A Cellular Automata Markov (CAM) model for land use change prediction using GIS and Python

by

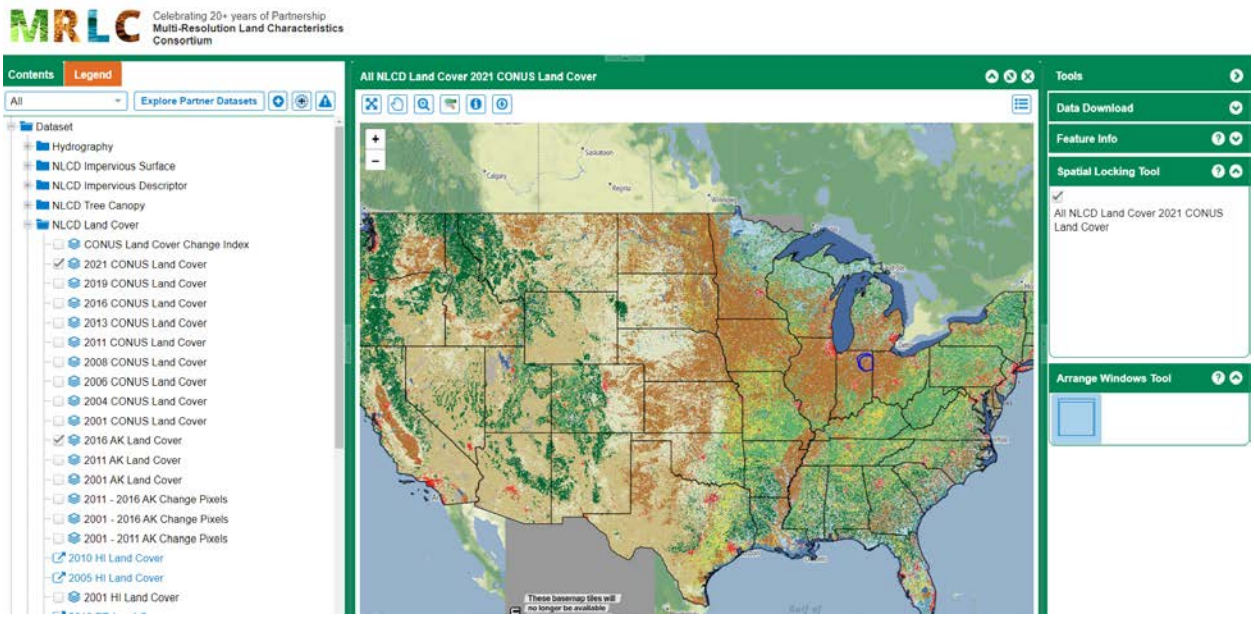
Angelos Alamanos

Supporting Information for the model execution

This file provides guidance for the basic steps of the analysis in GIS, as well as a description of the code.

Input Data Download

The first step for this application is to get spatial data of land uses. These were obtained from the United States Geological Survey (USGS) website (USGS, 2021), as shape files for the years 2006, 2011, 2016 and 2021 for Cedar Creek (Indiana, USA). To get these data, the website uses the MRLC NLCD Viewer web application, which is an interactive map allowing the user to select the study area (blue circle in the Figure below) and download the specified data from the left-side menu. So, the data of Figure 1 were downloaded.



The downloaded datasets look like this for each year: 

The data resolution (cell size (X,Y)) is 30x30.

The default coordinate projected system is GCS_WGS_1984.

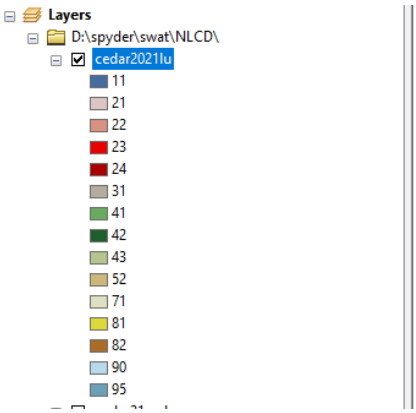
For this example, we changed this into the NAD_1983_UTM_Zone_16N because the DEM we used was in NAD_1983_UTM_Zone_16N, so we need consistency, i.e. all layers to have the same coordinate projected system.

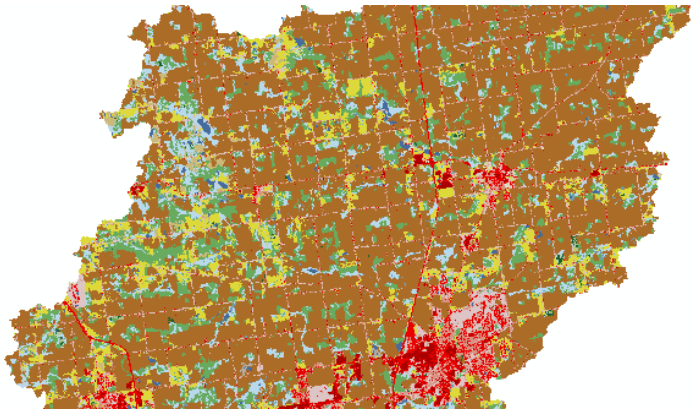
Input data preprocessing

We used the Digital Elevation Model (DEM) of the watershed (CCW) as a 'mask' in GIS (ArcMap 10.7.1), to *clip* the full land use map into just the part referring to our watershed. So, we got the land uses for only the CCW.

[files: cedar2001lu, cedar2006lu, cedar2011lu, cedar2016lu, cedar2021lu].

These look like this for our watershed:





All these values in the layer legend in the left refer to the classes of land uses, which contain many similar land cover types. These are explained in the file: *NLCD_landcover_legend_2018_12_17_FU4zwptaXAXzmpmpNBl3* (which comes with the data download from USGS):

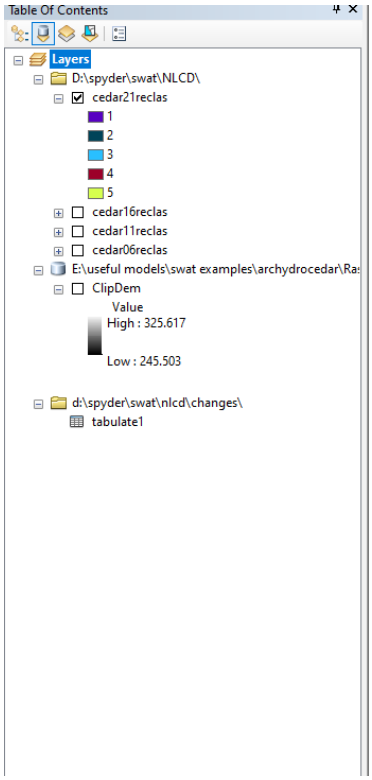
In order to reduce the computational load and effort, and handle easier the land use changes from year to year these land use categories were reclassified in GIS, with the Reclassification command, as follows:

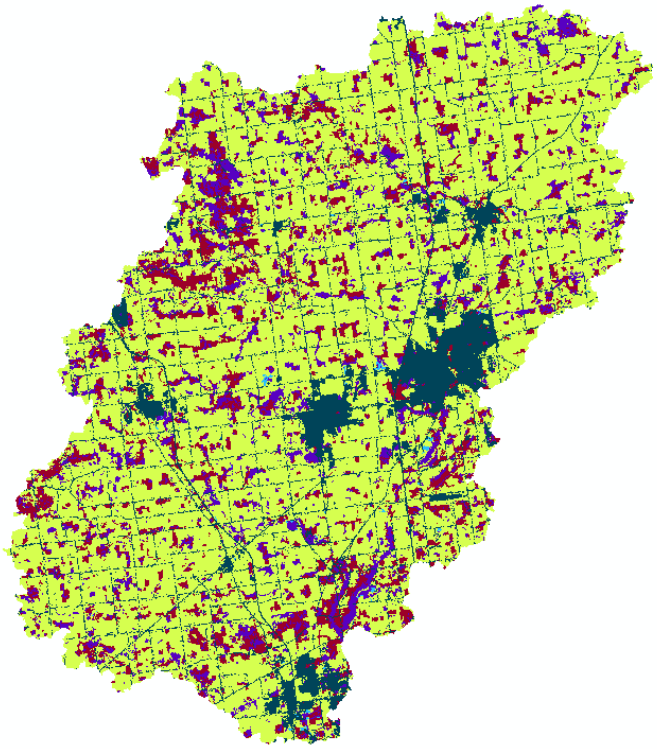
New Land Use class	Includes the following 'old' classes
'Water': 1	11 + 12 + 90 + 95
'Urban': 2	21+ 22 + 23 + 24
'Barren Land': 3	31
'Forest': 4	41 + 42 + 43 + 51 + 52 + 71 + 72 + 73 + 74
'Crops': 5	81 + 82

Please note that this reclassification is indicative for the example, and there can be many other ways, depending on the level of the detail and the specific land use categories one might want to study.

The reclassified files are the: cedar01reclas, cedar06reclas, cedar11reclas, cedar16reclas, cedar21reclas (as shown in the screenshot below).

And they contain less land cover categories, as follows:





<u>The Cellular Automata Markov (CAM) model</u>

Creating a Cellular Automata Markov (CAM) model for land use change prediction involves the estimation of the transition probability matrix, and simulation of land use changes over time. The CAM model can be mathematically represented as follows:

$$L_{t+1} = P_{ij} \cdot L_t \text{ , for land use types i,j = 1,2,…,n} \quad (1)$$

Where L_t and L_{t+1} are the land use maps at the year t and t+1 respectively, and P_{ij} is the transition probability matrix expressing the probability of each cell (pixel) to change from the land use type i in the year t to the land use type j in year t+1. So, this matrix can be expressed as follows:

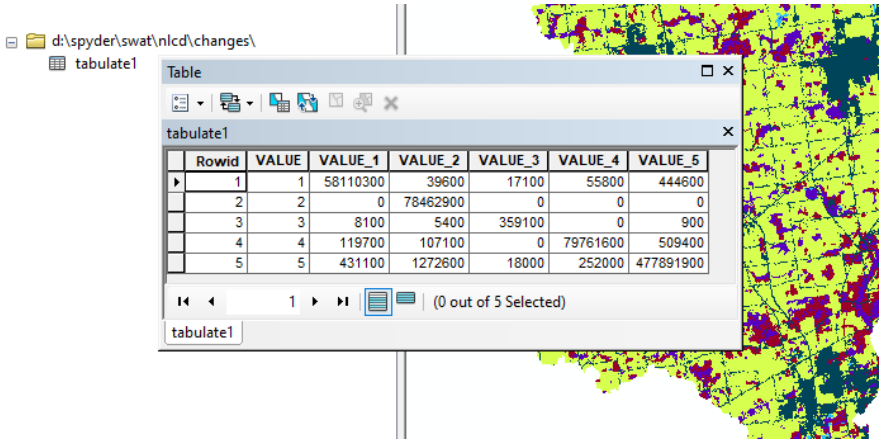
$$P_{ij} = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \cdots & & & \cdots \\ P_{n1} & \cdots & \cdots & P_{nn} \end{bmatrix}, \text{ with } 0 \le P_{ij} \le 1, \text{ and } \sum_{i,j=1}^{n} P_{ij} = 1 \quad (2)$$

Practically, the transition probability matrix is estimated by “cross tabulation” of two maps from different years (t, t+1), and it determines the probability of a pixel in a land-use class i to change into another class j during that time. To estimate these probabilities, we need first to know the number of pixels that changed from each land use i to another type j, over time. Namely, this process returns each element of the change (transition change) matrix (not as probability though, i.e. not the P_{ij} matrix, just as cells changed).

These changes were calculated within GIS with the "*Tabulate Area*" tool (Spatial Analyst Tool → Zonal → Tabulate Area), using the historic data (reclassified land use maps of 2006, 2011, 2016, 2021).

In GIS, this command works as follows:

- Input raster or feature zone data = the 2006 land cover (values 1,2,3,4,5).
- Input raster or feature class data = the 2011 land cover (values 1,2,3,4,5).
- We need to ensure that each land use category in these maps is assigned a unique numerical code (e.g., 1 for Water, 2 for Urban, etc. – same for each map), which corresponds to the keys in the change matrix.
- The output table is a 5x5 value table.



Each cell in the table represents the count of changes (pixels) from the land use category of the corresponding row that changes into the land use category of the corresponding column. For example, the value in cell (row = 1, column = 3) represents the count of changes (pixels) from land use category 1 to category 3 (i.e., 'Water' that became 'Barren Land').

The 'tabulate' table is actually the change matrix, and is interpreted as follows:

Water to Water	Water to Urban	Water to Barren Land	Water to Forest	Water to Crops
Urban to Water	Urban to Urban	Urban to Barren Land	Urban to Forest	Urban to Crops
Barren Land to Water	Barren Land to Urban	Barren Land to Barren Land	Barren Land to Forest	Barren Land to Crops
Forest to Water	Forest to Urban	Forest to Barren Land	Forest to Forest	Forest to Crops
Crops to Water	Crops to Urban	Crops to Barren Land	Crops to Forest	Crops to Crops

This process was repeated for all time steps of the historic data, i.e.:

- tabulate1 = 2006 to 2011
- tabulate2 = 2011 to 2016
- tabulate3 = 2016 to 2021,

and as we proceeded into the predicted land uses (details below), we also estimated the respective Change Matrices (tabulate area) for each one:

- tabulate2026to31
- tabulate2031to36
- tabulate36to41
- tabulate41to46
- tabulate46to51

At this stage we have completed our data preparation, we have change matrices for each pair of years we want to analyze (e.g., from 2006 to 2011, etc.), and we can get the final Transition Probability Matrices (Equation 2) from these change matrices. This requires 2 steps:

1. Normalize the Change Matrices: Calculate the total number of changes (transitions) from land use category i to any other land use category j. This is done by summing the values in row i of the change matrix. Then, we divide each cell in the change matrix by the corresponding total to normalize the matrix. This will give us the transition probabilities.
2. Construct the Transition Probability Matrix by populating it with the normalized values from the change matrix. The cells in the transition probability matrix (i, j) will represent the probability of a cell transitioning from category i in the earlier year to category j in the later year.

A Python script was used (Spyder, Anaconda) to assist us with this process.

It uses the results of the GIS "tabulate areas", normalizes their values and returns the Transition Probability Matrices for each year studied.

Indicatively, the resulted Transition Probability Matrices for each time-step are (pasted here as the arrays of the script:

 # Results:
transition_probability_matrix1 (2006 to 2011)

```
[[0.9905   0.0007   0.0003   0.001    0.0076]
  [0.         1.         0.         0.         0. ]
[0.0217   0.0145   0.9614    0.      0.0024]
[0.0015   0.0013   0.        0.9909   0.0063]
[0.0009   0.0027   0.        0.0005   0.9959]]
```

---

transition_probability_matrix2 (2011 to 2016)

```
[[0.99       0.0005     0.0005     0.0015   0.0075]
 [0.           1.          0.         0.        0.   ]
 [0.0434    0.0137    0.9384     0.0046     0.   ]
 [0.0015    0.001      0.         0.9907     0.0069]
 [0.0008   0.0028      0.         0.0002     0.9962]]
```

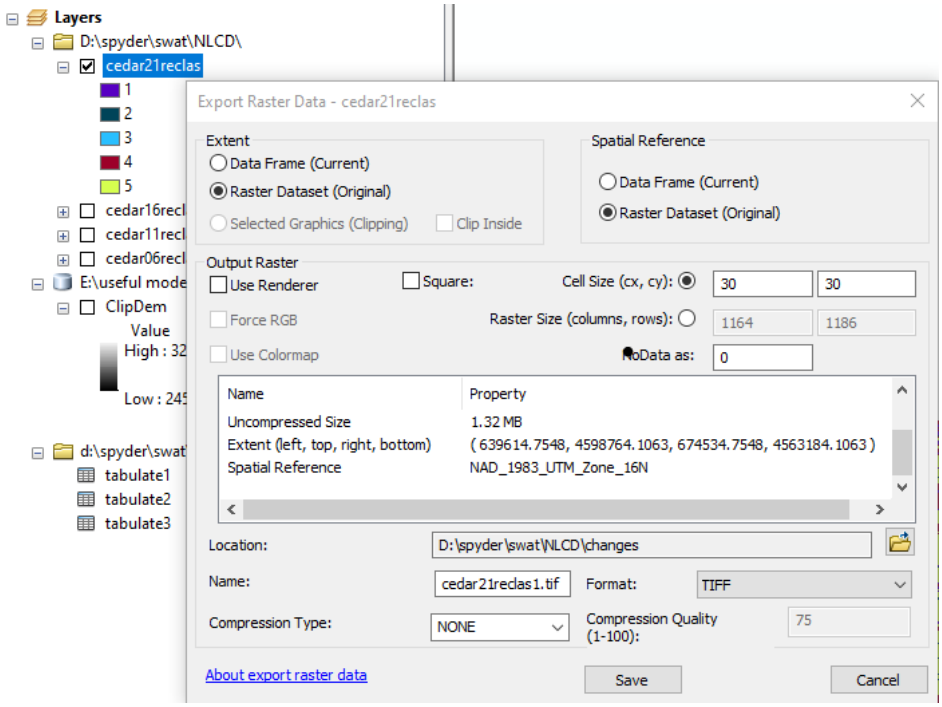
---

transition_probability_matrix2 (2016 to 2021)

```
[[0.9984   0.0005   0.0006   0.0005   0.0001]
 [0.       0.9998   0.       0.       0.0002]
 [0.0089   0.0201   0.9284   0.0425   0.    ]
 [0.0004   0.0021   0.       0.9952   0.0023]
 [0.0004   0.0024   0.0015   0.       0.9956]]
```

Having the Transition Probability Matrices, the CAM model can be applied to predict future land uses, as described in Equation 1, using the results obtained for Equation 2 (P_{ij}). The L_t in Equation 1 is going to be the historic land use map.

A necessary step at this stage is to save our historic land use maps as .tiff files (they are currently raster folder- files, but we need them as .tiff files to be easily workable, because the Python script reads them as tiff files to execute the Equation 1). The next screenshot shows how we save the raster layer file as tiff by 'exporting raster data'.



The Python script uses:

- The reclassified tiff file of the historic land use (L_t in Equation 1).
- The Transition Probability Matrix (P_{ij} in Equation 1). This is the result of the previous step of the script (normalizing the 'tabulate area' output). In the script the Transition Probability Matrix (P_{ij}) is inserted manually because if we read it from the previous step, there are usually reading file issues and it is more difficult to control its numeric format with several decimals. We also need to be sure that the conditions of Equation 2 apply, so it is easier to do this manually.

These are showed in the script indicatively for the prediction from 2016 to 2021.

```
print(transition_probability_matrix3)



##################### 2016 to 2021 ############################

import numpy as np
import os
```

For the prediction of any other time-step, the process is exactly the same.

The user needs only to change the:

- input data of the change matrices

```
# Using the Change Matrices (tabulate areas) from ArcGIS (replace with your own values)

import numpy as np

# Example change matrix for a specific year change (e.g., from 2006 to 2011)
change_matrix_06to11 = np.array([
    [58110300, 39600, 17100, 55800, 444600],   # From Water (1)
    [0, 78462900, 0, 0, 0],  # From Urban (2)
    [8100, 5400, 359100, 0, 900],   # From Barren Land (3)
    [119700, 107100, 0, 79761600, 509400],     # From Forest (4)
    [431100, 1272600, 18000, 252000, 477891900]   # From Crops (5)
])
```

- the working directory and the input files path

```
# Define the data directory and file paths
data_directory = r'D:\your\path'
land_use_2016 = r'D:\your\path\cedar16reclas1.tif'
```

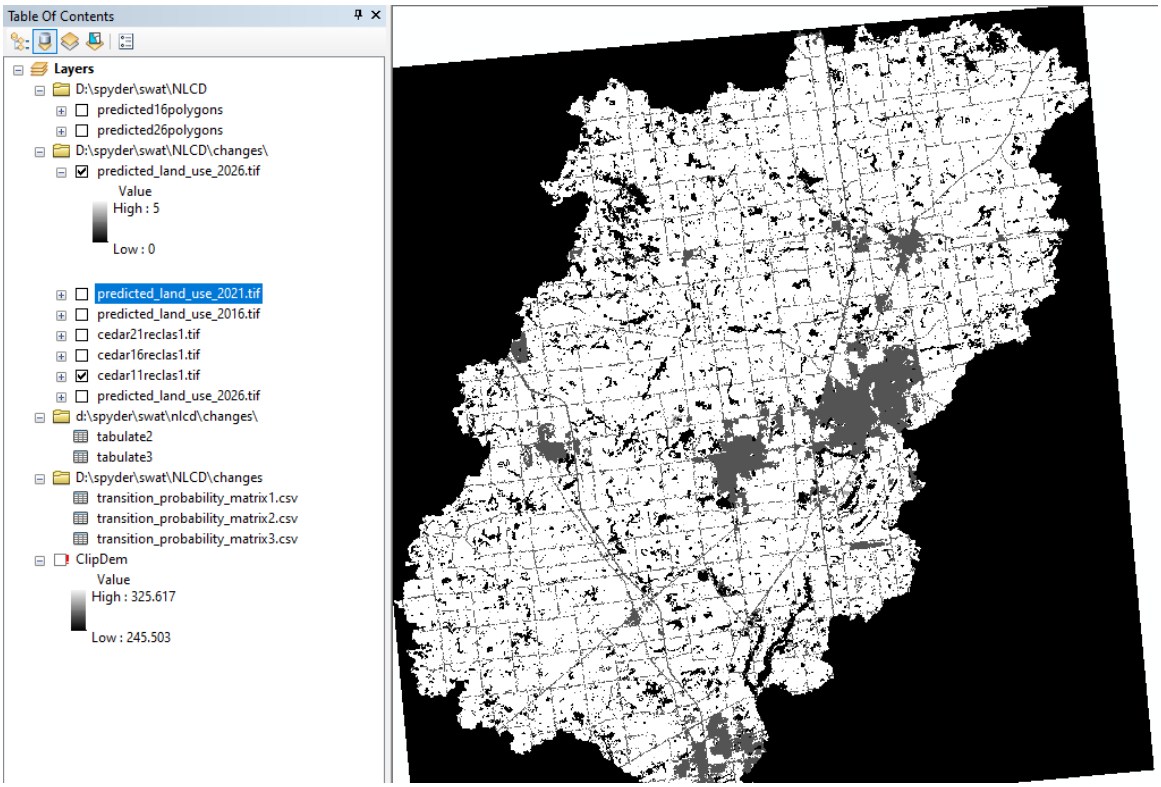
- to define the transition probabilities matrix manually, to ensure correct format, avoiding thus reading errors. Next, note that the script has a step to check that the probabilities are ‘valid’, i.e. all values between 0 and 1 and each row adds up to 1.0 exactly – otherwise the script warns up with the message “Invalid transition probabilities” (because it will not work).

The model defines a function to apply the transition based on the probabilities we inserted, and with these, it goes to each cell of the current land use map, and based on the transition probabilities, it converts it to another land use, or maintains the existing land use.

The output file is the predicted land use map, which uses the same coordinates and georeferenced system as the input land use file.

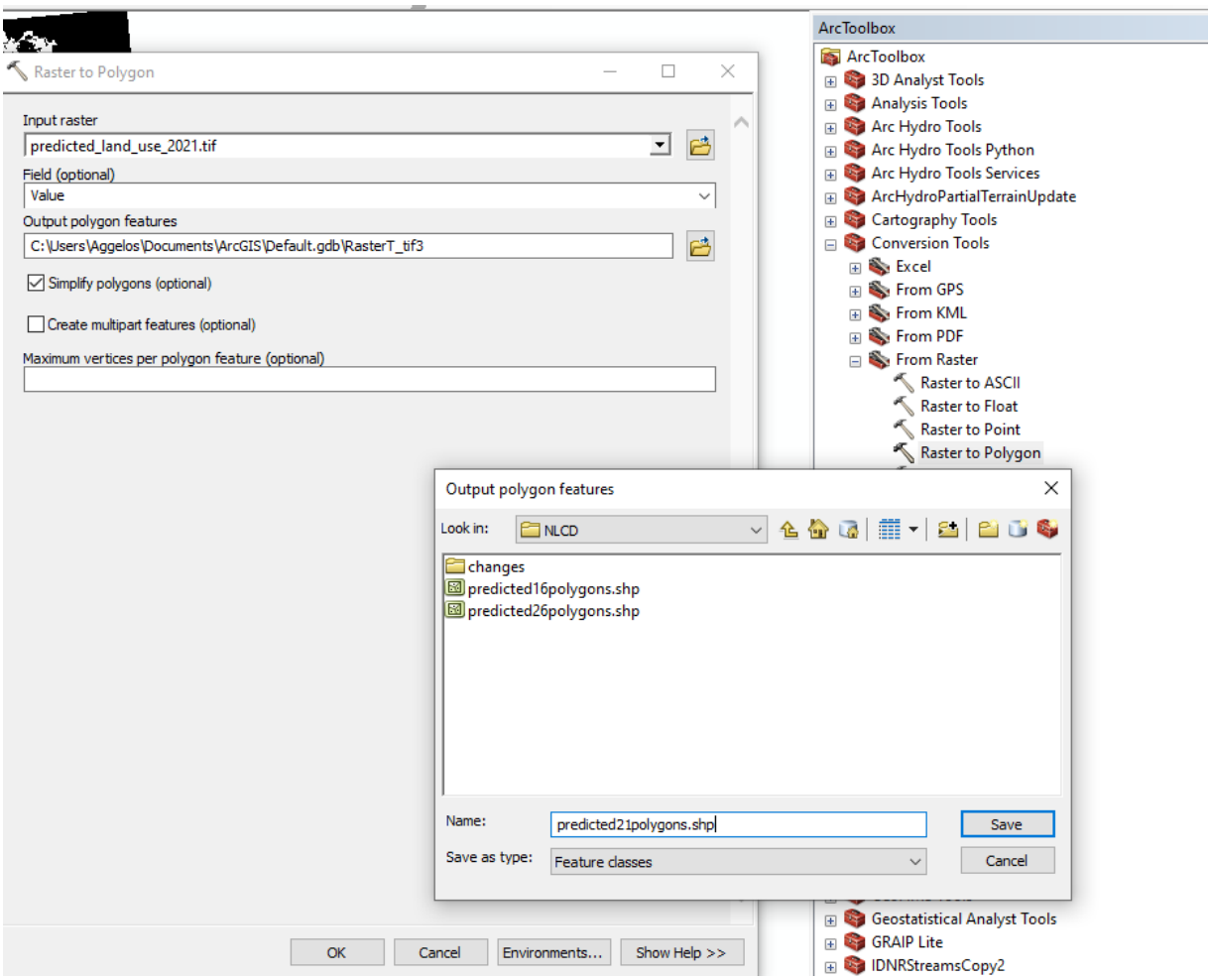
The script creates a tiff file for this output, i.e. the L_{t+1} of Equation 1, and saves it inside our pre-specified working directory folder.

From there we can open it in ArcMap:



We can convert this "high-low" tiff output into a file with an attribute table, showing the land use categories, as we want them, i.e. classes of 1,2,3,4,5.

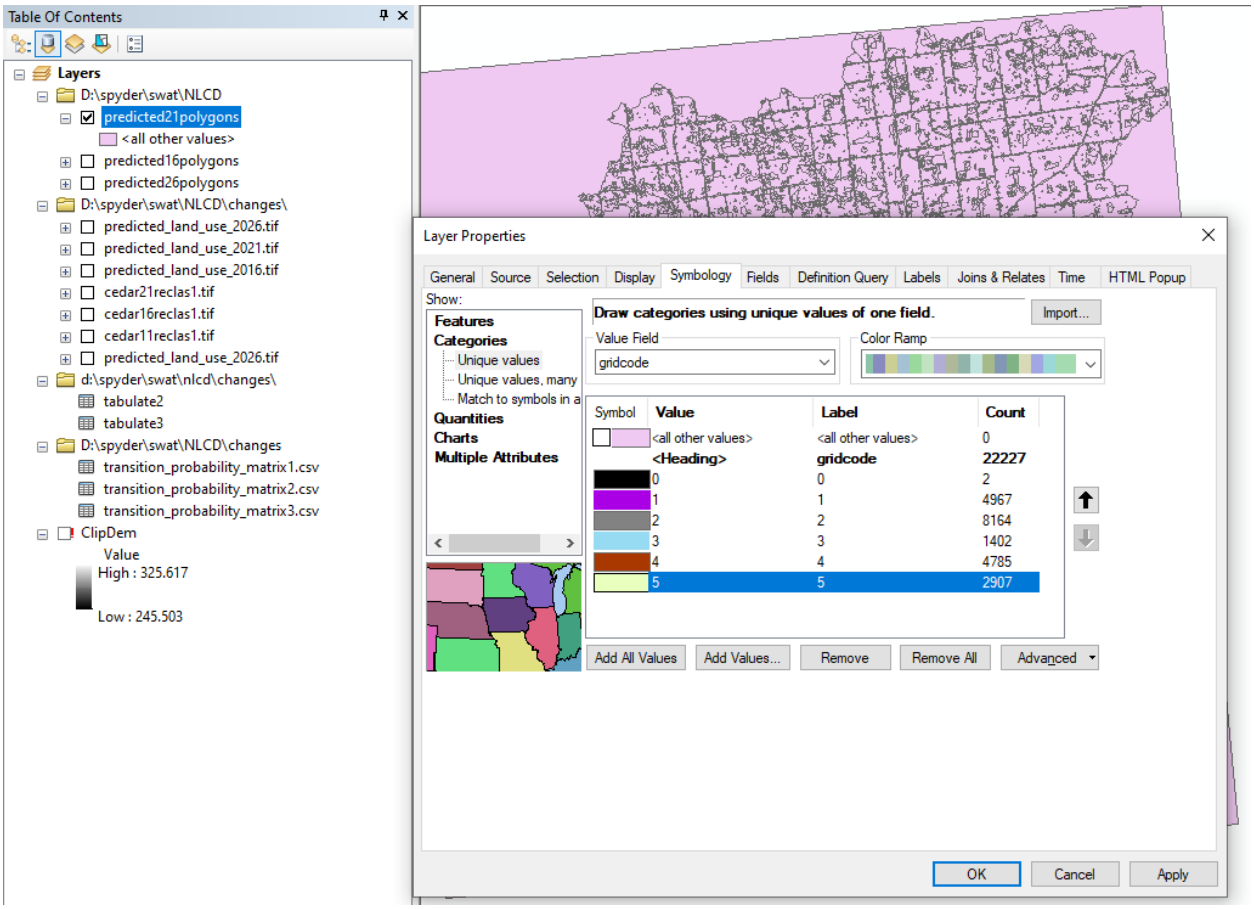
This can be achieved with the "raster to polygon" command. This converts the predicted land uses tiff (L_{t+1}) into a polygon file (we save it as a shp, as shown below):



In this new polygon file we saved, we can open it and do:

Right click → Properties → Symbology → Category / Unique values → Add all values

(and change the colours as desired).



This results in the final land use map (L_{t+1}) with the land use categories as we have defined them from the beginning.

This process in the script and GIS can be repeated as many times necessary, for the time-steps we need to predict.

It is recommended that we copy and paste the example script as many times as necessary per time-step, rather than editing it always by putting the new data names and transition probability matrices. This way, we can minimize mistakes and errors, and secure a template for the process.

Validation

Validating the predicted land use maps is an essential step to assess the accuracy of the model and make future predictions 'safely'. In this case, we can use tests such as statistical techniques based on:

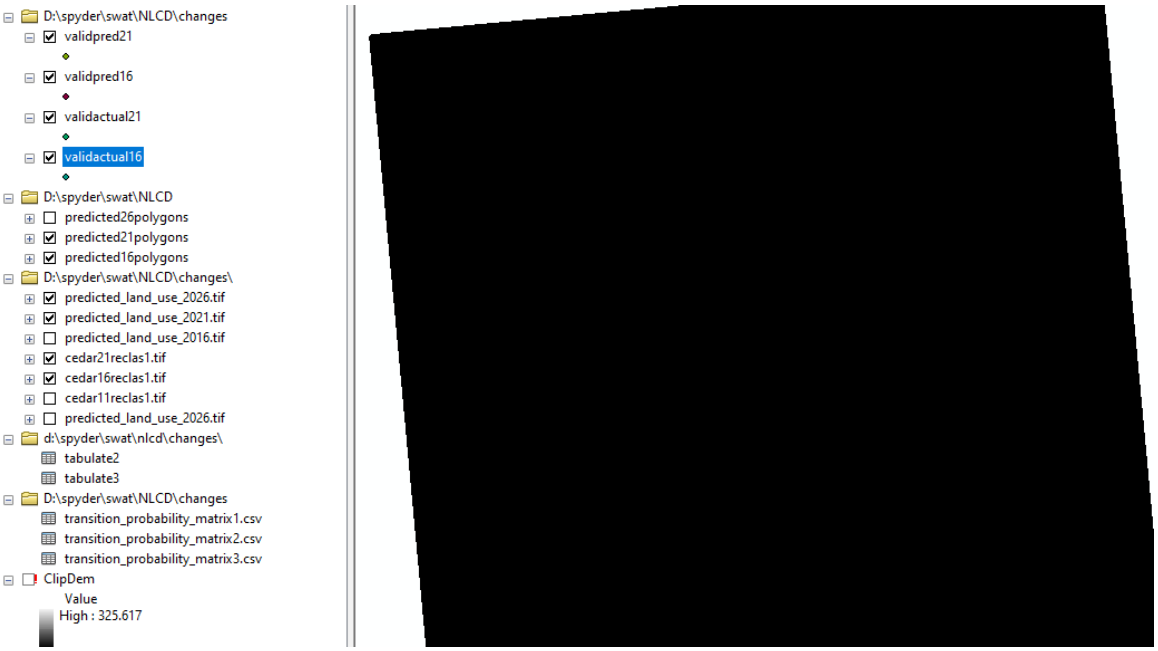
- Percentage of Accurate Results (Overall Accuracy):

Overall accuracy is a straightforward measure that calculates the percentage of correctly classified pixels compared to the total number of pixels. It provides a clear and easily interpretable metric of the model's accuracy (0 = bad, 1 = perfect).

- Mean Absolute Error (MAE): It quantifies the magnitude of errors between predicted and actual land use values. Lower MAE values indicate better model performance (closer to 0 is perfect).
- Root Mean Square Error (RMSE): It measures the square root of the average squared difference between predicted and observed values. It penalizes larger errors more heavily than smaller ones. Lower RMSE values indicate better model performance (closer to 0 is perfect).
- Kappa (κ): Cohen's K measures the level of agreement between two raters or classifiers, often used in the context of classification tasks like land use mapping. It quantifies the agreement between the predicted and true labels while taking into account the possibility of agreement occurring by chance.

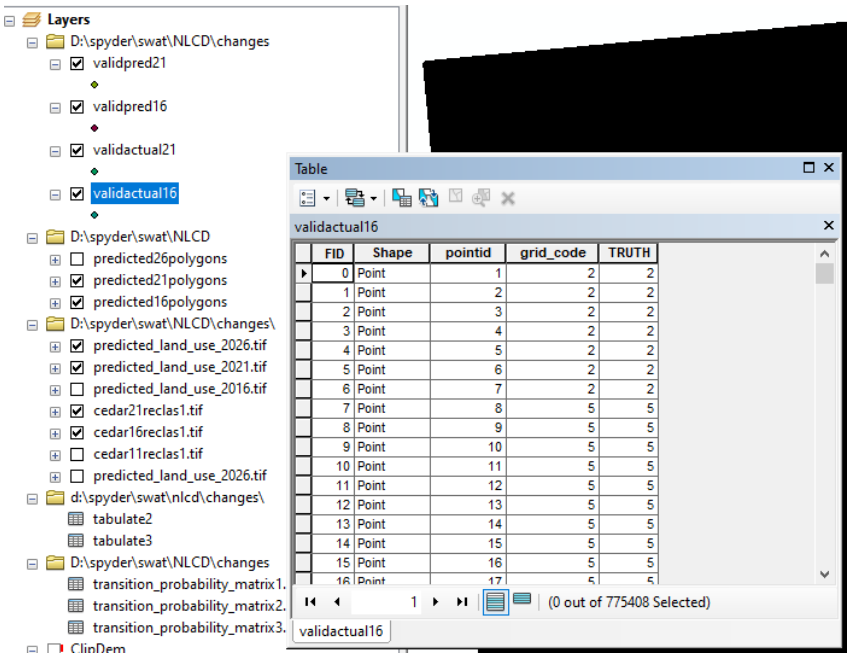
<u>Data preparation:</u>

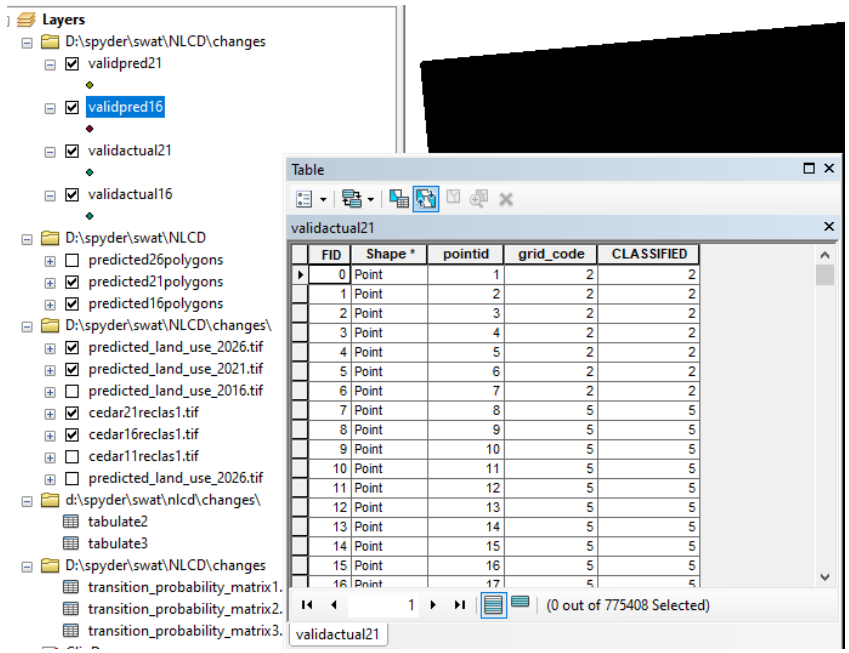
Within ArcGIS → "Raster to Points" command, so we create the actual (truth) point datasets, and the predicted (classified) point datasets for the years we have. That way, the script can use these as inputs to estimate the accuracy. [The next screenshot shows how this first step looks (an image full of points!)]



We can add fields to their Attribute Tables.

So, for example here we added a column "TRUTH" in the actual dataset, and a column "CLASSIFIED" in the predicted datasets. These were equal to their grid_code which stand for the land use categories in each case (i.e, the categories 1,2,3,4,5).





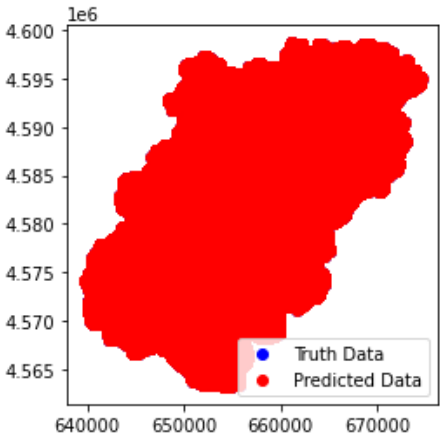
The validation script works as follows:

1. Opens the actual and predicted datasets (named “validactual” and “validpred” respectively).

Please make sure you replace the paths to your data:

```
########################  VALIDATION   ################

import geopandas as gpd
from sklearn.metrics import accuracy_score, mean_absolute_e
import numpy as np

# Paths to truth and predicted point feature classes
truth_path = r'your\path\validactual21.shp'
predicted_path = r'your\path\validpred21.shp'
```

2. Rounds their spatial coordinates of each point to 6 decimal places to ensure their correct merging, based on the spatial coordinates (in case we want to merge them based on each point's coordinates). This step is necessary, as the calculations will be executed in a single file among its columns of its attribute table. So, we merge them based on their spatial coordinates.
3. Generates a unique identifier (unique_id) based on spatial coordinates to ensure that the same points are compared.
4. We will use this unique_id to merge the actual and predicted datasets into a single dataset, in order to calculate from it the statistical metrics we want for our validation.
5. The script offers then a number of checks to ensure that the datasets are properly merged, based on the correct points, and they look as expected. These checks include:
    a. Checking if the datasets have the same number of points.
    b. Checking if the datasets have the same (common) points.
    c. Checking the datasets' data and columns to ensure that we are comparing the correct ones and the merged file looks as expected.
    d. Plots the truth_data and predicted_data points together, to ensure they are the same points (showing them overlapping actually, as below). e.g.:



6. Estimates the following accuracy metrics:

- Accuracy = accuracy_score
- MAE = mean_absolute_error
- RMSE = mean_squared_error RMSE
- Kappa = cohen_kappa_score
- Confusion matrix and associated statistics

PLOTS

Finally, the script provides two kinds of plots:

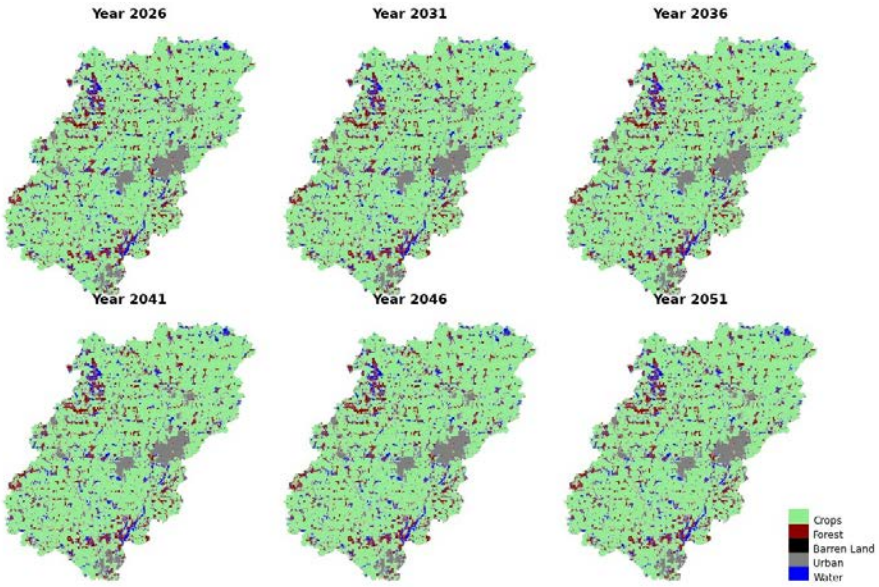
1. A Figure with multiple maps (e.g. the predicted maps that were generated).

The user can save these results to a path (change to “your path”), and the script will read them from there.

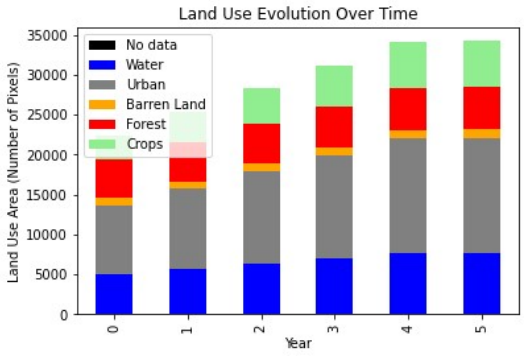
Also, the years corresponding to each month and the land use categories with their colours are defined manually by the user (along with other details of the map, e.g. fonts, size, spaces between maps):

```python
import geopandas as gpd

# Paths to the shapefiles for each year
shapefile_paths = [
    r'your\path\predicted26polygons.shp',
    r'your\path\predicted31polygons.shp',
    r'your\path\predicted36polygons.shp',
    r'your\path\predicted41polygons.shp',
    r'your\path\predicted46polygons.shp',
    r'Dyour\path\predicted51polygons.shp',
]

# Corresponding years
years = [2026, 2031, 2036, 2041, 2046, 2051]

# Land use categories and their colors
land_use_colors = {
    0: 'white',
    1: 'blue',
    2: 'gray',
    3: 'black',
    4: 'darkred',
    5: 'lightgreen',
}


# Land use labels for the legend
land_use_labels = {
    0: 'No data',
    1: 'Water',
    2: 'Urban',
    3: 'Barren Land',
    4: 'Forest',
    5: 'Crops',
}
```

The result map from the code looks like this for our example:



2. A plot showing the predicted areas, as pixels' count. The user needs to change the same things as in the previous plot (path to the data, years, land use categories, colours, map layout details) and the result looks like this for our example:



Please note that for both plots, the script takes the land use categories from the "gridcode" field of the attribute table of the saved shape files. If the land use categories are in a different field, then you can modify this accordingly:

```
259     ax = axes[i // 3, i % 3]
260
261     # Plot the GeoDataFrame with specified colors and legend
262     for land_use_code, color in land_use_colors.items():
263         gdf[gdf['gridcode'] == land_use_code].plot(ax=ax, color=color)
264
265     # Set title and remove axes
```

```
329
330     # Count the number of pixels for each land use category
331     category_counts = {land_use_categories[i]: (gdf['gridcode'] == i).sum() for i in land_use_c
332
333     # Append the counts to the DataFrame
334     land_use_counts = land_use_counts.append(category_counts, ignore_index=True)
```