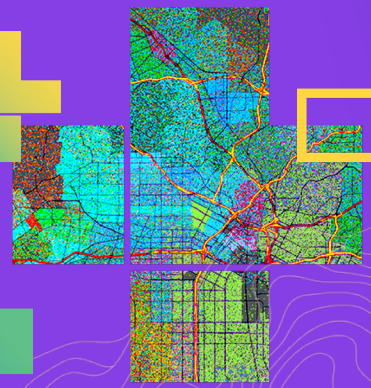
Advanced Analysis of Satellite Imagery Using Python

Hong Xu, Kaixi Zhang

2020 ESRI DEVELOPER SUMMIT | Palm Springs, CA

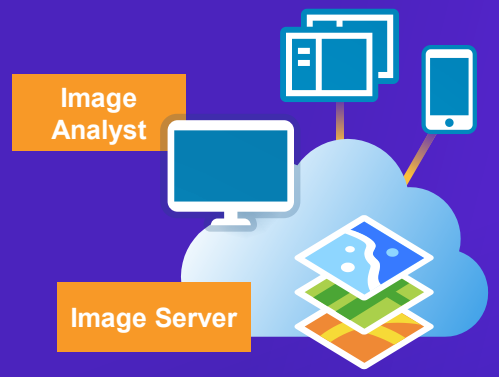


Workshop Outline

- ArcGIS is a platform for imagery
- ArcGIS has rich image analysis APIs
  - This workshop will focus on Python, specifically ArcPy in Pro 2.5 release
- Use cases
  - Annual Max NDVI Trend Analysis using time series Landsat images
  - Analyze land cover changes using linear spectral unmixing

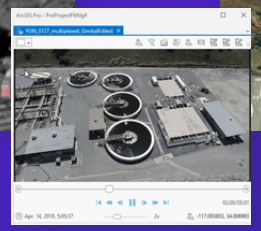
ArcGIS **Provides a Comprehensive Platform for Imagery and Remote Sensing**

Management & Dissemination



Visualization & Exploration

Image Space and Mensuration

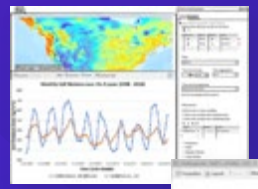


Full-Motion Video

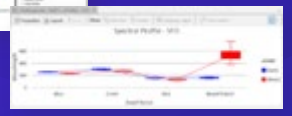
Oriented Imagery

Street Views

Charting



Temporal Profile



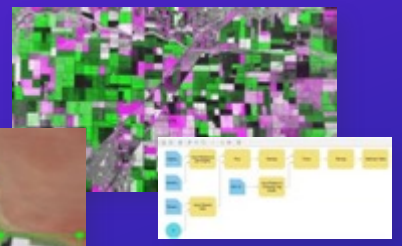
Spectral Profile

Analysis

Indexes

Classification

Change Analysis



AI / Deep Learning

Raster Processing Functions

Map Production

Stereo

Drone Mapping

DTM Generation

Drone2Map

Ortho Mapping

Seamless Orthophotos

Ortho Maker

Content

Landsat

Weather

Small Sats

High-Resolution Satellites

Community Imagery

Radar

Video

Aerial Photos

All Major Sensors

Sentinel

Drones

Integrated and Massively Scalable

Data Management

Making Imagery Accessible – Image Management and Dissemination

- **Mosaic datasets - optimal data model for managing imagery and rasters**
  - 20+ satellite data: Landsat, MODIS, Sentinel, etc.
  - Reference imagery and maintain metadata
  - Define processing to be applied
  - Scalable and flexible

- **Multidimensional raster**
  - Supports multidimensional raster analysis

- **Image server**
  - Dynamic Mosaicking
  - On-the-fly processing
  - Persist large datasets as required
  - Scalable
  - Server as Image Services, WMS, WCS, KML

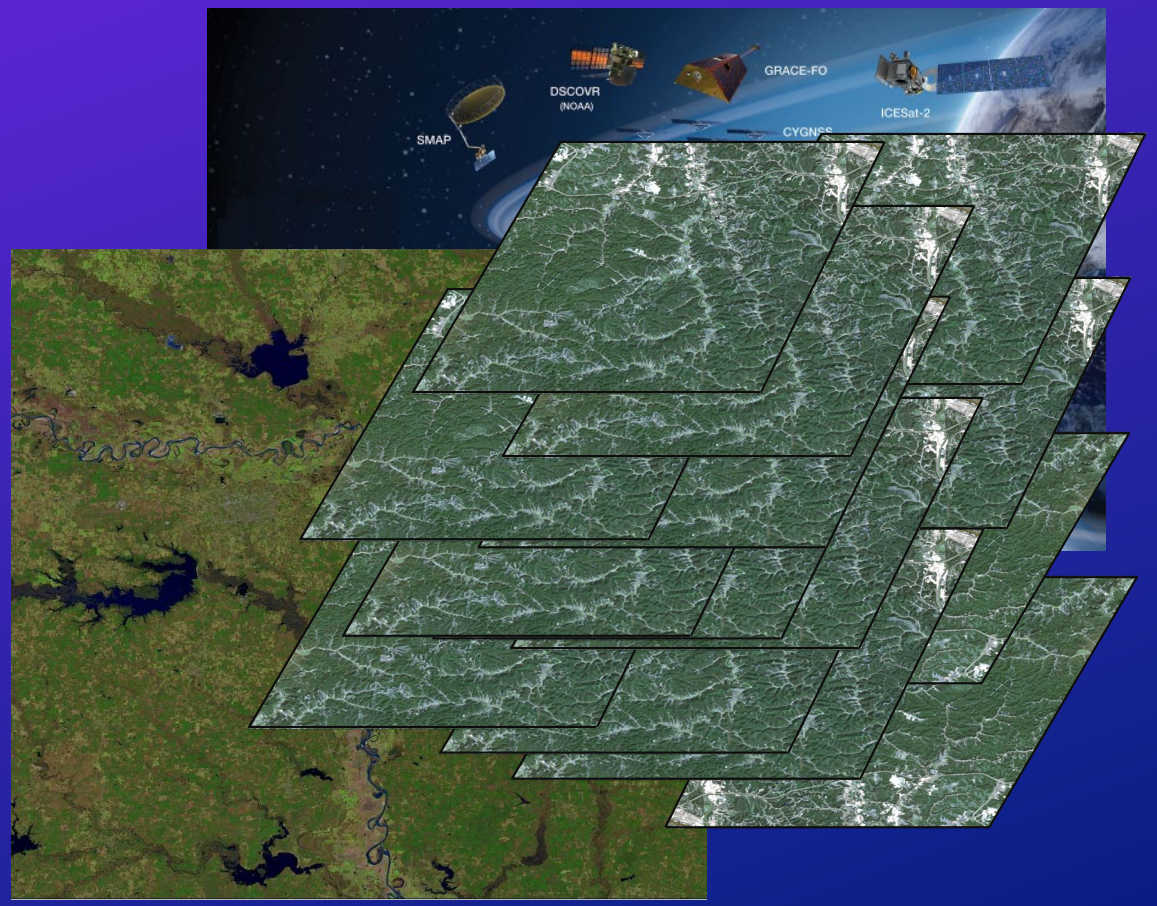
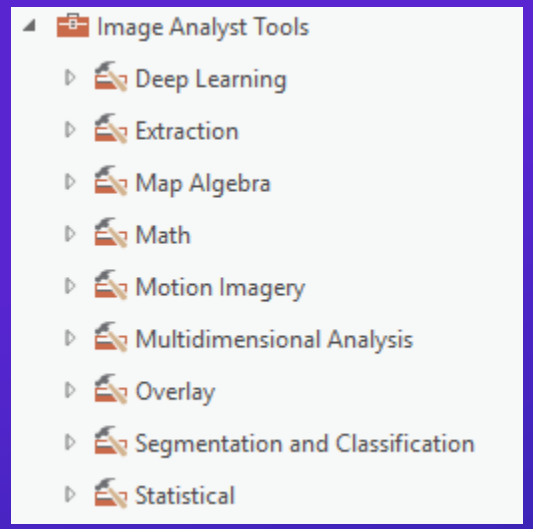


Image Analysis

Extracting Information from Imagery

Classification, deep learning, machine learning, prediction, multidimensional raster analysis

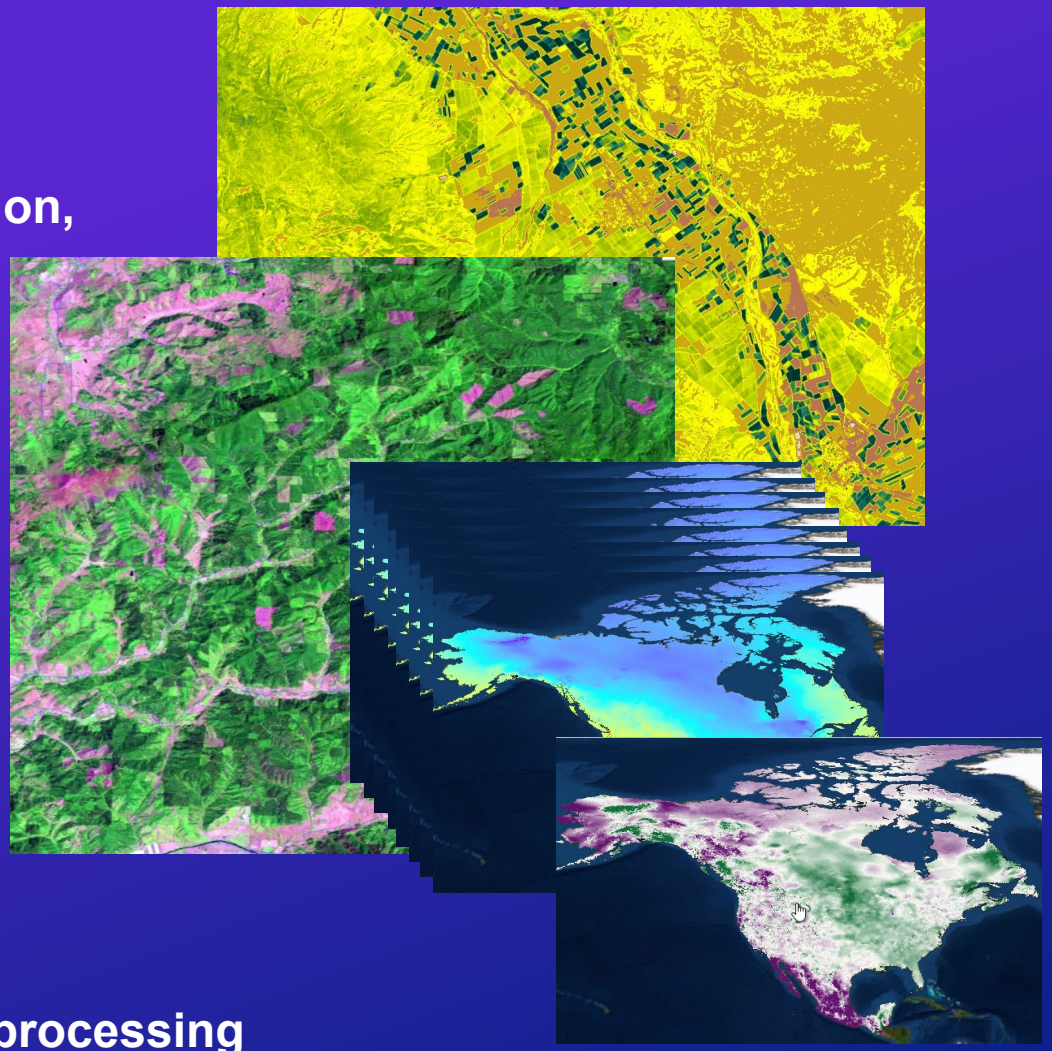
- **Image Analyst tools and ArcGIS Pro UX**



-Complete image classification workflow

-Interface with deep learning toolkits to create training data

-Inferencing to identify, label, or classify imagery



- **Raster functions - fast on-the-fly and persisted image processing**
- **Raster analytics - scale up and distribute analysis for large datasets**

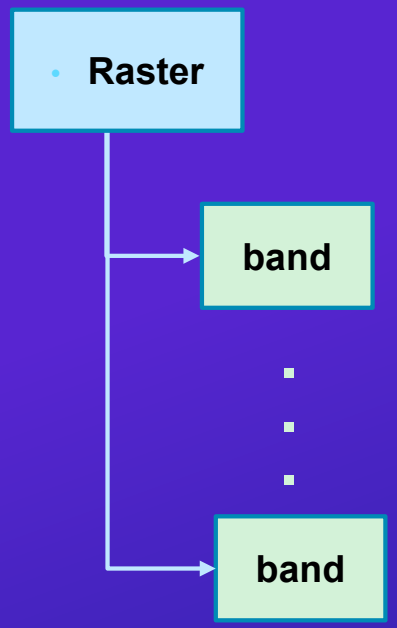
ArcGIS Python Environment for Image Processing and Analysis

- **Python packages**
  - ArcPy
    - ArcPy.ia (Image Analyst)
  - ArcGIS API for Python

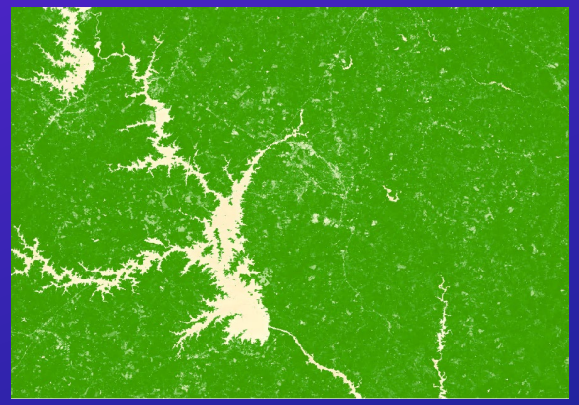
- **Code snippets and demos use**
  - ArcPy in Pro 2.5
  - ArcGIS Notebooks

Processing One Image

Raster → Function



NDVI →

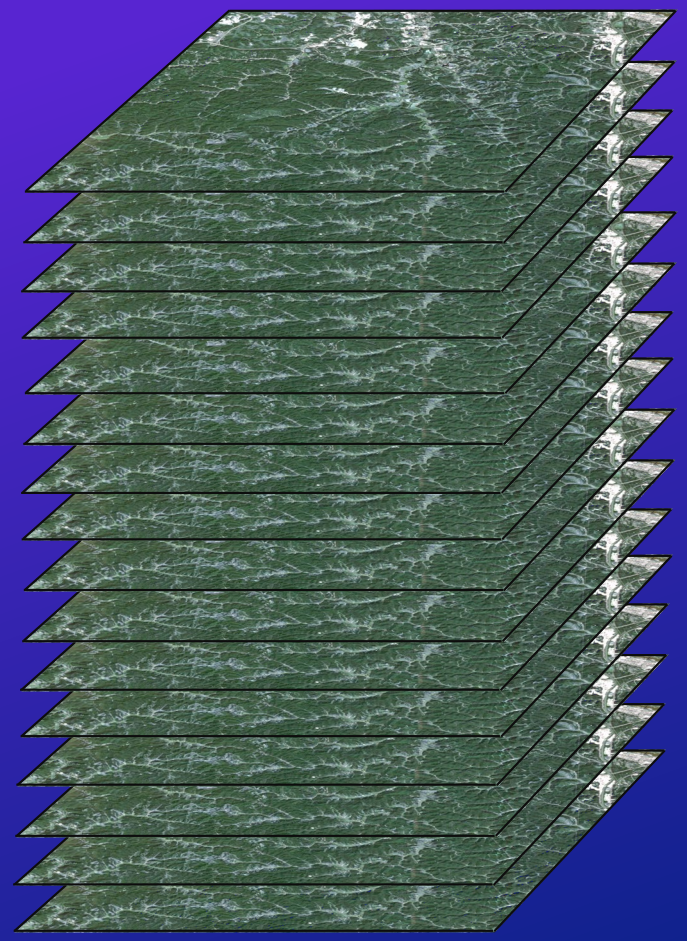


```
import arcpy
arcpy.CheckOutExtension("ImageAnalyst")
#create a Raster
scene = arcpy.Raster(r"C:\Landsat\LC80420342013236LGN00\LC80420342013236LGN00_MTL.txt\Multispectral")
#Display
sceneRGB = arcpy.ia.Render(scene, rendering_rule={'bands':[4,3,2], 'NumberOfStandardDeviation:2'})
#Compute an NDVI
sceneNDVI = arcpy.ia.NDVI(scene,  5, 4)
scene_ndvi_render = arcpy.ia.Render(sceneNDVI, rendering_rule={'min': -1, 'max': 1}, colormap="NDVI")
```

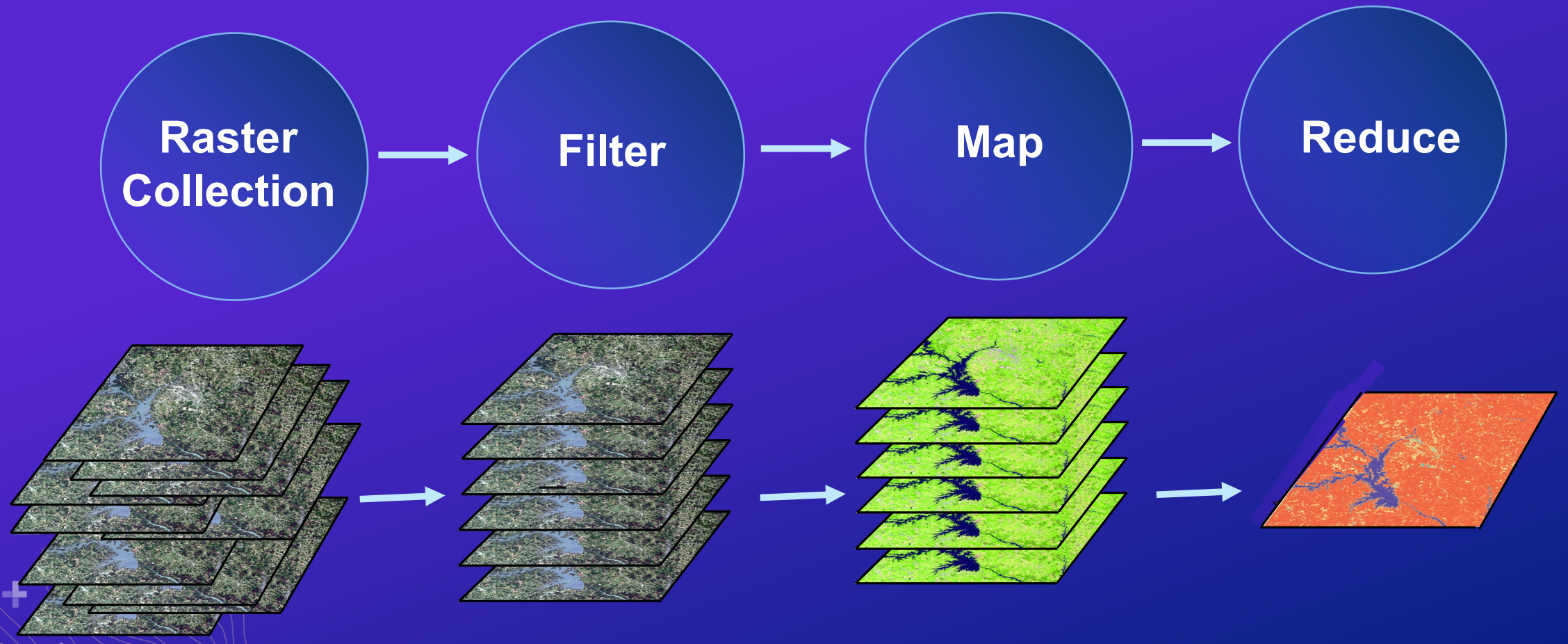
Process a Collection of Images

- **Landsat time series imagery**
  - 16-day cycle, 40 years
  - 30m/10m
  - Blue, Green, Red, NIR, SWIR-1, SWRI-2, TIR, Pan

- **To discover**
  - What is the water occurrence rate in past 10 years?
  - Did deforestation happen in this area?
  - What are the land cover changes?
  - What are the pattern and trend?
  - Etc.

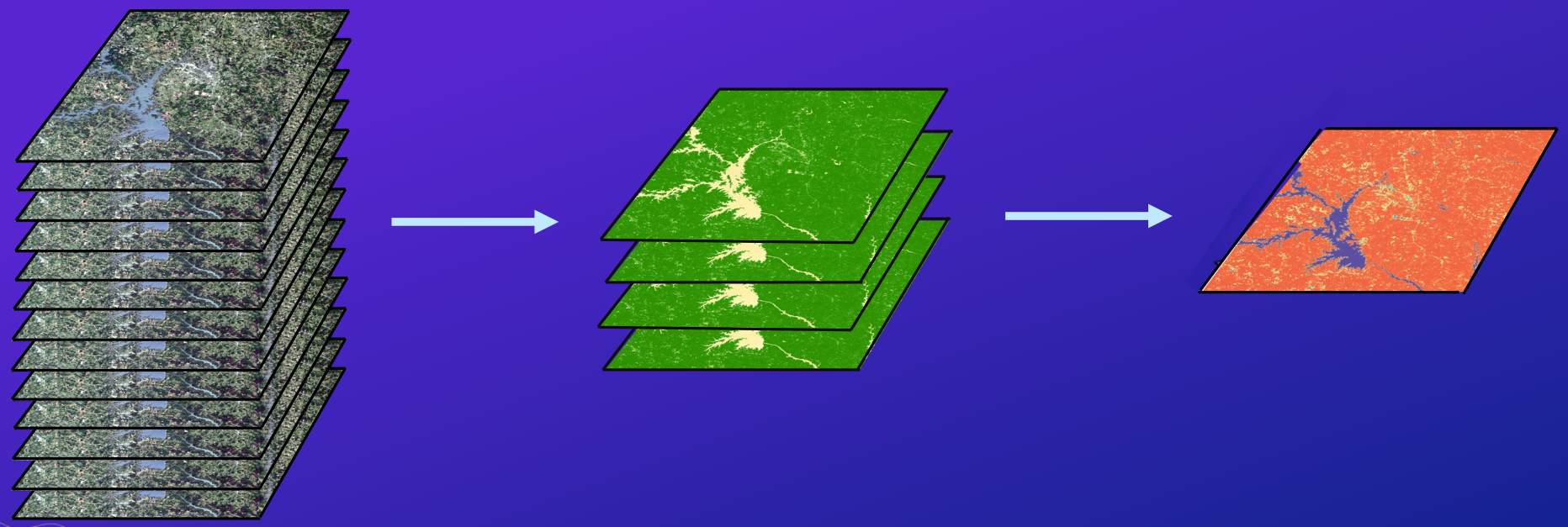


Workflow of Processing a Collection of Images Using Python

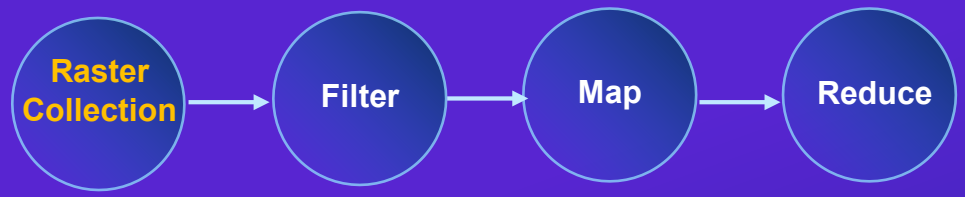


Use Case 1 : Annual Max NDVI Trend Analysis using time series Landsat images

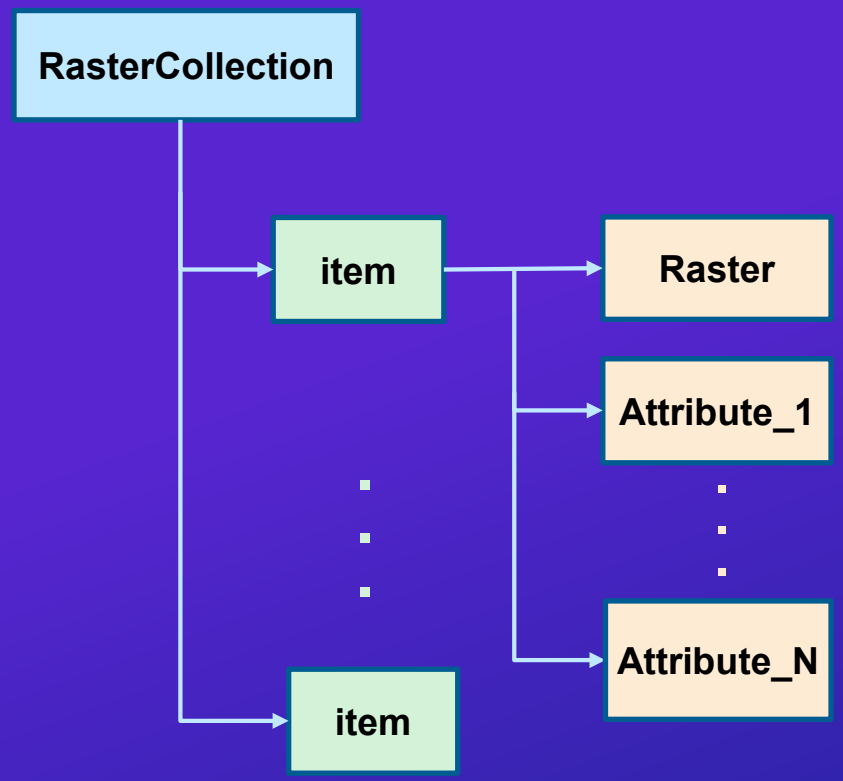
- Input: a folder contains 173 Landsat scenes
- Output: a time series Annual Maximum NDVI and a trend raster



Raster Collection

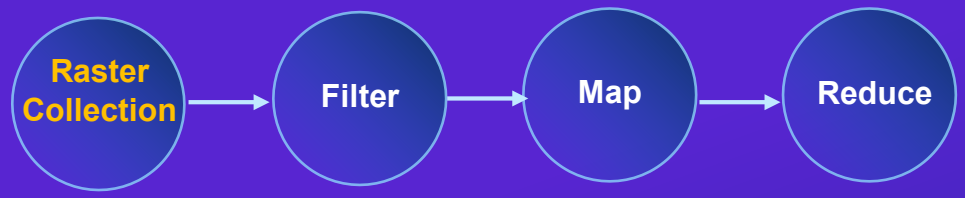


RasterCollection (rasters, {attribute_dict})



	Variable	Name	StdTime	Raster
0	Landsat7	LE07_L1TP_194055_20000314_20170213_01_T1	2000-03-14T00:00:00+00:00	C:\AppData\Local\Temp\1\x7477b6...
1	Landsat7	LE07_L1TP_194055_20000415_20170212_01_T1	2000-04-15T00:00:00+00:00	C:\AppData\Local\Temp\1\x6218e4...
2	Landsat7	LE07_L1TP_194055_20000517_20170211_01_T1	2000-05-17T00:00:00+00:00	C:\AppData\Local\Temp\1\x65db79...
3	Landsat7	LE07_L1TP_194055_20000720_20170210_01_T1	2000-07-20T00:00:00+00:00	C:\AppData\Local\Temp\1\x311095...
4	Landsat7	LE07_L1TP_194055_20001109_20170209_01_T1	2000-11-09T00:00:00+00:00	C:\AppData\Local\Temp\1\x0b483f...
5	Landsat7	LE07_L1TP_194055_20001125_20170209_01_T1	2000-11-25T00:00:00+00:00	C:\AppData\Local\Temp\1\xf34fb1...
6	Landsat7	LE07_L1TP_194055_20001227_20170208_01_T1	2000-12-27T00:00:00+00:00	C:\AppData\Local\Temp\1\x32111b...
7	Landsat7	LE07_L1TP_194055_20010301_20170206_01_T1	2001-03-01T00:00:00+00:00	C:\AppData\Local\Temp\1\xa2ad2c...
8	Landsat7	LE07_L1TP_194055_20010317_20170206_01_T1	2001-03-17T00:00:00+00:00	C:\AppData\Local\Temp\1\x894b56...
9	Landsat7	LE07_L1TP_194055_20010402_20170206_01_T1	2001-04-02T00:00:00+00:00	C:\AppData\Local\Temp\1\x284b09...
10	Landsat7	LE07_L1TP_194055_20010418_20170205_01_T1	2001-04-18T00:00:00+00:00	C:\AppData\Local\Temp\1\x7b7698...
11	Landsat7	LE07_L1TP_194055_20030307_20170126_01_T1	2003-03-07T00:00:00+00:00	C:\AppData\Local\Temp\1\x1a3d3a...
12	Landsat7	LE07_L1TP_194055_20040426_20170121_01_T1	2004-04-26T00:00:00+00:00	C:\AppData\Local\Temp\1\x34f0e3...
13	Landsat7	LE07_L1TP_194055_20041206_20170117_01_T1	2004-12-06T00:00:00+00:00	C:\AppData\Local\Temp\1\x61d2fd...
14	Landsat7	LE07_L1TP_194055_20050123_20170116_01_T1	2005-01-23T00:00:00+00:00	C:\AppData\Local\Temp\1\xc09d53...
15	Landsat7	LE07_L1TP_194055_20050208_20170116_01_T1	2005-02-08T00:00:00+00:00	C:\AppData\Local\Temp\1\x7687ff...

Raster Collection



- Created from
  - List of rasters

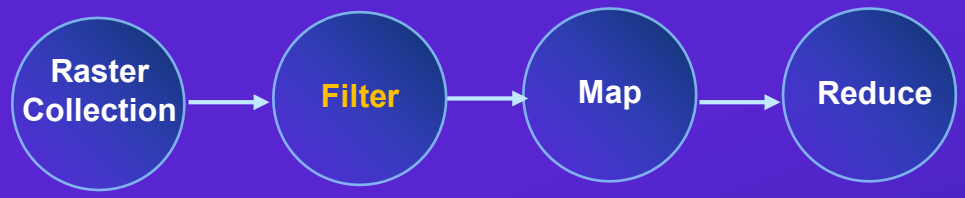
```
import arcpy
# from a list of rasters
in_rasters = [r"D:\data\L7_2000.tif", r"D:\data\L7_2001.tif", r"D:\data\L7_2002.tif"]
rc=arcpy.ia.RasterCollection(in_rasters, {"Variable": "Landsat7",
                                          "StdTime": ["2000-01-01", "2001-01-01", "20002-01-01"]})
```

  - mosaic dataset , Image service, multidimensional raster

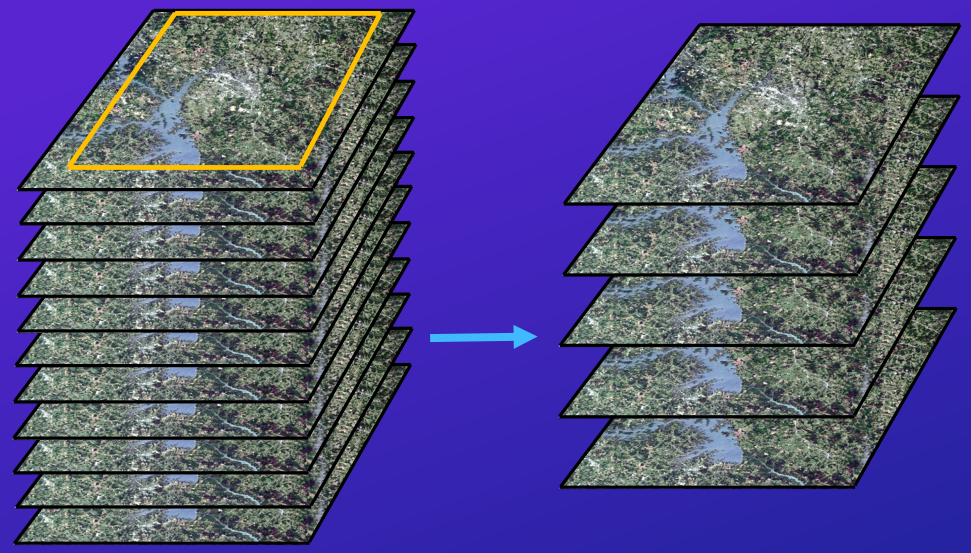
```
#create from a mosaic dataset
rc=arcpy.ia.RasterCollection(r"D:\data\Landsat7.gdb\CA_1990_2000")

#create from an image service
rc=arcpy.ia.RasterCollection("https://localhost/server/rest/services/CA/ImageServer")

#create from multidimensional raster
rc=arcpy.ia.RasterCollection(r"D:\data\CA_raster.crf")
```

https://pro.arcgis.com/en/pro-app/help/data/imagery/mosaic-datasets.htm
https://pro.arcgis.com/en/pro-app/help/data/imagery/an-overview-of-multidimensional-raster-data.htm

Filter



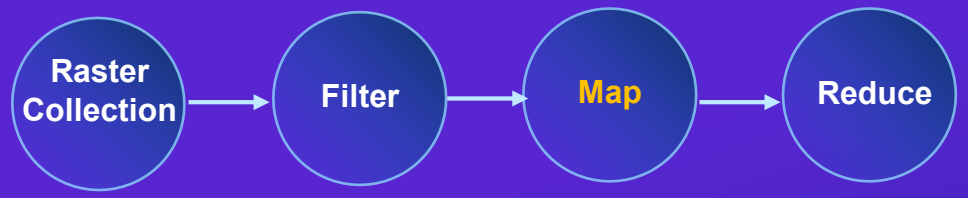
- **Filter items by**
  - Spatial extent
  - Time extent
  - Geometry
  - Where clause and attributes
  - Raster properties



```
filtered_rc = rc.filterByTime(end_time = '2009-01-01T00:00:00', time_field_name = 'AcquisitionDate')
filtered_rc = rc.filterByGeometry(line)
filtered_rc = rc.filterByAttribute("SensorName", "CONTAINS", "Landsat7")
filtered_rc = rc.filterByCalendarRange(calendar_field="DAY_OF_YEAR", start=1, end=5)
```

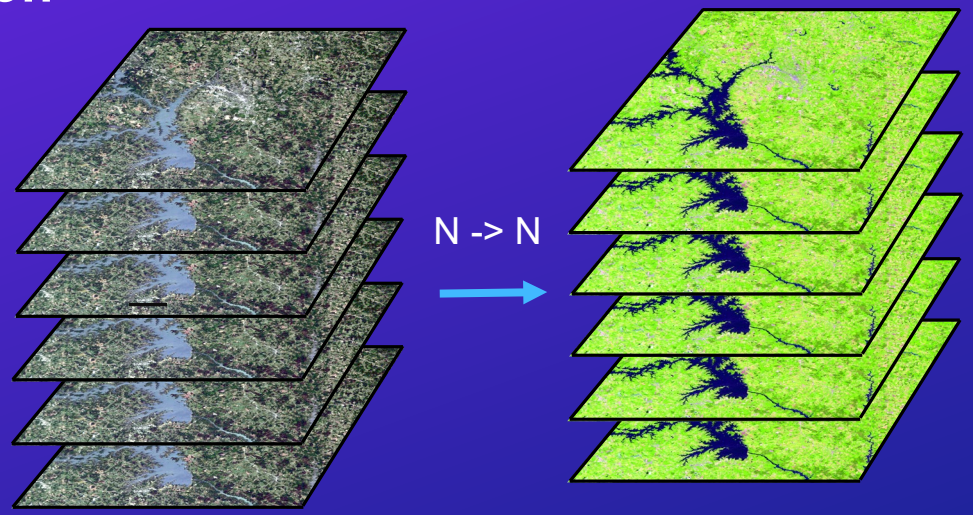
```
filtered_rc = rc.filterByGeometry(study_area).filterByTime(time_field_name = 'AcquisitionDate',
                        start_time = '1990-01-01 00:00:00', end_time = '1995-12-31 00:00:00')
```

Map



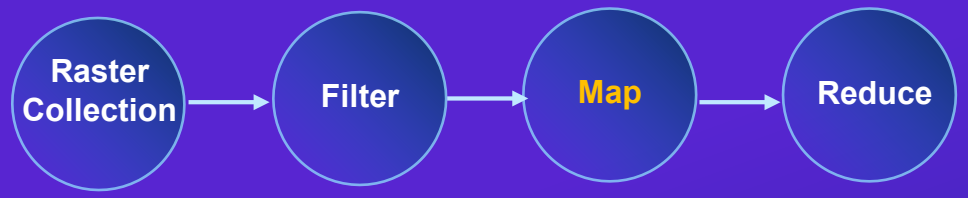
- **Map a function for each item in the RasterCollection**
  - N -> N
- **RasterCollection.map(func)**

```
def MyFunc(item):
        in_raster = item['Raster']
        processed_raster = process_function(in_raster…)
        return {'raster':processed_raster, …}

processed_rc = rc.map(MyFunc)
```




```python
def calcNDVI(item):
    ndvi = arcpy.ia.NDVI(item["Raster"], nir_band_id = 4, red_band_id = 3)
    return {"raster": ndvi, "Name": item["Name"], "AcquisitionDate": item["AcquisitionDate"]}

ndvi_rc = filtered_rc.map(calcNDVI)
```

Map



- **Support many built-in functions**

Band index functions

BandArithmatic, Cig, Cire,ClayMinerals, EVI, FerroursMinerals, GEMI, GNDVI, GVITM, IronOxide, MSAVI, MTVI2, NDVI, NDVIre, NDWI, PVI, RTVICore, SAVI, SRre, Sultan, TSAVI, VARI

Classification

Random Tree, Support Vector Machine, Maximum Likelihood, ISO Cluster, Linear Spectral Unmixing.

Pixel stretch functions

Stretch, convolution, Pansharpen, and etc.

Analysis functions

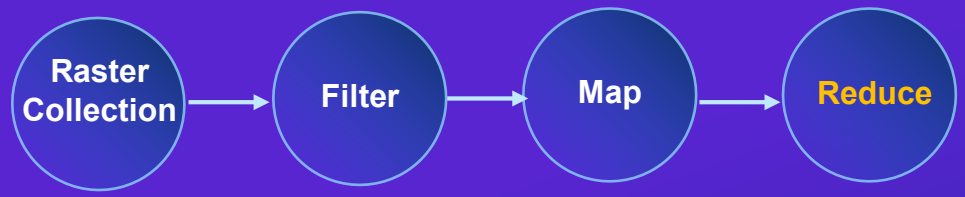
TasseledCap, Threshold, HeatIndex, WindChill, Apply and etc.

arcpy.ia.Apply (in_raster, raster_function, {raster_function_arguments})

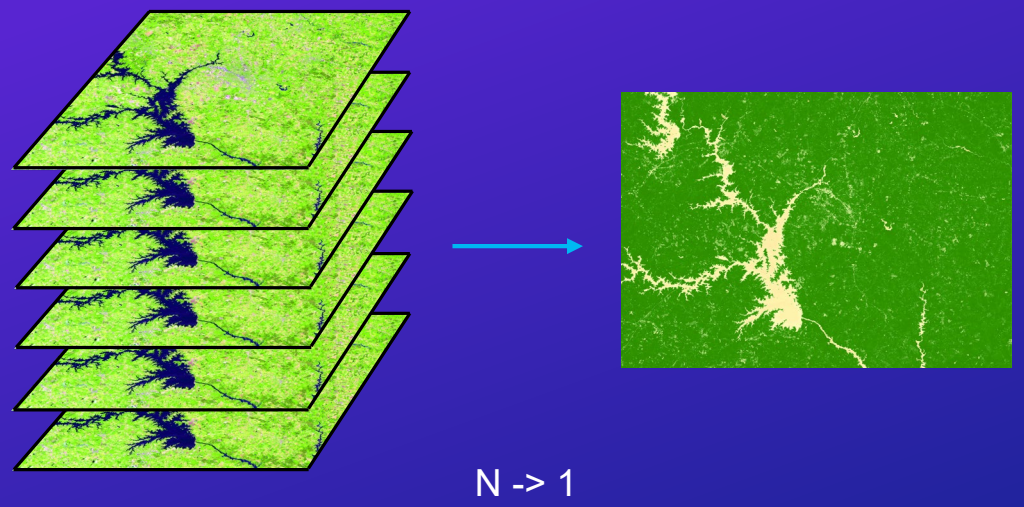
- **more…**

- **Custom functions**

Reduce

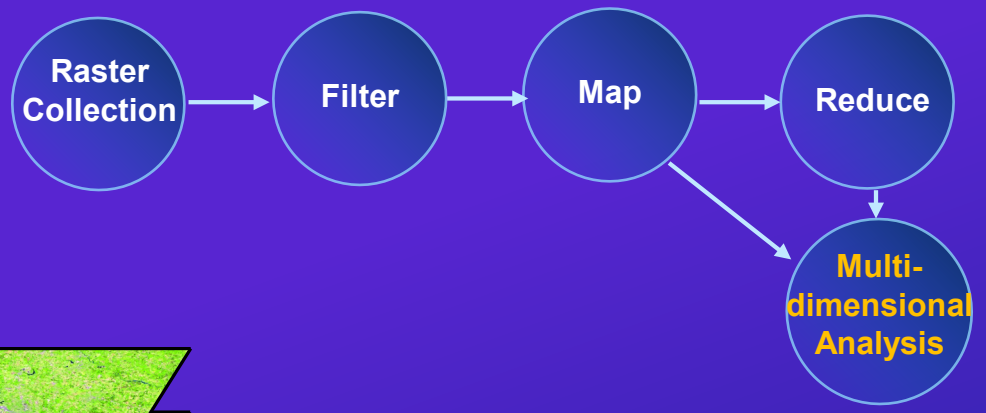


- **Reduce by computing pixel values from items**
  - N->1
- **Supported methods**
  - Min, Max, Mean, Sum, Median, Majority
  - Mosaic (Frist, Last)

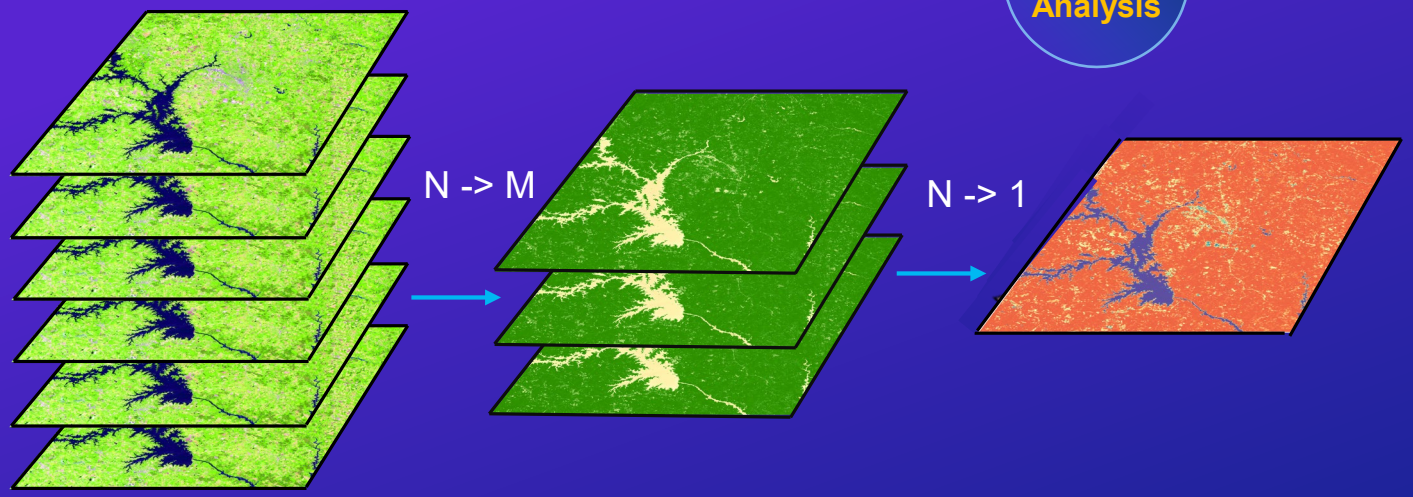



```
# get max NDVI in each year
annual_max_ndvi_list = []
for year in range(1990, 2000):
    # choose all items in the given year
    ndvi_in_one_year = ndvi_rc.filterByCalendarRange(calendar_field = "YEAR", start = year, time_field_name = "AcquisitionDate")
    # get max NDVI
    annual_max_ndvi_list.append(ndvi_in_one_year.max())
annual_max_ndvi_rc = arcpy.ia.RasterCollection(annual_max_ndvi_list, {"years": [yr for yr in range(1990, 2000)]})
```

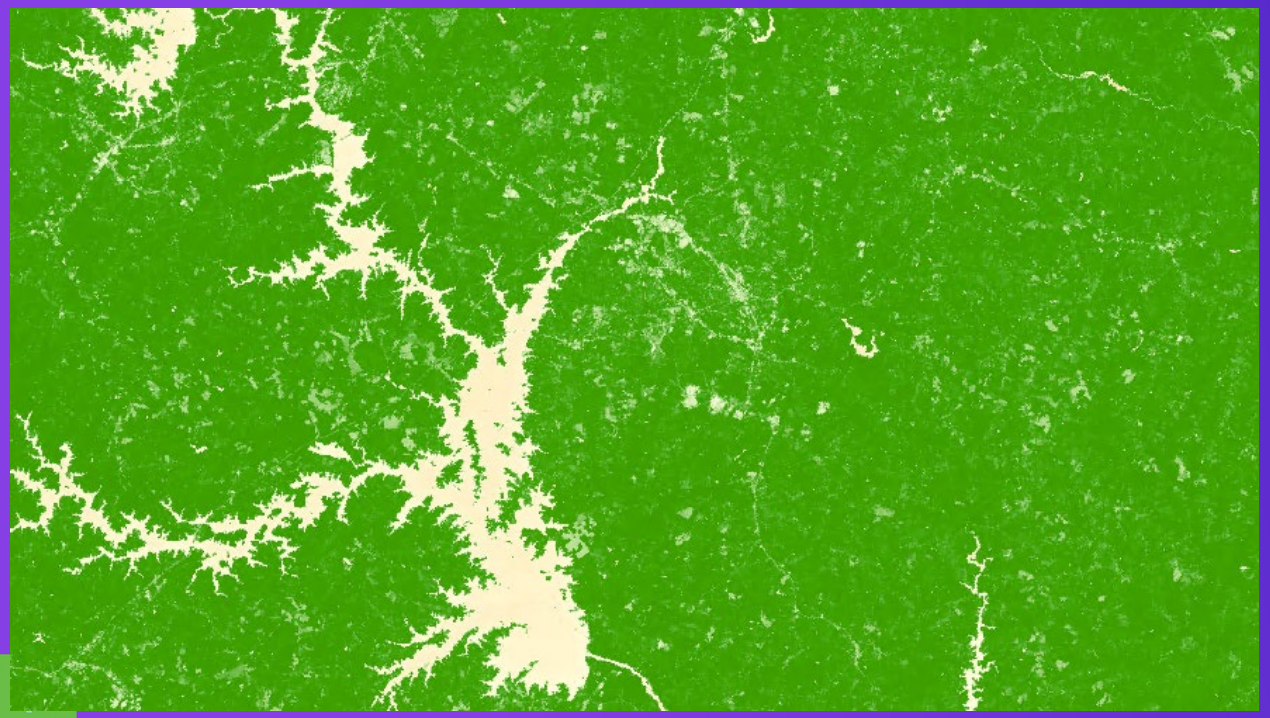
Multidimensional Analysis



- **Convert to a multidimensional raster**
  - RasterCollection::ToMultidimensionalRaster
- **Perform analysis using functions**
  - Aggregate
  - Anomaly
  - Trend
  - Multidimensional algebra
  - …

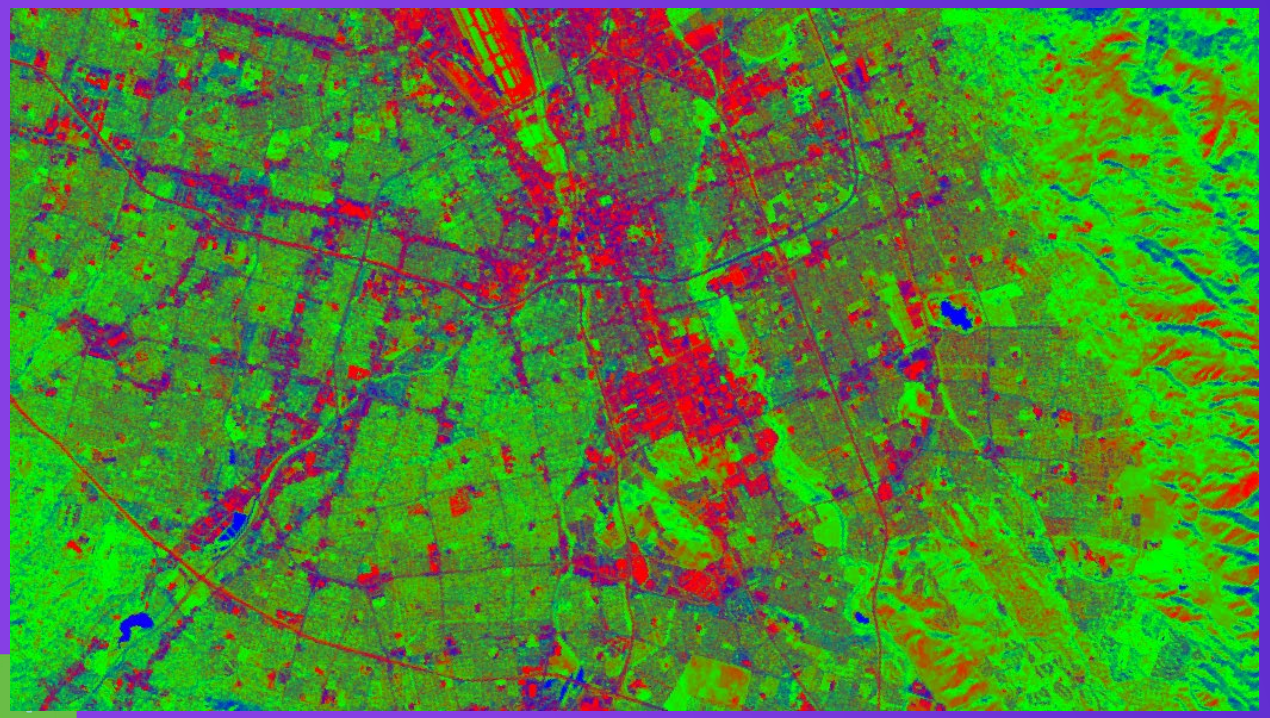



```python
# convert raster collection to a multidimensional raster
ndvi_md_raster = ndvi_rc.toMultidimensionalRaster(variable_field_name='Variable', dimension_field_names='StdTime')
# get max NDVI per year
annual_max_ndvi = arcpy.ia.Aggregate(ndvi_md_raster, dimension_name = "StdTime", raster_function = "MaxIgnoreNoData",
                                     aggregation_definition = "yearly")
# apply trend
max_ndvi_linear_trend = arcpy.ia.Apply(annual_max_ndvi, "TrendAnalysis",
                                       raster_function_arguments = {"DimensionName": "StdTime", "RegressionType": 0})
```



Annual Max NDVI Trend Analysis using time series Landsat images

Kaixi

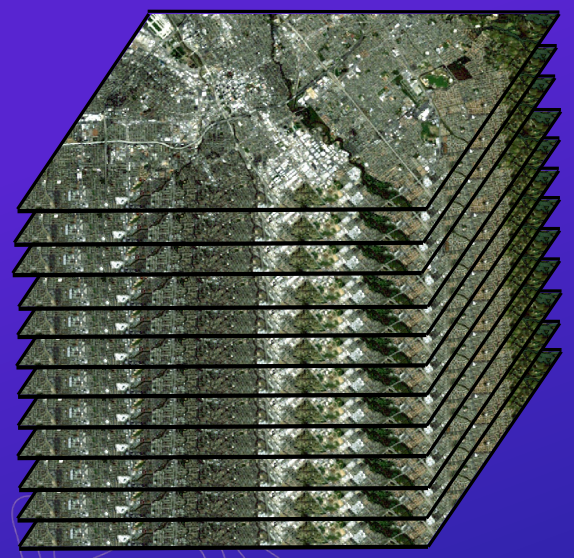


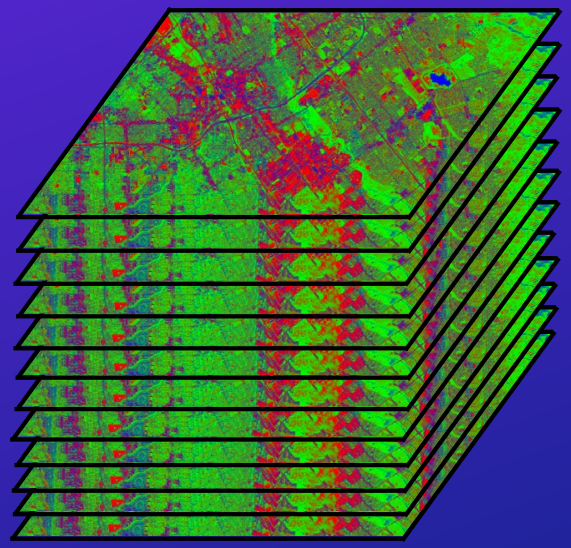
Analyze land cover changes using linear spectral unmixing

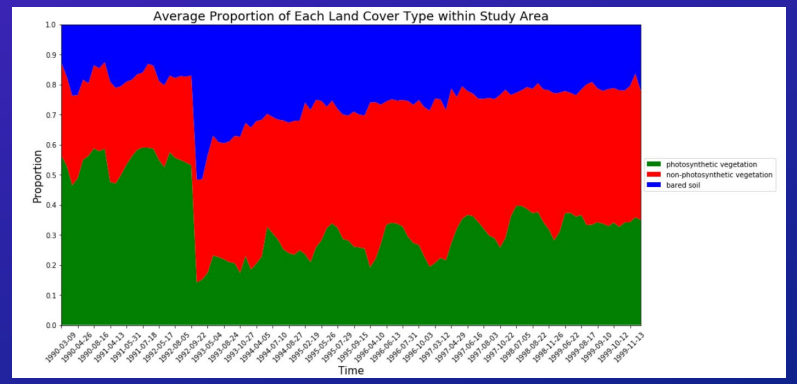
Kaixi

Use Case 2: Analyze Land Cover Change Using Linear Unmixing

- Input: a mosaic dataset contains 1087 Images
- Processing: linear unmixing function
- Output: time series fraction + statistics output







Mosaicking a Collection of Images

- **RasterCollection.mosaic (*mosaic_method*)**
  - rc.mosaic(‘LAST’)
  - rc.mosaic(‘FIRST’)

- Code snippet: mosaic RGB band of most recent images

```
# sort the collection from earlist to latest
sorted_rc = rc.sort('StdTime')
# choose Blue, Green, Red bands from each item in the collection
rgb_collection = sorted_rc.selectBands([2,3,4])
# mosaic
mosaic_raster = rgb_collection.mosaic('LAST')
# save
mosaic_raster.save(r"c:\data\mosaic_output.crf")
```

Processing Settings

- **Output formats**
  - CRF (.crf) : output one item or multiple items as a multidimensional raster
  - TIFF (.tif) : Output single item
- **Output compression**

```
arcpy.env.compression = "LERC 0.01"        # for controlled accuracy
arcpy.env.compression = "LERC 0"           # lossless
arcpy.env.compression = " LZ77 75"         # lossless
arcpy.env.compression = " JPEG 75"         # lossy
```

- **Parallel processing factor**
  - arcpy.env.parallelprocessingfactor = "6"
  - arcpy.env.parallelprocessingfactor = "90%"

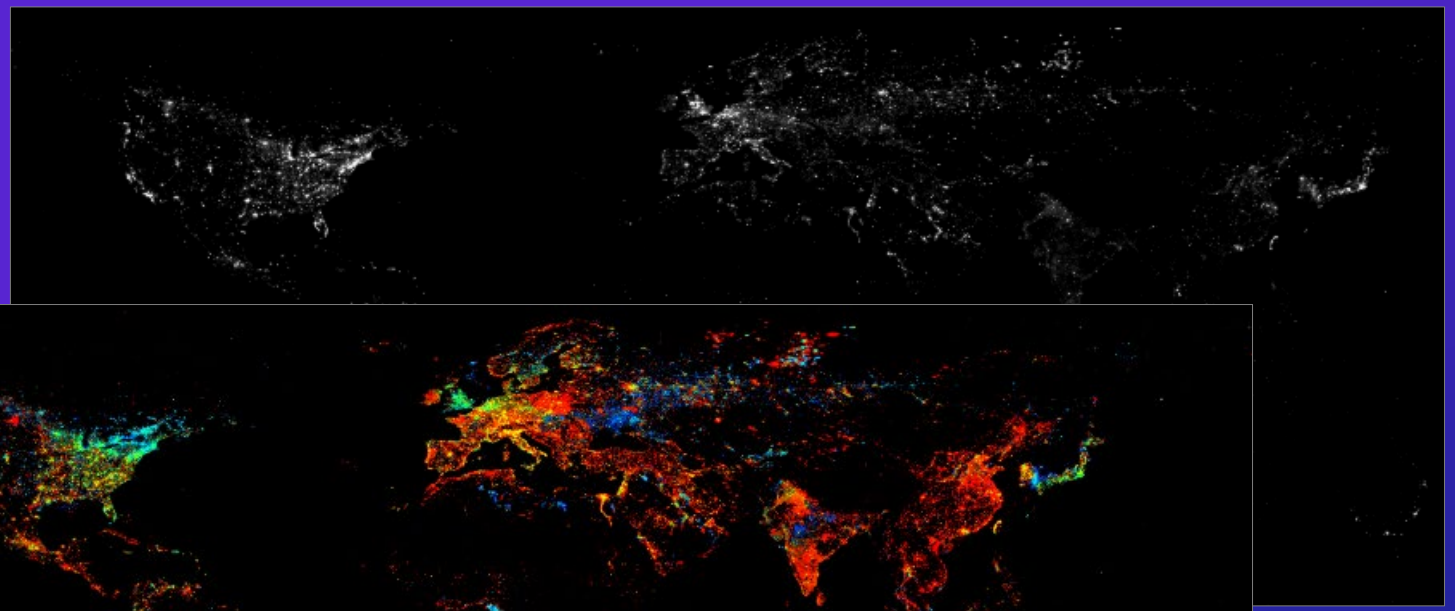
Help References

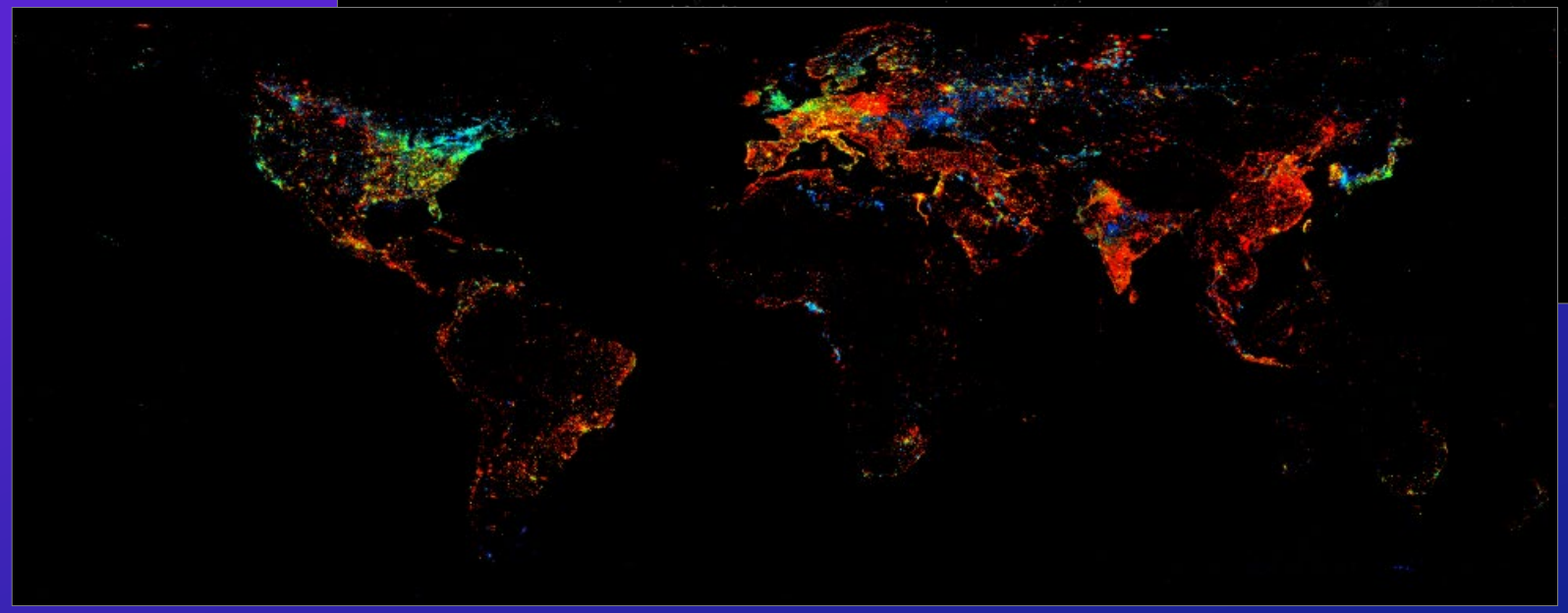
- https://pro.arcgis.com/en/pro-app/arcpy/image-analyst/rastercollection-class.htm
- https://pro.arcgis.com/en/pro-app/arcpy/image-analyst/an-overview-of-the-image-analyst-functions.htm

- **Same capabilities and APIs for *ArcGIS for Python* will be available in ArcGIS 10.8.1**

Understanding Urban Expansion Using Nightlight Imagery

- Yearly nightlight images (1992-2013)
- Trend function
- Render as RGB





Red: Increase_Trend
Green: P_Value
Blue: Decrease_Trend

```
# Fit Linear Regression Model
nightlight_linear_trend = arcpy.ia.Apply(nightLights_1992_2013, "TrendAnalysis",
                                         raster_function_arguments = {"DimensionName": "StdTime", "RegressionType": 0})
# check the dimension value of the trend output
nightlight_linear_trend.getDimensionValues("NightLightData", "StdTime")
```