National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

Centre for Environmental Data Archival
SCIENCE AND TECHNOLOGY FACILITIES COUNCIL
NATURAL ENVIRONMENT RESEARCH COUNCIL

# File formats and metadata conventions

Thanks to all contributors:

Alison Pamment, Sam Pepler, Ag Stephens, Stephen Pascoe, Kevin Marsh, Anabelle Guillory, Graham Parton, Esther Conway, Eduardo Damasio Da Costa, Wendy Garland, Alan Iwi, Matt Pritchard and Sarah Callaghan.

# Overview

- Why use standard formats?
- netCDF file format
- CF conventions
- Text files
- NASA Ames
- BADC CSV

# Why use standard formats?

- Well documented
- Standard tools available to read them
- You/your program only has to learn to read one file then you can read them all
- Ability to read/write data not dependent on specialized application
- Easier to preserve long-term and convert to a more modern formats (data curation as well as preservation)

# NetCDF

# What is NetCDF ?

NetCDF stands for “Network Common Data Form”

1 ) It is a software toolbox for the creation, access, and sharing of array oriented scientific data

- It is freely distributed and can be used with C, Fortran, C++, Java, Matlab, Octave, IDL, Python, and other languages.

2 ) It is a file format (.nc) that can be manipulated by the software toolbox

- Heavily used by atmosphere and ocean scientists, climate modellers, software tool developers and data providers. It is easily convertible from and into ASCII (text) format.

# Why use NetCDF ?

The netCDF file format has been very widely adopted by the environmental science community:

- **Self-Describing.** A netCDF file includes information about the data it contains.
- **Portable.** A netCDF file can be accessed by computers with different ways of storing integers, characters, and floating-point numbers.

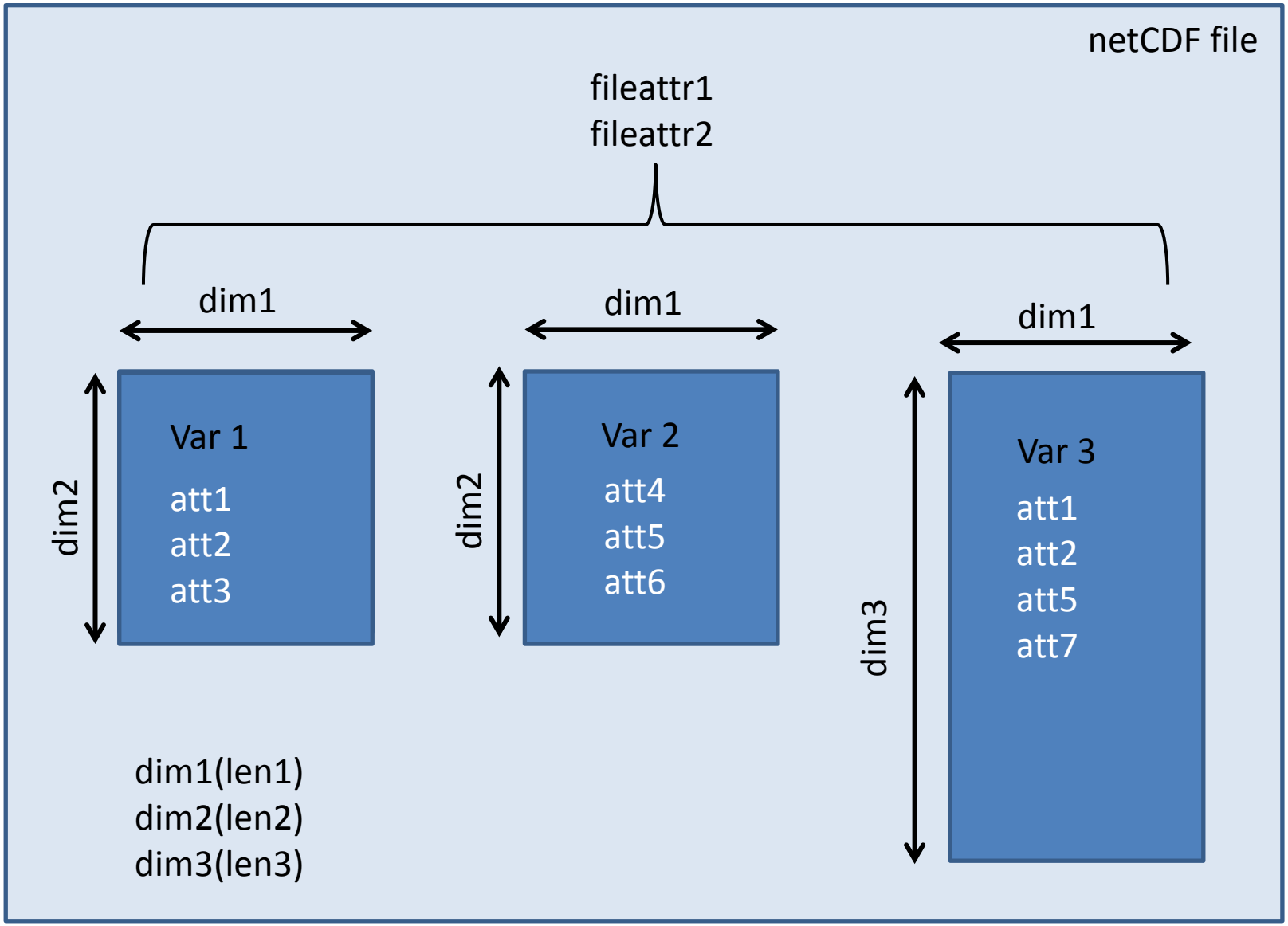
# Why use NetCDF ?

- NetCDF is maintained by UNIDATA (Boulder, USA: UCAR/Unidata Program Center)

- Users' guides for C, Fortran, Java, C++ and other languages interfaces to netCDF data; tutorials for new users, workshop material and program examples.

http://www.unidata.ucar.edu/software/netcdf/

# netCDF3 (“classic”) data model



# How to read NetCDF files

- ncdump is a command line utility that can produce human readable CDL text from a binary netCDF3 or netCDF4 input
- Display the contents of the netCDF dataset mydata.nc on the standard output:

  ncdump mydata.nc
- Display only the metadata in mydata.nc:

  ncdump –h mydata.nc
- Display only the data in mydata.nc:

  ncdump –c mydata.nc

# Using CDL to describe NetCDF contents

```
netcdf example_1 {  // example of CDL notation for a netCDF dataset

dimensions:         // dimension names and lengths are declared first
        lat = 5, lon = 10, level = 4, time = unlimited;

variables:          // variable types, names, shapes, attributes
        float   temp(time,level,lat,lon);
                    temp:long_name     = "temperature";
                    temp:units         = "celsius";
        float   rh(time,lat,lon);
                    rh:long_name = "relative humidity";
                    rh:valid_range = 0.0, 1.0;      // min and max
        int     lat(lat), lon(lon), level(level);
                    lat:units       = "degrees_north";
                    lon:units       = "degrees_east";
                    level:units     = "millibars";
        short   time(time);
                    time:units      = "hours since 1996-1-1";
        // global attributes
                    :source = "Fictional Model Output";
```

# Using CDL to describe NetCDF contents

```
data:                    // optional data assignments
        level   = 1000, 850, 700, 500;
        lat     = 20, 30, 40, 50, 60;
        lon     = -160,-140,-118,-96,-84,-52,-45,-35,-25,-15;
        time    = 12;
        rh      =.5,.2,.4,.2,.3,.2,.4,.5,.6,.7,
                 .1,.3,.1,.1,.1,.1,.5,.7,.8,.8,
                 .1,.2,.2,.2,.2,.5,.7,.8,.9,.9,
                 .1,.2,.3,.3,.3,.3,.7,.8,.9,.9,
                  0,.1,.2,.4,.4,.4,.4,.7,.9,.9;
}
```

# How to write NetCDF files

- ncgen is a command line utility that can produce binary netCDF-3 or netCDF-4 files from a CDL input

- From the CDL file example_1.cdl, generate an equivalent binary netCDF file named mydata.nc:

  ncgen -o mydata.nc example_1.cdl

- From the CDL file example_1.cdl, generate a C program that will create an equivalent binary netCDF dataset:

  ncgen –c example_1.cdl > write_netcdf.c

# How to modify pre-existing NetCDF files

- Command line tools exist for more sophisticated interaction with netCDF files:
  - NCO (NetCDF Operators)
  - CDO (Climate Data Operators)
- Between them these tools allow the user to:
  - modify variables, dimensions, attributes;
  - modify record dimension;
  - extracting geographical regions and time periods of interest;
  - append data to an existing file;
  - read from a file, perform computations (e.g. averaging), write the results to another file.

# Advanced NetCDF tools

- Many programming and scripting languages contain libraries for interacting with netCDF files.

- For example:
  - IDL
  - MATLAB
  - PYTHON
  - OCTAVE
  - R

# Metadata Conventions

- Just as we have standard file formats to facilitate data exchange we can also standardize the way we provide metadata

- In the atmospheric and earth system science the **CF (Climate and Forecast)** metadata conventions are an important aid to data sharing

# What do the CF metadata conventions allow us to do?

- Extend the netCDF metadata conventions
- Provide a detailed description of the contents of a file, thus allowing unambiguous interpretation the data
- All the the standard netCDF file tools still work
- Additional standard tools can be used to read and write the metadata and exploit the information it carries

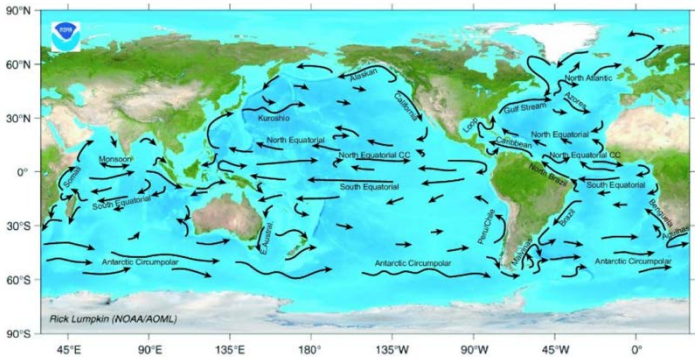
# Goals of CF

(as stated by Jonathan Gregory)

- Locate data in space–time and as a function of other independent variables, to facilitate processing and graphics
- Identify data sufficiently to enable users of data from different sources to decide what is comparable, and to distinguish variables in archives
- Framed as a netCDF standard, but most CF ideas relate to metadata design in general, hence can be contained in other formats such as XML

British Atmospheric Data Centre
NATIONAL CENTRE FOR ATMOSPHERIC SCIENCE
NATURAL ENVIRONMENT RESEARCH COUNCIL


Mesosphere
Stratosphere
Troposphere


90°N
60°N
30°N
0°
30°S
60°S
90°S
45°E
90°E
135°E
180°
135°W
90°W
45°W
0°
Rick Lumpkin (NOAA/AOML)
Kuroshio
North Equatorial
North Equatorial CC
South Equatorial
Monsoon
Somali
South Equatorial
Agulhas
Antarctic Circumpolar
Antarctic Circumpolar
Antarctic Circumpolar
Alaskan
California
North Atlantic
Gulf Stream
Canaries
Labrador
Caribbean
North Equatorial
North Equatorial CC
North Brazil
South Equatorial
Brazil
Benguela
Falkland
PeruChile
Malvinas

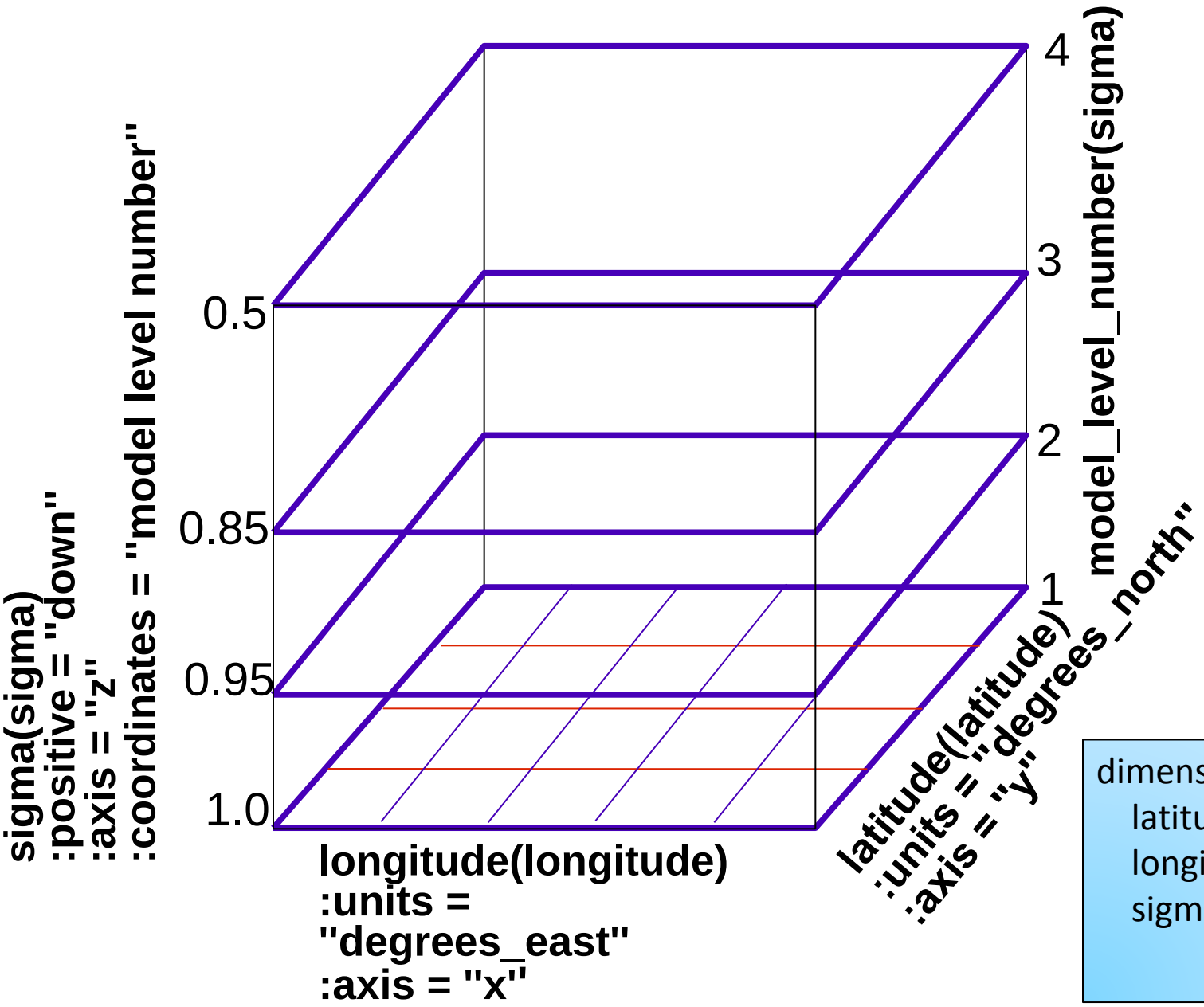
# File attributes

- conventions E.g. conventions = “CF1.0”
- title What's in the file
- institution* Where it was produced
- source* E.g. Name of model, instrument
- history Audit trail of processing
- references* Publications, web pages
- comment* Miscellaneous information

# Variable attributes

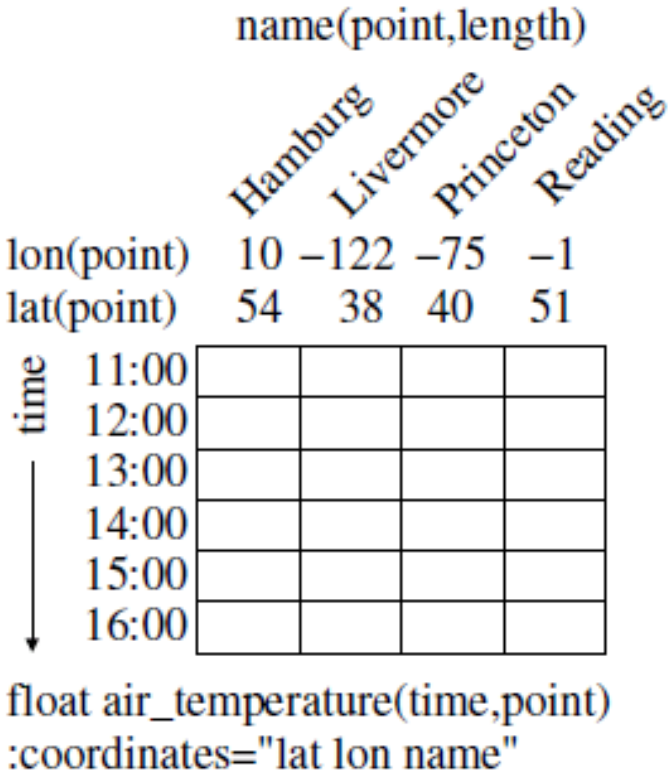
- standard_name from standard name table
- units mandatory unless dimensionless quantity
- long_name not standardised
- cell_methods variation within a cell e.g. max, mean
- cell_measures area or volume of a cell
- valid_max, valid_min, valid_range for numeric variables
- _FillValue, missing_value CF deprecates missing_value in favour of _FillValue
- flag_values, flag_meanings to make “flag” variables self describing

# Coordinate Variables




dimensions:
latitude 360;
longitude 180;
sigma 100;

# Coordinate variables




- Label coordinates, e.g. place names, surface types
- Standardized regions
- Index coordinates
- Scalar (size 1) coords
- Non geospatial coords, e.g. electromagnetic frequency

# Time

Time (year, month, day, hour, second) is recorded as:

time_unit since reference_time

e.g. days since 1990-1-1

seconds since 2013-12-31 00:00

# Calendars

- The calendar is indicated by the calendar attribute of the time coordinate variable
- Default is to use the “standard” calendar, but it is good practice to always specify a value.
- Possible values are:

| Gregorian or standard (the default) | Proleptic_gregorian |
|---|---|
| Noleap or 365_day | All_leap or 366_day |
| 360_day | julian |

# CF Standard Name Table

- Currently 2500+ names in the table – and growing!
- Updated monthly
- Version numbers and date stamp introduced in 2006
- Once names are added they are not removed

British Atmospheric Data Centre
NATIONAL CENTRE FOR ATMOSPHERIC SCIENCE
NATURAL ENVIRONMENT RESEARCH COUNCIL

# CF Standard Name Table

| Standard Name | Canonical Units | AMIP | GRIB |
|---|---|---|---|
| air_density | kg m-3 | | |
| air_potential_temperature | K | theta | 13 |
| air_pressure | Pa | plev | 1 |
| air_pressure_anomaly | Pa | | 26 |
| air_pressure_at_cloud_base | Pa | | |

# Canonical units

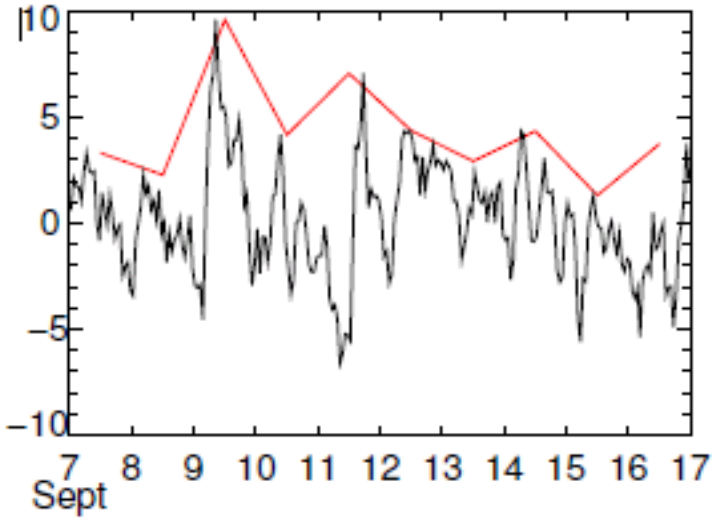
- Canonical_units are agreed at same time as standard name – they go hand in hand, e.g.

  mass_concentration → units = “kg m-3”

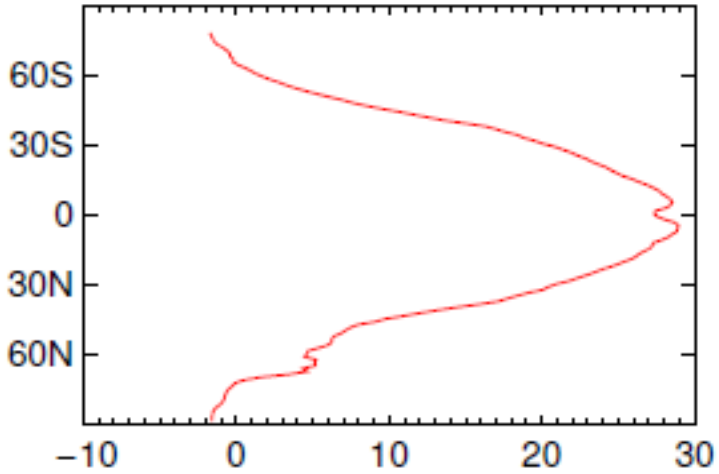
  mole_concentration → units = “mol m-3”
- String valued
- Must be supported by Unidata UDUNITS package which converts between recognized

# Cell methods

- The cell_methods attribute of a data variable indicates how variation within the cells is represented. The method may be different for each axis.
- The order of cell_methods listed from left to right indicates the order in which the operations were applied to the data.



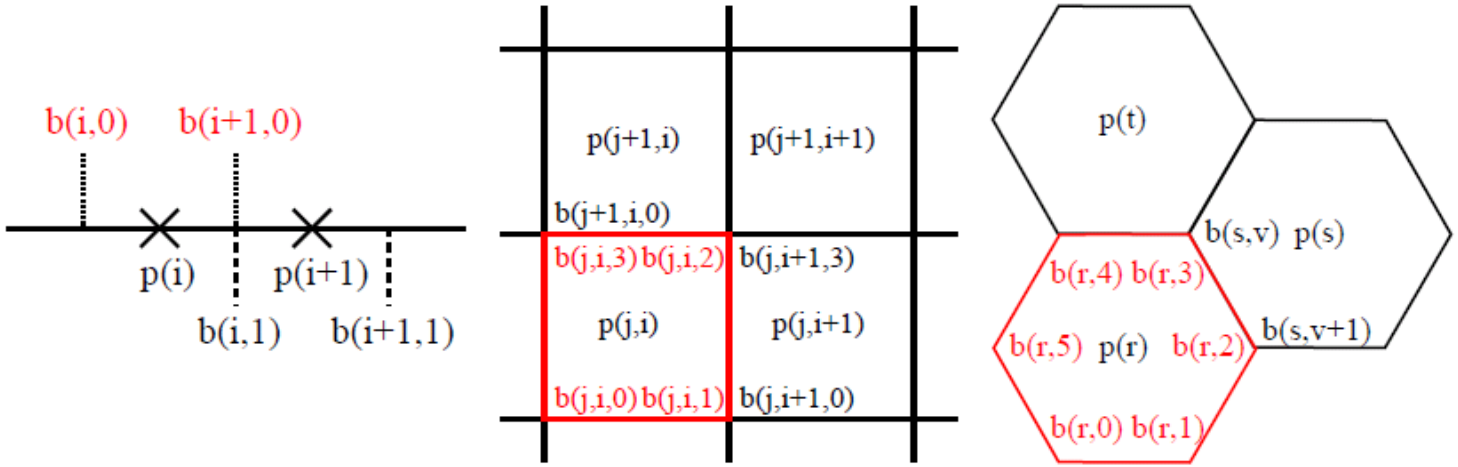

:cell_methods="time: maximum"



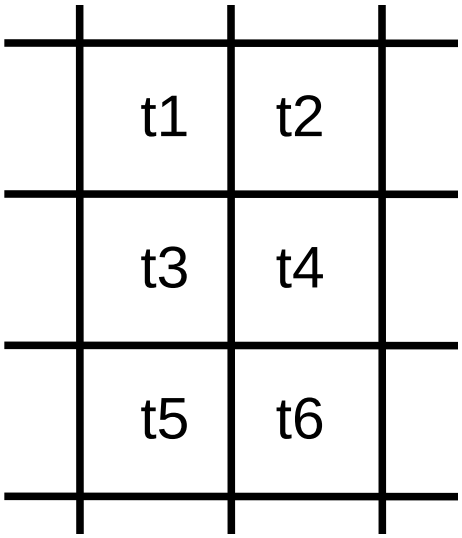

:cell_methods="longitude: mean"

# Cell Bounds

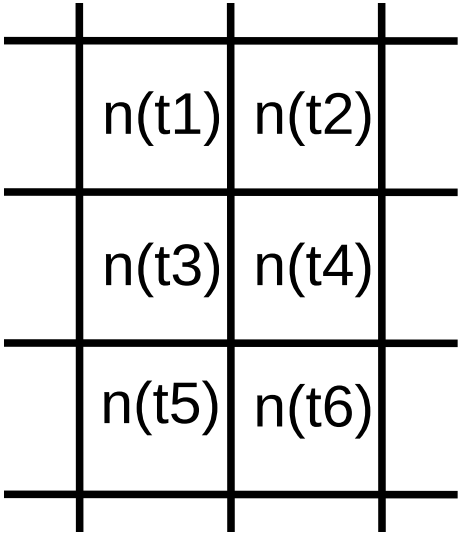
- cell_bounds attribute describes the extent of a cell
- E.g. area of lat-lon grid box, thickness of a vertical layer, length of a time-mean period
- Can be especially important for size-one coordinate variables



# Ancillary variables

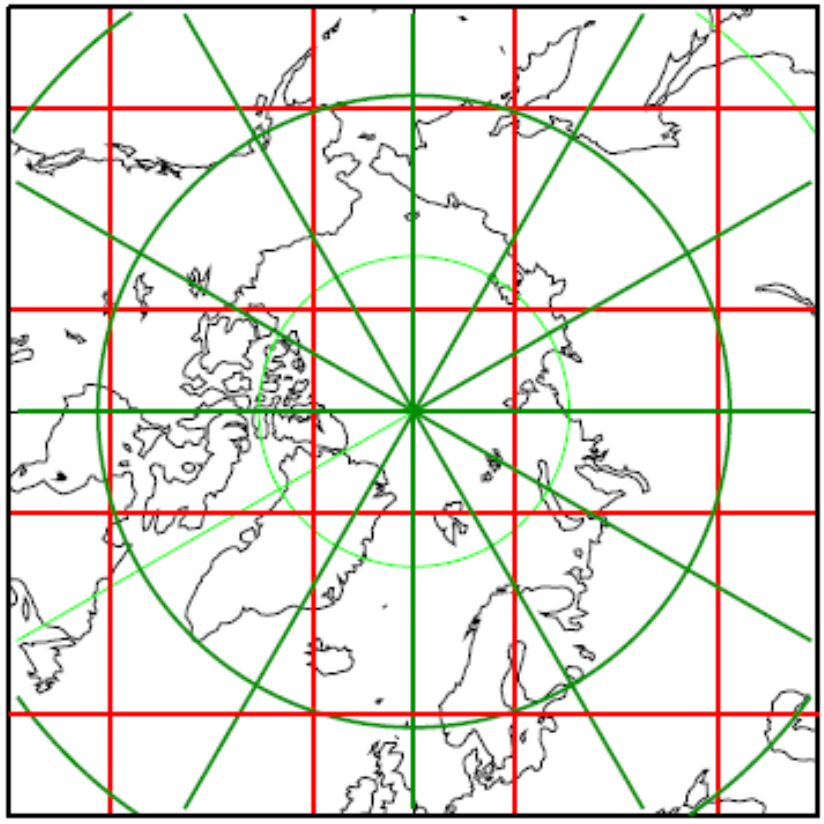



t(lat,lon)
t:standard_name = "air_temperature"
t:ancillary_variables = "numt"
t:units = "K"




numt(lat,lon)
numt:standard_name = "air_temperature number_of_observations"
numt:units = "1"

# Map projections




float p (y,x)
p:standard_name=“air_pressure_at_sea_level”
p:coordinates=“lat lon”
p:grid_mapping=“polar_stereographic”
p:latitude_of_projection_origin=“=90.”
p:false_easting=“ ”
p:false_northing=“ ”
float lat(y,x)

# Recommended text file formats

# What is a text file?

- Computers don't do text, they only do binary
- 01001010011110110101 – does not mean anything to humans
- "Hi there!" – does not mean anything to computers
- One of the simplest bridges between computers and people is simple standard text encoding
- There are 2 main flavours – ASCII and Unicode

ASCII Encoding table

| Dec | Hex | Char | Dec | Hex | Char | Dec | Hex | Char | Dec | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 00 | Null | 32 | 20 | Space | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 01 | Start of heading | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 02 | Start of text | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 03 | End of text | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 04 | End of transmit | 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 5 | 05 | Enquiry | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 06 | Acknowledge | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 07 | Audible bell | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 08 | Backspace | 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 9 | 09 | Horizontal tab | 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 10 | 0A | Line feed | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | 0B | Vertical tab | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | 0C | Form feed | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | 0D | Carriage return | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | 0E | Shift out | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | 0F | Shift in | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | Data link escape | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | Device control 1 | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | Device control 2 | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | Device control 3 | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | Device control 4 | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | Neg. acknowledge | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | Synchronous idle | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | End trans. block | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | Cancel | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | End of medium | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | Substitution | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | Escape | 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 28 | 1C | File separator | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 29 | 1D | Group separator | 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 30 | 1E | Record separator | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | Unit separator | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | □ |

ASCII Encoding table

*Unicode (UTF-8) has a similar concept but extends to many thousands of characters*

| Dec | Hex | Char | Dec | Hex | Char | Dec | Hex | Char | Dec | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 00 | Null | 32 | 20 | Space | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 01 | Start of heading | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 02 | Start of text | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 03 | End of text | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 04 | End of transmit | 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 5 | 05 | Enquiry | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 06 | Acknowledge | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 07 | Audible bell | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 08 | Backspace | 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 9 | 09 | Horizontal tab | 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 10 | 0A | Line feed | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | 0B | Vertical tab | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | 0C | Form feed | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | 0D | Carriage return | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | 0E | Shift out | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | 0F | Shift in | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | Data link escape | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | Device control 1 | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | Device control 2 | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | Device control 3 | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | Device control 4 | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | Neg. acknowledge | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | Synchronous idle | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | End trans. block | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | Cancel | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | End of medium | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | Substitution | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | Escape | 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 28 | 1C | File separator | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 29 | 1D | Group separator | 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 30 | 1E | Record separator | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | Unit separator | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | □ |

Unprintable Characters

Numbers

Capital Letters

Lower case

Printable Characters

# When to use text formats

- Ideal for Simple (smallish) measurement datasets against one distinct, incremental variable, like a time stamp
- Is the data just a table of stuff
- The users are most comfortable with data they can see in a human readable form

# The NASA Ames format

- Grew out of NASA aircraft campaigns and was first formalised at the Ames Research Centre
- They needed to **facilitate the data exchange** between campaign participants and …
- allow **shared use of a minimised amount of software** to analyse and display different datasets.
- The format should be **portable** (readable on any machine by any programming language) and **self-describing** (that is contain metadata);
- It had to be **readable by humans**.
- The **portable** and **readable by humans** requirements implied the adoption of a text format (namely **ASCII**).
- The **self-describing** requirement was met by including in each data file a **header** containing the *metadata*.
- Very well suited to field campaigns involving several teams that need to share their observations, the NASA Ames Format is not well adapted to very voluminous datasets. In this case, although less portable, a binary format is recommended.

# File structure - header and data

- All NASA-Ames files have a header and then the data.
- The header has lines that describes the data in the file – metadata.
- You can tell what each piece of metadata means by the position in the file. For example - The third line is always the Author's institute.
- There are several sub-types for dealing with data with different numbers of dimensions, but it's best to use it for the 1D, or possibly 2D data.

# What's in NASA Ames

**Format specification**

Number of header lines  NASA-Ames Format Index
Author's name
Author's institute
Instrument name
Project name
1 1
Date of observations  Date file produced
Size of intervals in time (use 0.0 if non-uniform)
Name for time variable with units
Number of variables for each time point
Scaling factors for each variable
Missing values for each variable
Name of first variable
Name of second variable
…
Number of lines of specific comments
Comment line 1
Comment line 2 etc
Number of lines of general comments
Comment line 1
Comment line 2
Comment line 3
Comment line 4 etc
Data
Data
Data
…

Who

How

Why

When

What

Extra Details

Data

# A simple NASA Ames example

**Format specification**

Number of header lines  NASA-Ames Format Index
Author's name
Author's institute
Instrument name
Project name
1 1
Date of observations  Date file produced
Size of intervals in time (use 0.0 if non-uniform)
Name for time variable with units
Number of variables for each time point
Scaling factors for each variable
Missing values for each variable
Name of first variable
Name of second variable
…
Number of lines of specific comments
Comment line 1
Comment line 2 etc
Number of lines of general comments
Comment line 1
Comment line 2
Comment line 3
Comment line 4 etc
Data
Data
Data
…

**Example**

27 1001
Whalley, Lisa
School of Chemistry, University of Leeds, UK
FAGE HO2
RHaMBLe Cape Verde 2007
1 1
2007 05 21  2013 05 29
0
Time (decimal days since 2007-01-01 00:00:00 +0.00)
1
1
9999999999
HO2 (molecule cm-3)
0
11
University of Leeds FAGE HO2 data
Campaign: Reactive Halogens in the Marine Boundary Layer (RHaMBLe) Part of the UK SOLAS programme
"Instrument inlet was located on the FAGE container roof, approx height = 3.5 m"  LAT:  16.85 N  LONG: 24.87 W
Units in molecule cm-3
This data is roughly a 7.5 minute average taking all data points in each 7.5 minute period  Time represents mid time of data period
[HO2] 2 sigma standard deviation = 44%
Contact Lisa Whalley prior to use:  L.K.Whalley@leeds.ac.uk;
decimal time  HO2
140.478  4.54E+08
140.499  3.67E+08
140.504  4.20E+08
140.510  4.34E+08
140.515  4.62E+08
140.520  3.59E+08

# More NASA Ames info

https://badc.nerc.ac.uk/help/formats/NASA-Ames/

Go here for…

- Examples
- Code
- Documentation

# NASA Ames gotchas

- Time values must be monotonic and increasing.
- Missing values (same format but larger than any possible data value)
- If exporting from excel watch out for excel formatting of date
- No non-ASCII characters – cut and paste from other places
- You can write valid NASA-Ames that's no help to anyone. Author's name = Sam, Variable name = sam3.

# Tools list

- Nappy - python library
- Excel – use fixed with import
- Fortran – NASA-Ames was originally written with Fortran import in mind
- Other languages, MatLab, IDL fairly easy to read NASA-Ames data.
- Downloading data from the BADC? The data browser that can give basic plot or allow file to be exported into excel/CSV.
- Preparing data to archive? There is a web based file format checker.

# Encoding CF in NASA-Ames

- We recommend the use of CF standard names and units where possible.
- Write information in the comment lines if you need to

# The BADC-CSV format

- The BADC has used NASA-Ames formatted data for many years. But,
  - NASA-Ames can be complex and confusing for users.
  - Users have to strip the header off to import the text file into Excel.
  - A lot of effort to support data producers in the creation of NASA-Ames files.
  - Can't be simply written from spread sheet packages like Excel.
  - The metadata fields offered by NASA-Ames are fixed and inflexible.

# The BADC-CSV format

- Model data stored at the BADC often uses the NetCDF format with CF conventions. This provides a format framework with good flexible metadata. The format can be read by a number of analysis programs including FORTRAN, Matlab and IDL. It is however difficult for a researcher with little technical knowledge to use.
- To solve these problems a new file format was developed to bring the advantages from the NetCDF file format into a simple text file. The approach was to use metadata conventions on top of comma separated values files (CSV) as produced by applications like Excel.

# Details of the BADC-CSV format

```
Conventions, G, BADC-CSV,1
title, G, My data file
creator, G, Prof W E Ather,Reading
contributor, G, Sam Pepler,BADC
creator, G, A. Pdra
long_name, x, time, days since 2007-03-14
long_name, y, air temperature
long_name, z, met station air temperature
creator, z, unknown, Met Office
coordinate_variable, x, x
location_name, G, Rutherford Appleton Lab
Data
x, y, z
0.8,2.4,2.3
1.1,3.4,3.3
2.4,3.5,3.3
3.7,6.7,6.4
4.9,5.7,5.8
end data
```

Who

How

Why

When

What

Extra Details

Data

# Details of the BADC-CSV format

```
Conventions, G, BADC-CSV,1
title, G, My data file
creator, G, Prof W E Ather, Reading
contributor, G, Sam Pepler, BADC
creator, G, A. Pdra
long_name, x, time, days since 2007-03-14
long_name, y, air temperature
long_name, z, met station air temperature
creator, z, unknown, Met Office
coordinate_variable, x, x
location_name, G, Rutherford Appleton Lab
Data
x,y,z
0.8,2.4,2.3
1.1,3.4,3.3
2.4,3.5,3.3
3.7,6.7,6.4
4.9,5.7,5.8
end data
```

G = Global

Column names

Metadata

Data

# More BADC-CSV info

http://badc.nerc.ac.uk/help/formats/badc-csv/

Go here for…

- Examples
- Code
- Documentation

Where do people use text formats?

- Aircraft data
- BADC data from Met Office obs networks
- Clearflo – Air quality time series
- MSTRF – surface obs http://badc.nerc.ac.uk/browse/badc/mst/data/surface-met/2014/02/met-sensors_capel-dewi_20140211.na
- Ozone sonde profiles

Where don't text formats work

- Met Office Model data
- ECMWF data
- MST radar
- Satellite observations

# Markup-based formats

- XML
- Good for complex data structures, but cumbersome to interact with programmatically
- Its semantic-less. There is nothing to stop you writing data in XML, but you need to write the tools too.
- If you do put meaning into the data you need to find/write the schema too.

# Excel format

- Why not keep data in excel format?
- Not very open (though that has changed in recent years)
- Subject to change
- Not always implemented in the same one from application to application
- There is a temptation to encode information into the style – “All red, italics are invalid”. Just hope the font has italics in the future.

# Other text-based formats

- There are many other formats, but please use ones that are
  - metadata-rich
  - Standardised (widely used)