Python: Working with Raster Data

Nawajish Noman
Elizabeth Graham
Outline

• Managing rasters with tools and performing analysis with Map Algebra

• How to access the analysis capability
  - Demonstration

• Complex expressions and optimization
  - Demonstration

• Additional modeling capability using classes
  - Demonstration

• Extending modeling capability using NumPy arrays
  - Demonstration
The problem that is being addressed

• You have a complex modeling problem

• You are working with rasters, features and tables

• You want to write a script that is
  - **Reusable** - repeat the workflow with the same or different set of data
  - **Dynamic** – repeat analysis by using different parameter values
  - **Extends capabilities** - by taking advantage of 3rd party python packages
  - **Performs well** – optimized to improve execution speed
The ash borer model

• Movement by flight
  - Fly up to a half mile under its own power
  - Vegetation type and ash density (suitability surface)

• Movement by hitchhiking
  - Roads
  - Camp sites
  - Mills
  - Population
  - Current location of the borer (suitability surface)

• Random movement
Raster analysis

• To prepare and manage raster data
  - Displaying
  - Adding, copying, deleting, etc.
  - Mosaic, Clip, etc.
  - Raster object
  - NumPy, ApplyEnvironment, etc.

• To perform the analysis use raster analysis/modeling
  - 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)
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
Importing Spatial Analyst Module

- Module of ArcPy

- Like all modules must be imported

- To access the operators and tools in an algebraic format the imports are important

```python
import arcpy
from arcpy import env  # Analysis environment
from arcpy.sa import *  # BEST Practice for Map Algebra

import arcpy.sa  # NOT Recommended for Map Algebra
```
General syntax

• Write algebraic expression to perform Map Algebra

• 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
  - Operators, Tools and Parameters
  - Output
Inputs for analysis

- Rasters
- Features
- Numbers
- Text
- Objects
- Constants
- Variables

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

- Symbols for mathematical operations

- Many operators in both Python and Spatial Analyst

\[
\text{outRas1} = \text{inRaster1} + \text{inRaster2}
\]

- Cast the raster (Raster class constructor) to indicate operator should be applied to rasters

\[
\text{outRas1} = \text{Raster(“rastername1”) + Raster(“rastername2”)}
\]
\[
\text{outRas2} = \text{outRas1} + 8
\]
\[
\text{outRas3} = \text{outRas2} + \text{Raster(“rastername2”) \times 8}
\]
Map Algebra tools

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

```python
outRas = Slope(inRaster, "DEGREE", 0.3048)
outRas = Aspect("rastername")
```

- 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)

Slope($in\_raster$, \{output\_measurement\}, \{z\_factor\})

- $outRas = $Slope($inRaster$, “DEGREE”, 0.3048)
- $outRas = $Slope($inRaster$, “”, 0.3048)
- $outRas = $Slope($inRaster$)

*Tip: Keywords are in quotes and it is recommended they are capitalized*
Map Algebra output

- Returns the results as a **Raster object**
- Object with methods and properties
- The output is **temporary**
Demo 1
Data management

- Raster management tools
- Raster Calculator
- Python window
- Simple expressions
Complex expressions

- Multiple operators and tools can be executed in a single expression
- Output from one expression can be the input to a subsequent expression

*Tip: It is a good practice to set the input to a variable and use the variable in the expression*
More on the raster object

• A variable pointing to a dataset

• Output from a Map Algebra expression or pointing to an existing dataset

• The associated dataset is temporary (when created from Map Algebra) but has a save method

• A series of properties describing the associated dataset
  - Description of raster (e.g., number of rows)
  - Description of the values (e.g., mean)
Optimization

- Operators and local tools work on a per-cell basis

- A series of local tools (Abs, Sin, Cell Statistics, etc.) and operators can be optimized

- When entered into a single expression all local tools and operators are processed together on a per cell basis
The iterative aspects of the ash borer model

• Movement by hitchhiking
  - Based on highly susceptible areas
  - Nonlinear decay
  - Random points and check susceptibility

• Movement by flight
  - Depends on the year how far it can move in a time step
  - “Is there a borer in my neighborhood”
  - “Will I accept it” – suitability surface

• Random movement
  - Nonlinear decay from known locations (NumPy array)
Demo 2
Movement by hitchhiking

- Roads, Campsites, Mills, Population, and current location (suitability)
- Complex expressions
- Raster object
- Optimization
Classes

• Objects that are used as inputs to tools
  - Varying number of arguments depending on the selected parameter choice (neighborhood type)
  - The number of entries into the parameters can vary depending on the specific situation (a remap table)

• More flexible

• You can query and modify the individual arguments
Classes (contd.)

- Creating
  
nbr01 = NbrCircle(4, “MAP”)

- Querying
  
  radius01 = nbr01.radius

- Changing arguments
  
nbr01.radius = radius01 + 2
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.
The iterative aspects of the ash borer model

- **Movement by hitchhiking**
  - Based on highly susceptible areas
  - Nonlinear decay
  - Random points and check susceptibility

- **Movement by flight**
  - Depends on the year how far it can move in a time step
  - “Is there a borer in my neighborhood”
  - “Will I accept it” – suitability surface

- **Random movement**
  - Nonlinear decay from known locations (NumPy array)
Demo 3
Movement by flight

- 20 km per year
- Vegetation type/ash density (suitability)
- Classes
- Using variables
- Vector integration
NumPy

- An extension package to Python
  - Adds support for large, multi-dimensional arrays
  - Provides a large library of high-level mathematical functions

- Can be used to extend raster analysis capabilities by creating
  - custom functions
  - custom tools

- Access the wealth of free functions built by the scientific community
  - Clustering
  - Filtering
  - Linear algebra
  - Optimization
  - Fourier transformation
  - Morphology
NumPy Arrays

- Two functions to work with raster
  - `RasterToNumPyArray(in_raster, {lower_left_corner}, {ncols}, {nrows}, {nodata_to_value})`
  - `NumPyArrayToRaster(in_array, {lower_left_corner}, {x_cell_size}, {y_cell_size}, {value_to_nodata})`
The iterative aspects of the ash borer model

• Movement by flight
  - Depends on the year how far it can move in a time step
  - “Is there a borer in my neighborhood”
  - “Will I accept it” – suitability surface

• Movement by hitchhiking
  - Based on highly susceptible areas
  - Nonlinear decay
  - Random points and check susceptibility

• Random movement
  - Nonlinear decay from known locations (NumPy array)
Demo 4
The random movement

- Random movement based on nonlinear decay from existing locations
- Custom function
- NumPy array
Summary

• When the problem become 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 http://www.arcgis.com/home/item.html?id=5a42c7c368e444818615c584fdddb13a