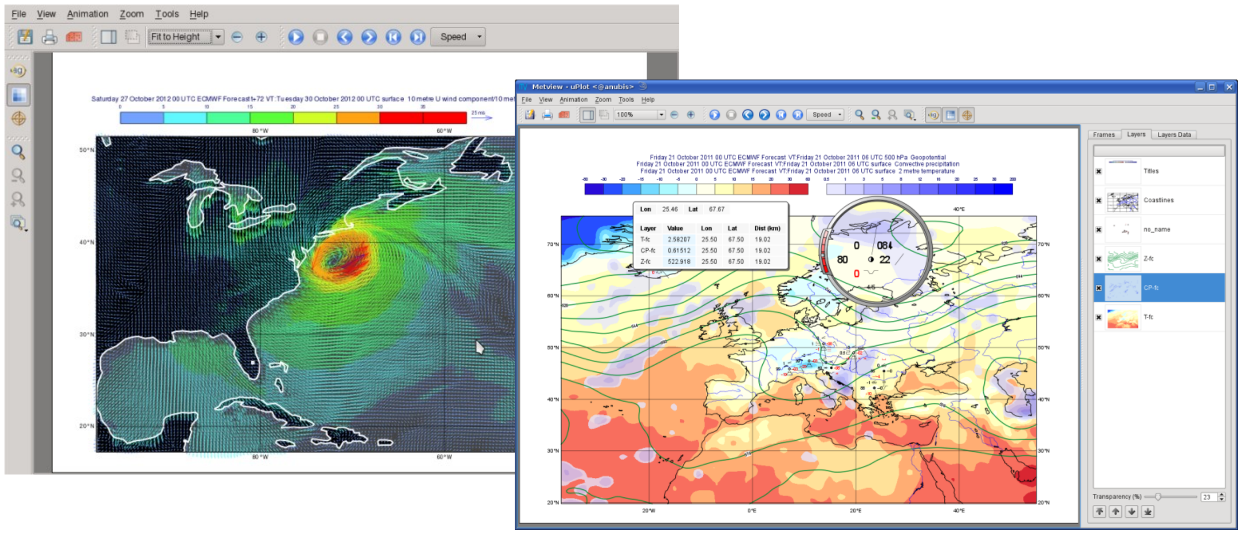
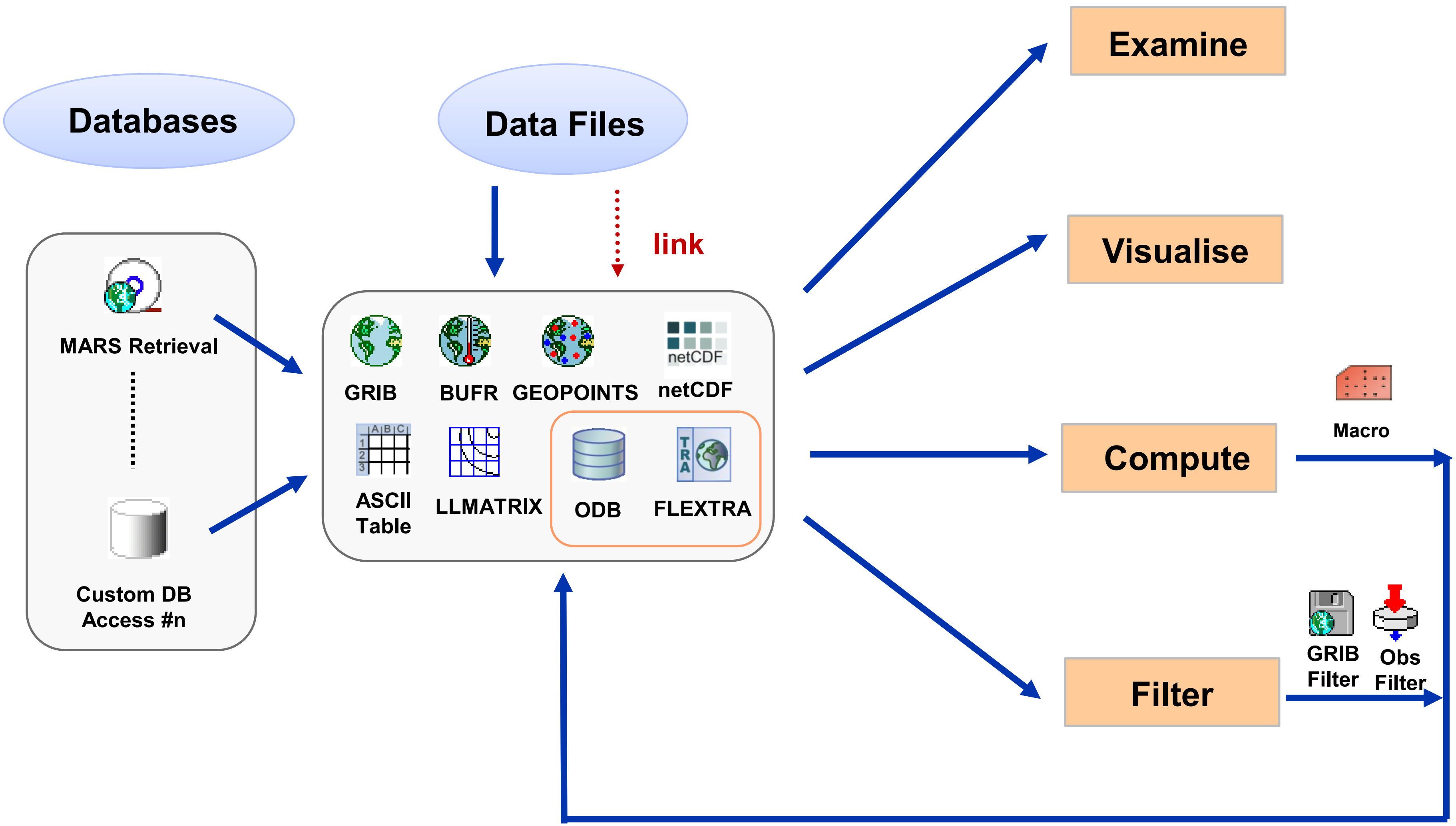
Data Handling with Metview



Sándor Kertész

Development Section
ECMWF

Data handling in Metview

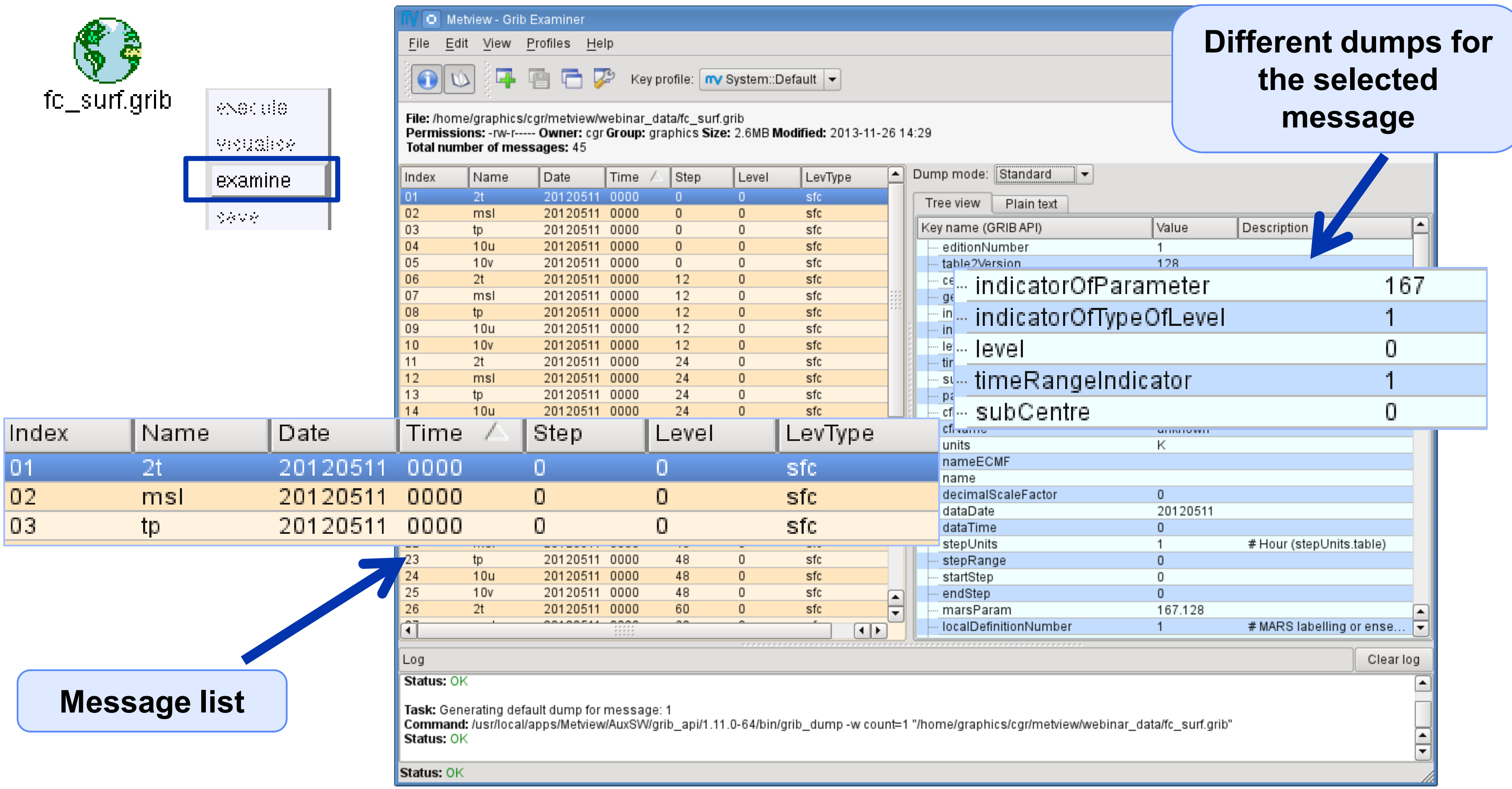


GRIB

- WMO's binary format for gridded data
- The Metview interface is based on GRIB API
- Access to both Edition 1 and 2 files

GRIB Examiner

- GRIBs contents can be checked with the **GRIB Examiner**



GRIB Examiner – Values dump

Dump mode: Values

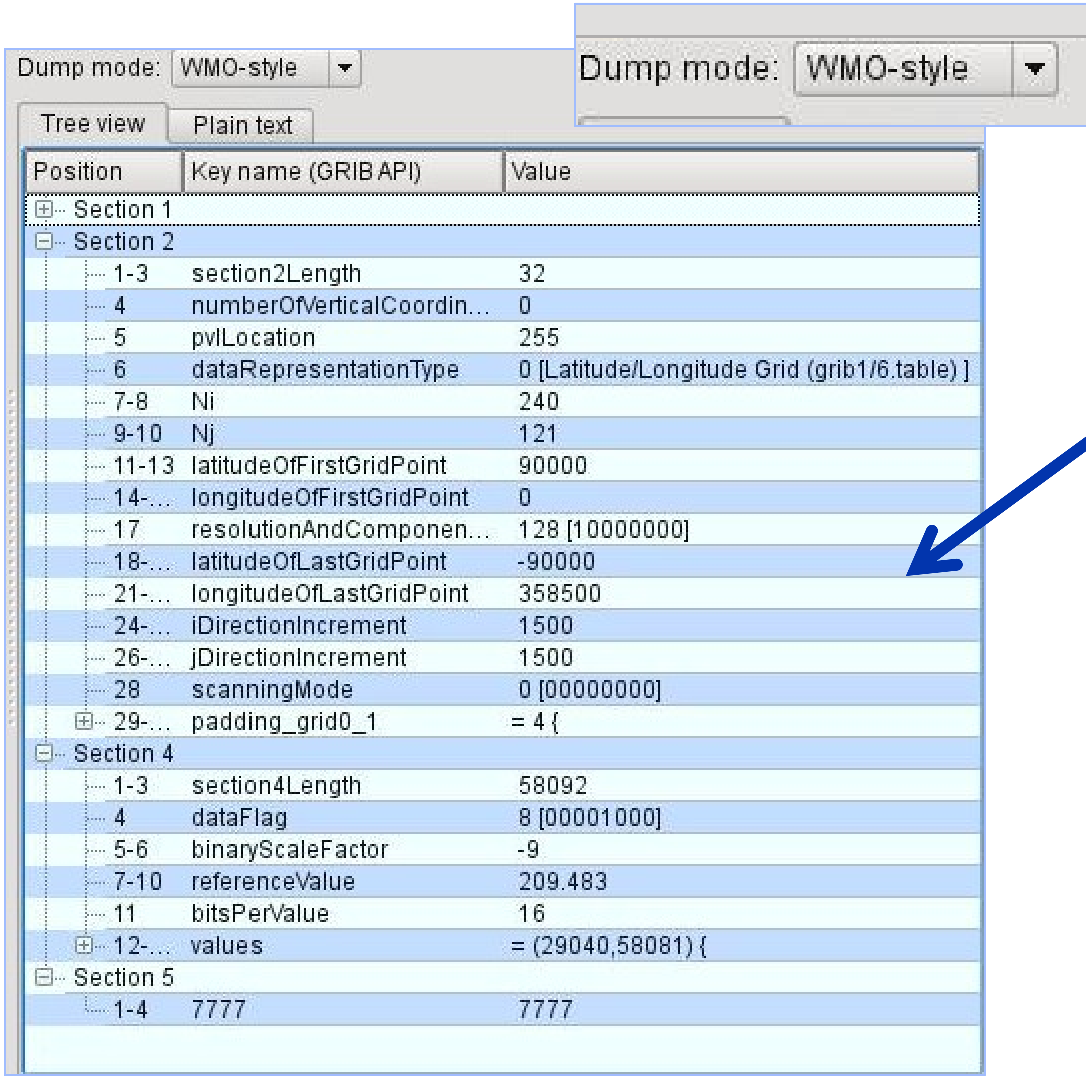
Go to row: 1 (Number of points: 29040)

Index	Latitude	Longitude	Value
10559	25.500	357.000	301.6919
10560	25.500	358.500	300.3052
10561	24.000	0.000	303.8774
10562	24.000	1.500	304.2954
10563	24.000	3.000	301.1665
10564	24.000	4.500	298.9282
10565	24.000	6.000	298.7759
10566	24.000	7.500	297.1509
10567	24.000	9.000	297.6567
10568	24.000	10.500	296.5220
10569	24.000	12.000	293.8872
10570	24.000	13.500	297.1079
10571	24.000	15.000	297.9028
10572	24.000	16.500	296.8403
10573	24.000	18.000	296.9438
10574	24.000	19.500	294.5200
10575	24.000	21.000	295.1958
10576	24.000	22.500	296.6899
10577	24.000	24.000	296.4712
10578	24.000	25.500	290.8188
10579	24.000	27.000	293.4263
10580	24.000	28.500	295.9556
10581	24.000	30.000	296.5669

Dump mode: Values

All the values for the selected message

GRIB Examiner – WMO-style dump


Dump mode: WMO-style

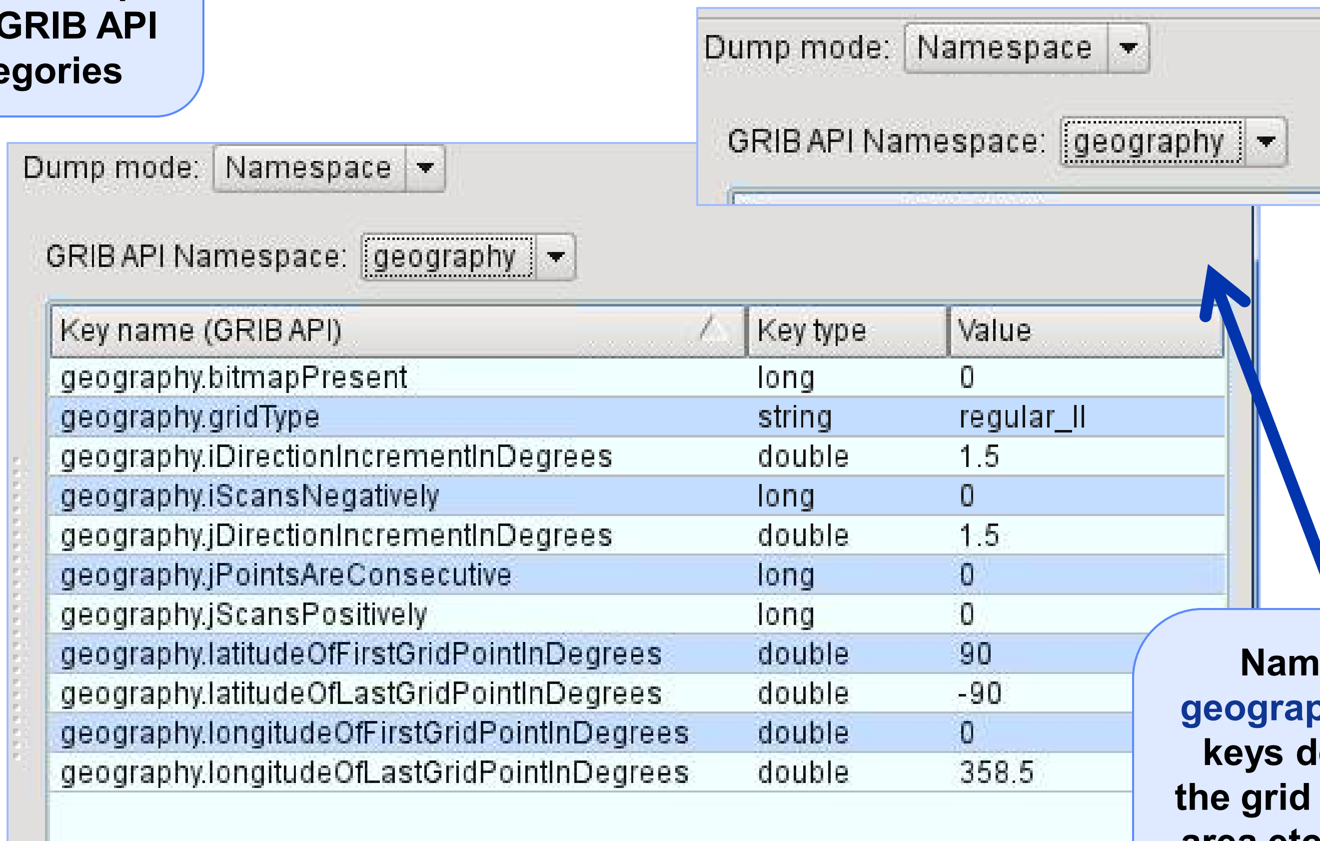
Tree view | Plain text

Position	Key name (GRIB API)	Value
Section 1		
Section 2		
1-3	section2Length	32
4	numberOfVerticalCoordin...	0
5	pvlLocation	255
6	dataRepresentationType	0 [Latitude/Longitude Grid (grib1/6.table)]
7-8	Ni	240
9-10	Nj	121
11-13	latitudeOfFirstGridPoint	90000
14-...	longitudeOfFirstGridPoint	0
17	resolutionAndComponen...	128 [10000000]
18-...	latitudeOfLastGridPoint	-90000
21-...	longitudeOfLastGridPoint	358500
24-...	iDirectionIncrement	1500
26-...	jDirectionIncrement	1500
28	scanningMode	0 [00000000]
29-...	padding_grid0_1	= 4 {
Section 4		
1-3	section4Length	58092
4	dataFlag	8 [00001000]
5-6	binaryScaleFactor	-9
7-10	referenceValue	209.483
11	bitsPerValue	16
12-...	values	= (29040,58081) {
Section 5		
1-4	7777	7777

Each section of the GRIB message is shown in a tree view

GRIB Examiner – Namespace dump

Namespace is a GRIB API concept to define GRIB API key categories


Dump mode: Namespace

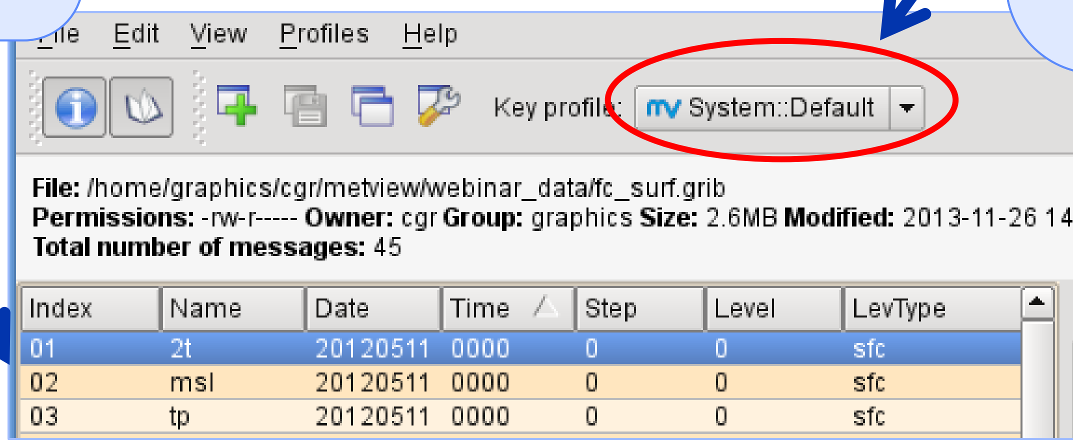
GRIB API Namespace: geography

Key name (GRIB API)	Key type	Value
geography.bitmapPresent	long	0
geography.gridType	string	regular_ll
geography.iDirectionIncrementInDegrees	double	1.5
geography.iScansNegatively	long	0
geography.jDirectionIncrementInDegrees	double	1.5
geography.jPointsAreConsecutive	long	0
geography.jScansPositively	long	0
geography.latitudeOfFirstGridPointInDegrees	double	90
geography.latitudeOfLastGridPointInDegrees	double	-90
geography.longitudeOfFirstGridPointInDegrees	double	0
geography.longitudeOfLastGridPointInDegrees	double	358.5

Namespace geography groups keys describing the grid resolution, area etc. GRIB API concept

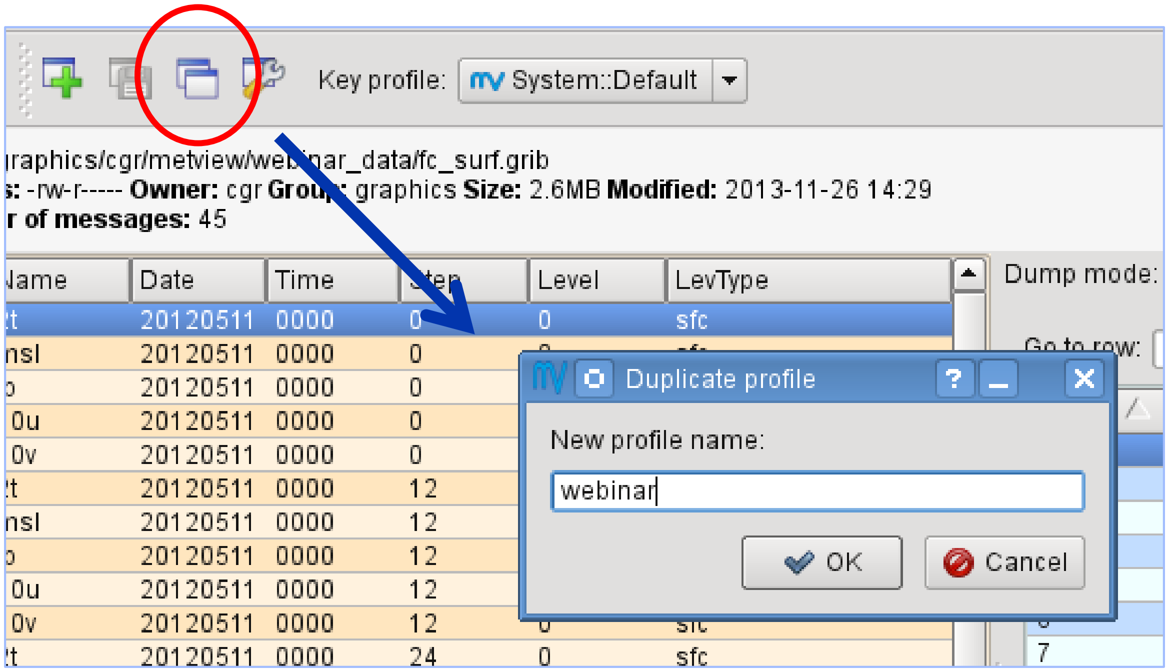
GRIB Examiner – Key profiles

The message list is presented using a set of GRIB API keys. A group of these keys is called a **key profile**.



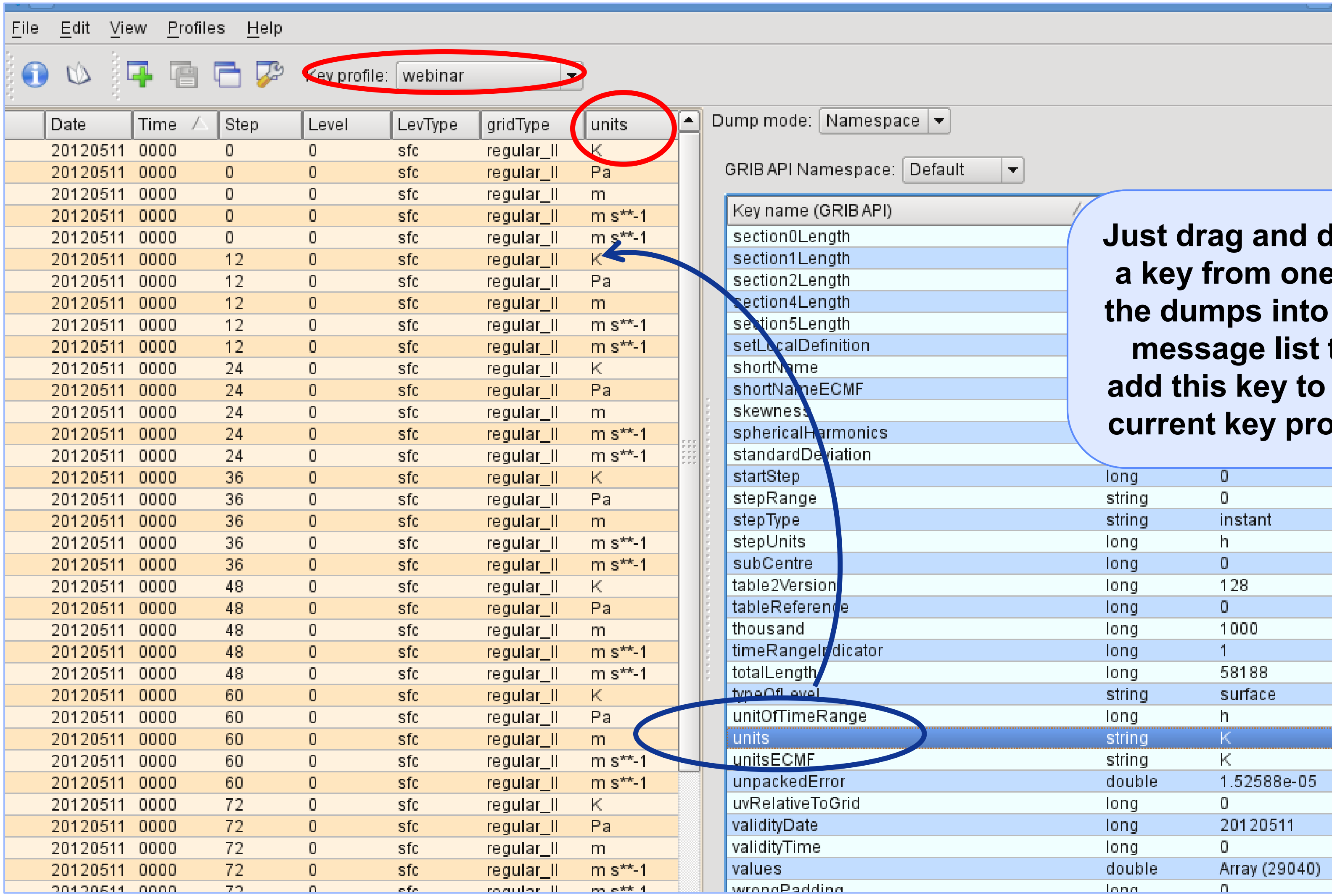
The default profile is called **System::Default**. It is **read-only**! However you can create any number of additional profiles.

GRIB Examiner – Create a new key profile



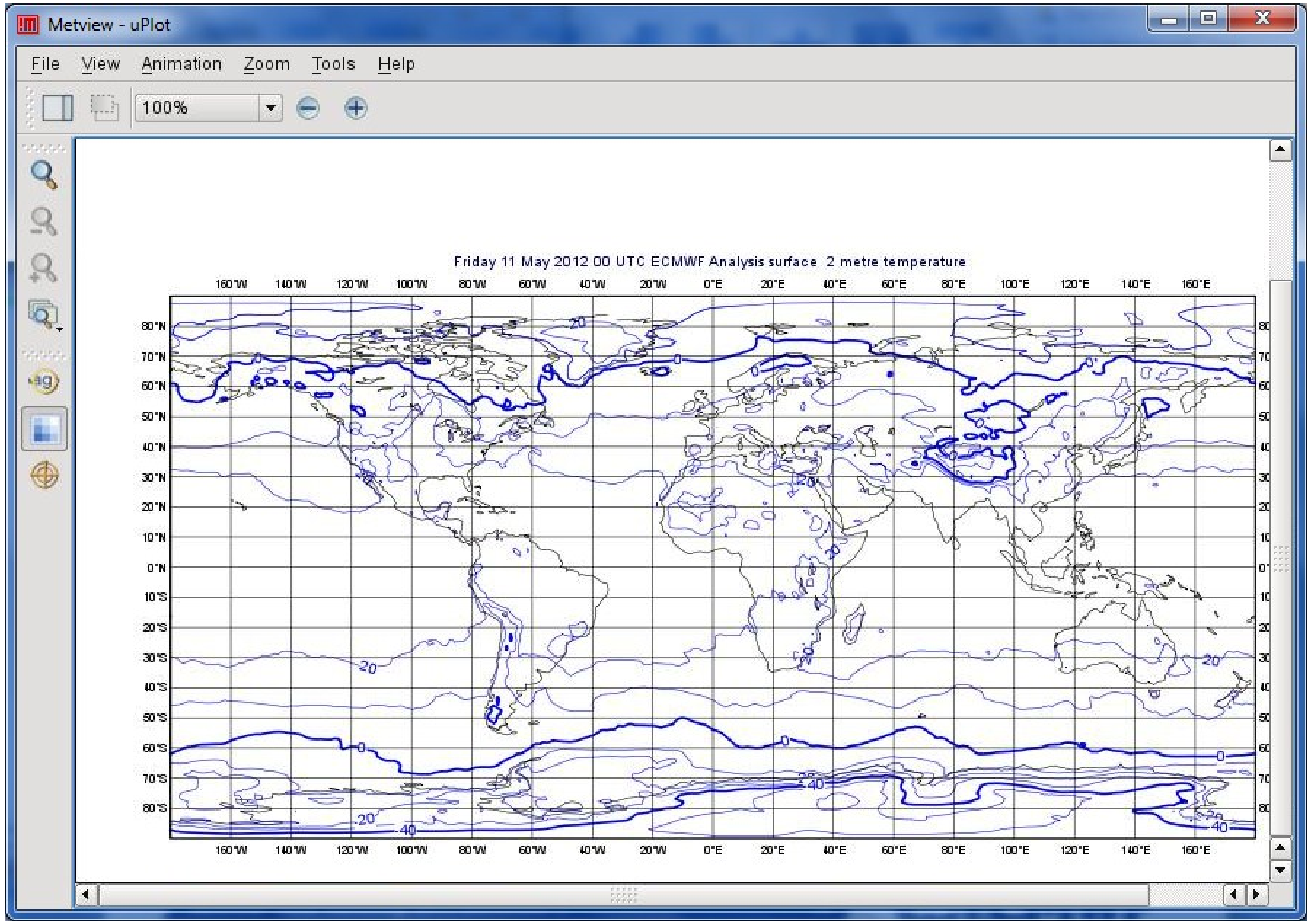
The easiest way to create a new key profile is to duplicate an existing one

GRIB Examiner – Populate key profiles



Just drag and drop a key from one of the dumps into the message list to add this key to the current key profile

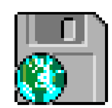
GRIB plotting



Overlaying fields from the same GRIB file

<u>Example:</u> overlay T2 and MSLP forecasts from file fc_surf.grib

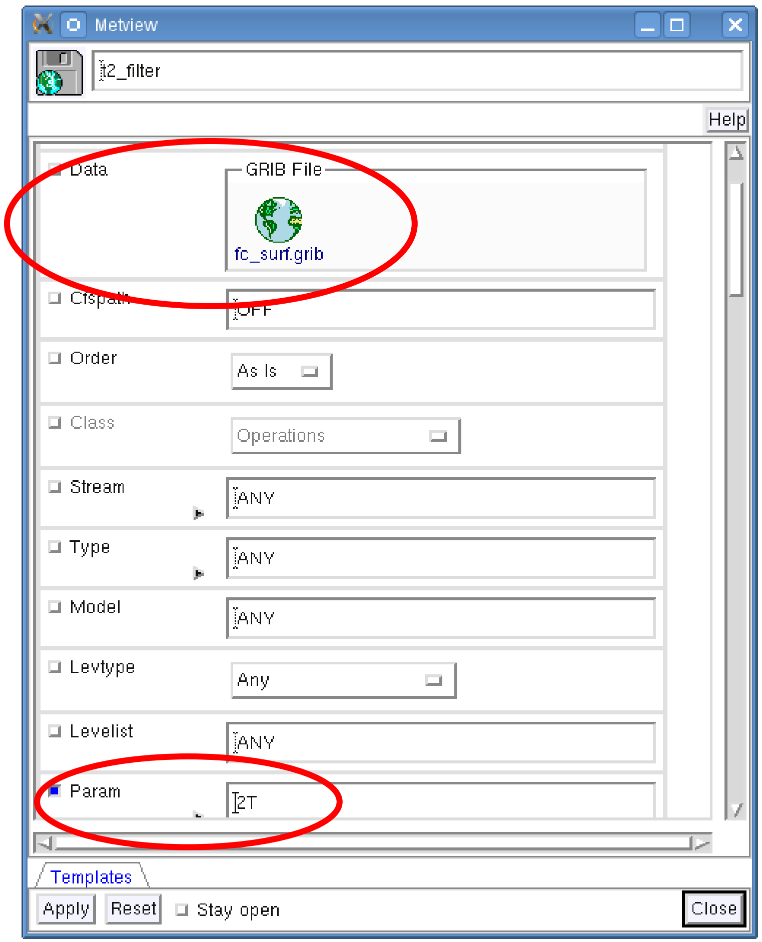
- **We need to filter out each parameter into a separate file**
- **We will use the GRIB Filter icon**

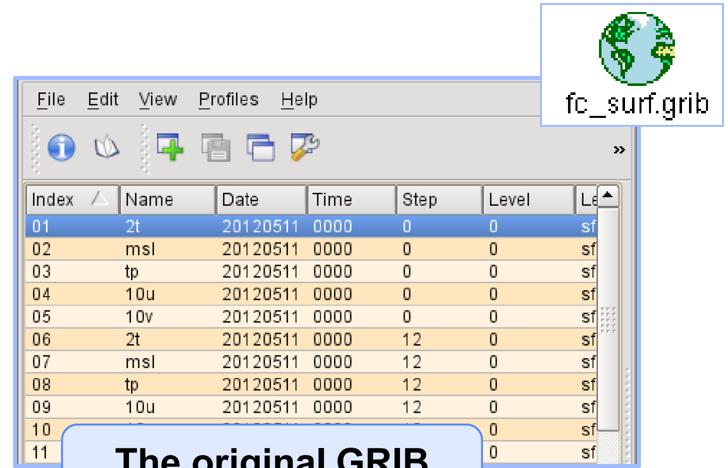


GRIB Filter

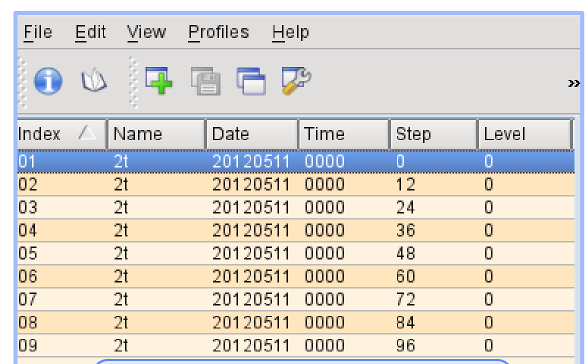
- **It allows filtering according to parameter, date, time, level etc.**
- **It caches the results (name turns green) and can be used directly in the same way as GRIB icon**

GRIB Filter: Parameter selection



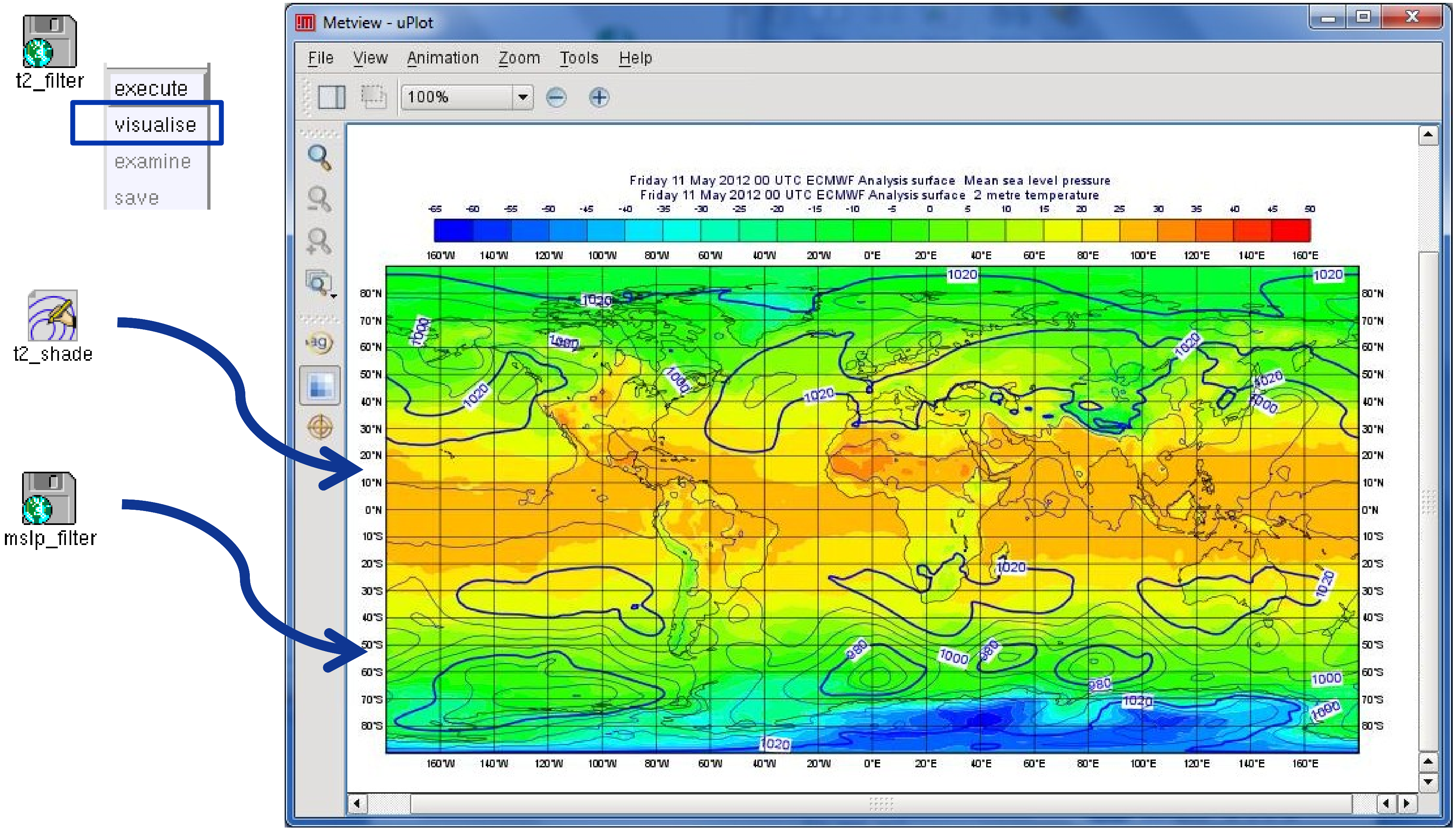


The original GRIB

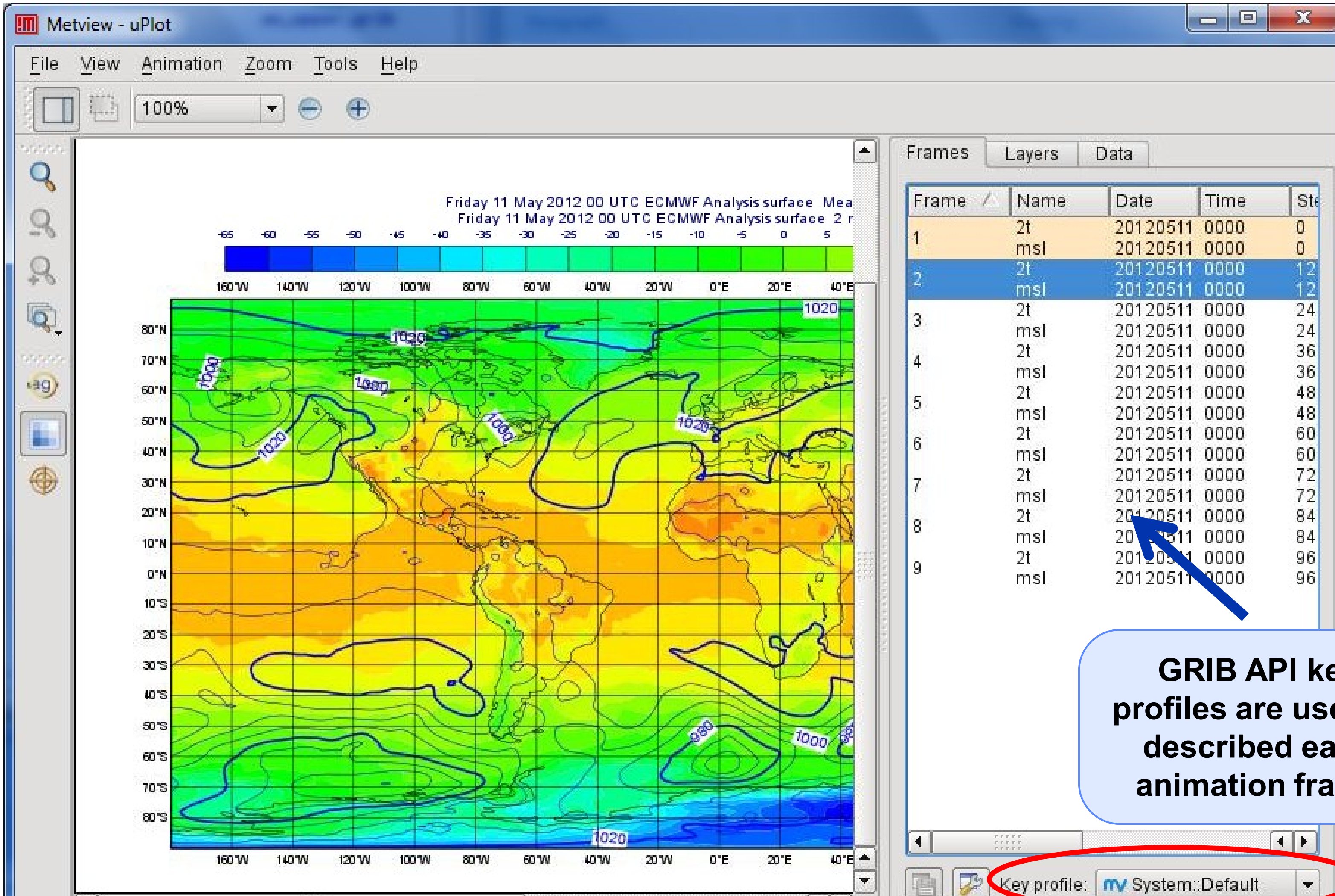


The resulting GRIB

Overlaying GRIB fields

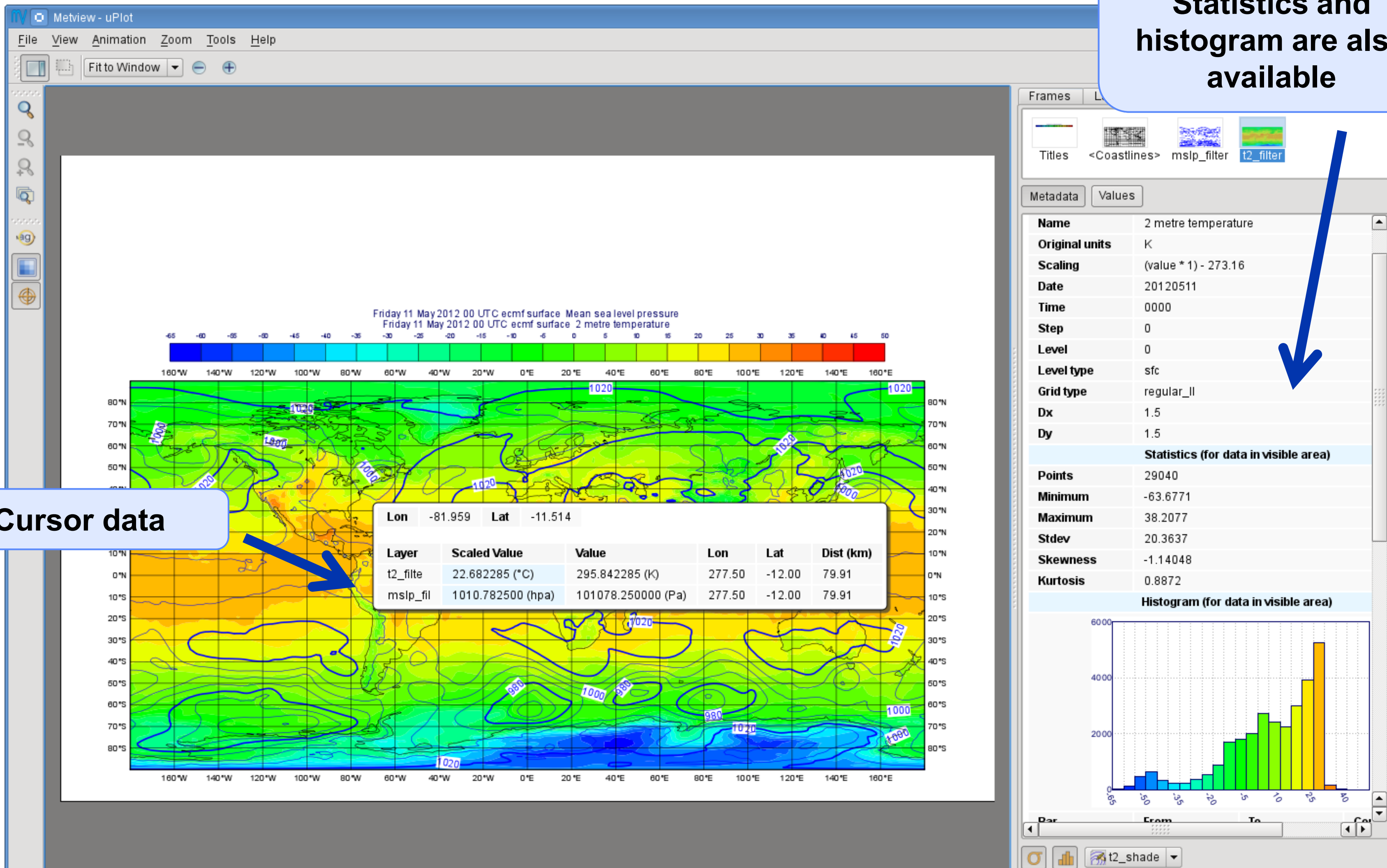


Overlaying GRIB fields



GRIB API key profiles are used to described each animation frame

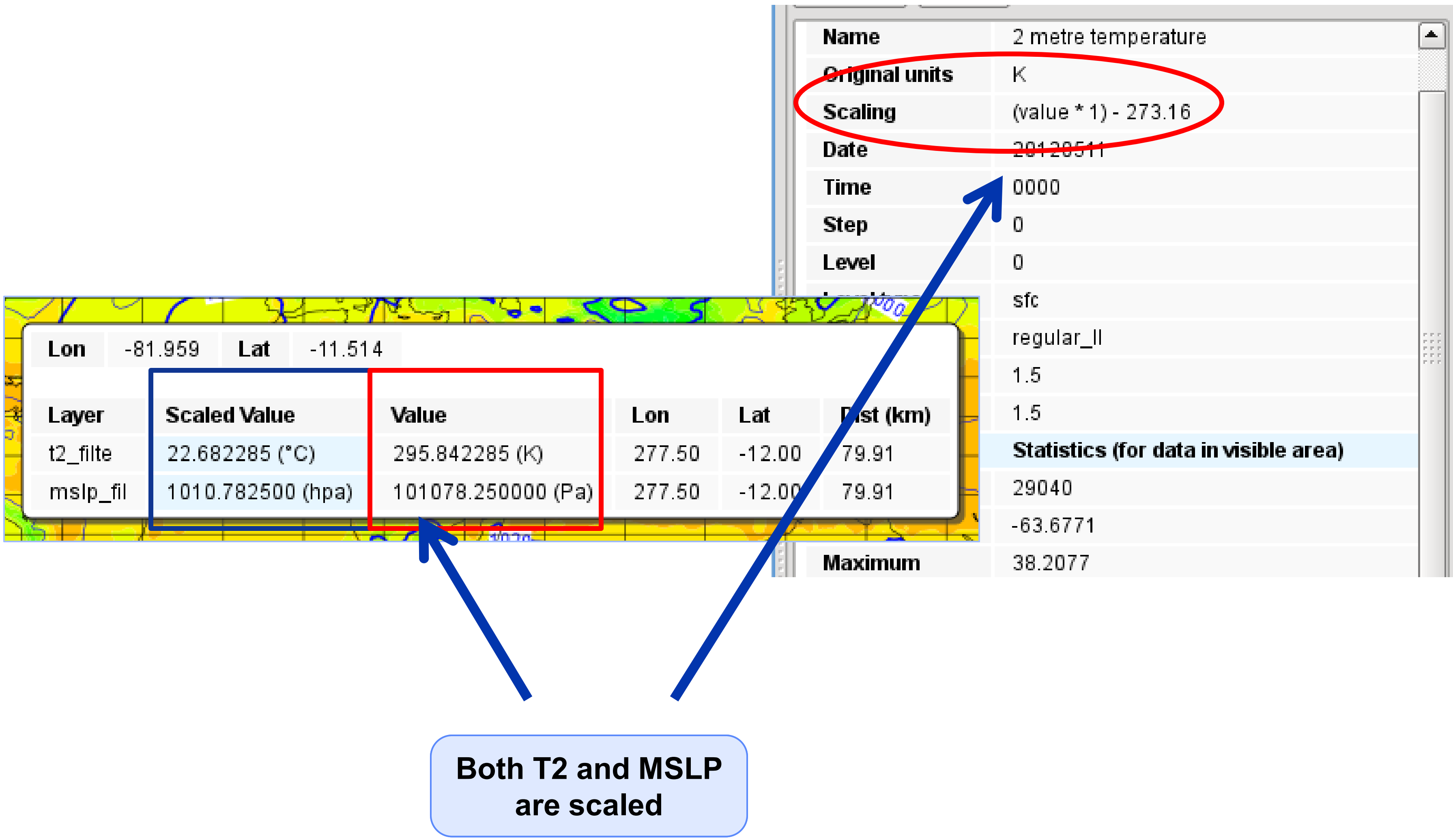
GRIB data inspection



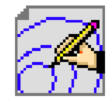
Statistics and histogram are also available

Cursor data

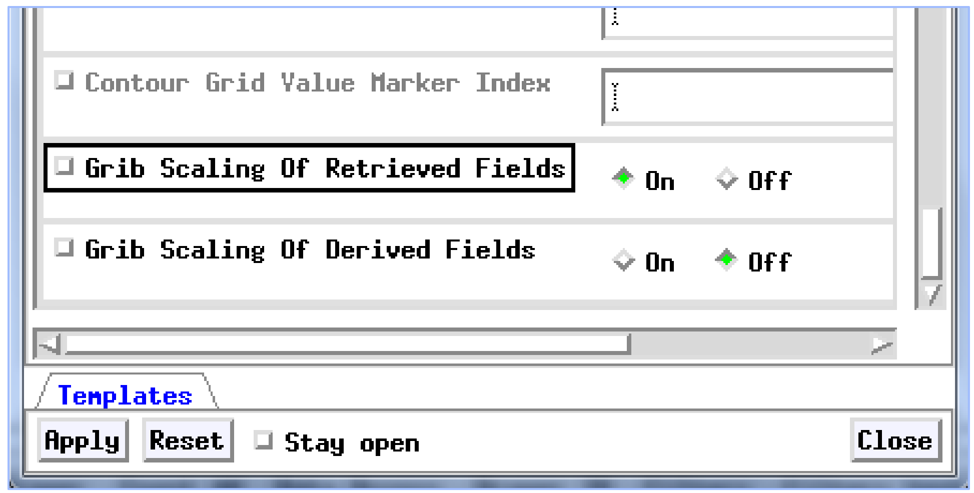
GRIB scaling for plotting



GRIB scaling for plotting

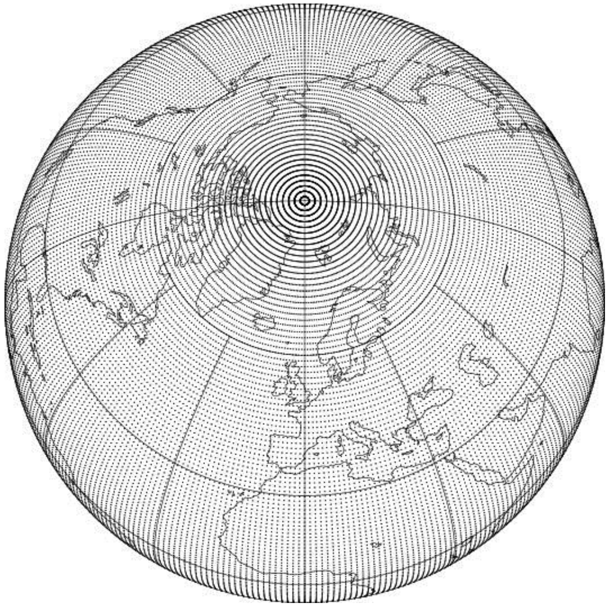

t2_shade

This parameter tells Metview to apply scaling for certain fields for contouring



Other usage of GRIB Filter: interpolation

- Spherical harmonics to gridpoint transformation
- Interpolation between different grids
  - Regular Gaussian grid
  - Reduced Gaussian grid
  - lat-lon grids etc.
- Currently it is based on EMOS lib



How to use the interpolation?

Example: compute the difference between two different resolution T500 fields

fc_upper.grib

an_upper.grb

Forecast
1.5x1.5
global grid

Analysis
0.25x0.25 limited
area grid

- **The steps involved:**
  1. Filter T500 for the matching date and time
  2. Interpolate the global field to the LAM grid
  3. Compute the difference

→ **We will write a macro!**

Macro: Compute difference #1

Filter parameter and level from analysis. In Macro the GRIB Filter is invoked via command: read

```
#Metview Macro

#Read T500 analysis
g_an=read(source: "an_upper.grib",
          param: "t",
          level: 500)

#Read target area borders
lat1=grib_get_double(g_an,"latitudeOfFirstGridPointInDegrees") #north
lat2=grib_get_double(g_an,"latitudeOfLastGridPointInDegrees") #south
lon1=grib_get_double(g_an,"longitudeOfFirstGridPointInDegrees") #west
lon2=grib_get_double(g_an,"longitudeOfLastGridPointInDegrees") #east

#Read target grid resolution
dx=grib_get_double(g_an,"iDirectionIncrementInDegrees")
dy=grib_get_double(g_an,"jDirectionIncrementInDegrees")

```

Here we read a set of GRIB API keys

Macro: Compute difference #2

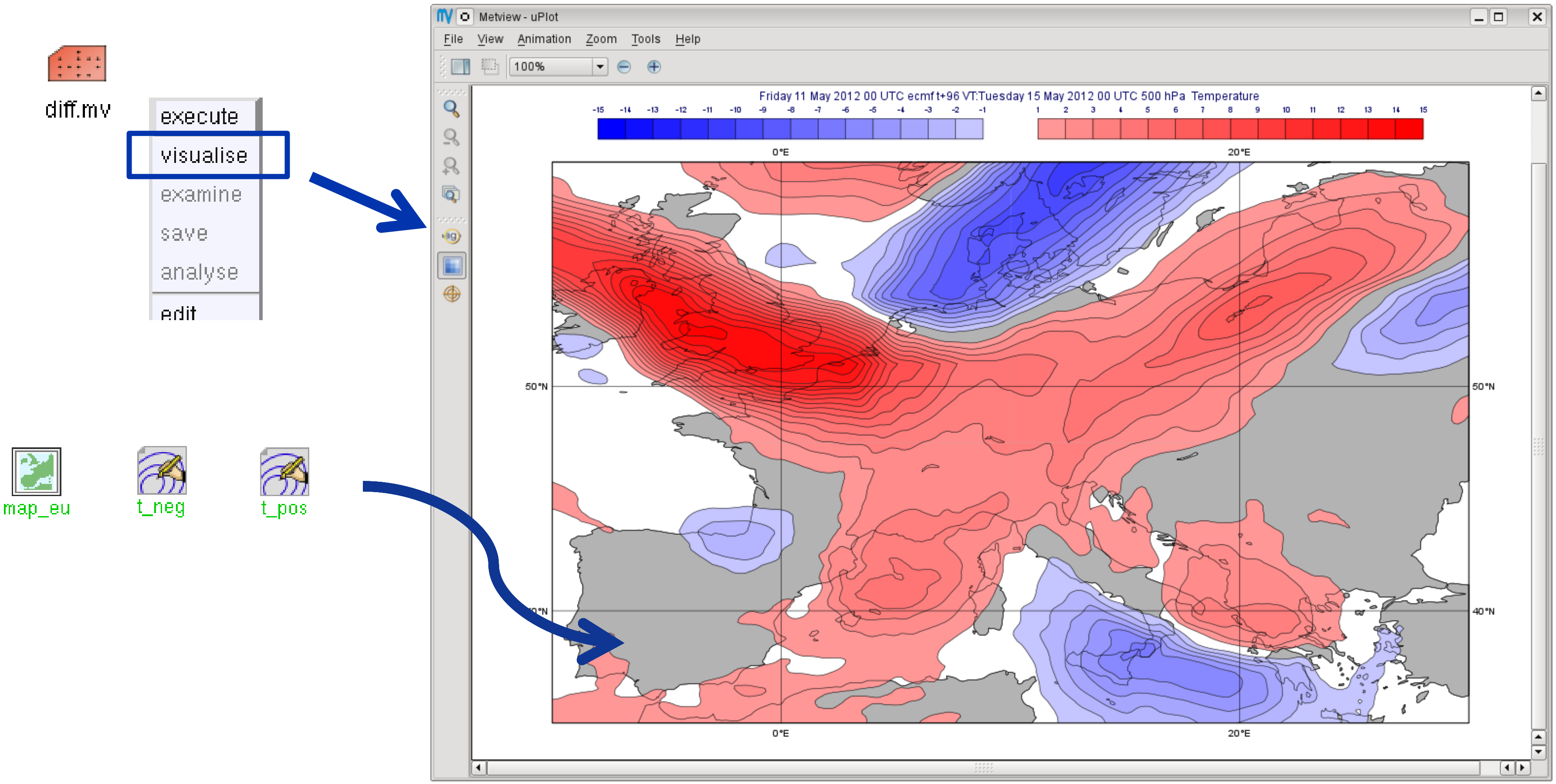
Filter parameter and level and step from forecast

Interpolation

```
#Read T500 forecast for the analysis date (+96h)
#and interpolate it to the target grid
g_fc=read(source: "fc_upper.grib",
          param: "t",
          level: 500,
          step: 96,
          area: [lat2,lon1,lat1,lon2],  #[s,w,n,e]
          grid: [dx,dy] )

#Compute difference
g_res=g_fc-g_an

#Return results
return g_res
```

Difference operator only works between grids with the same number of points

Macro: Compute difference #2



Macro usage: compute wind speed

Example: compute 10 windspeed from u and v components

```
#Metview Macro

#read GRIB
g=read(source: "fc_surf.grib")

#Filter 10u
u=read(data: g,
       param: "10u"
       )

#Filter 10v
v=read(data: g,
       param: "10v"
       )

#Compute sp
sp=sqrt(u*u+v*v)

#Set shortName to the correct value
sp=grib_set_string(sp,["shortName","10si"])

#Return results
return sp
```

Fieldset operation

Here we set the GRIB header

New value for the key

GRIB API key

Macro usage: compute precipitation for intervals

- Precipitation is often stored as an accumulated quantity
- We want to see precipitation for a given interval (e.g. 12h, 24h)

Example: compute precipitation for 12 h intervals from file **fc_surf.grib**

We filter tp

Operation between slices of fieldsets: we compute the difference of adjacent fields

tp.mv

```
#Metview Macro

#Read tp fields from the GRIB file
g=read(
        source: "fc_surf.grib",
        param: "tp"
)

#Number of tp mesages
num=count(g)

#Compute tp for 12h intervals
gr=g[2,num]-g[1,num-1]

#Return results
return gr
```

Precipitation computation explained

g f1 f2 f3 .... fnum

Our original fieldset has num fields

```
#Compute tp for 12h intervals
gr=g[2,num]-g[1,num-1]
```

g[2,num] f2 f3 .... fnum

-

g[1,num-1] f1 f2 .... fnum-1

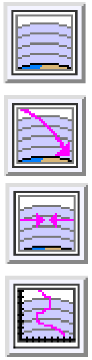
g[2,num]-g[1,num-1] f2-f1 f3-f2 .... fnum-fnum-1

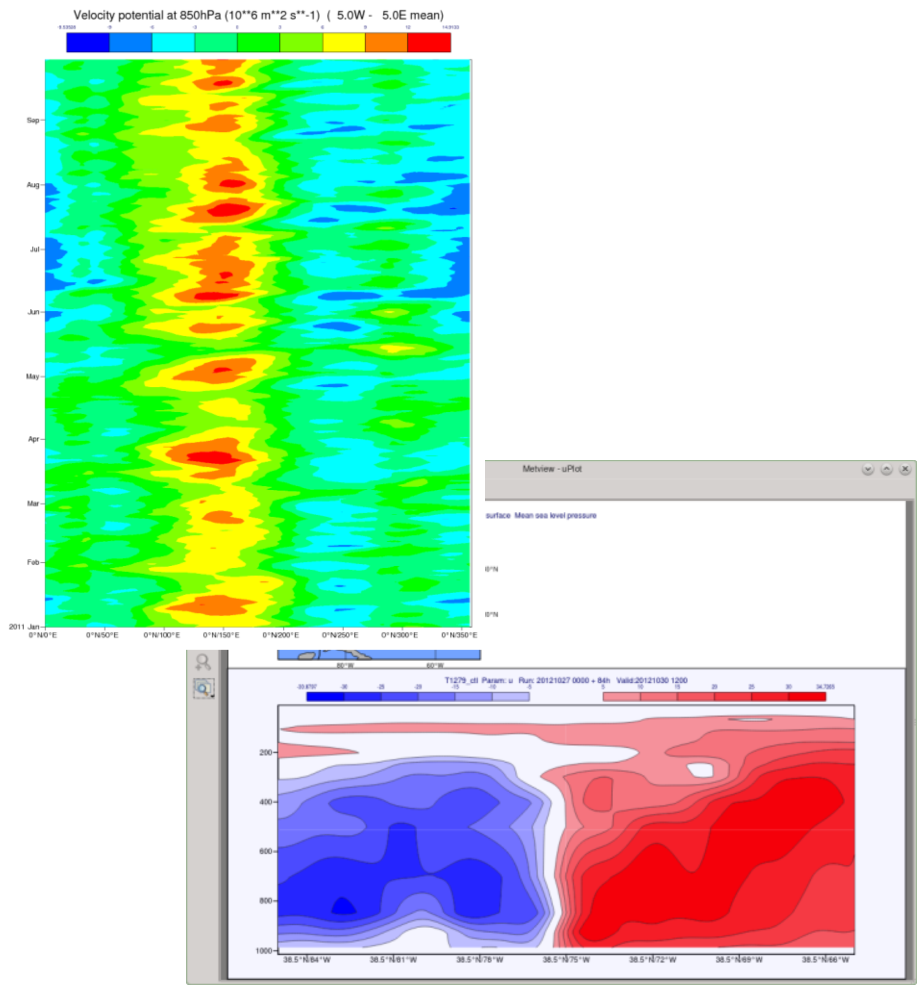
Macro usage: more functions

- A rich set of macro functions exists for GRIB. A few examples:
  - latitudes(), longitudes(), values(): read the latitudes, longitudes and values of a field into vectors (in-memory arrays)
  - average(): compute average
  - mask(): set field values to 0 or 1 using an area mask
  - bitmap(): assign missing values to a field using a mask
  - nobitmap(): replace missing values
- See Macro Tutorial 3 for some elaborated examples, such as masking one field based on the values of another (e.g. apply a land sea mask to a field to remove (i.e. to bitmap) points over sea)

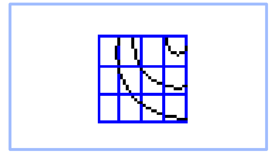
Complex plot types for GRIB

- These plots require data extraction from multiple fields and some computations as well
- There are a set of GRIB specific icons to generate:
  - Cross sections
  - Hovmøller diagrams
  - Zonal mean plots
  - Vertical profiles

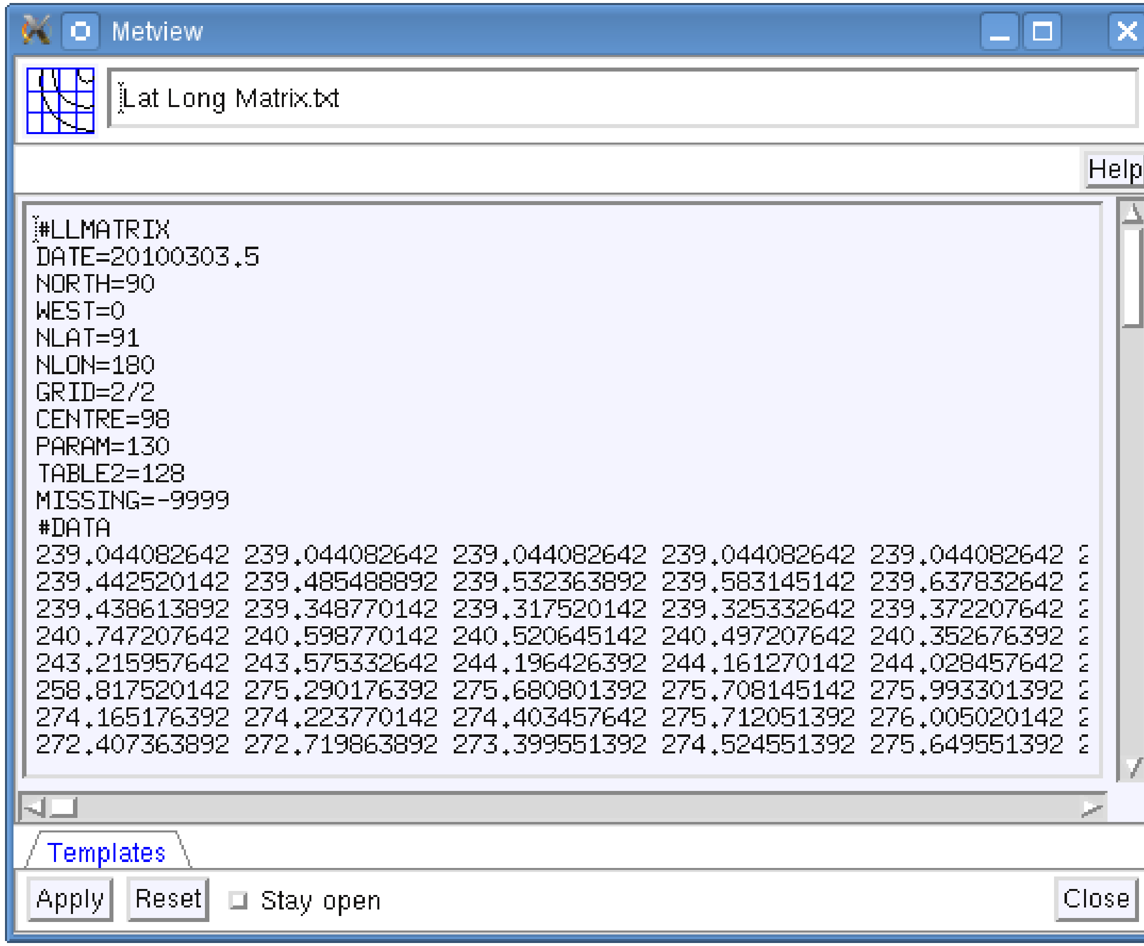




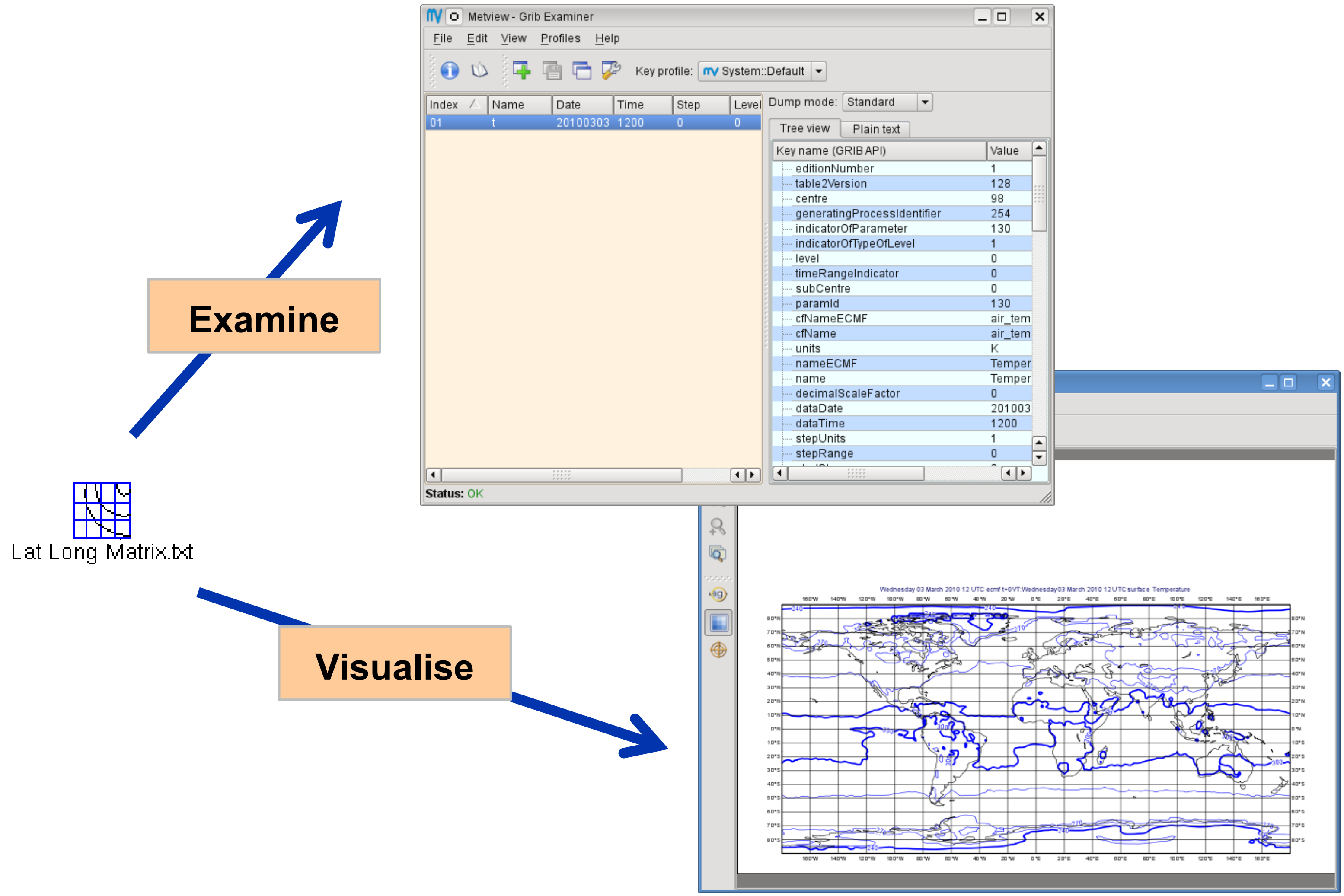
Lat Long Matrix



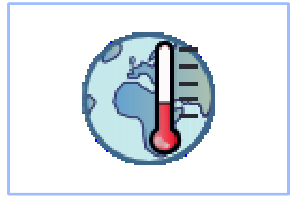
- Metview's ASCII format for gridded data
- Turned into GRIB internally
- Can be edited as a text file



Lat Long Matrix – Behaves like a GRIB



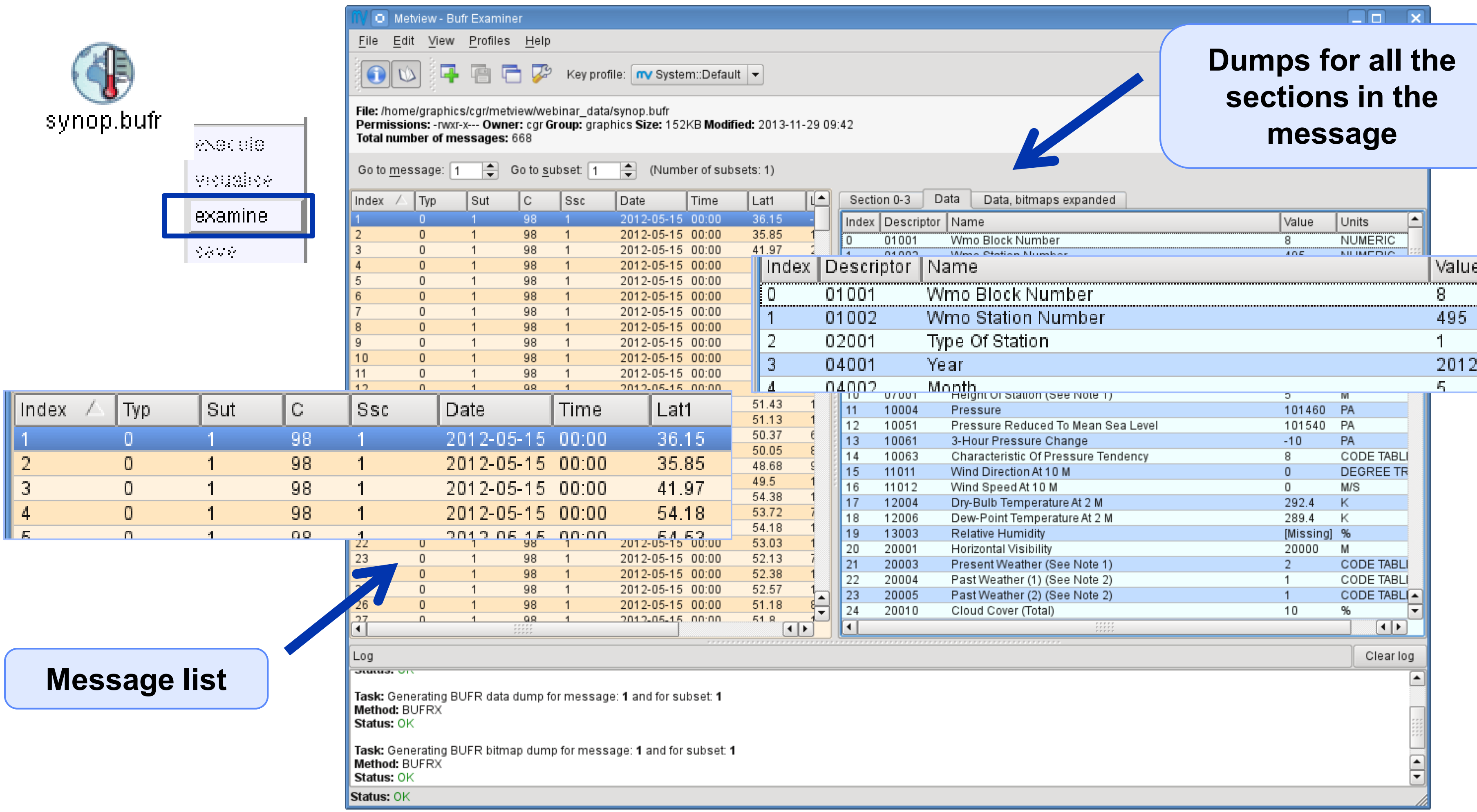
BUFR



- WMO’s binary format for observation data
- Metview offers a high level interface to work with BUFR
- Internally we use BUFRDC (part of EMOS lib) to decode BUFR messages
- There is a BUFR tutorial available on the Metview web page

BUFR Examiner

- BUFRs contents can be checked with the **BUFR Examiner**

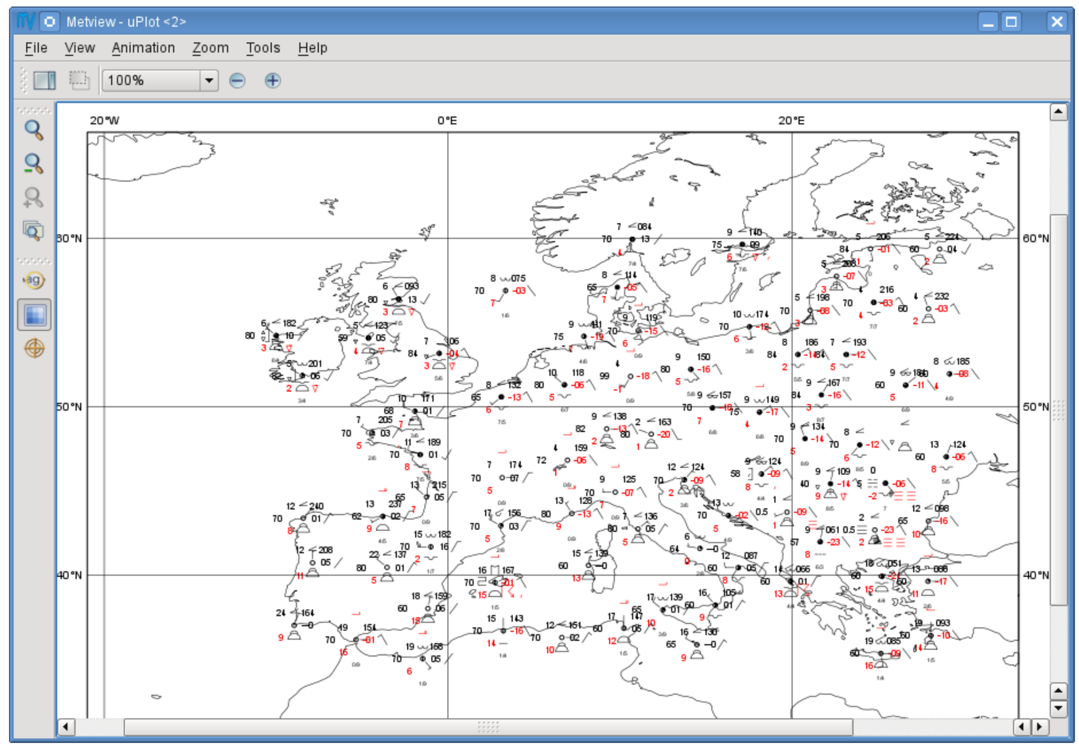


Dumps for all the sections in the message

Message list

BUFR Plotting

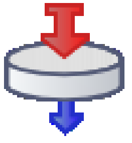
- We can directly visualise BUFR files with conventional observations (e.g. SYNOP)



BUFR: Accessing data

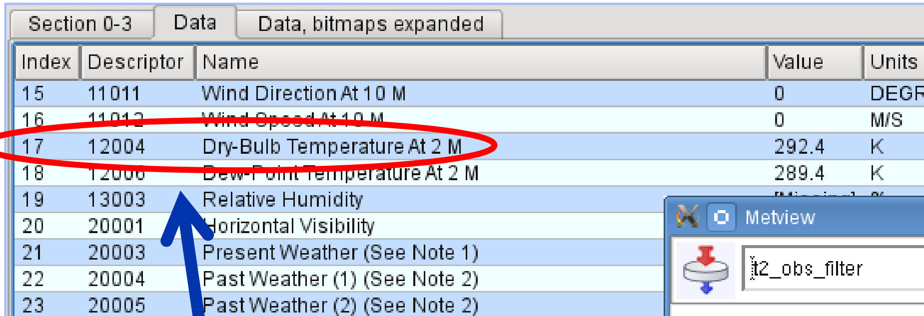
Example: extract and plot T2 with symbol plotting from file synop.bufr

- **We need to use the Observation Filter icon**

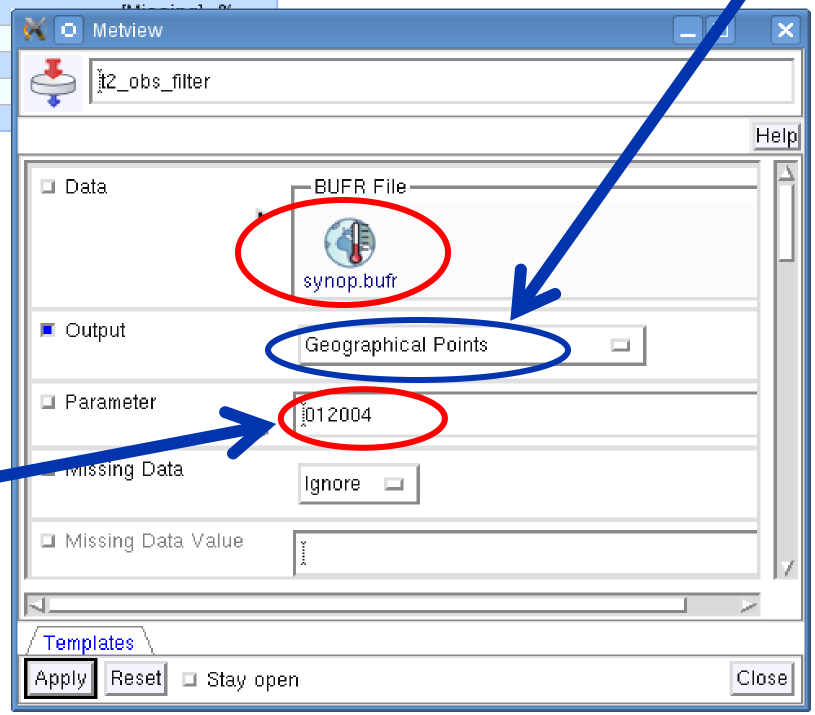


- **It can perform filtering according to parameter, level, area, time, channel etc.**

BUFR: Filtering

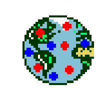


Index	Descriptor	Name	Value	Units
15	11011	Wind Direction At 10 M	0	DEGR
16	11012	Wind Speed At 10 M	0	M/S
17	12004	Dry-Bulb Temperature At 2 M	292.4	K
18	12006	Dew-Point Temperature At 2 M	289.4	K
19	13003	Relative Humidity		
20	20001	Horizontal Visibility		
21	20003	Present Weather (See Note 1)		
22	20004	Past Weather (1) (See Note 2)		
23	20005	Past Weather (2) (See Note 2)		



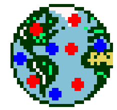
We set the output of the filtering operation to **Geopoints**

We can save the result into a file



Parameters are defined by their BUFR descriptors

Geopoints

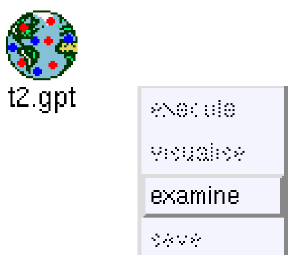


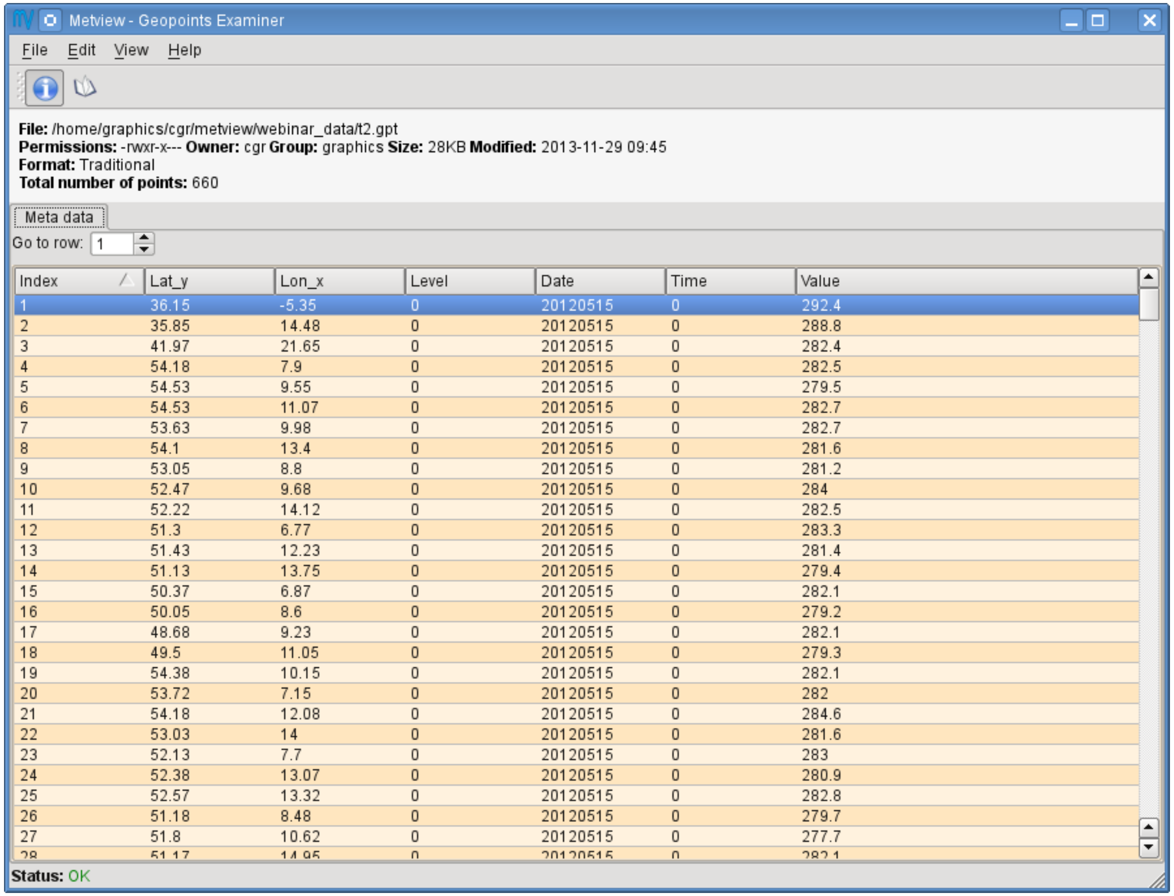
- Metview's custom format to store scattered geo-referenced data
- ASCII files with 4 different types: The default is shown here:

```
#GEO
PARAMETER = 12004
lat      long     level    date       time    value
#DATA
36.15    -5.35    0        20120515   0000    292.4
35.85    14.48    0        20120515   0000    288.8
41.97    21.65    0        20120515   0000    282.4
```

Geopoints Examiner

- Geopoints contents can be checked with the **Geopoints Examiner**
- This is how the result of the BUFR filtering looks like

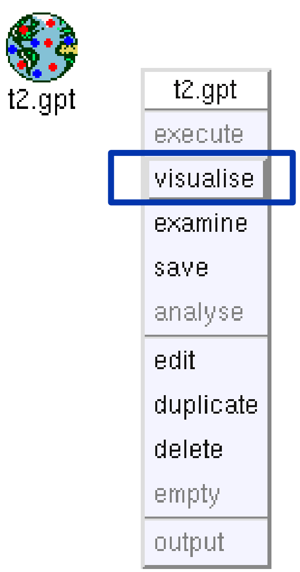


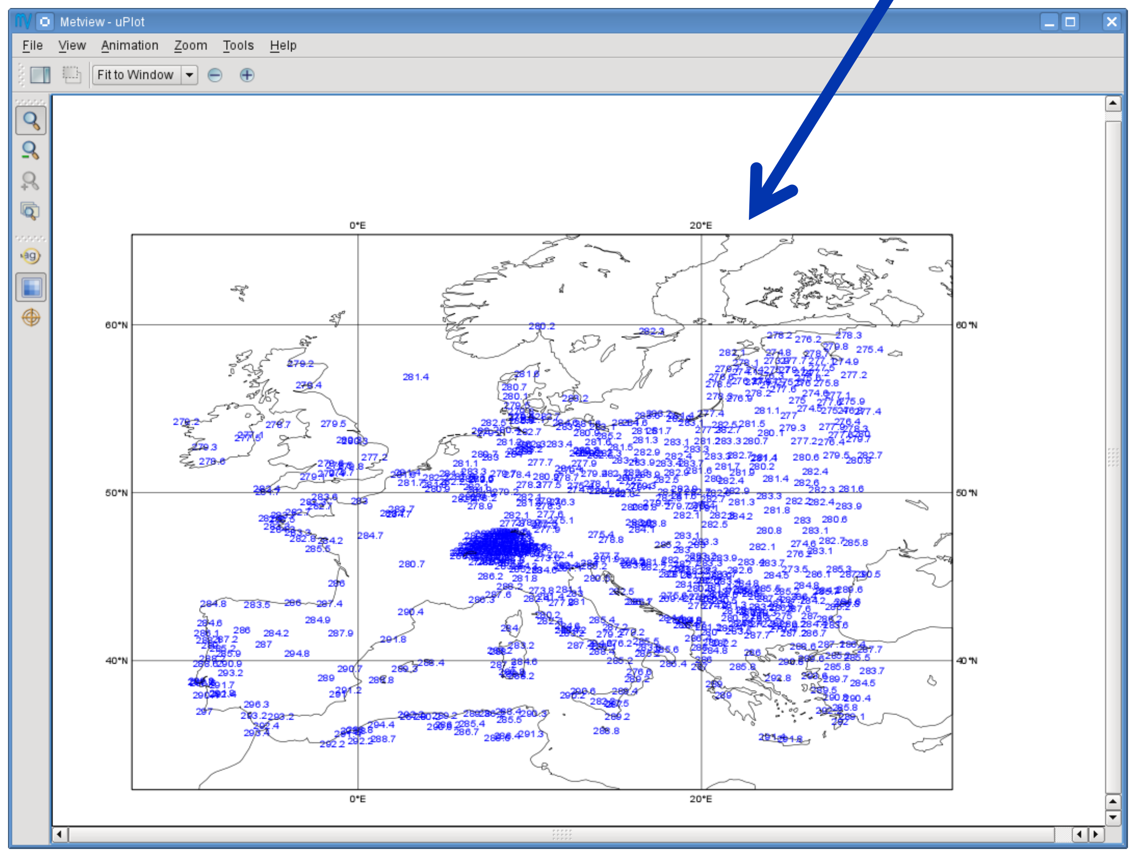


Geopoints Plotting

- We can directly visualise Geopoints
- It is based on symbol plotting

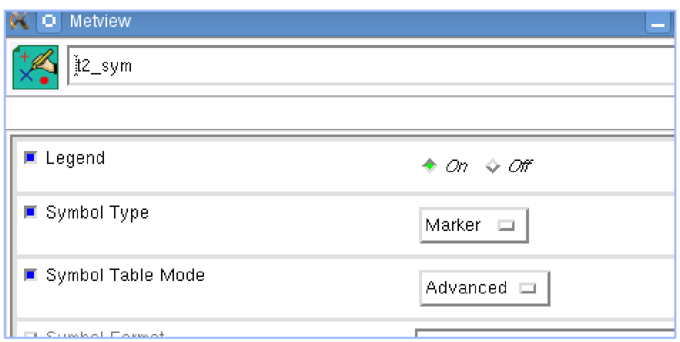
By default the numbers are plotted to the map

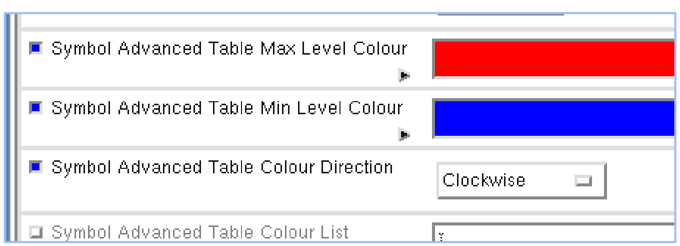


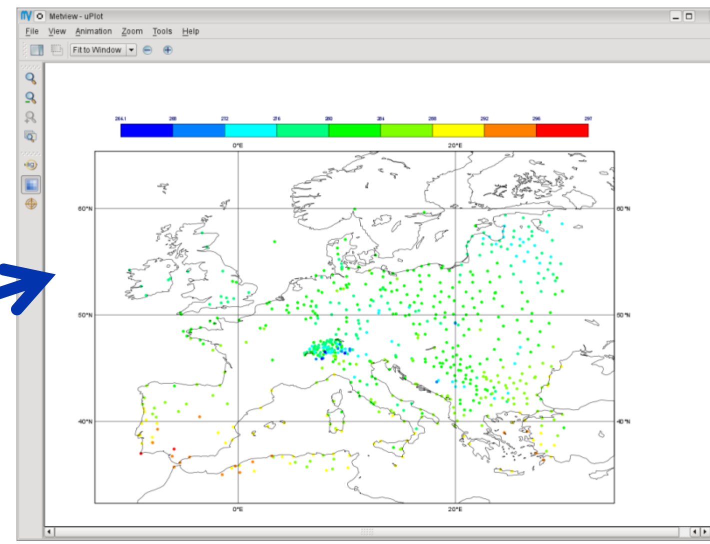


Customisation with Symbol Plotting

- The **Symbol Plotting** icon offers a large number of options for plot customisation
- We can use the **Advanced Table Mode** to define a nice colour palette between the min and max colours (just like for Contouring)





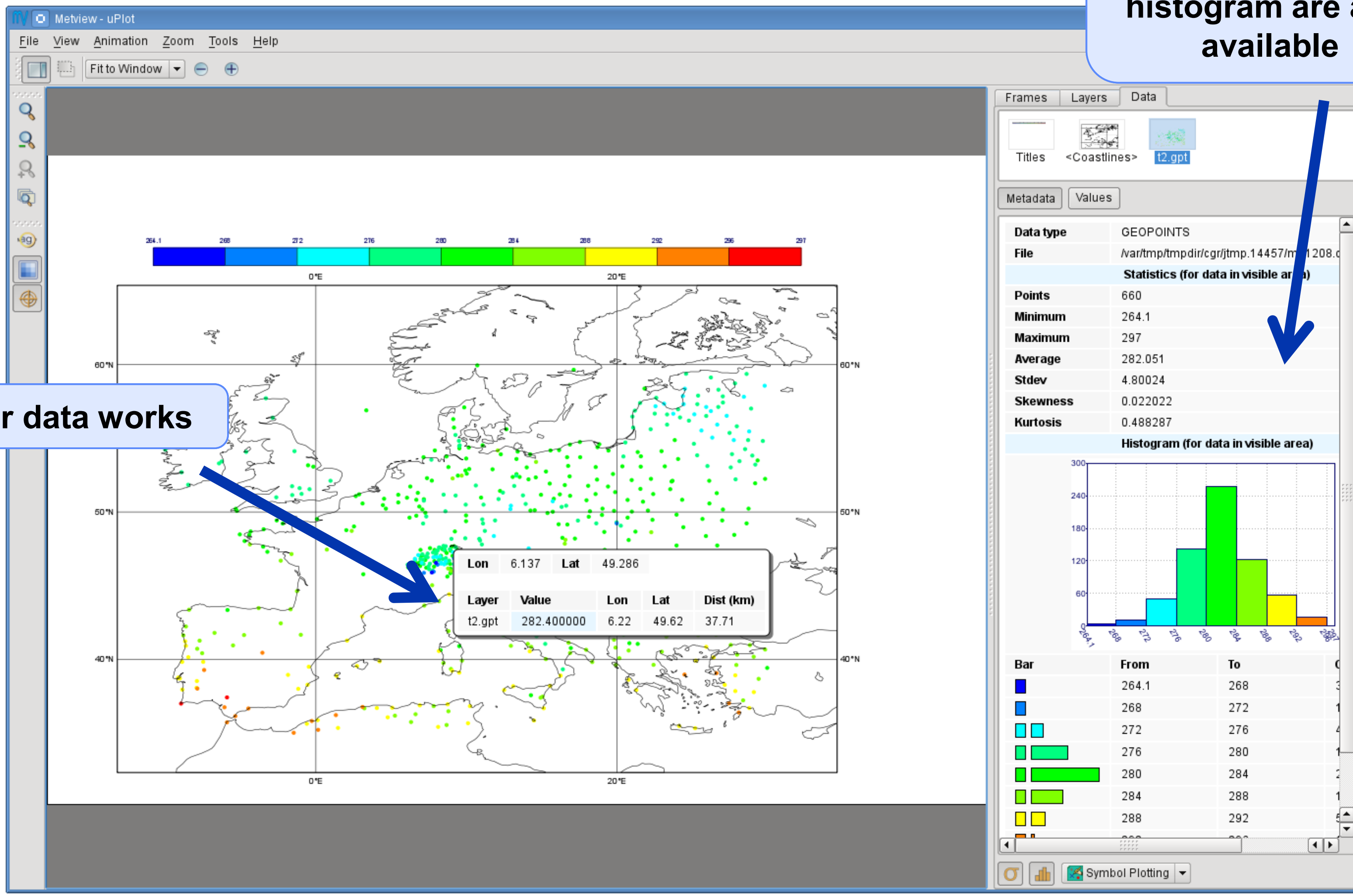


ECMWF

Geopoints Plotting

Statistics and histogram are also available

Cursor data works



Macro: difference between GRIB and Geopoints

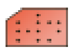
Example:
compute the difference between the T2 forecast and observations

```
#Metview Macro

#Read t2 field forecast (96h) from the GRIB file
g=read(
        source: "fc_surf.grib",
        param: "2t",
        step: 96
)

#Read observations from gepoints
gpt=read("t2.gpt")

#Compute the difference
res=g-gpt

#Return results
return res
```

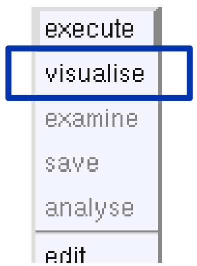
This step involves interpolation of the GRIB data to the Geopoints locations

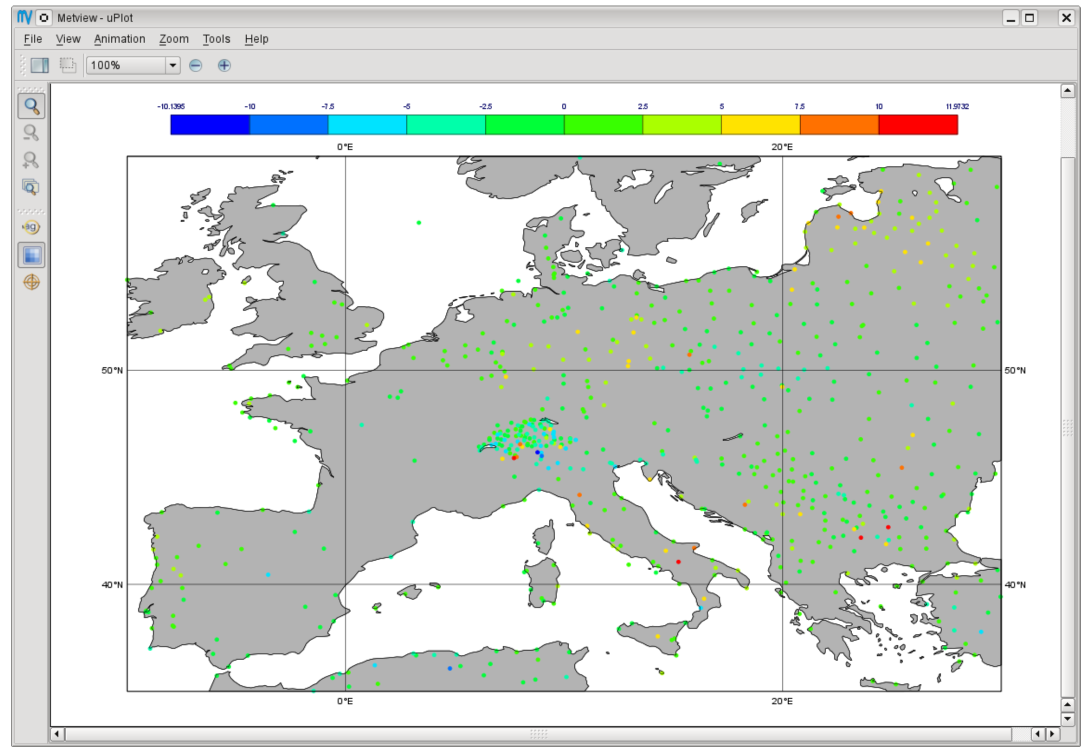
The result is another Geopoints

Compute difference between GRIB and Geopoints



grib_minus_geo.mv



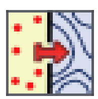


Forecast minus observation differences

Geopoints to GRIB

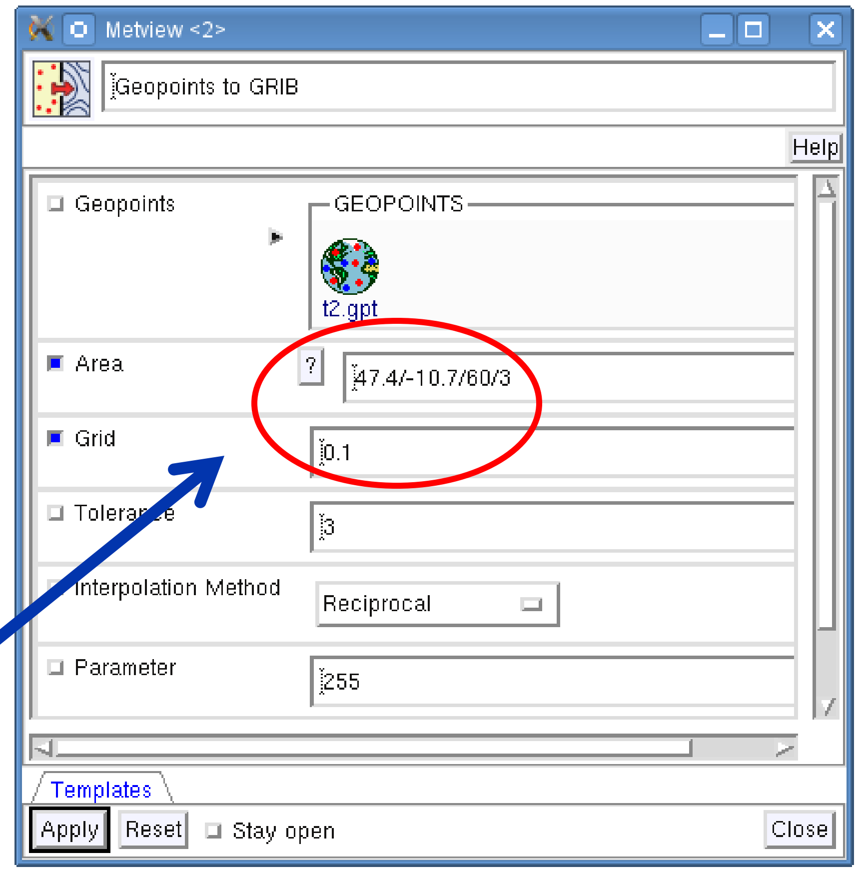
<u>Example:</u> interpolate T2 observations onto a grid then apply contouring

- **We need to use the Geopoints to GRIB icon**



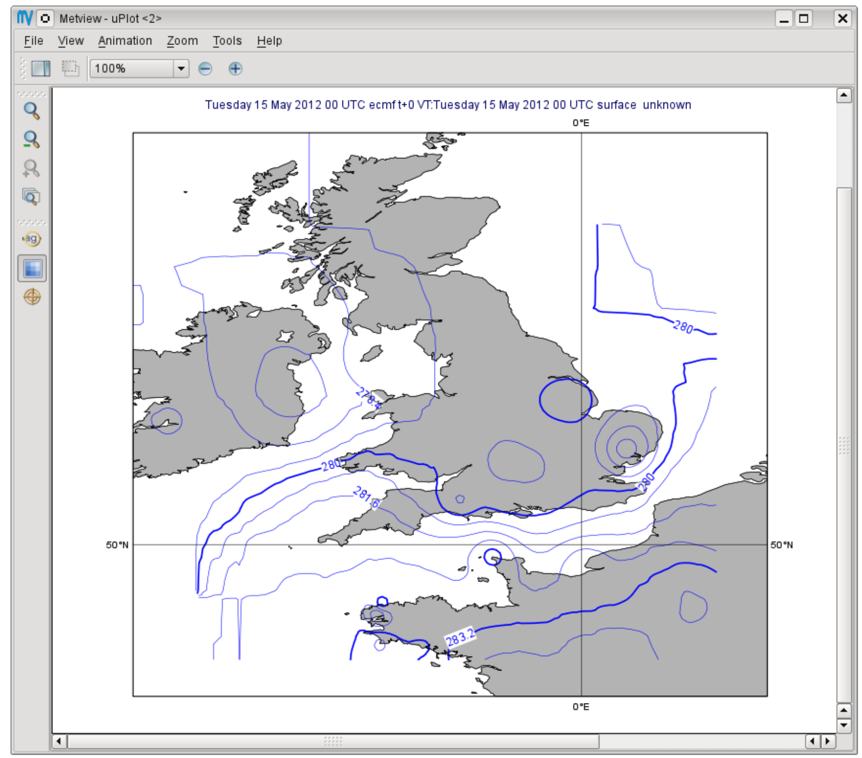
Geopoints to GRIB

- **This icon interpolates Geopoints data onto a regular lat-lon grid and encodes it into GRIB**



The grid definition

Geopoints to GRIB



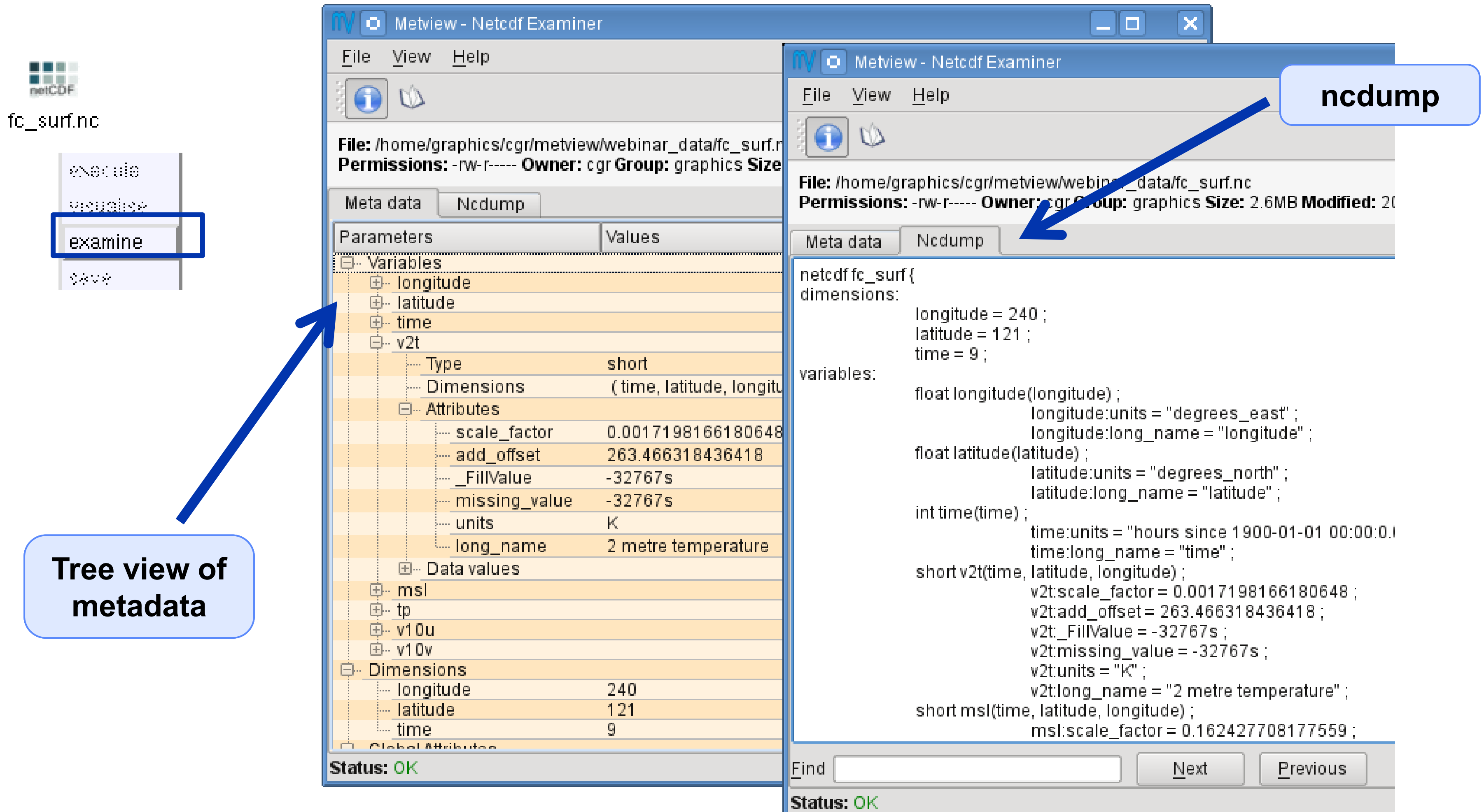
T2 observations interpolated onto a 0.1x0.1 grid

NetCDF

- UNIDATA's binary format for multidimensional arrays
- Metview's NetCDF plotting interface was added a few years ago

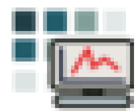
NetCDF Examiner

- NetCDF contents can be checked with the NetCDF Examiner



NetCDF: How to plot it?

- NetCDF is so flexible it can contain almost any kind of data
- We need to use the NetCDF Visualiser icon

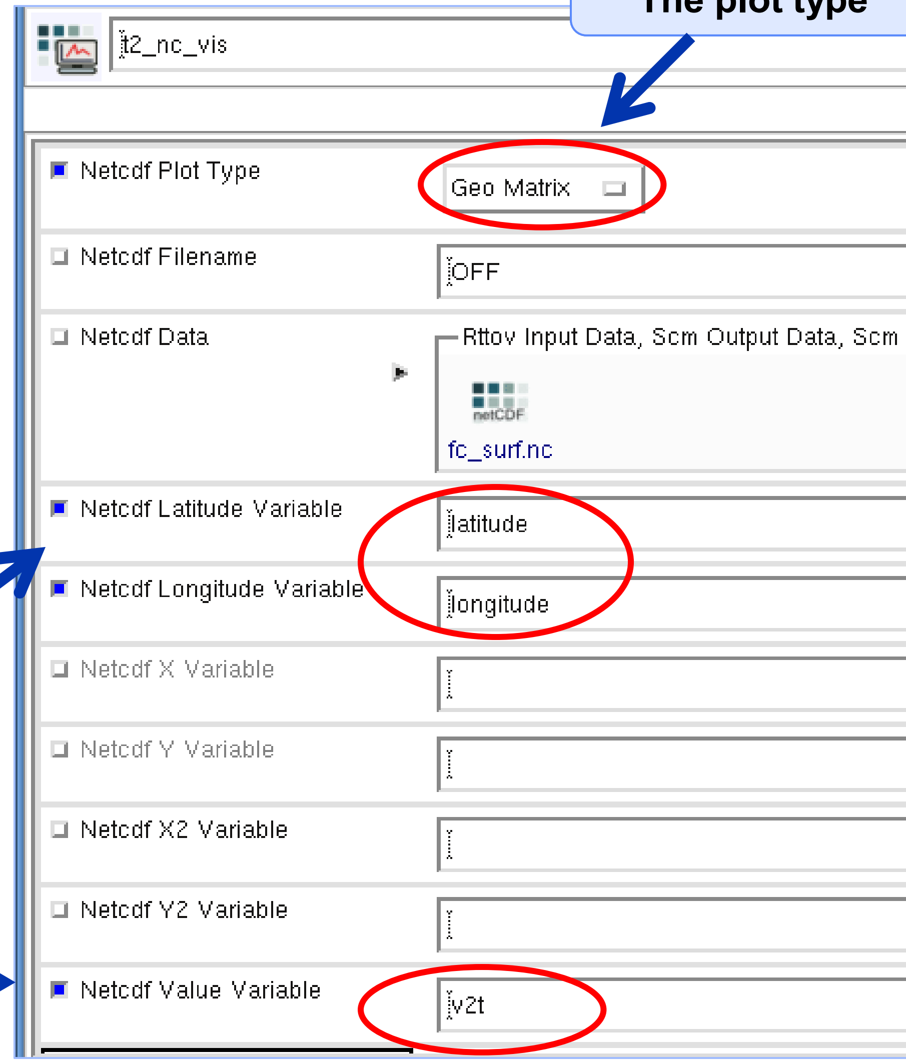


- It defines the way variables/dimensions are used for plotting

Plotting NetCDF data

Example:
plot T2 from file
fc_surf.nc

t2_nc_vis

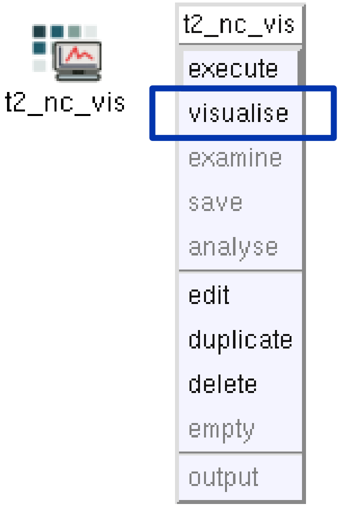


The plot type

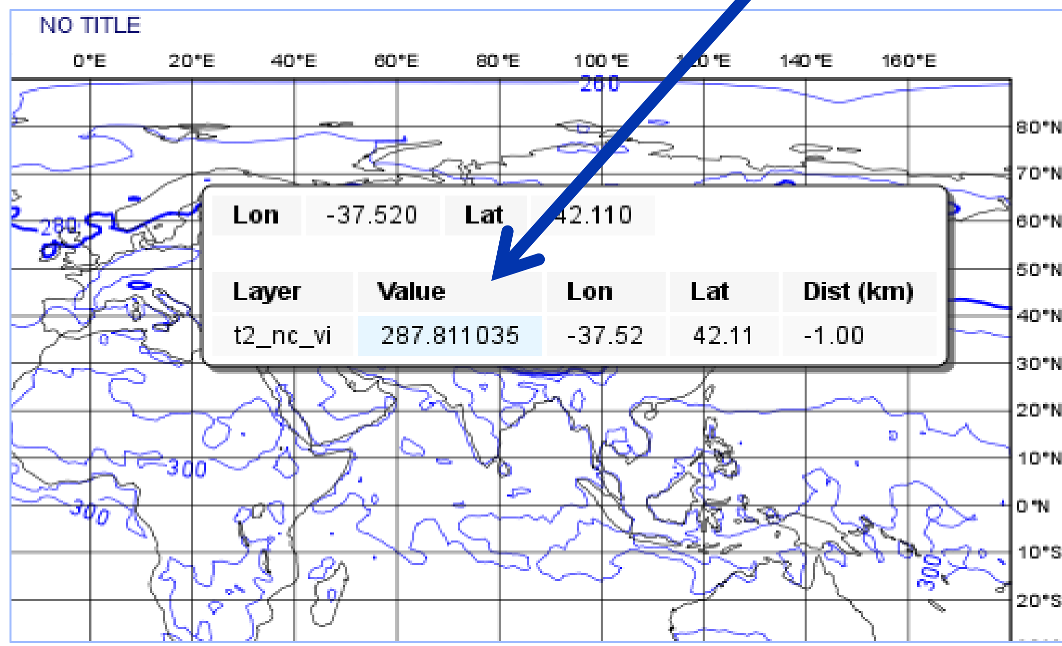
Latitude and longitude variables

The variable to plot (T2 is called v2t in our file)

NetCDF: Plotting



No scaling applies for NetCDF values: we have values in Kelvins



ECMWF

NetCDF: Macro Usage

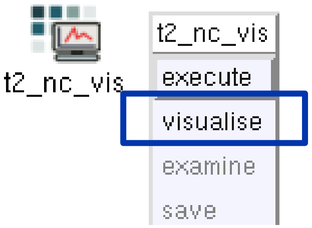
Example: convert values of T2 from Kelvin to Celsius

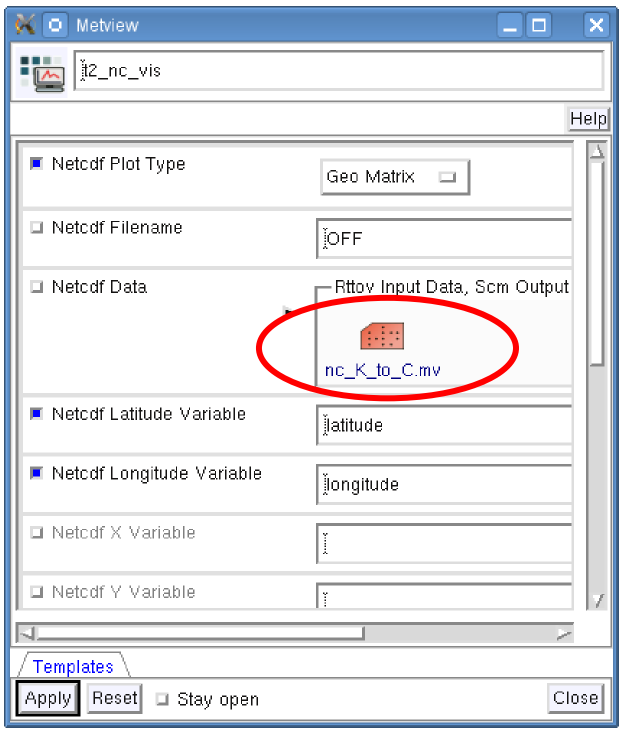
- **The NetCDF macro interface is based on the current variable concept: all operations are only valid to the currently selected NetCDF variable!**

nc_K_to_C.mv

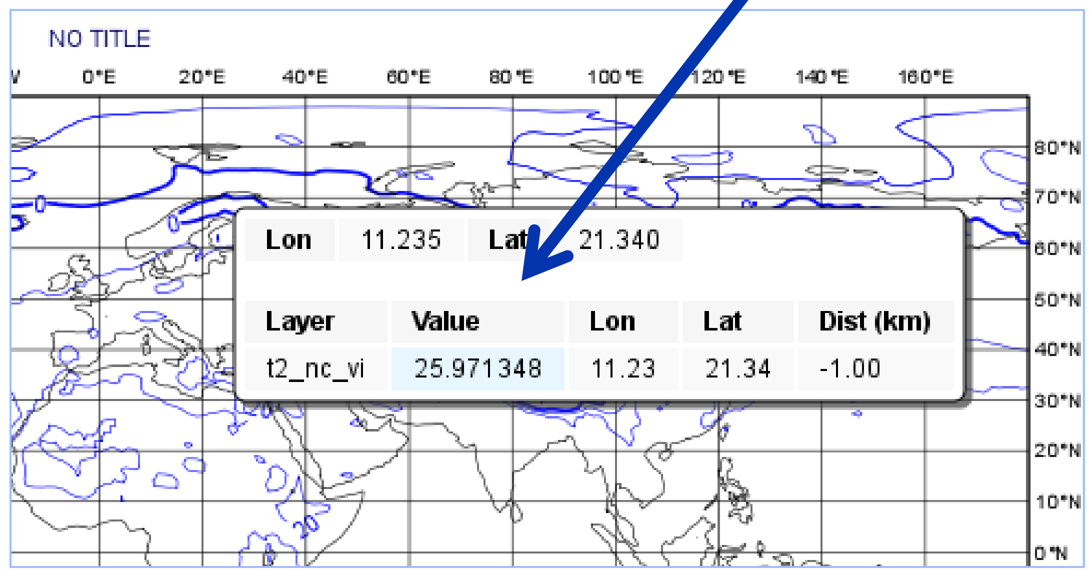
```
#Metview Macro

#Read netcdf file
nc=read("fc_surf.nc")

#Get the list of netcdf variables
var_list = variables(nc)

#Find index for t2
idx=find(var_list,"v2t")

#Set the current variable to t2
setcurrent(nc,idx)

#Change the values of the current variable
nc = nc - 273.16

#Return results
return nc
```

NetCDF: Plotting the modified data

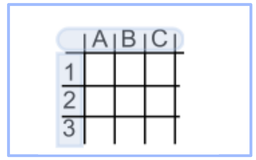




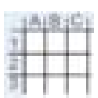
Now we have values in Celsius



ASCII Table Data



- ASCII file with data arranged with one variable per column
- Can contain a header
- CSV files can be handled as Table Data
- Geopoints files can be treated as Table Data as well

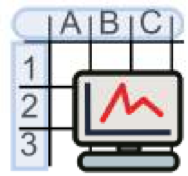


data.csv

```
latitude,longitude,fc,an,fc-an
90,0,-30.29,-25.81,4.48
90,4,-30.29,-25.81,4.48
90,8,-30.29,-25.81,4.48
90,12,-30.29,-25.81,4.48
90,16,-30.29,-25.81,4.48
90,20,-30.29,-25.81,4.48
90,24,-30.29,-25.81,4.48
```

Plotting Table Data

- Table Data plotting is based on the Table Visualiser icon



- It defines the way columns are used for plotting

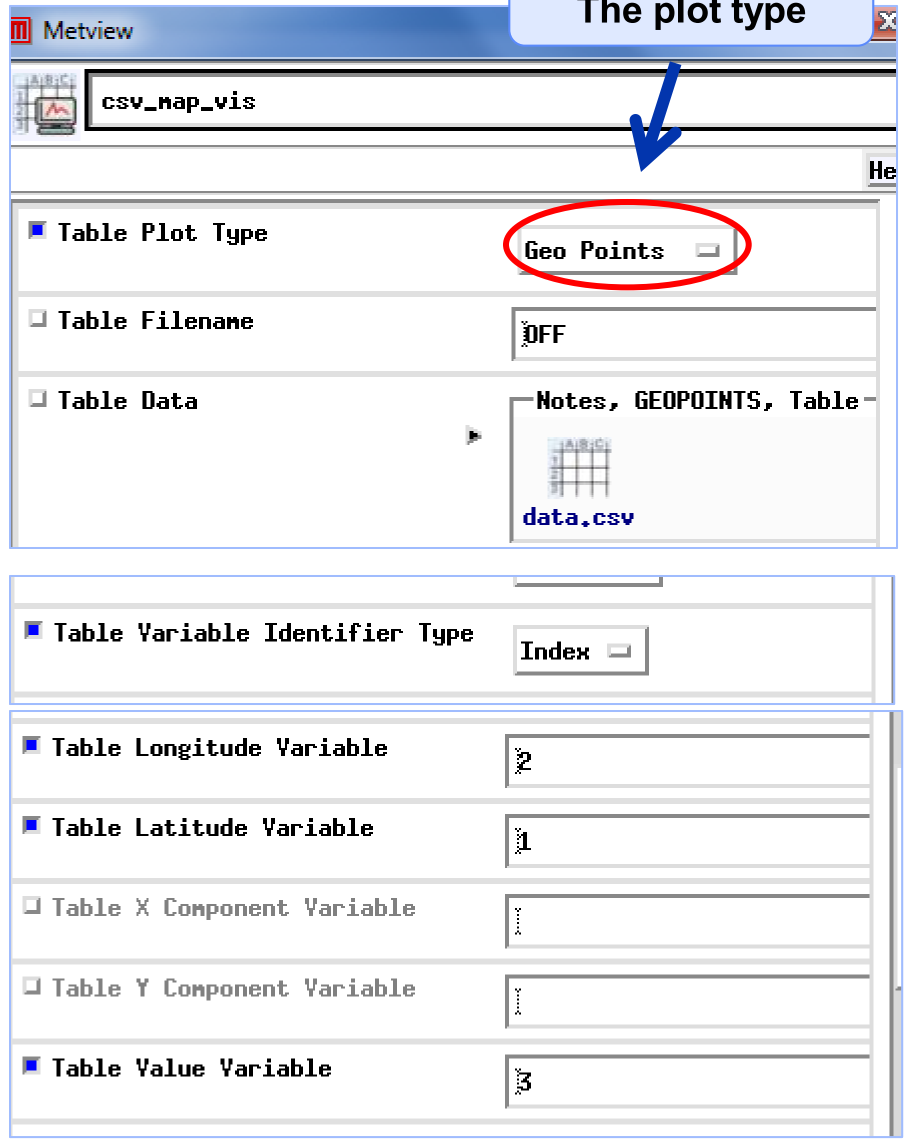
Plotting Table Data

__Example:__ plot the forecast values from file data.csv

csv_map_vis

We need to tell the visualiser which columns should be used from the file

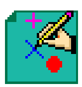
The plot type



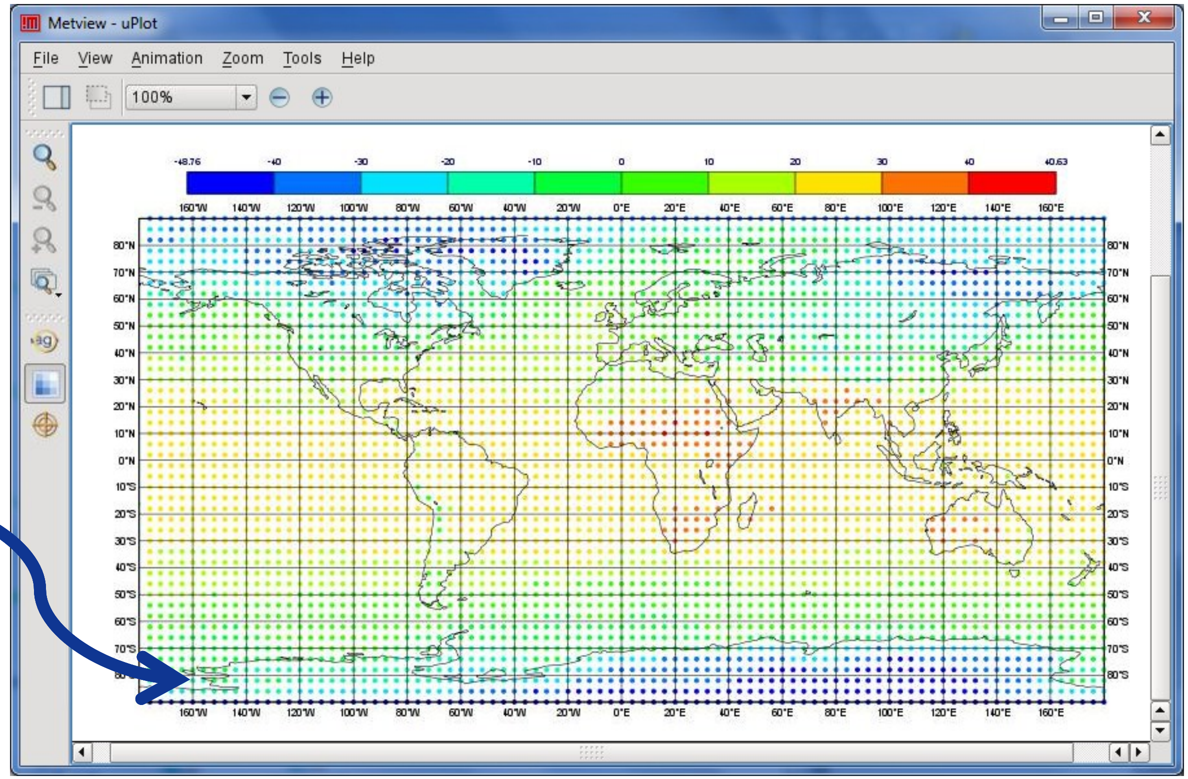
Plotting Table data

csv_map_vis

- execute
- visualise
- examine
- save
- analyse



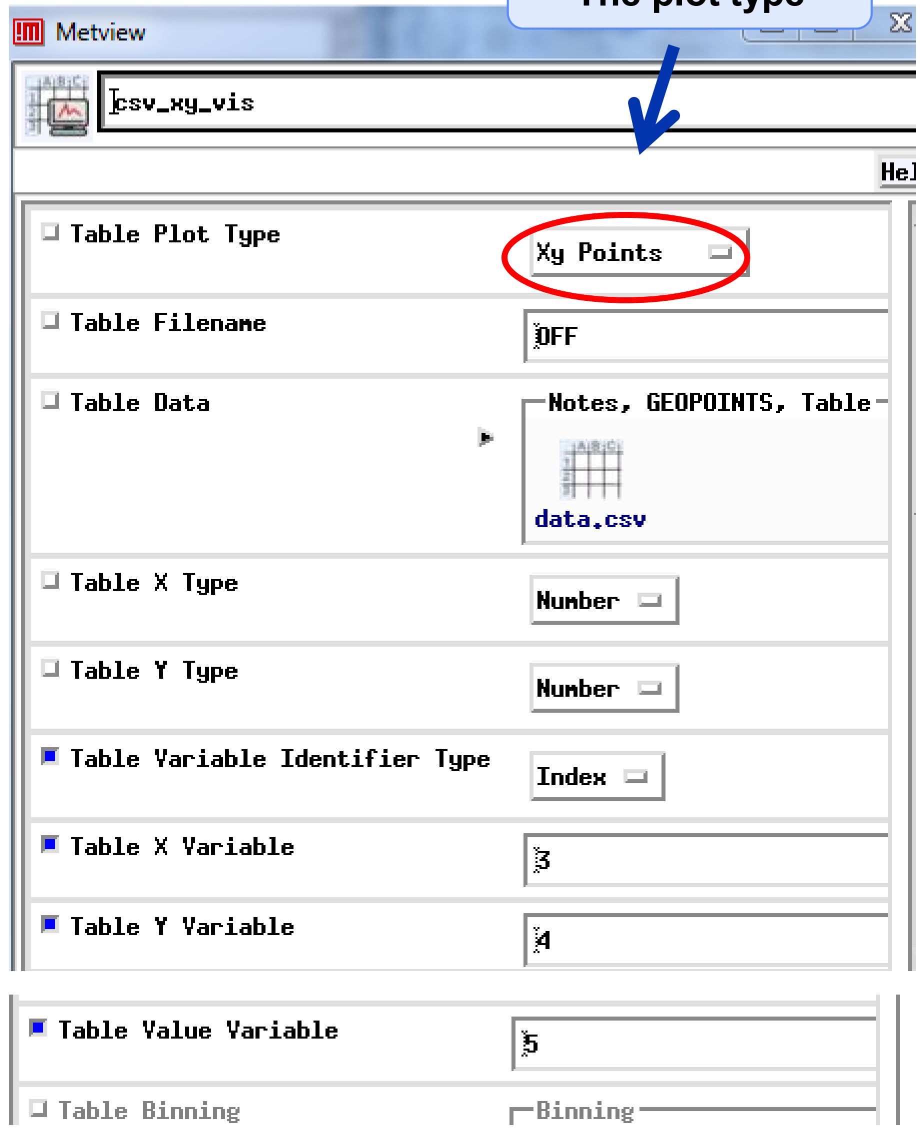
t_sym



Scatterplots from Table data

Example: generate a scatterplot from our CSV file with forecast in X axis and analysis in Y axis, and values (for colouring) taken from fc-an.

The plot type

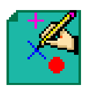


We need to tell the visualiser which columns should be used for X, Y and value

Scatterplots from Table data

csv_xy_vis

execute
visualise
examine
save
analyse



t_diff

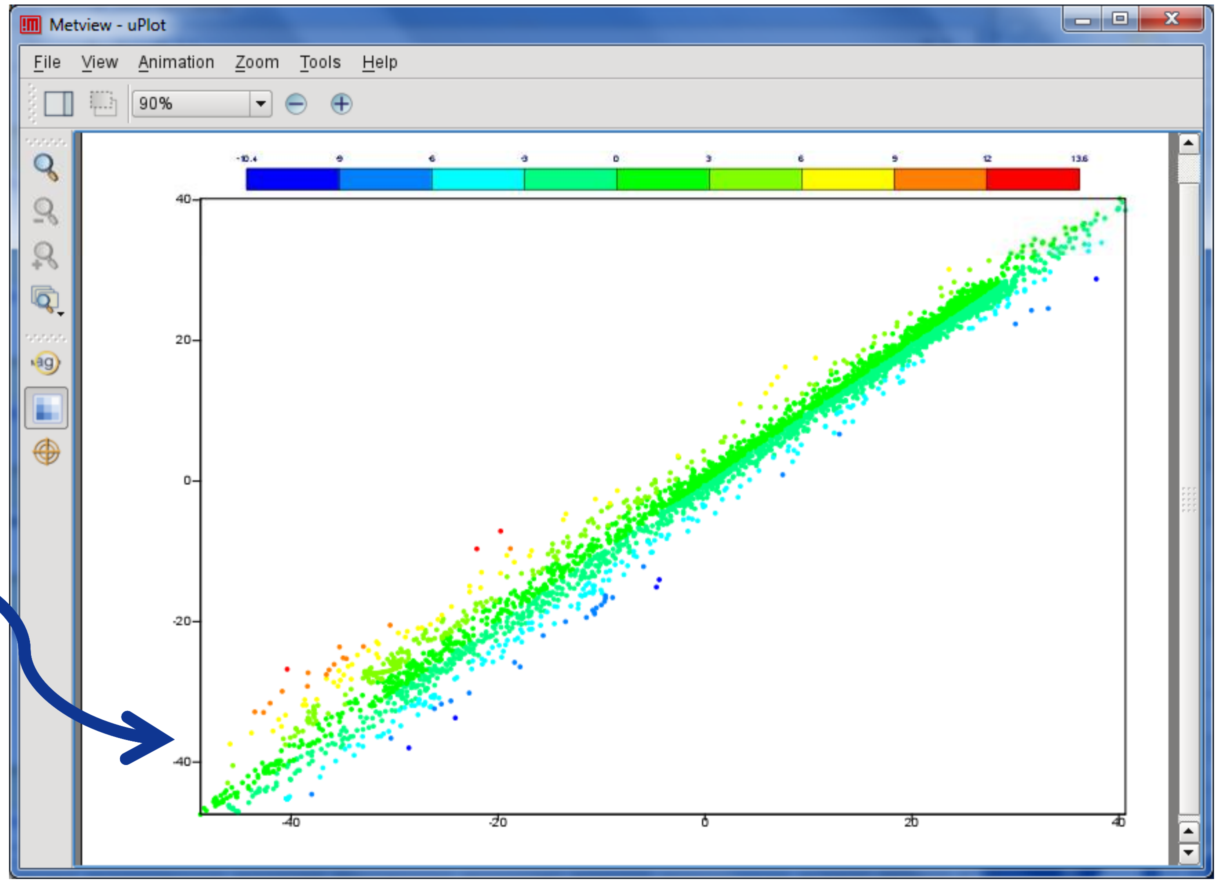


Table Data: macro usage

Example: compute the mean of the forecast-analysis values from our CSV file (fifth column)

```
#Metview Macro

#Read netcdf file
t=read_table(table_filename: "data.csv")

#Read the fifth column into a vector
v = values(t,5)

#Print mean
print("mean=",mean(v))
```

The output of the macro →

```
mean=-0.0265241545894
```

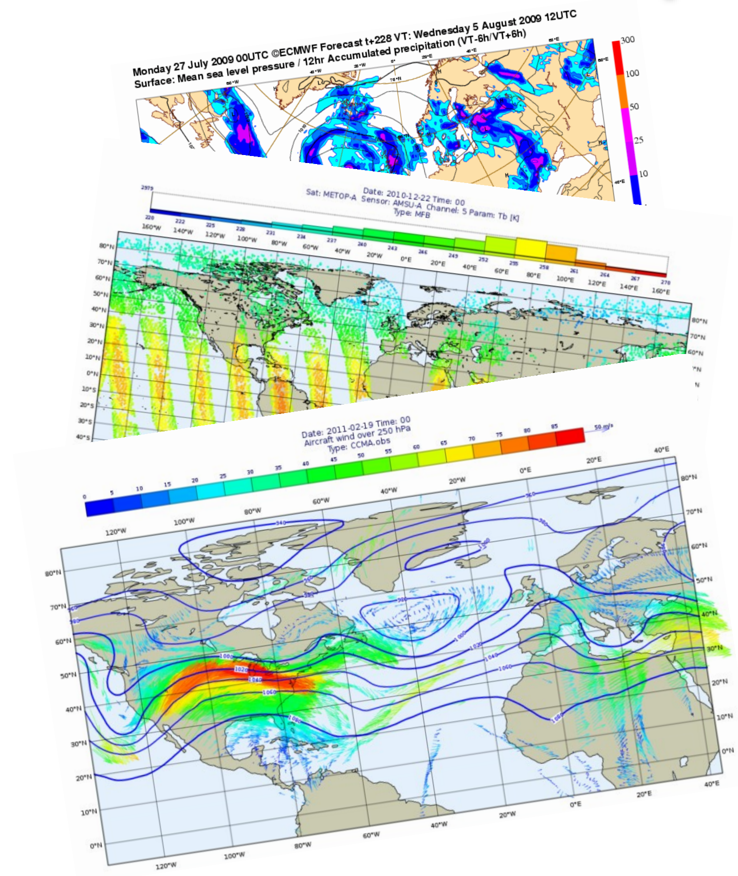
For more information ...

email us:

- **Metview:** metview@ecmwf.int

visit our web pages:

- **https://software.ecmwf.int/metview**
  - **Documentation and tutorials**
  - **Download the virtual machine**



<u>Thursday, 5th December, 9:30 AM UTC: Q&A</u>

www.hipchat.com/gRuxxenlY