FINDING THE SUITABLE ROAD-TESTING SITES FOR AUTONOMOUS VEHICLES IN PHILADELPHIA

XINYI QIU

TABLE OF CONTENTS

INTRODUCTION

The widespread of Autonomous Vehicle (AV) has never been seen as close as it seems now with the confluence of advanced technologies and supportive regulations from local governments. The United States (US) is home to companies that lead the world in AV development, contributing to strong performance in technology and innovation ranking No.1 around the world[1]. In the Automated vehicles 3.0 published in October 2018 by the US Department of Transportation, it declared principles towards AVs including prioritizing safety and preparing for automation through guidance and pilot programs, etc[2].

Pennsylvania, with two world-class research universities, Carnegie Mellon University and University of Pennsylvania, has emerged as a leading location for on-road testing of HAVs. To ensure the safety on road, Pennsylvania has passed two laws related to autonomous vehicles and regulations by the department of transportation. However, there is still a long way to go to avoid the fatal accident in Arizona. The safety of AVs on road depends on the operation system, human behavior and external environment. While engineers are working on the technologies of AV itself to improve the safety and accuracy of system, we, as a transportation planner, will work on the road infrastructure and regulations on AVs. The term project aims to provide a baseline for PennDOT to regulate the location where AVs could test on road, considering the AV's reaction to the external environments, including road conditions, geological conditions, weather conditions and demographic conditions.

[1] World Economic Forum Networked Readiness Index, 2018.

[2] Preparing for the Future of Transportation: Automated Vehicles 3.0. https://www.transportation.gov/av/3

DATA SOURCES

- Philadelphia City limits shape file
- Philadelphia 2010 Census tracts shapefile
- Philadelphia Street Centerlines line shapefile
- Philadelphia High Injury Network shapefile
- Pennsylvania Traffic Volumes shapefile
- SRTM Digital Elevation Data Version 4 image
- Philadelphia population, age, employment and disabilities census data

METHODS

The project will combine the consideration of road conditions, slope, and demographic conditions and choose the suitable places and roads for AVs' road tests.

Road conditions take high injury network, traffic volume and street class into consideration and extract suitable roads for road tests where have low injury, low traffic volume and low speed limit.

Demographic conditions take population density, percentage of the elderly over 65 and the kids, the percentage of the disabilities and the number of vehicles. Use weights to calculate the total score of each census tracts and choose the best places for road tests.

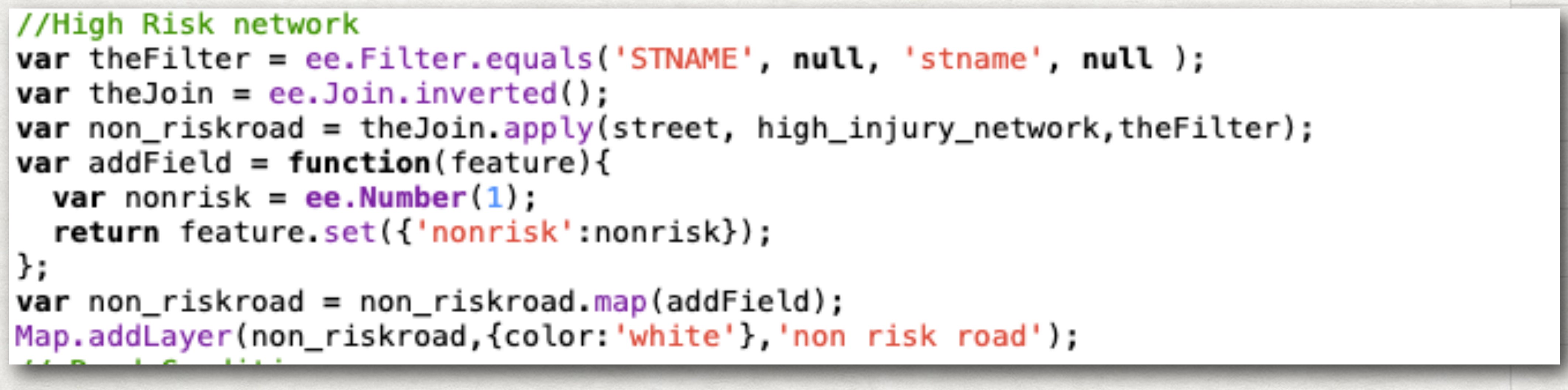
Finally, combine the suitable road and the census tracts to find the best places for Philadelphia to have a pilot project before it is widely used.

ROAD CONDITIONS

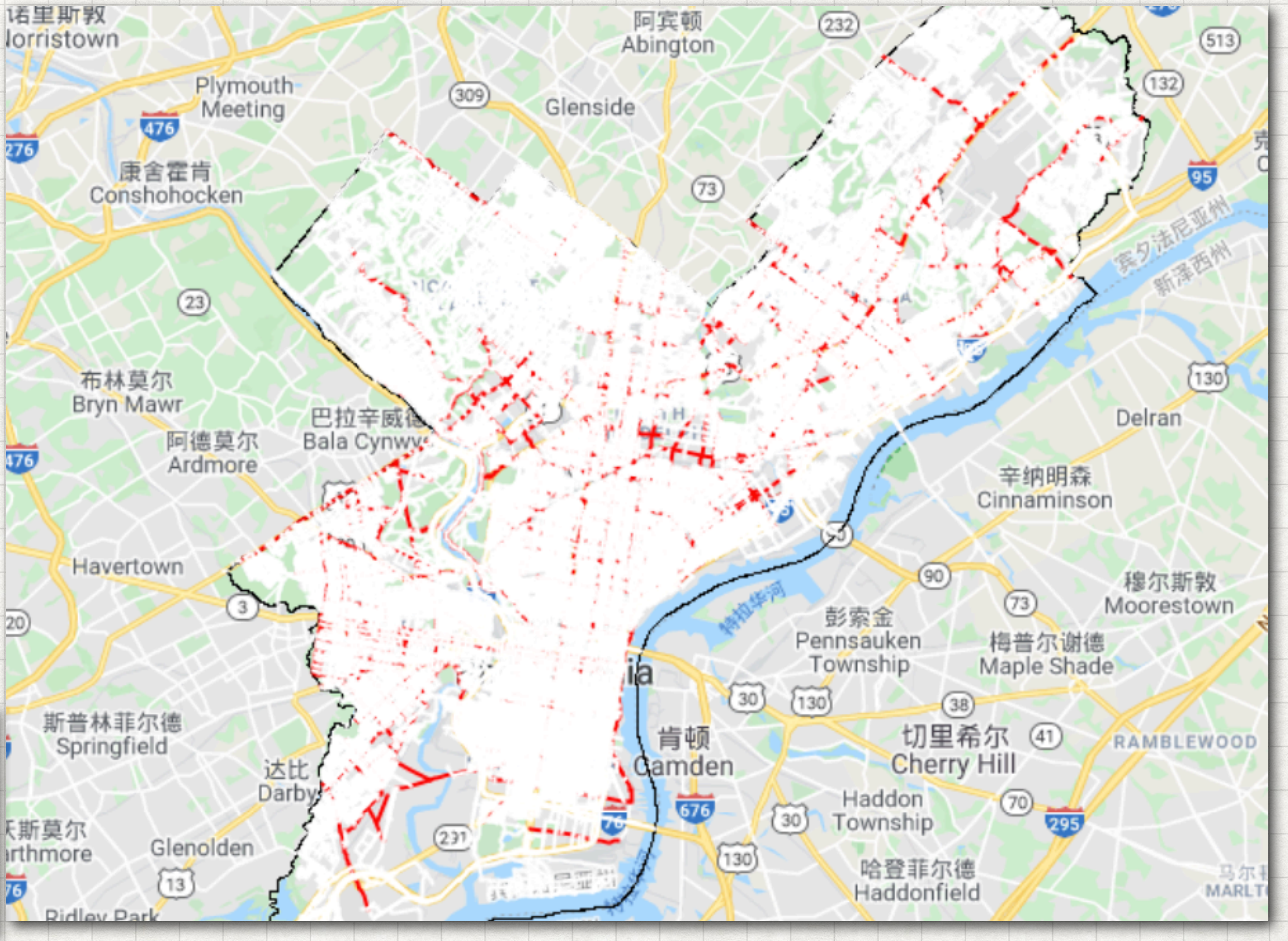
HIGH INJURY NETWORK

-Use *ee.Join.invert()* to exclude high injury network from the streets of Philadelphia, for those high injury network could not be used as road test sites for Autonomous Vehicles

-The image on the left has two layers, one is the white road network which is not risky, and the red road network which is more likely to have crashes due to the analysis of Vision Zero



```
//High Risk network
var theFilter = ee.Filter.equals('STNAME', null, 'stname', null );
var theJoin = ee.Join.inverted();
var non_riskroad = theJoin.apply(street, high_injury_network,theFilter);
var addField = function(feature){
  var nonrisk = ee.Number(1);
  return feature.set({'nonrisk':nonrisk});
};
var non_riskroad = non_riskroad.map(addField);
Map.addLayer(non_riskroad,{color:'white'},'non risk road');
```



White represent non-risk road, and red represent high injury network

ROAD CONDITIONS

STREET CLASS

-Street class :

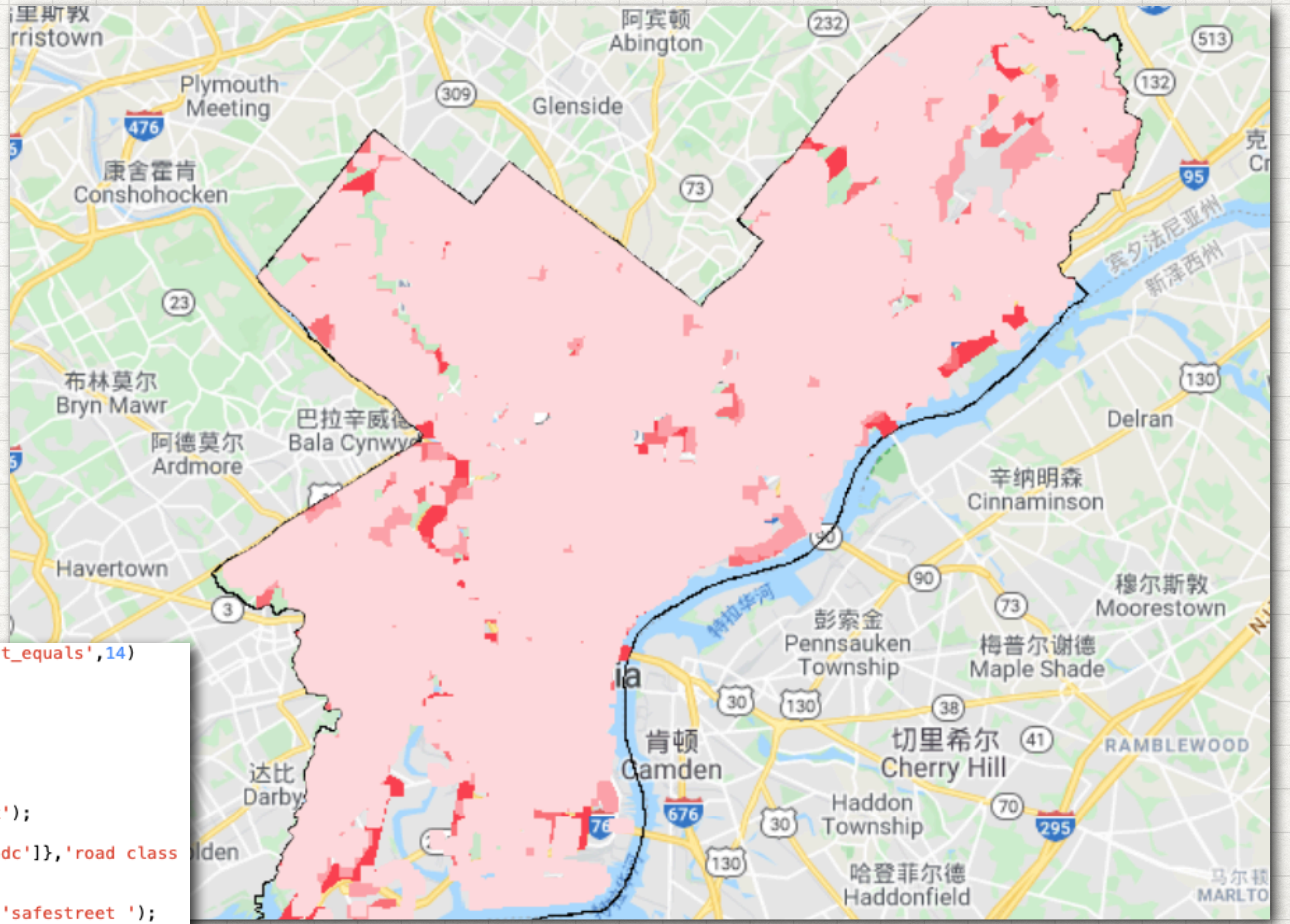
0-Navy Yard; 1-Expressways; 2-Major Arterial; 3-Minor Arterial; 4-Collector; 5-Local; 6-Driveway; 9-Low Speed Ramps; 10-High Speed Ramps; 12-Non Travelable; 14-City Boundary; 15-Walking Connector

-Exclude street class of 12, 14 and 15 where vehicles are not permitted

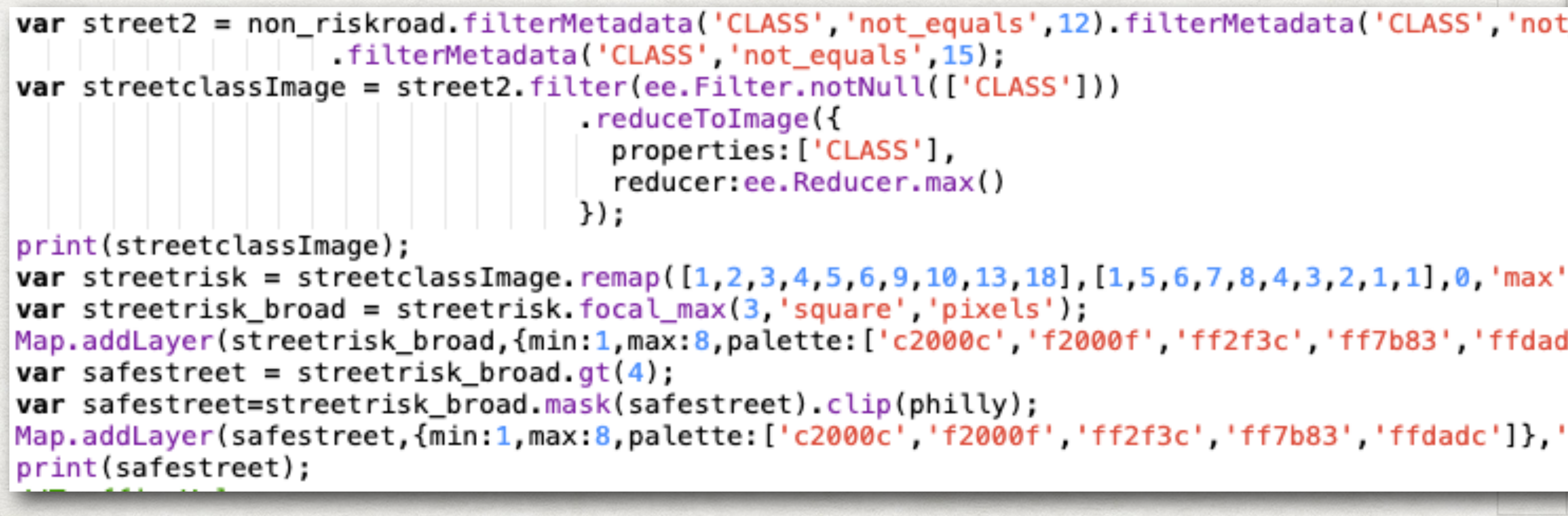
-Transfer the street centerline feature collection into raster and then remap the value of CLASS into the risk score of each class (1-risky; 10-safe)

-For the further analysis, use *image.focal_max()* to widen the range of streets

-Choose the road with value greater than 4 and display the map



Safe Street with suitable street class



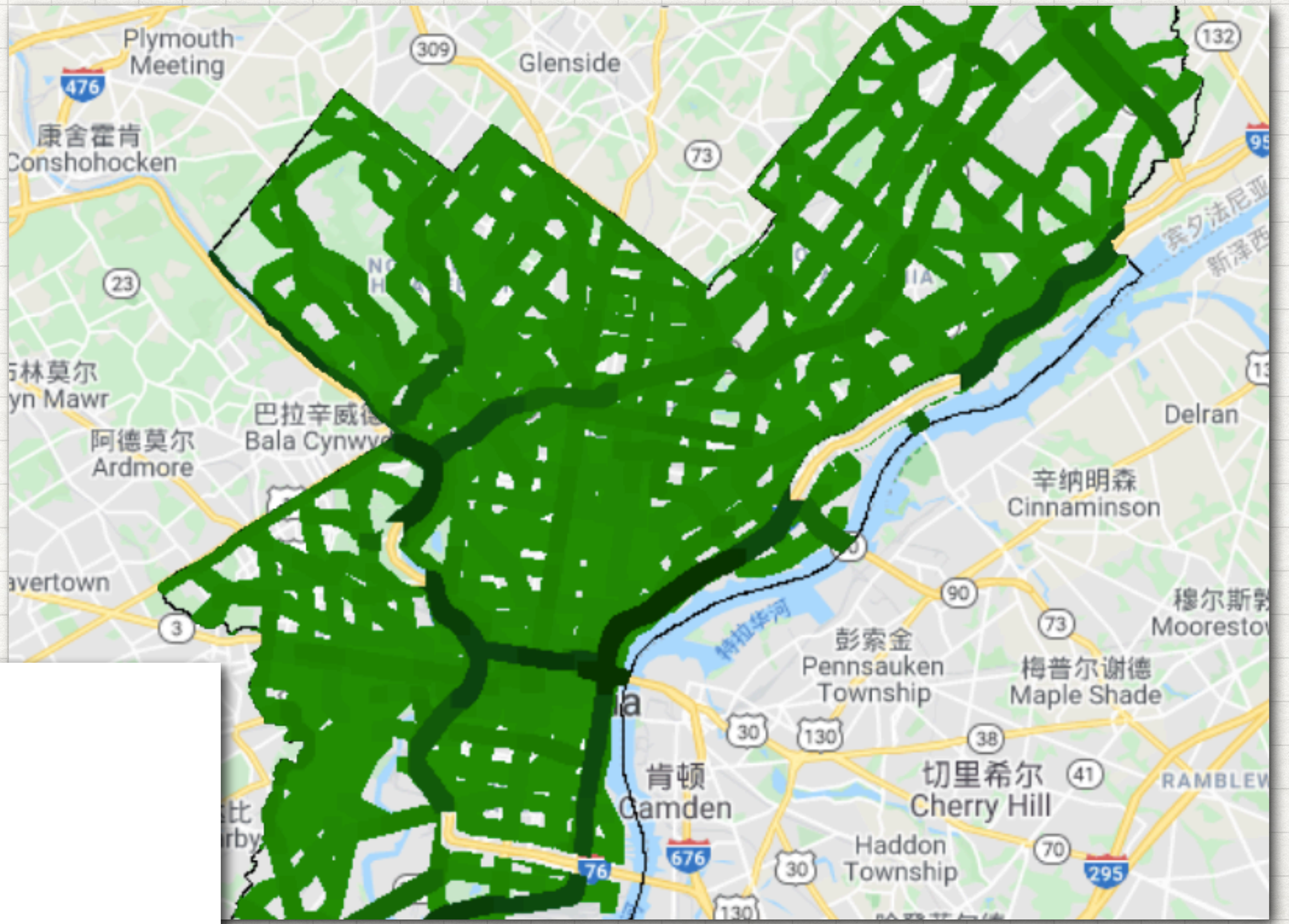
```
var street2 = non_riskroad.filterMetadata('CLASS','not_equals',12).filterMetadata('CLASS','not_equals',14)
                 .filterMetadata('CLASS','not_equals',15);
var streetclassImage = street2.filter(ee.Filter.notNull(['CLASS']))
                                .reduceToImage({
                                  properties:['CLASS'],
                                  reducer:ee.Reducer.max()
                                });
print(streetclassImage);
var streetrisk = streetclassImage.remap([1,2,3,4,5,6,9,10,13,18],[1,5,6,7,8,4,3,2,1,1],0,'max');
var streetrisk_broad = streetrisk.focal_max(3,'square','pixels');
Map.addLayer(streetrisk_broad,{min:1,max:8,palette:['c2000c','f2000f','ff2f3c','ff7b83','ffdadc']},'road class
var safestreet = streetrisk_broad.gt(4);
var safestreet=streetrisk_broad.mask(safestreet).clip(philly);
Map.addLayer(safestreet,{min:1,max:8,palette:['c2000c','f2000f','ff2f3c','ff7b83','ffdadc']},'safestreet ');
print(safestreet);
```

ROAD CONDITIONS

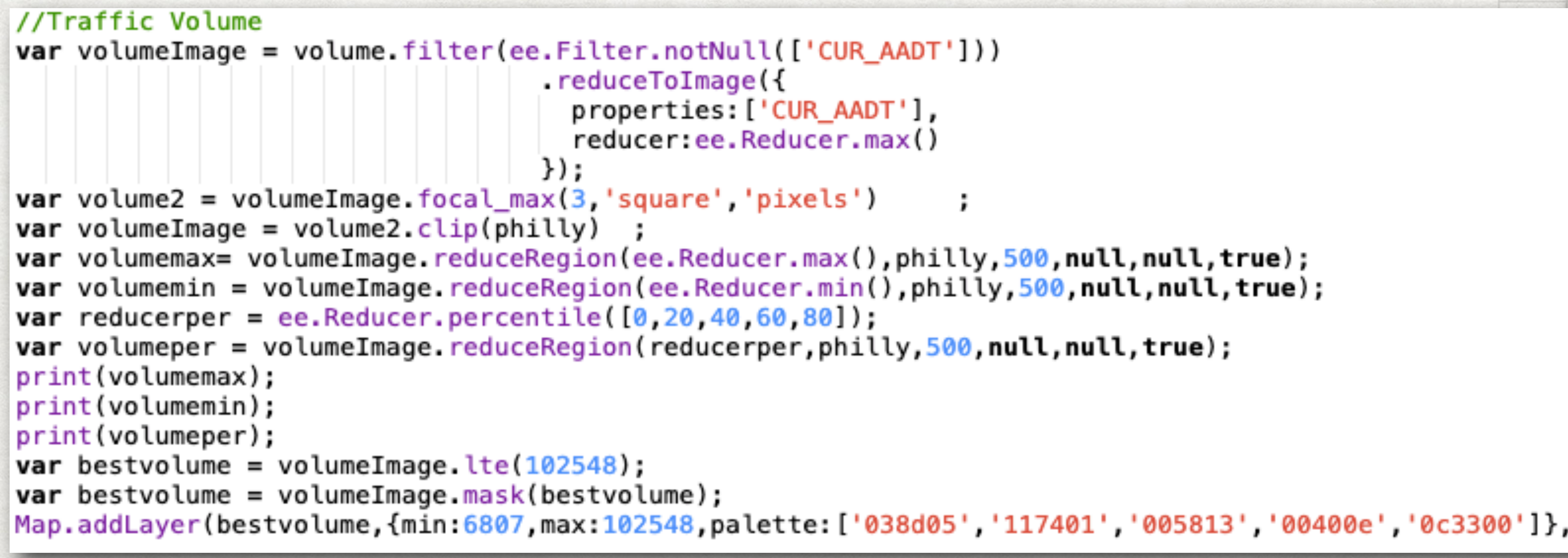
TRAFFIC VOLUME

- Transfer the traffic volume feature collection into raster, using *current annual average daily traffic* as the value of each pixel

-Choose the streets with volume less than 80 percentile of the traffic volume in Philadelphia



Street with volume less than 80 percentile of the traffic volume in Philadelphia

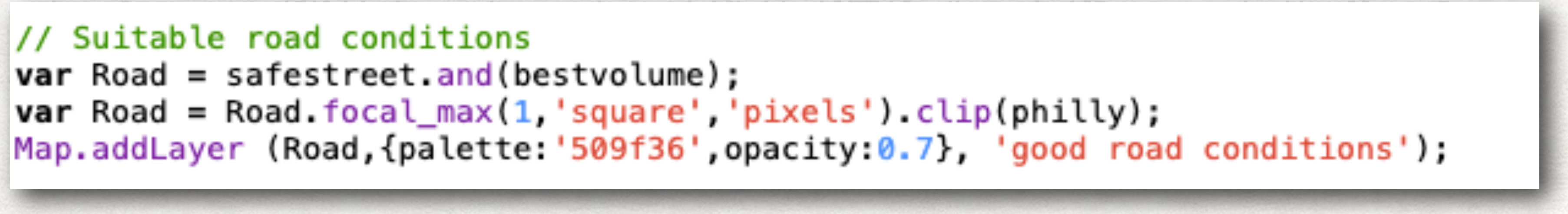


```
//Traffic Volume
var volumeImage = volume.filter(ee.Filter.notNull(['CUR_AADT']))
                                       .reduceToImage({
                                         properties:['CUR_AADT'],
                                         reducer:ee.Reducer.max()
                                       });
var volume2 = volumeImage.focal_max(3,'square','pixels')      ;
var volumeImage = volume2.clip(philly)  ;
var volumemax= volumeImage.reduceRegion(ee.Reducer.max(),philly,500,null,null,true);
var volumemin = volumeImage.reduceRegion(ee.Reducer.min(),philly,500,null,null,true);
var reducerper = ee.Reducer.percentile([0,20,40,60,80]);
var volumeper = volumeImage.reduceRegion(reducerper,philly,500,null,null,true);
print(volumemax);
print(volumemin);
print(volumeper);
var bestvolume = volumeImage.lte(102548);
var bestvolume = volumeImage.mask(bestvolume);
Map.addLayer(bestvolume,{min:6807,max:102548,palette:['038d05','117401','005813','00400e','0c3300']},'suitable volume');
```

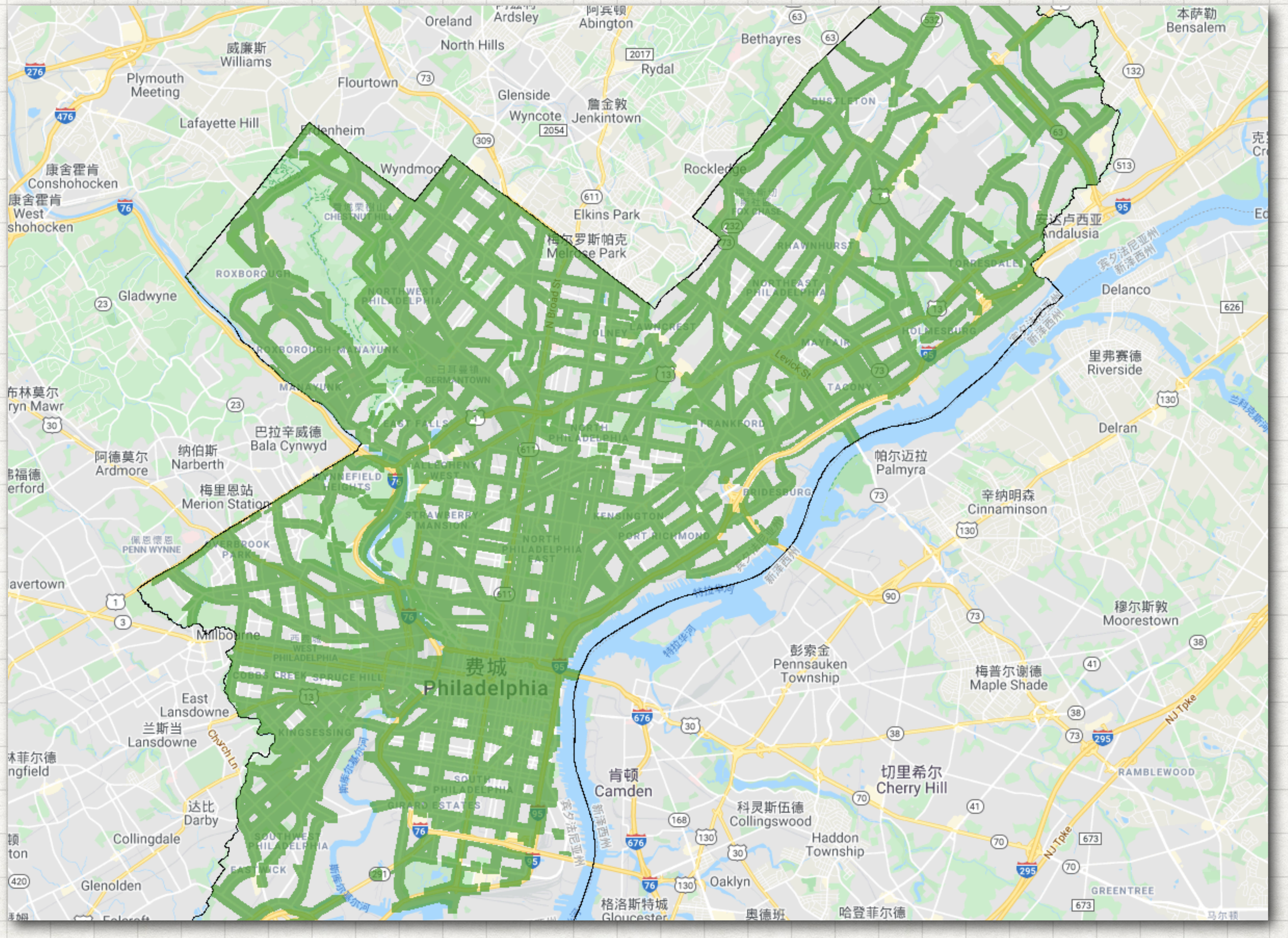
ROAD CONDITIONS

FINAL MAP

- *image.focal_max()* is used for fulfill a street to avoid multiple points in one street



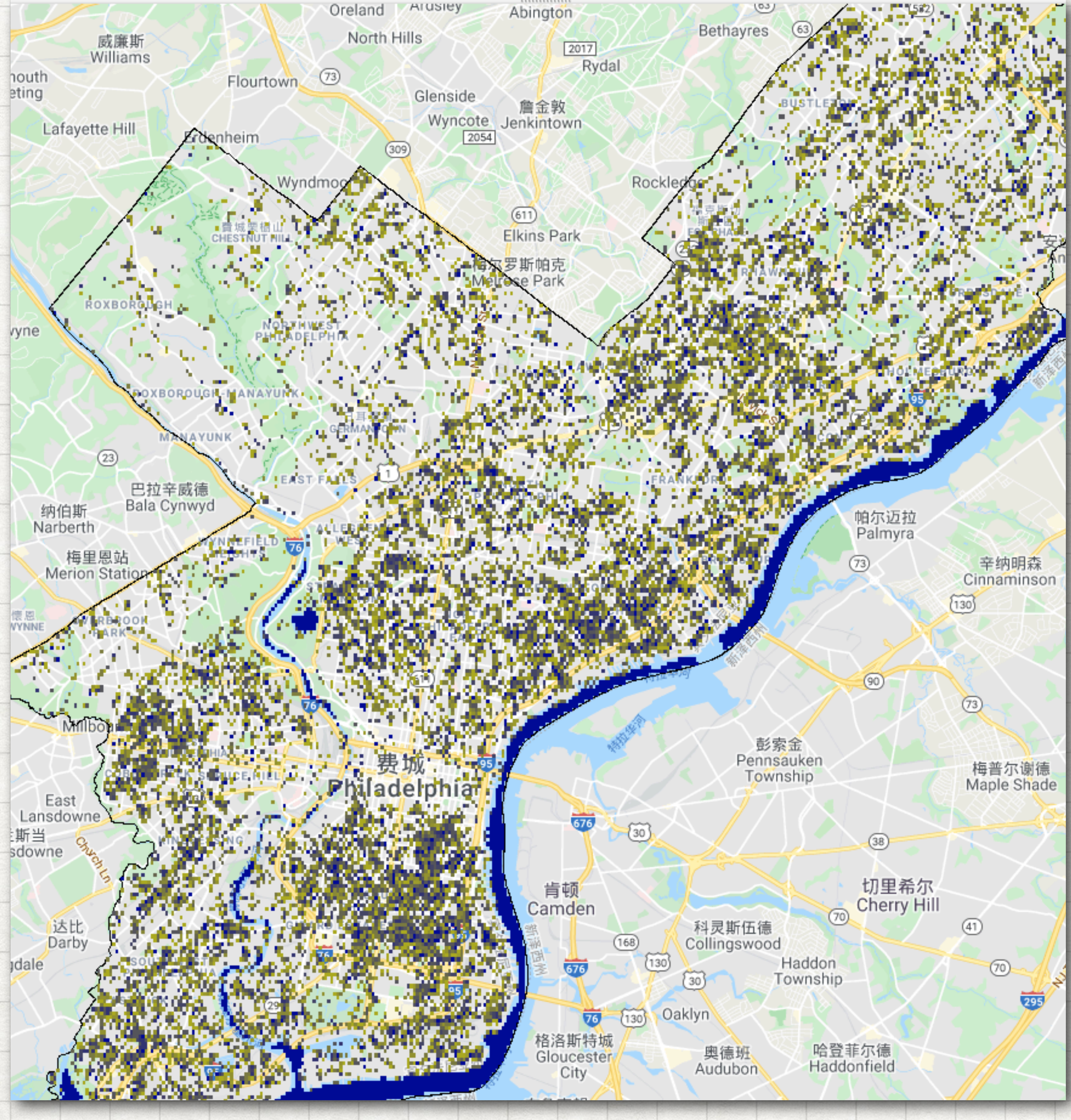
```
// Suitable road conditions
var Road = safestreet.and(bestvolume);
var Road = Road.focal_max(1,'square','pixels').clip(philly);
Map.addLayer (Road,{palette:'509f36',opacity:0.7}, 'good road conditions');
```



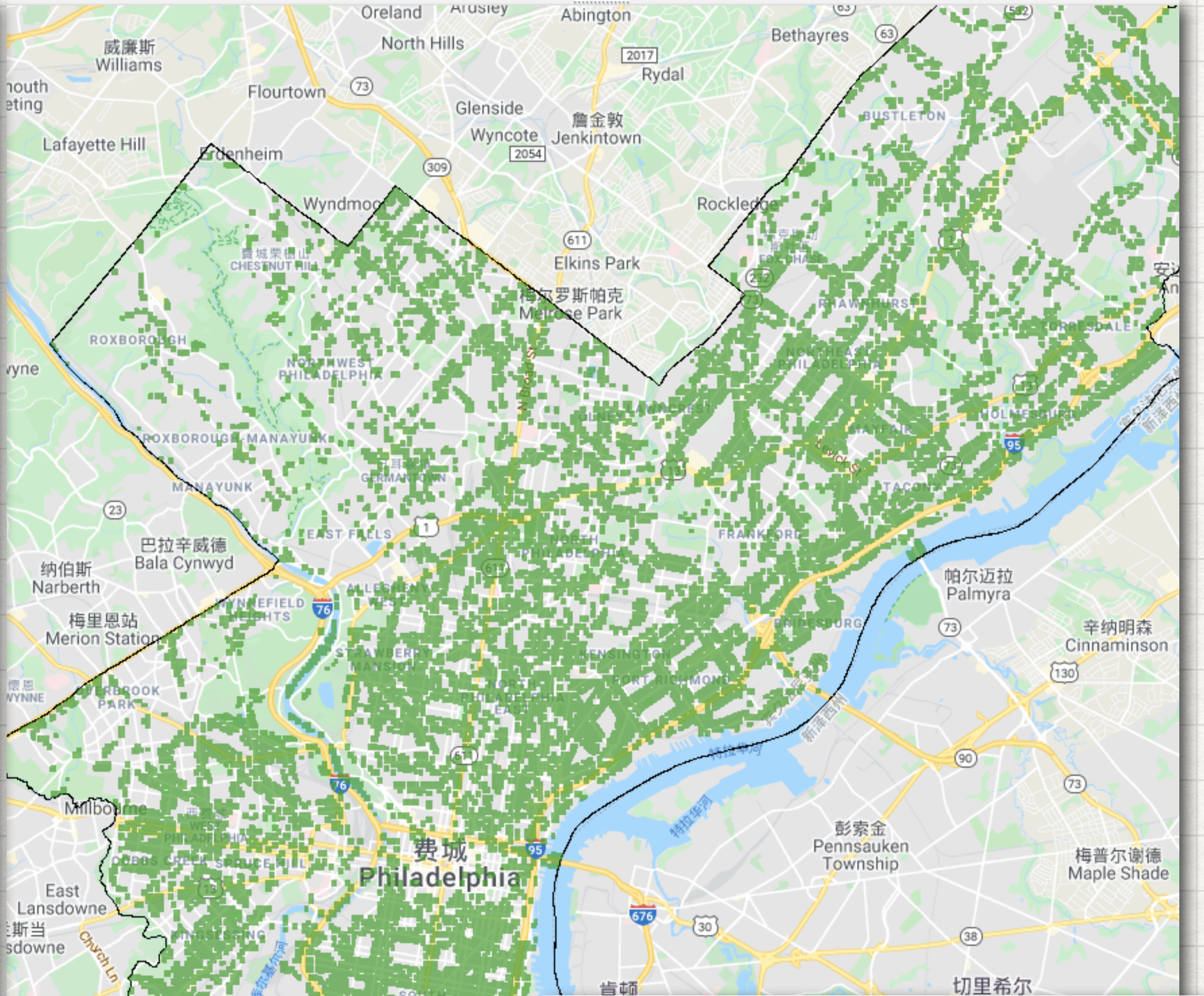
Suitable Streets considering the road conditions

GEOLOGICAL CONDITION

SLOPE

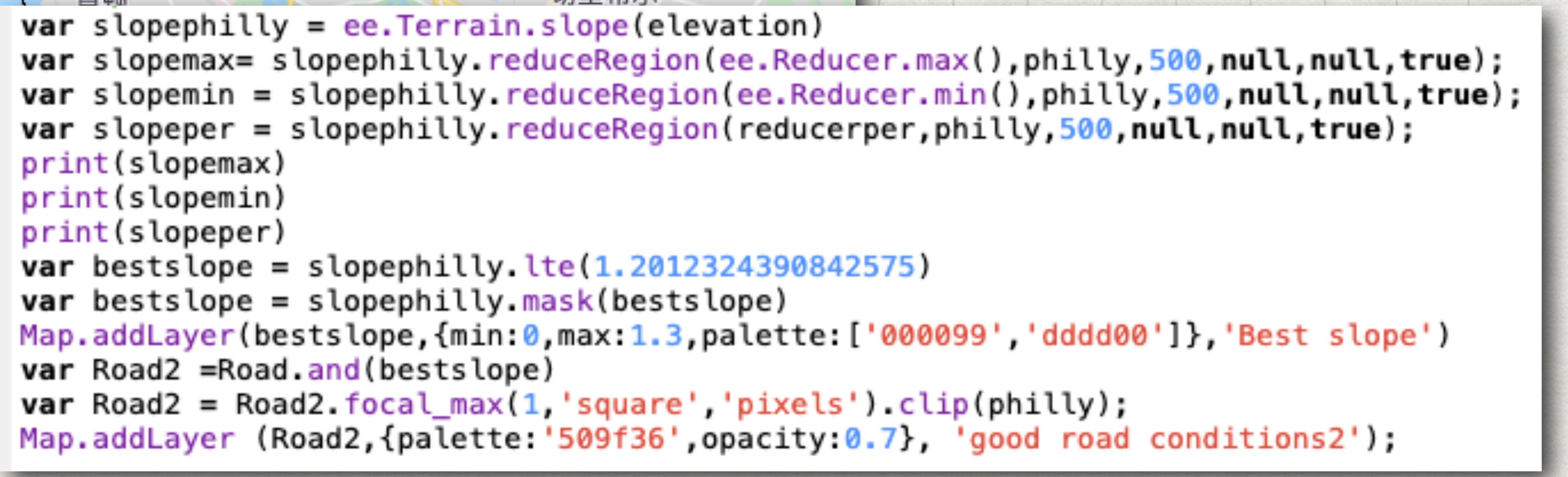


Suitable Slops



Suitable Streets

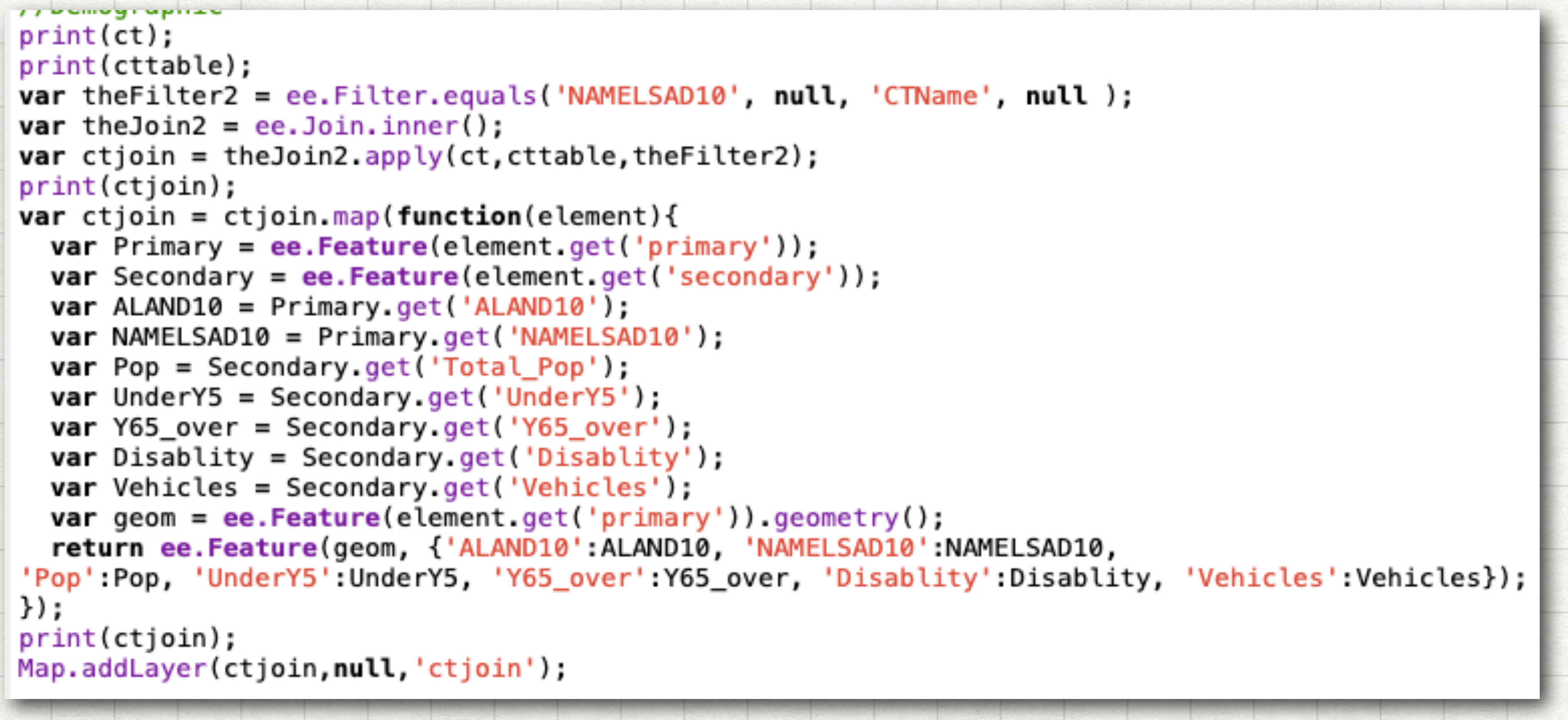
Slope is one of the most important factors that affect the reaction of Autonomous Vehicles. Therefore, I choose the slope less than 1.2, which is the 80 percentile of the slope in Philadelphia.



```
var slopephilly = ee.Terrain.slope(elevation)
var slopemax= slopephilly.reduceRegion(ee.Reducer.max(),philly,500,null,null,true);
var slopemin = slopephilly.reduceRegion(ee.Reducer.min(),philly,500,null,null,true);
var slopeper = slopephilly.reduceRegion(reducerper,philly,500,null,null,true);
print(slopemax)
print(slopemin)
print(slopeper)
var bestslope = slopephilly.lte(1.2012324390842575)
var bestslope = slopephilly.mask(bestslope)
Map.addLayer(bestslope,{min:0,max:1.3,palette:['000099','dddd00']},'Best slope')
var Road2 =Road.and(bestslope)
var Road2 = Road2.focal_max(1,'square','pixels').clip(philly);
Map.addLayer (Road2,{palette:'509f36',opacity:0.7}, 'good road conditions2');
```

DEMOGRAPHIC CONDITIONS

IMPORT .CSV

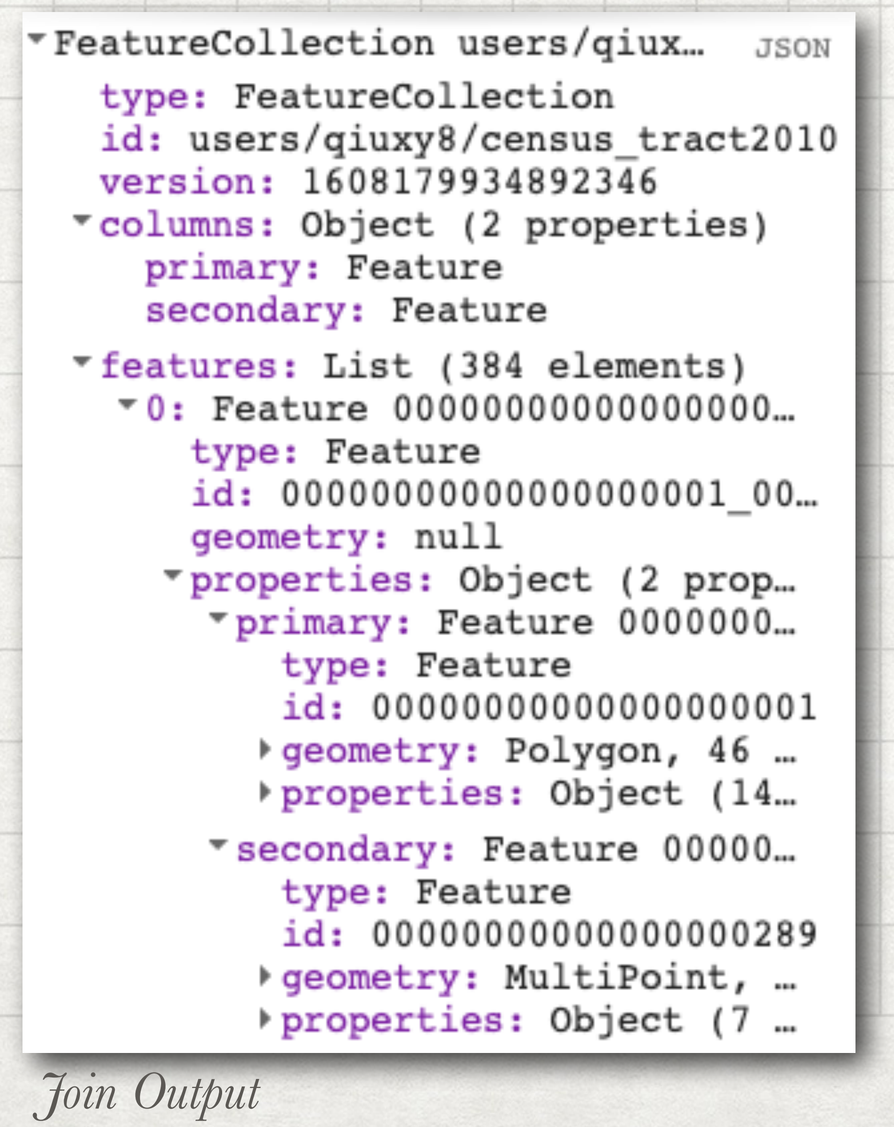


```
//Demographic
print(ct);
print(cttable);
var theFilter2 = ee.Filter.equals('NAMELSAD10', null, 'CTName', null );
var theJoin2 = ee.Join.inner();
var ctjoin = theJoin2.apply(ct,cttable,theFilter2);
print(ctjoin);
var ctjoin = ctjoin.map(function(element){
  var Primary = ee.Feature(element.get('primary'));
  var Secondary = ee.Feature(element.get('secondary'));
  var ALAND10 = Primary.get('ALAND10');
  var NAMELSAD10 = Primary.get('NAMELSAD10');
  var Pop = Secondary.get('Total_Pop');
  var UnderY5 = Secondary.get('UnderY5');
  var Y65_over = Secondary.get('Y65_over');
  var Disablity = Secondary.get('Disablity');
  var Vehicles = Secondary.get('Vehicles');
  var geom = ee.Feature(element.get('primary')).geometry();
  return ee.Feature(geom, {'ALAND10':ALAND10, 'NAMELSAD10':NAMELSAD10,
'Pop':Pop, 'UnderY5':UnderY5, 'Y65_over':Y65_over, 'Disablity':Disablity, 'Vehicles':Vehicles});
});
print(ctjoin);
Map.addLayer(ctjoin,null,'ctjoin');
```

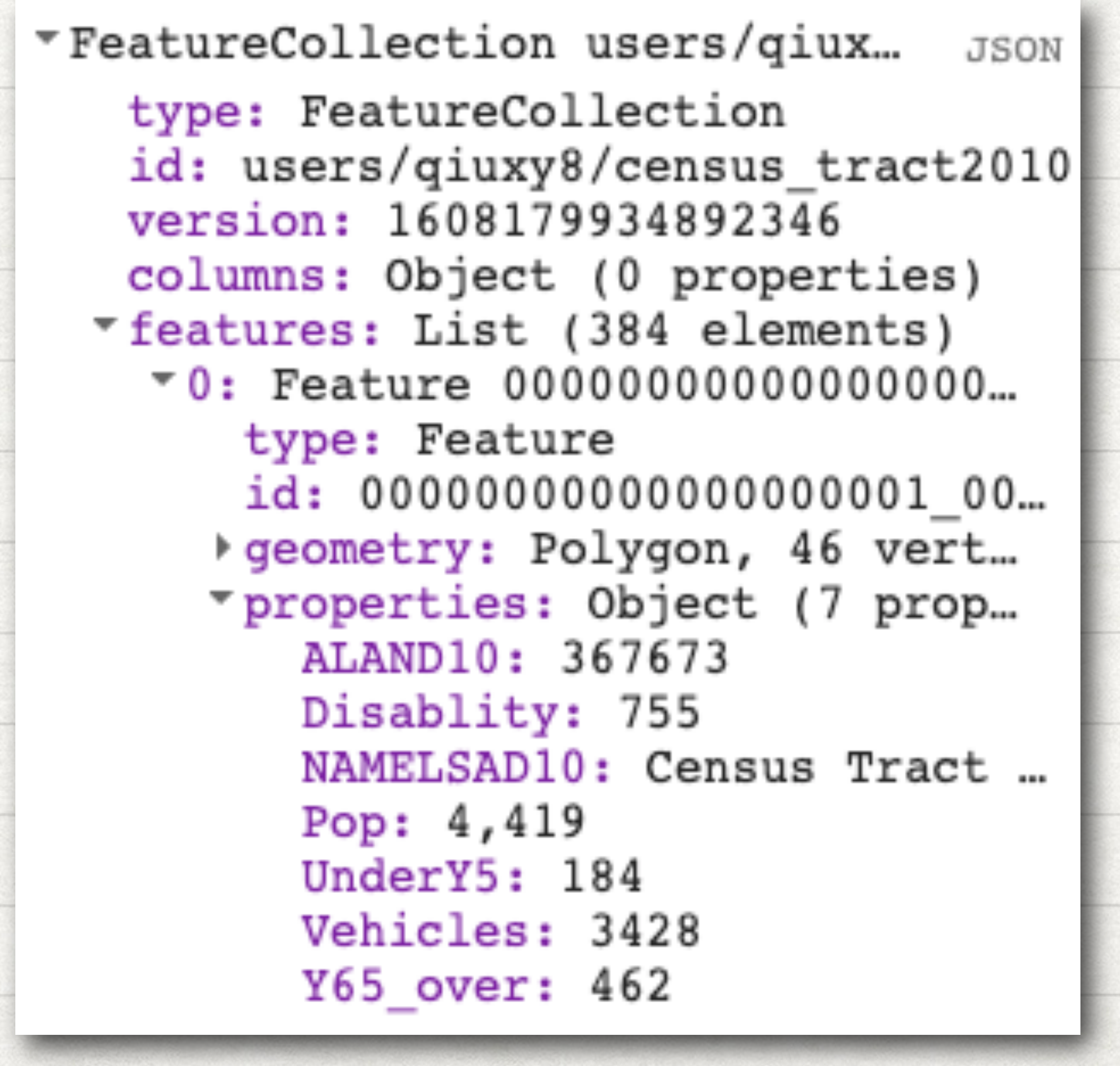
Credits to: Ian Schwarzenberg's "Congestion and Crash Severity along the Philadelphia Vision Zero High Injury Network"

The census tract of Philadelphia Shapefile does not include the census data. Therefore, I collect the data for the further analysis from data.census.gov and import the .csv file into GEE, including the total population, population under 5 and over 65, disability population and the number of vehicles.

Use *ee.Join.inner()* to join the data I collected with the census tract feature collection which has geometry information. However, the join output divides the properties into primary and secondary features, which cannot be mapped. Hence, I create a function to select the properties I need from primary and secondary properties. And the new output has seven properties I extract from the inner join's output



```
▾FeatureCollection users/qiux…   JSON
   type: FeatureCollection
   id: users/qiuxy8/census_tract2010
   version: 1608179934892346
  ▾columns: Object (2 properties)
     primary: Feature
     secondary: Feature
  ▾features: List (384 elements)
    ▾0: Feature 00000000000000000…
       type: Feature
       id: 00000000000000000001_00…
       geometry: null
      ▾properties: Object (2 prop…
        ▾primary: Feature 0000000…
           type: Feature
           id: 00000000000000000001
          ▸geometry: Polygon, 46 …
          ▸properties: Object (14…
        ▾secondary: Feature 00000…
           type: Feature
           id: 00000000000000000289
          ▸geometry: MultiPoint, …
          ▸properties: Object (7 …
```



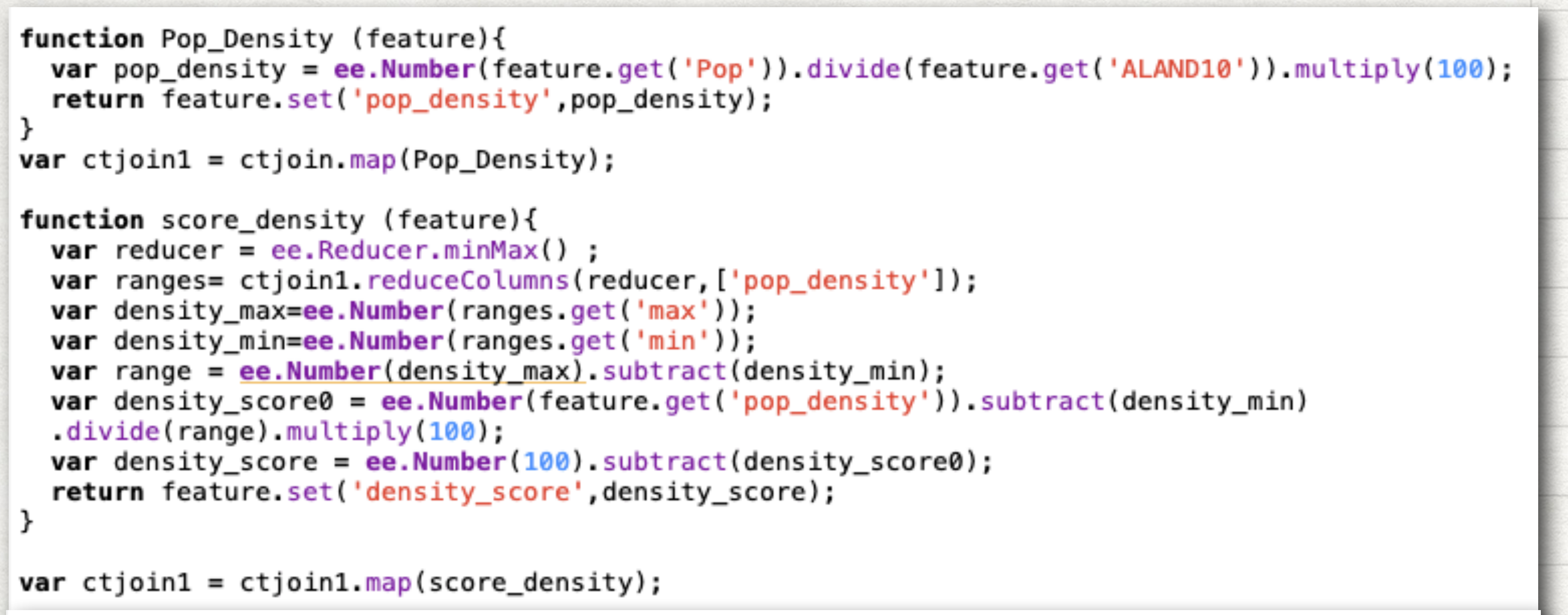
```
▾FeatureCollection users/qiux…   JSON
   type: FeatureCollection
   id: users/qiuxy8/census_tract2010
   version: 1608179934892346
   columns: Object (0 properties)
  ▾features: List (384 elements)
    ▾0: Feature 00000000000000000…
       type: Feature
       id: 00000000000000000001_00…
      ▸geometry: Polygon, 46 vert…
      ▾properties: Object (7 prop…
         ALAND10: 367673
         Disablity: 755
         NAMELSAD10: Census Tract …
         Pop: 4,419
         UnderY5: 184
         Vehicles: 3428
         Y65_over: 462
```

Join Output

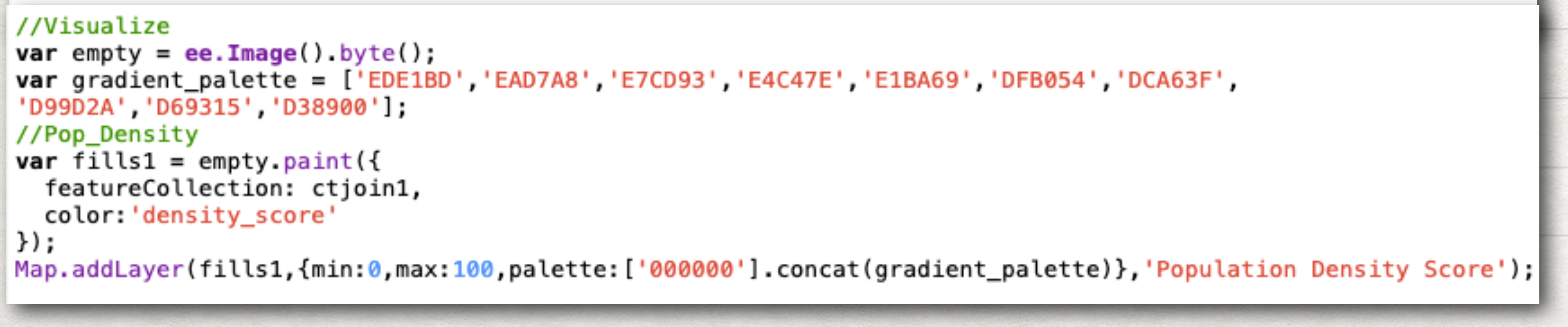
DEMOGRAPHIC CONDITIONS

POPULATION DENSITY

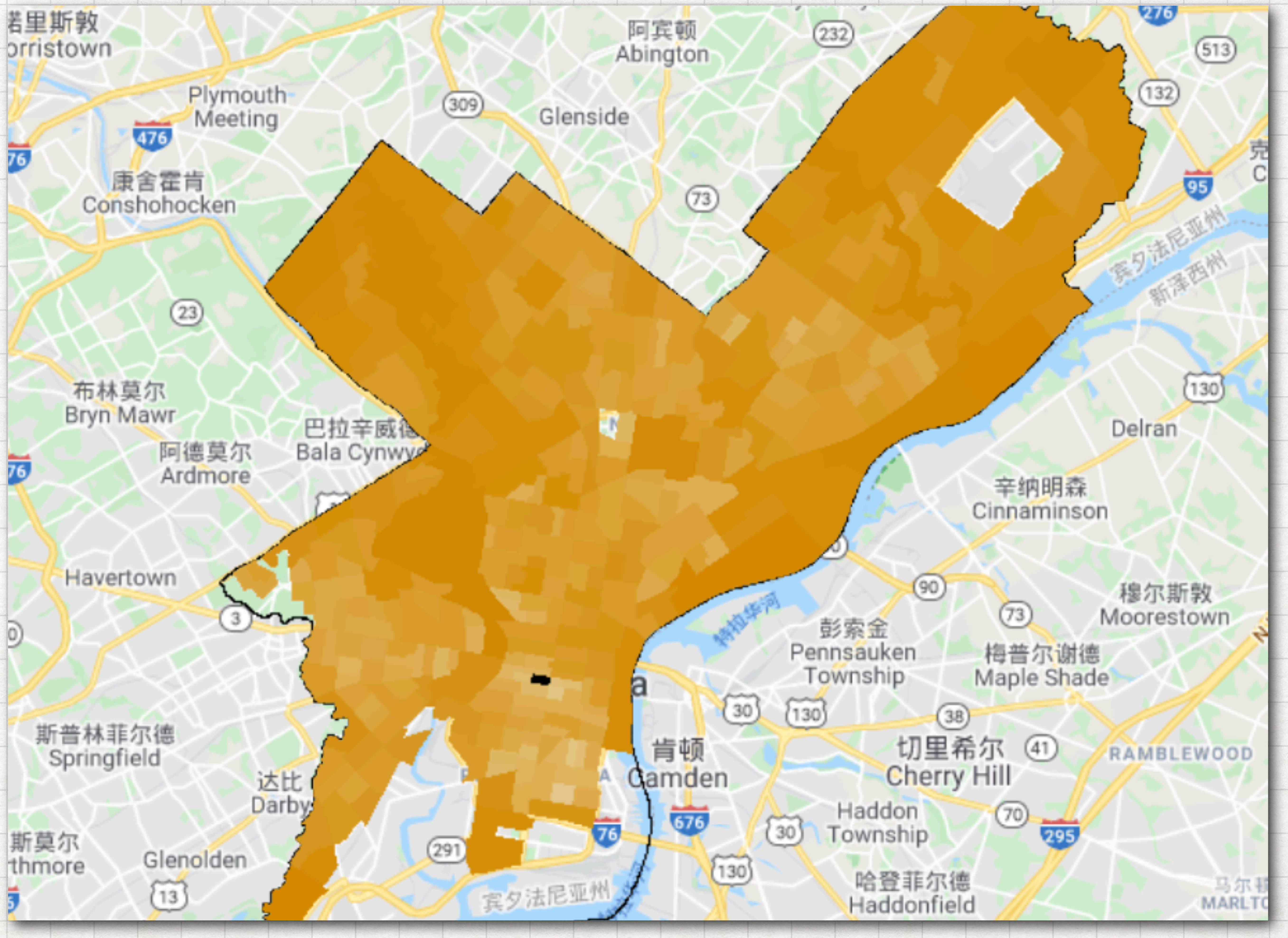
- Calculate the density of population and score the density from 0 to 100. 0 means the highest population density in Philadelphia, while 100 means the lowest population density in Philadelphia except for the census tracts with zero population



```
function Pop_Density (feature){
  var pop_density = ee.Number(feature.get('Pop')).divide(feature.get('ALAND10')).multiply(100);
  return feature.set('pop_density',pop_density);
}
var ctjoin1 = ctjoin.map(Pop_Density);

function score_density (feature){
  var reducer = ee.Reducer.minMax() ;
  var ranges= ctjoin1.reduceColumns(reducer,['pop_density']);
  var density_max=ee.Number(ranges.get('max'));
  var density_min=ee.Number(ranges.get('min'));
  var range = ee.Number(density_max).subtract(density_min);
  var density_score0 = ee.Number(feature.get('pop_density')).subtract(density_min)
  .divide(range).multiply(100);
  var density_score = ee.Number(100).subtract(density_score0);
  return feature.set('density_score',density_score);
}

var ctjoin1 = ctjoin1.map(score_density);
```



```
//Visualize
var empty = ee.Image().byte();
var gradient_palette = ['EDE1BD','EAD7A8','E7CD93','E4C47E','E1BA69','DFB054','DCA63F',
'D99D2A','D69315','D38900'];
//Pop_Density
var fills1 = empty.paint({
  featureCollection: ctjoin1,
  color:'density_score'
});
Map.addLayer(fills1,{min:0,max:100,palette:['000000'].concat(gradient_palette)},'Population Density Score');
```

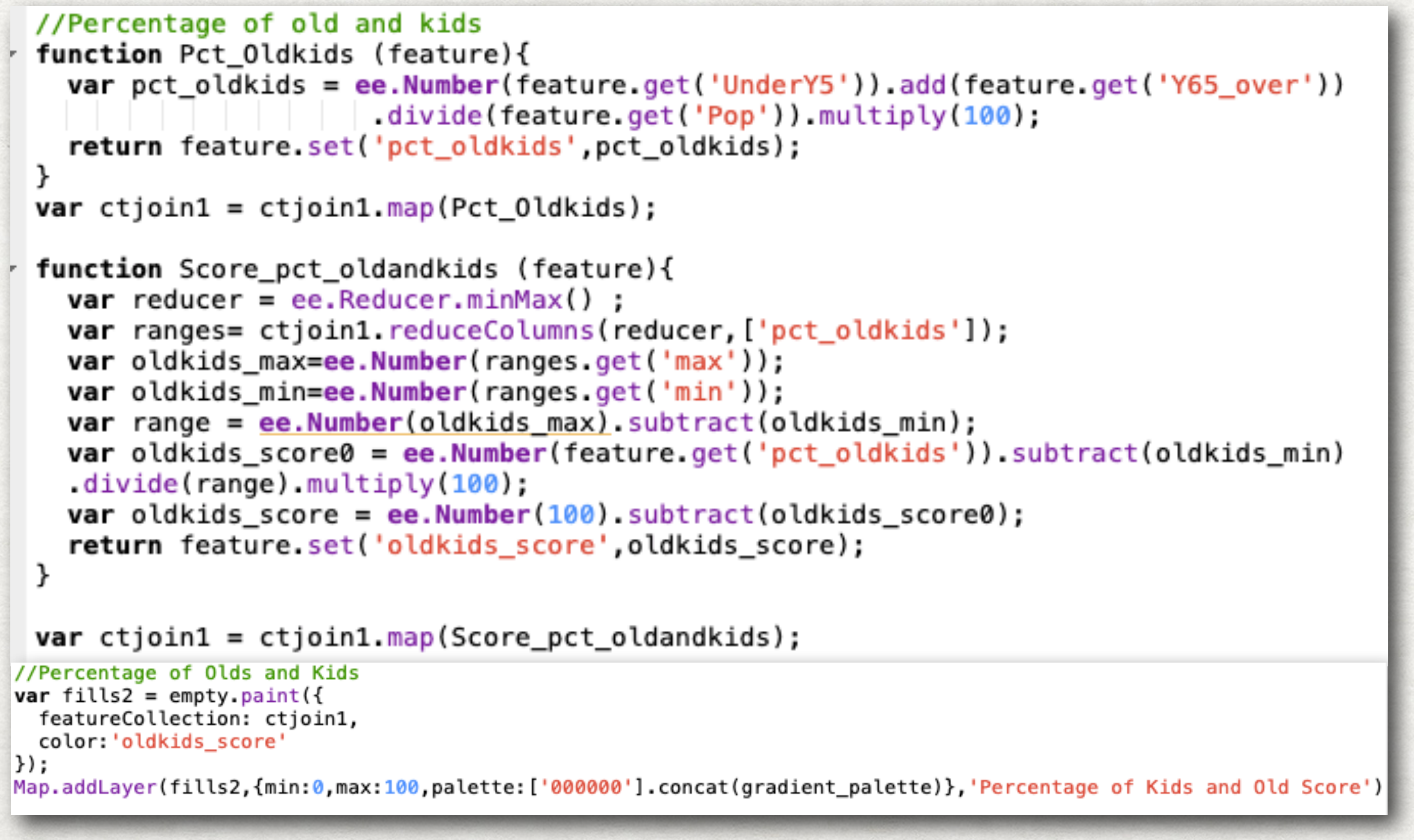


Population Density Score
(Blank means no data, Black represents 0)

DEMOGRAPHIC CONDITIONS

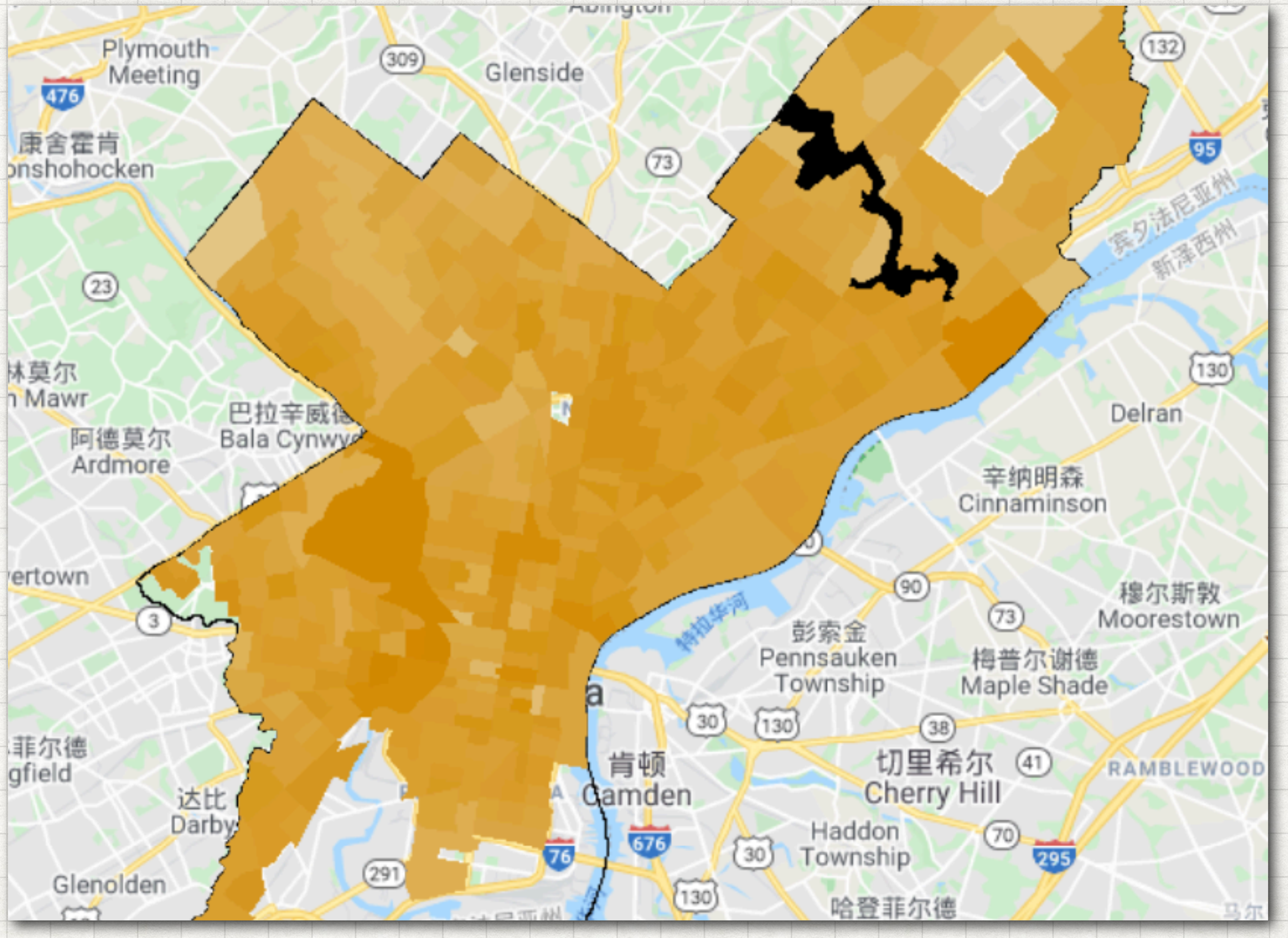
PERCENTAGE OF OLD AND KIDS

- Calculate the percentage of kids under 5 and the elderly over 65, and score the percentage from 0 to 100. 0 means the highest concentration of kids and the elderly in Philadelphia, while 100 means the opposite.



```
//Percentage of old and kids
function Pct_Oldkids (feature){
  var pct_oldkids = ee.Number(feature.get('UnderY5')).add(feature.get('Y65_over'))
                             .divide(feature.get('Pop')).multiply(100);
  return feature.set('pct_oldkids',pct_oldkids);
}
var ctjoin1 = ctjoin1.map(Pct_Oldkids);

function Score_pct_oldandkids (feature){
  var reducer = ee.Reducer.minMax() ;
  var ranges= ctjoin1.reduceColumns(reducer,['pct_oldkids']);
  var oldkids_max=ee.Number(ranges.get('max'));
  var oldkids_min=ee.Number(ranges.get('min'));
  var range = ee.Number(oldkids_max).subtract(oldkids_min);
  var oldkids_score0 = ee.Number(feature.get('pct_oldkids')).subtract(oldkids_min)
  .divide(range).multiply(100);
  var oldkids_score = ee.Number(100).subtract(oldkids_score0);
  return feature.set('oldkids_score',oldkids_score);
}

var ctjoin1 = ctjoin1.map(Score_pct_oldandkids);
```

```
//Percentage of Olds and Kids
var fills2 = empty.paint({
  featureCollection: ctjoin1,
  color:'oldkids_score'
});
Map.addLayer(fills2,{min:0,max:100,palette:['000000'].concat(gradient_palette)},'Percentage of Kids and Old Score')
```

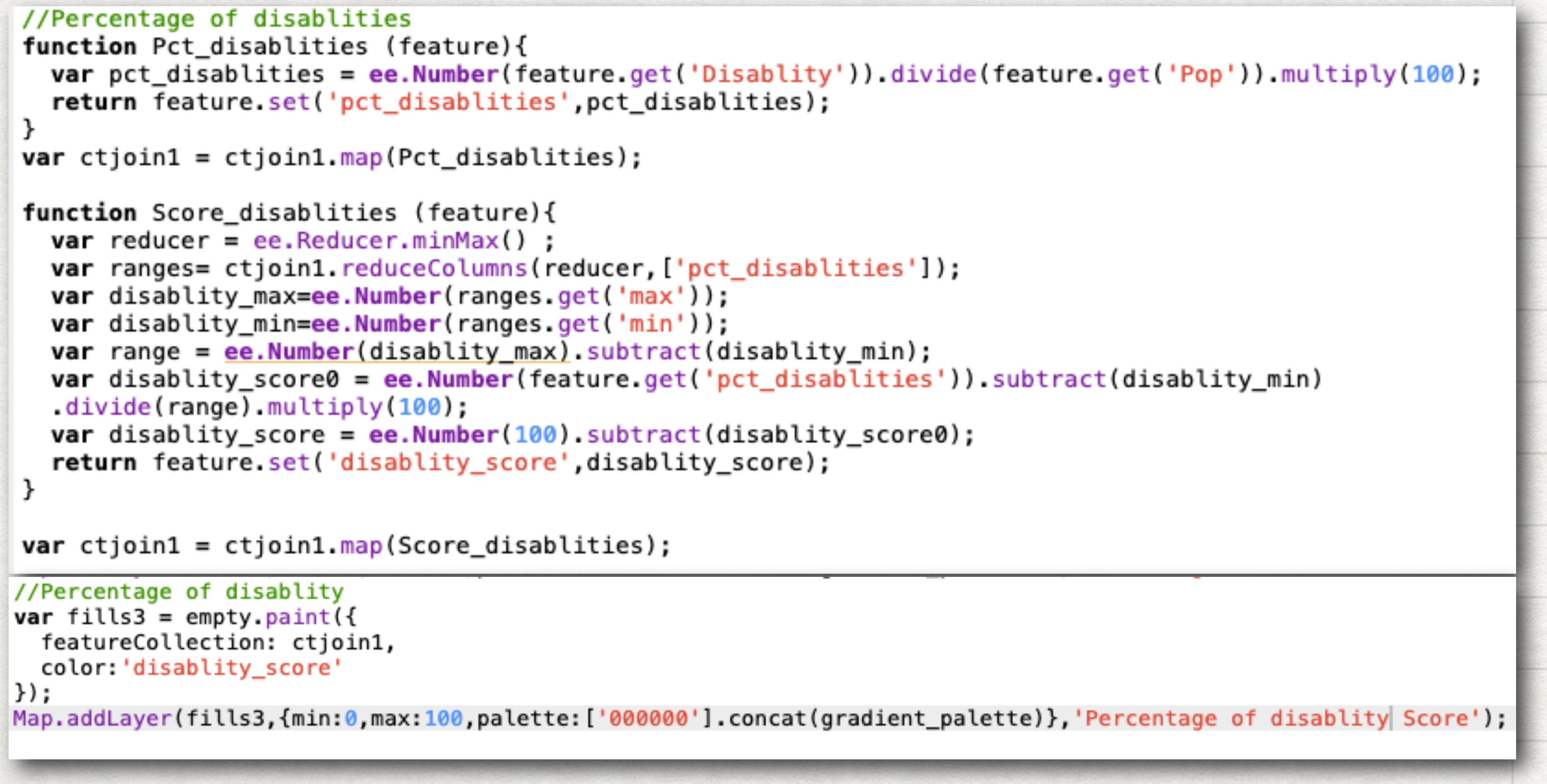


Percentage of Kids under 5 and elderly over 65 score
(Blank means no data, Black represents 0)

DEMOGRAPHIC CONDITIONS

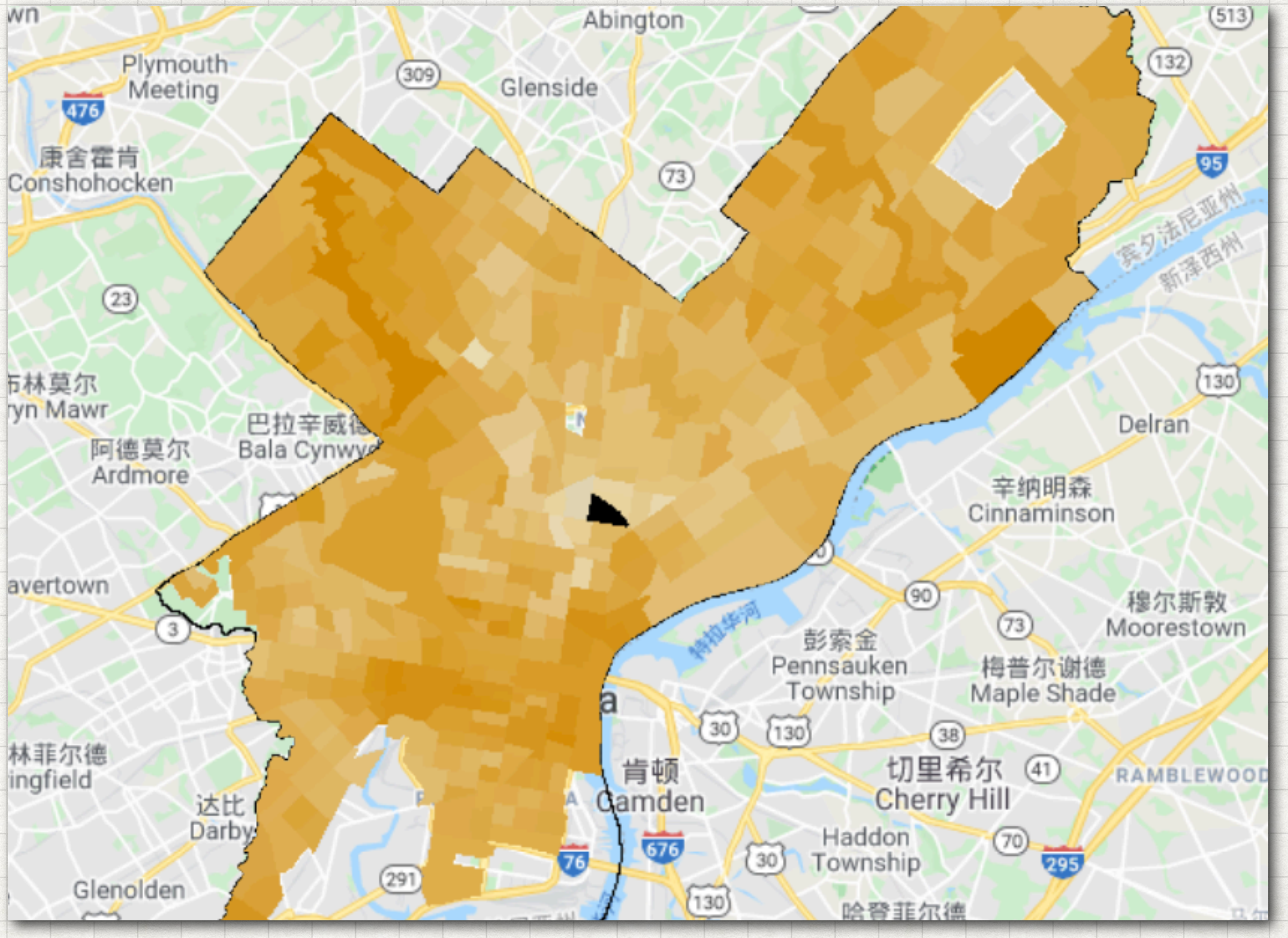
PERCENTAGE OF DISABILITIES

- Calculate the percentage of disabilities, and score the percentage from 0 to 100. 0 means the highest concentration of disabilities in Philadelphia, while 100 means the opposite.



```
//Percentage of disablities
function Pct_disablities (feature){
  var pct_disablities = ee.Number(feature.get('Disablity')).divide(feature.get('Pop')).multiply(100);
  return feature.set('pct_disablities',pct_disablities);
}
var ctjoin1 = ctjoin1.map(Pct_disablities);

function Score_disablities (feature){
  var reducer = ee.Reducer.minMax() ;
  var ranges= ctjoin1.reduceColumns(reducer,['pct_disablities']);
  var disablity_max=ee.Number(ranges.get('max'));
  var disablity_min=ee.Number(ranges.get('min'));
  var range = ee.Number(disablity_max).subtract(disablity_min);
  var disablity_score0 = ee.Number(feature.get('pct_disablities')).subtract(disablity_min)
  .divide(range).multiply(100);
  var disablity_score = ee.Number(100).subtract(disablity_score0);
  return feature.set('disablity_score',disablity_score);
}

var ctjoin1 = ctjoin1.map(Score_disablities);
```

```
//Percentage of disablity
var fills3 = empty.paint({
  featureCollection: ctjoin1,
  color:'disablity_score'
});
Map.addLayer(fills3,{min:0,max:100,palette:['000000'].concat(gradient_palette)},'Percentage of disablity Score');
```

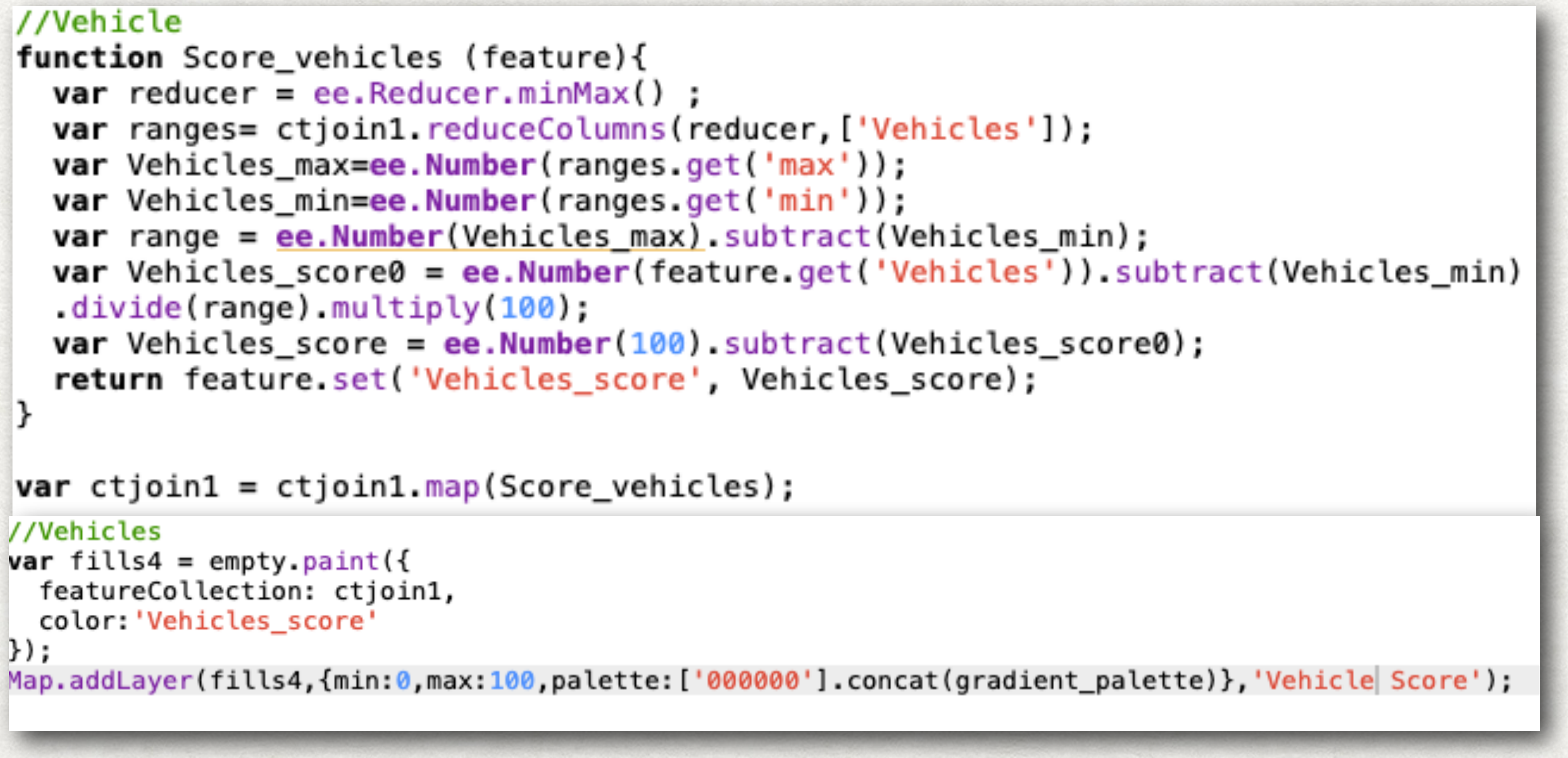


Percentage of disability score
(Blank means no data, Black represents 0)

DEMOGRAPHIC CONDITIONS

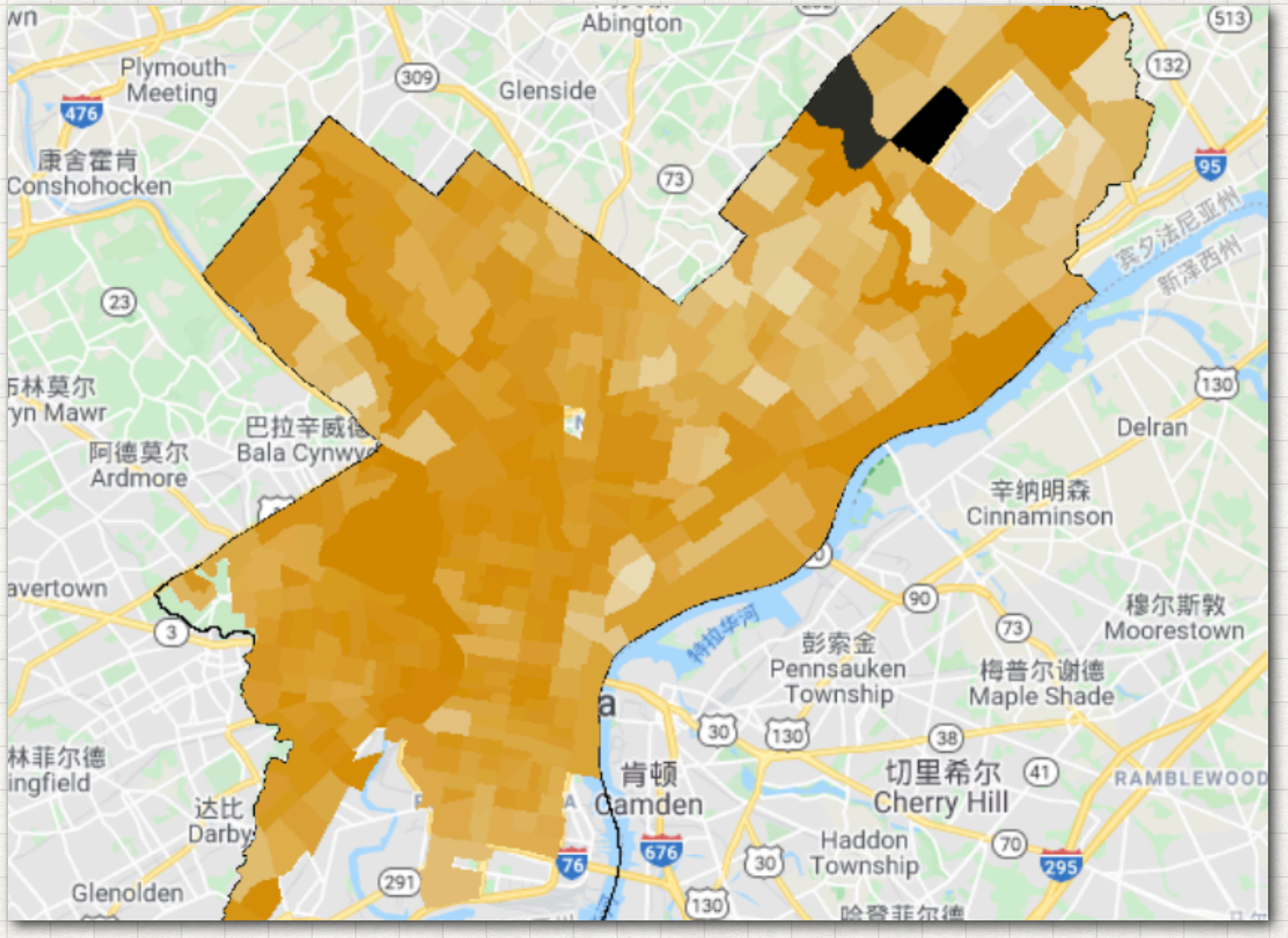
NUMBER OF VEHICLES

- Score the number of vehicles from 0 to 100. 0 means the census tract has the most vehicles in Philadelphia, while 100 means the opposite.



```
//Vehicle
function Score_vehicles (feature){
  var reducer = ee.Reducer.minMax() ;
  var ranges= ctjoin1.reduceColumns(reducer,['Vehicles']);
  var Vehicles_max=ee.Number(ranges.get('max'));
  var Vehicles_min=ee.Number(ranges.get('min'));
  var range = ee.Number(Vehicles_max).subtract(Vehicles_min);
  var Vehicles_score0 = ee.Number(feature.get('Vehicles')).subtract(Vehicles_min)
  .divide(range).multiply(100);
  var Vehicles_score = ee.Number(100).subtract(Vehicles_score0);
  return feature.set('Vehicles_score', Vehicles_score);
}

var ctjoin1 = ctjoin1.map(Score_vehicles);
```

```
//Vehicles
var fills4 = empty.paint({
  featureCollection: ctjoin1,
  color:'Vehicles_score'
});
Map.addLayer(fills4,{min:0,max:100,palette:['000000'].concat(gradient_palette)},'Vehicle Score');
```

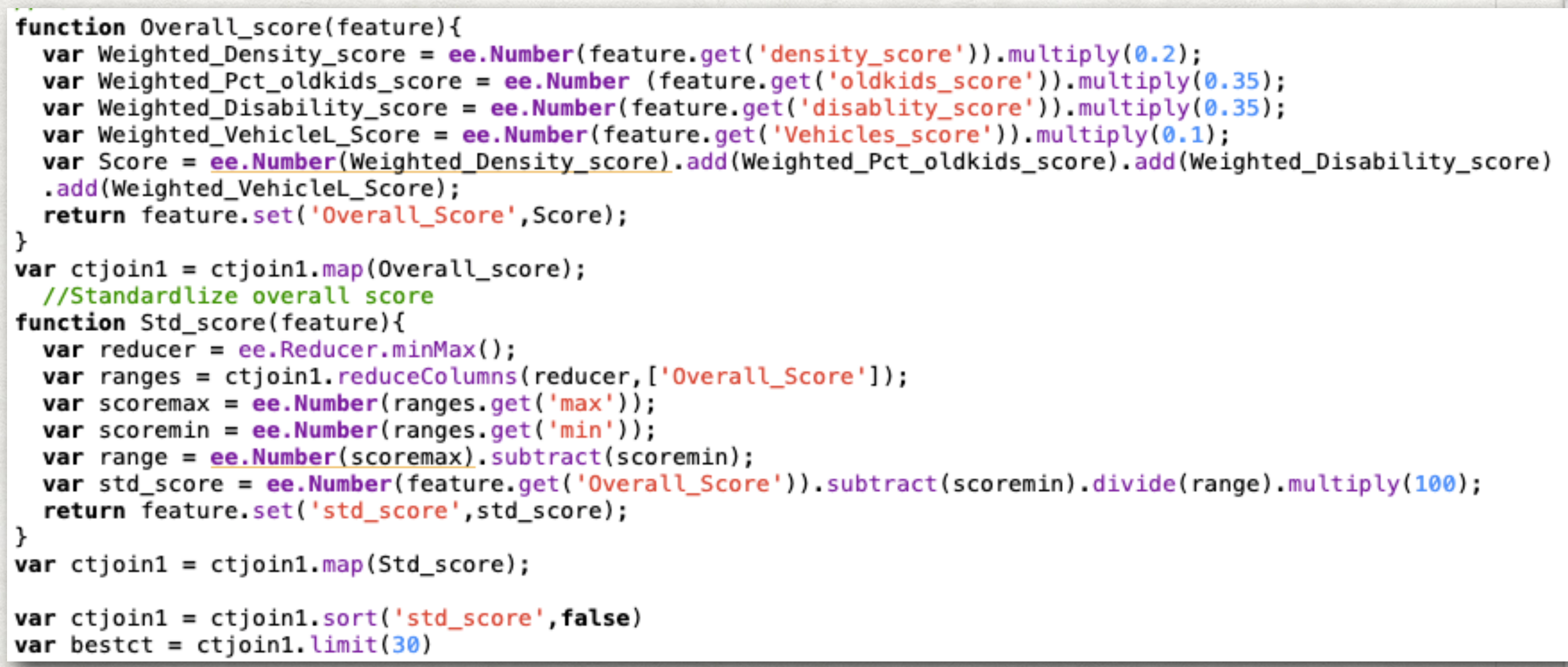


Number of Vehicle score
(Blank means no data, Black represents 0)

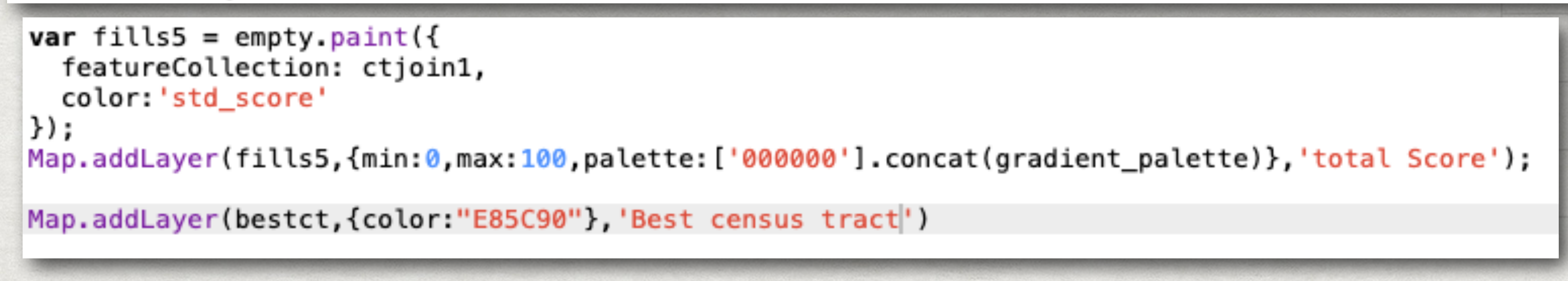
DEMOGRAPHIC CONDITIONS

TOTAL SCORE

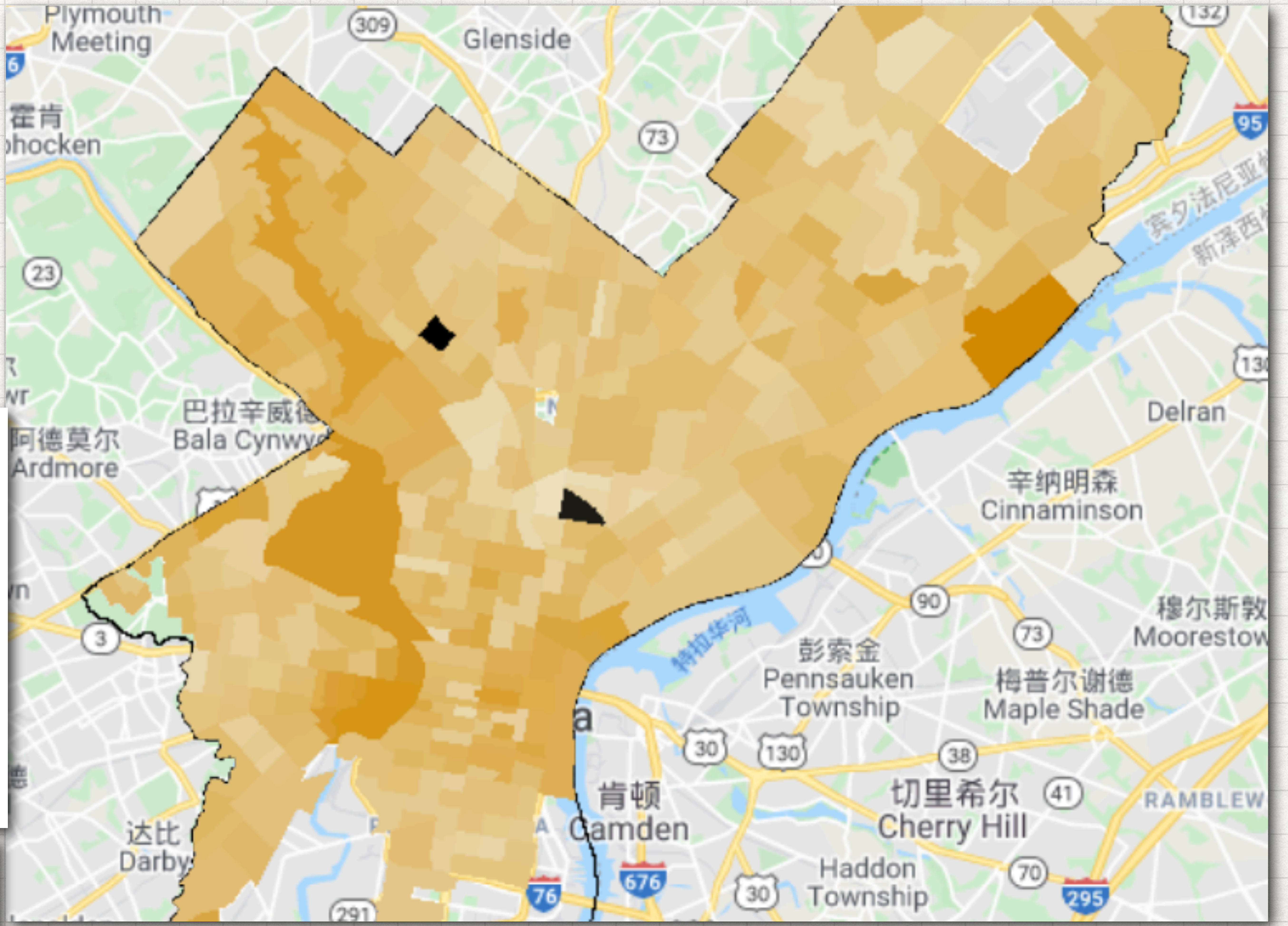
- Calculate the total score of each census tracts by adding the weighed scores of factors. After standardize the total score, choose 30 census tracts from the highest score to lowest score, which are the suitable places for road tests of AVs



```
function Overall_score(feature){
  var Weighted_Density_score = ee.Number(feature.get('density_score')).multiply(0.2);
  var Weighted_Pct_oldkids_score = ee.Number (feature.get('oldkids_score')).multiply(0.35);
  var Weighted_Disability_score = ee.Number(feature.get('disablity_score')).multiply(0.35);
  var Weighted_VehicleL_Score = ee.Number(feature.get('Vehicles_score')).multiply(0.1);
  var Score = ee.Number(Weighted_Density_score).add(Weighted_Pct_oldkids_score).add(Weighted_Disability_score)
  .add(Weighted_VehicleL_Score);
  return feature.set('Overall_Score',Score);
}
var ctjoin1 = ctjoin1.map(Overall_score);
  //Standardlize overall score
function Std_score(feature){
  var reducer = ee.Reducer.minMax();
  var ranges = ctjoin1.reduceColumns(reducer,['Overall_Score']);
  var scoremax = ee.Number(ranges.get('max'));
  var scoremin = ee.Number(ranges.get('min'));
  var range = ee.Number(scoremax).subtract(scoremin);
  var std_score = ee.Number(feature.get('Overall_Score')).subtract(scoremin).divide(range).multiply(100);
  return feature.set('std_score',std_score);
}
var ctjoin1 = ctjoin1.map(Std_score);

var ctjoin1 = ctjoin1.sort('std_score',false)
var bestct = ctjoin1.limit(30)
```



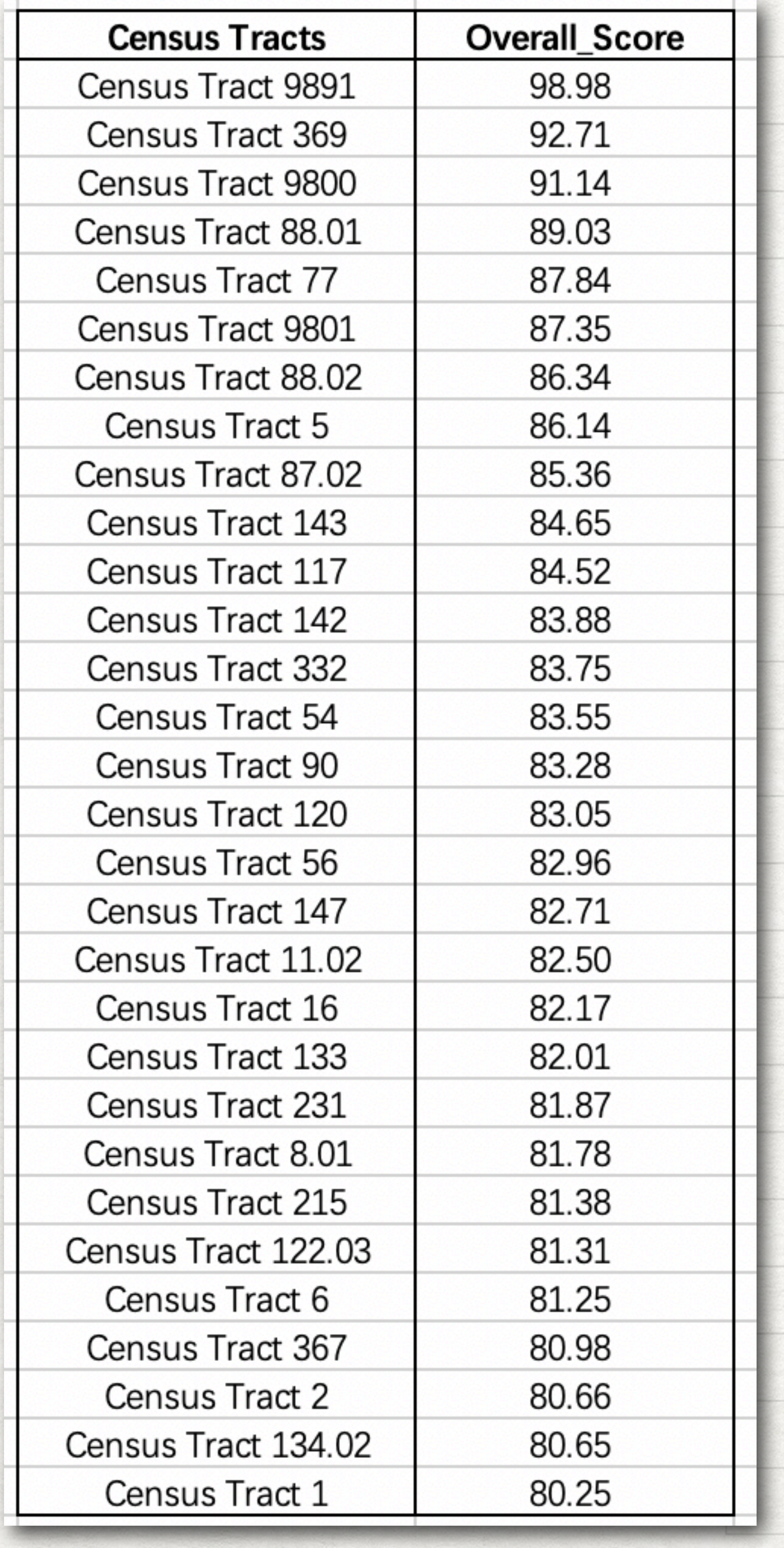
```
var fills5 = empty.paint({
  featureCollection: ctjoin1,
  color:'std_score'
});
Map.addLayer(fills5,{min:0,max:100,palette:['000000'].concat(gradient_palette)},'total Score');

Map.addLayer(bestct,{color:"E85C90"},'Best census tract')
```



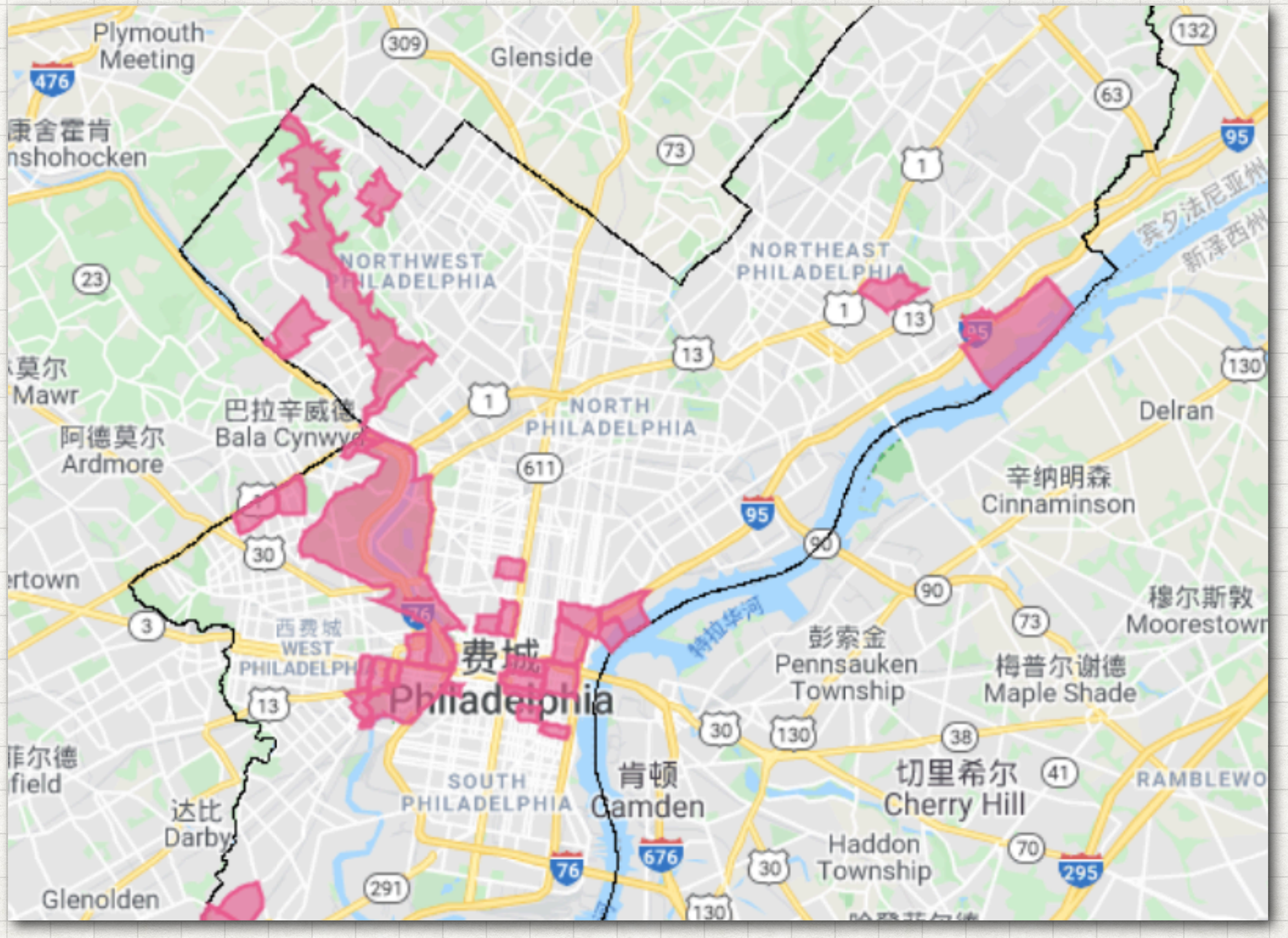
CONCLUSIONS

BEST CENSUS TRACTS

Suitable Census tracts for road tests concentrate in **Center City, University City, Fairmount Park, China Town, Spring Garden and Northern Liberties**



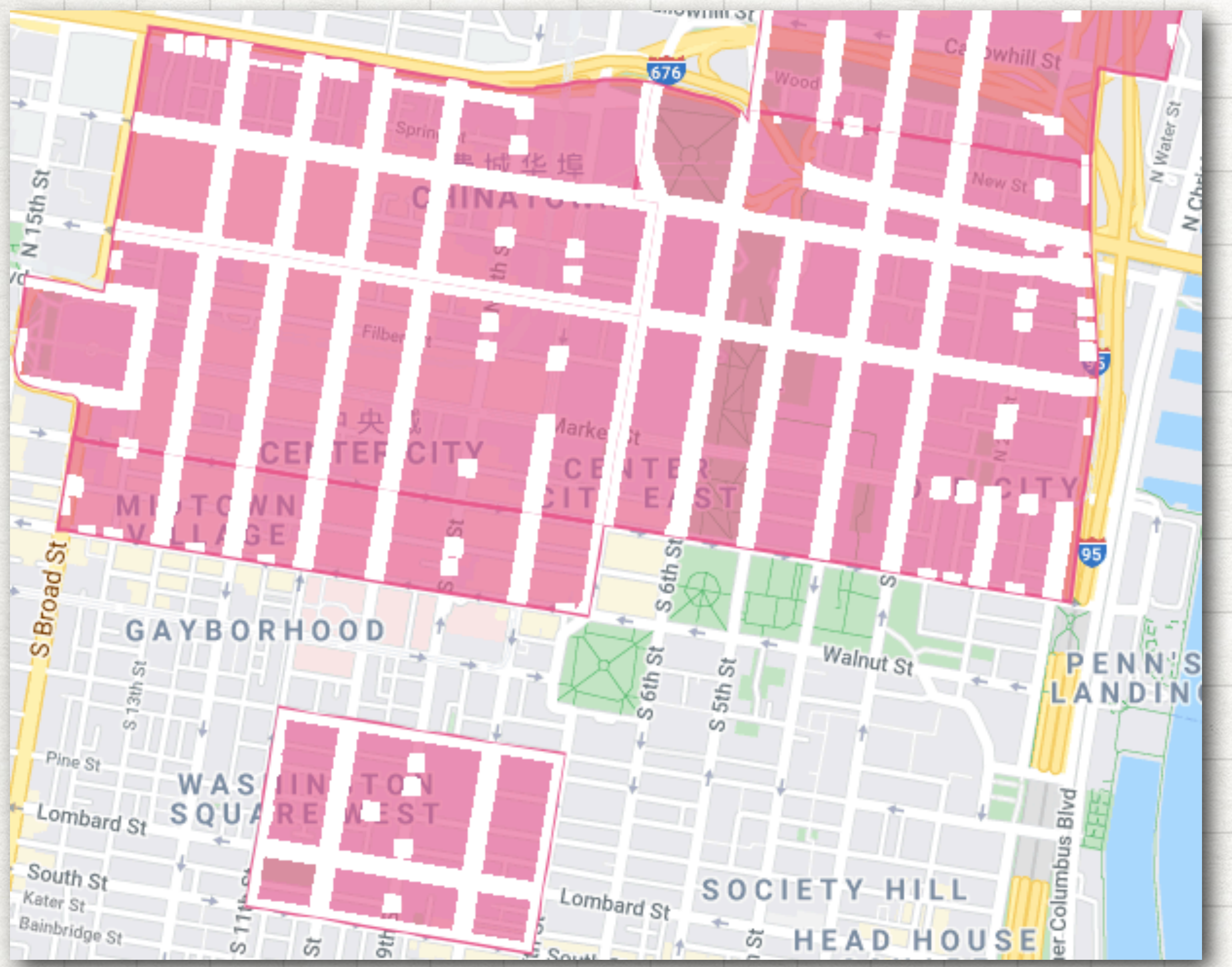
Census Tracts	Overall_Score
Census Tract 9891	98.98
Census Tract 369	92.71
Census Tract 9800	91.14
Census Tract 88.01	89.03
Census Tract 77	87.84
Census Tract 9801	87.35
Census Tract 88.02	86.34
Census Tract 5	86.14
Census Tract 87.02	85.36
Census Tract 143	84.65
Census Tract 117	84.52
Census Tract 142	83.88
Census Tract 332	83.75
Census Tract 54	83.55
Census Tract 90	83.28
Census Tract 120	83.05
Census Tract 56	82.96
Census Tract 147	82.71
Census Tract 11.02	82.50
Census Tract 16	82.17
Census Tract 133	82.01
Census Tract 231	81.87
Census Tract 8.01	81.78
Census Tract 215	81.38
Census Tract 122.03	81.31
Census Tract 6	81.25
Census Tract 367	80.98
Census Tract 2	80.66
Census Tract 134.02	80.65
Census Tract 1	80.25



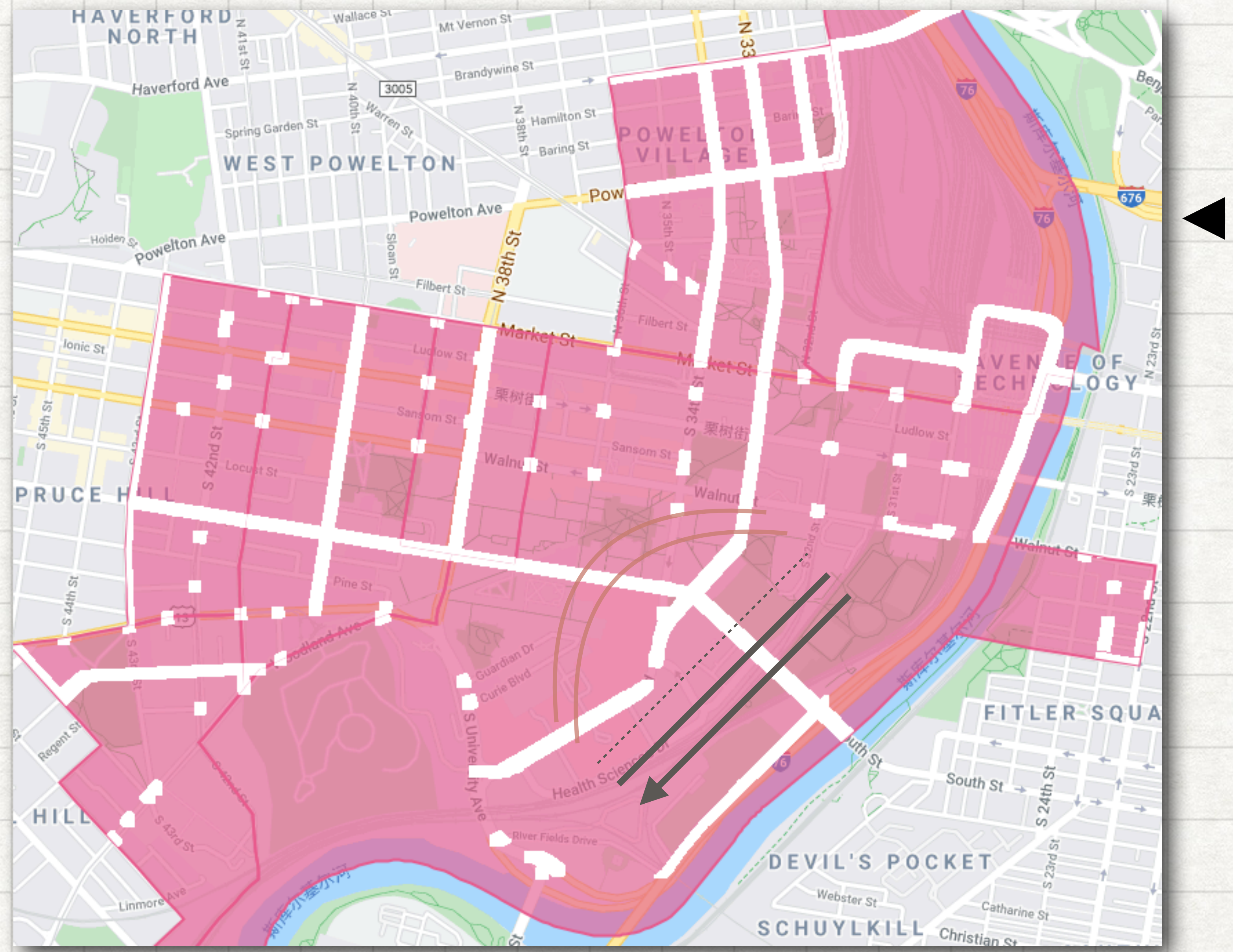
Suitable Census tracts for Avs

CONCLUSIONS

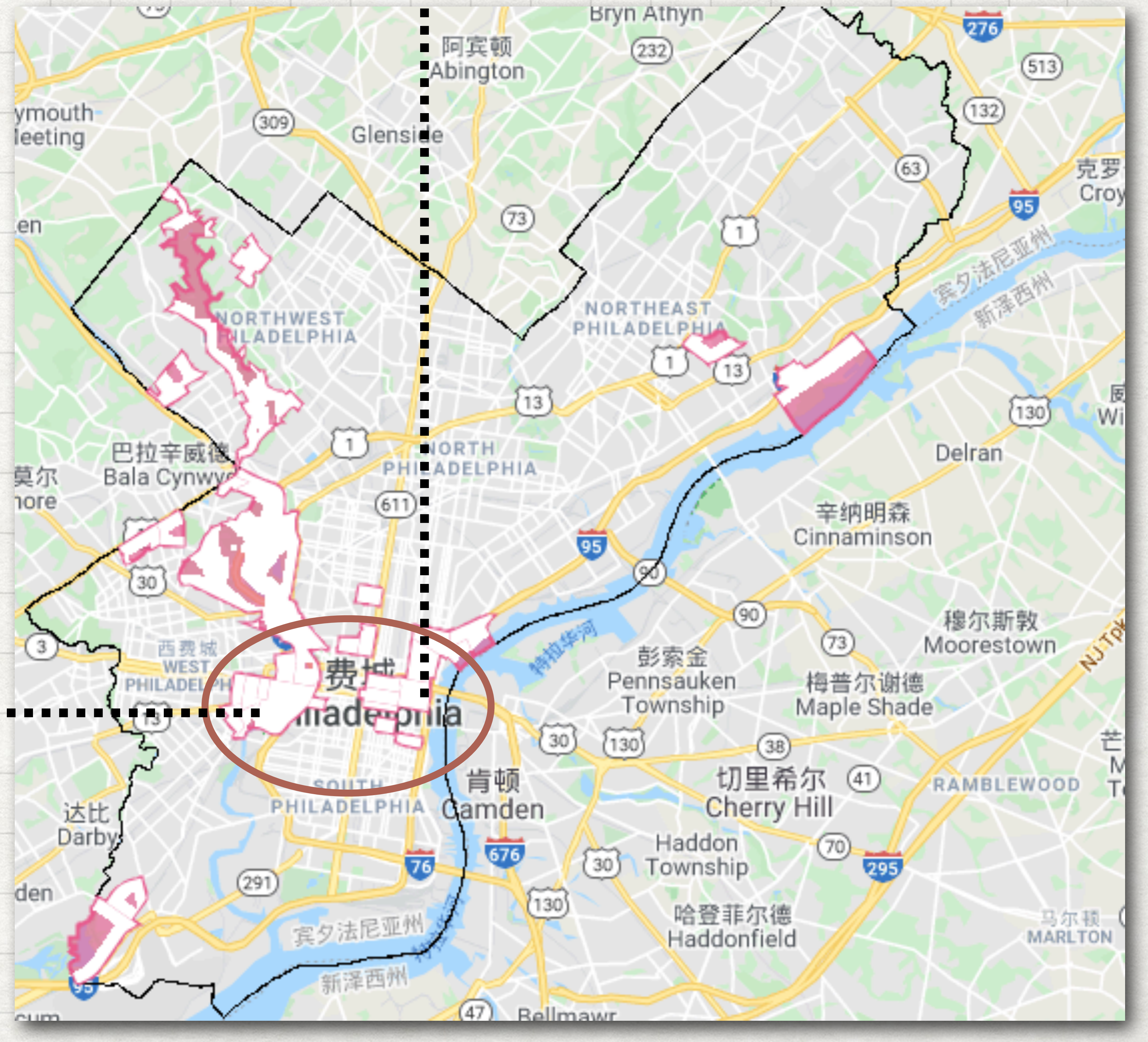
BEST ROADS



SUITABLE STREETS IN CENTER CITY: N 3TH STREET TO N 13TH STREET, RACE STREET, ARCH STREET, BENJAMIN FRANKLIN BRIDGE AND SPRING GARDEN STREET



SUITABLE STREETS IN UNIVERSITY CITY: N 33TH, 34TH, 38TH, 40TH AND 43TH STREET, POWELTON AVE, SPRUCE STREETAND SPRING GARDEN STREET



GEE CODE SCRIPT

https://code.earthengine.google.com/?scriptPath=users%2Fqiuxy8%2FAssignment%3Aterm%20project

```
// Import Dataset
var philly = ee.FeatureCollection('users/qiuxy8/City_limits');
var ct = ee.FeatureCollection('users/qiuxy8/census_tract2010');
var street = ee.FeatureCollection('users/qiuxy8/Street_Centerline');
var volume = ee.FeatureCollection('users/qiuxy8/Traffic_Volumes');
var high_injury_network = ee.FeatureCollection('users/qiuxy8/
high_injury_network_2017');
var elevation = ee.Image("CGIAR/
SRTM90_V4").select('elevation').clip(philly);
var cttable = ee.FeatureCollection("users/qiuxy8/census_tract1");
var empty = ee.Image().byte();
var Philly_Boundary_Transparent = empty.paint({
featureCollection: philly,
color: 1,
width: 1.25});
var CT_Transparent = empty.paint({
featureCollection: ct,
color: 1,
width: 0.75});
Map.addLayer(Philly_Boundary_Transparent,{color:'000000'},'Philly');
Map.centerObject(philly,11.5);
Map.addLayer(CT_Transparent,null,'census tract');
Map.addLayer(street,{color:'5c87ab'},'street');
Map.addLayer(high_injury_network,{color:'ff0000'},'High injury network');
```

```
//High Risk network
var theFilter = ee.Filter.equals('STNAME', null, 'stname', null );
var theJoin = ee.Join.inverted();
var non_riskroad = theJoin.apply(street, high_injury_network,theFilter);
var addField = function(feature){
  var nonrisk = ee.Number(1);
  return feature.set({'nonrisk':nonrisk});
};
var non_riskroad = non_riskroad.map(addField);
Map.addLayer(non_riskroad,{color:'white'},'non risk road');
```

GEE CODE SCRIPT

```
// Road Conditions
//Road Class
   /*Street class of arc:
0-Navy Yard
1-Expressways
2-Major Arterial
3-Minor Arterial
4-Collector
5-Local
6-Driveway
9-Low Speed Ramps
10-High Speed Ramps
12-Non Travable
14-City Boundary
15-Walking Connector*/
var street2 =
non_riskroad.filterMetadata('CLASS','not_equals',12).filterMetadata('CLAS
S','not_equals',14)
                    .filterMetadata('CLASS','not_equals',15);
var streetclassImage = street2.filter(ee.Filter.notNull(['CLASS']))
                                    .reduceToImage({
                                      properties:['CLASS'],
                                      reducer:ee.Reducer.max()
                                    });
print(streetclassImage);
var streetrisk = streetclassImage.remap([1,2,3,4,5,6,9,10,13,18],
[1,5,6,7,8,4,3,2,1,1],0,'max');
```

```
var streetrisk_broad = streetrisk.focal_max(3,'square','pixels');
Map.addLayer(streetrisk_broad,{min:1,max:8,palette:
['c2000c','f2000f','ff2f3c','ff7b83','ffdadc']},'road class risk ');
var safestreet = streetrisk_broad.gt(4);
var safestreet=streetrisk_broad.mask(safestreet).clip(philly);
Map.addLayer(safestreet,{min:1,max:8,palette:
['c2000c','f2000f','ff2f3c','ff7b83','ffdadc']},'safestreet ');
print(safestreet);
```

GEE CODE SCRIPT

```
//Traffic Volume
var volumeImage = volume.filter(ee.Filter.notNull(['CUR_AADT']))
                                  .reduceToImage({
                                    properties:['CUR_AADT'],
                                    reducer:ee.Reducer.max()
                                  });
var volume2 = volumeImage.focal_max(3,'square','pixels')     ;
var volumeImage = volume2.clip(philly)  ;
var volumemax=
volumeImage.reduceRegion(ee.Reducer.max(),philly,500,null,null,true);
var volumemin =
volumeImage.reduceRegion(ee.Reducer.min(),philly,500,null,null,true);
var reducerper = ee.Reducer.percentile([0,20,40,60,80]);
var volumeper =
volumeImage.reduceRegion(reducerper,philly,500,null,null,true);
print(volumemax);
print(volumemin);
print(volumeper);
var bestvolume = volumeImage.lte(102548);
var bestvolume = volumeImage.mask(bestvolume);
Map.addLayer(bestvolume,{min:6807,max:102548,palette:
['038d05','117401','005813','00400e','0c3300']},'suitable volume');
```

```
// Suitable road conditions
var Road = safestreet.and(bestvolume);
var Road = Road.focal_max(1,'square','pixels').clip(philly);
Map.addLayer (Road,{palette:'509f36',opacity:0.7}, 'good road conditions');
```

```
// Geographical
//Slope
var slopephilly = ee.Terrain.slope(elevation)
var slopemax=
slopephilly.reduceRegion(ee.Reducer.max(),philly,500,null,null,true);
var slopemin =
slopephilly.reduceRegion(ee.Reducer.min(),philly,500,null,null,true);
var slopeper =
slopephilly.reduceRegion(reducerper,philly,500,null,null,true);
print(slopemax)
print(slopemin)
print(slopeper)
var bestslope = slopephilly.lte(1.2012324390842575)
var bestslope = slopephilly.mask(bestslope)
Map.addLayer(bestslope,{min:0,max:1.3,palette:['000099','dddd00']},'Best
slope')
var Road2 =Road.and(bestslope)
var Road2 = Road2.focal_max(1,'square','pixels').clip(philly);
Map.addLayer (Road2,{palette:'509f36',opacity:0.7}, 'good road
conditions2');
```

GEE CODE SCRIPT

```
//Demographic
print(ct);
print(cttable);
var theFilter2 = ee.Filter.equals('NAMELSAD10', null, 'CTName', null );
var theJoin2 = ee.Join.inner();
var ctjoin = theJoin2.apply(ct,cttable,theFilter2);
print(ctjoin);
var ctjoin = ctjoin.map(function(element){
  var Primary = ee.Feature(element.get('primary'));
  var Secondary = ee.Feature(element.get('secondary'));
  var ALAND10 = Primary.get('ALAND10');
  var NAMELSAD10 = Primary.get('NAMELSAD10');
  var Pop = Secondary.get('Total_Pop');
  var UnderY5 = Secondary.get('UnderY5');
  var Y65_over = Secondary.get('Y65_over');
  var Disablity = Secondary.get('Disablity');
  var Vehicles = Secondary.get('Vehicles');
  var geom = ee.Feature(element.get('primary')).geometry();
  return ee.Feature(geom, {'ALAND10':ALAND10,
'NAMELSAD10':NAMELSAD10,
'Pop':Pop, 'UnderY5':UnderY5, 'Y65_over':Y65_over, 'Disablity':Disablity,
'Vehicles':Vehicles});
});
print(ctjoin);
```

```
//Clean the data
var ctjoin =
ctjoin.filterMetadata('ALAND10','greater_than',0).filterMetadata('Pop','greater_than',0)
                    .filterMetadata('UnderY5','not_less_than',0).filterMetadata('Y65_over','not_less_than',0)
                    .filterMetadata('Disablity','not_less_than',0).filterMetadata('Vehicles','not_less_than',0)
```

```
//Population Density
function Pop_Density (feature){
  var pop_density =
ee.Number(feature.get('Pop')).divide(feature.get('ALAND10')).multiply(100);
  return feature.set('pop_density',pop_density);
}
var ctjoin1 = ctjoin.map(Pop_Density);
function score_density (feature){
  var reducer = ee.Reducer.minMax() ;
  var ranges= ctjoin1.reduceColumns(reducer,['pop_density']);
  var density_max=ee.Number(ranges.get('max'));
  var density_min=ee.Number(ranges.get('min'));
  var range = ee.Number(density_max).subtract(density_min);
  var density_score0 =
ee.Number(feature.get('pop_density')).subtract(density_min)
  .divide(range).multiply(100);
  var density_score = ee.Number(100).subtract(density_score0);
  return feature.set('density_score',density_score);
}
var ctjoin1 = ctjoin1.map(score_density);
```

GEE CODE SCRIPT

```
//Percentage of old and kids
function Pct_Oldkids (feature){
  var pct_oldkids =
ee.Number(feature.get('UnderY5')).add(feature.get('Y65_over'))
                     .divide(feature.get('Pop')).multiply(100);
  return feature.set('pct_oldkids',pct_oldkids);
}
var ctjoin1 = ctjoin1.map(Pct_Oldkids);

function Score_pct_oldandkids (feature){
  var reducer = ee.Reducer.minMax() ;
  var ranges= ctjoin1.reduceColumns(reducer,['pct_oldkids']);
  var oldkids_max=ee.Number(ranges.get('max'));
  var oldkids_min=ee.Number(ranges.get('min'));
  var range = ee.Number(oldkids_max).subtract(oldkids_min);
  var oldkids_score0 =
ee.Number(feature.get('pct_oldkids')).subtract(oldkids_min)
  .divide(range).multiply(100);
  var oldkids_score = ee.Number(100).subtract(oldkids_score0);
  return feature.set('oldkids_score',oldkids_score);
}

var ctjoin1 = ctjoin1.map(Score_pct_oldandkids);
```

```
//Percentage of disablities
function Pct_disablities (feature){
  var pct_disablities =
ee.Number(feature.get('Disablity')).divide(feature.get('Pop')).multiply(100);
  return feature.set('pct_disablities',pct_disablities);
}
var ctjoin1 = ctjoin1.map(Pct_disablities);

function Score_disablities (feature){
  var reducer = ee.Reducer.minMax() ;
  var ranges= ctjoin1.reduceColumns(reducer,['pct_disablities']);
  var disablity_max=ee.Number(ranges.get('max'));
  var disablity_min=ee.Number(ranges.get('min'));
  var range = ee.Number(disablity_max).subtract(disablity_min);
  var disablity_score0 =
ee.Number(feature.get('pct_disablities')).subtract(disablity_min)
  .divide(range).multiply(100);
  var disablity_score = ee.Number(100).subtract(disablity_score0);
  return feature.set('disablity_score',disablity_score);
}

var ctjoin1 = ctjoin1.map(Score_disablities);
```

GEE CODE SCRIPT

```
//Vehicle
function Score_vehicles (feature){
  var reducer = ee.Reducer.minMax() ;
  var ranges= ctjoin1.reduceColumns(reducer,['Vehicles']);
  var Vehicles_max=ee.Number(ranges.get('max'));
  var Vehicles_min=ee.Number(ranges.get('min'));
  var range = ee.Number(Vehicles_max).subtract(Vehicles_min);
  var Vehicles_score0 =
ee.Number(feature.get('Vehicles')).subtract(Vehicles_min)
  .divide(range).multiply(100);
  var Vehicles_score = ee.Number(100).subtract(Vehicles_score0);
  return feature.set('Vehicles_score', Vehicles_score);
}

var ctjoin1 = ctjoin1.map(Score_vehicles);
```

```
//Total
function Overall_score(feature){
  var Weighted_Density_score =
ee.Number(feature.get('density_score')).multiply(0.2);
  var Weighted_Pct_oldkids_score = ee.Number
(feature.get('oldkids_score')).multiply(0.35);
  var Weighted_Disability_score =
ee.Number(feature.get('disablity_score')).multiply(0.35);
  var Weighted_VehicleL_Score =
ee.Number(feature.get('Vehicles_score')).multiply(0.1);
  var Score =
ee.Number(Weighted_Density_score).add(Weighted_Pct_oldkids_score).add
(Weighted_Disability_score)
  .add(Weighted_VehicleL_Score);
  return feature.set('Overall_Score',Score);
}
var ctjoin1 = ctjoin1.map(Overall_score);
```

GEE CODE SCRIPT

```
//Standardlize overall score
function Std_score(feature){
  var reducer = ee.Reducer.minMax();
  var ranges = ctjoin1.reduceColumns(reducer,['Overall_Score']);
  var scoremax = ee.Number(ranges.get('max'));
  var scoremin = ee.Number(ranges.get('min'));
  var range = ee.Number(scoremax).subtract(scoremin);
  var std_score =
ee.Number(feature.get('Overall_Score')).subtract(scoremin).divide(range).mult
iply(100);
  return feature.set('std_score',std_score);
}
var ctjoin1 = ctjoin1.map(Std_score);
var ctjoin1 = ctjoin1.sort('std_score',false)
var bestct = ctjoin1.limit(30)
print(bestct)
```

```
//Visualize
var empty = ee.Image().byte();
var gradient_palette =
['EDE1BD','EAD7A8','E7CD93','E4C47E','E1BA69','DFB054','DCA63F',
'D99D2A','D69315','D38900'];
//Pop_Density
var fills1 = empty.paint({
  featureCollection: ctjoin1,
  color:'density_score'
});
Map.addLayer(fills1,{min:0,max:100,palette:
['000000'].concat(gradient_palette)},'Population Density Score');
```

```
//Percentage of Olds and Kids
var fills2 = empty.paint({
  featureCollection: ctjoin1,
  color:'oldkids_score'
});
Map.addLayer(fills2,{min:0,max:100,palette:
['000000'].concat(gradient_palette)},'Percentage of Kids and Old Score');
//Percentage of disablity
var fills3 = empty.paint({
  featureCollection: ctjoin1,
  color:'disablity_score'
});
Map.addLayer(fills3,{min:0,max:100,palette:
['000000'].concat(gradient_palette)},'Percentage of disablity Score');
//Vehicles
var fills4 = empty.paint({
  featureCollection: ctjoin1,
  color:'Vehicles_score'
});
Map.addLayer(fills4,{min:0,max:100,palette:
['000000'].concat(gradient_palette)},'Vehicle Score');
//Total and Best

var fills5 = empty.paint({
  featureCollection: ctjoin1,
  color:'std_score'
});
Map.addLayer(fills5,{min:0,max:100,palette:
['000000'].concat(gradient_palette)},'total Score');
Map.addLayer(bestct,{color:"E85C90"},'Best census tract');
```