py-stint: PYthon Spatial/Temporal INtersection Toolset

IndEcol @ Norges teknisk-naturvitenskapelige universitet
Wiley Bogren
2014-09-11
— Version 1.1

Abstract

This document describes the capabilities and use of the `py-stint` toolset for spatial and temporal intersections. The primary expected use for py-stint is to produce datasets combining land-use and climate data with MODIS albedo timeseries.

Contents

1 Directory Structure

The `py-stint` project relies on four essential directories. The user should have write access to `INPUT.txt` and `STINT/`. These directories can be located anywhere relative to each other, but the structure of each directory should be as follows:

```
py-stint/ :  INPUT.txt
             run.py
             Tools/
                *.py modules
                Data/
                   MODIS_tiles.shp

MODIS_ARCHIVE/ :  MCD43A3/ –One directory per MODIS product
                     h19v02/ –One directory per MODIS tile
                        *.hdf

ERA_ARCHIVE/ :  era_climate_t2m_1of3.nc
                era_climate_t2m_2of3.nc
                era_climate_t2m_3of3.nc
                climate_field2_1of1.nc

STINT/ :  Holds output directories, which are made while running py-stint.
```

2 Getting Data

2.1 Corine Landcover

2006 Corine Land Cover seamless vector data is part of the European Commission programme to COoRdinate INformation on the Environment (Corine).

EEA grants free access to all its data/applications provided that the user agrees:

- to acknowledge the source as follows: Copyright EEA, Copenhagen, 2007
- to display a link to the EEA web site http://www.eea.europa.eu
- not to use the data/applications for commercial purposes unless the Agency has expressly granted the right to do so

A spatialite database containing all landcover classes can be downloaded from the following link:

http://www.eea.europa.eu/data-and-maps/data/clc-2006-vector-data-version-3/

The spatialite database can be converted to shapefile and cropped to an Area of Interest via commandline:

```
$ ogr2ogr -f "ESRI Shapefile" <path/to/output> <path/to/clc06.sqlite>
-dsco SPATIALITE=yes -clipsrc <xmin> <ymin> <xmax> <ymax>
```

Where `<xmin> <ymin> <xmax> <ymax>` are the bounding x and y coordinates of the Area of Interest, in the LAEA-ETRS89 projection.

2.2 ERA-Interim Climate

ERA Interim climate data can be downloaded from the following website. The server may require an user account before allowing the download.

download: http://apps.ecmwf.int/datasets/data/interim_full_daily/

terms of use: http://apps.ecmwf.int/datasets/data/interim_full_daily/licence/

! → The ECMWF server sets a limit of about 2GB per file download. Download data in netcdf format, in either 6-hour or 24-hour resolution, with a single field (attribute) per file.

If the desired timespan will not fit on a single file, break it into as many downloads as necessary, with the start-date of each successive file one time-step later than the end-date of the previous file. e.g. if file1 ends 18:00 31. December 2011, then file2 should start 00:00 1. January 2012.

2.3 MODIS Datasets

MODIS products can be downloaded from two main sources, given below. Future versions of `py-stint` will incorporate dataset retrieval tools and functions based on the `modis_download.py` script from the `pyModis` project: `http://pymodis.fem-environment.eu/`

For now the best plan is to either figure out `modis_download.py` yourself (recommended, as it is excellent) or to (1) write a script in python or R to compile a list of files you want to download, then (2) use a command line utility like `axel` to retrieve all the files in the list.

After all files have been retrieved, ensure that the `MODIS_Archive/` directory is organized in the `archive/product/tile/*.hdf` structure, as illustrated in Section 1.

Core MODIS products such as albedo and surface temperature:
`http://e4ftl01.cr.usgs.gov/MOTA/`

Snow cover (National Snow and Ice Data Center):
`ftp://n5eil01u.ecs.nsidc.org/MOST/`

3 Workflow

Python programs given in the workflow guide should be called from the `py-stint/` directory. These commands are presented in the format `$ python ...`

3.1 Setup

- `$ python Tools/MODIS_aoi.py path/to/aoi.shp`
  - Print a list of `MODIS` tiles to include in `MODIS_ARCHIVE/`
- Create project directory
- Prepare `ERA` and `MODIS` archives as per Section 2. `MODIS_ARCHIVE/` should be complete with regard to:
  - products (MCD43A3 etc)
  - dates within timeframe
  - `MODIS` tiles which have been identifed within the `AOI`
- Fill out `INPUT.ex` with actual project parameters, then save as `py_stint/INPUT.txt`

3.2 Processing

After completing the setup, execute each of the 7 processing stages consecutively via the command line. Each stage needs to execute successfully before the next stage can run, but nothing else needs to be done in between stages.

3.2.1 Stage 1: `$ python run.py 1`

The stage begins by parsing project parameters from `INPUT.txt`, as well as loading the projection and extent from the area of interest (`AOI`). Technically, these are loaded at the start of every stage, but this is where you'll be alerted to any errors in the input parameters.

The stage then checks for corrupt or missing files within the `MODIS_archive` given the region, timeframe and datasets specified in the input parameters.

Finally, the time variable is loaded from each NetCDF file in the `ERA` archive. An error is raised if there are any gaps or overlaps in timesteps.

Requires : `INPUT.txt`
`AOI`/landcover shapefile
`MODIS_archive`
`ERA_archive`
`MODIS_tiles.shp`

Produces : Prints a report on any issues found in the `MODIS` or `ERA` archives.

Creates subdirectories within the `STINT/` directory, if they are not already present.

Explore : This stage produces no new data to explore.

3.2.2 **Stage 2:** `$ python run.py 2`

This stage aggregates each `MODIS` and `ERA` dataset into a single HDF5 array file, with dimensions [time, y, x].

The [time] dimension is shared by `MODIS` and `ERA` arrays, each timestep covering one 16-day `MODIS` observation period.

The spatial dimensions [y,x] are defined by the dataset's native projection (`MODIS sinusoidal` or `WGS84` geographic coordinates).

HDF5 arrays are cropped to the project timeframe and the minimum rectangle necessary to completely contain the `AOI`.

`MODIS` datasets: tiles are merged, subsets determined, and days aggregated.

`ERA` datasets: subsets determined, 6- or 24-hour records averaged to match 16-day `MODIS` intervals.

Requires : `INPUT.txt`
`AOI`/landcover shapefile
`MODIS_archive`
`ERA_archive`
`MODIS_tiles.shp`

Produces : `ERA` and `MODIS` HDF5 array files, native projections:
`STINT/Processing/HDF/<dataset>.hdf5`

Explore : The HDF5 array file is probably the richest, most exciting format generated in `py-stint` processing. Data for a given region and timeframe is stored in a single, compact and efficient format. It has excellent potential for visualization of spatial and temporal trends within albedo and climate datasets.

The first tool for exploring HDF5 data will be a python program for converting individual timesteps to geoTIFF files, which can be viewed in a desktop GIS browser.

* Python: h5py and numpy

* R: rhdf5, h5r, or ncdf4

* matlab: h5disp and h5read

3.2.3 **Stage 3:** `$ python run.py 3`

Produce `ERA` and `MODIS` shapefiles from HDF5 subset arrays.

The shapefiles consist of a polygon grid, each raster cell represented by rectangle with attribute fields for cell id, cell center coordinates, and grid coordinates which allow cell values to be retrieved from the HDF5 array file. For example `y_ind` = 0, `x_ind` = 0 is the top left cell in the grid.

The spatial reference system of the shapefile geometries as well as the cell center coordinates is the same as the HDF5 from which the grid was derived, `MODIS` sinusoidal or `ERA WGS84`.

Requires : `STINT/Processing/HDF/<project>_<dataset>.hdf5`

Produces : `ERA` and `MODIS` shapefiles, native projections:
`STINT/Processing/SHP/<project>_era.shp`
`STINT/Processing/SHP/<project>_modis.shp`

Explore : The products of this stage are standard ESRI shapefiles that can be explored in any GIS browser. Attribute fields and projection of each file are summarized below:

modis.shp	
Spatial reference system	MODIS sinusoidal projection
id	MODIS cell id
x_ctr,y_ctr	(x, y) sinusoidal coordinates, cell center
x_ind,y_ind	grid coordinate HDF5 array file

era.shp	
Spatial reference system	WGS84 geographic coordinates
id	ERA cell id
x_ctr,y_ctr	(lon, lat) coordinates, cell center
x_ind,y_ind	grid coordinate HDF5 array file

3.2.4 **Stage 4:** `$ python run.py 4`

Transform ERA & MODIS shapefiles to AOI/landcover spatial reference system.

Construct r-tree spatial index (*.idx) for each to speed up spatial queries such as the intersection in Stage 5.

Cell center coordinates in the attribute fields y_ctr and y_ctr remain in the dataset's original spatial reference system.

Requires : STINT/Processing/SHP/<project>_era.shp
STINT/Processing/SHP/<project>_modis.shp

Produces : ERA and MODIS shapefiles with index, AOI projections:
STINT/Processing/SHP/<project>_era_reprj.shp
STINT/Processing/SHP/<project>_era_reprj.idx
STINT/Processing/SHP/<project>_modis_reprj.shp
STINT/Processing/SHP/<project>_modis_reprj.idx

Explore : The products of this stage are standard ESRI shapefiles that can be explored in any GIS browser. Attribute fields and projection of each file are summarized below:

modis_reprj.shp	
Spatial reference system	Native AOI projection
id	MODIS cell id
x_ctr, y_ctr	(x, y) sinusoidal coordinates, cell center
x_ind, y_ind	grid coordinate HDF5 array file

era_reprj.shp	
Spatial reference system	Native AOI projection
id	ERA cell id
ctr_x, ctr_y	(lon, lat) coordinates, cell center
x_ind, y_ind	grid coordinate HDF5 array file

3.2.5 **Stage 5:** `$ python run.py 5`

Intersect AOI/landcover shapefile with era_reprj.shp to produce landcover/climate shapefile, lcc.shp. Each feature in the intersection product is wholly within a single ERA climate cell and a single landcover type.

Attribute fields specified by lc_ in INPUT.txt are preserved from AOI/landcover shapefile.

Takes advantage of `era_reprj.idx` to speed up intersection.

Requires : `AOI`/landcover shapefile
`STINT/Processing/SHP/<project>_era_reprj.shp`
`STINT/Processing/SHP/<project>_era_reprj.idx`

Produces : land cover/climate shapefile with index, `AOI` projection:
`STINT/Processing/SHP/<project>_lcc.shp`
`STINT/Processing/SHP/<project>_lcc.idx`

Explore : `lcc.shp` is a standard ESRI shapefile that can be explored in any GIS browser. Attribute fields and projection are summarized below:

	`lcc.shp`
Spatial reference system	Native `AOI` projection
`id`	lcc cell id
`era_id`	`ERA` cell id
`era_ctr_x`, `era_ctr_y`	(lon, lat) coordinates, `ERA` cell center
`era_x_ind`, `era_y_ind`	grid coordinate of matching record from `ERA` HDF5 array file
`lc_id`	`AOI`/landcover feature id
`lc_*`	Any attribute fields, such as land type, selected for preservation from `AOI`/landcover shapefile

3.2.6 **Stage 6:** `$ python run.py 6`

Intersect `lcc.shp` with `modis_reprj.shp` to produce landcover/climate/modis shapefile, `lcm.shp`.

Each feature in the intersection product is wholly within a single `MODIS` cell, single `ERA` climate cell, and a single landcover type. This combination of attribute fields is illustrated in Figure 1.

Attribute fields from `lcc.shp` are preserved, as well as additionl fields identifying which MODIS cell each feature belongs to. Additionally, an Area field describes the size in square meters of each feature in the aoi projection.

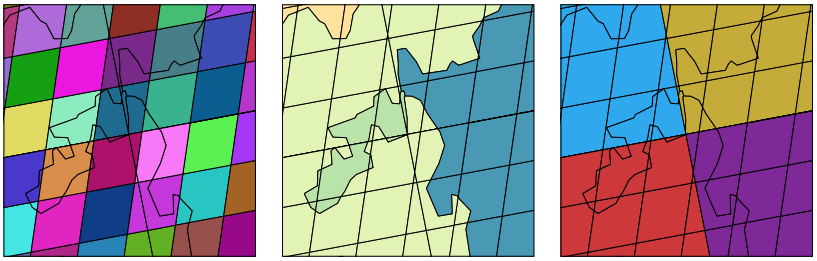
Takes advantage of `modis_reprj.idx` to speed up intersection.

`lcm.shp` is effectively the central product of this workflow, breaking each landcover feature into polygons wholly within and linked to individual MODIS and ERA cells

Requires : `STINT/Processing/SHP/<project>_modis_reprj.shp`
`STINT/Processing/SHP/<project>_modis_reprj.idx`
`STINT/Processing/SHP/<project>_lcc.shp`

Produces : landcover/climate/modis shapefile with index, `AOI` projection:
`STINT/Processing/SHP/<project>_lcm.shp`
`STINT/Processing/SHP/<project>_lcm.idx`

Explore : `lcm.shp` is a standard ESRI shapefile that can be explored in any GIS browser. Attribute fields and projection are summarized below:



(a) MODIS id (b) Landcover (c) ERA id

Figure 1: lcm.shp, shaded by different attribute fields

	lcm.shp
Spatial reference system	Native AOI projection
id	lcc cell id
area	area of feature in AOI projection, m^2
era_id	ERA cell id
era_ctr_x, era_ctr_y	(lon, lat) coordinates, ERA cell center
era_x_ind, era_y_ind	grid coordinate of matching record from ERA HDF5 array file
mod_id	MODIS cell id
mod_ctr_x, mod_ctr_y	(x, y) coordinates, MODIS cell center
mod_x_ind, mod_y_ind	grid coordinate of matching record from MODIS HDF5 array file
lc_id	AOI/landcover feature id
lc_*	Any attribute fields, such as land type, selected for preservation from AOI/landcover shapefile

3.2.7 **Stage 7:** `$ python run.py 7`

This stage exports a CSV spreadsheet file for each MODIS and ERA dataset, corresponding to the HDF5 array files. It then exports a landcover CSV which gives the area and landcover attributes of each feature in lcm.shp and links the feature to the corresponding MODIS and ERA rows.

The CSV system is basically a two dimensional representation of the HDF5 array files combined with lcm.shp. Each row in lc.csv represents a single feature in lcm.shp, with landcover attributes as well as the MODIS_id and ERA_id linked to the feature.

Each MODIS and ERA cell identified in lcm.shp is represented as a row within each of the respective dataset CSV files, with columns corresponding to MODIS intervals within the timeframe specified.

Requires : AOI/landcover shapefile
STINT/Processing/HDF/<project>_<dataset>.hdf5
STINT/Processing/SHP/<project>_lcm.shp

Produces : STINT/Output/CSV/<project>_lc.csv
STINT/Output/CSV/<project>_<dataset>.csv

Explore : lc.csv: each row is a landcover feature with MODIS and ERA links

`MODIS` and `ERA` dataset CSVs: each row is a full timeseries for an individual cell identified within `lcm.shp`.

3.3 Troubleshooting

While I tried hard to write code that catches problems and prints clear messages describing the nature of the error, there may be situations where unclear errors slip through. If such errors emerge while running `py-stint`, they will probably come in one of the following flavors:

1. Input: a problem with the format or structure of input parameters, project directories, or files linked to the project.

   Going through the INPUT.txt file and the directory structure can help resolve this kind of error.

2. File Access: a problem opening or loading the contents of a file, such as a shapefile or hdf.

   This can arise through an input error, a problem with the installation of the module being used to access the data, or a problem with the library that module depends on.

3. Data Operations: a problem manipulating or transforming data, such as managing data types, transforming spatial coordinates or geometries between projections, or sorting multiple `MODIS` arrays into a single subset array.

   This could be due to a module/library installation problem, or because of a previous failed operation. Errors do not always end the process at the point when they occur. It could be that a file access or transformation step failed earlier in the processing, but the resulting incorrect or None type object did not immediately provoke an error or Exception.

Each stage described in Section 3.2 includes a list of input files in order to help in tracking down any errors that do arise.

Appendix

A Requirements & Installation

Python 2.7 is required for `py-stint`.

On an Ubuntu system, the following commands are sufficient to set up the libraries and python modules listed in Sections A.1 and A.2.

```
$ sudo apt-get install ipython python-numpy python-pip python-scipy
python-matplotlib python-h5py python-software-properties
```

```
$ sudo add-apt-repository ppa:ubuntugis/ubuntugis-unstable
```

```
$ sudo apt-get install libspatialindex-dev python-gdal gdal-bin
```

```
$ sudo pip-install rtree pyproj
```

A.1 Library Dependencies

The following software libraries must be compiled before many of the python modules can be installed. When possible, try to compile them in the order listed, as some (such as gdal) have capabilities which depend on the previous libraries already being present.

GEOS : Required for geometric intersections.
http://trac.osgeo.org/geos/

libspatialindex : Required for rtree spatial indexing module for python.
http://libspatialindex.github.io/

netCDF4 : Required for ERA climate data and gdal support for MODIS hdf.
https://www.unidata.ucar.edu/software/netcdf/

hdf5 : Required for h5py and the format used to aggregate MODIS and ERA subset arrays.
http://www.hdfgroup.org/HDF5/

GDAL : Required for geospatial data I/O.
http://www.gdal.org/

proj.4 : required for reprojecting shapefile and raster datasets.
http://trac.osgeo.org/proj/

A.2 Python Dependencies

Basic modules are included in most installations of python. Scientific and Geospatial modules can be installed via package management, pip, or easyinstall. Geospatial modules in particular may require the libraries listed in Section A.1 to be available.

Basic Modules : os, sys, math, glob, multiprocessing, datetime, cPickle, csv

Scientific Modules : numpy, scipy, netCDF4, h5py

Geospatial Modules : gdal+ogr+osr, rtree