pyGr

A port and extension of the Gradual Rasterization Tool

Background

pyGr is a python port of the gradual zoning algorithm gr, developed to automatically define automatically sized zoned based on population and employment rasters. The model documentation can be found here:

Moeckel, R., Donnelly, R. (2014) Gradual Rasterization: Redefining the spatial resolution in transport modeling. In: Proceedings of the 93rd Annual Meeting of the Transportation Research Board (TRB). Washington, D.C. January 12-16, 2014. [available for download at: http://moeckel.github.io/rm/doc/2014\_moeckel\_donnelly\_trb.pdf]

pyGr also provides two extensions:

1. **Aligning of zones to municipal boundaries**

Due to privacy requirements, statistical data are often only available at a municipal level. These municipalities rarely line up with an ideal grid based zoning. After generating a grid of zones, our tool splits the grid based zones along municipal boundaries, distributing the zone’s value accordingly, and merging zones within a region where necessary to avoid slivers along the municipal boundaries.

1. **Iterative behaviour**

When designing a zoning system, the number of zones in a model needs to be tailored to balance the usefulness and complexity of the model. In the original gr, a threshold must be specified that is used to divide larger zones into smaller ones. Previously this was a manual process of trial and error. With pyGr, the desired number of zones can be specified, and the the zoning algorithm is iterated to find the parameters that best provide the desired number of zones.

pyGr can be run without these extensions, by setting flags in the configuration file.

Design

The tool is split into two scripts, a pre-processing procedure and the zoning algorithm itself.

The zoning algorithm takes as input the following:

1. A polygon shapefile of the study area and metropolitan regions
2. Two raster inputs of the same resolution and size, for population and employment figures respectively.
3. If the land use statistics for each zone are desired, a configuration file is required with the following:

* Input folder: The folder with the land use files (we used ALKIS data)
* Class Field: the attribute representing the land use type
  + Class Values: A list (with a translations) of land use types to be included. Values not in this list will be ignored
* Scaling factors:
  + Comma separated values of the weighting for each land use type when distributing a certain statistic. We only apply scaling to employment in our example. The list must be the same length as the number of land use types in ‘Class Values’, and sum to 1

The pre processing is designed specifically for the data availability that we are working with in developing the CITE Munich Regional Model. However, care has been taken to make many of the steps generic, and will be useful to modellers preparing their data for the zoning algorithm.

Notes on Inputs

In recognition that population, employment, and land use data sources and availability varies widely from use case to use case, the tool is designed to be as accommodating to different data sources and formats as possible. The code is provided as open source, and can be modified to suit the needs of the modeller. As provided, we make the following assumptions:

All shapefiles and rasters need to have an embedded Coordinate Reference System (CRS). Where this is not supplied with the data, it is easily embedded through the use of GIS software.

pyGr assumes that all input files have the same CRS embedded. An update will follow in the future to enable the tool to automatically apply projections between the shapefiles, however this is not yet implemented.

The tool is designed to work with German datasets. As such the AGS ‘Amtlicher Gemeindeschlüssel’ is used as a numerical identifier for each muncipality. Some modifications may be required to work with other regional typologies.

For our zone system in Munich, we take advantage of 100m2 resolution census population data (CITE) to disaggregate municipal population counts, and fall back on ALKIS land use data for disaggregating employment.

Installation and Execution

Our tool requires some external packages and libraries to handle the geospatial operations.

{rasterstats}, rasterio, shapely and Fiona need to be installed, along with the GDAL library.

For details on running the pre-processing script, run:

build\_rasters.py --help

The zoning algorithm is run as followed:

generate\_zones.py {zoning config file} {land use config file}