Python – Raster Analysis

Kevin M. Johnston

Nawajish Noman

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

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

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

Input for analysis

- Rasters
- Features
- Numbers and 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.

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

Map Algebra operators

- Symbols for mathematical operations
- Many operators in both Python and Spatial Analyst

```
outRas = inRaster1 + inRaster2
```

- Creating a raster object (Raster class constructor - casting) 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

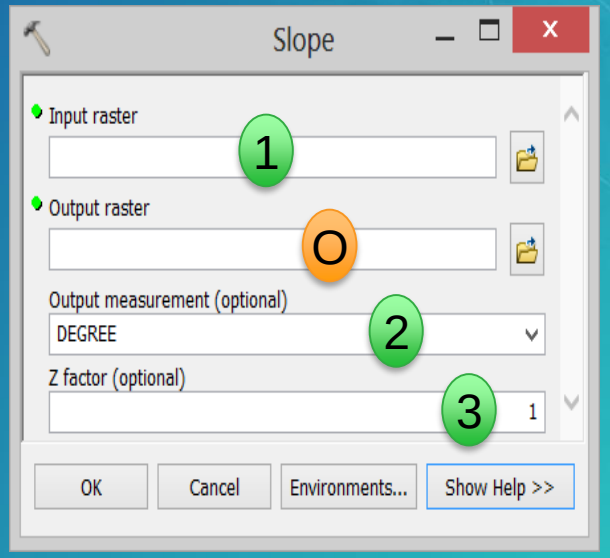
- Defines how the tool is to be executed
- Each tool has its own unique set of parameters
- Some are required, others are optional
- Numbers, strings, and objects (classes)

1 2 3

```
Slope(in_raster, {output_measurement}, {z_factor})
```

O

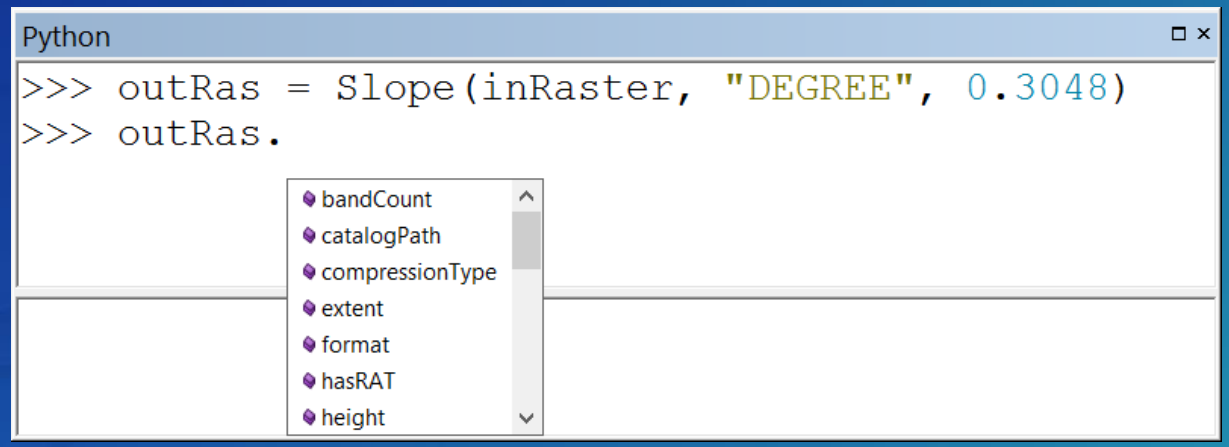
```
outRas = Slope(inRaster, "DEGREE", 0.3048)
outRas = Slope(inRaster, "", 0.3048)
outRas = Slope(inRaster)
```



Tip: Keywords are in quotes

Map Algebra output

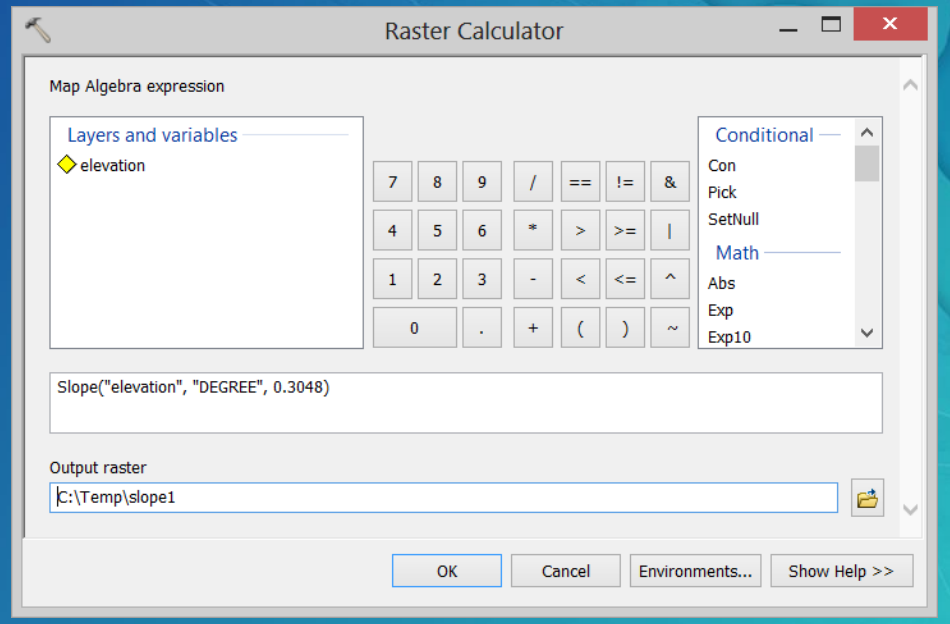
- Stores the results as a Raster object
- Object with methods and properties

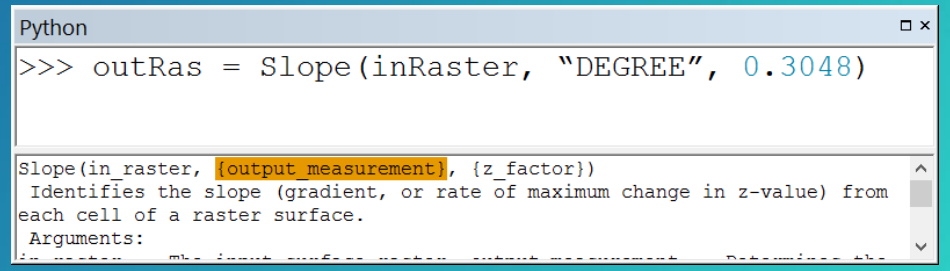


- In scripting the output is temporary
- Associated data will be deleted if not explicitly saved

Access to Map Algebra

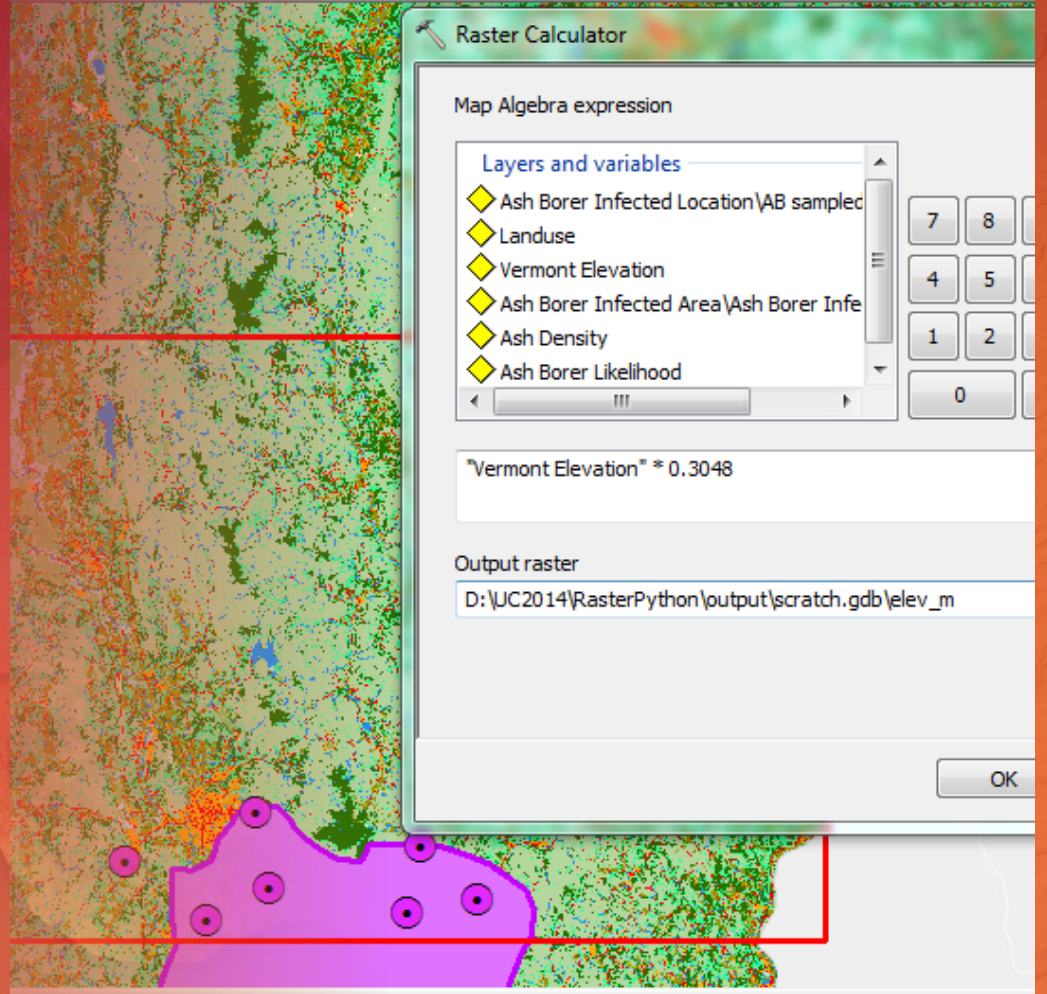
- **Raster Calculator**
  - Spatial Analyst tool
  - Easy to use calculator interface
  - Stand alone or in ModelBuilder
- **Python window**
  - Single expression or simple exploratory models
- **Scripting**
  - Complex models
  - Line completion and colors





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)**




```
tRaster = Slope(Raster("Vermont Elevation") * 0.3048)
```

```
_raster, {output_measurement}, {z_factor})
ies the slope (gradient, or rate of maximum change in z-value) from
 surface.
ts:
r -- The input surface raster. output_measurement -- Determines the
egrees or percentages) of the output slope data. z_factor -- Number
```

Demo

Data management and accessing the capability

- Raster management tools
- Raster Calculator
- Python window
- Model Builder
- Simple expression

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

```
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

```
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

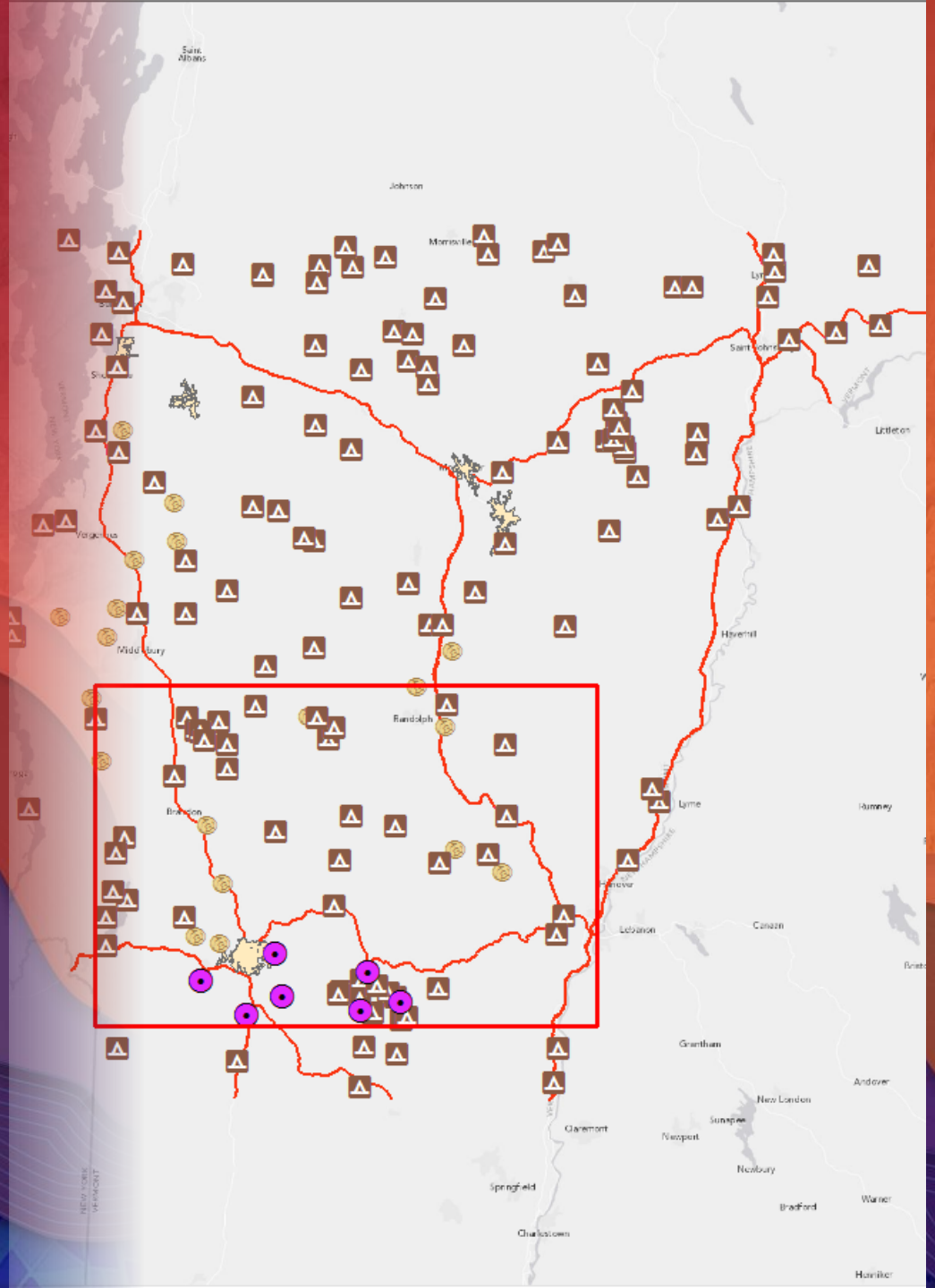
- A series of local tools (Abs, Sin, CellStatistics, etc.) and operators can be optimized
- When entered into a single expression each tool and operator is processed on a per cell basis

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 susceptibility surface
  - Vegetation types and density of ash
  - Nonlinear decay
- Random points and check susceptibility



Demo

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

Syntax

NbrRectangle ({width}, {height}, {units})

Parameter	Explanation	Data Type
width	The width of the rectangle neighborhood. If only the `width` is specified, the resulting neighborhood is a square. (The default value is 3)	Double
height	The height of the rectangle neighborhood. If only the `height` is specified, the resulting neighborhood is a square. (The default value is 3)	Double
units	Defines the units of the neighborhood. • CELL —The unit of measurement is in cells. • MAP —The units are in map coordinates. (The default value is CELL)	String

Properties

Property	Explanation	Data Type
width (Read and Write)	The width of the rectangle neighborhood. If only the `width` is specified, the `height` will default to the same as the `width`, resulting in a square neighborhood.	Double
height (Read and Write)	The height of the rectangle neighborhood. If only the `height` is specified, the `width` will default to the same as the `height`, resulting in a square neighborhood.	Double
units (Read and Write)	Defines the units of the neighborhood.	String

Classes - Categories

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

General classes - Capability

- Creating

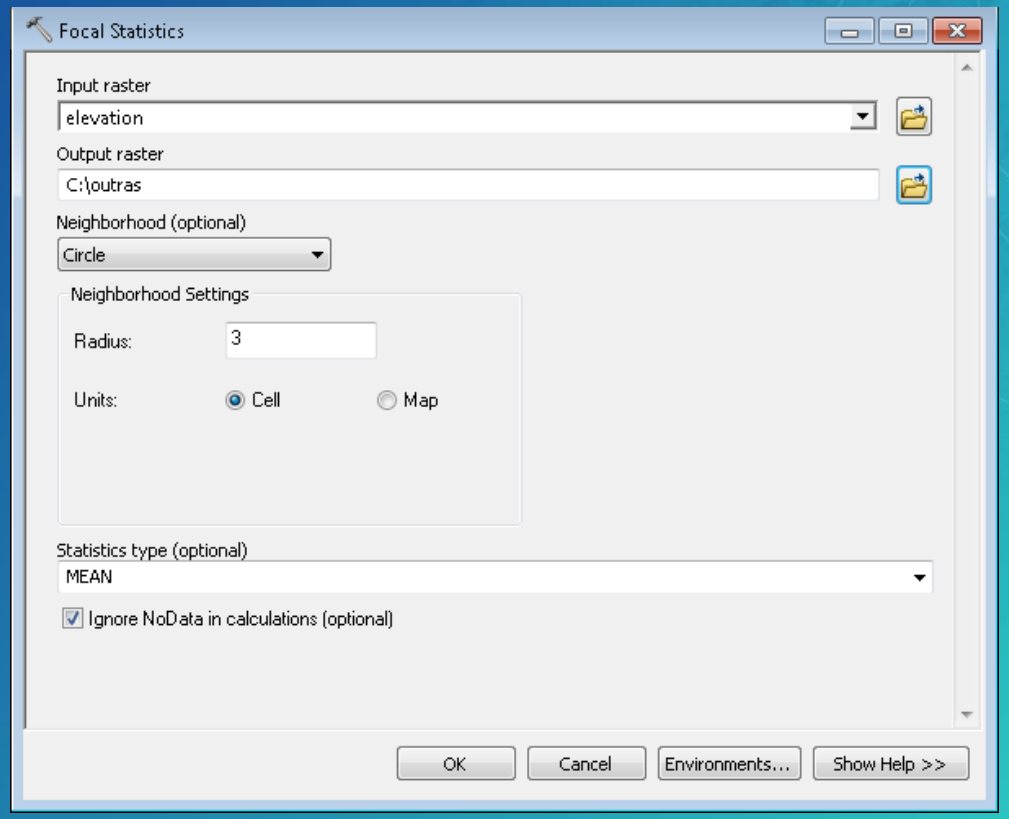
```
neigh = NbrCircle(4, "MAP")
```

- Querying

```
radius = neigh.radius
```

- Changing arguments

```
neigh.radius = 6
```



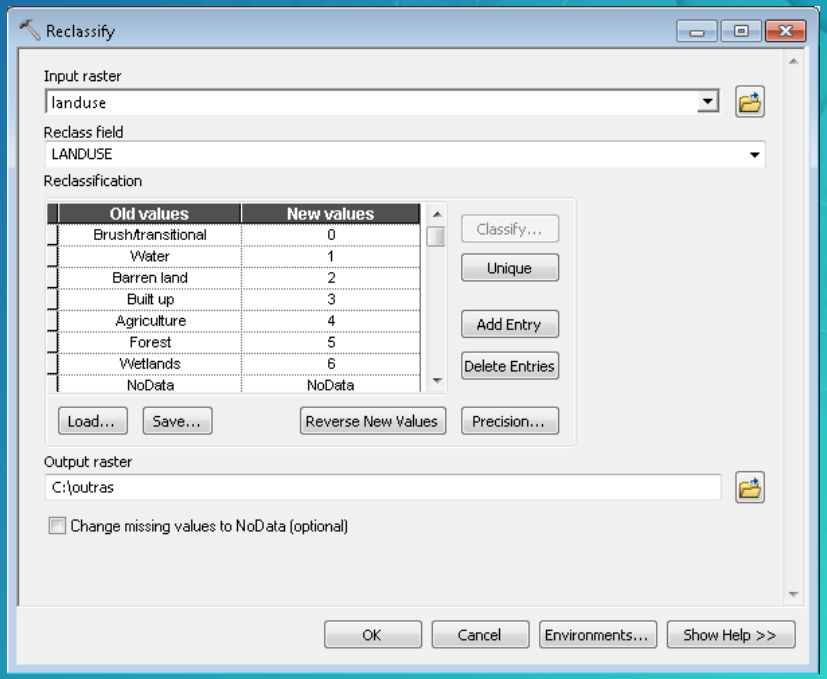
Classes composed of lists

- **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.

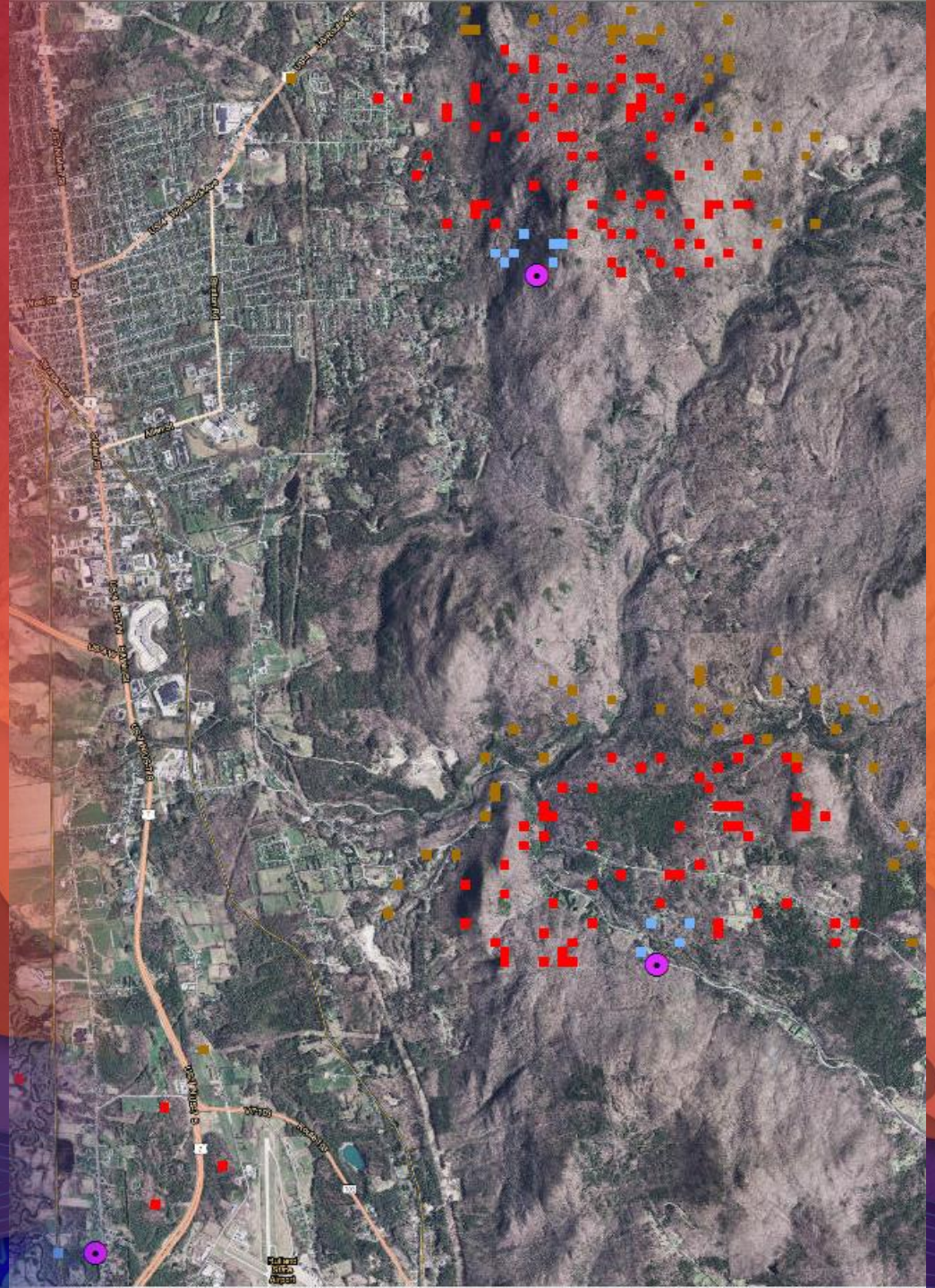
```
dist = EucDistance(arcpy.Select_analysis("schools", "#", "Pop>2000"))
```

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

Movement by flight

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

Outline

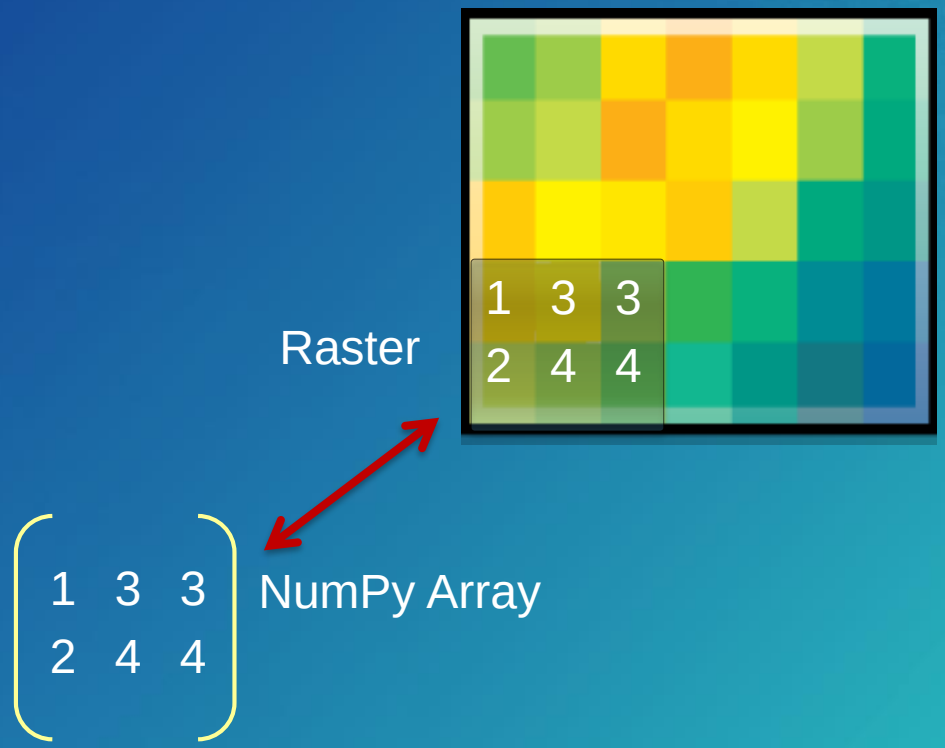
- Managing rasters and performing analysis with Map Algebra
- How to access the analysis capability
  - Demonstration
- Complex expressions and optimization
  - Demonstration
- Additional modeling capability: classes
  - Demonstration
- Full modeling control: NumPy arrays
  - Demonstration

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**



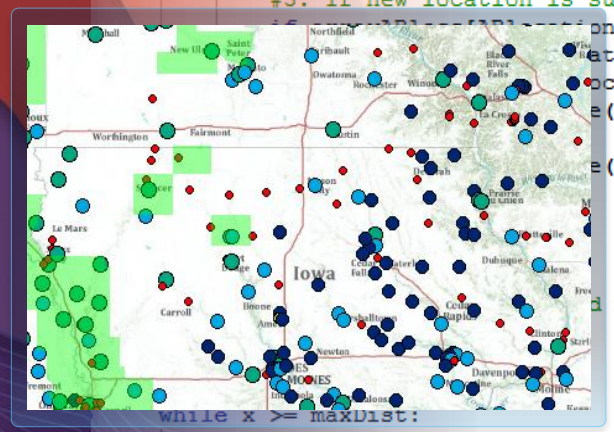
The Ash Borer model

- 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

```
'''#########################################################
# AB_RandomMovement.py
#
# Created: 03/27/13
# Author: ESRI
#########################################################'''

def getABlocations(array1):
    ind = array1.nonzero()
    return ind

def randomABMovment(arrayABlocs, arrayABsus, maxDist):
    #Get current Ash Borer locations from array (1 = infested; 0 = not in
    ABlocations = getABlocations(arrayABlocs)
    #Reduce array for every third
    ABlocations3 = ABlocations[0][::3],ABlocations[1][::3]
    nlocs = len(ABlocations3[0])
    #for each input AB location
    for i in range(nlocs):
        p = ABlocations3[0][i],ABlocations3[1][i]
        arcpy.AddMessage("current location: %s" %(str(p)))
        # Test three random movements
        for i in range(3):
            #1. Generate random movement distance.
            randomXYdist = getRandomXYDist(maxDist)

            #2. New location for Ash Borer movement
            ABlocation_new = p[0] - randomXYdist[0], p[1] + randomXYdist[
            arcpy.AddMessage("new location %s" %(str(ABlocation_new)))

            #3. If new location is susceptible AB Infest = TRUE
                                                    on_new] != 1:
                                                    ation_new] >= 8:
                                                    ocation_new] = 1
                                                    e("Infested")

                                                    e("Not Infested")

                                                    d mean. It should be nonzero.

        while x >= maxDist:
            x = int(random.expovariate(1.0/lambd))
        while y < - maxDist or y > maxDist:
            while ypos >= maxDist:
                ypos = int(random.expovariate(1.0/lambd))
            while yneg <= -(maxDist):
                yneg = int(random.expovariate(-1.0/lambd))
```



Demo

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

Other Spatial Analyst sessions

- **Spatial Analyst: An Introduction**
  - Tues 10:15 – 11:30
  - Wed 10:15 – 11:30
- **Finding the Best Locations Using Suitability Modeling**
  - Tues 1:30 – 2:45
  - Thurs 8:30 – 9:45
- **Identifying the Best Paths with Cost Distance**
  - Tues 3:15 – 4:30
  - Wed 1:30 – 2:45
- **Suitability Modeling and Cost Distance Analysis Integrated Workflow (Demo Theater)**
  - Wed 4:30 – 5:15
- **Python: Raster Analysis**
  - Tues 8:30 – 9:45
- **Getting Started With Map Algebra Using the Raster Calculator and Python (Demo Theater)**
  - Thurs 9:30 – 10:15

Other Spatial Analyst sessions

- Modeling Renewable Energy Potential Using ArcGIS (Demo Theater)
  - Tues 1:30 – 2:15
- Creating Watersheds and Stream Networks
  - Wed 10:00 – 10:30
- Hydrologic and Hydraulic Modeling
  - Wed 3:15 – 4:30
  - Thurs 1:30 – 2:45
- GIS Techniques for Floodplain Delineation (Demo Theater)
  - Tues 12:30 – 1:15
- Creating a Hydrologically Conditioned DEM (Demo Theater)
  - Tues 10:30 – 11:15
- Creating Surfaces from Various Data Sources
  - Tues 3:15 – 4:30
  - Thurs 3:15 – 4:30
- Choosing the Best Kriging Model for Your Data (Demo Theater)
  - Wed 11:30 – 12:15

Other Spatial Analyst sessions

- Surface Interpolation in ArcGIS (Demo Theater)
  - Thurs 10:30 – 11:15
- Creating Watersheds and Stream Networks (Demo Theater)
  - Wed 10:00 – 10:30
- Working with Elevation Services (Demo Theater)
  - Tues 10:30 – 11:15
  - Wed 9:30 – 10:15
- Building Python Raster Functions (Demo Theater)
  - Tues 10:30 – 11:15
- Raster Analytics in Image Server: An Introduction
  - Wed 3:15 – 4:30
- Raster Classification with ArcGIS Desktop (Demo Theater)
  - Thurs 9:30 – 10:15
- Raster Function Processing (Demo Theater)
  - Thurs 10:30 – 11:15

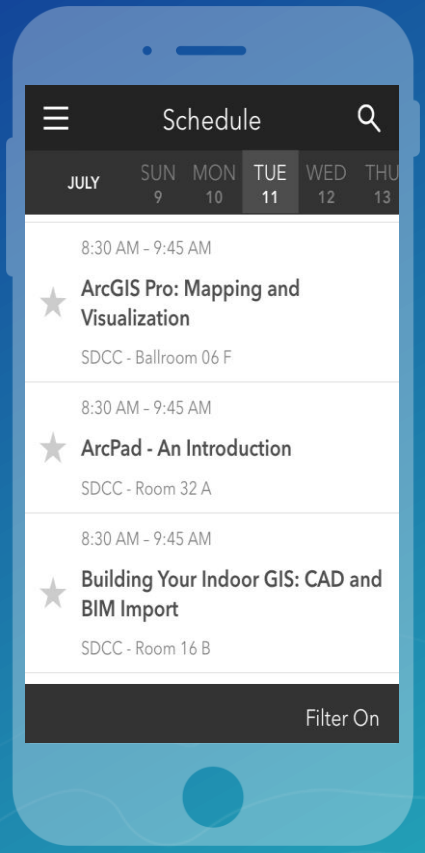
Want to learn more?

- **Documentation**
  - ArcGIS Pro Help
  - Terminology and user interface reference guide
- **Related Esri Training and Tutorials**
  - Introduction to ArcGIS Pro for GIS Professionals (Instructor Led)
  - Getting Started with ArcGIS Pro (Virtual Campus)
  - Get Started with ArcGIS Pro (Learn ArcGIS)
- **Additional Resources**
  - ArcGIS Pro Site

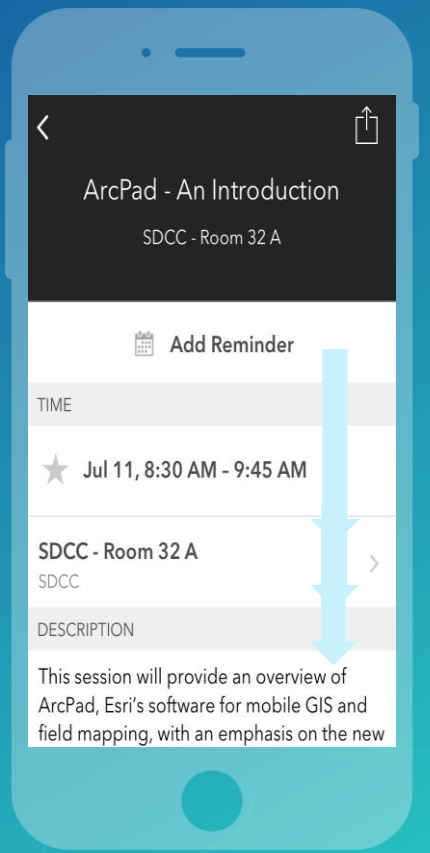
Please Take Our Survey on the **Esri Events App!**

Download the Esri Events app and find your event

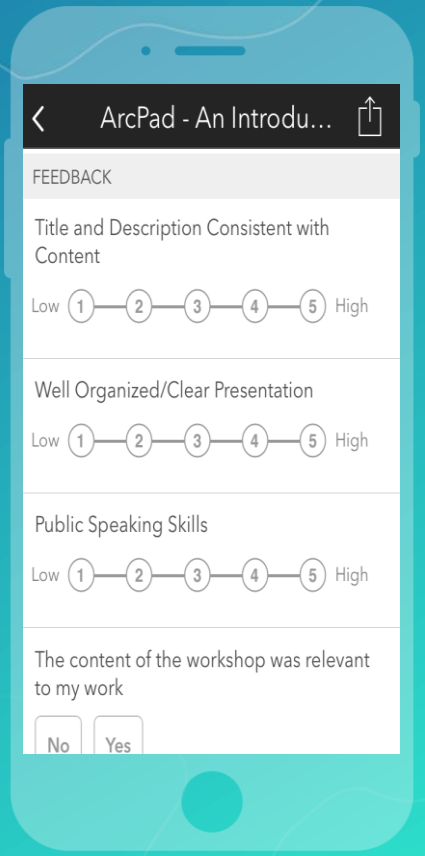
Select the session you attended



Scroll down to find the survey



Complete Answers and Select "Submit"




esri
THE
SCIENCE
OF
WHERE