The PyNIO multi-format scientific data I/O module

SEA2015 • April 16, 2015

Dave Brown • Mary Haley
Wei Huang • Rick Brownrigg

CISL/TDD/DVAT
(Data Visualization and Analysis Tools)
• Develop geoscientific analysis and visualization software in close collaboration with climate and weather scientists
• Create extensive examples and documentation
• Answer user questions on a daily (hourly!) basis
• Provide hands-on training workshops 4-6x a year
• Collaborate with outside groups and users to enhance software
PyNIO tutorial schedule

- Introduction to scientific data (10 minutes)
- Introduction to PyNIO (15 minutes)
- Demos and exercises (1.5 hours)
Quick survey

• What types of data do you work with?
  • ASCII, CSV
  • Binary (Fortran, C)
  • Excel
  • Self-describing (NetCDF, HDF, GRIB, Shapefile)
  • Not sure

• What programming software do you currently use?
  • Fortran, C, C++
  • Scripting languages: MATLAB, IDL, Python, R, Ruby, Java, NCL, etc.
Effectively managing data is critical to the science we do

NCAR invests quite a bit of time and money researching ways to improve on all aspects of data creation, management, analysis, maintenance, visualization, integration, etc.

The tools DVAT develops are mainly tuned to the data formats used by climate and weather researchers.
Definition: data format

The organization of information (data) according to preset specifications.
Types of data formats we use:

- ASCII

- Binary (Fortran sequential, . . .)

- Self-describing formats

  - This is where PyNIO comes in...
Data formats: self-describing

- Self-describing data formats are files that contain data and descriptive information about the data ("metadata")

- Metadata is information about the file itself and/or about the variables on the file
Data formats: **self-describing**

Metadata generally includes:

- **Attributes**
  - Describes “attributes” of the file or variable
  - “creation_date”, “long_name”, “units”

- **Dimension names and sizes**
  - “time” “level” “lat” “lon”

- **Coordinate information**
  - Latitude/Longitude arrays, time array, level array
Pros of self-describing formats

- Well-written files describe themselves
- Query file for information
- Essential for subsetting and aggregation
Cons of self-describing formats

• Complex – requires special software to read/write
• Some formats still evolving
• Can be large
• Standards not always adopted

“Look at your data”
PyNIO supports these formats:

- NetCDF-3 / NetCDF-4
- HDF4 / HDF5
- HDF-EOS2 / HDF-EOS5
- GRIB1 / GRIB2
- Shapefiles
Self-describing data formats

**NetCDF (Network Common Data Form)**

- Very common in climate sciences
- Developed and supported by Unidata
- Two versions: NetCDF-3 and NetCDF-4

http://www.unidata.ucar.edu/software/netcdf/
Self-describing data formats

HDF (Hierarchical Data Format)

• Tailored for large and complex datasets
• Used by a wide variety of scientific disciplines
• Two versions HDF4 / HDF5

http://www.hdfgroup.org
Self-describing data formats

**HDF-EOS (HDF Earth Observing System)**

- HDF4 and HDF5 subset with conventions, data types, and metadata
- Used for NASA EOS missions (mostly satellite)

Geo-located data

http://hdfeos.net
Self-describing data formats

**GRIB (Gridded Binary)**

- World Meteorological Organization standard
- Historical / forecast weather data
- Actually a record format
- Requires supplemental files to describe data

General Regularly-distributed Information in Binary form
Self-describing data formats

**Shapefiles**

- ESRI/GIS format
- Can be points, lines, polygons
- Example: population, roads, country boundaries, election data, airport locations
- Used for graphics, masking data
Climate Data Guide

The Climate Data Guide provides concise and reliable information on the strengths and limitations of the key observational data sets, tools and methods used to evaluate Earth system models and to understand the climate system.

https://climatedataguide.ucar.edu/

Includes information about tools useful for climate research
Examples of using Shapefiles to mask data

Original data with shapefile outlines

Original data masked against land

Original data masked against ocean

Data averaged over counties in Georgia

2005-10-11 17:24:27

Wind Speed: 10m

Wind Power Classification
- Weak: 0.00-5.4
- Moderate: 5.5-13.8
- Strong: 13.9-27.9
- Extreme: Over 27.9
What is PyNIO?

- Community-supported free open source Python tool for accessing multiple scientific data formats
- Based on well-tested and supported NCL data I/O code
- Interfaces with numpy
- A single call accesses all supported formats
- Handles data with missing values, e.g.:
  - Ocean data where no data exists on land
  - No data because of lack of instrumentation in remote areas
  - Instrument failure for some period of time
PyNIO: current state

- Current release (1.4.1) is circa 2011 – pretty old now
- Solid support for the NetCDF3 interface
  - Including NetCDF4 classic (adds compression)
- Other formats appear like NetCDF3:
  - GRIB 1 & 2, HDF, HDF-EOS and shapefiles
- All may have:
  - global attributes
  - shared dimensions (one may be ‘unlimited’)
  - variables
  - variable-specific attributes
  - coordinate variables for locating the data in space
PyNIO 1.5.0 almost ready

- Nearly complete support for NetCDF4
  - Groups, multiple unlimited dimensions, variable-len arrays, compound variables
- And most features of HDF5
- Many improvements for other formats
- Interface similar to netcdf4-python
  - Both modeled on ScientificIO module from Konrad Hinson
- Main type classes: NioFile and NioVariable
>>> import Nio
>>> print Nio.__doc__
....
Class NioFile:
f = Nio.open_file(filepath, mode='r', options=None, history='', format='')
attributes:
    name -- the name of the file or group
dimensions -- dictionary of dimension lengths with dimension name keys
variables -- dictionary of variable objects with variable name keys
attributes -- dictionary of global file or group attributes with attribute name keys
    (the following are applicable for advanced formats NetCDF4 and HDF5 only)
groups -- dictionary of group objects with group name keys
ud_types -- dictionary of user-defined data type definitions with data type name keys
chunk_dimensions -- dictionary of chunking dimension sizes with dimension name keys
parent -- reference to the parent group, parent file for the root group, or None for a file
path -- the path of a group relative to the root group (’/’), or the file path for a file

* Red text indicates new feature for PyNIO 1.5.0
* Dark red means not working yet
Methods:
close(history='') -- close the file
create_dimension(name, length) -- create a dimension in the file
create_variable(name, type, dimensions) -- create a variable in the file
unlimited(dimension_name) -- returns True if dimension_name refers to an unlimited dimension; False otherwise

(the following are applicable for advanced formats NetCDF4 and HDF5 only)
create_group(name) -- create a group in the file or group.
create_vlen(name,type,dimensions) -- create a variable length array variable in the file or group.
create_compound(name,type,dimensions) -- create a compound variable with the given type and dimensions.
create_compound_type(name, type) -- create a user-defined compound type; with member names, sizes, and types as defined in the type sequence argument.
NioVariable

Attributes:
- rank -- a scalar value indicating the number of dimensions
- shape -- a tuple containing the number of elements in each dimension
- dimensions -- a tuple containing the dimensions names in order
- attributes -- a dictionary of variable attributes with attribute name keys
- size -- a scalar value indicating the size in bytes of the variable
- name -- the name of the variable
- parent -- reference to the group or file to which the variable belongs
- path -- the path of the variable relative to the root group ('/')

Methods:
- assign_value(value) -- assign a value to a variable in the file.
- get_value() -- retrieve the value of a variable in the file.
- typecode() -- return a character code representing the variable's type.
- set_option(option,value) -- set certain options.
Groups

- Groups are NioFile class types
- Just like file variables except their path is relative to the root group (‘/’)
- And they have a parent
- Both a flat representation and a hierarchical directory-like representation of the groups
- Variables, dimensions and attributes have a similar scheme
Both flat and hierarchical
Same object in both cases

```python
>>> import Nio
>>> f = Nio.open_file('tnested.nc')
>>> g = f.groups['forecasts/model1']
>>> g1 = f.groups['forecasts'].groups['model1']
>>> print f.groups is f.groups['forecasts'].groups
False
>>> g1.name
'model1'
>>> g.name
'model1'
>>> g1.parent.name
'forecasts'
>>> g.parent.name
'forecasts'
>>> print g is g1
True
```
Variables

```python
>>> print g.variables.keys()
['lat', 'time', 'lon', 'temp', 'level']

>>> print f.variables.keys()
```
Future

• Finish 1.5.0:
  – Better packaging
  – Compound data, HDF5 issues, testing, fixing
  – Planned coordinate mesh attribute with aligned slicing across all dimensions and data variable
    • Rectangular, curvilinear, & unstructured

• Post 1.5.0:
  – Support for parallel writes
  – Support for in memory files
http://www.pyngl.ucar.edu/Training/SEA2015/
http://www.pyngl.ucar.edu/Nio.shtml

Questions?