
Met Office
Iris
A python package for the analysis and visualisation of gridded Met/Ocean data
Philip Elson
9th Dec 2016
www.metoffice.gov.uk
© Crown copyright

Who am I?

- Scientific Python technical lead in the Met Office's analysis, visualisation and data (AVD) team
- Seconded to the Bureau for 6 months – primary focus is the Model Data Services project which aims to deliver standard APIs for model data access
- Contributor to many open source Scientific Python packages, including:
  - Member of the matplotlib executive committee & core contributor for several years – led on the “notebook” backend
  - Creator of conda-forge & cartopy
  - Co-creator of Iris & biggus

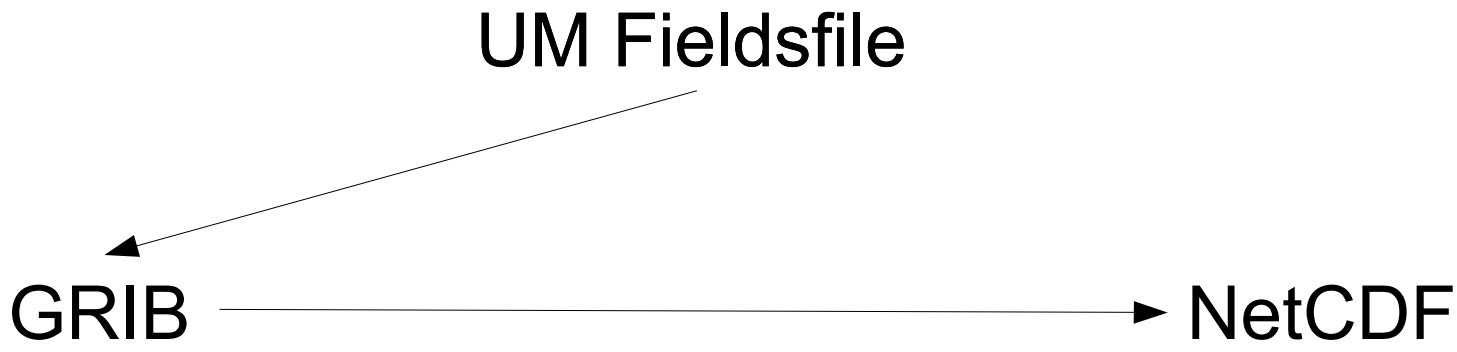
Outline

- What is Iris?
- Iris demo
- Using Iris for novel analysis
- Opportunities for combining Iris with other tools

Audience of this talk:

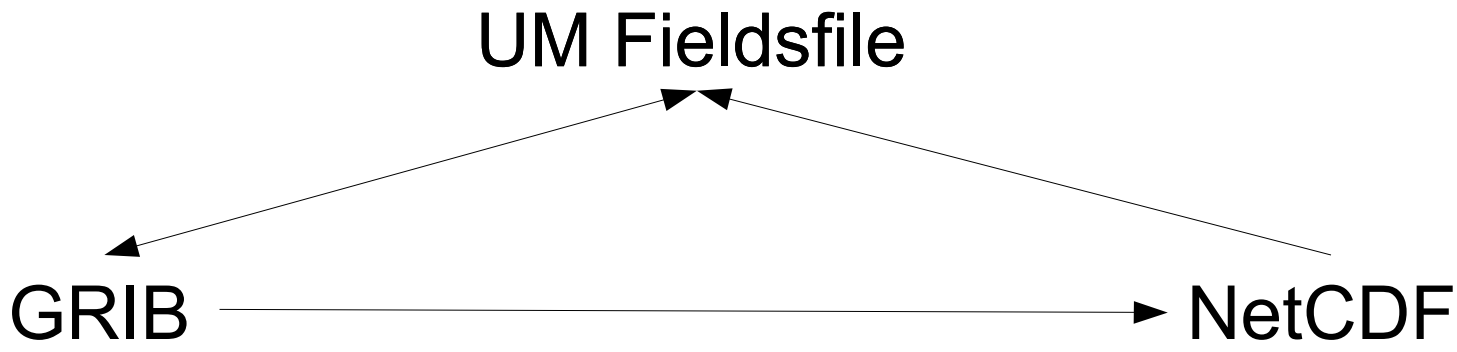
- Those who do data analysis and visualisation of Met/Ocean gridded data

The problem space (circa 2010)



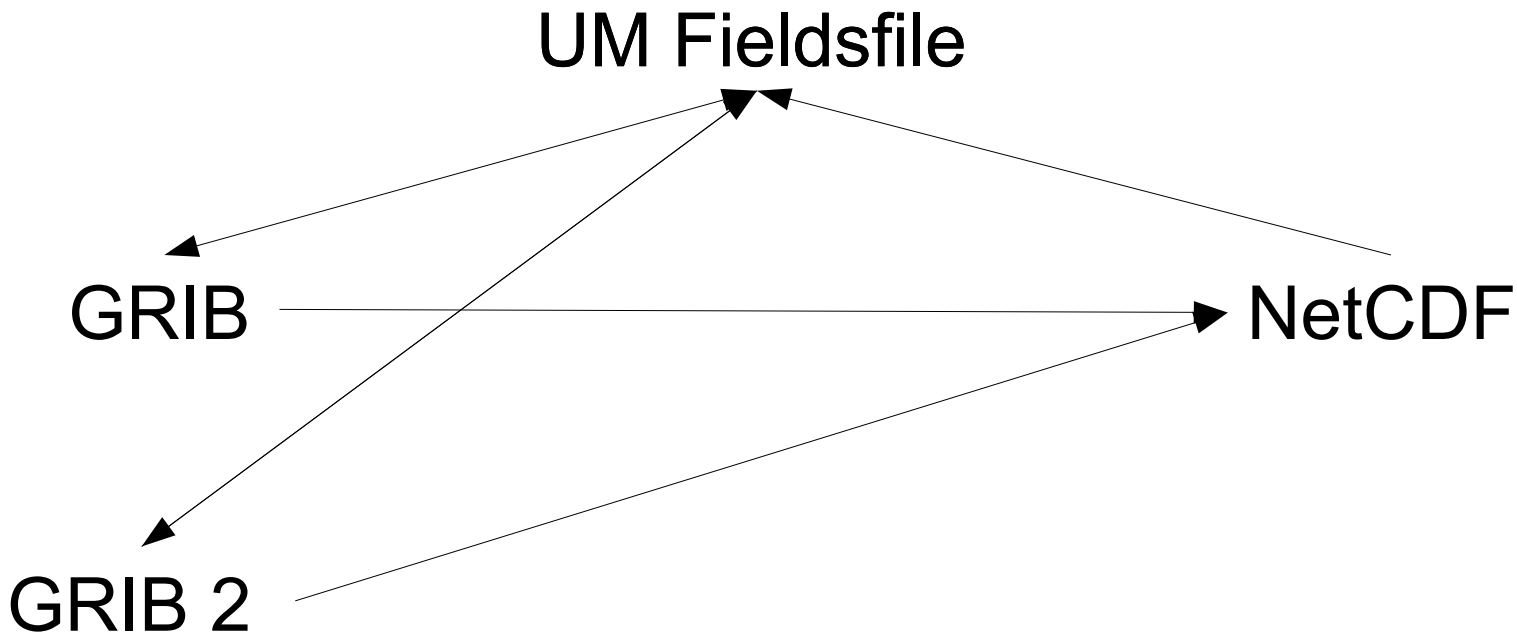
(Downstreaming data)

The problem space (circa 2010)



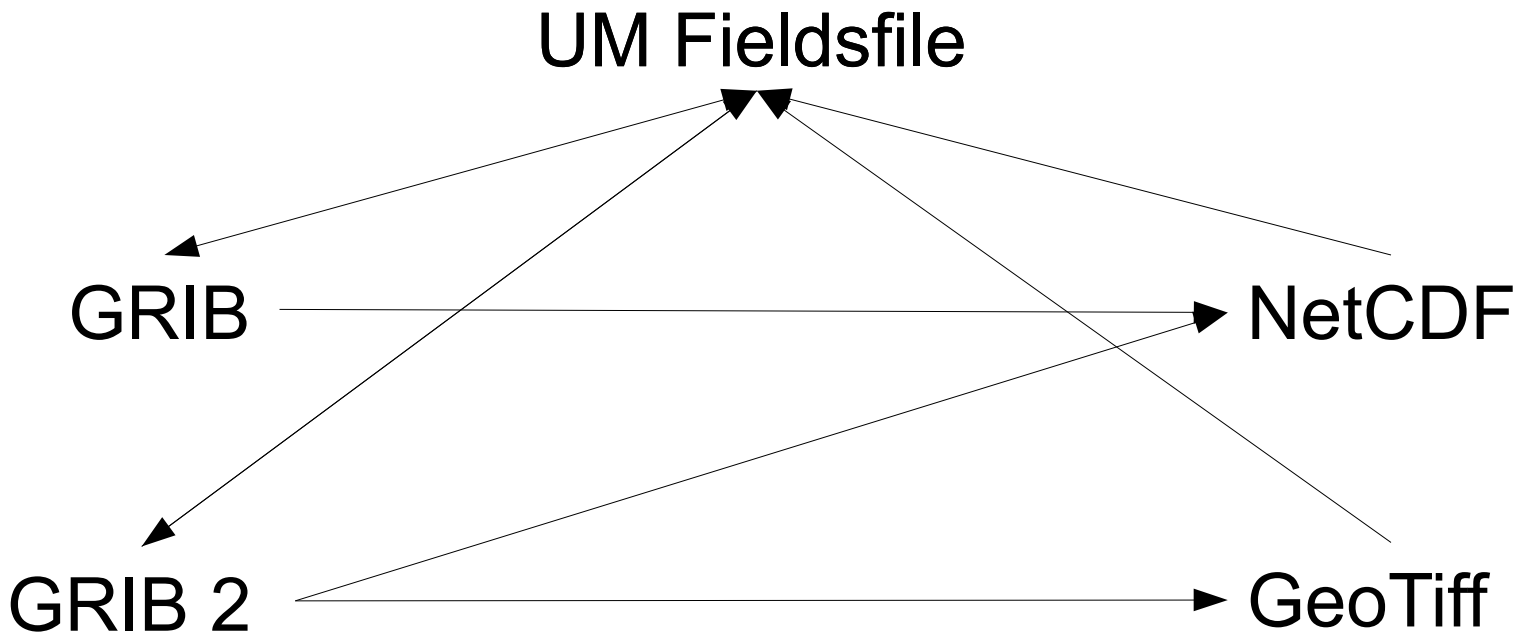
(Data assimilation / Ancillary production)

The problem space (circa 2010)



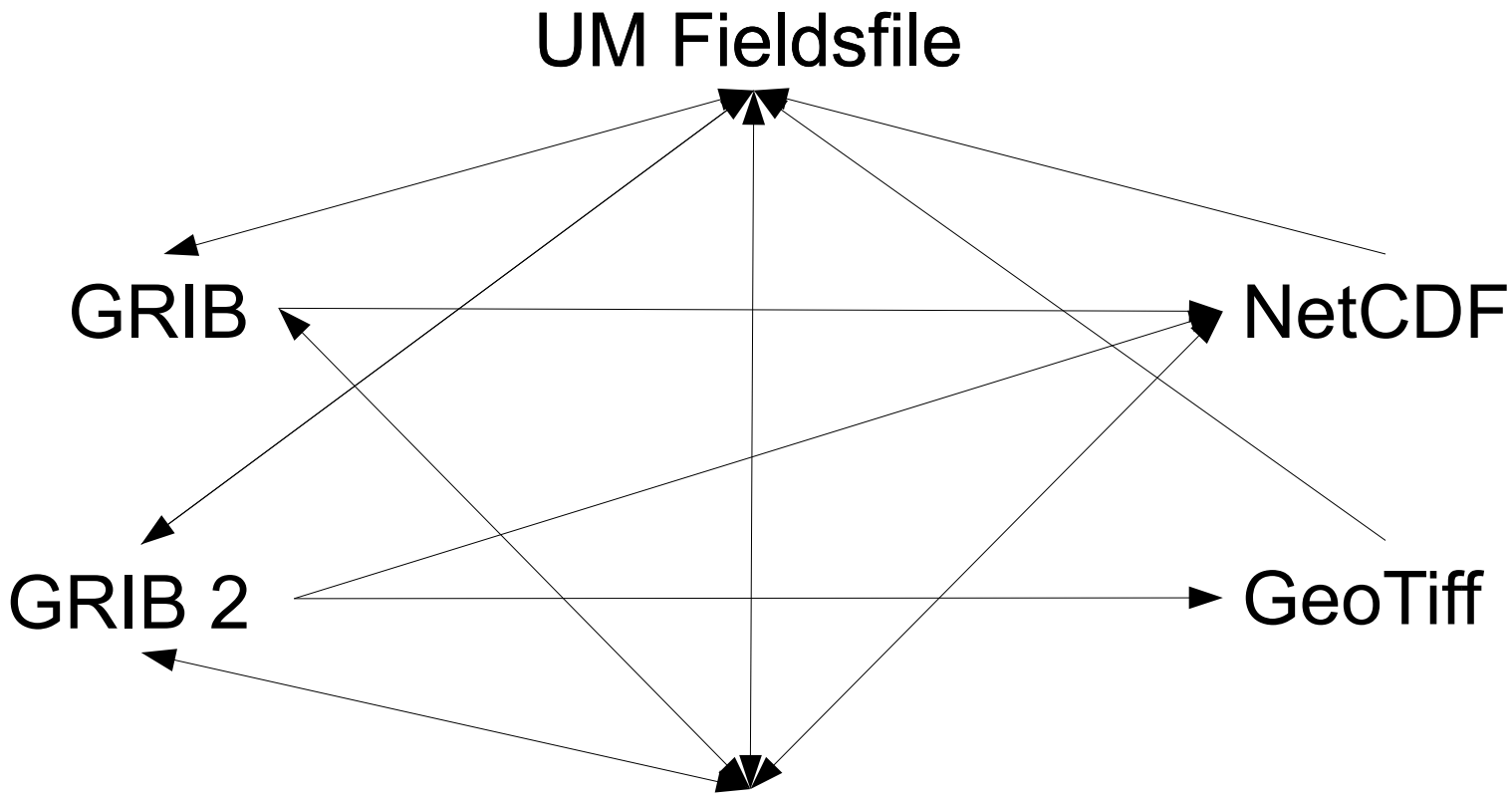
(GRIB 2 is now widespread)

The problem space (circa 2010)



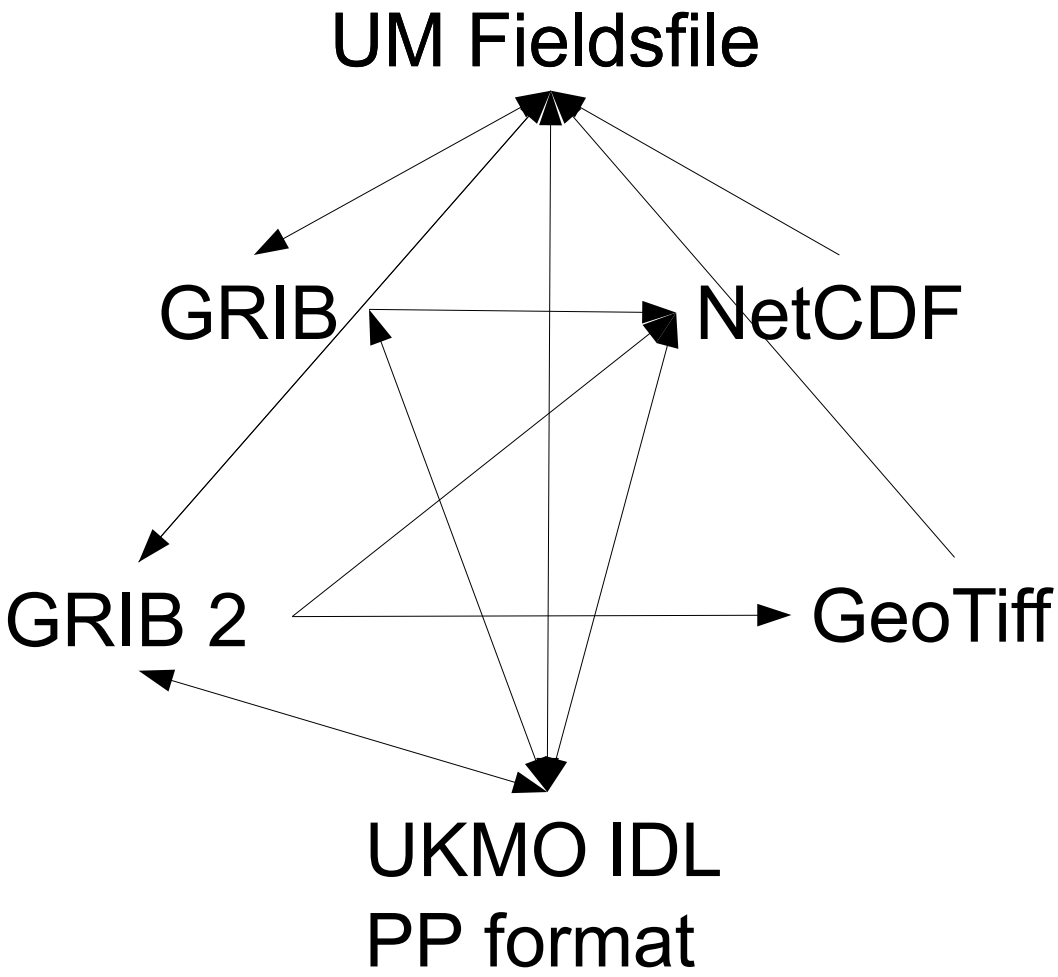
(GeoTiff as a GIS product &
incoming GeoTiff for model ancil)

The problem space (circa 2010)

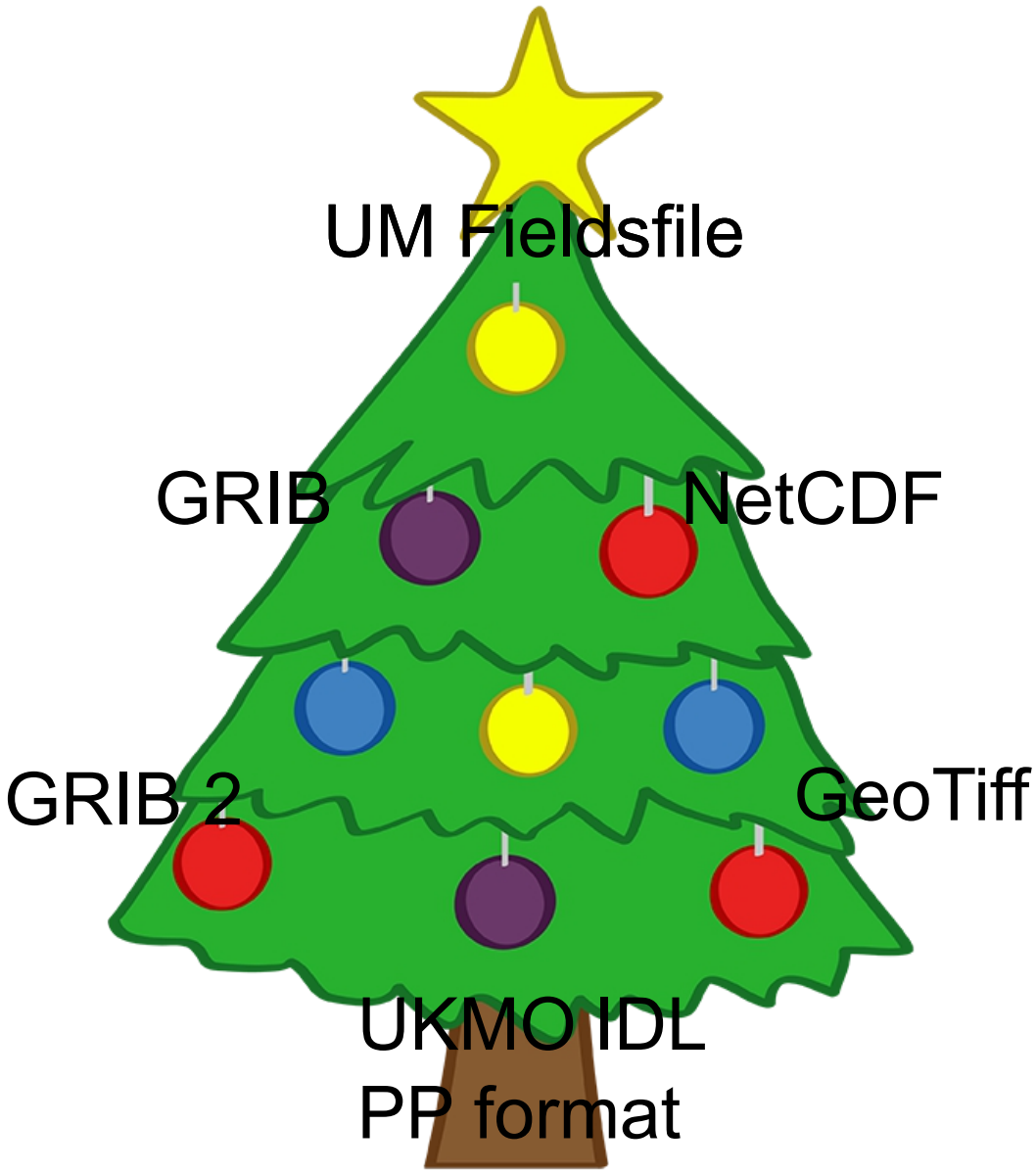


(Met Office's IDL UKMO library
for pre/post-processing,
designed for PP files)

The problem space (circa 2010)

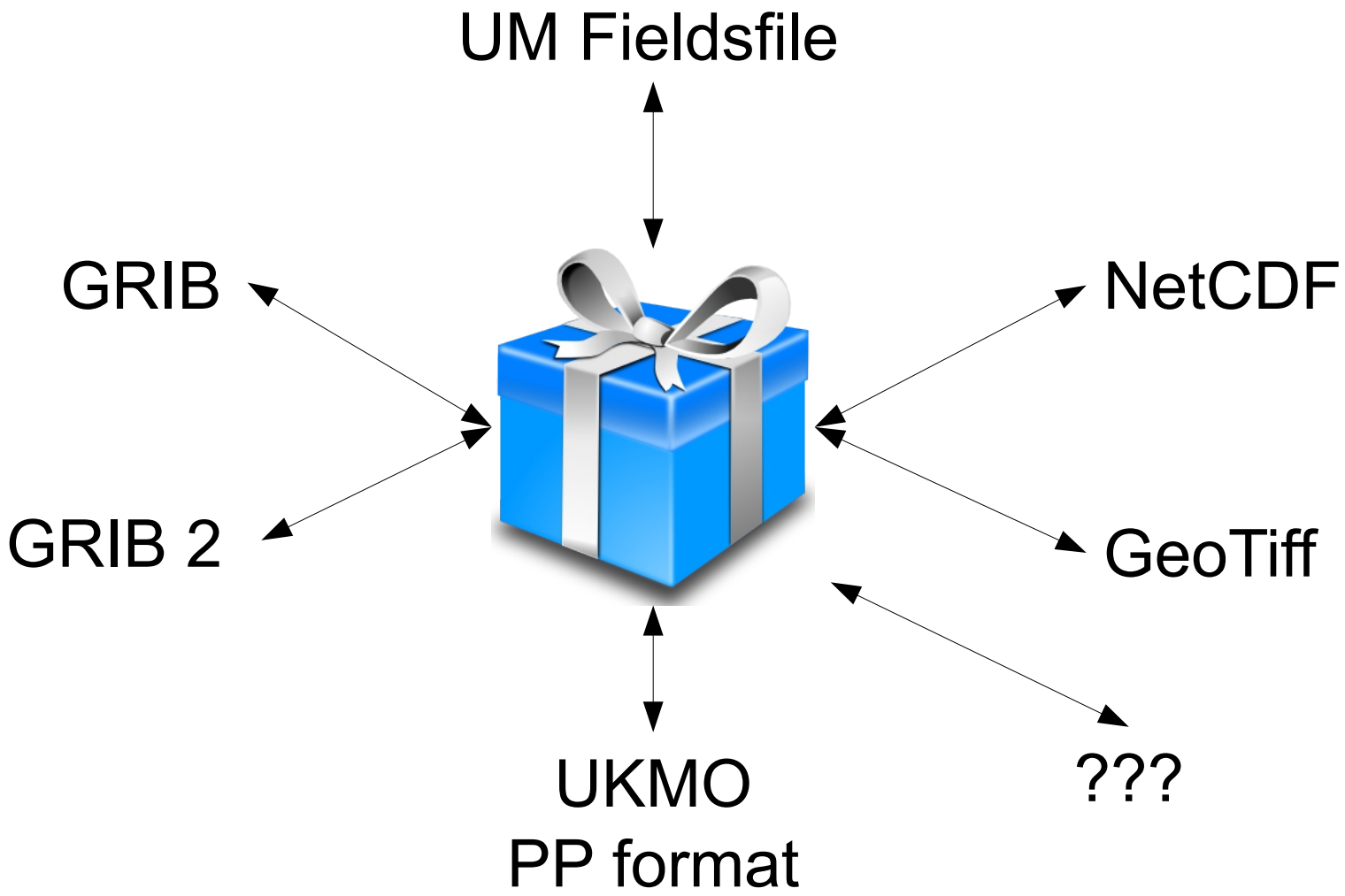


The problem space (circa 2010)

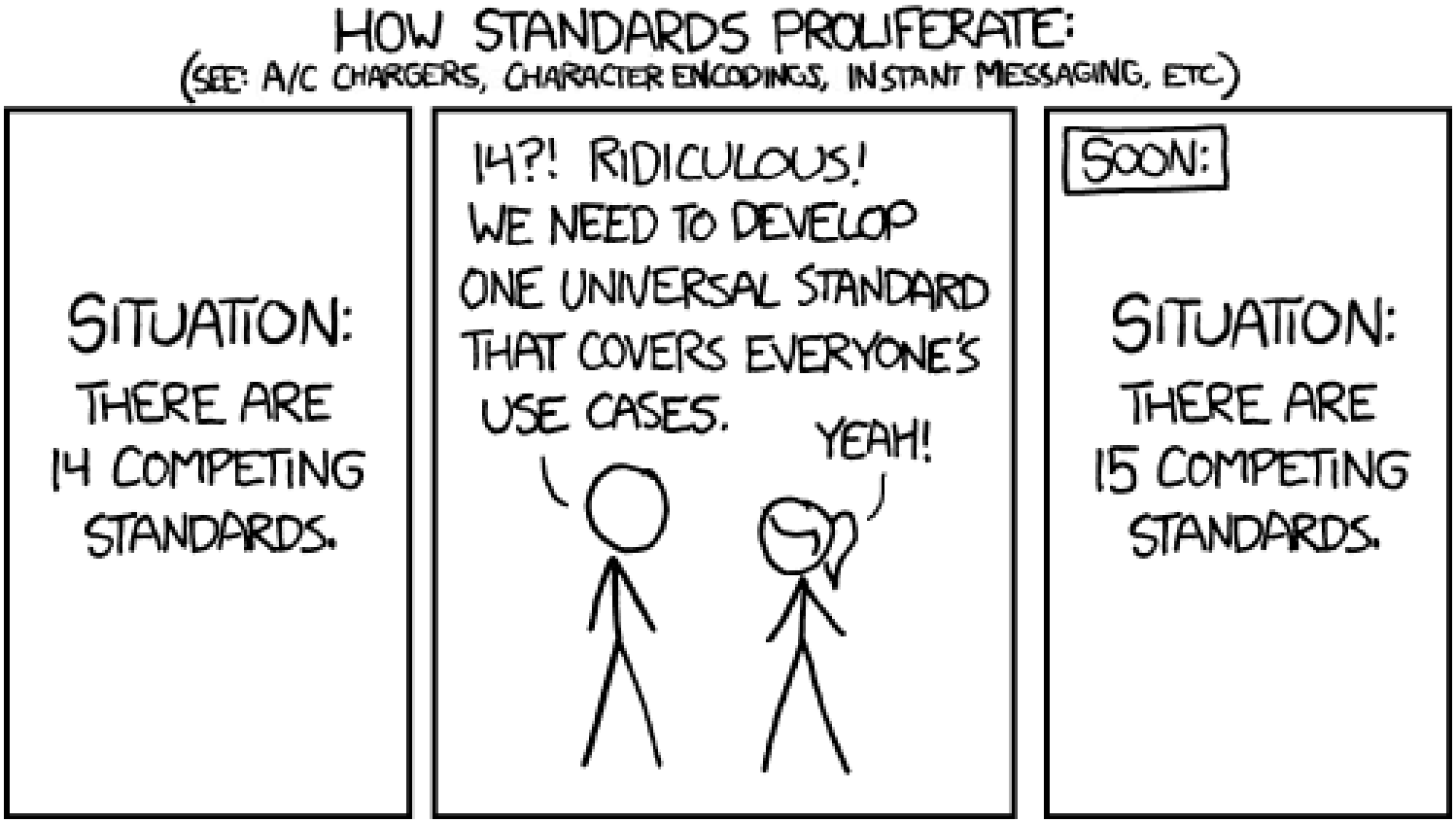


A solution

A solution



Caution



https://xkcd.com/927/

Met Office

What is  Iris ?

python powered

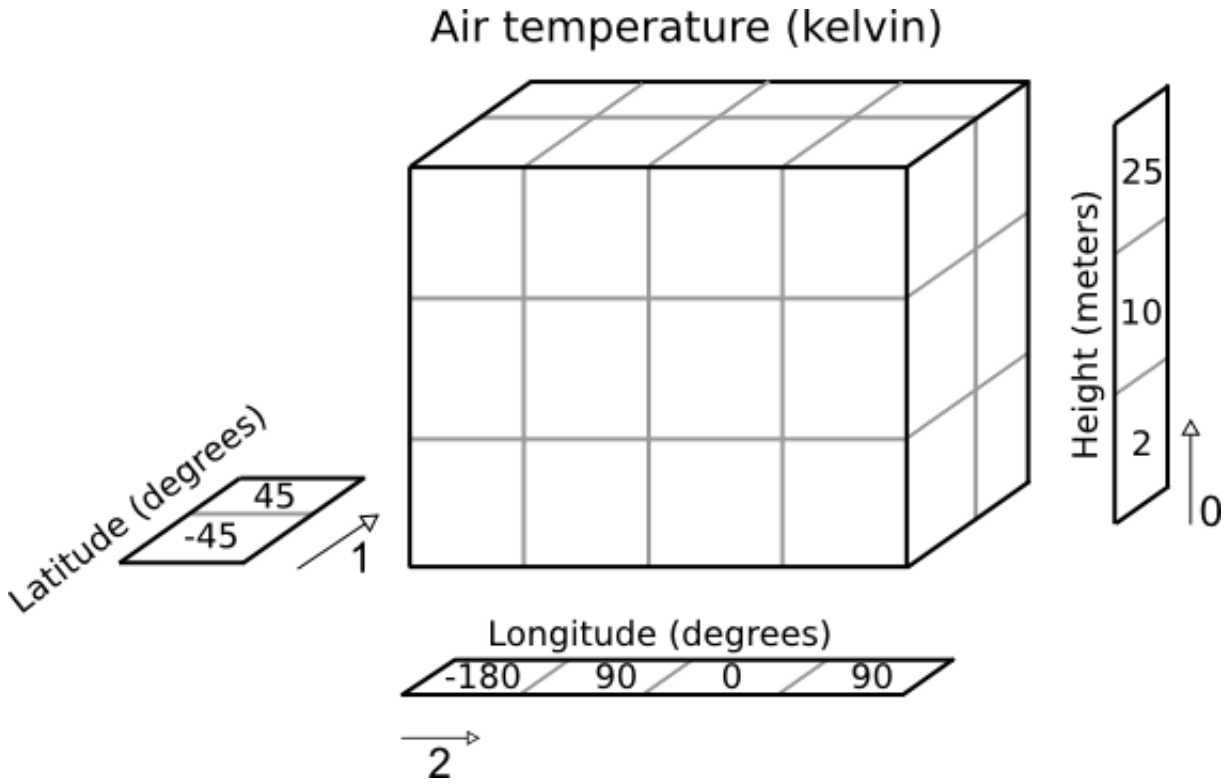
open source initiative

CF Metadata

NetCDF Climate and Forecast Metadata Convention

www.metoffice.gov.uk

Met Office



Met Office

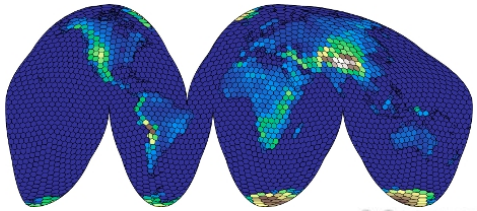
What is Iris ?

netCDF
GRIB
PP
…

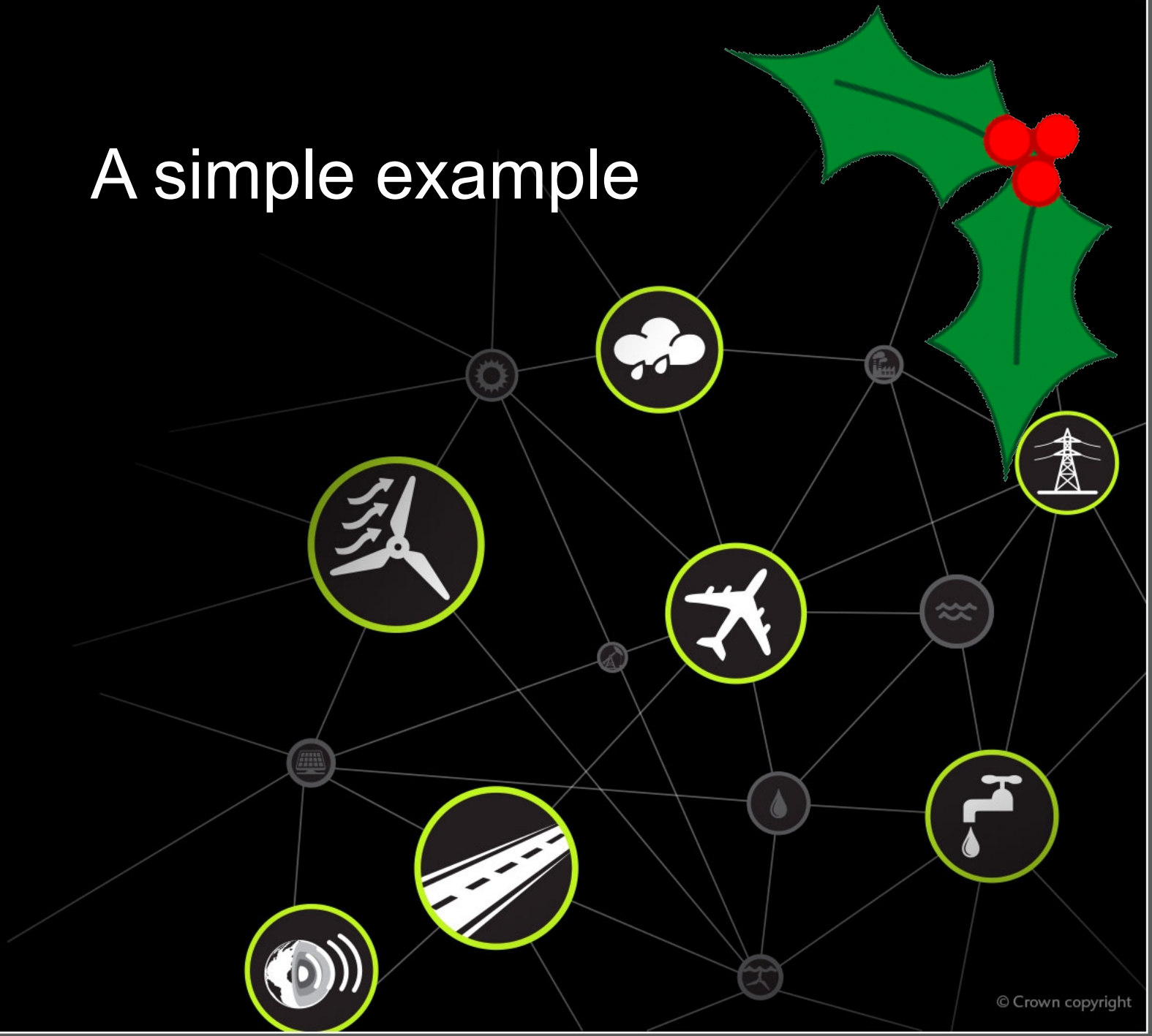
Iris

scikit-image
image processing in python

SciPy



A simple example

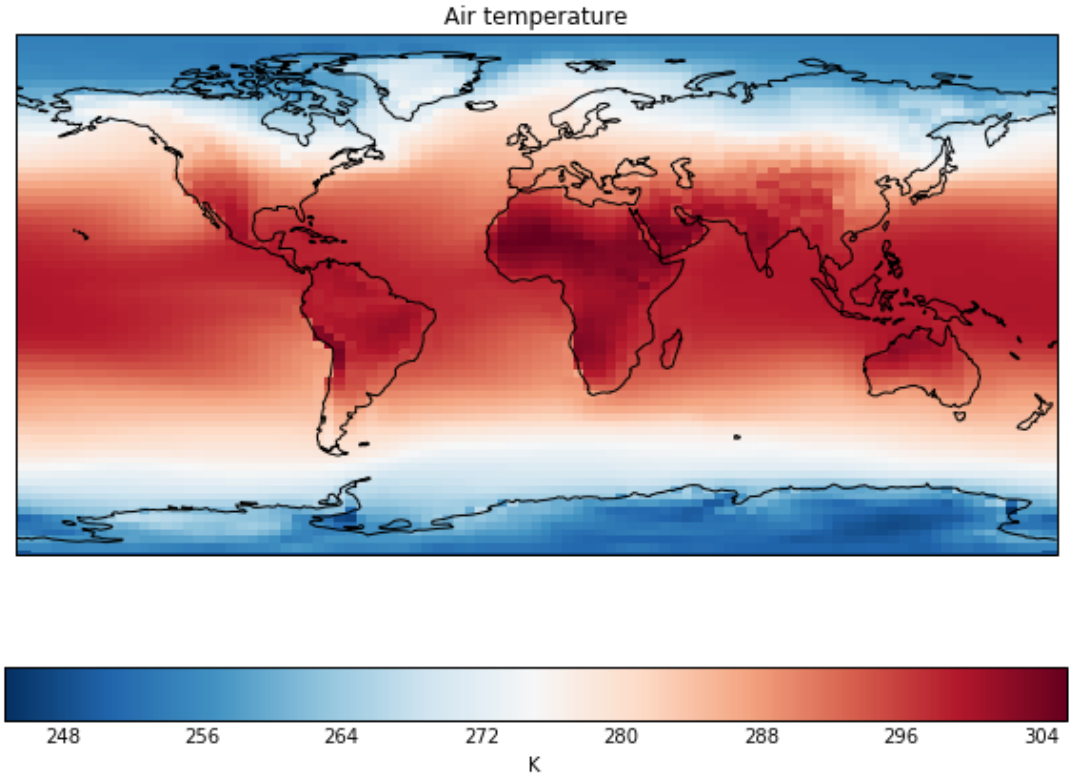


Loading a cube

```
>>> import iris

>>> air_temp = iris.load_cube(filename,
                              'air_temperature')
>>> print(air_temp)

air_temperature / (K) (latitude: 73; longitude: 96)
Scalar coordinates:
   pressure:                  1000.0 hPa
   time: 1998-12-01 00:00:00, bound=(1994-12-01 00:00:00,
                                     1998-12-01 00:00:00)
Attributes:
   STASH: m01s16i203
   source: Data from Met Office Unified Model
```

Plotting with matplotlib

```
>>> import matplotlib.pyplot as plt
>>> import iris.quickplot as qplt

>>> qplt.pcolormesh(air_temp, cmap='RdBu_r')
>>> plt.gca().coastlines()
```



Output:

- PNG
- PDF
- PS
- ...

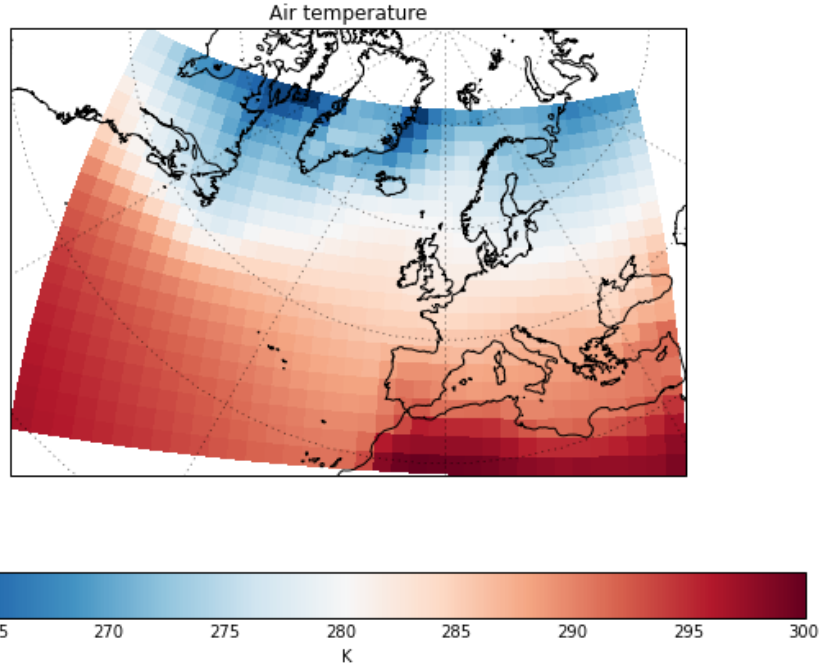
Regridding and interpolation

```
>>> from iris.analysis import Linear

>>> exeter = [('longitude', [-3.5]),
              ('latitude', [50.7])]
>>> exeter_temp = air_temp.interpolate(exeter,
                                       Linear())


>>> mslp_euro = iris.load_cube(filename2)

>>> air_temp_euro = air_temp.regrid(mslp_euro,
                                    Linear())
```

Typically, Iris functions take cubes as input, and return cubes as output.

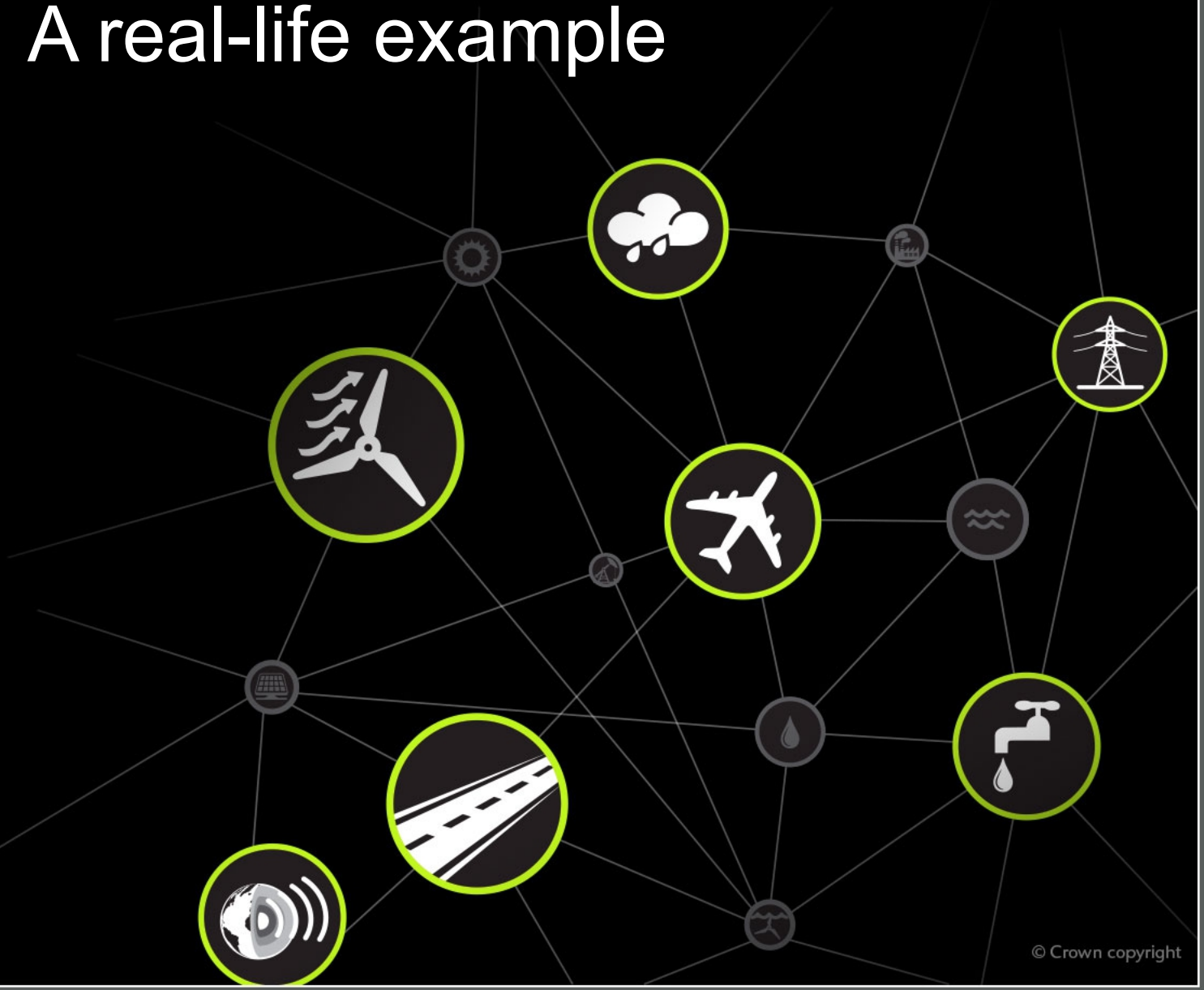
Maps with cartopy

Maps in Iris are drawn by cartopy, a python package developed to solve common dateline and pole problems seen with traditional mapping libraries.



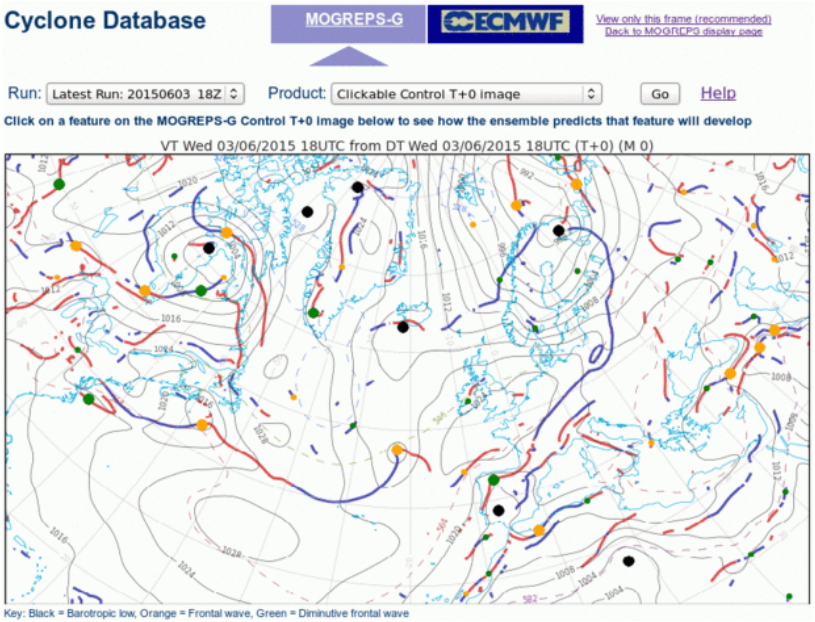

```
>>> from cartopy.crs as ccrs

>>> ax = plt.axes(projection=ccrs.NorthPolarStereo())
>>> qplt.pcolormesh(air_temp_euro, cmap='RdBu_r')

>>> ax.coastlines('50m')
>>> ax.gridlines()
```

A real-life example



www.metoffice.gov.uk

MOGREPS-G Cyclone Database



An algorithm to identify and track fronts and cyclonic features, based on:

Hewson, T.D. & H.A. Titley, 2010: Objective identification, typing and tracking of the complete life-cycles of cyclonic features at high spatial resolution. Meteorol. Appl., 17, 355-381.

Implementing the algorithm

- Load the phenomenon

- Regrid and interpolate data to specific to vertical levels

- Compute isolines for locating phenomenon + isosurfaces for masking phenomenon, based on thresholds from paper

- Compute intersection of isosurfaces and isolines to identify cyclonic features

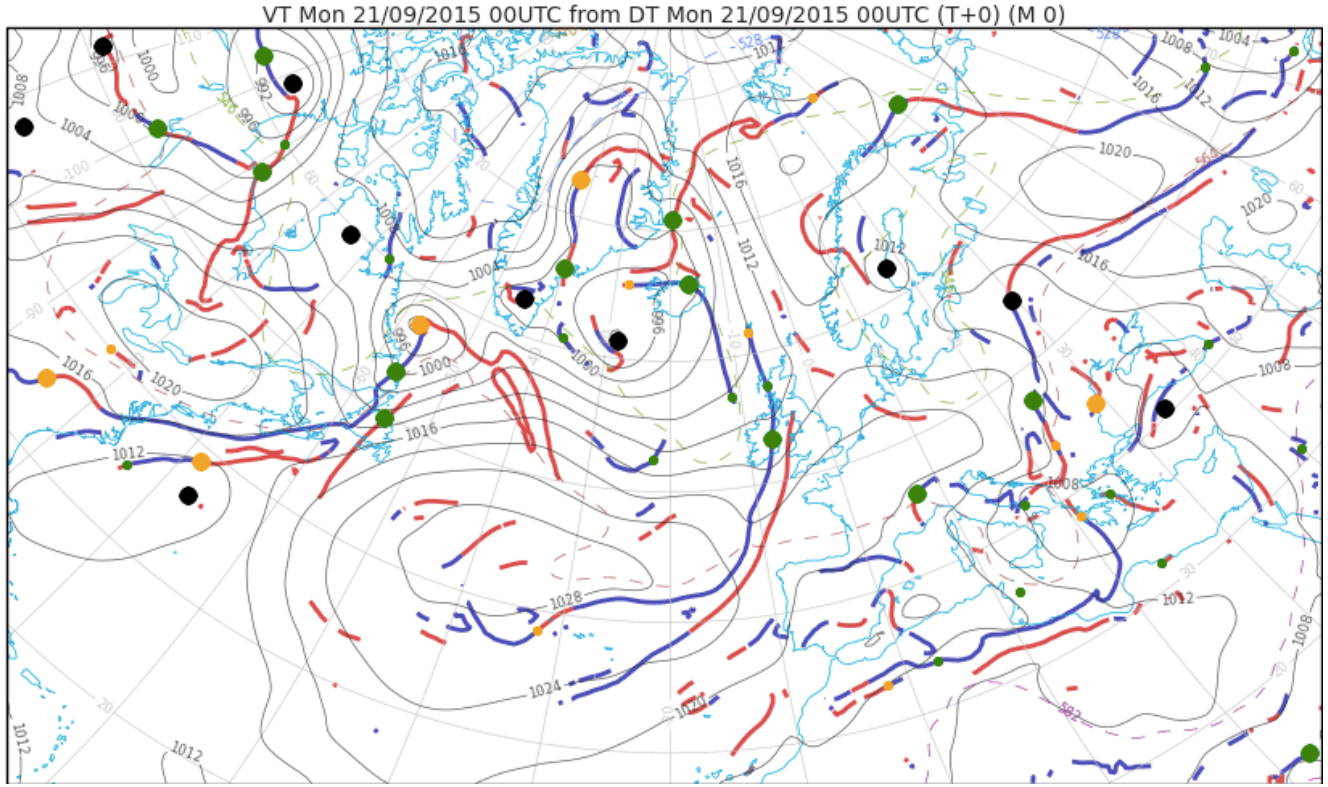
Met Office

- Classify cyclonic features based on phenomenon values

Iris

- Visualise cyclonic features and the underlying diagnostics

Iris matplotlib Cartopy



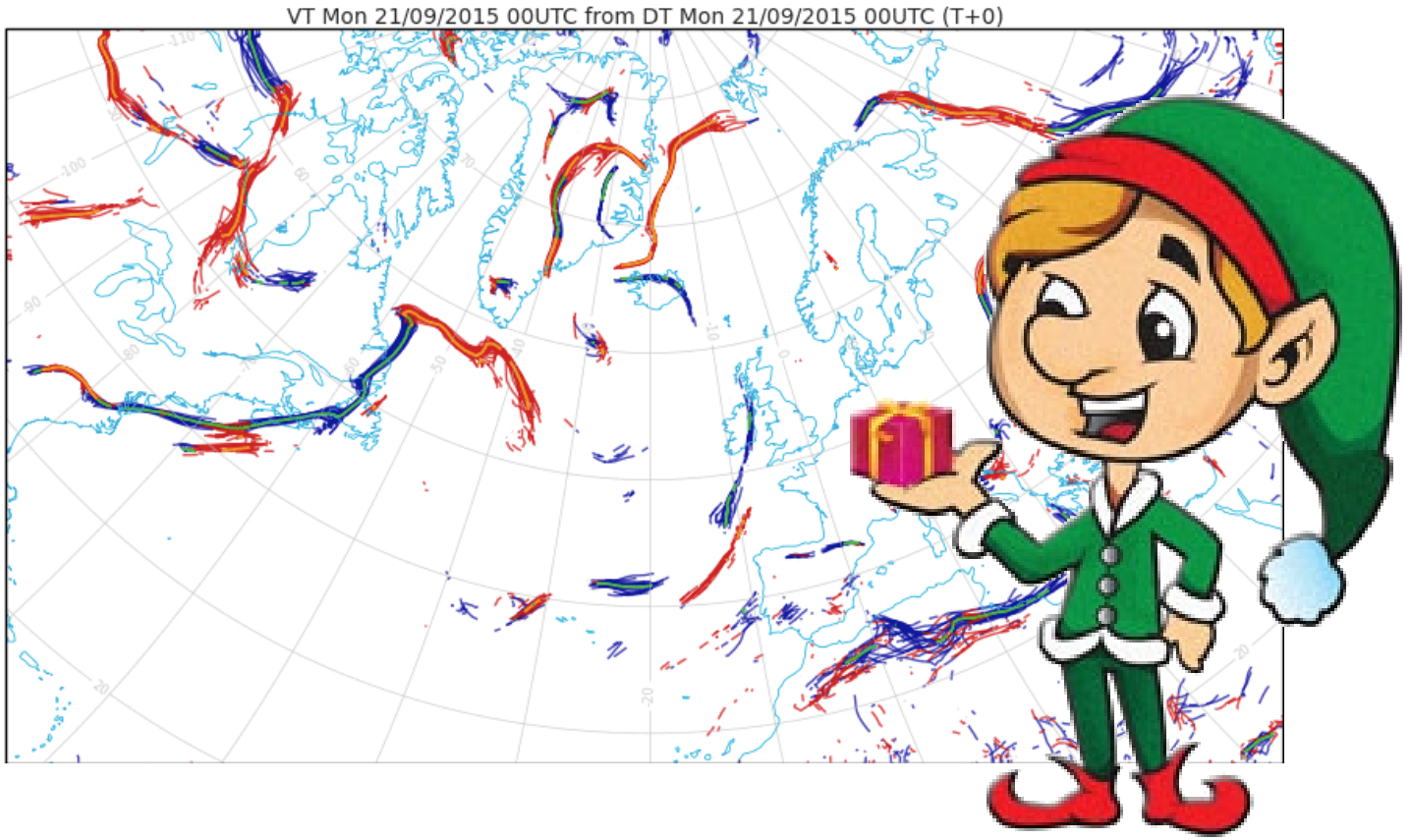
Barotropic Lows **Frontal Waves** **Diminutive Waves**

Met Office

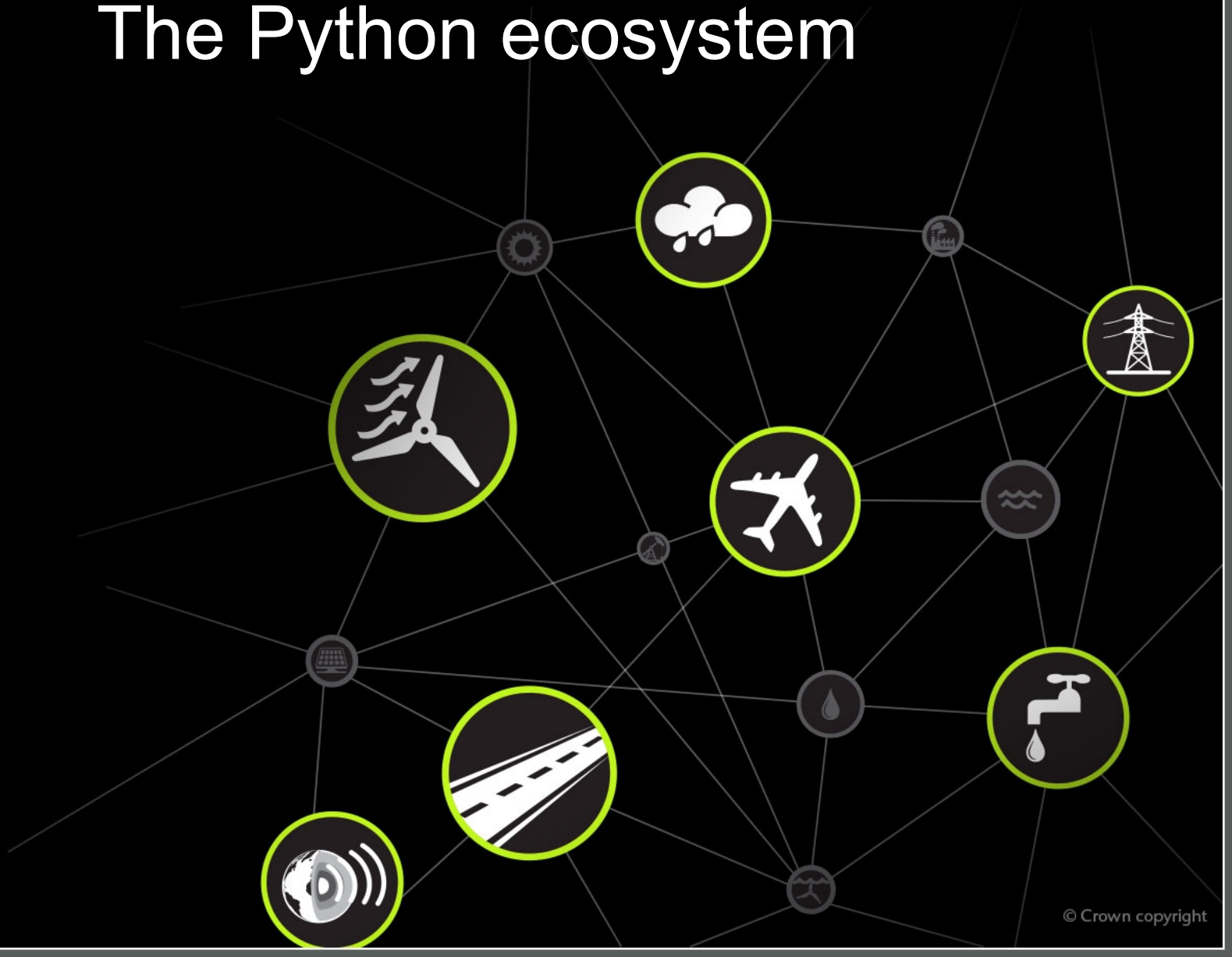
- Visualise fronts as a spaghetti plot

matplotlib

Cartopy



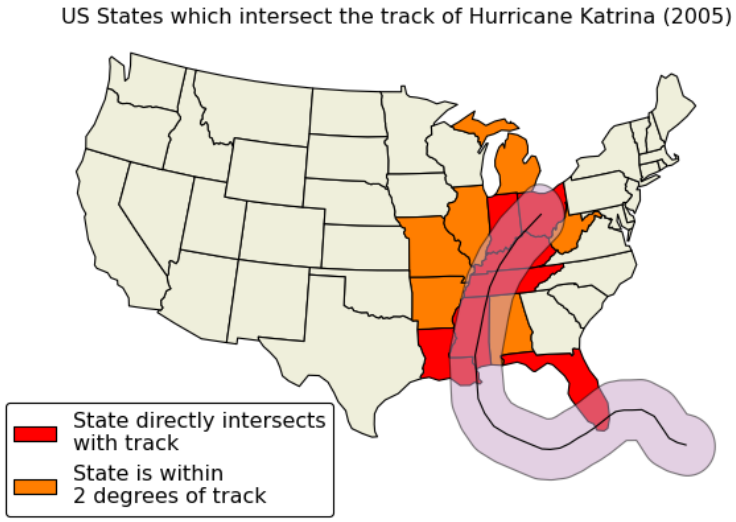
The Python ecosystem



Opportunities within Python

GIS tools:

- Shapely
- Cartopy
- Fiona
- RasterIO
- QGIS



http://scitools.org.uk/cartopy/docs/latest/examples/hurricane_katrina.html

Opportunities within Python

Large data manipulation:

- Cython
- Numba

Tools to optimise slow for-loops using static typing and JIT compilation for C-like performance

- Biggus
- Dask

Opportunities within Python

Large data manipulation:

- Cython
- Numba

- Biggus
- Dask

Biggus example:

```
>>> print(data)
<Array shape=(80640, 4, 144, 192)
        dtype=dtype('float32') size=33.22 GiB>

>>> stats = [biggus.mean(data, axis=0),
             biggus.max(data, axis=0),
             biggus.min(data, axis=0)]

>>> biggus.ndarrays(stats)
```

Result in ~4m45s on an Intel Xeon E5520 with 8GiB memory, bound by I/O not CPU.

Iris is using Biggus for many of it operations. This means that we can load, analyse and save cubes *way* beyond the available system memory.

Next steps

How does it perform?

Already does out-of-core for many operations (though not *yet* interpolations) and is becoming lazier and lazier...

Iris' UM Fieldsfile loading (python) was benchmarked in a like-for-like comparison with UM utils (modern Fortran) and found to be 4x faster. New UM utils to be based off of Iris' fieldsfile code (extended and packaged as “mule”). For heavy I/O good algorithms are king.

Interpretation of metadata is a bottleneck within Iris (particularly FF/GRIB → cube) – many opportunities for optimisation, including loading concurrently across a cluster

For cases that work out-of-core on data from a network mounted disk (e.g. GPFS, NFS), a proof-of-concept exists to distribute the processing across many nodes with *no* change needed to user code

What about xarray?

- xarray is an Nd datamodel for netCDF files, written in Python
- Superbly written, naturally extends Pandas to Nd
- Developed by Stephan Hoyer from Iris concept[1]
- Target's a wider audience – more community buy-in, less domain specific utilities (e.g. no interpolation, no masked array out-of-core, limited interpretation of common standards such as CF)

Long term aim is looking to Iris using the underlying xarray data structures, with Iris providing the bells and whistles of domain specific functionality.

1. http://xarray.pydata.org/en/stable/faq.html#what-other-netcdf-related-python-libraries-should-i-know-about

Installing Iris

On a non-BoM machine:

```
conda install -c conda-forge iris
```

Iris installation in /apps anticipated as part of MDS project, already available on some systems (e.g. Raijin).

Met Office

Questions

Further reading:
github.com/scitools/courses

Slides from presentation:
github.com/pelson/BoM-Iris-presentation

Email: philip.elson@bom.gov.au
Github: github.com/pelson
Twitter: @pypelson