**separate\_attributes\_to\_tif.py**

**workflow**

Hello! Please use this document to get a basic understanding of the script before running or editing it. Please update this document with changes you make.

*Please note, columns and attributes are the same thing and are referenced interchangeably.***What should I read?**

*I am running the script from scratch on a new or existing year.*

Environment -> Essential Parameters -> Additional Parameters -> Metadata Templates -> Creating Tifs -> Symbology Files -> Packaging

*I want to regenerate the attribute tifs’ metadata files without reprocessing the image.*

Environment -> Essential Parameters -> Additional Parameters -> Metadata Templates -> Metadata Regeneration

**Environment**

The follow packages must be installed in your python environment for the script to run. The install commands are provided for convenience:

**Install Commands**

conda install conda-forge::gdal

pip install numpy simpledbf pandas bs4

**Essential Parameters**

These parameters should be checked and changed as needed each time the script is run. They are located near the top.

Text

Description automatically generated

chunk\_size – the size of each processing chunk. 29060 or less is recommended for systems with 32GB of RAM.

treeMapTif – the filepath to the main dataset. This is the dataset that will have its attributes separated.

treeMapDbf – the filepath to the main dataset’s vat.dbf file. This is the attribute table used by the main dataset.

outputFolder – the output folder for the separated tifs, their metadata, and their packaged forms.

**Additional Parameters**

These parameters should be checked and updated when switching between versions of TreeMap.

Text

Description automatically generated

Text

Description automatically generated

treeMapDatasetNoDataValue – the NoData value used in the main dataset’s dbf.

creation\_options – the creation options for GeoTIFF compression, tiling, and file format.

cols – the attributes to be separated from the main dataset and their desired data types.

col\_descriptions – the attributes and their associated descriptions.

discrete\_cols – the attributes, that will have attribute tables generated, and their associated palettes from the palettes.json.

col\_units – the units of each attribute. Used in the metadata.

**Metadata Templates**

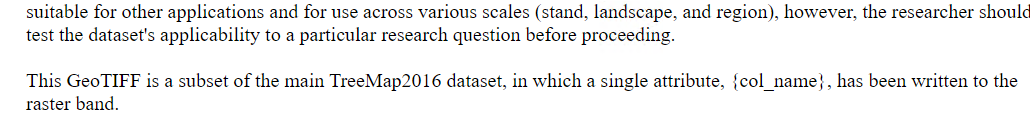
Metadata template files need to be created for each year of TreeMap. These templates will be used when generating each attribute’s metadata. Existing metadata templates can be edited if desired. There are four metadata files required, in two categories (basic + arc). All metadata files are, and must be, stored in gtac-treemap\data\_portal\_scripts\supp\_files\metadata\_templates

Graphical user interface, text

Description automatically generated

**Basic**

These are the basic xml + html metadata files; They’re based off the main dataset’s. These templates must be created for each year of TreeMap. If you’re reprocessing an existing year of TreeMap, you do NOT need to recreate these files, but feel free to make changes if you desire.



Steps for creating templates for new TreeMap years:

1. Copy and paste the main dataset’s xml metadata into gtac-treemap\data\_portal\_scripts\supp\_files\metadata\_templates.
2. Open the file with Visual Studio Code or another program that allows you to edit xml/html.
3. Edit the text in the file as you please. (e.g., remove/add information)
   1. If deleting any xml or html tags, make sure you know what you’re doing.
4. Add the following text in places where you want each attribute to populate its unique information:
   1. {col\_name}
      1. Insert this text, with the curly brackets, anywhere you want each attribute’s unique name to appear. (e.g., FORTYPCD will appear in its own metadata while ALSTK will appear in its own metadata)
   2. {col\_description}
      1. Insert this text, with the curly brackets, anywhere you want each attribute’s unique description to appear.
   3. {date}
      1. Insert this text, with the curly brackets, anywhere you want the date the metadata was generated to appear.
   4. {data\_gateway\_link}
      1. Insert this text, with the curly brackets, anywhere where you want the data gateway to appear.
5. Save the file and rename it in this format:
   1. YEAR\_metadata\_template.xml (e.g., 2016\_metadata\_template.xml)
6. Repeat all steps for the html metadata.

**Arc**

These are the ESRI compatible metadata files; They are based on the attributes’ basic metadata that gets generated. These templates do NOT need to be updated or created for new years of TreeMap. They may need to be updated for new versions of ArcGIS Pro, although this is unlikely.

**Creating Tifs**

Once your environment, parameters, and metadata templates have been setup, you’re ready to use the script to generate attribute tifs.

Steps:

1. Open whatever you use to run python and activate the environment containing the appropriate packages. I use Anaconda.

Text

Description automatically generated with medium confidence

1. Run *separate\_attributes\_to\_tif.py*



1. Review use considerations and type ‘images’ for the mode (do not include quotes)

A screenshot of a computer program

Description automatically generated

1. Type ‘all’ for image mode if you want to process all the attributes OR type the name of the specific attribute you want to process.
   1. ‘all’ will skip existing attribute tifs in the output folder

A screenshot of a computer

Description automatically generated with low confidence

1. Review the paths, chunk size, and TreeMap version and press ‘enter’ if they are correct. Otherwise, press q to quit and review the Essential Parameters section of this script. Please note that the TreeMap version is derived automatically from the main dataset’s filename.

A picture containing text

Description automatically generated

1. Wait for your attribute(s) to process!

**Symbology Files**

We created symbology files for continuous (NOT thematic or ordinal) attributes of TreeMap 2016 so users could synchronize the visualization with the TreeMap viewer in ArcGIS Pro and QGIS. These must be manually created before the ‘Packaging’ step. Each year’s layer files must be stored in their own folder, named after the year, in the symbology\_files folder. (e.g., C:\Users\NicholasStorey\gtac-treemap\data\_portal\_scripts\symbology\_files\2016)

Table

Description automatically generated

Steps:

1. Add all the attribute tifs to ArcGIS Pro.
2. Design their symbology.
3. Export each layer as a layer file.
4. Save each one to the folder of its respective year (e.g., C:\Users\NicholasStorey\gtac-treemap\data\_portal\_scripts\symbology\_files\2020)
   1. Create the year folder if needed
5. Repeat in QGIS with qml exports

**Packaging**

Once the attribute tifs and metadata have been generated, and you’ve designed the symbology files for each attribute, you can run the packaging portion of the script to zip all the files together. The script generates a readme with explanations of each of the files included in the zip.

Text

Description automatically generated with medium confidence Graphical user interface, text

Description automatically generated

Steps:

1. Open whatever you use to run python and activate the environment containing the appropriate packages. I use Anaconda.

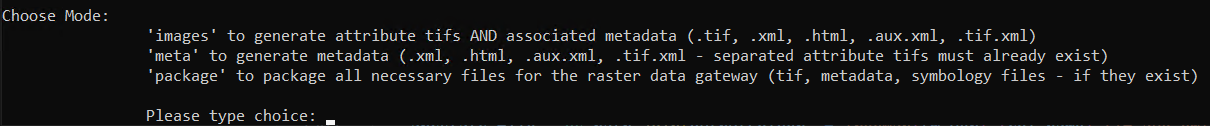
Text

Description automatically generated with medium confidence

1. Run *separate\_attributes\_to\_tif.py*



1. Type ‘package’ for mode



1. Review the paths and TreeMap version. Press enter if correct.
   1. The ‘attribute tifs and metadata folder’ should be the folder where the tifs + metadata are stored. In the script, this is just the outputFolder.

A picture containing graphical user interface

Description automatically generated

1. Wait for processing to finish. A folder, 00\_Zipped\_Files, will be created in the outputFolder with all the zipped files.

Graphical user interface, text, application, email

Description automatically generated

**Metadata Regeneration**

When processing with the ‘images’ mode of the script, metadata for each attribute gets generated alongside the images. However, it’s common to make a mistake when creating the metadata templates or to want to change something about the images’ metadata after the images have been created. Instead of reprocessing the imagery **and** metadata, we can use a mode integrated in the script that will only regenerate the metadata. **NOTE: The tif for each attribute you want to generate metadata for MUST already exist in the directory assigned to the ‘outputFolder’ variable. Also, your metadata template files must already exist and be in the correct directory (see Metadata Templates section of this document)**

1. Open whatever you use to run python and activate the environment containing the appropriate packages. I use Anaconda.

Text

Description automatically generated with medium confidence

1. Run *separate\_attributes\_to\_tif.py*



1. Review use considerations and type ‘meta’ for the mode (do not include quotes)

A picture containing graphical user interface

Description automatically generated

1. Specify which metadata you want to generate.

Graphical user interface

Description automatically generated with medium confidence

1. Review the paths and TreeMap version. Press enter if correct.
   1. The ‘Metadata template folder’ should be the folder where the metadata templates are stored.

A picture containing graphical user interface

Description automatically generated