2013 Esri International User Conference
July 8–12, 2013 | San Diego, California
Technical Workshop

Python – Raster Analysis

Kevin M. Johnston
Ryan DeBruyn
Nawajish Noman

The problem that is being addressed

- You have a complex modeling problem
- You are mainly working with rasters
- Some of the spatial manipulations that you trying to implement are difficult or not possible using standard ArcGIS tools
- Due to the complexity of the modeling problem, processing speed is a concern

Outline

- Managing rasters and performing analysis with Map Algebra
- How to access the analysis capability – Demonstration
- Complex expressions and optimization – Demonstration
- Additional modeling capability: classes – Demonstration
- Full modeling control: NumPy arrays – Demonstration
- Pre-10 Map Algebra

The complex model

Emerald Ash Borer

Originated in Michigan

Infest ash trees

100% kill

Coming to Vermont

The *Ash Borer* model

- Movement by flight
  - 20 km per year
  - Vegetation type and ash density (suitability surface)
- Movement by hitchhiking
  - Roads
  - Camp sites
  - Mills
  - Population
  - Current location of the borer (suitability surface)
- Random movement

The *Ash Borer* model

- Prepare the data
- An iterative model – based on a year
- Three sub models run individually each iteration and the results are combined
  - Movement by flight (run 3 different seasons)
  - Movement by hitchhiking (run once)
  - Random movement (run once)

Raster analysis – Preparing the data

- To prepare and manage raster data
  - Displaying
  - Adding, copying, deleting, etc.
  - Mosaic, Clip, etc.
  - Raster object
  - NumPy, ApplyEnvironment, etc.
- To perform analysis
  - Spatial Analyst
  - Map Algebra

What is Map Algebra

- Simple and powerful algebra to execute Spatial Analyst tools, operators, and functions to perform geographic analysis
- The strength is in creating complex expressions
- Available through Spatial Analyst module
- Integrated in Python (all modules available)

Importing Spatial Analyst

- Module of ArcPy site package
- Like all modules must be imported
- To access the operators and tools in an algebraic format the imports are important

```
import arcpy
from arcpy.sa import *
```

General syntax

- Map Algebra available through an algebraic format
- Simplest form: output raster is specified to the left of an equal sign and the tool and its parameters on the right

```
from arcpy.sa import  *
outRas = Slope(indem)
```

- Comprised of:
  - Input data
  - Tools
  - Output
  - Operators
  - Parameters

Input data

- Input elements
  - Dataset name or layer
  - Dataset with full path
  - Variable pointing to a raster
  - Raster object
  - Numbers and constants

```
outRas = Slope(inRaster)
```

Tip: It is good practice to set the input to a variable and use the variable in the expression. Dataset names are quoted.

```
inRaster1 = "C:/Data/elevation"
outRas = Slope(inRaster1)
```

Map Algebra operators

- Symbols for mathematical operations
- Many operators in both Python and Spatial Analyst
- Creating a raster object (Raster class) indicates operator should be applied to rasters

```
elevMeters = Raster("C:\data\elevation") * 0.3048
outSlope = Slope(elevMeters)
```

Map Algebra tools

- All Spatial Analyst tools are available (e.g., Sin, Slope, Reclassify, etc.)

```
outRas = Aspect(inRaster)
```

- Can use any Geoprocessing tools

Tip: Tool names are case sensitive

Tool parameters

- Defines how the tool is to be executed
- Each tool has its own unique set of parameters
- Some are required, others are optional
- Numbers, strings, and objects (classes)

```
outRas = Slope(inRaster, "PERCENT_RISE",  0.3048)
```

Map Algebra output

- Stores the results as a Raster object
- Object with methods and properties
- In scripting the output is temporary
- Associated data will be deleted if not explicitly saved

```
outRas = Hillshade(inRaster)
```

Access to Map Algebra

- Raster Calculator
  - Spatial Analyst tool
  - Easy to use calculator interface
  - Stand alone or in ModelBuilder
- Python window
  - Single expression or simple exploratory models
- Scripting
  - Complex models
  - Line completion and colors

The *Ash Borer* model

- Prepare the data
- An iterative model – based on a year
- Three sub models run individually each iteration and the results are combined
  - Movement by flight (run 3 different seasons)
  - Movement by hitchhiking (run once)
  - Random movement (run once)

Demo 1: Data management

Raster management tools

Raster Calculator

Python window

ModelBuilder

Simple expressions

Outline

- Managing rasters and performing analysis with Map Algebra
- How to access the analysis capability – Demonstration
- Complex expressions and optimization – Demonstration
- Additional modeling capability: classes – Demonstration
- Full modeling control: NumPy arrays – Demonstration
- Pre-10 Map Algebra

Complex expressions

- Multiple operators and tools can be implemented in a single expression

- Output from one expression can be input to a subsequent expression

```
inRaster = ExtractByAttributes(inElevation, "Value > 1000")
out = Con(IsNull(inRaster),inRaster,0)
```

More on the raster object

- A variable with a pointer to a dataset
- Output from a Map Algebra expression or from an existing dataset
- The associated dataset is temporary (from Map Algebra expression) - has a save method

```
outRas = Slope(inRaster)
outRas.save("sloperaster")
```

- Properties describing the associated dataset
  - Description of raster (e.g., number of rows)
  - Description of the values (e.g., mean)

Optimization

- A series of local tools (Abs, Sin, CellStatistics, etc.) and operators can be optimized
- When entered into a single expression each tool and operator is processed on a per cell basis

The *Ash Borer* model

- Prepare the data
- An iterative model – based on a year
- Three sub models run individually each iteration and the results are combined
  - Movement by flight (run 3 different seasons)
  - **Movement by hitchhiking (run once)**
  - Random movement (run once)

Movement by hitchhiking

- Hitchhike on cars and logging trucks
- Most likely spread around
  - Roads
  - Populated areas (towns and camp areas)
  - Commercial area (mills)
- Have a susceptible surface
  - Vegetation types and density of ash
  - Nonlinear decay
- Random points and check susceptibility

Demo 2: Movement by hitchhiking

Roads, Campsites, Mills, Population,and current location (suitability)

Complex expressions

Raster object

Optimization

Outline

- Managing rasters and performing analysis with Map Algebra
- How to access the analysis capability – Demonstration
- Complex expressions and optimization – Demonstration
- Additional modeling capability: classes – Demonstration
- Full modeling control: NumPy arrays – Demonstration
- Pre-10 Map Algebra

Classes

- Objects that are used as parameters to tools
  - Varying number of arguments depending on the parameter choice (neighborhood type)
  - The number of entries can vary depending on situation (remap table)
- More flexible
- Query the individual arguments

Classes - Categories

- General
  - Fuzzy
  - Horizontal Factor
  - KrigingModel
  - Neighborhood
  - Time
  - Vertical Factor
  - Radius
- Composed of lists
  - Reclass
  - Topo
  - Weighted reclass tables

General classes – some capability

- Creating

```
neigh = NbrCircle(4, "MAP")
```

- Querying

```
radius = neigh.radius
```

- Changing arguments

```
neigh.radius = 6
```

Classes composed of lists

- Topo

```
inContours = TopoContour([['contours.shp', 'spot_meter']])
```

- Reclassify

```
remap = RemapValue([["Brush/transitional", 0],
        ["Water", 1],["Barren land", 2]])
```

- Weighted Overlay

```
myWOTable = WOTable([[inRaster1, 50, "VALUE", remapsnow],
        [inRaster2, 20, "VALUE", remapland],
        [inRaster3, 30, "VALUE", remapsoil] ], [1, 9, 1])
```

Vector integration

- Feature data is required for some Spatial Analyst Map Algebra
  - IDW, Kriging, etc.
- Geoprocessing tools that operate on feature data can be used in an expression
  - Buffer, Select, etc.

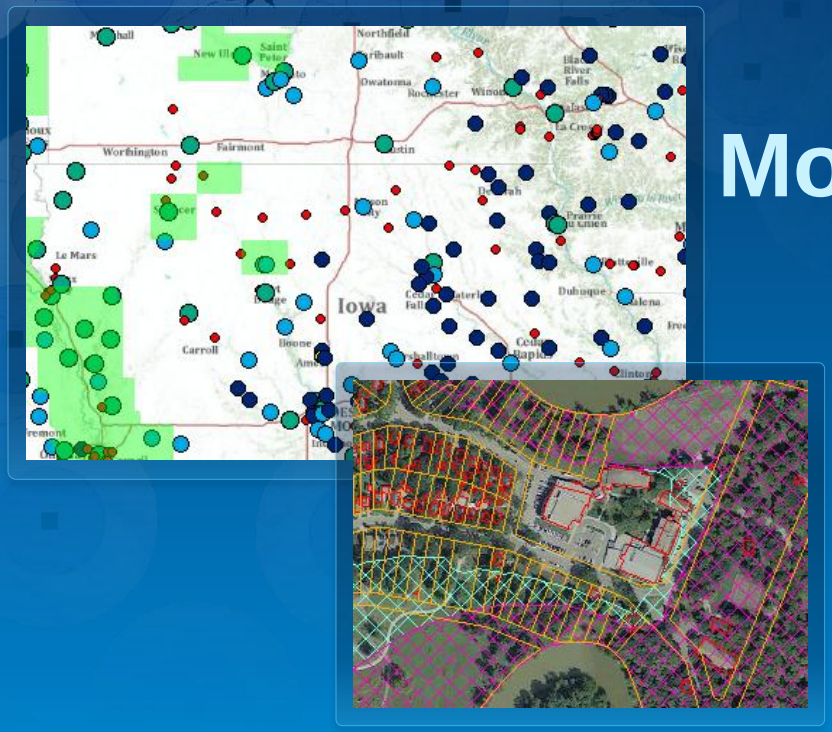
```
dist = EucDistance(arcpy.Select_analysis("schools", "#", "Pop>2000"))
```

The *Ash Borer* model

- Prepare the data
- An iterative model – based on a year
- Three sub models run individually each iteration and the results are combined
  - **Movement by flight (run 3 different seasons)**
  - Movement by hitchhiking (run once)
  - Random movement (run once)

Movement by flight

- Fly from existing locations - 20 km per year
- Based on iterative time steps
  - Spring, summer, fall, and winter
- Time of year determines how far it can move in a time step
- Suitability surface based on vegetation type and ash density
- Iterative movement logic
  - “Is there a borer in my neighborhood”
  - “Will I accept it” – suitability surface



Demo 3: Movement by flight

20 km per year

Vegetation type/ash density (suitability)

Classes

Using variables

Vector integration

Outline

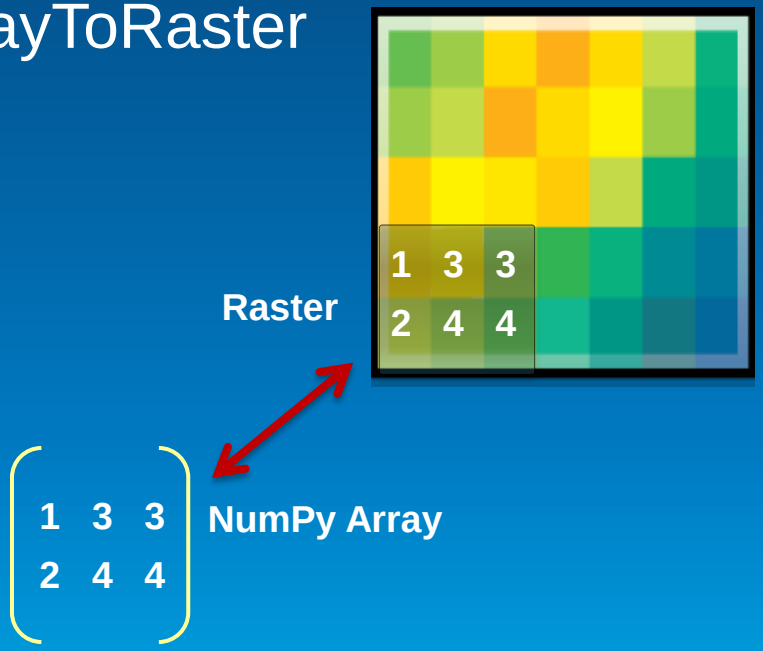
- Managing rasters and performing analysis with Map Algebra
- How to access the analysis capability – Demonstration
- Complex expressions and optimization – Demonstration
- Additional modeling capability: classes – Demonstration
- Full modeling control: NumPy arrays – Demonstration
- Pre-10 Map Algebra

NumPy Arrays

- A generic Python storage mechanism
- Create custom tool
- Access the wealth of free tools built by the scientific community
  - Clustering
  - Filtering
  - Linear algebra
  - Optimization
  - Fourier transformation
  - Morphology

NumPy Arrays

- Two tools
  - RasterToNumPyArray
  - NumPyArrayToRaster

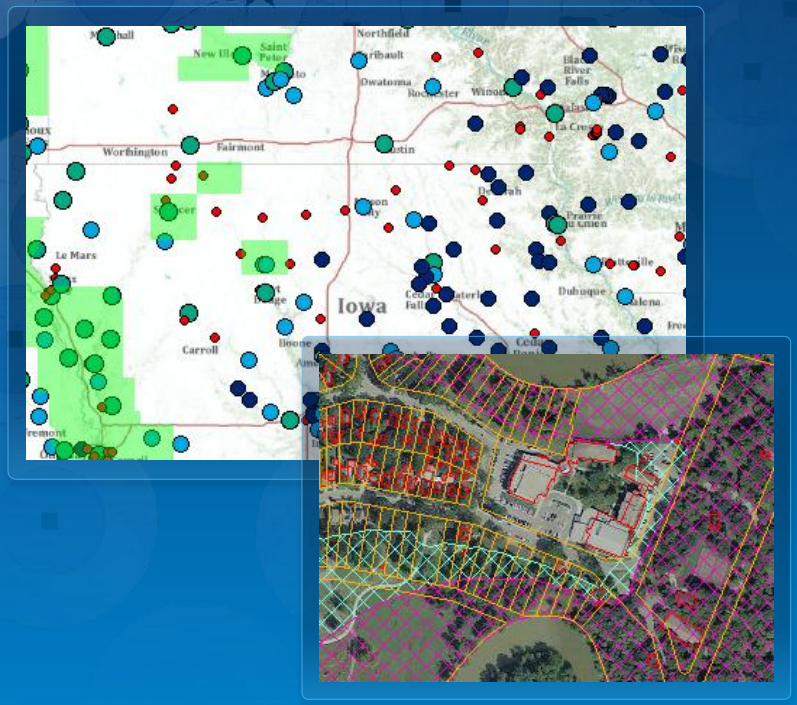


The *Ash Borer* model

- Prepare the data
- An iterative model – based on a year
- Three sub models run individually each iteration and the results are combined
  - Movement by flight (run 3 different seasons)
  - Movement by hitchhiking (run once)
  - **Random movement (run once)**

Random movement

- Some of the movement cannot be described deterministically
- Nonlinear decay from known locations
- Specific decay function not available in ArcGIS
- NumPy array
  - Export raster
  - Apply function
  - Import NumPy array back into a raster
- Return to ash borer model and integrate three movement sub models



Demo 4: The random movement

Random movement based on

nonlinear decay from existing locations

Custom function

NumPy array

Outline

- Managing rasters and performing analysis with Map Algebra
- How to access the analysis capability – Demonstration
- Complex expressions and optimization – Demonstration
- Additional modeling capability: classes – Demonstration
- Full modeling control: NumPy arrays – Demonstration
- Pre-10 Map Algebra

Pre-10.0 Map Algebra

- Similar to Map Algebra 10.0
- Faster, more powerful, and easy to use (line completion, colors)
- Any changes are to take advantage of the Python integration
- Raster Calculator at 10.0 replaces the Raster Calculator from the tool bar, SOMA, and MOMA
- SOMA in existing models will still work

Thank you…

Please fill out the session evaluation

First Offering ID: 1190

Second Offering ID: 1390

Online – www.esri.com/ucsessionsurveys

Paper – pick up and put in drop box

Spatial Analyst - Technical Sessions

- An Introduction - Rm 03

Tuesday, July 9, 8:30AM – 9:45AM
Wednesday, July 10, 1:30PM – 2:45PM

- Suitability Modeling - Rm 03

Tuesday, July 9, 10:15 AM – 11:30PM
Wednesday, July 10, 3:15PM – 4:30PM

- Python – Raster Analysis - Rm 03

Tuesday, July 9, 3:15PM – 4:30PM
Thursday, July 11, 8:30AM – 9:45PM

- Creating Surfaces – Rm 03

Wednesday, July 10, 8:30AM – 9:45PM

Spatial Analyst Technical Sessions (short)

- Creating Watersheds and Stream Networks – Rm 31C

Thursday, July 11, 10:15AM – 11:45AM

- Regression Analysis Using Raster Data – Hall G Rm 2

Wednesday, July 10, 10:30AM – 11:00AM

Demo Theater Presentations – Exhibit Hall B

- Modeling Rooftop Solar Energy Potential

Tuesday, July 9, 5:30PM – 6:00PM

- Surface Interpolation in ArcGIS

Wednesday, July 10, 9:00AM – 10:00AM

- Getting Started with Map Algebra

Tuesday, July 9, 10:00AM – 11:00AM

- Agent-Based Modeling

Wednesday, July 10, 1:00PM – 2:00PM

- Image Classification with Spatial Analyst

Tuesday July 9, 3:00PM – 3:30PM

Understanding our world.