
gazar Documentation

Release 0.0.2

Alan D. Snow

Aug 22, 2017

Contents:

1	GDALGrid	3
1.1	GDALGrid	3
1.2	ArrayGrid	6
2	gazar.grid	7
3	gazar.shape	9
4	Indices and tables	11

A collection of functions to use with GDAL.

Also, the Mongolian word for land, point, place or station ().

GitHub: <https://github.com/snowman2/gazar>

1.1 GDALGrid

A Python wrapper for the `gdal.Dataset()` with additional functionality.

class `gazar.grid.GDALGrid(grid_file, prj_file=None)`

Wrapper for `gdal.Dataset()` with `osr.SpatialReference()` object.

Parameters

- **grid_file** – The grid file to be wrapped.
- **prj_file** (`str`, optional) – Path to projection file.

bounds (`as_geographic=False`, `as_utm=False`, `as_projection=None`)

Returns bounding coordinates for the dataset.

Parameters

- **as_geographic** (`bool`, optional) – If True, this will return the bounds in EPSG:4326. Default is False.
- **as_utm** (`bool`, optional) – If True, it will attempt to find the UTM zone and will return bounds in that UTM zone.
- **as_projection** (`osr.SpatialReference()`, optional) – Output projection for bounds.

Returns (`x_min`, `x_max`, `y_min`, `y_max`) Bounds for the grid in the format

Return type `tuple`

coord2pixel (`x_coord`, `y_coord`)

Returns base-0 raster index using global coordinates to pixel center

Parameters

- **x_coord** (`float`) – The projected x coordinate of the cell center.
- **y_coord** (`float`) – The projected y coordinate of the cell center.

Returns (col, row) - The 0-based column and row index of the pixel.

Return type `tuple`

epsg

`str` – EPSG code

geotransform

`tuple` – The geotransform for the dataset.

latlon

Returns latitude and longitude arrays representing the grid.

Returns

- **proj_lats** (`numpy.array()`) – The latitude array.
- **proj_lons** (`numpy.array()`) – The longitude array.

lonlat2pixel (*longitude, latitude*)

Returns base-0 raster index using longitude and latitude of pixel center

Parameters

- **longitude** (*float*) – The longitude of the cell center.
- **latitude** (*float*) – The latitude of the cell center.

Returns (col, row) - The 0-based column and row index of the pixel.

Return type `tuple`

np_array (*band=1, masked=True*)

Returns the raster band as a numpy array.

Parameters

- **band** (*obj:int, optional*) – Band number (1-based). Default is 1. If ‘all’, it will return all of the data as a 3D array.
- **masked** (*bool, optional*) – If True, will return the array masked with the NoData value. Default is True.

Returns

Return type `numpy.array()` or `numpy.ma.array()`

num_bands

int – number of bands in raster

pixel2coord (*col, row*)

Returns global coordinates to pixel center using base-0 raster index.

Parameters

- **col** (*int*) – The 0-based column index.
- **row** (*int*) – The 0-based row index.

Returns (x_coord, y_coord) - The x, y coordinate of the pixel center in the dataset’s projection.

Return type `tuple`

pixel2lonlat (*col, row*)

Returns latitude and longitude to pixel center using base-0 raster index

Parameters

- **col** (*int*) – The 0-based column index.
- **row** (*int*) – The 0-based row index.

Returns (longitude, latitude) - The lat, lon of the pixel center in the dataset's projection.

Return type `tuple`

proj

func – `pyproj.Proj` – Proj4 object

proj4

str – proj4 string

to_arc_ascii (*file_path*, *band=1*, *print_nodata=True*)

Writes data to Arc ASCII file format.

Parameters

- **file_path** (*str*) – Path to output ascii file.
- **band** (*obj:int*, optional) – Band number (1-based). Default is 1.
- **print_nodata** (*bool*, optional) – If True, it will write out the NoData value for the raster band. Default is False.

to_grass_ascii (*file_path*, *band=1*, *print_nodata=True*)

Writes data to GRASS ASCII file format.

Parameters

- **file_path** (*str*) – Path to output ascii file.
- **band** (*obj:int*, optional) – Band number (1-based). Default is 1.
- **print_nodata** (*bool*, optional) – If True, it will write out the NoData value for the raster band. Default is False.

to_projection (*dst_proj*, *resampling=<Mock id='140403181044816'>*)

Reproject dataset to new projection.

Parameters **dst_proj** (`osr.SpatialReference()`) – Output projection.

Returns

Return type `GDALGrid()`

to_tif (*file_path*)

Write out as geotiff.

Parameters **file_path** (*str*) – Output path for file.

wkt

str – WKT projection string

write_prj (*out_projection_file*, *esri_format=False*)

Writes projection file.

Parameters

- **out_projection_file** (*str*) – Output path for file.
- **esri_format** (*bool*, optional) – If True, it will convert the projection string to the Esri format. Default is False.

x_size

int – size of x dimensions

y_size
int – size of y dimensions

1.2 ArrayGrid

Class for constructing a GDALGrid from an array.

```
class gazar.grid.ArrayGrid(in_array, wkt_projection, geotransform, gdal_dtype=<Mock
                                id='140403181045008'>, nodata_value=None)
```

Bases: *gazar.grid.GDALGrid*

Loads `numpy.array()` into a `GDALGrid()`.

Parameters

- **in_array** (`numpy.array()`) – 2D or 3D array of data.
- **wkt_projection** (`str`) – WKT projection string.
- **geotransform** (`tuple`) – Geotransform for array.
- **gdal_dtype** (`gdalconst()`, optional) – The data type of the *in_array* for GDAL. Default is `gdalconst.GDT_Float32`.
- **nodata_value** (*int or float, optional*) – The value used in the grid for No-Data. Default is None.

`gazar.grid.utm_proj_from_latlon(latitude, longitude, as_wkt=False, as_osr=False)`

Returns UTM projection information from a latitude, longitude coordinate pair.

Parameters

- **latitude** (*float*) – The center latitude.
- **longitude** (*float*) – The center longitude.
- **as_wkt** (*bool, optional*) – If True, will return the WKT projection string.
- **as_osr** (*bool, optional*) – If True, will return the `osr.SpatialReference()` object.

Returns Defaults to the proj.4 string.

Return type `str` or `osr.SpatialReference()`

`gazar.grid.geotransform_from_yx(y_arr, x_arr, y_cell_size=None, x_cell_size=None)`

Calculates geotransform from arrays of y and x coords. Assumes Y max and X min are at [0,0].

Parameters

- **y_arr** (`numpy.array()`) – Array of latitudes or y coordinates.
- **x_arr** (`numpy.array()`) – Array of longitudes or x coordinates.
- **y_cell_size** (*float, optional*) – Y cell size in projected coordinates.
- **x_cell_size** (*float, optional*) – X cell size from projected coordinates.

Returns geotransform: (x_min, x_cell_size, x_skew, y_max, y_skew, -y_cell_size)

Return type `tuple`

`gazar.grid.resample_grid(original_grid, match_grid, to_file=False, output_datatype=None, resample_method=<Mock id='140403181045136'>, as_gdal_grid=False)`

This function resamples a grid and outputs the result to a file.

Based on: <http://stackoverflow.com/questions/10454316/how-to-project-and-resample-a-grid-to-match-another-grid-with-gdal-python>

Parameters

- **original_grid** (`str` or `gdal.Dataset()` or `GDALGrid()`) – The original grid dataset.
- **match_grid** (`str` or `gdal.Dataset()` or `GDALGrid()`) – The grid to match.
- **to_file** (`str` or `bool`, optional) – Default is `False`, which returns an in memory grid. If `str`, it writes to file.
- **output_datatype** (`osgeo.gdalconst()`, optional) – A valid datatype from `gdalconst` (Ex. `gdalconst.GDT_Float32`).
- **resample_method** (`osgeo.gdalconst()`, optional) – A valid resample method from `gdalconst`. Default is `gdalconst.GRA_Average`.
- **as_gdal_grid** (`bool`, optional) – Return as `GDALGrid()`. Default is `False`.

Returns If `to_file` is a `str`, then it returns `None`. Otherwise, if `to_file` is `False` then it returns a `gdal.Dataset()` unless `as_gdal_grid` is `True`. Then, it returns `GDALGrid()`.

Return type `None` or `gdal.Dataset()` or `GDALGrid()`

```
gazar.grid.gdal_reproject(src, dst=None, src_srs=None, dst_srs=None, epsg=None, error_threshold=0.125, resampling=<Mock id='140403181045264'>, as_gdal_grid=False)
```

Reproject a raster image.

Based on: https://github.com/OpenDataAnalytics/gaia/blob/master/gaia/geo/gdal_functions.py

Parameters

- **src** (`str` or `gdal.Dataset()` or `GDALGrid()`) – The source image.
- **dst** (`str`, optional) – The filepath of the output image to write to.
- **src_srs** (`osr.SpatialReference()`, optional) – The source image projection.
- **dst_srs** (`osr.SpatialReference()`, optional) – The destination projection. If not provided, the code will use `epsg`.
- **epsg** (`int`, optional) – The EPSG code to reproject to. If not provided, the code will use `dst_srs`.
- **error_threshold** (`float`, optional) – Default is 0.125 (same as `gdalwarp` commandline).
- **resampling** (`osgeo.gdalconst()`) – Method to use for resampling. Default method is `gdalconst.GRA_NearestNeighbour`.
- **as_gdal_grid** (`bool`, optional) – Return as `GDALGrid()`. Default is `False`.

Returns By default, it returns `gdal.Dataset`. It will return `GDALGrid()` if `as_gdal_grid` is `True`.

Return type `gdal.Dataset()` or `GDALGrid()`

`gazar.shape.reproject_layer(in_path, out_path, out_spatial_ref)`

Reprojects a shapefile layer.

Based on: <https://pcjericks.github.io/py-gdalogr-cookbook/projection.html>

Parameters

- **in_path** (`str`) – The path to the input shapefile layer.
- **out_path** (`str`) – The path to the output shapefile layer.
- **out_spatial_ref** (`osr.SpatialReference()`) – The output spatial reference.

`gazar.shape.rasterize_shapefile(shapefile_path, out_raster_path=None, shapefile_attribute=None, x_cell_size=None, y_cell_size=None, x_num_cells=None, y_num_cells=None, match_grid=None, raster_wkt_proj=None, convert_to_utm=False, raster_dtype=<Mock id='140403173282256'>, raster_nodata=-9999, as_gdal_grid=False)`

Convert shapefile to raster from specified attribute

Parameters

- **shapefile_path** (`str`) – Path to shapefile.
- **out_raster_path** (`str`, optional) – Path to raster to be generated.
- **shapefile_attribute** (`str`, optional) – Attribute to be rasterized.
- **x_cell_size** (`float`, optional) – Longitude cell size in output projection.
- **y_cell_size** (`float`, optional) – Latitude cell size in output projection.
- **x_num_cells** (`int`, optional) – Number of cells in latitude.
- **y_num_cells** (`int`, optional) – Number of cells in longitude.
- **match_grid** (`str` or `gdal.Dataset()` or `GDALGrid()`, optional) – Grid to match for output.

- **raster_wkt_proj** (*str*, optional) – WKT projections string for output grid.
- **convert_to_utm** (*bool*, optional) – Convert grid to UTM automatically. Default is False.
- **raster_dtype** (*osgeo.gdalconst()*) – Output grid datatype (GDT). Default is `gdal.GDT_Int32`.
- **raster_nodata** (*float or int*, optional) – No data value for output raster. Default is -9999,
- **as_gdal_grid** (*bool*, optional) – Return as `GDALGrid()`. Default is False.

Returns It will return `GDALGrid()` if *as_gdal_grid* is True. Otherwise, it will not return anything.

Return type None or `GDALGrid()`

Example Default:

```
from gloat.grid import rasterize_shapefile

shapefile_path = 'shapefile.shp'
new_grid = 'new_grid.tif'
rasterize_shapefile(shapefile_path,
                    new_grid,
                    x_num_cells=50,
                    y_num_cells=50,
                    raster_nodata=0,
                    )
```

Example GDALGrid to ASCII with UTM:

```
from gazar.grid import rasterize_shapefile

shapefile_path = 'shapefile.shp'
new_grid = 'new_grid.asc'
gr = rasterize_shapefile(shapefile_path,
                        x_num_cells=50,
                        y_num_cells=50,
                        raster_nodata=0,
                        convert_to_utm=True,
                        as_gdal_grid=True,
                        )
gr.to_grass_ascii(new_grid, print_nodata=False)
```

CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`

A

ArrayGrid (class in gazar.grid), 6

B

bounds() (gazar.grid.GDALGrid method), 3

C

coord2pixel() (gazar.grid.GDALGrid method), 3

E

epsg (gazar.grid.GDALGrid attribute), 4

G

gdal_reproject() (in module gazar.grid), 8

GDALGrid (class in gazar.grid), 3

geotransform (gazar.grid.GDALGrid attribute), 4

geotransform_from_yx() (in module gazar.grid), 7

L

latlon (gazar.grid.GDALGrid attribute), 4

lonlat2pixel() (gazar.grid.GDALGrid method), 4

N

np_array() (gazar.grid.GDALGrid method), 4

num_bands (gazar.grid.GDALGrid attribute), 4

P

pixel2coord() (gazar.grid.GDALGrid method), 4

pixel2lonlat() (gazar.grid.GDALGrid method), 4

proj (gazar.grid.GDALGrid attribute), 5

proj4 (gazar.grid.GDALGrid attribute), 5

R

rasterize_shapefile() (in module gazar.shape), 9

reproject_layer() (in module gazar.shape), 9

resample_grid() (in module gazar.grid), 7

T

to_arc_ascii() (gazar.grid.GDALGrid method), 5

to_grass_ascii() (gazar.grid.GDALGrid method), 5

to_projection() (gazar.grid.GDALGrid method), 5

to_tif() (gazar.grid.GDALGrid method), 5

U

utm_proj_from_latlon() (in module gazar.grid), 7

W

wkt (gazar.grid.GDALGrid attribute), 5

write_prj() (gazar.grid.GDALGrid method), 5

X

x_size (gazar.grid.GDALGrid attribute), 5

Y

y_size (gazar.grid.GDALGrid attribute), 5