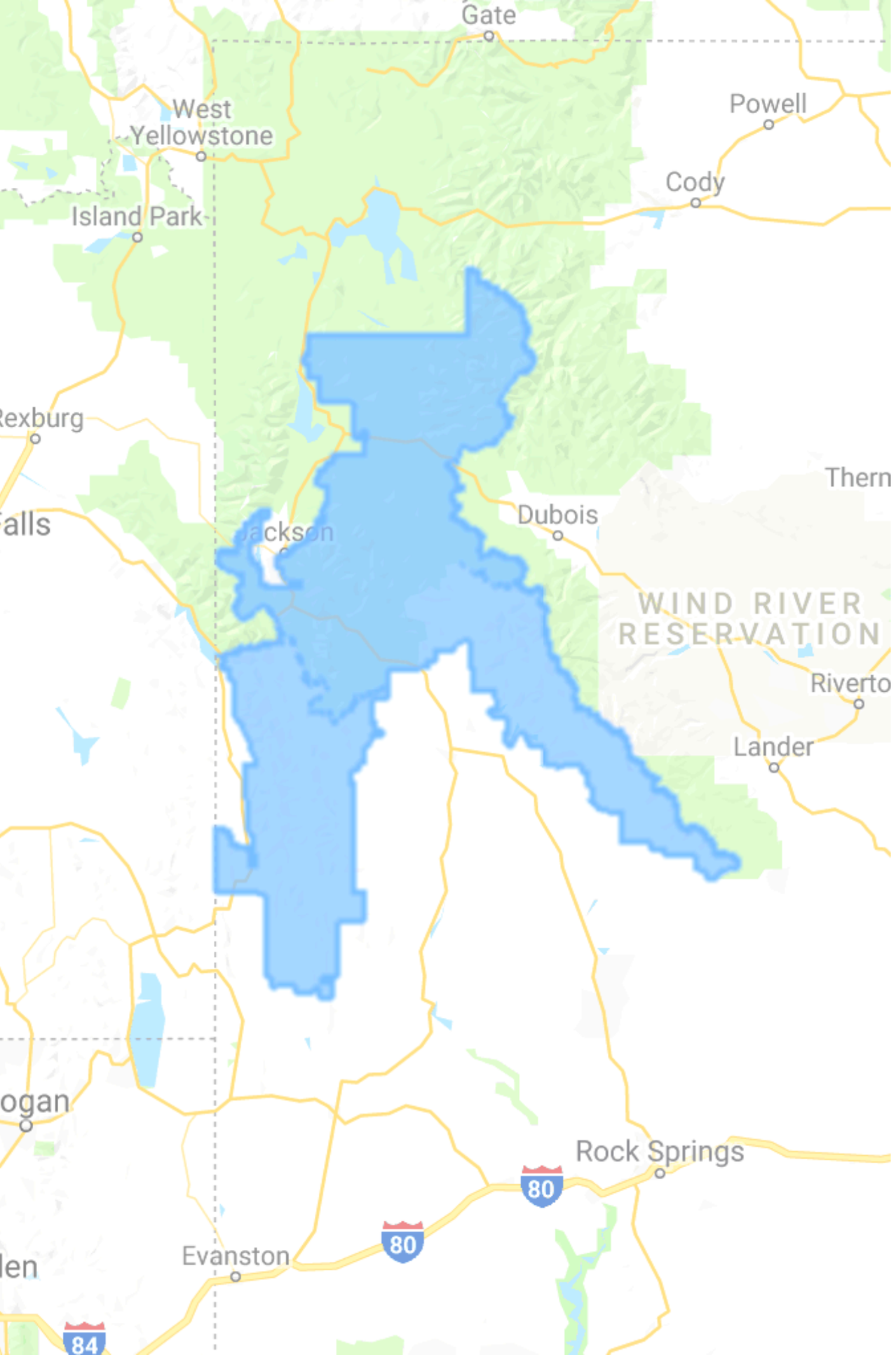
Examining Resident Populations and Development around National Forests in the U.S. – A Spatial and Socioeconomic Analysis Tool with Google Earth Engine

Google Earth Engine Project
F&ES 754 Geospatial Software Design
Instructor: Professor Dana Tomlin

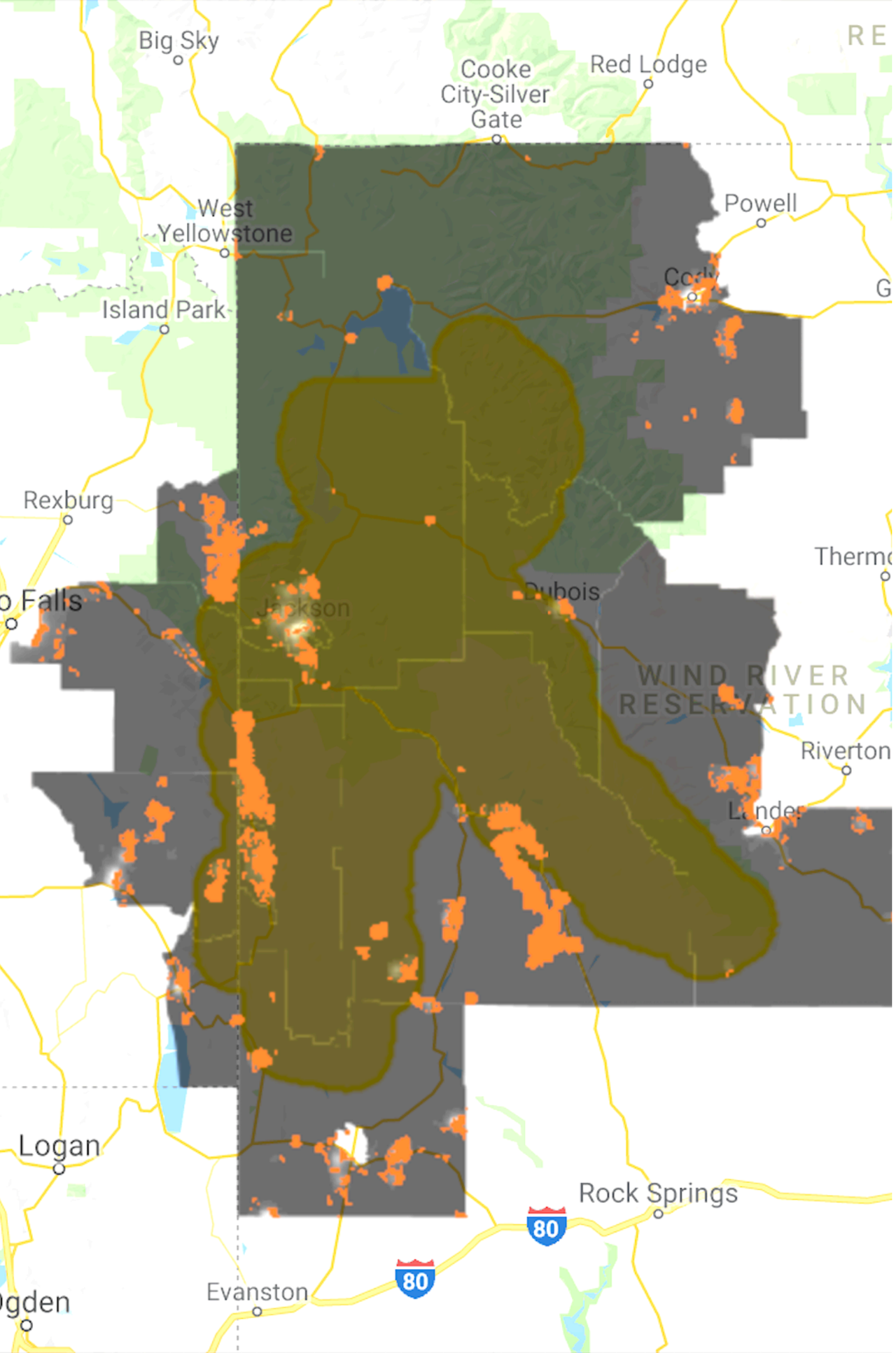
Chengcheng Qiu | Fall 2018
Yale School of Public Health



Introduction

This project demonstrates the application of Google Earth Engine in developing a spatial analysis tool for examining the population context and socioeconomic development of resident populations around the National Forests in the United States of interest.

Understanding the relationship between human society and the environment has become crucial in conservations of the national forests; however, difficulties exist in examining the human settlements and visualizing the socioeconomic changes around national forests across different places and times. Thus, being able to easily visualize the demographic profile and socioeconomic changes over years in populations around any of the national forests of interest is a key step in ultimately being able to interpret and use the information for research and environmental conservation.



Objective

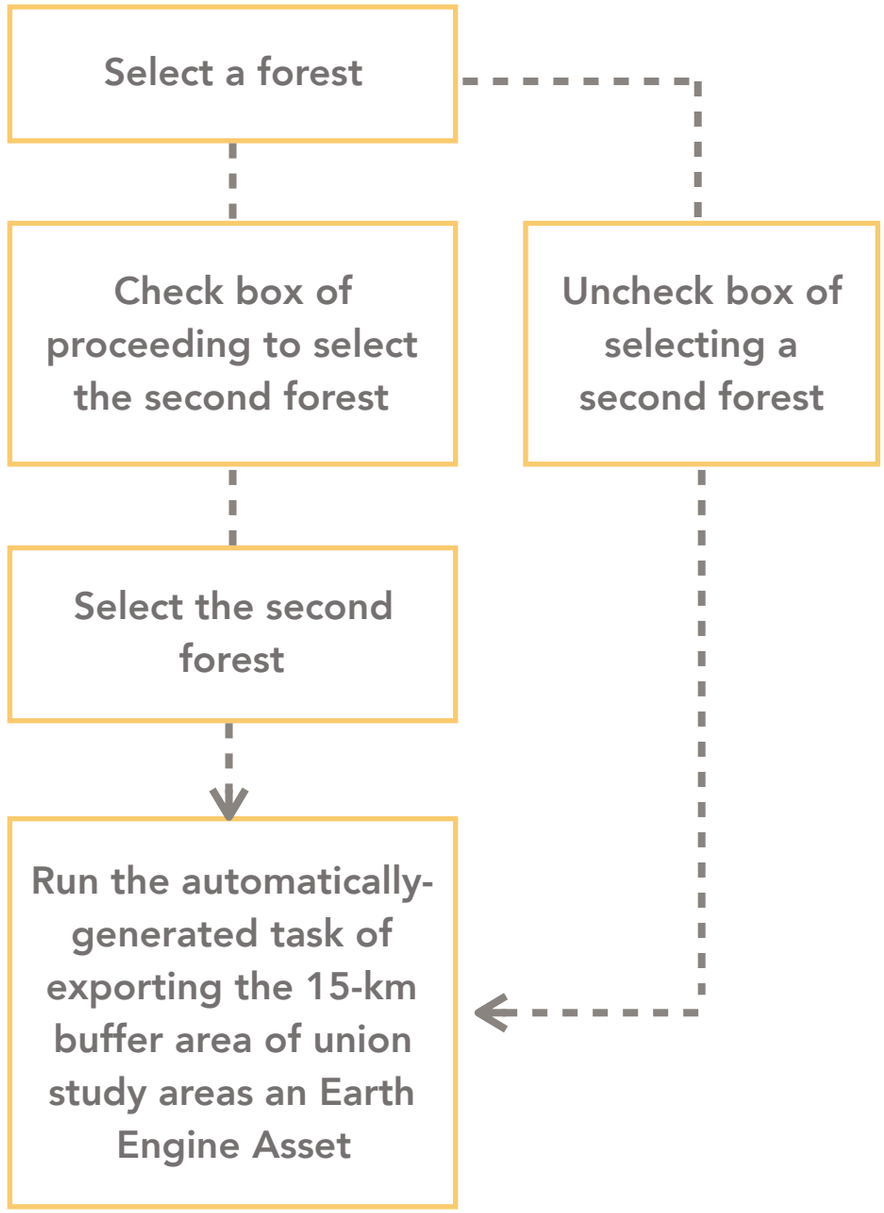
The objective was to create a tool that can be used for examining human settlements around any of the 152 original proclaimed national forests in the United States whose proclaimed boundaries were made available by the U.S. Forest Service[1].

This tool has been created to help easily visualize demographic information of the resident populations living within a defined distance from the national forests of interest, and to visualize socioeconomic growth through analysis of nighttime stable lights, which have been widely used as a proxy measure for intensity of economic activities. The approach to creating this tool can be easily modified to look at populations living within different distances from the national forests, and visualize economic development between different years.

[1] U.S. Forest Service, 2018.

Part 1. User Selection

Framework

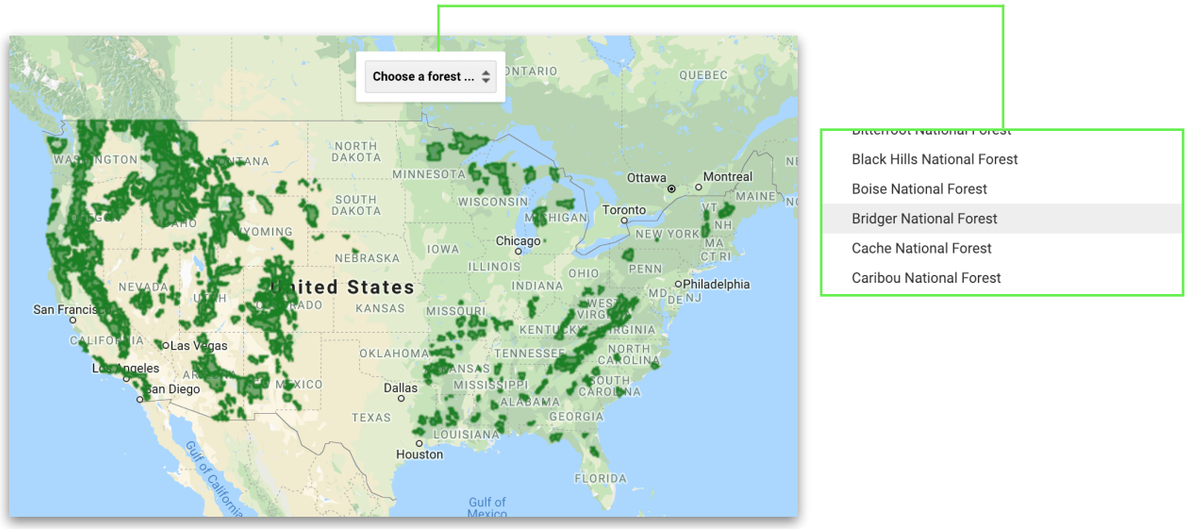


Approach

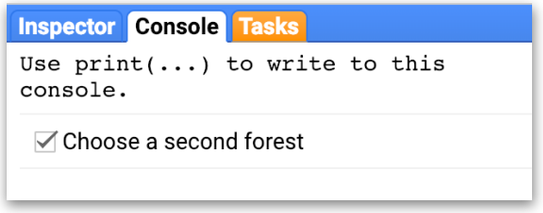
Link to code:
https://code.earthengine.google.com/6fb0841c2428220b211d6e76978f682b

All proclaimed national forests in the U.S. are presented in dark green on the interactive Google Map. The user is allowed to select up to 2 national forests for their analysis at one time.

1. Choose the national forest of interest from menu (alphabetically listed in order)



2. In the console window, select whether to proceed with selecting a second national forest to be analyzed in union with the first selected forest.

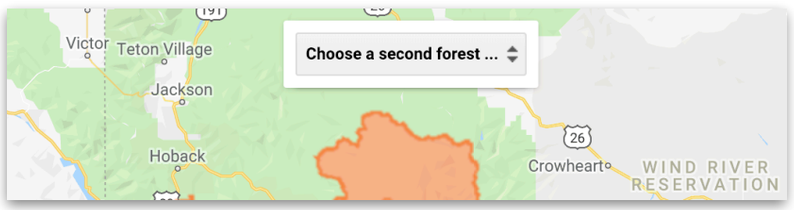


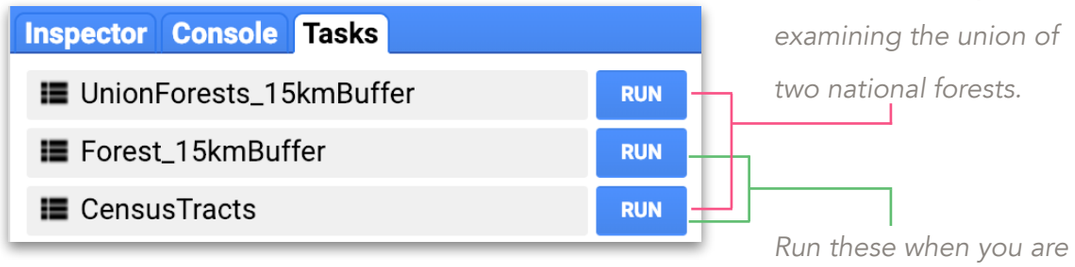
This option was enabled for the consideration that some adjacent national forests may be analyzed together as a single study area.

Approach (cont.)

3. If do not wish to choose a second forest, go to tasks window and run the two auto-generated tasks of creating Earth Engine assets for selected national forest and U.S. census tracts for later analysis.

Otherwise, check the box, select the second national forest from the menu, and then run the tasks of generating assets for 15km buffer area around the union of selected national forests and census tracts.





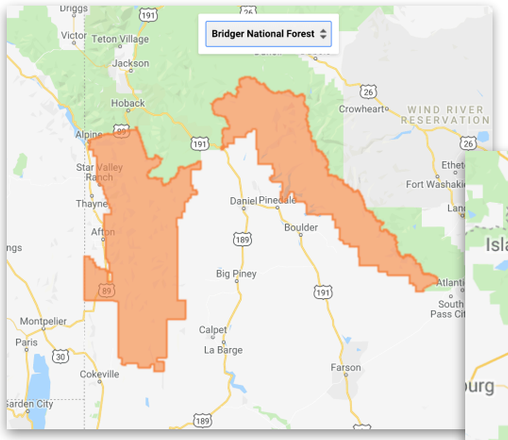
Run these when you are examining the union of two national forests.

Run these when you are examining only one national forest.

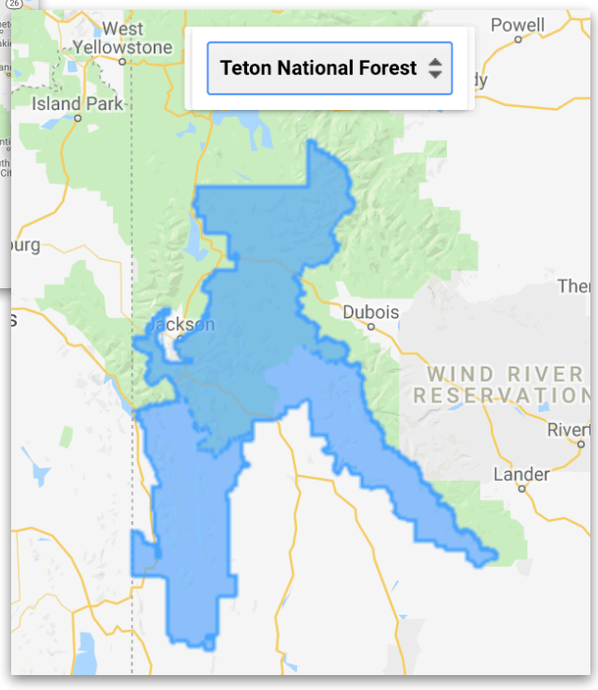
Example

In the project, I used the Bridger Teton National Forest for analysis and for demonstration of this tool. The study area is the union of Bridger and Teton National Forests.

Bridger National Forest



Bridger-Teton National Forest



Part 2. Demographic Profile

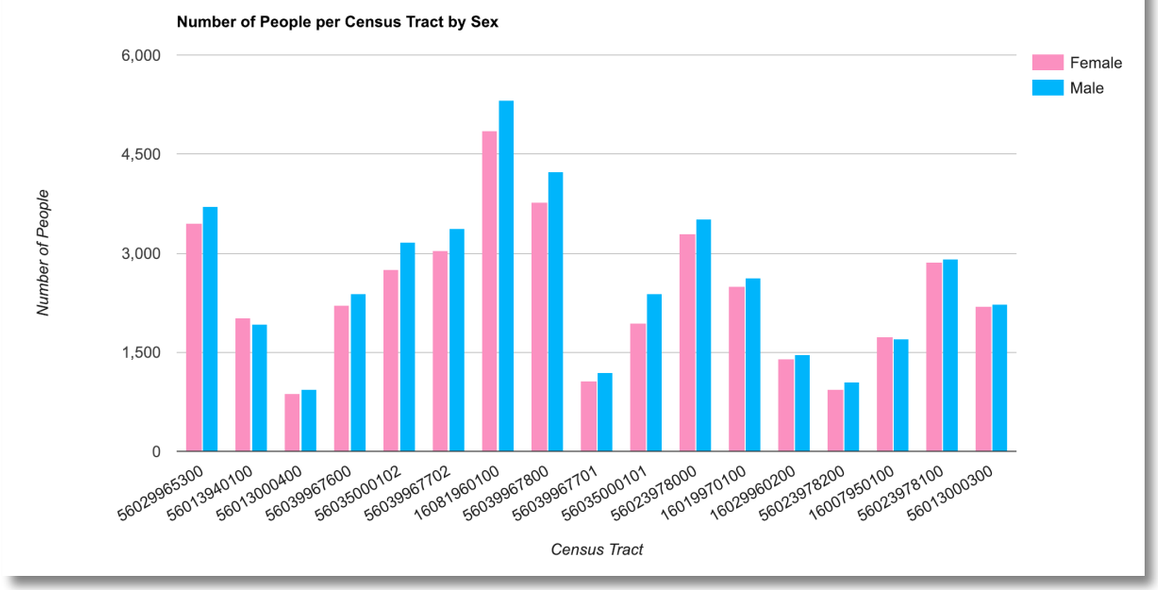
Approach

Link to code:
https://code.earthengine.google.com/9b66ac12b32098cc43c31a84554bae0b

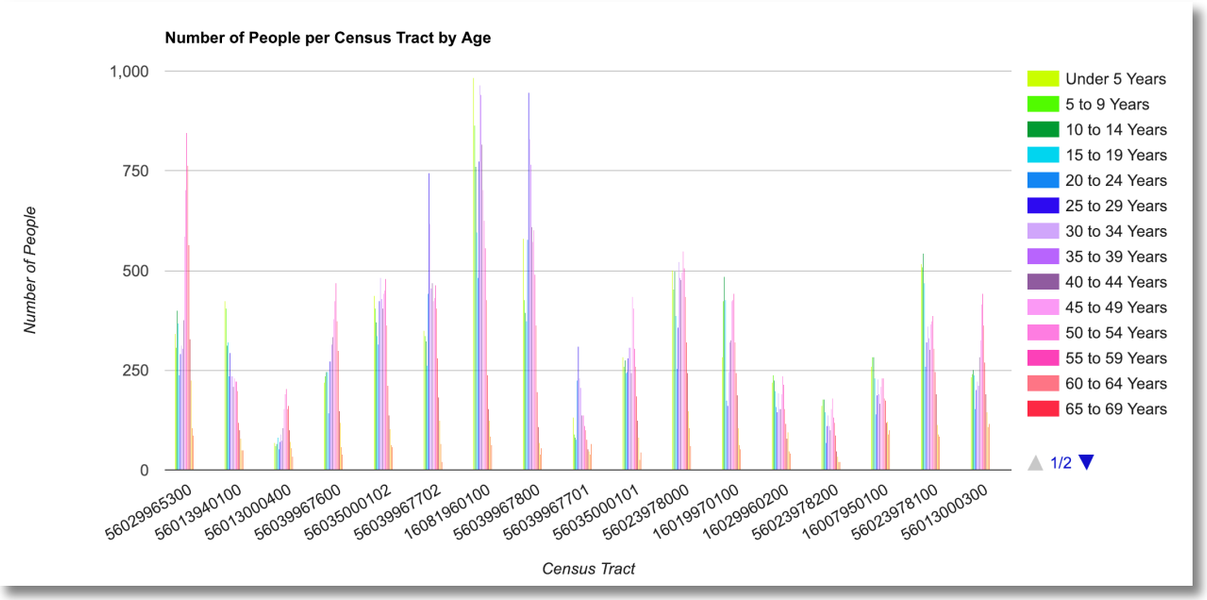
The script of this project has been divided into different pieces based on different purposes. After generating assets of 15-km buffer area of selected study sites and census tracts, run the second script to automatically read in the assets and generate demographic profile of the resident populations living within the buffer area. Information is available on sex, age, ethnicity, housing occupancy, and household distribution.

Example

```
Total Population                                    JSON
85050

Sex distribution (%)                                JSON
▼ Object (2 properties)                             JSON
    Female: 48.10817166372722
    Male: 51.89182833627278
```



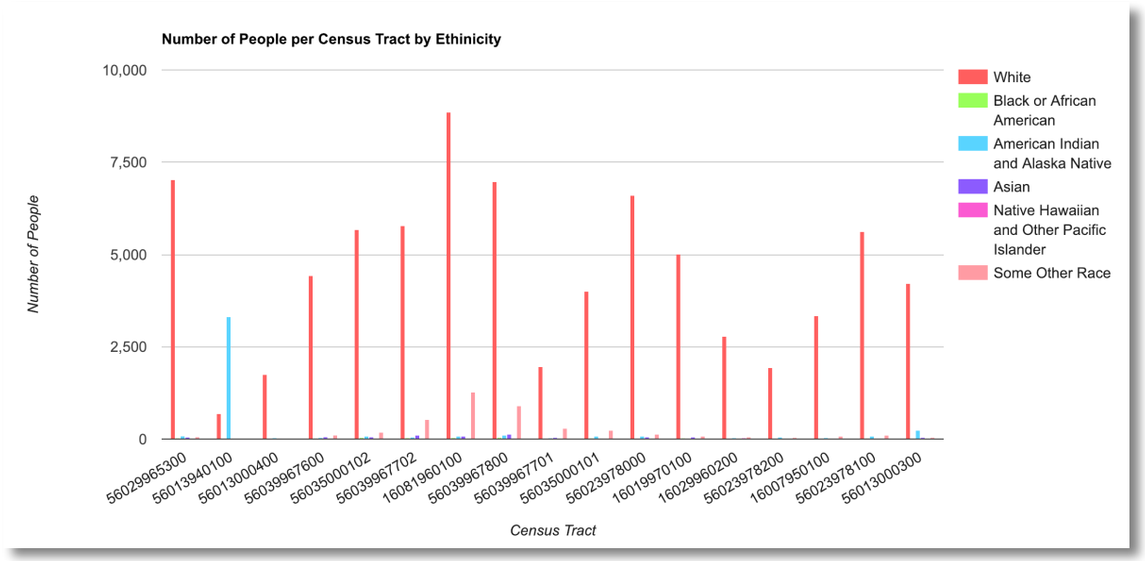

```
Age distribution (%)                                JSON
▼ Object (18 properties)                            JSON
    10 to 14 Years: 6.708994708994709
    15 to 19 Years: 5.893004115226337
    20 to 24 Years: 4.908877131099353
    25 to 29 Years: 6.940623162845386
    30 to 34 Years: 7.366255144032922
    35 to 39 Years: 6.867724867724869
    40 to 44 Years: 6.493827160493828
    45 to 49 Years: 7.327454438565549
    5 to 9 Years: 6.732510288065844
    50 to 54 Years: 7.777777777777778
    55 to 59 Years: 7.659024103468548
    60 to 64 Years: 6.225749559082892
    65 to 69 Years: 4.466784244562022
    70 to 74 Years: 2.9476778365667258
    75 to 79 Years: 2.136390358612581
    80 to 84 Years: 1.313345091122869
    85 Years and Over: 1.1746031746031746
    Under 5 Years: 7.059376837154615
```



The 2010 U.S. census tracts demographic data was obtained from the geodatabase named TIGER from the United States Census Bureau. Census tracts that are partly within the 15km buffer area around Bridger-Teton National Forest were auto-selected by the tool and were included in the demographic profile.

In general, there are slightly more males than females. There are less older adults that are aged more than 65 years of old.

Example (cont.)



Ethinicity distribution (%) JSON

▼ Object (6 properties) JSON

- American Indian and Alaska Native: 5.17225161669…
- Asian: 0.8865373309817755
- Black or African American: 0.3574368018812463
- Native Hawaiian and Other Pacific Islander: 0.15…
- Some Other Race: 4.858318636096414
- White: 90.11287477954144

The White population is the main ethnicity group in the area, compositing 90% of the whole population. One interesting finding is that, in census tract whose ID is 56013940100, American Indian and Alaska native is the main ethnicity group, different from all other census tracts.

The total housing units in this area is 46195 in 2010. Half of the total housing units are owner-occupied, and the second largest occupancy of housing units is for seasonal and recreation use.

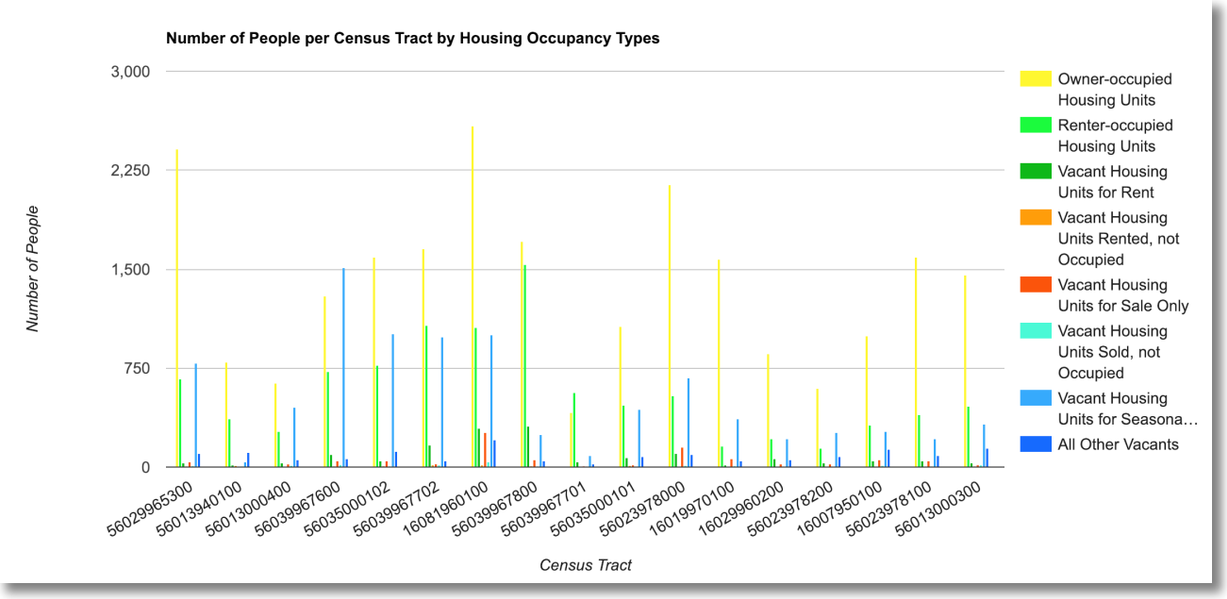
Total Housing Units JSON

46195

Housing occupancy distribution (%) JSON

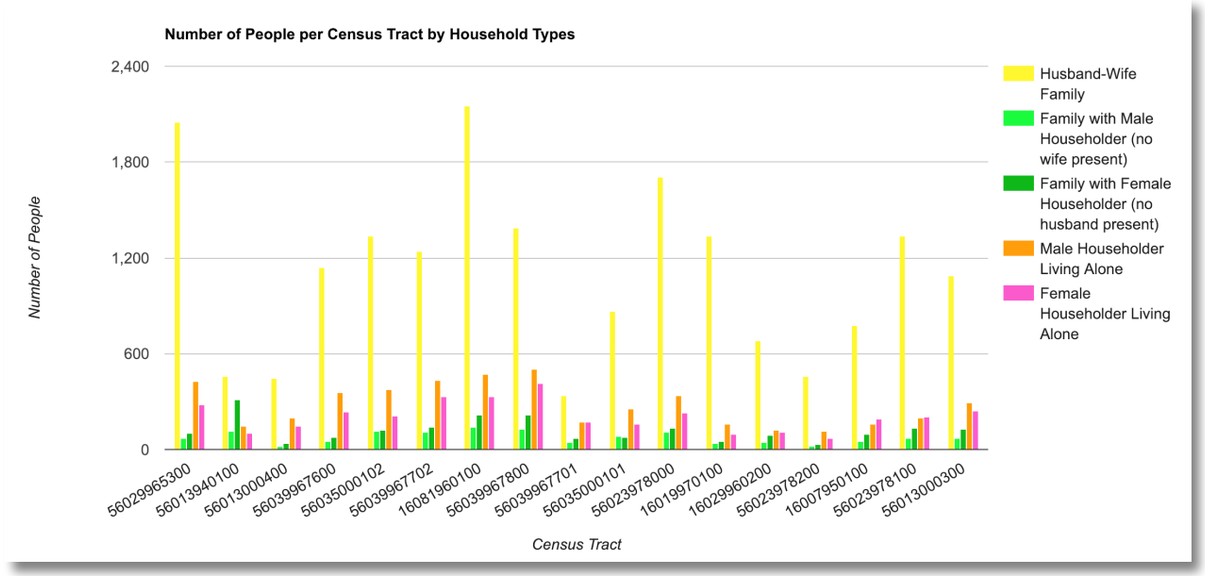
▼ Object (8 properties) JSON

- All Other Vacants: 3.268752029440416
- Owner-occupied Housing Units: 50.6288559367897
- Renter-occupied Housing Units: 21.110509795432407
- Vacant Housing Units Rented, not Occupied: 0.235956272323…
- Vacant Housing Units Sold, not Occupied: 0.37666414114081…
- Vacant Housing Units for Rent: 3.134538369953458
- Vacant Housing Units for Sale Only: 1.9699101634376013
- Vacant Housing Units for Seasonal/Recreational use: 19.27…



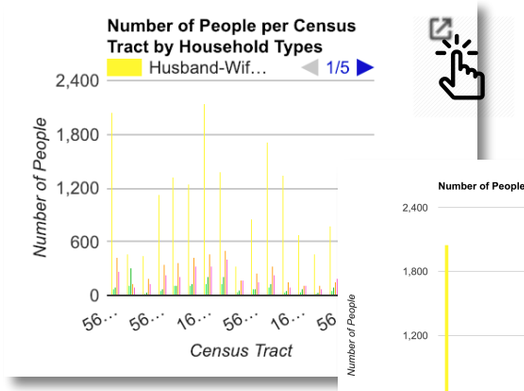
Example (cont.)

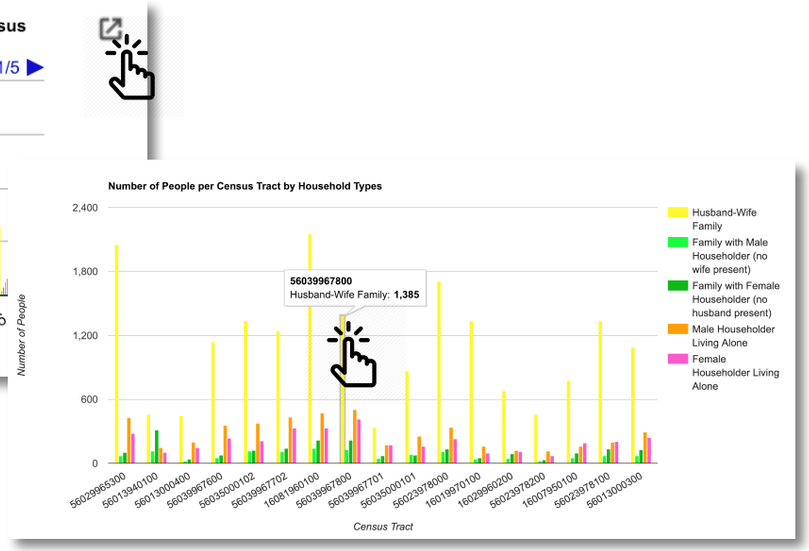
```
Population in occupied housing units distribution … JSON
▼ Object (2 properties)                              JSON
    Population in Owner-occupied Housing Units: 71.54758…
    Population in Renter-occupied Housing Units: 28.4524…

Household type distribution (%)                      JSON
▼ Object (5 properties)                              JSON
    Family with Female Householder (no husband present): 6.1194…
    Family with Male Householder (no wife present): 3.862401931…
    Female Householder Living Alone: 10.612552806276403
    Husband-Wife Family: 56.74109837054918
    Male Householder Living Alone: 14.227519613759807
```

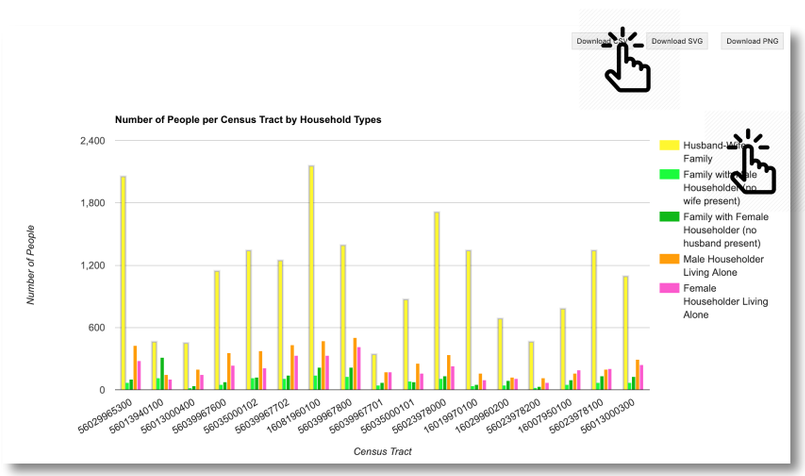


The majority of population live in owner-occupied houses, and most households are occupied by husband-wife family, followed by a relatively smaller percentage of non-family households occupied either by a female householder or a male householder.

The charts may be difficult to see in the console window, but by clicking on the button of opening in a new browser tab, the interactive chart would adjusts to the size of the window and allow the user to explore the distribution chart in greater details. The user can also move the mouse to whichever bar or category in the legends to examine the specific values, as shown below. The charts also can be downloaded in formats of CSV, SVG, or PNG.







Part 3. Development

Framework

Visualize stable nighttime light in 1992 and 2013

↓

Detect areas where nighttime light increases over the years

↓

Detect changes of suburban areas

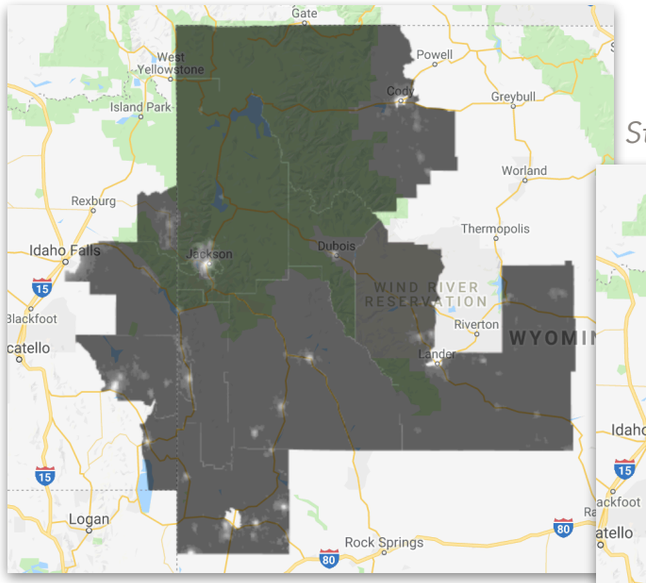
Approach

Link to code:
https://code.earthengine.google.com/a6c4c565e36f25f5122262b54ad6fff4

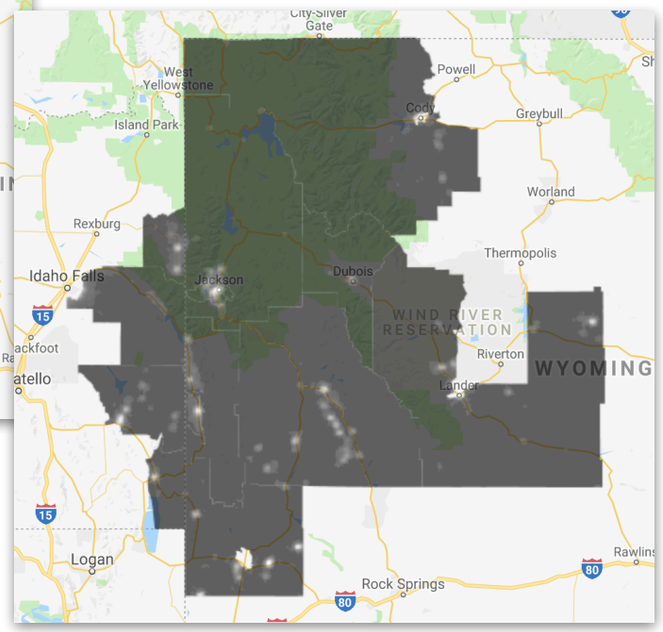
In examining the development around the study area, I chose to analyze the stable nighttime lights, which may serve as a proxy measure to economic development, and compared the light changes between 1992 and 2013. Also, this tool will highlight areas where stable nighttime light had increased, and areas that can be considered suburban, where the stable nighttime light has a value greater than 20.

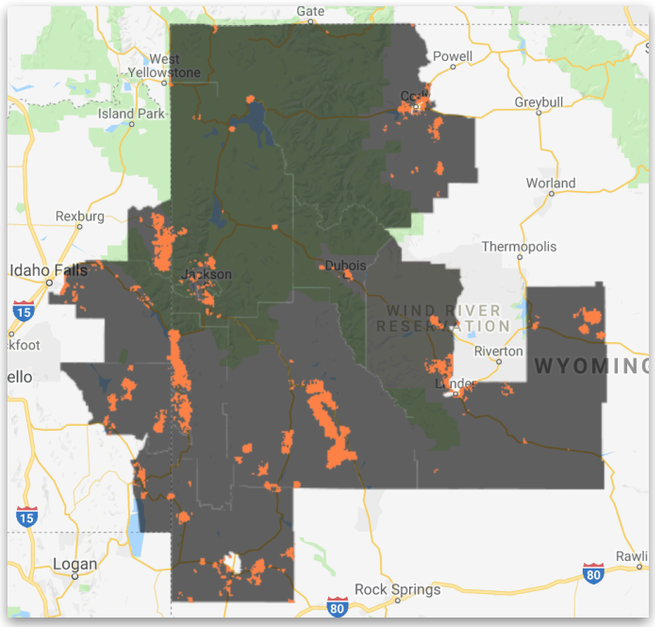
Example

Stable Nighttime Lights in 1992



Stable Nighttime Lights in 2013

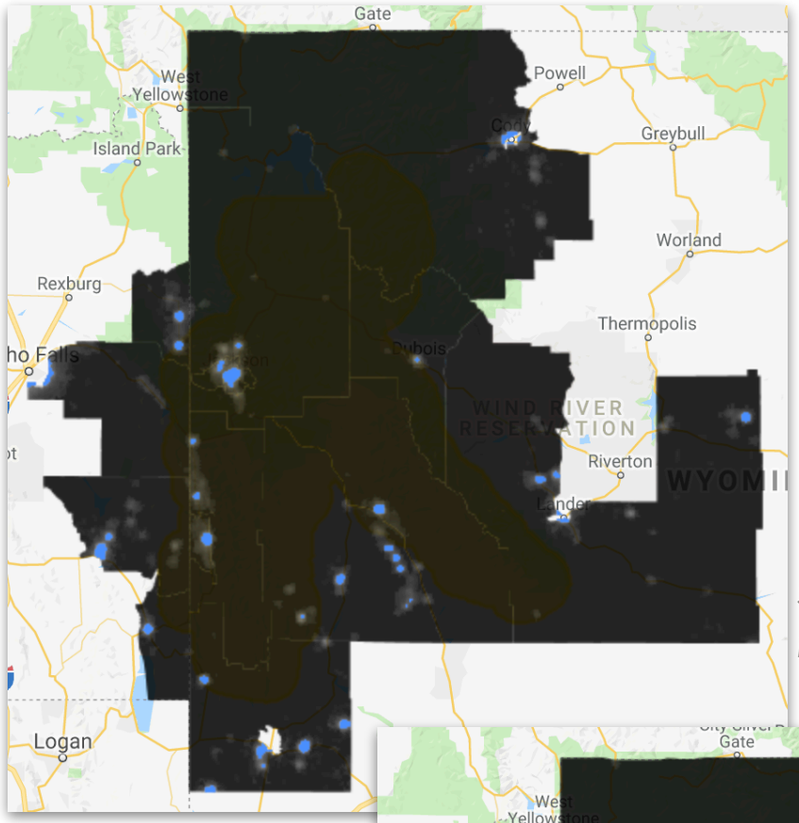


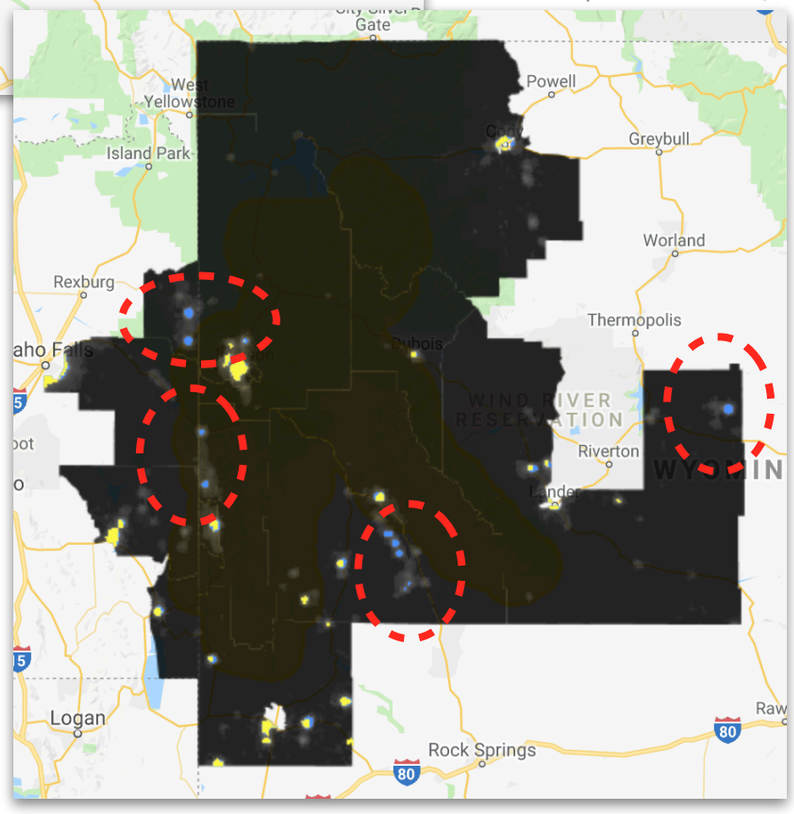


Highlighted areas indicate places where stable nighttime light had increased from 1992 to 2013

Example (cont.)

Suburban areas in 2013 (marked in blue)





Suburban areas in 2013 (marked in blue), and suburban areas in 1992 (on top, marked in yellow)

The spots circled in red indicate newly developed areas with increased popularity.

The coordinates for centroids of largest connected areas with continuously bright nighttime lights are made available to be viewed in the console window for different years, sorted in descending order.

```
point1992                                                    JSON
▼ FeatureCollection (4 elements, 3 col…                      JSON
    type: FeatureCollection
  ▶ columns: Object (3 properties)
  ▼ features: List (4 elements)
    ▼ 0: Feature -2057+898 (Point, 2 prope…
        type: Feature
        id: -2057+898
      ▼ geometry: Point (-110.81, 43.48)
          type: Point
        ▼ coordinates: [-110.8071902961429…
            0: -110.80719029614298
            1: 43.477910954725445
        geodesic: false
      ▶ properties: Object (2 properties)
    ▶ 1: Feature -2059+879 (Point, 2 prope…
    ▶ 2: Feature -2048+869 (Point, 2 prope…
    ▶ 3: Feature -2039+882 (Point, 2 prope…
```

```
point2013                                                    JSON
▼ FeatureCollection (4 elements, 3 colu…                     JSON
    type: FeatureCollection
  ▶ columns: Object (3 properties)
  ▼ features: List (4 elements)
    ▼ 0: Feature -2057+898 (Point, 2 propert…
        type: Feature
        id: -2057+898
      ▶ geometry: Point (-110.81, 43.48)
      ▶ properties: Object (2 properties)
    ▶ 1: Feature -2059+879 (Point, 2 propert…
    ▶ 2: Feature -2039+882 (Point, 2 propert…
    ▶ 3: Feature -2059+862 (Point, 2 propert…
```

Data Sources

1. Proclaimed National Forests boundaries shapefile by the U.S. Forest Services. https://data.fs.usda.gov/geodata/edw/edw_resources/meta/S_USA.ProclaimedForest.xml

2. 2010 U.S. Census Tracts Data. https://developers.google.com/earth-engine/datasets/catalog/TIGER_2010_Tracts_DP1

3. World nighttime light time series data. https://developers.google.com/earth-engine/datasets/catalog/NOAA_DMSP-OLS_NIGHTTIME_LIGHTS

References

Thanks for Professor Tomlin, Sabrina Szeto and TC Chakraborty's continuous help and support.

The analysis on determining urban areas and centroids of areas with continuously bright nighttime light regions credits to Qu, Ge (Chole)'s code for her Earth Engine term project written in 2017.

Script

```
// EE Term Project - National Forests (Part 1)

// Import U.S. census tracts demographic profile
var CensusTracts = ee.FeatureCollection( 'TIGER/2010/Tracts_DP1' );
// Map.addLayer( CensusTracts, { color: '98DEFF' }, 'Census Tract Profiles' ); (optionally hided)

// Export an ee.FeatureCollection as an Earth Engine asset
Export.table.toAsset({
  collection: CensusTracts,
  description:'CensusTracts',
  assetId: 'National_CensusTracts',
});

// Import proclaimed national forest boundaries shapefiles (polygons)
var forests = ee.FeatureCollection( 'ft:1ZsAuc_-VKhr0KNfnZUtDOfwUH4n1Wu0CLJBR32F_');
Map.addLayer(forests, { color: '1F8029' }, 'All National Forests')

// sort forest name in order
var forests = forests.sort( 'FORESTNAME');

// get the Forest names
var forestnames = forests.aggregate_array('FORESTNAME');

// Initialize a selection field
// Once a country has been selected, the redraw function is going
var select_forest = ui.Select({
                          items: forestnames.getInfo(),
                          onChange: view });

select_forest.setPlaceholder('Choose a forest ...');

// Add the drop-down 'select' widget to the map
Map.add(select_forest);

function view(key){

  // get the selected forest
  var selectedForest = ee.Feature(forests.filter(ee.Filter.eq('FORESTNAME', key)).first());
  Map.centerObject(selectedForest);

  // store the name of the selected forest
  var selectedForest_Strg = ee.String(selectedForest.get('FORESTNAME'));

  // add the forest geometry to the map
  var selectedForest_lyr = ui.Map.Layer(selectedForest, {color:'F78945'}, 'Selected forest');
  Map.layers().set(0, selectedForest_lyr);
```

```
// Buffer Zone of selected forest (within 15km)
  var fbuff15km  = selectedForest.geometry().buffer(15000);
  var buff15km  = ee.FeatureCollection([fbuff15km]);

  // Export the first ee.FeatureCollection as an Earth Engine asset.
  Export.table.toAsset({
    collection: buff15km,
    description:'Forest_15kmBuffer',
    assetId: 'Forest_15kmBuffer',
  });

  // initialize a checkbox field
  var checkbox = ui.Checkbox('Choose a second forest', false);

  checkbox.onChange(function(checked) {
  // Shows or hides the first map layer based on the checkbox's value.
  var select_2ndforest = ui.Select({
                              items: forestnames.getInfo(),
                              onChange: view2,
                              });

  select_2ndforest.setPlaceholder('Choose a second forest ...');

  // Add the drop-down 'select' widget to the map
  Map.add(select_2ndforest);
  });

  print(checkbox);

  function view2(key){

    // get the selected forest
    var selected2ndForest = ee.Feature(forests.filter(ee.Filter.eq('FORESTNAME', key)).first());

    // store the name of the selected forest
    var selected2ndForest_Strg = ee.String(selected2ndForest.get('FORESTNAME'));

    // union 1st and 2nd selected forest
    var UnionForests = selected2ndForest.union(selectedForest);

    // display unioned forests
    Map.centerObject(UnionForests);
    var UnionForests_lyr = ui.Map.Layer(UnionForests, {color:'49A4FF'}, 'Unioned selected
forest');
    Map.layers().set(0, UnionForests_lyr);
```

Script

```
    // Buffer Zone of selected forest (within 15km)
   var fbuff15km  = UnionForests.geometry().buffer(15000);
   var buff15km  = ee.FeatureCollection([fbuff15km]);

   // Export an ee.FeatureCollection as an Earth Engine asset.
   Export.table.toAsset({
     collection: buff15km,
     description:'UnionForests_15kmBuffer',
     assetId: 'Forest_15kmBuffer',
   });
  }
}
```

```
// EE Term Project - National Forests (Part 2)

// Import assets of census tracts and 10-km buffer of selected forest
var forest_buff15km = ee.FeatureCollection('users/chengchengqiu/Forest_15kmBuffer');
var censusTracts = ee.FeatureCollection('users/chengchengqiu/National_CensusTracts');

// Display 10-km buffer of selected forest
Map.addLayer(forest_buff15km, {color:'FFDE0A'} , '10km-Buffer Zone of Selected Forest');

// Select and display census tracts that intersect with the buffer area
var forest_buff15km_ct = censusTracts.filterBounds(forest_buff15km);
Map.centerObject(forest_buff15km_ct, 7);
Map.addLayer(forest_buff15km_ct,{color:'FF6100'},'Census Tracts within 10km-Buffer Zone');

// DEMOGRAPHIC PROFILE: sex, age, ethinicity, housing occupancy breakdown

// total population
var population = ee.Number(forest_buff15km_ct.aggregate_sum( 'dp0010001' ));
print( 'Total Population', population );

// plot histogram of the number of people per census tract by sex
var SexChart    = ui.Chart.feature.byFeature( forest_buff15km_ct, 'geoid10',
['dp0010039','dp0010020'] )
                .setChartType('ColumnChart');
var SexChart    = SexChart.setSeriesNames( ['Female','Male'] );
var SexChart    = SexChart.setOptions( { title: 'Number of People per Census Tract by Sex',
                                hAxis: { title: 'Census Tract' },
                                vAxis: { title: 'Number of People' },
                                colors: ['FF90C4', '0AB6FF'],
                                series: { 0: { lineWidth: 2, lineDashStyle: [3,2]},
                                          1: { lineWidth: 2, lineDashStyle: [3,2]},
                                        }
                              }   );
print( SexChart );


// sex breakdown
var female = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010039'));
var male = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010020'));

// calculate percentage
function Pct( tNUMBER, iNUMBER )
      { var    Percentage = iNUMBER.divide( tNUMBER ).multiply(100);
        return Percentage;
      }
```

Script

```
var pfemale = Pct(population, female);
var pmale = Pct(population, male);

// sex distribution in percentages
var sex_dist = ee.Dictionary({

  'Female':pfemale,
  'Male': pmale
});
print('Sex distribution (%)', sex_dist);

// plot histogram of the number of people per census tract by age group
var AgeChart    = ui.Chart.feature.byFeature( forest_buff15km_ct, 'geoid10',

['dp0010002','dp0010003','dp0010004','dp0010005','dp0010006',

'dp0010007','dp0010008','dp0010009','dp0010010','dp0010011',

'dp0010012','dp0010013','dp0010014','dp0010015','dp0010016',
                                    'dp0010017','dp0010018','dp0010019'] )
              .setChartType('ColumnChart');
var AgeChart    = AgeChart.setSeriesNames( ['Under 5 Years','5 to 9 Years','10 to 14 Years','15
to 19 Years',
                                    '20 to 24 Years','25 to 29 Years','30 to 34 Years','35 to 39 Years',
                                    '40 to 44 Years','45 to 49 Years','50 to 54 Years','55 to 59 Years',
                                    '60 to 64 Years','65 to 69 Years','70 to 74 Years','75 to 79 Years',
                                    '80 to 84 Years','85 Years and Over'] );
var AgeChart    = AgeChart.setOptions( { title: 'Number of People per Census Tract by Age',
                          hAxis: { title: 'Census Tract' },
                          vAxis: { title: 'Number of People' },
                          colors: ['CEFF00','54FF00','009C34',
                                  '0BD9F3','0B89F3','2D0BF3',
                                  'D0A6FF','BB64FF','925BA0',
                                  'FF9CF7','FF7EE2','FF45B8',
                                  'FF7787','FF2E47','B71E30',
                                  'FFD02E','FF902E','FF6A2E'],
                          lineWidth: 3,
                          lineDashStyle: [3,2],
                             }   );
print( AgeChart );

// age breakdown
var under5yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010002'));
var f5to9yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010003'));
var f10to14yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010004'));
var f15to19yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010005'));
```

```
var f20to24yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010006'));
var f25to29yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010007'));
var f30to34yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010008'));
var f35to39yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010009'));
var f40to44yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010010'));
var f45to49yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010011'));
var f50to54yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010012'));
var f55to59yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010013'));
var f60to64yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010014'));
var f65to69yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010015'));
var f70to74yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010016'));
var f75to79yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010017'));
var f80to84yr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010018'));
var f85overyr = ee.Number(forest_buff15km_ct.aggregate_sum('dp0010019'));

// calculate percentages
var punder5yr = Pct(population, under5yr);
var pf5to9yr = Pct(population, f5to9yr);
var pf10to14yr = Pct(population, f10to14yr);
var pf15to19yr = Pct(population, f15to19yr);
var pf20to24yr = Pct(population, f20to24yr);
var pf25to29yr = Pct(population, f25to29yr);
var pf30to34yr = Pct(population, f30to34yr);
var pf35to39yr = Pct(population, f35to39yr);
var pf40to44yr = Pct(population, f40to44yr);
var pf45to49yr = Pct(population, f45to49yr);
var pf50to54yr = Pct(population, f50to54yr);
var pf55to59yr = Pct(population, f55to59yr);
var pf60to64yr = Pct(population, f60to64yr);
var pf65to69yr = Pct(population, f65to69yr);
var pf70to74yr = Pct(population, f70to74yr);
var pf75to79yr = Pct(population, f75to79yr);
var pf80to84yr = Pct(population, f80to84yr);
var pf85overyr = Pct(population, f85overyr);

var age_dist = ee.Dictionary({'Under 5 Years': punder5yr, '5 to 9 Years': pf5to9yr,
                              '10 to 14 Years': pf10to14yr,'15 to 19 Years': pf15to19yr,
                              '20 to 24 Years': pf20to24yr,'25 to 29 Years': pf25to29yr,
                              '30 to 34 Years': pf30to34yr,'35 to 39 Years': pf35to39yr,
                              '40 to 44 Years': pf40to44yr,'45 to 49 Years': pf45to49yr,
                              '50 to 54 Years': pf50to54yr,'55 to 59 Years': pf55to59yr,
                              '60 to 64 Years': pf60to64yr,'65 to 69 Years': pf65to69yr,
                              '70 to 74 Years': pf70to74yr,'75 to 79 Years': pf75to79yr,
                              '80 to 84 Years': pf80to84yr,'85 Years and Over': pf85overyr});
print('Age distribution (%)', age_dist);
```

Script

```
// plot histogram of the number of people per census tract by ethinicity group
var EthChart    = ui.Chart.feature.byFeature( forest_buff15km_ct, 'geoid10',

['dp0090001','dp0090002','dp0090003','dp0090004','dp0090005',
                                'dp0090006'] )
            .setChartType('ColumnChart');
var EthChart    = EthChart.setSeriesNames( ['White','Black or African American','American
Indian and Alaska Native','Asian',
                                'Native Hawaiian and Other Pacific Islander','Some Other Race'] );
var EthChart    = EthChart.setOptions( { title: 'Number of People per Census Tract by Ethinicity',
                        hAxis: { title: 'Census Tract' },
                        vAxis: { title: 'Number of People' },
                        colors: ['FF5E5E','9BFF5E','5ED6FF','8C5EFF',
                                'FF5ED6','FFA0A3'],
                        lineWidth: 2,
                        lineDashStyle: [3,2],
                           }   );
print( EthChart );

// ethinicity breakdown
var white = ee.Number(forest_buff15km_ct.aggregate_sum('dp0090001'));
var black_aframer = ee.Number(forest_buff15km_ct.aggregate_sum('dp0090002'));
var amerindi_alaska = ee.Number(forest_buff15km_ct.aggregate_sum('dp0090003'));
var asian = ee.Number(forest_buff15km_ct.aggregate_sum('dp0090004'));
var hawaiian_opacific = ee.Number(forest_buff15km_ct.aggregate_sum('dp0090005'));
var other = ee.Number(forest_buff15km_ct.aggregate_sum('dp0090006'));

var pwhite = Pct(population, white);
var pblack_aframer = Pct(population, black_aframer);
var pamerindi_alaska = Pct(population, amerindi_alaska);
var pasian = Pct(population, asian);
var phawaiian_opacific = Pct(population, hawaiian_opacific);
var potherrace = Pct(population, other);

var ethinicity_dist = ee.Dictionary({'White': pwhite,
                        'Black or African American': pblack_aframer,
                        'American Indian and Alaska Native': pamerindi_alaska,
                        'Asian': pasian,
                        'Native Hawaiian and Other Pacific Islander': phawaiian_opacific,
                        'Some Other Race': potherrace});
print('Ethinicity distribution (%)', ethinicity_dist);


// plot histogram of the number of people per census tract by housing occupancy types
var HousingChart    = ui.Chart.feature.byFeature( forest_buff15km_ct, 'geoid10',

                        ['dp0210002','dp0210003','dp0180004','dp0180005','dp0180006',
                         'dp0180007','dp0180008','dp0180009'] )
                .setChartType('ColumnChart');
var HousingChart    = HousingChart.setSeriesNames( ['Owner-occupied Housing Units','Renter-
occupied Housing Units',
                        'Vacant Housing Units for Rent','Vacant Housing Units Rented, not
Occupied',
                        'Vacant Housing Units for Sale Only','Vacant Housing Units Sold,
not Occupied',
                        'Vacant Housing Units for Seasonal, Recreational, or Occasional
use',
                        'All Other Vacants'] );
var HousingChart    = HousingChart.setOptions( { title: 'Number of People per Census Tract by
Housing Occupancy Types',
                        hAxis: { title: 'Census Tract' },
                        vAxis: { title: 'Number of People' },
                        colors: ['FFF935','15FF3C','06BB1C','FF9F15',
                                'FF5515','4DFBD6','35ACFF','156AFF'],
                        lineWidth: 2,
                        lineDashStyle: [3,2],
                           }   );
print( HousingChart );

// total housing units
var housingunits = ee.Number(forest_buff15km_ct.aggregate_sum( 'dp0180001' ));
print( 'Total Housing Units', housingunits );

// housing occupancy
var occupiedhu_owner = ee.Number(forest_buff15km_ct.aggregate_sum('dp0210002'));
var occupiedhu_renter = ee.Number(forest_buff15km_ct.aggregate_sum('dp0210003'));
var vacanthu_rent = ee.Number(forest_buff15km_ct.aggregate_sum('dp0180004'));
var vacanthu_rented = ee.Number(forest_buff15km_ct.aggregate_sum('dp0180005'));
var vacanthu_sale = ee.Number(forest_buff15km_ct.aggregate_sum('dp0180006'));
var vacanthu_sold = ee.Number(forest_buff15km_ct.aggregate_sum('dp0180007'));
var vacanthu_occasional = ee.Number(forest_buff15km_ct.aggregate_sum('dp0180008'));
var vacanthu_other = ee.Number(forest_buff15km_ct.aggregate_sum('dp0180009'));

var poccupiedhu_owner = Pct(housingunits, occupiedhu_owner);
var poccupiedhu_renter = Pct(housingunits, occupiedhu_renter);
var pvacanthu_rent = Pct(housingunits, vacanthu_rent);
var pvacanthu_rented = Pct(housingunits, vacanthu_rented);
var pvacanthu_sale = Pct(housingunits, vacanthu_sale);
var pvacanthu_sold = Pct(housingunits, vacanthu_sold);
var pvacanthu_occasional = Pct(housingunits, vacanthu_occasional);
var pvacanthu_other = Pct(housingunits, vacanthu_other);
```

Script

```
var hoccupancy_dist = ee.Dictionary({'Owner-occupied Housing Units': poccupiedhu_owner,
                                     'Renter-occupied Housing Units': poccupiedhu_renter,
                                     'Vacant Housing Units for Rent':pvacanthu_rent,
                                     'Vacant Housing Units Rented, not Occupied':pvacanthu_rented,
                                     'Vacant Housing Units for Sale Only':pvacanthu_sale,
                                     'Vacant Housing Units Sold, not Occupied':pvacanthu_sold,
                                     'Vacant Housing Units for Seasonal/Recreational
use':pvacanthu_occasional,
                                     'All Other Vacants':pvacanthu_other});
print('Housing occupancy distribution (%)', hoccupancy_dist);

// population in occupied housing units
var pop_ownerOccupied = ee.Number(forest_buff15km_ct.aggregate_sum('dp0220001'));
var pop_renterOccupied = ee.Number(forest_buff15km_ct.aggregate_sum('dp0220002'));

// total population in occupied housing units
var pop_inOccupied = ee.Number(pop_ownerOccupied.add(pop_renterOccupied));

var ppop_ownerOccupied = Pct(pop_inOccupied, pop_ownerOccupied);
var ppop_renterOccupied = Pct(pop_inOccupied, pop_renterOccupied);

var popoccupancy_dist = ee.Dictionary({'Population in Owner-occupied Housing Units':
ppop_ownerOccupied,
                                       'Population in Renter-occupied Housing Units':
ppop_renterOccupied});
print('Population in occupied housing units distribution (in percentage)', popoccupancy_dist);


// plot histogram of the number of people per census tract by household types
var HouseholdChart    = ui.Chart.feature.byFeature( forest_buff15km_ct, 'geoid10',

['dp0130004','dp0130006','dp0130008','dp0130012','dp0130014'] )
                    .setChartType('ColumnChart');
var HouseholdChart    = HouseholdChart.setSeriesNames( ['Husband-Wife Family','Family with
Male Householder (no wife present)',
                                'Family with Female Householder (no husband present)','Male
Householder Living Alone',
                                'Female Householder Living Alone'] );
var HouseholdChart    = HouseholdChart.setOptions( { title: 'Number of People per Census
Tract by Household Types',
                               hAxis: { title: 'Census Tract' },
                               vAxis: { title: 'Number of People' },
                               colors: ['FFF935','15FF3C','06BB1C','FF9F15',
                                       'FF5ED0'],
                               lineWidth: 2,
                               lineDashStyle: [3,2],
                                     }   );
print( HouseholdChart );

// total households
var household = ee.Number(forest_buff15km_ct.aggregate_sum( 'dp0130001' ));

// household types
var fam_husbandwife = ee.Number(forest_buff15km_ct.aggregate_sum('dp0130004'));
var fam_male = ee.Number(forest_buff15km_ct.aggregate_sum('dp0130006'));
var fam_female = ee.Number(forest_buff15km_ct.aggregate_sum('dp0130008'));
var nonfam_male = ee.Number(forest_buff15km_ct.aggregate_sum('dp0130012'));
var nonfam_female = ee.Number(forest_buff15km_ct.aggregate_sum('dp0130014'));

var pfam_husbandwife = Pct(household, fam_husbandwife);
var pfam_male = Pct(household, fam_male);
var pfam_female = Pct(household, fam_female);
var pnonfam_male = Pct(household, nonfam_male);
var pnonfam_female = Pct(household, nonfam_female);

var household_dist = ee.Dictionary({'Husband-Wife Family': pfam_husbandwife,
                                    'Family with Male Householder (no wife present)': pfam_male,
                                    'Family with Female Householder (no husband present)':pfam_female,
                                    'Male Householder Living Alone':pnonfam_male,
                                    'Female Householder Living Alone':pnonfam_female});
print('Household type distribution (%)', household_dist);
```

Script

```
// EE Term Project - National Forests (Part 3)

// Import assets of census tracts and 15-km buffer of selected forest
var forest_buff15km = ee.FeatureCollection('users/chengchengqiu/Forest_15kmBuffer');
var censusTracts = ee.FeatureCollection('users/chengchengqiu/National_CensusTracts');

// Display 15-km buffer of selected forest
Map.centerObject(forest_buff15km, 7);
Map.addLayer(forest_buff15km, {color:'FFDE0A'} , '15km-Buffer Zone of Selected Forest');

// Select and display census tracts that intersect with the buffer area
var forest_buff15km_ct = censusTracts.filterBounds(forest_buff15km);
Map.addLayer(forest_buff15km_ct,{color:'FF6100'},'Census Tracts within 15km-Buffer Zone');

/* nighttime map - find human settlements - calculate distance to the forest */
// import the nighttime lights image collection (Jan 1,1992 - Jan 1,2014)
var world_nightlights = ee.ImageCollection('NOAA/DMSP-OLS/NIGHTTIME_LIGHTS');

// convert image collection to a list and get the images for 1992 and 2013
var listnightlights = world_nightlights.toList(35,0);
var nightlight1992 = ee.Image(listnightlights.get(0));
var nightlight2013 = ee.Image(listnightlights.get(34));
print('listofyears',listnightlights,'light1992',nightlight1992,'light2013',nightlight2013);

// select stable lights band from the images for 1992 and 2013
var stableLight1992 = nightlight1992.select(['stable_lights'],['1992StableLight']);
var stableLight2013 = nightlight2013.select(['stable_lights'],['2013StableLight']);

// clip nightlight image for census tracts within the 15-km buffer area of selected national forest
var ClippedLightImg1992 = stableLight1992.clip(forest_buff15km_ct);
var ClippedLightImg2013 = stableLight2013.clip(forest_buff15km_ct);

// visualize nightlight images
Map.addLayer( ClippedLightImg1992, {min:0, max:55, opacity:0.6},  'StableLight1992' );
Map.addLayer( ClippedLightImg2013, {min:0, max:55, opacity:0.6},  'StableLight2013' );

// stable light change from 1992 t0 2013
var lightChange92to13 =
ClippedLightImg2013.subtract(ClippedLightImg1992).set("lightChange92to13","lightChange92to
13")
var lightIncrease92to13 = lightChange92to13.gt(0);
var lightIncrease92to13 = lightChange92to13.mask(lightIncrease92to13);
Map.addLayer(lightChange92to13, {min:0, max:55, opacity:0.6},  'lightChange92to13' );
Map.addLayer( lightIncrease92to13, {palette:['#FF8548']},  'LightIncrease92to13' );

// define urban areas as having nightlight greater than 20
var urban1992 = ClippedLightImg1992.gt(20);
var urban2013 = ClippedLightImg2013.gt(20);

// visualize urban areas
var urban1992 = urban2013.mask(urban1992);
var urban2013 = urban2013.mask(urban2013);
Map.addLayer( urban2013, {palette:['#4B8FFF']},  'Urban Area in 2013' );
Map.addLayer( urban1992, {palette:['#FFF74B']},  'Urban Area in 1992' );

// identify connected areas with continously bright nighttime lights
var connectPixel1992 = urban1992.connectedPixelCount(1024, true);
var connectPixel2013 = urban2013.connectedPixelCount(1024, true);

// identify the centroids of these connected areas with continuous nighttime lights & record the
number of pixels contained
var forest_buff15kmfeature = ee.Feature(forest_buff15km.first());
var forest_buff15km_geo = forest_buff15kmfeature.geometry();

//The count of pixels in each component is a proxy for the size of each identified area
var point1992 = connectPixel1992.reduceToVectors(null,forest_buff15km_geo,
6000,'centroid',true,'numPixels','EPSG:3857');
var point2013 = connectPixel2013.reduceToVectors(null,forest_buff15km_geo,
6000,'centroid',true,'numPixels','EPSG:3857');

//Sort these centroids in descending order by the number of pixels contained in the components
var point1992 = point1992.sort('numPixels',false);
var point2013 = point2013.sort('numPixels',false);
print("point1992",point1992);
print("point2013",point2013);
```