Python – Raster Analysis

Kevin M. Johnston
Ryan DeBruyn
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
A 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**
Typical problem just like yours: The Characteristics

- **Complex**

- **Multiple input types**
  - Need to work with rasters along with features and tables

- **Scenarios**
  - Repeat analysis by using different parameter values

- **Dynamic**
  - Time is explicit, need to run sections multiple times

- **Enhanced capabilities**
  - Need to take advantage of 3rd party Python packages

- **Reusable**
  - Repeat the workflow with the same or different set of data

- **Performance and optimization**

  Ideal for Map Algebra and Python scripting
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 import env  # Analysis environment
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

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

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

- Rasters
- Features
- Numbers and text

\[
\text{outRas} = \text{Slope(} \text{inRaster} \text{)}
\]

- Objects
- Constants
- Variables

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

\[
\text{inRaster1} = "C:/Data/elevation"
\]
\[
\text{outRas} = \text{Slope(} \text{inRaster1} \text{)}
\]
Map Algebra operators

- Symbols for mathematical operations

- Many operators in both Python and Spatial Analyst

\[
\text{outRas} = \text{inRaster1} + \text{inRaster2}
\]

- Creating a raster object (\text{Raster class constructor - casting}) indicates operator should be applied to rasters

\[
\text{elevMeters} = \text{Raster("C:\data\elevation")} \times 0.3048
\]

\[
\text{outSlope} = \text{Slope(elevMeters)}
\]
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)

\[
\text{Slope}(\text{in\_raster}, \{\text{output\_measurement}\}, \{\text{z\_factor}\})
\]

\[
\text{outRas} = \text{Slope}(\text{inRaster}, \text{"DEGREE"}, 0.3048)
\]

\[
\text{outRas} = \text{Slope}(\text{inRaster}, \text{""}, 0.3048)
\]

\[
\text{outRas} = \text{Slope}(\text{inRaster})
\]

Tip: Keywords are in quotes
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
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 and accessing the capability

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
Complex expressions

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

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

```python
inRaster = ExtractByAttributes(inElevation, "Value > 1000")
out = Con(IsNull(inRaster), 0, inRaster)
```
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
  
  ```python
  outRas = Slope(inRaster)
  outRas.save("sloperaster")
  ```

- A series of 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
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

• Composed of lists
  - Reclass
  - Topo

• Time
  - Vertical Factor
  - Radius
  - Transformation functions

- Weighted reclass tables
General classes – Capability

• Creating
  
  \[\text{neigh} = \text{NbrCircle}(4, \text{"MAP")}\]

• Querying
  
  \[\text{radius} = \text{neigh.radius}\]

• Changing arguments
  
  \[\text{neigh.radius} = 6\]
Classes composed of lists

- Topo

\[
\text{inContours} = \text{TopoContour}([['\text{contours.shp}', '\text{spot\_meter}']])
\]

- Reclassify

\[
\text{remap} = \text{RemapValue}(["Brush/transitional", 0],
["Water", 1],["Barren land", 2])
\]

- Weighted Overlay

\[
\text{myWOTable} = \text{WOTable}([\text{inRaster1, 50, "VALUE", remapsnow}],
[\text{inRaster2, 20, "VALUE", remapland}],
[\text{inRaster3, 30, "VALUE", remapsoil} ]], [1, 9, 1])
\]
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.

\[
\text{dist} = \text{EucDistance(arcpy.Select\_analysis("schools", ",", "Pop>2000"))}
\]
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
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
Summary

- When the problem becomes more complex you may need additional capability provided by Map Algebra
- **Map Algebra** powerful, flexible, easy to use, and integrated into Python
- Accessed through: Raster Calculator, Python window, ModelBuilder (through Raster Calculator), and scripting
- Raster object and classes
- Create models that can better **capture interaction** of phenomena

- Demos are available online at [https://github.com/ryandebruyn/Working_with_raster_data_using_arcpy.git](https://github.com/ryandebruyn/Working_with_raster_data_using_arcpy.git)
Thank you...

- Please fill out the session survey:

  First Offering ID: 244/1191
  Second Offering ID: 244/1411

  Online – www.esri.com/ucsessionssurveys
  Paper – pick up and put in drop box