Marine Geospatial Ecology Tools

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.x, ArcGIS

Talk Outline

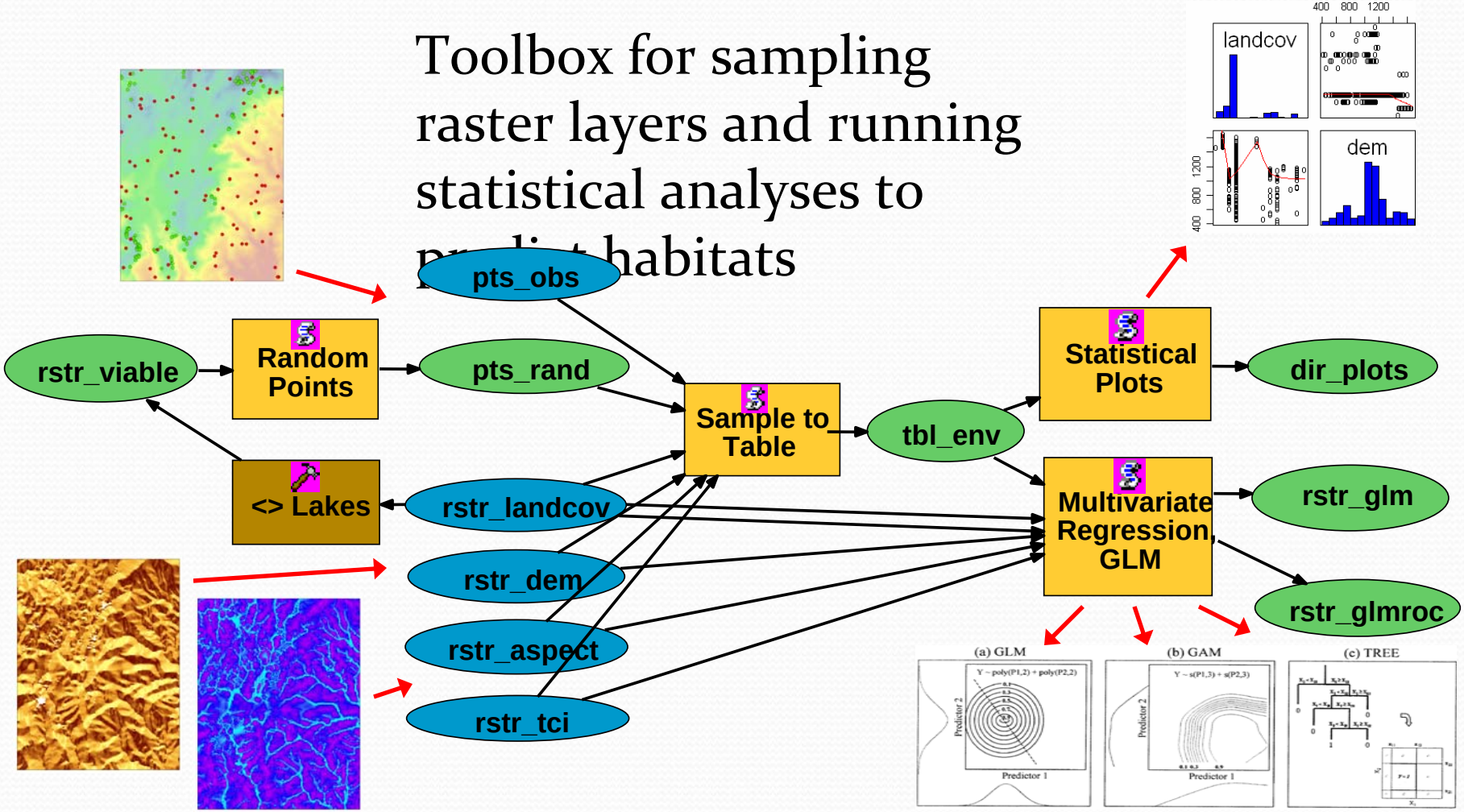
- History of MGET
- Walkthrough typical user scenario
  - Highlight interesting tools and features
- Invitation to collaborate
- Advanced topics: (time permitting)
  - Ben's Connectivity Modeler
  - How to build an MGET tool

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!

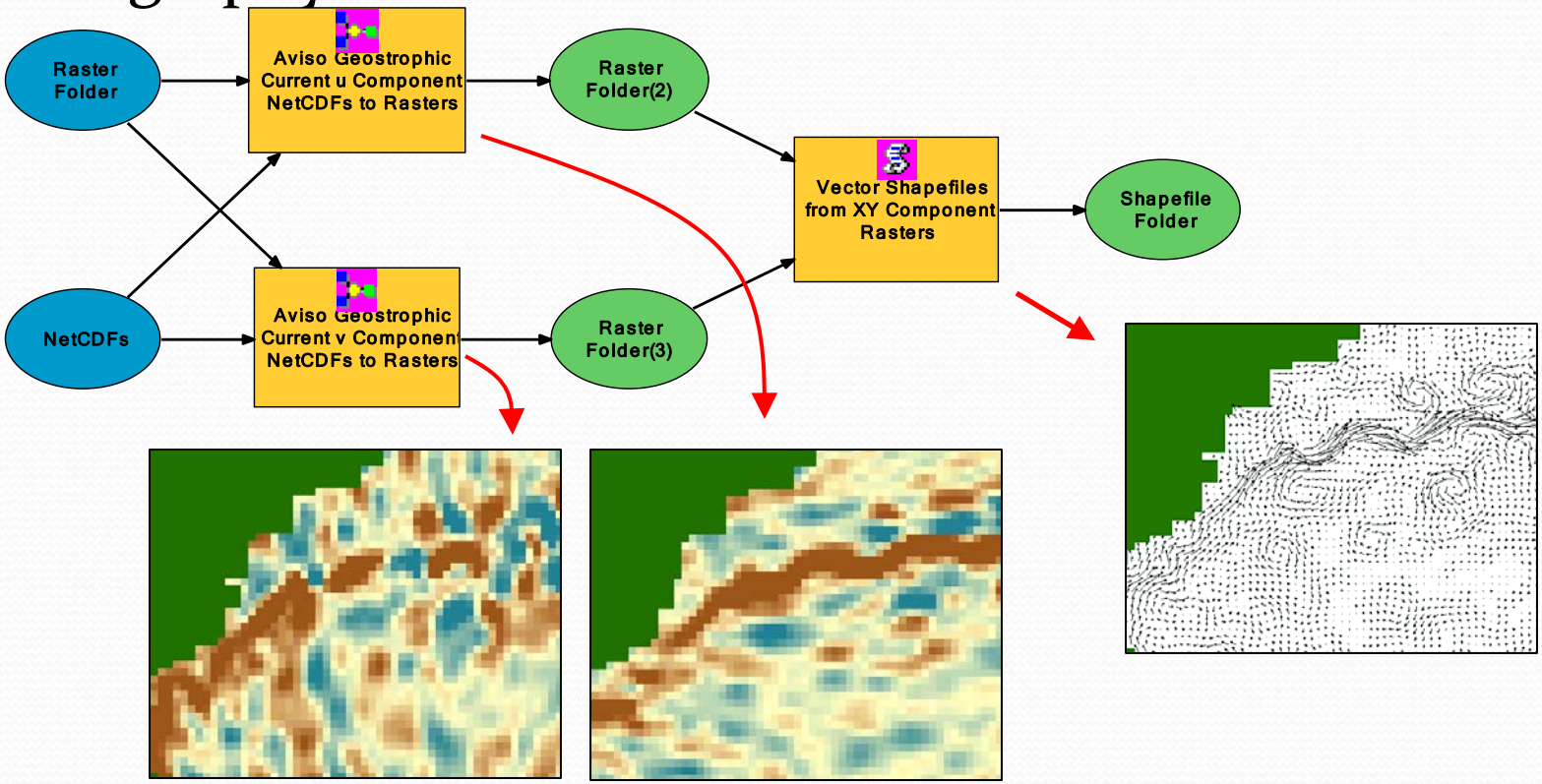
ArcRStats by Ben Best

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



Marine Ecology Tools by Jason Roberts

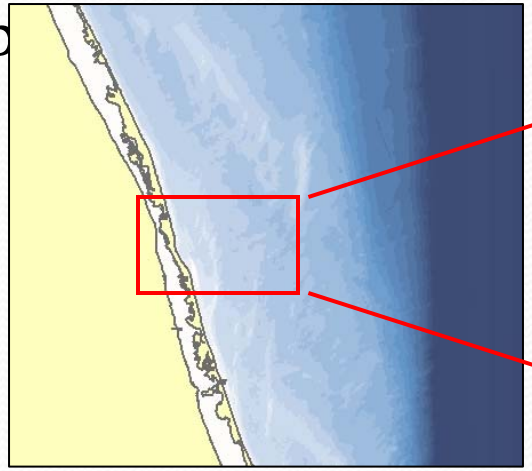
Unreleased toolbox for batch processing of oceanography



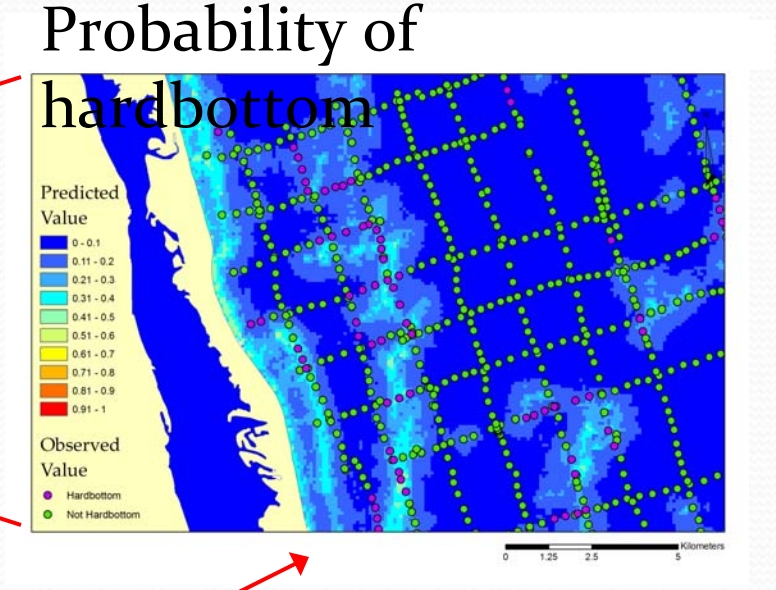
Benthic Complexity Modeler by Daniel Dunn

Predicts hardbottom from coarse-grain (90 m) bathymetry

GEODAS bathymetry



Probability of hardbottom



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

Statistical model: GLM with logit link

Unifying Our Tools

ArcRStats, HabMod, ConnMod

Marine Ecology Tools

Benthic Complexity Modeler

E. Treml's Connectivity Tools

And others

MGET

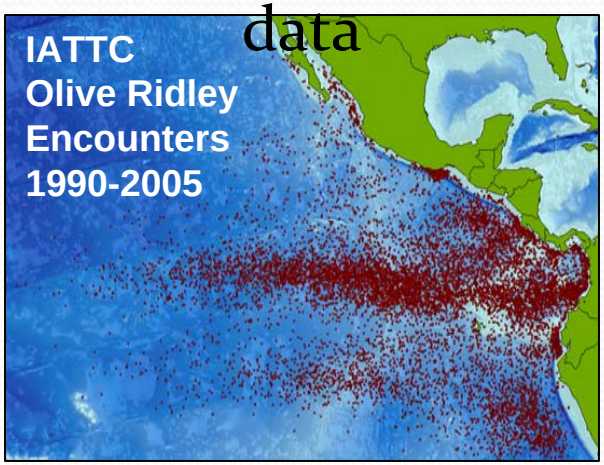
python powered

Typical User Scenario

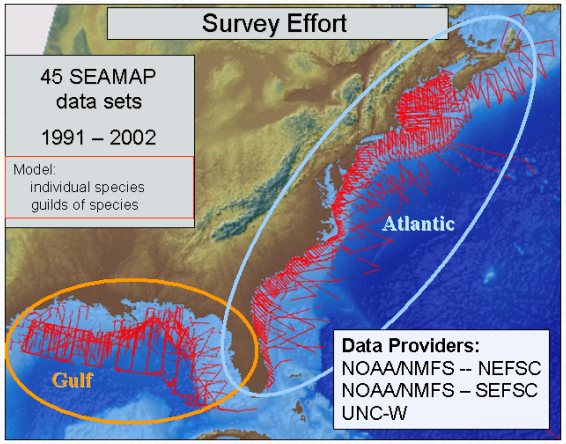
- Researcher has spatially-explicit observations of a species and wants to investigate why it is there
  - Are there spatial and temporal patterns?
  - Correlations with environmental conditions?
  - Correlations with occurrence of other species?
  - Can we predict its occurrence and thereby improve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

Typical Observation Data

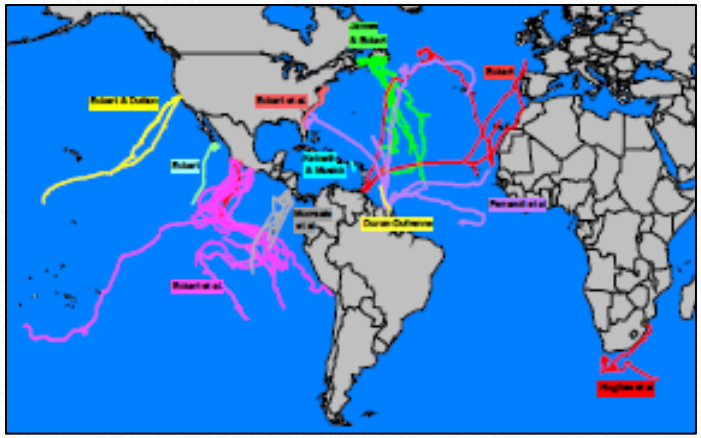
Fishery bycatch data



Surveys



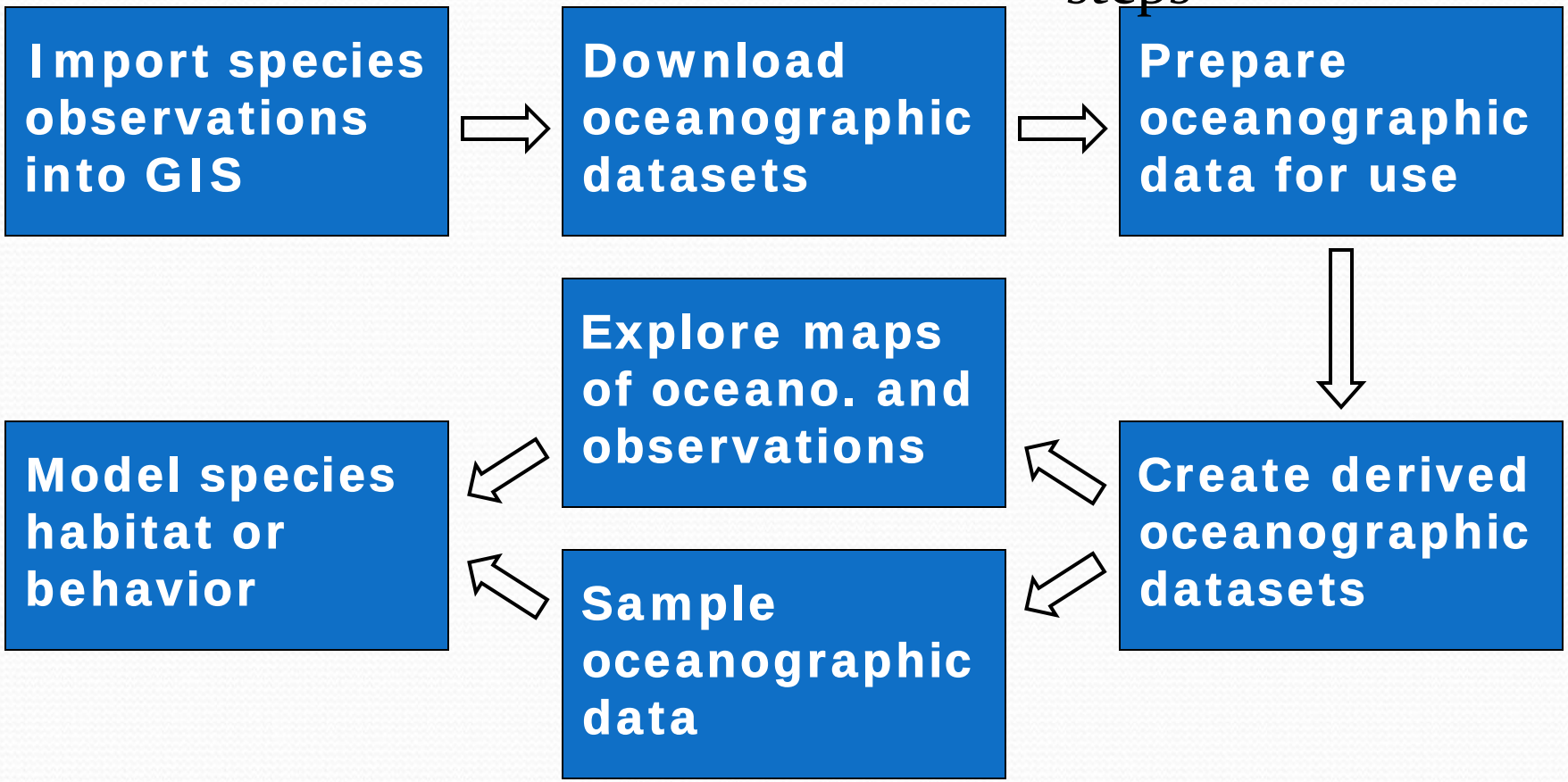
Argos satellite tracks



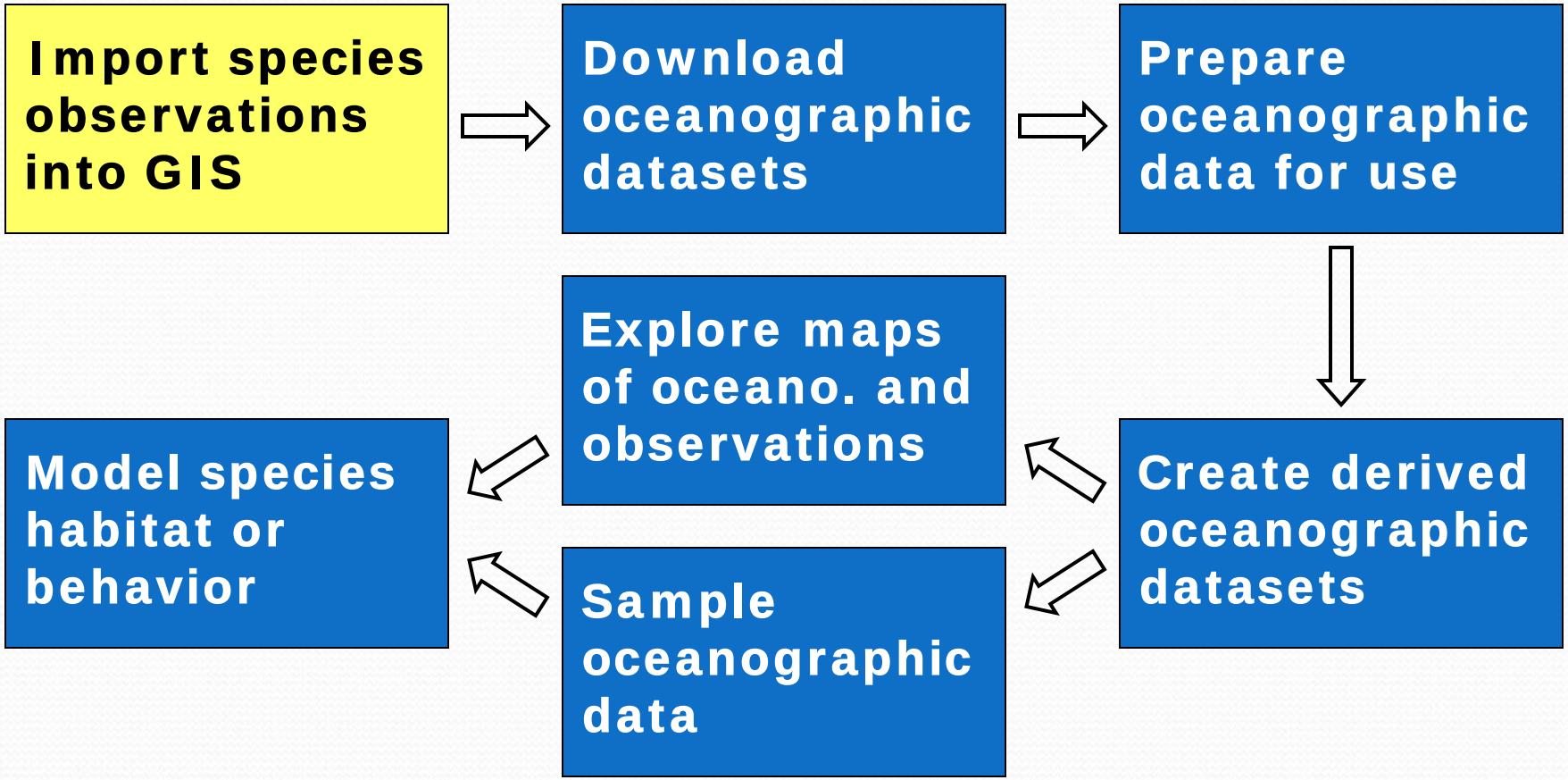
Scott Eckert

Typical Workflow

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



Typical Workflow



Import Argos Results Into Tables

DIAG record

```
37120  Date : 12.12.03 09:58:10  LC : 2  IQ : 50
     Lat1 : 34.282N  Lon1 :  77.010W  Lat2 : 32.640N  Lon2 :  69.121W
     Nb mes : 012  Nb mes>-120dB : 000  Best level : -121 dB
     Pass duration : 706s   NOPC : 2
     Calcul freq : 401 647695.6 Hz   Altitude :    0 m
             49           54           00           00
             00           00          218           17
             96           96           29          159
            255           94           98
```

DS record

```
11373 37120   9 31 J
     2003-12-03 09:49:11  1         00           00           00           00
                                    00          106           40          116
                                    02          230           97          145
                                   211          214           72           44
                                   158          139          118          119
                                   193           80           00           00
                                    00           00           00           00
                                    00           00           00
```

Import Argos Results Into Tables

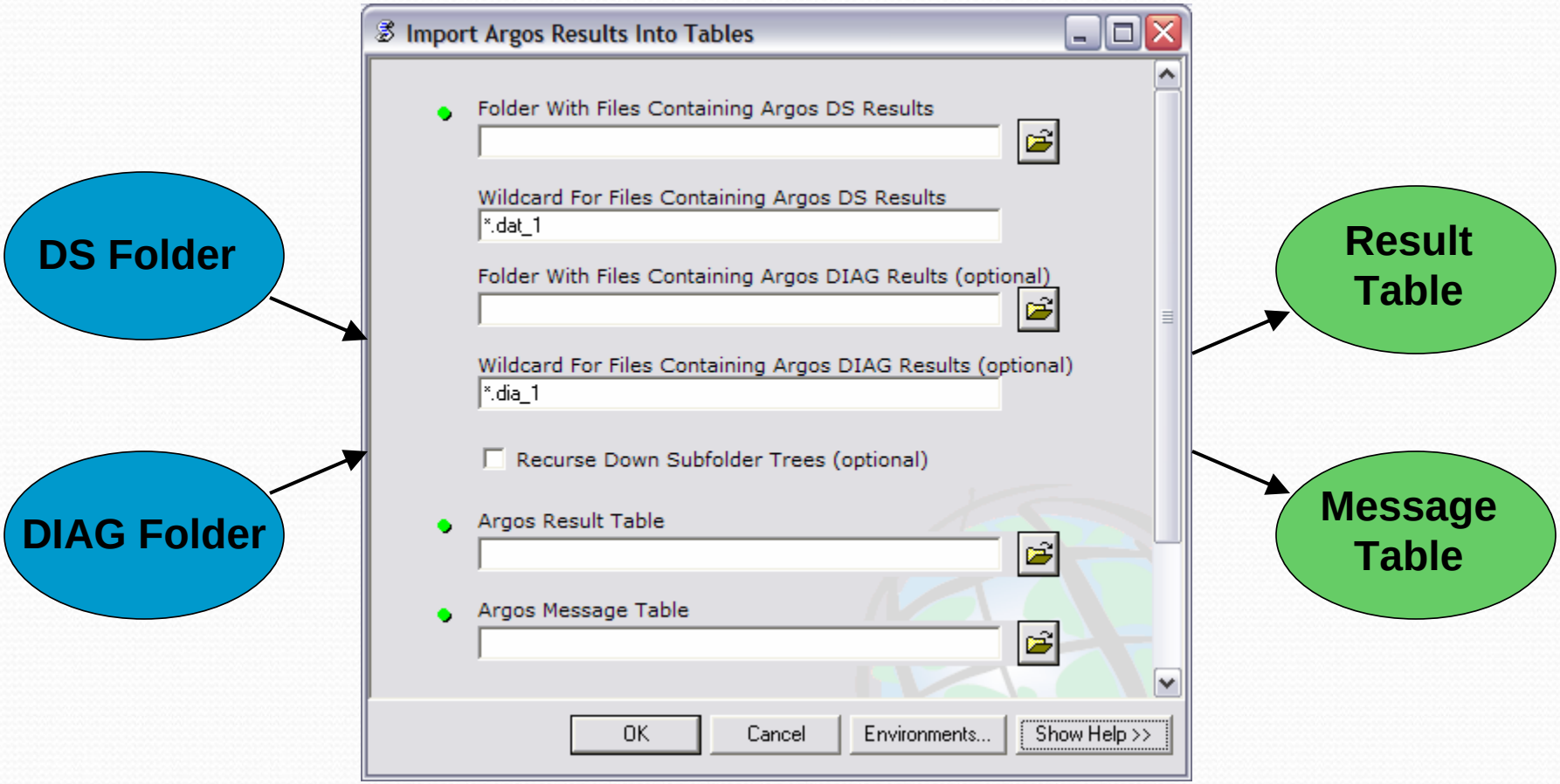
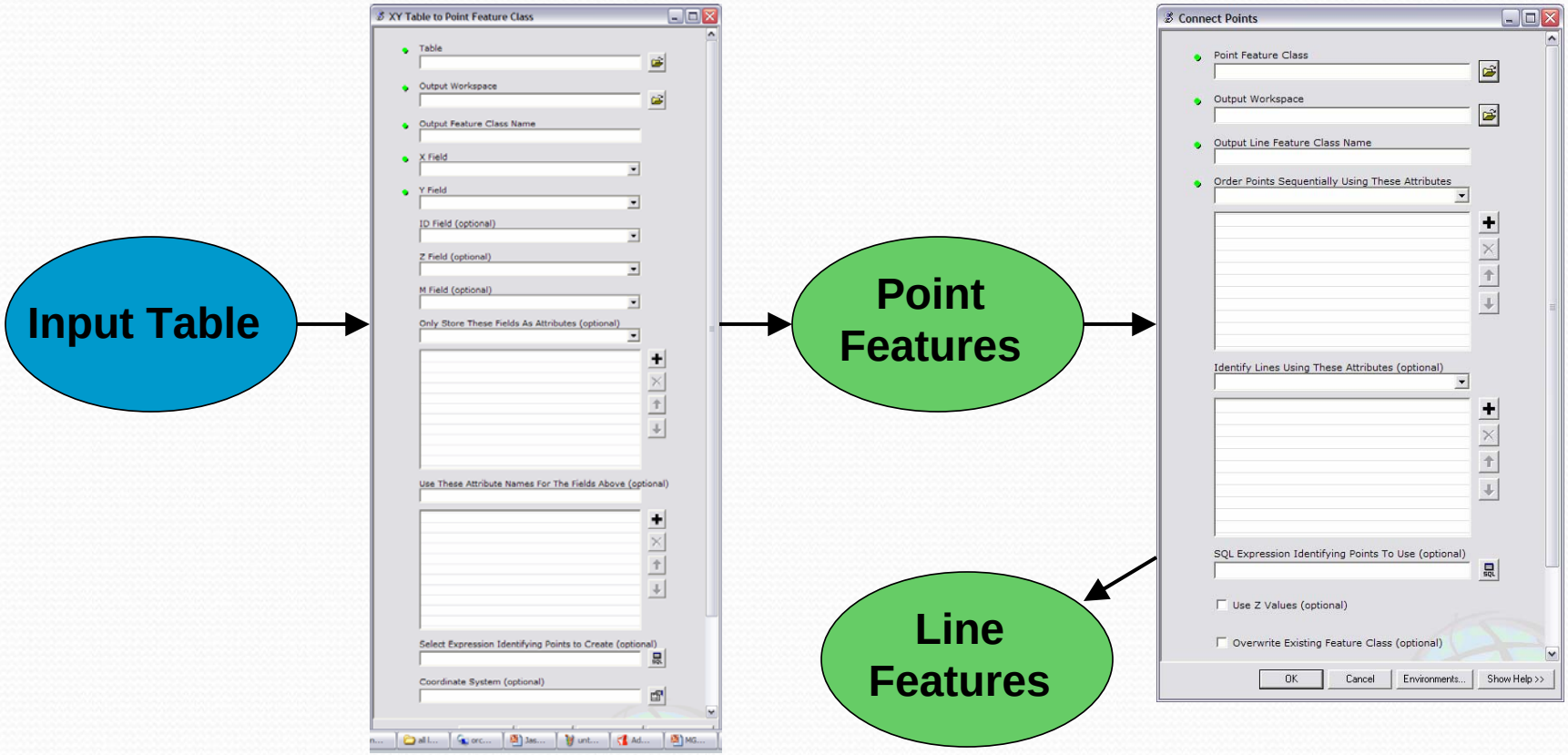


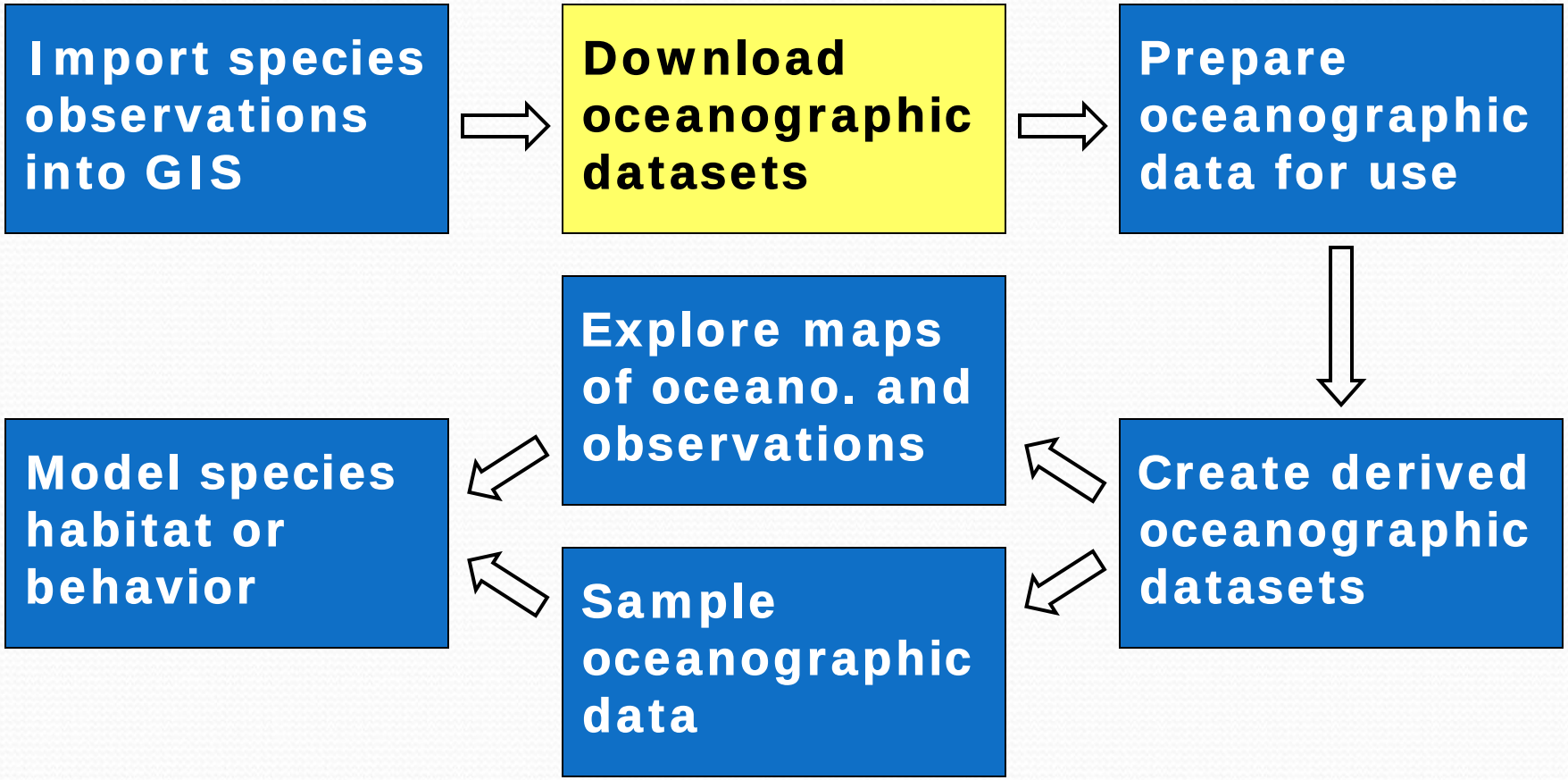
Table → Points → Lines

XY Table to Point Feature Class

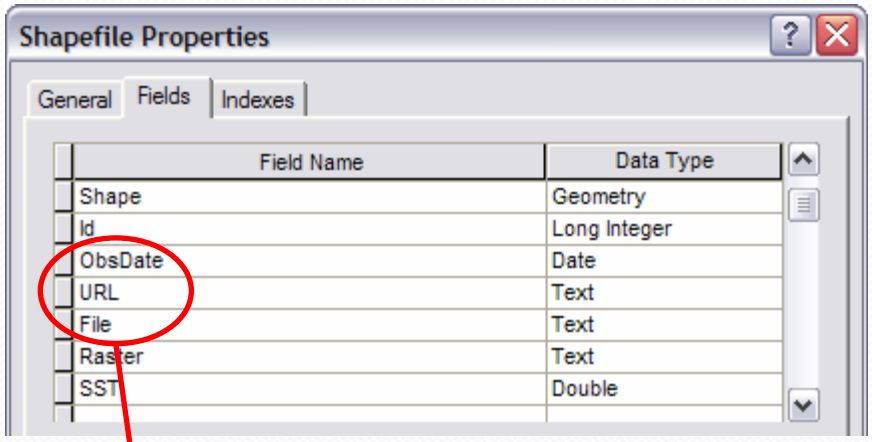
Connect Points

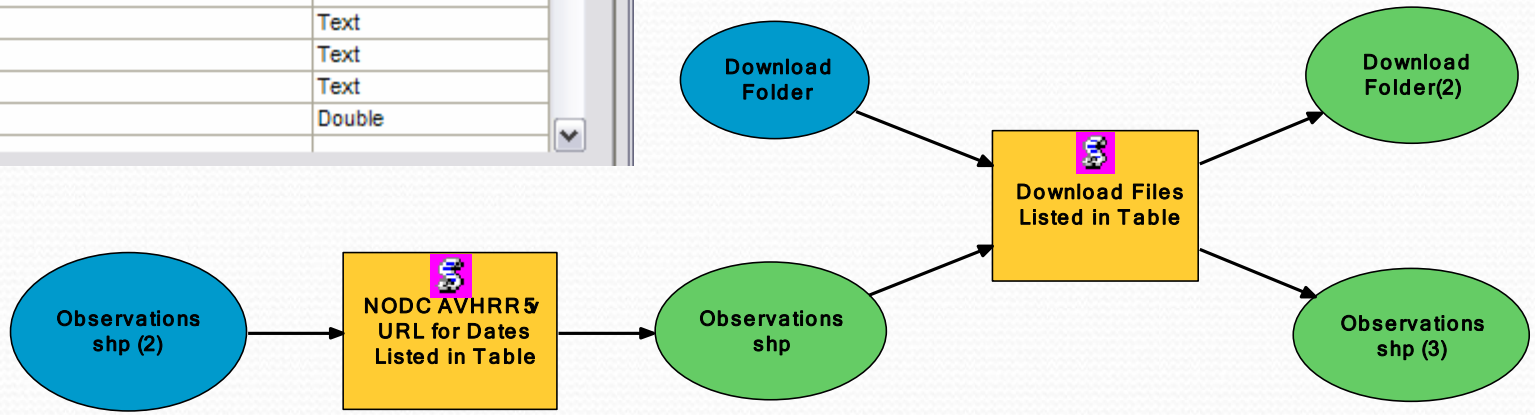


Typical Workflow



Calculate URLs and Download

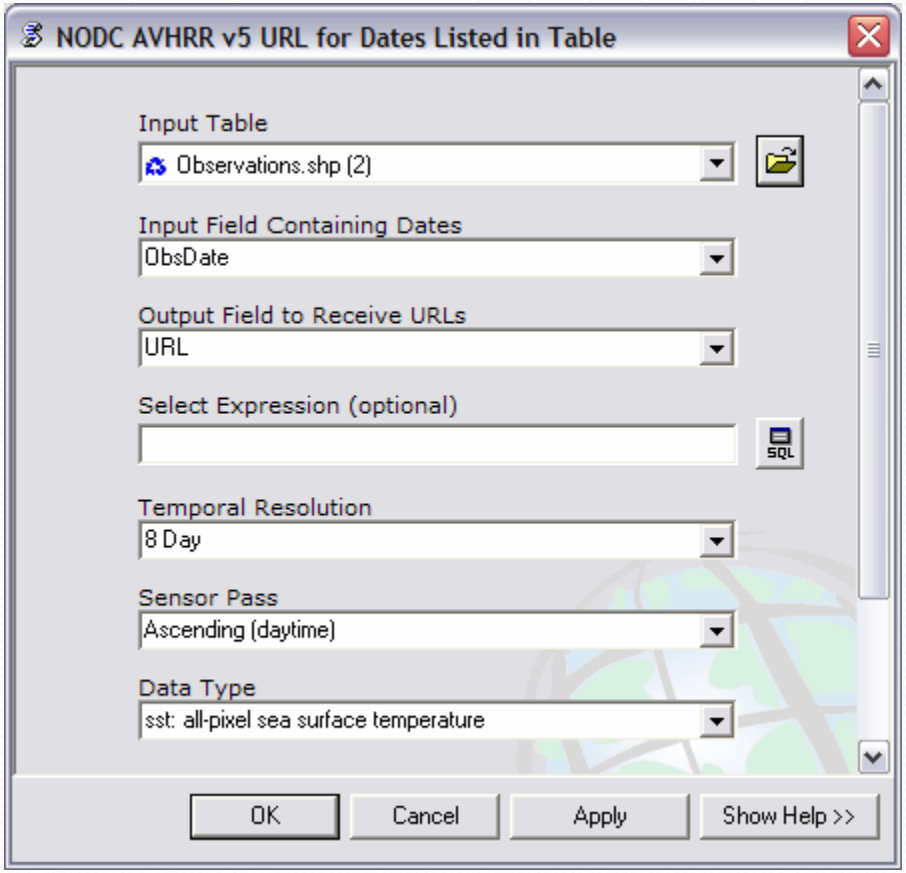




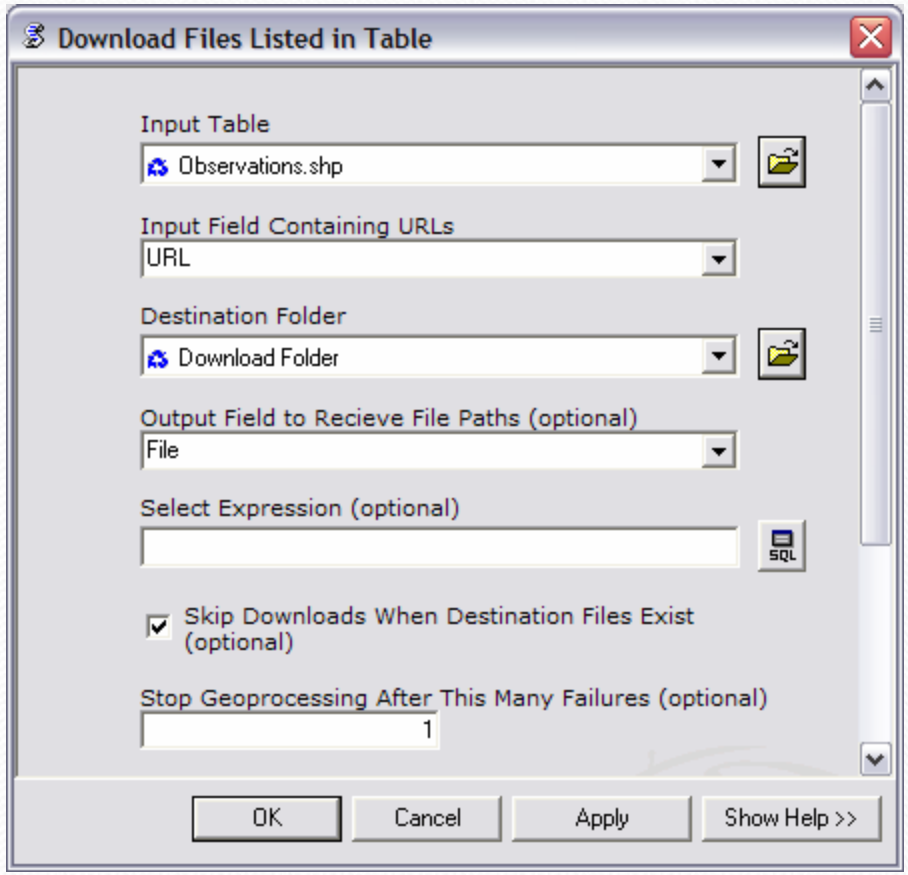
1. Given points' dates, calculates URL to NOAA SST HDFs
2. Given URLs, download the files to a given directory

Calculate URLs and Download

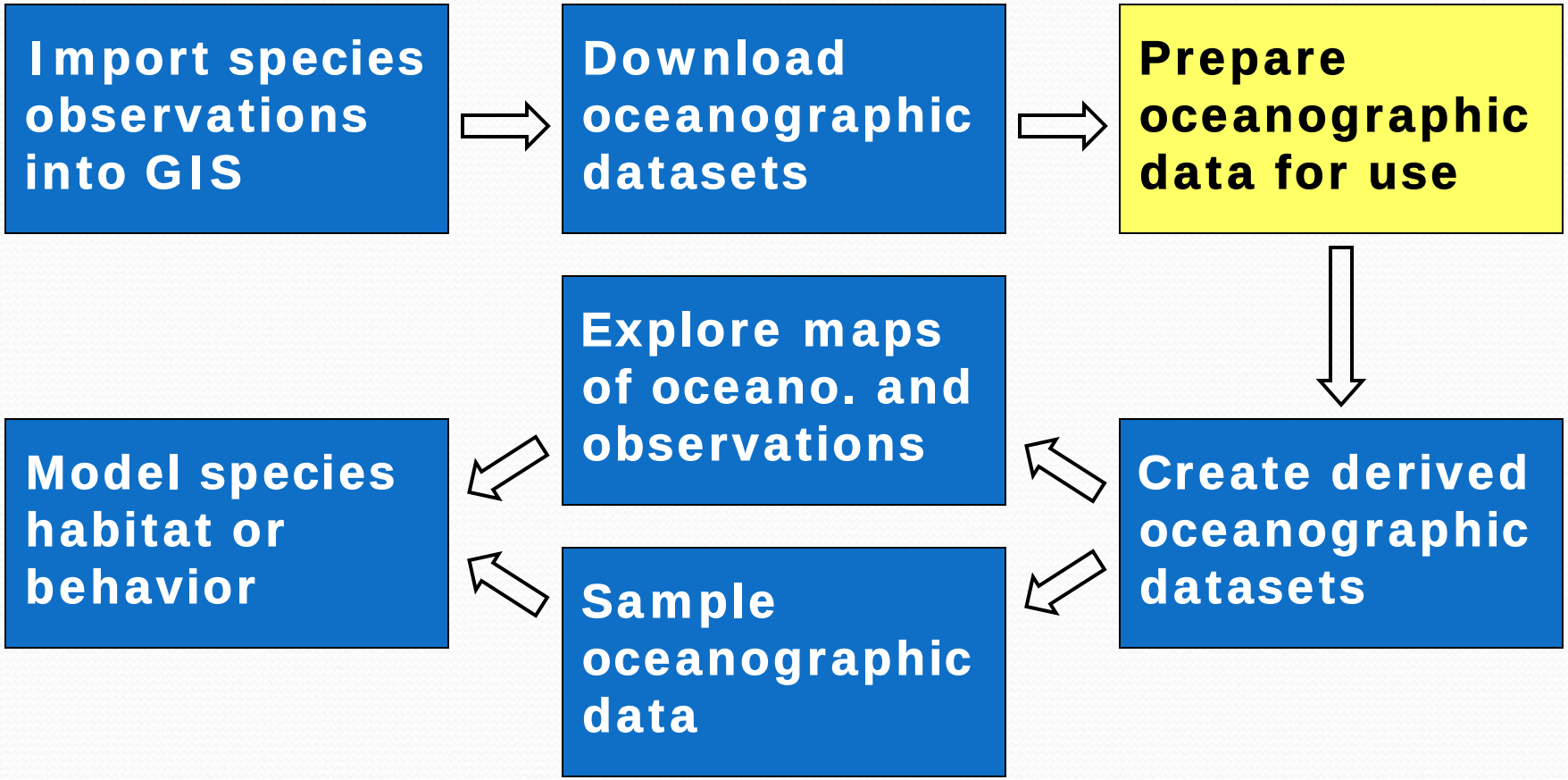
NODC AVHRR v5 URLs for Dates



Download Files



Typical Workflow



Preparing Oceanography For Use

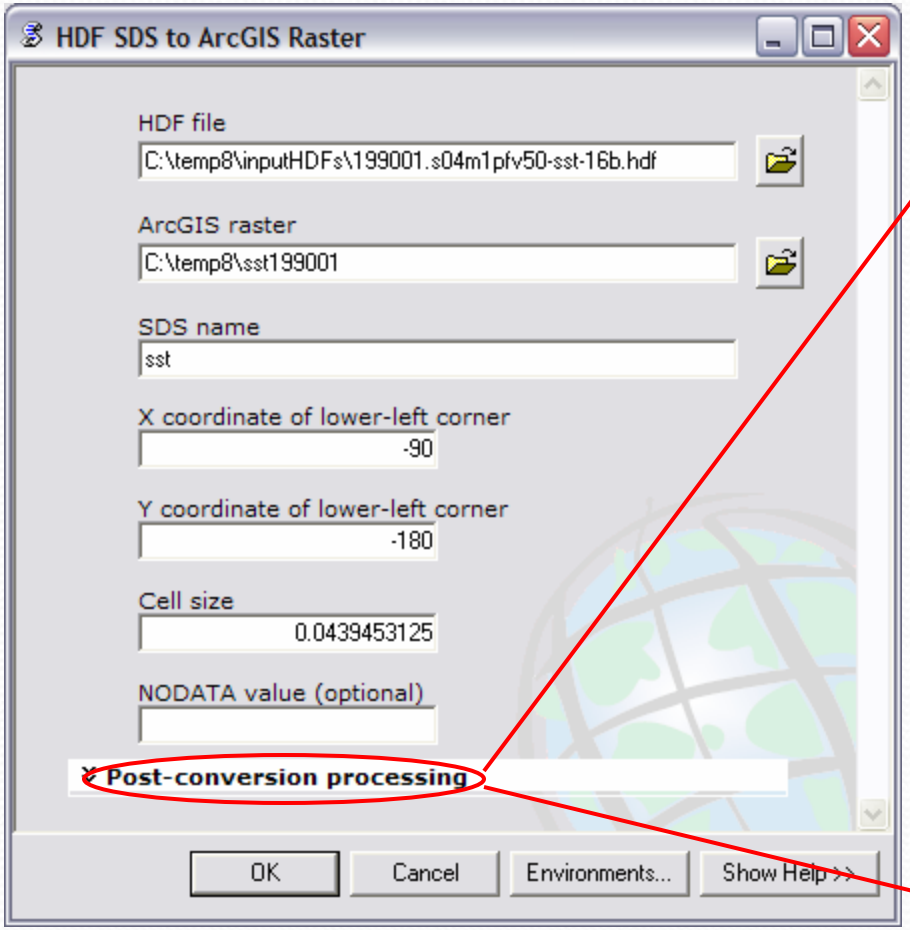
- Most oceanographic datasets are not immediately usable
- Common preprocessing steps include:
  - Converting to a supported format
  - Projecting to a desired projection
  - Clipping to region of interest
  - Performing arbitrary map algebra
  - Building pyramids

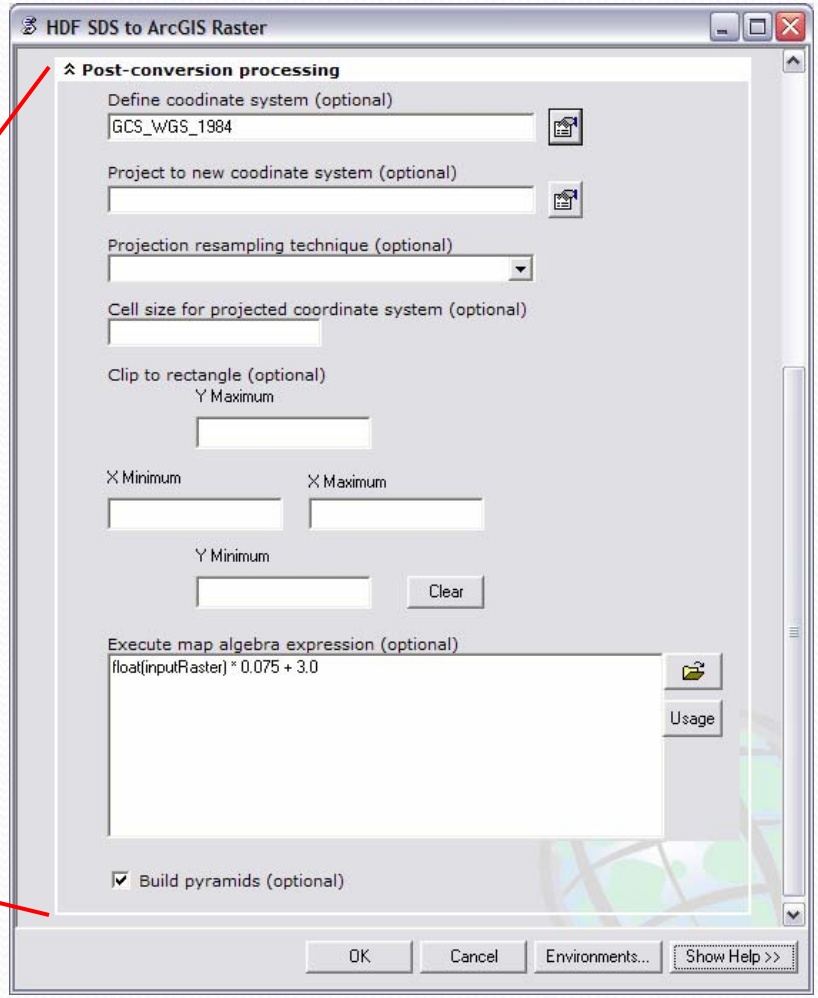
MGET Tools for Oceanography

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



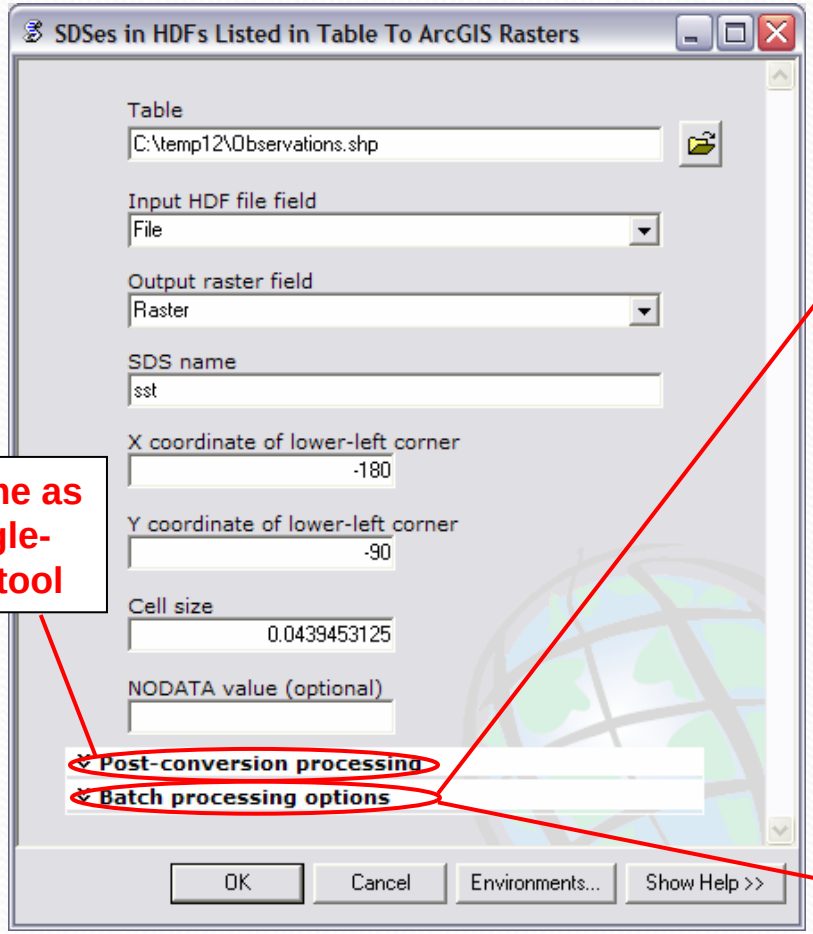


Batch Processing Design Pattern 1

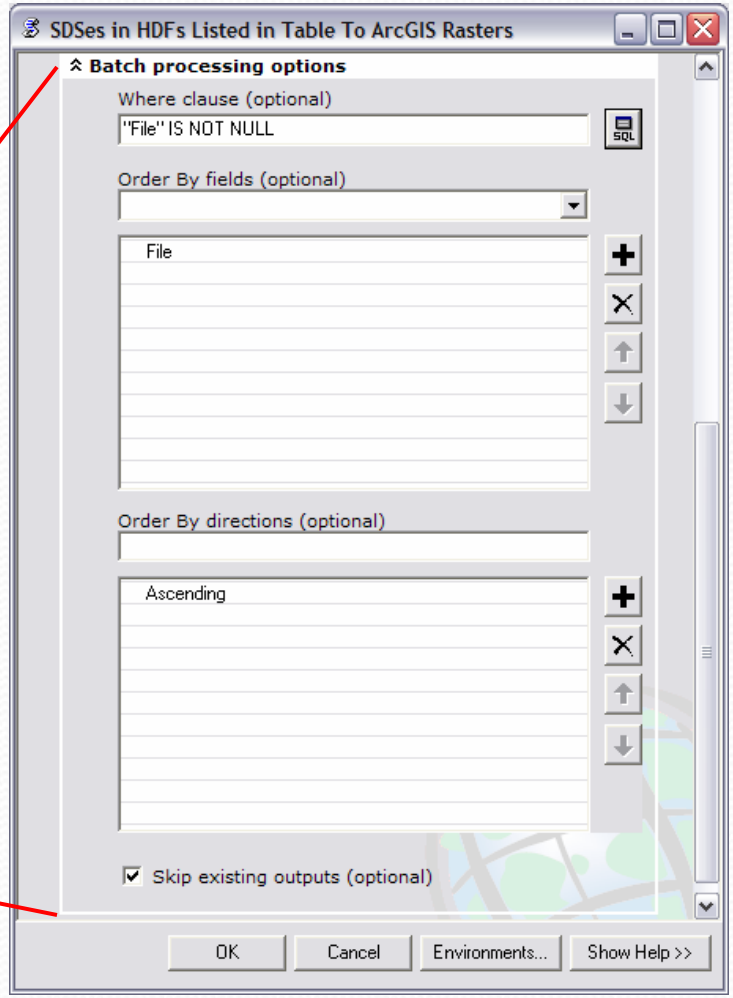
"Process inputs listed in table" pattern:

- 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

Example: SDSes in HDFs Listed in Table to ArcGIS Rasters



Same as single-file tool

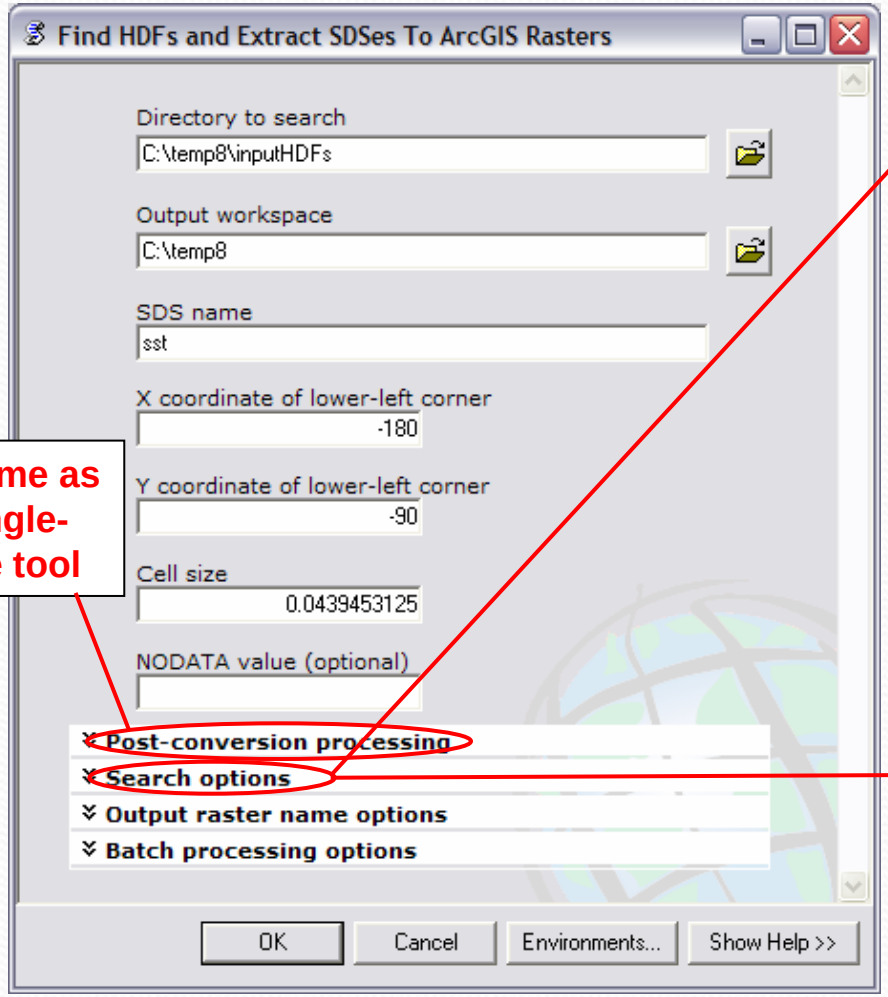


Batch Processing Design Pattern 2

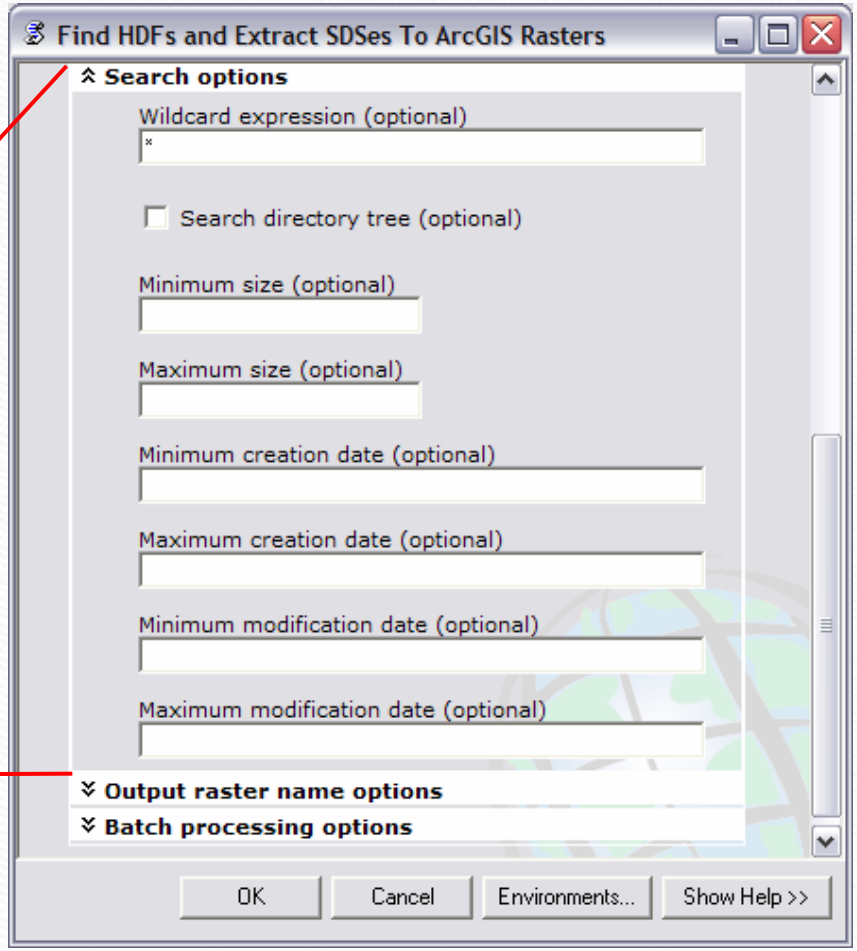
"Find and process inputs" pattern:

- 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

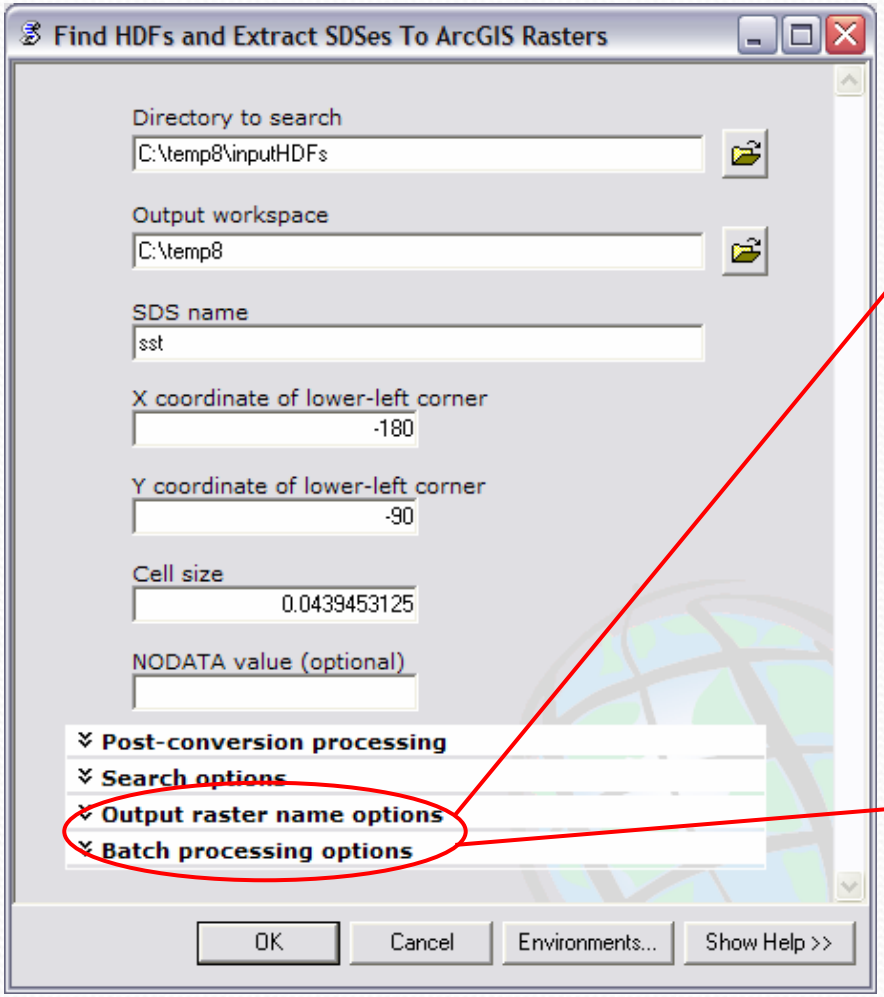
Example: SDSes in HDFs Listed in Table to ArcGIS Rasters

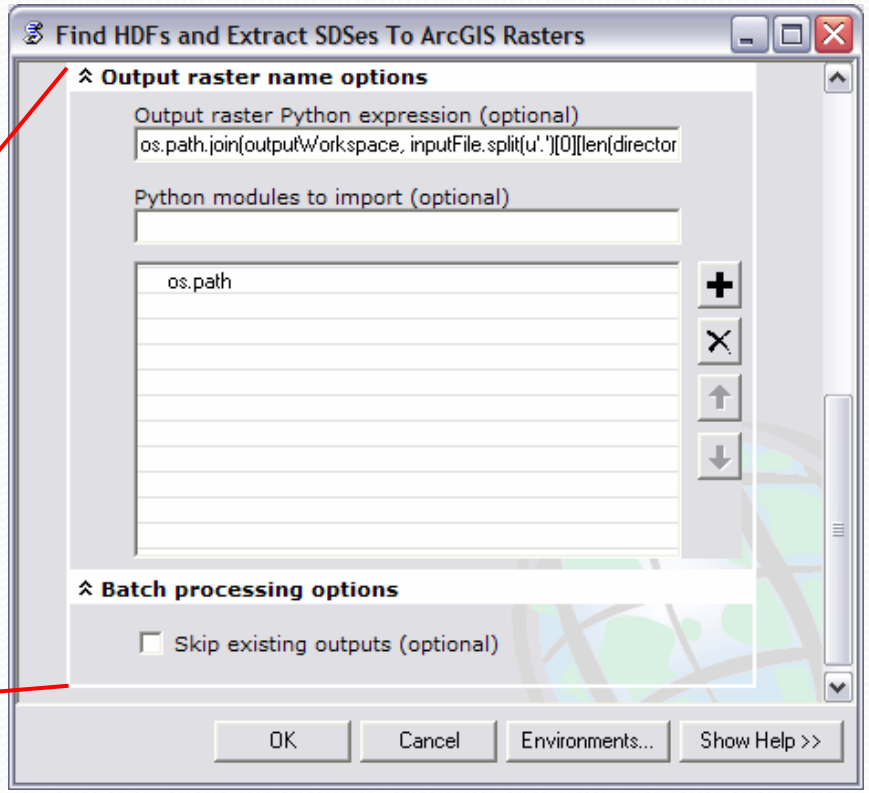


Same as single-file tool



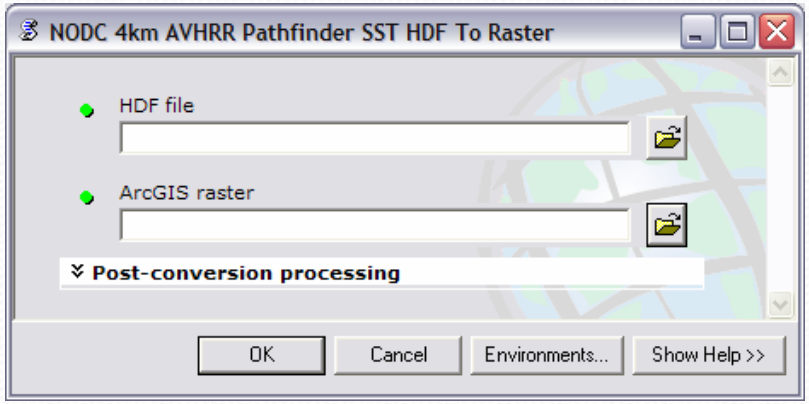
Example: SDSes in HDFs Listed in Table to ArcGIS Rasters

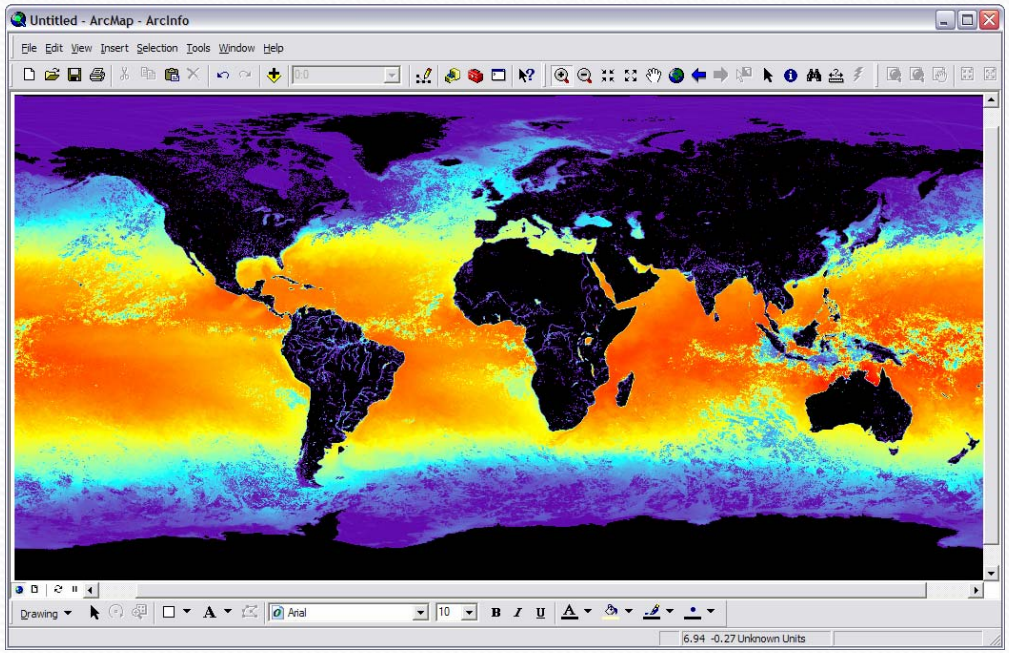




Example Product-Specific Tool

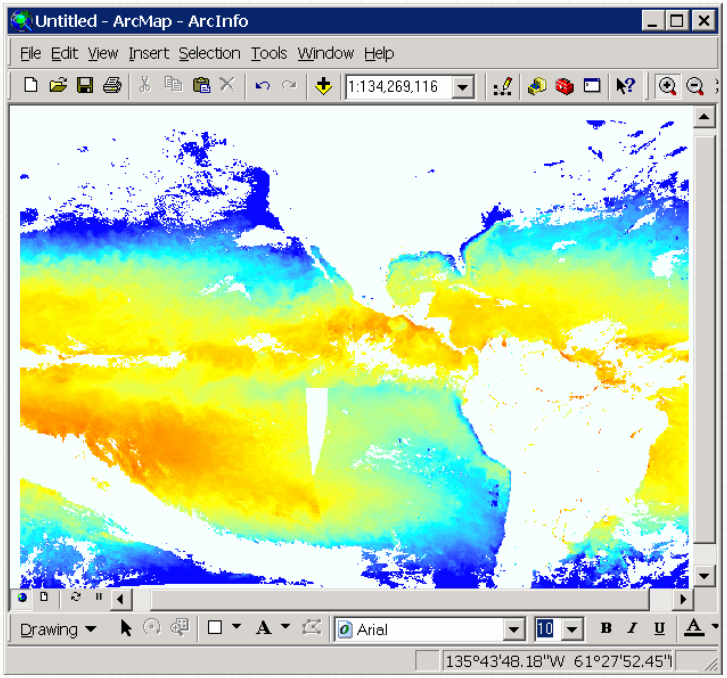
NOAA NODC 4km AVHRR Pathfinder v5 SST



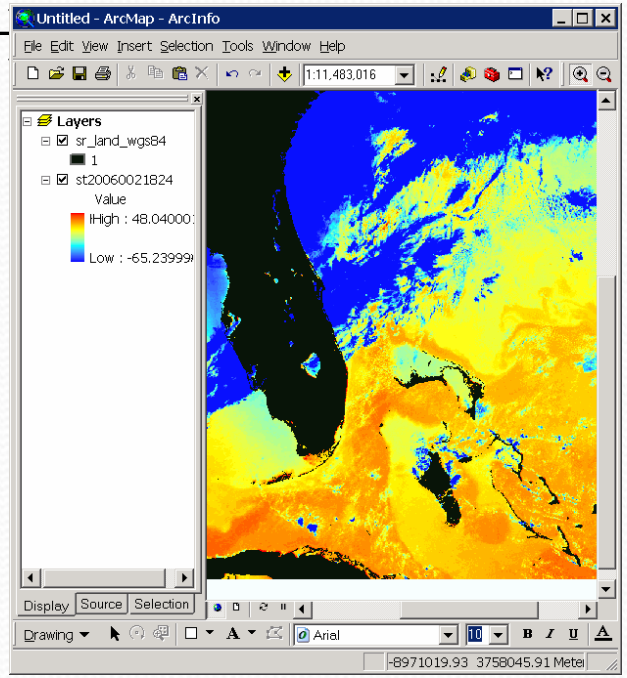


Other SST Products

PO.DAAC GOES 10/12



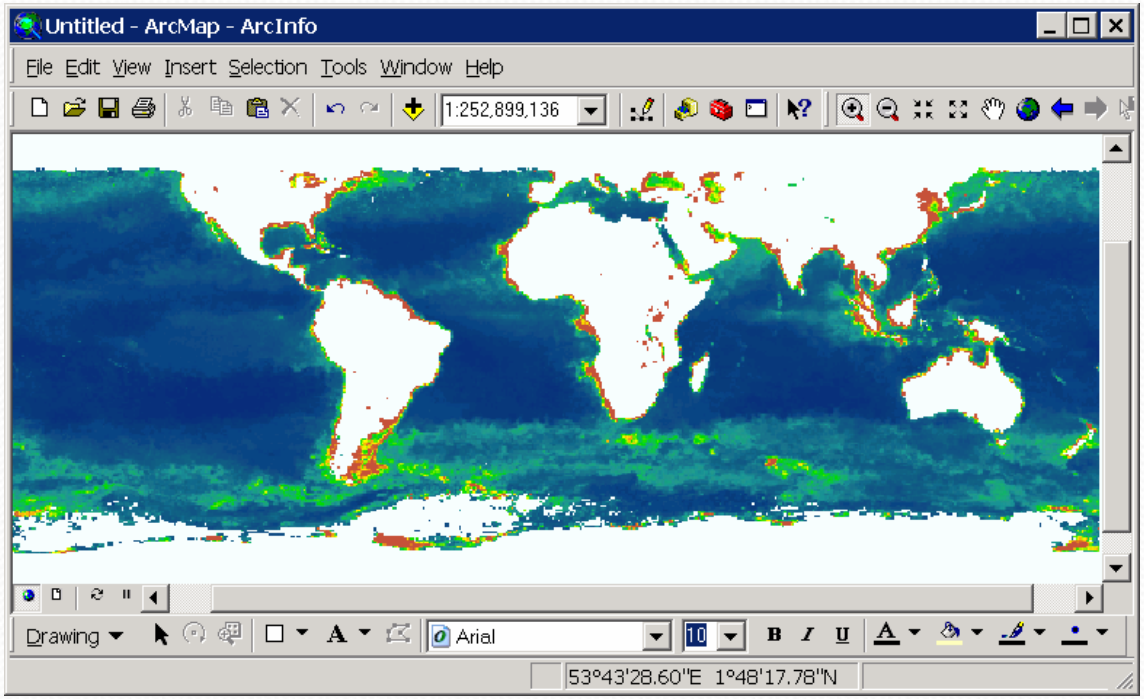
NOAA CoastWatch AVH



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

Sea Surface Chlorophyll

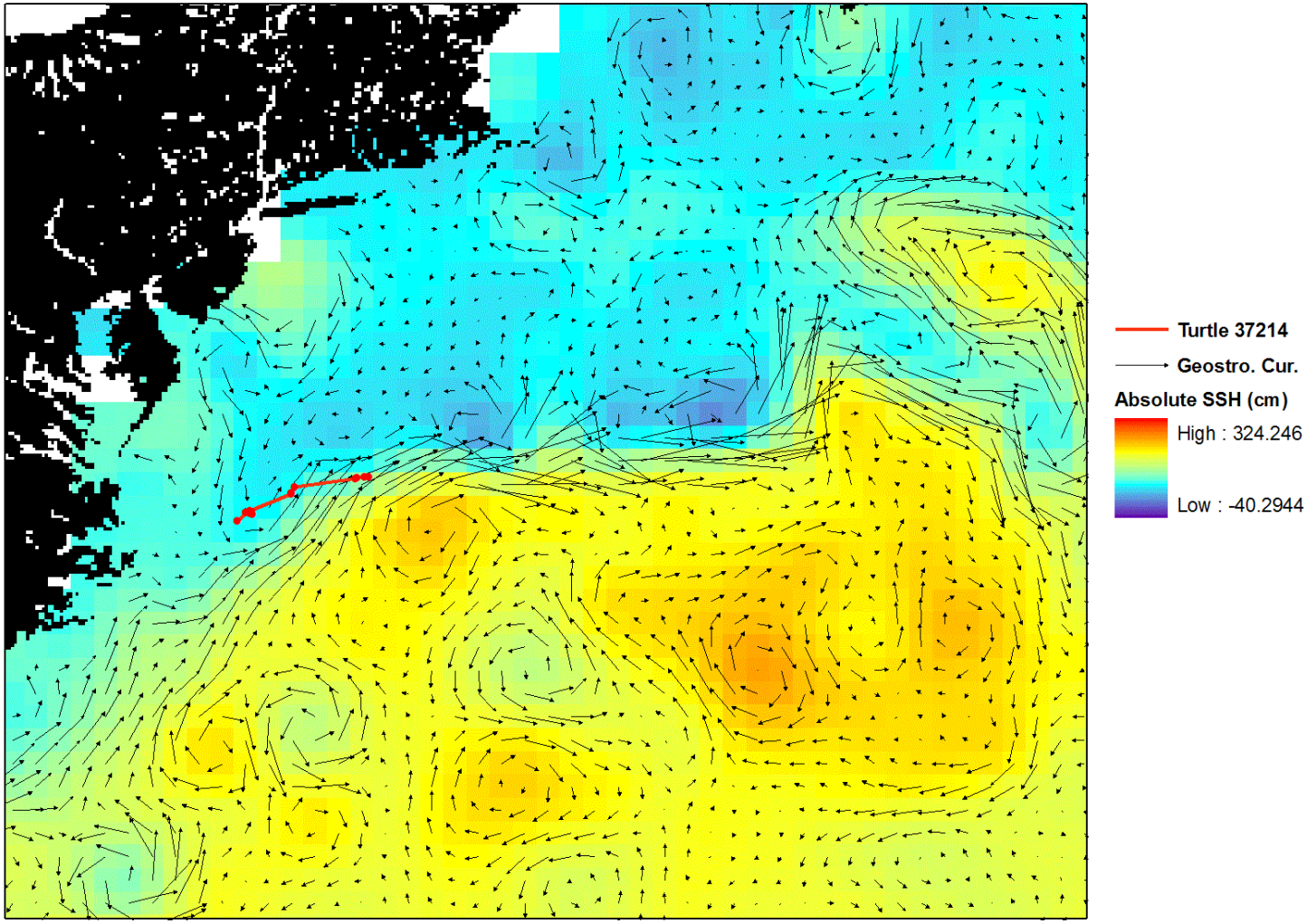
NASA OceanColor Group SeaWiFS



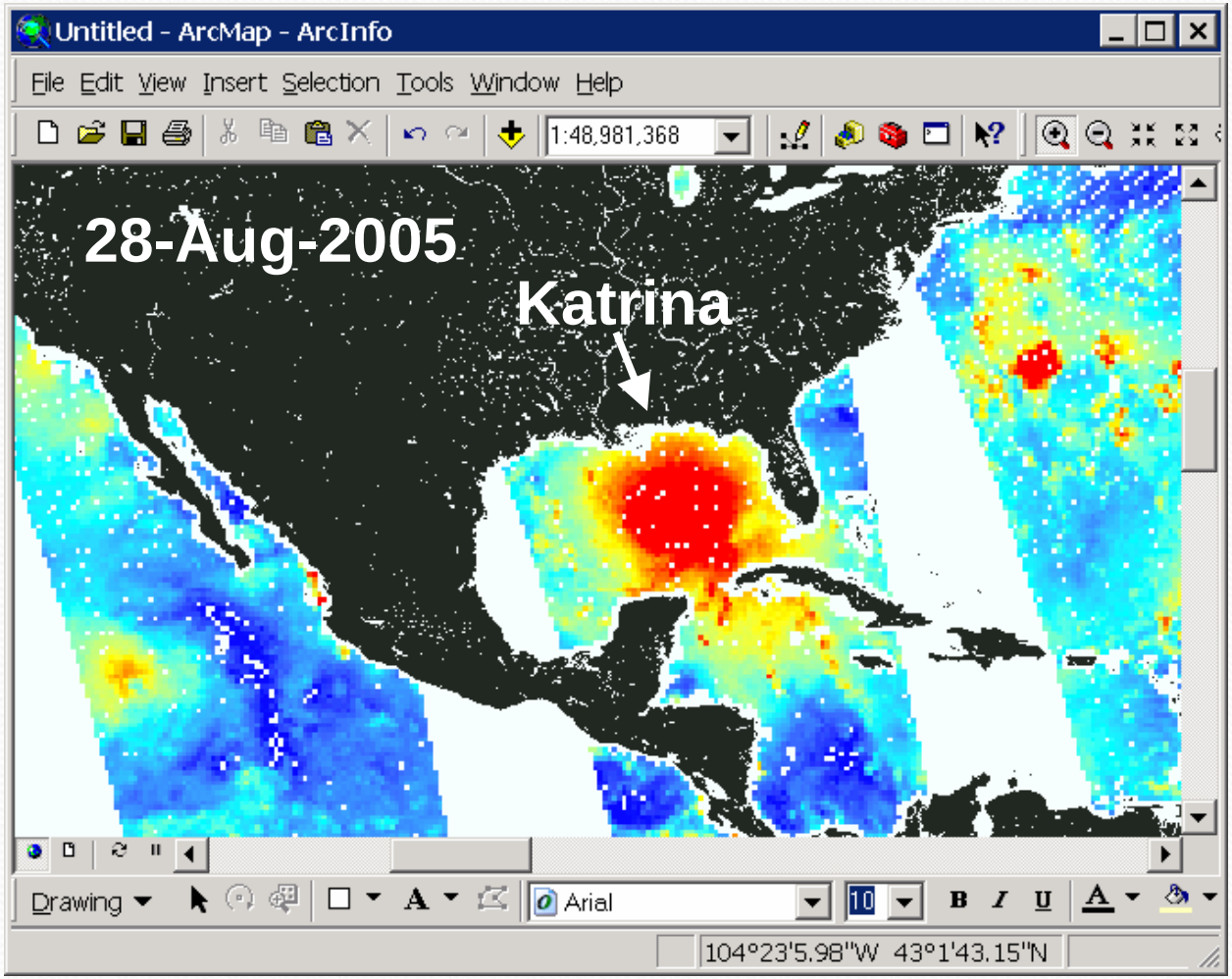
Also: MODIS Aqua and combined MODIS/SeaWiFS

AVISO SSH and Geostrophic Currents

Turtle 37214: 15-Dec-2003 - 18-Dec-2003



PO.DAAC QuickSCAT Winds

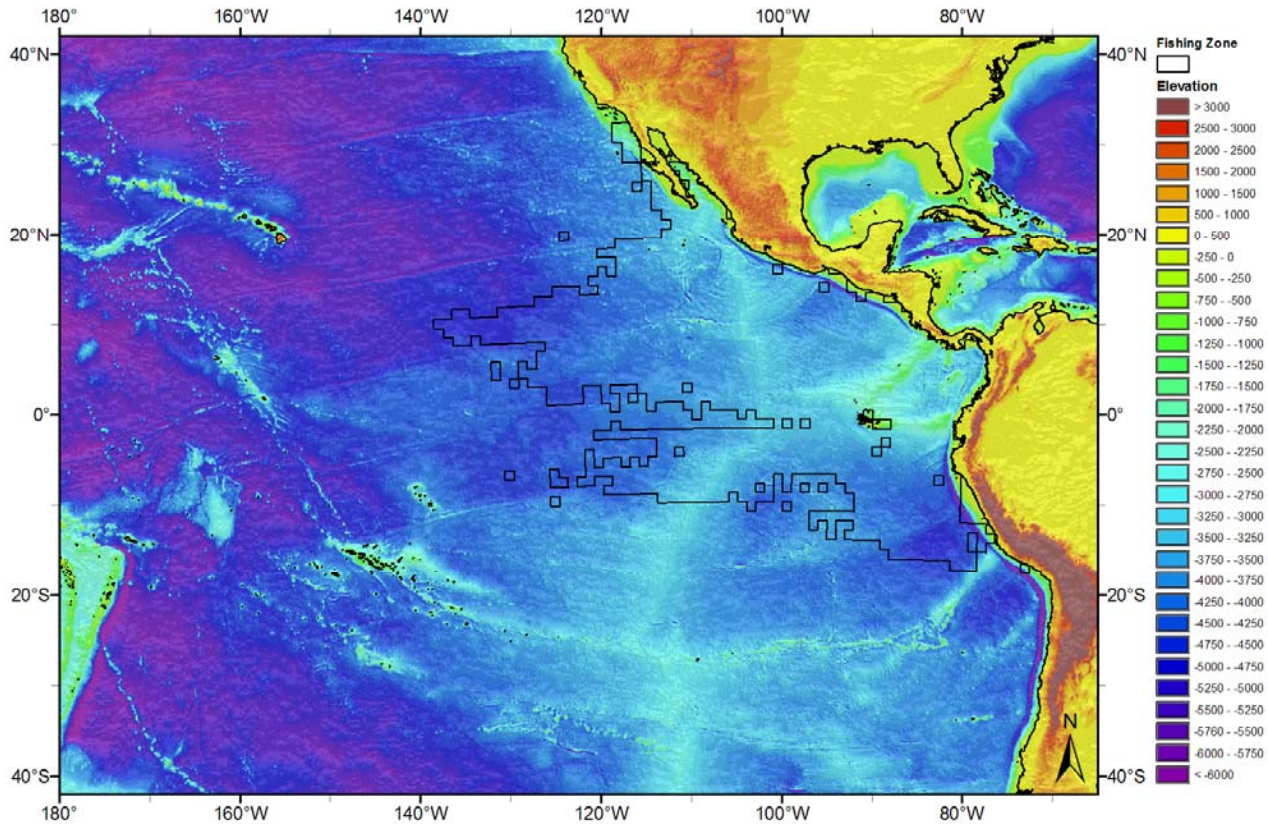


Also: BYU QuickSCAT Sigma-o (approximates sea surface rougness)

Global Bathymetries

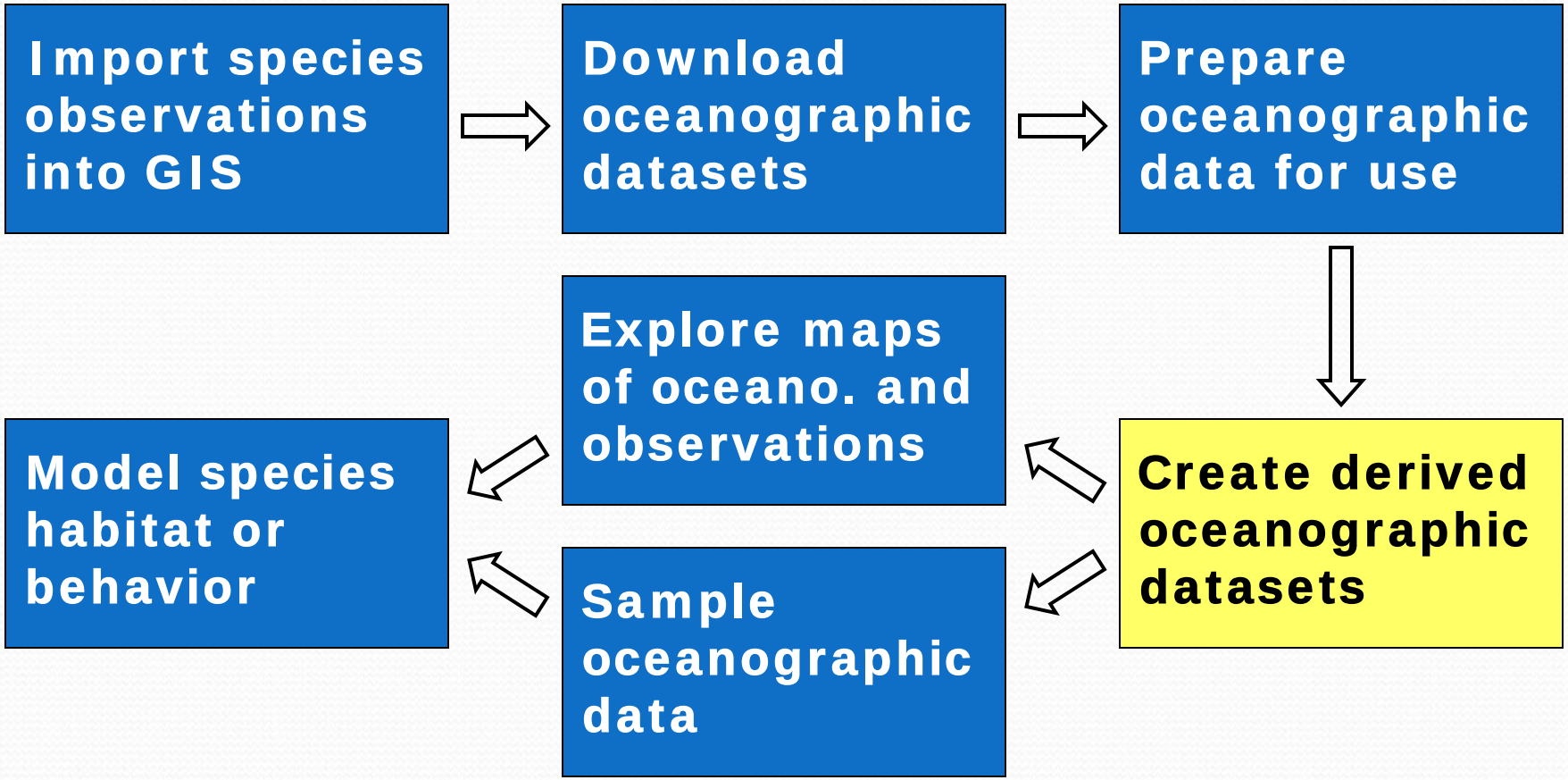
- ETOPO2
- GEBCO
- S2004

Map shows S2004 clipped to ETP



Data: Walter Smith S2004 Global Bathymetry (unpublished)

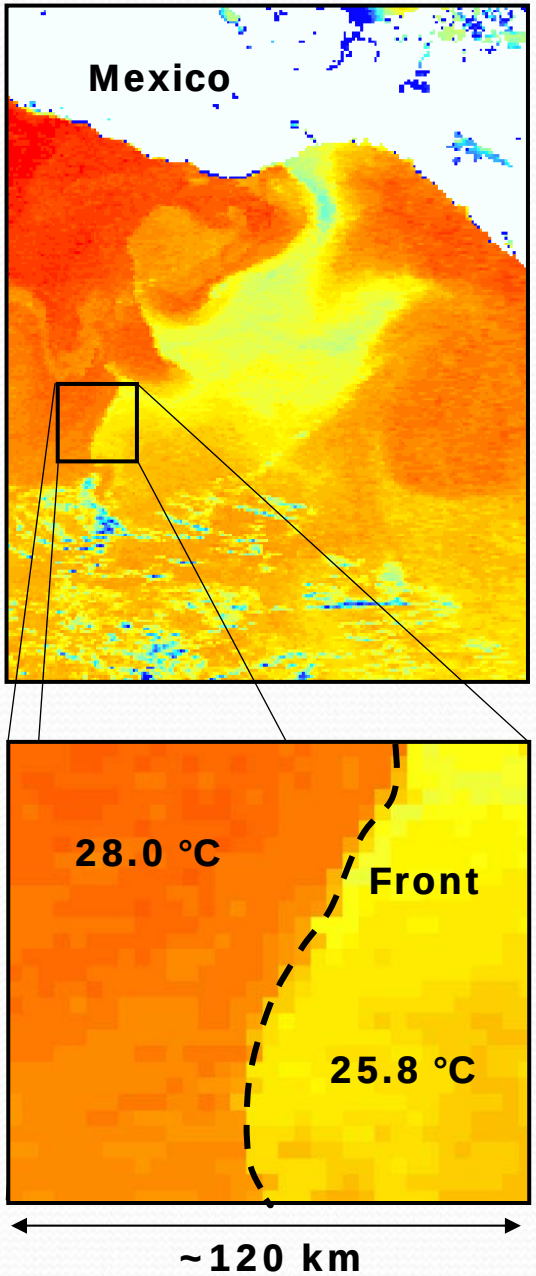
Typical Workflow



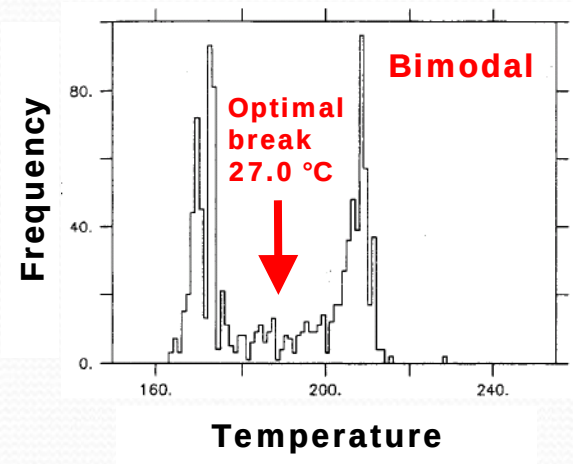
Identifying Surface Temperature Fronts

Cayula-Cornillion Edge Detection Algorithm (1992)

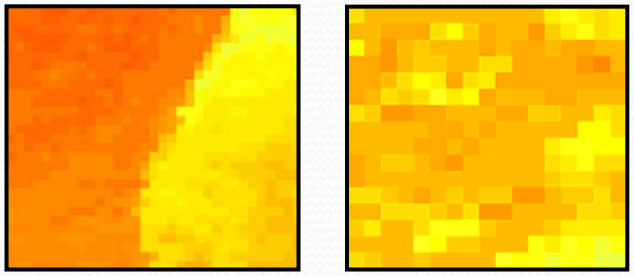
AVHRR Daytime SST
03-Jan-2005



Step 1: Histogram analysis



Step 2: Spatial cohesion test

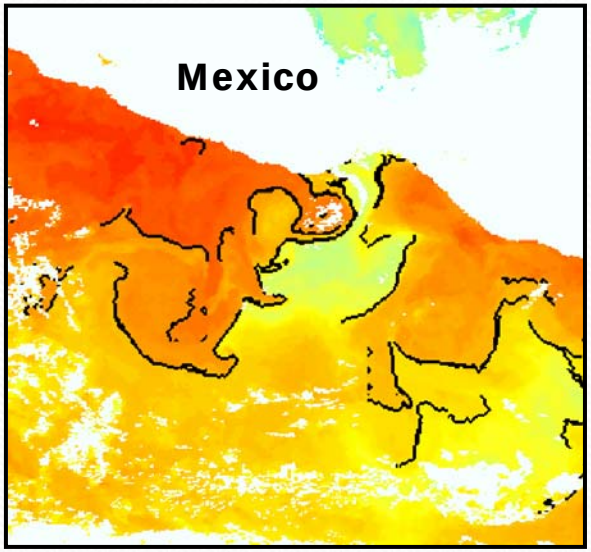


Strong cohesion → front present

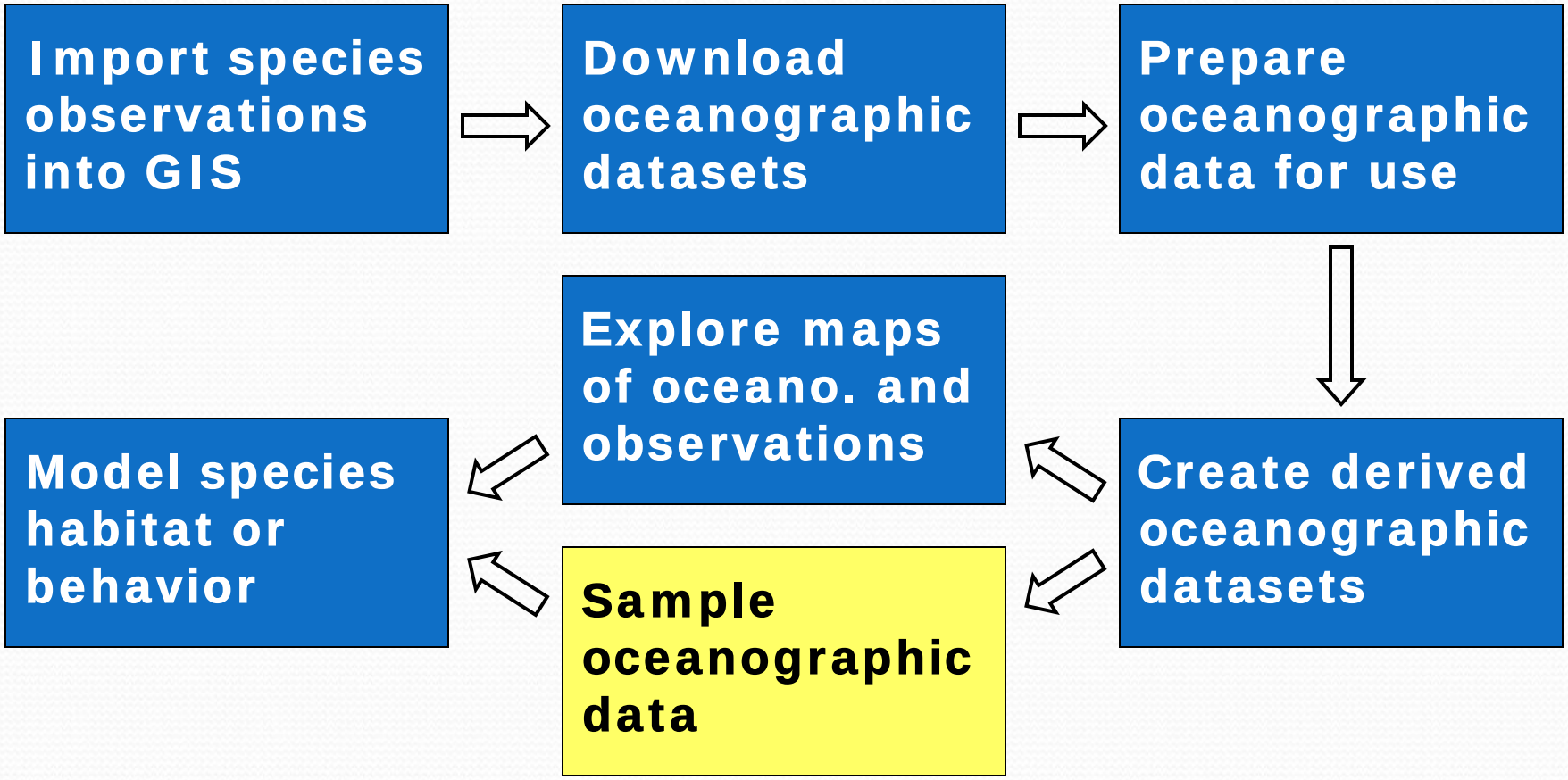
Weak cohesion → no front

- 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

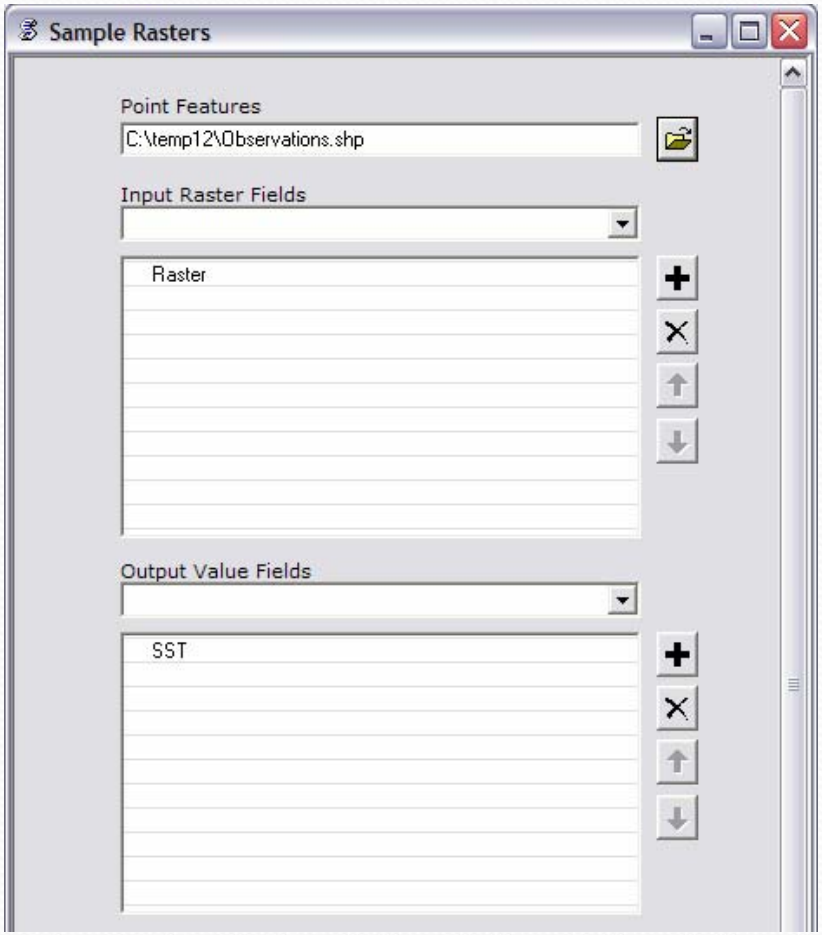
Example Output

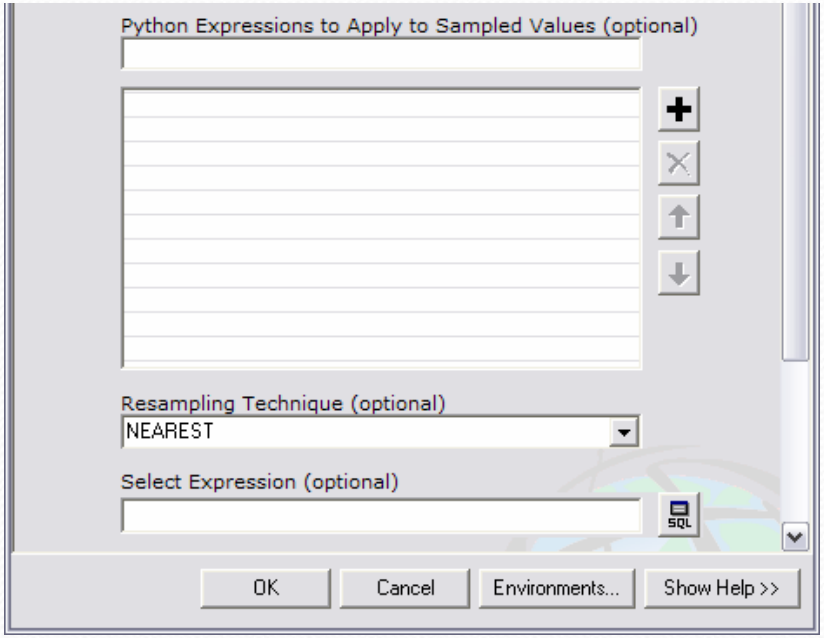


Typical Workflow



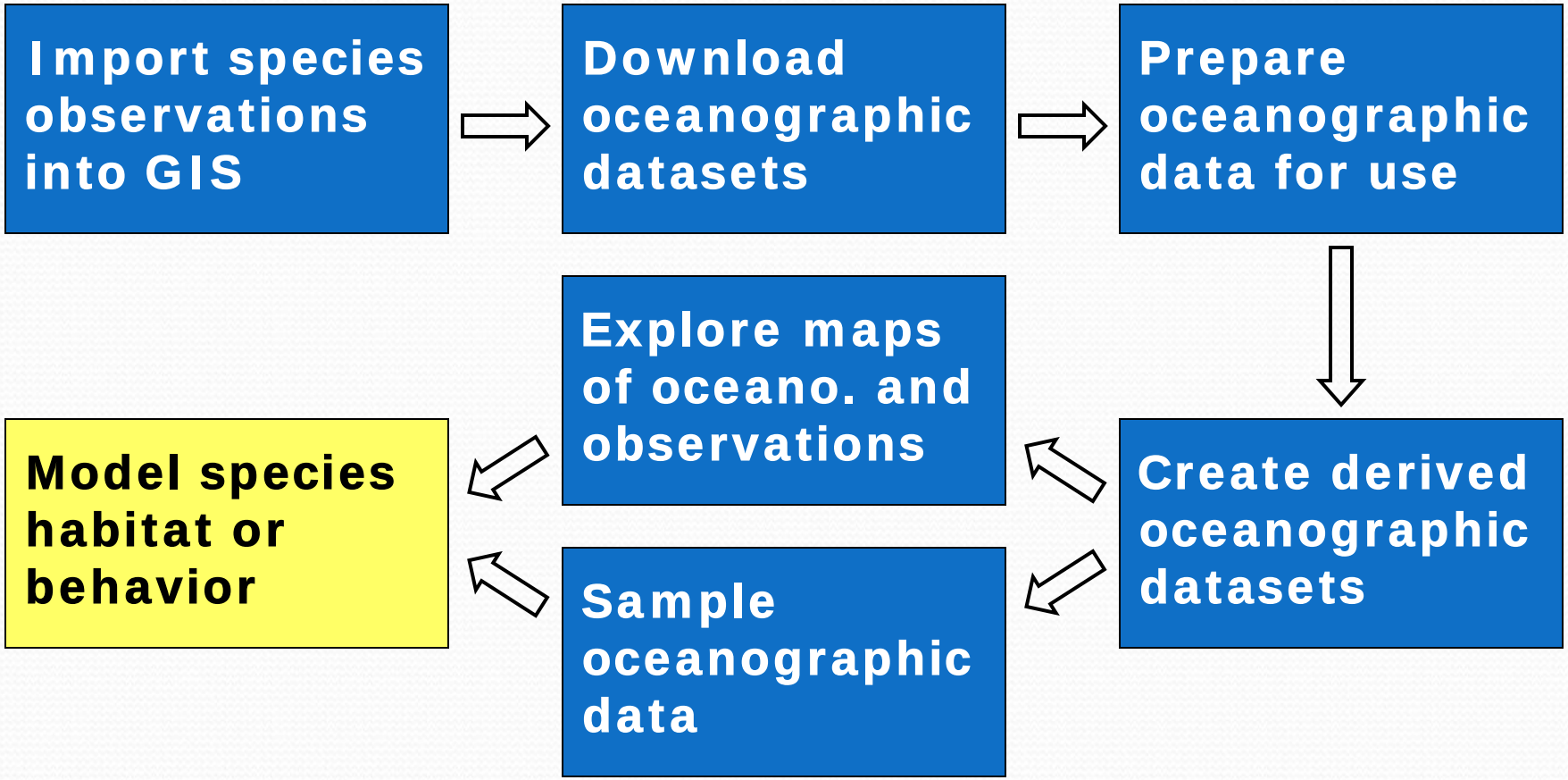
Batch Sampling Tool



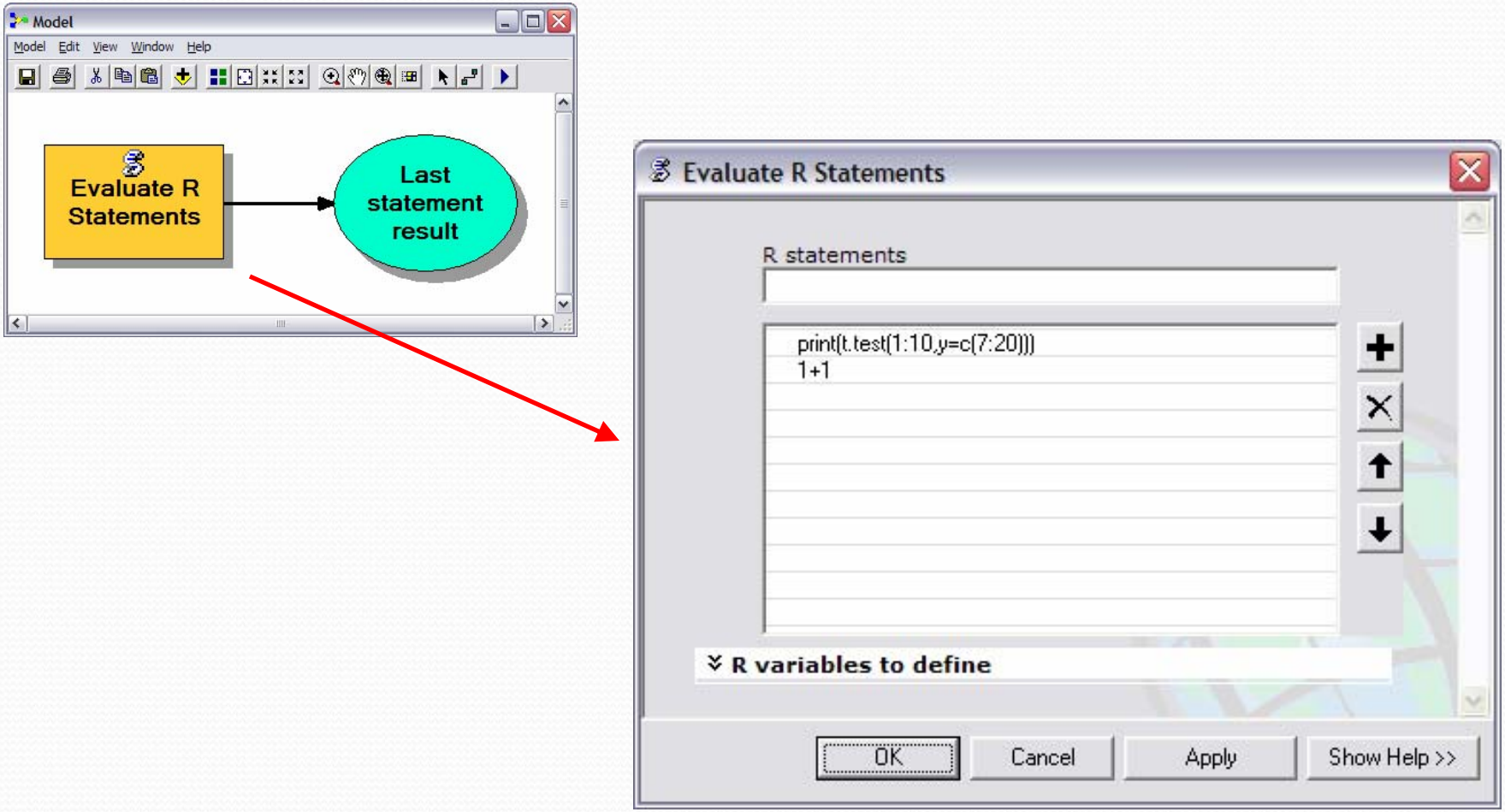


- Sample rasters in 1 or more fields
- Stores values directly in fields!
- Can apply Python expression to sampled values

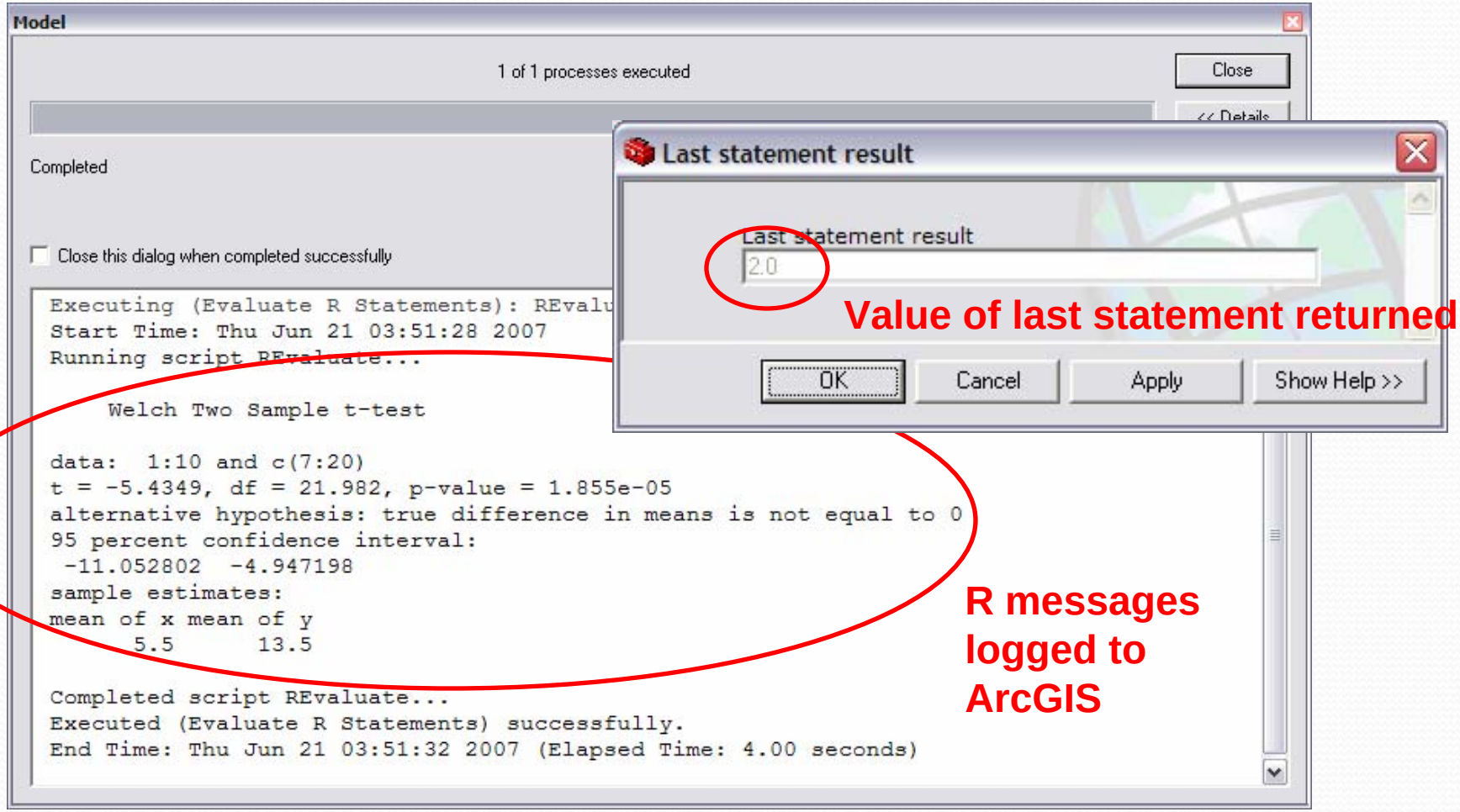
Typical Workflow



Invoking R from ArcGIS



Invoking R from ArcGIS

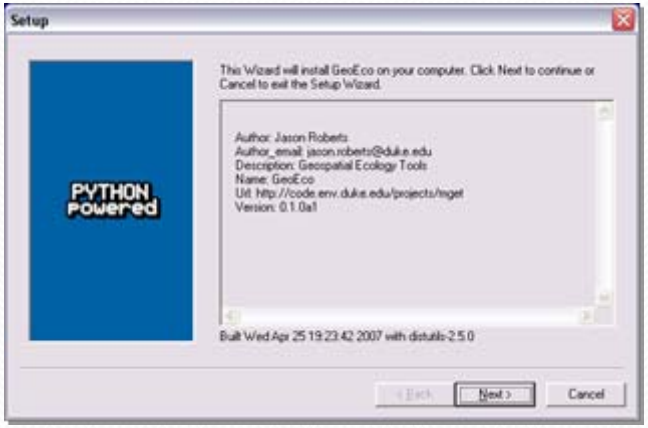


MGET Project Status

- 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

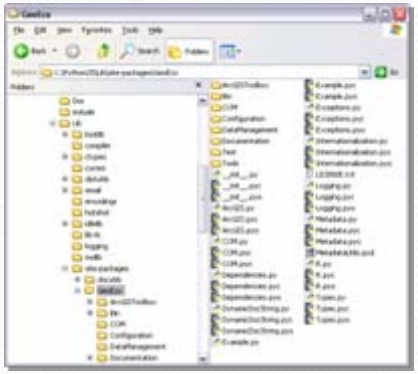
Installing MGET

User executes installation program

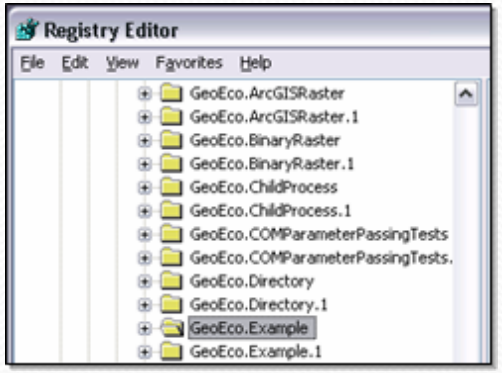


Registers ArcGIS toolbox

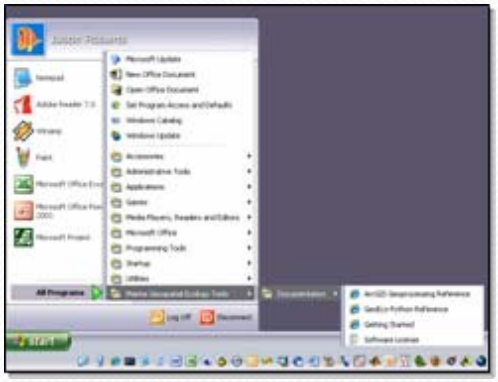
Installs Python package



Registers COM objects

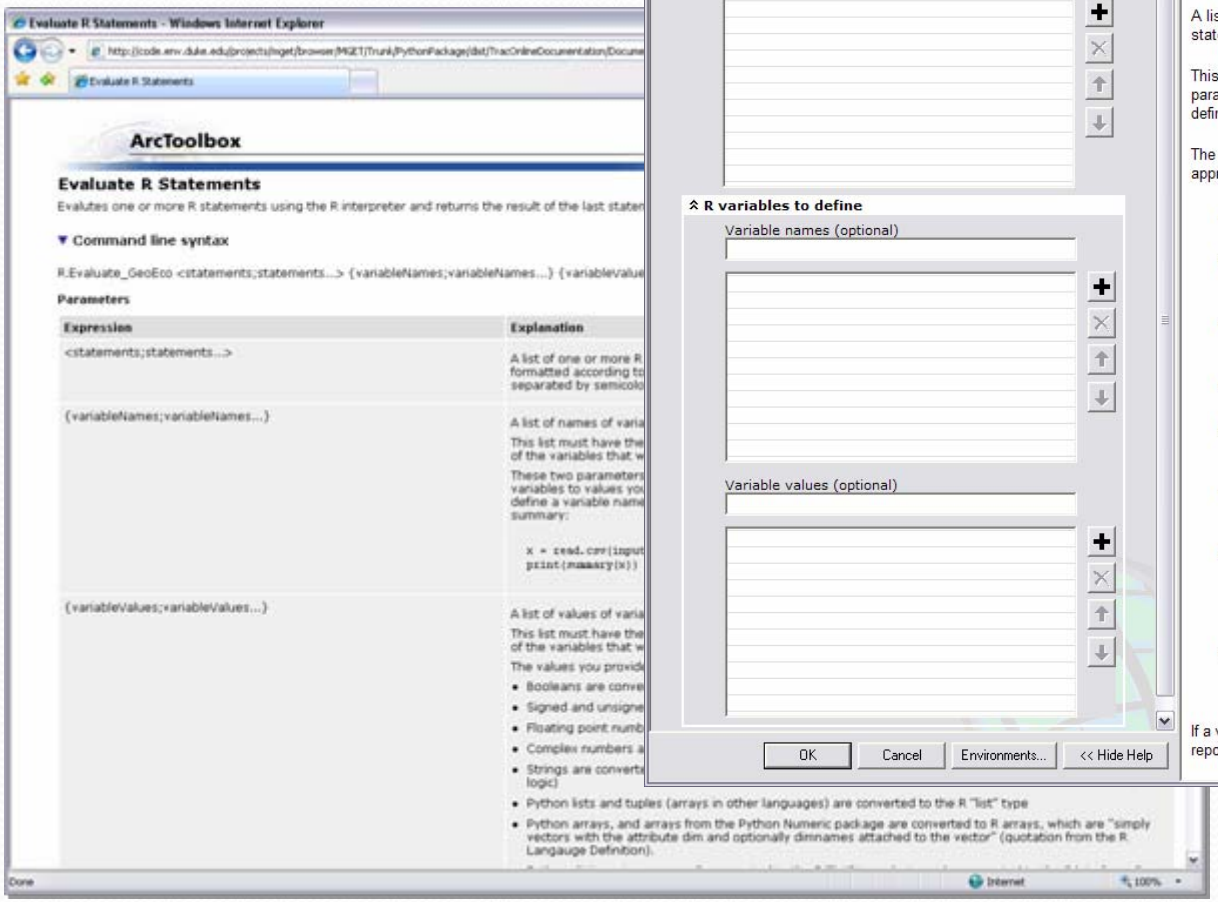


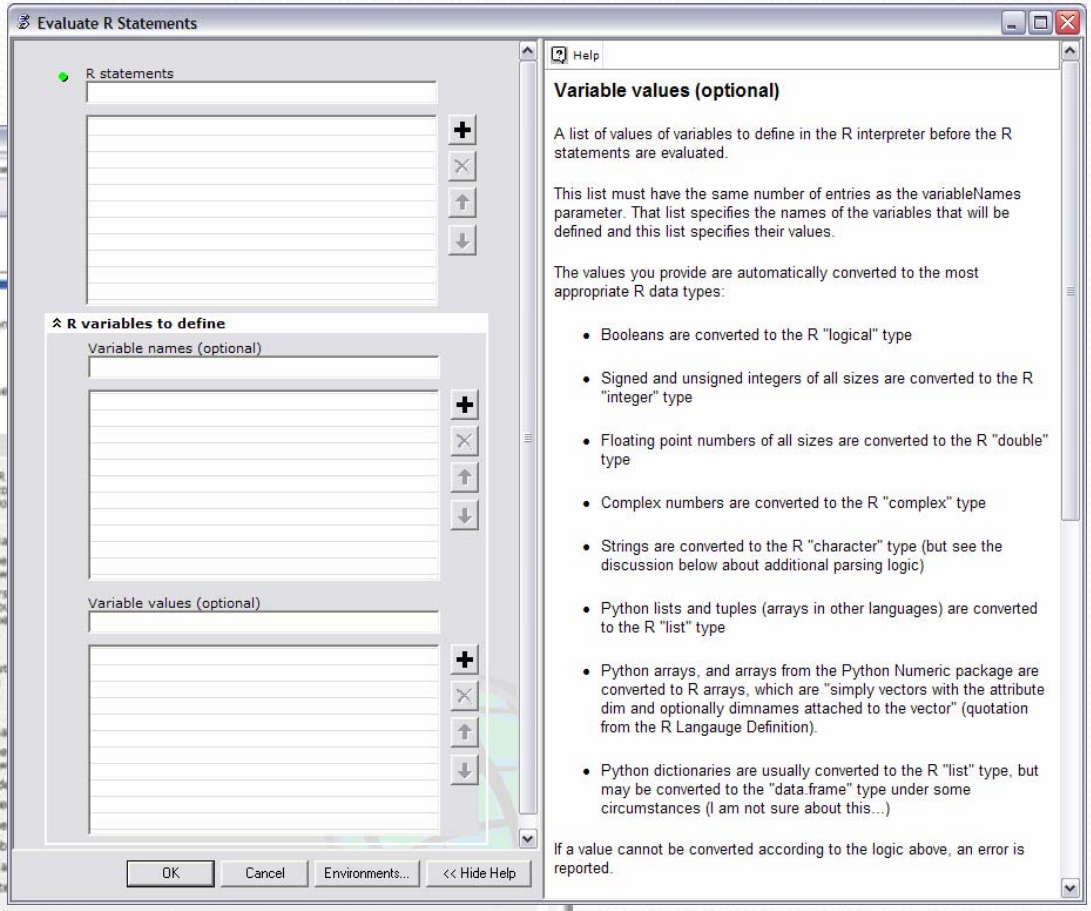
Configures Start menu



ArcGIS documentation

In HTML files





In Arc toolbox

Python documentation

HDF Class - Windows Internet Explorer

GeoEco Python Reference

Previous: GeoEco.DataManagement.HDFs.Module Up: Module Index Next: ExtractHeader Method

HDF Class

Provides methods for manipulating HDF files (HDF version 4 and earlier).

Module: GeoEco.DataManagement.HDFs
Intended use: All methods are recommended for external callers
COM: Exposed as COM class GeoEco.HDF
ArcGIS: Some methods are exposed as ArcGIS geoprocessing tools

Classmethods

ExtractHeader(*inputFile, outputFile*[, *overwriteExisting*])
Extracts an HDF file's header to a text file.

GetSDSDataType(*inputFile, sdsName*)
Returns the data type of a specified Scientific Data Set (SDS) in an HDF file.

GetSDSDimensions(*inputFile, sdsName*)
Returns a list of the dimensions of a specified Scientific Data Set (SDS) in an HDF file.

GetSDSNames(*inputFile*)
Returns a list of the names of Scientific Data Sets (SDSes) present in an HDF file.

SDSToArcGISRaster(*inputFile, outputRaster, sdsName, xLowerLeftCorner, yLowerLeftCorner, cellSize*[, *noDataValue*[, *coordinateSystem*[, *buildPyramids*[, *overwriteExisting*]]]])
Extracts a specified Scientific Data Set (SDS) from an HDF file to an ArcGIS raster.

SDSToArcInfoASCIIGrid(*inputFile, outputFile, sdsName, xLowerLeftCorner, yLowerLeftCorner, cellSize*[, *noDataValue*[, *overwriteExisting*]])
Extracts a specified Scientific Data Set (SDS) from an HDF file to an ArcInfo ASCII Grid.

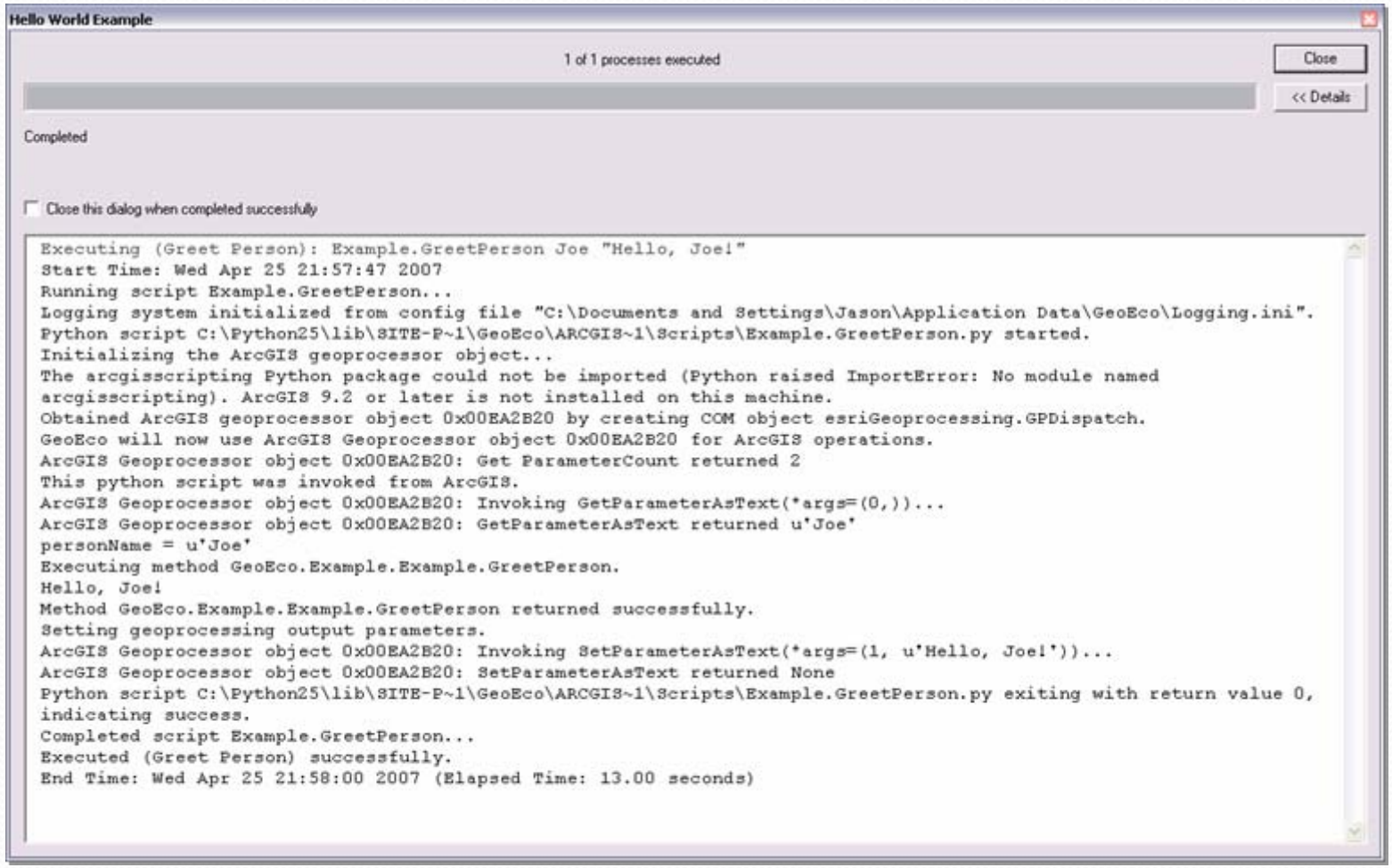
SDSToBinaryRaster(*inputFile, outputFile, sdsName*[, *overwriteExisting*])
Extracts a specified Scientific Data Set (SDS) from an HDF file to a binary raster.

Remarks

At the time this class was developed, two major versions of HDF were under continuing development: HDF, which was at release 4.2r1, and HDF5, which was at release 5-1.6.5. This class can process HDF files but not HDF5 files. It can also process HDF-EOS, a specialized version of HDF, but not HDF5-EOS.

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.

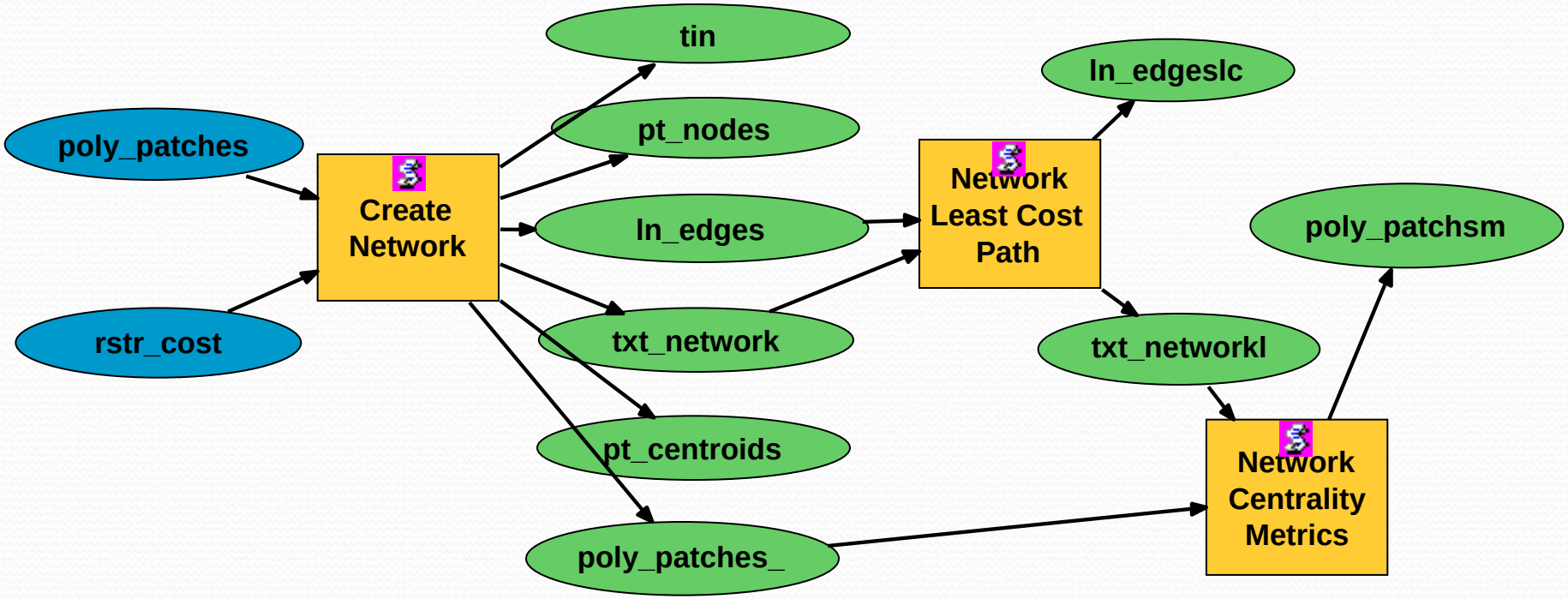
Thanks! Any questions?

Download site:

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

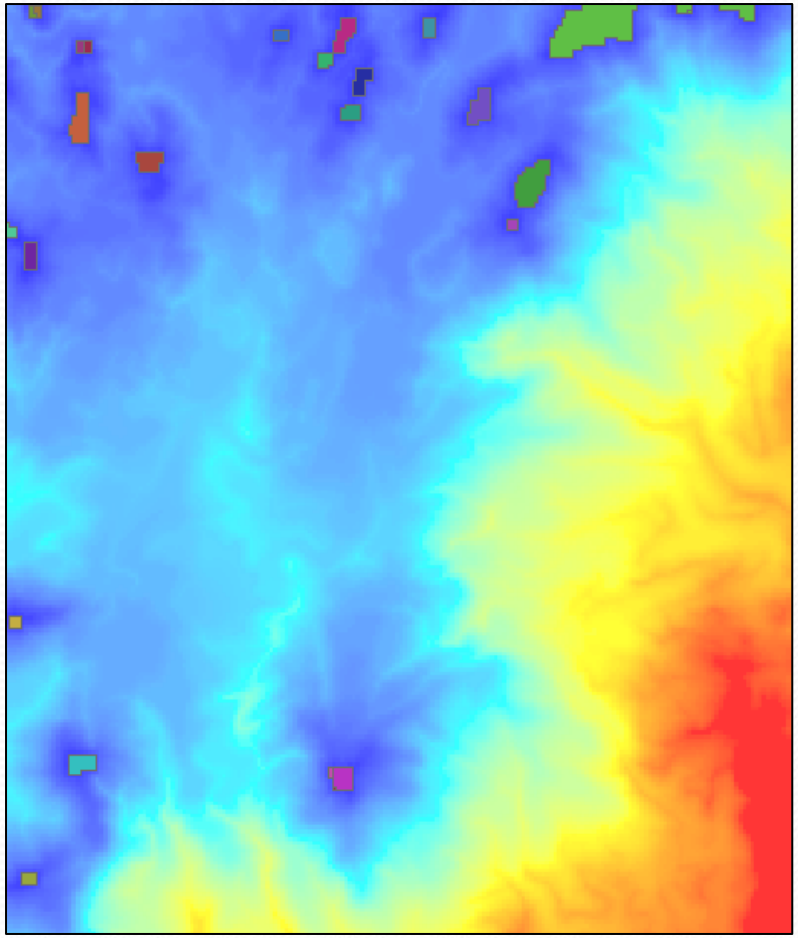
Contact: Jason Roberts
jason.roberts@duke.edu

Advanced Modeling: Ben's Connectivity Modeler



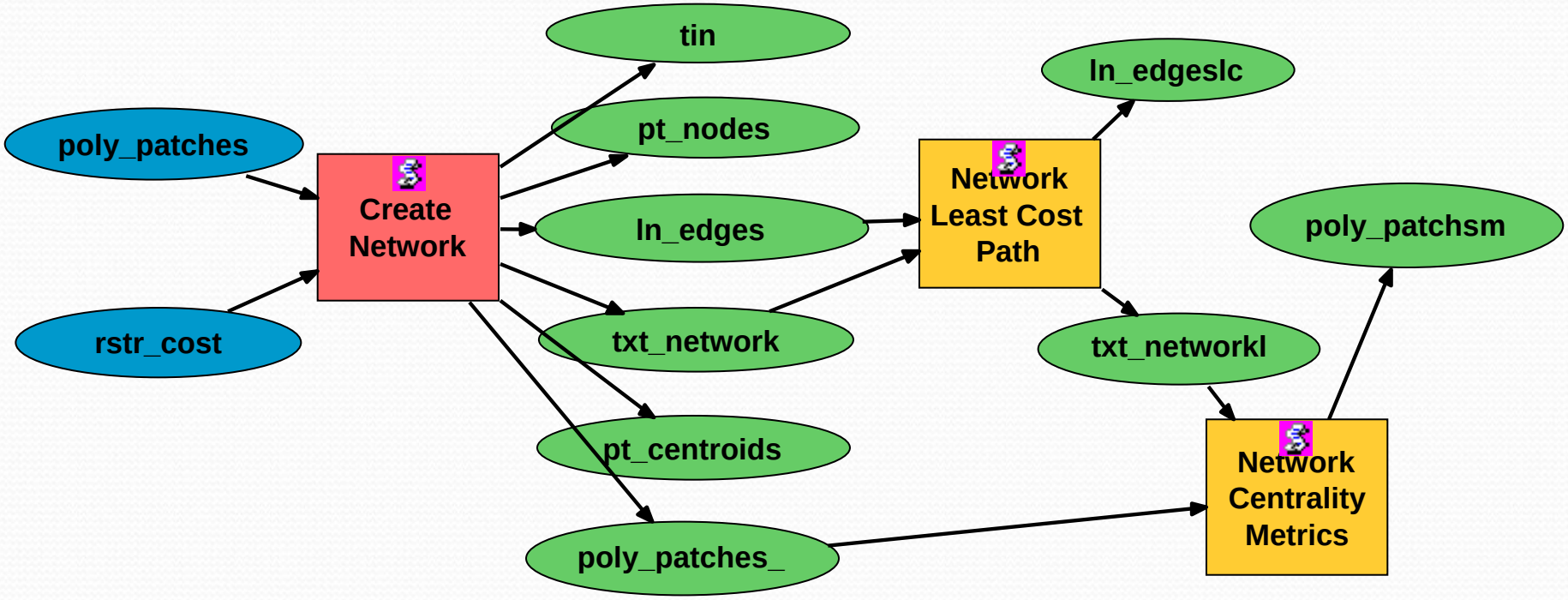
Best et al 2007 (In Review)

The Connectivity Problem

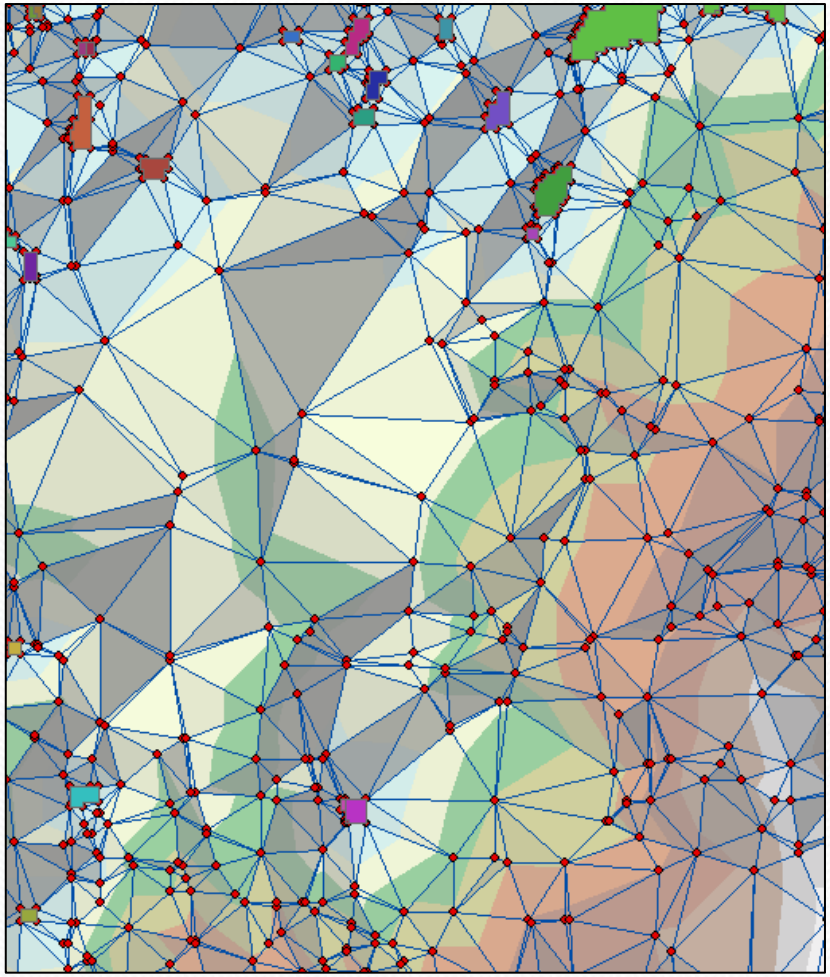


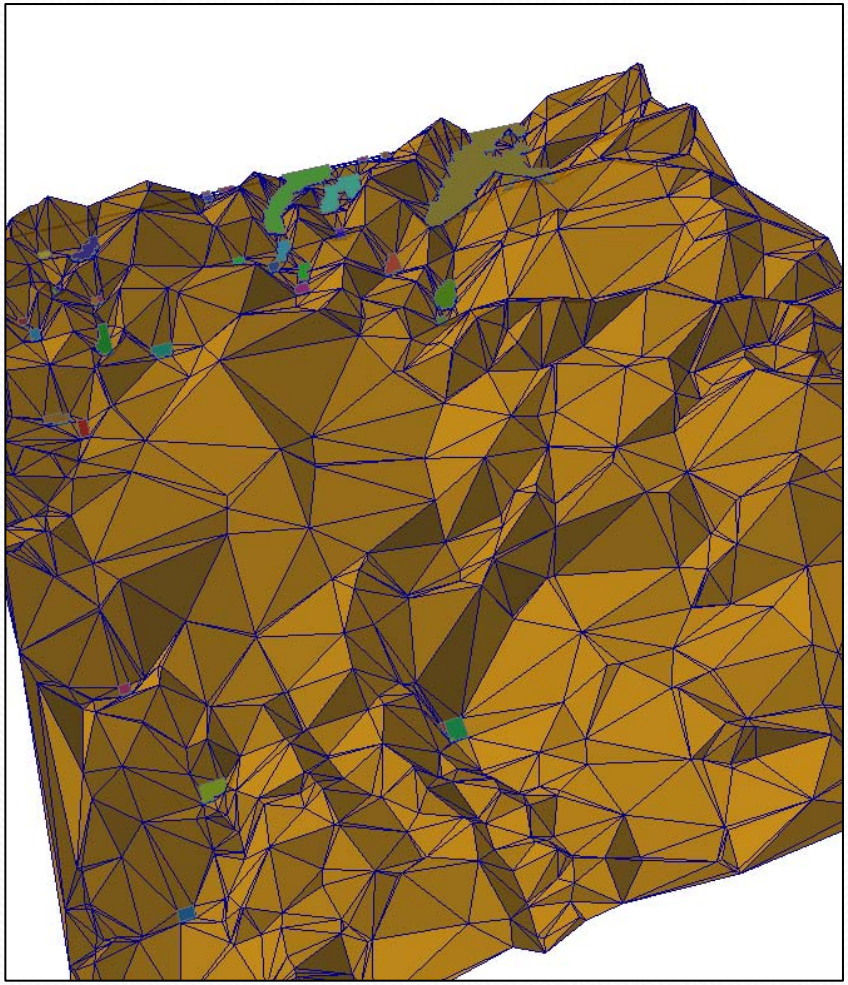
- Say you have a set of patches and a cost surface that describes migration cost
- How to efficiently compute how “connected” patches are to each other?

Step 1: create a network from the cost surface

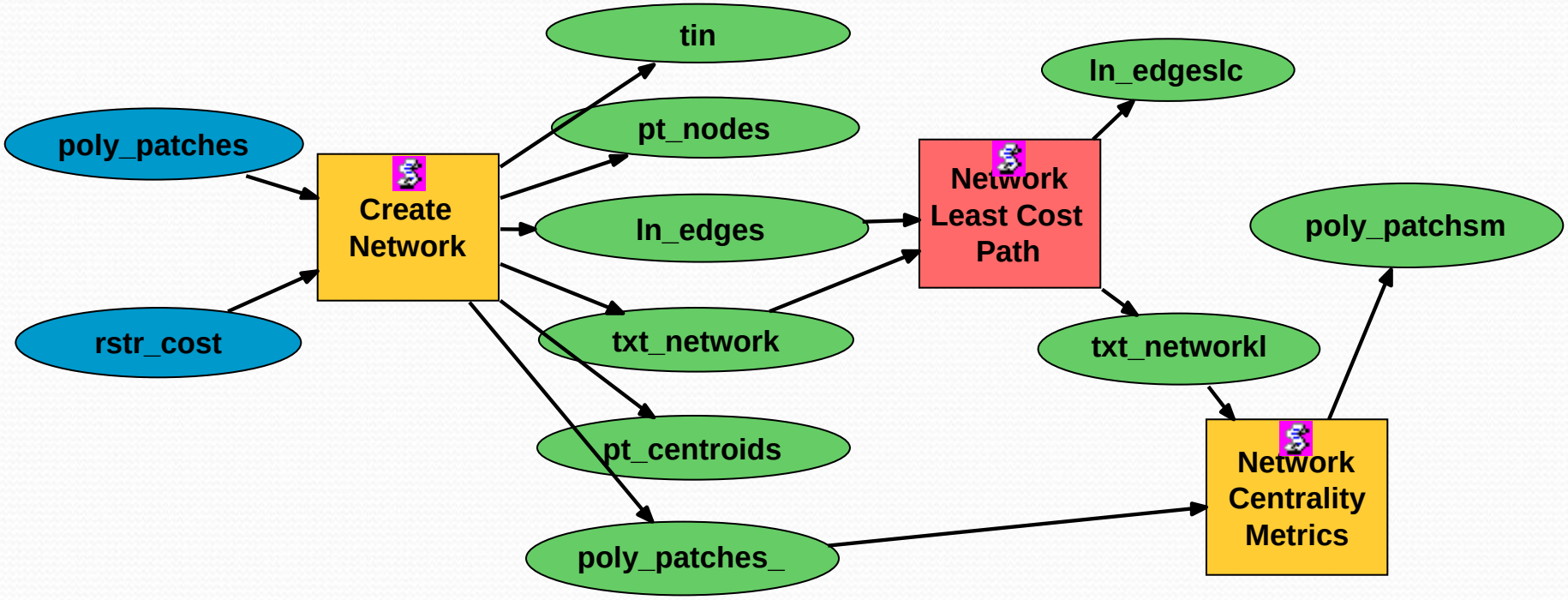


Cost surface is converted to a TIN to create the network

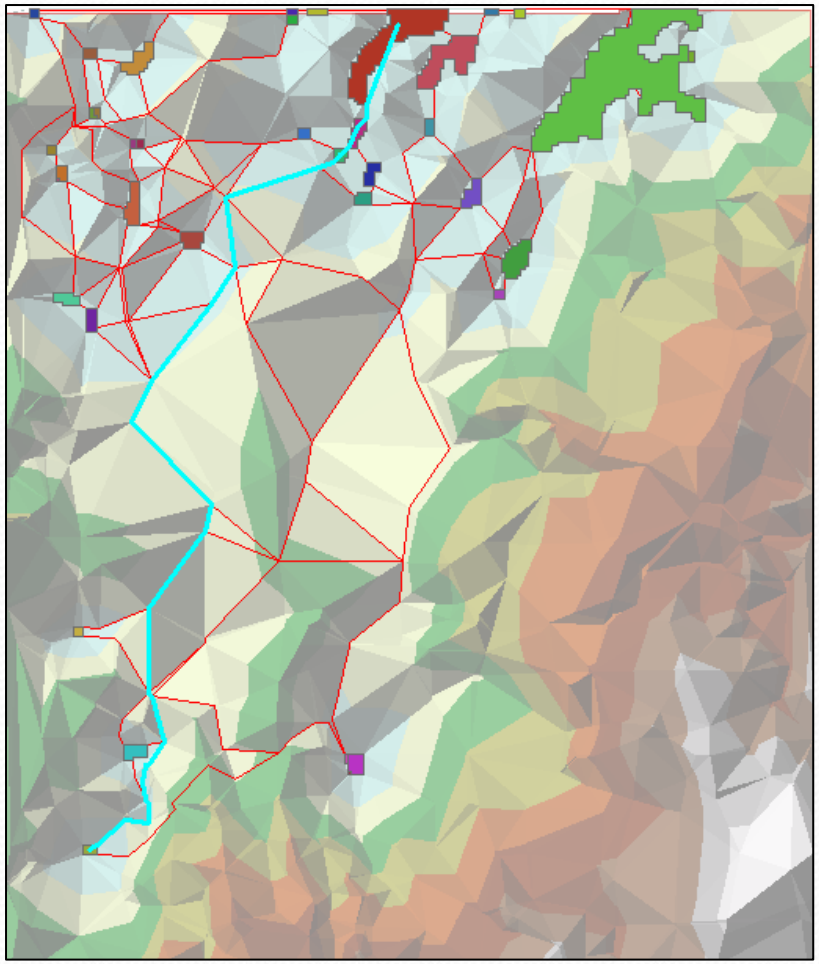




Step 2: calculate the least cost paths for the network

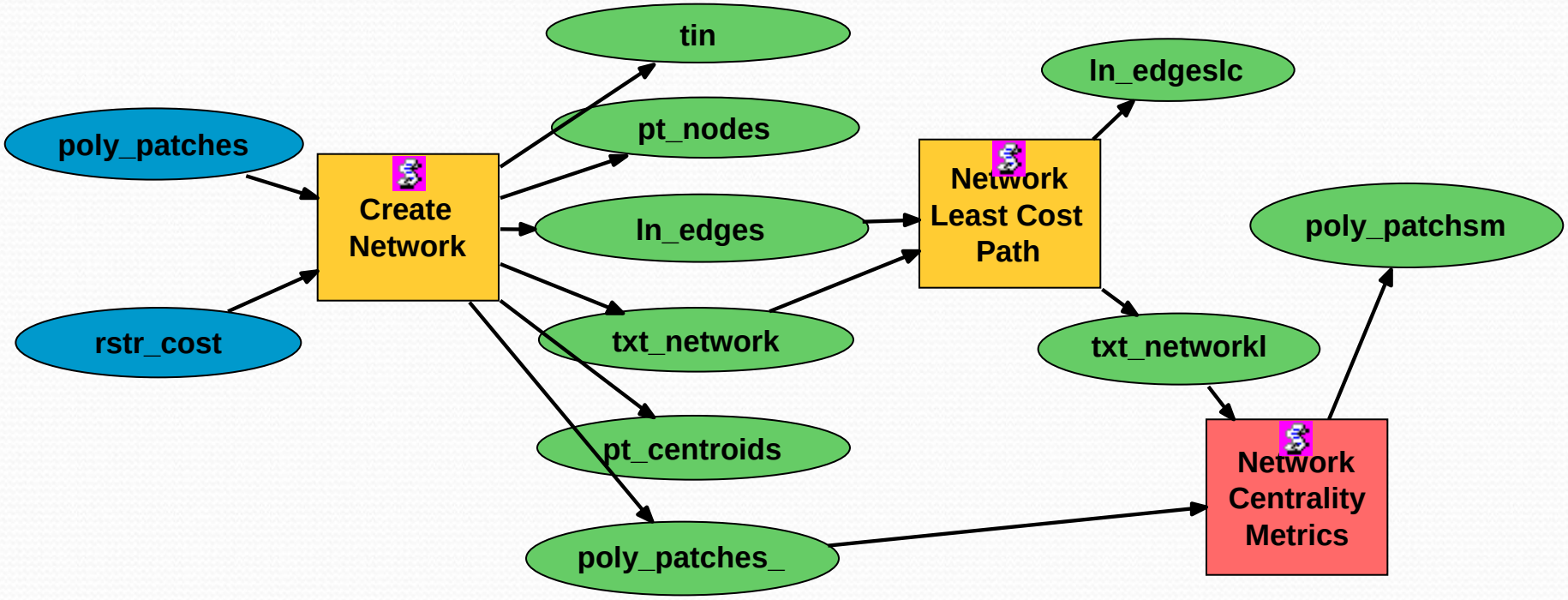


Network least cost paths



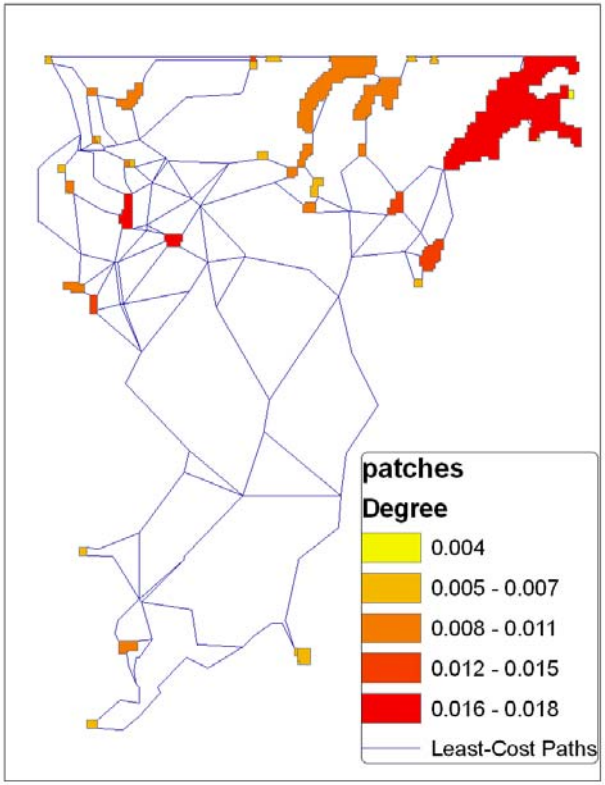
- Djikstra algorithm highly efficient over ArcGIS CostPath function
- Future: create corridors with CostDistance from paths

Step 3: compute network centrality metrics as indices of connectivity

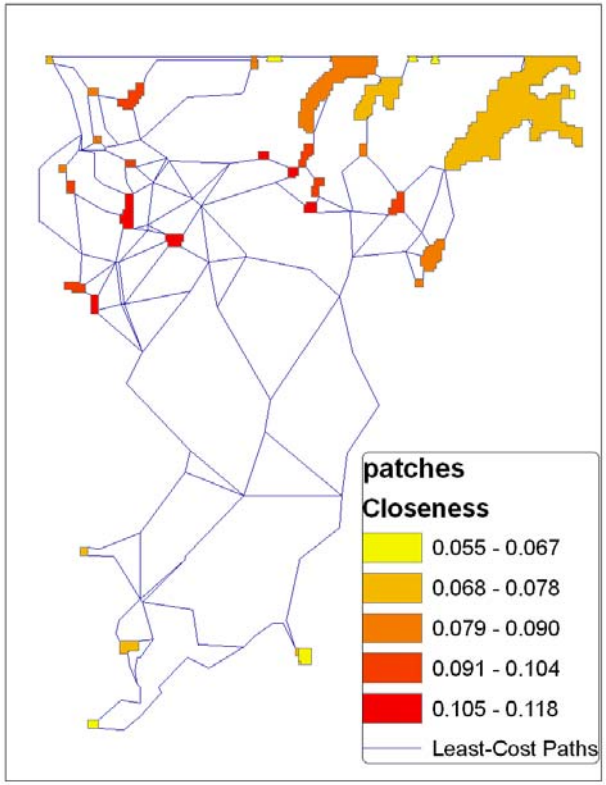


Network centrality metrics

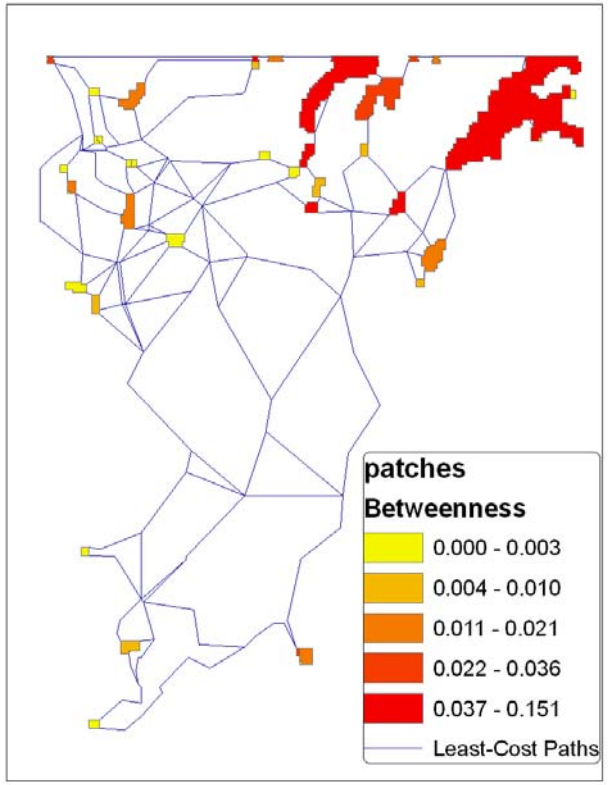
Degree



Closeness



Betweenness



Brandes, 2000. "Faster Evaluation of Shortest-Path Based Centrality Indices." CiteSeer.

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

```
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

2. Define the method

```
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

Input parameter

Method body, to be filled in as next step

3. Fill in the method's body

```
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 = _(u'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

The _ function and use of Unicode strings enable localization of MGET tools

Output parameter

4. Specify metadata at bottom of file

Some of the many things you can do with the metadata

```
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'))
```

Write documentation in reStructuredText

Configure interoperability

Specify appearance of ArcGIS toolbox

Specify strong parameter type and validation options

5. Execute the build script

```
CMD Shell

C:\MGET\MGET\Trunk\PythonPackage>Build_Win32_py24.cmd
Setting environment for using Microsoft Visual Studio 2005 x86 tools.
Python 2.4.4 (#71, Oct 18 2006, 08:34:43) [MSC v.1310 32 bit (Intel)]

*****************************************************************************
* Initializing
*****************************************************************************

GeoEco setup script (last modified Thu Mar 29 19:37:37 2007)
Building GeoEco on Microsoft Windows 5.1 build 2600 with Service Pack 2
Created directory C:\MGET\MGET\Trunk\PythonPackage\build.
Copying directory C:\MGET\MGET\Trunk\PythonPackage\src to C:\MGET\MGET\Trunk\PythonPackage\build\src...

*****************************************************************************
* Generating Metadata.xml
*****************************************************************************

Generating XML metadata for Python modules:
  GeoEco
  GeoEco.ArcGIS
  GeoEco.COM
  GeoEco.DataManagement
  GeoEco.DataManagement.ArcGISRasters
  GeoEco.DataManagement.BinaryRasters
  GeoEco.DataManagement.Directories
  GeoEco.DataManagement.Files
  GeoEco.DataManagement.HDFs
  GeoEco.DataManagement.Processes
  GeoEco.DataManagement.Shapefiles
  GeoEco.Dependencies
  GeoEco.DynamicDocString
  GeoEco.Exceptions
  GeoEco.Internationalization
  GeoEco.Logging
  GeoEco.Metadata
  GeoEco.R
  GeoEco.Test
  GeoEco.Test.ParameterPassingTests
  GeoEco.Tools
  GeoEco.Types
Writing C:\MGET\MGET\Trunk\PythonPackage\build\src\Metadata.xml...

*****************************************************************************
* Generating the ArcGIS toolbox
*****************************************************************************

Created C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts.
Writing ArcGIS wrapper scripts:
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\ArcGISRaster.Copy.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\ArcGISRaster.CopyToDirectory.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\ArcGISRaster.Delete.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\ArcGISRaster.Move.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\ArcGISRaster.MoveToDirectory.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\BinaryRaster.SwapBytes.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\BinaryRaster.ToArcGISRaster.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\BinaryRaster.ToArcInfoASCIIGrid.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\Directory.Copy.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\Directory.CopyToDirectory.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\Directory.Create.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\Directory.CreateSubdirectory.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\Directory.CreateTemporaryDirectory.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\Directory.Delete.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\Directory.Move.py"
  Writing "C:\MGET\MGET\Trunk\PythonPackage\build\src\GeoEco\ArcGISToolbox\Scripts\Directory.MoveToDirectory.py"
```

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:

```
PythonWin
File  Edit  View  Tools  Window  Help
PythonWin 2.5 (r25:51908, Sep 19 2006, 09:52:17) [MSC v.1310 32 bit (Intel)] on win32.
Portions Copyright 1994-2006 Mark Hammond - see 'Help/About PythonWin' for further copyright information.
>>> from GeoEco.Logging import Logger
>>> Logger.Initialize()
2007-04-25 20:00:16,213 INFO Logging system initialized from config file "C:\Documents a
>>> from GeoEco.Example import Example
>>> greeting = Example.GreetPerson(u'Jason')
2007-04-25 20:01:02,819 INFO Hello, Jason!
>>> print greeting
Hello, Jason!
>>> |
Ready                                                    SCRL  00011  005
```

Initialize logging (optional)

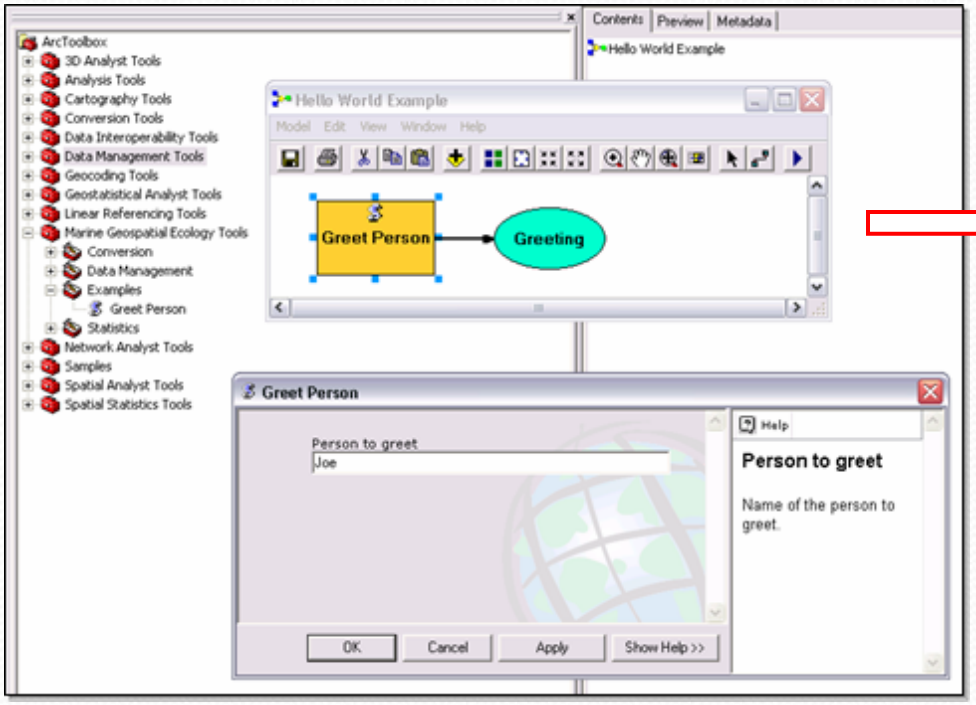
Import the module, invoke method

Log message (format is configurable)

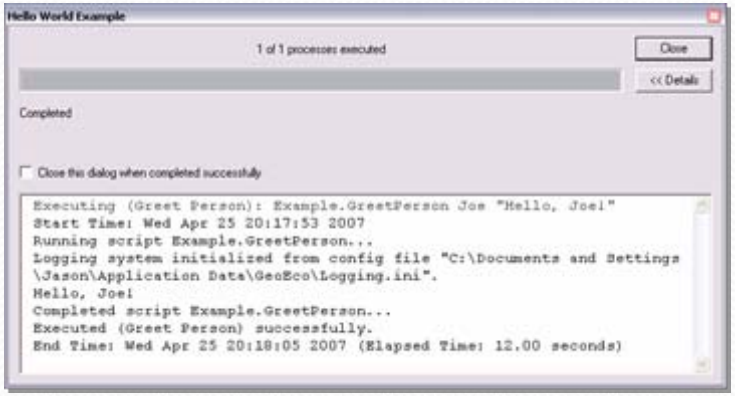
Print returned value

The core framework is platform-independent; it is up to you to determine what platforms your method supports.

Invoking your tool from ArcGIS



Log messages appear in progress window



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)

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

CMD Shell

```
C:\temp2>cscript test.vbs
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

2007-04-25 20:46:12,826 INFO Logging system initialized from config file
2007-04-25 20:46:12,887 INFO Hello, Joe!
Hello, Joe!

C:\temp2>_
```

Invoking your tool from .Net

Test - Microsoft Visual Studio

File Edit View Refactor Project Build Debug Data Tools Window Community Help

Start Page | Program.cs*

Test.Program | Main(string[] args)

```
using GeoEco;

namespace Test
{
    class Program
    {
        static void Main(string[] args)
        {
            Logger logger = new LoggerClass();
            logger.Initialize(false, null);
            Example example = new ExampleClass();
            string greeting = example.GreetPerson("Joe");
            Console.WriteLine(greeting);

            HDF hdf = new HDFClass();
            hdf.|
        }
    }
}
```

C# Example

ExtractHeader
GetHashCode
GetSDSDataType
GetSDSDimensions
GetSDSNames
GetType
SDSToArcGISRaster
SDSToArcInfoASCIIGrid
SDSToBinaryRaster

⇦ IntelliSense is fully functional!

Solution Explorer - Solution 'Test' (1 project)

- Solution 'Test' (1 project)
  - Test
    - Properties
    - References
      - GeoEco
      - System
      - System.Data
      - System.Xml
    - Program.cs

Properties

Add Reference

.NET | COM | Projects | Browse | Recent

Component Name	TypeLib Version	Path
FpWrap 3.0 Type Library	1.0	C:\Program Files\Comm
FStock 2.0 Type Library	2.0	C:\PROGRA~1\COMMO
fwUI 1.0 Type Library	1.0	C:\Program Files\Nortor
GBDetect 1.0 Type Library	1.0	C:\Program Files\Adobe
GeoEco 0.1 Type Library	0.1	C:\Python25\lib\site-pa
GPIProxy 1.0 Type Library	1.0	C:\Program Files\InterV
GXCovProp	2.0	C:\Program Files\ArcGIS
HelloWorld	1.0	C:\Documents and Setti
Help Center Behaviors 1.0 Typ...	1.0	C:\WINDOWS\PCHealt
Help Center UI 1.0 Type Library	1.0	C:\WINDOWS\PCHealt
Help Service 1.0 Type Library	1.0	C:\WINDOWS\pchealth'
Help System Services 1.0 Type ...	1.0	C:\Program Files\Comm
HHActiveX 1.0 Type Library	1.0	C:\WINDOWS\system3
HHCtrl 4.0 Type Library	4.0	C:\WINDOWS\system3

OK | Cancel

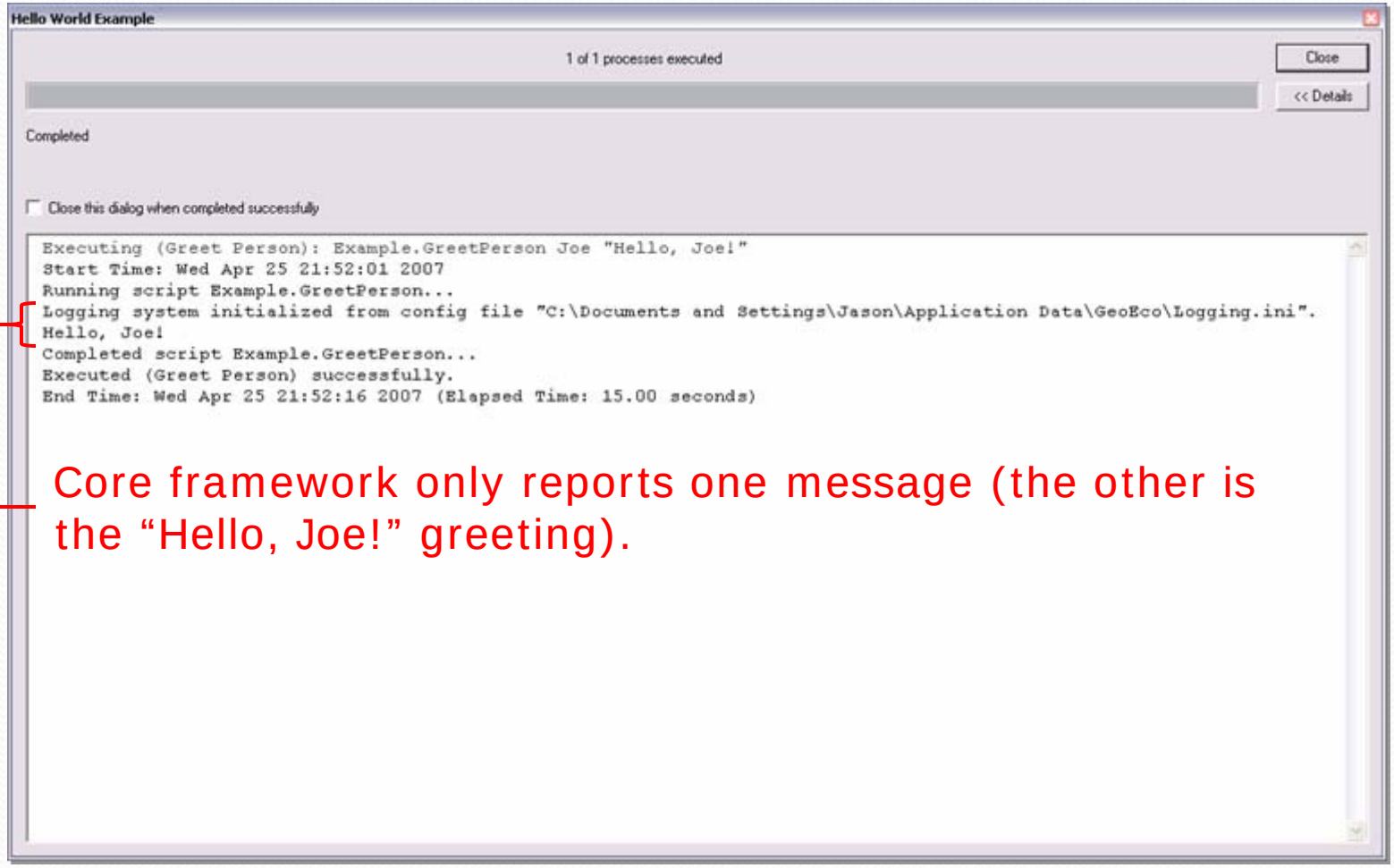
Add a reference to the GeoEco Type Library

Error List

Build succeeded | Ln 17 | Col 17 | Ch 17

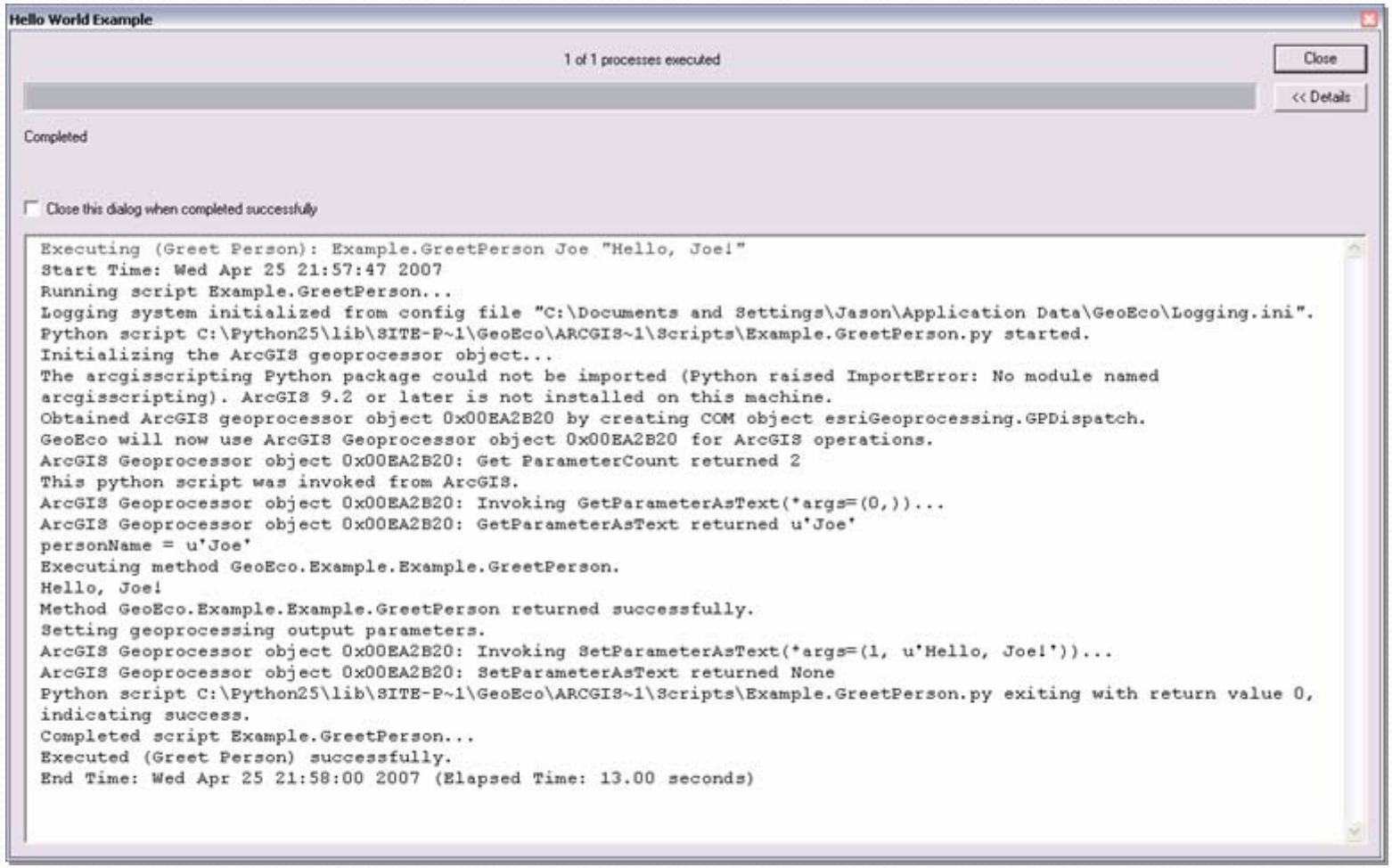
Invocation occurs through early-bound ("vtable") COM, not COM Automation

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

Dependencies checked and geoprocessor initialized here

Wrapped geoprocessor object logs all calls

```
@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')

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')])
```

ArcGIS dependencies

Resulting output (debug logging enabled)

PythonWin

File Edit View Tools Window Help

```
PythonWin 2.5 (r25:51908, Sep 19 2006, 09:52:17) [MSC v.1310 32 bit (Intel)] on win32.
Portions Copyright 1994-2006 Mark Hammond - see 'Help/About PythonWin' for further copyright information.
>>> from GeoEco.Logging import Logger
>>> Logger.Initialize()
2007-04-25 22:55:01,891 INFO Logging system initialized from config file "C:\Documents and Settings\Jason\Application Data\GeoEco\Logging.ini".
>>> from GeoEco.Example import Example
>>> Example.CreateZeroRaster(u'c:\\temp2\\foobar2')
2007-04-25 22:55:10,213 DEBUG Checking software dependency: ArcGIS version 9.1 or later.
2007-04-25 22:55:10,223 DEBUG Initializing the ArcGIS geoprocessor object...
2007-04-25 22:55:10,233 DEBUG The arcgisscripting Python package could not be imported (Python raised ImportError: No module named arcgisscripting). ArcGIS
9.2 or later is not installed on this machine.
2007-04-25 22:55:11,384 DEBUG Obtained ArcGIS geoprocessor object 0x01463918 by creating COM object esriGeoprocessing.GPDispatch.
2007-04-25 22:55:11,394 DEBUG GeoEco will now use ArcGIS Geoprocessor object 0x01463918 for ArcGIS operations.
2007-04-25 22:55:11,394 DEBUG Retrieving the ArcGIS version numbers from the Windows Registry.
2007-04-25 22:55:11,404 DEBUG Checking software dependency: Python win32api module.
2007-04-25 22:55:11,404 DEBUG Checking software dependency: Python win32con module.
2007-04-25 22:55:11,414 DEBUG ArcGIS 9.1 Service Pack 2 is installed.
2007-04-25 22:55:11,424 DEBUG Checking software dependency: ArcGIS "spatial" extension.
2007-04-25 22:55:11,424 DEBUG ArcGIS Geoprocessor object 0x01463918: Invoking CheckOutExtension(*args=(u'spatial',))...
2007-04-25 22:55:21,859 DEBUG ArcGIS Geoprocessor object 0x01463918: CheckOutExtension returned u'CheckedOut'
2007-04-25 22:55:21,859 DEBUG ArcGIS Geoprocessor object 0x01463918: Invoking Exists(*args=(u'c:\\temp2\\foobar2',))...
2007-04-25 22:55:27,697 DEBUG ArcGIS Geoprocessor object 0x01463918: Exists returned False
2007-04-25 22:55:27,697 DEBUG ArcGIS Geoprocessor object 0x01463918: Invoking CreateConstantRaster_sa(*args=(u'c:\\temp2\\foobar2', 2.5, 'FLOAT', 1.0, '0.0
0.0 1.0 1.0'))...
2007-04-25 22:55:30,061 INFO Executing (CreateConstantRaster_1): CreateConstantRaster c:\temp2\foobar2 2.5 FLOAT 1 "0 0 1 1"
2007-04-25 22:55:30,061 INFO Start Time: Wed Apr 25 22:55:27 2007
2007-04-25 22:55:30,071 INFO
2007-04-25 22:55:30,081 INFO
2007-04-25 22:55:30,081 INFO
2007-04-25 22:55:30,091 INFO Executed (CreateConstantRaster_1) successfully.
2007-04-25 22:55:30,091 INFO End Time: Wed Apr 25 22:55:29 2007 (Elapsed Time: 2.00 seconds)
2007-04-25 22:55:30,101 DEBUG ArcGIS Geoprocessor object 0x01463918: CreateConstantRaster_sa returned u'c:\\temp2\\foobar2'
>>>
```

Initialize geoprocessor

Other dependency checks

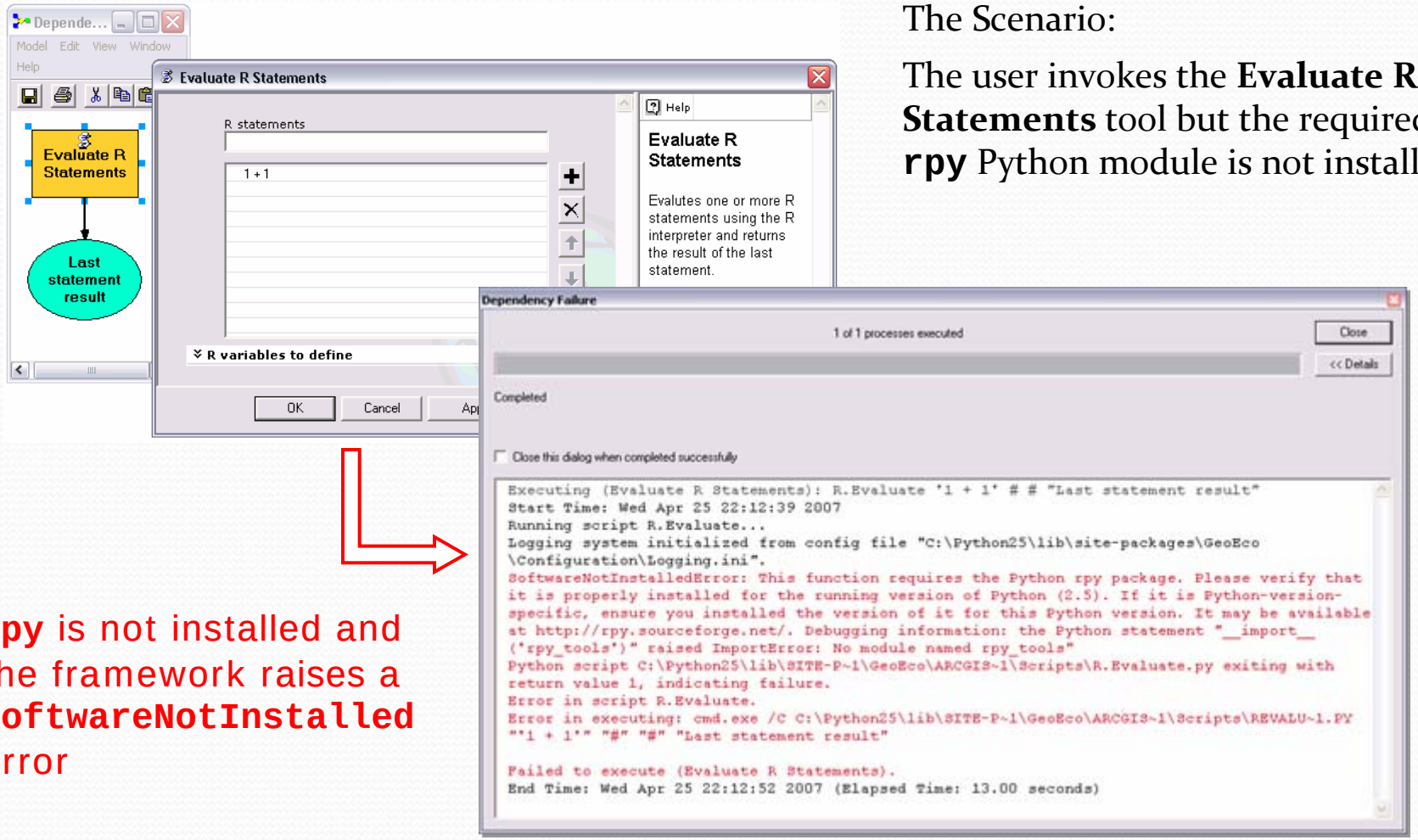
Logged ArcGIS interactions

Ready SCRL 00031 005

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



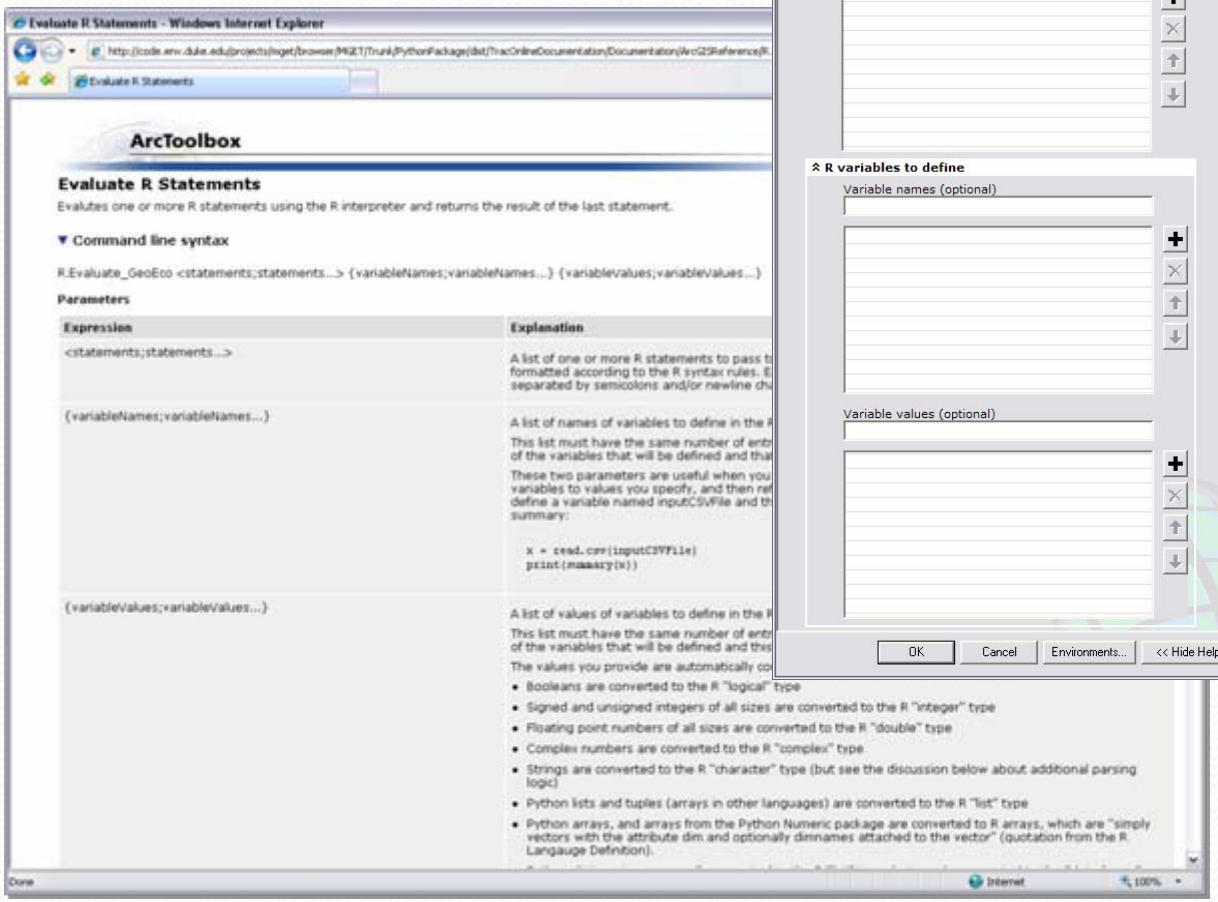
The Scenario:

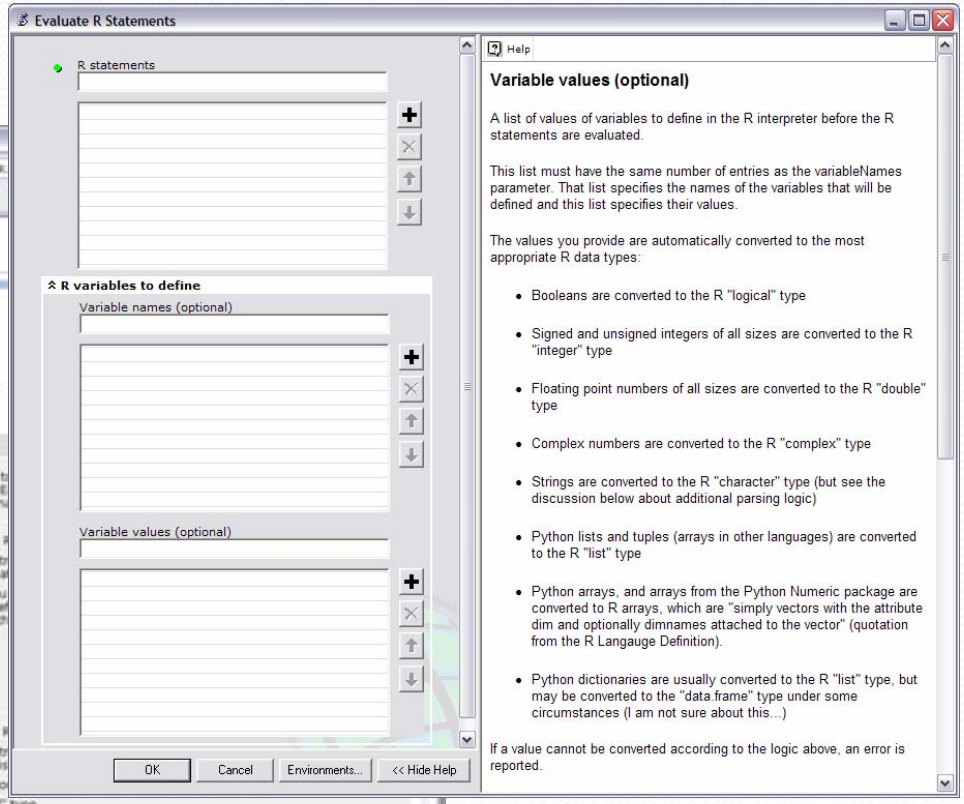
The user invokes the **Evaluate R Statements** tool but the required **`rpy`** Python module is not installed

`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

HDF Class - Windows Internet Explorer

GeoEco Python Reference

Previous: GeoEco.DataManagement.HDFs Module Up: Module Index Next: ExtractHeader Method

HDF Class

Provides methods for manipulating HDF files (HDF version 4 and earlier).

Module: GeoEco.DataManagement.HDFs
Intended use: All methods are recommended for external callers
COM: Exposed as COM class GeoEco.HDF
ArcGIS: Some methods are exposed as ArcGIS geoprocessing tools

Classmethods

ExtractHeader(*inputFile, outputFile*[, *overwriteExisting*])
Extracts an HDF file's header to a text file.

GetSDSDataType(*inputFile, sdsName*)
Returns the data type of a specified Scientific Data Set (SDS) in an HDF file.

GetSDSDimensions(*inputFile, sdsName*)
Returns a list of the dimensions of a specified Scientific Data Set (SDS) in an HDF file.

GetSDSNames(*inputFile*)
Returns a list of the names of Scientific Data Sets (SDSes) present in an HDF file.

SDSToArcGISRaster(*inputFile, outputRaster, sdsName, xLowerLeftCorner, yLowerLeftCorner, cellSize*[, *noDataValue*[, *coordinateSystem*[, *buildPyramids*[, *overwriteExisting*]]]])
Extracts a specified Scientific Data Set (SDS) from an HDF file to an ArcGIS raster.

SDSToArcInfoASCIIGrid(*inputFile, outputFile, sdsName, xLowerLeftCorner, yLowerLeftCorner, cellSize*[, *noDataValue*[, *overwriteExisting*]])
Extracts a specified Scientific Data Set (SDS) from an HDF file to an ArcInfo ASCII Grid.

SDSToBinaryRaster(*inputFile, outputFile, sdsName*[, *overwriteExisting*])
Extracts a specified Scientific Data Set (SDS) from an HDF file to a binary raster.

Remarks

At the time this class was developed, two major versions of HDF were under continuing development: HDF, which was at release 4.2r1, and HDF5, which was at release 5-1.6.5. This class can process HDF files but not HDF5 files. It can also process HDF-EOS, a specialized version of HDF, but not HDF5-EOS.

HTML pages generated from XML metadata using XSL transforms

Using metdata to generate batch-processing versions of your method

```
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