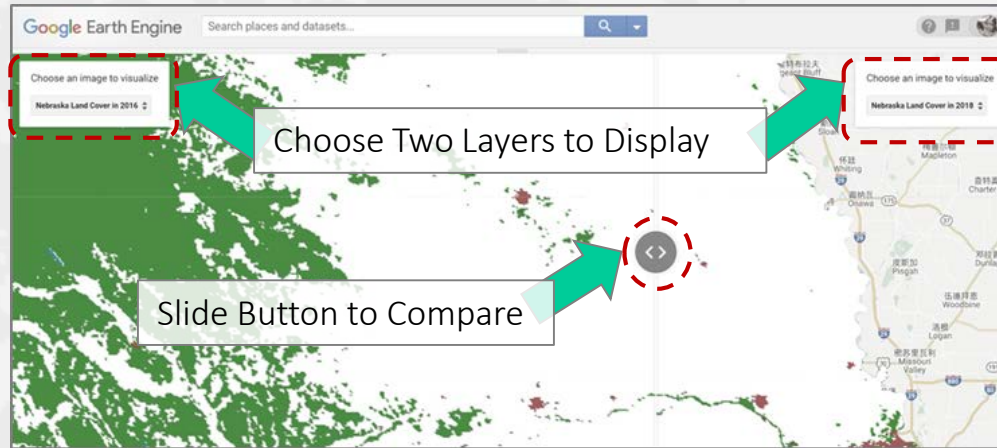


Fig 3 The Intersection of Two Criteria: NE and MA

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.

Introduction

Workflow

Code & Result

Conclusion

Appendix

```
2. // import all data, pre-process, and mapset
3. var land18 = ee.Image("MODIS/006/MCD12Q1/2018_01_01");
4. var land16 = ee.Image("MODIS/006/MCD12Q1/2016_01_01");
5. var state = ee.FeatureCollection("TIGER/2016/States");
6.
7. var nebraska = state.filterMetadata('NAME', 'contains', 'Nebraska');
8. var neland16 = land16.select('LC_Type1').clip(nebraska);
9. var neland18 = land18.select('LC_Type1').clip(nebraska);
10. var massachusetts = state.filterMetadata('NAME', 'contains', 'Massachusetts');
11. var maland16 = land16.select('LC_Type1').clip(massachusetts);
12. var maland18 = land18.select('LC_Type1').clip(massachusetts);
13.
14. var palette = ['26BB14', // 01 = Evergreen Needleleaf Forest
15.               '26BB14', // 02 = Evergreen Broadleaf Forest
16.               '26BB14', // 03 = Deciduous Needleleaf Forest
17.               '26BB14', // 04 = Deciduous Broadleaf Forest
18.               '26BB14', // 05 = Mixed Deciduous Forest
19.               '26BB14', // 06 = Closed Shrubland
20.               '26BB14', // 07 = Open Shrubland
21.               '26BB14', // 08 = Woody Savanna
22.               '26BB14', // 09 = Savanna
23.               '26BB14', // 10 = Grassland
24.               '26BB14', // 11 = Permanent Wetland
25.               'FFFFFF', // 12 = Cropland
26.               'CC3333', // 13 = Urban
27.               'FFFFFF', // 14 = Crops & Natural Vegetation
28.               'FFFFFF', // 15 = Permanent Snow & Ice 'DOCCAA',
29.               'FFFFFF', // 16 = Barren / Desert
30.               '06AFE8' // 17 = Water
31.               ].join(',');
32. var mapset = { min:1, max:17, opacity:0.7, palette:palette};
```

```
34. // choose the images
35. var images = {
36.   'Nebraska Land Cover in 2016': neland16.visualize(mapset),
37.   'Nebraska Land Cover in 2018': neland18.visualize(mapset),
38.   'Massachusetts Land Cover in 2016': maland16.visualize(mapset),
39.   'Massachusetts Land Cover in 2018': maland18.visualize(mapset),
40. };
41.
42. // create the left and right map, and have them display layer 0 and 1
43. var leftMap = ui.Map();
44. leftMap.setControlVisibility(false);
45. var leftSelector = addLayerSelector(leftMap, 0, 'top-left');
46. var rightMap = ui.Map();
47. rightMap.setControlVisibility(false);
48. var rightSelector = addLayerSelector(rightMap, 1, 'top-right');
49.
50. // add a layer selection widget to allow users to change
51. function addLayerSelector(mapToChange, defaultValue, position) {
52.   var label = ui.Label('Choose an image to visualize');
53.   function updateMap(selection){mapToChange.layers().set(0, ui.Map.Layer(images[selection]))}
54.   var select = ui.Select({items: Object.keys(images), onChange: updateMap});
55.   select.setValue(Object.keys(images)[defaultValue], true);
56.   var controlPanel = ui.Panel({widgets: [label, select], style: {position: position}});
57.   mapToChange.add(controlPanel);
58. }
59.
60. // create a split panel to hold the adjacent, linked maps
61. var splitPanel = ui.SplitPanel({firstPanel: leftMap, secondPanel: rightMap, wipe: true, style:
62.   {stretch: 'both'}});
63.
64. // set the split panel as the only thing in the UI root
65. ui.root.widgets().reset([splitPanel]);
66. 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:

```
1. var newneland16 = neland16.remap([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
2. [1,1,1,1,1,10,10,10,10,10,10,0,0,0,0,0,100], 0, 'LC_Type1');
3. var newneland18 = neland18.remap([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
4. [1,1,1,1,1,10,10,10,10,10,10,0,0,0,0,0,100], 0, 'LC_Type1');
5. var nelandchange = newneland18.subtract(newneland16);
6. var countnelandchange = nelandchange.reduceRegion({reducer: ee.Reducer.frequencyHistogram(),
7. geometry: nebraska, scale: 500});
8. print('Nebraska Land Change:',countnelandchange);
9. // then calculate green loss and gain
10. var negreenloss = nelandchange.eq(-10).or(nelandchange.eq(9).or(nelandchange.eq(-1)));
11. Map.addLayer(negreenloss,{palette:'FFFFFF,DC1010',opacity:0.7}, 'Nebraska Green Loss');
12. var countneloss = negreenloss.reduceRegion({reducer: ee.Reducer.histogram(), geometry: nebraska,
13. scale: 500});
14. print('Nebraska Green Loss:',countneloss);
15. var negreengain = nelandchange.eq(-9).or(nelandchange.eq(1).or(nelandchange.eq(10)));
16. var negain_remap = negreengain.remap([1], [1], null, 'remapped');
17. Map.addLayer(negain_remap,{palette:'ffffff,336633',opacity:0.7}, 'Nebraska Green Gain');
18. var countnegain = negreengain.reduceRegion({reducer: ee.Reducer.histogram(), geometry: nebraska,
19. scale: 500});
20. print('Nebraska Green Gain:',countnegain);
```

Nebraska Land Change:

Object (1 property)

remapped: Object (10 properties)

- 1: 4
- 10: 12226.184313725485
- 9: 56.5921568627451
- 90: 5
- 0: 785112.474509804
- 1: 17
- 10: 2732.6117647058827
- 100: 6
- 9: 5
- 90: 16

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

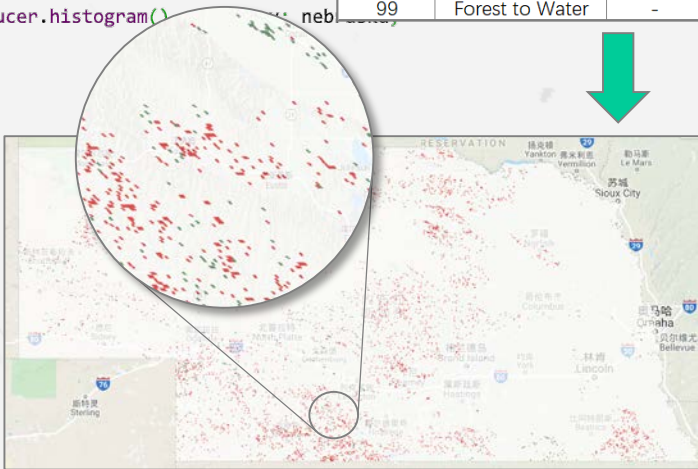


Fig 4 Green Land Gain and Loss in Nebraska

Nebraska Green Loss:

Object (1 property)

remapped: Object (4 properties)

- bucketMeans: [0,1]
- bucketMin: 0
- bucketWidth: 1
- histogram: [787950.6784313729, 12230.184313725485]
- 0: 787950.6784313729
- 1: 12230.184313725485

Nebraska Green Gain:

Object (1 property)

remapped: Object (4 properties)

- bucketMeans: [0,1]
- bucketMin: 0
- bucketWidth: 1
- histogram: [797374.65882353, 2806.203921568628]
- 0: 797374.65882353
- 1: 2806.203921568628

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
```

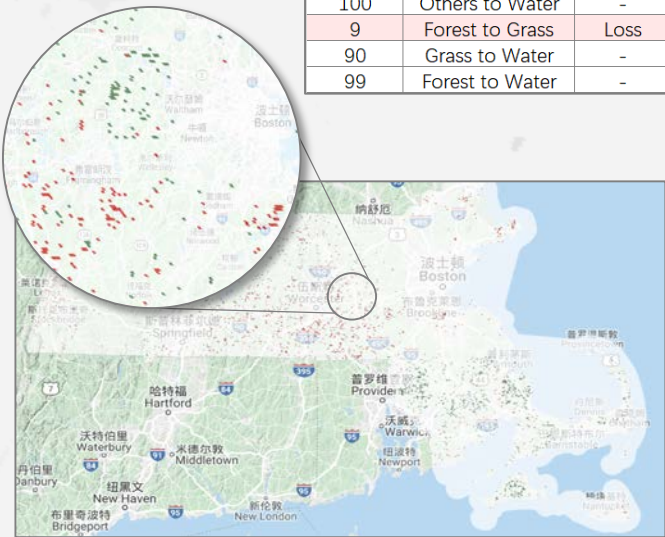


Fig 5 Green Land Gain and Loss in Massachusetts

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 PM_{2.5}, 4) annual SO₂.

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

```
1. # Import necessary modules
2. import sys, os, string, math, arcpy, traceback
3. arcpy.env.overwriteOutput = True
4.
5. try:
6.     # read and write the name of the output shapefile
7.     nameOfInputTable = arcpy.GetParameterAsText(0)
8.     nameOfXField = arcpy.GetParameterAsText(1)
9.     nameOfYField = arcpy.GetParameterAsText(2)
10.    nameOfOutputShapefile = arcpy.GetParameterAsText(3)
11.
12.    # set the local variables
13.    in_Table = nameOfInputTable
14.    x_coords = nameOfXField
15.    y_coords = nameOfYField
16.    out_Layer = nameOfOutputShapefile
17.    saved_Layer = r"C:/Users/USER/Desktop/upenn/LARP743/My Final Project/ArcPy part/output.lyr"
18.
19.    # make the new Layer
20.    arcpy.MakeXYEventLayer_management(in_Table, x_coords, y_coords, out_Layer)
21.
22.    # print the total rows and save to a layer file
23.    print(arcpy.GetCount_management(out_Layer))
24.    arcpy.SaveToLayerFile_management(out_Layer, saved_Layer)
25.
26. except Exception as e:
27.     # If unsuccessful, end gracefully by indicating why
28.     arcpy.AddError('\n' + "Script failed because: \t\t" + e.message)
29.     # ... and where
30.     exceptionreport = sys.exc_info()[2]
31.     fullmessage = traceback.format_tb(exceptionreport)[0]
32.     arcpy.AddError("at this location: \n\n" + fullmessage + "\n")
```

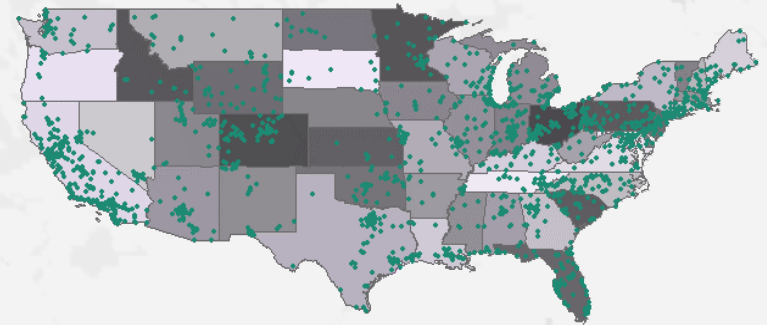
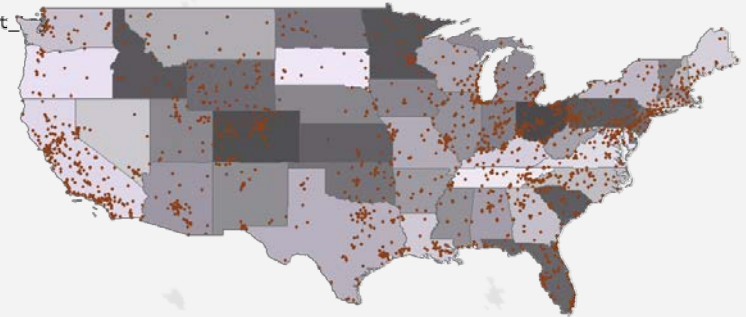


Fig 6 Location of Monitors in 2016 and 2018 (Alaska Excluded)



Introduction

Workflow

Code & Result

Conclusion

Appendix

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.

Introduction

Workflow

Code & Result

Conclusion

Appendix

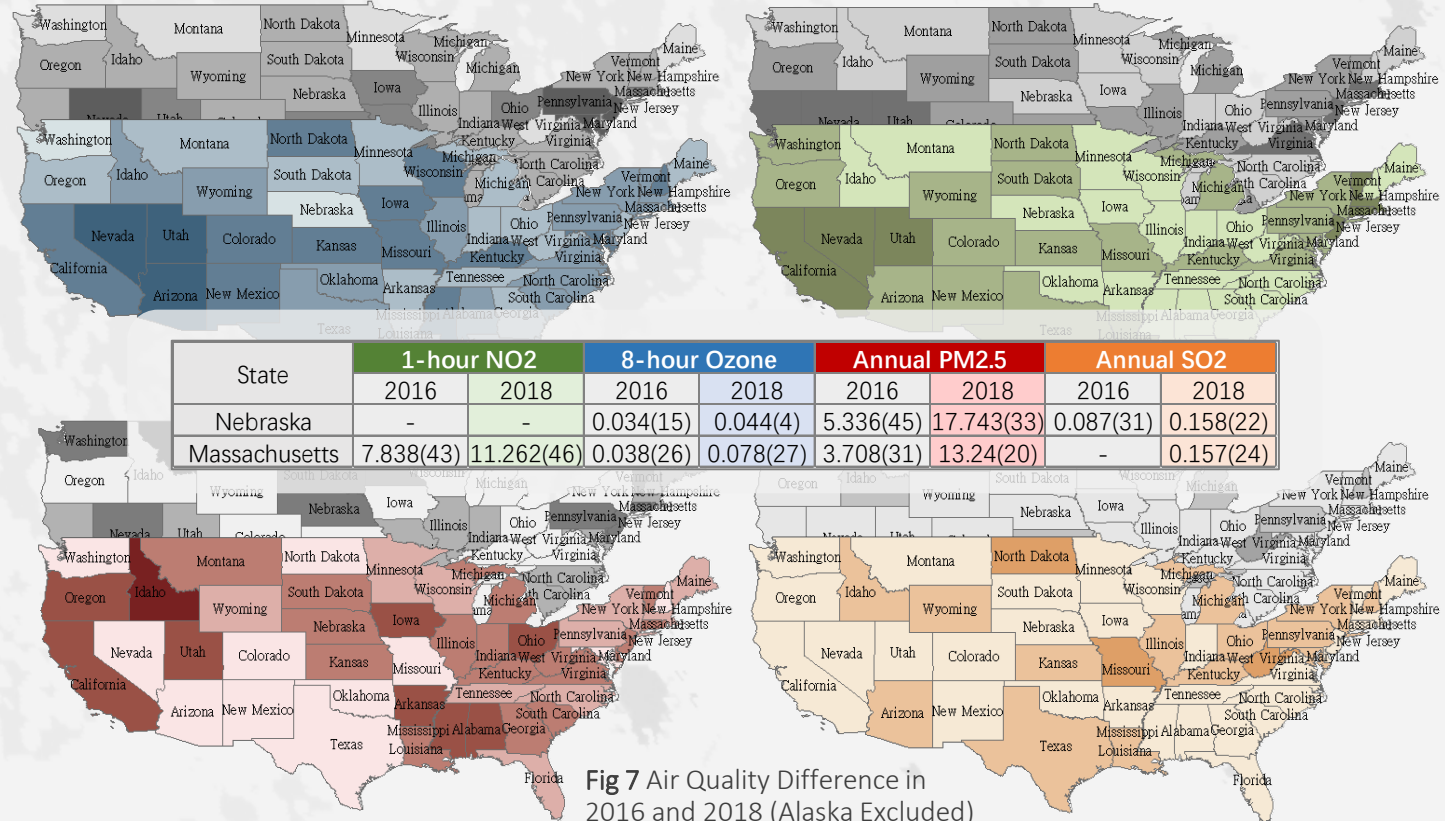


Fig 7 Air Quality Difference in 2016 and 2018 (Alaska Excluded)

Conclusion

Introduction

Workflow

Code & Results

Conclusion

Appendix

- 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 km² 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

Introduction

Workflow

Code & Results

Conclusion

Appendix

```
1. // import all data needed
2. var state = ee.FeatureCollection("TIGER/2016/States");
3. var rawdata = ee.FeatureCollection("users/andisann666/data16to18");
4. var land18 = ee.Image("MODIS/006/MCD12Q1/2018_01_01");
5. var land16 = ee.Image("MODIS/006/MCD12Q1/2016_01_01");
6.
7. // setup and some palettes
8. Map.centerObject(state, 4);
9. var palette1 = ['FFFFFF', 'FF8000'];
10. var palette2 = ['FE8B2', 'FF2828'];
11.
12. // join raw data with state features
13. var joinfilter = ee.Filter.equals("NAME", null, 'State', null);
14. var thejoin = ee.Join.inner();
15. var join = thejoin.apply(state, rawdata, joinfilter);
16. var datastate = join.map(function(featurepair) {
17.   var feature1 = ee.Feature(featurepair.get('primary'));
18.   var feature2 = ee.Feature(featurepair.get('secondary'));
19.   return feature1.set(feature2.toDictionary());
20. });
21.
22. // show joined results
23. print('Original Data', state);
24. print('Joined Data', datastate);
25.
26. // add sales change from 2016 to 2018
27. var salechange = function(feature){
28.   var sale16 = ee.Number(feature.get('EV Sales 2016'));
29.   var sale18 = ee.Number(feature.get('EV Sales 2018'));
30.   return feature.set({'sale change from 16 to
31. 18':sale18.subtract(sale16).divide(sale16).multiply(100).log()});
32. var salestate = datastate.map(salechange);
33.
34. // select and map top sale change states
35. var topchange = salestate.sort('sale change from 16 to 18', false).limit(10);
36. var topchangelist = topchange.aggregate_array('NAME');
37. print("Top EV Sales Change:", topchangelist);
38. var empty = ee.Image().byte();
39. var naptopsale = empty.paint({featureCollection: topchange, color: 'sale change from 16 to 18'});
40. Map.addLayer(naptopsale, {palette:palette2, min:5, max:7, opacity:0.7}, 'Top EV Sales change from 16
to 18');
41.
42. // add generation change from 2016 to 2018
43. var genechange = function(feature){
44.   var gene16 = ee.Number(feature.get('New Energy 2016'));
45.   var gene18 = ee.Number(feature.get('New Energy 2018'));
46.   return feature.set({'% generation change from 16 to
47. 18':gene18.subtract(gene16).divide(gene16).multiply(100)});
48. };
49. var genestate = datastate.map(genechange);
50.
51. // select and map top generation source change states
52. var toppgene = genestate.sort('% generation change from 16 to 18', false).limit(10);
53. var topgeneList = toppgene.aggregate_array('NAME');
54. print("Top Generations Changes", topgeneList);
55. var empty = ee.Image().byte();
56. var naptoggene = empty.paint({featureCollection: toppgene, color: '% generation change from 16 to
57. 18'});
58. Map.addLayer(naptoggene, {palette:palette1, max:100, opacity:0.7}, 'Top Generation Change from 16 to
59. 18');
60.
61. // intersect to get two states: Nebraska & Massachusetts
62. var topchangeft = ee.Feature(topchange.union().first());
63. var toppgeneft = ee.Feature(topgene.union().first());
64. var intersect = topgeneft.intersection(topchangeft);
65. Map.addLayer(intersect, {color:'7180FF', opacity:0.7}, 'Progressed New Energy Usage');
66.
67. // detect Nebraska land cover change
68. // first show the land cover from 16 to 18
69. var nebraska = datastate.filterMetadata('NAME', 'contains', 'Nebraska');
70. var neland16 = land16.select('LC_Type1').clip(nebraska);
71. var neland18 = land18.select('LC_Type1').clip(nebraska);
72. var palette3 = ['268B16', // 01 = Evergreen Needleleaf Forest
73. '26B316', // 02 = Evergreen Broadleaf Forest
74. '26B316', // 03 = Deciduous Needleleaf Forest
75. '26B316', // 04 = Deciduous Broadleaf Forest
76. '26B316', // 05 = Mixed Deciduous Forest
77. '7FD687', // 06 = Closed Shrubland
78. '7FD687', // 07 = Open Shrubland
79. '7FD687', // 08 = Woody Savanna
80. '7FD687', // 09 = Savanna
81. '7FD687', // 10 = Grassland
82. '80C79', // 11 = Permanent Wetland
83. 'FFFFFF', // 12 = Cropland
84. 'CC3333', // 13 = Urban
85. 'FFFFFF', // 14 = Crops & Natural Vegetation
86. 'FFFFFF', // 15 = Permanent Snow & Ice 'DCCAA',
87. 'FFFFFF', // 16 = Barren / Desert
88. '00AEE8', // 17 = Water
89. ].join(",");
90. var napsct = { min:1, max:17, opacity:0.7, palette:palette3};
91. Map.addLayer(neland16, napsct, 'Nebraska Land Cover in 2016');
92. Map.addLayer(neland18, napsct, 'Nebraska Land Cover in 2018');
93.
94. var newneland16 = neland16.remap([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
95. [1,1,1,1,1,10,10,10,10,10,0,0,0,0,0,0,0]);
96. var newneland18 = neland18.remap([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
97. [1,1,1,1,1,10,10,10,10,10,0,0,0,0,0,0,0]);
98. var nelandchange = newneland18.subtract(newneland16);
99. var countnelandchange = nelandchange.reduceRegion({reducer: ee.Reducer.frequencyHistogram(),
100. geometry: nebraska, scale: 500});
101. print("Nebraska Land Change:", countnelandchange);
102.
103. // then calculate green loss and gain
104. var negreenloss = nelandchange.eq(-10).or(nelandchange.eq(9).or(nelandchange.eq(-1)));
105. Map.addLayer(negreenloss, {palette:'FFFFFF,DC8108', opacity:0.7}, 'Nebraska Green Loss');
106. var countnegreenloss = negreenloss.reduceRegion({reducer: ee.Reducer.histogram(), geometry: nebraska,
107. scale: 500});
108. print("Nebraska Green Loss:", countnegreenloss);
109.
110. var negreengain = nelandchange.eq(-9).or(nelandchange.eq(1).or(nelandchange.eq(10)));
111. var negain_remap = negreengain.remap([1, 1], null, 'remapped');
112. Map.addLayer(negain_remap, {palette:'FFFFFF,336633', opacity:0.7}, 'Nebraska Green Gain');
113. var countnegain = negreengain.reduceRegion({reducer: ee.Reducer.histogram(), geometry: nebraska,
114. scale: 500});
115. print("Nebraska Green Gain:", countnegain);
```


Appendix – Full Code

Introduction

Workflow

Code & Results

Conclusion

Appendix

```
110. // detect Massachusetts land cover change
111. // first show the land cover from 16 to 18
112. var massachusetts = datastate.filterMetadata('NAME', 'contains', 'Massachusetts');
113. var maland16 = land16.select('LC_Type1').clip(massachusetts);
114. var maland18 = land18.select('LC_Type1').clip(massachusetts);
115. Map.addLayer(maland16, mapset, 'Massachusetts Land Cover in 2016');
116. Map.addLayer(maland18, mapset, 'Massachusetts Land Cover in 2018');
117. var newmaland16 = maland16.remap([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
118. [1,1,1,1,1,10,10,10,10,10,0,0,0,0,100], 0, 'LC_Type1');
119. var newmaland18 = maland18.remap([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
120. [1,1,1,1,1,10,10,10,10,10,0,0,0,0,100], 0, 'LC_Type1');
121. var malandchange = newmaland18.subtract(newmaland16);
122. var countmalandchange = malandchange.reduceRegion({'reducer': ee.Reducer.frequencyHistogram(),
123. geometry: massachusetts, scale: 500});
124. print('Massachusetts Land Change:', countmalandchange);
```

```
125. // then calculate green loss and gain
126. var magreenloss = malandchange.eq(10).or(malandchange.eq(9).or(malandchange.eq(1)));
127. Map.addLayer(magreenloss, {palette: 'FFFFFF,DC1010', opacity: 0.7}, 'Massachusetts Green Loss');
128. var countmaloss = magreenloss.reduceRegion({'reducer': ee.Reducer.histogram(), geometry: massachusetts,
129. scale: 500});
129. print('Massachusetts Green Loss:', countmaloss);
130. var magreengain = malandchange.eq(9).or(malandchange.eq(1).or(malandchange.eq(10)));
131. var magain_remap = magreengain.remap([1], [1], null, 'renapped');
132. Map.addLayer(magain_remap, {palette: 'FFFFFF,336633', opacity: 0.7}, 'Massachusetts Green Gain');
133. var countmagain = magreengain.reduceRegion({'reducer': ee.Reducer.histogram(), geometry: massachusetts,
134. scale: 500});
134. print('Massachusetts Green Gain:', countmagain);
135.
136. // detect air pollution change *see ArcPy part*
```

Google Earth Engine - Script TWO

```
1. // another method: compare 2016/2018 images with a split panel
2. // import all data, pre-process, and mapset
3. var land18 = ee.Image("MODIS/006/MCD12Q1/2018_01_01");
4. var land16 = ee.Image("MODIS/006/MCD12Q1/2016_01_01");
5. var state = ee.FeatureCollection("TIGER/2016/States");
6.
7. var nebraska = state.filterMetadata('NAME', 'contains', 'Nebraska');
8. var neland16 = land16.select('LC_Type1').clip(nebraska);
9. var neland18 = land18.select('LC_Type1').clip(nebraska);
10. var massachusetts = state.filterMetadata('NAME', 'contains', 'Massachusetts');
11. var maland16 = land16.select('LC_Type1').clip(massachusetts);
12. var maland18 = land18.select('LC_Type1').clip(massachusetts);
13.
14. var palette = ['268814', // 01 = Evergreen Needleleaf Forest
15. '268814', // 02 = Evergreen Broadleaf Forest
16. '268814', // 03 = Deciduous Needleleaf Forest
17. '268814', // 04 = Deciduous Broadleaf Forest
18. '268814', // 05 = Mixed Deciduous Forest
19. '268814', // 06 = Closed Shrubland
20. '268814', // 07 = Open Shrubland
21. '268814', // 08 = Woody Savanna
22. '268814', // 09 = Savanna
23. '268814', // 10 = Grassland
24. '268814', // 11 = Permanent Wetland
25. 'FFFFFF', // 12 = Cropland
26. 'CC3333', // 13 = Urban
27. 'FFFFFF', // 14 = Crops & Natural Vegetation
28. 'FFFFFF', // 15 = Permanent Snow & Ice 'DDCCAA',
29. 'FFFFFF', // 16 = Barren / Desert
30. '06AFE8', // 17 = Water
31. ].join(',');
32. var mapset = { min:1, max:17, opacity:0.7, palette:palette};
```

```
34. // choose the images
35. var images = {
36. 'Nebraska Land Cover in 2016': neland16.visualize(mapset),
37. 'Nebraska Land Cover in 2018': neland18.visualize(mapset),
38. 'Massachusetts Land Cover in 2016': maland16.visualize(mapset),
39. 'Massachusetts Land Cover in 2018': maland18.visualize(mapset),
40. };
41.
42. // create the left and right map, and have them display Layer 0 and 1
43. var leftMap = ui.Map();
44. leftMap.setControlVisibility(false);
45. var leftSelector = addLayerSelector(leftMap, 0, 'top-left');
46. var rightMap = ui.Map();
47. rightMap.setControlVisibility(false);
48. var rightSelector = addLayerSelector(rightMap, 1, 'top-right');
49.
50. // add a Layer selection widget to allow users to change
51. function addLayerSelector(mapToChange, defaultValue, position) {
52. var label = ui.Label('Choose an image to visualize');
53. function updateMap(selection){mapToChange.layers().set(0, ui.Map.Layer(images[selection]))}
54. var select = ui.Select({items: Object.keys(images), onChange: updateMap});
55. select.setValue(Object.keys(images)[defaultValue], true);
56. var controlPanel = ui.Panel({widgets: [label, select], style: {position: position}});
57. mapToChange.add(controlPanel);
58. }
59.
60. // create a split panel to hold the adjacent, linked maps
61. var splitPanel = ui.SplitPanel({firstPanel: leftMap, secondPanel: rightMap, wipe: true, style:
62. {stretch: 'both'}});
63.
64. // set the split panel as the only thing in the UI root
65. ui.root.widgets().reset([splitPanel]);
66. var linker = ui.Map.Linker([leftMap, rightMap]);
```

Appendix – Full Code

ArcPy- Script THREE

```
1. # Import necessary modules
2. import sys, os, string, math, arcpy, traceback
3. arcpy.env.overwriteOutput = True
4.
5. try:
6.     # read and write the name of the output shapefile
7.     nameOfInputTable = arcpy.GetParameterAsText(0)
8.     nameOfXField = arcpy.GetParameterAsText(1)
9.     nameOfYField = arcpy.GetParameterAsText(2)
10.    nameOfOutputShapefile = arcpy.GetParameterAsText(3)
11.
12.    # set the local variables
13.    in_Table = nameOfInputTable
14.    x_coords = nameOfXField
15.    y_coords = nameOfYField
16.    out_Layer = nameOfOutputShapefile
17.    saved_Layer = r"C:/Users/USER/Desktop/upenn/LARP743/My Final Project/Arcpy part/output.lyr"
18.
19.    # make the new Layer
20.    arcpy.MakeXYEventLayer_management(in_Table, x_coords, y_coords, out_Layer)
21.
22.    # print the total rows and save to a layer file
23.    print(arcpy.GetCount_management(out_Layer))
24.    arcpy.SaveToLayerFile_management(out_Layer, saved_Layer)
25.
26. except Exception as e:
27.     # If unsuccessful, end gracefully by indicating why
28.     arcpy.AddError('\n' + "Script failed because: \t\t" + e.message)
29.     # ... and where
30.     exceptionreport = sys.exc_info()[2]
31.     fullmessage = traceback.format_tb(exceptionreport)[0]
32.     arcpy.AddError("at this location: \n\n" + fullmessage + "\n")
```

ArcPy- Script FOUR

```
1. import sys, os, string, math, arcpy, traceback
2. arcpy.env.overwriteOutput = True
3.
4. try:
5.     nameOfStateInput = arcpy.GetParameterAsText(0)
6.     nameOfMonitorInput = arcpy.GetParameterAsText(1)
7.     nameOfFieldInput = arcpy.GetParameterAsText(2)
8.     nameOfMeanOutput = arcpy.GetParameterAsText(3)
9.
10.    # Want to join field value to states and calculate the mean value for each state
11.    targetFeatures = nameOfStateInput
12.    joinFeatures = nameOfMonitorInput
13.    fieldName = nameOfFieldInput
14.
15.    # Create a new fieldmappings and add the two input feature classes.
16.    fieldmappings = arcpy.FieldMappings()
17.    fieldmappings.addTable(targetFeatures)
18.    fieldmappings.addTable(joinFeatures)
```

```
20. # First get the Monitor fieldmap.
21. # The output will have the states with the attributes of the specific field.
22. # Setting the field's merge rule to mean will aggregate the values
23. # for all of the prices for each state into an average value.
24. # The field is also renamed to be more appropriate for the output.
25. fieldIndex = fieldmappings.findFieldMapIndex(fieldName)
26. fieldmap = fieldmappings.getFieldMap(fieldIndex)
27. field = fieldmap.outputField
28.
29. # Rename the field and pass the updated field object back into the field map
30. newField = "mean_" + fieldName[:3]
31. field.name = newField
32. field.aliasName = newField
33. fieldmap.outputField = field
34.
35. # Set the merge rule to mean and then replace the old fieldmap in the mappings object
36. # with the updated one
37. fieldmap.mergeRule = "mean"
38. fieldmappings.replaceFieldMap(fieldIndex, fieldmap)
39.
40. # spatial join, add new field
41. newName = "new" + fieldName[:3]
42. arcpy.SpatialJoin_analysis(targetFeatures, joinFeatures, nameOfMeanOutput, "M", "M",
43.                             fieldmappings)
44. arcpy.AddField_management(nameOfMeanOutput, newName, "DOUBLE", 10, 4)
45. enumerationOfRecords = arcpy.UpdateCursor(nameOfMeanOutput)
46.
47. # Loop through that enumeration, set mean sale value for each record
48. for nextRecord in enumerationOfRecords:
49.     meanValue = nextRecord.getValue(newField)
50.     nextRecord.setValue(newName, meanValue)
51.     enumerationOfRecords.updateRow(nextRecord)
52.
53. # Delete row and update cursor objects
54. del nextRecord
55. del enumerationOfRecords
56.
57. except Exception as e:
58.     arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
59.     exceptionreport = sys.exc_info()[2]
60.     fullmessage = traceback.format_tb(exceptionreport)[0]
61.     arcpy.AddError("at this location: \n\n" + fullmessage + "\n")
```

Introduction

Workflow

Code & Results

Conclusion

Appendix