Met Office

# Iris

## A python package for the analysis and visualisation of Meteorological data

Philip Elson

30th Sept 2015

© Crown copyright

# What is Iris ?

python powered

CF Metadata

NetCDF Climate and Forecast Metadata Convention

open source initiative

© Crown copyright

Met Office

# What is Iris ?

netCDF

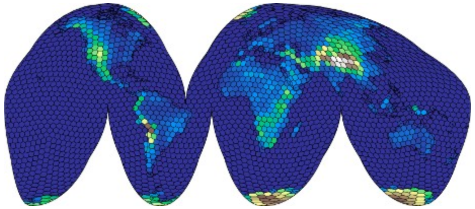
GRIB

PP

...

scikit-image
image processing in python

SciPy



© Crown copyright

Met Office

# An example

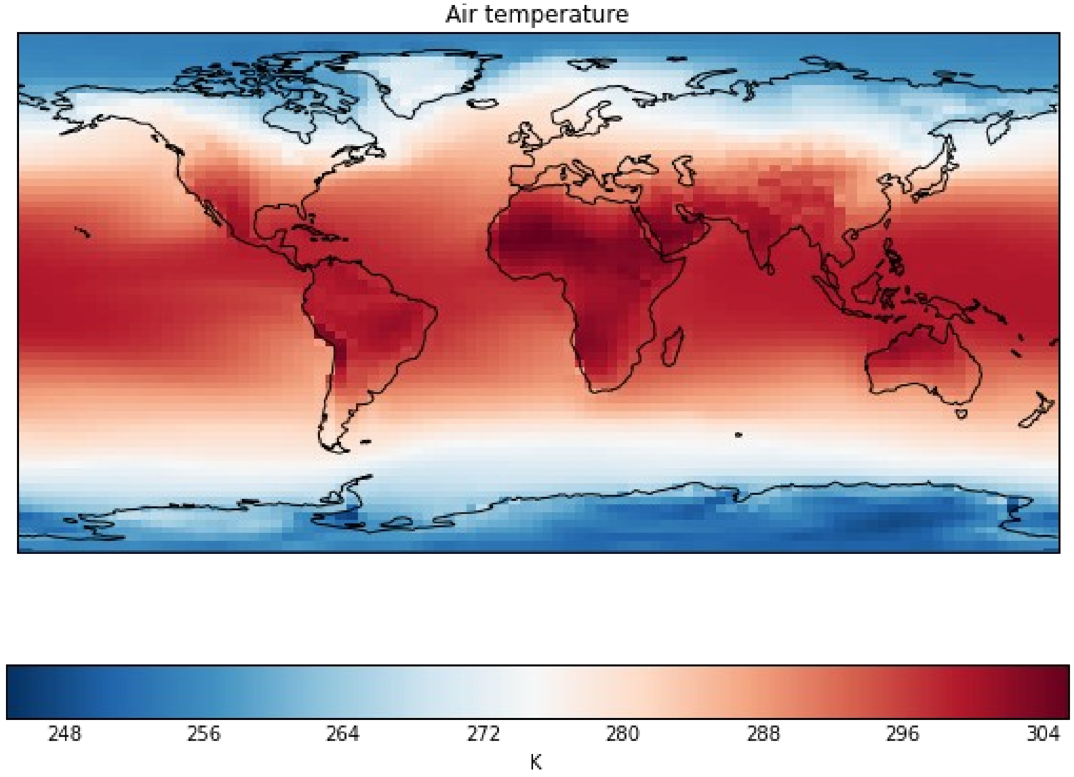
© Crown copyright

# 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:
   forecast_period:          6477 hours
   forecast_reference_time: 1998-03-01 03:00:00
   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()
```

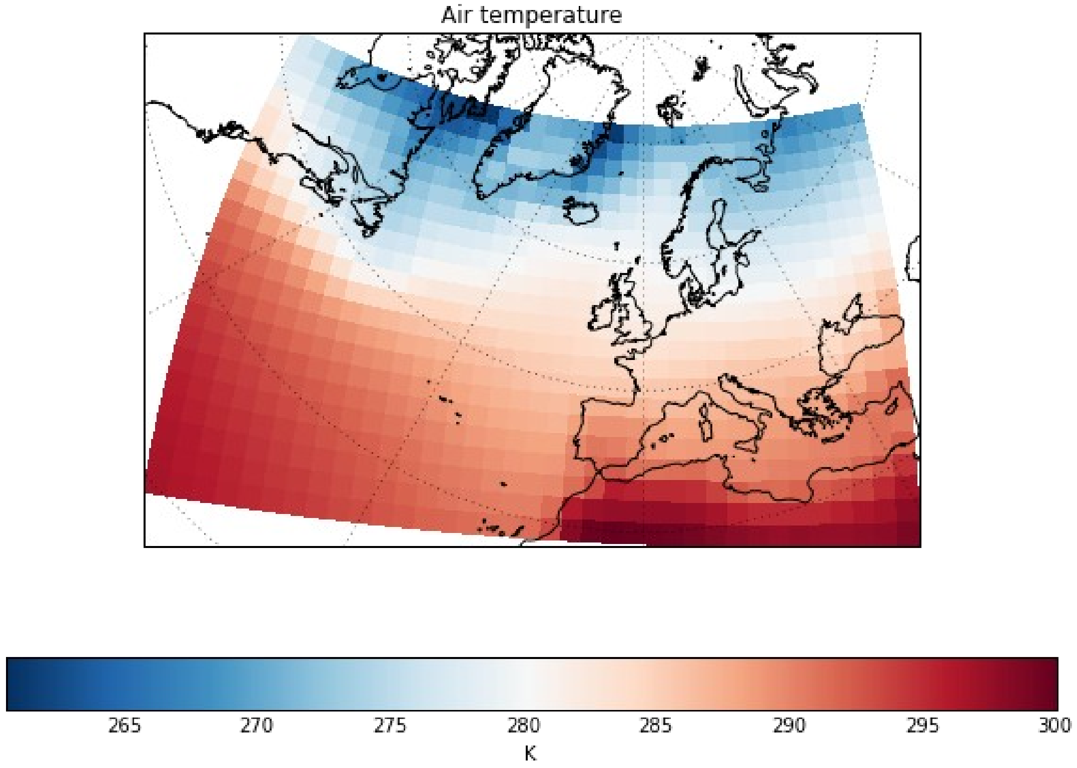


# Regridding and interpolation

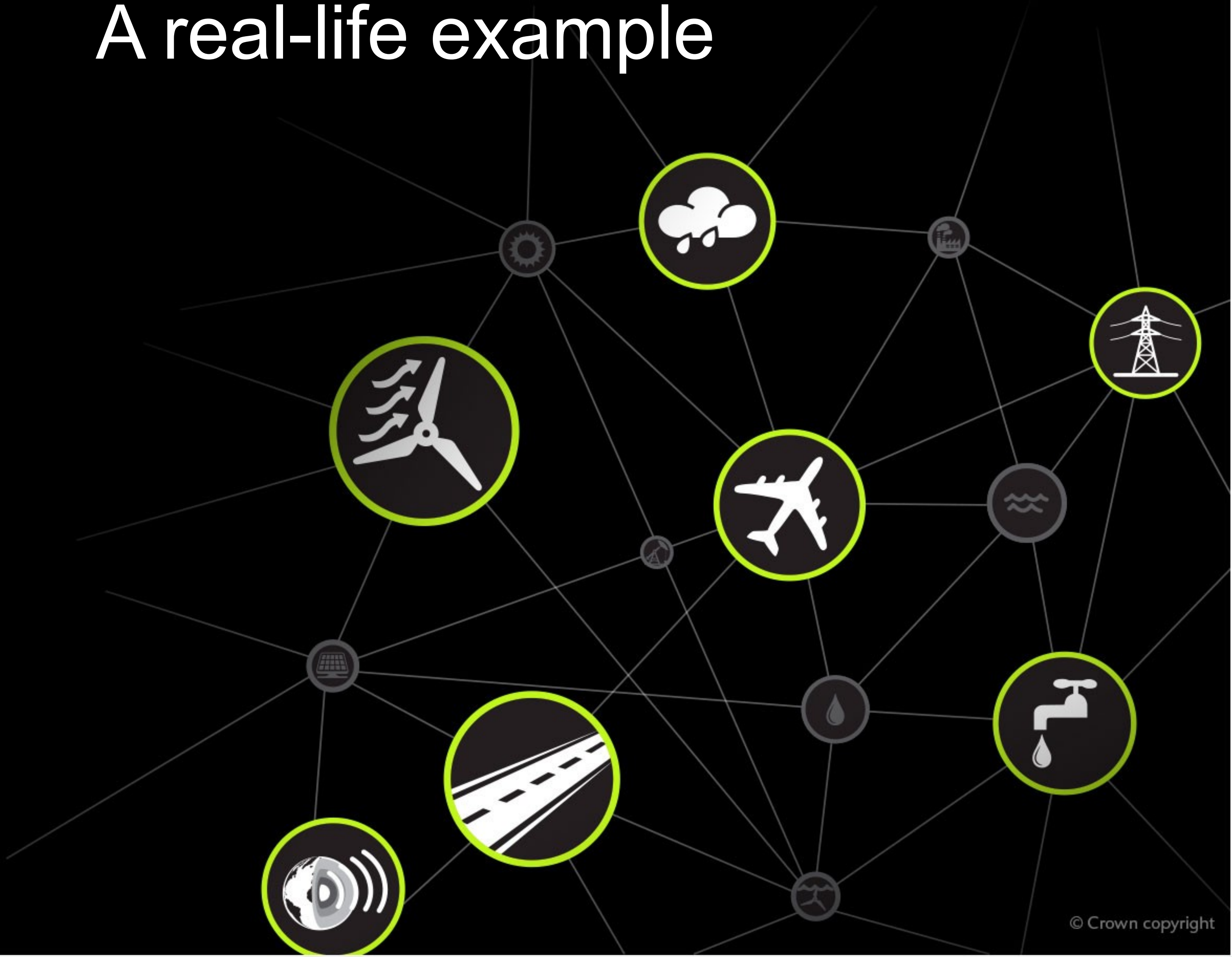
```
>>> from iris.analysis import Linear

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


>>> mslp_euro = iris.load_cube(filename2)

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

# Maps with cartopy

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

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

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

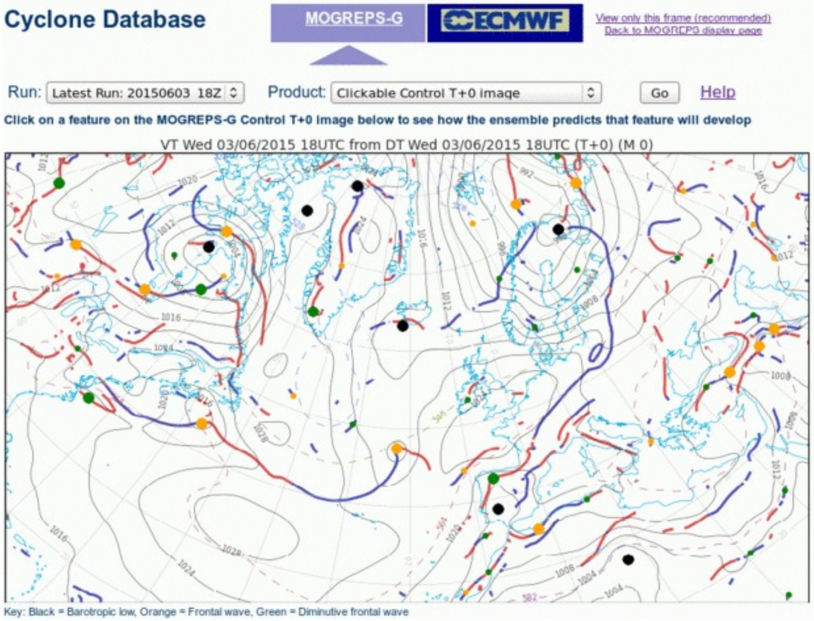


# A real-life example



© Crown copyright

# MOGREPS-G Cyclone Database



Cyclone Database

MOGREPS-G
ECMWF
View only this frame (recommended)
Back to MOGREPS display page

Run: Latest Run: 20150603 18Z

Product: Clickable Control T+0 image

Click on a feature on the MOGREPS-G Control T+0 image below to see how the ensemble predicts that feature will develop

VT Wed 03/06/2015 18UTC from DT Wed 03/06/2015 18UTC (T+0) (M 0)

Key: Black = Barotropic low, Orange = Frontal wave, Green = Diminutive frontal wave

Based on the algorithm developed in

*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.

© Crown copyright

Met Office

# Implementing the algorithm

- Load the phenomenon

Iris

- Regrid and interpolate data to specific to vertical levels

Iris

- Compute isolines for locating phenomenon + isosurfaces for masking phenomenon

matplotlib Cartopy

- Compute intersection of isosurfaces and isolines to identify cyclones

Shapely

© Crown copyright

Met Office

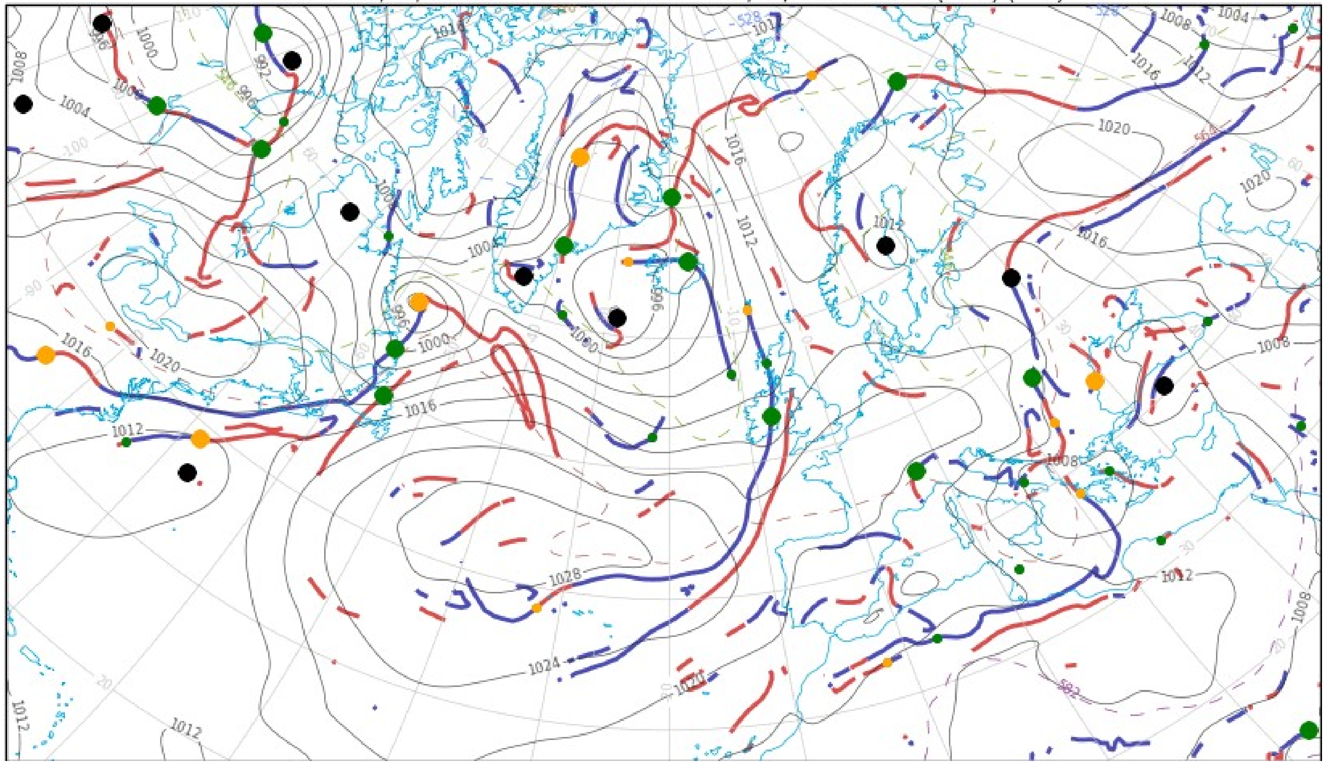
- Classify cyclones based on phenomenon values

Iris NumPy

- Visualise cyclones and the underlying phenomenon

Iris matplotlib cartopy

VT Mon 21/09/2015 00UTC from DT Mon 21/09/2015 00UTC (T+0) (M 0)



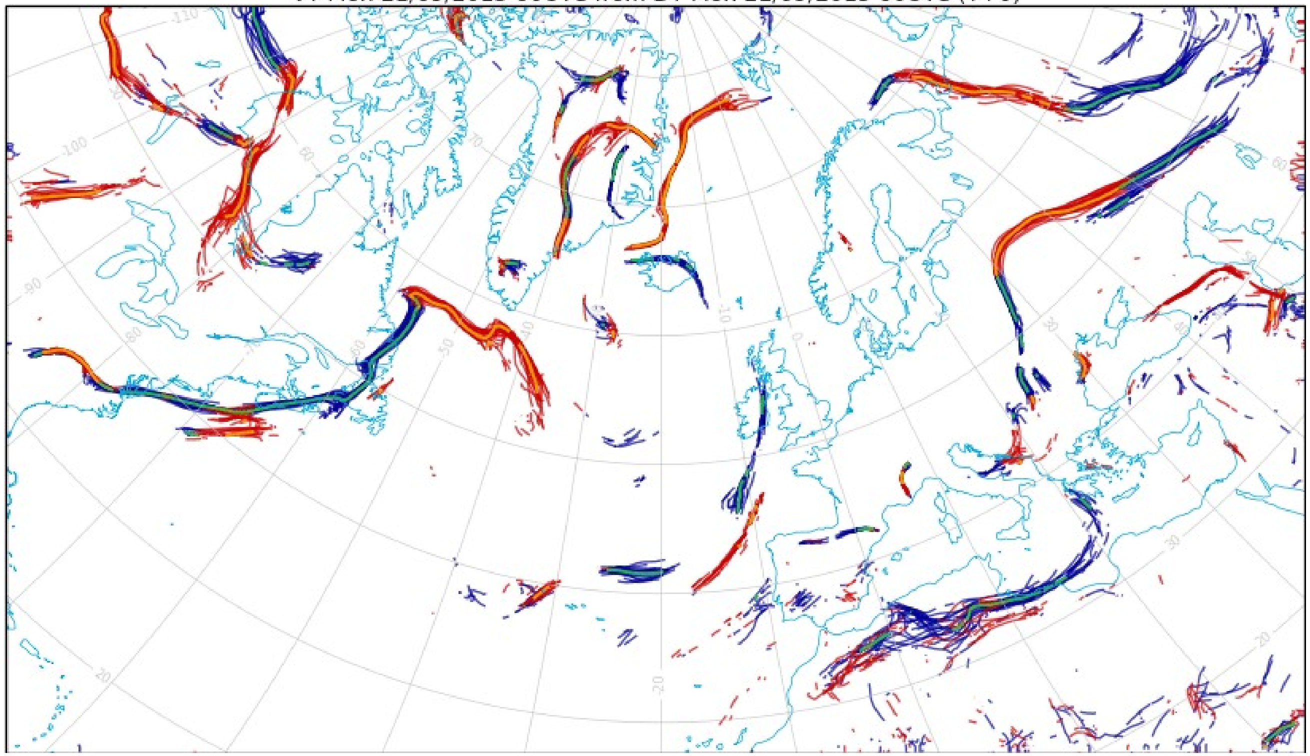
**Barotropic Lows** **Frontal Waves** **Diminutive Waves**

© Crown copyright

- Visualise cyclones as a spaghetti plot

Iris matplotlib cartopy

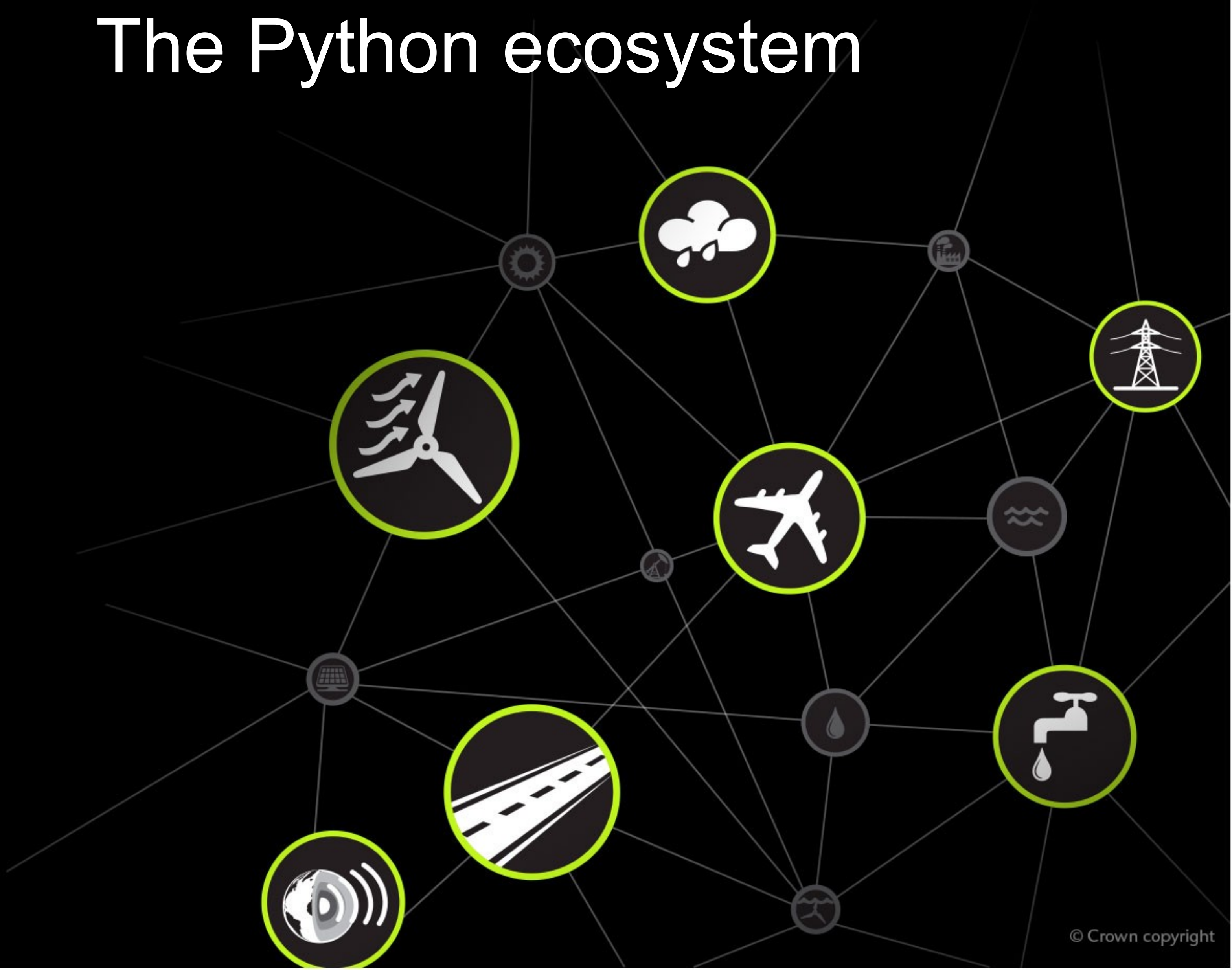
VT Mon 21/09/2015 00UTC from DT Mon 21/09/2015 00UTC (T+0)



© Crown copyright

Met Office

# The Python ecosystem

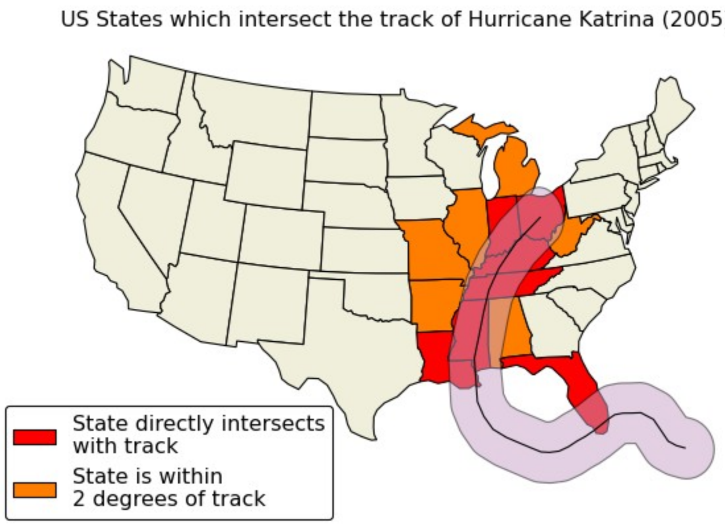


© Crown copyright

# Opportunities within Python

GIS:

- Shapely
- Cartopy
- Fiona
- RasterIO
- QGIS


US States which intersect the track of Hurricane Katrina (2005)
🟥 State directly intersects with track
🟧 State is within 2 degrees of track

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

Camp, J., Roberts, M., MacLachlan, C., Wallace, E., Hermanson, L., Brookshaw, A., Arribas, A., Scaife, A. A., Mar. 2015. *Seasonal forecasting of tropical storms using the met office GloSea5 seasonal forecast system*. Quarterly Journal of the Royal Meteorological Society

© Crown copyright

Met Office

# Opportunities within Python

Large data manipulation:

- Numpy
- Cython
- Numba
- Biggus
- Dask

**Biggus example (already used within Iris):**

```
>>> 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.**

© Crown copyright

# Installing Iris

```
conda install --channel SciTools iris
```

Conda can be downloaded as part of “**miniconda**”: http://conda.pydata.org/miniconda.html

© Crown copyright

Met Office

# Questions

**Links from presentation:**
github.com/pelson/ecmwf-vis-2015

Github: github.com/pelson
Twitter: @pypelson

© Crown copyright

Met Office

# Iris

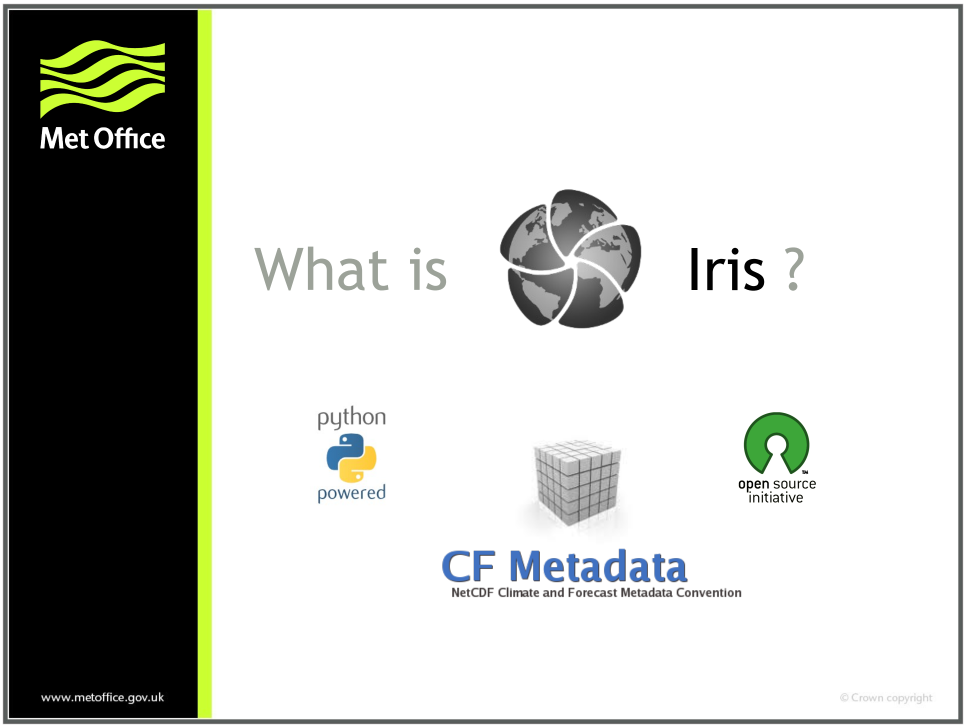
## A python package for the analysis and visualisation of Meteorological data

Philip Elson

30th Sept 2015

www.metoffice.gov.uk

© Crown copyright



Met Office

## What is Iris ?

python powered

open source initiative

CF Metadata
NetCDF Climate and Forecast Metadata Convention

www.metoffice.gov.uk

© Crown copyright

Iris is an open source Python package, at the core of which is Iris' data model – the cube.

A cube is an interpretation of the CF Metadata conventions implemented as an in-memory data model with a Pythonic interface.

Despite the name, the cube is really a hyperrectangle – it can have an arbitrarily large number of dimensions, each of different length.

Attached to each cube is extensible metadata which describes the phenomenon it represents, as well as coordinates which describe the dimensions of the data.

Met Office

What is Iris ?

netCDF
GRIB
PP
…

scikit-image
image processing in python

SciPy

www.metoffice.gov.uk

© Crown copyright

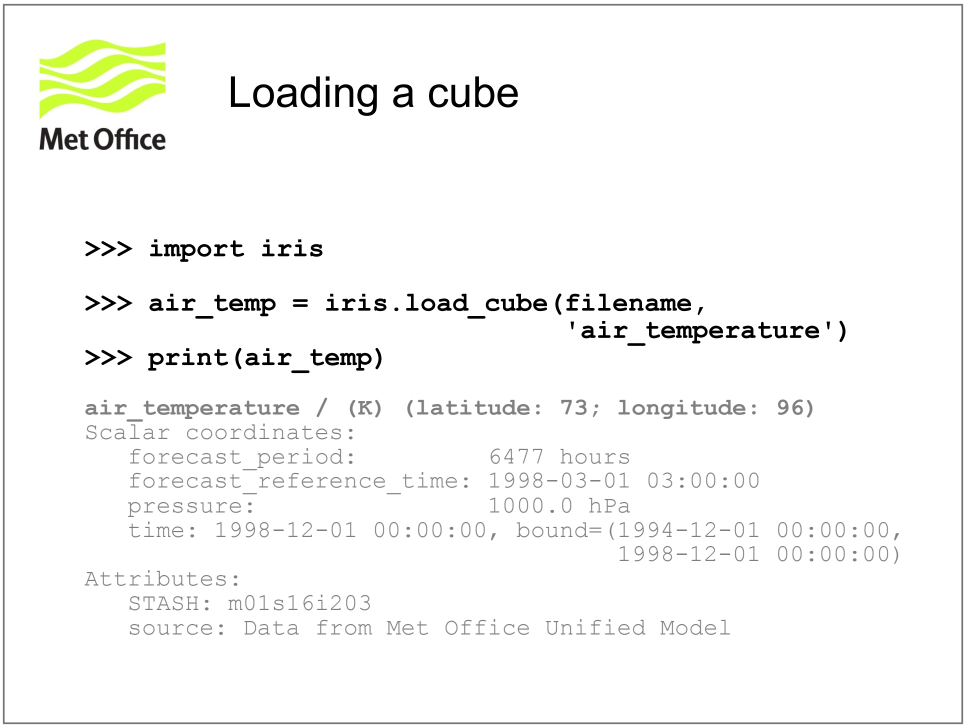
Iris’s cubes aren’t bound to specific input fileformats – provided it *could* be stored as a CF-1.5 compliant NetCDF file, it can be represented as a cube. (e.g. ORCA data)

We already have import support for commonly seen files, such as NetCDF, GRIB and UM PP and Fieldsfiles, and it is relatively simple to add new loading capabilities, or to just create a cube by hand.

Once you have a cube, there are common built-in operations, such as interpolation, aggregations, rolling windows, as well as a matplotlib plotting interface. Critically, it is easy to get access to your data, so if you want to do some novel analysis that Iris doesn't have, you can just call out to the amazing Python ecosystem.

## An example

To best demonstrate some of Iris capabilities, let's look at a simple worked example.



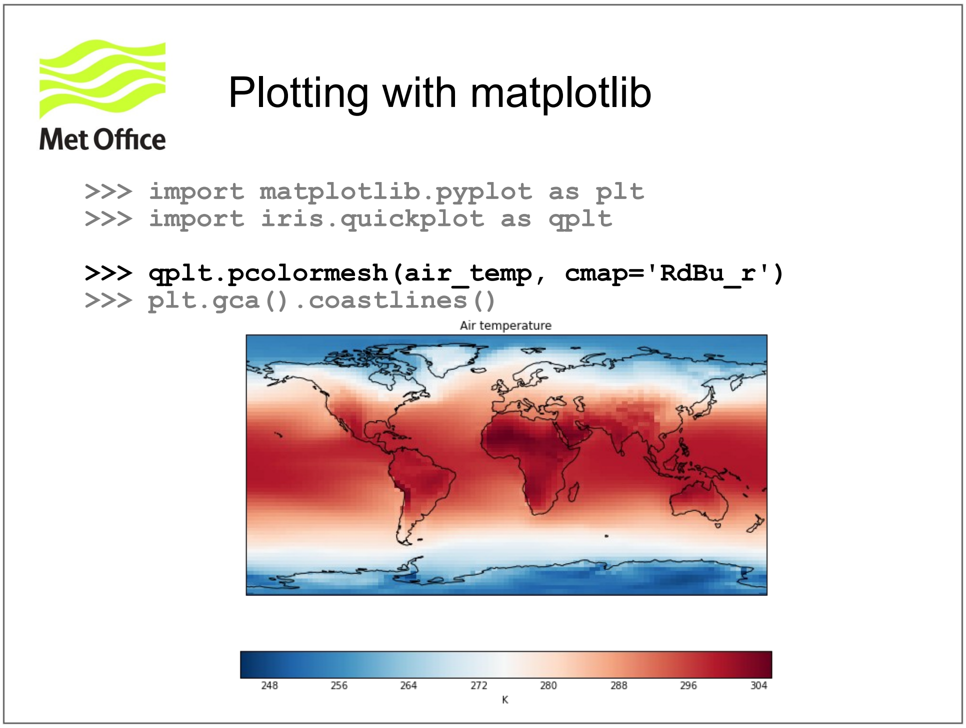
## Loading a cube

```
>>> import iris

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

air_temperature / (K) (latitude: 73; longitude: 96)
Scalar coordinates:
   forecast_period:         6477 hours
   forecast_reference_time: 1998-03-01 03:00:00
   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
```

Iris is a python package rather than a system, so we simply “import iris” in our usual python interpreter.

We can load data with Iris using the “load_cube” function. In this case we are asking of air_temperature from the file.

The string representation of a cube, as seen when printing it, is a short summary of some of the key metadata.



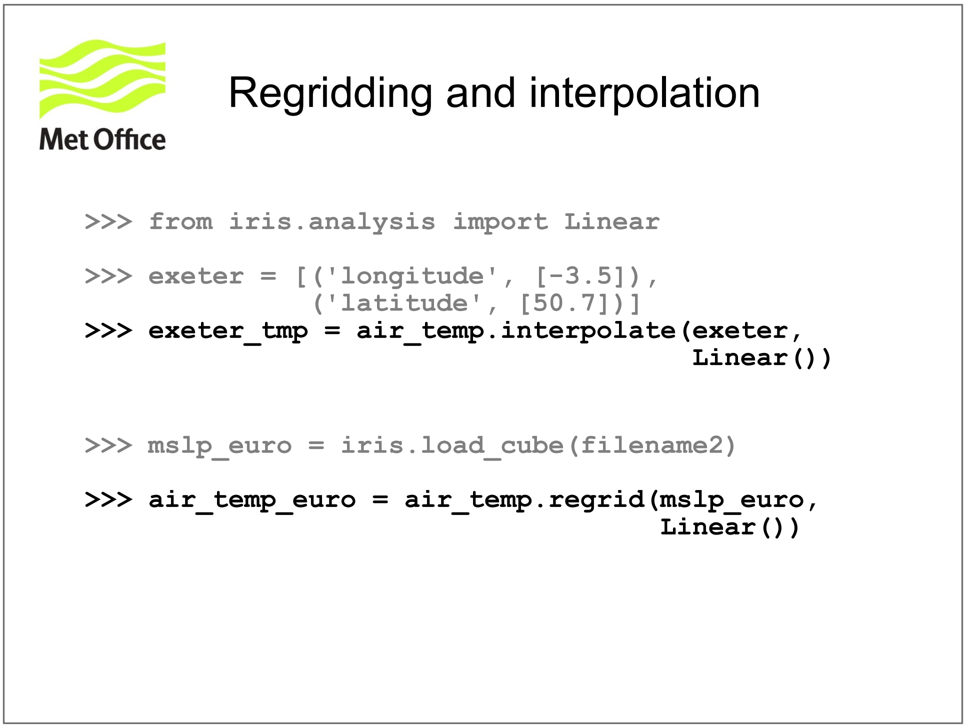
## Plotting with matplotlib

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

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

Met Office
Air temperature
248
256
264
272
280
288
296
304
K

The visualisation layer in iris is a very thin wrapper on matplotlib. Essentially, our wrappers understand cubes, and pass the appropriate information on to matplotlib to do the visualisation itself. This includes things such as the title, the x and y coordinate values as well as the coordinate system of those coordinates, and the data itself.

In this case we have used matplotlib's **pcolormesh**, which is essentially a block plot, but we could equally have asked for filled contours using the **contourf** function.



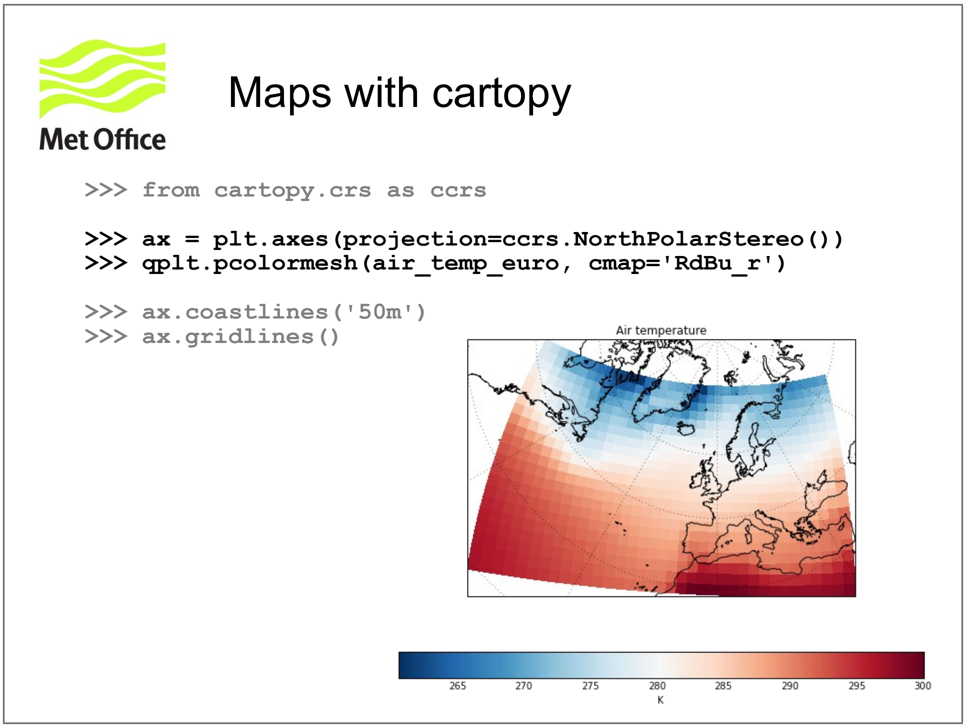
## Regridding and interpolation

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

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


>>> mslp_euro = iris.load_cube(filename2)

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

Iris has interpolation options built-in, simple interpolation to do flat-surface interpolation (point, line, plane, etc.), as well as regridding interpolation where one cube defines the grid for another to be interpolated onto.

In this case, initially we compute the temperature data at a single point.

In the other case, we regrid our temperature data onto the grid of another variable (mslp_euro – which is on a limited area model European domain with a rotated pole).



# Maps with cartopy

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

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

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

Iris' mapping capabilities come from cartopy – a spin off python package which solves the dateline and pole problems seen with traditional mapping libraries.

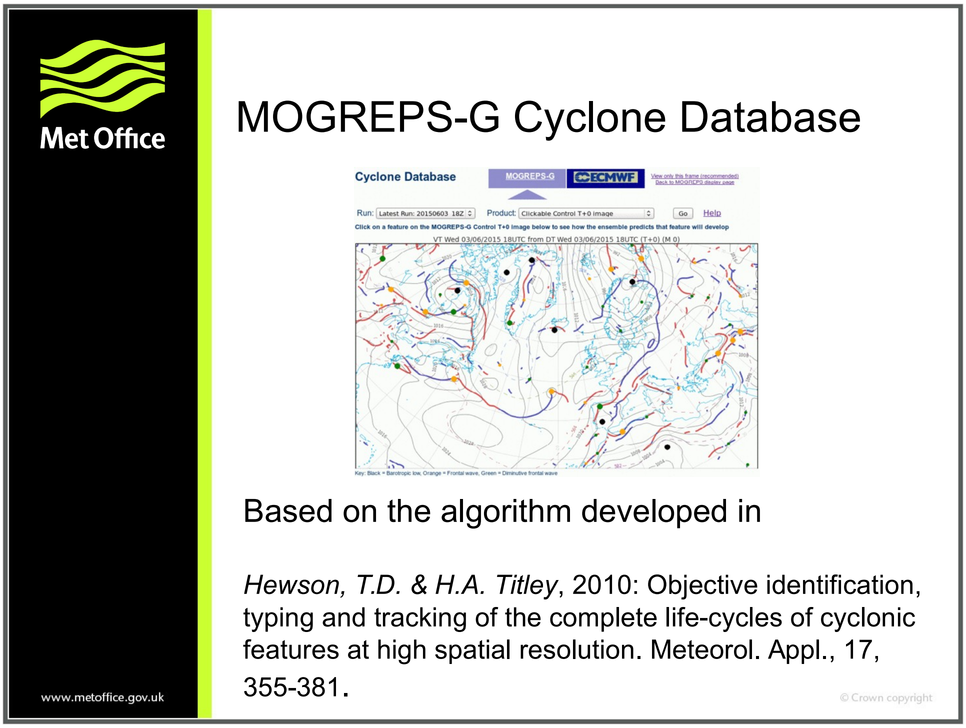
In this case, we tell matplotlib that we want to produce a north polar stereographic map, and then use Iris to put our data (which has now been regridded onto a rotated pole) on the map.

Met Office

# A real-life example

www.metoffice.gov.uk

© Crown copyright


Met Office
www.metoffice.gov.uk
MOGREPS-G
ECMWF
View only this frame (recommended)
Back to MOGREPS display page
Go
Help

# MOGREPS-G Cyclone Database

Cyclone Database

Run: Latest Run: 20150603 18Z

Product: Clickable Control T+0 image

Click on a feature on the MOGREPS-G Control T+0 image below to see how the ensemble predicts that feature will develop

VT Wed 03/06/2015 18UTC from DT Wed 03/06/2015 18UTC (T+0) (M 0)

Key: Black = Barotropic low, Orange = Frontal wave, Green = Diminutive frontal wave

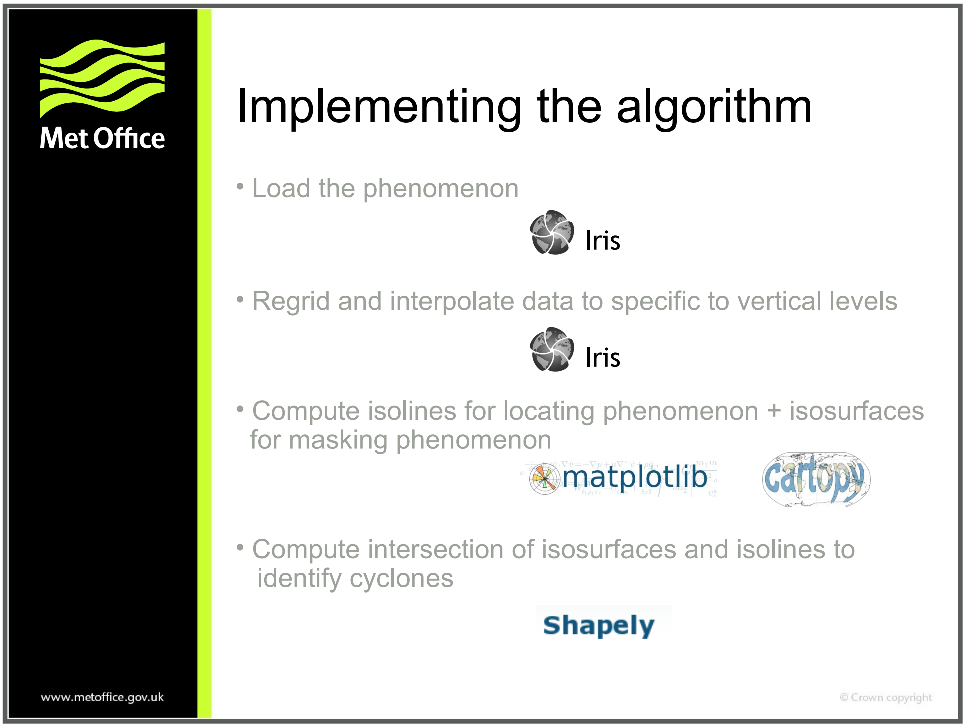
Based on the algorithm developed in

*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.

© Crown copyright

In practice, for novel product generation, we want to implement algorithms which don't belong in Iris' core.

As an example, I worked with Helen Titley to implement the “Objective identification, typing and tracking of the complete life-cycles of cyclonic features” algorithm described in this 2010 publication.



Met Office

# Implementing the algorithm

- Load the phenomenon

Iris

- Regrid and interpolate data to specific to vertical levels

Iris

- Compute isolines for locating phenomenon + isosurfaces for masking phenomenon

matplotlib cartopy

- Compute intersection of isosurfaces and isolines to identify cyclones

Shapely

www.metoffice.gov.uk

© Crown copyright

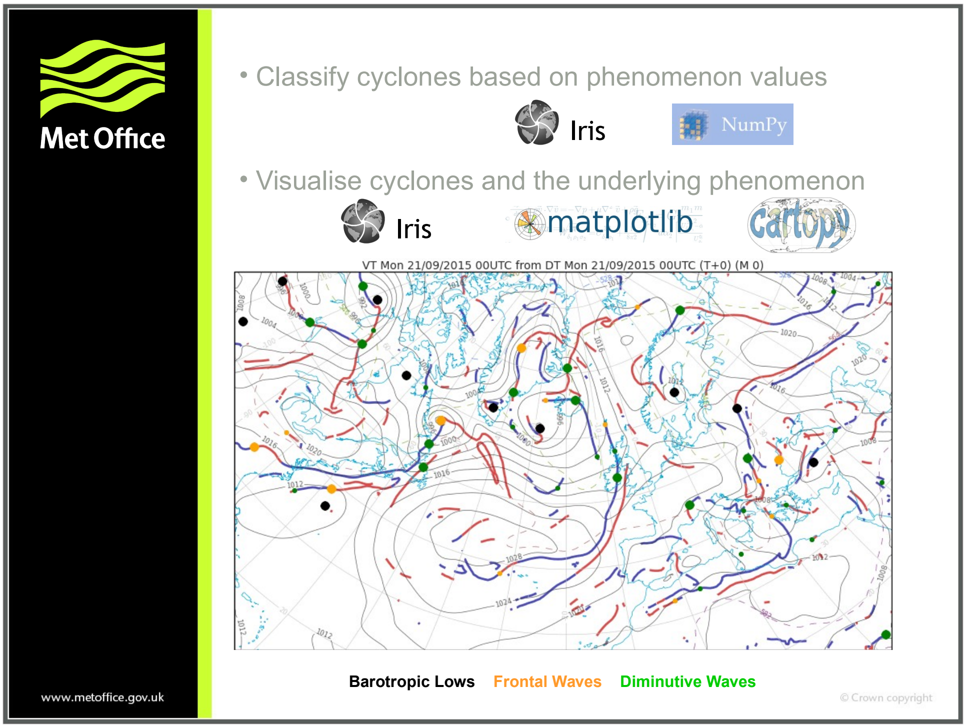
The algorithm has several steps, each of which benefited from having powerful tools already available within Python:

Roughly, we start by loading the algorithm inputs straight from model output.

Next we put the diagnostics onto the desired grid and vertical levels.

We then used matplotlib in conjunction with cartopy to compute geolocated isolines and isosurfaces.

Using shapely, we compute the intersection of these surfaces to identify fronts, and cyclonic centers.



Met Office

- Classify cyclones based on phenomenon values

Iris NumPy

- Visualise cyclones and the underlying phenomenon

Iris matplotlib cartopy

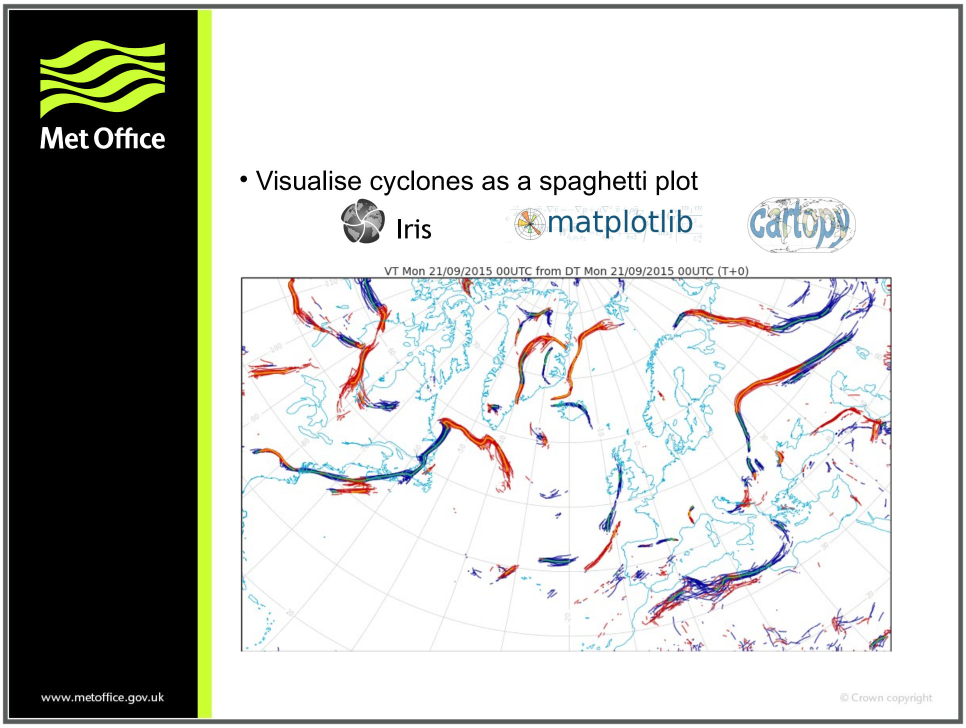
VT Mon 21/09/2015 00UTC from DT Mon 21/09/2015 00UTC (T+0) (M 0)
Barotropic Lows
Frontal Waves
Diminutive Waves

www.metoffice.gov.uk

© Crown copyright

Using Iris, we then interpolate the diagnostics for these locations, to classify warm/cold fronts, and the type of cyclonic feature identified.

Finally, we visualise the fronts and cyclonic features, along with some of the diagnosics, to produce a web based animation. On the web page, each of the cyclonic features are clickable to give detailed diagnostic information and trends.



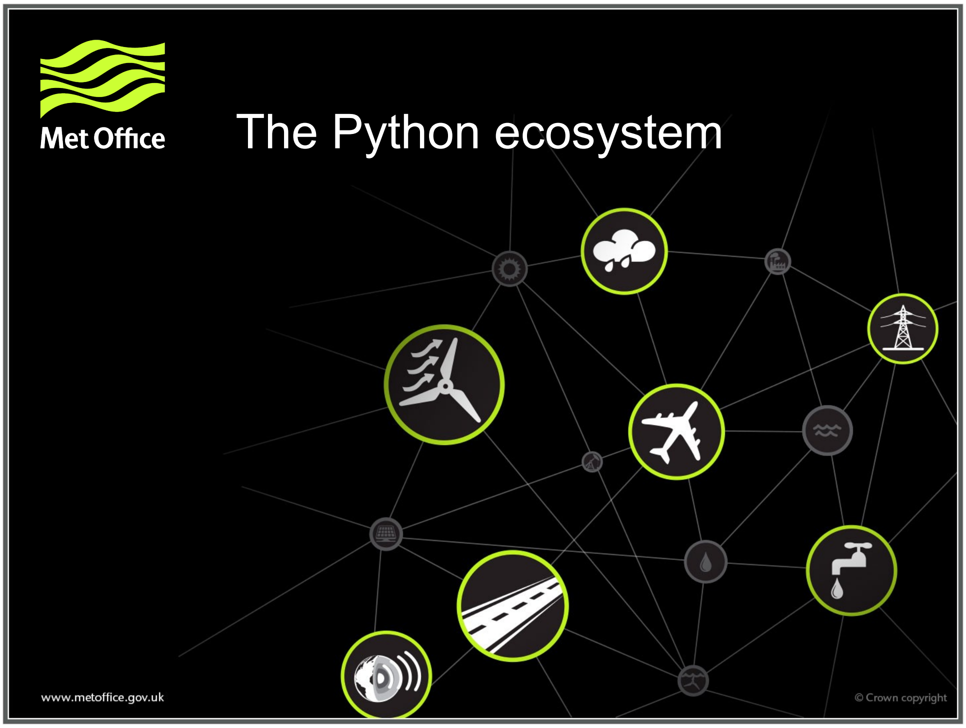
Met Office

- Visualise cyclones as a spaghetti plot

www.metoffice.gov.uk

© Crown copyright

Because we had saved each of the geometries of the identified fronts, we were also able to compose the fronts into a multi-ensemble animation, which gives a nice indication of the spread of frontal development.



Met Office

## The Python ecosystem

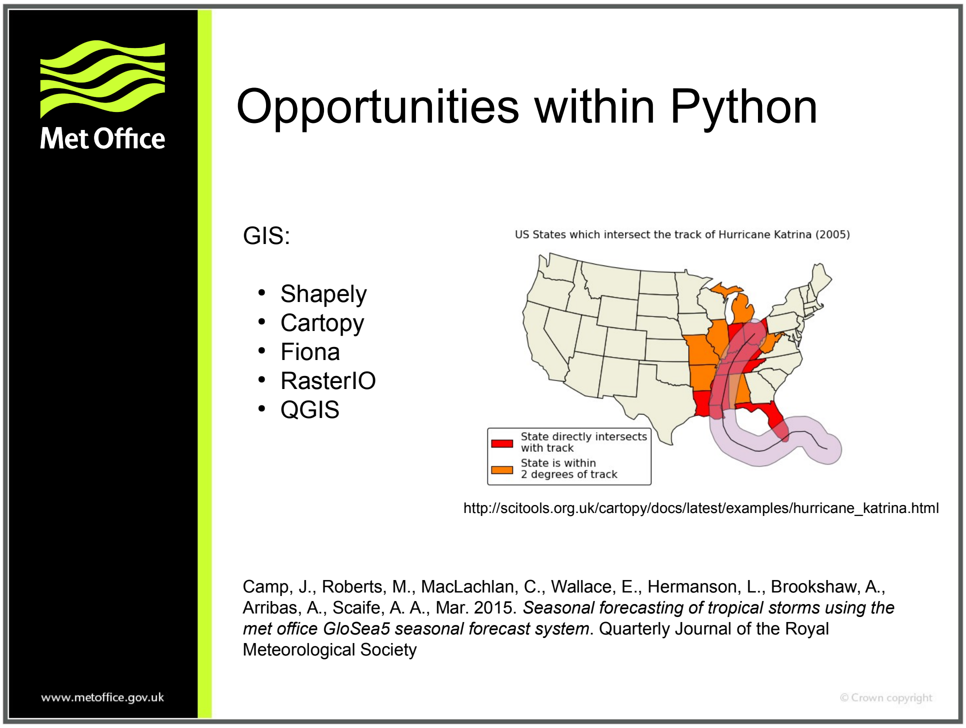
www.metoffice.gov.uk

© Crown copyright

The most powerful part of Iris is the communities it plugs into.

Beyond the Iris basics there is a whole world of cross-domain packages which can be leveraged.

I will highlight just a few of these opportunities which represent possibilities in delivering novel science, or simply more effective calculation of ever increasing data payloads.



Met Office

## Opportunities within Python

GIS:

- Shapely
- Cartopy
- Fiona
- RasterIO
- QGIS

US States which intersect the track of Hurricane Katrina (2005)

🟥 State directly intersects with track
🟧 State is within 2 degrees of track

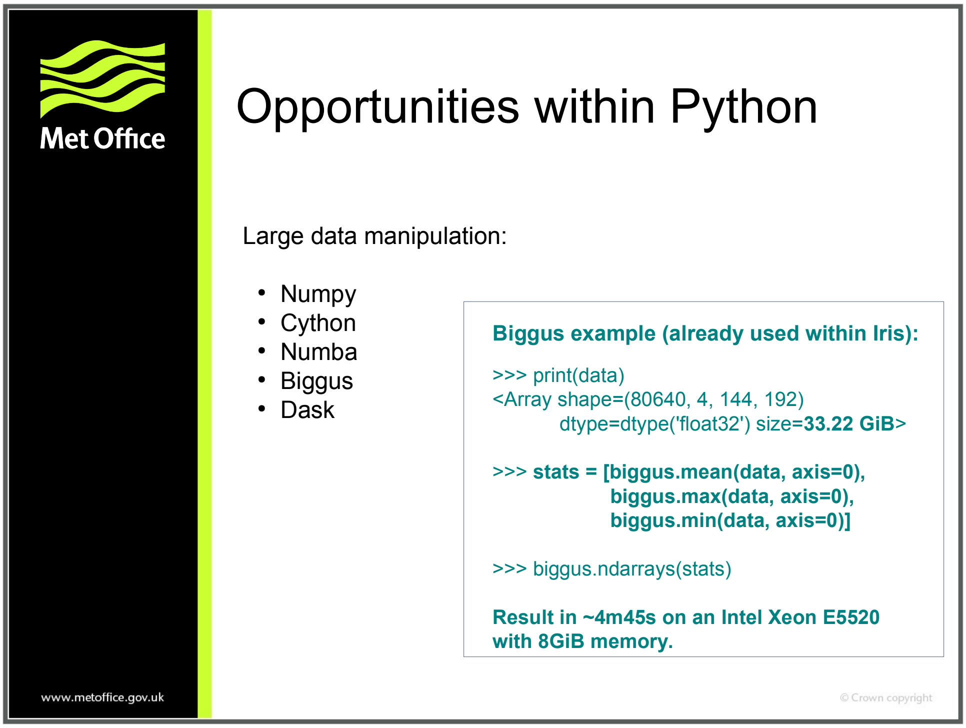
http://scitools.org.uk/cartopy/docs/latest/examples/hurricane_katrina.html

Camp, J., Roberts, M., MacLachlan, C., Wallace, E., Hermanson, L., Brookshaw, A., Arribas, A., Scaife, A. A., Mar. 2015. *Seasonal forecasting of tropical storms using the met office GloSea5 seasonal forecast system*. Quarterly Journal of the Royal Meteorological Society

© Crown copyright

The geographic information system (GIS) world has long been a place for scientists skilled with GIS (desktop) tools to do novel temporal and spatial analyses, but the evolution of these algorithms into user-friendly Python packages has opened up a whole field of opportunity.

This toy example from the cartopy gallery highlights the US states which were within a distance from the track of hurricane Katrina. Although this is just a toy example, a similar technique is being used by Joanne Camp et al. to produce a seasonal forecast for land-falling Hurricane frequencies.



Met Office

# Opportunities within Python

Large data manipulation:

- Numpy
- Cython
- Numba
- Biggus
- Dask

**Biggus example (already used within Iris):**

```
>>> 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.**

www.metoffice.gov.uk

© Crown copyright

With ever increasing HPC comes ever increasing model output.

Numpy has long been the array language of choice within Python, however as a languague Python remains relatively slow - recently though, there has been work to improve the performance of Python in tight for-loop optimisation, Cython and Numba allow optimisations, such as static typing, to give C-like performance in certain situations.

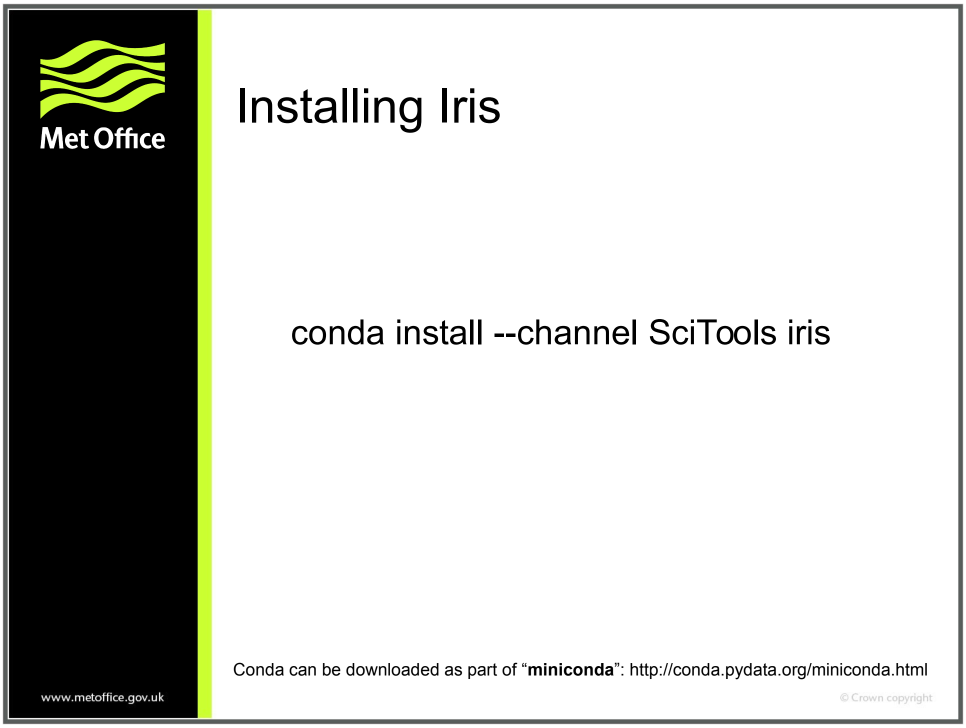
Increasingly there is work going on to extend the numpy interface to deal with arrays beyond that which can realistically be fit into memory. Biggus and dask share a common goal of delivering “out-of-core” computation to stream through data quickly and efficiently.

The following example shows a biggus array of 32GiB, over which we want to compute the mean, max and min on the first dimension (time in this case). It is easy to imagine the algorithm to compute these values without ever needing to look at all of the data in one go. The computation of these values is not only achievable, it is **fast and lean on memory**, with the overall execution time being overwhelmingly slowed by disk I/O rather than CPU.

Iris makes use of biggus under the hood, allowing cubes to be loaded and processed, which are far beyond what is allocatable in memory.

We are IO bound, as is shown by cat-ing the file to /dev/null

```
$> time cat the_big_file > /dev/null
4m19.61s
```



# Installing Iris

```
conda install --channel SciTools iris
```

Conda can be downloaded as part of “**miniconda**”: http://conda.pydata.org/miniconda.html

www.metoffice.gov.uk

© Crown copyright

Iris is reasonably well established in the community, and although traditionally it was difficult to install with all of its dependencies, the conda package manger has made the installation trivial on all major platforms (Linux, OSX, Windows).

# Questions

**Links from presentation:**
github.com/pelson/ecmwf-vis-2015

Github: github.com/pelson
Twitter: @pypelson

www.metoffice.gov.uk

© Crown copyright