unrasterize Documentation

Release 1.0.0

Brian Lewis

Jun 11, 2019

Contents

CHAPTER 1

Motivation

Raster data formats have become increasingly popular to represent global population density (see, for example, CIESIN's Gridded Population of the World). But the sheer number of pixels can make working with raster data difficult for certain use cases, especially in live applications where calculations are performed on the fly.

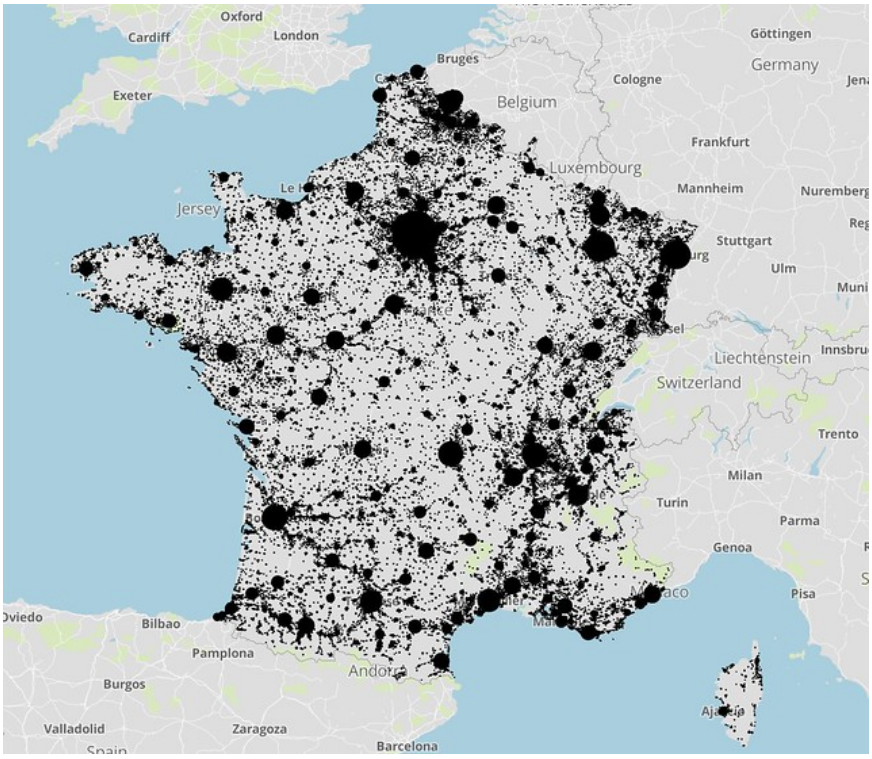
Enter `unrasterize`, a lightweight package to extract representative population points from raster data. The library returns points in vector format (i.e., GeoJSON), selecting certain pixels as "representative" and aggregating nearby population values to preserve the total population.

For a concrete example, consider Encompass, a tool used to measure spatial access to critical social services. Measuring the driving time between every raster pixel and every service provider is computationally infeasible. Unrasterizing the data yields a manageable number of points and allows the massive pairwise distance computation to terminate before the sun becomes a red giant.

Note that some loss of fidelity is inevitable. In fact, that's the point.

CHAPTER 2

Sample Output



CHAPTER 3

Contents

3.1 Usage

3.1.1 Algorithm walkthrough

Let's walk through the algorithm behind the default `Unrasterizer` class. The basic use pattern is as follows:

```
u = Unrasterizer(mask_width=4, threshold=0.5)
representative_points = u.select_representative_pixels(raster_data)
```

`mask_width` indicates minimum number of non-selected pixels between adjacent selected pixels.

`threshold` indicates the minimum value required for a pixel to be selected. Measured in the same units as value (often population per pixel).

First, all pixels with value below the given threshold will be ignored. The remaining pixels are sorted according to their values and considered for selection one by one.

After each point is selected, all points within a circle of radius `mask_width` in the L^∞ norm (i.e., a square in the standard Euclidean norm) of the newly chosen point are masked and removed from further consideration. This process continues until every point is either selected or part of the mask.

The `Unrasterizer` class performs this selection process on the entire raster file, while the `WindowedUnrasterizer` applies the same process across each individual raster block and combines the results into a single array of selected pixels.

3.1.2 Demonstration

Here, the initial red triangle in the west represents pixels that fall below the population threshold. This could represent an uninhabited geographical feature, like a mountainside or part of a lake.

The initial purple pixel indicates the pixel with the highest value. It is chosen first.

All points within a square of radius `mask_width` of the chosen pixel are added to the mask (moving from turquoise to orange to red). These points will never be selected going forward.

Next, the remaining (turquoise) pixel with the highest value is selected and the pixels around it are masked. This process repeats ad infinitum.

3.1.3 Examples

For an example of `unrasterize` in action, see this Jupyter notebook.

3.2 Unrasterize API

The following classes exist to convert raster data to GeoJSON.

For large raster files, the WindowedUnrasterizer is the most memory efficient, with the caveat that it may select some points from adjacent windows that are very close together.

3.2.1 Classes

unrasterize.BaseUnrasterizer

unrasterize.Unrasterizer

unrasterize.WindowedUnrasterizer

3.3 Contributing

View `unrasterize` on GitHub.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search