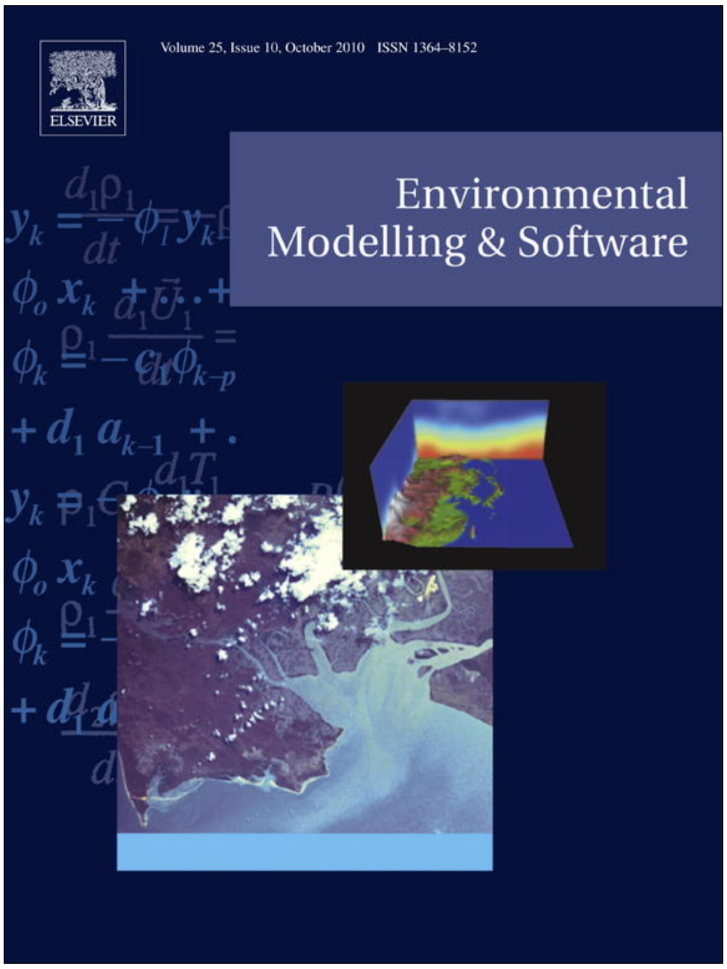
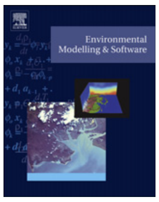

Volume 25, Issue 10, October 2010 ISSN 1364-8152
ELSEVIER
Environmental Modelling & Software

Environmental Modelling & Software 25 (2010) 1197–1207

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft



Marine Geospatial Ecology Tools: An integrated framework for ecological geoprocessing with ArcGIS, Python, R, MATLAB, and C++

Jason J. Roberts [a,*], Benjamin D. Best [a], Daniel C. Dunn [a], Eric A. Treml [b], Patrick N. Halpin [a]

[a] Marine Geospatial Ecology Laboratory, Nicholas School of the Environment, Duke University, Durham, NC 27708, USA
[b] The University of Queensland, School of Biological Sciences, St. Lucia, Queensland 4072, Australia

ARTICLE INFO

Article history:
Received 16 January 2009
Received in revised form
18 March 2010
Accepted 26 March 2010
Available online 24 April 2010

Keywords:
Marine ecology
Spatial ecology
Software integration
Interoperability
Informatics
Habitat modeling
Oceanography
GIS

ABSTRACT

With the arrival of GPS, satellite remote sensing, and personal computers, the last two decades have witnessed rapid advances in the field of spatially-explicit marine ecological modeling. But with this innovation has come complexity. To keep up, ecologists must master multiple specialized software packages, such as ArcGIS for display and manipulation of geospatial data, R for statistical analysis, and MATLAB for matrix processing. This requires a costly investment of time and energy learning computer programming, a high hurdle for many ecologists. To provide easier access to advanced analytic methods, we developed Marine Geospatial Ecology Tools (MGET), an extensible collection of powerful, easy-to-use, open-source geoprocessing tools that ecologists can invoke from ArcGIS without resorting to computer programming. Internally, MGET integrates Python, R, MATLAB, and C++, bringing the power of these specialized platforms to tool developers without requiring developers to orchestrate the interoperability between them.

In this paper, we describe MGET's software architecture and the tools in the collection. Next, we present an example application: a habitat model for Atlantic spotted dolphin (*Stenella frontalis*) that predicts dolphin presence using a statistical model fitted with oceanographic predictor variables. We conclude by discussing the lessons we learned engineering a highly integrated tool framework.

© 2010 Elsevier Ltd. All rights reserved.

Software availability

Name of software: Marine Geospatial Ecology Tools (MGET)
Developer: Duke University, Marine Geospatial Ecology Laboratory
Contact: mget-help@nicholas.duke.edu
Year first available: 2007
Hardware required: PC (2 GB RAM, 2 GHz CPU recommended)
Software required: Windows XP or later, Python 2.4 or later; ArcGIS 9.1 or later is optional but highly recommended
Program languages: Python, C++, C#, R, MATLAB
Program size: ~20 MB
Availability: Download from http://mgel.env.duke.edu/tools
Cost: Free, licensed under the GNU GPL

1. Introduction

The Duke University Marine Geospatial Ecology Laboratory specializes in the development of spatially-explicit ecological modeling techniques and the application of those methods in marine ecology studies and conservation projects. By publishing reusable and interoperable software tools in addition to journal articles, we hope to allow other researchers, managers, and conservation practitioners to repeat our analyses without reengineering them from scratch, to integrate them into larger scientific and management workflows, and ultimately to leverage them in an operational context. To maximize our pool of potential users, we target ecologists and analysts with moderate expertise in geographic information systems (GIS) and little experience in computer programming.

Developing tools for this community is hard. The community requires that tools be easy to install and operate, with graphical user interfaces (GUIs), ideally integrated with a GIS. For simple tools, such as a tool for building polygons for the regions traversed by drifting longline fishing gear (Dunn et al., 2008), we found we could satisfy these requirements by writing geoprocessing tools for ArcGIS using the Python programming language (ESRI, 2008; Python Software Foundation, 2008). ArcGIS is well known: an ongoing survey of nearly 40,000 GIS professionals found that ArcGIS is the dominant GIS platform, with 78% of respondents reporting that they used ArcGIS or related ESRI products and only

* Corresponding author. Tel.: +1 206 619 2736.
E-mail address: jason.roberts@duke.edu (J.J. Roberts).

1364-8152/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.envsoft.2010.03.029

27% reporting that they used the next most popular GIS product (GISJobs.com, 2008). Python is a modern, open-source language that has been integrated into ArcGIS. Developing Python-based geoprocessing tools for ArcGIS is easy: all tools share a common graphical user interface provided by ArcGIS, and developers must implement only the geospatial analysis tasks performed by the tool.

For more complicated tools, we found that ArcGIS and Python did not provide all of the analytic functionality we needed. For predictive modeling tools, such as a system for predicting marine animal habitats from oceanographic conditions (Best et al., 2007), or a tool for predicting hard bottom substrate from coarse-resolution bathymetry (Dunn and Halpin, 2009), we needed multivariate statistical modeling functions. The platform we preferred for this was R, a popular statistics programming language (R Development Core Team, 2008). For math-intensive modeling tools, such as a hydrodynamic simulation of the dispersal of larvae between coral reefs (Treml et al., 2008), we preferred MATLAB, a popular programming platform for numeric processing (MathWorks, 2008). Unfortunately, R and MATLAB proved unsuitable as user interfaces for our target community because they both require programming expertise to operate effectively.

To provide our target community with an acceptable user interface and our developers with sufficient analytic functionality, we concluded that we must integrate ArcGIS and Python with R and MATLAB. To avoid reengineering this integration on a tool-by-tool basis, we decided to rewrite them all under a common software framework and release them as a unified collection called Marine Geospatial Ecology Tools (MGET).

Besides providing engineering benefits, this approach benefitted our tool development process as well. Our development process essentially follows Argent's (2004) four-level process for developing environmental models, in which a model (or tool) is first developed by a researcher for a specific problem, then generalized and tested for a range of similar problems, then reengineered, repackaged, and documented for widespread operational use, and finally adopted by planners and policy makers as a reliable "black box". Most of our tools start out as rough prototypes developed for specific research projects, but when they have high potential utility, our goal is to take them to at least the third level of Argent's process. To successfully transition to the third level, in which a tool is of suitable quality to be used operationally, we have found it is almost always necessary to throw away the prototype and rewrite the tool from scratch. By developing a unified framework and release vehicle for all of our tools, we reduced both the temptation to release low quality prototypes and the effort required to rewrite them as high quality members of a consistent collection.

2. Software architecture of MGET

Although the ArcGIS/Python/R/MATLAB integration was a key requirement for MGET, it was not the only one. Here, we enumerate the other important requirements, present the architecture of MGET tools, and describe MGET's code-generation functionality, a key component of the architecture that facilitates the integration.

2.1. Key requirements

As mentioned, we wanted to give developers access to ArcGIS, Python, R, and MATLAB for implementing their tools. We also wanted to allow development in C++ to implement CPU-intensive tools, such as a multi-threaded version of the Cayula and Cornillon (1992) algorithm for identifying fronts in sea surface temperature images.

As mentioned, we wanted to expose MGET tools in ArcGIS as geoprocessing tools, allowing us to capitalize on ArcGIS's familiar user interface and its ModelBuilder graphical workflow system. To facilitate interoperability with other programming languages, we also wanted to expose each tool using the Microsoft Component Object Model (COM) (see Rogerson, 1997). Finally, we wanted the architecture to facilitate the exposure of tools to other workflow systems such as Kepler (Ludäscher et al., 2006) in the future.

Much of the functionality we required, such as functions for reading scientific file formats like NetCDF and HDF or functions for calculating biodiversity indices, was not available in the core Python, R, and MATLAB software, but in extension packages and toolboxes. In predecessors to MGET (Best, 2006a,b, 2007), we found that these extensions speeded development but complicated installation, frustrating our users. Users experienced many failures related to missing software, but often could not determine that this was the root cause of the problem. With MGET, we wanted each tool to check its software dependencies prior to executing and to report a clear error message and installation instructions when required software was missing. Also, we wanted the MGET installation program to automatically install as much required software as possible.

Finally, to minimize user costs, we wanted MGET to be a free, open-source project and to leverage free software whenever possible. Even though Windows ArcGIS users constituted our primary user community, we wanted MGET tools to be independent of Windows and ArcGIS whenever possible.

2.2. Tool architecture

MGET is a collection of *tools*. A *tool* is analogous to a synchronous subroutine: it accepts inputs, performs some processing, and produces outputs. While it is executing, it blocks the thread of execution of the calling program. We selected this simple construct because it facilitates interoperability: all common programming languages support it, ArcGIS geoprocessing tools must conform to it, and it is easily represented in workflow systems.

An MGET tool consists of a Python class method (van Rossum, 2001) that implements the tool's processing, and some metadata. The metadata specifies the method's software dependencies, documentation, and the data types and validation steps for its input and output parameters, and designates how it should be exposed for invocation by other programs. Using the metadata, MGET automatically checks dependencies, validates input parameter values, and implements interoperability plumbing, freeing the developer to focus his code on the scientific problem addressed by the tool.

Fig. 1 shows the invocation flow for a hypothetical tool called `MyTool`, implemented in the Python module `MyTool.py`. In this example, the developer designated `MyTool` for maximal interoperability. ArcGIS users invoke `MyTool` via a geoprocessing tool in the MGET ArcGIS toolbox. Python programs invoke it by importing the `MyTool.py` module and calling it directly. Early-bound COM clients, such as programs written in C, C++, C#, or FORTRAN, invoke it by compiling with an MGET COM type library (not shown) and calling through the `IMyTool` COM interface. Late-bound COM clients, such as programs written in Visual Basic, VBA, VBScript, Jscript, R, and MATLAB, invoke it through COM Automation, which operates through the `IDispatch` COM interface. To expose MyTool to both types of COM callers, MGET leverages the `win32com` module from the Python for Windows extensions (also known as pywin32) (Hammond et al., 2008).

When the tool is invoked through any of the mechanisms described above, MGET checks dependencies, validates input parameter values, and executes the Python code written by the tool developer. The developer's Python code can call functions from any of the six sources: Python, C/C++, MATLAB, R, ArcGIS, and COM.

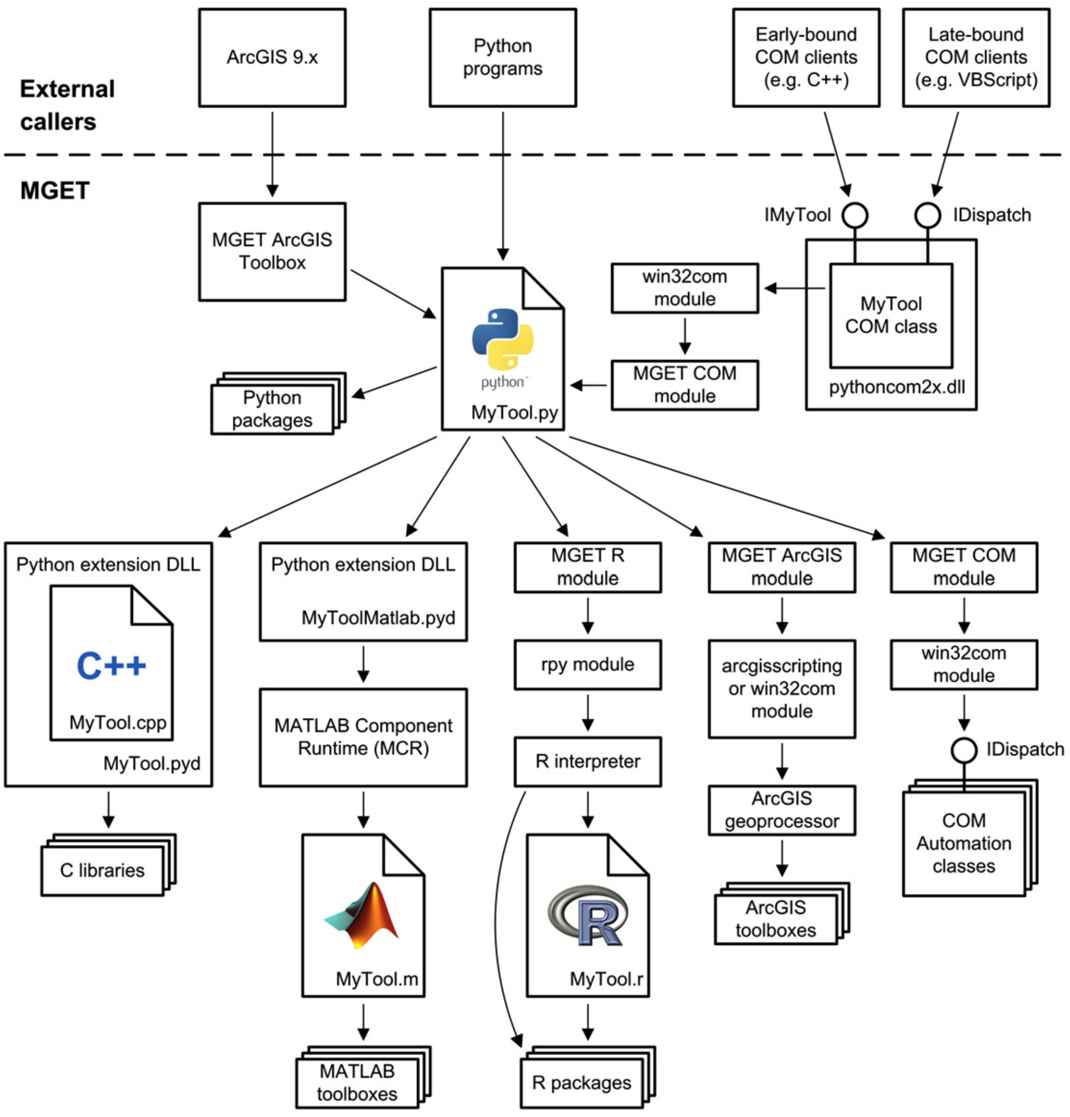


Fig. 1. MGET's software architecture. Boxes show major software modules and arrows show invocation flow between modules. The four documents labeled MyTool represent source code files for a hypothetical MGET tool that is implemented in Python, C++, MATLAB, and R.

MGET provides substantial infrastructure to facilitate this integration, allowing the developer to quickly access his preferred platforms. If the developer chooses to implement his tool in a language other than Python, his Python code can be as short as two lines, the minimum needed to access another language through MGET's infrastructure. If he needs to access multiple languages, he can orchestrate the overall flow from Python.

We selected Python as the "core" language for development of MGET tools in part because it provides excellent facilities for interoperating modules implemented in other programming languages (see, for example, Schmitz et al., 2009). In the example in Fig. 1, the developer authored functions in C++, MATLAB, and R, in the files `MyTool.cpp`, `MyTool.m`, and `MyTool.r`, respectively, in addition to the Python code in `MyTool.py`. Python supports compilation of C/C++ functions into Python extension modules (van Rossum, 2008), allowing the functions to be called just like those written in Python itself. MGET performs this compilation automatically. MATLAB functions work similarly: MGET wraps them with C++ and compiles the wrappers into Python extension modules, which invoke the `.m` files through the MATLAB Component Runtime (MCR), a free redistributable from the publisher of MATLAB. To provide access to R, MGET leverages the `rpy` module (Moreira and Warnes, 2008), which wraps the R interpreter in a Python interface.

2.3. Code generation

It has been shown that developers can be more productive and introduce fewer errors when they can leverage programs that generate code for them (Weigert and Dietz, 2003; Kieburtz et al., 1996). MGET's code-generation capabilities relieve tool developers of much of the coding needed for the integration and interoperability described above. After a developer writes a tool and its metadata, he executes a build script. As illustrated in Fig. 2, the build script generates the files needed for integration and interoperability, as well as the tool's documentation, and packages it all into a setup program. When the user runs the setup program, it installs the files on the user's machine, registers the MGET toolbox with ArcGIS, and registers the MGET COM classes with Windows.

3. Tools in the MGET collection

At the time of this writing, MGET included over 180 tools, grouped into seven categories: Conversion, Data Management,

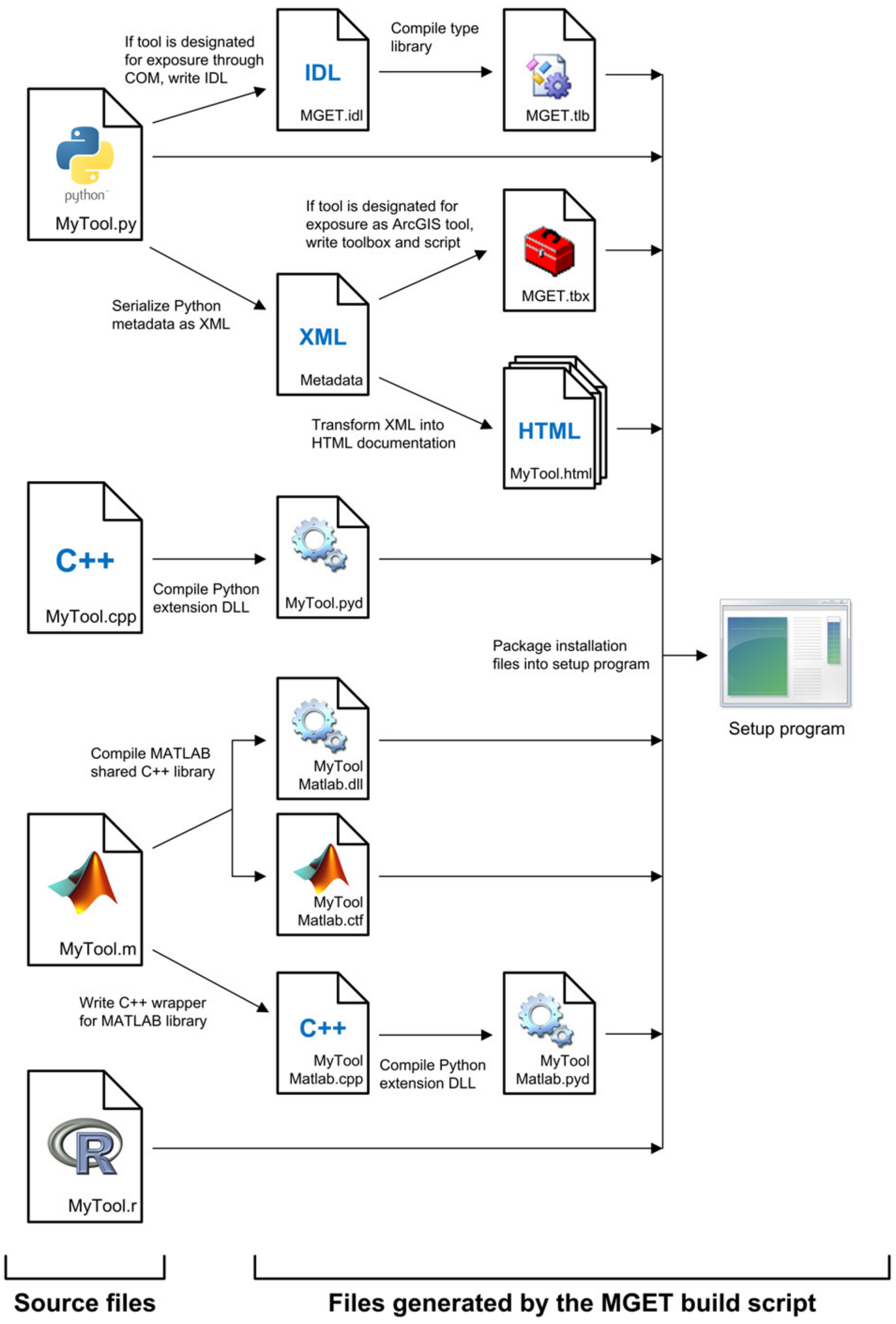

If tool is designated for exposure through COM, write IDL
IDL
MGET.idl
Compile type library
MGET.tlb
python
MyTool.py
If tool is designated for exposure as ArcGIS tool, write toolbox and script
MGET.tbx
Serialize Python metadata as XML
XML
Metadata
Transform XML into HTML documentation
HTML
MyTool.html
C++
MyTool.cpp
Compile Python extension DLL
MyTool.pyd
Package installation files into setup program
Setup program
Compile MATLAB shared C++ library
MyTool Matlab.dll
MyTool.m
MyTool Matlab.ctf
Write C++ wrapper for MATLAB library
C++
MyTool Matlab.cpp
Compile Python extension DLL
MyTool Matlab.pyd
R
MyTool.r
Source files
Files generated by the MGET build script

Fig. 2. Files automatically generated by the MGET build script for a hypothetical MGET tool called MyTool that is implemented in Python, C++, MATLAB, and R. After generating the files, the build script packages them into the MGET setup program.

Spatial Analysis, Oceanographic Analysis, Connectivity Analysis, Statistics, and Data Products. In this section, we briefly summarize each category and highlight some of the tools they contain.

The Conversion tools convert geospatial data from one format to another, allowing ArcGIS users to convert oceanographic data from popular formats such as HDF, NetCDF, and binary flat files to formats that ArcGIS can read. Also included is a tool for downloading data from an OPeNDAP server into ArcGIS format.

The Data Management tools perform common data manipulation tasks such as finding, copying, and deleting files, directories, and other data, extracting headers from geospatial data files, and executing programs and database commands. While not directly relevant to ecology, these tools fill important gaps in the collection of general-purpose tools that come with ArcGIS.

The Spatial Analysis tools analyze and manipulate vector and raster data and are useful in a variety of scenarios. For example, MGET's sampling tools intersect points with rasters or polygons and write the values of the rasters' cells or the polygons' attributes to fields of the points. By matching points to rasters by date, these tools can extract the values of oceanographic parameters such as sea surface temperature from remotely-sensed time-series data. The extracted values can then be analyzed with MGET's Create Fishnet for Points tool, which produces grids that statistically summarize the points contained by each cell.

The Oceanographic Analysis tools execute analytic algorithms on oceanographic data sets to derive data useful for marine ecological modeling and other research. For example, oceanographic fronts have long been known to aggregate and attract marine organisms (Mann and Lazier, 1996). MGET's Cayula–Cornillon Fronts tool identifies fronts in sea surface temperature images using Cayula and Cornillon's(1992) single image edge detection algorithm. The output can be used in combination with biological surveys to test whether organisms are associated with fronts.

The Connectivity Analysis tools analyze the connectivity of marine ecosystems. Currently, MGET includes only one such tool. Using the method developed by Treml et al. (2008), this tool simulates the dispersal of coral larvae from reefs by ocean currents using an Eulerian advection–diffusion algorithm implemented in MATLAB and outputs a graph structure representing the connections between reefs. Researchers can then identify reefs of biological or conservation interest by analyzing the graph structure to find those that are sources and sinks of larvae, are hydrodynamically isolated from others, are stepping stones in the connectivity network, and so on.

The Statistics tools extend the statistical analysis and modeling capabilities of R to ArcGIS users. From ArcGIS, users can easily perform tasks like plotting density histograms and scatterplots, fitting and evaluating statistical models, and executing arbitrary R statements from the ArcGIS ModelBuilder. Section 4 below gives an in-depth example of some of these tools.

The Data Products tools are Conversion, Data Management, and analysis tools that are customized to specific widely-used oceanographic data sets. Many of these are essentially pre-parameterized, easier-to-use versions of the general-purpose tools described above. For example, the Cayula–Cornillon Fronts in CoastWatch Image tool detects fronts in National Oceanographic and Atmospheric Administration (NOAA) CoastWatch Advanced Very High Resolution Radiometer (AVHRR) SST images by integrating the NOAA CoastWatch Software Library and Utilities (Hollemans, 2008) with a customized version of the MGET front detection tool described above.

4. Example application: predictive habitat modeling using ArcGIS

The introduction of advanced GIS software and statistical modeling techniques has spawned a burgeoning assortment of spatially-explicit statistical approaches to ecology. One such approach is predictive habitat modeling, in which the investigator attempts to relate spatiotemporal observations of a species to environmental conditions using statistics or other quantitative techniques and then predicts the distribution of the species across a region, timeframe, and range of environmental conditions. Many variations on this approach have been presented in the ecology literature (for a review, see Guisan and Zimmermann, 2000). In this paper, we illustrate how to build a habitat model using point observations of the presence or absence of a species using a binomial generalized additive model (GAM, Hastie and Tibshirani, 1990) that estimates the probability of species presence using oceanographic conditions as predictor variables. We assess the model's predictive performance using receiver operating characteristic (ROC) analysis (Metz, 1978) and then predict maps of the species' distribution by evaluating the model across every pixel in a stack of satellite images. This approach has been used to model the habitat of marine mammals (Redfern et al., 2006), seabirds (Vilchis et al., 2006), and fish (Valavanis et al., 2008), as well as many terrestrial fauna and flora (Guisan and Zimmermann, 2000).

Our goal is to briefly illustrate how to implement this modeling technique using tools present in MGET. To keep this paper short, we do not provide a complete description of the modeling technique, review all of the factors that must be considered to apply it successfully, or specify the complete set of parameters provided to the MGET tools. Also, we skipped some steps that are recommended to arrive at a robust model (such as the separation of the input data set into model calibration and evaluation sets). For a detailed discussion of the technique and recommended modeling steps, please consult the references provided above.

4.1. Data

For this example, we used sightings of Atlantic spotted dolphin (*Stenella frontalis*) recorded during a ship-based marine mammal abundance survey conducted by the NOAA Southeast Fisheries Science Center (SEFSC) in August and September of 1999 along the east coast of the United States (Roden, 1999). Following standard NOAA protocols for ship-based line-transect surveys, NOAA placed trained observers on the vessel's flying bridge and equipped them with high power binoculars. As the ship progressed along a planned survey route, the observers noted the times, locations, and species of marine mammals they sighted.

For oceanographic variables, we used bathymetry, sea surface temperature (SST), and chlorophyll density, all from remotely-sensed data sets. We chose bathymetry because NOAA's most recent stock assessment for the Western North Atlantic stock of *S. frontalis* described its distribution in terms of depth: "Atlantic spotted dolphins regularly occur in the inshore waters south of Chesapeake Bay and near the continental shelf edge and continental slope waters north of this region" (Waring et al., 2007). We chose SST because marine mammals are believed to be adapted to specific temperature regimes (Stevick et al., 2002). We chose chlorophyll density because the presence of algae may indicate the presence of herbivorous fish and other species preyed upon by dolphins. For bathymetry data, we used ETOPO2v2 (NGDC, 2006). For SST, we used daytime monthly images for August and September 1999 from the AVHRR Pathfinder SST data set (Casey and Evans, 2008). For chlorophyll, we used monthly images from the SeaWiFS Level 3 data set (Feldman and McClain, 2008).

4.2. Methods

4.2.1. Data acquisition and preparation

From the Ocean Biogeographic Information System–Spatial Ecological Analysis of Megavertebrate Populations (OBIS–SEAMAP) web portal (Halpin et al., 2006), we downloaded the "SEFSC Atlantic surveys 1999" data set, consisting of two shapefiles: one of lines representing the ship's route while the marine mammal observers were "on effort," and one of points representing animals observed along the route. We extracted the 75 points where Atlantic spotted dolphins was observed (Fig. 3) and used these as presence points in the habitat model. For absence points, we generated 525 points along the effort lines, distributed randomly, with the constraint that they be no closer than 20 km to any presence point. (While this method for generating absence points was very easy to implement in ArcGIS, alternative, more difficult methods may yield more accurate models; see Redfern et al. (2006) for further discussion.)

Next, we downloaded the three oceanographic data sets and converted them with an MGET tool from their original formats to raster formats compatible with ArcGIS. To keep this example simple, we created a single raster for each predictor variable, representing the average value of the variable for the duration of the NOAA cruise (see Fig. 3). For bathymetry, a static variable, we just

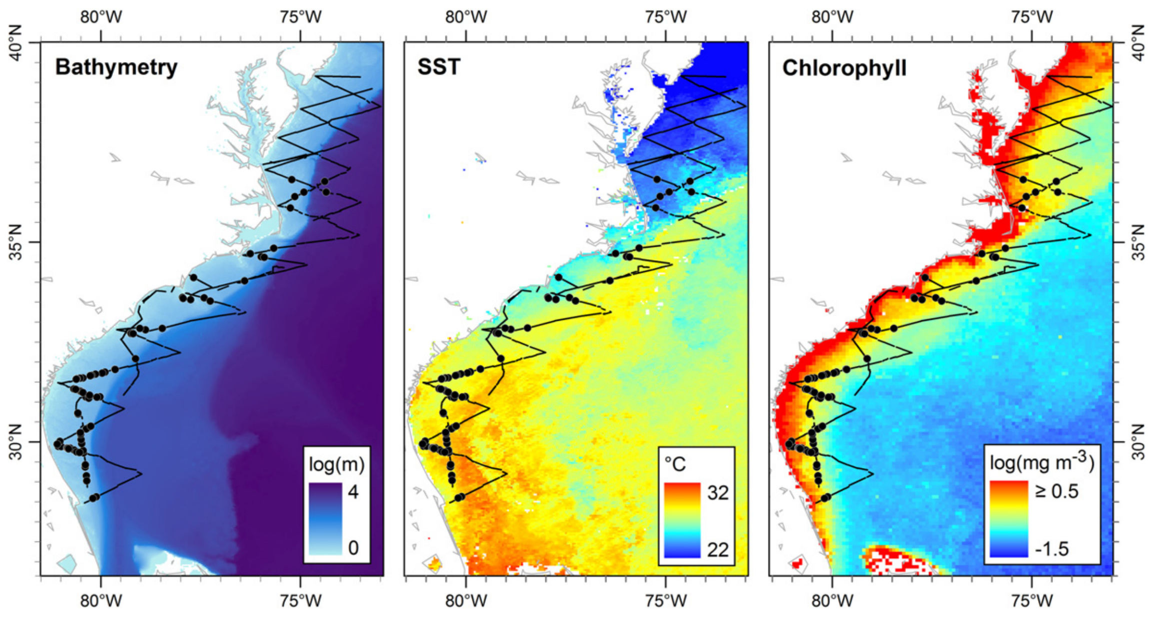


Fig. 3. Data used in the *Stenella frontalis* habitat model. The three panels show the three oceanographic predictor variables used in the model, averaged over the months of August and September 1999. The heavy line shows the track of the NOAA survey vessel while observers were on effort. The heavy dots show locations where observers spotted *Stenella frontalis*.

used the ETOPO2v2 raster. For SST and chlorophyll, we created 2-month mean rasters by averaging the August and September rasters. (By averaging the dynamic oceanographic variables in this way, we created a model that is simple but that has low temporal resolution. Higher resolution could be achieved by matching presence and absence points to weekly or daily SST and chlorophyll images.)

4.2.2. Model fitting, evaluation, and habitat prediction

After preparing the presence and absence points and predictor variable rasters, we constructed an ArcGIS geoprocessing model (Fig. 4) to perform the remaining steps of the analysis using tools from MGET. First, we sampled the values of the predictor variable rasters at the presence and absence points. Next, we constructed density histograms (Fig. 5) showing the distribution of each variable for the presence and absence points, to determine if the presence points could be statistically separated from the absence points using values of the variable, and to gain an understanding of the variable's distribution. Armed with this information, we fitted the following binomial GAM, expressed here in R formula syntax, using an MGET tool that invoked functions in the R mgcv package (Wood and Augustin, 2002):

```
Presence ~ log10(Bathymetry) + s(SST) + s(log10
(Chlorophyll))                                        (1)
```

This model predicts dolphin presence as an additive multiple regression of bathymetry, SST, and chlorophyll density with a logarithm fit to bathymetry, a smoothed spline fit to SST, and a smoothed spline fit to the logarithm of chlorophyll density (see Section 4.3 below for discussion of these terms). To characterize the shape of the influence of each predictor on the probability of presence, we plotted each term against its fitted range of predictor values (Fig. 6).

We assessed the model's performance by plotting an ROC curve (Fig. 7) using an MGET tool that invoked functions in the R `rocr` package (Sing et al., 2005). ROC analysis assists the modeler in selecting a *cutoff value* for classifying the model response, a continuous probability ranging from 0 to 1, into a binary value of either 0 or 1 (i.e. species absent or present). Response values less than the cutoff value are classified as 0, the rest are classified as 1. An ROC curve compares the rates of true positive predictions and false positive predictions obtained from the model for all possible cutoff values. The MGET tool, using the `rocr` package, identified the cutoff value on the ROC curve that was closest to the point of perfect classification (the upper left corner of the plot, where the true positive rate is 1 and false positive rate is 0), and calculated a variety of model performance statistics for this value.

Using the fitted model, we created a map showing the probability of presence predicted from the oceanographic rasters (Fig. 8). We also created corresponding maps showing the estimated standard errors of the prediction and the binary presence/absence classification, produced by classifying the probability of presence using the cutoff value selected during the ROC analysis. The binary classification represents our prediction of Atlantic spotted dolphin habitat for this region and time period.

4.3. Results

All three predictor variables appeared to be good candidates for separating the presence points from the absence points. Because bathymetry and chlorophyll were left-skewed, we log-transformed them (log base 10) to produce a more even distribution for these plots and the modeling procedure. For bathymetry, nearly all of the presence points were clustered between 10 and 100 m while the absence points were fairly evenly distributed (Fig. 5, left). In the GAM, we fit a line to the log-transformed values of bathymetry. For SST, most of the presence values were clustered between 27 and 30 °C, but some occurred at lower temperatures (Fig. 5, center). The absence points displayed a bimodal distribution. Due to this complexity, we fitted a spline smooth to SST in the GAM. For chlorophyll, the presence points spiked around 0.5 mg m^{-3} (−0.3 on the log scale) while the absence points spiked at a lower value and tailed off gradually to the right, encompassing the entire range

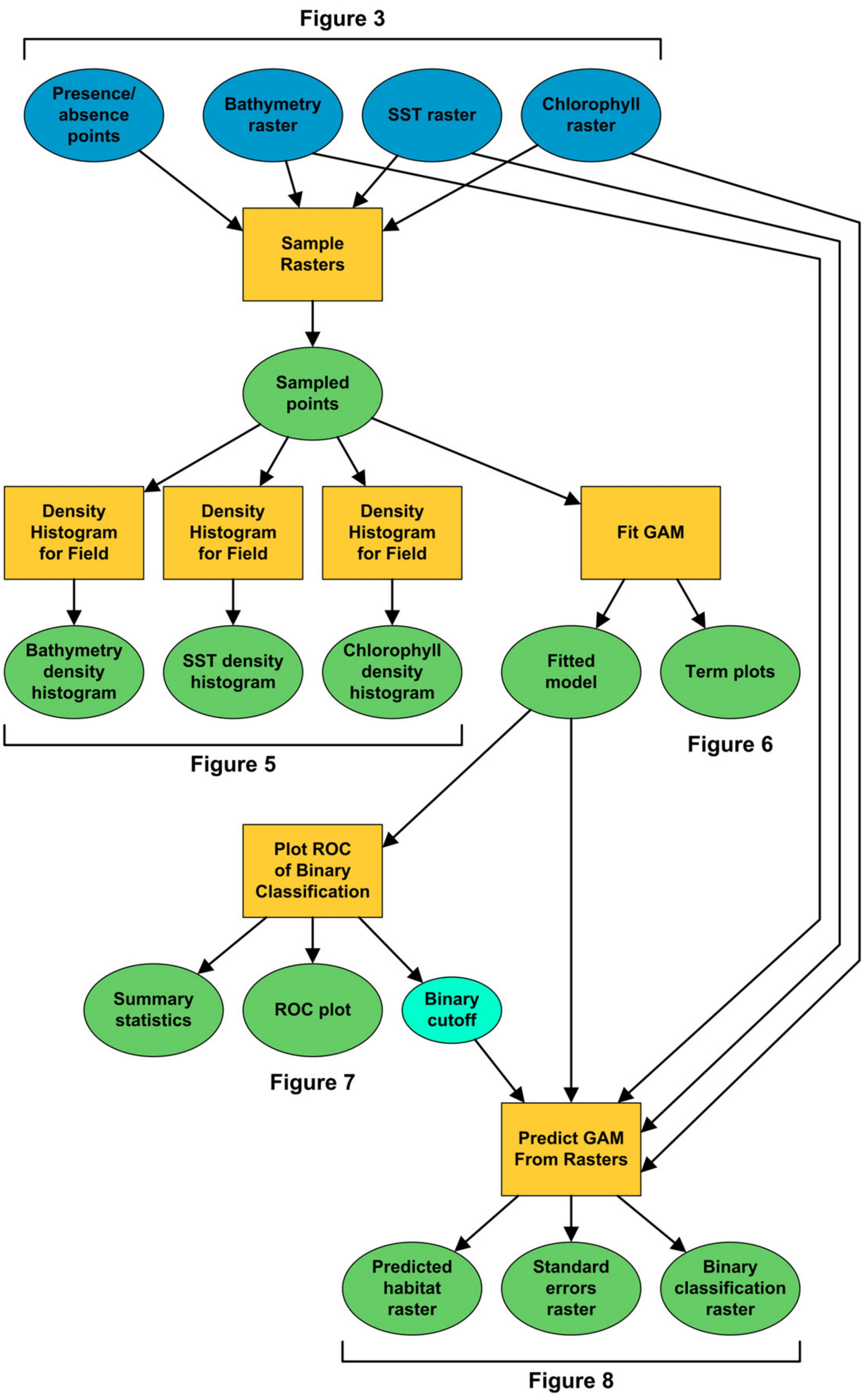


Fig. 4. ArcGIS geoprocessing model showing the analysis workflow for the *Stenella frontalis* habitat model. The four ovals at the top represent the inputs to the analysis. The boxes represent MGET tools. The remaining ovals represent outputs produced by the tools. The inputs and some outputs are presented in other figures, as labeled in the workflow.

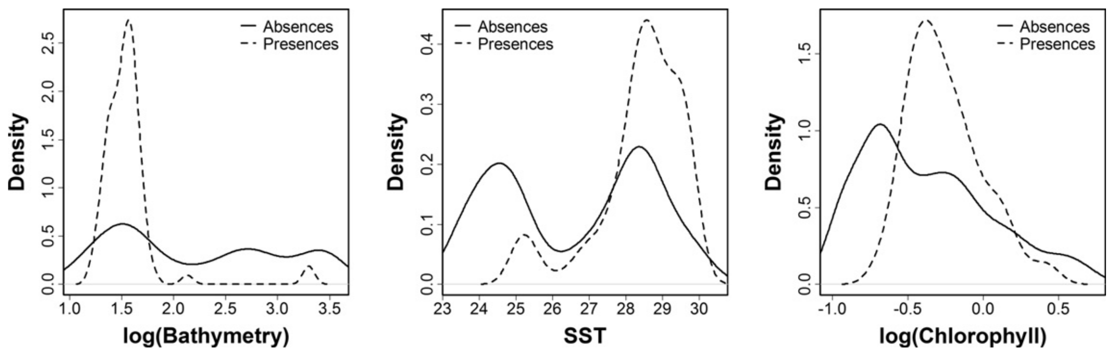


Fig. 5. Density histograms for the *Stenella frontalis* habitat model. Each panel shows the distribution of a particular predictor variable for the points where dolphins were present (solid line) and the points where they were absent (dashed line).

of presence values (Fig. 5, right). This suggested that dolphin presence was most probable between a range of chlorophyll values. To capture this correlation in the GAM, we fitted a spline smooth to the log-transformed values of chlorophyll.

The GAM explained 56% of the null deviance and had an unbiased risk estimator (UBRE) score of -0.63. The estimated p-values for all model terms were significant ($p < 0.02$). Bathymetry showed a negative correlation with dolphin presence that grew weaker at shallow depths (Fig. 6, left). SST showed a negative correlation at the coldest values, almost no correlation at mid-range values, and a positive correlation at the warmest values (Fig. 6, center). At extreme warm or cold values the correlation became uncertain, due to the sparse number of points available for those temperatures. Chlorophyll showed a negative correlation at the extremes, and a positive correlation in between (Fig. 6, right). As with SST, correlation became uncertain at the extremes due to insufficient data.

A commonly-used summary statistic for ROC analysis is the area under the ROC curve (AUC). A perfect model has a true positive rate of 1 and false positive rate of 0, yielding an AUC of 1. A model that gives random predictions has the same true positive rate and false positive rate, yielding an AUC of 0.5. Our model yielded an AUC of 0.959 (Fig. 7), indicating strong predictive power.

The highest predicted probability of dolphin presence occurred in the southern part of the surveyed region, on the continental shelf (Fig. 8, left). This area was shallow and warm, and had a moderate density of chlorophyll. The probability remained above zero northward along the shelf but dropped to zero north of Chesapeake Bay, as water temperatures dropped. Off the shelf, the probability approached zero everywhere except east of the Outer Banks of North Carolina, above the northern edge of the Gulf Stream current, where temperatures and chlorophyll density were high enough to slightly outweigh the negative effect of deep water.

Regions of high standard error occurred close to shore and along the edge of the continental shelf, where both presences and absences were observed but the sparse number of data points provided weak statistical coverage of the oceanographic conditions that occurred there (Fig. 8, center).

The binary classification map (Fig. 8, right) shows the regions where the predicted probability of presence exceeded the cutoff value selected by the ROC analysis (0.155). The true positive rate, or sensitivity, of the model for this value was 0.920, and the false positive rate, or 1 minus the specificity, was 0.128.

5. Lessons learned

The principal challenge we faced in creating MGET was an engineering problem, not a science problem: how do we build a suite of geospatial ecology tools that are accessible to non-programmers, that are modular and highly interoperable, and that are cheap to build and deploy? Our solution was to integrate a number of different programming platforms and software packages, both commercial and free, into a unified framework, and develop our tools on top of the framework. Here, we discuss some of the engineering challenges we faced in attempting such a large integration project.

Early in the design process, we had to decide how we would architect MGET tools so they could be invoked from ArcGIS. ArcGIS provides several alternatives, such as the Python-based scripting approach that we selected, and the ArcObjects approach, which

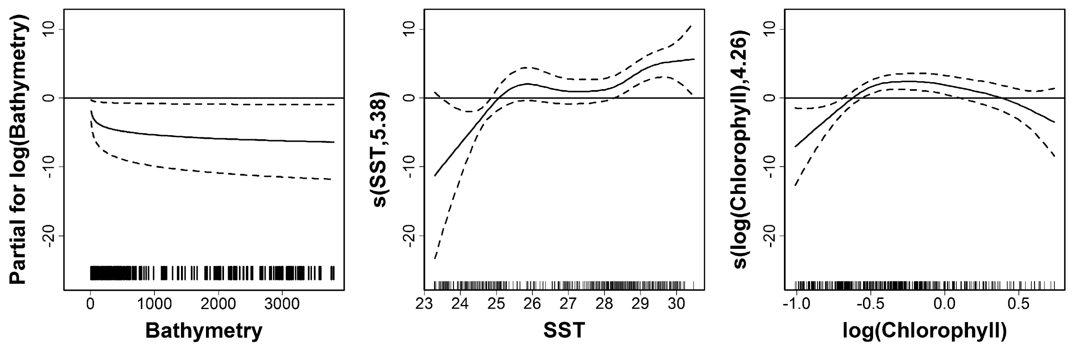


Fig. 6. Partial residual plots for the GAM terms in the *Stenella frontalis* habitat model. Each panel plots the value of a model predictor variable (x axis) against the partial residuals (y axis), i.e. the residuals after removing the effects of the other predictors. The dashed lines show the estimated 95% confidence limits. In the model, bathymetry was fitted logarithmically, while SST and the log of chlorophyll were fitted with a spline smooth. The plots use the same y axis scale, so that the relative influence of predictors may be easily compared. For the two spline plots, the y axis label indicates the estimated degrees of freedom.

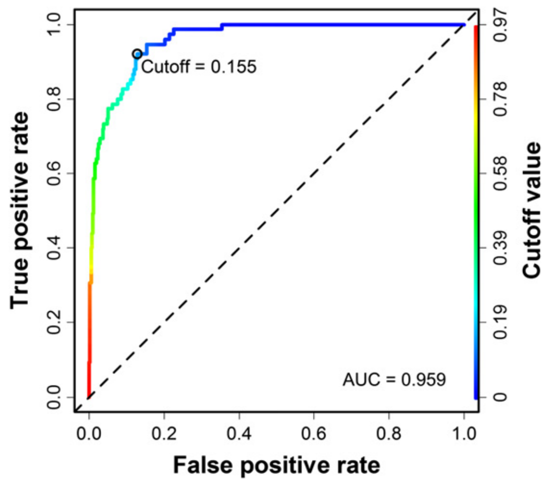


Fig. 7. Receiver operating characteristic (ROC) plot for the *Stenella frontalis* habitat model. The ROC curve is labeled with the cutoff value selected for classifying the continuous probability of dolphin presence into a binary value. The plot also specifies the area under the ROC curve (AUC).

involves writing programs in a compiled language such as C# on top of the set of COM components that ArcGIS itself is built upon. We selected the Python scripting approach because it allowed faster development, had a lower learning curve, and was based on open-source technologies. In particular, we liked that ArcGIS provided the entire user interface (UI) required for running Python-based tools, freeing us from having to write any UI code. But by selecting this approach, we limited ourselves to the UI offered by ArcGIS. This UI provides only a basic collection of graphical user input elements, such as text boxes and drop-down menus. Consequently, MGET tools cannot leverage sophisticated input elements, such as clickable maps, tables of editable cells, and so on.

We encountered many problems trying to integrate so many programming platforms. Platforms that initially appeared to be fully compatible usually proved to be subtly incompatible. Often, not all of the data types available from one platform would be available for the other. For example, Python natively offers only one floating point numeric type (IEEE 754 double precision), while MATLAB offers two (IEEE 754 single and double precision). Although we could rely on previously-developed interoperability packages such as `win32com` and `rpy` to translate data types, these packages never provided all of the translations we required, forcing us to write additional translation code. The worst case was date and time data types, which were represented differently in each programming language and never handled by the interoperability packages.

Whenever we took a dependency on software written by someone else, we always had to set aside time to fix compatibility problems that surfaced when they released a new version of their software. For example, when ArcGIS 9.3 was released, it included a feature allowing Python-based tools such as MGET to run as part of the ArcCatalog or ArcMap processes, rather than as a separate process as was done in previous versions of ArcGIS. This feature was intended to be a performance optimization and was enabled by default. But ArcGIS 9.3 exhibited a subtle compatibility problem with the then-current version of MATLAB (version 2007b): both programs compiled a third-party library called `xerces-c_2_7.dll`, but because they used different compilers, their versions of the library were incompatible with each other. As a result, MGET tools that invoked MATLAB would not work with ArcGIS 9.3 until we forced ArcGIS to run them as “out of process” tools.

Dependencies on other software proved risky because we could never be sure of the level of support we would receive from their developers or how well the software would be maintained in the future. This risk existed both for large commercial package such as ArcGIS and for small, free, open-source projects. In the case of ArcGIS, we found that ESRI was unresponsive to our bug reports, requiring us to develop many workarounds for bugs in ArcGIS. For the open-source projects, the level of support and maintenance varied according to the interest and availability of the original author. In one case, the `rpy` package, the original developer ceased participation in the project. Because `rpy` must be rebuilt whenever a new version of R is released, we had to start building our own

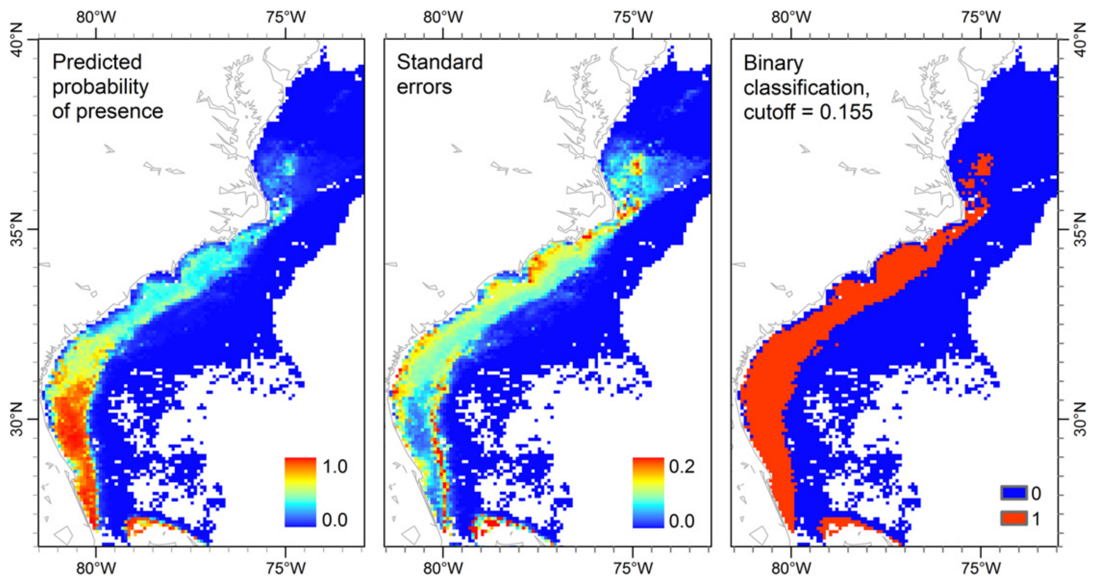

Predicted probability of presence
Standard errors
Binary classification, cutoff = 0.155
80°W
75°W
40°N
35°N
30°N
1.0
0.0
0.2
0.0
0
1

Fig. 8. Predictions for the *Stenella frontalis* habitat model. The left plot shows the probability of *Stenella frontalis* presence predicted from the oceanographic rasters (Fig. 3) by the GAM. The center plot shows the estimated standard errors for the prediction. The right plot shows a binary classification of the left plot using the cutoff value selected by the ROC analysis. White pixels represent areas where no prediction was performed because the values of one or more oceanographic rasters were outside the ranges of values used to fit the model.

private copy of rpy to ensure that MGET would be compatible with future releases of R. (Recently, another developer released a new package called rpy2, but it is not fully compatible with the original rpy, so we must rewrite portions of MGET before we can use it.)

Each time we took a dependency on another package, we had to write additional code for detecting whether the package was installed and for installing it if it was missing. This code was usually tedious and time consuming, although some programming platforms provided built-in infrastructure that made it fairly easy. For example, most R packages are distributed through the Comprehensive R Archive Network (CRAN) and can be downloaded and installed by simply calling a few R functions. A similar infrastructure exists for Python, but it is not as widely adopted by package developers. To work around this, we had to incorporate private copies of Python packages directly into MGET, allowing them to be installed by MGET's own installation program.

A last notable problem we encountered was the difficulty of debugging MGET tools that integrated multiple programming languages. While each language provides interactive debuggers that can step through each line of code, one at a time, most of these require you to execute the code within the debugger itself. We could rarely use these to debug MGET tools, which execute in a considerably more complicated manner. For example, a typical MGET tool that performs statistical analysis begins execution in an ArcGIS process. ArcGIS loads the Python interpreter and invokes MGET Python code, which loads the rpy module. rpy, written in C, loads the R interpreter, which finally invokes the MGET tool's R code. As far as we know, in this situation, there is no easy way to step through the R code with R's interactive debugger. Our solution to this problem was two-fold. First, we instrumented MGET with extensive logging that chronicles the tool's execution in detail. Second, we implemented exception handlers for each programming environment that catch unhandled exceptions and report detailed information about the problem. While not as effective as interactive debugging, this solution provides us with enough information to resolve many bugs. It can also be activated by MGET users so they can send us a log whenever a problem occurs, allowing us to diagnose the situation without accessing their machines.

6. Conclusion

In this paper, we presented Marine Geospatial Ecology Tools (MGET), a collection of modular software tools designed for ecologists who are familiar with GIS but have little experience with computer programming. By integrating ArcGIS, Python, R, MATLAB, and C++ through interoperability modules and code generation, MGET allows tool developers to easily access the power of several popular scientific programming platforms without having to write tedious integration code. At the time of this writing, MGET contained over 180 tools addressing problems such as converting geospatial data from one format to another, identifying ecologically relevant features in oceanographic images, modeling hydrodynamic connectivity of coral reefs, sampling time-series raster data, and building and evaluating predictive habitat models. Future releases of MGET will include additional connectivity and oceanographic analysis tools, as well as tools for analyzing fisheries and animal movements.

To illustrate the use of several MGET tools in an analytic workflow, we developed a presence/absence habitat model for Atlantic spotted dolphin (*S. frontalis*) sightings by sampling oceanographic images, fitting a generalized additive model to the sampled values, and predicting a map of the probability of dolphin presence by processing the images through the fitted model. This example showed how an ecologist can use MGET to accomplish a common multivariate statistical modeling scenario from ArcGIS without doing any statistical programming, while, behind the scenes, the MGET tools leverage best-of-breed statistical functions available in R.

Finally, we reviewed our experience tackling the engineering challenge of integrating ArcGIS, Python, R, MATLAB, and C++ into a tool development framework. We learned that we could save considerable development time by leveraging interoperability packages developed by others, but that these dependencies sometimes came with hidden costs, such as unanticipated coding needed to address subtle compatibility problems, and risks, such as poor support from the package developer. Also, we found it was challenging to debug tools developed with the framework because the flow of execution was usually too complicated to allow traditional interactive debuggers to be used. Others that attempt projects with this level of integration should beware of these problems.

Acknowledgements

We thank Dave Ullman and Stephanie Hansen for providing example code for detecting SST fronts and geostrophic eddies. Several MGET tools are based on those examples. We thank Michelle Sims for providing example code and suggestions for improving several statistical tools. We thank four anonymous reviewers for providing suggestions that improved this manuscript. The development of MGET was made possible by a grant from the David and Lucile Packard Foundation, with additional support from NASA. Finally, MGET was built atop a lot of software developed by others, often for free. We acknowledge the generous contributions of these developers, too numerous to list here, on the MGET web site.

References

Argent, R.M., 2004. An overview of model integration for environmental applications—components, frameworks and semantics. Environmental Modelling & Software 19, 219–234.

Best, B.D., 2006a. ArcRstats: Multivariate Habitat Prediction Using ArcGIS and the Open-source R Statistical Package. <http://mgel.env.duke.edu/tools>.

Best, B.D., 2006b. ConnMod: Connectivity Modeling Toolbox. <http://mgel.env.duke.edu/tools>.

Best, B.D., 2007. Geospatial Modeling of Habitat and Connectivity. M.Sc. thesis. Duke University, 67 pp.

Best, B.D., Halpin, P.N., Fujioka, E., Read, A.J., Qian, S.S., Hazen, L.J., Schick, R.S., 2007. Geospatial web services within a scientific workflow: predicting marine mammal habitats in a dynamic environment. Ecological Informatics 2 (3), 210–223.

Casey, K.S., Evans, R., 2008. Global AVHRR 4 km SST for 1985–2005, Pathfinder v5.0, NODC/RSMAS. NOAA National Oceanographic Data Center. <http://pathfinder.nodc.noaa.gov> (accessed November 2008).

Cayula, J.-F., Cornillon, P., 1992. Edge detection algorithm for SST images. Journal of Atmospheric and Oceanic Technology 9 (1), 67–80.

Dunn, D.C., Halpin, P.N., 2009. Filling a marine spatial planning data gap: rugosity as a mesoscale proxy for hard-bottom habitat. Marine Ecology Progress Series 377, 1–11.

Dunn, D.C., Kot, C.Y., Halpin, P.N., 2008. A comparison of methods to spatially represent pelagic longline fishing effort in catch and bycatch studies. Fisheries Research 92, 268–276.

ESRI, 2008. ArcGIS – A Complete Integrated System. Environmental Systems Research Institute, Inc., Redlands, California. <http://esri.com/arcgis>.

Feldman, G.C., McClain, C.R., 2008. Ocean Color Web, SeaWiFS Reprocessing 5.2. In: Kuring, N., Bailey, S.W. (Eds.). NASA Goddard Space Flight Center. <http://oceancolor.gsfc.nasa.gov/> (accessed November 2008).

GISJobs.com, 2008. Salary Survey. <http://www.gisjobs.com/survey> (accessed October 2008).

Guisan, A., Zimmermann, N.E., 2000. Predictive habitat distribution models in ecology. Ecological Modelling 135, 147–186.

Halpin, P.N., Read, A.J., Best, B.D., Hyrenbach, K.D., Fujioka, E., Coyne, M.S., Crowder, L.B., Freeman, S.A., Spoerri, C., 2006. OBIS–SEAMAP: developing a biogeographic research data commons for the ecological studies of marine mammals, seabirds, and sea turtles. Marine Ecology Progress Series 316, 239–246. <http://seamap.env.duke.edu>.

Hammond, M., da Silva, S., Coombs, J.R., Cole, V., Immisch, L., Upole, R., Heller, T., Golden, T., Mick, T., 2008. Python for Windows Extensions. <http://sourceforge.net/projects/pywin32/>.

Hastie, T.J., Tibshirani, R.J., 1990. Generalized Additive Models. Chapman and Hall/CRC, Boca Raton, FL.

Hollemans, P., 2008. CoastWatch Software Library and Utilities v3.2.2. NOAA National Environmental Satellite Data Information Service. <http://coastwatch.noaa.gov/cwn/cw_software.html>.

Kieburtz, R.B., McKinney, L., Bell, J.M., Hook, J., Kotov, A., Lewis, J., Oliva, D.P., Sheard, T., Smith, I., Walton, L., 1996. A software engineering experiment in software component generation. In: Proceedings of the 18th International Conference on Software Engineering (ICSE '96). IEEE Computer Society Press, Berlin, Germany.

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger-Frank, E., Jones, M., Lee, E., Tao, J., Zhao, Y., 2006. Scientific workflow management and the Kepler system. Concurrency and Computation: Practice & Experience 18 (10), 1039–1065.

Mann, K.H., Lazier, J.R.N., 1996. Dynamics of Marine Ecosystems: Biological–Physical Interactions in the Oceans, second ed. Blackwell Science, Cambridge, Massachusetts.

MathWorks, 2008. MATLAB – The Language of Technical Computing. The MathWorks, Inc., Natick, Massachusetts. <http://mathworks.com/matlab>.

Metz, C.E., 1978. Basic principles of ROC analysis. Seminars in Nuclear Medicine 8, 283–298.

Moreira, W., Warnes, G., 2008. RPy: A Simple and Efficient Access to R from PYTHON. <http://rpy.sourceforge.net/>.

NGDC, 2006. 2-Minute Gridded Global Relief Data (ETOPO2v2). NOAA National Geophysical Data Center. <http://www.ngdc.noaa.gov/mgg/fliers/06mgg01.html> (accessed November 2008).

Python Software Foundation, 2008. Python Programming Language. Python Software Foundation, Hampton, New Hampshire. <http://python.org>.

R Development Core Team, 2008. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0. <http://www.R-project.org>.

Redfern, J.V., Ferguson, M.C., Becker, E.A., Hyrenbach, K.D., Good, C., Barlow, J., Kaschner, K., Baumgartner, M.F., Forney, K.A., Ballance, L.T., Fauchald, P., Halpin, P., Hamazaki, T., Pershing, A.J., Qian, S.S., Read, A., Reilly, S.B., Torres, L., Werner, F., 2006. Techniques for cetacean-habitat modeling. Marine Ecology Progress Series 310, 271–295.

Roden, C., 1999. Cruise Results; Summer Atlantic Ocean Marine Mammal Survey; NOAA Ship Oregon II Cruise OT 99-05 (236). NOAA Southeast Fisheries Science Center.

Rogerson, D.E., 1997. Inside COM: Microsoft's Component Object Model. Microsoft Press, Redmond, Washington.

Schmitz, O., Karssenberg, D., van Deursen, W.P.A., Wesseling, C.G., 2009. Linking external components to a spatio-temporal modelling framework: coupling MODFLOW and PCRaster. Environmental Modelling & Software 24, 1088–1099.

Sing, T., Sander, O., Beerenwinkel, N., Lengauer, T., 2005. ROCR: visualizing classifier performance in R. Bioinformatics 21 (20), 3940–3941.

Stevick, P.T., McConnell, B.J., Hammond, P.S., 2002. Patterns of movement. In: Hoelzel, A.R. (Ed.), Marine Mammal Biology: An Evolutionary Approach. Blackwell Science, Oxford.

Treml, E.A., Halpin, P.N., Urban, D.L., Pratson, L.F., 2008. Modeling population connectivity by ocean currents, a graph-theoretic approach for marine conservation. Landscape Ecology 23, 19–36.

Valavanis, V.D., Pierce, G.J., Zuur, A.F., Palialexis, A., Saveliev, A., Katara, I., Wang, J., 2008. Modelling of essential fish habitat based on remote sensing, spatial analysis and GIS. Hydrobiologia 612, 5–20.

van Rossum, G., 2001. Making Types Look More Like Classes. Python PEP 252, Version 56033. <http://www.python.org/dev/peps/pep-0252/>.

van Rossum, G., 2008. Extending Python with C or C++. In: Extending and Embedding the Python Interpreter, Release 2.5.2. <http://www.python.org/doc/2.5.2/ext/>.

Vilchis, L., Ballance, L.T., Fiedler, P.C., 2006. Pelagic habitat of seabirds in the eastern tropical Pacific: effects of foraging ecology on habitat selection. Marine Ecology Progress Series 315, 279–292.

Waring, G.T., Josephson, E., Fairfield, C.P., Maze-Foley, K. (Eds.), 2007. U.S.Atlantic and Gulf of Mexico Marine Mammal Stock Assessments – 2007. NOAA Technical Memorandum NMFS NE 205, 415 pp.

Weigert, T., Dietz, P., 2003. Automated generation of marshaling code from high-level specifications. In: Reed, R. (Ed.), SDL 2003: System Design. Lecture Notes in Computer Science 2708. Springer-Verlag, Berlin.

Wood, S.N., Augustin, N.H., 2002. GAMs with integrated model selection using penalized regression splines and applications to environmental modelling. Ecological Modelling 157, 157–177.