PyRaster - Python spatial image processing

API Documentation

July 30, 2012

Contents

Contents **1**

1 Module rasterIO **2**
1.1 Functions 3
1.2 Variables 4

Index **5**

1 Module rasterIO

rasterIO - Library of functions to convert geospatial raster formats to/from Numpy masked arrays.

Introduction

This library contains wrapper functions for GDAL Python bindings, converting data to Numerical Python multi-dimensional array's in memory for processing. Subsequent generated arrays can be written to disk in the standard geospatial GeoTiff format.

Notes

- Error checking - rasterIO contains minimal user-level error checking

Supported Formats

- Input:
  - rasterIO supports reading any GDAL supported raster format
- Output:
  - rasterIO generates GeoTiff files by default (this can be modified in the source code)
  - GeoTiffs are created with embedded binary header files containing geo information

Supported Datatypes

- Raster IO supports Float32 and Int16 data types
- The default datatype is Float32
- Boolean datasets use Int16 datatypes

NoDataValue

If the input data has no recognisable NoDataValue (readable by GDAL) then the input NoDataValue is assumed to be 9999. This can be changed by manually specifying an input NoDataVal when calling readrasterbands() In accordance with GDAL the output data NoDataValue is 9999 or 9999.0 or can be manually set by when writrasterbands() When using unsigned integer data types the default output NoDataValue will be 0

How to use documentation

Documentation for module functions is provided as Python docstrings, accessible from an interactive Python terminal Within docstrings examples from an interactive Python console are identified using '>>>' Further information is given to developers within the source code using '#' comment strings To view this text and a list of available functions call the Python in-built help command, specifying module name

```
>>> import rasterIO
>>> help(rasterIO)
...this text...
```

For help on a specific function call the Python in-built help command, specifying module.function

```
>>> import rasterIO
>>> help(rasterIO.wkt2epsg)
Help on function wkt2epsg in module rasterIO
wkt2epsg(wkt)
Accepts well known text of Projection/Coordinate Reference System and generates EPSG code
```

How to access functions

To access functions, import the module to Python and call the desired function, assigning the output to a named variable Note that the primary input datatype (default) for all functions is either a Numpy array or a Numpy masked array Within this module the term "raster" is used to signify a Numpy/Numpy masked array of raster values Use the rasterIO module to convert Numpy arrays to/from Geospatial raster data formats

```
>>> import rasterIO
>>> band_number = 1
>>> rasterdata = rasterIO.readrasterband(gdal_file_pointer, band_number)
```

Optional function arguments are shown in document strings in brackets [argument]

Dependencies

Python 2.5 or greater Numerical python (Numpy) 1.2.1 or greater (1.4.1 recommended)

- Note that due to bugs in Numpy.ma module, Numpy 1.4.1 or greater is required to support masked arrays of integer values * See comments in reasrasterband() for more information

License & Authors

Version: 1.1.1

1.1 Functions

opengdalraster(*fname*)

Accepts gdal compatible file on disk and returns gdal pointer.

readrastermeta(*dataset*)

Accepts GDAL raster dataset and returns, gdal_driver, XSize, YSize, projection info(well known text), geotranslation data.

readrasterband(*dataset*, *aband*, *NoDataVal*=`None`, *masked*=`True`)

Accepts GDAL raster dataset and band number, returns Numpy 2D-array.

newgdalraster(*outfile*, *format*, *XSize*, *YSize*, *geotrans*, *epsg*, *num_bands*, *gdal_dtype*)

Accepts file_path, format, X, Y, geotransformation, epsg, number_of_bands, gdal_datatype and returns gdal pointer to new file.

This is a lower level function that allows users to control data output stream directly, use for specialist cases such as varying band data types or memory limited read-write situations. Note that users should not forget to close file once data output is complete (dataset = None).

newrasterband(*dst_ds*, *rasterarray*, *band_num*, *NoDataVal*=`None`)

Accepts a GDAL dataset pointer, rasterarray, band number, [NoDataValue], and creates new band in file.

writerasterbands(*outfile*, *format*, *XSize*, *YSize*, *geotrans*, *epsg*, *NoDataVal*=`None`, **rasterarrays*)

Accepts raster(s) in Numpy 2D-array, outputfile string, format and geotranslation metadata and writes to file on disk

writerasterband(*rasterarray*, *outfile*, *format*, *aXSize*, *aYSize*, *geotrans*, *epsg*, *NoDataVal*=`None`)

Legacy function for backwards compatability with older scripts. Use writerasterbands instead.

Accepts raster in Numpy 2D-array, outputfile string, format and geotranslation metadata and writes to file on disk

wkt2epsg(*wkt*)

Accepts well known text of Projection/Coordinate Reference System and generates EPSG code

band2txt(*band*, *outfile*)

Accepts numpy rasterand writes to specified text file on disk.

1.2 Variables

Name	Description
gdt2npy	**Value:** `{1: 'uint8', 2: 'uint16', 3: 'int16', 4: 'uint32', 5: 'in...`
npy2gdt	**Value:** `{'float32': 6, 'float64': 7, 'int16': 3, 'int32': 5, 'uin...`
gdt2struct	**Value:** `{1: 'B', 2: 'H', 3: 'h', 4: 'I', 5: 'i', 6: 'f', 7: 'd'}`
___package___	**Value:** `None`

Index

rasterIO *(module)*, 2–4
    rasterIO.band2txt *(function)*, 4
    rasterIO.newgdalraster *(function)*, 3
    rasterIO.newrasterband *(function)*, 3
    rasterIO.opengdalraster *(function)*, 3
    rasterIO.readrasterband *(function)*, 3
    rasterIO.readrastermeta *(function)*, 3
    rasterIO.wkt2epsg *(function)*, 4
    rasterIO.writerasterband *(function)*, 4
    rasterIO.writerasterbands *(function)*, 4