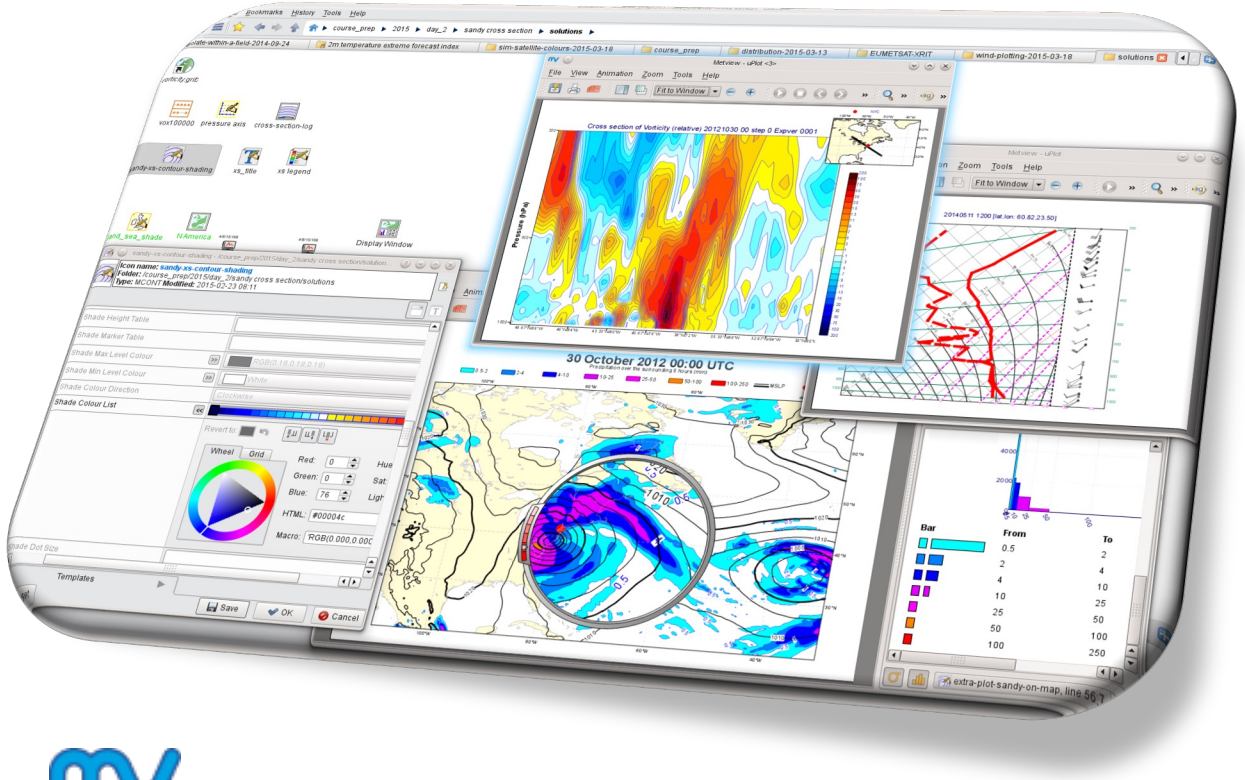
Metview Introduction

ECMWF

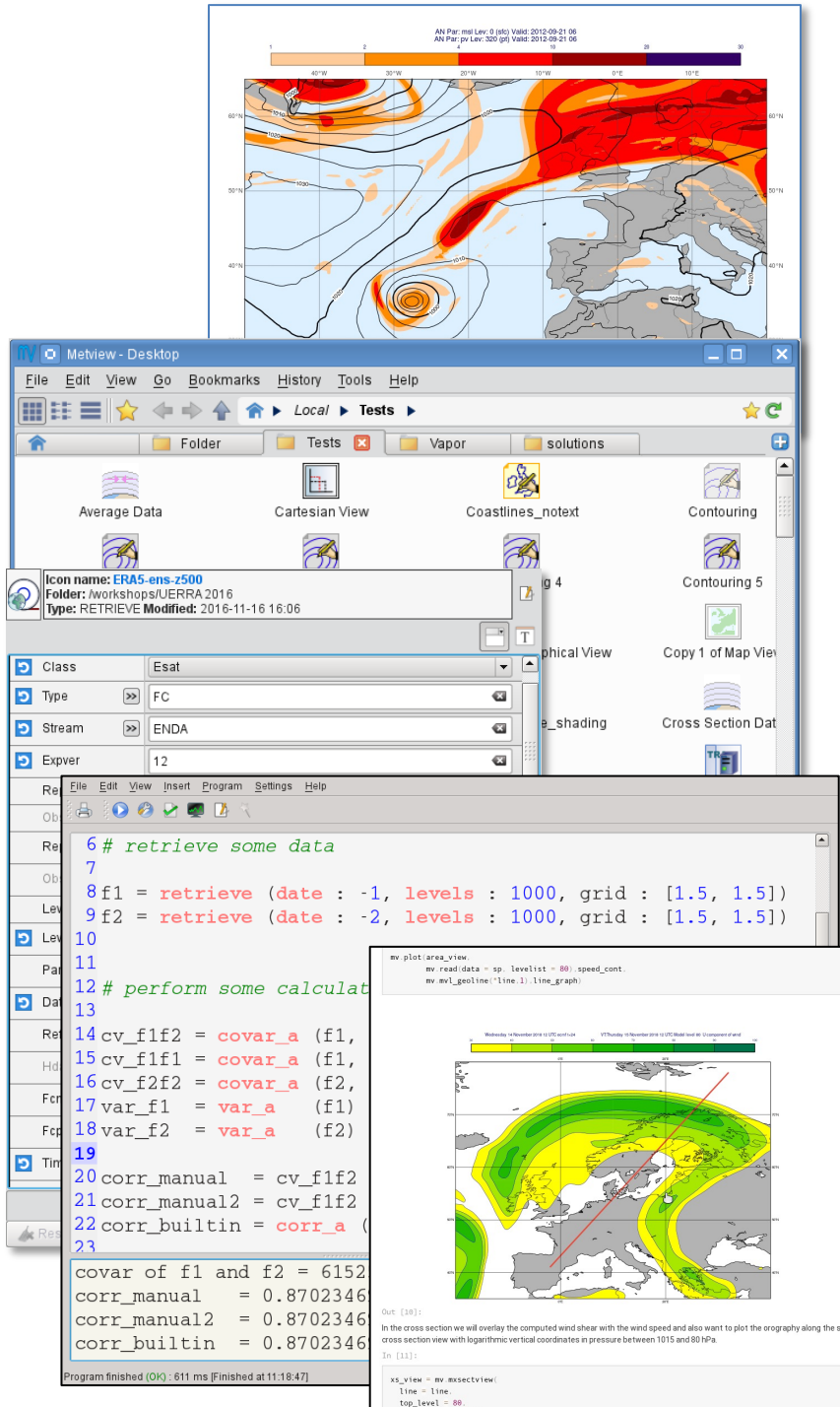
Iain Russell,
Sándor Kertész

Development Section, ECMWF



What is Metview?

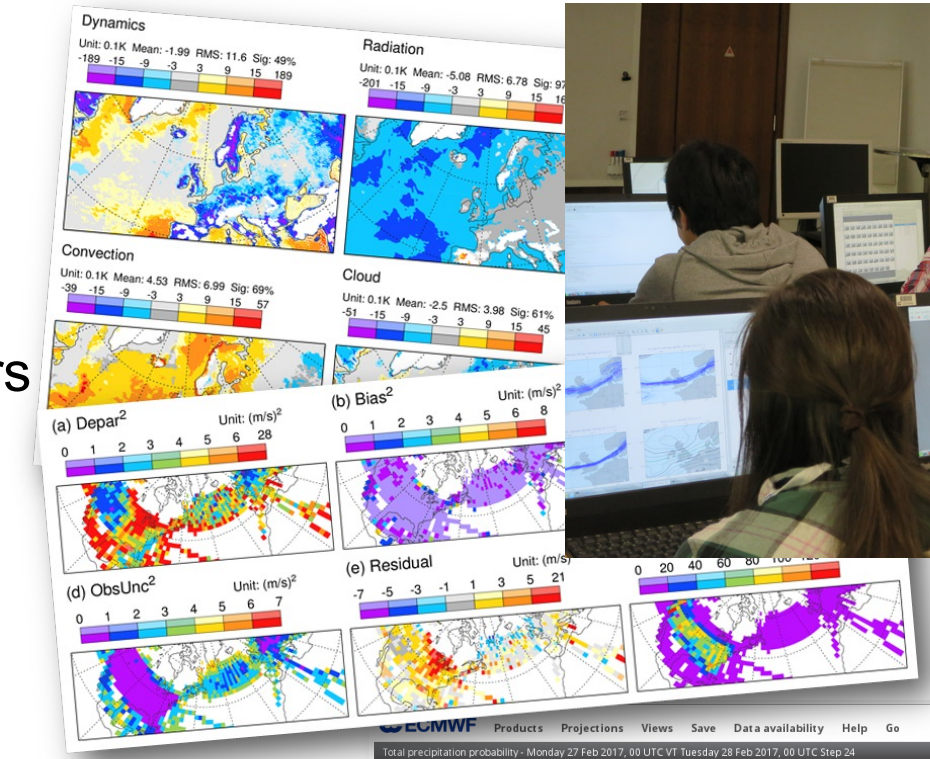
- Workstation software, runs on UNIX, from laptops to supercomputers (including macOS)
- Developed at ECMWF, built on other ECMWF libraries
- Open source, Apache 2.0 license
- Data access
- Data processing
- Data visualisation
- Icon based user interface
- Powerful scripting languages - Macro and **Python**
- Metview is a co-operation project with INPE (Brazil)



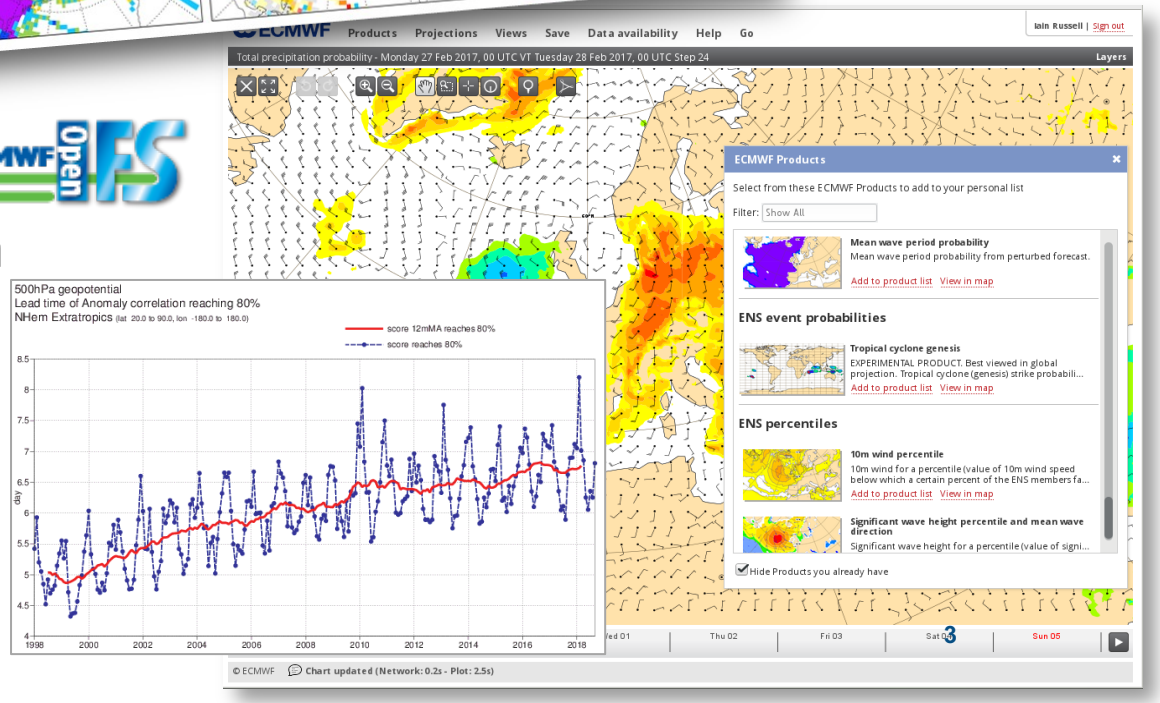
30 years of Metview so far

- Serving users of ECMWF data since 1993

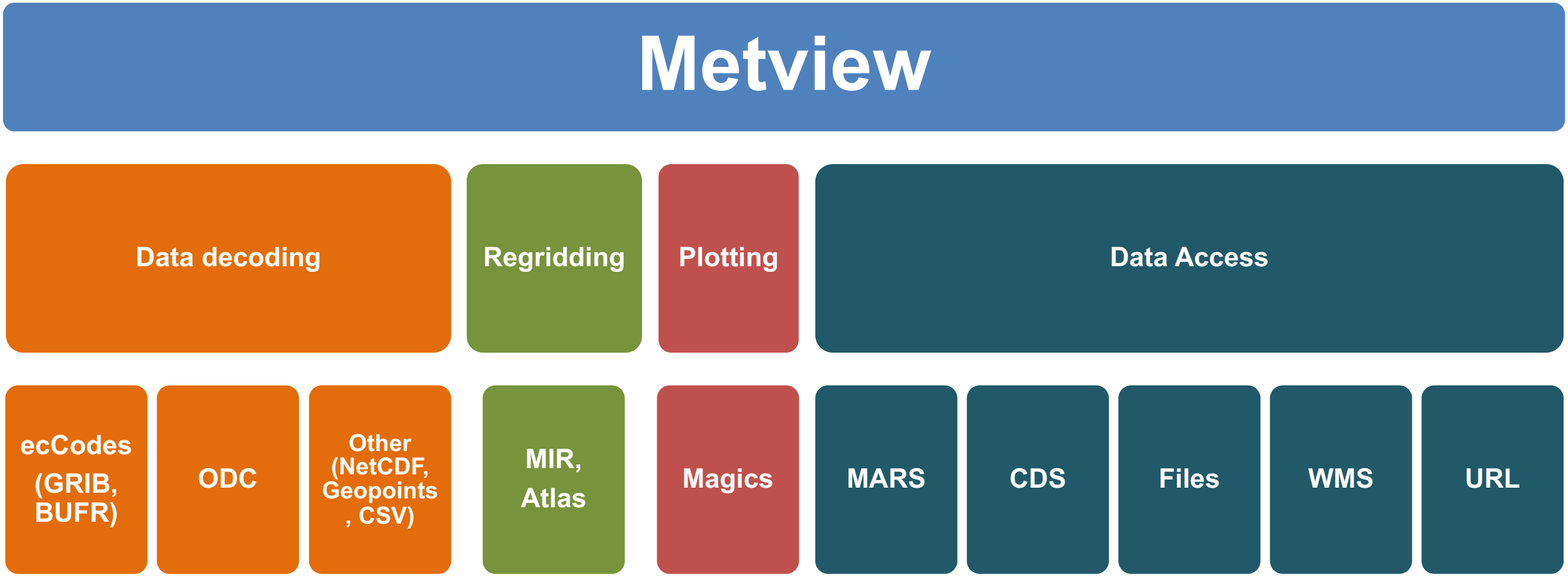
- Used daily by many analysts and researchers
  - inside and outside ECMWF
  - also by commercial users of our data



- Some large developments, e.g. the Diagnostics Toolbox, OpenIFS workshops, Quaver (verification package) are based on top of Metview
- ecCharts is based on Metview's architecture and takes it onto the web



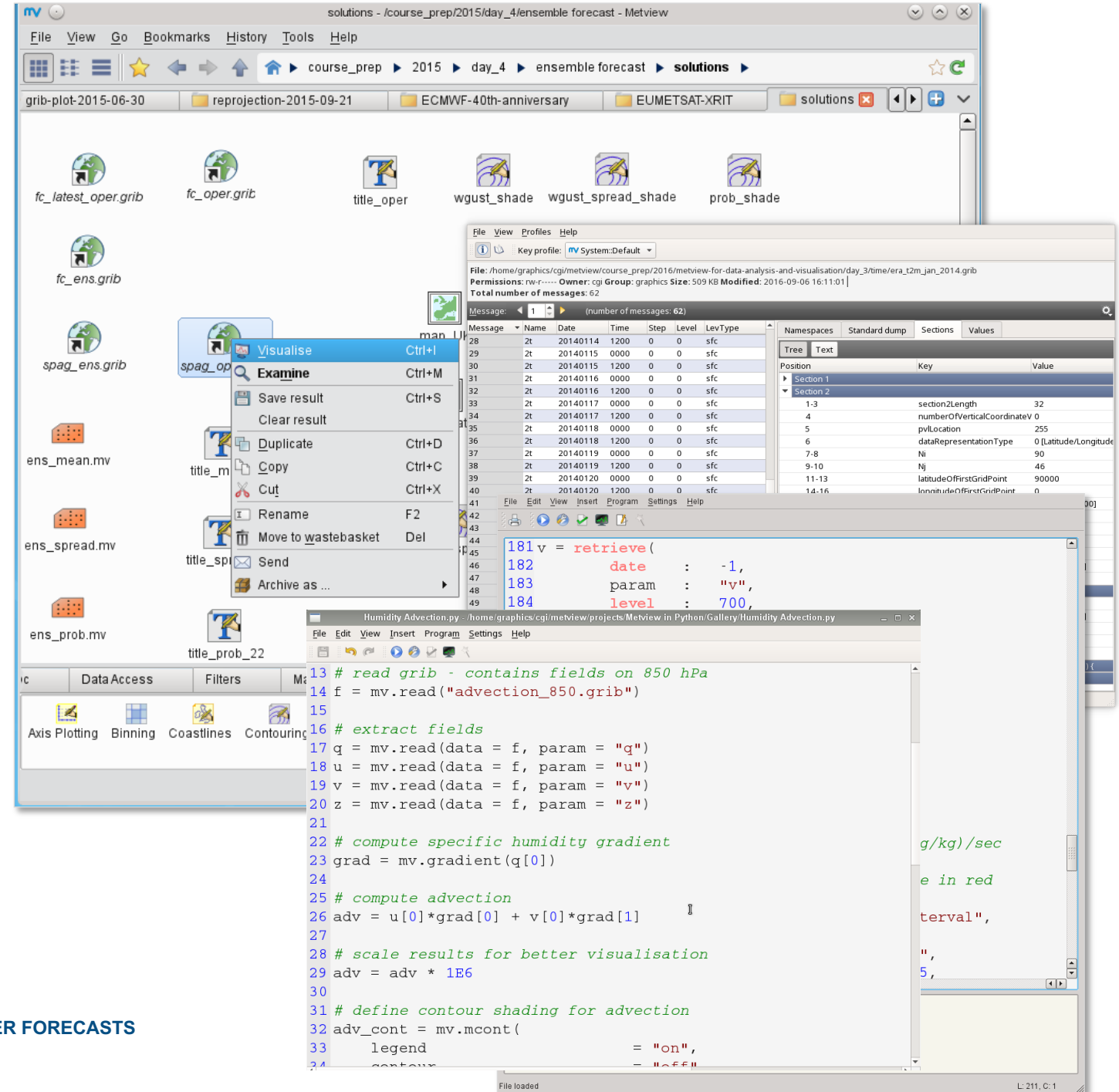
Built on top of ECMWF software packages



Using Metview

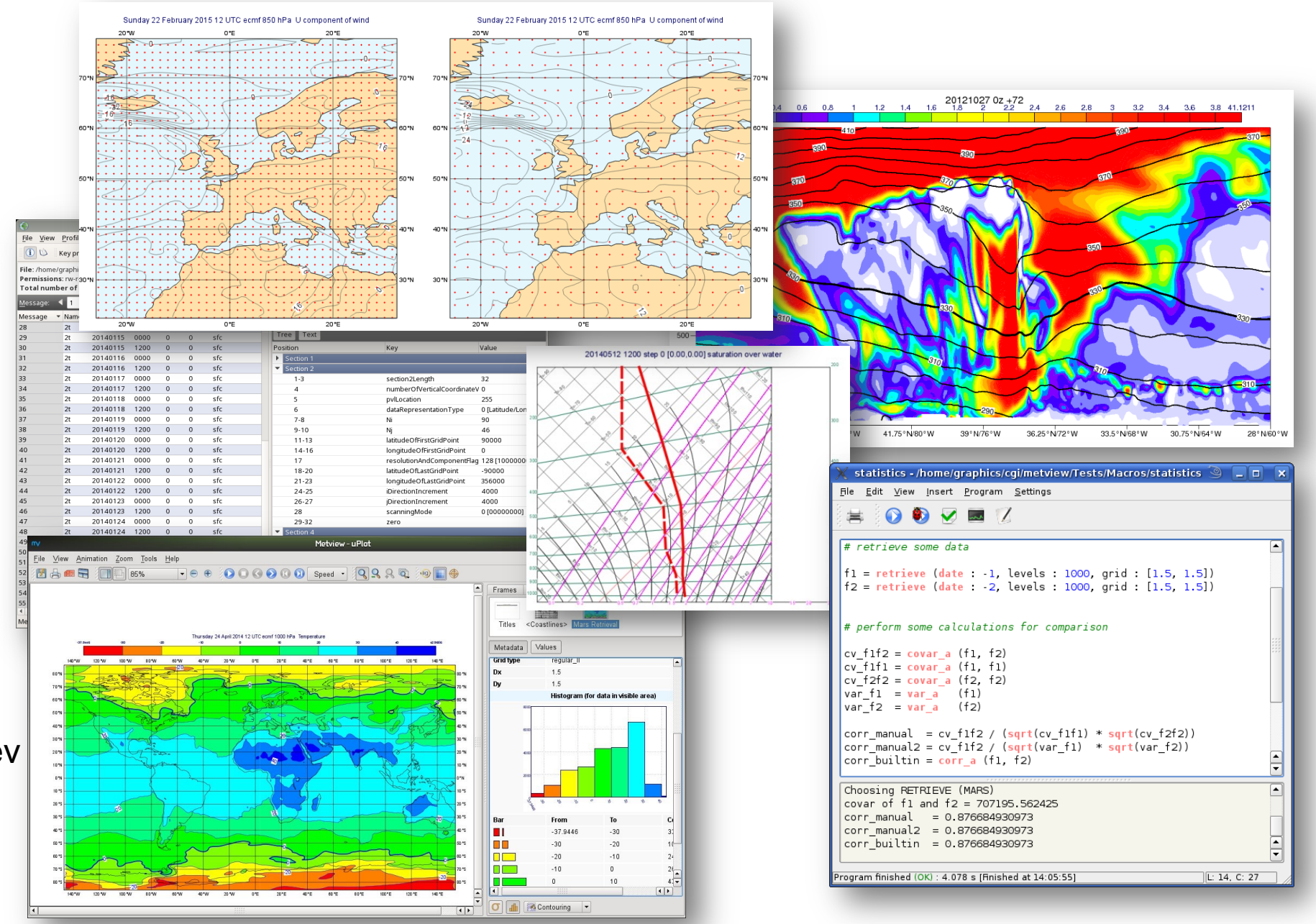
- Icon-based user interface
  - interactive investigation of data
  - icons represent data, settings and processes
  - icons can be chained together – output from one is input to another
    - e.g. filter fields from a certain date, then pass that to the Cross Section icon

- Powerful Python interface
  - more serious computations
  - batch or interactive usage
  - can generate code from UI icons
  - based on its predecessor, Metview Macro



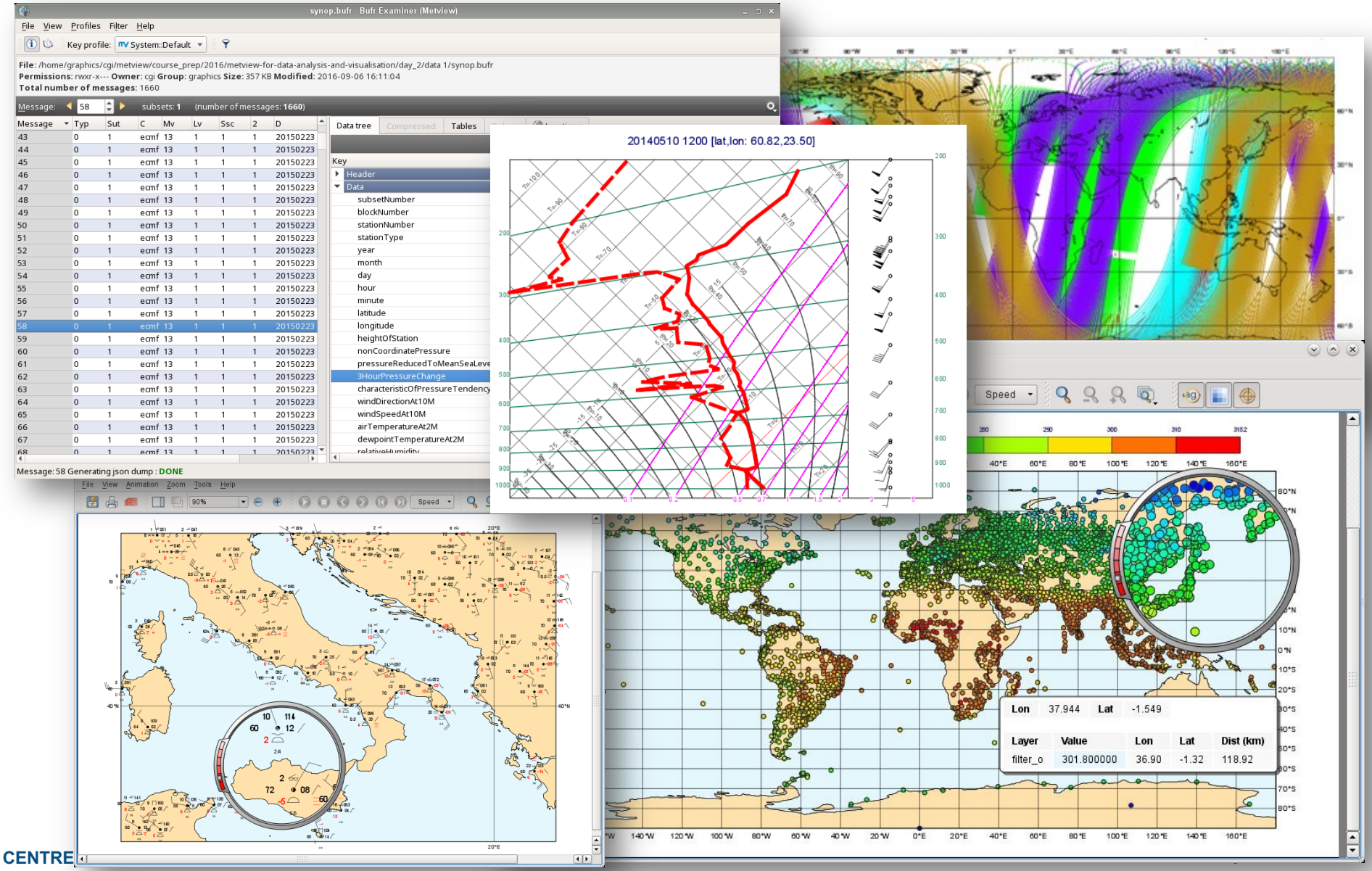
Metview + GRIB

- Plot
- Examine
- Filter, regrid
- Crop, mask
- Maths, Boolean
- Specialised:
  - Cross section
  - Thermodynamics
  - Gradient
  - Vertical integration
  - Model to pressure lev
  - Etc



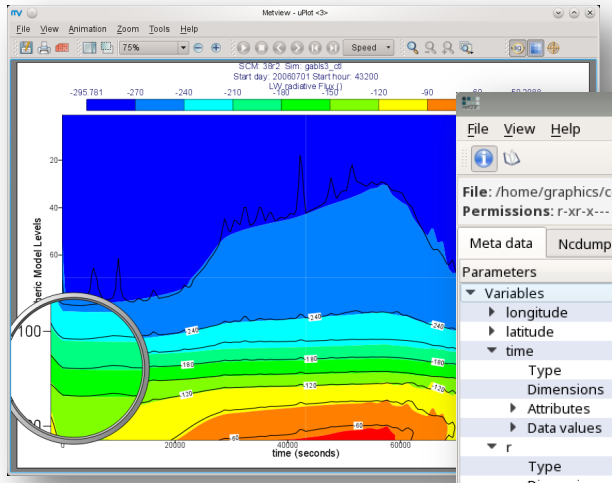
Metview + BUFR

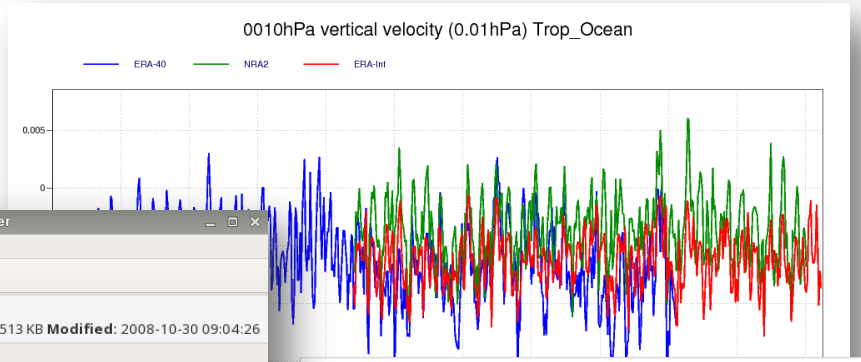
- Plot
- Examine
- Filter
- Extract values
- Convert to Geopoints
- Thermodynamics

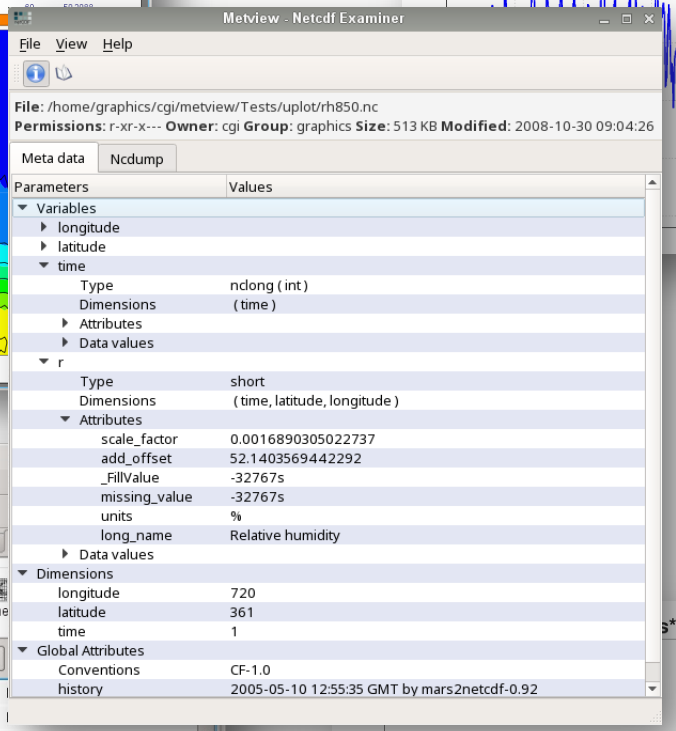


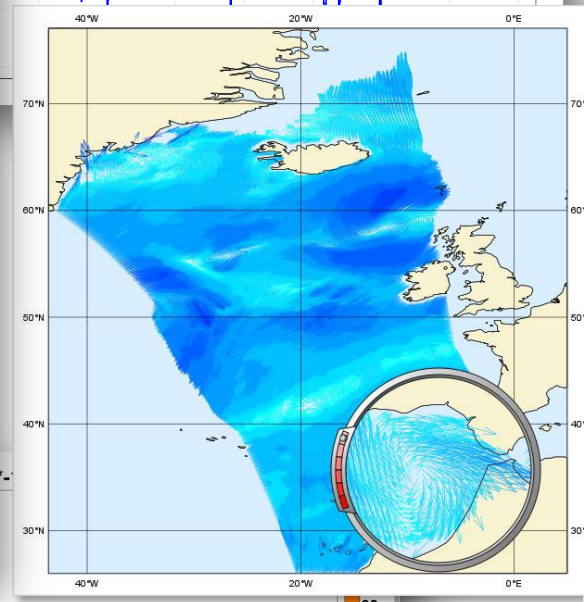
Metview + NetCDF

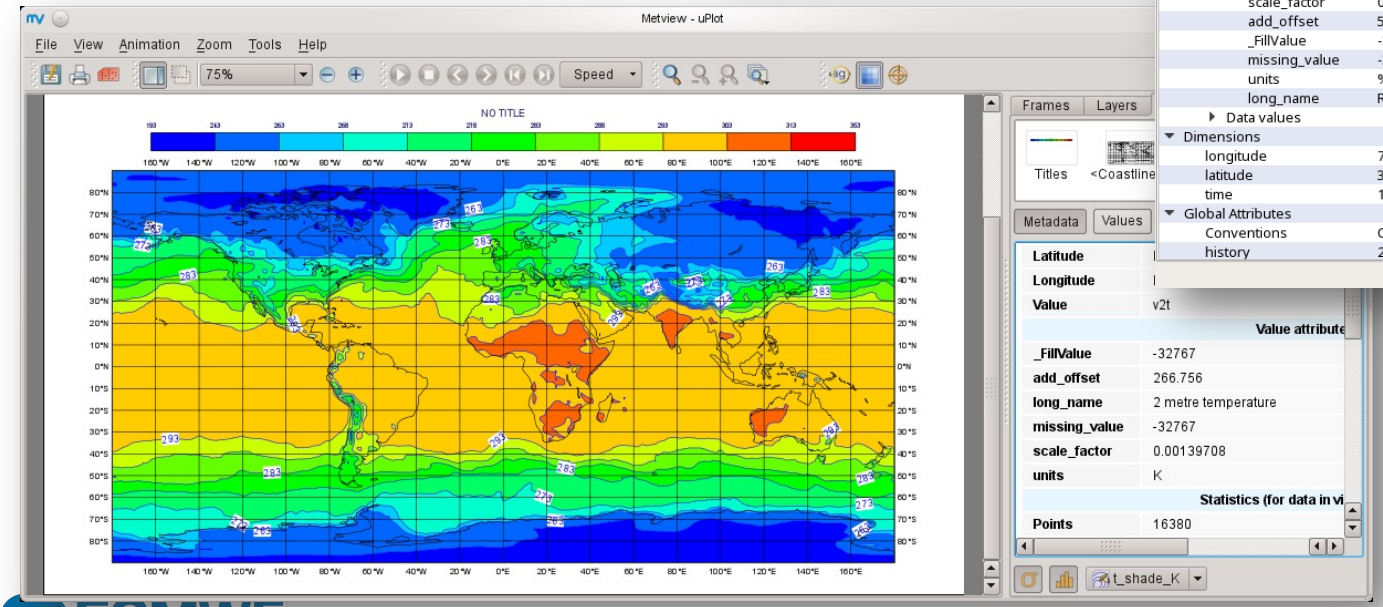
- Plot
- Examine
- Maths, Boolean

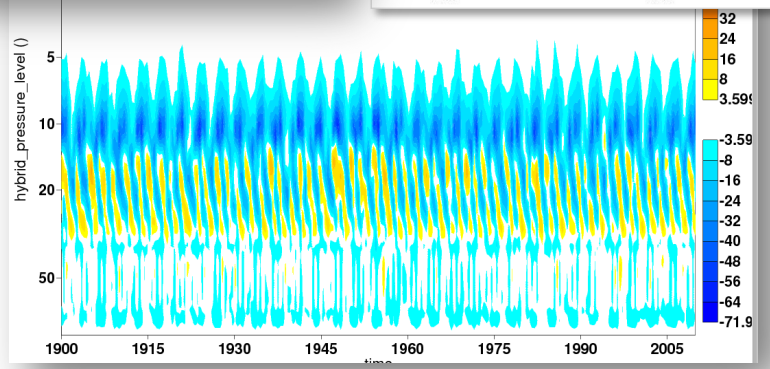






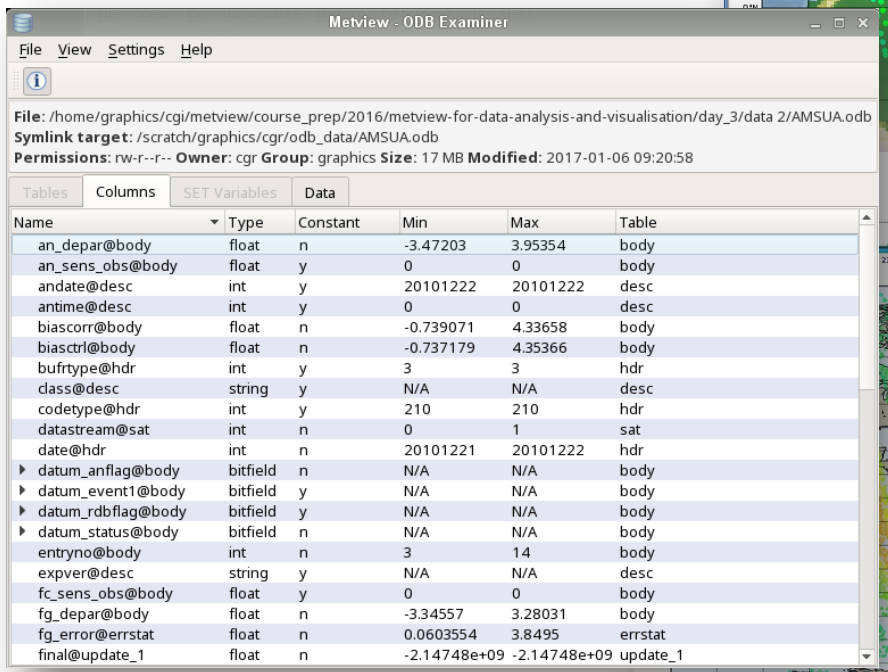


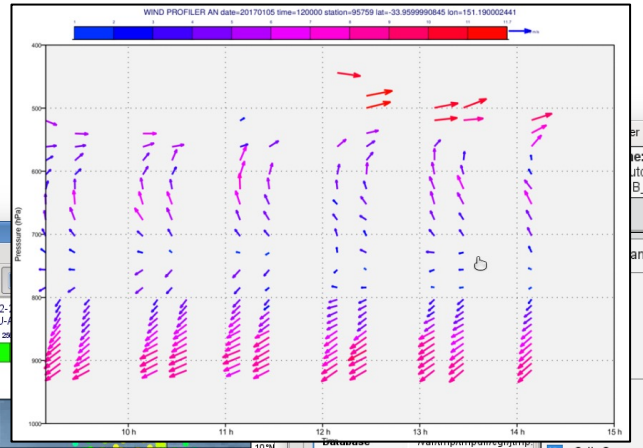


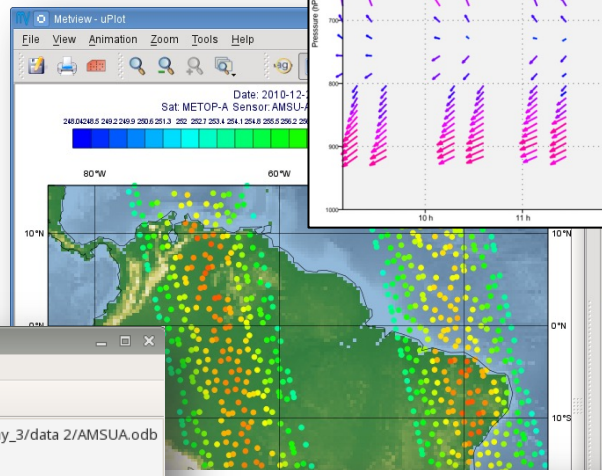


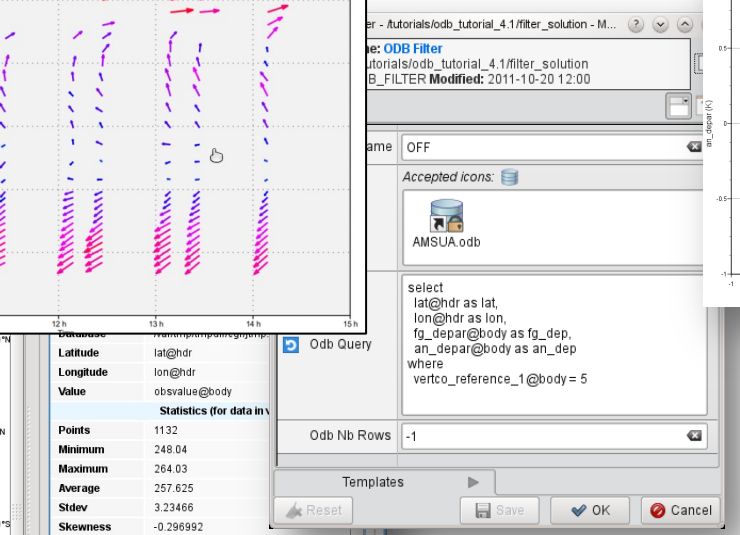
Metview + ODB

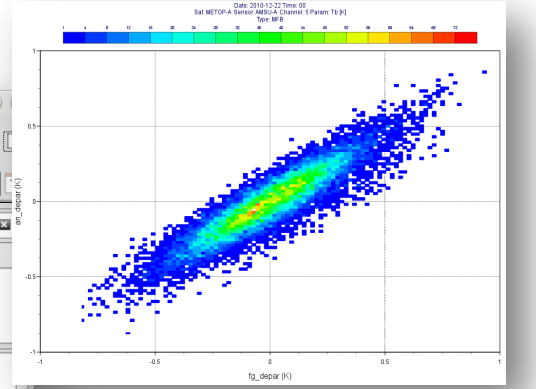
- Plot
- Examine
- Filter
- Convert to Geopoints

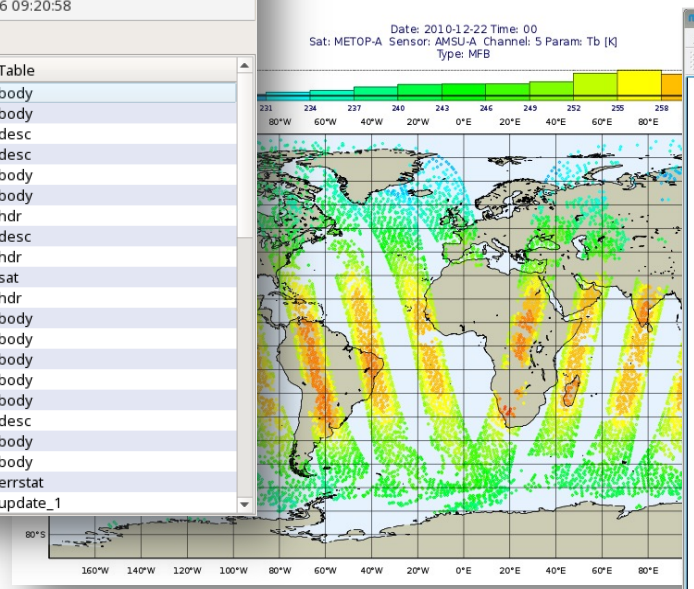


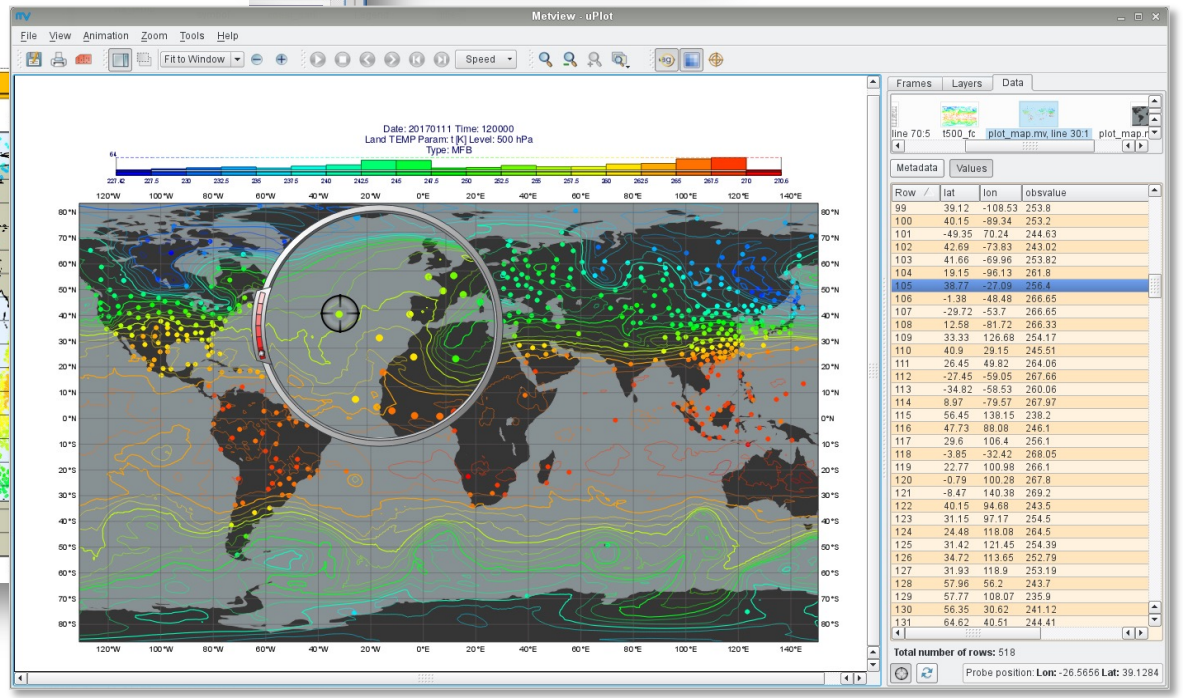








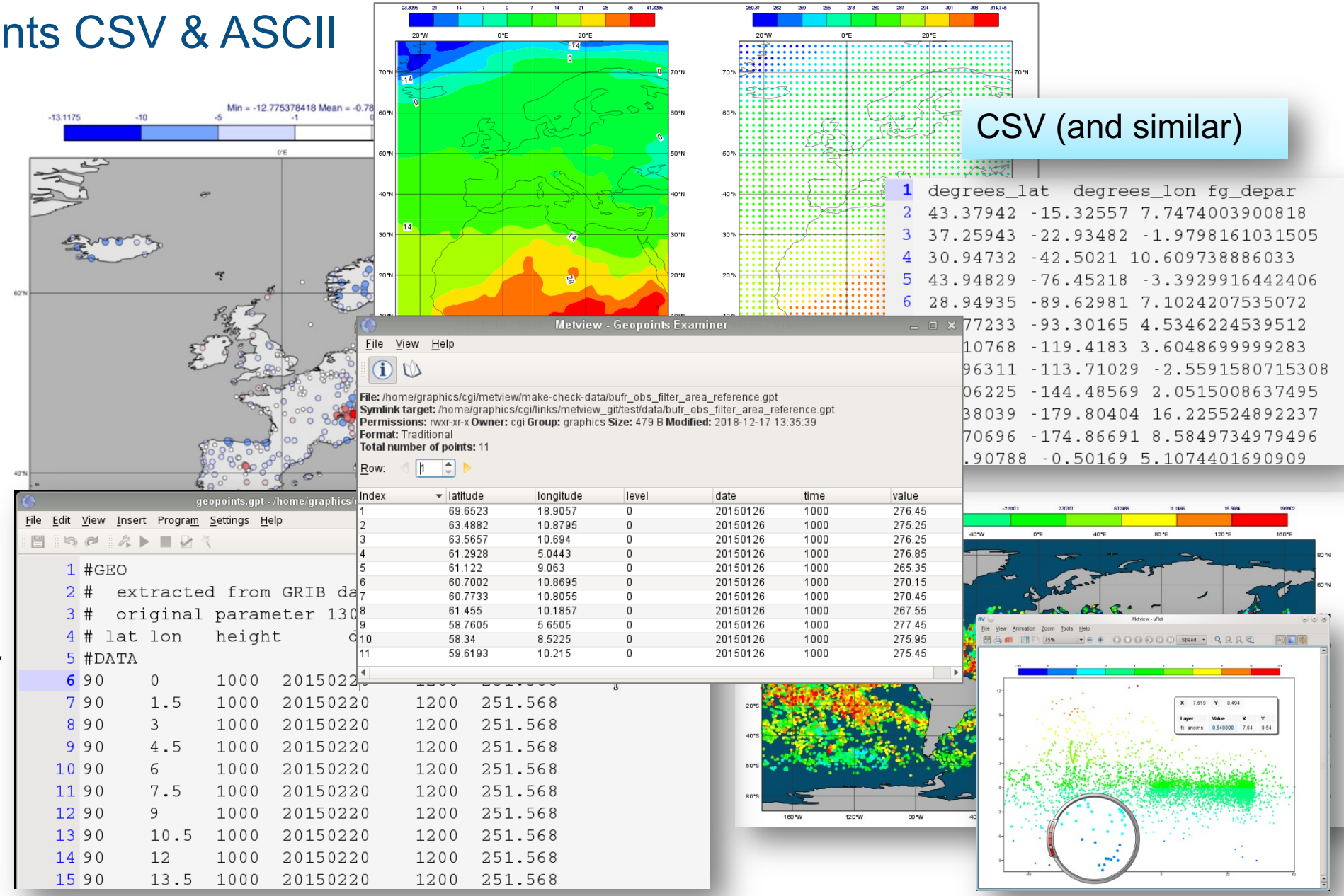




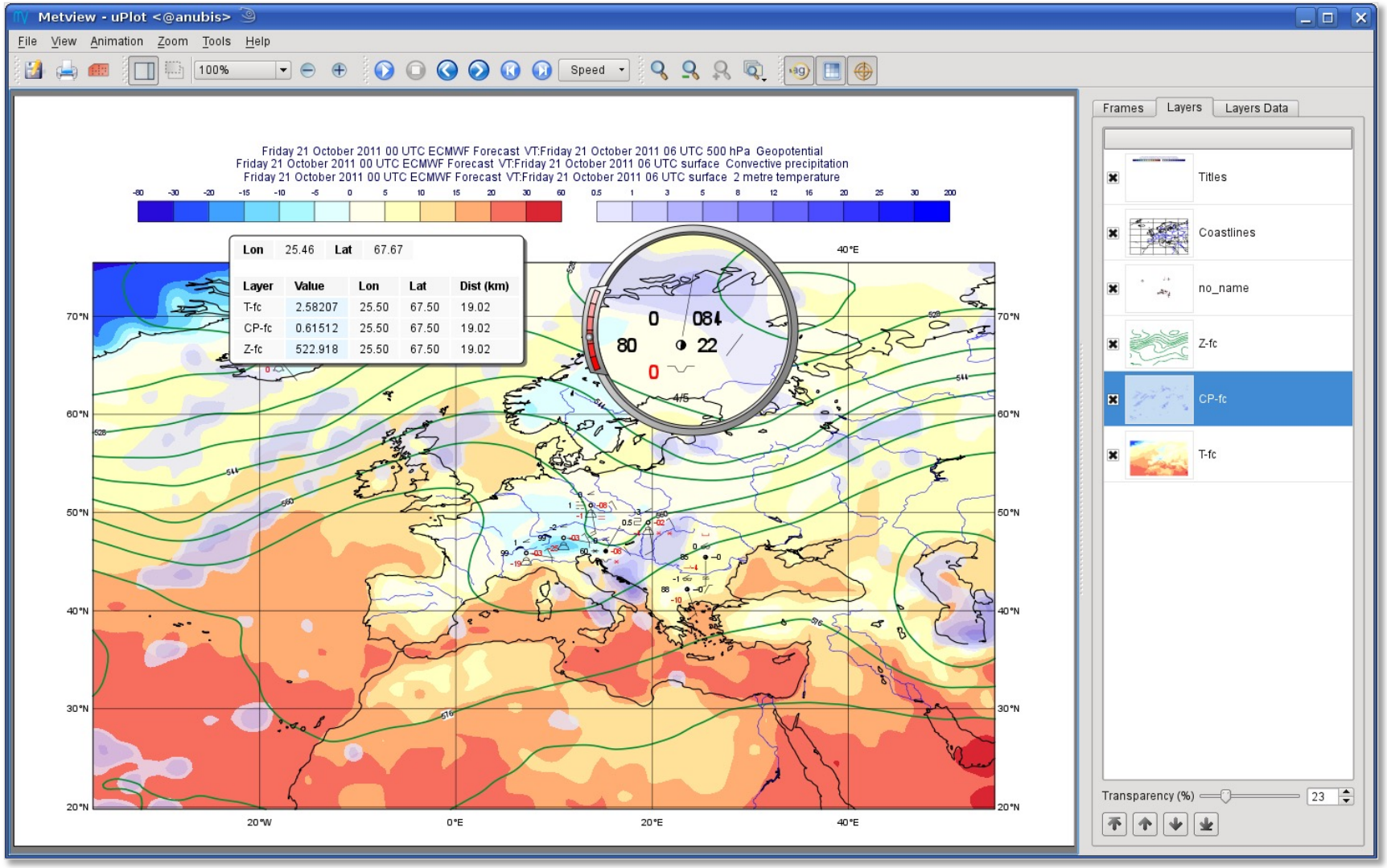
ECMWF EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS

Metview + Geopoints CSV & ASCII

- Geopoints – geo-located values
- Plot
- Examine
- Filter
- Maths, Boolean
- Geo functions
- Convert between GRIB, BUFR and Geopoints
- Can also read CSV

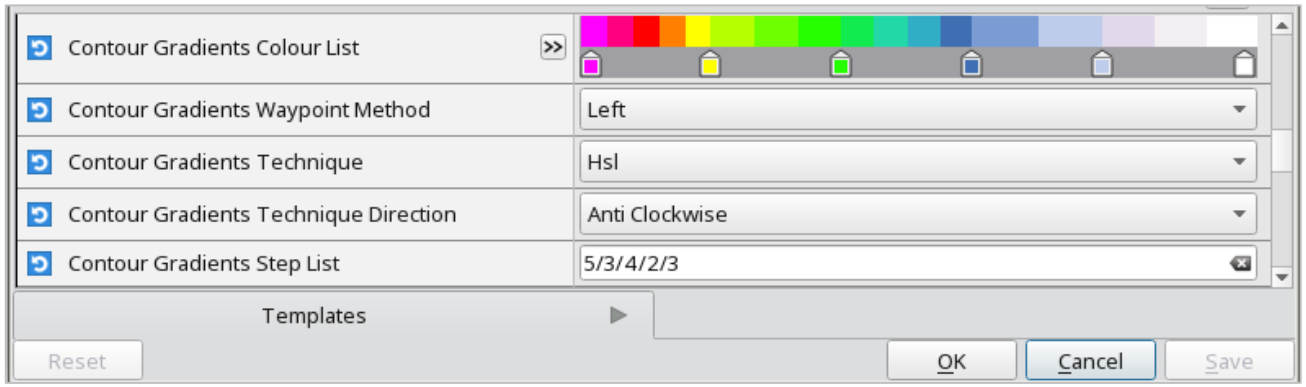


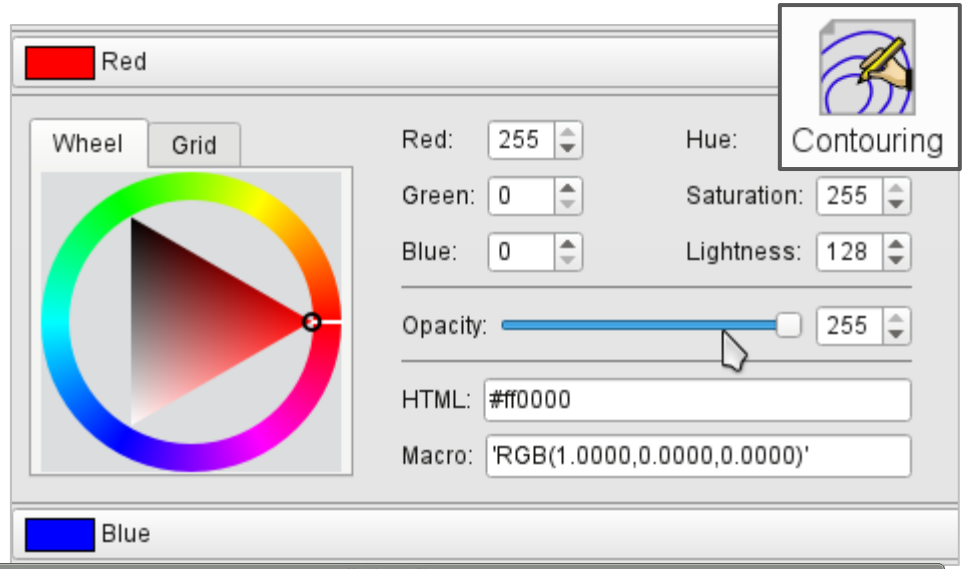
Visualisation - Overlay

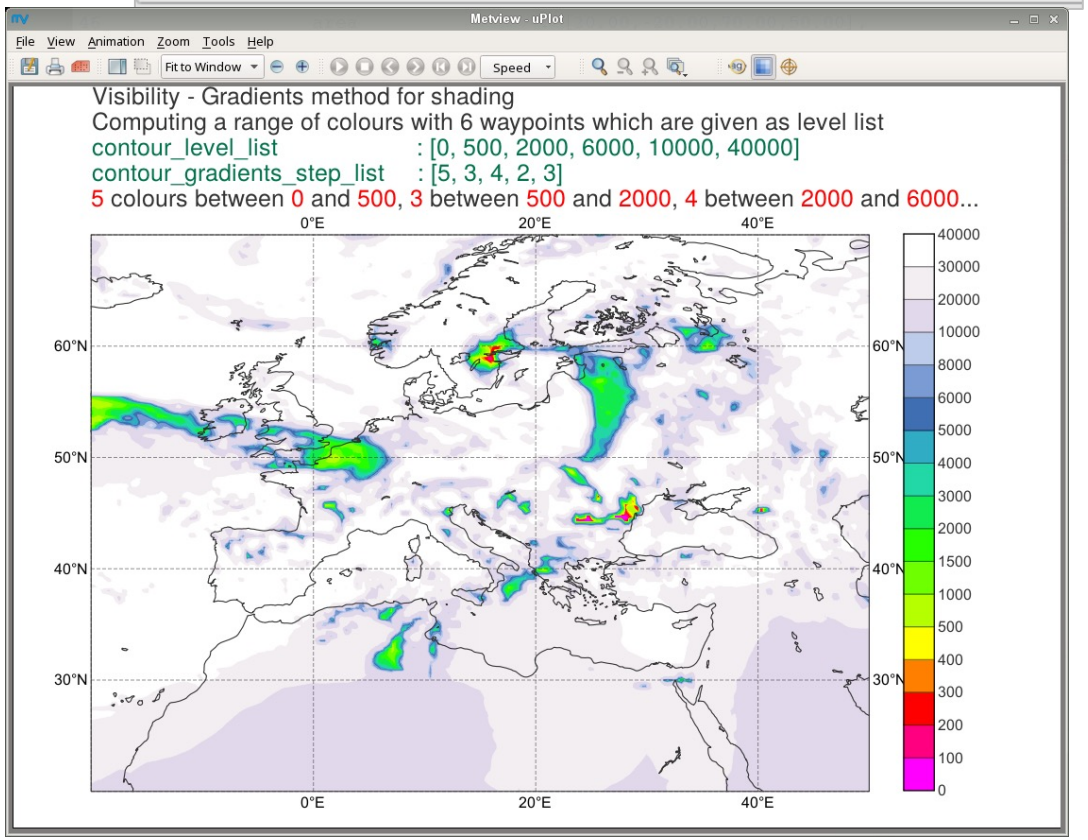


Contouring schemes

- Plenty of options for complete customisation of palettes

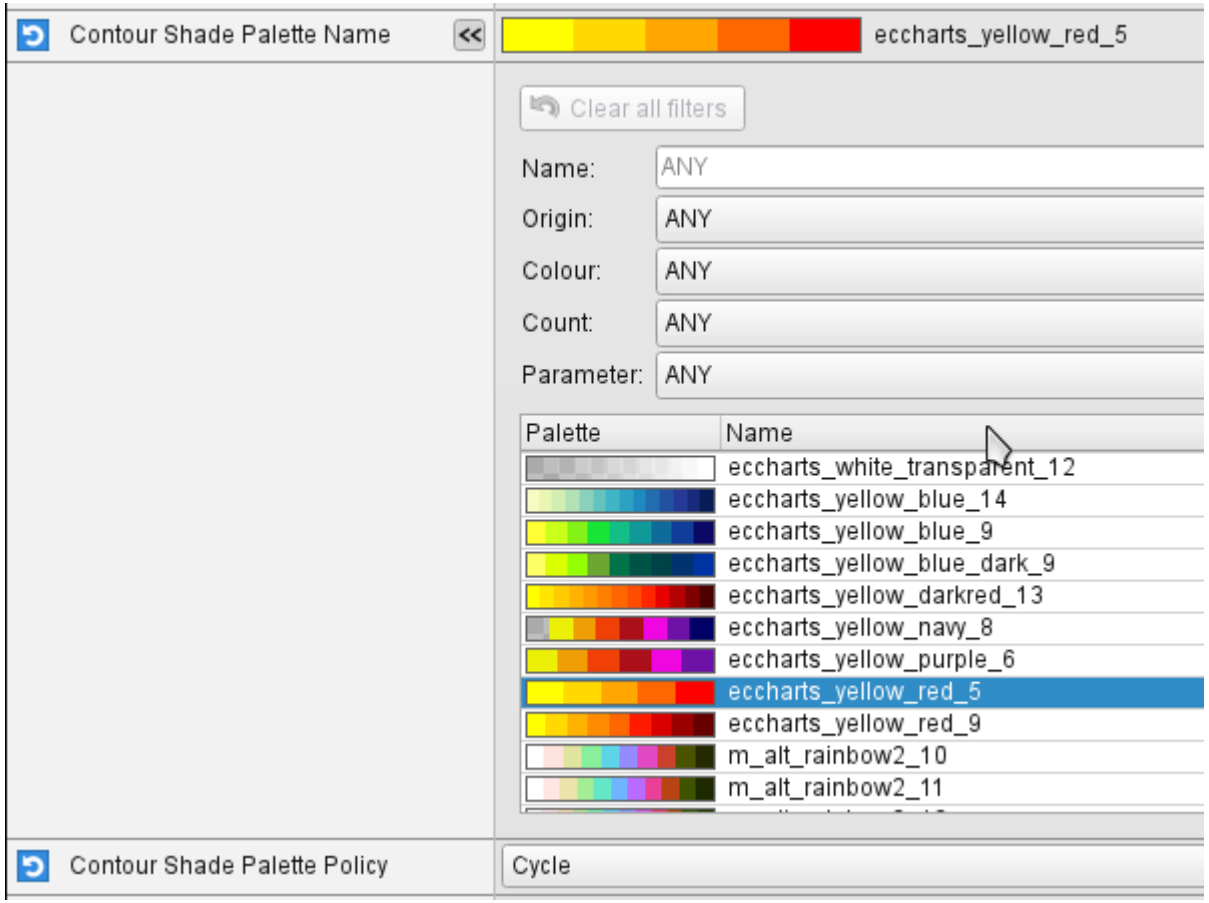


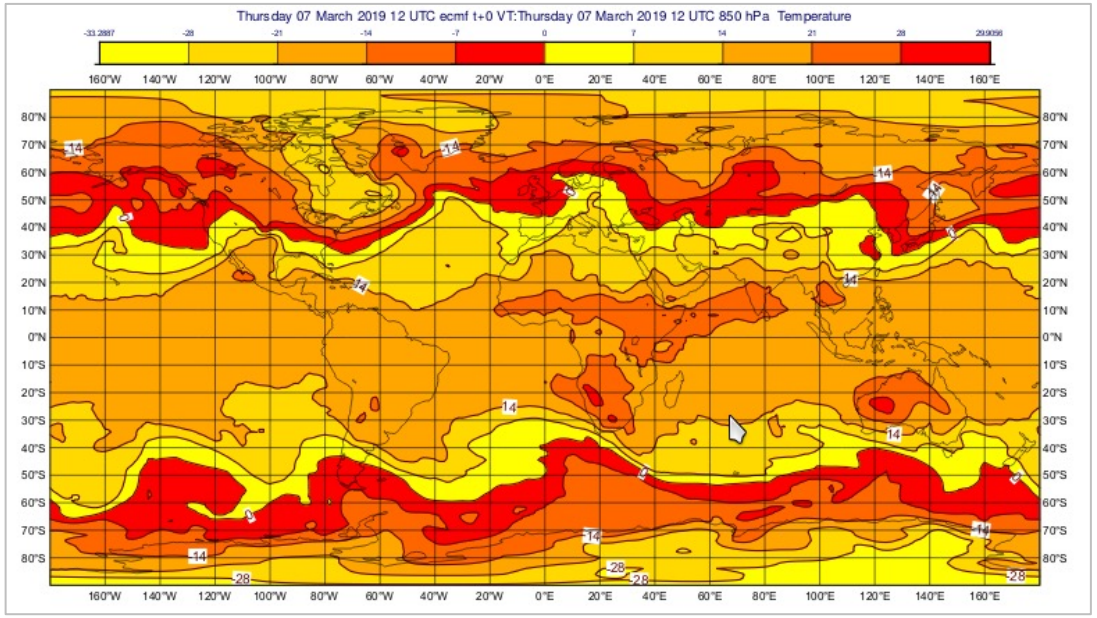




Contouring schemes

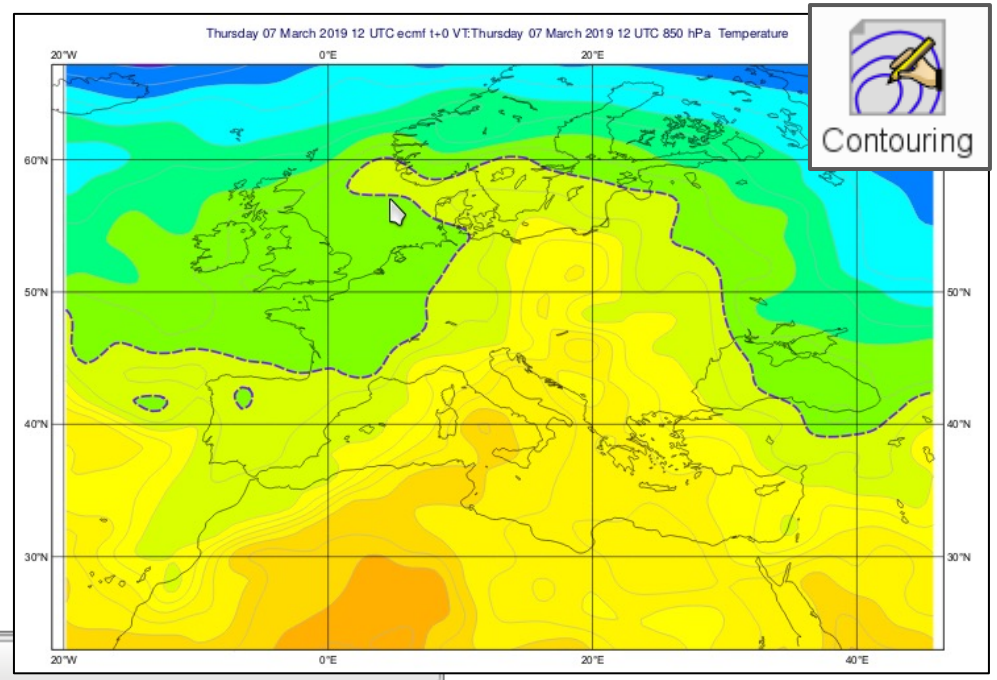
- A set of pre-defined palettes is also available
  - But you still have to supply the mapping between values and colours

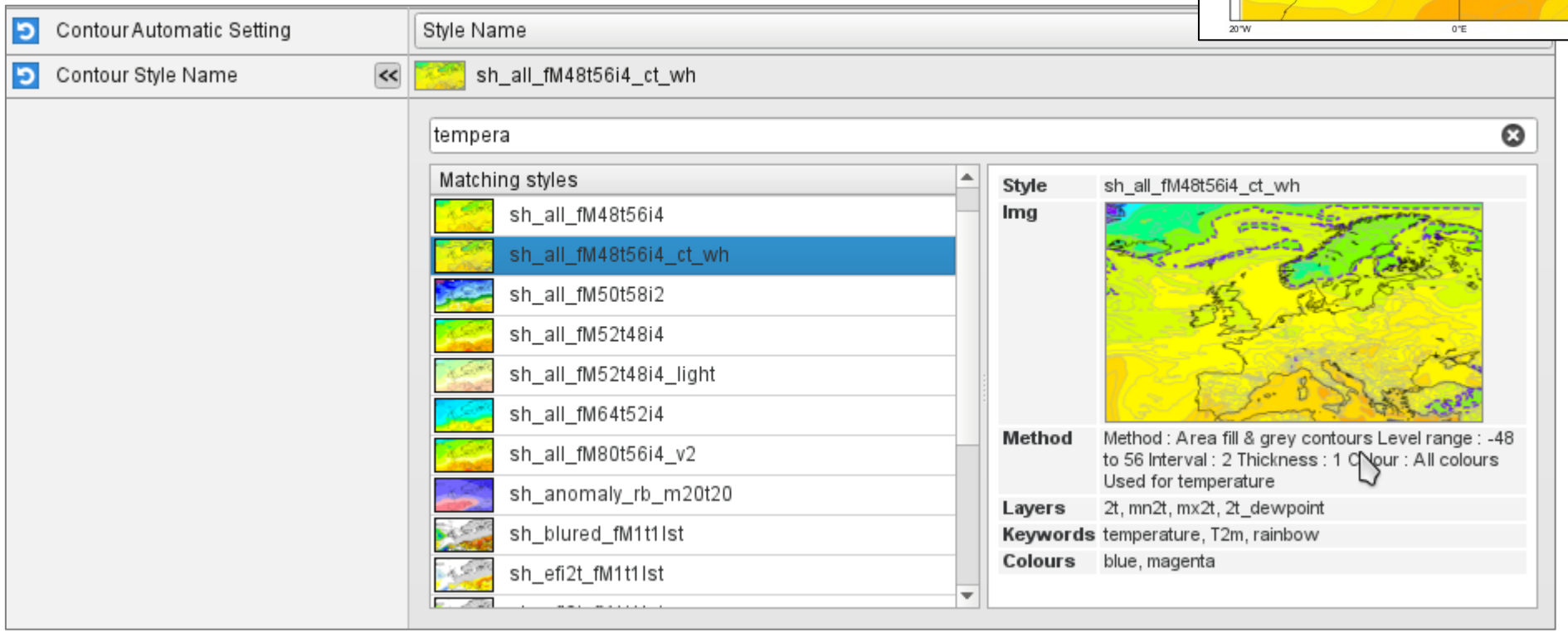




Contouring schemes

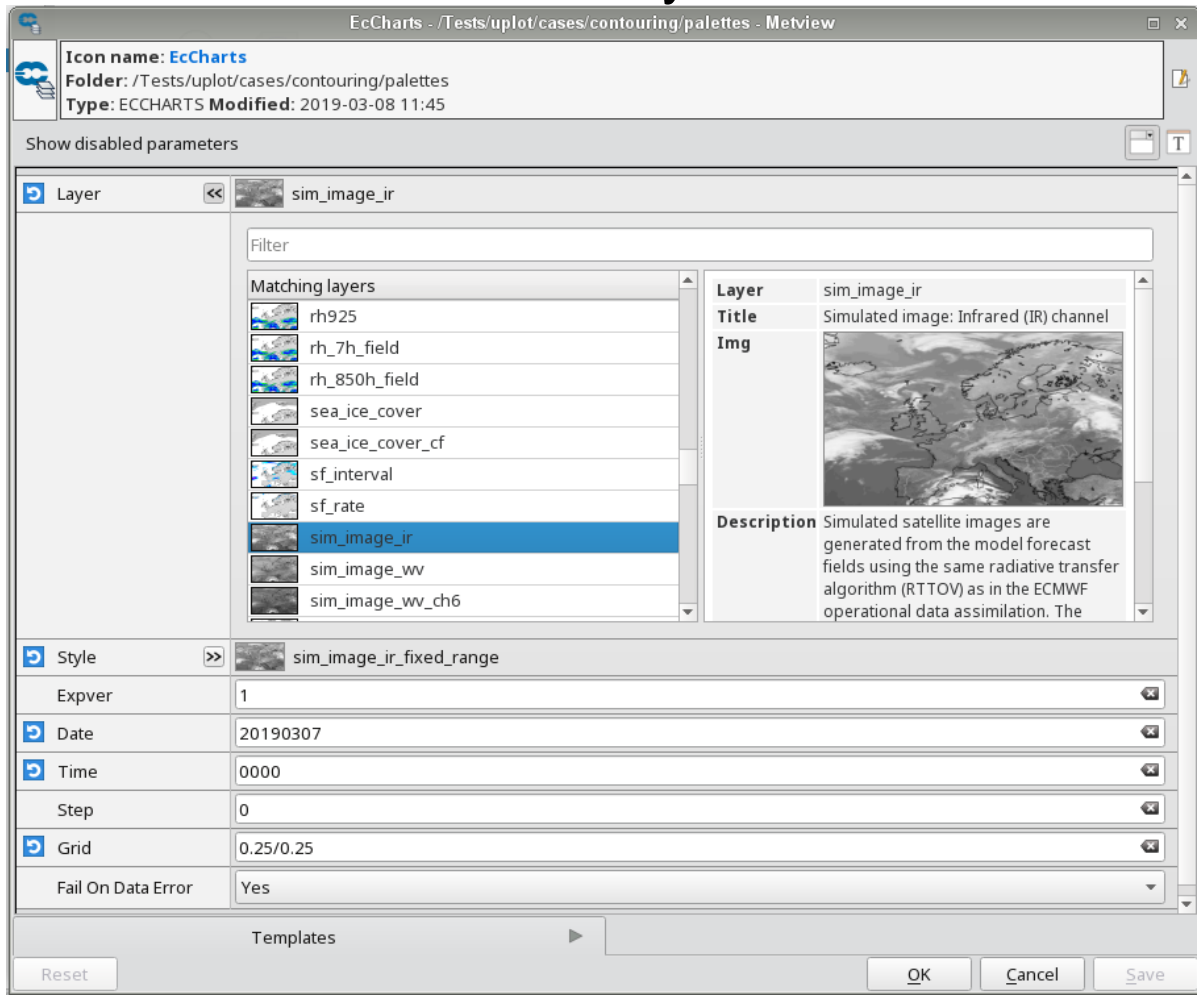
- Can select from pre-defined styles
  - the styles come from ecCharts
  - everything is done for you
  - or choose “Contour Automatic Setting = ECMWF” – style will be chosen based on meta-data

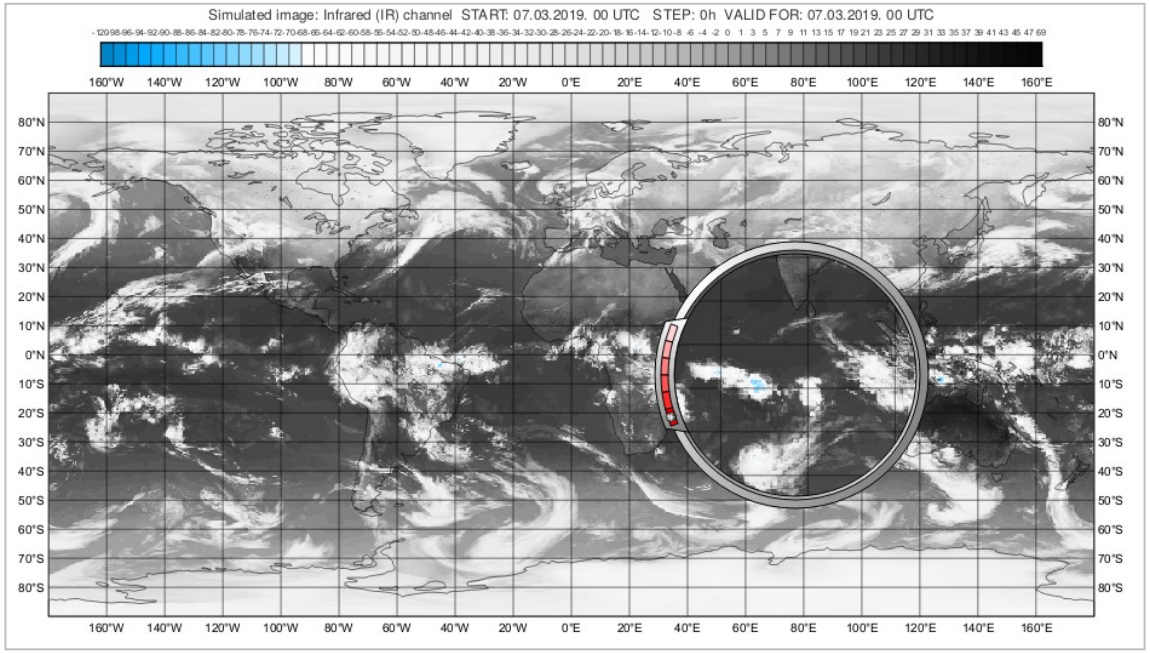




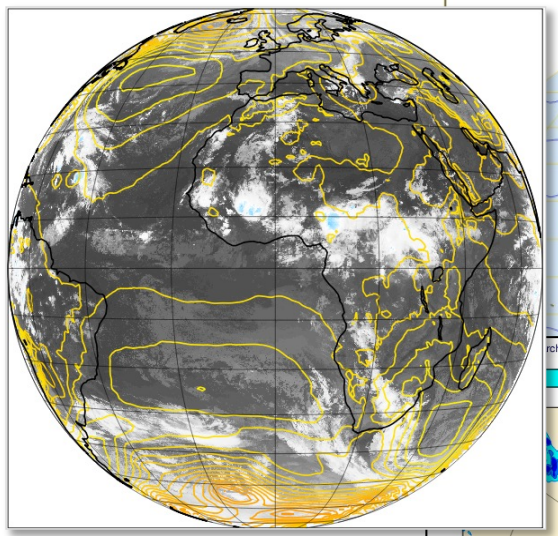
Complete ecCharts Layers

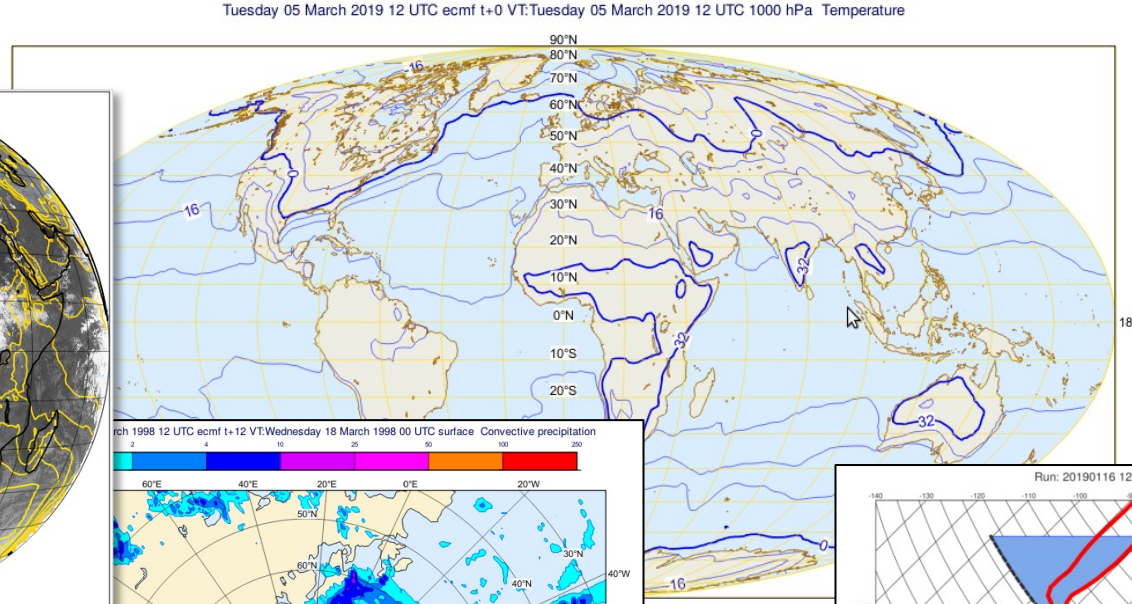
- The ecCharts icon goes further – retrieves data from MARS and styles it

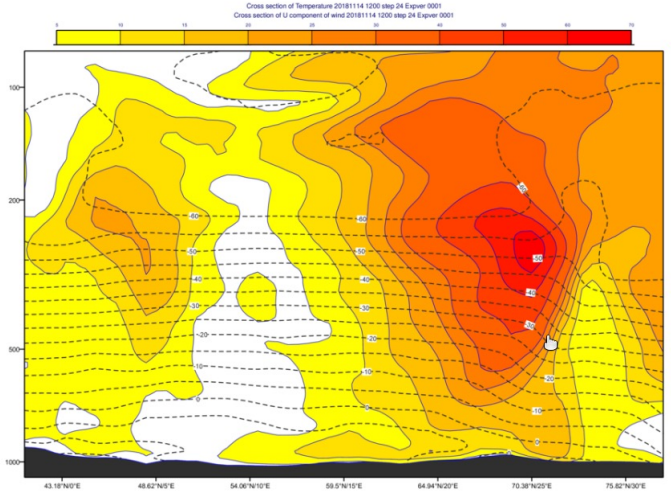


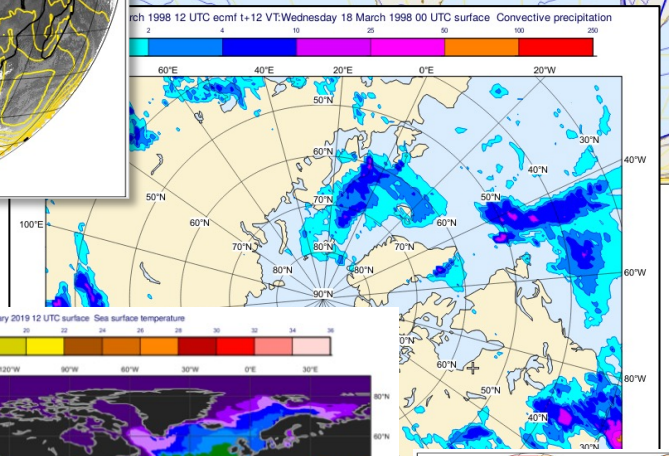


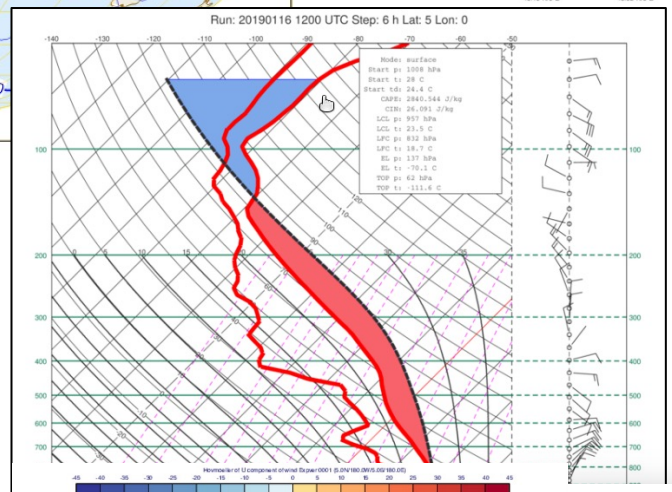
Views

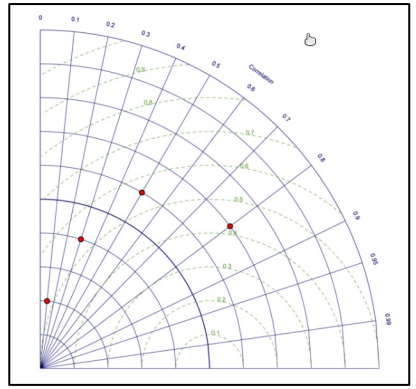


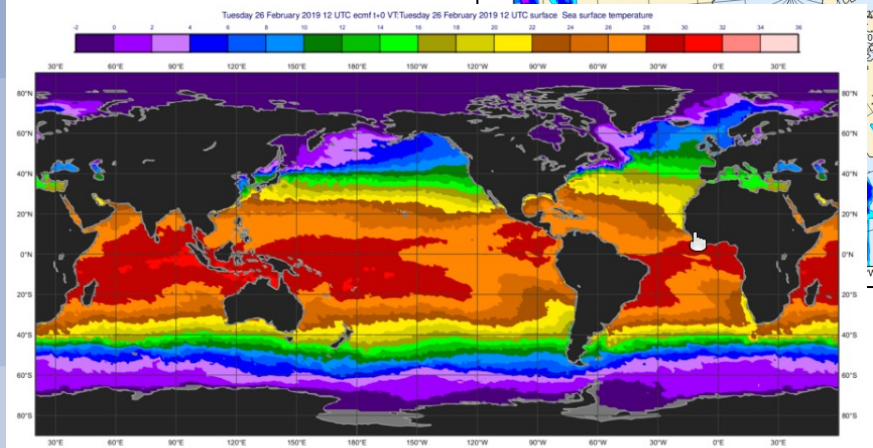


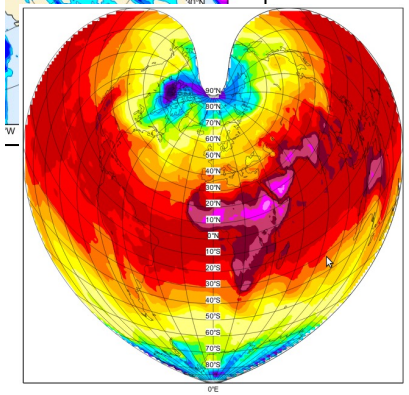


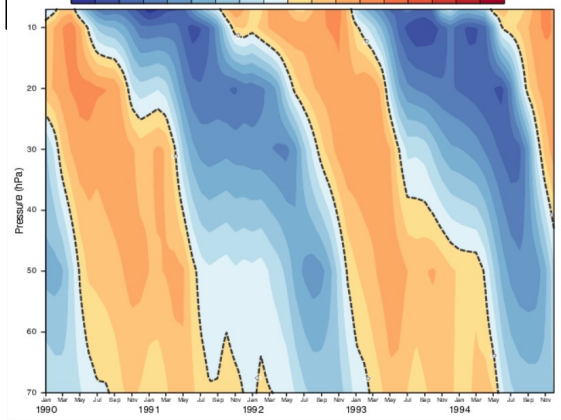


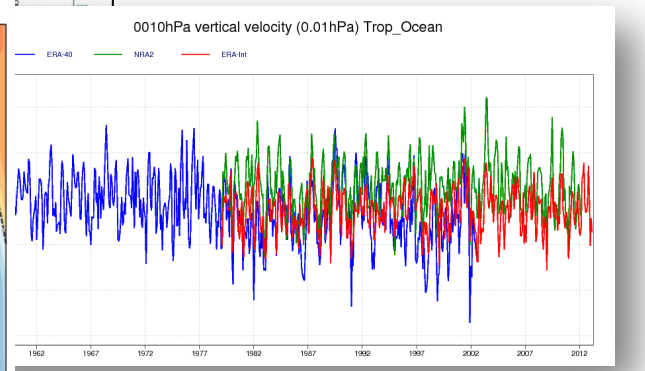










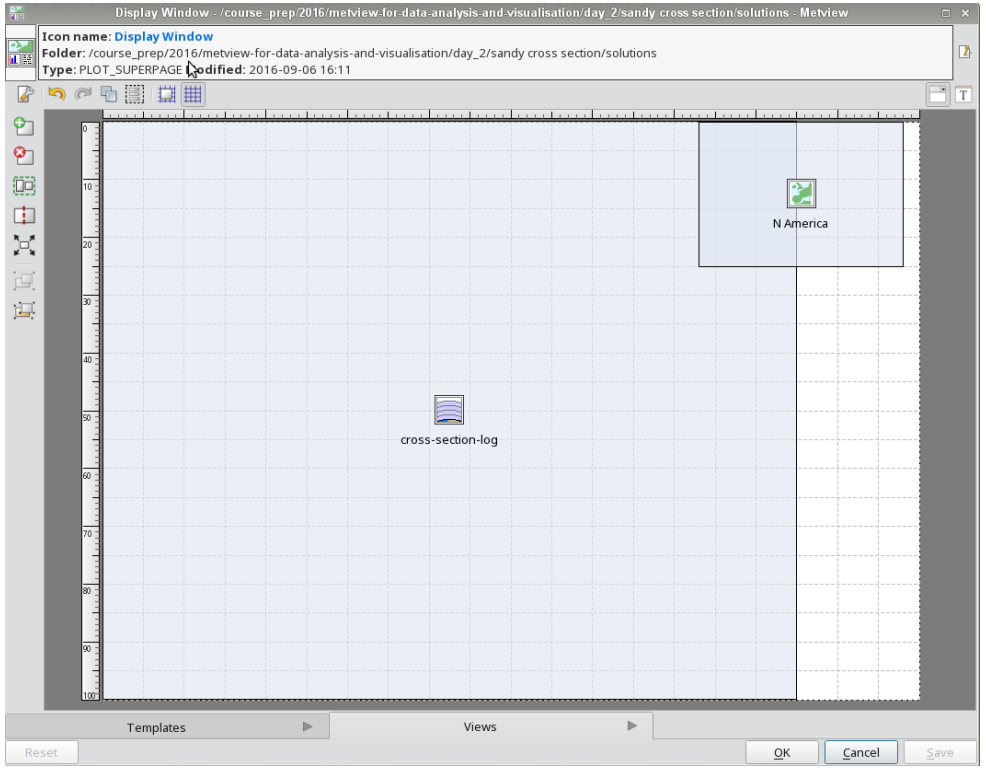


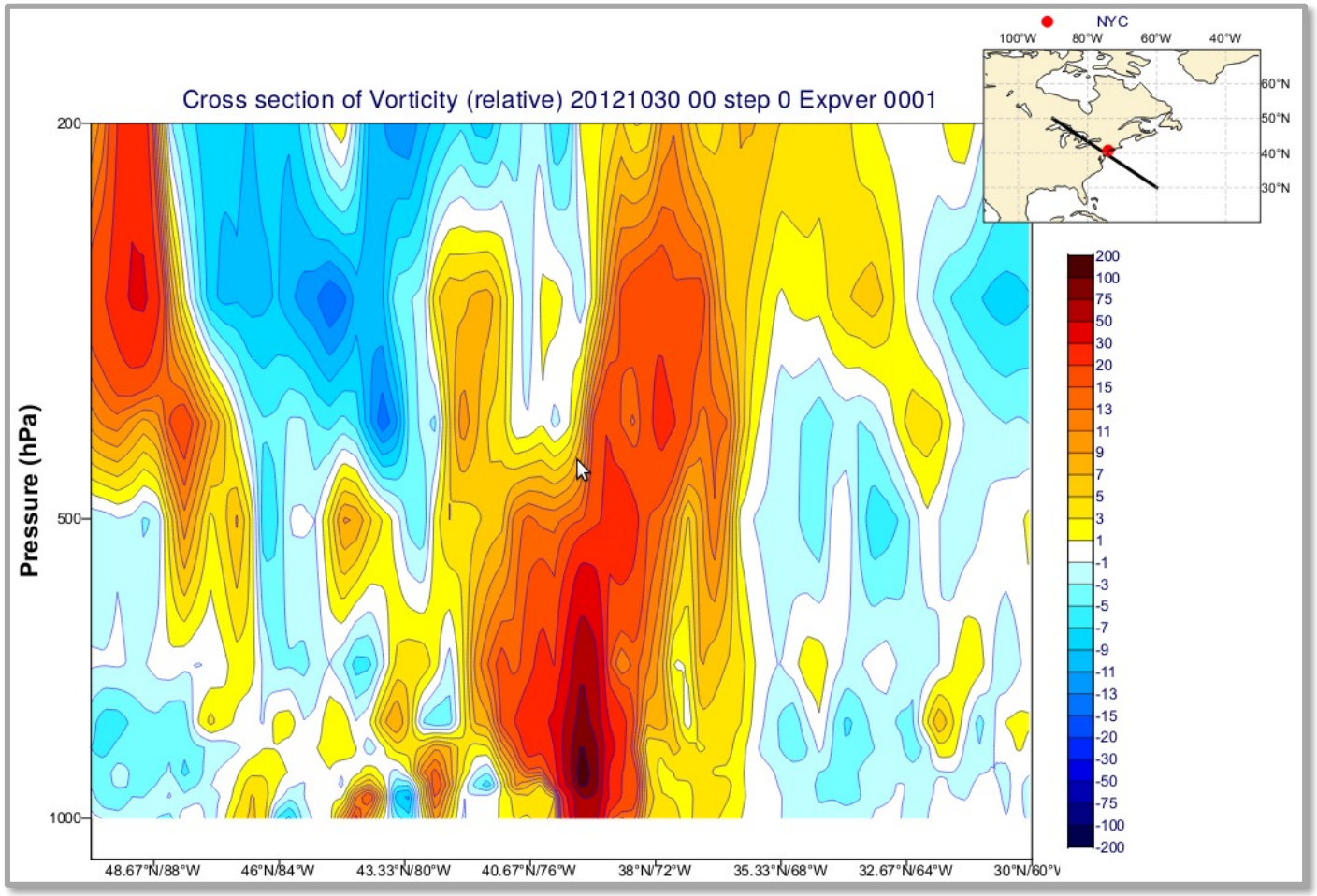
ECMWF

EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS

Visualisation - Layout

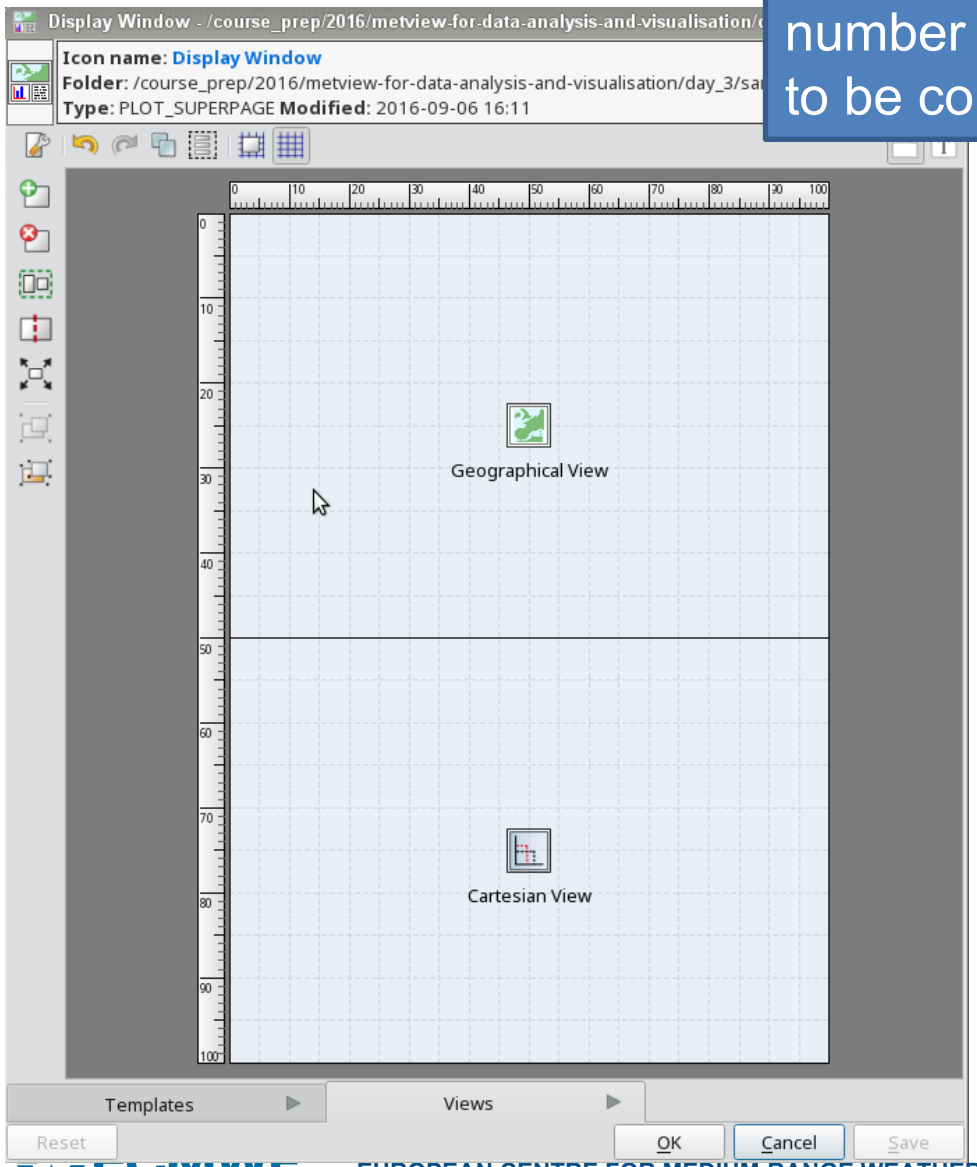
Layout editor allows any number of different views to be combined

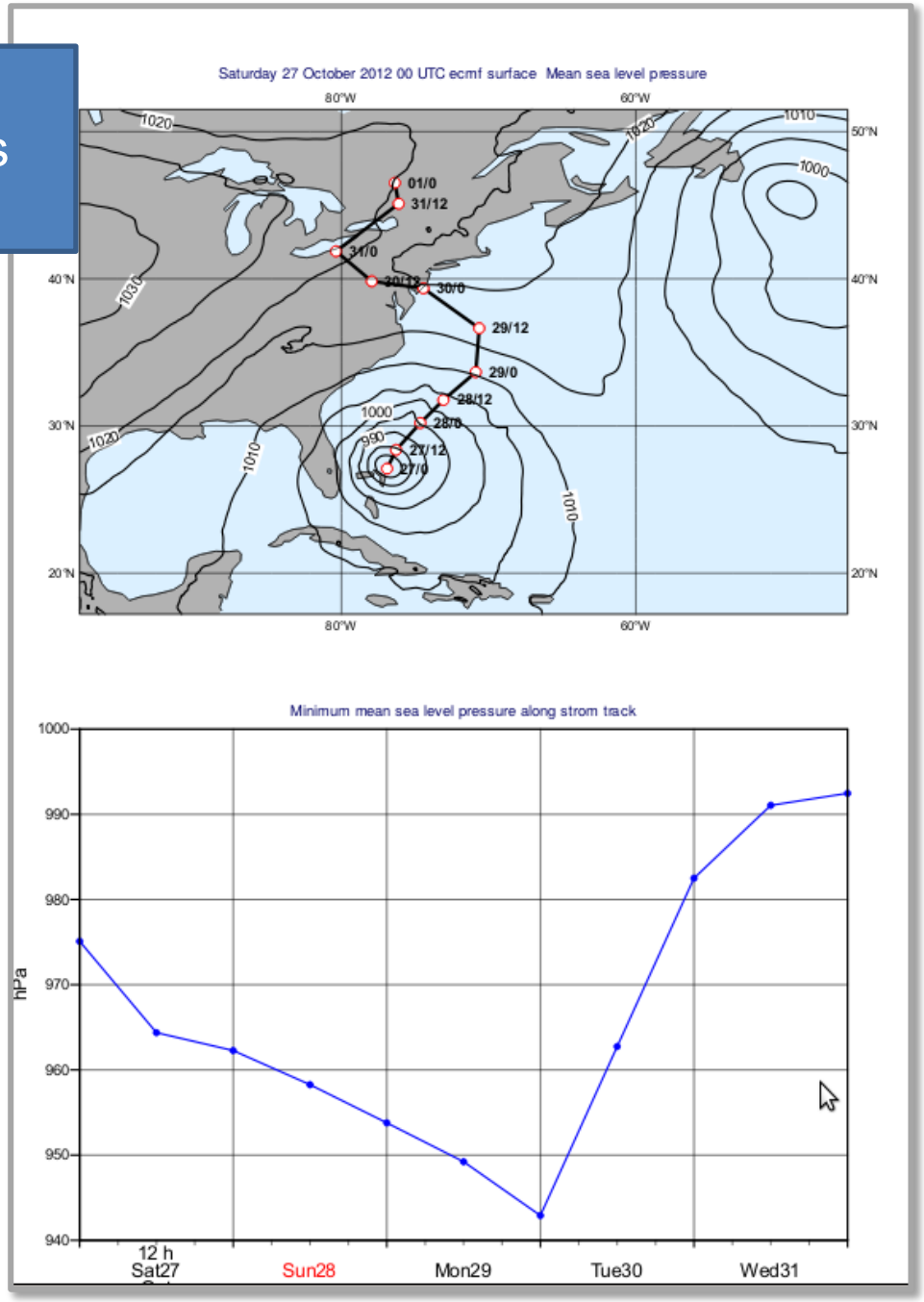




Visualisation - Layout

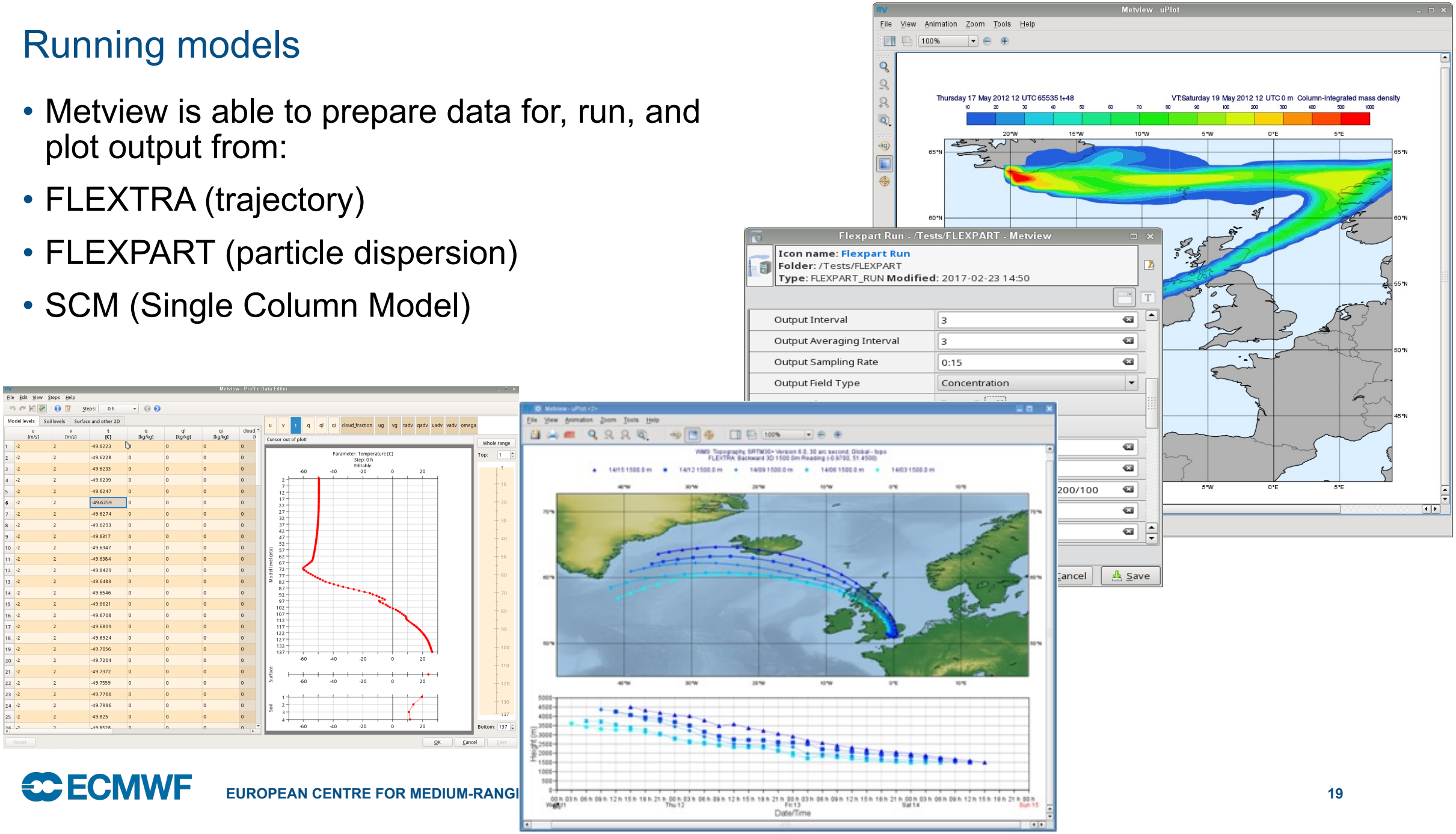
Layout editor allows any number of different views to be combined





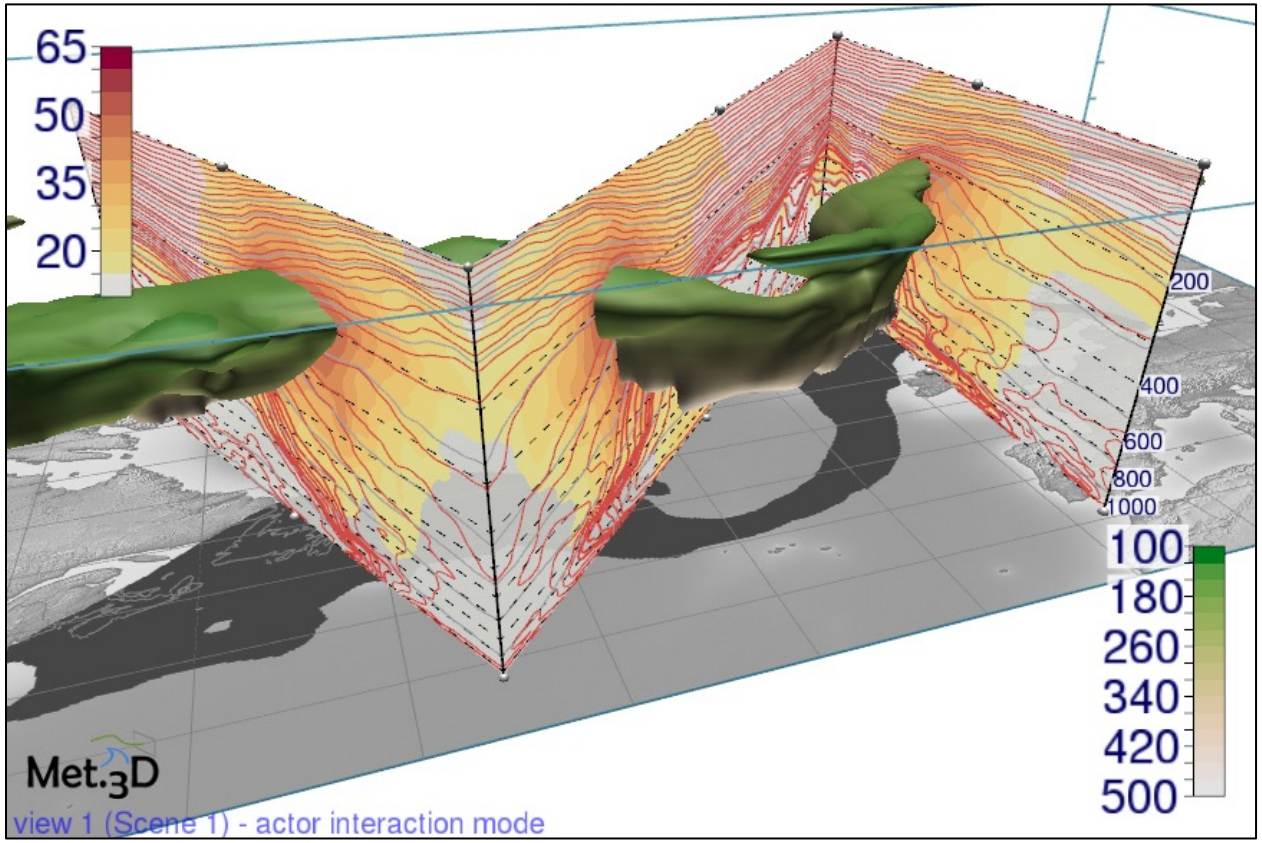
Running models

- Metview is able to prepare data for, run, and plot output from:
- FLEXTRA (trajectory)
- FLEXPART (particle dispersion)
- SCM (Single Column Model)

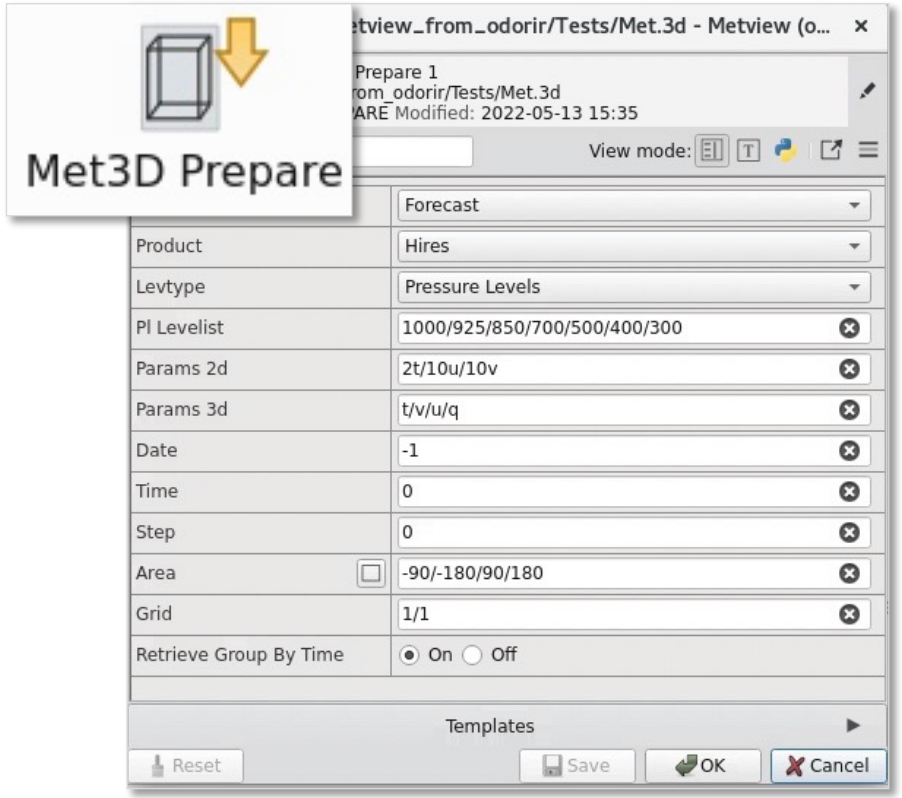


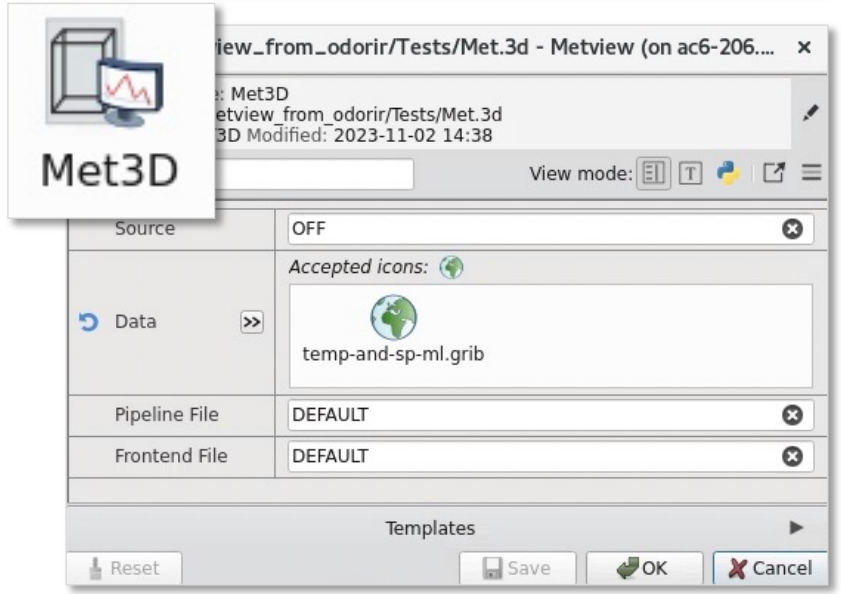
3D

- Metview can prepare data for, and launch Met.3D



Imagery produced by Met.3D (met3d.wavestoweather.de)





Python API

- Powerful, high-level scripting library
- Native handling of major data types
- Some nice features, e.g. model-obs differences (gridded minus scattered data)
- Can extract numpy, pandas and xarrays from Metview's data types

```
# filter out a timestep
wg = mv.read(data=g, step=78)

# create mask (with 0s and 1s) for windgust > 22 m/s
wg_mask = wg > 22

# compute probability
prob = mv.mean(wg_mask) * 100

# define contour shading
cont = mv.mcont(
    legend="on",
    contour_line_colour="charcoal",
    contour_highlight="off",
```

`w_from_omega()`	Computes the vertical velocity in m/s from pressure velocity

Thermodynamics

`dewpoint_from_relative_humidity()`	Computes the dewpoint for a given temperature and relative humidity
`dewpoint_from_specific_humidity()`	Computes the dewpoint for a given specific humidity and pressure
`eqpott_m()`	Computes the equivalent potential temperature on model levels
`eqpott_p()`	Computes the equivalent potential temperature on pressure levels
`lifted_condensation_level()`	Computes the Lifted Condensation Level (LCL) using the parcel method

ecific humidity

re for a given

lifted_condensation_level

`lifted_condensation_level`(*t, td, p*)

Computes the Lifted Condensation Level (LCL) for a parcel ascending from a given temperature, dewpoint and pressure.

Parameters
- **t** (number, ndarray or `Fieldset`) – initial temperature (K)
- **td** (number, ndarray or `Fieldset`) – initial dew point temperature (K)
- **p** (number, ndarray or `Fieldset`) – initial pressure (Pa)

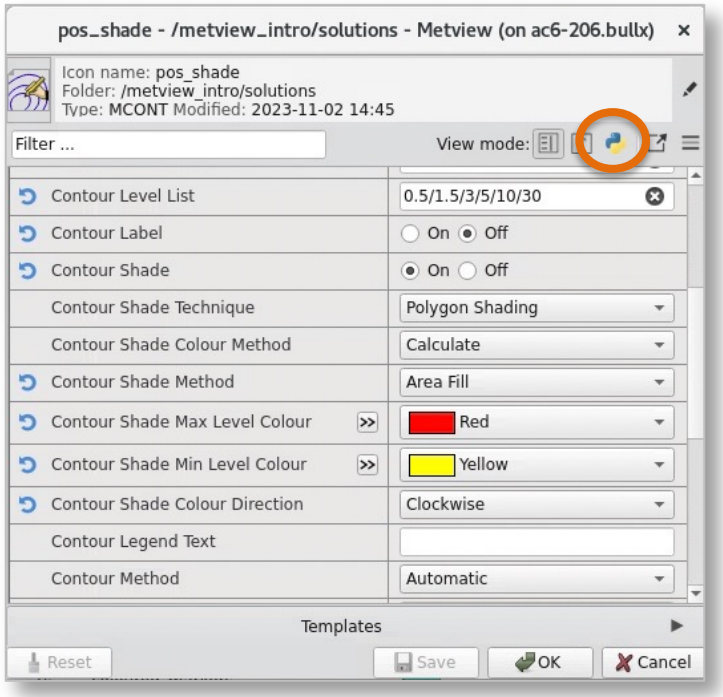
Return type dict or None

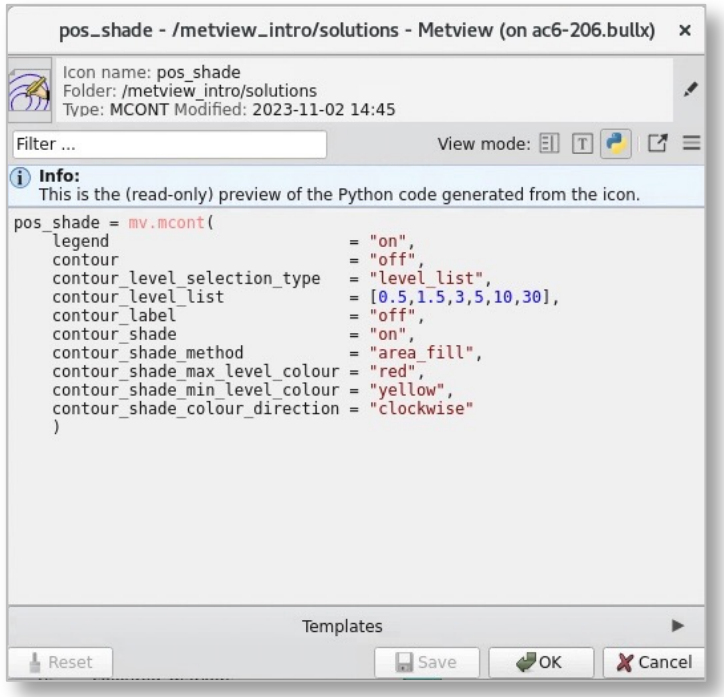
The LCL is the level where the parcel becomes saturated during an adiabatic ascend. First, the LCL temperature is computed with the formula from [Bolton1980]:

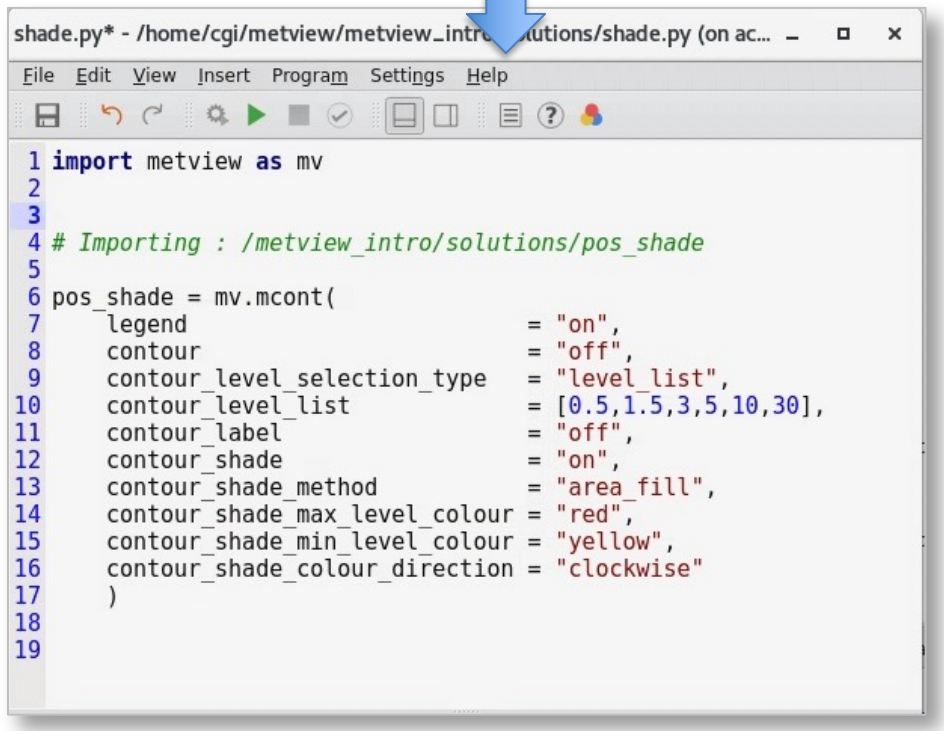
$$t_{LCL} = 56.0 + \frac{1}{\frac{1}{td-56} + \frac{log(\frac{t}{td})}{800}}$$

Python API

- Every icon has a corresponding Python function call
- Two ways to generate Python code from icons:
- 1) click on the Python icon inside an icon editor, then copy & paste the code from there
- 2) drop an icon into Metview's Code Editor

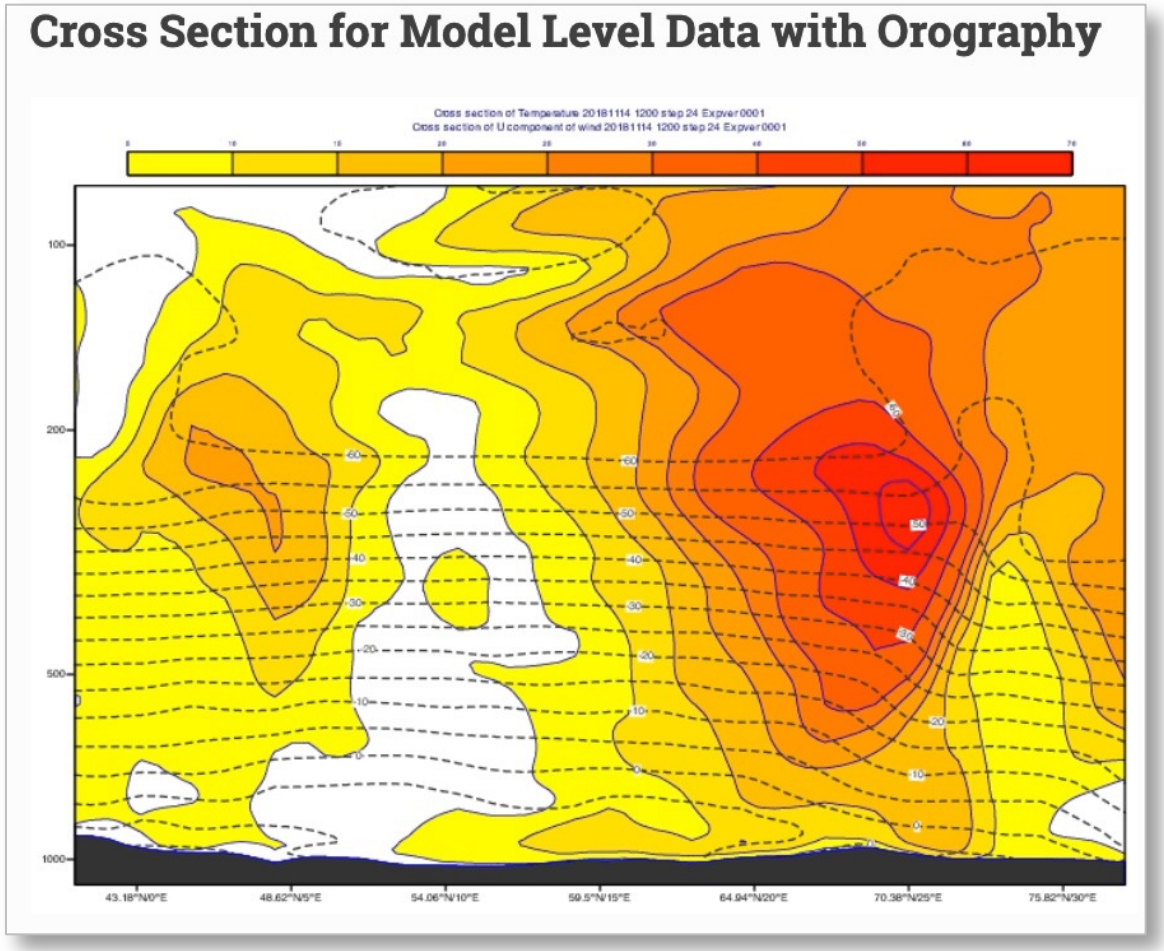


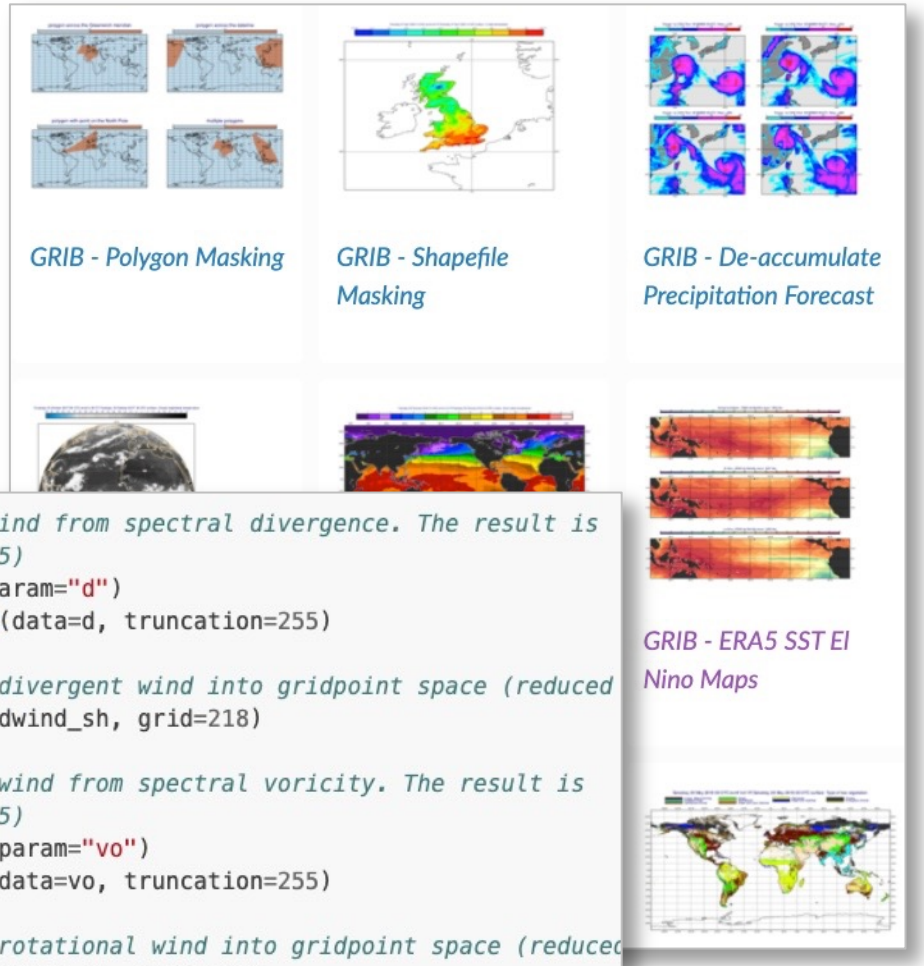




Examples

- See the Gallery for Python examples

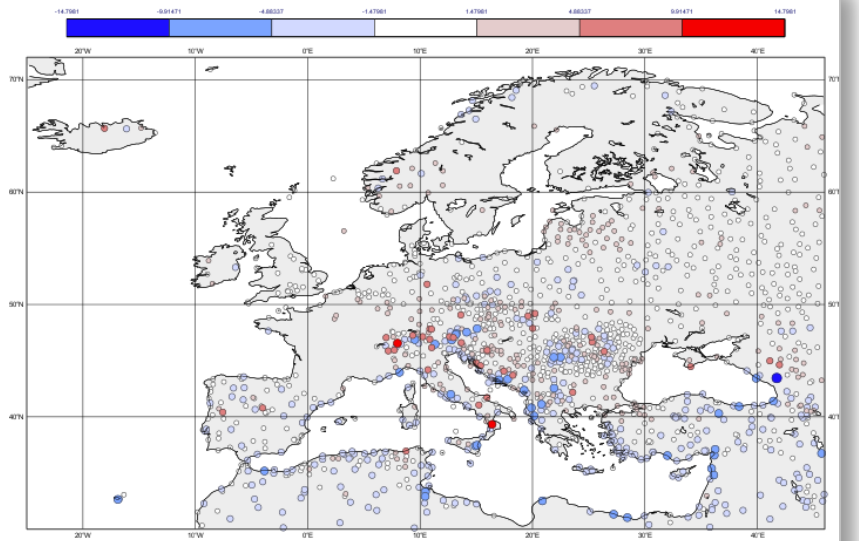





```
# compute divergent wind from spectral divergence. The result is
# still spectral (T255)
d = mv.read(data=g, param="d")
dwind_sh = mv.divwind(data=d, truncation=255)

# transform spectral divergent wind into gridpoint space (reduced
dwind = mv.read(data=dwind_sh, grid=218)

# compute rotational wind from spectral voricity. The result is
# still spectral (T255)
vo = mv.read(data=g, param="vo")
rwind_sh = mv.divrot(data=vo, truncation=255)

# transform spectral rotational wind into gridpoint space (reduced
rwind = mv.read(data=rwind_sh, grid=218)

# define coastlines
coast = mv.mcoast(
    map_coastline_colour="cream",
    map_coastline_land_shade="on",
    map_coastline_land_shade_colour="RGB(0.2,0.2,0.2)",
    map_coastline_sea_shade="on",
    map_coastline_sea_shade_colour="RGB(0.3,0.3,0.3)",
    map_grid_colour="RGB(0.9333,0.9333,0.9333)",
)
```

Examples

- See also the Jupyter Notebooks



Out [14]:

We can easily convert this to a pandas dataframe for further analysis.

In [15]:

```
df = diff.to_dataframe()
```

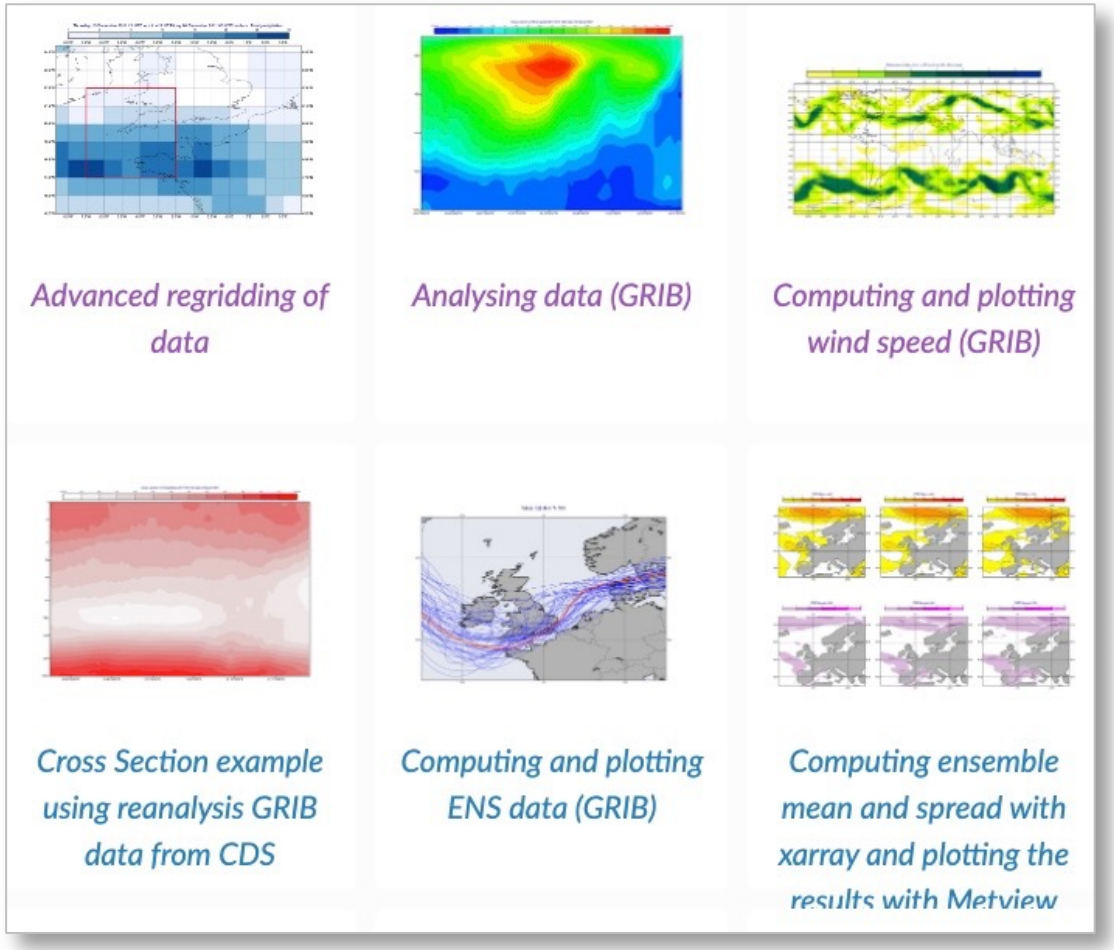
Print a summary of the whole data set:

In [16]:

```
df.describe()
```

Out [16]:

	latitude	longitude	value	level
count	1471.000000	1471.000000	1471.000000	1471.0
mean	46.557104	21.160707	-0.201723	0.0
std	8.350950	14.272239	2.417394	0.0
min	30.110000	-22.590000	-10.236664	0.0

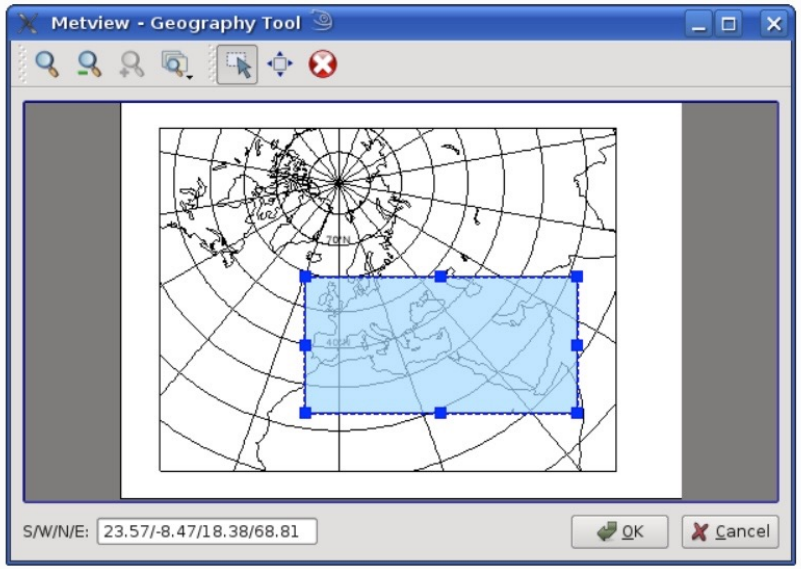


Tutorials

- Plenty of material online including tutorials

Now we want to set the area used in the view. Although we can interactively zoom into smaller areas in the **Display Window**, we now want to store a particular area so that we can use exactly the same one again and again. Set the **Map Area Definition** to Corners and click on the **Geography Tool** button next to the **Area** parameter (shown in the picture below).

This tool helps you define a region.



Course modules

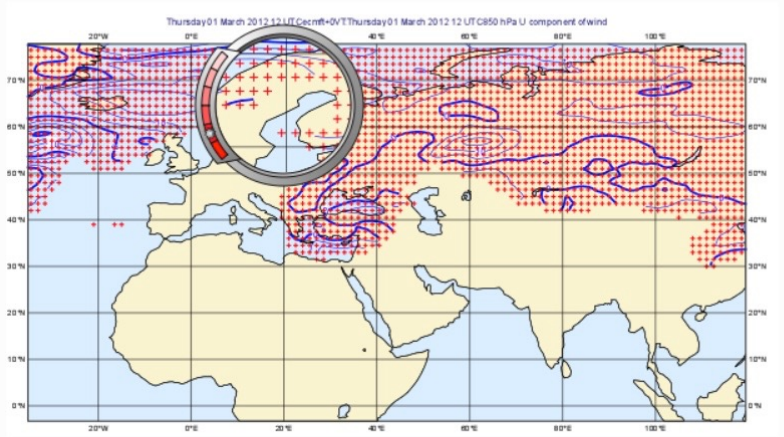
- A Simple Visualisation
- Customising Your Plot
- Case Study: Plotting Hurricane Sandy on a Map
- Data Part 1
- Processing Data
- Analysis Views
- Layout in Metview
- Case Study: Cross Section of Sandy
- Data Part 2
- Handling Time in Metview
- Graph Plotting in Metview
- Case study: Plotting the Track of Hurricane Sandy
- Working with graphical output
- Organising Macros
- Missing Values and Masks
- Optimising Your Workflow
- Customising Your Plot Title
- Case study: Ensemble Forecast
- Running Metview in Batch Mode
- Working with Folders and Icons
- Exploring Metview

Tutorials on specific topics

- Webinar (2022): Slicing and dicing GRIB data and associated notebook Slicing and dicing GRIB data
- Webinar (2021): Stop, think, interpolate! and associated notebook Advanced regridding of data
- Workshop presentation from UEF 2022: Interactive data visualisation and pre-processing with ECMWF's Metview software

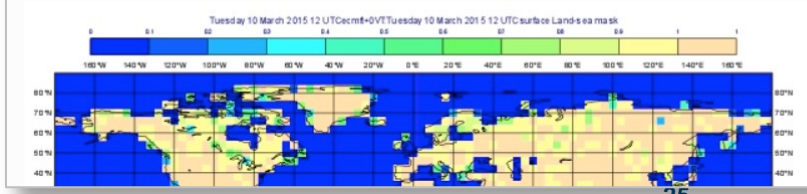
- BUFR tutorial
- ODB Tutorial 1
- ODB Tutorial 2
- FLEXPART tutorial
- FLEXTRA tutorial
- VAPOR Tutorial
- Metview WMS Tutorial
- The SCM Interface in M

Overview



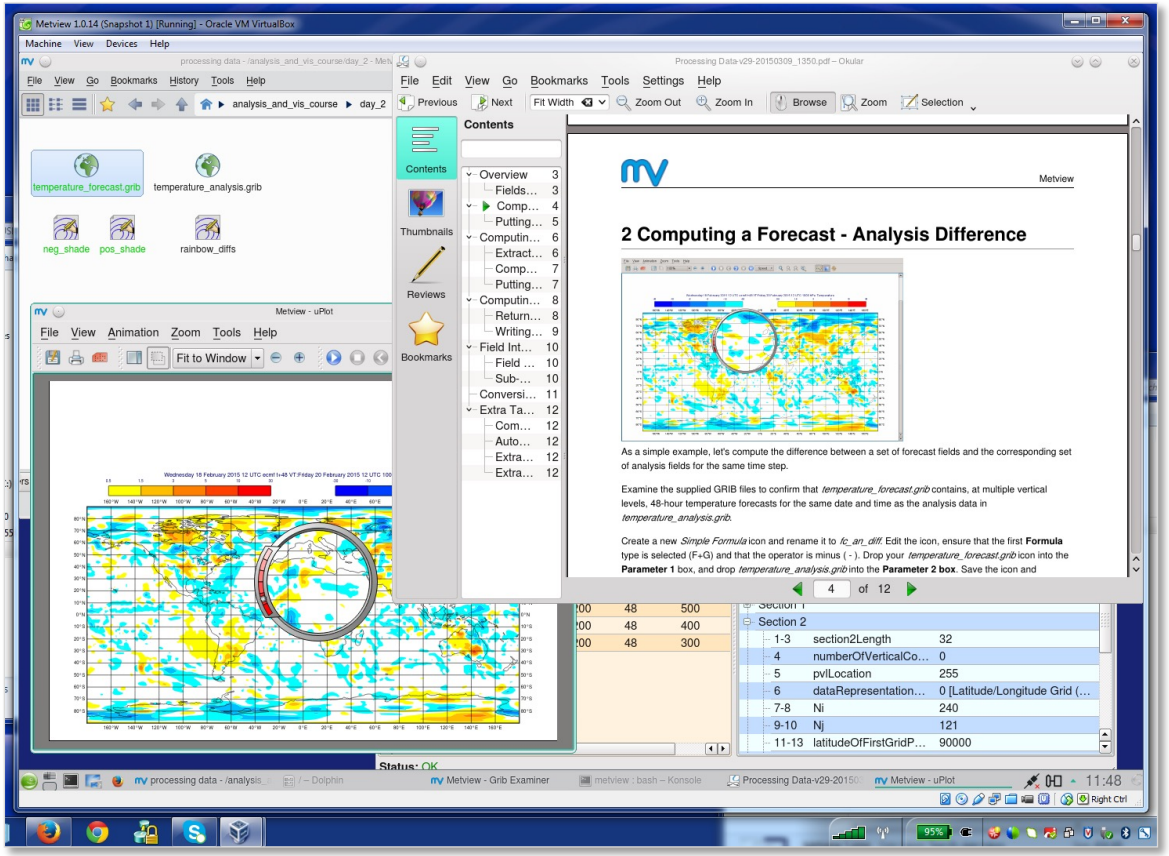
Fields and observations can often contain missing values - it can be important to understand the implications of these, and also how to use them to remove unwanted data points. Using a mask of missing values can enable Metview to perform computations on a specific subset of points.

Computing the mean surface temperature over land



Metview availability

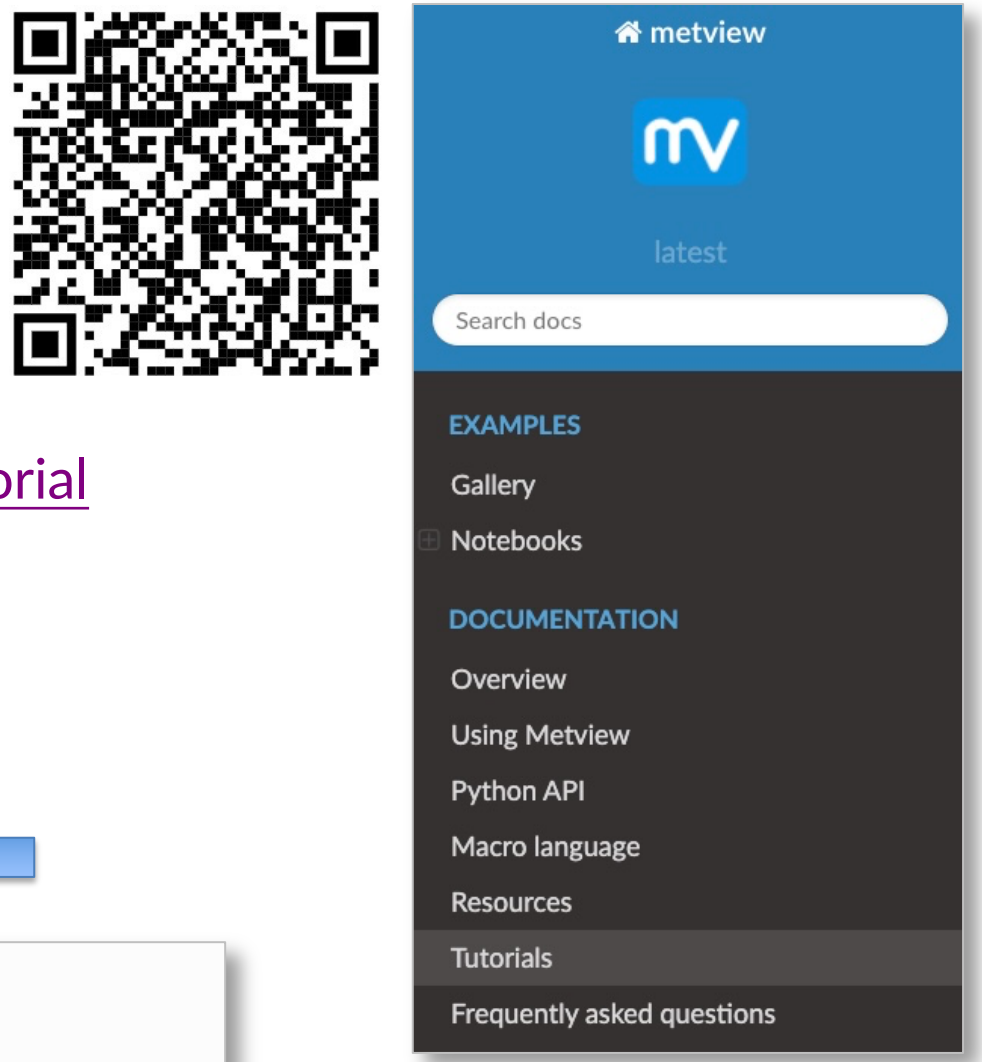
- Available for Linux and macOS
- Inside ECMWF
  - `module load ecmwf-toolbox ; metview`
- Install from binaries
  - openSUSE, Fedora, Ubuntu
- Conda
  - `conda install metview -c conda-forge`
    `conda install metview-batch -c conda-forge`
    `conda install metview-python -c conda-forge`
- Homebrew
  - `brew install metview`
- Build from source
- The Metview Python interface can be installed separately if not in conda:
  - `pip install metview`

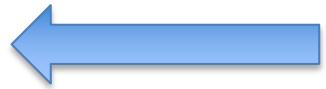


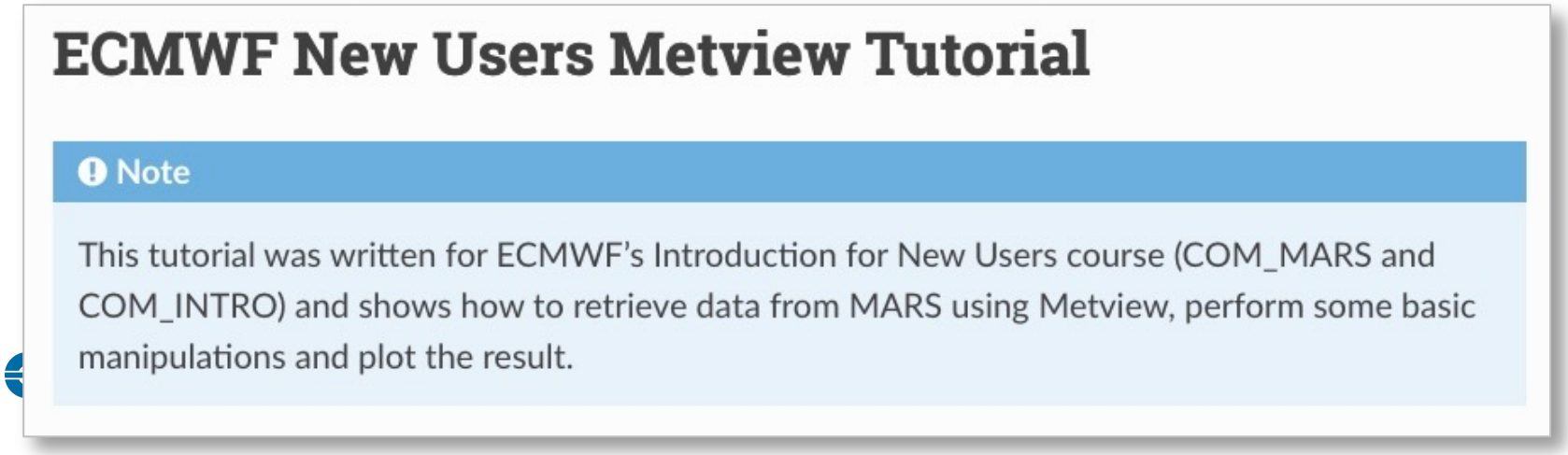
Exercise (in GatherTown)

- Find the Metview documentation on ReadTheDocs:
- https://metview.readthedocs.io/en/latest/index.html

- Navigate: Tutorials -> ECMWF New Users Metview Tutorial

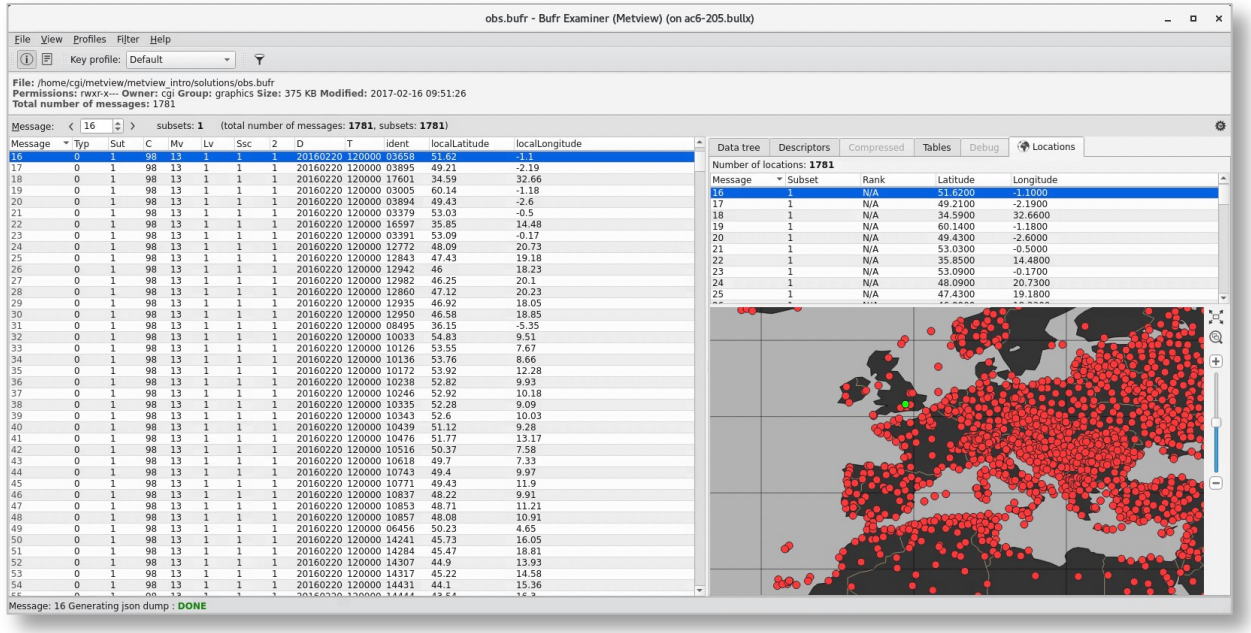




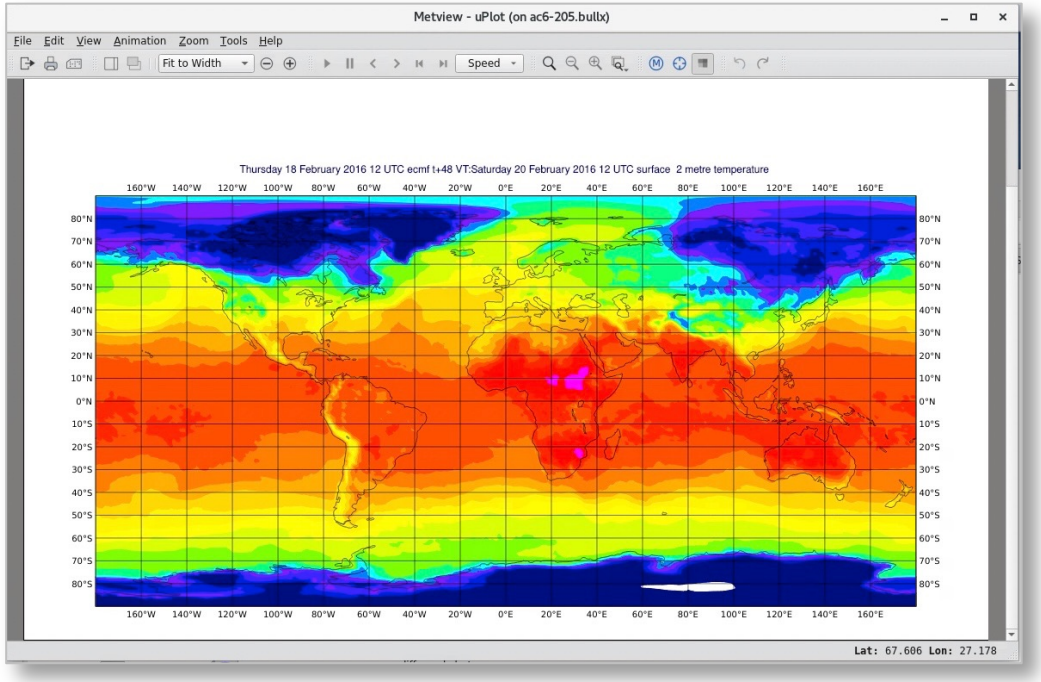


Pro Tips (command-line tools)

- `metview -e <GRIB/BUFR/… file>`
  - Starts up the examiner

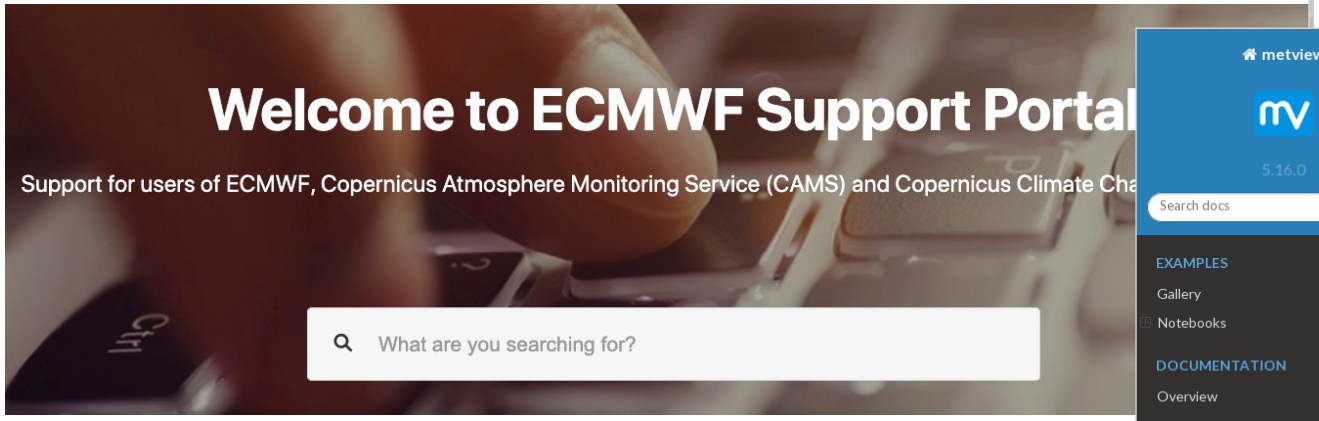


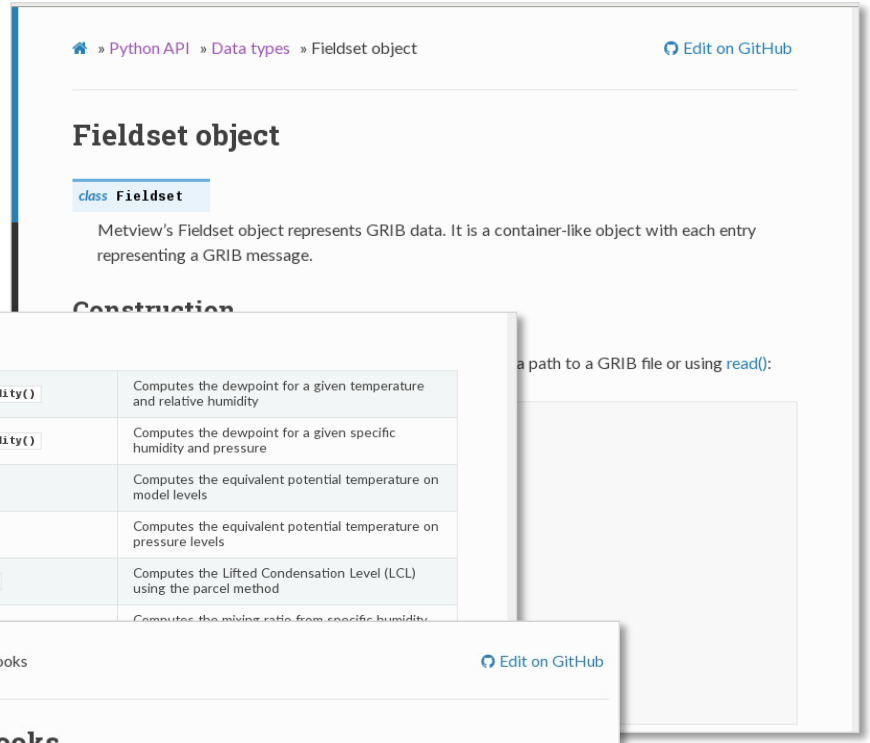
- `metview -p <file>`
  - Starts up the plot window with the given data – tries to find the corresponding ecCharts style if available
- `metview -slog`
  - Gives more debug output to stdout
- `metview -u /path/to/new/metview/home`
  - Use another home dir ($HOME/metview is default)

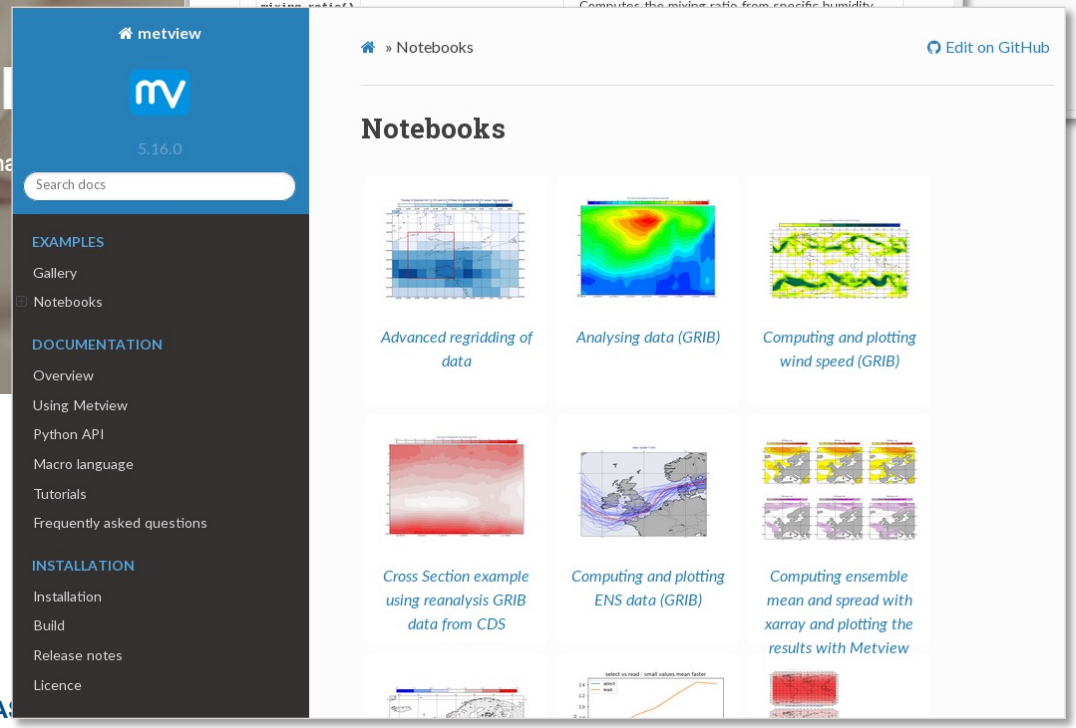


For more information…

- Ask for help:
  - https://www.ecmwf.int/en/support
- Visit our web pages:
  - https://metview.readthedocs.io/en/latest/index.html







earthkit

ECMWF's next-generation data-processing toolkit family

python™

earthkit-data | earthkit-plots

earthkit-meteo | earthkit-climate

earthkit-maps | earthkit-regrid

https://earthkit.readthedocs.io/

earthkit
latest
Search docs
COMPONENTS
Overview
earthkit-data
earthkit-maps
INSTALLATION
Installation
License

Welcome to earthkit's documentation
Edit on GitHub

Welcome to earthkit's documentation

Warning

This project is **BETA** and will ... **Experimental** for the foreseeable future. Interfaces and functionality are ... to chan... and the project itself may be scrapped. **DO NOT** use this software in any p... ...ftwar... ...hat is operational.

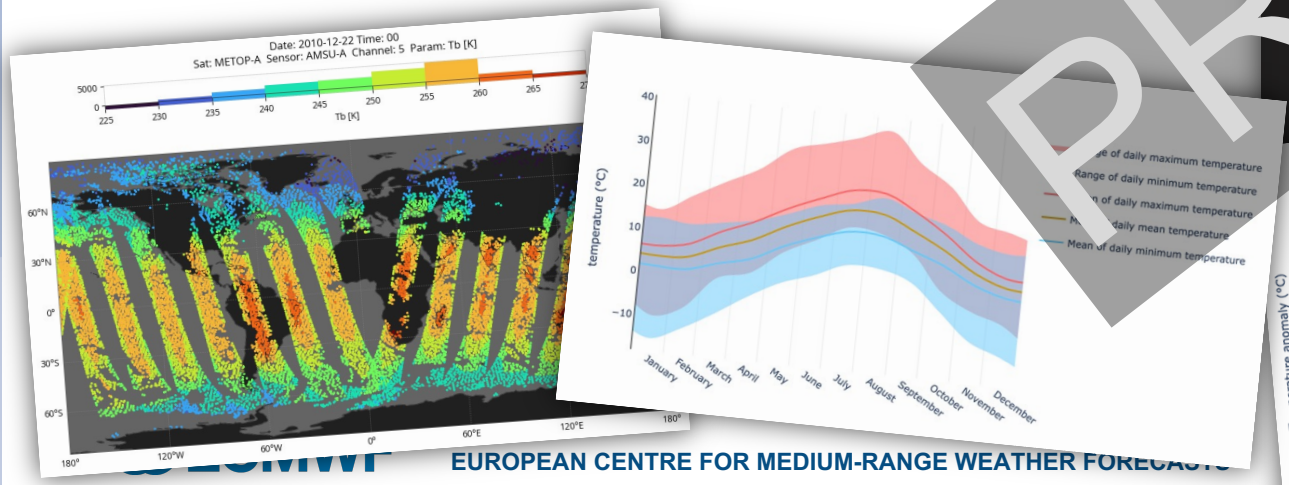
Warning

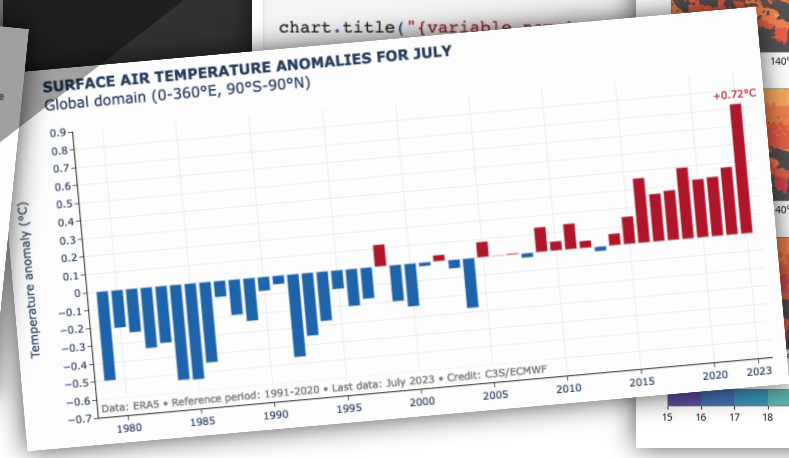
This documentatio... ... work in progress and can only be regarded as a **DRAFT**.

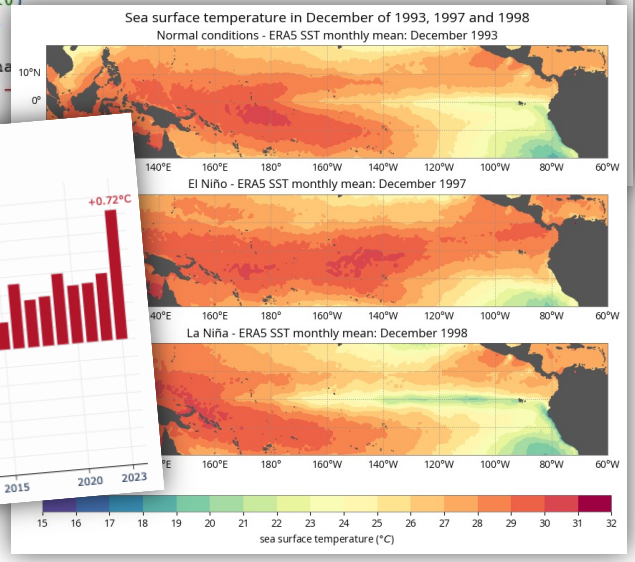
...kit is a new ...en-source Python project led by ECMWF, providing powerful tools for speeding ... weather ... climate science workflows by simplifying data access, processing, analysis, vis... ... and much more.

...arthkit offers **multiple interoperable** software components built on top of well-established open-source Python libraries like numpy, pandas and matplotlib. earthkit also integrates and leverages

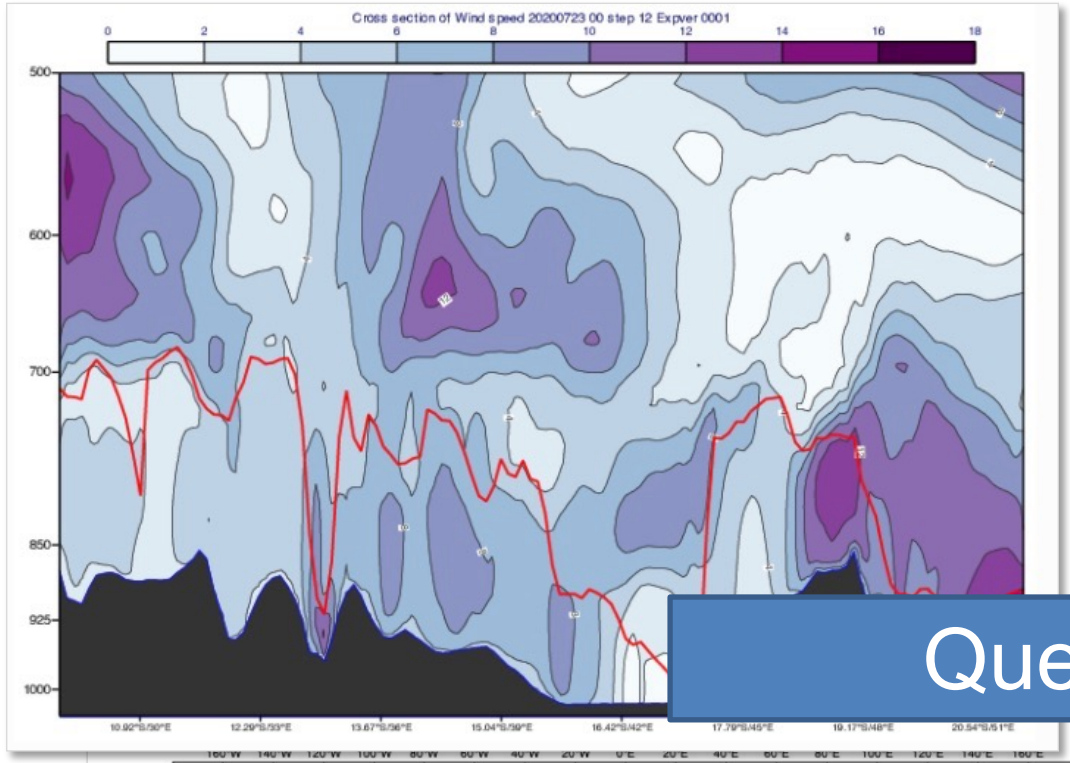
```
...t = earthkit.maps.Superplot(domain=[100, 300, -20, 20], rows=3)

chart.plot(data, style=style)

chart.land(color="#555", zorder=10)
chart.gridlines(xlocs=range(-180,

for subplot, conditions in zip(cha
    subplot.title(f"{conditions} -

chart.title("{variable
```







PREVIEW



Finding a range of extreme values

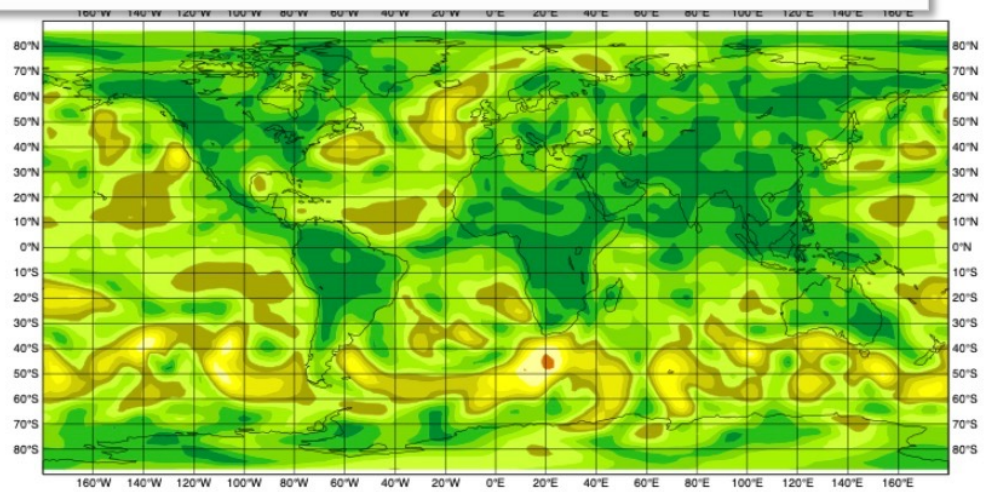
Find the locations where the value is within 95% of the maximum by supplying a range of values:

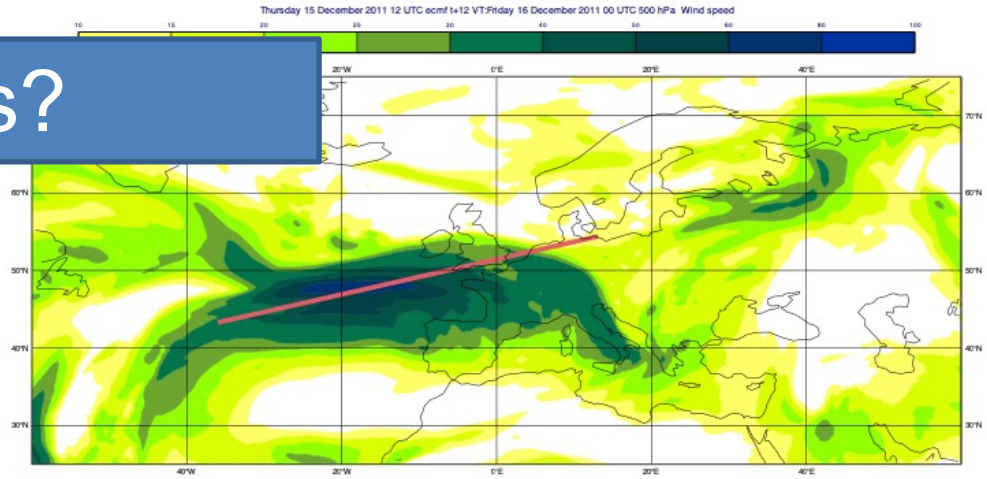
```
[12]: mv.find(wg0, [max0*0.95, max0])
[12]: [[48.5, -5.0], [48.5, -4.5], [47.0, -7.0], [47.0, -3.5], [47.0, -3.0]]
```

If we want to work with these points in Metview, the easiest way is to use the gfind() function to return a **Geopoints** variable:

```
[13]: max_points = mv.gfind(wg0, max0, max0*0.05)
```

Questions?





Define a cross section view along an area of interest

```
[36]: xs_view = mv.mxsectview(
          bottom_level = 1000.0,
          top_level    = 100,
          line         = line
          )
```

Create a colour scale to plot the data with

```
[37]: xs_shade = mv.mcont(
          legend                    = "on",
```