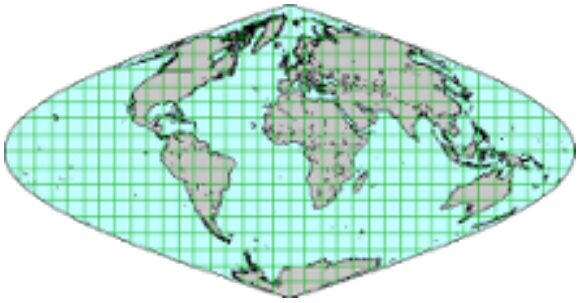

pyModis Documentation

Release 0.6.2

Luca Delucchi

April 08, 2013

CONTENTS

`pyModis` library was developed to replace old bash scripts developed by Markus Neteler to download MODIS data from NASA FTP server. It is very useful for GIS and Remote Sensing Platform of Fondazione Edmund Mach to update its large collection of MODIS data.

It has several features:

- it is very useful for downloading large numbers of MODIS HDF/XML files and for using this in a cron job for automated continuous updating
- it can parse the XML file to obtain information about the HDF files
- it can convert a HDF MODIS file to GEOTIF file using MODIS Reprojection Tool
- it can create a mosaic of several tiles using MODIS Reprojection Tool and can create the xml metadata file with the information of all tiles used in mosaic

CHAPTER

ONE

ABOUT PYMODIS

1.1 Requirements

The only required software is MODIS Reprojection Tool to convert or mosaic MODIS HDF files.

For *download* or *parse* HDF files only standard Python modules are used by `pyModis` library and tools.

1.2 How to install pyModis

Installing `pyModis` is very simple. First you need to download `pyModis` source code from github repository.

You can use git to download the latest code (with the whole history and so it contain all the different stable versions, from the last to the first)

```
git clone git://github.com/lucadelu/pyModis.github
```

or download the latest stable version from the repository and decompress it.

Now enter the `pyModis` folder and launch as administrator of your computer

```
python setup.py install
```

If the installation doesn't return any errors you should be able to use `pyModis` library from a Python console. Then, launch a your favorite Python console (I really suggest `ipython`) and digit

```
import pymodis
```

If the console doesn't return any error like this

```
ImportError: No module named pymodis
```

the `pyModis` library has been installed properly and you can use it or one of the tools distributed with `pyModis`.

1.3 How to report a bug

If you find any problems in `pyModis` library you can report it using the issues tracker of github.

1.4 How to compile documentation

This documentation has been made with Sphinx, so you need to install it to compile the original files to obtain different output formats.

Please enter the `docs` folder of `pyModis` source and run

```
make <target>
```

with one of the following target to obtain the desired output:

- **html**: to make standalone HTML files
- **dirhtml**: to make HTML files named index.html in directories
- **singlehtml**: to make a single large HTML file
- **pickle**: to make pickle files
- **json**: to make JSON files
- **htmlhelp**: to make HTML files and a HTML help project
- **qthelp**: to make HTML files and a qthelp project
- **devhelp**: to make HTML files and a Devhelp project
- **epub**: to make an epub
- **latex**: to make LaTeX files, you can set PAPER=a4 or PAPER=letter
- **latexpdf**: to make LaTeX files and run them through pdflatex
- **text**: to make text files
- **man**: to make manual pages
- **texinfo**: to make Texinfo files
- **info**: to make Texinfo files and run them through makeinfo
- **gettext**: to make PO message catalogs
- **changes**: to make an overview of all changed/added/deprecated items
- **linkcheck**: to check all external links for integrity
- **doctest**: to run all doctests embedded in the documentation (if enabled)

1.4.1 PDF link in HTML

To insert a link to PDF file of pyModis documentation into HTML documentation (the link will be added on the sidebar) you have to compile first the PDF and after the HTML, so you need to launch:

```
make latexpdf
make html
```

If PDF file is missing no link will be added

1.5 Ohloh statistics

For more information about `pyModis` please visit the pyModis Ohloh page

CHAPTER

TWO

PYMODIS SCRIPTS

The `pyModis` **scripts** provide you with a complete toolkit to work with MODIS data, you can download, analyze and convert data. They are developed to work from command line and inside scripts to automatically update your MODIS files dataset.

Currently the tools are 5:

- *modis_download.py*
- *modis_parse.py*
- *modis_multiparse.py*
- *modis_mosaic.py*
- *modis_convert.py*

2.1 modis_download.py

modis_download.py is a script to download MODIS data from NASA FTP servers. It can download large amounts of data and it can be profitably used with cron jobs to receive data with a fixed delay of time.

2.1.1 Usage

```
modis_download.py [options] destination_folder
```

2.1.2 Options

```
-h  --help        show the help
-P  --password    password to connect to ftp server,
                  usually your email address  [required]
-U  --username    username to connect to ftp server
                  [default=anonymous]
-u  --url         ftp server url [default=e4ftl01.cr.usgs.gov]
-t  --tiles       string of tiles separated from comma
                  ([default=None] for all tiles)
-s  --source      directory on the ftp
                  ([default=MOLT/MOD11A1.005] for Terra LST data)
-D  --delta       delta of day from the first day [default=10]
-f  --firstday    the day to start download, if you want change
                  data you have to use this format YYYY-MM-DD
                  ([default=None] is for today)
-e  --enddaythe   day to finish download, if you want change
                  data you have to use this format YYYY-MM-DD
                  ([default=None] use delta option)
-x                this is useful for debugging the download
                  [default=False]
-j                download also the jpeg files [default=False]
-O                download only one day, it set delta=1 [default=False]
-A                download all days, it usefull for first download of a
                  product. It overwrite the 'firstday' and 'endday'
                  options [default=False]
-r                remove files with size ugual to zero from
                  'destination_folder'  [default=False]
```

2.1.3 Examples

Download Terra LST data for a month for Europe

```
modis_download.py -P your.mail@prov.org -t TODO -f 2008-01-01 -e 2008-01-31
```

Download the last 15 days of Aqua LST data

```
modis_download.py -P your.mail@prov.org -s MOLA/MYD11A1.005 -t TODO -D 15
```

Download all tiles of NDVI for one day (you have know the rigth day otherwise it download nothing)

```
modis_download.py -P your.mail@prov.org -s TODO -f 2010-12-31 -O
```

2.2 modis_parse.py

modis_parse.pys is a script to parse the XML metadata file for a MODIS tile and return the requested value. It can also write the metadata information in a text file.

2.2.1 Usage

```
modis_parse.py [options] hdf_file
```

2.2.2 Options

```
-h  --help     show the help
-a             print all possible values of metadata
-b             print the values related to the spatial max extent
-d             print the values related to the date files
-e             print the values related to the ECSDataGranule
-i             print the input layers
-o             print the other values
-p             print the values related to platform
-q             print the values related to quality
-s             print the values related to psas
-t             print the values related to times
-w  --write    write the chosen information into a file
```

2.2.3 Examples

Return all values of metadata

```
modis_parse.py -a FILE
```

Write all values to a file

```
modis_parse.py -a -w metadata_FILE.txt FILE
```

Print spatial extent and quality

```
modis_parse.py -b -q FILE
```

2.3 modis_multiparse.py

modis_multiparse.py is a script to parse several XML metadata files for MODIS tiles. It is very usefull to create XML metadata file for a mosaic.

2.3.1 Usage

```
modis_multiparse.py [options] hdf_files_list
```

2.3.2 Options

```
-h  --help     show the help
-b             print the values related to the spatial max extent
-w  --write    write the MODIS XML metadata file for MODIS mosaic
```

2.3.3 Examples

Print values of spatial bounding box

```
modis_multiparse.py -b FILE1 FILE2 ...
```

Write xml file to use with hdf file create by *modis_convert.py*

```
modis_multiparse.py -w FILE_mosaic.xml FILE1 FILE2 ...
```

2.4 modis_mosaic.py

modis_mosaic.py is a script to create a mosaic of several MODIS tiles in HDF format.

2.4.1 Usage

```
modis_mosaic.py [options] hdflist_file
```

2.4.2 Options

```
-h  --help      show the help
-m  --mrt       the path to MRT software    [required]
-o  --output    the name of output file    [required]
-s  --subset    a subset of product's layers. The string
                should be similar to: 1 0 [default: all layers]
```

2.4.3 Examples

Convert all the layers of several tiles:

```
modis_mosaic.py -m "/usr/local/bin/" -o FILE_mosaik FILE1 FILE2 ...
```

Convert LAYERS of several LST MODIS tiles:

```
modis_mosaic.py -s "1 0 1 0" -m "/usr/local/bin/" -o FILE_mosaik FILE1 FILE2 ...
```

2.5 modis_convert.py

modis_convert.py is a script to convert MODIS data to TIF formats and different projection reference system. It is an interface to MRT mrtmosaic software, the best application for work with HDF MODIS data.

2.5.1 Usage

```
modis_convert.py [options] hdf_file
```

2.5.2 Options

```
-h  --help              show the help
-s  --subset            a subset of product's layers. The string
                        should be similar to: 1 0    [required]
-m  --mrt               the path to MRT software    [required]
-o  --output            the name of output file
-g  --grain             the spatial resolution of output file
-d  --datum             the code of datum
-r  --resampl           the type of resampling
-p  --proj_parameters   a list of projection parameters
-t  --proj_type         the output projection system
-u  --utm               the UTM zone if projection system is UTM
```

Warning: You can find the supported projections in the 'Appendix C' of MODIS reprojection tool user's manual and the datums at section `Datum Conversion` of the same manual

2.5.3 Examples

Convert LAYERS from LST MODIS data with output resolution in 250 meters with latitude and longitude reference system

```
modis_convert.py -s "1 0 1 0" -m "/usr/local/bin/" -g 250 FILE
```

Convert LAYERS from NDVI MODIS data with output resolution in 500 meters with UTM projection in the 32 zone

```
modis_convert.py -s "1 0 1 0" -m "/usr/local/bin/" -g 500 -p UTM -u 32 FILE
```

CHAPTER

THREE

EXAMPLE OF A FULL PROCESS

In this short example you can understand how to concatenate the scripts to obtain a GeoTIFF file for each band of the chosen product.

> **Warning:** This example is based on a Linux based system. Please if you use other OS change the paths where data will be saved

3.1 Downloading data

For first you need to obtain data, so you need to use *modis_download.py*

```
modis_download.py -f 2012-12-05 -O -t h28v05,h29v05,h28v04
                  -P yourmail@mail.org /tmp/
```

> **Warning:** In this example we are working on Japan extension, so please change the name of tiles according with your region.
> In this example we download data for only one day (2012-12-05) using the option "-O".
> Please change yourmail@mail.org with your mail.

Inside `/tmp/` directory you will find a file called *listfileMOD11A1.005.txt* containing the names of files downloaded. The name of file it is related to the product that you download.

> **Warning:** Every time that you download new files of same product it will be overwrite, so if you need it, you should rename the file

3.2 Mosaic data

At this point you need to create the mosaic of the tiles downloaded. *modis_mosaic.py* is the script to use.

```
modis_mosaic.py -m /path/to/mrt/ -o /tmp/outputfile /tmp/listfileMOD11A1.005.txt
```

> **Warning:** `/path/to/mrt/` is the directory where Modis Reprojection Tools is stored

The output of this command are *outputfile.hdf* and *outputfile.hdf.xml* inside the directory `/tmp`. It's reading the input files contained in *listfileMOD11A1.005.txt*

3.3 Convert data

The last part of the procedure is to convert the mosaic, from HDF format and sinusoidal projection, to GeoTIFF with several projection. You have to use *modis_convert.py*

```
modis_convert.py -s '( 1 1 1 1 1 1 1 1 1 1 1 1 )' -m /path/to/mrt/
                 -o /tmp/finalfile.tif -g 250 /tmp/outputfile.hdf
```

CHAPTER

FOUR

PYMODIS LIBRARY

`pyModis` library it is a Python library to work with MODIS data.

It can easily used in your application to download, analyze and convert MODIS data, it is already used by GRASS GIS in r.in.modis addons tools.

It is compose by three module:

4.1 pyModis Package

4.1.1 `downmodis` module: is very useful to download MODIS data from NASA FTP server

class `pymodis.downmodis.`**downModis**(*password*, *destinationFolder*, *user='anonymous'*, *url='e4ftl01.cr.usgs.gov'*, *tiles=None*, *path='MOLT/MOD11A1.005'*, *today=None*, *enddate=None*, *delta=10*, *jpg=False*, *debug=False*)

A class to download MODIS data from NASA FTP repository

checkDataExist(*listNewFile*, *move=0*)

Check if a file already exists in the directory of download

listNewFile = list of all files, returned by getFilesList function

move = it is useful to know if a function is called from download or move function

closeFTP()

Close ftp connection

connectFTP(*ncon=20*)

Set connection to ftp server, move to path where data are stored and create a list of directories for all days

ncon = number maximum of test to connection at the ftp server

dayDownload(*listFilesDown*)

Downloads tiles are in files_hdf_consider

listFilesDown = list of the files to download, returned by checkDataExist function

debugDays()

This function is useful to debug the number of days

debugLog()
Function to create the debug file

debugMaps()
This function is useful to debug the number of maps to download for each day

downloadsAllDay(*clean=False*, *allDays=False*)
Downloads all the tiles considered

getAllDays()
Return a list of all days

getFilesList()
Create a list of files to download, it is possible choose to download also the jpeg files or only the hdf files

getListDays()
Return a list of all selected days

getNewerVersion(*oldFile*, *newFile*)
Return newer version of a file

oldFile = one of the two similar file

newFile = one of the two similar file

removeEmptyFiles()
Check if some file has size ugual 0

setDirectoryIn(*day*)
Enter in the directory of the day

setDirectoryOver()
Come back to old path

4.1.2 parsemodis module: is very simple library to parse MODIS metadata file, it can also write the XML metadata file for a mosaic.

class pymodis.parsemodis.**parseModis**(*filename*)
Class to parse MODIS xml files, it also can create the parameter configuration file for resampling MODIS DATA with the MRT software or convertmodis Module

confResample(*spectral*, *res=None*, *output=None*, *datum='WGS84'*, *resample='NEAREST_NEIGHBOR'*, *projtype='GEO'*, *utm=None*, *projpar='(0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)'*)
Create the parameter file to use with resample MRT software to create tif file

spectral = the spectral subset to be used, look the product table to understand the layer that you want use. For example:

- NDVI (1 1 1 0 0 0 0 0 0 0 0 0) copy only layer NDVI, EVI and QA VI the other layers are not used
- LST (1 1 0 0 1 1 0 0 0 0 0 0) copy only layer daily and nightly temperature and QA

res = the resolution for the output file, it must be set in the map unit of output projection system. The software will use the original resolution of input file if res it isn't set

output = the output name, if it doesn't set will use the prefix name of input hdf file

utm = the UTM zone if projection system is UTM

resample = the type of resampling, the valid values are:

- NN (nearest neighbor)
- BI (bilinear)
- CC (cubic convolution)

projtype = the output projection system, the valid values are:

- AEA (Albers Equal Area)
- ER (Equirectangular)
- GEO (Geographic Latitude/Longitude)
- HAM (Hammer)
- ISIN (Integerized Sinusoidal)
- IGH (Interrupted Goode Homolosine)
- LA (Lambert Azimuthal)
- LCC (LambertConformal Conic)
- MERCAT (Mercator)
- MOL (Mollweide)
- PS (Polar Stereographic)
- SIN (Sinusoidal)
- UTM (Universal TransverseMercator)

datum = the datum to use, the valid values are:

- NAD27
- NAD83
- WGS66
- WGS76
- WGS84
- NODATUM

projpar = a list of projection parameters, for more info check the Appendix C of MODIS reprojection tool user manual https://lpdaac.usgs.gov/content/download/4831/22895/file/mrt41_usermanual_032811.pdf

confResample_swath(*sds*, *geoloc*, *res*, *output=None*, *sphere='8'*, *resample='NN'*, *projtype='GEO'*, *utm=None*, *projpar='0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0'*)

Create the parameter file to use with resample MRT software to create tif file

sds = Name of band/s (Science Data Set) to resample

geoloc = Name geolocation file (example MOD3, MYD3)

res = the resolution for the output file, it must be set in the map unit of output projection system. The software will use the original resolution of input file if res it isn't set

output = the output name, if it doesn't set will use the prefix name of input hdf file

sphere = Output sphere number. Valid options are:

- 0=Clarke 1866
- 1=Clarke 1880
- 2=Bessel
- 3=International 1967
- 4=International 1909
- 5=WGS 72
- 6=Everest
- 7=WGS 66
- 8=GRS1980/WGS 84
- 9=Airy
- 10=Modified Everest
- 11=Modified Airy
- 12=Walbeck
- 13=Southeast Asia
- 14=Australian National
- 15=Krassovsky
- 16=Hough
- 17=Mercury1960
- 18=Modified Mercury1968
- 19=Sphere 19 (Radius 6370997)
- 20=MODIS Sphere (Radius 6371007.181)

resample = the type of resampling, the valid values are:

- NN (nearest neighbor)
- BI (bilinear)
- CC (cubic convolution)

projtype = the output projection system, the valid values are:

- AEA (Albers Equal Area)
- ER (Equirectangular)
- GEO (Geographic Latitude/Longitude)
- HAM (Hammer)
- ISIN (Integerized Sinusoidal)

- IGH (Interrupted Goode Homolosine)
- LA (Lambert Azimuthal)
- LCC (LambertConformal Conic)
- MERCAT (Mercator)
- MOL (Mollweide)
- PS (Polar Stereographic),
- SIN ()Sinusoidal)
- UTM (Universal TransverseMercator)

utm = the UTM zone if projection system is UTM

projpar = a list of projection parameters, for more info check the Appendix C of MODIS reprojection tool user manual https://lpdaac.usgs.gov/content/download/4831/22895/file/mrt41_usermanual_032811.pdf

getGranule()
Set the GranuleURMetaData element

getRoot()
Set the root element

retBoundary()
Return the maximum extend (Bounding Box) of the MODIS file as dictionary

retBrowseProduct()
Return the BrowseProduct element

retCollectionMetaData()
Return the CollectionMetaData element as dictionary

retDTD()
Return the DTDVersion element

retDataCenter()
Return the DataCenterId element

retDataFiles()
Return the DataFiles element as dictionary

retDataGranule()
Return the ECSDataGranule elements as dictionary

retDbID()
Return the DbID element

retGranuleUR()
Return the GranuleUR element

retInputGranule()
Return the input files (InputGranule) used to process the considered file

retInsertTime()
Return the InsertTime element

retLastUpdate()
Return the LastUpdate element

retMeasure()
Return statistics of QA as dictionary

retPGEVersion()
Return the PGEVersion element

retPSA()
Return the PSA values as dictionary, the PSAName is the key and and PSAValue is the value

retPlatform()
Return the platform values as dictionary.

retRangeTime()
Return the RangeDateTime elements as dictionary

class pymodis.parsemodis.**parseModisMulti**(*hdflist*)
A class to obtain some variables for the xml file of several MODIS tiles. It can also create the xml file

valBound()
Function return the Bounding Box of mosaic

valCollectionMetaData(*obj*)
Function to add CollectionMetaData

obj = element to add CollectionMetaData

valDTD(*obj*)
Function to add DTDVersion

obj = element to add DTDVersion

valDataCenter(*obj*)
Function to add DataCenter

obj = element to add DataCenter

valDataFiles(*obj*)
Function to add DataFileContainer

obj = element to add DataFileContainer

valDbID(*obj*)
Function to add DbID

obj = element to add DbID

valGranuleUR(*obj*)
Function to add GranuleUR

obj = element to add GranuleUR

valInputPointer(*obj*)
Function to add InputPointer

obj = element to add InputPointer

valInsTime(*obj*)
Function to add the minimum of InsertTime

obj = element to add InsertTime

valMeasuredParameter(*obj*)
Function to add ParameterName

obj = element to add ParameterName

valPGEVersion(*obj*)
Function to add PGEVersion

obj = element to add PGEVersion

valPlatform(*obj*)
Function to add Platform elements

obj = element to add Platform elements

valRangeTime(*obj*)
Function to add RangeDateTime

obj = element to add RangeDateTime

writexml(*outputname*)
Write a xml file for a mosaic

outputname = the name of xml file

4.1.3 convertmodis module: using MRT software, can convert MODIS HDF file to GeoTiff file or create a HDF mosaic file for several tiles.

class pymodis.convertmodis.**convertModis**(*hdfname*, *confile*, *mrtpath*)
A class to convert modis data from hdf to tif using resample (from MRT tools)

executable()
Return the executable of resample MRT software

run()
Exec the convertion process

class pymodis.convertmodis.**createMosaic**(*listfile*, *outprefix*, *mrtpath*, *subset=False*)
A class to convert several MODIS tiles into a mosaic

executable()
Return the executable of mrtmosaic MRT software

run()
Exect the mosaic process

write_mosaic_xml()
Write the XML metadata file for MODIS mosaic

class pymodis.convertmodis.**processModis**(*hdfname*, *confile*, *mrtpath*, *inputhdf=None*, *outputhdf=None*, *geolocfile=None*)
A class to process raw modis data from hdf to tif using swath2grid (from MRT Swath tools)

executable()
Return the executable of resample MRT software

run()
Exec the convertion process

We acknowledge the Fondazione Edmund Mach for promoting the development of free and open source software.

PYTHON MODULE INDEX

p

INDEX

S

V

W