6.0 [Actual Step #4] Masking Raster Data Using Polygons from Census Data

6.1 Defining and reading geometries

The script below masks three raster files by using a vector shapefile's polygonal outline. The census file below contains an outline for Philadelphia County, the study area for this analysis. The first two lines of code create two lists: *input_files* and *output_files*.

```
# File paths
input_files = [landcover_reprojected, treecover_reprojected, landsat_reprojected]
output_files = ["land_cover_mask.tif", "tree_cover_mask.tif", "land_surface_temp_mask.tif"]

# Read the geometry shapes from shapefile
with fiona.open(census_reprojected, "r") as shapefile:
    shapes = [feature["geometry"] for feature in shapefile]
```

The code uses the Fiona Python library to open the census reprojected shapefile and reads it into a variable named *shapefile*. Then the shapefile variable has the column for geometry called within each feature or row of the shapefile. This data is then stored in the variable *shapes*.

6.2 Masking raster datasets

The code then starts a loop by using the zip function to loop through the lists *input_files* and *output_files* simultaneously with the variables named *input_path* and *output_path*. The Rasterio Python library is used to open the *input_path* variable as it is being looped through the *input_files* list. It then stores the *input_path* variable as the *src* variable while under "with". Then the variables *out_image* and *out_transform* are set equal to the result of the **rasterio.mask** function. The *out_image* variable stores the masked raster data as a NumPy array. The *out_transform* variable stores the updated transformation matrix for the clipped raster. The function uses *src* for the input file argument, *shapes* for the geometry data argument, and crop=True for the argument that crops the *src* raster data to the geometry extents of *shapes*, which contains the census vector shapefile. Finally, the *src* meta data for the original input rasters are stored in the new variable *out_meta*.

```
# Loop through each raster, apply mask, and save the output
for input_path, output_path in zip(input_files, output_files):
    with rasterio.open(input_path) as src:
        # Mask the raster with the shapefile geometries
        out_image, out_transform = rasterio.mask.mask(src, shapes, crop=True)
        out_meta = src.meta



   # Update metadata
    out_meta.update({
        "driver": "GTiff",
        "height": out_image.shape[1],
        "width": out_image.shape[2],
        "transform": out_transform
    })
```

6.3 Saving output files

The above function updates the *out_meta* variable to reflect the new metadata output of the masking function. The driver makes sure the new file type is a Geotiff, the height argument and width arguments takes its data from the geometry of the new *out_image* variable from the mask function result. The transform argument uses the *out_transform* variable which also uses the result data from the masking function.

```
with rasterio.open(output_path, "w", **out_meta) as dest:
    dest.write(out_image)
```

Rasterio.open is used to open and write to the *output_path* and *out_meta* variables data as the variable “dest”. The variable *out_image* is then written or added to the *dest* variable which is the same as the *output_path* variable.

```
print(f'{input_path} has been masked and saved to {output_path}')
```

This print statement prints out the input_path and output_path names every time a loop is completed. For our study, the loop will be completed three times.

6.4 Masking with multiple shapefiles

If you want to construct a script where multiple vector shapefiles are used to mask a raster dataset, you are able to do so by adding *shapes* as a variable at the beginning of the script.

```
input_files = [landcover_reprojected, treecover_reprojected, landsat_reprojected]
output_files = ["land_cover_mask.tif", "tree_cover_mask.tif", "land_surface_temp_mask.tif"]
shapes = [census_reprojected, blockgroup_reprojected, block_reprojected]
```

The for looping mentioned in 6.2 would be moved up a few extra lines to include the reading geometry section of the code. Variable *shapes* would also be added into the for loop and **zip()** function. The rest of the script remains the same, with the exception of census_reprojected from the previous code being changed to *shapes* to successfully complete the loop.

```
for input_path, output_path, shapes in zip(input_files, output_files, shapes):
    with fiona.open(shapes, "r") as shapefile:
        shapes = ([feature["geometry"] for feature in shapefile])

    # Apply mask to raster
    with rasterio.open(input_path) as src:
        out_image, out_transform = rasterio.mask.mask(src, shapes, crop=True)
        out_meta = src.meta

        # Update metadata to reflect the new dimensions and transform
        out_meta.update({
            "driver": "GTiff",
            "height": out_image.shape[1],
            "width": out_image.shape[2],
            "transform": out_transform
        })

    # Save the masked raster to the output file
    with rasterio.open(output_path, "w", **out_meta) as dest:
        dest.write(out_image)
```

6.5 Exercises

Exercise 1 (Easy): A member of a community activist group is awaiting their result for the 2023 copy of the land cover dataset to assess how the census tract has changed in the past year and the impacts it might have to the local nature preserve at the northern end of the census tract. However, the dataset arrived unmasked and includes a larger area than the census tract. Mask the land cover dataset to only include data from within the census tract.

Exercise 2 (Advanced): State Representative Margaret Croke and Congressman Mike Quigley have established a joint cooperative to improve tree canopy cover within their respective districts. However, their districts don't entirely overlap with each other and some areas would require finance from a different member of legislature. Mask the tree canopy dataset using both Margaret's and Mike's districts to only include areas that fall under both of their districts. Hint: Look to section 6.4 for guidance.