Using Python to Automate Production of Species Conservation Range Maps

Don Katnik
Maine Department of Inland Fisheries and Wildlife

Organization of Fish & Wildlife Information Managers 2015 Conference
Williamsburg, VA

DISCLAIMER: I don't work for Python
Using Python to Automate Production of Species Conservation Range Maps

Background

• Maine wanted to generate a “Conservation Range Map” for each Species of Greatest Conservation Need in our 2015 State Wildlife Action Plan update.

• A good task for automation!

• Python is a free, easy-to-use programming language for automating data workflows.

• In this presentation, I’ll explain how I used Python to automate the production of hundreds of SGCN range maps.
Using Python to Automate Production of Species Conservation Range Maps

Presentation Overview

- Desired Outputs
- Available Inputs
- Intermediate “Process-Generated” Components
- Python Scripts
- Conclusions

Python Script(s)?

Desired Outputs – What's an "SGCN"?

- Species of Greatest Conservation Need
- A species at-risk of becoming “listed” as Endangered/Threatened
- Maine identified 354 SGCNs
- Across all taxa, habitat types, and parts of the state

Python Script(s)?


354 SGCNs
What's a "Species Conservation Range" (SCR) Map?

- Shows where a Conservation Action may benefit the SGCN
- Based on where the SGCN is believed to occur and what habitats the SGCN is believed to use (and could occupy)
- Generalized to sub-watersheds and “Townships” (sub-counties)

Python Script(s)?

Raw Data? → Intermediate Components? → Sub-watershed SCR Map → Township SCR Map

354 SGCNs
Using Python to Automate Production of Species Conservation Range Maps

Desired Outputs – Map Format

- Adobe PDF files
- Most supported document format
- Familiar to many users
- Can turn layers on/off like in GIS (including data sources)
- Cannot be altered by users
Using Python to Automate Production of Species Conservation Range Maps

Desired Outputs – Ability to Revise/Update

- Changes in map appearance (colors, text, etc.)
- Additional/updated input data
- Additional SGCNs
- Changes in habitat associations / geographic filters

Python Script(s)?
- Raw Data?
- Intermediate Components?
- 354 SGCNs
- Revise/Update

Sub-watershed SCR Map
- Township SCR Map
# Available Inputs – Observation Data

<table>
<thead>
<tr>
<th>Format</th>
<th>Content</th>
<th>Data Set(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIS Shapefile</td>
<td>Polygons</td>
<td>MDIFW Essential Wildlife Habitats</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MDIFW Endangered, Threatened, &amp; Special Concern Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MDIFW Shorebird Areas</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MDIFW Vernal Pools</td>
</tr>
<tr>
<td>GIS Shapefile</td>
<td>Points</td>
<td>MDFIW Fish Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MDFIW Lynx Database</td>
</tr>
<tr>
<td>Excel Spreadsheet</td>
<td>Lat/Long Coordinates</td>
<td>eBird</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maine Bee Surveys</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maine Damselfly/Dragonfly Surveys</td>
</tr>
<tr>
<td>Access Database</td>
<td>Lat/Long Coordinates</td>
<td>Maine Mussel Surveys</td>
</tr>
<tr>
<td>Excel Spreadsheet</td>
<td>Township</td>
<td>Maine Amphibian/Reptile Atlas Project</td>
</tr>
<tr>
<td>Excel Spreadsheet</td>
<td>Lat/Long or UTM Coordinates</td>
<td>Maine Rare Mayfly Surveys</td>
</tr>
<tr>
<td>Excel Spreadsheet</td>
<td>Lat/long or UTM Coordinates or Township or Sub-watershed</td>
<td>&quot;Historical&quot; (Miscellaneous Source) Data</td>
</tr>
<tr>
<td>Excel Spreadsheet</td>
<td>Township or Sub-watershed</td>
<td>Maine Butterfly Surveys</td>
</tr>
<tr>
<td>Excel Spreadsheet</td>
<td>&quot;Stop&quot; IDs Points</td>
<td>Breeding Bird Survey</td>
</tr>
<tr>
<td>GIS Shapefile</td>
<td>Points</td>
<td></td>
</tr>
</tbody>
</table>

"Historical" (Miscellaneous Source) Data
We'll focus on the "Townships" (sub-county) maps, but there is a parallel set of everything for making the sub-watershed maps.
Using Python to Automate Production of Species Conservation Range Maps

Output – PDF Map for Each SGCN

• Townships are "lit up" based on observations (i.e., presence) of the SGCN

• Colors (e.g., red, yellow) identify the observation data sources

• To make all these PDF maps, I wanted a single map layout template that I could customize for each SGCN...
Using Python to Automate Production of Species Conservation Range Maps

Intermediate Components – ArcGIS Map Template

CUSTOM LEGEND
Each of these composite graphic/text elements is the legend item for an observation data source. By default they are "docked" off the page. For each SGCN, Python finds the legend items for its observation data sources and moves them onto the page to build a custom legend.

TITLE = SPECIES NAME
This "text element" in the template layout has a unique name that allows Python to find and manipulate it (i.e., change the text string)
Intermediate Components – ArcGIS Map Template

- Table of Contents contains layer files that point to the temporary single-SGCN Township shapefiles
- There is a shapefile for each observation data source
- Each shapefile contains only those Townships where the SGCN was observed based on that data source
Intermediate Components – Lookup Tables

• One table for each observation data source (parallel structure to the shapefiles)
• Each record is a unique Species × Township occurrence
• Only presence data
Using Python to Automate Production of Species Conservation Range Maps

Python Scripts – Populating Lookup Tables

For each Observation Data Source:

- Get its Lookup Table (OUTPUT)
- Get its Observation Data Source Shapefile (INPUT)

For each SGCN:

- Select Observations of the SGCN from the Observation Shapefile
- Overlay Selected Observations with Township Layer

For each Township where SGCN was observed:

- Write SGCN name and Township code to Lookup Table

Write summary of SGCN analysis to log file
Python Scripts – Speed Tricks

- Takes a LONG time to process all data sources and SGCNs
- Script allows user to specify which data sources to run
- If records already exist for an SGCN in the Lookup Table, it is skipped (to refresh data for an SGCN, its previous output records in the table(s) were deleted)
- For code development, can indicate a "test run" = only one SGCN will be processed to test if code works
- Log files = easier debugging
Using Python to Automate Production of Species Conservation Range Maps

Python Scripts – Making the PDF Maps

For each SGCN:

1. Empty the temporary, single-species Township shapefiles
2. For each Lookup Table (Data Source):
   a. Get its temporary, single-species Township shapefile (now empty)
   b. Query Lookup Table for records for the SGCN:
   c. Load resulting Townships into the generic, single-species Township shapefile
3. Customize map layout Title (SGCN name)
4. Build custom legend
5. Toggle layers on/off depending on which have data for the SGCN
6. Export ArcMap Layout to PDF
Python Scripts – Speed Tricks

• Takes a LONG time to process all SGCNs
• If PDF files already exist for an SGCN, it is skipped (so to refresh maps for an SGCN, the existing PDF files must be deleted)
• Load Township feature geometries into a Python dictionary
• Load Lookup Tables into a nested Python dictionary
• Log files = easier debugging
Conclusions

• I have refreshed the entire set of SGCN maps several times
• Have frequently replaced maps for select SGCNs
• Would have been almost impossible without script automation
• Generating log files as the scripts processed data was invaluable for debugging problems

And the peasants rejoiced...

...and gave praise to Python!
Conclusions, Cont.

Dynamic generation of SGCN maps from Lookup Tables:

- Minimal file sizes
- Produced desired output with reasonable processing time
- Easy to add additional data sources and SGCNs
- Map template easy to change

- Users sometimes wanted the saved shapefiles for an SGCN (they don’t exist)
- Changing an SGCN's map required re-running the script (cannot just edit an MXD)
- Multi-species queries require custom scripts
- Not intuitive to most users; requires detailed documentation to ensure someone else could use the process
The Obligatory "Got Questions?" Dog Pic
• Location coordinates stored in different coordinates systems and formats (even within the same database)

• Python can convert text/numeric formats with a series of "If...Then" statements BUT have to know beforehand all the variations that need to be addressed

• Coordinating taxonomy:
  – Strip superfluous words/symbols
  – Identify inconsistent species
  – Hard-code “if-then” substitutions

• Unique script for every observation data source (but most reasonably short)
Input – Observation Shapefiles

• One shapefile for each observation data source
• Most contain buffered points or mapped polygons depicting species observations
• Overlaid with Township features to determine which SGCNs were observed in which Townships
• Overlay results summarized into Observation Tables
Using Python to Automate Production of Species Conservation Range Maps

Available Inputs – Habitat Data

- National Hydrography Dataset
- U.S. Fish and Wildlife Service National Wetlands Inventory
- Maine Natural Areas Program Freshwater Tidal Marshes
Using Python to Automate Production of Species Conservation Range Maps

Intermediate Components – Generic, Single-SGCN Township Shapefiles

- The layers in the Map Template point to these shapefiles
- Initially they are empty
- As each SGCN is processed, only those Townships where the SGCN was observed are loaded into them (no wasted space storing Townships where the SGCN was not observed)
- Each shapefile is for a different observation source (allows users to view them individually)