--- *Chapter One* ---

Introduction

The aim of this workbook is to help marine biologists familiarise themselves with using GIS in their research. To do this, it uses the same Task Oriented Learning (TOL) approach first introduced in *An Introduction To Using GIS In Marine Biology* to provide five exercises based around the creation of custom GIS tools. Developing such custom GIS tools allows you to quickly and easily repeat tasks for which there are no existing tools. As such, this volume does not represent a stand alone GIS book and it is meant to act as a companion guide to the original book rather than to replace it in any way. It does not provide any background information on using GIS as this has already been covered within *An Introduction To Using GIS In Marine Biology* itself. Instead, it simply provides instructions for doing the exercises themselves.

Thus, this workbook is primarily aimed at those who have read some or all of *An Introduction To Using GIS In Marine Biology*. If you have not already done so, it is recommended that, at a minimum, you read chapters seven ('*Translating biological tasks into the language of GIS*'), eleven ('*How to use the 'How To...' sections of this book*') and twenty ('*How to combine instruction sets for basic tasks to create instruction sets for more complex tasks*') of *An Introduction To Using GIS In Marine Biology* before working through any of these exercises. It will also help if you are familiar with the basics of GIS (chapter two), common concepts and terms in GIS (chapter three), the importance of projections, coordinate systems and datums (chapter four), types of GIS data layers (chapter five), starting a GIS project (chapter six) and how to set up a GIS project (chapter thirteen). Finally, it is worth at least flicking through chapters thirteen to nineteen to familiarise yourself with how instruction sets are laid out using the TOL approach introduced in *An Introduction To Using GIS In Marine Biology*.

This supplementary workbook uses ArcGIS® 10.2 software to illustrate how the tasks related to creating custom GIS tools for automating frequently repeated tasks can be done.

However, similar processes are likely to be used to achieve similar outcomes in other GIS software packages and using other automation processes. The exercises provided in this book are designed to be worked through in a sequential manner. This is because the same data sets are used throughout and you will need to use some of the data layers generated in earlier exercises for later ones. In addition, the exercises lead on from each other in a manner that develops your familiarity with creating custom GIS tools. For example, exercise one covers the creation of a tool for plotting locational data contained in a spreadsheet and transforming them into a custom projection/coordinate system, while exercise two builds on this tool to make a more complex custom tool for creating a presence-absence data layer of species distribution from such data. These tools are continually built up throughout the workbook until, by exercise four, you will be creating a complex tool which allows you to automate the process of calculating abundance per unit survey effort for a species using a polygon grid data layer. Those familiar with other supplementary workbooks in this series will see that in many cases, the tasks which will be automated are ones which have previously been done manually. Thus, the exercises in this workbook build on those contained in the earlier ones.

The exercises are provided using the same flow diagram based format introduced in the '*How To...*' reference guide section of *An Introduction To Using GIS In Marine Biology*, and specifically in chapter twenty which outlined how to combine individual instruction sets to work out how to do more complex tasks. This means that for each exercise, you will first find an outline of what will be achieved by the end of it, why it is useful for marine biologists to be able to do this and what data layers you will need to start with. You will then find a summary flow diagram which will detail the order in which individual instruction sets for basic tasks must be done. Finally, you will find a set of numbered instruction sets based on those provided in *An Introduction To Using GIS In Marine Biology*. These have been customised to make them specific to the data set used for each example. In order to complete a specific exercise, you will need to work through each of these instruction sets in the order given in the summary flow diagram. To allow you to know whether you are progressing correctly, figures will be provided at regular intervals which will show you what the contents of various windows in the individual software packages being used for each exercise should look like at that specific stage.

The data sets used in each exercise can be downloaded from *www.gisinecology.com/books/marinebiologysupplementaryworkbook*.

Before you start working through the exercises in this supplementary workbook, six terms need to be defined. These are component, module, parameter, user-defined parameter, precondition and template data layer. Components are the basic building blocks from which custom GIS tools are built and they can be thought of as being equivalent to individual sections in a flow diagram. This means that they form the simple stepping stones which are used to carry out more complex tasks. In general, each component will consist of a single function that does a single specific action, such as selecting data within a data layer or adding a field to an attribute table. In ArcGIS 10.2 software, each component consists of either a single pre-existing tool which will be customised in some way, or a specially written script. However, only the former will be used in the exercises in this supplementary workbook.

A module is a set of components within a custom tool which, when taken together, do a specific complex operation. Effectively, a module is the smallest element within a custom tool which does a complete task from start to finish. For example, a module might consist of all the components required to plot the locational data for a species contained in a spreadsheet (see exercise one), or the creation of a polygon grid data layer (see exercise three). The simplest custom tools might only have a single module consisting of a single component, while more complicated ones can have many different modules, each made up of many different components. In general, when building a custom tool, you will create each module individually and check that it works before moving onto the next one.

A parameter is any input or output variable which needs to be set before a specific component within a custom tool will be work properly. Parameters can be external files (for example a spreadsheet file containing locational information for a species of interest), data layers (for example a line data layer which represents survey coverage), or a value for a specific characteristic of the output data layers which will be produced by the specific component (such as the cell size or extent of a raster data layer which will be created by the component). There are two ways which parameters can be set. These can either be fixed in advance when the component is built into the custom tool or they can be user-defined.

When a parameter is fixed in advance the same value, file or file name will be used every time the component is run (e.g. it may always use the same cell size or extent when generating a raster data layer). In contrast, when a parameter is set to be user-defined, the user will specify the exact value, file or file name to be used each time the component is run within the custom tool. Depending on exactly what you wish a specific component to do when you use your custom tool, it will be important to decide if a parameter should be set to use a fixed value, file or file name or whether it should be user-defined. In general, user-defined parameters allow you to create more generic custom tools than fixed parameters. However, they also leave more room for the user to decide on exactly what the parameters should be (potentially giving less standardised results).

A precondition is a parameter in a model, typically an output data layer from a specific component or module, which must exist before a later component or module within the same custom tool can be run. For example, you may have one set of modules in a custom tool which is working with the locational data for a target species, such as plotting it a the GIS project and converting it into the required projection/coordinate system, and another set of modules which is working with the survey data, such as working out the total level of survey effort in each grid cell in a polygon grid. If the aim of your custom GIS tool is to calculate the number of animals recorded in each grid cell per unit of survey effort (see exercise four), then the outputs of these two modules would need to be set as preconditions for any module which combines both the point data of species locations and information about the amount of survey effort in each grid cell. This is because they would have to be completed in order for this last module to be able to function. Thus, preconditions effectively set the order in which different modules within the custom tool are run whenever the tool is used. While most preconditions will be set within the custom tool itself, there are other preconditions which are set by the structure of the data layers or files required for the custom tool to work. For example, there might be a precondition for a tool which specifies that there must be specific fields with specific names in the attribute table of a data layer. An example of this can be found in exercise four, where the custom tool which is built requires that there is a field in the attribute table of the species locational data layer called 'Number' which contains the number of animals recorded at each location. If this field is not present, then the tool will not work properly.

Related to preconditions are template data layers. Template data layers are intermediate data layers generated or used within a custom GIS tool. Often they are required to have specific names and structures, and to be stored in a specific location on the users computer. Thus, template data layers are simply intermediate data layers which will be over-written every time the custom tool is run. The reason they are used is so that they can act as fixed parameters, meaning that the user does not need to define the names of a large number of intermediate file names and locations each time the custom tool is run. In addition, it prevents the user's hard drive becoming clogged with files for data layers which do not need to be kept after the custom tool has been successfully run. However, if these template data layers do not exist at the right location on the computer on which a given custom tool is being used, the tool will not work, and this is one of the main reasons that custom tools stop working. Thus, it is important to ensure that all template data layers are stored in a specific folder for which the address will not change from one computer to another.

The issue of template data layers leads onto the issue of absolute or relative addresses for saving the data layers and other files generated by a specific tool. If an absolute address is used for specific files within a custom tool, then this file will need to have exactly the same address on every computer where it will be used (e.g. C:\Toolbox\Tool1). While this works well for computers where you are free to store files or create folders on the main hard drives, it does not work as well when users have to store files on their own individual directories on the computers they use. This is because each user's file space will have a different address. For example, one user might have the address C:\Alice at the start of all their files names, and another C:\Bob. In this case, Bob will not be able to use any custom tools which Alice creates using absolute addresses. This is because all the absolute addresses will start C:\Alice, and Bob will not have access to any files with such addresses. This is where relative addresses come in. Relative addresses will generally look like this: …\Toolbox\Tool1. When a custom tool is set to use a relative address, it will look for files relative to where it is stored and not the absolute address. Thus, if Alice wanted to share a custom tool with Bob, she could use a relative address such as …\Toolbox\Tool1 to store the tool itself and any files it requires to function. This would mean that it would not matter if the absolute address on her computer was C:\Alice\Toolbox\Tool1, while on Bob's it was C:\Bob\Toolbox\Tool1. However, for the custom tool to work properly, all files,

including the tool itself, would need to be stored in the same relative folders and sub-folders.

For all the exercises in this supplementary workbook, absolute addresses will be used. This is because it is simpler for those trying to create custom tools to automate their GIS processes for the first time. However, they will use a relatively simple absolute address to store both the tools created and the files which they require (in this case C:\GIS_EXERCISES_6). This means they could be transferred to any other computer with a C: drive simply by copying this folder from the C: drive of one computer to the C: drive of another. As you become more proficient with building custom GIS tools, you may wish to explore the possibility of using relative addresses within them. In ArcGIS 10.2 software, the ModelBuilder module, which provides one way to create custom tools, uses absolute addresses by default. If you wish to use relative addresses, then as soon as you have created your new tool (and before you add any components or modules to it), right click on its name in the TOOLBOX window and select PROPERTIES. If you then click on the GENERAL tab, you will see a box towards the bottom which says STORE RELATIVE PATH NAMES (INSTEAD OF ABSOLUTE PATHS). If you tick this box, your custom tool will save all the addresses relative to the folder where the tool itself is stored rather than as absolute addresses.

Whenever you make a custom tool, you should create metadata for it. This will describe what it does, why and when it was created, what software you used to create it (and what version of the software), whether there are any preconditions for the use of a specific custom tool (such as a requirement for data layers to have specific structures), what parameters it requires which can be user-defined, your name and contact information (if you are going to share it with other people) and any limitations on its use or usefulness. This can be done by creating a 'read me' file (a simple text file stored with the custom tool which provides this information and which can be opened by any word processing software package), by creating specific metadata attached to the tool itself or through a combination of the two (see exercise five for an example).

In the ArcGIS 10.2 software package, used to illustrate how to create custom GIS tools in this supplementary workbook, there are two possible ways to construct them. The first is to

use the ModelBuilder module of the software, and this is the approach which will be used here. This allows for custom tools to be constructed from existing tools and functions available within the software package using a flow diagram approach similar to the TOL Approach for teaching GIS used in the *Introduction To Using GIS In Marine Biology* series of books. The second is to use an open-source package called PYTHON. This is a command-based programming language and requires that you learn how to write Python code. However, you can move between these two approaches, and you can generate the Python code for any custom tool you create in the ModelBuilder module of ArcGIS by clicking on the MODEL menu and selecting EXPORT> TO PYTHON SCRIPT… If you wish to create custom tools in Python, this can be done from within the ArcGIS 10.2 software environment by clicking on GEOPROCESSING on the main menu bar of the ArcMap user interface and selecting PYTHON. If you wish to see the Python code for a specific tool you have created in the ModelBuilder module, you can export it as a Python script and then open it in this Python module.

NOTE: The instruction sets provided here are for training purposes only, and they are only meant to be an aid to learning how to use GIS in marine biological research. While every effort has been made to ensure that these instructions are complete and error-free, they come with no guarantee of accuracy and, as with all technical books, some errors may have slipped through undetected. Whenever I become aware of any such issues, I will post corrections on *www.GISinEcology.com/books/marinebiology/corrections* rather than waiting to correct them in the next edition of this book. Before doing any of the exercises in this book, you should check this webpage to see if any corrections have been posted there. In addition, it is important to realise that there is no guarantee that these instructions will produce the desired outcome in every circumstance. As a result, if you are using the instruction sets provided here to learn how to do critical tasks, it is essential that you check (and then double check) that they work for your given circumstances rather than blindly following them without thinking. The author will not be responsible for any errors which occur because of the application of these instruction sets to real world situations.

NOTE: As with many things in GIS, there may be more than one way to do the exercises outlined in this book. The instructions presented here will work for the data sets provided and for the exercises outlined in this book. If you find an alternative way to do them which

works for your data, or if you have someone who can show you how to do them in another way, feel free to do them differently. In particular, there are many different ways for generating the coding required to automate GIS-based tasks. This includes using the ModelBuilder module of the ArcGIS 10.2 software package, which uses a graphic-based format similar to the flow diagrams used in this supplementary workbook, using the open source Python programming language or using the free statistical software R. However, in all cases, similar steps will be required to be incorporated into the programming to allow the task to be correctly automated.