Pandarus Documentation

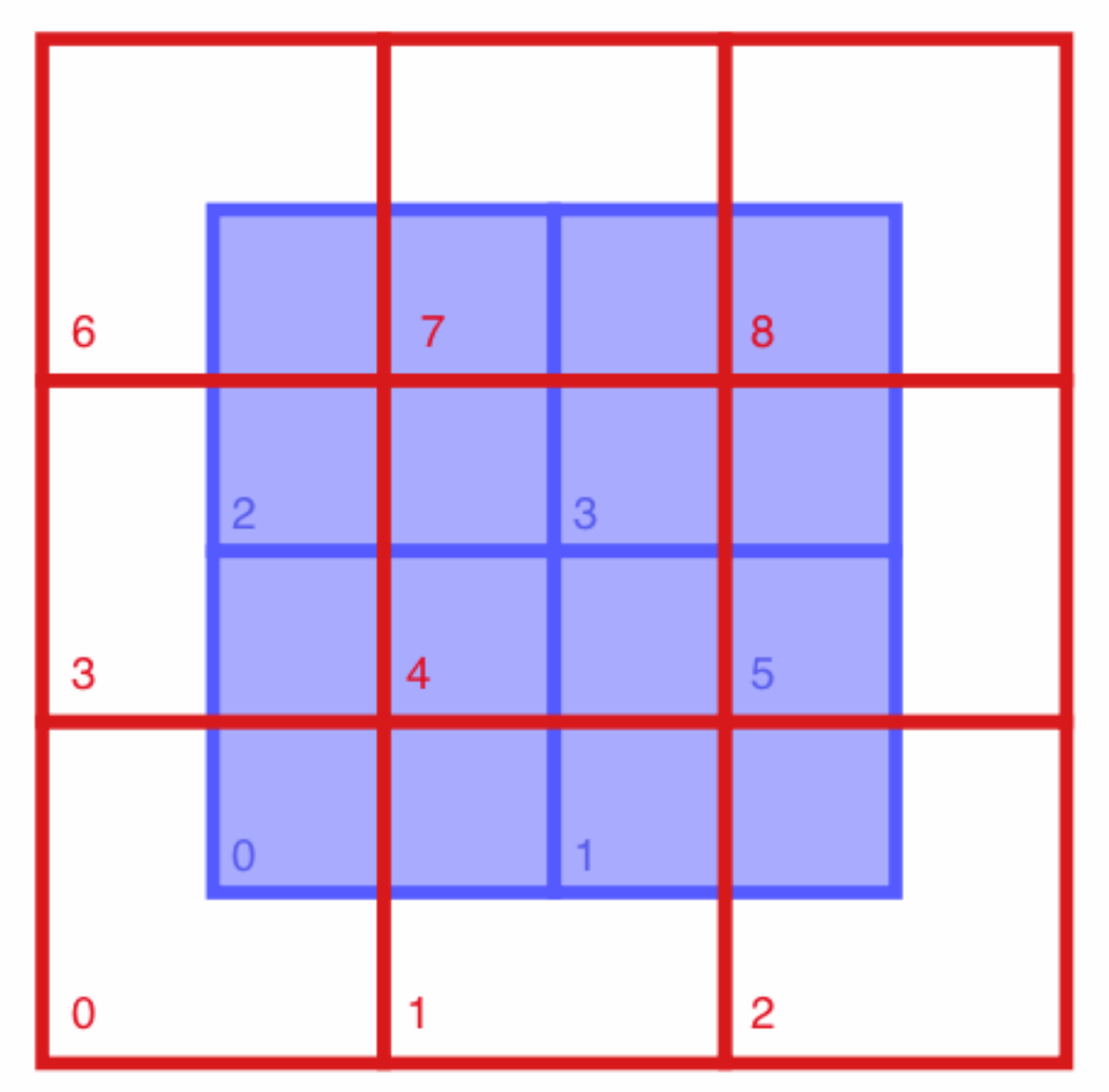
Release 0.4.1

Chris Mutel

April 30, 2015

Contents

Pandarus is software for taking two geospatial data sets (either raster or vector), and calculating their combined intersected areas. Here is an example of two input maps, one in blue, the other in red:



Pandarus would calculate the intersected areas of each spatial unit of both maps, and output the following:

```
{(0, 0): 0.25,
 (0, 1): 0.25,
 (0, 3): 0.25,
 (0, 4): 0.25,
 (1, 1): 0.25,
 (1, 2): 0.25,
 (1, 4): 0.25,
 (1, 5): 0.25,
 (2, 3): 0.25,
 (2, 4): 0.25,
 (2, 6): 0.25,
 (2, 7): 0.25,
```

```
(3, 4): 0.25,
(3, 5): 0.25,
(3, 7): 0.25,
(3, 8): 0.25}
```

The intersected areas are given in square meters. Because Pandarus was designed for global data sets, the Mollweide projection is used as the default projection for calculating areas. Although no projection is perfect, the Mollweide has been found to be a reasonable compromise (e.g. [1])

Warning: Pandarus is still in development, and given how people misuse and even abuse geospatial data, it will certainly fail for some maps and use cases. Please feel free to file bug reports if things don't work as you expect.

[1] Usery, E.L., and Seong, J.C., (2000) A comparison of equal-area map projections for regional and global raster data

CHAPTER 1

Referencing spatial features

There is no standard way to reference vector features or raster cells. Internally, Pandarus iterates over vector features in the file order, and assigns each an integer id starting from zero. For rasters, a similar procedure is followed, again with an incrementing integer id, iterating over cells starting from the bottom left of the raster, and iterating over rows and then columns.

For vector data sets, the label of a data column which uniquely identifies each feature can be given, and Pandarus will translate the integer id to that data column value. Raster data sets will have their integer ids automatically translated to the label `"Cell(x, y)`, where `x` and `y` are the longitude and latitude of the raster cell centroid.

CHAPTER 2

Why Pandarus?

The software matches two different maps against each other, and Pandarus was a bit of a matchmaker himself. Plus, ancient names are 200% more science-y.

CHAPTER 3

Usage

Pandarus installs a command line programs, `pandarus`. It is called in the command shell or terminal:

```
pandarus <map1> [--field1=<field1>] <map2> [--field2=<field2>] <output> [csv|json|pickle]
```

After the program name, you need to give the locations of the two spatial data sets, as well as the name of the file to create for the output:

- `<map1>` is the filepath of the first raster or vector spatial data set
- `<map2>` is the filepath of the second raster or vector spatial data set
- `<output>` is the filepath of the outputted file
- `<field1>` and `<field2>`, if specified, are the names of the columns used to uniquely identify each feature in `<map1>` and/or `<map2>`. For rasters, this value is ignored.
- `csv` or `json` or `pickle`, if specified, is the output format. `json` is the default value.

For example, if I was matching the raster `/Users/cmutel/test.raster` against the shapefile `/Users/cmutel/test.shp`, which had a unique data column `name`, and wanted to create the file `/Users/cmutel/foo.bar`, using the `csv` format, I would enter:

```
pandarus /Users/cmutel/test.raster /Users/cmutel/test.shp --field2=name /Users/cmutel/foo.bar csv
```

3.1 From Python

Pandarus can also be used as a Python library. See the *Controller* technical documentation.

CHAPTER 4

Output formats

The basic output is, for each intersecting spatial unit in the first and second map, a unique ID for each spatial unit and the intersected area, e.g. `('Switzerland', 'Rhine watershed', 42)`. The are three ways of reformatting the data when written to a file.

4.1 JSON

JSON is a data format originally developed for Javascript but now used widely in many programming languages. This is the recommended and default output format for Pandarus. The JSON output format is:

```
[
    [map1 id, map2 id, intersected area],
]
```

4.2 CSV

Pandarus can also export to CSV files. It includes a copy of unicodecsv, so that map ids aren't limited to the ASCII character set. CSV files are UTF-8 encoded, and have the following format:

```
map1 id, map2 id, intersected area
```

4.3 Pickle

Pandarus results can also be serialized into Python pickles with the following format:

```
[
    (map1 id, map2 id): intersected area
]
```

CHAPTER 5

Technical Details

5.1 Computational efficiency

Pandarus is relatively computationally efficient. It uses R tree indices to screen out geometries where no intersections are possible, and uses Python multiprocessing to split work across all available CPUs. Given its use case in regionalized LCA, further optimizations didn't seem worthwhile.

5.2 Wrapping vector and raster data in a common interface

The *Map* object provides a common API for both vector and raster data sources. Vector data is loaded using Fiona, and raster data is loaded using the much less pleasant GDAL library. See the *Map* and *Raster* technical documentation.

Note: The common API is not perfect. For example, GDAL raster geometries are given as WKT strings, but Fiona vector geometries are given as GeoJSON dictionaries (see `pandarus.maps.to_shape`).

5.3 Projection

Projection between coordinate reference systems is done using a wrapper adapted from code by Sean Gillies based on pyproj. See the *Projection* function.

CHAPTER 6

Installation

Pandarus can be installed directly from PyPi using *pip* or *easy_install*, e.g.

```
pip install pandarus
```

Pandarus source code is on bitbucket.

6.1 Requirements

- docopt
- fiona
- GDAL
- progressbar
- pyproj
- Rtree
- shapely

6.2 Easier installation using Canopy

Enthought Canopy is an easy way to install the geospatial dependencies, which can be a pain, especially on Windows. Note that *Rtree* will still have to be manually installed, even when using Canopy. See the rtree docs.

6.3 Running tests

Warning: The current level of testing is, shall we say, inadequate. You have been warned.

To run the tests, install nose, and run `nosetests`.

6.4 Technical reference

6.4.1 Map

6.4.2 Raster

6.4.3 Projection

6.4.4 Controller