ArcGIS API for Python: Image Analysis

Rohit Singh
Nawajish Noman
ArcGIS API for Python

What is it?

- Python library for spatial analysis, mapping and GIS
- Powerful, modern and easy to use
- Powered by Web GIS

Made for AUTOMATION and DATA SCIENCE
API Overview
A GIS represented in Python
arcgis.raster module

- The **arcgis.raster** module contains classes and **raster analysis functions** for working with raster data and imagery layers in ArcGIS Enterprise.
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]:
```python
sdmap = gis.map('San Diego', zoomlevel=14)
sdmap
```

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

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

• **Correction**
  - Perform geometric and radiometric corrections to get rid of flaws caused by
    - the curved shape of the Earth,
    - the imperfectly transparent atmosphere,
    - daily and seasonal variations in the amount of solar radiation
    - imperfections in scanning instruments, among other things.

• **Visualization and Appearance**
  - Enhances the appearance by modifying the brightness and contrast
  - The Convolution functions sharpens an image, blurs an image, detects edges
  - The Pansharpening function fuses a higher-resolution panchromatic image with a lower-resolution, multiband raster dataset to increase the spatial resolution of the multiband image.
Image Processing (cont.)

- Band Math and Indices
  - Image indices are computed from multiband images to emphasize a specific phenomenon.
  - The normalized difference vegetation index (NDVI) is a standardized index allowing you to display greenness, also known as relative biomass.
    - High values represent forest areas and lush vegetation.
    - Moderate values represent areas of shrubs and grassland.
    - Very low values represent areas of little to no vegetation, such as concrete, rock, or bare soil.
    - Extremely low or negative values represent areas with no vegetation at all, such as cloud, water, snow.
  - The NDVI Colorized function applies the NDVI function on the input image, and then uses a color map or color ramp to display the result.
Demo

Burn Index
California wildfires 2017 - Thomas Fire analysis

The Thomas Fire is a massive wildfire burning in Ventura and Santa Barbara Counties, and one of multiple wildfires that started in early December 2017 in Southern California. It has burned approximately 270,500 acres (423 sq mi; 1,095 km²), becoming the largest wildfire of the 2017 California wildfire season, and the third-largest wildfire in modern California history.

```python
30 [29]: from arcgis.gis import *
    ...: import pandas as pd
```
Image Processing (cont.)

- Image Segmentation and Classification
  - Perform object based image classification on image segments or pixels using a variety of supervised and unsupervised techniques through an easy-to-use wizard workflow.
    - Supervised/unsupervised classification
    - Object based and pixel based segmentation
    - Accuracy assessment
    - SVM, random trees, ISO clustering, MLC
Demo

Flood Classification
Raster Analysis

• Distance Analysis
  • Apply distance functions to create maps of distance, and cost distance from locations, the shortest path two locations, or the optimal travel path between multiple locations.

• Suitability Modeling
  • Determine an ideal location for operations, considering multiple factors that affect suitability. Combine categorial (land use, soil type) and continuous (slope, property value) variables to create a suitability map. Then use Locate Regions to find the best locations.

• Hydrological Analysis
  • Derive surface flow patterns from a DEM. Create watersheds, stream networks, flow distance, and other hydrological characteristics. Summarize landscape data as input to hydrologic and hydraulic models, as well as flood inundation and erosion models.
Demo
Cross Country Mobility
Cross country mobility

Find most efficient paths for off-road vehicles
Use Geoprocessing for offroad routing

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

```python
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]:

![Terrain Map](image_url)
Land Cover
Barren, developed or cultivated?

```python
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]:

![Map of land cover](image-url)
Vegetation characteristics
Forest, shrub or pasture?

In [36]:

```
vegetation = pd.read_csv('CCM/LUTables/veg.csv')
vegetation
```

<table>
<thead>
<tr>
<th>mapunit</th>
<th>mapdesc</th>
<th>ttadbmapunit</th>
<th>stemd</th>
<th>stems</th>
<th>vr</th>
<th>v1</th>
<th>v2</th>
<th>f3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Evergreen Forest</td>
<td>C</td>
<td>0.183</td>
<td>3.030</td>
<td>0.60</td>
<td>NaN</td>
<td>NaN</td>
<td>0.60</td>
</tr>
<tr>
<td>1</td>
<td>Mixed Forest</td>
<td>E</td>
<td>0.167</td>
<td>3.515</td>
<td>0.60</td>
<td>NaN</td>
<td>NaN</td>
<td>0.60</td>
</tr>
<tr>
<td>2</td>
<td>Shrub/Scrub</td>
<td>B1, B2, H</td>
<td>NaN</td>
<td>NaN</td>
<td>0.68</td>
<td>NaN</td>
<td>NaN</td>
<td>0.68</td>
</tr>
<tr>
<td>3</td>
<td>Grassland/Herbaceous</td>
<td>G1, G2</td>
<td>NaN</td>
<td>NaN</td>
<td>0.75</td>
<td>NaN</td>
<td>NaN</td>
<td>0.75</td>
</tr>
<tr>
<td>4</td>
<td>Pasture/Hay</td>
<td>A1</td>
<td>NaN</td>
<td>NaN</td>
<td>0.80</td>
<td>NaN</td>
<td>NaN</td>
<td>0.80</td>
</tr>
<tr>
<td>5</td>
<td>Cultivated Crops</td>
<td>A2, A3, A5, A6, A7, FC, FD, FE, L</td>
<td>NaN</td>
<td>NaN</td>
<td>0.56</td>
<td>NaN</td>
<td>NaN</td>
<td>0.56</td>
</tr>
<tr>
<td>6</td>
<td>Open Water</td>
<td>W</td>
<td>NaN</td>
<td>NaN</td>
<td>0.00</td>
<td>NaN</td>
<td>NaN</td>
<td>0.00</td>
</tr>
<tr>
<td>7</td>
<td>Developed, Open Space</td>
<td>X</td>
<td>NaN</td>
<td>NaN</td>
<td>0.00</td>
<td>NaN</td>
<td>NaN</td>
<td>0.00</td>
</tr>
<tr>
<td>8</td>
<td>Developed, Low Intensity</td>
<td>X</td>
<td>NaN</td>
<td>NaN</td>
<td>0.00</td>
<td>NaN</td>
<td>NaN</td>
<td>0.00</td>
</tr>
<tr>
<td>9</td>
<td>Developed, Medium Intensity</td>
<td>X</td>
<td>NaN</td>
<td>NaN</td>
<td>0.00</td>
<td>NaN</td>
<td>NaN</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>Developed High Intensity</td>
<td>X</td>
<td>NaN</td>
<td>NaN</td>
<td>0.00</td>
<td>NaN</td>
<td>NaN</td>
<td>0.00</td>
</tr>
<tr>
<td>11</td>
<td>Barren Land (Rock/Sand/Clay)</td>
<td>N</td>
<td>NaN</td>
<td>NaN</td>
<td>1.00</td>
<td>NaN</td>
<td>NaN</td>
<td>1.00</td>
</tr>
<tr>
<td>12</td>
<td>Deciduous Forest</td>
<td>D</td>
<td>0.150</td>
<td>4.000</td>
<td>0.60</td>
<td>NaN</td>
<td>NaN</td>
<td>0.60</td>
</tr>
<tr>
<td>13</td>
<td>Woody Wetlands</td>
<td>I C, D, J, E</td>
<td>NaN</td>
<td>NaN</td>
<td>0.40</td>
<td>NaN</td>
<td>NaN</td>
<td>0.40</td>
</tr>
<tr>
<td>14</td>
<td>Emergent Herbaceous Wetlands</td>
<td>J, K</td>
<td>NaN</td>
<td>NaN</td>
<td>0.55</td>
<td>NaN</td>
<td>NaN</td>
<td>0.55</td>
</tr>
</tbody>
</table>
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('RodsAndRails')[0]
transportation

Out[38]:

RoadsAndRails_SD
Binary Roads and Rails for SD county

Imagery Layer by bgerItRA
Last Modified: July 18, 2017
0 comments, 4 views
Vehicle Characteristics
Jeep, truck or tank?

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

Out[39]:
```
<table>
<thead>
<tr>
<th></th>
<th>name</th>
<th>maxmph</th>
<th>onslope</th>
<th>offslope</th>
<th>vehiclewidth</th>
<th>oversidediameter</th>
<th>vci</th>
<th>vci50</th>
<th>minturnrad</th>
<th>vehiclelength</th>
<th>mlc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>M1</td>
<td>71.0</td>
<td>68.7</td>
<td>55.0</td>
<td>3.65</td>
<td>0.25</td>
<td>25.0</td>
<td>58.0</td>
<td>9.90</td>
<td>9.90</td>
<td>60.0</td>
</tr>
<tr>
<td>1</td>
<td>M60A1</td>
<td>48.0</td>
<td>60.0</td>
<td>45.0</td>
<td>3.63</td>
<td>0.15</td>
<td>20.0</td>
<td>48.0</td>
<td>9.40</td>
<td>9.40</td>
<td>54.0</td>
</tr>
<tr>
<td>2</td>
<td>M109</td>
<td>56.0</td>
<td>60.0</td>
<td>45.0</td>
<td>3.10</td>
<td>0.12</td>
<td>25.0</td>
<td>57.0</td>
<td>6.60</td>
<td>6.60</td>
<td>24.0</td>
</tr>
<tr>
<td>3</td>
<td>M113</td>
<td>48.0</td>
<td>60.0</td>
<td>45.0</td>
<td>2.69</td>
<td>0.10</td>
<td>17.0</td>
<td>40.0</td>
<td>4.00</td>
<td>4.00</td>
<td>12.0</td>
</tr>
<tr>
<td>4</td>
<td>M35A2</td>
<td>56.0</td>
<td>64.0</td>
<td>30.0</td>
<td>2.43</td>
<td>0.06</td>
<td>26.0</td>
<td>59.0</td>
<td>5.30</td>
<td>6.70</td>
<td>10.0</td>
</tr>
<tr>
<td>5</td>
<td>M191</td>
<td>50.0</td>
<td>60.0</td>
<td>28.0</td>
<td>1.69</td>
<td>0.04</td>
<td>19.0</td>
<td>44.0</td>
<td>5.00</td>
<td>3.35</td>
<td>NaN</td>
</tr>
<tr>
<td>6</td>
<td>T62</td>
<td>50.0</td>
<td>62.0</td>
<td>45.0</td>
<td>3.37</td>
<td>0.15</td>
<td>21.0</td>
<td>49.0</td>
<td>9.33</td>
<td>9.33</td>
<td>42.0</td>
</tr>
<tr>
<td>7</td>
<td>T72</td>
<td>60.0</td>
<td>62.0</td>
<td>45.0</td>
<td>3.38</td>
<td>0.18</td>
<td>25.0</td>
<td>60.0</td>
<td>9.20</td>
<td>9.20</td>
<td>45.0</td>
</tr>
</tbody>
</table>
```
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]:
Raster Analysis (contd.)

• Interpolation and Density
  • Predicts values and creates a continuous surface from sample data points, such as elevation, rainfall, chemical concentrations, and noise levels.

• Derivation of land surface characteristics
  • Create hillshade, slope, aspect, curvature, solar insolation, and visibility to help solve spatial problems

• Statistical Analysis
  • Perform calculations on a per-cell basis between multiple datasets, based on local, neighborhood, or zonal functions.

• Generalization
  • Either clean up small erroneous data in the raster or generalize the data to get rid of unnecessary detail for a more general analysis.

• Map Algebra
  • An easy-to-use and powerful way to perform raster analysis as algebraic expressions.
Demo
Map Algebra
Running in San Diego

Least Important  
20%

Moderately Important  
30%

Most Important  
50%

Low Elevation  
Flat, Not Hilly  
Natural, Not Built
Inputs - Elevation

In [8]:

```python
# 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),
                [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(runRaster, {'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('San Diego Places To Run')

In [26]:
resultlyr

Out[26]:
San Diego Places To Run
Analysis Image Service generated from GenerateRaster
Imagery Layer by arcgis_python
Last Modified: July 06, 2017
0 comments, 0 views
Image Processing & Raster Analysis Functions

**Multiband Math**
- Arithmetic
- Band Arithmetic

**Math**
- Calculator
- Abs
- Divide
- Exp
- Exp10
- Exp2
- Float
- Int
- Ln
- Log10
- Log2
- Minus
- Mod
- Negate
- Plus
- Power
- Round Down
- Round Up
- Square
- Square Root
- Times

**Correction**
- Apparent Reflectance
- Geometric Correction
- Speckle Filtering (Lee, Frost, Kuan)
- Thermal Noise
- Radiometric Calibration

**Data Management & Conversion**
- Raster to Vector
- Vector to Raster
- Colormap
- Colormap To RGB
- Complex
- Grayscale
- Remap / Reclass
- Spectral Conversion
- Unit Conversion
- Vector Field
- LAS to Raster
- LAS Dataset to Raster
- Clip
- Composite
- Extract Bands
- Mask
- Mosaic Rasters
- Rasterize Features
- Reproject
- Nibble

**Interpolation**
- Interpolate Irregular Data
- Nearest Neighbor
- IDW
- EBK
- swath

**Visualization & Appearance**
- Contrast and Brightness
- Convolution
- Pan-sharpening
- Resample
- Statistics and Histogram
- Stretch

**Surface Generation & Analysis**
- Aspect
- Curvature
- Elevation Void Fill
- Hillshade
- Shaded Relief
- Slope
- Viewshed
- Contour

**Analysis: Image Segmentation & Classification**
- Segmentation (Mean Shift)
- Training (ISO, SVM, ML)

**Analysis: Distance & Density**
- Euclidean Distance
- Cost Distance
- Least Cost Path
- Kernel Density

**Analysis: Band Math & Indices**
- NDVI / NDVI Colorized
- SAVI / MSAVI / TSAVI
- GEMI
- GVI (Landsat TM)
- PVI
- Tasseled Cap (Kauth-Thomas)
- Binary Thresholding

**Analysis: Overlay**
- Weighted Overlay
- Weighted Sum

**Analysis: Hydrology**
- Fill
- Flow Accumulation
- Flow Direction
- Flow Distance
- Stream Link
- Watershed

**Python**
- Custom Algorithms

**Statistics:**
- Zonal Statistics
- Cell Statistics
- ArgStatistics
Demo
Hydrologic Analysis
Demo
Deep Learning
Resources

A powerful Python library for spatial analysis, mapping and GIS

ArcGIS API for Python is a Python library for working with and exploring geographic data, powered by ArcGIS. It provides simple and efficient tools for integrated mapping and data analysis, processing, visualization, and distribution, as well as being integrated and managing a GIS with users, groups and environments. In addition to working with ArcGIS data, the library includes access to ready-to-use maps and created geographic data from the ArcGIS API for Python. It also integrates well with the scientific Python ecosystem and includes full support for Pandas and NumPy notebooks.

https://developers.arcgis.com/python/

Road Ahead: Enterprise Image Server  *(big data raster analytics)*

- New suitability modeling dashboard web interface
- Function chain editor UI for building raster analytic workflows
- Continue to grow the collection of scalable distributed analytics tools
  - Path distance tools
  - Locate Regions
  - Neighborhood (focal) statistics tools
  - More new hydro tools
Please Take Our Survey on the App

Download the Esri Events app and find your event

Select the session you attended

Scroll down to find the feedback section

Complete answers and select “Submit”