Data Conversion & Cleaning

Timothy Brown

June 24th 2015

Overview

Common Data Formats

Binary

NetCDF

Metadata

Common Data Formats

The problem is there is **no** universally common data format.

Data coming off instruments is often

- text file (comma separated value)
- proprietary binary blob

Very rarely is it in a *nice* open standard binary format. Such as

- NetCDF
- HDF5

CSV

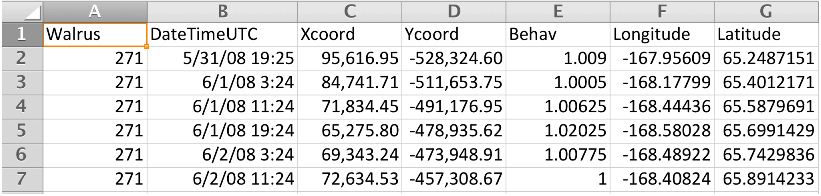
Comma separated value is a plain text file with the data written as ASCII (or Unicode). For example the first few rows and columns of the Walrus data is:

```
Walrus,DateTimeUTC,Xcoord,Ycoord,Behav,
271,5/31/2008 19:25,"95,616.95","-528,324.60",1.009,
271,6/1/2008 3:24,"84,741.71","-511,653.75",1.0005,
271,6/1/2008 11:24,"71,834.45","-491,176.95",1.00625,
```

There is no standard in CSV formats.

- No comments or metadata.
- Non standard date format (ISO 8601).
- Position data is quoted and is grouped with commas.

It is possible to read in Excel. It is able to parse most columns.



	A	B	C	D	E	F	G
1	Walrus	DateTimeUTC	Xcoord	Ycoord	Behav	Longitude	Latitude
2	271	5/31/08 19:25	95,616.95	-528,324.60	1.009	-167.95609	65.2487151
3	271	6/1/08 3:24	84,741.71	-511,653.75	1.0005	-168.17799	65.4012171
4	271	6/1/08 11:24	71,834.45	-491,176.95	1.00625	-168.44436	65.5879691
5	271	6/1/08 19:24	65,275.80	-478,935.62	1.02025	-168.58028	65.6991429
6	271	6/2/08 3:24	69,343.24	-473,948.91	1.00775	-168.48922	65.7429836
7	271	6/2/08 11:24	72,634.53	-457,308.67	1	-168.40824	65.8914233

CSV in Python

There is more than one CSV module

- CSV

```
import csv
with open('Walruses.csv', 'rb') as f:
    reader = csv.reader(f)
    for row in reader:
        print row
```

- Numpy

```
import numpy as np
data = np.genfromtxt('Walruses.csv', delimiter=',')
```

- Pandas

```
import pandas as pd
data = data = pd.read_csv('Walruses.csv',
                          parse_dates=True,
                          thousands=',')
```

Binary Files

Nearly impossible to decode.

- Endianess
- Data format, precision

There are a few tools

- od, `hexdump`
- A lot of spare time
- Pressure the vendor/author

```
0000200 <E0>   {   <CC>   <BF>   O   <EF>   <E3>   -   <CD>   <C1>   <CB>
         31712   49100   61263   11747   49613   49099   55930   30216
0000220 _   <CF>   <CC>   <BF>   <E1>   a   <B2>   A   M   <DF>   <CE>
            53087   49100   25057   16818   57165   49102   34662   16219
```

NetCDF

Network Common Data Form

- Provides a platform-independent file format conducive to sharing data.
- A number of software tools and programming languages can read/write it.

NetCDF is a set of interfaces for array-oriented data access and a freely distributed collection of data access libraries.

- C
- Fortran
- R
- Python

NetCDF Formats

- Started in 1989.
- Formats
  - Version 3 Classic
  - Version 3 64-Bit
  - Version 4 Classic (built on HDF5)
  - Version 4 64-Bit (built on HDF5)

NetCDF Data Model

Dimensions Describe the axes of the data arrays. A dimension has a name and a length. An unlimited dimension has a length that can be expanded at any time, as more data are written to it. NetCDF classic files can contain at most one unlimited dimension.

Variables N-dimensional arrays of data. Variables in NetCDF files can be one of six types (char, byte, short, int, float, double).

Attributes Annotate variables or files with small notes or supplementary metadata. Attributes are always scalar values or 1D arrays, which can be associated with either a variable or the file as a whole. Although there is no enforced limit, the user is expected to keep attributes small.

NetCDF Example

```
laptop$ ncdump foo.nc
netcdf foo {
dimensions:
        time = UNLIMITED ; // (10 currently)
        latitude = 10 ;
        longitude = 10 ;
variables:
        double time(time) ;
        double latitude(latitude) ;
        double longitude(longitude) ;
data:
 time =
  {1403654520, 1403654640, 1403654760, 1403654880,
   1403655000, 1403655120, 1403655240, 1403655360,
   1403655480, 1403655600}
```

NetCDF in Python

- Anaconda by default does not contain NetCDF.
  - Search for a package

    ```
    laptop ~$ conda search NetCDF
    Fetching package metadata: ....
    libnetcdf           4.3.2                0  defaults
                     *  4.3.2                1  defaults
                        4.3.3.1              0  defaults
    netcdf4             1.0.2       np17py27_0  defaults
    ```

  - Install NetCDF

    ```
    laptop ~$ conda install netCDF4
    ```

  - There is also `update`, `remove`

- Unidata's github page contains a very good introduction.
- There are really only 4 commands:
  - Create a file

    ```python
    import netCDF4 as nc
    ncf = nc.Dataset('data.nc', mode='w')
    ```

  - Create a dimension

    ```python
    ncf.createDimension('time', None)
    ```

  - Create a variable

    ```python
    time = ncf.createVariable('time', 'f8', ('time',))
    ```

  - Close the file

    ```python
    ncf.close()
    ```

Lets take a look at the file:

- Version of NetCDF

  ```
  laptop ~$ ncdump -k data.nc
  netCDF-4
  ```

- Dump the whole file

  ```
  laptop ~$ ncdump data.nc
  ncdump data.nc
  netcdf data {
  dimensions:
          time = UNLIMITED ; // (0 currently)
  variables:
          double time(time) ;
  data:
  }
  ```

- Adding data to the file
  - Open the file for reading and writing

    ```
    ncf = nc.Dataset('data.nc', mode='r+')
    ```

  - Add the data

    ```
    time = ncf.variables['time']
    time[:] = xrange(0,11)
    ```

  - Close the file

    ```
    ncf.close()
    ```

- Adding an attribute
    - Global attribute

      ```
      ncf.institution  = 'USGS'
      ```

    - Variable attribute

      ```
      time.units = 'seconds since 1970-01-01 00:00:00 UTC'
      ```

Metadata

Metadata is “data about data”.

- Structural
  - Dimensions
  - Format
- Descriptive
  - What the data is
  - Who collected it
  - Algorithm used

Conventions

The purpose of conventions is to require conforming datasets to contain sufficient metadata that they are self-describing in the sense that each variable in the file has an associated description of what it represents, including physical units if appropriate, and that each value can be located in space and time.

- Climate and Forecast (CF) Conventions
- Attribute Convention for Dataset Discovery (ACDD) Conventions
- CGNS

CF Conventions

- Origin of the data

title	What's in the file
institution	Where it was produced
source	How it was produced e.g. model version, instrument type
history	Audit trail of processing operations
references	Pointers to publications or web documentation
comment	Miscellaneous

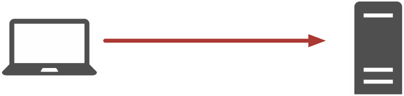
- Description of the data

units	The units of the data
standard_name	The name identifying quantity
missing_value	A value to use if data is missing e.g. -99999
valid_min	The valid minumum value
valid_max	The valid maximum value

Exercise On Yeti

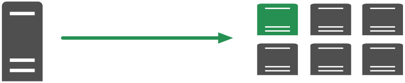
1. Log in to Yeti.

```
laptop ~$ ssh yeti.cr.usgs.gov
```



2. Start a compute job.

```
yeti-login01 ~$ sinteractive -A training \
                    -p prod             \
                    -t 01:00:00 -n 1    \
                    --reservation=training_prod_2
```



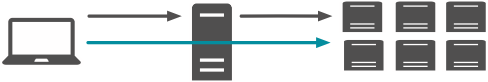
3. Start the ipython notebook server.

```
compute80 ~$ module load python/anaconda
compute80 ~$ ipython notebook        \
                     --no-browser     \
                     --ip=$(hostname) \
                     --port=8888
```

The notebook will bind to the local host on port 8888 by default.

4. Create a tunnel.

```
laptop ~$ ssh -f -N -L 8888:compute80:8888 \
                yeti.cr.usgs.gov
```



The tunnel takes the form of

```
-L local_port:remote_host:remote_port
```

So we are forwarding the default Notebook port (8888) on our laptop to the compute node port 8888, through the login node.

5. Open a browser window pointing to `localhost:8888`.

Questions?

Survey

License