I/O

INPUT AND OUTPUT, PART I

Homework Review

Questions or comments?

The Lecture of Today

Overview of basic input and output (I/O or IO) techniques as it relates to Atmospheric Science programming and how to manage competing techniques to arrive at a practical solution.

- Discussion of file formats and data types.
- Examples of working with common ATMOS data types.
- Deconstruct the thought process with approaching data tasks.

Workflow Overview: as data centric

Framing the overall task and subtasks; asking the right questions to yourself.

- What's the end goal I'm after?
- How is "this" part related to that end goal?
- What data do I need to satisfy the task?
- Where do I get it?
- How do I use it? (reading, converting)
- How do I store it?

File Types & Formats

File format “classes”:

- Structured
- Unstructured (free form)
- Relational and hierarchal
- Flat or mixed
- Custom and 1000’s more

https://en.wikipedia.org/wiki/List_of_file_formats

Data types and attributes:

- Text or ascii
- Hexadecimal
- Binary with or without headers

Data format types:

- Little and big endian
- Signed and unsigned
- Float(32,64), int(8,16,32,64), string, char, Boolean, to name a few.

Files as IO = Fundamental

Did you know most things in Unix communicate as files?

"Everything is a file" describes one of the defining features of Unix, and its derivatives — that a wide range of input/output resources such as documents, directories, hard-drives, modems, keyboards, printers and even some inter-process and network communications are simple streams of bytes exposed through the filesystem name space.

https://en.wikipedia.org/wiki/Everything_is_a_file

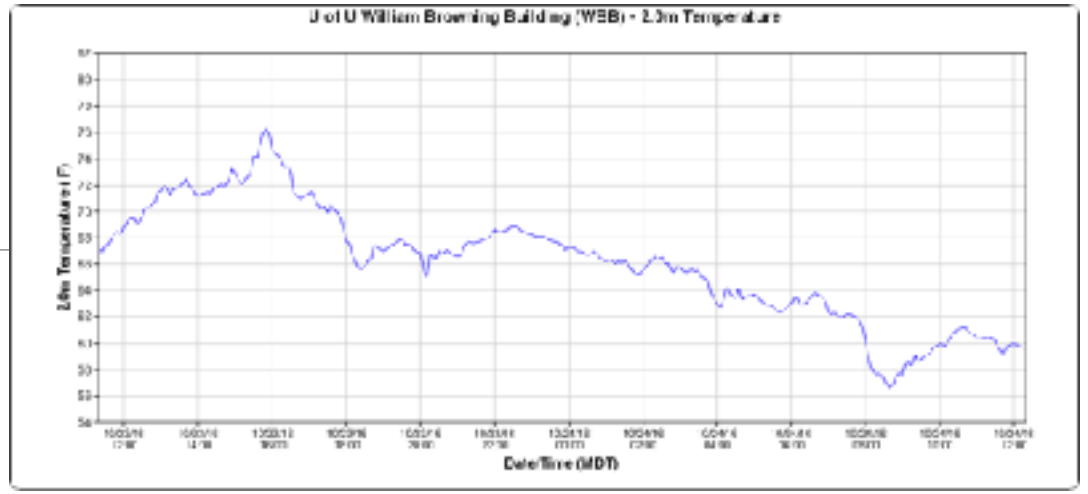
Examples of ATMOS Data Landscape

Some main players are:

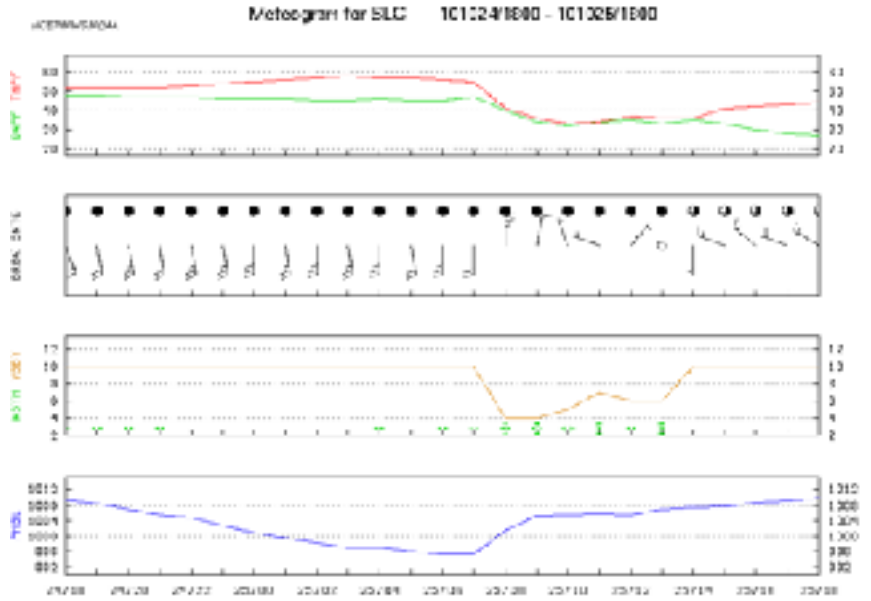
- In situ observations
- Model output
- Remote sensing
- Text products (discussions, warnings)
- Supportive (GIS for context)

In Situ Observations

Many file formats exist for this type of data. Most common are csv, tab delimited, NetCDF, METAR, SHEF, BUFR.



- Data is almost always represented as a time-series in a tabular or graphical form.
- Metograms (collection of many weather variables standardized to same time scale) are also common.



Model Output

Example products include GFS, NAM, RAP, NDFD, RTMA, HRRR, etc. See these for more info:

http://www.weather.gov/forecastmaps

http://www.nco.ncep.noaa.gov/pmb/nwprod/prodstat/

Output is commonly grib, grib2 or NetCDF file format. Examples at NWS operational ftp download site.

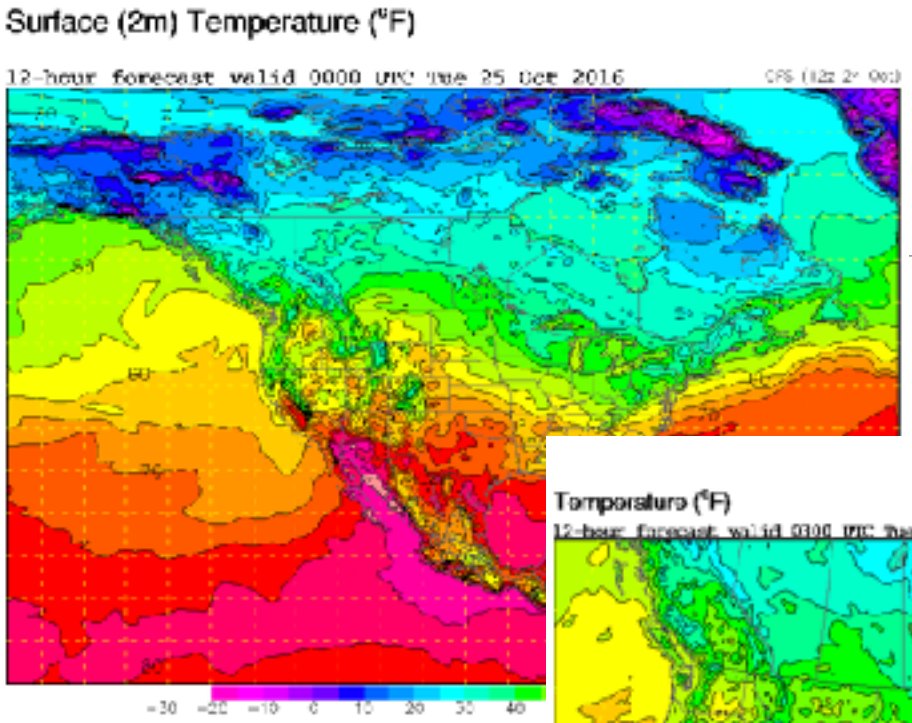
ftp://tgftp.nws.noaa.gov/SL.us008001/ST.opnl

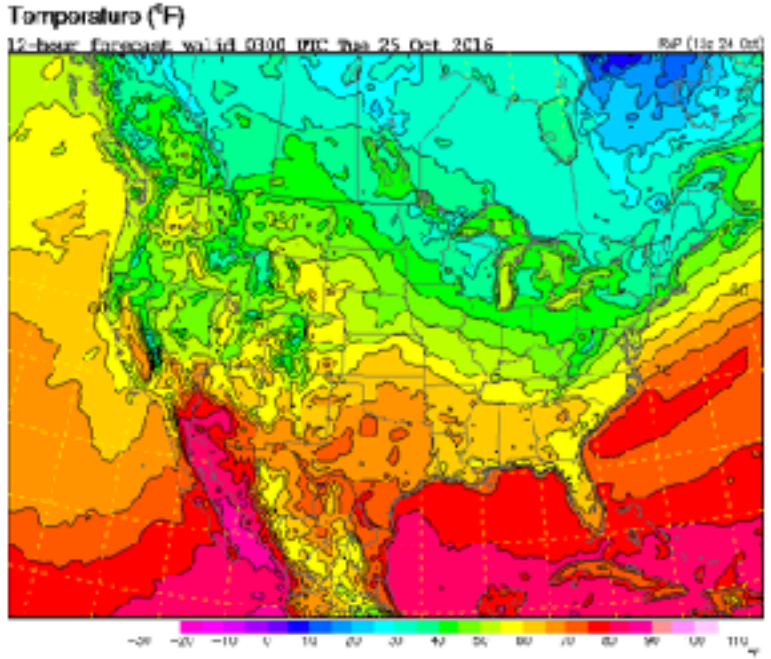
No matter how the data is stored, data for a model output will ultimately take the shape of values for lat/lon pairs. Locations may be explicit or implied.

[[284.6, 286.1, 281.5, …],

[282.7, 283.4, 287.0, …],

]

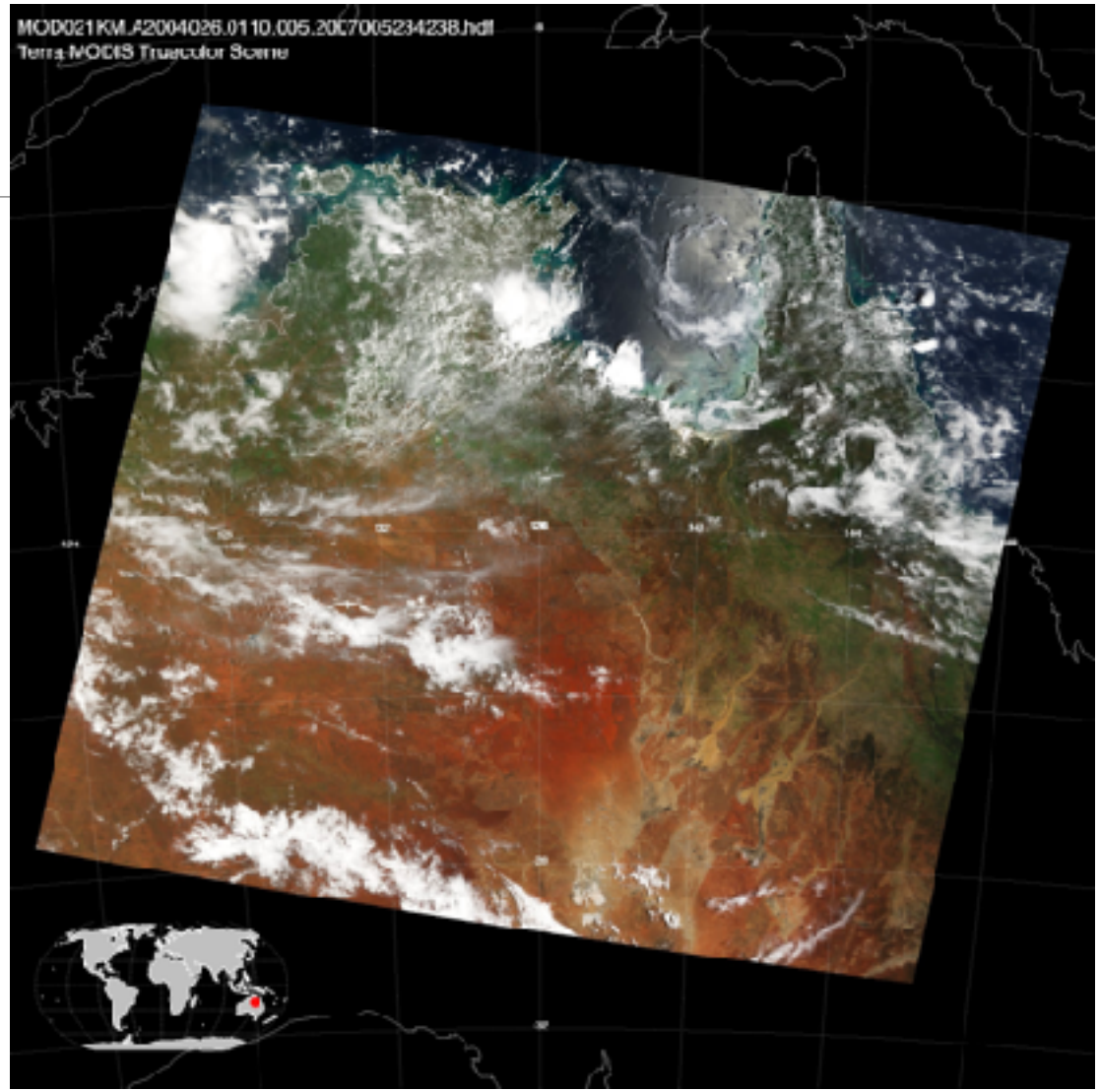




Remotely Sensed

Remote sensing is the science of obtaining information about objects or areas from a distance, typically from aircraft or satellites.

- Data is commonly stored in HDF, NetCDF, GeoTIFF, or raw binary formats.
- Compression and packing techniques are often used due to the size of these data.
- see https://worldview.earthdata.nasa.gov/ for many examples of visualized products.



Text Products

These data are often for forecasts, discussions, automated messages such as warnings, watches, public statements, etc.

We’re not going to spend time with these, other than to note they have predictable formats, and therefore can be parsed and processed effectively.

URGENT - WEATHER MESSAGE
NATIONAL WEATHER SERVICE MEDFORD OR
252 AM PDT MON OCT 24 2016

CAZ085-ORZ030-031-250000-
/O.CON.KMFR.HW.W.0018.161024T1200Z-161025T0600Z/
/O.CON.KMFR.WI.Y.0036.161024T1200Z-161025T0600Z/
MODOC COUNTY-
NORTHERN AND EASTERN KLAMATH COUNTY AND WESTERN LAKE COUNTY-
CENTRAL AND EASTERN LAKE COUNTY-
INCLUDING THE CITIES OF...ALTURAS...BEATTY...BLY...
SPRAGUE RIVER...LAKEVIEW
252 AM PDT MON OCT 24 2016

...HIGH WIND WARNING REMAINS IN EFFECT UNTIL 11 PM PDT THIS
EVENING...
...WIND ADVISORY REMAINS IN EFFECT UNTIL 11 PM PDT THIS EVENING...

* WINDS IN ADVISORY AREA...SOUTH 20 TO 30 MPH WITH GUSTS UP TO
  50 MPH.

* WINDS IN WARNING AREA...SOUTH TO SOUTHWEST 40 TO 50 MPH WITH
  GUSTS UP TO 70 MPH POSSIBLE.

* TIMING...WINDS WILL INCREASE EARLY THIS MORNING...PEAK LATE THIS
  MORNING INTO THIS AFTERNOON...THEN DIMINISH TONIGHT.

* LOCATIONS IN ADVISORY INCLUDE...KLAMATH FALLS...LAKEVIEW...
  ALTURAS.

* LOCATIONS IN WARNING INCLUDE...HIGHWAY 31 NEAR SUMMER LAKE...WARNER
  MOUNTAINS.

* IMPACTS...TRAVEL COULD BECOME DANGEROUS ALONG HIGHWAY 31 NEAR
  SUMMER LAKE... ESPECIALLY FOR HIGH PROFILE VEHICLES. TRAVEL
  ALONG HIGHWAYS 140... 395... AND 31 WILL BECOME DIFFICULT FOR
  HIGH PROFILE VEHICLES. UNSECURED OBJECTS MAY BE LOST OR
  DAMAGED.

* VIEW THE HAZARD AREA IN DETAIL AT HTTP://WEATHER.GOV/MEDFORD/HAZARD

PRECAUTIONARY/PREPAREDNESS ACTIONS...

A WIND ADVISORY MEANS THAT WINDS OF 35 MPH ARE EXPECTED WITH
HIGHER GUSTS POSSIBLE. WINDS THIS STRONG CAN MAKE DRIVING
DIFFICULT...ESPECIALLY FOR HIGH PROFILE VEHICLES. USE EXTRA
CAUTION.

A HIGH WIND WARNING MEANS A HAZARDOUS HIGH WIND EVENT IS EXPECTED
OR OCCURRING. SUSTAINED WIND SPEEDS OF AT LEAST 40 MPH OR GUSTS
OF 58 MPH OR MORE CAN LEAD TO PROPERTY DAMAGE. TRAVEL WILL BE
IMPACTED... ESPECIALLY FOR HIGH PROFILE VEHICLES.

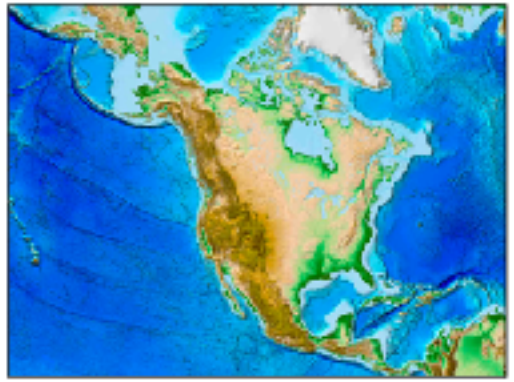
&&

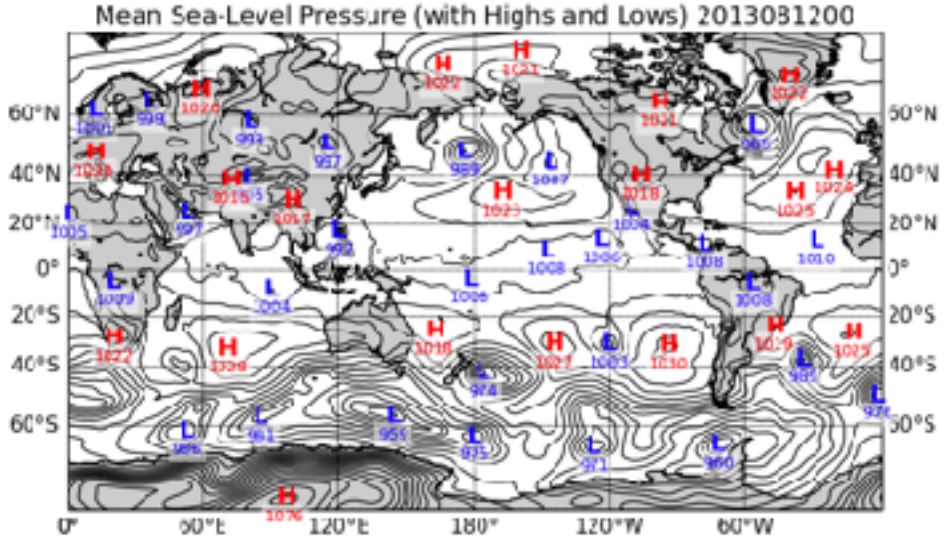
$$

Supportive Data

Some data is supportive for our needs.

- Digital elevations (DEMs) or contours, shorelines, water masks, water bodies, coastlines, zones, smoke plumes, fire perimeters, points.
- Commonly seen as ESRI ShapeFile, raw binary, GeoTIFF, geojson, topojson.
- Much of the contextual data needed for visualizations is contained within most programming environments, such as IDL, Matlab, Python (Matplotlib), R.





Why So Many Data Formats?

There are usually very good reasons why data have certain formats.

- Self describing data is very useful for archival storage and distribution
- Self contained metadata and provenance
- Spatial representation, and extents. (ShapeFiles)
- Structure for multiple and arbitrary dimensions
- Repetitive data patterns compress really well (grib2, HDF5)
- Offset and scale factors work great for some data (NetCDF)

Any other reasons?

Some Gotchas

Always double check you are working with the data you expected

- Be careful with using filenames as metadata.
- Be skeptical that the data you expect is actually contained within any given file.
  - Example: NDFD files are distributed from NOAA every hour as the same name in the ftp/http download directories. If they fail to update, you could be using a forecast for the wrong hour, or even day!

Data Sources

Some are easy to get, while others are purposefully difficult. Why is that?! Lots of reasons and sometimes politics, data sensitivity, costs of distribution, etc.

Research considerations when using data

- Get data closest to the source as possible. There are a lot of unofficial sources of data
- Get the metadata whenever possible, and ask about the data. Give a brief use case to the owner of the data. Why? Because all data have “warts” and caveats.
- Ask how to cite the data.

Measured / Observed & Derived Data

Measured and observed data are from techniques to capture the state of something physical and real at a given time and location. Examples: air temperature, ozone concentrations, wave height, battery voltage, 500mb dew point.

A derived data element is a data element derived from other data elements using a mathematical, logical, or other type of transformation, e.g. arithmetic formula, composition, aggregation*. (An interesting read: https://blog.ldodds.com/2015/09/05/what-is-derived-data/) * The OECD Glossary of Statistical Terms

** Wind speed at a surface station is often an average over a period of time (such as every 5 minutes). Is this observed or derived data?

A Note on Big Data

Big data is a term for data sets that are so large or complex that traditional data processing applications are inadequate to deal with them. Challenges include analysis, capture, data curation, search, sharing, storage, transfer, visualization, querying, updating and information privacy.

- Wikipedia, https://en.wikipedia.org/wiki/Big_data

A Note on Big Data

Be skeptical of having big data problems before considering the en vogue solutions for working through big data. Solutions for processing and analyzing big data are very expensive. Spending $1,000 an hour is not that uncommon when spinning up compute nodes in clouds.

Don’t turn your data into an “Unnecessarily Big Data Problem.” Consider a divide and conquer approach and break out steps into reasonable tasks whenever possible.

Potential Big Data Case Examined

A research question necessitates your needing to find the difference in observed 10m temperatures from two stations near each other over a period of time for exact matching moments. The stations report at high temporal frequency, every 2 and 3 seconds respectively.

A data request resulted in you getting two 100+ Gb files that contain text rows with the format of Date-time in UTC, temperature, QC Boolean. The files contain over 10 million rows each. Both files are already sorted by date-time. Some times are missing in both sets of data.

The data is too large to fit into memory all at once on any computer available to you.

Do we have a big data problem here?

Hint: Did someone say it was already sorted?

Example 1: Be “Lazy”

1. Realize that sorting is your best friend.
2. Discover that opening both files lazily at the same time is valuable.
3. Realize that the solution is as simple as looping over one of the files once and reading rows from the other until times are greater than the current row from the first file.
4. When times do match, write the rows out to a file for both stations with the temperature difference as last csv element.
5. Close the files and reflect back on your genius.

Example 1: but not too Lazy!

1. Don’t open the second file more than once (such as within the first loop)
2. Don’t rewind or seek back to 0 position on the second file every time you find the date to be greater than the row from file one.
3. Don’t store your results in memory and then write it out to disk at the end of the reading; you’ll consume all memory before that happens anyway.
4. Don’t run the program without making sure all previous instances are finished/killed and all files are closed.
5. Try not to place all files on the same network disk if that location is known to “feel” slow. This only exacerbates the disk IO limits.
6. Try writing to a local place on disk.
7. Don’t zip/gzip your file every time you open and close it from your program until you have it running 100%.

A Note

There are not right ways and wrong ways of writing programs. There are just **ways**.

If we need to label them, let's say there are practical ways and not-so-practical ways, which yield different results in speed, memory optimization, reading, writing, and generally all things "performance" related.

Ergo, you decide what the right balance is for the task, and try to recognize a reasonable balance during the approach.

A Tip

Find the community and official docs for the file formats you're working with. There is a very good chance you'll find a working example that opens and reads the file in question.

It's OK to copy and paste code examples from public sources to get you going. Credit when necessary (e.g., functions are taken wholesale) and don't worry about crediting code that has been modified to fit your needs. That is NOT an infringement of licensing.

Iteration, Drafts & Prototypes

You will write code …

to learn how to write code …

to throw away that code …

to learn how to write code …

to write code that works and is useful

Iteration, Drafts & Prototypes

… And that's perfectly OK to do.

Tips:

- At the very least you get rid of Analysis Paralysis.
- Never think you're going to get it right the first time or draft.
- Don't spend too much time worrying about any of the details until the critical path is proven and understood.
- Focus on function over form to start with. e.g., a pretty looking plot with bad data is useless.

CSV Processing

The common tasks (most apply to any file format):

- Reading a file
  - Use a module or built-in feature vs code your own
  - Line by line vs all at once
  - Reading until a marker is found (1,2,3EOF)
  - Stripping leading and trailing characters
  - Splitting on row delimiters
  - Splitting columns on read
  - Be careful of commas in double or single quotes

CSV Processing

The common tasks (most apply to any file format):

- Arranging
  - Arrays vs other data object types
  - Load rows-wise vs column-wise
- Casting and converting
  - Cast to proper data type from string
  - Apply units
  - Date-time standardizations *
    - Use built-in date-time formats. DON'T invent your own date or time code.

CSV Processing

The common tasks (most apply to any file format):

- Sorting and comparing
  - Sort on useful keys
  - Get index positions of sorted array to use on ancillary arrays
    - e.g., you have two arrays as date and temperature. Make sure to apply the sorting index positions to temperature if you sort on date.
  - Double and triple check your data types if you're sorting on them.
    - [1,2,3,10,20,30,100,200,300] will sort differently as string vs integer.
  - Take advantage of built-in comparers and sorting features.

CSV Processing

The common tasks (most apply to any file format):

- Bookkeeping
    - Build supportive arrays and counters

Working with CSV

When working with multiple and/or disparate CSV files, consider the following:

- Standardize the main data “key” elements. These include time, data types, precision, locations, and more.
- To find data assets closest to each other, such as the closest grid cell in a model output file to a specific surface station, compare distance via latitude and longitude.
- To find data assets closest to each other in time with thresholds, you would compare the observed values closest to the top of each hour, within 15 minutes in either direction.

NetCDF

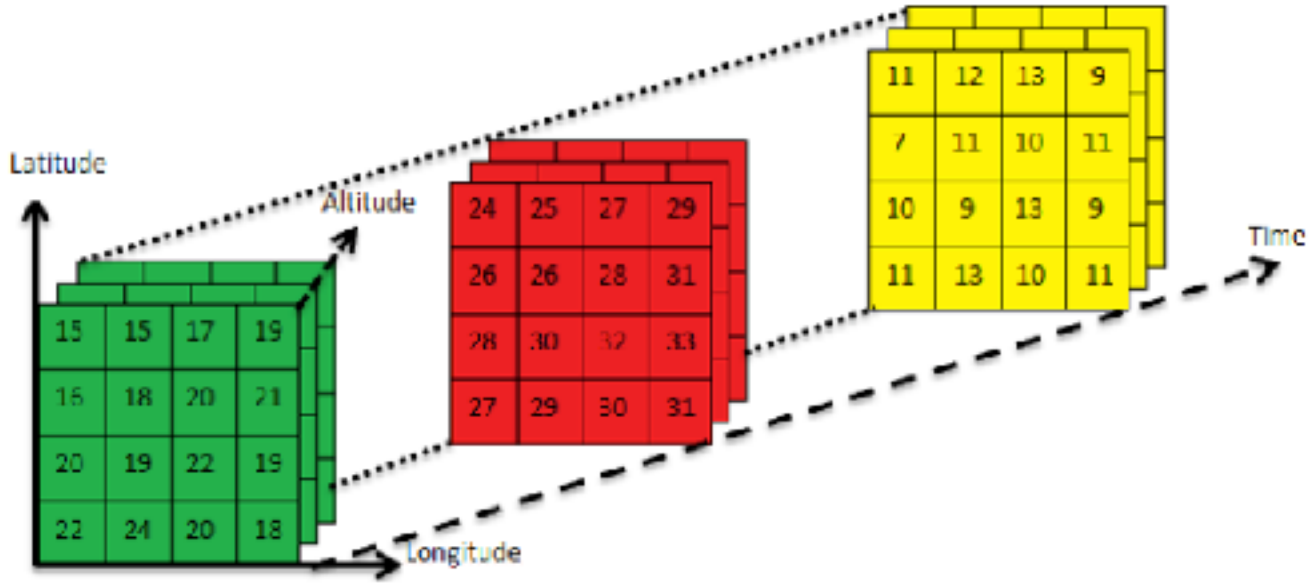
- NetCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data.
- The purpose of the Network Common Data Form (netCDF) interface is to allow you to **create, access, and share** array-oriented data in a form that is **self-describing and portable**. "Self-describing" means that a dataset includes information defining the data it contains. "Portable" means that the data in a dataset is represented in a form that can be accessed by computers with different ways of storing integers, characters, and floating-point numbers.
- The Network Common Data Form, or netCDF, is an interface to a library of data access functions for storing and retrieving data in the form of arrays. An array is an n-dimensional (where n is 0, 1, 2, ...) rectangular structure containing items which all have the same data type (e.g., 8-bit character, 32-bit integer). A scalar (simple single value) is a 0-dimensional array.

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

NetCDF 4 dimension example

A grid of relative humidity for multiple times each at multiple levels.

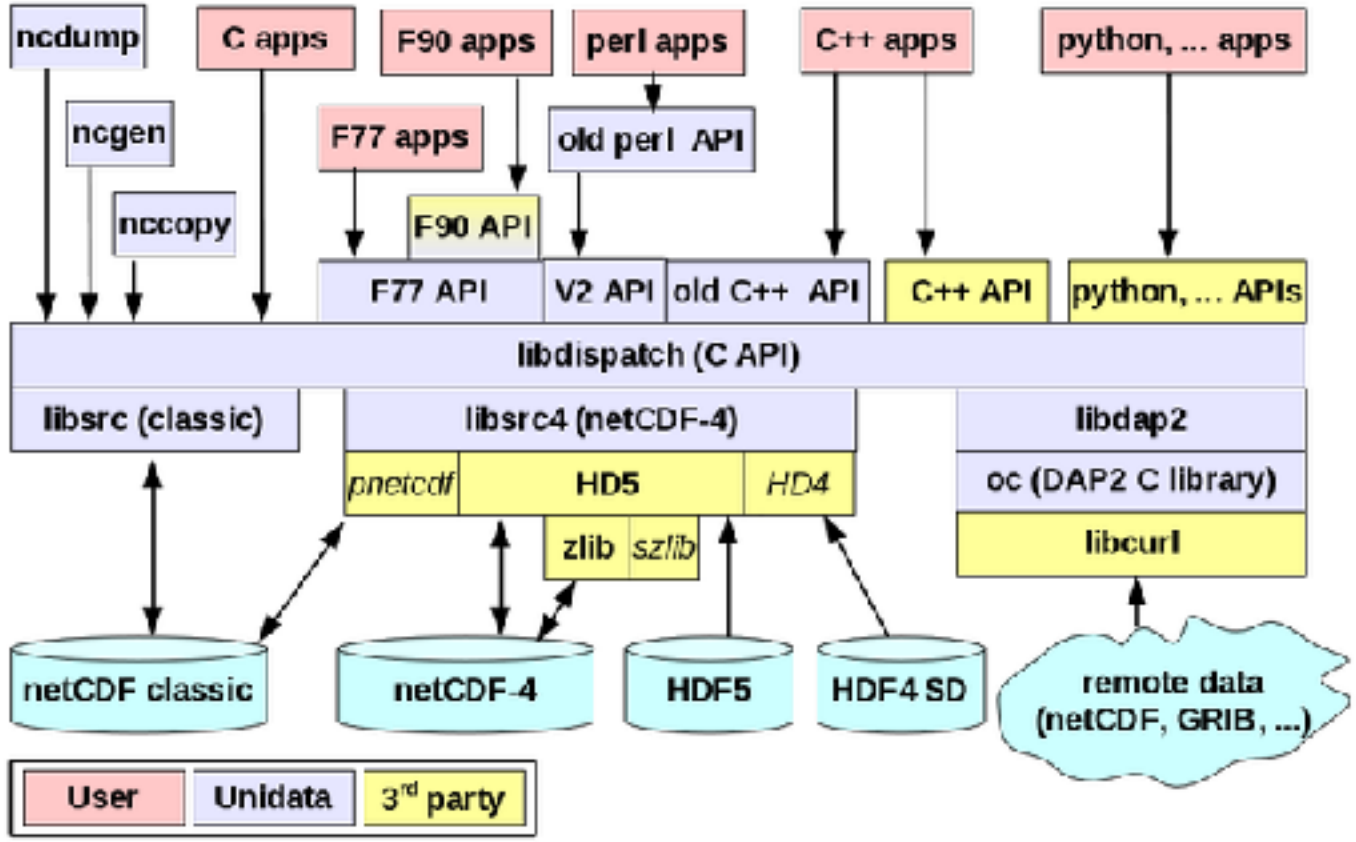
Such as, latitude, longitude, pressure level, time.



Unidata & NetCDF Library Ecosystem

More than a file format…

It's a way of life.



ncdump - h mtmet.nc

```
netcdf mtmet {
dimensions:
    time = 25920 ;
variables:
    int date(time) ;
        date:units = "epoch seconds since Jan 1, 1970 00:00:00z" ;
    float temp(time) ;
        temp:units = "c degrees" ;
    float rh(time) ;
        rh:units = "percentage" ;
    float wind_speed(time) ;
        wind_speed:units = "meters per second" ;
    float wind_direction(time) ;
        wind_direction:units = "direction degrees" ;
    float wind_gust(time) ;
        wind_gust:units = "meters per second" ;

// global attributes:
        :description = "Time series for Mountain Meteorology (MTMET) station on University of Utah campus." ;
        :location = "Salt Lake City, Utah" ;
        :latitude = "40.766573" ;
        :longitude = "-111.828211" ;
}
```

Working with NetCDF

In Python

https://www.getdatajoy.com/learn/Read_and_Write_NetCDF_Files_from_Python

In MATLAB

https://www.mathworks.com/help/matlab/ref/ncread.html

In IDL

http://www.harrisgeospatial.com/docs/NCDF_Overview.html

In R

https://www.getdatajoy.com/learn/Read_and_Write_NetCDF_from_R

Homework #3

Homework can be found here:

http://www.inscc.utah.edu/~u0079358/atmos6910/Class_3/