



ArcGIS API for Python for ANALYSTS and DATA SCIENTISTS

ROHIT SINGH
ANDREW CHAPKOWSKI

2018 Esri DEVSummit Conference | Palm Springs, CA

Analysts, Data Scientists and Developers

Analyst

- Uses graphical tools
- Can call functions, cut & paste code
- Can change some variables

Gets paid for:
Insight

Excel, VB, Tableau,
Python

Data Scientist

- Builds simple apps & workflows
- Used to be "just an analyst"
- Likes coding to solve problems
- Doesn't want to be a "full-time programmer"

Gets paid (like a rock star) for:
Code that produces insight

SAS, R, Matlab,
Python

Developer

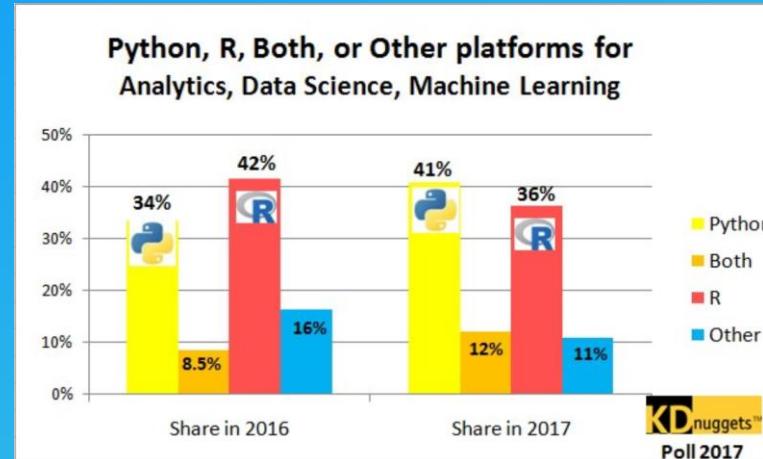
- Creates frameworks & compilers
- Uses IDEs
- Degree in CompSci
- Knows multiple languages

Gets paid for:
Code

C, C++, Java, JS,
Python

Why Python?

- Popularity
- Productivity
- Interoperability
- Solves the “two-language” problem
- Scientific Python ecosystem
- Community



CAME FOR THE
LANGUAGE
STAYED FOR THE
COMMUNITY



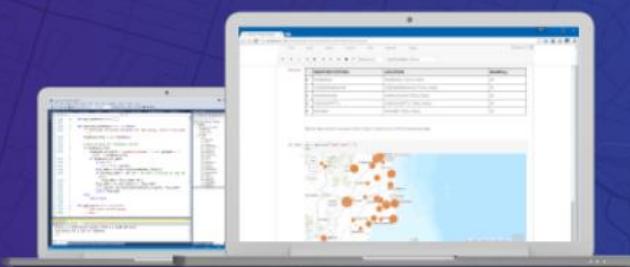


Source: [Gateway Data Sciences Courses Reach Enrollment Milestones](#)

ArcGIS API for Python

[Install the API](#)

Version 1.3 · December 2017

[Home](#) [Guide](#) [Sample Notebooks](#) [API Reference](#) [Community](#)

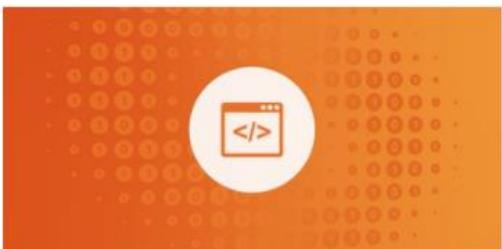
A powerful Python library for spatial analysis, mapping and GIS

ArcGIS API for Python is a Python library for working with maps and geospatial data, powered by web GIS. It provides simple and efficient tools for sophisticated vector and raster analysis, geocoding, map making, routing and directions, as well as for organizing and managing a GIS with users, groups and information items. In addition to working with your own data, the library enables access to ready to use maps and curated geographic data from Esri and other authoritative sources. It also integrates well with the scientific Python ecosystem and includes rich support for Pandas and Jupyter notebook.

[Install the API](#) | [Get started](#) | [View samples](#)

Understand your GIS

This "hello world" style notebook shows how to get started with the GIS and visualize its contents.

[Get started with the GIS class](#)

Manage your GIS

The ArcGIS API for Python provides APIs and samples for ArcGIS Online administrators to manage their online organization.

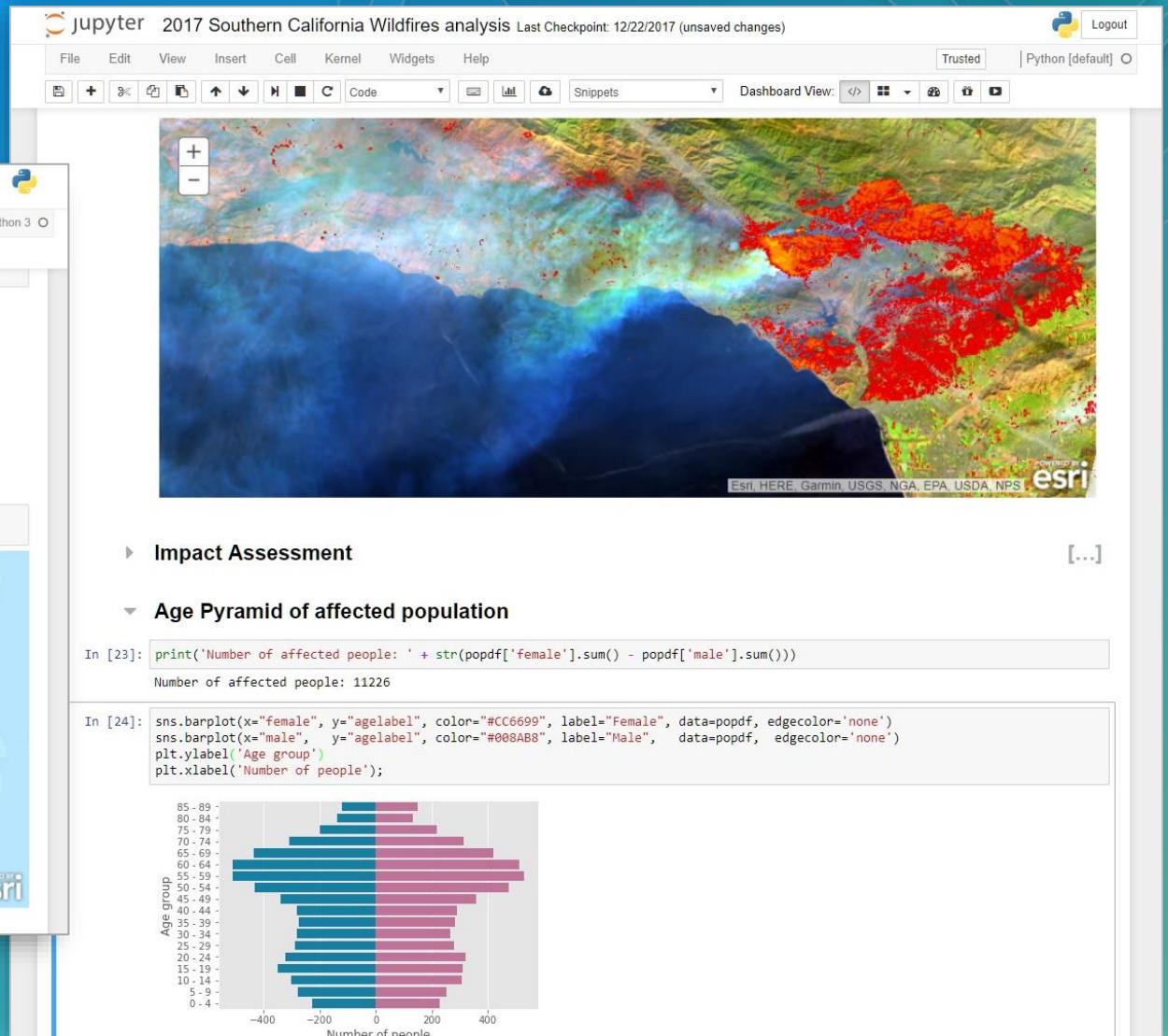
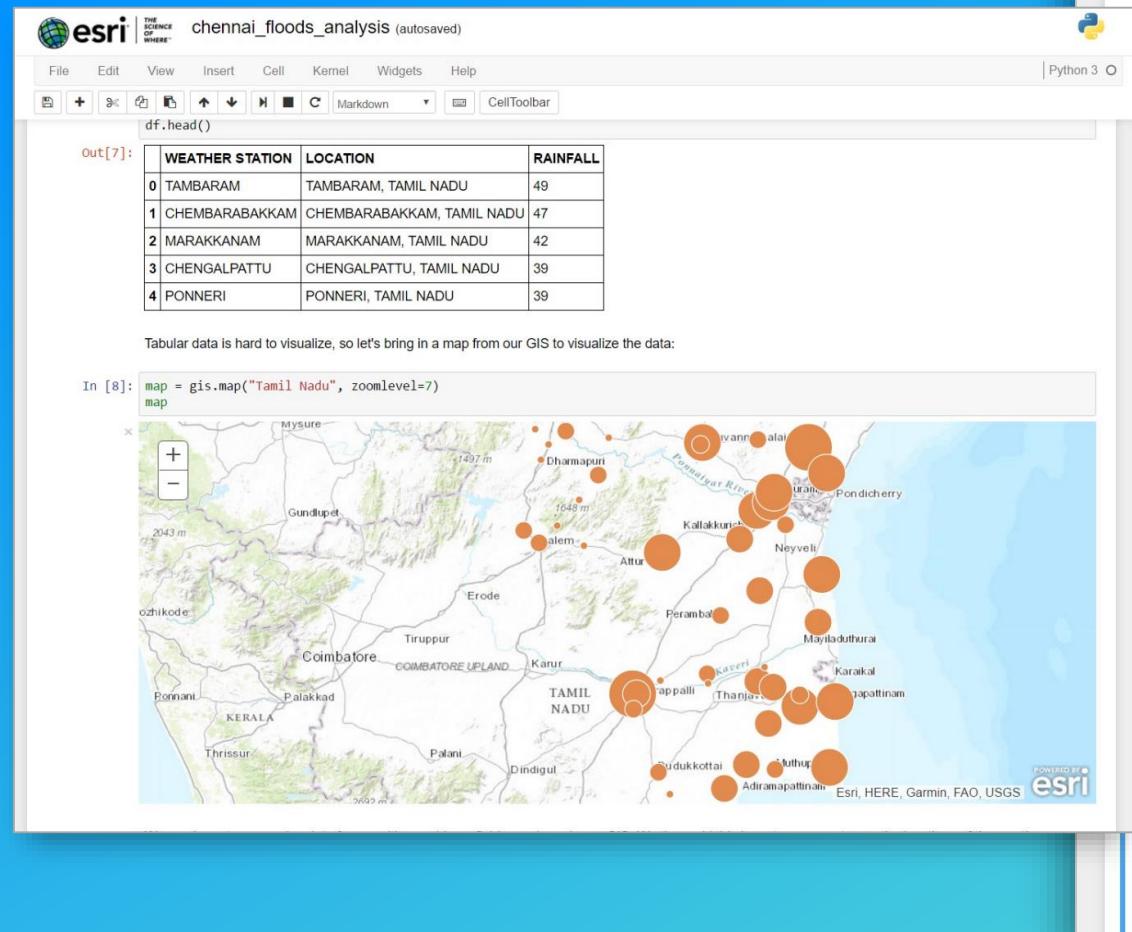
[Clone a portal](#)

Perform Spatial Analysis

Call sophisticated spatial analysis tools that work with online content, using a few lines of code.

[Chennai floods analysis](#)

ArcGIS + Jupyter = ❤



It all starts with your GIS

```
In [1]: from arcgis.gis import GIS
```

```
In [2]: gis = GIS('https://deldev.maps.arcgis.com', 'demo_deldev')
```

```
In [3]: enterprise = GIS('https://python.playground.esri.com/portal', 'arcgis_python',
```

Search for content

```
In [4]: items = gis.content.search('San Diego')
```

```
In [5]: for item in items:  
    display(item)
```



Places to see in San Diego

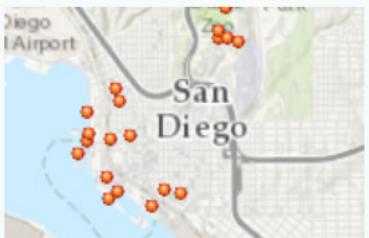
Places to see in San Diego



Feature Collection by deldev

Last Modified: July 01, 2017

0 comments, 512 views



San Diego attractions web map

Esri Story Maps team member and San Diego resident Rupert Essinger selects some places you might enjoy.



Web Map by deldev

Last Modified: July 01, 2017

0 comments, 3 views



San Diego Trolley stations

San Diego Trolley stations

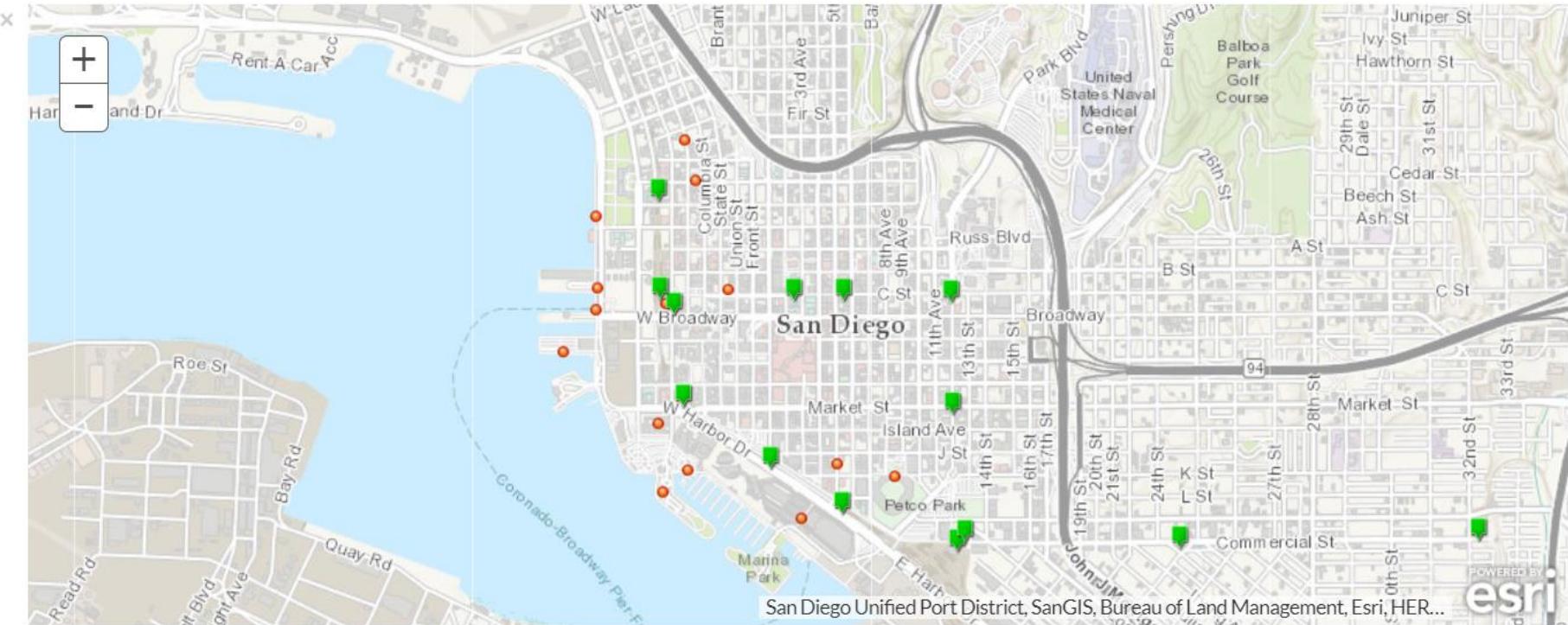


Feature Collection by deldev

Last Modified: June 23, 2017

Visualize layers on map widget

```
In [7]: sdmap = gis.map('San Diego', zoomlevel=14)  
sdmap
```



```
In [8]: sdmap.add_layer(sd_attractions)
```

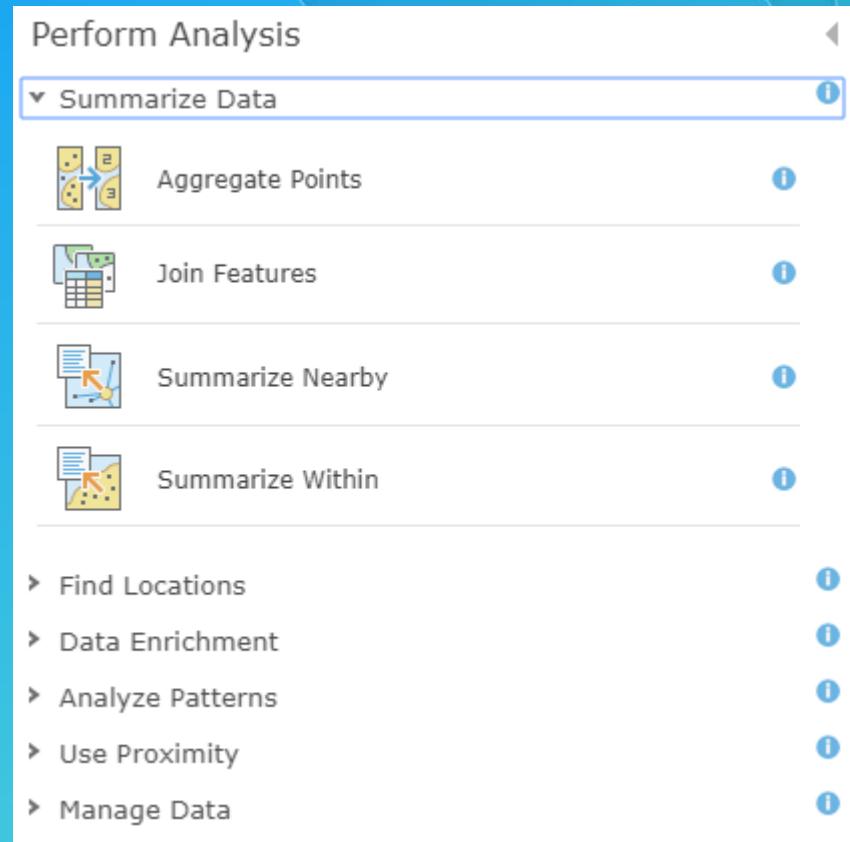
```
In [9]: sdmap.add_layer(trolley_stations)
```

Spatial Analysis

Discover relationships, patterns and trends in data
arcgis.feature submodules

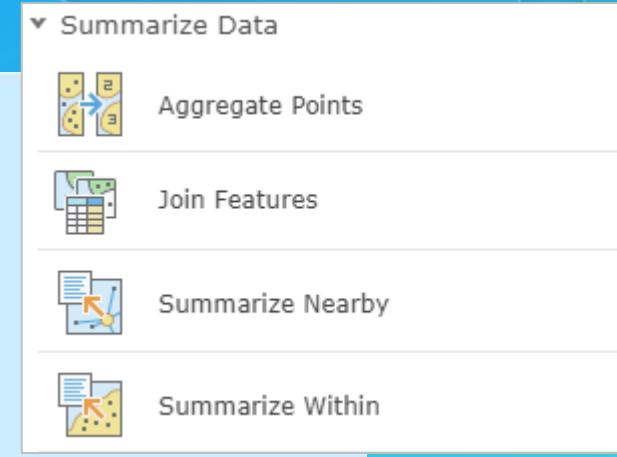
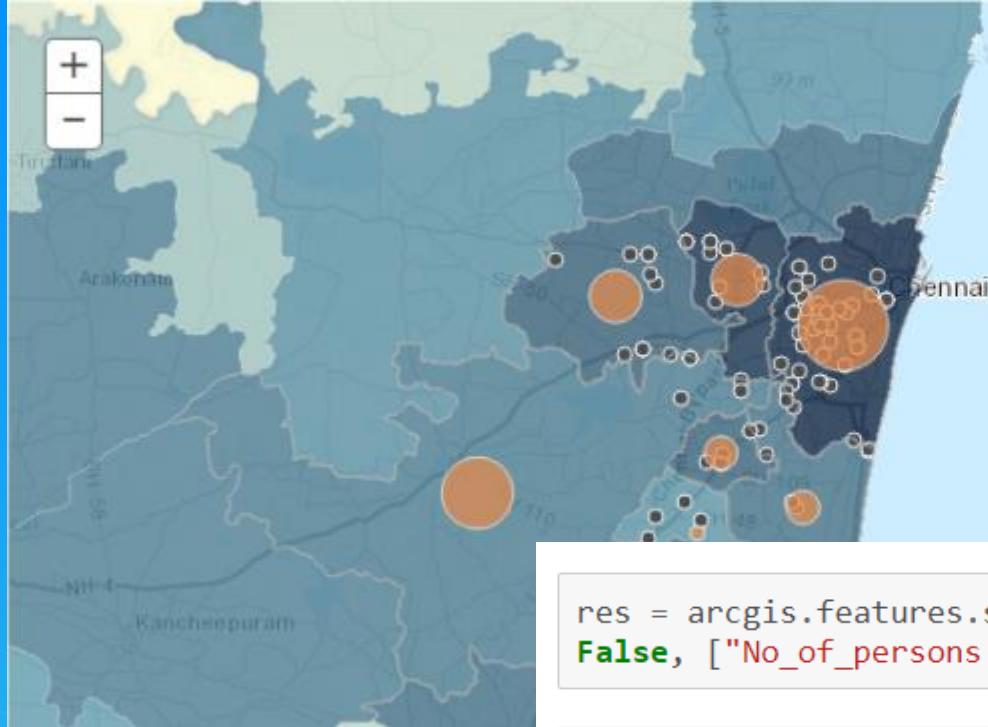
Spatial Analysis Tools

- Summarize Data
- Find Locations
- Data Enrichment
- Analyze Patterns
- Use Proximity
- Manage Data



Summarize Data

Calculate summary statistics for features and attributes



```
res = arcgis.features.summarize_data.aggregate_points(relief_centers, chennai_pop_featurelayer,  
False, ["No_of_persons Sum"])
```

```
aggr_lyr = res['aggregated_layer']
```

```
reliefmap.add_layer(aggr_lyr, { "renderer": "ClassedSizeRenderer",  
"field_name": "SUM_No_of_persons"})
```

Enrich Layer

Add detailed demographic data and statistics to your analysis

```
enriched_crime = enrich_data.enrich_layer(police_beats,  
                                         analysis_variables=analysis_variables)
```

Submitted.

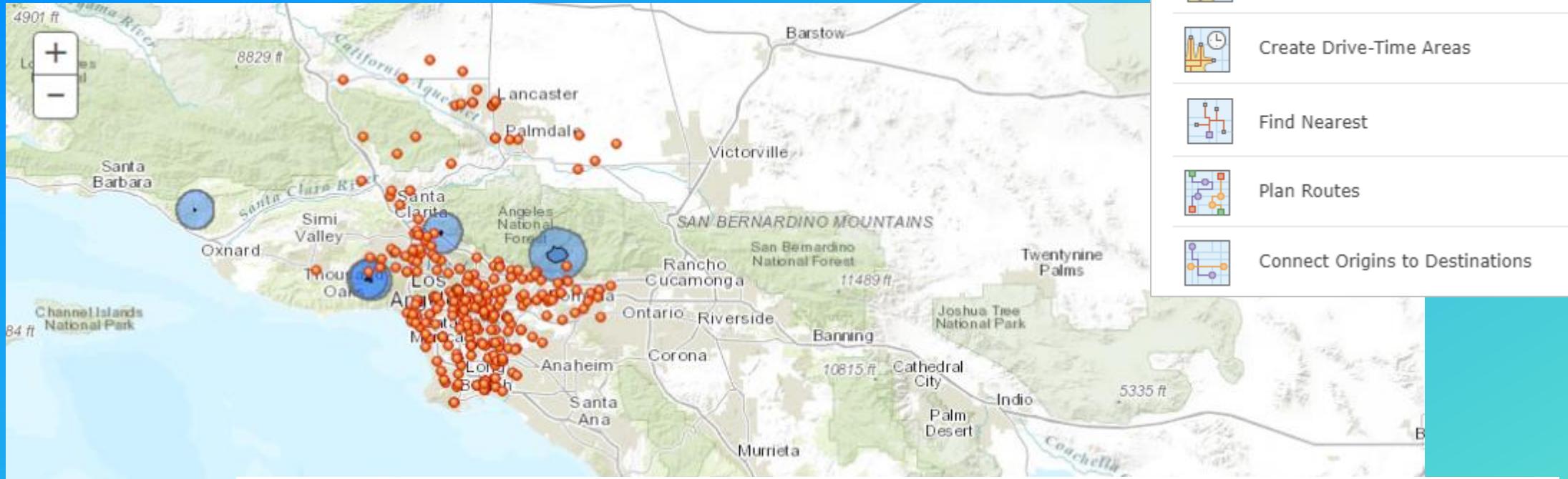
Executing...

```
enriched_df = enriched_crime.query().df  
enriched_df.head()
```

	ASSCDEG_CY	AVGFMSZ_CY	AVGHHSZ_CY	AVGHINC_CY	BACHDEG_CY	DIVINDX_CY	EDUCBASECY	ENRICH_FID
0	0.0	0.00	0.00	0.0	0.0	0.0	0.0	1.0
1	0.0	3.57	2.29	93321.0	16.0	79.6	30.0	2.0
2	545.0	2.34	1.35	99836.0	1142.0	72.8	12007.0	3.0
3	869.0	2.55	1.64	112053.0	8333.0	63.8	20489.0	4.0

Use Proximity

“What is near what?”



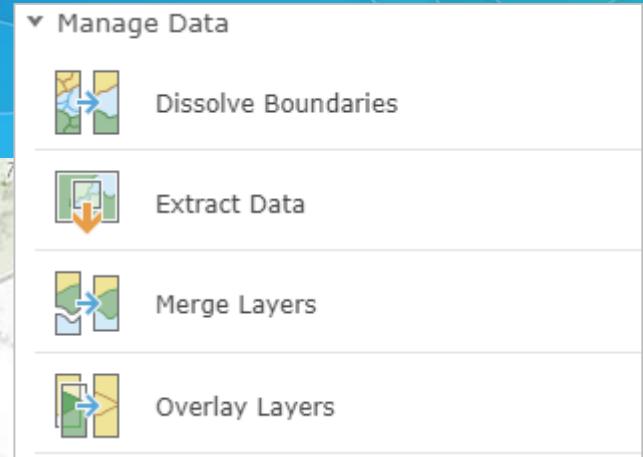
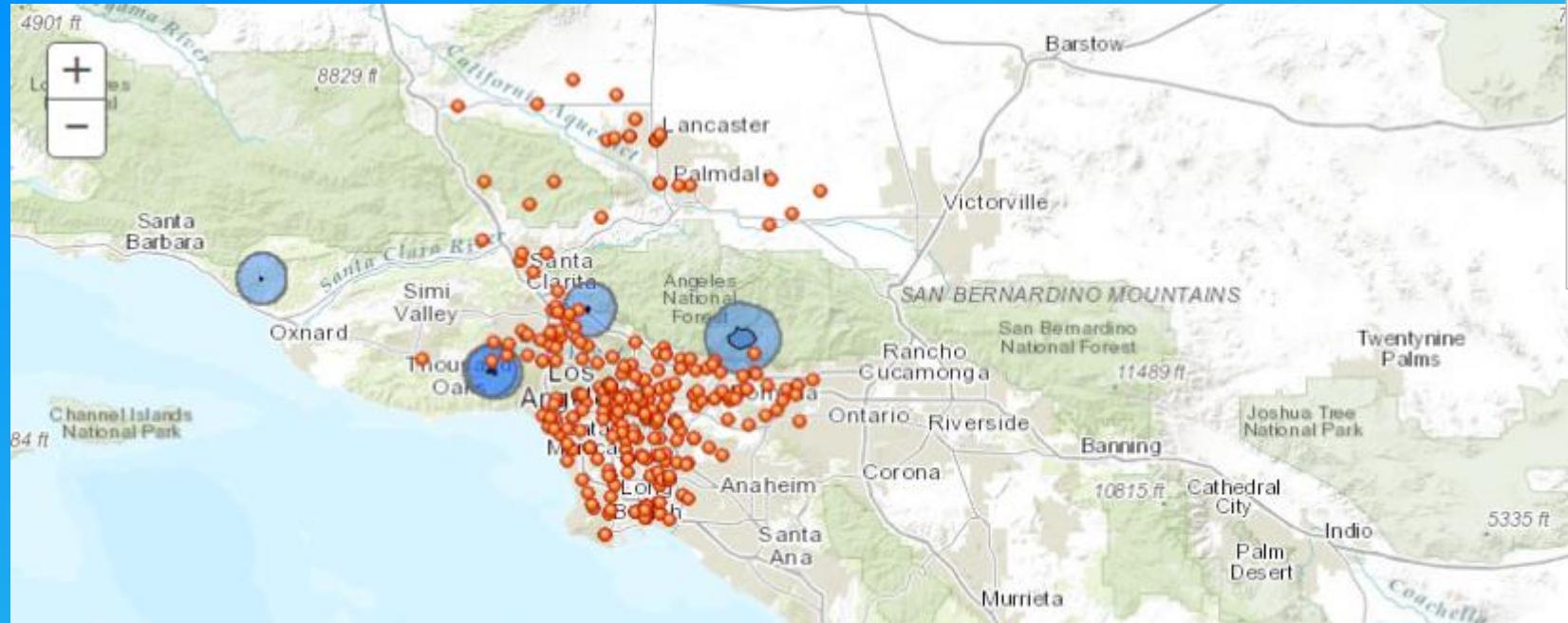
from arcgis.features.use_proximity import create_buffers

```
# buffer the active fire boundaries and add as new content
```

```
timestamp = '{:%Y_%m_%d_%H_%M_%S}'.format(datetime.datetime.now())
firebuffers = create_buffers(fires, [4], None, 'Miles', output_name="Fire_Buffers_" + timestamp)
```

Manage Data

Manage geographic data, overlay layers

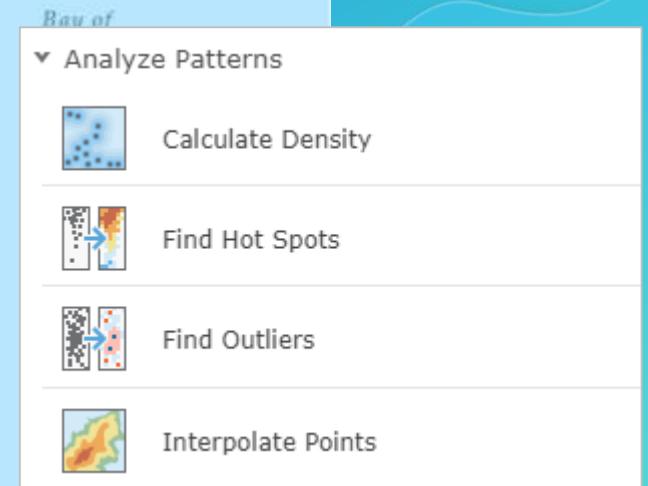
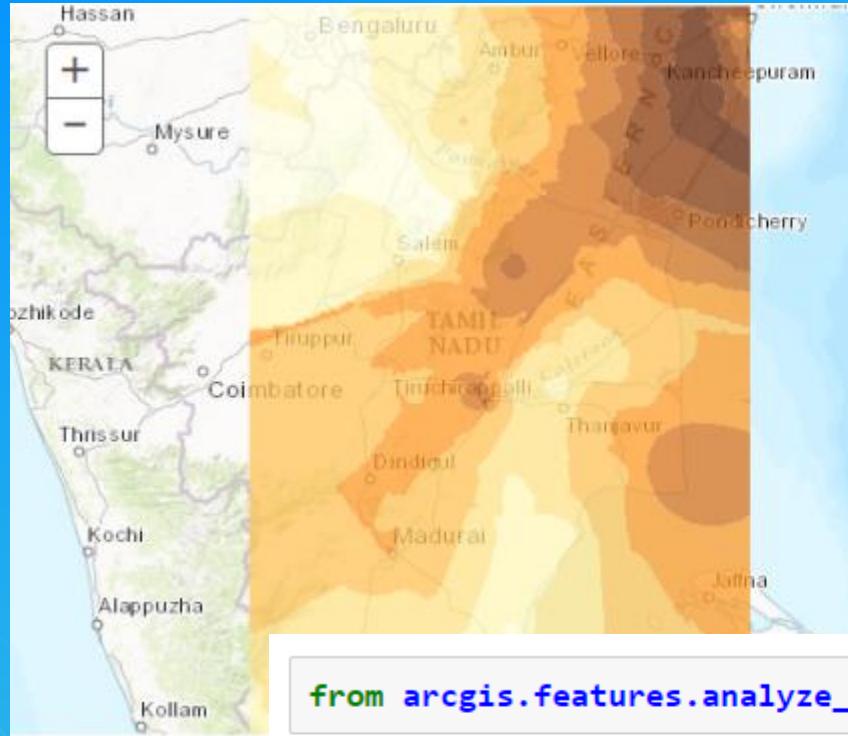


```
from arcgis.features.manage_data import overlay_layers

# run analysis to determine critical infrastructure within the risk boundaries
riskinfra = overlay_layers(firebuffers, infra,
                           overlay_type="Intersect",
                           output_name="At_Risk_Infrastructure_" + timestamp)
```

Analyze Patterns

Identify, quantify, and visualize spatial patterns in your data.



```
from arcgis.features.analyze_patterns import interpolate_points
```

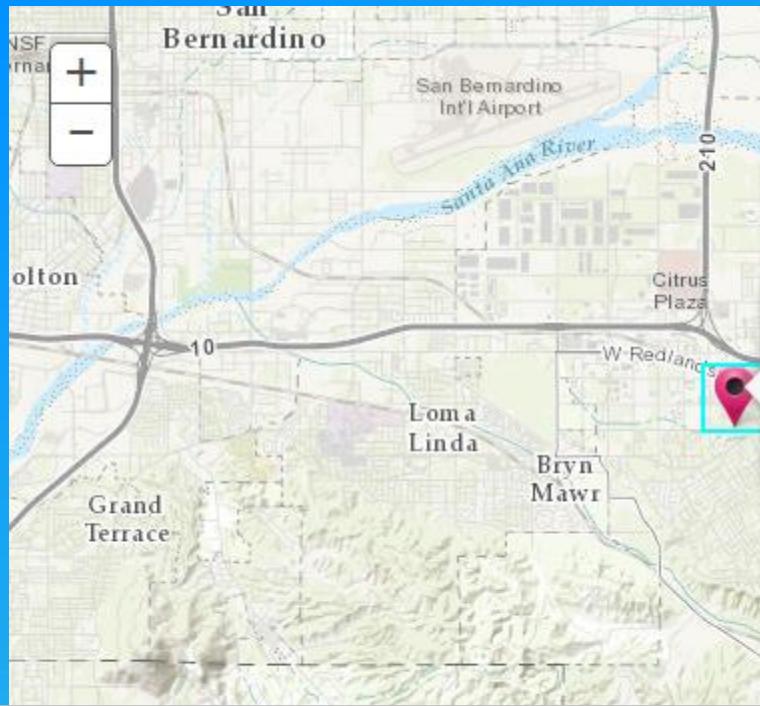
```
interpolated_rf = interpolate_points(rainfall, field='RAINFALL')
```

Geocoding

Geocoding, Batch geocoding, reverse geocoding
`arcgis.geocoding` module

Geocode

Single line or multi field addresses

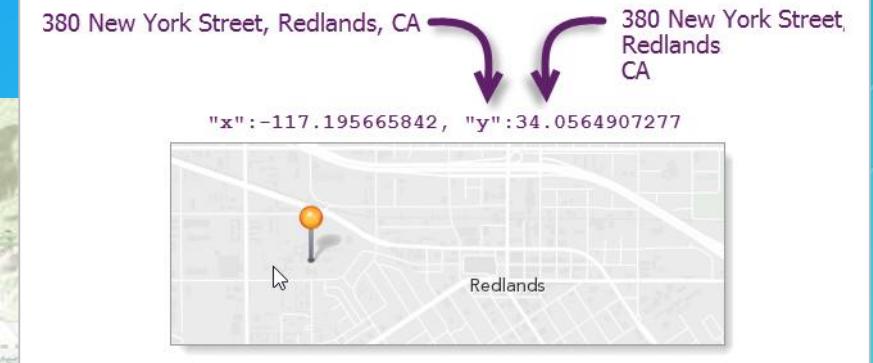


```
multi_field_address = {  
    "Address" : "380 New York Street",  
    "City" : "Redlands",  
    "Region" : "CA",  
    "Postal" : 92373  
}
```



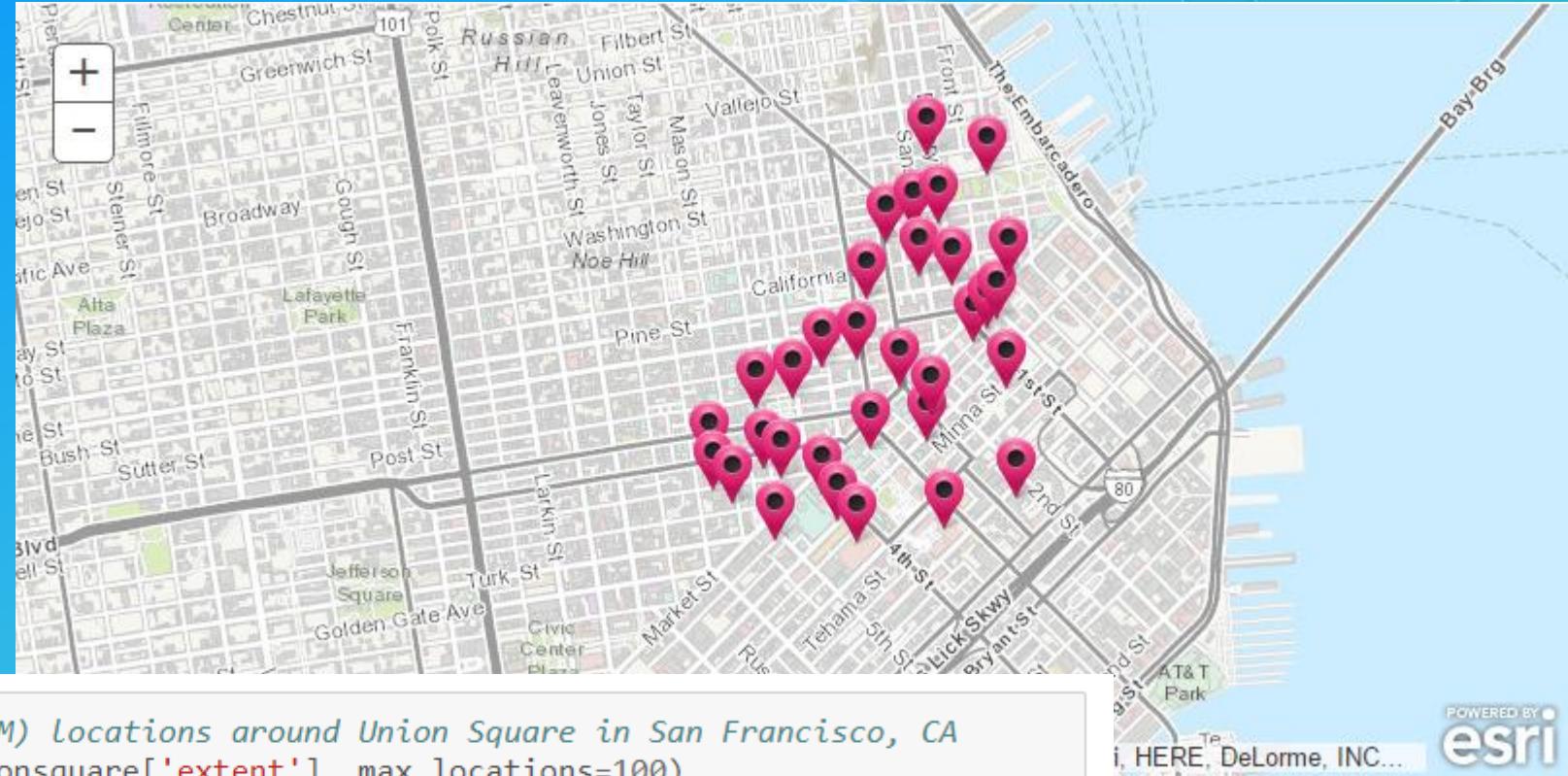
```
single_line_address = "380 New York Street, Redlands, CA 92373"
```

```
# geocode the single line address and plot the location of the first geocode result on the map  
esrihq = geocode(single_line_address)[0]  
  
# add a popup to the matched location  
popup = {  
    "title" : "Esri Headquarters",  
    "content" : esrihq['address']  
}  
map.draw(esrihq['location'], popup)
```



Geocode Points of Interest

- Cultural or geographic landmarks
- Businesses by name or category
- Administrative divisions



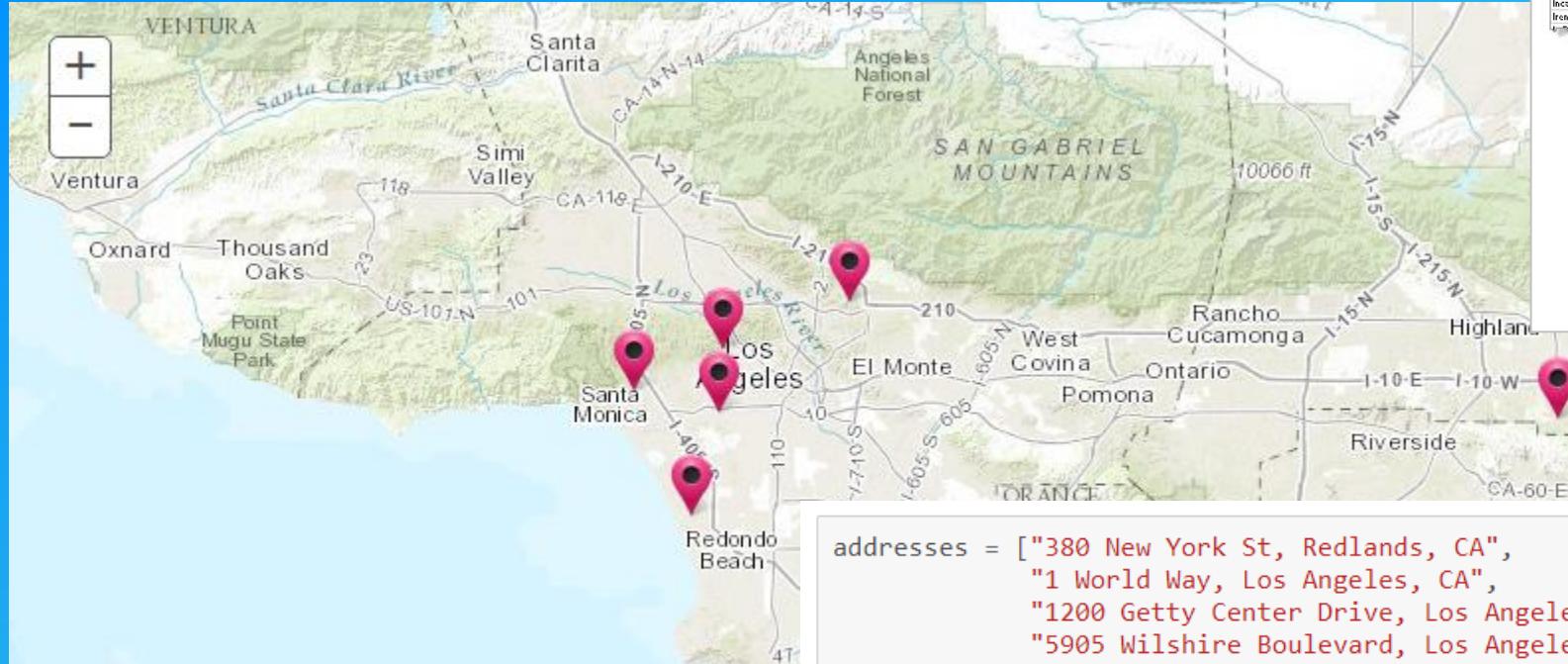
```
# find and plot upto 100 Starbucks(TM) Locations around Union Square in San Francisco, CA
starbucks = geocode("Starbucks", unionsquare['extent'], max_locations=100)
for starbuck in starbucks:
    map.draw(starbuck['location'])
```

POWERED BY
esri

i, HERE, DeLorme, INC...

Batch Geocoding

Geocode an entire list of single line or multi field addresses



Berkenstraat 26, 4462 VX GOES
Borlaagstraat 41, HEERLEN
Bornhoutstraat 28, 4532 AT TERNEUZEN
BEUKBERGENPL 103, 3712 BX HUISTER HEIDE UT
Beverendstraat 1-G2, 5612 BD NUIJENEN
Bremelan 318, 2116 TV BENTVELD
Broekhofstraat 3, 5312 TP VENLO
Gippingstraat 61G, 3026 RJ ROTTERDAM
Gildstraat 18, 5663 CG GELREDOORP
Grijfiersveld 51H, 7501 DV APELDOORN
Houtstraat 10, 3024 XG HAGENBURG
Huizingenstraat 33, 3006 BC ROTTERDAM
Houtwal 2, 3715 BL BARneveld
Inadred 23, 3562 XA Utrecht
Irenestraat 1, 6524 LM ARNHEM
Wijngaardsberg 2, 3142 VH MiddelgaarT



```
addresses = ["380 New York St, Redlands, CA",
             "1 World Way, Los Angeles, CA",
             "1200 Getty Center Drive, Los Angeles, CA",
             "5905 Wilshire Boulevard, Los Angeles, CA",
             "100 Universal City Plaza, Universal City, CA 91608",
             "4800 Oak Grove Dr, Pasadena, CA 91109"]
```

```
results = batch_geocode(addresses)
```

```
for address in results:
    map.draw(address['location'])
```

Reverse Geocode

Determines address at a particular x/y location

```
result = reverse_geocode([4.366281,50.851994], lang_code="fr")
```

```
result
```

```
{'address': {'Address': 'Rue de la Sablonnière 15',
 'City': 'Bruxelles',
 'CountryCode': 'BEL',
 'Loc_name': 'BEL.PointAddress',
 'Match_addr': 'Rue de la Sablonnière 15, 1000, Bruxelles',
 'Neighborhood': 'Bruxelles',
 'Postal': '1000',
 'PostalExt': None,
 'Region': 'Bruxelles',
 'Subregion': 'Bruxelles'},
 'location': {'spatialReference': {'latestWkid': 4326, 'wkid': 4326},
 'x': 4.366265813154625,
 'y': 50.85196404988331}}
```



Network Analysis

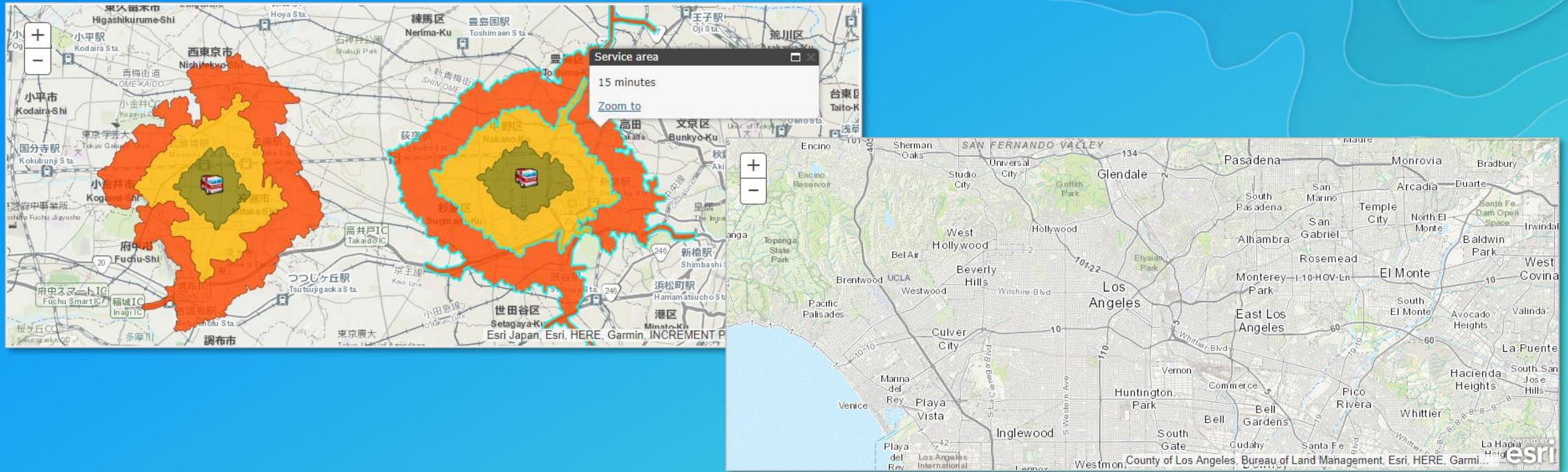
Routing and directions, location allocation, service areas...
`arcgis.network.analysis` module

Network analysis capabilities and tools

Commercial grade, traffic aware routing and directions for multiple travel modes



Network analysis – Service Areas



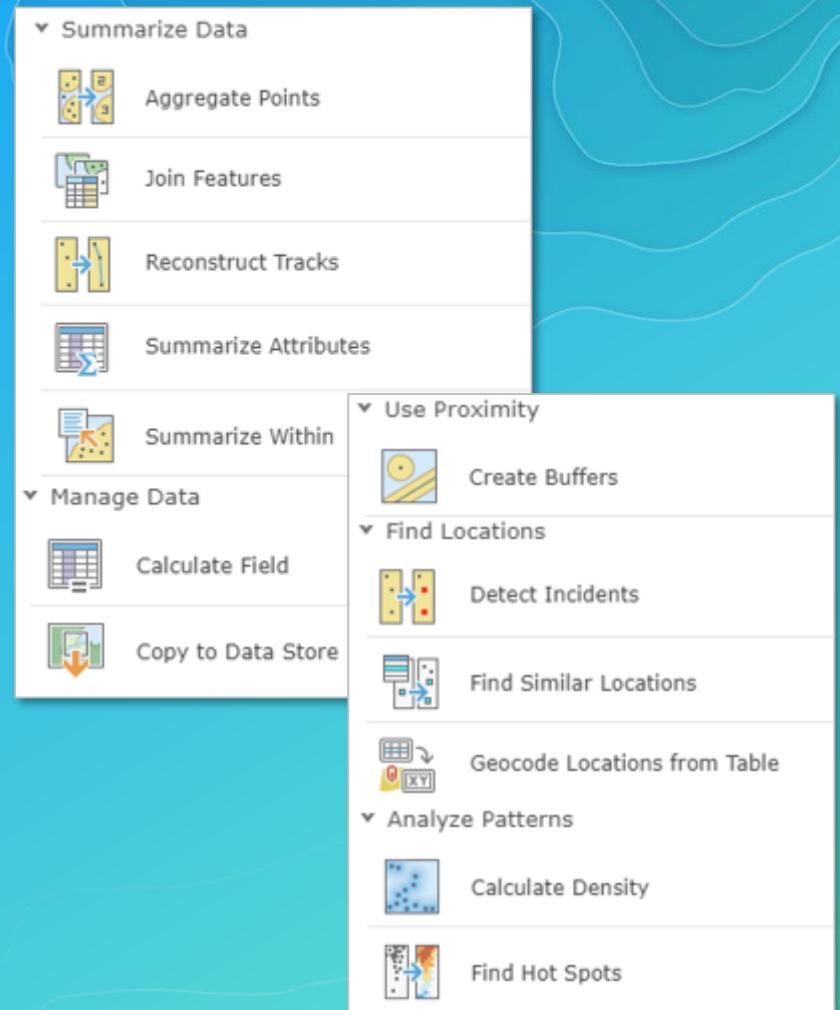
GeoAnalytics

Fast distributed spatio-temporal analysis of large vector and tabular data
`arcgis.geoanalytics` module

GeoAnalytics Tools

Tools that works with tabular data that is usually spatially enabled:

- **Summarize data** – calculate descriptive statistics of features and their attributes within areas or near other features
- **Analyze patterns** – identify, quantify, and visualize spatial patterns
- **Find locations** – identify areas that meet a number of different criteria
- **Use proximity** – answer “what is near what?”
- **Manage data** – day-to-day management of geographic data



The screenshot shows a web browser window for the Houston Police Department's website. The header includes the City of Houston logo and links for Home, I Want To, Govt, Residents, Business, Departments, Visitors, and Espanol. Below the header is a large banner featuring the Houston skyline and the text "Welcome to the City of Houston eGovernment Center". The main content area is titled "POLICE DEPARTMENT" and contains a "Crime Statistics" section with tables for 2015, 2014, 2013, and 2012. Each table row provides links to "Access or Excel" files for each month. To the right of the tables is a sidebar titled "POLICE DEPARTMENT LINKS" containing links to various city services like HoustonPolice.org, Organization, Get Informed, Join Us, Police Stations / Storefronts, Department Phone Directory, File a Report Online, Public Information Request, Multimedia, Registrations, Services, Contact, and Helpful Links.

2015			2014		
January: Access or Excel	February: Access or Excel	March: Access or Excel	January: Access or Excel	February: Access or Excel	March: Access or Excel
April: Access or Excel	May: Access or Excel	June: Access or Excel	April: Access or Excel	May: Access or Excel	June: Access or Excel
July: Access or Excel	August: Access or Excel	September: Access or Excel	July: Access or Excel	August: Access or Excel	September: Access or Excel
October: Access or Excel	November: Access or Excel	December: Access or Excel	October: Access or Excel	November: Access or Excel	December: Access or Excel

2013			2012		
January: Access or Excel	February: Access or Excel	March: Access or Excel	January: Access or Excel	February: Access or Excel	March: Access or Excel
April: Access or Excel	May: Access or Excel	June: Access or Excel	April: Access or Excel	May: Access or Excel	June: Access or Excel

Demo

Analysis of crime patterns in Houston

Attach data

```
In [27]: datastores = arcgis.geoanalytics.get_datastores()
datastores.add_bigdata('Houston_crime_yearly',
                      r'\\teton\\atma_shared\\datasets\\HoustonCrime')
```

Big Data file share exists for Houston_crime_yearly

```
Out[27]: <Datastore title:"/bigDataFileShares/Houston_crime_yearly" type:"bigDataFileShare">
```

```
In [28]: houston_yearly = houston_gis.content.search('Houston_crime_yearly',
                                                    'big data file share')[0]
houston_yearly
```

Out[28]:



bigDataFileShares_Houston_crime_yearly
Big Data File Share by admin
Last Modified: March 03, 2017
0 comments, 0 views

```
In [30]: houston_yearly.layers
```

```
Out[30]: [<Layer url:"https://dev003247.esri.com/gax/rest/services/DataStoreCatalogs/big
DataFileShares_Houston_crime_yearly/BigDataCatalogServer/houstoncrime2010">,
<Layer url:"https://dev003247.esri.com/gax/rest/services/DataStoreCatalogs/big
DataFileShares_Houston_crime_yearly/BigDataCatalogServer/houstoncrime2011">,
<Layer url:"https://dev003247.esri.com/gax/rest/services/DataStoreCatalogs/big
DataFileShares_Houston_crime_yearly/BigDataCatalogServer/houstoncrime2012">]
```

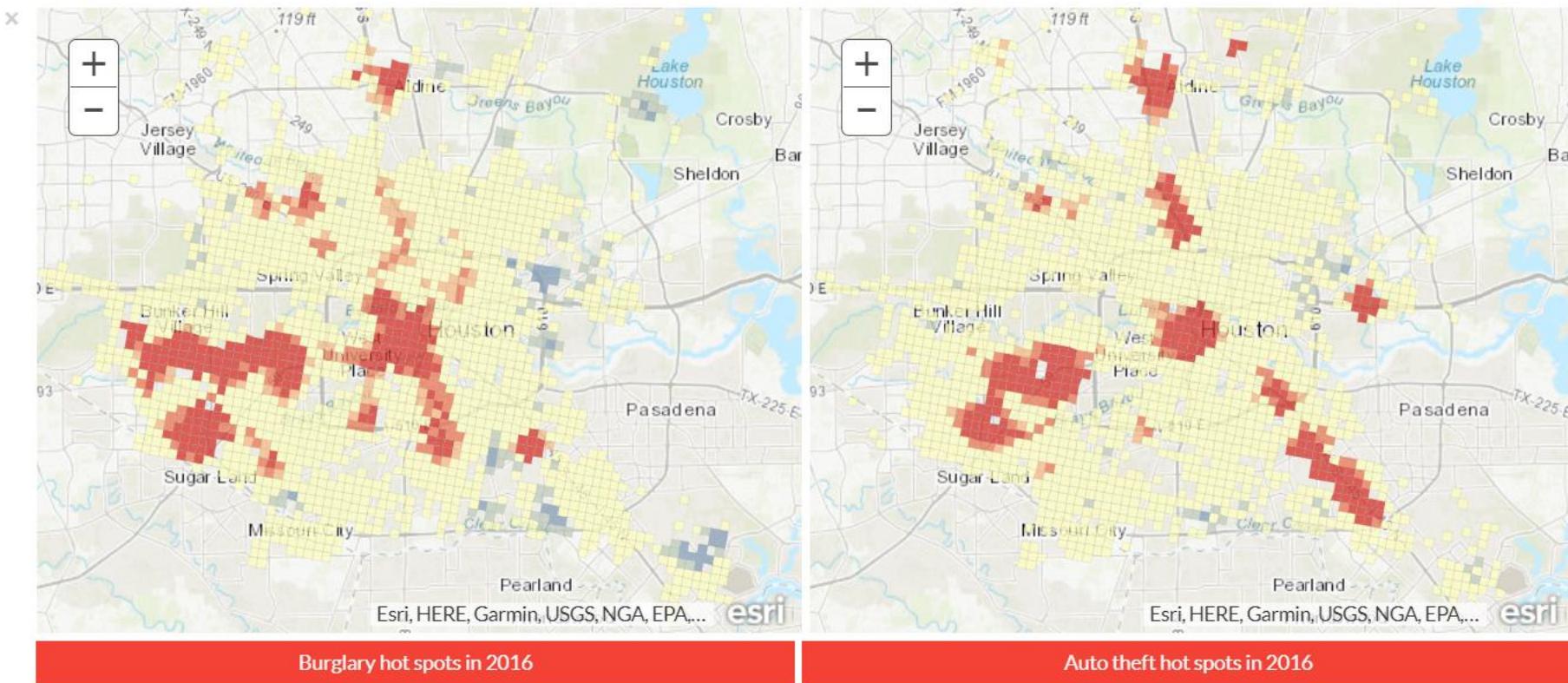
Invoke batch analytics

```
In [ ]: for category in df.Category.unique()[:-1]:  
    lyrid = 0  
    for year in range(2010, 2017):  
        output_name='Houston_' + category.replace(' ', '_') + '_Hotspot_' + str(year)  
        print('Generating ' + output_name)  
        layer = houston_yearly.layers[lyrid]  
        layer.filter = "Category='{}'".format(category)  
  
        find_hot_spots(layer, bin_size=0.5, bin_size_unit='Miles',  
                        neighborhood_distance=1, neighborhood_distance_unit='Mi']  
        output_name=output_name)  
  
    lyrid = lyrid + 1
```

View results

Hot Spots across crime categories

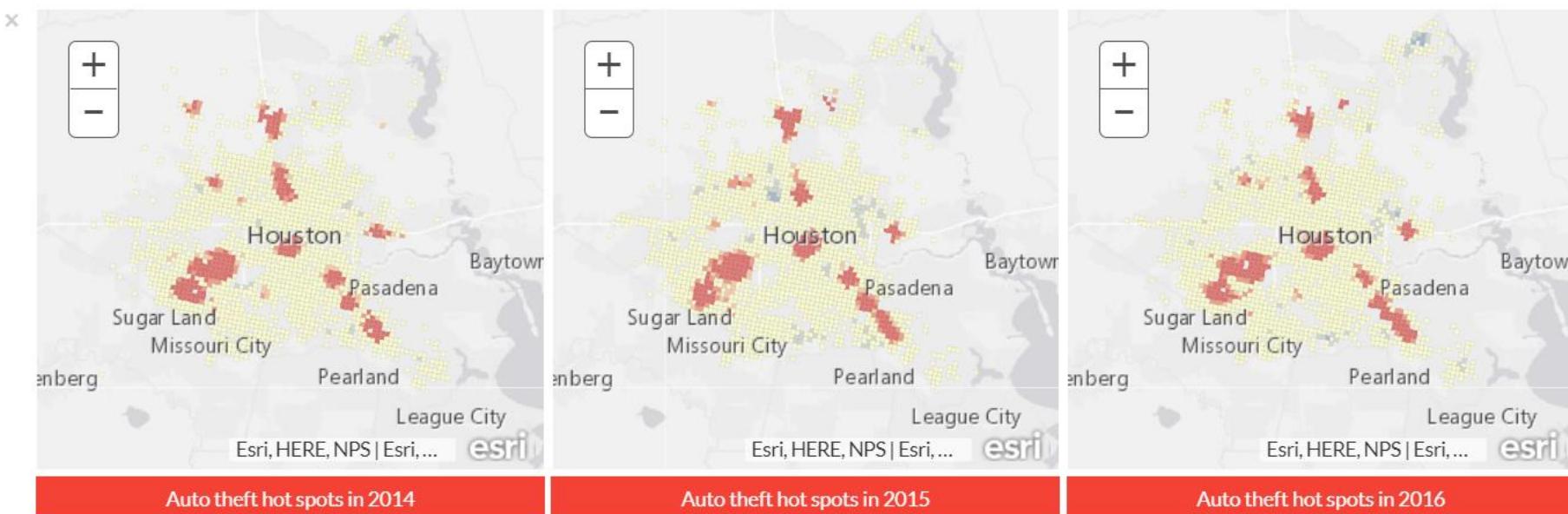
```
In [33]: display(HBox([hotmap1, hotmap2]))  
display(HBox(children=[Button(description='Burglary hot spots in 2016', layout=Layout(width='100%')),  
             Button(description='Auto theft hot spots in 2016', layout=Layout(width='100%'))]))
```



Compare Hot Spots over time

```
In [35]: for year in range(2014, 2017):
    layer = houston_gis.content.search('Houston_Auto_Theft_Hotspot_' + str(year))
    hotspotmap = houston_gis.map(houston)
    hotspotmap.add_layer(layer)
    hotspotmap.layout=Layout(flex='1 1', padding='3px')
    maps.append(hotspotmap)
    hotspotmap.basemap='gray'
    labels.append(Button(description='Auto theft hot spots in ' + str(year),
                         layout=items_layout, button_style='danger'))

display(HBox([maps[0], maps[1], maps[2]], layout=layout))
display(HBox(children=labels, layout=Layout(width='100%')))
```



GeoEnrichment

Enrich your analysis with demographic and business data
`arcgis.geoenrichment` module

GeoEnrichment

- Get facts about a location or area
 - Street addresses
 - Points, lines and polygon geometries
 - Within a drive time or service area
 - Named geographical areas
 - Counties or block groups in California
 - Districts and subdistricts in India
- Create charts and choropleth maps

```
df = usa.data_collections

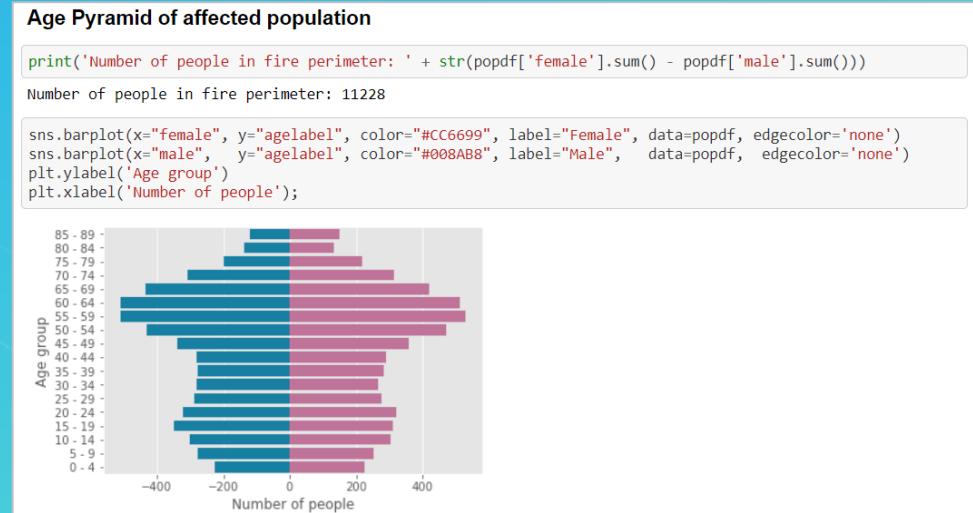
# print a few rows of the DataFrame
df.head()
```

	analysisVariable	alias	fieldCategory	vintage
dataCollectionID				
1yearincrements	1yearincrements.AGE0_CY	2017 Population Age <1	2017 Age: 1 Year Increments (Esri)	2017
1yearincrements	1yearincrements.AGE1_CY	2017 Population Age 1	2017 Age: 1 Year Increments (Esri)	2017
1yearincrements	1yearincrements.AGE2_CY	2017 Population Age 2	2017 Age: 1 Year Increments (Esri)	2017

```
ca_counties = usa.subgeographies.states['California'].counties

counties_df = enrich(study_areas=ca_counties, data_collections=['Age'])
counties_df.head(10)
```

	FEM0	FEM10	FEM15	FEM20	FEM25	FEM30	FEM35	FEM40	FEM45	FEM5	...	MALE75	MALE80
0	47323	50068	50590	57125	60205	61600	59673	56910	56069	49093	...	16318	10422
1	24	33	29	20	24	24	23	25	34	33	...	17	10
2	706	871	867	725	747	790	809	821	1019	769	...	789	510

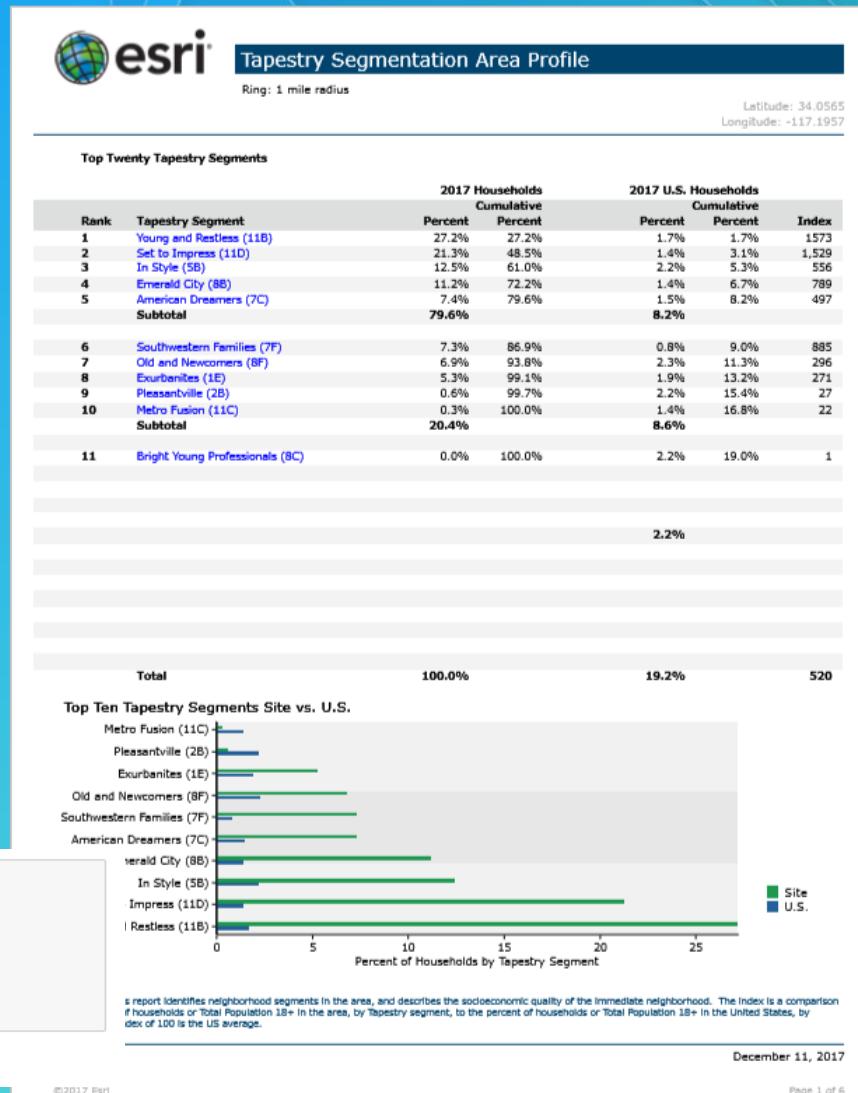


GeoEnrichment – create reports

```
# print a sample of the reports available for USA  
usa.reports.head(10)
```

	id	title	categories	formats
0	census2010_profile	2010 Census Profile	[Demographics]	[pdf, xlsx]
1	acs_housing	ACS Housing Summary	[Demographics]	[pdf, xlsx]
2	acs_population	ACS Population Summary	[Demographics]	[pdf, xlsx]
3	55plus	Age 50+ Profile	[Demographics]	[pdf, xlsx]
4	agesexrace	Age by Sex by Race Profile	[Demographics]	[pdf, xlsx]
5	agesex	Age by Sex Profile	[Demographics]	[pdf, xlsx]
6	cex_auto	Automotive Aftermarket Expenditures	[Consumer Spending]	[pdf, xlsx]
7	business_loc	Business Locator	[Business]	[pdf, xlsx]
8	business_summary	Business Summary	[Business]	[pdf, xlsx]
9	community_profile	Community Profile	[Demographics]	[pdf, xlsx]

```
report = create_report(study_areas=["380 New York Street, Redlands, CA"],  
                      report="tapestry_profileNEW",  
                      export_format="PDF",  
                      out_folder=r"c:\xc", out_name="esri_tapestry_profile.pdf")  
  
report  
  
'c:\\xc\\esri_tapestry_profile.pdf'
```



Imagery and Raster Analysis

On-the-fly and distributed batch analysis of raster data
`arcgis.raster` module

Imagery and Raster Analysis

```
In [3]: landsat_item = gis.content.search('title:Multispectral Landsat',
                                         'Imagery Layer', outside_org=True)[0]
```

```
In [4]: landsat_item
```

Out[4]:



Multispectral Landsat

Landsat 8 OLI, 30m Multispectral 8 band scenes with visual renderings and indices. Updated daily. Based on the Landsat on AWS collections.



Imagery Layer by esri

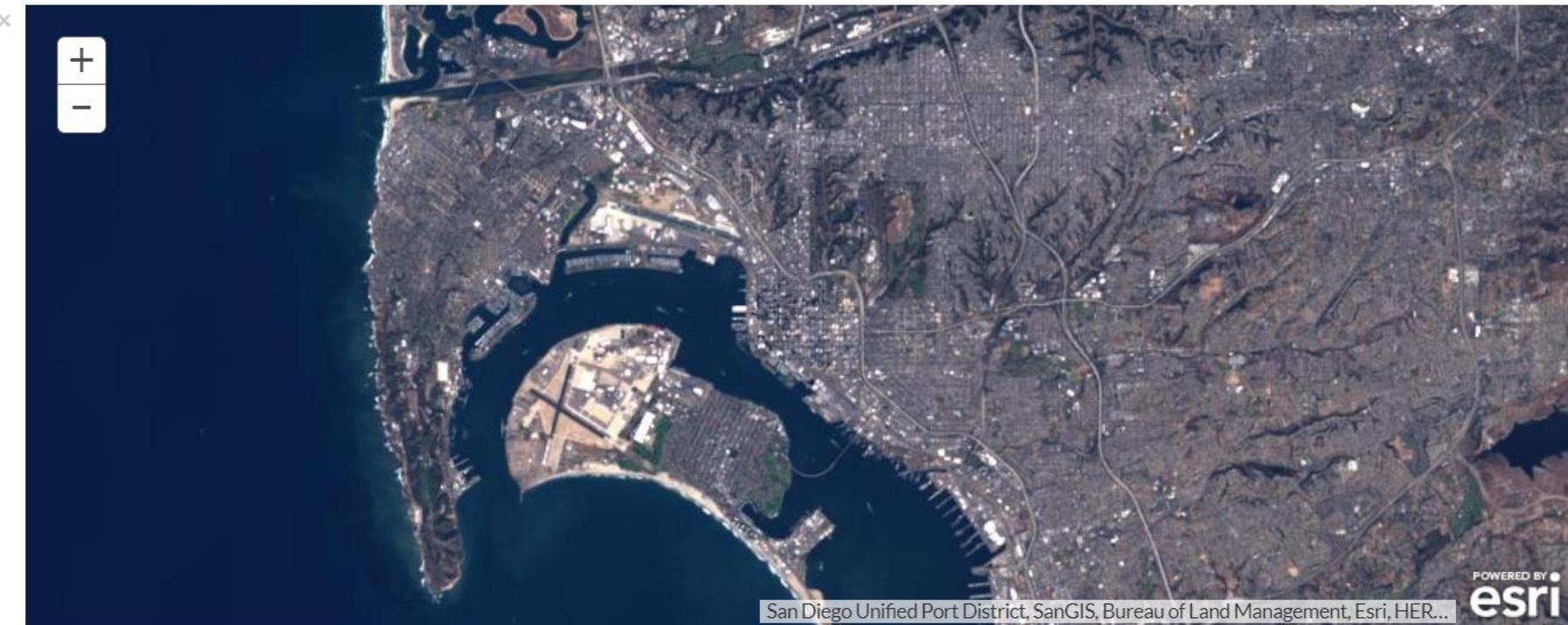
Last Modified: December 07, 2016

0 comments, 118,273 views

```
In [5]: landsat = landsat_item.layers[0]
```

Visualizing imagery layers

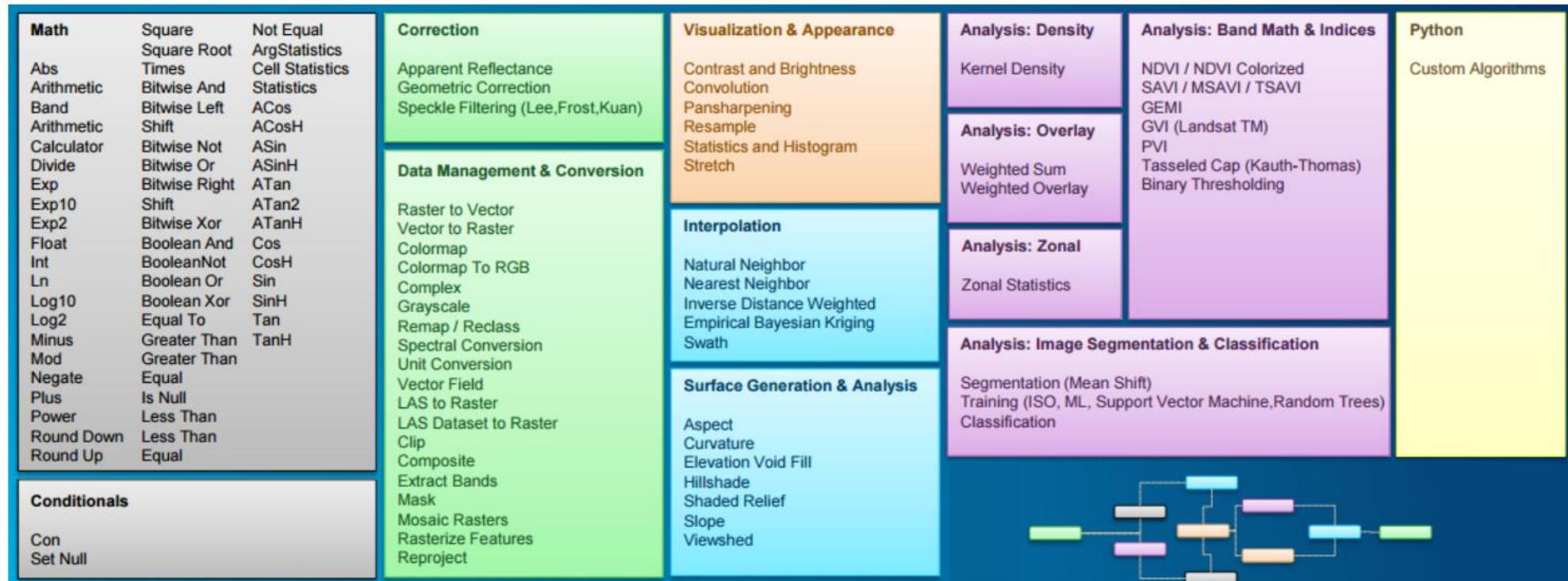
```
In [6]: imagery_map = gis.map('San Diego, CA', zoomlevel=12)
imagery_map.add_layer(landsat)
imagery_map
```



```
In [27]: for rasterfunc in landsat.properties.rasterFunctionInfos:
    print(rasterfunc.name)
    imagery_map.add_layer(apply(landsat, rasterfunc.name))
    time.sleep(2)
```

Agriculture with DRA
Bathymetric with DRA

Custom raster processing using Raster Functions



Running in San Diego

Least Important

20%



Low Elevation

Moderately Important

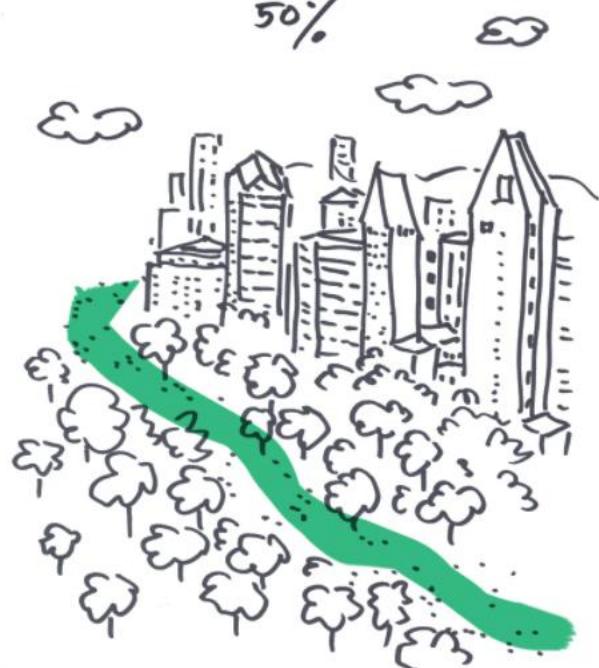
30%



Flat, Not Hilly

Most Important

50%



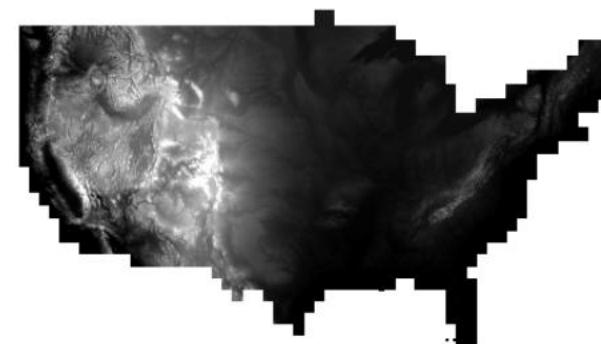
Natural, Not Built

Inputs - Elevation

In [8]: *# Digital elevation model for the US*

```
elevation_item = enterprise.content.search('elevation_270m')[0]
elevation_lyr = elevation_item.layers[0]
elevation_lyr
```

Out[8]:



Natural areas

```
In [9]: # Human Modified Index imagery Layer  
# This dataset is based on research on the degree of human modification to  
# the Landscape, on a scale of 0 - 1, where 0.0 indicates unmodified natural  
# Landscape and 1.0 indicates the Landscape is completely modified by human aci  
  
naturalareas_item = enterprise.content.search('human_modification_index')[0]  
naturalareas_lyr = naturalareas_item.layers[0]  
naturalareas_lyr
```

Out[9]:



Interactive raster processing in Jupyter Notebook

```
In [12]: clipped_elev = clip(elevation_lyr, sd_geom)  
clipped_elev
```

Out[12]:

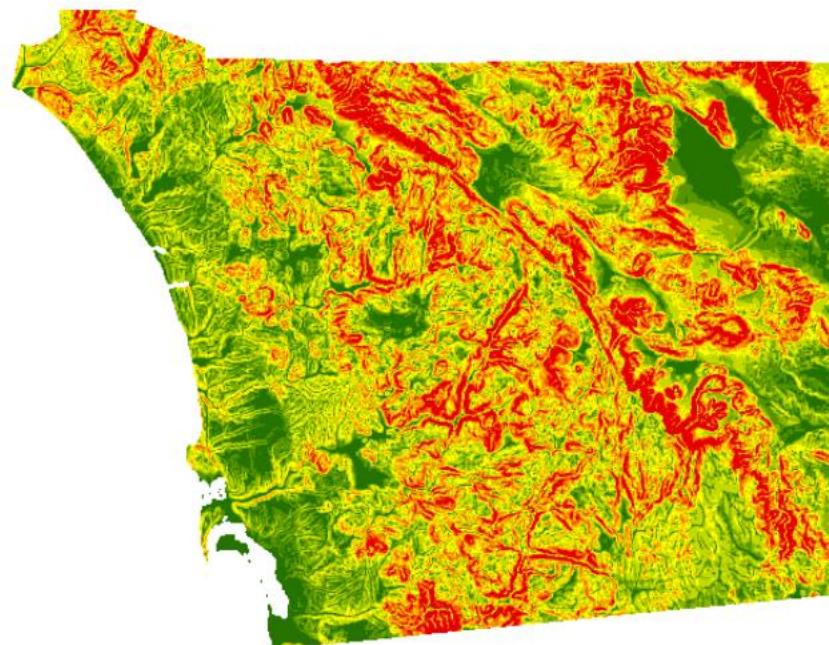


Chaining raster functions

```
In [14]: output_values = [1,2,3,4,5,6,7,8,9]

colormap(remap(slope(clipped_elev,
                     slope_type='DEGREE',
                     z_factor=1),
               input_ranges=[0,1, 1,2, 2,3, 3,5, 5,7, 7,9, 9,12, 12,15, 15,100],
               output_values=output_values),
          colormap=red_green)
```

Out[14]:



Prepare input layers

```
In [17]: elevation = remap(elevation_lyr,  
                         [-90,250, 250,500, 500,750, 750,1000, 1000,1500, 1500,2000],  
                         output_values)
```

```
In [18]: terrain = remap(slope(elevation_lyr, slope_type='DEGREE', z_factor=1), # Slope  
                         [0,1, 1,2, 2,3, 3,5, 5,7, 7,9, 9,12, 12,15, 15,100],  
                         output_values)
```

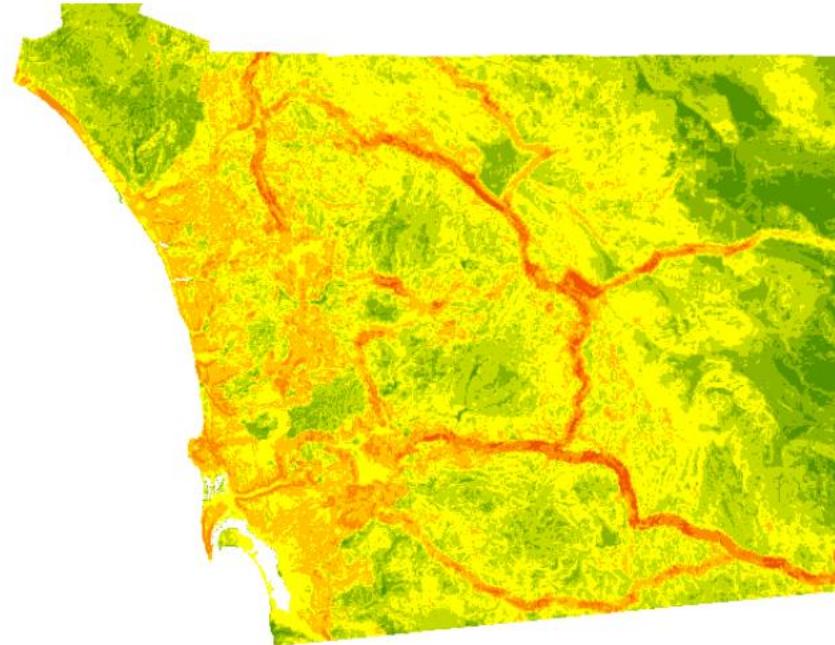
```
In [19]: natural_areas = remap(naturalareas_lyr,  
                           [0.0,0.1, 0.1,0.2, 0.2,0.3, 0.3,0.4, 0.4,0.5,0.5,0.6, 0.6,  
                           output_values])
```

Map Algebra for the Web GIS

```
In [20]: result = 0.2*elevation + 0.3*terrain + 0.5*natural_areas
```

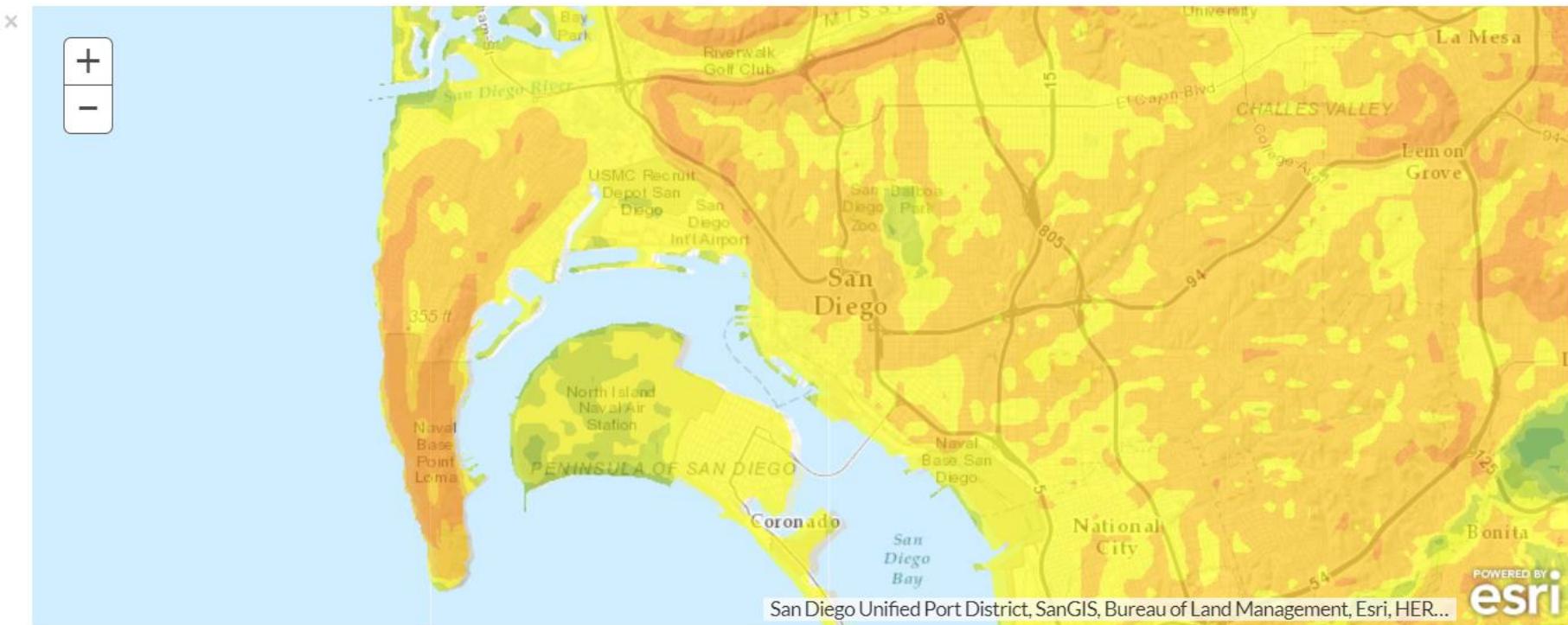
```
In [21]: run_raster = colormap(clip(result, sd_geom), colormap=red_green)  
run_raster
```

Out[21]:



Visualize results using map widget

```
In [22]: surface_map = gis.map('San Diego, CA', zoomlevel=12)  
surface_map
```



```
In [23]: surface_map.add_layer(run_raster, {'opacity': 0.6})
```

Persist results as an imagery layer

```
In [24]: # Generate a persistent result at source resolution using Raster Analytics  
resultlyr = run_raster.save('SanDiego_PlacesToRun')
```

```
In [26]: resultlyr
```

Out[26]:



[SanDiego_PlacesToRun](#)

Analysis Image Service generated from GenerateRaster



Imagery Layer by arcgis_python

Last Modified: July 06, 2017

0 comments, 0 views

Cross country mobility

Find **most efficient paths** for off-road vehicles

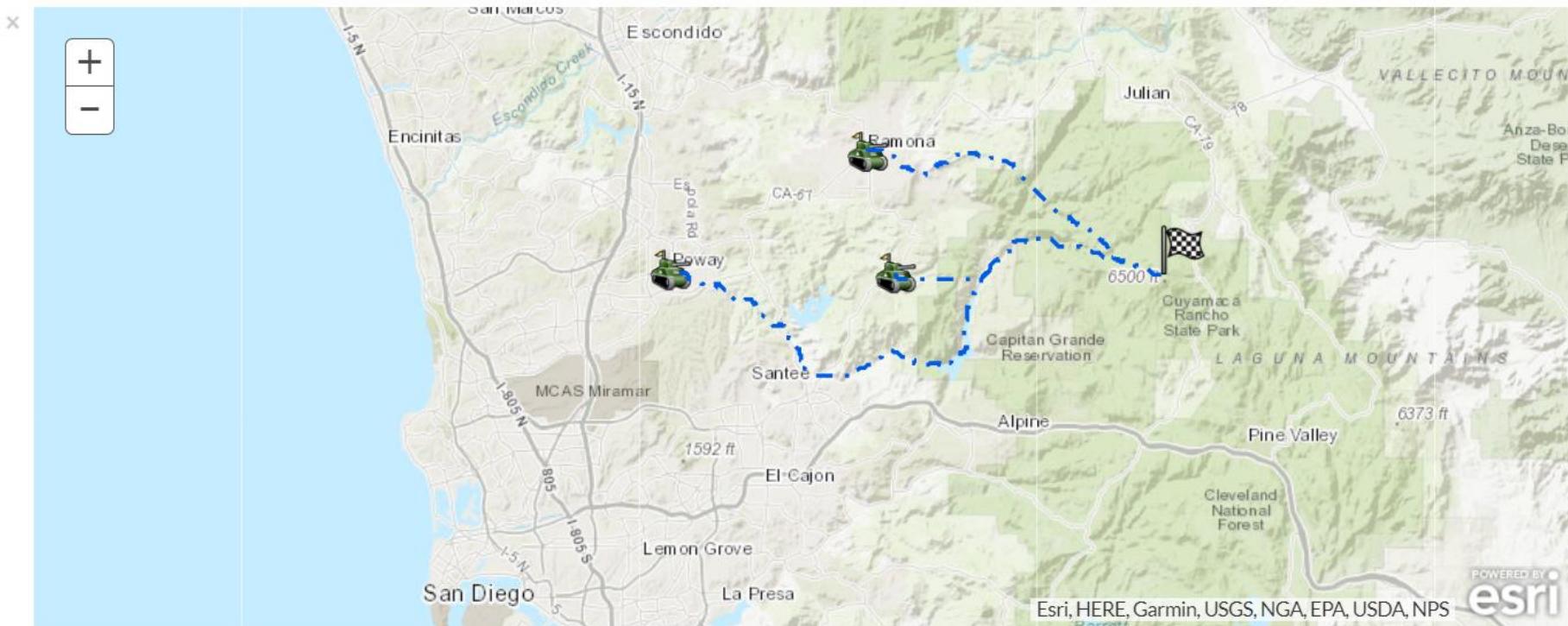


Use Geoprocessing for offroad routing

```
In [25]: from arcgis.geoprocessing import import_toolbox  
  
ccmurl='https://maps.esri.com/apl3/rest/services/LCP/LCP/GPServer/LeastCostPath  
ccm = import_toolbox(ccmurl)
```

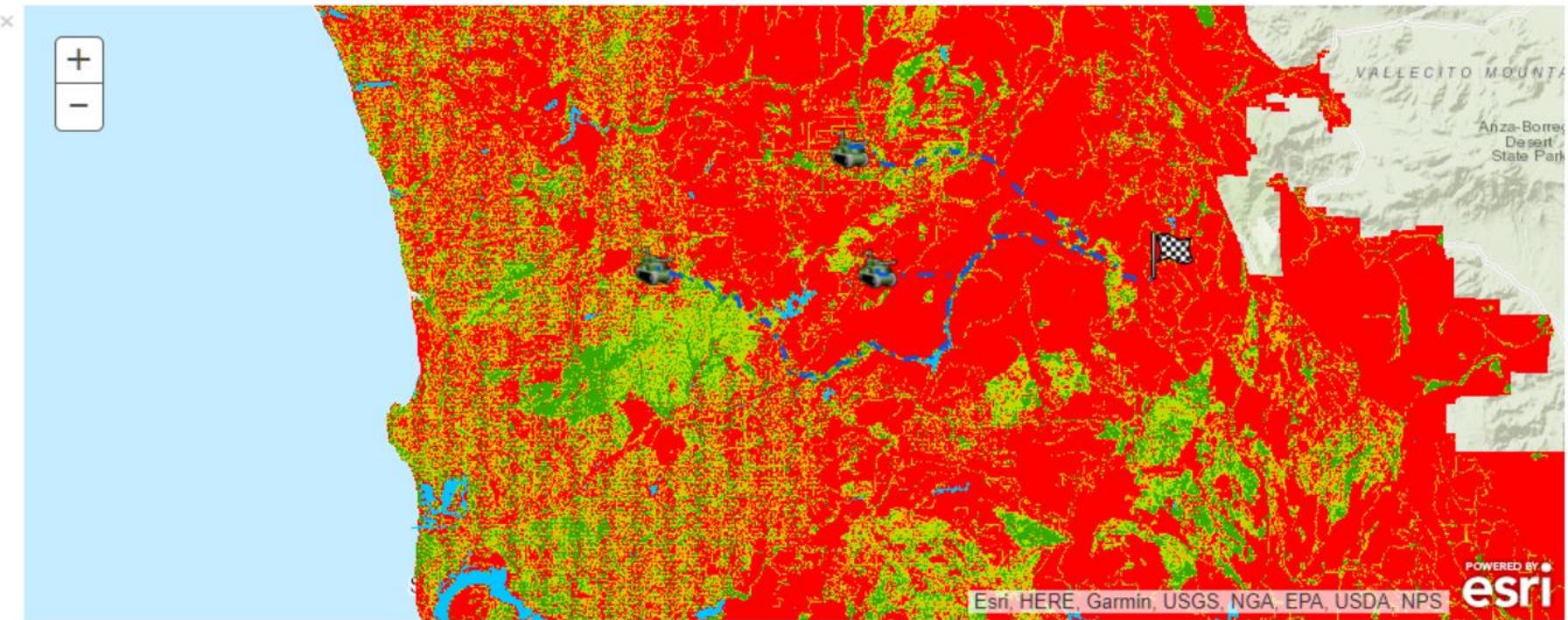
```
In [26]: def find_path(m, pt):  
    m.draw(pt, symbol=finish_symbol)  
    paths = ccm.least_cost_path(destination=FeatureSet([Feature(pt)]),  
                                origins=origins)  
    m.draw(paths, symbol=dash_dot)
```

```
In [28]: ccm_map = gis.map('San Vicente Reservoir', zoomlevel=10)  
ccm_map.on_click(find_path)  
ccm_map
```



```
In [29]: ramona = geocode("Ramona, CA")[0]  
poway = geocode("Poway, CA")[0]  
barona = geocode("Barona Reservation, CA")[0]  
  
ccm_map.draw(ramona, symbol=tank_symbol)  
ccm_map.draw(poway, symbol=tank_symbol)  
ccm_map.draw(barona, symbol=tank_symbol)
```

```
In [35]: ccm_map.add_layer(surface)
```



Inputs

Which factors affect cross country mobility?

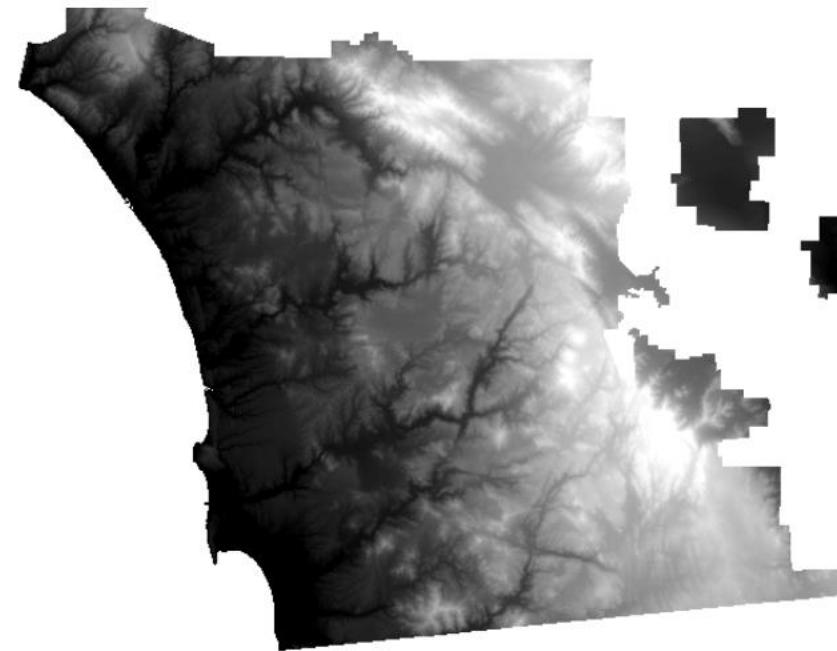
Terrain

Flat, rolling or steep?

```
In [34]: dtm_sd = enterprise_b.content.search('DTM_SD')[0]
elevation = dtm_sd.layers[0]
elevation.extent = sd_extent

elevation
```

Out[34]:



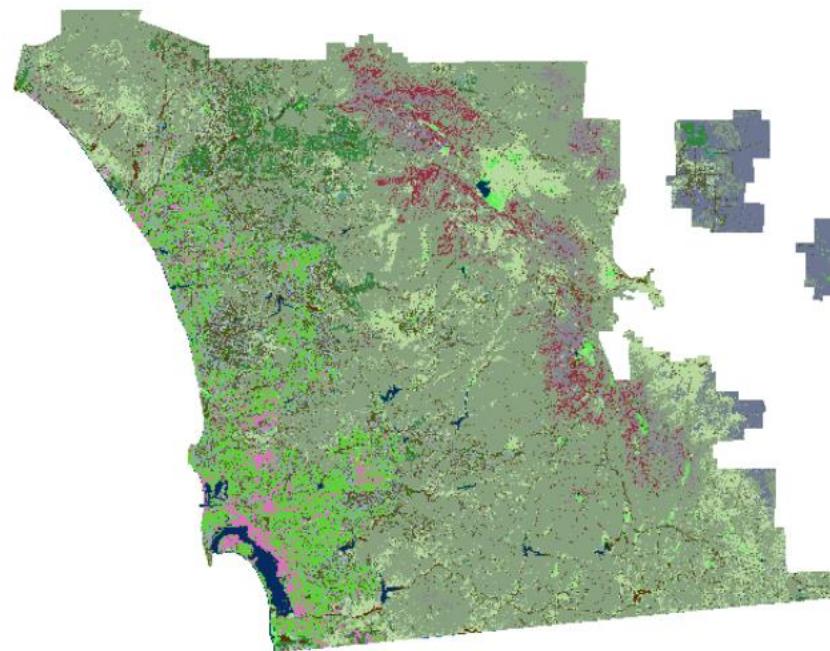
Land Cover

Barren, developed or cultivated?

```
In [35]: nlcd_sd = enterprise_b.content.search('NLCD_SD')[0]
land_cover = nlcd_sd.layers[0]
land_cover.extent = sd_extent

land_cover
```

Out[35]:



Vegetation characteristics

Forest, shrub or pasture?

```
In [36]: vegetation = pd.read_csv('CCM/LUTables/veg.csv')  
vegetation
```

Out[36]:

	mapunit	mapdesc	ttadbmappingunit	stemd	stems	vr	v1	v2	f3
0	42	Evergreen Forest	C	0.183	3.030	0.60	NaN	NaN	0.60
1	43	Mixed Forest	E	0.167	3.515	0.60	NaN	NaN	0.60
2	52	Shrub/Scrub	B1, B2, H	NaN	NaN	0.68	NaN	NaN	0.68
3	71	Grassland/Herbaceous	G1, G2	NaN	NaN	0.75	NaN	NaN	0.75
4	81	Pasture/Hay	A1	NaN	NaN	0.80	NaN	NaN	0.80
5	82	Cultivated Crops	A2, A3, A5, A6, A7, FC, FD, FE, L	NaN	NaN	0.56	NaN	NaN	0.56
6	11	Open Water	W	NaN	NaN	0.00	NaN	NaN	0.00
7	21	Developed, Open Space	X	NaN	NaN	0.00	NaN	NaN	0.00
8	22	Developed, Low Intensity	X	NaN	NaN	0.00	NaN	NaN	0.00
9	23	Developed, Medium Intensity	X	NaN	NaN	0.00	NaN	NaN	0.00
10	24	Developed High Intensity	X	NaN	NaN	0.00	NaN	NaN	0.00
11	31	Barren Land (Rock/Sand/Clay)	N	NaN	NaN	1.00	NaN	NaN	1.00
12	41	Deciduous Forest	D	0.150	4.000	0.60	NaN	NaN	0.60
13	90	Woody Wetlands	IC, ID, IE	NaN	NaN	0.40	NaN	NaN	0.40
14	95	Emergent Herbaceous Wetlands	J, K	NaN	NaN	0.55	NaN	NaN	0.55

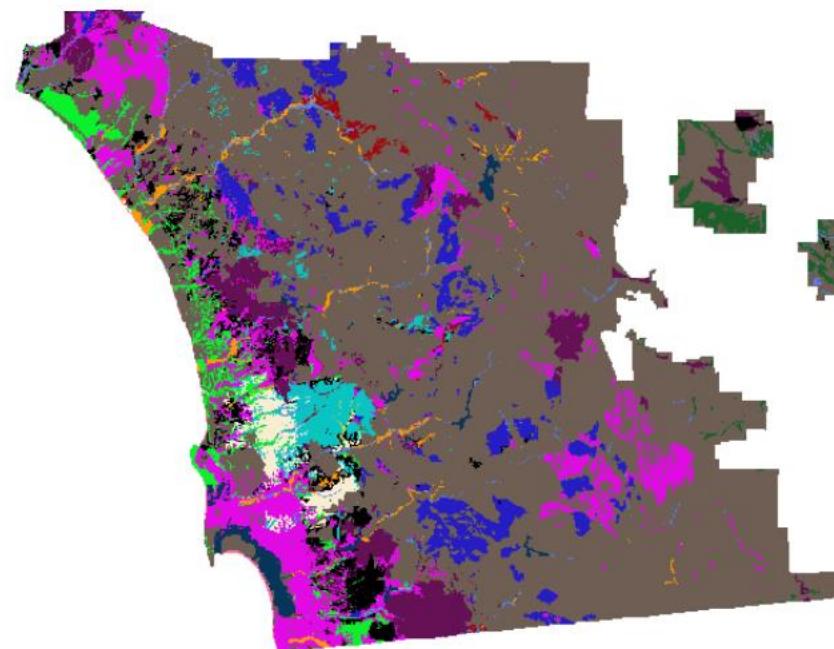
Soil Type

Rock, clay or sand?

```
In [37]: soils_sd = enterprise_b.content.search('Soils_SDv3')[0]
soils = soils_sd.layers[0]

soils
```

Out[37]:



Transportation infrastructure

In [38]:

```
transportation = enterprise_b.content.search('RoadsAndRails')[0]
transportation
```

Out[38]:



RoadsAndRails_SD

Binary Roads and Rails for SD county



Imagery Layer by bgerltRA

Last Modified: July 18, 2017

0 comments, 4 views

Vehicle Characteristics

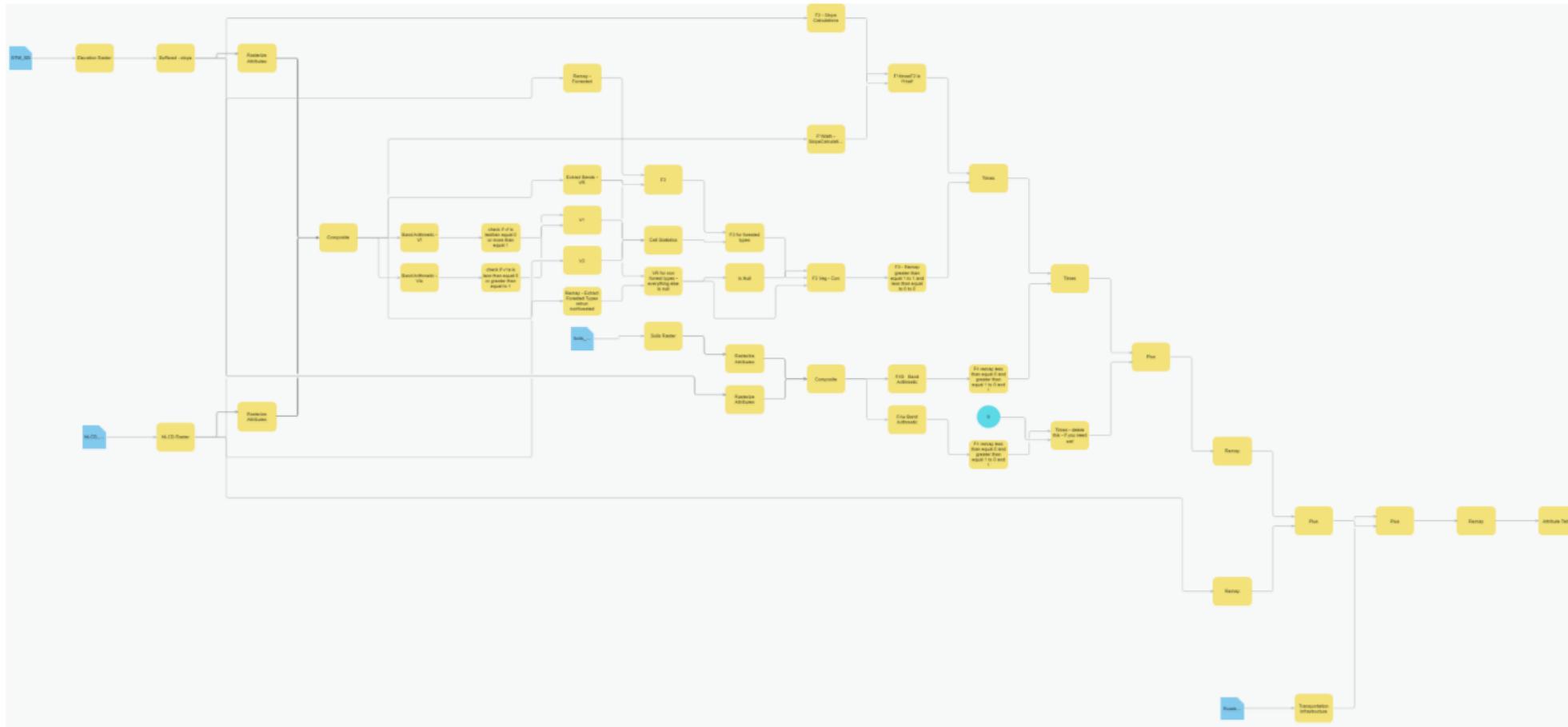
Jeep, truck or tank?

```
In [39]: import pandas as pd  
vehicle_characteristics = pd.read_csv('CCM/LUTables/VCTable.csv')  
vehicle_characteristics
```

Out[39]:

	name	maxkph	onslope	offslope	vehiclewidth	overridediameter	vci1	vci50	minturnrad	vehiclelength	mle
0	M1	71.0	68.7	53.0	3.65	0.25	25.0	58.0	9.90	9.90	60.0
1	M60A1	48.0	60.0	45.0	3.63	0.15	20.0	48.0	9.40	9.40	54.0
2	M109	56.0	60.0	45.0	3.10	0.12	25.0	57.0	6.60	6.60	24.0
3	M113	48.0	60.0	45.0	2.69	0.10	17.0	40.0	4.80	4.80	12.0
4	M35A2	56.0	64.0	30.0	2.43	0.06	26.0	59.0	5.30	6.70	10.0
5	M151	50.0	60.0	28.0	1.69	0.04	19.0	44.0	5.80	3.35	NaN
6	T62	50.0	62.0	45.0	3.37	0.15	21.0	49.0	9.33	9.33	42.0
7	T72	60.0	62.0	45.0	3.38	0.18	25.0	60.0	9.20	9.20	45.0

Raster function chain for cross country mobility



Generate cost surface

```
In [54]: with open("CCM_FunctionChain.rft.xml", "r", encoding='utf-8-sig') as rft:  
    raster_fn = rft.read()
```

```
In [55]: %%time  
from arcgis.raster.analytics import generate_raster  
  
surface = generate_raster(raster_fn, output_name='Cross_Country_Mobility')
```

Wall time: 5min 48s

```
In [56]: surface
```

Out[56]:



[Cross_Country_Mobility](#)

Analysis Image Service generated from GenerateRaster



Imagery Layer by rsinghRA

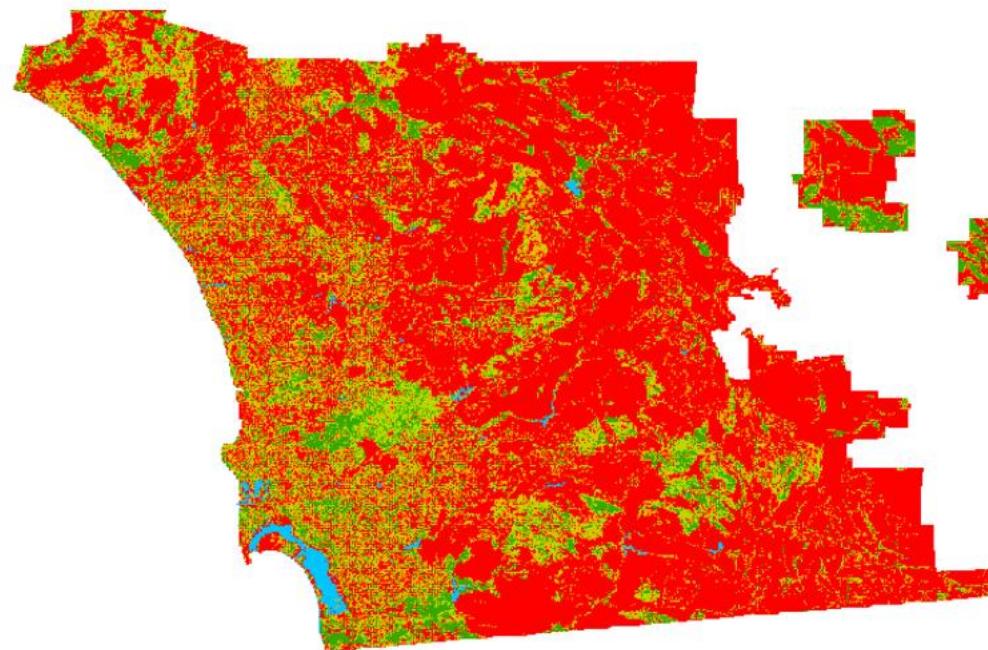
Last Modified: October 22, 2017

0 comments, 0 views

```
In [57]: cost_surface = surface.layers[0]
cost_surface.extent = sd_extent

cost_surface
```

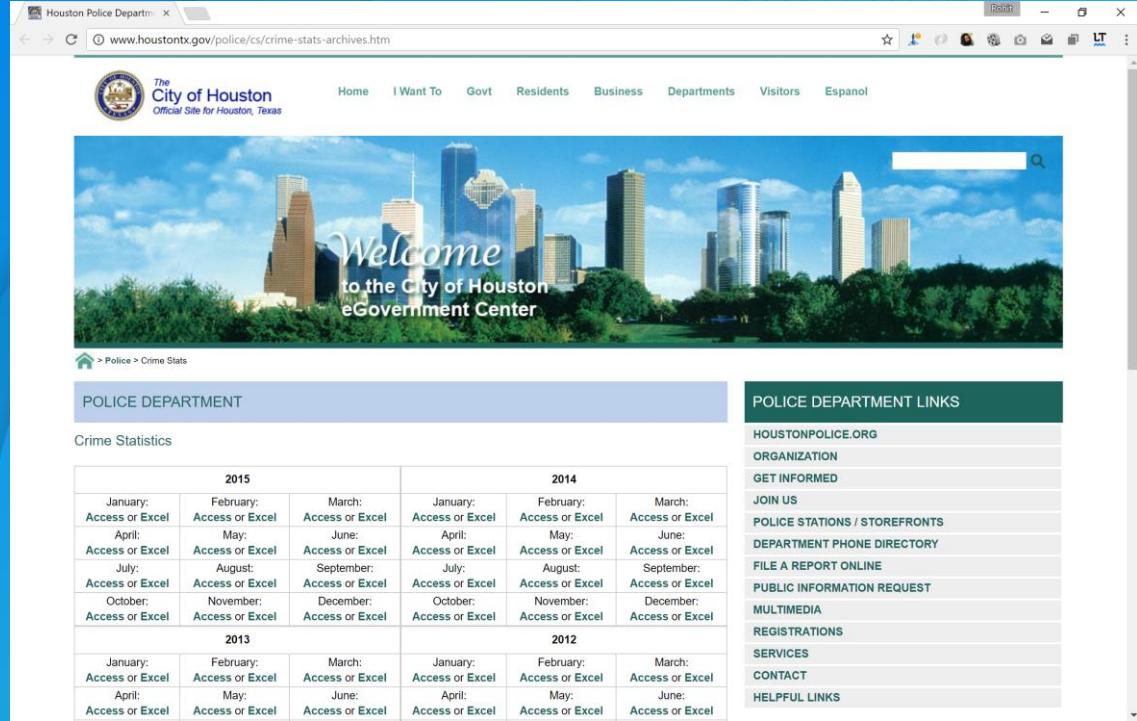
Out[57]:



Spatial Data Frame

Feature layers as a Pandas dataframe

`arcgis.features.SpatialDataFrame` class



Demo

Using SpatialDataFrame

Data Science

Find the patterns hidden in data

Data Science with ArcGIS - Data

- Esri curated content – Living Atlas
 - Multi-spectral, temporal, dynamic imagery layers
 - Landsat, NAIP, MODIS, Elevation
 - Basemaps, Imagery, Demographics, Transport
 - Boundaries & places, Landscape, Oceans
 - Earth Observations, Urban Systems, Historical Maps, ...
- Your data, org's data, data shared with you
 - Shapefiles, File geodatabase, CSV, Excel, HTML, ...
 - File shares, cloud share
 - HDFS, Hive and databases
- Public data
 - Maps, layers and datasets shared by users worldwide

A screenshot of a web-based GIS application interface showing search results for datasets titled "SAM_Sites". The results are displayed in three cards:

- Final Site Sam Schoenmann**
Stowe Final Site
Feature Layer Collection by Schoes34
Last Modified: November 18, 2016
0 comments, 12 views
- SAM_Site99**
SAM sites
Feature Layer Collection by dbooth_TechsupportUK
Last Modified: September 08, 2017
0 comments, 110 views
- Sam Sites**
sam sites
Web Map by deldev
Last Modified: February 08, 2018
0 comments, 0 views

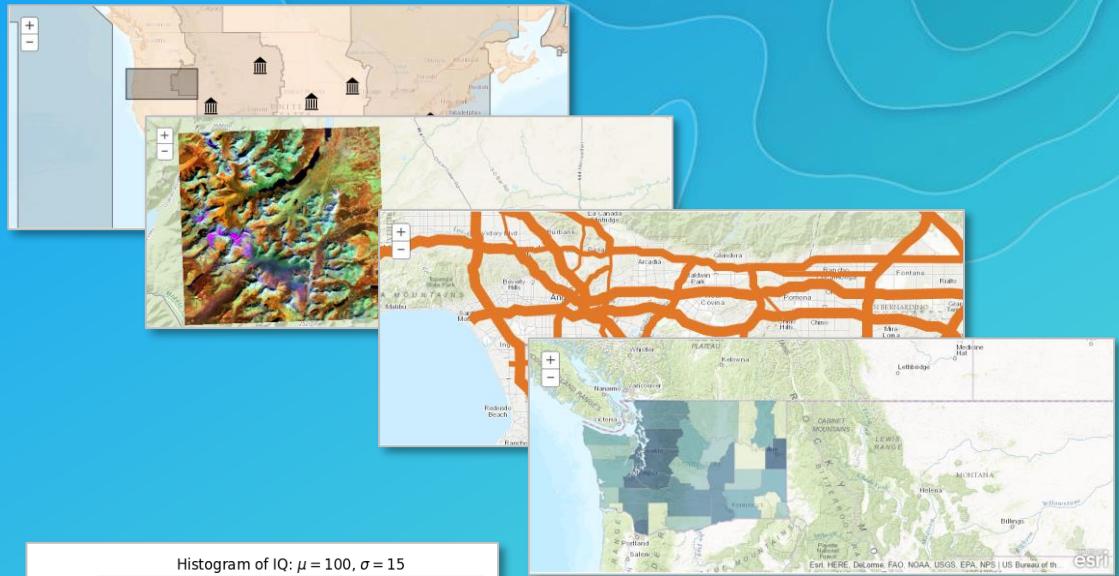
At the top left, there is a snippet of Python code:

```
gis = GIS(profile='deldev')  
  
items = gis.content.search('title:"SAM_Sites"', outside_org=True)  
for item in items:  
    display(item)
```

Data Science with ArcGIS - Visualization

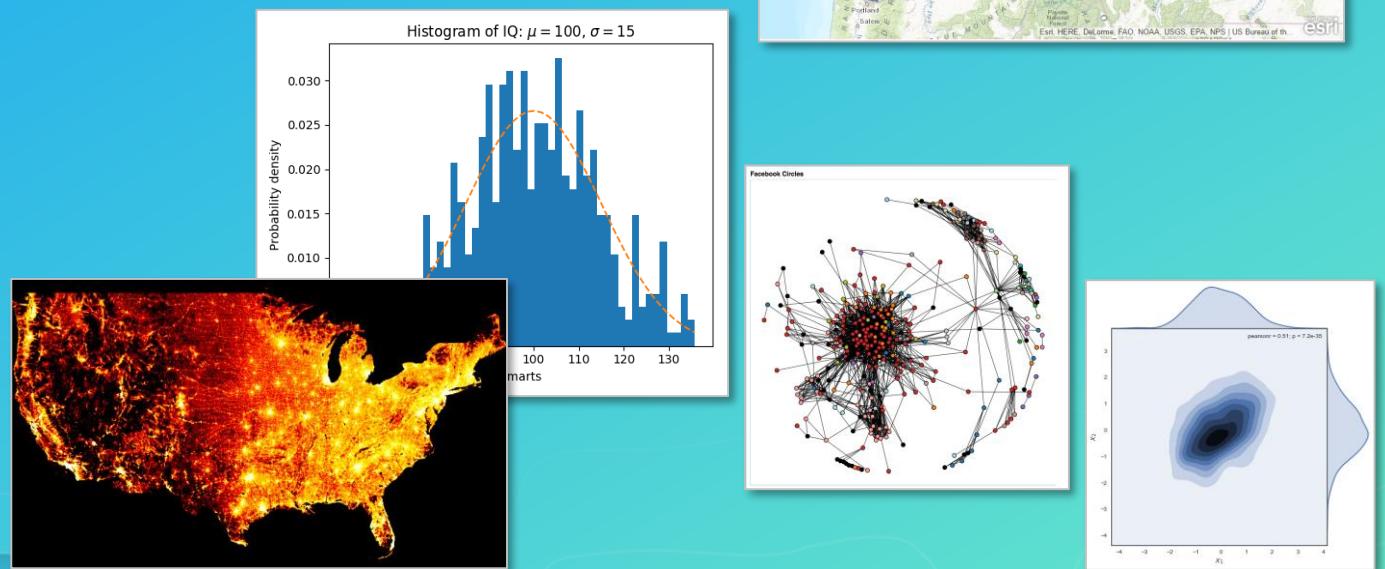
- Visualize with ArcGIS

- Map widget in Jupyter notebook
- Web Maps and Web Scene
- Feature layers
- Raster and imagery layer
- Smart mapping
- Pythonic renderers and symbology



- Visualize with Python

- Matplotlib, Seaborn, Bokeh, Plotly, ...
- Datashader, Holoviews, Mayavi, ...



Data Science with ArcGIS - Analysis

- Analysis with ArcGIS

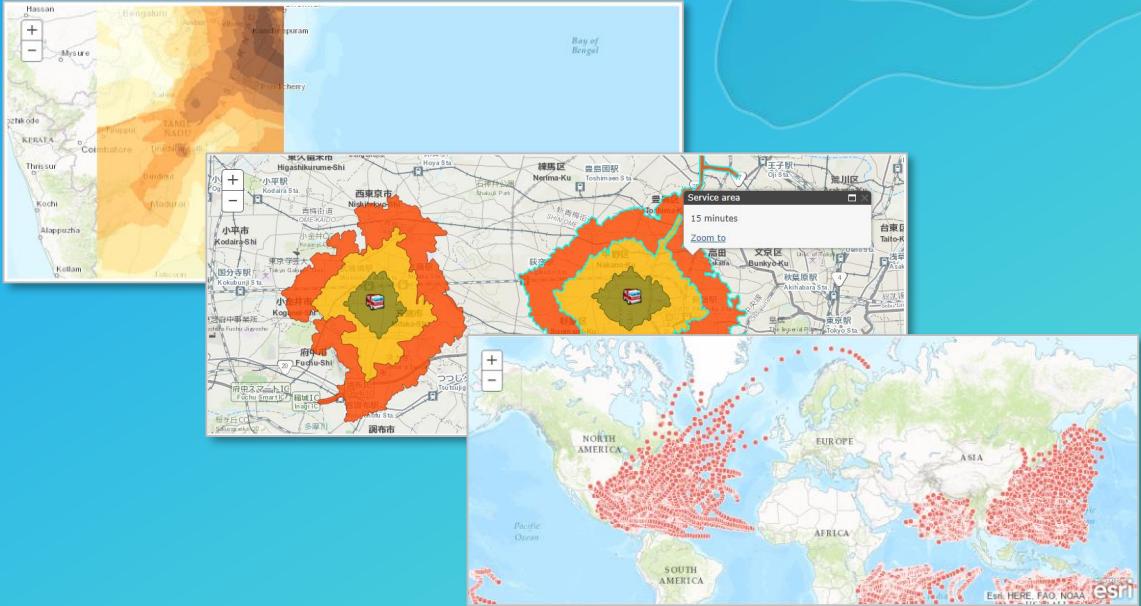
- Geoprocessing in Web GIS

- Spatial analysis, Routing and directions
 - Network analysis, Geocoding, Geoenrichment...

- Imagery and Raster Analysis

- On the fly dynamic image processing
 - Distributed raster analysis

- GeoAnalytics – large tabular and vector data



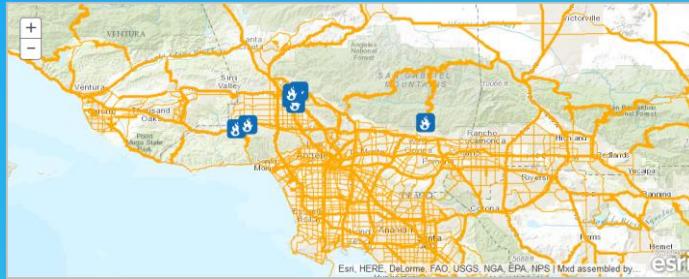
Data Science with ArcGIS - Analysis

- Analysis with Python libraries
 - Data wrangling
 - Pandas, numpy, scipy
 - Machine learning
 - Scikit-learn, tensorflow, keras, pytorch, fastai
 - Geospatial analysis
 - PySAL, GDAL, Shapely, Fiona, ...
 - Image processing and computer vision
 - PIL, OpenCV, scikit-image



Data Science workflows - Deployment

- Deploy as information products
 - WebMaps
 - Web scenes
 - Layers
- Deploy as web tools
 - Geoprocessing script tools
 - Binder projects
- Deploy as dashboards
 - ArcGIS Operations Dashboard
 - Jupyter dashboard
 - Plotly dashboard



Questions?

Please fill out the surveys!



esri

THE
SCIENCE
OF
WHERE