fixnc Documentation

Release 0.0.1

Nikolay Koldunov

Nov 28, 2018

Contents

fixnc helps to change meta information of the netCDF files. You can easilly add, delete and rename dimensions, variables and attributes.

CHAPTER 1

Quick start:

In the *tests* directory you will find a netCDF file *test.nc*, that have *X*, *Y* and *T* dimensions.:

```
netcdf test {
dimensions:
    X = 10 ;
    Y = 10 ;
    T = UNLIMITED ; // (5 currently)
variables:
    float T(T) ;
        T:unuts = "hours since 2001-01-01 00:00:00" ;
    float mytemp(T, X, Y) ;
        mytemp:longname = "Temperature" ;
        mytemp:shortname = "temp" ;
}
```

We would like to change the names of this dimensions to lon lat and time:

```
import fixnc as fnc
from netCDF4 import Dataset

fl = Dataset('./tests/test.nc') # create netCDF4 instance
nc = fnc.ncfile(fl)             # create ncfile instance, that is just a collection
                                # of ordered dictionaries.
# rename dimensions
nc.rename_dim('X','lon')
nc.rename_dim('Y','lat')
nc.rename_dim('T','time',)

# save output
nc.save('out.nc')
```

This should generate a new netCDF file (*out.nc*), that have exactly the same content as the original one, but with dimension names changed.:

```
netcdf out {
dimensions:
        lon = 10 ;
        lat = 10 ;
        time = UNLIMITED ; // (5 currently)
variables:
        float T(time) ;
                T:unuts = "hours since 2001-01-01 00:00:00" ;
        float mytemp(time, lon, lat) ;
                mytemp:longname = "Temperature" ;
                mytemp:shortname = "temp" ;
}
```

In this case dimension names in the variables will be also changed.

To add an at:tribute to T variable you simply:

```
nc.add_attr('T','standard_name', 'time')
```

CHAPTER 2

Documentation

2.1 Installation

2.1.1 Required dependencies

- netCDF4
- sh

2.1.2 Installation

If you are lucky simple

```
pip install fixnc
```

should install dependencies and fixnc itself. However there is a big chance, that netcdf4 will have some problems.

The better way is to first install netcdf4 with Miniconda and then pip install fixnc.

And you always can just clone github repository .

2.2 Basic Tutorial

Let't open some netCDF file with netCDF4. We will use *test.nc* located in *tests* directory

```
In [1]: from netCDF4 import Dataset
In [2]: fl = Dataset('./tests/test.nc')
```

And create ncfile instance:

```
In [3]: from fixnc import ncfile
In [4]: nc = ncfile(fl)
```

Here is how the header of the netCDF file will look like:

```
In [5]: nc
Out[5]:
File format: NETCDF4
Dimentions: X(10), Y(10), T(5)
variables:
         float32 T(T)
           unuts: hours since 2001-01-01 00:00:00
         float32 mytemp(T, X, Y)
           longname: Temperature
           shortname: temp
```

We would like to rename dimentions, update variable names and add some attributes. The netCDF header is represented as set of Ordered dictionaries. For example here are our dimentions:

```
In [6]: nc.dims
Out[6]:
OrderedDict([(u'X',
              OrderedDict([('name', u'X'),
                           ('size', 10),
                           ('isunlimited', False)])),
             (u'Y',
              OrderedDict([('name', u'Y'),
                           ('size', 10),
                           ('isunlimited', False)])),
             (u'T',
              OrderedDict([('name', u'T'),
                           ('size', 5),
                           ('isunlimited', True)]))])
```

We can change the names of dimentions with **rename_dim** method:

```
In [7]: nc.rename_dim('X','lon')
   ...: nc.rename_dim('Y','lat')
   ...: nc.rename_dim('T','time',)
   ...: nc
   ...:
Out[7]:
File format: NETCDF4
Dimentions: lon(10), lat(10), time(5)
variables:
         float32 T(time)
           unuts: hours since 2001-01-01 00:00:00
         float32 mytemp(time, lon, lat)
           longname: Temperature
           shortname: temp
```

Note, that names of the dimentions in the description of the variables are also changed. You can change this behaviour by setting *renameall=False*.

Now rename variables:

```
In [8]: nc.rename_var('T','time')
   ...: nc.rename_var('mytemp', 'temp')
   ...: nc
   ...:
Out[8]:
```

(continues on next page)

(continued from previous page)

```
File format: NETCDF4
Dimentions: lon(10), lat(10), time(5)
variables:
         float32 time(time)
           unuts: hours since 2001-01-01 00:00:00
         float32 temp(time, lon, lat)
           longname: Temperature
           shortname: temp
```

Note that unit attribute for time is wrong, let's fix it:

```
In [9]: nc.rename_attr('time','unuts','units')
   ...: nc
   ...:
Out[9]:
File format: NETCDF4
Dimentions: lon(10), lat(10), time(5)
variables:
         float32 time(time)
           units: hours since 2001-01-01 00:00:00
         float32 temp(time, lon, lat)
           longname: Temperature
           shortname: temp
```

Add a bit more information about time by providing additional attributes:

```
In [10]: nc.add_attr('time','standard_name', 'time')
    ...: nc.add_attr('time','calendar','proleptic_gregorian')
    ...: nc
    ...:
Out[10]:
File format: NETCDF4
Dimentions: lon(10), lat(10), time(5)
variables:
         float32 time(time)
           units: hours since 2001-01-01 00:00:00
           standard_name: time
           calendar: proleptic_gregorian
         float32 temp(time, lon, lat)
           longname: Temperature
           shortname: temp
```

And add some global attribute as well:

```
In [11]: nc.add_gattr('history','fixed with fixnc')
    ...: nc
    ...:
Out[11]:
File format: NETCDF4
Dimentions: lon(10), lat(10), time(5)
variables:
         float32 time(time)
           units: hours since 2001-01-01 00:00:00
           standard_name: time
           calendar: proleptic_gregorian
         float32 temp(time, lon, lat)
           longname: Temperature
```

(continues on next page)

(continued from previous page)

```
            shortname: temp

          history:fixed with fixnc
```

Now we can save the result:

```
In [12]: nc.save('out.nc')
```

And compare once again the original and the resulting files:

```
In [14]: !ncdump -h ./tests/test.nc
netcdf test {
dimensions:
        X = 10 ;
        Y = 10 ;
        T = UNLIMITED ; // (5 currently)
variables:
        float T(T) ;
                T:unuts = "hours since 2001-01-01 00:00:00" ;
        float mytemp(T, X, Y) ;
                mytemp:longname = "Temperature" ;
                mytemp:shortname = "temp" ;
}
```

```
In [15]: !ncdump -h ./out.nc
netcdf out {
dimensions:
        lon = 10 ;
        lat = 10 ;
        time = UNLIMITED ; // (5 currently)
variables:
        float time(time) ;
                time:units = "hours since 2001-01-01 00:00:00" ;
                time:standard_name = "time" ;
                time:calendar = "proleptic_gregorian" ;
        float temp(time, lon, lat) ;
                temp:longname = "Temperature" ;
                temp:shortname = "temp" ;

// global attributes:
                :history = "fixed with fixnc" ;
}
```

2.3 API

fixnc.**create_variable**(*data*, *dimensions*, *hasunlimdim=False*, *datatype='float32'*, *FillValue=None*, *attributes={}*)

Creates dictionary that can be added as a variable to the netCDF file.

Create dictionary that contains information necessary for creation of the netCDF variable.

Parameters

- **data** (*array-like*) – Numpy array, array-like object that contains the actual data values.

- **dimensions** (*tuple*) – tuple with dimension names, like ('time', 'lat', 'lon'). dimensions should exist in the source file, or should be added in the ncfile object.
- **hasunlimdim** (*bool*) – True if variable have unlimited dimension, otherwise False. !!NOTE!! At present unlimited dimension in your new variable has to be the same size as in the original data (e.g. number of time steps). This should be changed.
- **datatype** (*datatype*) – numpy datatype as a string, like "float32".
- **FillValue** (*number, optional*) – If your data should have one, otherwise None
- **attributes** (*OrderedDict*) – Orderd dictionary with attributes and their values.

Returns Ordered dictionary that can be used to add data to netCDF file.

Return type OrderedDict

fixnc.**dump_variable**(*var*, *filename*)
Use *pickle* to dump OrderedDict to the file.

Parameters

- **var** (*OrderedDict*) – OrderedDict, supposedly produced by *create_variable* function.
- **filename** (*str*) – name of the file.

Returns True is succes.

Return type bool

fixnc.**load_variable**(*filename*)
Use *pickle* load OrderedDict from the file to the variable.

Parameters **filename** (*str*) – name of the file.

Returns

Return type OrderedDict

class fixnc.**ncfile**(*ifile*)
Main class of the fixnc.

This class is initiated with original netCDF file object created by Dataset from the netCDF4 package. The properties of the file are copied to the attributes of the class and cna be then saved together with data of the original file. The purpose is to be able to fix metadata in the netCDF file, like dimension names, attributes and so on, but to save the rest of the structure of the original file as much as possible.

Initial version of the class is based on the code from netCDF4 library https://github.com/Unidata/netcdf4-python/blob/master/netCDF4/utils.py

Parameters **ifile** (*Dataset*) – Instance of the Dataset class from the netCDF4 library.

add_attr(*var*, *attr*, *value*)
Add attribute to the variable.

Parameters

- **var** (*str*) – Name of the variable.
- **attr** (*str*) – Name of the new attribute.
- **value** (*str*) – Value of the new attribute.

add_dim(*name*, *size*, *isunlimited=False*)
Add dimension.

Parameters

- **name** (*str*) – Name of the dimension.
- **size** (*int*) – Size of the dimension.
- **isunlimited** (*bool*) – Flag to indicate if the dimension is unlimited or not.

add_gattr(*attr*, *value*)
Add global attribute.

Parameters

- **attr** (*str*) – Name of the new global attribute.
- **value** (*str*) – Value of the new global attribute.

add_var(*varname*, *var*)
Add variable.

Parameters

- **varname** (*str*) – Name of the new variable.
- **var** (*OrderedDict*) – Should be OrderedDict, prepared with *create_variable* function.

change_attr(*var*, *attrname*, *newvalue*)
Change the value of the attribute in specified variable.

The name of the attribute stays the same.

Parameters

- **var** (*str*) – Variable, that should have an attribute with *attrname*.
- **attrname** (*str*) – Name of the attribute.
- **newvalue** (*str*) – New value of the attribute.

change_data(*var*, *data*)
Change data values in the existing variable.

Should be exactly the same shape as original data. Data should be numpy array, or array-like object.

Parameters

- **var** (*str*) – Name of the variable.
- **data** (*array-like*) – Array with new data values of the variable. The size should be the same as for the original data.

change_dtype(*var*, *dtype*)
Change data type values in the existing variable.

Should be exactly the same shape as original data. Data should be numpy array, or array-like object.

Parameters

- **var** (*str*) – Name of the variable.
- **dtype** (*numpy dtype, e.g. npumpy.dtype('float16')*) – Array with new data values of the variable. The size should be the same as for the original data.

change_gattr(*attrname*, *newvalue*)
Change the value of existing global attribute.

The name of the global attribute stays the same.

Parameters

- **attrname** (*str*) – Name of the global attribute.

- **newvalue** (*str*) – New value of the global attribute.

del_attr(*var*, *attr*)
Delete attribute from the variable.

Parameters

- **var** (*str*) – Name of the variable.
- **attr** (*str*) – Name of the attribute to delete.

del_var(*var*)
Delete variable.

Parameters **var** (*str*) – Name of the variable.

rename_attr(*var*, *oldname*, *newname*)
Rename existing attribute of the variable.

The value of the attribute stays the same.

Parameters

- **var** (*str*) – Variable, that should have an attribute with *oldname*.
- **oldname** (*str*) – Old name of the attribute.
- **newname** (*str*) – New name of the attribute.

rename_dim(*oldname*, *newname*, *renameall=True*)
Rename existing dimension.

Parameters

- **oldname** (*str*) – Name of existing dimension.
- **newname** (*str*) – New name for the dimension.
- **renameall** (*bool*) – If renameall is True, rename corresponding dimensions in the variables as well.

rename_dim_invar(*var*, *oldname*, *newname*)
Rename dimension in the variable.

Parameters

- **var** (*str*) – Variable, that should have dimension with *oldname*.
- **oldname** (*str*) – Old name of the dimension.
- **newname** (*str*) – New name of the dimension.

rename_gattr(*oldname*, *newname*)
Rename existing global attribute.

The value of the attribute stays the same.

Parameters

- **oldname** (*str*) – Old name of the global attribute.
- **newname** (*str*) – New name of the global attribute.

rename_var(*oldname*, *newname*)
Rename existing variable.

Attributes, dimensions and data stays the same.

Parameters

- **oldname** (*str*) – Old name of the variable.
- **newname** (*str*) – New name of the variable.

reorder_dims(*neworder*)
Reorder dimensions.

Parameters **neworder** (*list*) – List with dimension names, positioned in the desired order.

reorder_vars(*neworder*)
Reorder variables.

Parameters **neworder** (*list*) – List with name of the variables, positioned in the desired order.

save(*fname*)
Save the file to the disk.

Create netCDF file from the ncfile object.

Parameters **fname** (*str*) – File name.

fixnc.**reorder**(*odict*, *neworder*)
Reorder values in the OrderedDict

Reorder Ordered Dictionary according to the list of keys provided in neworder.

Parameters

- **odict** (*OrderedDict*) –
- **neworder** (*list*) – list with keys from the odict, positioned in the new order.

Returns Ordered dictionary with key-value pairs reordered according to the *neworder*.

Return type OrderedDict

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index