OpenClimateGIS: A Python Library for Geospatial Manipulations of Climate Datasets

Ben Koziol (NESII/CIRES/NOAA-ESRL/GSD)
February 2018
NESII Group Overview

● NESII → NOAA Environmental Software Infrastructure & Interoperability Group

● NESII builds software infrastructure for Earth system modeling, data analysis, and scientific collaboration using open source, community development approaches

● NESII has been at ESRL / CIRES since November, 2009 - formerly the Earth System Modeling Infrastructure section at the National Center for Atmospheric Research

● Joined Global Systems Divisions at ESRL in 2016

● Partners and customers are from research and operational centers, weather and climate, across U.S. agencies and international organizations
Presentation Outline

1. Overview
2. Subsetting
3. Structured Computation
4. Data Format Conversion
5. Next Steps
6. Notebook Demo
What is OpenClimateGIS?

- OpenClimateGIS (OCGIS) is a pure Python, open source software library enabling dynamic access to and manipulation of climate data
- Software goal is to overcome barriers of usability of climate projections in adaptation planning and resource management
  - Translate out of climate data formats
  - Select geographical regions of interest
  - Select times/levels of interest
  - Compute application-relevant indices
  - Convert to end-user and analysis-ready formats
  - Provide comprehensive metadata
  - Enable access to big, high-resolution climate data files
- Builds on numerous open source software libraries:

<table>
<thead>
<tr>
<th>Required</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>netCDF4</td>
<td>rtree</td>
</tr>
<tr>
<td>numpy</td>
<td>cf_units</td>
</tr>
<tr>
<td>shapely</td>
<td>ESMF</td>
</tr>
<tr>
<td>fiona</td>
<td>mpi4py</td>
</tr>
<tr>
<td>osgeo</td>
<td></td>
</tr>
</tbody>
</table>
Software Architecture

- Written in pure Python
- Modular/extensible design for data interface, format conversion, and computations
- Operations manipulate coordinate variables to limit the amount of “value data” requested
- Built with generator functions at the operations API
- With few exceptions, all operations parallelized with MPI

* This is the library used for reading NetCDF. Depending on the input data format, other data APIs may be desirable (i.e. OGR for vector formats).
Subsetting

- Handles many types of geospatial subsetting:
  - Points
  - Arbitrary Polygons
  - Lines
  - Bounding Boxes
  - Multi-geometries

- Reads geometries directly from OGC and ESRI formats (anything that supports Python)

- Temporal subsetting - time ranges or “regions” (i.e. arbitrary month and year combinations)

- Level subsetting - lower and upper bounds

- Label-based slicing

- Reads and writes CF and PROJ.4 coordinate systems

- Wrapping and unwrapping for 360 geographic coordinate systems
Structured Computation

- Framework designed to accommodate a variety of climate indices and metrics:
  - Temporally grouped functions → monthly means, annual maximums, seasonal aggregations
  - String-based functions → ‘diff=tasmax-tasmin’
  - Simple transforms → natural logarithm
  - Multivariate functions → heat indices

- Provide a “straightforward” and “painless” method for introducing new indices

- Integrated ICCLIM ECA indices

```python
class ocgis.calc.library.index.heat_index.HeatIndex(*args, **kwds):
    # source


calculate(tas=None, rhs=None, units=None)
    # source
```

```python
def calculate(self, tas=None, rhs=None, units=None):
    if units == 'k':
        tas = 1.8*(tas - 273.15) + 32
    else:
        raise(NotImplementedError)

c1 = -42.379
C2 = 2.04901523
C3 = 10.14333127
C4 = -0.22475341
C5 = -6.837836e-3
C6 = -5.681717e-2
C7 = 1.22874e-3
C8 = 8.5282e-4
C9 = -1.99e-6
idz = tas < 88
if idz:
    tas_mask = np.logical_or(idx, tas.mask)
    if rhs:
        rhs_mask = np.logical_or(idx, rhs.mask)
        rhssq = np.square(rhs)
    else:
        rhssq = np.square(rhs)
```
Data Format Conversion

- A general framework for data conversion for streaming to multiple formats
- Sequential parallel writes implemented for all formats.
  a. Asynchronous writes available for NetCDF provided its built with parallel support
- Common set of headers for tabular output files that may be adjusted to suit a user’s needs (i.e. a user may only be interested in a timestamp and associated data value)
- Uses Fiona to write to GIS vector formats
- Currently supported formats: CSV, Keyed-CSV Shapefile, GeoJSON, netCDF (with and without geometries!), ESRI Shapefile, array-based collections
  a. NetCDF Metadata Interpretation → CF-Grid, UGRID, SCRIP
Some Recent OCGIS Developments

- **National Hydrography Dataset Conservative Regridding**
  - **Motivation**: Develop and profile a high performance regridding solution for complex, irregular meshes and fine resolution rectangular grids over large scales.

- **Chunked Regridding**
  - **Motivation**: Develop software infrastructure for grid manipulations that scale to high spatial resolutions and accuracies while accommodating arbitrary compute environments and grid structures.
Next Steps

- **2.1.0 (March 2018)**
  - UGRID/SCRIP/ESMF Unstructured Format
  - Asynchronous NetCDF IO (adds write capability)
  - Chunked regrid weight generation for big grids
  - Memory and performance improvements

- **2.2.0 (...future...)**
  - Memory tiling for computations
  - Redistribution
  - Parallel regridding w/ ESMF mesh support
  - CF-Discrete Sampling Geometries
  - ?
Contacts & Links

● Questions, comments, suggestions, or “hidden features”:
  ○ ocgis_support@list.woc.noaa.gov or https://github.com/NCPP/ocgis/issues

● Mailing lists and releases:
  ○ ocgis_info@list.woc.noaa.gov

● Software links:
  ○ http://www.earthsystemcog.org/projects/openclimategis/
  ○ http://www.earthsystemcog.org/projects/esmpy/
  ○ http://www.esrl.noaa.gov/nesii/
Backup Slides
Status

- Current Release: 2.0.0 (2.1.0 planned in coming months)
- Project is fully open source under the University of Illinois-NCSA License ([http://opensource.org/licenses/NCSA](http://opensource.org/licenses/NCSA))
- Hosted on GitHub: [https://github.com/NCPP/ocgis](https://github.com/NCPP/ocgis)
- Mature test harness (1000+ unit tests)
import ocgis

ops = ocgis.OcgOperations(dataset={'uri': '/data/tas_kelvin.nc'},
                          time_region={'month': [6, 7, 8]},
                          geom=[-121, 38, -122, 40],
                          conform_units_to='celsius',
                          output_format='nc')

path = ops.execute()
ESMPy Overview

- Python interface to the high performance, parallel regridding functionality of ESMF - Uses NumPy for data array management

- Supported coordinate representations:
  - **Grid**: 2D/3D, logically rectangular, regional/global, stagger options
  - **Mesh**: 2D/3D unstructured
  - **LocStream**: 2D/3D unconnected points (point cloud)

- Source data is represented with **Field**, built on a **Grid**, **Mesh** or **LocStream**

- Regridding uses two **Fields** (source and destination)

- Methods include first order conservative, bilinear, nearest neighbor, and more

- Data may be read directly from file. Formats include Gridspec, UGRID, and SCRIP

- Other notable features include masking, ignoring unmapped points, options for line paths between points, and a variety of pole handling capabilities
ESMPy Integration

- OCGIS has been integrated with ESMPy to support bilinear and first order conservative regridding between structured grids
- Regridding is part of the operations “ecosystem” and may be chained with subsetting, etc.
- Current development is adding support for meshes and location streams in addition to grids
  - Use mesh regridding in place of the nonoptimal spatial averaging algorithms inside OCGIS
  - Use location stream for unstructured/observational regridding sources and targets
- With the release of ESMPy 7.0, ESMPy fields will be interoperable with OCGIS fields - proof-of-concept code in feature branch
Extensibility

**Example Calculation Subclassing**

```python
class Min(base.AbstractUnivariateSetFunction):
    description = 'Min value for the series.'
    key = 'min'
    dtype = constants.np_float

    def calculate(self, values):
        return np.ma.min(values, axis=0)

class Max(base.AbstractUnivariateSetFunction):
    description = 'Max value for the series.'
    key = 'max'
    dtype = constants.np_float

    def calculate(self, values):
        return np.ma.max(values, axis=0)
```

**Example NetCDF Data Reading**

```python
class NcVectorDimension(VectorDimension):
    def _set_value_from_source_(self):
        try:
            ds = self._data._open()
            var = ds.variables[self.meta['name']]
        except KeyErr as e:
            if self.axis == 'R' :
                self.src_idx += 1
                var = self._src_idx + 1
            else:
                ocsis_lh(logger='interface.nc', exc=e)
```

Dataset Bundling

- Bundles or packages are groups of data over which to apply a common set of operations → idea is to extend ensembles

- OCGIS consolidates coordinate systems for the datasets and subset geometry(s) and applies selected operations to each in sequence

- The example data displayed below is from a CSV output from three datasets:
  a. CMIP5 Decadal Simulation (3 degrees, 360 lat/lon)
  b. NARCCAP CRCM-CGCM3 (50 km, Polar Stereographic)
  c. Maurer Gridded Observational (⅛ degrees, 180 lat/lon)

- Example description:
  a. Pull out all January dates
  b. Spatially subset and area-weight the values for grid cells intersecting the Nebraska state boundary
  c. Calculate the monthly mean and standard deviation
  d. Write data to CSV

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>ALIAS</th>
<th>CALC_KEY</th>
<th>CALC_ALIAS</th>
<th>TIME</th>
<th>YEAR</th>
<th>MONTH</th>
<th>DAY</th>
<th>LEVEL</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>tas</td>
<td>cmip5_tas</td>
<td>mean</td>
<td>mean_cmip5_tas</td>
<td>2001-01-16 00:00:00</td>
<td>2001</td>
<td>1</td>
<td>16</td>
<td></td>
<td>271.785</td>
</tr>
<tr>
<td>tas</td>
<td>cmip5_tas</td>
<td>std</td>
<td>stdev_cmip5_tas</td>
<td>2001-01-16 00:00:00</td>
<td>2001</td>
<td>1</td>
<td>16</td>
<td></td>
<td>5.59953</td>
</tr>
<tr>
<td>pr</td>
<td>narccap_pr</td>
<td>mean</td>
<td>mean_narccap_pr</td>
<td>1981-01-16 00:00:00</td>
<td>1981</td>
<td>1</td>
<td>16</td>
<td></td>
<td>1.0143e-05</td>
</tr>
<tr>
<td>pr</td>
<td>narccap_pr</td>
<td>std</td>
<td>stdev_narccap_pr</td>
<td>1981-01-16 00:00:00</td>
<td>1981</td>
<td>1</td>
<td>16</td>
<td></td>
<td>4.5889e-06</td>
</tr>
<tr>
<td>pr</td>
<td>narccap_pr</td>
<td>mean</td>
<td>mean_narccap_pr</td>
<td>1981-02-16 00:00:00</td>
<td>1981</td>
<td>1</td>
<td>16</td>
<td></td>
<td>2.38535e-05</td>
</tr>
<tr>
<td>pr</td>
<td>narccap_pr</td>
<td>std</td>
<td>stdev_narccap_pr</td>
<td>1981-02-16 00:00:00</td>
<td>1981</td>
<td>1</td>
<td>16</td>
<td></td>
<td>3.57291e-06</td>
</tr>
<tr>
<td>tasmax</td>
<td>mauer_tasmax</td>
<td>mean</td>
<td>mean_maurer_tasmax</td>
<td>1971-01-16 00:00:00</td>
<td>1971</td>
<td>1</td>
<td>16</td>
<td></td>
<td>1.37631</td>
</tr>
<tr>
<td>tasmax</td>
<td>mauer_tasmax</td>
<td>std</td>
<td>stdev_maurer_tasmax</td>
<td>1971-01-16 00:00:00</td>
<td>1971</td>
<td>1</td>
<td>16</td>
<td></td>
<td>7.20884</td>
</tr>
</tbody>
</table>
Core Capabilities of OpenClimateGIS

- Read local or remotely served (i.e. OPeNDAP) ~CF-compliant netCDF datasets
- Geospatial subsetting by arbitrary vector geometries (e.g. watersheds) and time/level bounds
- Common spatial operations such as intersects, clip, and aggregation on point or polygon (e.g. bounded coordinates) data representations
- Geometry wrapping and unwrapping to maintain a “GIS-friendly” -180 to 180 longitudinal spatial domain
- Support for geographic (e.g. latitude/longitude) and projected climate datasets (e.g. Lambert Conformal)
- Option to apply temporally-grouped computations to data subsets
- Write climate data to GIS and tabular formats