OpenClimateGIS
A Python Geoprocessing Framework for Climate Datasets

Presented by Ben Koziol (ben.koziol@noaa.gov)
NESII/CIRES/NOAA-ESRL*
May 2013

*NOAA Environmental Software Infrastructure and Interoperability Group / Cooperative Institute for Research in the Environmental Sciences / NOAA Earth System Research Laboratory
Acknowledgements

- Dr. Richard Rood, Project PI, Atmospheric, Oceanic, & Space Science, University of Michigan-Ann Arbor
- Dr. Tyler Erickson, Former Co-I, Google (formerly Michigan Tech Research Institute)
- K. Arthur Endsley, Web Development, Michigan Tech Research Institute
- Dr. Luca Cinquini, NESII/NASA-JPL
- Dr. Seth McGinnis, NCAR
- NCPP (National Climate Predictions and Projections) Platform
- NOAA Climate Program Office for funding support
- Developers, maintainers, and open source software advocates:
  - The HDF Group
  - Unidata
  - GDAL/OSGEO
  - netCDF4-python
  - Shapely
  - NumPy
  - Django Project
What is OpenClimateGIS?
Why was it developed?

- **Problem:** The climate modeling community and the GIS user community tend to work with very different data formats and spatial scales, which makes it difficult to integrate climate model datasets into regional analyses.
- OpenClimateGIS offers a solution to this technical barrier by translating gridded netCDF climate products into formats compatible with modern GIS and statistical software (interoperability tool).
- **Other Major Features (in addition to conversion):**
  - Built with free and open source software using Python 2.7
  - Geoprocessing (e.g. intersects, clip, and aggregation with point or polygons) and projection management
  - Temporally-grouped computations
  - Domain wrapping and unwrapping
- There are numerous high-quality climate data analysis software packages and toolkits: UV-CDAT, CDAT, NCL, R, Sage, etc.
- OpenClimateGIS focuses on extracting and translating data for web services and use in other softwares
  - Output intended for further statistical analysis, visualization, or as input to other models such as heat mortality, water resources models, etc.
Some Potentially Foreign Terminology

- **HDF** - Hierarchical Data Format
  - Self-describing binary data format used for the storage and distribution of complex, n-dimensional datasets

- **netCDF** - Network Common Data Format
  - Refinement of HDF base library which formalizes data structures and exposes a number of "user-friendly" tools

- **CF** - Climate and Forecasting Convention
  - Metadata and formatting standard for netCDF data used in climate/weather applications

- **OPeNDAP/TDS** - Open Data Access Protocol / THREDDS (Thematic Realtime Environmental Distributed Data Services) Data Server
  - Server and data frameworks often used for distributing climate data
Features of Climate Data

- Valuable!
- Varying resolutions and dimensionality
- Long time series...
- High volume...terabytes to petabytes
- Standards, compliance, and data quality specific to data provenance
- Lots of data accessible to a subset of technically trained/knowledgeable practitioners with sufficient computing resources to meet data analysis demands
- Community is aware of accessibility problem and have developed or are actively developing services and software to mitigate it
  - e.g. ESGF, NCPP, USGS Geo Data Portal, climate.gov, lcat, ferret, Live Access Server, etc.
A Climate Data File (netCDF)

- Array-oriented
- Composed of variables constructed from dimensions
- Variables and the dataset may have attributes
- CF convention provides a structure for naming, formatting, and general file organization

```plaintext
float tas(time, lat, lon) ;
    tas:standard_name = "air_temperature" ;
    tas:long_name = "Near-Surface Air Temperature" ;
    tas:units = "K" ;
    tas:original_name = "ST" ;
    tas:cell_methods = "time: mean (interval: 15 minutes)" ;
    tas:cell_measures = "area: areacella" ;
    tas:history = "2011-05-08T01:01:51Z altered by CMOR:..."

double time(time) ;
    time:bounds = "time_bnds" ;
    time:units = "days since 1850-1-1" ;
    time:calendar = "365_day" ;
    time:axis = "T" ;
    time:long_name = "time" ;
    time:standard_name = "time" ;

double lat(lat) ;
    lat:bounds = "lat_bnds" ;
    lat:units = "degrees_north" ;
    lat:axis = "Y" ;
    lat:long_name = "latitude" ;
    lat:standard_name = "latitude" ;
```

Point representations come from cell centroids constructed from row (lat) and column (lon) vector dimensions. When bounds on the row and column dimensions are present, polygon representations may be constructed.
A Word on Projections (netCDF)

```python
S1 polar_stereographic() ;
   polar_stereographic:straight_vertical_longitude_from_pole = "263.0"
;
   polar_stereographic:standard_parallel = "60.0"
   polar_stereographic:false_easting = "3475000.0"
   polar_stereographic:false_northing = "7475000.0"
   polar_stereographic:latitude_of_projection_origin = "90.0"
   polar_stereographic:grid_mapping_name = "polar_stereographic"
   polar_stereographic:resolution_at_standard_parallel = "50000.0"
   polar_stereographic:earth_radius = "6371000.0"
;
+proj=stere +lat_0=90 +lat_ts=60 +lon_0=263 +k=1
+x_0=3475000 +y_0=7475000 +ellps=WGS84 +units=m +no_defs
```

- Named variable containing projection parameters reduced to PROJ4 string and converted to OSR SpatialReference objects
- Attribute structure defined by CF convention
- This is one of the most confounding elements when working with netCDF climate data
- Proper management critical to regional modeling applications
Software Overview

- Core Design Principles:
  - Portable and open source
  - Modular to accommodate a variety of input data sources
  - Output explanatory metadata
  - Operate efficiently in a web service or script
  - Minimize data transfer
  - Operate from source data!
  - Tool-based ideology
  - Initially tuned to netCDF
Data Interface

- Each dimension exposes subset, slice, and iteration methods
- Shared data connection for loading from source
- On-demand data access - data pulled following subset attempting to minimize read times
- Customizable to the specifics of the input dataset
- netCDF4-python used for data access in the netCDF data interface - OGR for shapefile access
Subsetting (Spatial Dimension)

- **Spatial Operations:**
  - Intersects
  - Clip/Intersection
  - Aggregation/Union
- **Projection handling**
- **Attempts to optimize spatial operations through easily available means:**
  - Simple spatial index
  - Array operations where appropriate
  - Perform "contains" before "intersection"
  - Geometry preparations
- **Vector geometries stored in masked NumPy arrays as Shapely geometries**
- **Aggregation implies area-weighting and used at the dataset-level**
Computation

- Built around the idea of climate indices and/or metrics
  - Annual descriptive statistics from daily inputs
  - Daily exceedances over a threshold grouped by month
  - Heat indices
- Atomic at the level of a temporal group
- Calculations abstracted into general categories:
  - Functions with no parameters
  - Parameterized functions
  - Multivariate functions
- All calculations have:
  - Descriptive attributes
  - Spatial and temporal aggregation behaviors
Conversion & Metadata

● Conversion:
  ○ Data streamed through a series of iterators yielding dictionary representations of data rows
  ○ Standard set of headers for raw or derived (computed) data packages
  ○ Only limited by Python for conversion options
  ○ Shapefile, CSV (also "linked" version), and netCDF currently available -- adding additional converters on an as-needed basis
  ○ Also NumPy-based dictionary output for use in scripting, etc.

● Metadata:
  ○ Provenance is very important for climate science and contributes to establishing "salience" - the data must be relevant to the problem otherwise it is so much noise!
  ○ Maintaining the "chain of custody" for data operations is hence a much needed feature
  ○ OCGIS does track operations and records them in an output text file -- more formats and increased verbosity desirable
An Explanatory Use Case
More Explanation

- Data is from NARCCAP\(^1\) (North American Regional Climate Change Assessment Program)
  - Example data and structure for use case provided by Seth McGinnis, NCAR (National Center for Atmospheric Research)
  - NARCCAP provides a series of RCMs (regional climate models) nested in atmosphere-ocean global circulation models (AOGCMs)
  - For this use case, the data is composed of:
    - 11 GCM-RCM combinations
    - ~50 km spatial resolution
    - Ten years of data at 3-hour intervals
    - Multiple projections and grid definitions
    - Temperature and precipitation variables
- Extract model output nearest a city centroid
- Perform a series of temporally aggregated calculations
  - This example computed descriptive statistics grouped by variable, month, year, and RCM-AOGCM (i.e. mean, median, min, max)
- Write results to a shapefile with a common projection

1. [http://www.narccap.ucar.edu/](http://www.narccap.ucar.edu/)
Models used in NARCCAP do not share the same coordinate systems leading to the staggered appearance of the points in the graphic. For the graphic in the upper left, the points following closest to the selection geometry (in red) are enclosed in the black polygon.
Basic Overview of Python API

1. Set environment variables
2. Construct request dataset objects comprised of:
   a. Data location - OPeNDAP URLs or paths to data at locally accessible location
   b. Target variable
   c. Additional parameters to overload if not readable from input data (e.g. calendar, projection)
3. Specify subset parameters
   a. Time or level ranges
   b. Selection geometry (WGS84 geographic) - point, bounding box, or shapefile
4. Set any other operational parameters (e.g. geometric abstraction, spatial operation, output format)

```python
ocgis.env.WRITE_TO_REFERENCE_PROJECTION = True
calc = [{'func': 'mean', 'name': 'mean'},
        {'func': 'median', 'name': 'median'},
        {'func': 'max', 'name': 'max'},
        {'func': 'min', 'name': 'min'}]
calc_grouping = ['month', 'year']
ops = ocgis.OcgOperations(dataset=rds, calc=calc, calc_grouping=calc_grouping,
                          output_format='shp', geom=[-97.74278, 30.26694],
                          abstraction='point')
ret = ops.execute()
```
Results

- In the case of a shapefile output, the software returns a path to the shapefile contained in a folder with additional files:
  - Metadata output file
  - Dataset descriptor CSV file
  - Shapefile representation of selection geometry
- Shapefile output okay in this case - many time slices + complex geometries results in large, inefficiently stored data (CSV may make more sense)
Another Example

For a HUC8 watershed geometry in Colorado, aggregate coincident point data and calculate descriptive statistics for the model combinations (intercomparison). Write data to CSV file which will minimize size by not repeating the geometry for every shapefile row. (Thanks to Joe Barsugli for use case suggestion.)
OpenClimateGIS v0.05b-dev Metadata File
Generated (UTC): 2013-05-16 01:04:17.462648

== Potential Header Names with Definitions ==

ALIAS :: Dataset alias - may be equivalent to VARIABLE.
CALC_NAME :: User-supplied name for a calculation.
CID :: Unique identifier for a calculation name.
DAY :: Day extracted from time string.
DID :: Dataset identifier see *_did.csv file for additional information on dataset requests.
GID :: Geometry identifier assigned by OpenClimateGIS to a dataset geometry. In the case of "aggregate=True" this is equivalent to "UGID".
LEVEL :: Level name.
LID :: Level identifier unique within a variable.
MONTH :: Month extracted from time string.
TGID :: Unique grouped time identifier.
TID :: Unique time identifier.
TIME :: Time string.
UGID :: User geometry identifier pulled from a provided set of selection geometries. Reduces to "1" for the case of no provided geometry.
URI :: Path to input data at execution time.
VALUE :: Value associated with a variable or calculation.
VARIABLE :: Name of request variable.
VID :: Unique variable identifier.
VLID :: Globally unique level identifier.
YEAR :: Year extracted from time string.

== Argument Definitions and Content Descriptions ==

abstraction=point
   Spatial dimension abstracted to point.

agg_selection=False
   Selection geometries left as is.

aggregate=False
   Selected geometries are not aggregated (unioned).

allow_empty=False
   Empty returns NOT allowed. If a selection geometry has no intersecting geometries from the target dataset, an exception is raised.
What's Next?

- Integration with NESII's regridding and coupling software ESMF (Earth System Modeling Framework) via its Python interface ESMPy
- Remote access to climate data in ESGF (Earth System Grid Federation)
- Interactive web interface
- Move software out of beta (3 to 6 months)
- Expand climate index catalog through collaboration with NCPP
- Support for OGC-compliant services - specifically WPS
- Bring in Fiona\(^1\) for I/O operations with OGC data formats
- Improve handling of operations on large data files - prototype support for large array computations
- Continue documentation and usability improvements - lots of moving parts
- Applications, collaborations, users, and developers are welcome!
  - Addition of output formats, etc. specific to user/application needs
  - Plugins for other software packages

1. [https://github.com/sgillies/Fiona](https://github.com/sgillies/Fiona)
Links and Contact Information

- NCPP: http://earthsystemcog.org/projects/ncpp/
- OpenClimateGIS Documentation: http://ncpp.github.io/ocgis/
- GitHub Code Repository: https://github.com/NCPP/ocgis
- NESII Group: http://www.esrl.noaa.gov/nesii/
- OpenClimateGIS Support Email List: ocgis_support@list.woc.noaa.gov