Overview for the 2007 ESRI User Conference
21-Jun-2007

Jason Roberts, Ben Best, Daniel Dunn, and Pat Halpin
Duke University Marine Geospatial Ecology Lab
Marine Geospatial Ecology Tools

• Geoprocessing toolbox for marine ecology
  • Oceanographic data management and analysis
  • Sophisticated sampling and statistical modeling
  • Emphasis on batch processing and interoperability
• Open source, implemented mostly in Python
  • Tools are platform independent, when possible
  • Some tools do not even require ArcGIS
• Minimum requirements: Python 2.4, ArcGIS
Talk Outline

- History of MGET
- Walkthrough typical user scenario
  - Highlight interesting tools and features
- Invitation to collaborate
- Advanced topics: (time permitting)
  - Ben’s Connectivity Modeler
  - How to build an MGET tool
History of MGET

- We have produced many geoprocessing tools but have done a poor job sharing them
  - Staff developed tools independently
  - Tools shared ad hoc with collaborators
  - Little effort to package and document tools for easy re-use by anonymous users
- It is time to unify our efforts!
ArcRStats by Ben Best

Toolbox for sampling raster layers and running statistical analyses to predict habitats

- Random Points
- pts_rand
- Sample to Table
- tbl_env
- Statistical Plots
- dir_plots
- Multivariate Regression GLM
- rstr_glm
- rstr_glmroc
- <> Lakes
- rstr_landcov
- rstr_dem
- rstr_aspect
- rstr_tci
- pts_obs
- rstr_viable

Statistical Plots

Multivariate Regression GLM
Marine Ecology Tools by Jason Roberts

Unreleased toolbox for batch processing of oceanography
Benthic Complexity Modeler by Daniel Dunn

Predicts hardbottom from coarse-grain (90 m) bathymetry.

Statistical model: GLM with logit link.

Substrate Type ~ depth difference + aspect variety + slope + error

Probability of hardbottom
Unifying Our Tools

ArcRStats, HabMod, ConnMod
Marine Ecology Tools
Benthic Complexity Modeler
E. Treml’s Connectivity Tools
And others

MGET
Typical User Scenario

- Researcher has spatially-explicit observations of a species and wants to investigate why it is there
  - Are there spatial and temporal patterns?
  - Correlations with environmental conditions?
  - Correlations with occurrence of other species?
  - Can we predict its occurrence and thereby improve our management of it?
Typical Observation Data

Fishery bycatch data

IATTC Olive Ridley Encounters 1990-2005

Surveys

45 SEAMAP data sets
1991 – 2002

Scott Eckert

Argos satellite tracks

Data Providers:
NOAA/NMFS – NEFSC
NOAA/NMFS – SEFSC
UNC-W
Typical Workflow

MGET includes (or will include) tools that assist with all steps.
Typical Workflow

1. **Import species observations into GIS**
2. **Download oceanographic datasets**
3. **Prepare oceanographic data for use**
4. **Sample oceanographic data**
5. **Explore maps of oceanography and observations**
6. **Create derived oceanographic datasets**
7. **Model species habitat or behavior**
### Import Argos Results Into Tables

#### DIAG record

<table>
<thead>
<tr>
<th>Date</th>
<th>LC</th>
<th>IQ</th>
<th>Lat1</th>
<th>Lon1</th>
<th>Lat2</th>
<th>Lon2</th>
<th>Nb mes</th>
<th>Nb mes&gt;-120dB</th>
<th>Best level</th>
<th>Calcul freq</th>
<th>Altitude</th>
<th>Pass duration</th>
<th>NOPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.12.03</td>
<td>2</td>
<td>50</td>
<td>34.282N</td>
<td>77.010W</td>
<td>32.640N</td>
<td>69.121W</td>
<td>012</td>
<td>000</td>
<td>-121 dB</td>
<td>401 647695.6 Hz</td>
<td>0 m</td>
<td>706s</td>
<td>2</td>
</tr>
</tbody>
</table>

#### DS record

<table>
<thead>
<tr>
<th>Date</th>
<th>LC</th>
<th>IQ</th>
<th>Lat1</th>
<th>Lon1</th>
<th>Lat2</th>
<th>Lon2</th>
<th>Nb mes</th>
<th>Nb mes&gt;-120dB</th>
<th>Best level</th>
<th>Calcul freq</th>
<th>Altitude</th>
<th>Pass duration</th>
<th>NOPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003-12-03</td>
<td>1</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>02</td>
<td>106</td>
<td>40</td>
<td>230</td>
<td>97</td>
<td>145</td>
<td>44</td>
<td>116</td>
<td>119</td>
<td>119</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>158</td>
<td>139</td>
<td>118</td>
<td>193</td>
<td>80</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>211</td>
<td>214</td>
<td>72</td>
<td>218</td>
<td>17</td>
<td>29</td>
<td>159</td>
<td>218</td>
<td>17</td>
<td>17</td>
<td>29</td>
<td>159</td>
<td>218</td>
<td>17</td>
</tr>
</tbody>
</table>
Import Argos Results Into Tables

- DS Folder
- DIAG Folder
- Result Table
- Message Table
Table $\rightarrow$ Points $\rightarrow$ Lines

XY Table to Point Feature Class

Connect Points

Input Table $\rightarrow$ Point Features $\rightarrow$ Line Features
Typical Workflow

1. Import species observations into GIS
2. Download oceanographic datasets
3. Explore maps of oceanographic and observations
4. Sample oceanographic data
5. Prepare oceanographic data for use
6. Create derived oceanographic datasets
7. Model species habitat or behavior
Calculate URLs and Download

1. Given points’ dates, calculates URL to NOAA SST HDFs
2. Given URLs, download the files to a given directory
Calculate URLs and Download

NODC AVHRR v5 URLs for Dates

- **Input Table**: Observations.shp (2)
- **Input Field Containing Dates**: ObsDate
- **Output Field to Receive URLs**: URL
- **Select Expression (optional)**
- **Temporal Resolution**: 8 Day
- **Sensor Pass**: Ascending (daytime)
- **Data Type**: sst: all-pixel sea surface temperature

Download Files

- **Input Table**: Observations.shp
- **Input Field Containing URLs**: URL
- **Destination Folder**: Download Folder
- **Output Field to Receive File Paths (optional)**: File
- **Select Expression (optional)**
- **Skip Downloads When Destination Files Exist (optional)**
- **Stop Geoprocessing After This Many Failures (optional)**: 1
Typical Workflow

1. Import species observations into GIS
2. Download oceanographic datasets
3. Prepare oceanographic data for use
4. Explore maps of oceanogr. and observations
5. Sample oceanographic data
6. Create derived oceanographic datasets
7. Model species habitat or behavior
Preparing Oceanography For Use

• Most oceanographic datasets are not immediately usable

• Common preprocessing steps include:
  • Converting to a supported format
  • Projecting to a desired projection
  • Clipping to region of interest
  • Performing arbitrary map algebra
  • Building pyramids
MGET Tools for Oceanography

Implemented in three layers:

1. Single-input, single output for general format
   - HDF to ArcGIS Raster

2. Batch processing versions for general format
   - HDFs Listed in Table to ArcGIS Rasters
   - Find HDFs and Convert to ArcGIS Rasters

3. Specialized versions for particular products
   - NODC AVHRR v5 HDF to ArcGIS Raster
   - NODC AVHRR v5 HDFs Listed in Table to Rasters
   - Find NODC AVHRR v5 HDFs and Convert to Rasters
Example: HDF to ArcGIS Raster
Batch Processing Design Pattern 1

“Process inputs listed in table” pattern:

- Table fields contain the paths to the inputs to process and the outputs to produce
- User can populate these columns using any technique (e.g. Download Files tool)
- The batch tool accepts a SQL where clause to select the rows to process, and an order by clause to specify the
  id
Example: SDSes in HDFs Listed in Table to ArcGIS Rasters

Same as single-file tool
Batch Processing Design Pattern 2

“Find and process inputs” pattern:

- User specifies:
  - Input and output locations (e.g. workspaces)
  - Optional search parameters (e.g. wildcard)
  - Python expression for naming outputs (a sensible default is always provided)
- The batch tool searches the input location, processes all inputs that are found and stores them in the output
Example: SDSes in HDFs Listed in Table to ArcGIS Rasters

Same as single-file tool
Example: SDSes in HDFs Listed in Table to ArcGIS Rasters
Example Product-Specific Tool

NOAA NODC 4km AVHRR Pathfinder v5 SST
Other SST Products

PO.DAAC GOES 10/12

NOAA CoastWatch AVHRR

Also: PO.DAAC MODIS Aqua and Terra, GOES 9
Sea Surface Chlorophyll

NASA OceanColor Group SeaWiFS

Also: MODIS Aqua and combined MODIS/SeaWiFS
AVISO SSH and Geostrophic Currents

PO.DAAC QuickSCAT Winds

Also: BYU QuickSCAT Sigma-o
(approximate sea surface roughness)
Global Bathymetries

- ETOPO2
- GEBCO
- S2004

Map shows S2004 clipped to ETP

Typical Workflow

1. Import species observations into GIS
2. Download oceanographic datasets
3. Prepare oceanographic data for use
4. Explore maps of oceanographic and observations
5. Sample oceanographic data
6. Model species habitat or behavior
7. Create derived oceanographic datasets
Identifying Surface Temperature Fronts

Cayula-Cornillion Edge Detection Algorithm (1992)

Step 1: Histogram analysis

- Bimodal distribution with an optimal break at 27.0°C

Step 2: Spatial cohesion test

- Strong cohesion → front present
- Weak cohesion → no front

Example Output

- Processed all daily AVHRR 4 km images from 1985-2005
- Over 15,000 images, requiring 2 months of computer time
- Also processed GOES 10 and 12 images
Typical Workflow

1. Import species observations into GIS
2. Download oceanographic datasets
3. Prepare oceanographic data for use
4. Explore maps of oceanographic and observations
5. Model species habitat or behavior
6. Create derived oceanographic datasets
7. Sample oceanographic data
Batch Sampling Tool

- Sample rasters in 1 or more fields
- Stores values directly in fields!
- Can apply Python expression to sampled values
Typical Workflow

1. Import species observations into GIS
2. Download oceanographic datasets
3. Prepare oceanographic data for use
4. Explore maps of oceanographic and observations
5. Sample oceanographic data
6. Create derived oceanographic datasets
7. Model species habitat or behavior
Invoking R from ArcGIS

Evaluate R Statements

Last statement result

Evaluate R Statements

R statements

print(t.test(1:10,y=c(7:20)))
1+1

R variables to define

OK  Cancel  Apply  Show Help >>
Invoking R from ArcGIS

R messages logged to ArcGIS

Value of last statement returned
MGET Project Status

- Version 0.2 just released
- “Framework” nearly complete, but only simple building-block tools are implemented
  - HDF to raster converter might interest you
- Oceanographic processing tools will be released in July
- Ben and Jason will integrate ArcRStats
Installing MGET

User executes installation program

Installs Python package

Registers COM objects

Configures Start menu

GeoEco-1.0.win32-py2.5.exe

Registers ArcGIS toolbox
ArcGIS documentation

In HTML files

ArcToolbox

Evaluate R Statements

Evaluate one or more R statements using the R interpreter and returns the result of the last statement.

Command line syntax:

`R.Evalute.GeLoOn <ctations;statements,...> (variablenames;variablenames...) (variablevalues;variablevalues...)`

Parameters:

- `Expression`: A list of one or more R statements formatted according to the specified syntax.
- `variablenames`: A list of names of variables to values. This list must have the same length as the number of variables in the expression. These two parameters are used to define a variable name summary.
- `variablevalues`: A list of values of variables. This list must have the same length as the number of variables in the expression. The values you provide:
  - Booleans are converted to the R "logical" type
  - Signed and unsigned integers of all sizes are converted to the R "integer" type
  - Floating point numbers of all sizes are converted to the R "double" type
  - Complex numbers are converted to the R "complex" type
  - Strings are converted to the R "character" type (but see the discussion below about additional parsing logic)
  - Python lists and tuples (arrays in other languages) are converted to the R "list" type
  - Python arrays, and arrays from the Python Numeric package are converted to R arrays, which are "simple vectors with the attribute dim and optionally dimnames attached to the vector" (hardcopy from the R Language Definition).
  - Python dictionaries are usually converted to the R "list" type, but may be converted to the "data.frame" type under some circumstances (I am not sure about this...)

If a value cannot be converted according to the logic above, an error is reported.

Variable values (optional)

A list of values of variables to define in the R interpreter before the R statements are evaluated.

This list must have the same number of entries as the `variablenames` parameter. That list specifies the names of the variables that will be defined and this list specifies their values.

The values you provide are automatically converted to the most appropriate R data types:

- Booleans are converted to the R "logical" type
- Signed and unsigned integers of all sizes are converted to the R "integer" type
- Floating point numbers of all sizes are converted to the R "double" type
- Complex numbers are converted to the R "complex" type
- Strings are converted to the R "character" type (but see the discussion below about additional parsing logic)
- Python lists and tuples (arrays in other languages) are converted to the R "list" type
- Python arrays, and arrays from the Python Numeric package are converted to R arrays, which are "simple vectors with the attribute dim and optionally dimnames attached to the vector" (quoted from the R Language Definition).
- Python dictionaries are usually converted to the R "list" type, but may be converted to the "data.frame" type under some circumstances (I am not sure about this...)

If a value cannot be converted according to the logic above, an error is reported.

In Arc toolbox
Python documentation

In HTML, formatted like Python library documentation
MGET includes extensive validation and logging; log levels are configurable.
Invitation to Collaborate

• Do you need a specific tool developed for your project?
  • We would consider developing it for MGET, especially if it would be widely applicable

• Do you develop tools yourself?
  • Become a contributor/coauthor! We could help you integrate your tools into MGET.
Thanks! Any questions?

Download site:
http://code.env.duke.edu/projects/mget

Contact: Jason Roberts
jason.roberts@duke.edu
Advanced Modeling: Ben’s Connectivity Modeler

Create Network

- poly_patches
- rstr_cost
- tin
- pt_nodes
- in_edges
- txt_network
- pt_centroids
- poly_patches

Network Least Cost Path

- In_edgeslc
- txt_networkl

Network Centrality Metrics

Best et al 2007 (In Review)
The Connectivity Problem

- Say you have a set of patches and a cost surface that describes migration cost
- How to efficiently compute how “connected” patches are to each other?
Step 1: create a network from the cost surface
Cost surface is converted to a TIN to create the network.
Step 2: calculate the least cost paths for the network

- Create Network
  - poly_patches
  - rstr_cost
- tin
- pt_nodes
- ln_edges
- poly_patches
- pt_centroids
- txt_network
- Network Least Cost Path
  - ln_edgeslc
  - txt_networkl
- poly_patchsm
- Network Centrality Metrics
Network least cost paths

- Djikstra algorithm highly efficient over ArcGIS CostPath function
- Future: create corridors with CostDistance from paths
Step 3: compute network centrality metrics as indices of connectivity
Network centrality metrics

Degree | Closeness | Betweenness

Developer walk-through:

How do you develop an MGET tool?
Goals for a common development framework for Duke’s tools

- Let developers select the best technologies for the job
  - Require tools to formally declare their dependencies
  - Encourage devs to choose set of standard technologies
- Automate tedious stuff, such as:
  - Tool dependency and input parameter validation
  - Interoperability plumbing (Arc toolboxes, COM objects)
  - Generation of installation packages
  - Generation of documentation
- Provide a library of common utility functions:
  - Invoking Arc/R/MATLAB, manipulating files and data,
Marine Geospatial Ecology Tools

- Core framework implemented in Python and C++
  - Python was selected due to wide appeal to Duke developers and perceived strategic importance to ESRI
- Core framework is platform independent
  - Individual tools determine their own dependencies
- Tools may be implemented in any language
  - But language interoperability is expensive to develop
  - Currently planning for Python, R and MATLAB tools
- Provides all features from previous slide, and more!
Creating a “Hello, World” tool in Python

You implement a Python-based “tool” in MGET by creating a Python instance method or classmethod:

1. Create the module and class that will receive the new method (or locate an existing module and class)
2. Define the method’s name and input parameters
3. Fill in the method’s body (i.e., write the code)
4. Specify some metadata about the method
5. Run a script that rebuilds the MGET installation package
1. Create the module and class

```python
from GeoEco.DynamicDocString import DynamicDocString
from GeoEco.Internationalization import _
from GeoEco.Logging import Logger

class Example(object):
    __doc__ = DynamicDocString()

# Class definition
# Hack to allow metadata to be added to class

# Import needed modules provided by the core MGET framework

GeoEco is the name of the MGET Python package; I chose this name after Ben expressed a desire that the package name not imply that the tools are not only for marine problems.
```
from GeoEco.DynamicDocString import DynamicDocString
from GeoEco.Internationalization import _
from GeoEco.Logging import Logger

class Example(object):
    __doc__ = DynamicDocString()

    @classmethod
    def GreetPerson(cls, personName):
        pass

        # Method definition
        Method body, to be filled in as next step
        # Input parameter
        Method definition
from GeoEco.DynamicDocString import DynamicDocString
from GeoEco.Internationalization import _
from GeoEco.Logging import Logger

class Example(object):
    __doc__ = DynamicDocString()

    @classmethod
def GreetPerson(cls, personName):
        cls.__doc__.obj.ValidateMethodInvocation()
        message = _("Hello, %s!") % personName
        Logger.Info(message)
        return message

Call to function that validates the tool’s dependencies and input parameter values
Call to logging system
Output parameter

The _ function and use of Unicode strings enable localization of MGET tools
from GeoEco_Metadata import *
from GeoEco_Types import *

AddMethodMetadata(Example.GreetPerson,
    shortDescription=_(u’Greets a specified person’),
    isExposedToPythonCallers=True,
    isExposedByCOM=True,
    isExposedAsArcGISTool=True,
    arcGISDisplayName=_(u’Greet Person’),
    arcGISToolCategory=_(u’Example Tools’))

AddArgumentMetadata(Example.GreetPerson, u’personName’,
    typeMetadata=UnicodeStringTypeMetadata(canBeNone=False),
    description=_(u’The person to greet with a friendly message.’),
    arcGISDisplayName=_(u’Person to greet’))

Some of the many things you can do with the metadata

Configure interoperability

Specify appearance of ArcGIS toolbox

Specify strong parameter type and validation options

Write documentation in reStructuredText
5. Execute the build script
The build script

Using the tool metadata as input, the script generates:

- ArcGIS toolbox (*Marine Geospatial Ecology Tools.tbx*)
- Python wrapper scripts for invoking tools from toolbox
- Microsoft COM type library and registration scripts, so tools can be invoked as COM objects
- Python reference documentation (HTML)
- ArcGIS geoprocessing documentation (HTML)
- COM documentation (HTML, not implemented yet)
- Installation package (*GeoEco-1.0.win32-py25.exe*)
Invoking your tool from Python

Python callers can import your module and invoke your method directly:

```python
>>> from GeoEco.Logging import Logger
>>> Logger.Initialize()
2007-04-25 20:00:16,213 INFO Logging system initialized from config file "C:\Documents and
>>> from GeoEco.Example import Example
>>> greeting = Example.GreetPerson(u'Jason')
2007-04-25 20:01:02,819 INFO Hello, Jason!
>>> print greeting
Hello, Jason!
```
Invoking your tool from ArcGIS

Documentation appears in geoprocessing UI

Fancy formatting is supported (e.g. bullets, hyperlinks, indentation, code, images)
Invoking your tool through COM Automation

VBScript example (many other languages supported)

```vbscript
Set logger = WScript.CreateObject("GeoEco.Logger")
logger.Initialize
Set example = WScript.CreateObject("GeoEco.Example")
greeting = example.GreetPerson("Joe")
WScript.StdOut.WriteLine(greeting)
```

![CMD Shell output](image)
Invoking your tool from .Net

C# Example

Invocation occurs through early-bound ("vtable") COM, not COM Automation
Logging with Debug messages disabled

Core framework only reports one message (the other is the “Hello, Joe!” greeting).
Logging with Debug messages enabled
Example: Declaring dependencies and calling ArcGIS geoprocessor functions

```python
@classmethod
def CreateZeroRaster(cls, outputRaster):
    cls.__doc__.Obj.ValidateMethodInvocation()
    gp = GeoprocessorManager.GetWrappedGeoprocessor()
    gp.CreateConstantRaster_sa(outputRaster, 0.0, 'FLOAT', 1.0, '0.0 0.0 1.0 1.0')
```

Dependencies checked and geoprocessor initialized here

Wrapped geoprocessor object logs all calls

ArcGIS dependencies:

- Dependencies checked and initialized
- Wrapped geoprocessor object logs all calls

```python
AddMethodMetadata(Example.CreateZeroRaster,
    shortDescription=_(u'Creates a raster with all cells set to zero.'),
    isExposedToPythonCallers=True,
    isExposedByCOM=True,
    isExposedAsArcGISTool=True,
    arcGISDisplayName=_(u'Create Zero Raster'),
    arcGISToolCategory=_(u'Examples'),
    dependencies=[ArcGISDependency(9, 1),
                  ArcGISExtensionDependency(u'spatial')])
```
Resulting output (debug logging enabled)

Initialize geoprocessor
Other dependency checks
Logged ArcGIS interactions
Dependencies implemented so far

- WindowsDependency – minimum Windows version
- PythonDependency – minimum Python version
- PythonModuleDependency – module is installed
- ArcGISDependency – minimum ArcGIS version
- ArcGISProductDependency – minimum product level
- ArcGISExtensionDependency – extension is available
- RDependency – minimum R version
- RPackageDependency – package is installed, minimum version optional
Example of dependency failure

The Scenario:
The user invokes the Evaluate R Statements tool but the required rpy Python module is not installed.

rpy is not installed and the framework raises a SoftwareNotInstalled error.
ArcGIS documentation

HTML pages generated from XML metadata using XSL transforms

Documentation also added to Arc toolbox by build script
Python documentation

HTML pages generated from XML metadata using XSL transforms
Using metadata to generate batch-processing versions of your method

```python
from GeoEco.BatchProcessing import BatchProcessing
from GeoEco.DataManagement.Fields import Field

BatchProcessing.GenerateForMethod(HDF.ExtractHeader,
    inputParamNames=[u'inputFile'],
    inputParamFieldArcGISDisplayNames=[u'Input HDF file field'],
    inputParamDescriptions=[u'%s paths of the input HDF files.'],
    outputParamNames=[u'outputFile'],
    outputParamFieldArcGISDisplayNames=[u'Output text file field'],
    outputParamExpressionArcGISDisplayNames=[u'Output file Python expression'],
    outputParamDescriptions=[u'%s paths of the text files to write.'],
    ...
    processListMethodName=u'ExtractHeaderList',
    processTableMethodName=u'ExtractHeaderTable',
)
```

Batch processing methods to generate