fauxgeo Documentation

Release dev

Will Bierbower

August 19, 2015

Contents

CHAPTER 1

Affine Module

CHAPTER 2

Raster Module

CHAPTER 3

RasterFactory Module

CHAPTER 4

Vector Module

CHAPTER 5

Fauxgeo

A python library that generates simple OSGeo-supported rasters and vectors. The primary purpose for this library is to help test geoprocessing functions.

- Free software: BSD license
- Documentation: https://fauxgeo.readthedocs.org.

5.1 Requirements

- Numpy
- Scipy
- Matplotlib
- GDAL
- PyGeoprocessing
- Shapely

5.2 Installation

```
pip install fauxgeo
```

5.3 Features

- Raster Class
- RasterFactory Class
- Vector Class

5.4 Usage

The Raster Class

```
import numpy as np
import gdal

# set arguments
array = np.ones((3, 3))
affine = Raster.create_simple_affine(0, 0, 1, 1)
proj = 4326
datatype = gdal.GDT_Float64
nodata_val = -9999.0

# initialize raster
raster = Raster.from_array(array, affine, proj, datatype, nodata_val)
raster = Raster.from_file('path/to/geotiff')

raster.uri  # equals '/path/to/geotiff'
raster.driver  # e.g. 'GTiff'
raster.dataset

raster.get_band(1)  # returns 2d numpy masked array
raster.get_bands()  # returns 3d numpy masked array
raster.get_nodata()  # returns nodata value
raster.get_shape()  # returns 2-tuple (rows, cols)
raster.get_projection()
raster.get_affine()
raster.get_bounding_box()
raster.get_cell_area()

raster2 = raster1.set_datatype(gdal.GDT_Int32)
raster2 = raster1.set_nodata(-9999)
raster2 = raster1.set_datatype_and_nodata(gdal.GDT_Int32, -9999)

# Operations
raster3 = raster1.align(raster2)
raster3 = raster1.align_to(raster2)
assert(raster3.is_aligned(raster2))

raster4 = raster3 * raster2
raster4 = raster3 / raster2
raster4 = raster3 + raster2
raster4 = raster3 - raster2
raster4 = raster3 ** raster2

raster4 = raster3 * 4.5
raster4 = 4.5 / raster3
raster4 = raster3 + 4.5
raster4 = 4.5 - raster3
raster4 = raster3 ** 4.5

# get and set slices of a raster
print raster4[1:9:2]
raster4[1:9:2] = 6

raster4 = raster3.minimum(raster2)

# returns base rasters with same nodata and datatype
zeros_raster = raster3.zeros()
ones_raster = raster3.ones()
```

```
raster4 = raster3.clip('/path/to/aoi_shapefile')
raster4 = raster3.reproject(epsg_code)

reclass_table = {
    1: 2,
    2: 1
}
raster4 = raster3.reclass(reclass_table)

raster4 = raster3.resize_pixels(pixel_size, resample_method)

# visualization
image = raster4.get_grayscale_image()  # returns PIL Image object

raster.save_raster('/path/to/dst.tif')
del raster  # cleans up temporary file on object deletion or program exit
```

The RasterFactory Class

```
import gdal

# set arguments
shape = (3, 3)
affine = Affine.identity()
proj = 4326
datatype = gdal.GDT_Float64
nodata_val = -9999

# initialize factory
factory = RasterFactory(proj, datatype, nodata_val, shape[0], shape[1], affine=affine)

# create test rasters
test_raster_1 = factory.uniform(5)  # returns raster with 1 band filled with 5's
test_raster_2 = factory.alternating(0, 1)
test_raster_3 = factory.random()
test_raster_4 = factory.horizontal_ramp(1, 10)  # interpolated from 1 to 10 across columns
test_raster_5 = factory.vertical_ramp(1, 10)  # interpolated from 1 to 10 across rows
```

The Vector Class (a work in progress...)

```
from shapely.geometry import *

# set arguments
shapely_object = Polygon([(0, 0), (0, 1), (1, 1)])
proj = 4326

# initialize vector
vector = Vector.from_shapely(shapely_object, proj)
vector = Vector.from_file('/path/to/shapefile')

shapely_object = vector.get_geometry()

vector.save_vector('/path/to/dst.shp')
del vector
```

5.5 Tests

```
python setup.py test
```

5.6 Planning

- Add basic visualization functionality
- **Raster Operations**
  - Reclass
  - Overlay - intersection, union, clip
  - Dissolve
  - Buffer
  - Raster_to_Vector
  - Slope
  - Aspect

CHAPTER 6

Other Useful Info

6.1 Unit Testing with Fauxgeo

```
import unittest
from fauxgeo import RasterFactory, Raster
import pygeoprocessing as pygeo

class TestIO(unittest.TestCase):

  def setUp(self):
    # arguments for raster factory
    pixel_size = 0.083333
    size = 180/pixel_size
    shape = (size, size*2)
    affine = Affine(pixel_size, 0, -180, 0, -pixel_size, 90)
    proj = 4326
    datatype = gdal.GDT_Int32
    nodata_val = NODATA_INT

    # create raster factory
    self.global_int_factory = RasterFactory(
      proj, datatype, nodata_val, shape[0], shape[1], affine=affine)

  def test(self):
    # create test data
    global_raster = self.global_int_factory.uniform(5)

    # arguments for vectorize_datasets
    dataset_uri_list = [global_raster.uri]
    dataset_out_uri = pygeo.geoprocessing.temporary_filename()
    # ... other inputs ...
    pygeo.geoprocessing.vectorize_datasets(...)

    # validate that output matches expected value
    r = Raster.from_file(dataset_out_uri)
    assert(r[0, 0] == 5)
```

6.2 History

6.2.1 0.2.5 (2015-05-28)

- Removed pyproj as a dependency

6.2.2 0.2.4 (2015-05-27)

- Slicing added to Raster class

6.2.3 0.2.3 (2015-04-28)

- Test cases added
- Reverse math operations added to Raster class

6.2.4 0.2.0 (2015-04-21)

- Local math operations added to Raster class, intergrated with Pygeoprocessing

6.2.5 0.1.1 (2015-03-15)

- Raster, TestRaster, and RasterFactory classes added

6.2.6 0.1.0 (2015-01-21)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search