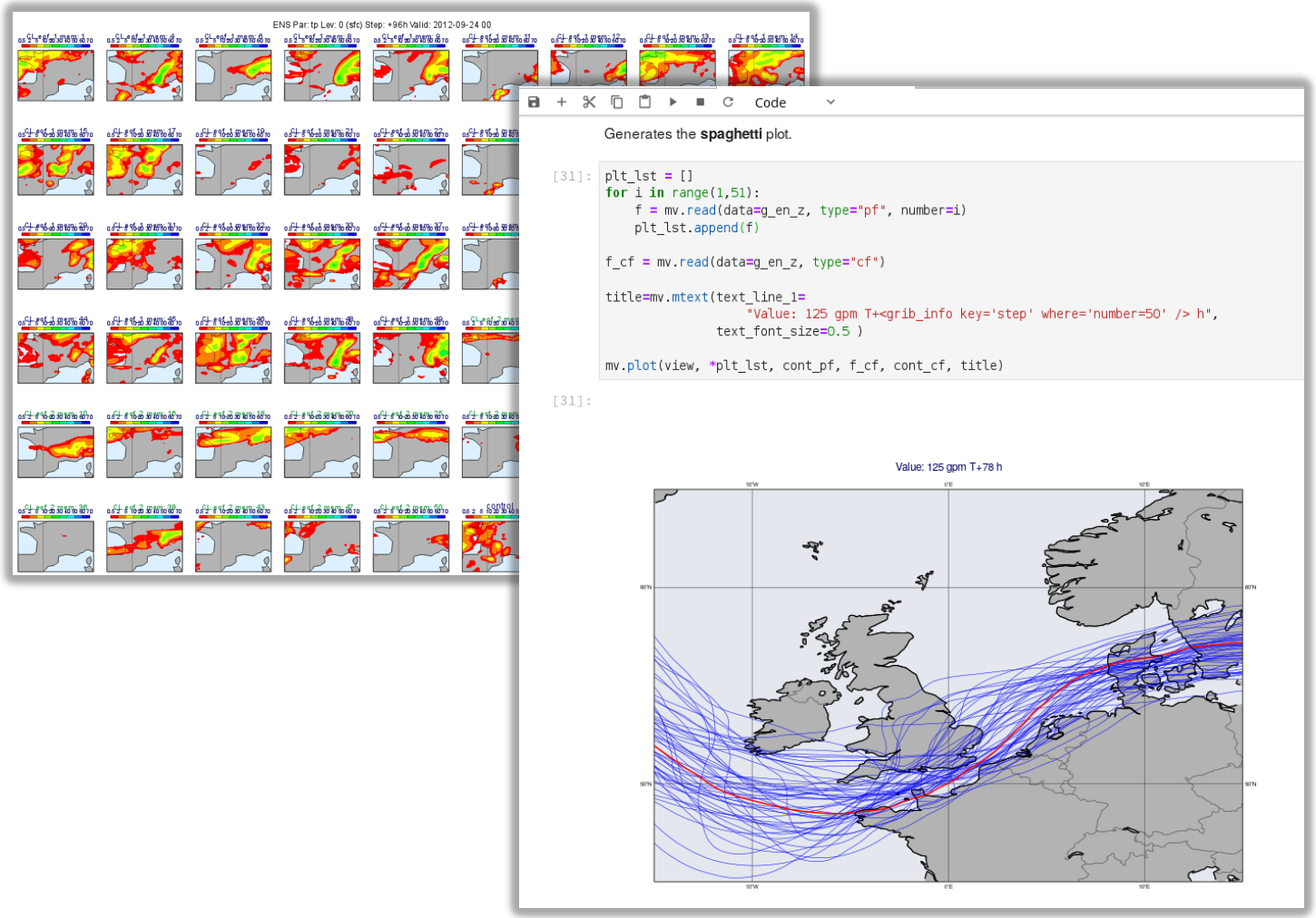
Processing and Visualizing ECMWF Ensemble Data

Webinar - May 12, 2020

Sándor Kertész
Iain Russell

Development Section, ECMWF



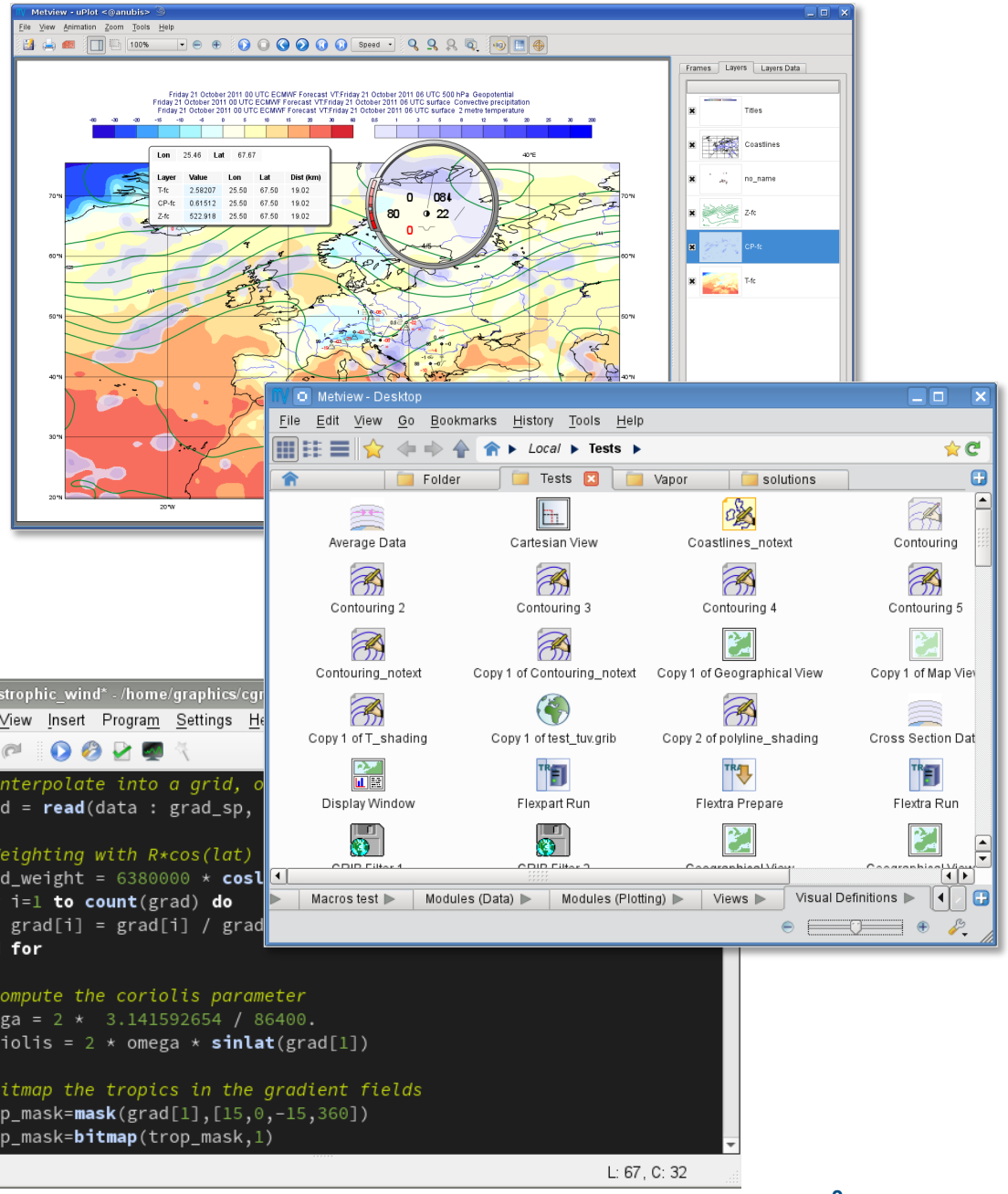
Outline

The webinar will be based on Jupyter notebooks
(Using Metview's Python interface)

- Plotting basics
- Introduction to ensemble data processing and plotting
  - Ensemble mean and spread
  - Stamp plots
  - Spaghetti plots
  - Probabilities and percentiles
  - CDF curves
- Where to find out more

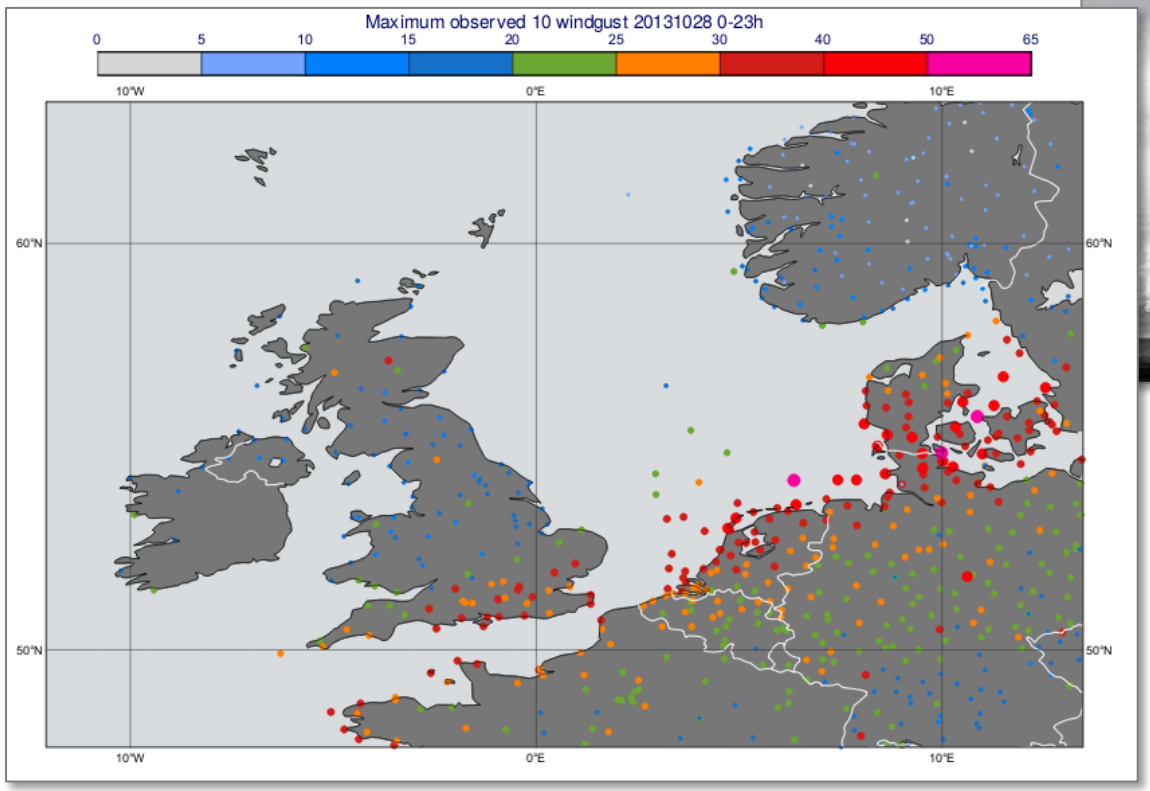
What is Metview?

- Workstation software, runs on UNIX, from laptops to supercomputers (including Mac OS X)
- Open source
- Visualisation
- Data processing
- Icon based user interface
- Powerful scripting languages (Macro and Python)



The forecast data we use

St Jude storm, 2013 October 28



- All data is in **GRIB** format:
  - Deterministic forecast
  - Ensemble forecast (ENS)
- Retrieved from the **MARS** archive and post-processed with Metview

We start with showing how to build a plot using the deterministic forecast

Step 1: Getting the data

grib_get()

to access **ecCodes** keys from the GRIB header.

Jupyter Notebook

```
import metview as mv
```

Reads the data into a **fieldset**.

```
g_fc = mv.read('fc_storm.grib')
```

Check field **metadata**.

```
mv.grib_get(g_fc, ['date', 'time', 'shortName', 'level', 'step'])
```

```
[['20131025', '0000', '10fg3', '0', '72'],
 ['20131025', '0000', '10fg3', '0', '78'],
 ['20131025', '0000', '10fg3', '0', '84'],
 ['20131025', '0000', 'z', '850', '78']]
```

Filter the data to work with.

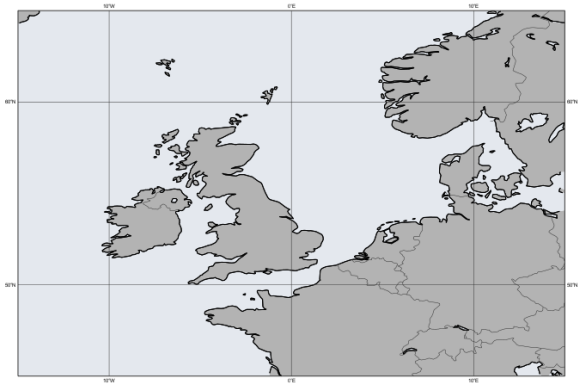
```
fc = mv.read(data=g_fc, param="10fg3", step=78)
```

Step 2: Defining contour, legend and title settings

Define **contour shading**, **legend** and **title** for wind gust.

```
wgust_shade = mv.mcont(
    legend                          = "on",
    contour_line_colour             = "navy",
    contour_highlight               = "off",
    contour_level_selection_type    = "level_list",
    contour_level_list              = [15,20,25,30,35,40,50],
    contour_label                   = "off",
    contour_shade                   = "on",
    contour_shade_colour_method     = "list",
    contour_shade_method            = "area_fill",
    contour_shade_colour_list       = ["sky","greenish_blue","avocado",
                                       "orange","orangish_red","violet"]
    )

legend = mv.mlegend(legend_text_font_size = 0.35)

title = mv.mtext(text_font_size = 0.5)
```

Step 3: Defining the map



Defines **coastlines** settings.

```
coast = mv.mcoast(
    map_coastline_land_shade        = "on",
    map_coastline_land_shade_colour = "grey",
    map_coastline_sea_shade         = "on",
    map_coastline_sea_shade_colour  = "RGB(0.8944,0.9086,0.933)",
    map_coastline_thickness         =  2,
    map_boundaries                  = "on",
    map_boundaries_colour           = "charcoal",
    map_grid_colour                 = "charcoal",
    map_grid_longitude_increment    = 10
    )
```

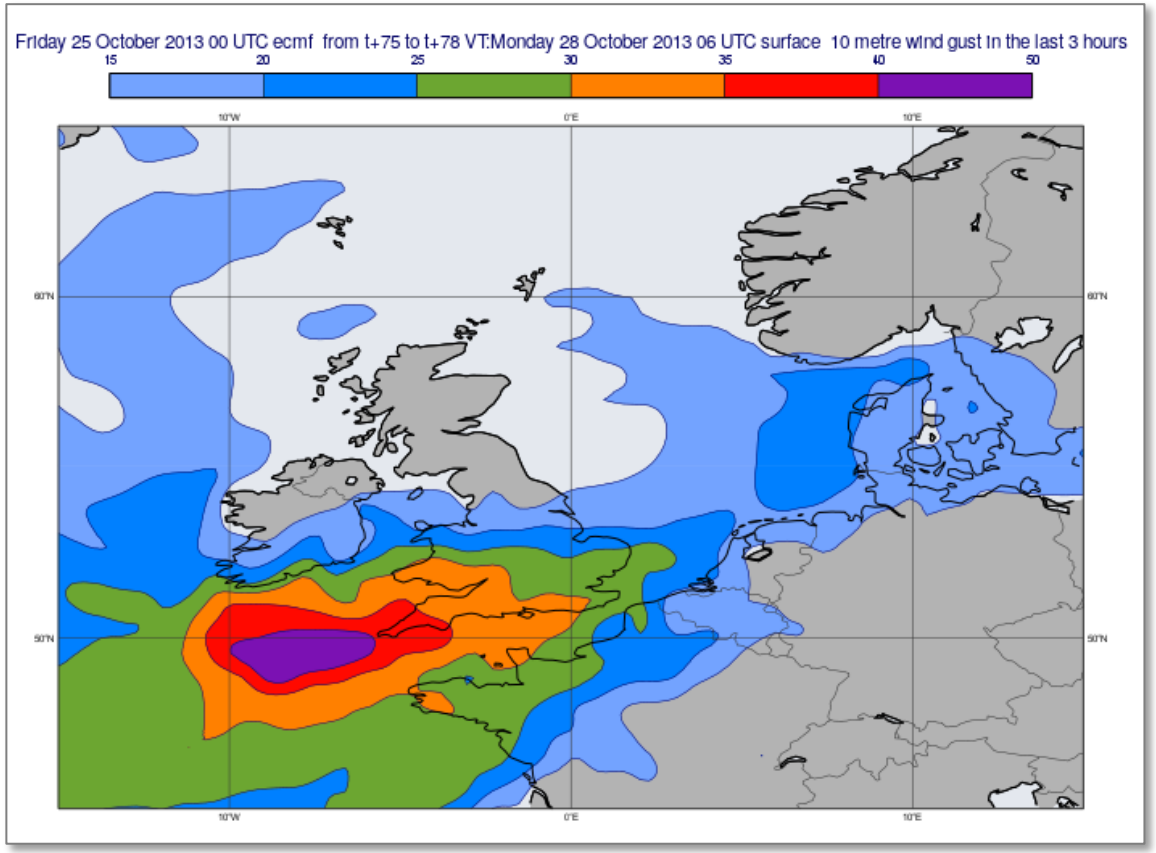
Creates a **map view**.

```
view = mv.geoview(
    map_area_definition = 'corners',
    area = [45,-15,65,15],
    coastlines = coast
    )
```

Step 4: Generating the plot

Tells Metview to generate a plot inside the **Jupyter** notebook

```
mv.setoutput('jupyter')

mv.plot(view, fc, wgust_shade, legend, title)
```



Icon functions

Many functions represent **icons** from the user interface. There they can be edited, dropped into a plot etc.

Grib Filter

mv.read(…)

Coastlines

mv.mcoast(…)

Contouring

mv.mcont(…)

Geographical view

mv.geoview(…)

Legend

mv.legend(…)

Text plotting

mv.mtext(…)

The ENS data

Two kinds of ENS members

Control forecast

type = **cf**

Perturbed forecast

type = **pf**

number = *specifies the ENS member (1-50 in ENS)*

Reads and **filters** the ENS data.

```
en = mv.read(source="ens_storm.grib", param="10fg3", step=78)
```

Checks field **metadata**.

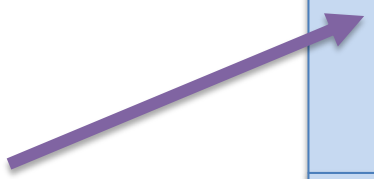
```
mv.grib_get(en, ['date', 'time', 'type', 'number',
                 'shortName', 'step'])[:8]
```

```
[['20131025', '0000', 'cf', '0', '10fg3', '78'],
 ['20131025', '0000', 'pf', '1', '10fg3', '78'],
 ['20131025', '0000', 'pf', '2', '10fg3', '78'],
 ['20131025', '0000', 'pf', '3', '10fg3', '78'],
 ['20131025', '0000', 'pf', '4', '10fg3', '78'],
 ['20131025', '0000', 'pf', '5', '10fg3', '78'],
 ['20131025', '0000', 'pf', '6', '10fg3', '78'],
 ['20131025', '0000', 'pf', '7', '10fg3', '78']]
```

Making a 2x2 ensemble plot

This is what we want to achieve

We can use what we already created



Deterministic forecast	Control forecast
Ensemble mean	Ensemble spread

The control forecast

Filter data using the **type** key

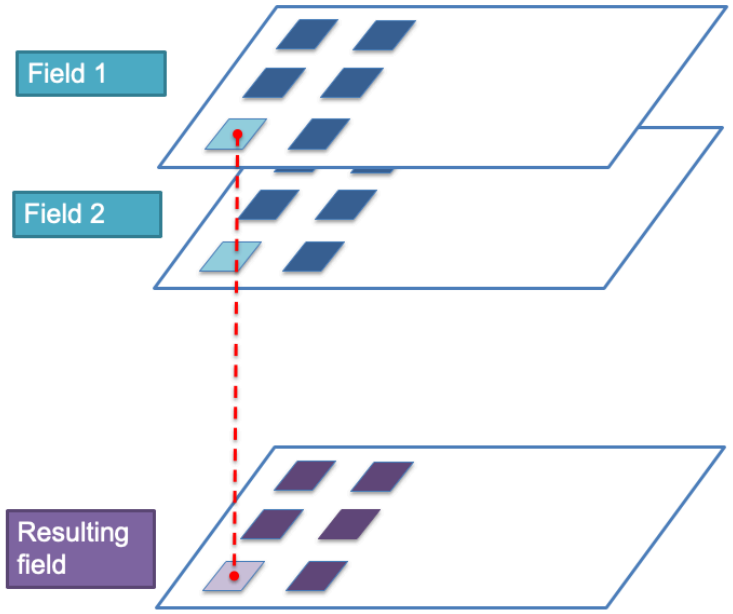
Reads the control forecast and creates a **title** for it.

```
cf = mv.read(data=en, type="cf")

cf_title = mv.mtext(text_lines=["CF"],
                    text_font_size=0.4)
```

The ensemble mean

mean()

computes the mean in each grid point, the result is a new fieldset



Computes the ensemble mean and creates a **title** for it.

```
e_mean = mv.mean(en)

mean_title = mv.mtext(text_lines=["ENS Mean"],
                      text_font_size=0.4)
```

The ensemble spread

stdev()

computes the **standard deviation** (=spread) in each grid point

We need different contour settings for the standard deviation

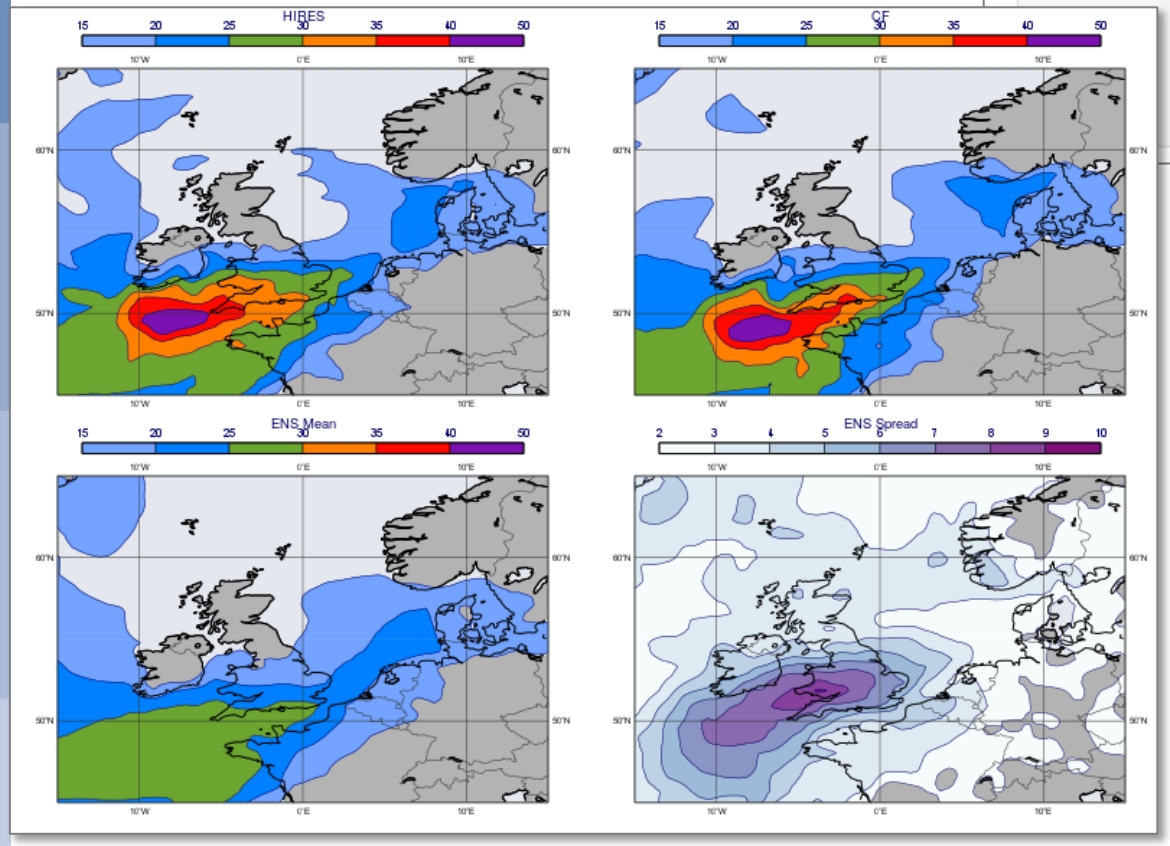
Computes the ensemble spread and defines **contruring** and **title** for it.

```
e_spread = mv.stdev(en)

spread_shade = mv.mcont(
    legend                          = "on",
    contour_line_colour             = "navy",
    contour_highlight               = "off",
    contour_level_selection_type = "level_list",
    contour_level_list              = [2,3,4,5,6,7,8,9,10],
    contour_label                   = "off",
    contour_shade                   = "on",
    contour_shade_colour_method  = "palette",
    contour_shade_method            = "area_fill",
    contour_shade_palette_name   = "m_purple_9"
    )

spread_title = mv.mtext(text_lines=["ENS Spread"],
                        text_font_size=0.4)
```

Putting it all together

Defines a 2x2 plot **layout** and generates the **plot**.

```
dw = mv.plot_superpage(pages = mv.mvl_regular_layout(view,2,2,1,1))

mv.plot(dw[0], fc, wgust_shade,
        mv.mtext(text_lines=["HIRES"], text_font_size=0.4), legend,
        dw[1], cf, wgust_shade, cf_title, legend,
        dw[2], e_mean, wgust_shade, mean_title, legend,
        dw[3], e_spread, spread_shade, spread_title, legend)
```



For a given plot in Jupyter we combine everything in a **single plot command**!

Making a stamp plot

Stamp plot

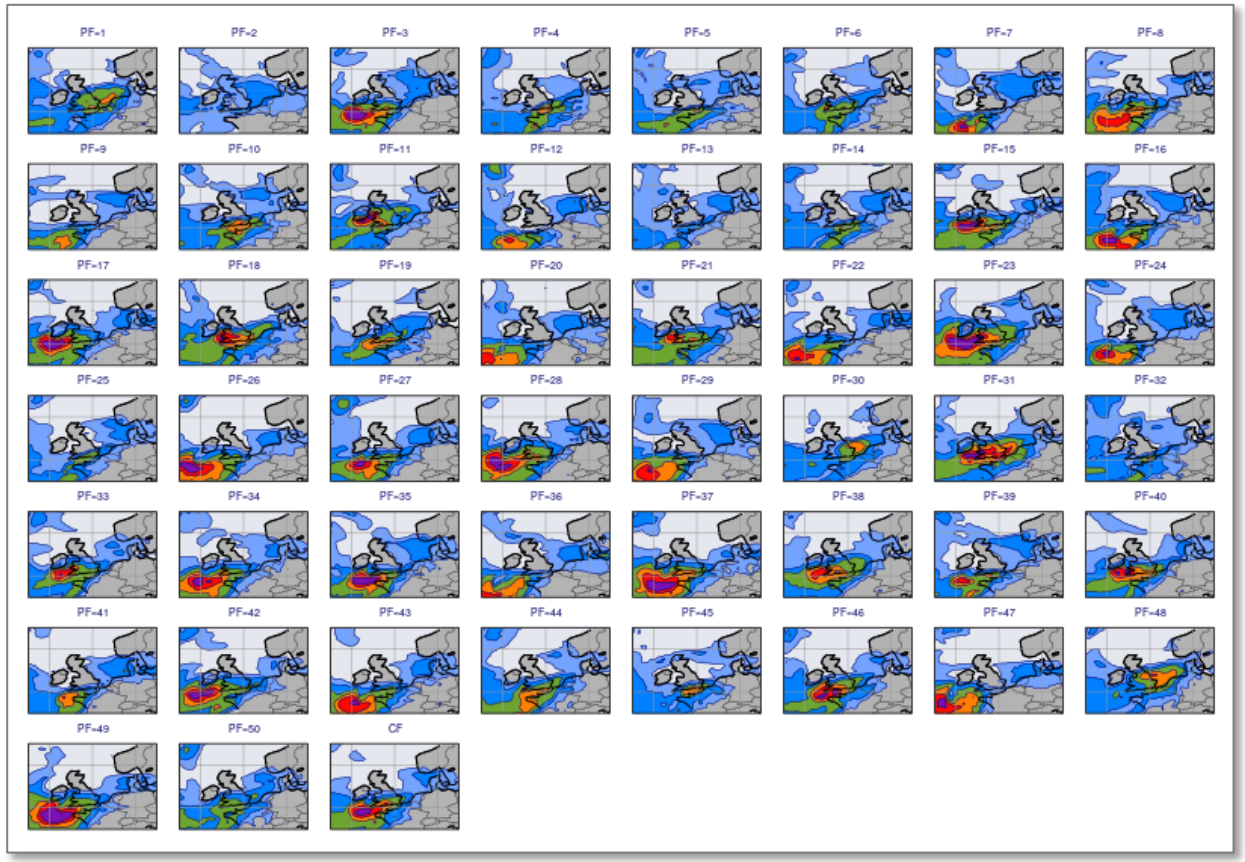
We plot each ENS member into a **different map** within the same plot

Creates a **layout** with 8 columns and 7 rows.

```
dw = mv.plot_superpage(
    pages = mv.mvl_regular_layout(view_stamp,8,7,1,1, [5,100,0,100]))

mv.plot(dw)
```

Stamp plot

We use a **list** to define the arguments to the **plot()** command




```
pl_lst = []

# perturbed forecasts
for i in range(1, 51):
    f = mv.read(data=en, type="pf", number=i)
    title = mv.mtext(
        text_lines=["PF=" + str(i)],
        text_font_size=0.3)
    pl_lst.append(
        [dw[i-1], f, wgust_shade, title])

# control forecast
cf = mv.read(data=en, type="cf")
title = mv.mtext(text_lines=["CF"],
                 text_font_size=0.3)
pl_lst.append(
    [dw[50], f, wgust_shade, title])

mv.plot(pl_lst)
```

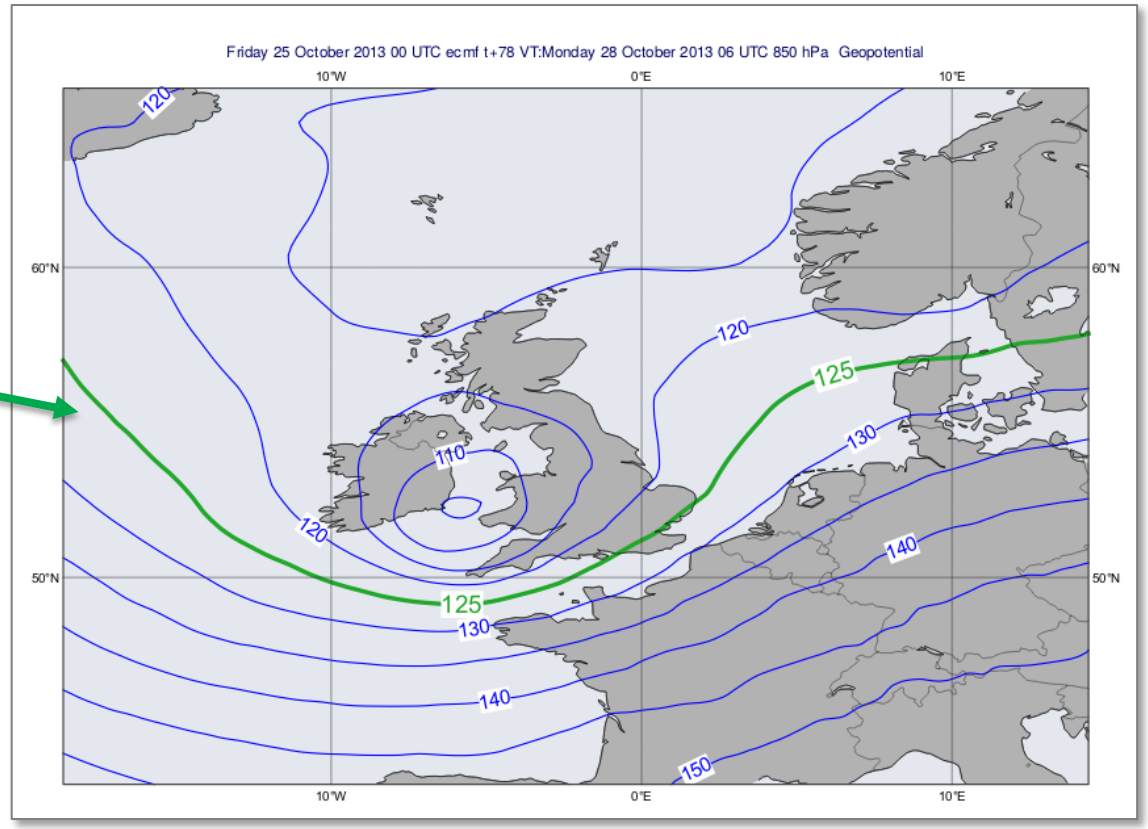
Making a spaghetti plot

Spaghetti plot

Pick an **isoline** value and plot only this isoline for **all the ENS members** into the same map

We will select the **125 dkm** line, good indicator for the position of the trough

Deterministic 850 hPa Geopotential forecast



Making a spaghetti plot

Blue contour for the perturbed members

Thick red line for the control forecast

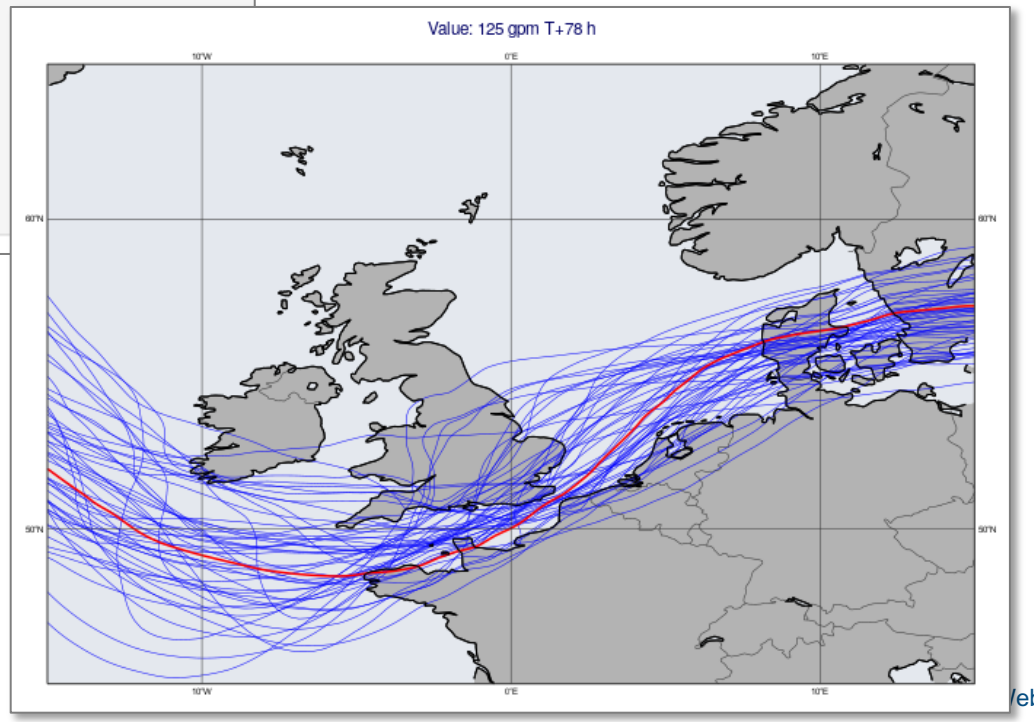
Reads the 850 hPa geopotential ENS data and defines **contours** for the spaghetti plot.

```
en_z = mv.read(source="ens_storm.grib", param="z", levelist=850, step=78)

cont_pf = mv.mcont(
    contour_label="off",
    contour_level_selection_type="level_list",
    contour_level_list=125,
    contour_line_colour="blue",
    contour_highlight="off",
    grib_scaling_of_derived_fields="on"
 )

cont_cf = mv.mcont(
    cont_pf,
    contour_line_colour="red",
    contour_line_thickness=3
)
```

Making a spaghetti plot

```
pf = []
for i in range(1,51):
    f = mv.read(data=en_z, type="pf", number=i)
    pf.append(f)

cf = mv.read(data=en_z, type="cf")

title=mv.mtext(
    text_line_1=
        "Value: 125 gpm T+<grib_info key='step' where='number=50' /> h",
    text_font_size=0.5 )

mv.plot(view,
        pf, cont_pf,
        cf, cont_cf, title)
```

This prevents plotting the title 51 times!



Computing the ENS probability

Computes the probabilty of having wind gust > 28 m/s (~100 km/h).

```
prob = en > 28
prob = mv.mean(prob) * 100
```

Step1 = Masking

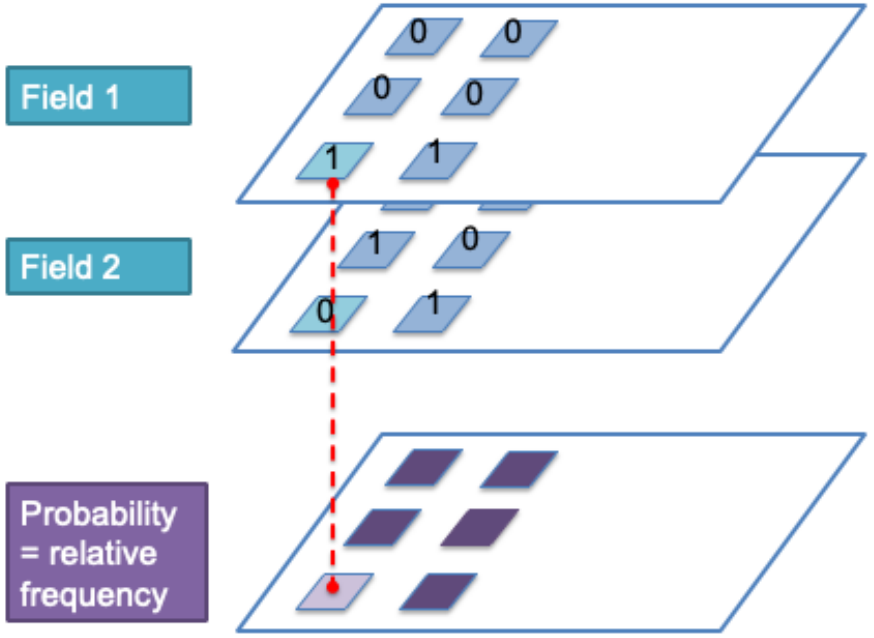
Logical operation on a field turning values into 0s and 1s

prob = en > 28

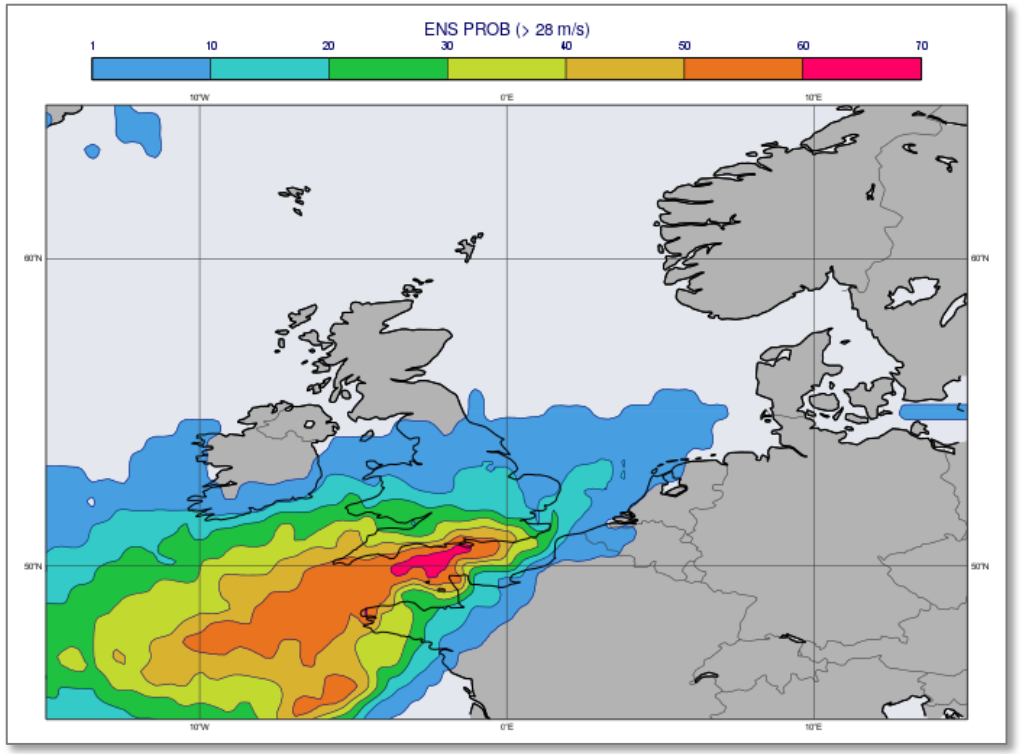
In the resulting fieldset (51 fields) all points with values greater than 28 will be 1s, while all other points will be 0s

Step 2 = Aggregation

The probability is simply the pointwise **mean** of these fields



Plotting the ENS probability



Plots the probability with custom **contour shading** and **title**.

```
prob_shade = mv.mcont(
    legend                          = "on",
    contour_line_colour             = "navy",
    contour_highlight               = "off",
    contour_level_selection_type    = "level_list",
    contour_level_list              = [1, 10, 20, 30, 40, 50, 60, 70],
    contour_label                   = "off",
    contour_shade                   = "on",
    contour_shade_colour_method     = "palette",
    contour_shade_method            = "area_fill",
    contour_shade_palette_name      = "eccharts_rainbow_blue_red_7"
    )

prob_title = mv.mtext(text_lines=["ENS PROB (> 28 m/s)"],
                      text_font_size=0.5)

mv.plot(view, prob, prob_shade, legend, prob_title)
```

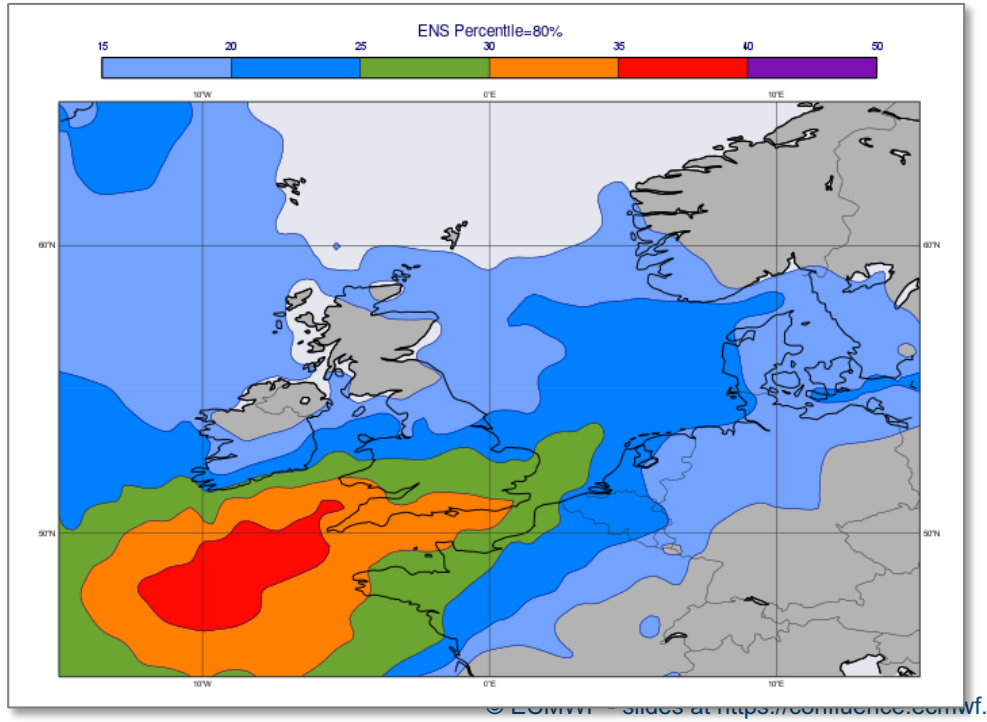
Plotting ENS percentiles maps

ENS percentile map

E.g. 80% percentile = the forecast value below which 80% of the ENS members fall

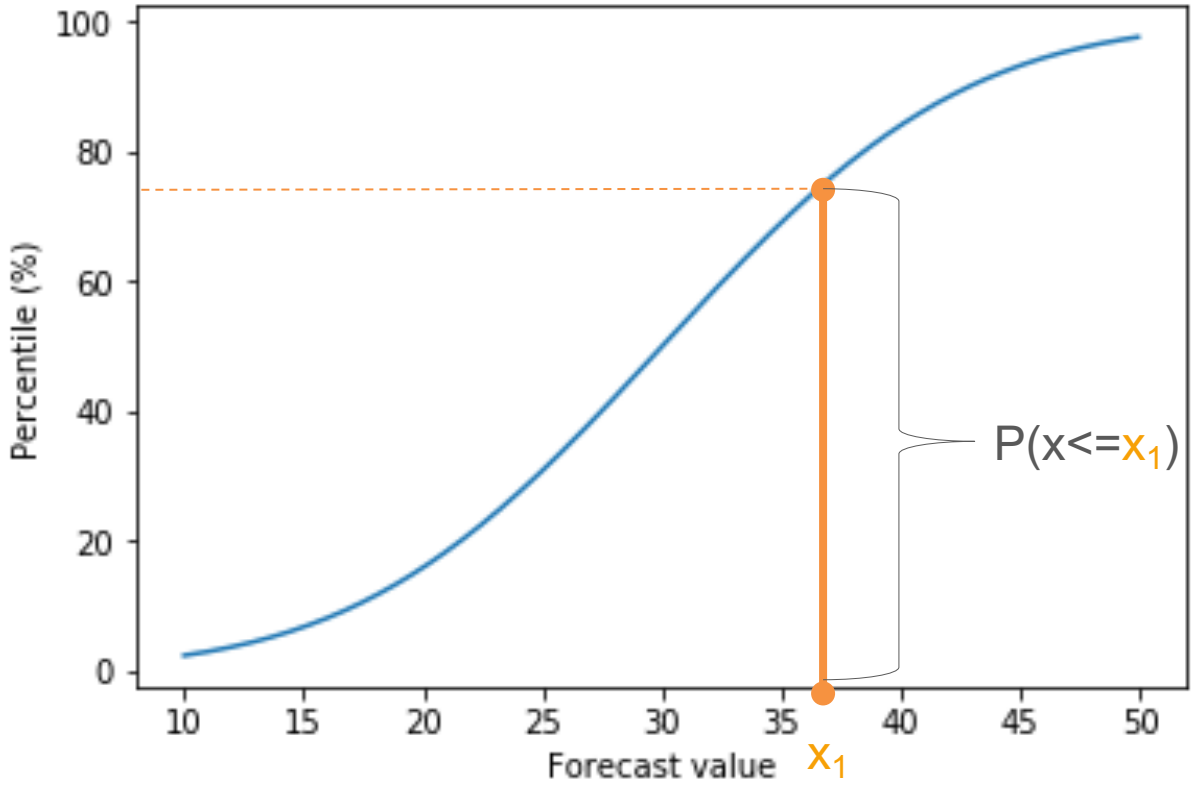
Generates plot for 80% **percentile** of wind gust.

```
perc = mv.percentile(data=en, percentiles=80)

perc_title = mv.mtext(text_lines=["ENS Percentile=80%"],
                      text_font_size=0.5)

mv.plot(view, perc, wgust_shade, legend, perc_title)
```



Making a CDF plot (for a location)

CDF = Cumulative Distribution Function



Probability that the forecast <= x1

Extracting values for a given location

nearest_gridpoint()

The result is a **numpy** array

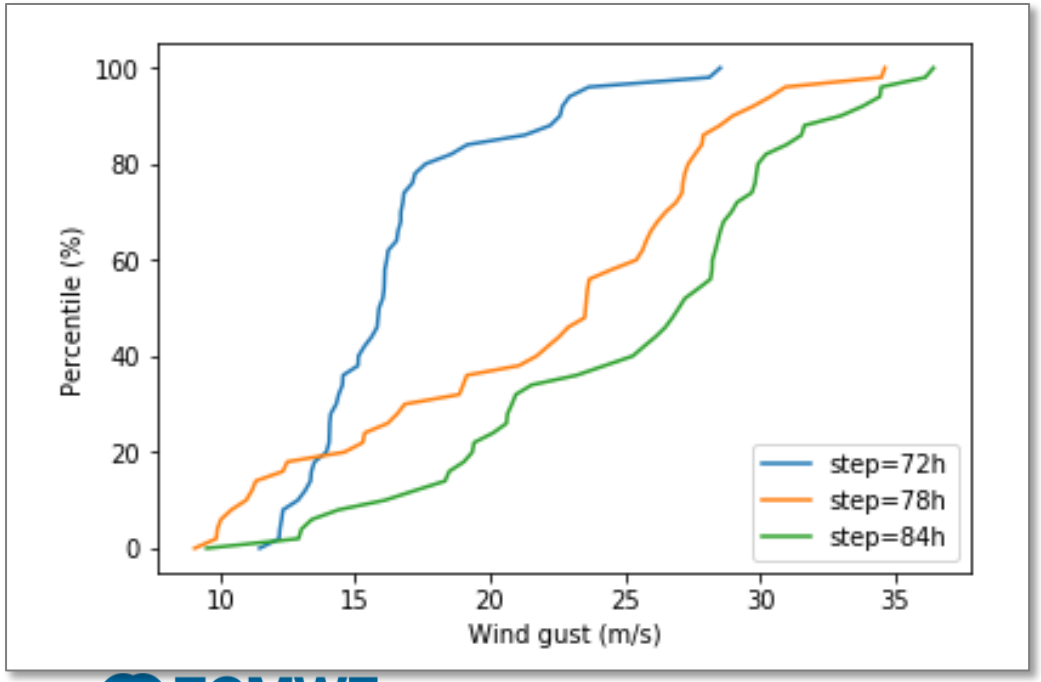

```
pos = [51.5, -1]
f = mv.read(source="ens_storm.grib", param="10fg3", step=78)
x = mv.nearest_gridpoint(f, pos)
x
```

```
[27.21945285797119,
 29.75399875640869,
 16.206149101257324,
 26.90046215057373,
 22.559062004089355,
 10.962868690490723,
 27.9066162109375,
 9.893465042114258,
 23.510653495788574,
 9.05254077911377,
 30.948326110839844,
 23.543956756591797
```

Making a CDF plot: full example

Generating a CDF for three time steps

We use **matplotlib** for the plotting




```python
import numpy as np
import matplotlib.pyplot as plt

pos = [51.5, -1]
lines = []
for step in [72, 78, 84]:
    f = mv.read(source="ens_storm.grib", param="10fg3", step=step)
    x = mv.nearest_gridpoint(f, pos)

    # form cdf
    y = np.arange(0, 101)
    x = np.percentile(x, y)

    # make line plot object
    line, = plt.plot(x, y, label="step={}h".format(step))
    lines.append(line)

plt.legend(handles=lines, loc='lower right')
plt.xlabel('Wind gust (m/s)')
plt.ylabel('Percentile (%)')
plt.show()
```

Where to find out more

Function documentation

ECMWF Spaces Calendars Create ... Search 9+

- Change History
- User Guide
  - Using Metview
  - The Macro Language
    - Macro syntax
    - Macro Data Types
    - List of Operators an
      - Information Functi
      - The nil Operand
      - Number Functions
      - String Functions
      - Date Functions
      - List Functions
      - Vector Functions
      - **Fieldset Functions**
      - Geopoints Functio
      - Geopointset Funct
      - NetCDF Functions
      - ODB Functions
      - Table Functions
      - Observations Func
      - Definition Functior
      - File I/O Functions
      - Timing Functions

Space tools

`fieldset` **`geostrophic_wind_pl`** `(z: fieldset)`

Computes the geostrophic wind from geopotential fields defined on pressure levels. For a given `z` geopotential field the computation of the geostrophic wind components is based on the following formulas:

$$u_g = -\frac{1}{f}\frac{1}{R}\frac{\partial z}{\partial \phi}$$

$$v_g = \frac{1}{f}\frac{1}{R\,cos\phi}\frac{\partial z}{\partial \lambda}$$

where

- `R` is the radius of the Earth
- `φ` is the latitude
- `λ` is the longitude
- `f=2Ωsinφ` is the Coriolis parameter, where Ω is the Earth's angular velocity.

The derivatives are computed with a second order finite-difference approximation. The resulting fieldset contains two fields for each input field: the u and v geostrophic wind components. In each output field the points close to the poles and the Equator are bitmapped (they contain missing values). Please note that this function is only implemented for regular latitude-longitude grids.

`geopoints` **`gfind`** `( fieldset,number )`
`geopoints` **`gfind`** `( fieldset,number,number )`

A filtering function that returns a geopoints holding the grid points whose value is equal to the *value* of the first number. Missing values in the input field are not returned. If a second number is given as the third argument it is a tolerance *threshold* and the geopoints will hold the grid points for which :

```
abs(data-value) <= threshold
```

Where to find out more

Icon reference

ECMWF Spaces Log in

- User Guide
  - Using Metview
  - The Macro Language
  - Metview's Python Interface
  - **Icon Reference**
    - Annotation View
    - Average Data
    - Average View
    - Axis Plotting
    - Binning
    - Bufr Picker
    - Cartesian View
    - Clean File
    - Coastlines
    - Common View Paramete
    - Contouring
    - Cross Section Data
    - Cross Section View
    - Display Window
    - Download from URL
    - ECCHARTS
    - ECFS

Space tools

Data access icons

Download from URL | ECCHARTS | ECFS | FLEXPART Prepare | FLEXTRA Prepare | MARS Retrieval | Met3D Prepare | Stations

VAPOR Prepare | WMS Client

Data filter icons

Bufr Picker | Clean File | GRIB Filter | ODB Filter | Observation Filter | Table Reader

Data processing icons

Average Data | Cross Section Data | FLEXPART Release | FLEXPART Run | FLEXTRA Run | Formula | Grib To Geopoints | Geopoints To Grib

Geopoints To KML | Hovmoeller Data | Percentile | Potential Temperature | Relative Humidity | Reprojection | Rotational or Divergent Wind | RTTOV Run

ECMWF

Where to find out more

Gallery

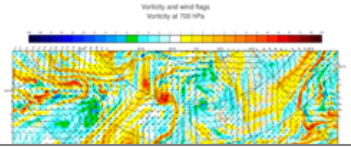
ECMWF Spaces

Search

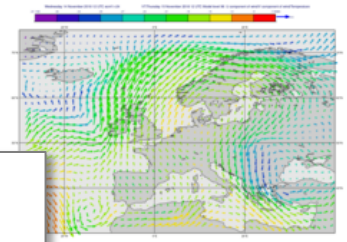
Log in

- Change History
- User Guide
- FAQ
- Training
- **Gallery**
  - Python Jupyter Notebooks
  - OpenIFS Workshop 2016

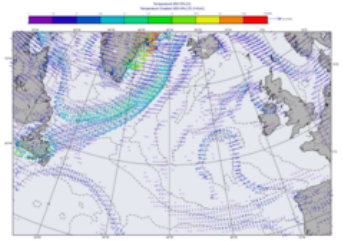
Vorticity and Wind Example
GRIB, Polar Stereographic



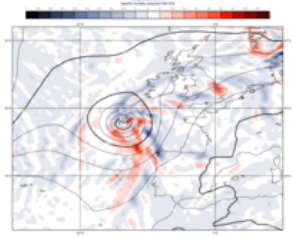
Wind Coloured By Temperature Example
GRIB, Cylindrical



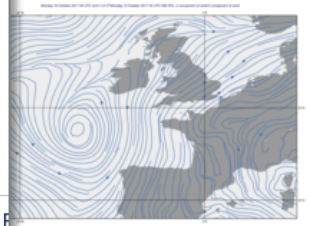
Temperature Gradient Vector Example
GRIB, Polar Stereographic



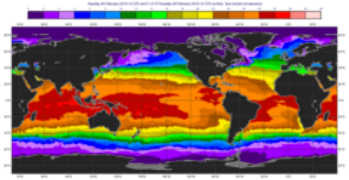
Humidity advection Example
GRIB, Cylindrical



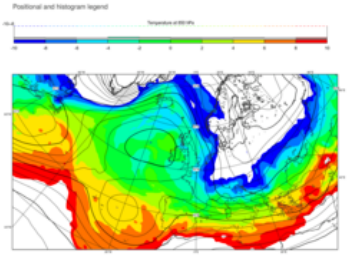
lines Example
ndrical



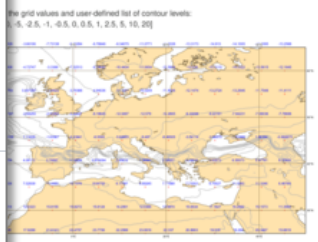
SST on Extended Cylindrical Map Example
GRIB, Cylindrical



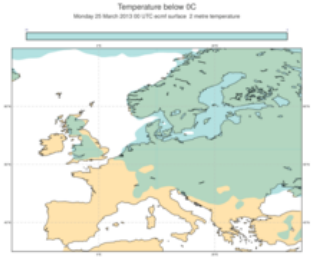
Histogram Legend Example
GRIB, Polar Stereographic



alues and Contour Levels Example
ndrical



Temperature Below 0C Example
GRIB, Cylindrical



Boundaries, Cities and Rivers Example
Cylindrical

ontouring Example
ndrical

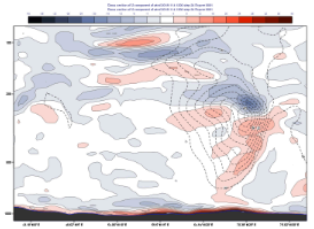
Model-Obs Difference Example
GRIB, BUFR, Cylindrical

BUFR Synop Example
BUFR, Cylindrical

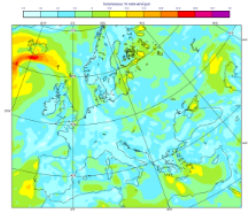
Python Jupyter Notebooks

Created by Milana Vuckovic, last modified by Sandor Kertesz about 6 hours ago

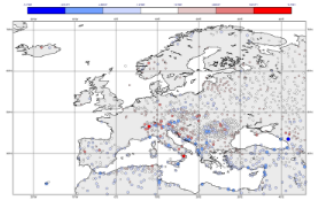
Vertical cross section - Wind shear



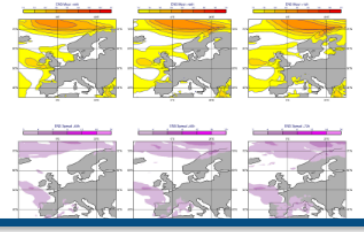
NetCDF from CDS



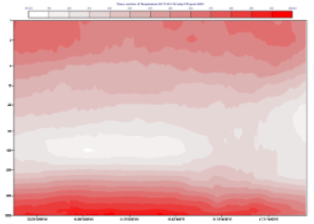
Forecast/Observations difference



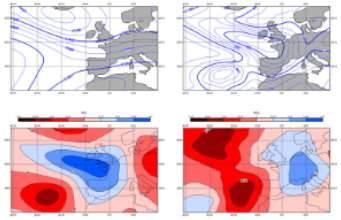
Using Xarray for computing



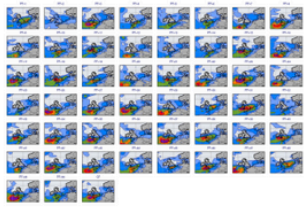
Vertical cross section - CDS data



Principal component analysis



Computing and plotting ensemble data

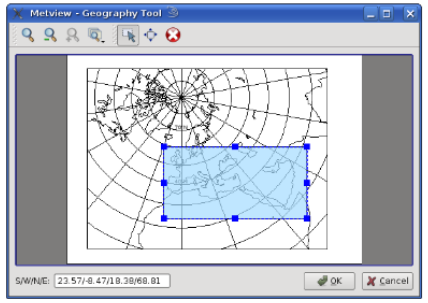


Where to find out more

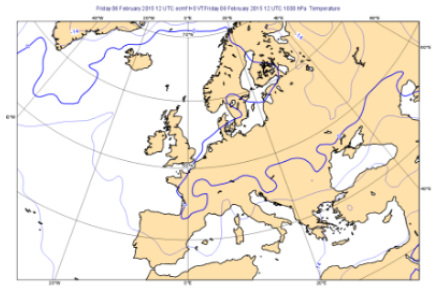
Lots of material online including tutorials

Now we want to set the area used in the view. Although we can interactively zoom into smaller areas in the **Display Win** use exactly the same one again and again. Set the **Map Area Definition** to Corners and click on the **Geography Tool** but

This tool helps you define a region.

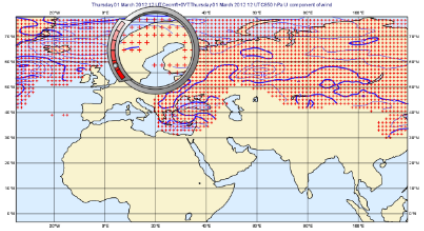


Use the **Zoom** tools to enlarge the European area and use the **Area** tool to select a region over Europe. Click **Ok** to save *Geographical View* editor. Click **Apply** in the *Geographical View* editor to save everything. Plot your data in this view to co



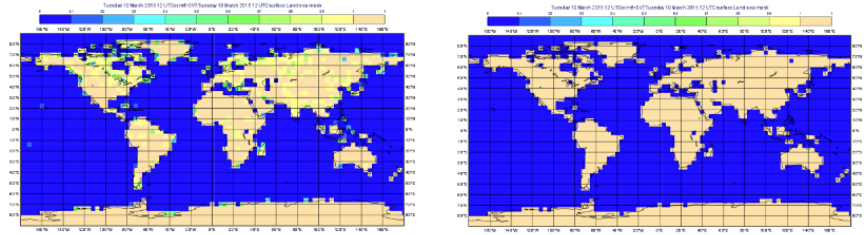
- A Quick Tour of Metview
- Data analysis and visualisation usi...
  - A Simple Visualisation
  - Customising Your Plot
  - Case Study: Plotting Hurricane S...
  - Data Part 1
  - Processing Data
  - Analysis Views
  - Layout in Metview
  - Case Study: Cross Section of Sa...
  - Data Part 2
  - Handling Time in Metview
  - Graph Plotting in Metview
  - Case study: Plotting the Track o...
  - 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

Overview



Fields and observations can often contain missing values - it can be important to understand the implications of the 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



As an example, we will use a land-sea mask field as the basis of performing a computation on only the land points, e

Visualise the supplied *land_sea_mask.grib* icon using the *grid_shade* icon. This *Contouring* icon is set up to shade the interpolation. To help illustrate what's going on, we've chosen low-resolution fields - this one is 4x4 degrees. The val between 0 and 1 on points which are close to both sea and land. Before we can use this field as a mask, we must do whether they count as land or sea! Let's say that a value of 0.5 or more is land.

Metview Availability – on ECMWF systems

- Versioned using the ‘module’ system

Interactive or batch session

```
module load python3
module swap metview/new
metview
```

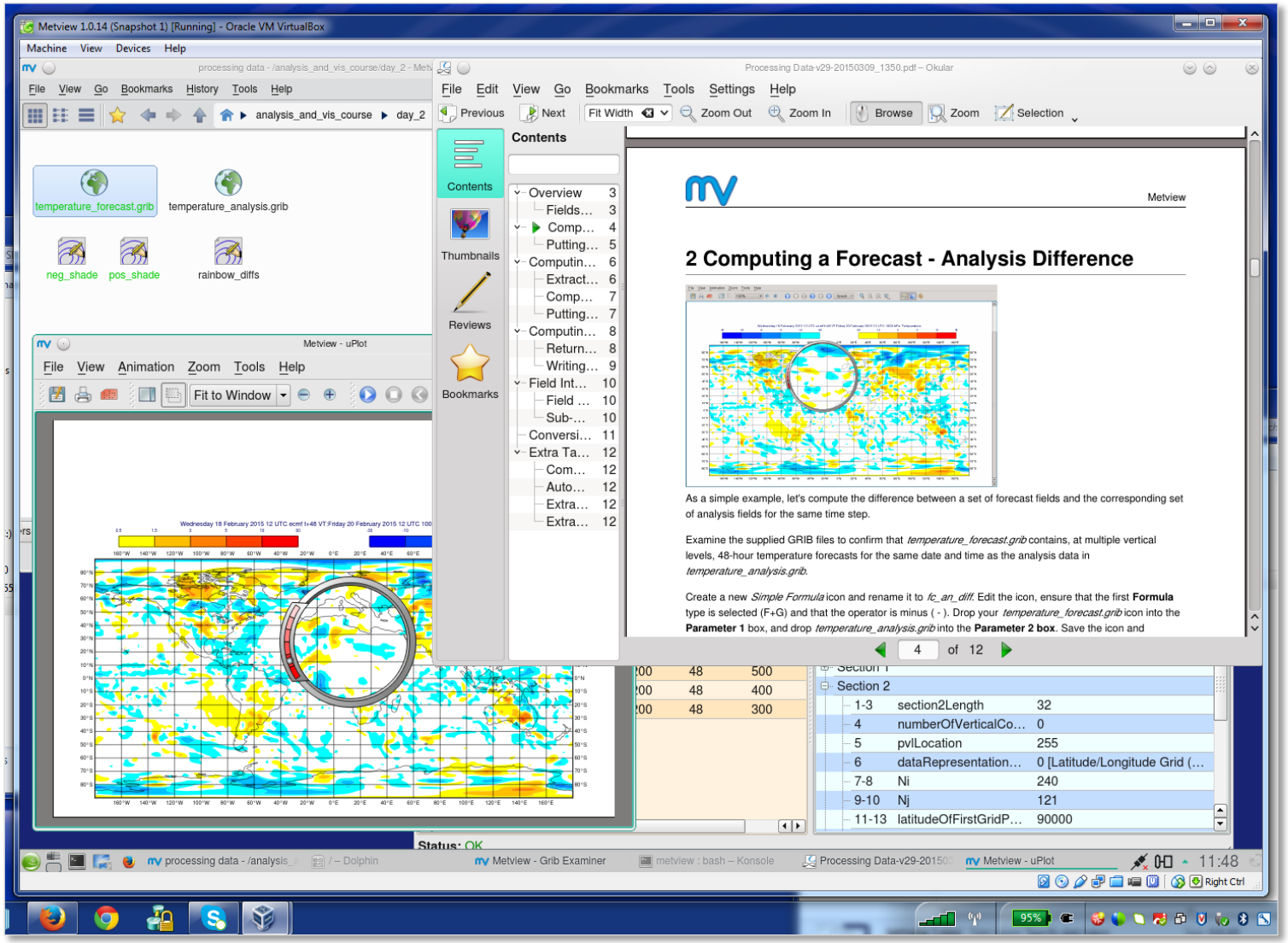
Jupyter notebook

```
module load python3
module swap metview/new
jupyter-notebook <path>
```

Metview availability – outside ECMWF

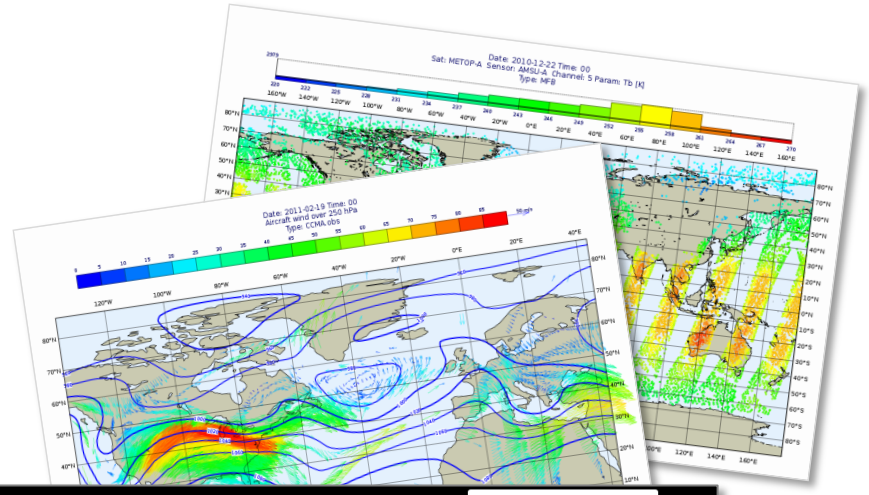
- Install from binaries
- Conda (via conda-forge)
- Build from source
- Build from bundle
- The Metview Python interface has to be installed separately:

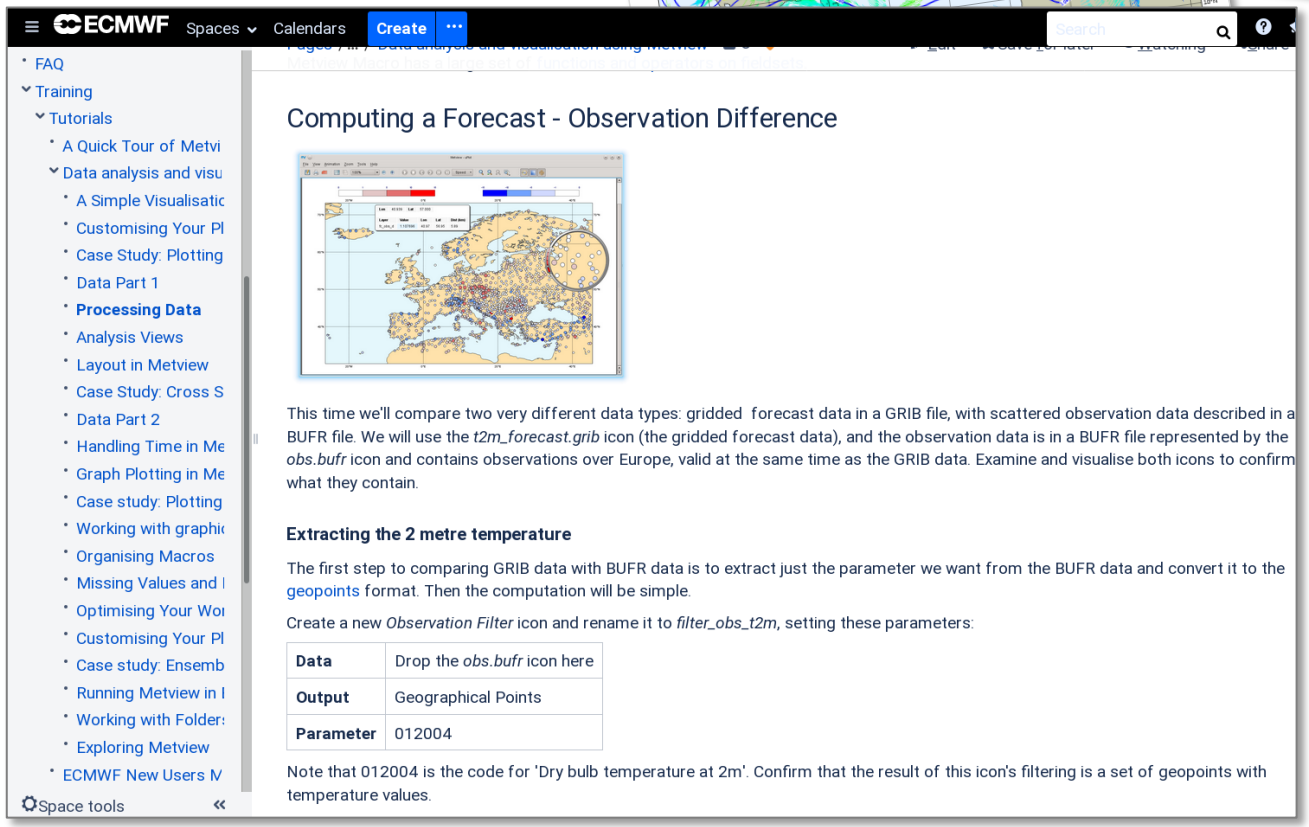
```
pip install metview
```



For more information…

- Ask for help:
  - Software.Support@ecmwf.int
- Visit our web pages:
  - http://confluence.ecmwf.int/metview





Questions?