**Programmatically Using Shapefiles to Clip Daymet Precipitation**

1. **Introduction**

When using Python to extract cell information from raster-type data based on shapefiles, each cell must be iterated over to identify whether it is within the shapefile’s polygon or not. While Python can capably iterate over a small number of cells in this manner, it is poor at doing so with large quantities of data. Despite Python’s ability to incorporate expansive matrices, the iterative process can lead to RAM being filled and crashing before completion. A solution can be found in the form of ArcGIS Pro, in which the Clipping tool can be simply and easily implemented to “cut out” the necessary data from a feature layer or raster. Again, this works ideally with smaller datasets and few raster cells. Even simply trying to load a high-enough-resolution raster will crash ArcGIS Pro. Another problem with this software is the type of data capable of being converted into a raster. The Earth Science Data and Information System’s (ESDIS) and Terrestrial Ecology Program’s Daymet dataset is an example that struggles with usability within ArcGIS Pro. Not only is Daymet data at a 1 km by 1 km resolution across the entire North American continent creating agonizing wait times—if not outright breakage—but it is also on a Lambert Conformal Conic projection. This projection features cells with differing, elongated shapes the further poleward it extends resulting in a failure to load the data as a raster when opened in ArcGIS Pro. Daymet data can be opened as a feature layer instead of a raster, but stopping there loses the visualization and data alteration often the case with rasters.

As part of a pilot study examining downscaled global climate models (GCMs) and effects on end-century precipitation, these problems were encountered first-hand. The said project’s focus is the Mississippi River Basin, and to compare the precipitation values within the Daymet Precipitation data—a raster netCDF file type—against the GCMs, Mississippi River Basin shapefiles were attempted to mask outside of the basin.

Ultimately the solution is the union of Python and ArcGIS Pro using ArcGIS Pro’s tools in the form of ArcPy. Without the visualization in the software to break, the two systems can achieve what neither is capable of alone. The objective of this report is to present a repeatable, workable solution incorporating programmatic methods and ArcPy’s toolset to extract cells from a Daymet precipitation file based on any provided shapefile, then output the extracted cells into its own netCDF usable in later research.

1. **Methodology**

The Daymet precipitation data was downloaded from NASA’s EarthData online repository while the Mississippi River Basin shapefile was downloaded from USGS’s ScienceBase-Catalog. Although *much* time was spent on trial and error—testing to see how the netCDF Daymet data could be inputted as a non-raster and then output as a netCDF again—the overall process ultimately is simple and straightforward using only the ArcPy package.

The first step is to use the MakeNetCDFFeatureLayer tool to input the desired Daymet precipitation file as a feature layer. This turns each netCDF cell into a point value and gets around the inconsistent cell sizes from the Lambert Conformal Conic projection. Second, is to use the Clip tool—extracting only the points within the polygon of the shapefile to a new layer. Finally, this new layer of points in the shape of the Mississippi River Basin (in this instance) can then be converted back into a netCDF with the FeatureToNetCDF tool. Two versions of this process have been created. A script that is capable of iterating over several shapefiles and outputting netCDFs per shapefile, and an ArcGIS Pro toolbox script that incorporates a user interface but lacks the potential breadth the raw script has.

1. **Code Documentation**
   1. **Script Method**

Within the MakeNetCDFFeatureLayer method, there are a few necessary variables that must be specified (Figure 1). These include which netCDF variables are to be added to a point’s information on the feature layer (prcp;lat;lon), the variables to be used as the x and y coordinates within ArcGIS Pro, the name of the output feature layer (daymet\_prcp), and the netCDF dimensions the method should extract cells from (y;x).

A computer screen with text on it

Description automatically generated

Figure 1. Variables within the ArcGIS Pro tool MakeNetCDFFeatureLayer.

Figure 2 shows the incorporation of a shapefile and ArcPy’s Clip tool to clip the newly created point feature layer and extract only the points that fit within the shapefile’s polygon. In this instance, there is only one shape file included, but it technically iterates over every shape file in the list provided thus allowing for iterative clipping. The print operation shows progress, and in the case of multiple shape files, the use of the variable hucName will keep each different shapefile from overwriting the clipped layers. The Clip tool uses the output feature layer name from the previous step.

A screen shot of a computer code

Description automatically generated

Figure 2. Use of shapefiles to Clip from the provided Daymet data. The commented section of code allows for a greater number of shapefiles to base the Clip tool on, if desired.

The last step takes the newly clipped feature layer(s) and converts them back into netCDFs. To accomplish this, variables and dimensions need to be set (Figure 3). Note the input feature layer is the same as the previous step’s output feature layer. Under the fields\_to\_variables variable in Figure 3, each of prcp, lat, and lon is listed twice. The first time indicates the ArcPy field to convert from, and the second indicates the netCDF variable to convert to. The “#” indicates what unit to put each variable in. The fields\_to\_dimensions variable works similarly, except it converts ArcPy’s x and y fields to the netCDF’s dimension variables.

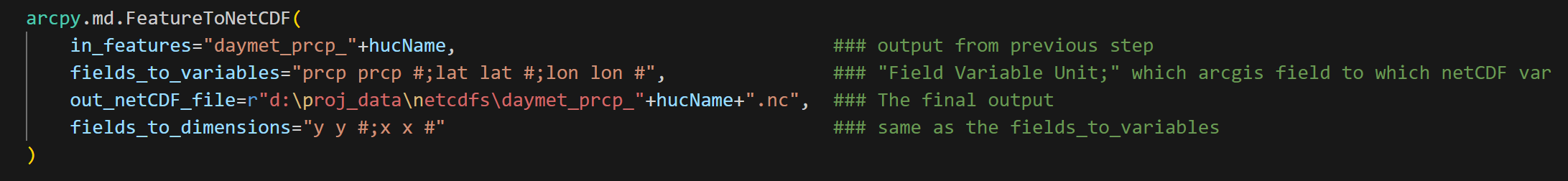


Figure 3. Method converting a clipped feature layer to a netCDF.

* 1. **Toolbox Method**

Although similar in function, the toolbox method does not take advantage of the iterative capabilities of python. Instead, it allows for a single instance of clipping and converting a Daymet file through a user interface. After defining the script\_tool’s function (Figure 4), the MakeNetCDFFeatureLayer is the same as it was in the script version but with the “daymet” variable being a path defined by user input in ArcGIS Pro’s user interface. Similarly, the clipping and converting back to netCDF processes use the same tools as the original script but are now specified by the user within the ArcGIS Pro interface (Figure 5).

A screenshot of a computer code

Description automatically generated

Figure 4. The function and first step to loading in the Daymet precipitation within the ArcGIS Pro toolbox.

A screenshot of a computer code

Description automatically generated

Figure 5. Steps 2 and 3 for clipping and converting the Daymet precipitation data within the ArcGIS Pro toolbox

1. **Results and Visualization**

Visualization of the clipped Daymet data shows exactly what was hoped for (Figure 6). Figure 6 shows the clipped output of annual precipitation for 1996. It is important to note that the x and y axis are in the form of meters rather than latitude and longitude. This is important as it means that further plotting requires the conversion of these coordinates or use of the indexes.

A blue and white map

Description automatically generated

Figure 6. Clipped 1996 annual precipitation.

This Daymet data is used with pilot testing model output from the Weather Research and Forecasting (WRF) system to compare which simulation would be the best model for further long-term simulation studies. Over such a large region as the Mississippi River Basin, there is much room for error and there is going to be no set of parameterization schemes resolving precipitation perfectly. This then means the most important regions to the Mississippi River Basin need to be prioritized and modeled as accurately as possible, while less important regions have some leeway to be less accurate.

Figure 7 shows the five-year (1996 — 2000) mean annual precipitation percent error and weighted percent error for six simulations testing different cumulus parameters and microphysics schemes. Each of the six simulations is compared with clipped Daymet precipitation data attained by this report’s method. The purpose of Figure 7 is to quantify the percent error and define by how much watersheds are diverging from the observed mean precipitation. The weighted percent error introduces a scalar that weighs the percent error by how important the watershed’s river contribution is to the Mississippi River Basin.

Within numerical weather models, precipitation is always a difficult metric to attain accurately. Because the end goal is to run a twenty-year climate simulation, year-to-year precipitation may not be accurate, but the trend and long-term amount should resemble the observed trends. The three most important basin contributors are the HUC05, HUC06, and HUC07 watersheds corresponding to the Ohio River, Tennessee River, and Upper Mississippi River and contributing 29.04%, 17.44%, and 20.05% respectively. The simulation cu1mp8 has the smallest combined error of these three watersheds of a 13.4% dry bias and 12.86% dry bias for the percent error and weighted percent error respectively. This simulation was chosen to move forward with due to this smaller error across the HUCs with the knowledge that the basin will have a dry bias of 42.38% and 29.84% with respect to the percent error and weighted percent error.

A map of different colors

Description automatically generated

Figure 7. Annual precipitation percent error and weighted percent error over each watershed of the Mississippi River Basin over the five-year period of 1996 — 2000.

A map of countries/regions with different colored areas

Description automatically generated

Figure 8. Mississippi River Basin’s Hydrologic Units (HUC) and Mean Annual Precipitation Weighted by River Discharge Contribution (mm).

**Challenges and Future Work**

The largest challenge with the output from this ArcPy method is that the dimensions are no longer in degrees latitude/longitude but in meters. The workaround is that in these clipped netCDFs, the x and y dimensions are the same as in the original Daymet data. This allows for easy searching using numpy’s numpy.where function to match the cells from the clipped data to the original. The desired cell indices can then be written to a text file and Python can easily use those with the original dataset to compare against the simulation output in degrees latitude/longitude.

The other limitation of this project is that each a different dataset is going to be using different variables and subsequently variable names. This introduces a problem as each step requires the specific variable names meaning the script cannot be implemented for a non-precipitation Daymet netCDF unless it is manually changed. Further development of the extraction of netCDF variables and dimensions within the script could be accomplished but would require no small amount of work beyond the scope of this project. Additionally, because this type of high-resolution data is uncommon and the particular use niche, the presented solution is workable and spending the time to universalize the code is not a priority.

Another nuisance coming from the Daymet data being so high-resolution is the clipping process takes roughly 30 minutes to run. If the data being clipped uses multiple shape files, however, the raw-script-version takes 30 minutes to run the first shapefile, but each subsequent shapefile takes less than a minute each.

**References**

U.S. Geological Survey, 2023, Watershed Boundary Dataset, accessed 2024 at URL <https://apps.nationalmap.gov/downloader/#/>

Thornton, M. M., Shrestha, R., Wei, Y., Thornton, P. E., & Kao, S.-C. (2022). Daymet: Annual Climate Summaries on a 1-km Grid for North America, Version 4 R1. ORNL DAAC, Oak Ridge, Tennessee, USA. <https://doi.org/10.3334/ORNLDAAC/2130>