Advanced Geoprocessing and ModelBuilder Concepts

A. Michelle Lawing
Ecosystem Science and Management
Texas A&M University
College Station, TX 77843
alawing@tamu.edu
Announcements

• OPTIONAL: Revise and resubmit your Proposal! One week – you get up to half the points back that you lost.

• Dr. Smith is coming next week (Nov. 10th) to lecture about ArcGIS Online.

• Establish your ArcGIS Online log in
  Username: email_tamu
  Password: password1
ModelBuilder

• Now you have had some practice with ModelBuilder

• Today we will talk about some advanced processing options

• These are common tools used in any geoprocessing framework
Managing Intermediate Data

• GIS analysis can get messy
• Most of the tools that you run produce an output dataset
• When you chain many tools together those datasets start piling up
• It's easy to wind up with a folder full of datasets with names like buffers1, clippedbuffers1, intersectedandclippedbuffers1, raster2reclassified, etc.
Managing Intermediate Data

• In most cases, you are concerned with just the final output dataset

• *Intermediate data* are just temporary

• You only need to keep it around to run the model, and then it can be deleted

• ModelBuilder can manage your intermediate data for you
Managing Intermediate Data

- ModelBuilder will place *intermediate data* in a temporary directory called the *scratch workspace*.

- The scratch workspace is your operating system's temp directory.

- You can configure it to exist in another location.
Managing Intermediate Data

• You can force data to go into the scratch workspace by using the %SCRATCHWORKSPACE% variable in the path

• For example: %SCRATCHWORKSPACE%
  myOutput.shp
Managing Intermediate Data

• You can also mark any element in ModelBuilder as *Intermediate* and it will be deleted after the model is run.

• By default, all *derived* data are Intermediate data.
Looping in ModelBuilder

• Looping, or iteration, is the act of repeating a process

• A main benefit of computers is their ability to quickly repeat tasks that would otherwise be mundane, cumbersome, or error-prone for a human to repeat and record

• Looping is a key concept in computer programming and you will use it often in geoprocessing
Looping in ModelBuilder

- ModelBuilder contains a number of elements called Iterators that can do looping in various ways.

- The names of these iterators, such as **For** and **While**, actually mimic the types of looping that you can program in Python and other languages.
Looping in ModelBuilder

Iterator: For

Description: Iterates over a starting and ending value by a given value.
Looping in ModelBuilder

Iterator: While

Description: Executes "while" a condition is true or false for the input or set of inputs
Looping in ModelBuilder

- Iterate Feature Selection - Iterates over features in a feature class
- Iterate Row Selection - Iterates over rows in a table
- Iterate Field Values - Iterates over each value in a field
- Iterate Multivalue - Iterates over a list of values
- Iterate Datasets - Iterates over datasets in a Workspace or Feature Dataset
Looping in ModelBuilder

- Iterate Feature Classes - Iterates over feature classes in a Workspace or Feature Dataset
- Iterate Files - Iterates over files in a folder
- Iterate Rasters - Iterates over rasters in a Workspace or a Raster Catalog
- Iterate Tables - Iterates over tables in a workspace
- Iterate Workspaces - Iterates over workspaces in a folder
Understanding an Iterator

• Each iterator has a set of parameters that may differ from the other iterators
• The overall structure of all iterator tools is very similar
Understanding an Iterator

- Iterate Feature Classes requires an Input Workspace where all the feature classes to iterate through are stored.
Understanding an Iterator

- Wildcard and Feature Type are used to restrict what feature classes in the workspace are iterated.
Understanding an Iterator

- Wildcard limits the feature classes by their names
- Feature Type limits the feature classes by their feature types
Understanding an Iterator

- Recursive is used to control the iteration over feature classes within subfolders in the workspace.
Understanding an Iterator

- Iterate Feature Classes has two output variables: the output feature class and the name of the feature class
Understanding an Iterator

• The output feature class can be connected to the next tool for processing
Understanding an Iterator

- Name can be used for inline variable substitution
Understanding an Iterator

• For example, if the Buffer tool was added to the model, and the Feature Class variable was connected to the tool, every feature class in the workspace would be buffered
Notes

• Only one iterator can be used per model.
• The options to add another iterator will be disabled if one iterator exists in the model.
• If an iterator is added to a model, all tools in the model iterate for each value in the iterator.
• If you do not want to run each tool in the model for each iterated value, create a submodel/model within a model that contains only the iterator and add it as a model tool into the main model.
Notes

• If a model containing an iterator is exported to a Python script, the script will **not include the iteration logic**.

• Python listing logic can be added to the script to achieve a similar effect.
Example using an Iterator

We will use the **Iterate Feature Classes** iterator to

1. Project every feature class in a workspace
2. Name each **Project** tool output dynamically based on the input name using **inline variable substitution**
3. Add a new field to each feature class.
Example using an Iterator

- An input workspace is specified for Iterate Feature Classes to iterate through every feature class in that workspace.
Example using an Iterator

- Feature classes to iterate through are restricted by specifying a **wildcard** and **feature type** (only **polygon** feature classes with with I will be iterated through)
Example using an Iterator

- Iterate Feature Classes produces two outputs: the feature class iterated through and the name of the feature class
Example using an Iterator

• The feature class variable is connected as the input to the **Project** tool
Example using an Iterator

• The feature class name variable is used for inline variable substitution to dynamically name each output from the Project tool with the same name of the input feature class.
Example using an Iterator

- A new field is added to every feature class by connecting the **Add Field** tool.
Accessing Iterators in ModelBuilder

• Insert menu - Click Insert > Iterators and choose an iterator to add to the model

• Shortcut menu - Right-click in open space in the model, click Iterators, then choose the iterator to add to the model