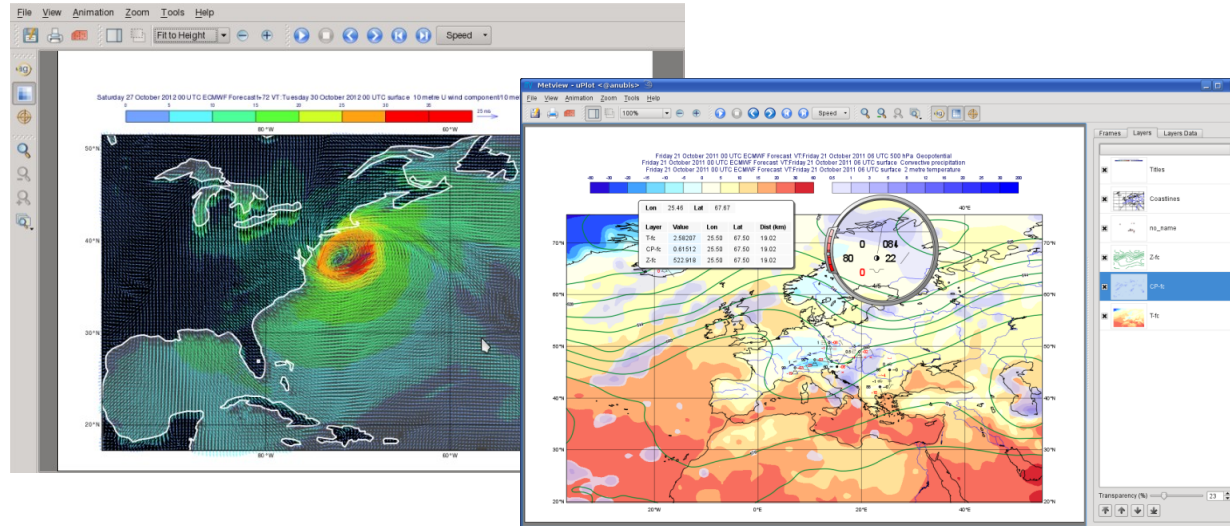


Data Handling with Metview

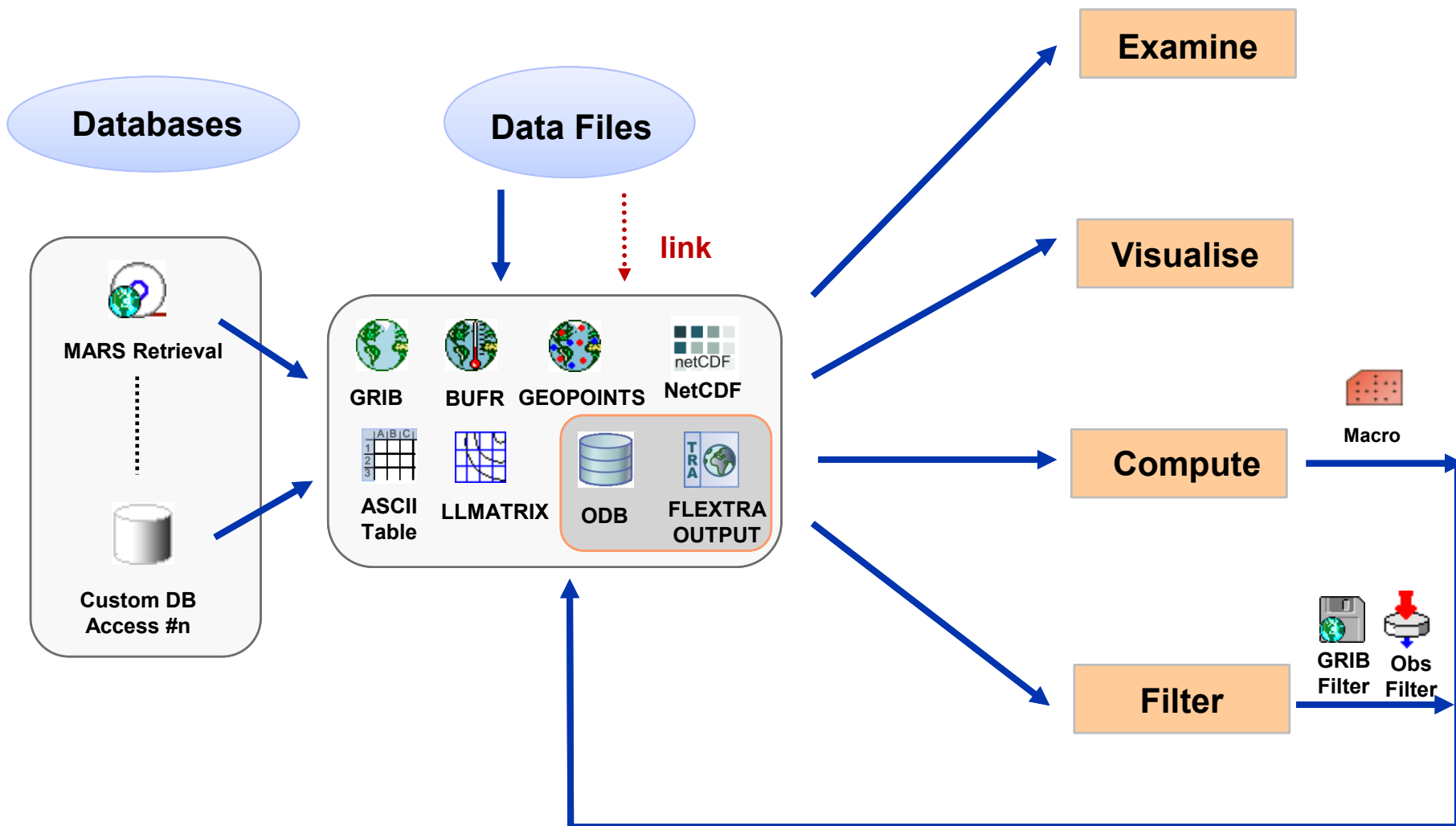


mv⁴

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**



fc_surf.grib

- execute
- visualise
- examine
- save

File: /home/graphics/cgr/metview/webinar_data/fc_surf.grib
 Permissions: -rw-r----- Owner: cgr Group: graphics Size: 2.6MB Modified: 2013-11-26 14:29
 Total number of messages: 45

Index	Name	Date	Time	Step	Level	LevType
01	2t	20120511	0000	0	0	sfc
02	msl	20120511	0000	0	0	sfc
03	tp	20120511	0000	0	0	sfc
04	10u	20120511	0000	0	0	sfc
05	10v	20120511	0000	0	0	sfc
06	2t	20120511	0000	12	0	sfc
07	msl	20120511	0000	12	0	sfc
08	tp	20120511	0000	12	0	sfc
09	10u	20120511	0000	12	0	sfc
10	10v	20120511	0000	12	0	sfc
11	2t	20120511	0000	24	0	sfc
12	msl	20120511	0000	24	0	sfc
13	tp	20120511	0000	24	0	sfc
14	10u	20120511	0000	24	0	sfc

Key name (GRIB API)	Value	Description
editionNumber	1	
table2Version	128	
indicatorOfParameter	167	
indicatorOfTypeOfLevel	1	
level	0	
timeRangeIndicator	1	
subCentre	0	

Log
 Status: OK
 Task: Generating default dump for message: 1
 Command: /usr/local/apps/Metview/AuxSW/grib_api/1.11.0-64/bin/grib_dump -w count=1 "/home/graphics/cgr/metview/webinar_data/fc_surf.grib"
 Status: OK

Different dumps for the selected message

Index	Name	Date	Time	Step	Level	LevType
01	2t	20120511	0000	0	0	sfc
02	msl	20120511	0000	0	0	sfc
03	tp	20120511	0000	0	0	sfc

Message list

GRIB Examiner – Values dump

Dump mode: Values

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

All the values for the selected message

GRIB Examiner – WMO-style dump

Dump mode: WMO-style

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

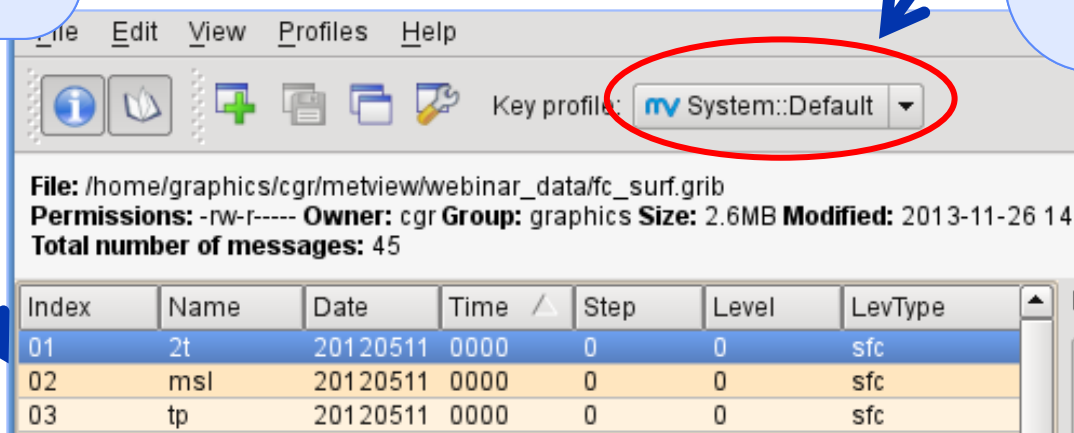
Key name (GRIB API)	Key type	Value
geography.bitmapPresent	long	0
geography.gridType	string	regular_ll
geography.iDirectionIncrementInDegrees	double	1.5
geography.iScansNegatively	long	0
geography.jDirectionIncrementInDegrees	double	1.5
geography.jPointsAreConsecutive	long	0
geography.jScansPositively	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**.

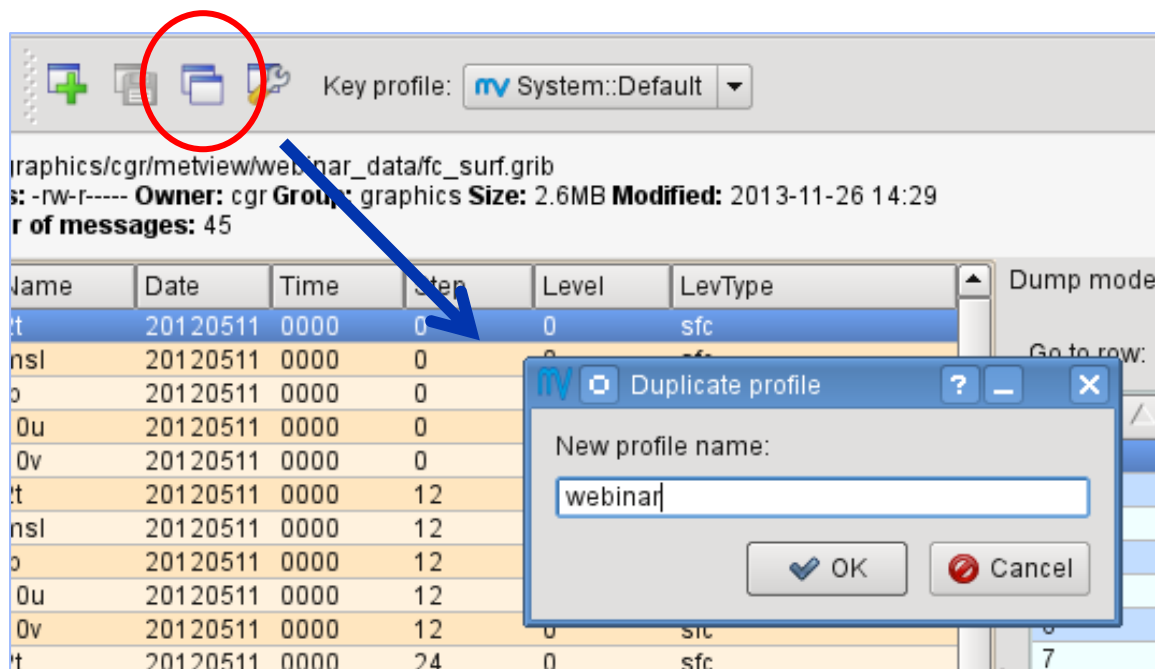
The default profile is called **System::Default**. It is **read-only!** However you can create any number of additional profiles.



File: /home/graphics/cgr/metview/webinar_data/fc_surf.grib
Permissions: -rw-r----- Owner: cgr Group: graphics Size: 2.6MB Modified: 2013-11-26 14:
Total number of messages: 45

Index	Name	Date	Time	Step	Level	LevType
01	2t	20120511	0000	0	0	sfc
02	msl	20120511	0000	0	0	sfc
03	tp	20120511	0000	0	0	sfc

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

Key profile: webinar

ne	Step	Level	LevType	units
00	0	0	sfc	K
00	0	0	sfc	Pa
00	0	0	sfc	m
00	0	0	sfc	m s**-1
00	0	0	sfc	m s**-1
00	12	0	sfc	K
00	12	0	sfc	Pa
00	12	0	sfc	m
00	12	0	sfc	m s**-1
00	12	0	sfc	m s**-1
00	24	0	sfc	K
00	24	0	sfc	Pa
00	24	0	sfc	m
00	24	0	sfc	m s**-1
00	24	0	sfc	m s**-1
00	36	0	sfc	K
00	36	0	sfc	Pa
00	36	0	sfc	m
00	36	0	sfc	m s**-1
00	36	0	sfc	m s**-1
00	48	0	sfc	K
00	48	0	sfc	Pa
00	48	0	sfc	m
00	48	0	sfc	m s**-1
00	48	0	sfc	m s**-1
00	60	0	sfc	K
00	60	0	sfc	Pa
00	60	0	sfc	m

GRIB API Namespace: Default

Key name (GRIB API)	type	value
shortName		
shortNameECMF		
skewness		
sphericalHarmonics		
standardDeviation		
startStep		
stepRange		
stepType		
stepUnits	long	
subCentre	long	0
table2Version	long	128
tableReference	long	0
thousand	long	1000
timeRangeIndicator	long	1
totalLength	long	5818
typeOfLevel	string	surface
unitOfTimeRange	long	h
units	string	K
unitsECMF	string	K
unpackedError	double	1.52
uvRelativeToGrid	long	0
validityDate	long	2013
validityTime	long	0
values	double	Array

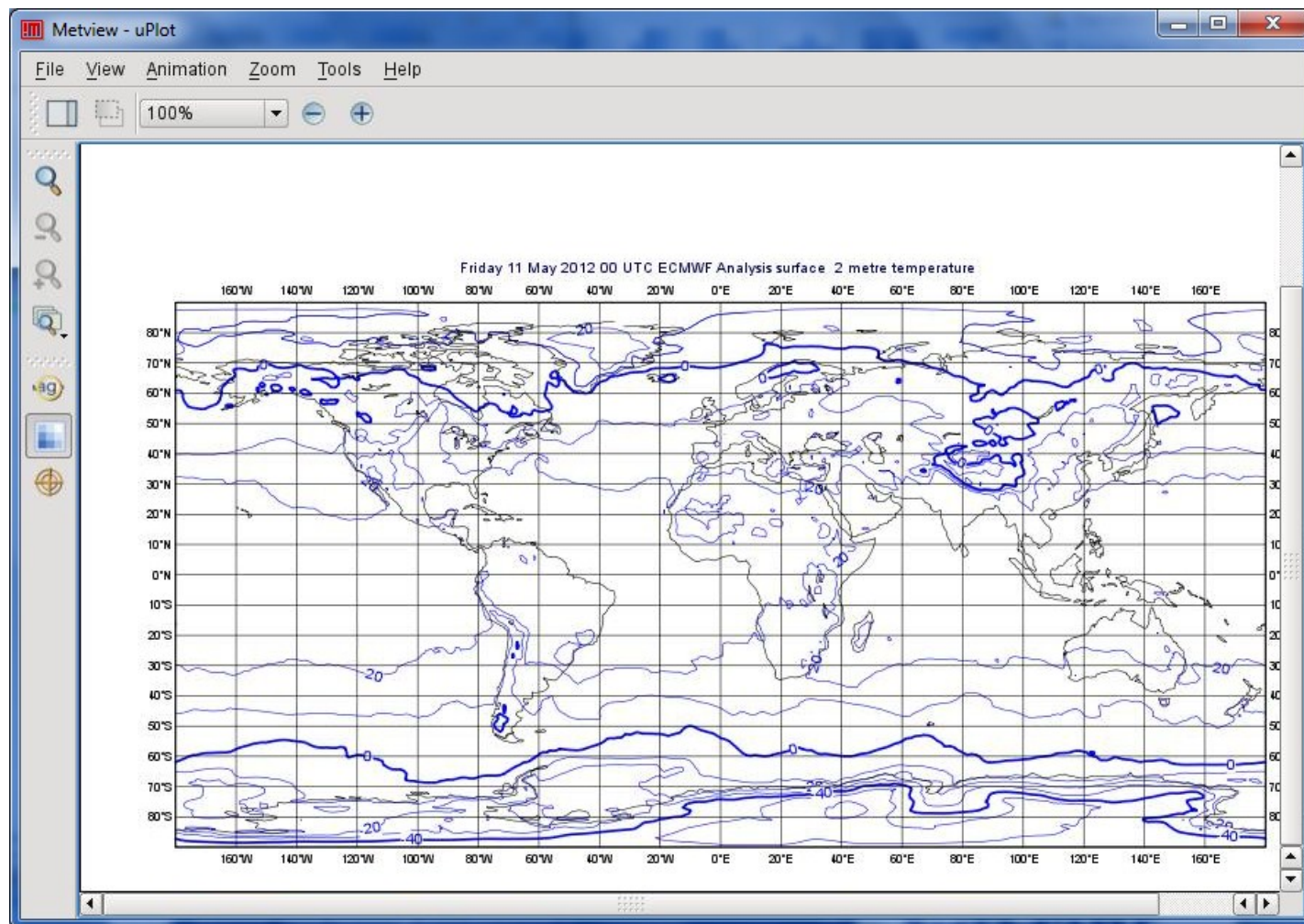
Status: OK

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

fc_surf.grib

- execute
- visualise
- examine
- save



Overlaying fields from the same GRIB file

Example: overlay T2 and MSLP forecasts from file `fc_surf.grib`

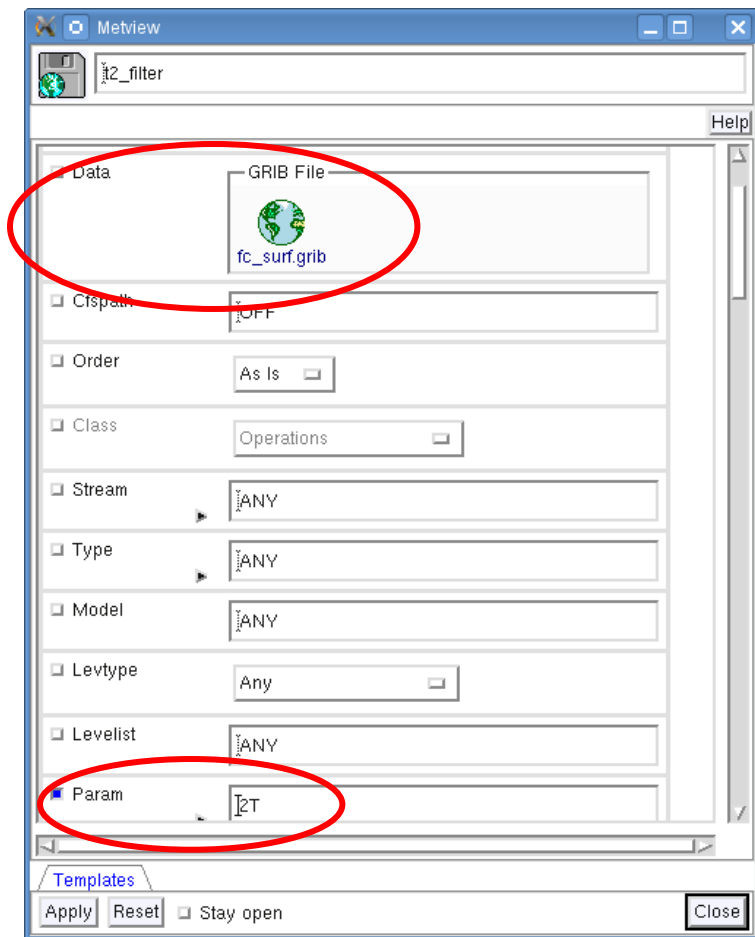
- We need to **filter** out each parameter into a separate file
- We will use the **GRIB Filter** icon



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



File Edit View Profiles Help

Index	Name	Date	Time	Step	Level	Le
01	2t	20120511	0000	0	0	sf
02	mssl	20120511	0000	0	0	sf
03	tp	20120511	0000	0	0	sf
04	10u	20120511	0000	0	0	sf
05	10v	20120511	0000	0	0	sf
06	2t	20120511	0000	12	0	sf
07	mssl	20120511	0000	12	0	sf
08	tp	20120511	0000	12	0	sf
09	10u	20120511	0000	12	0	sf
10						sf
11						sf

The original GRIB



File Edit View Profiles Help

Index	Name	Date	Time	Step	Level
01	2t	20120511	0000	0	0
02	2t	20120511	0000	12	0
03	2t	20120511	0000	24	0
04	2t	20120511	0000	36	0
05	2t	20120511	0000	48	0
06	2t	20120511	0000	60	0
07	2t	20120511	0000	72	0
08	2t	20120511	0000	84	0
09	2t	20120511	0000	96	0

The resulting GRIB

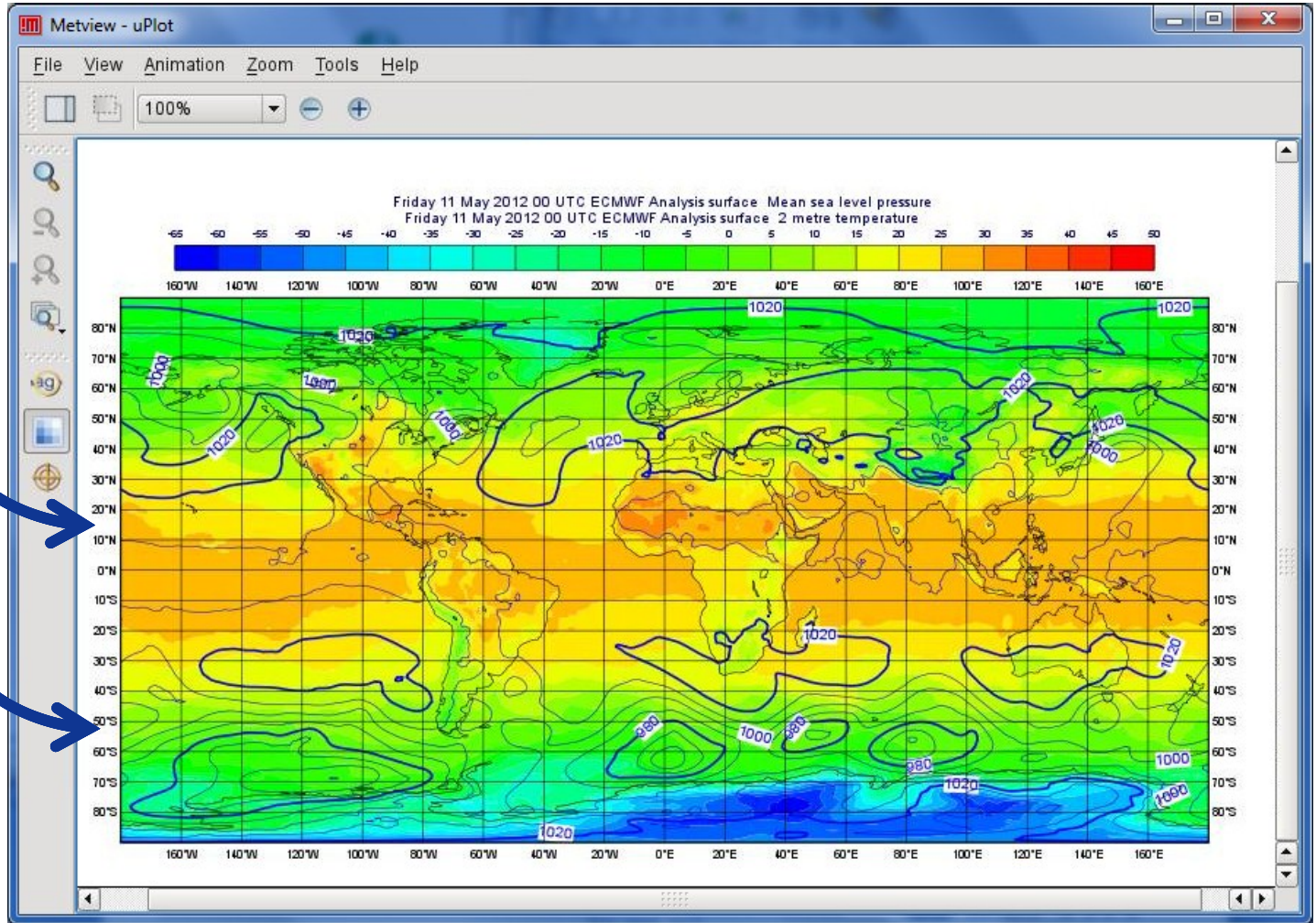
Overlaying GRIB fields

t2_filter

- execute
- visualise
- examine
- save

t2_shade

mstp_filter



Overlaying GRIB fields

Metview - uPlot

File View Animation Zoom Tools Help

100%

Friday 11 May 2012 00 UTC ECMWF Analysis surface Mea
Friday 11 May 2012 00 UTC ECMWF Analysis surface 2 r

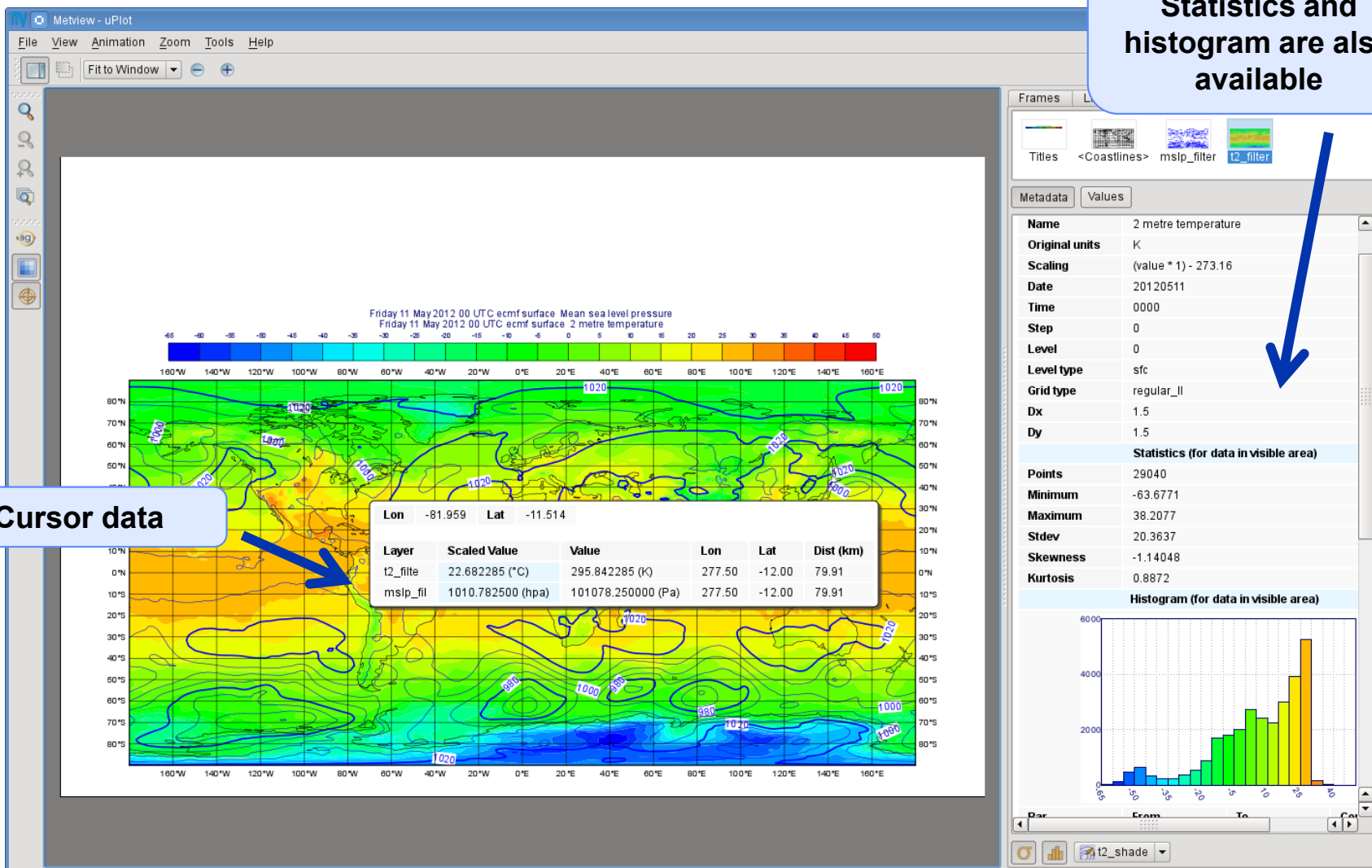
Frame	Name	Date	Time	Sta
1	2t	20120511	0000	0
	msl	20120511	0000	0
2	2t	20120511	0000	12
	msl	20120511	0000	12
3	2t	20120511	0000	24
	msl	20120511	0000	24
4	2t	20120511	0000	36
	msl	20120511	0000	36
5	2t	20120511	0000	48
	msl	20120511	0000	48
6	2t	20120511	0000	60
	msl	20120511	0000	60
7	2t	20120511	0000	72
	msl	20120511	0000	72
8	2t	20120511	0000	84
	msl	20120511	0000	84
9	2t	20120511	0000	96
	msl	20120511	0000	96

Key profile: mv System::Default

GRIB API key profiles are used to described each animation frame

GRIB data inspection

Statistics and histogram are also available



GRIB scaling for plotting

Name	2 metre temperature
Original units	K
Scaling	(value * 1) - 273.16
Date	20120511
Time	0000
Step	0
Level	0
	sfc
	regular_ll
	1.5
	1.5
Statistics (for data in visible area)	
	29040
	-63.6771
Maximum	38.2077

Lon	-81.959	Lat	-11.514			
Layer	Scaled Value	Value	Lon	Lat	Dist (km)	
t2_filte	22.682285 (°C)	295.842285 (K)	277.50	-12.00	79.91	
mslp_fil	1010.782500 (hpa)	101078.250000 (Pa)	277.50	-12.00	79.91	

**Both T2 and MSLP
are scaled**

GRIB scaling for plotting



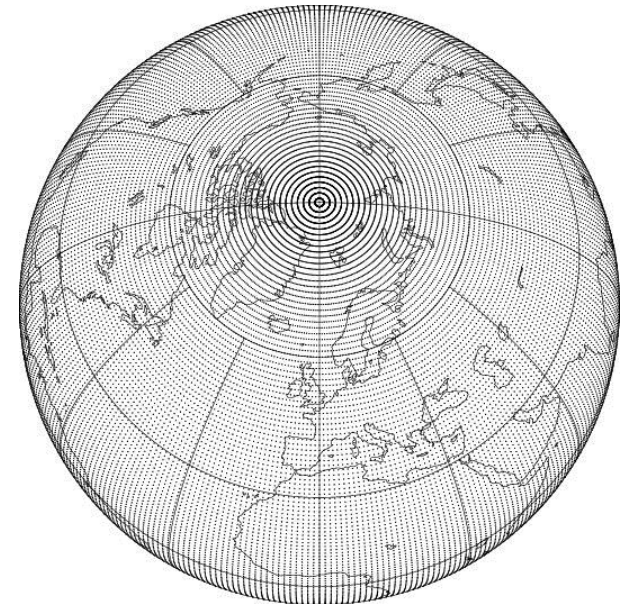
t2_shade

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

Forecast
1.5x1.5
global grid



an_upper.grb

Analysis at D4
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**

```
1 #Metview Macro
2
3 #Read T500 analysis
4 g_an=read(source: "an_upper.grib",
5           param: "t",
6           level: 500)
7
8 #Read target area borders
9 lat1=grib_get_double(g_an, "latitudeOfFirstGridPointInDegrees") #north
10 lat2=grib_get_double(g_an, "latitudeOfLastGridPointInDegrees") #south
11 lon1=grib_get_double(g_an, "longitudeOfFirstGridPointInDegrees") #west
12 lon2=grib_get_double(g_an, "longitudeOfLastGridPointInDegrees") #east
13
14 #Read target grid resolution
15 dx=grib_get_double(g_an, "iDirectionIncrementInDegrees")
16 dy=grib_get_double(g_an, "jDirectionIncrementInDegrees")
17
```

Here we read a set of GRIB API keys

Macro: Compute difference #2

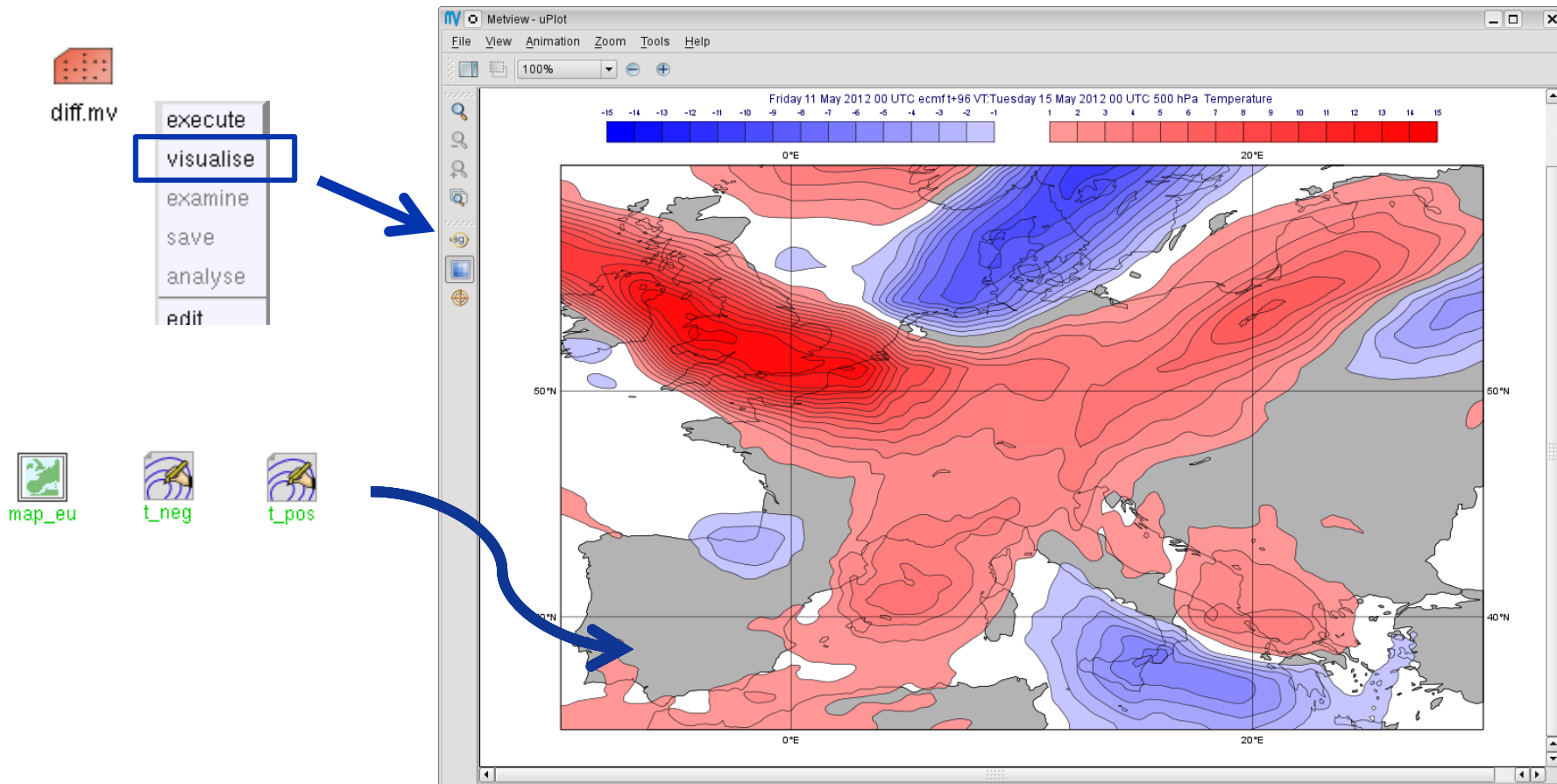
Filter parameter and level and step from forecast

```
18 #Read T500 forecast for the analysis date (+96h)
19 #and interpolate it to the target grid
20 g_fc=read(source: "fc_upper.grib",
21           param: "t",
22           level: 500,
23           step: 96,
24           area: [lat2,lon1,lat1,lon2], #[s,w,n,e]
25           grid: [dx,dy] )
26
27 #Compute difference
28 g_res=g_fc-g_an
29 |
30 #Return results
31 return g_res
```

Interpolation

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

Fieldset
operation

Here we set
the GRIB
header

```
1 #Metview Macro
2
3 #read GRIB
4 g=read(source: "fc_surf.grib")
5
6 #Filter 10u
7 u=read(data: g,
8       param: "10u"
9       )
10
11 #Filter 10v
12 v=read(data: g,
13       param: "10v"
14       )
15
16 #Compute sp
17 sp=sqrt(u*u+v*v)
18
19 #Set shortName to the correct value
20 sp=grib_set_string(sp, ["shortName", "10si"])
21
22 #Return results
23 return sp
```

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`

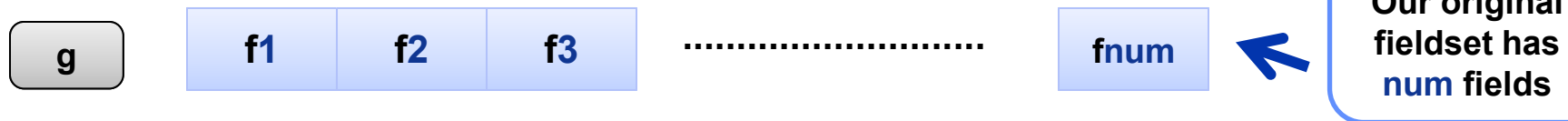
We filter `tp`

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

```
1 #Metview Macro
2
3 #Read tp fields from the GRIB file
4 g=read(
5     source: "fc_surf.grib",
6     param: "tp"
7 )
8
9 #Number of tp messages
10 num=count(g)
11
12 #Compute tp for 12h intervals
13 gr=g[2,num] - g[1,num-1]
14
15 #Return results
16 return gr
17
```



Precipitation computation explained



```

12 #Compute tp for 12h intervals
13 gr=g [2, num] - g [1, num-1]
14

```



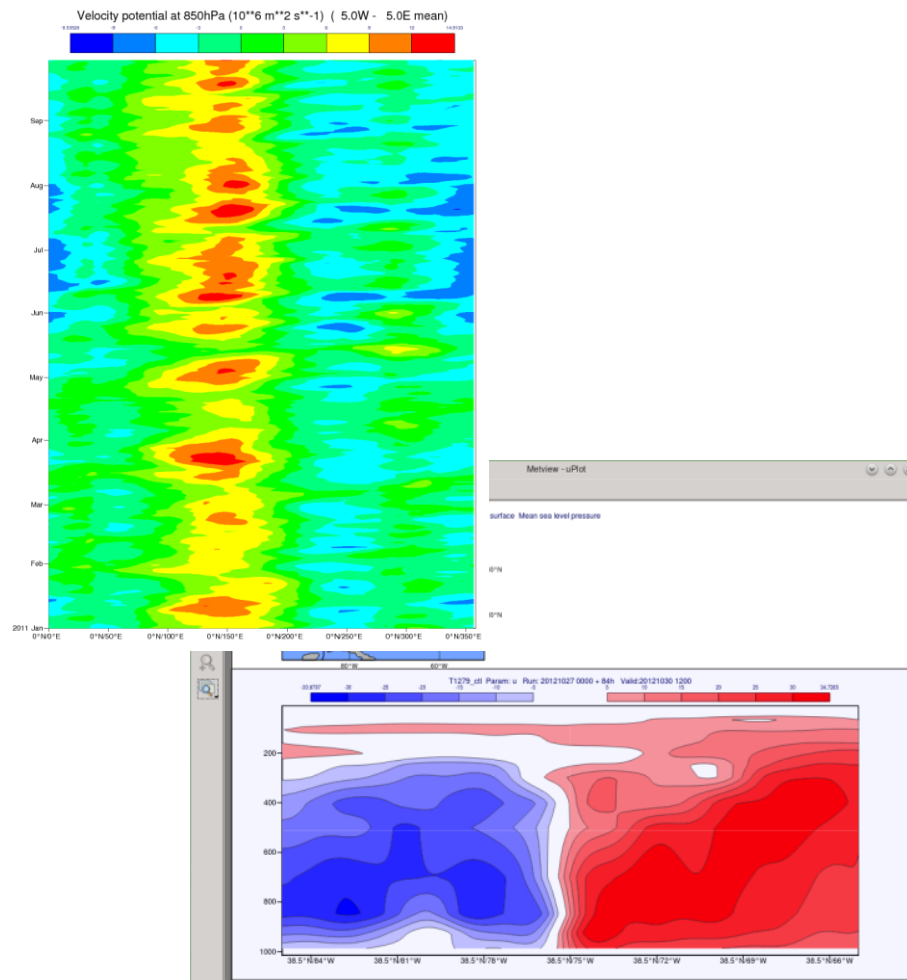
Macro usage: more functions

- 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



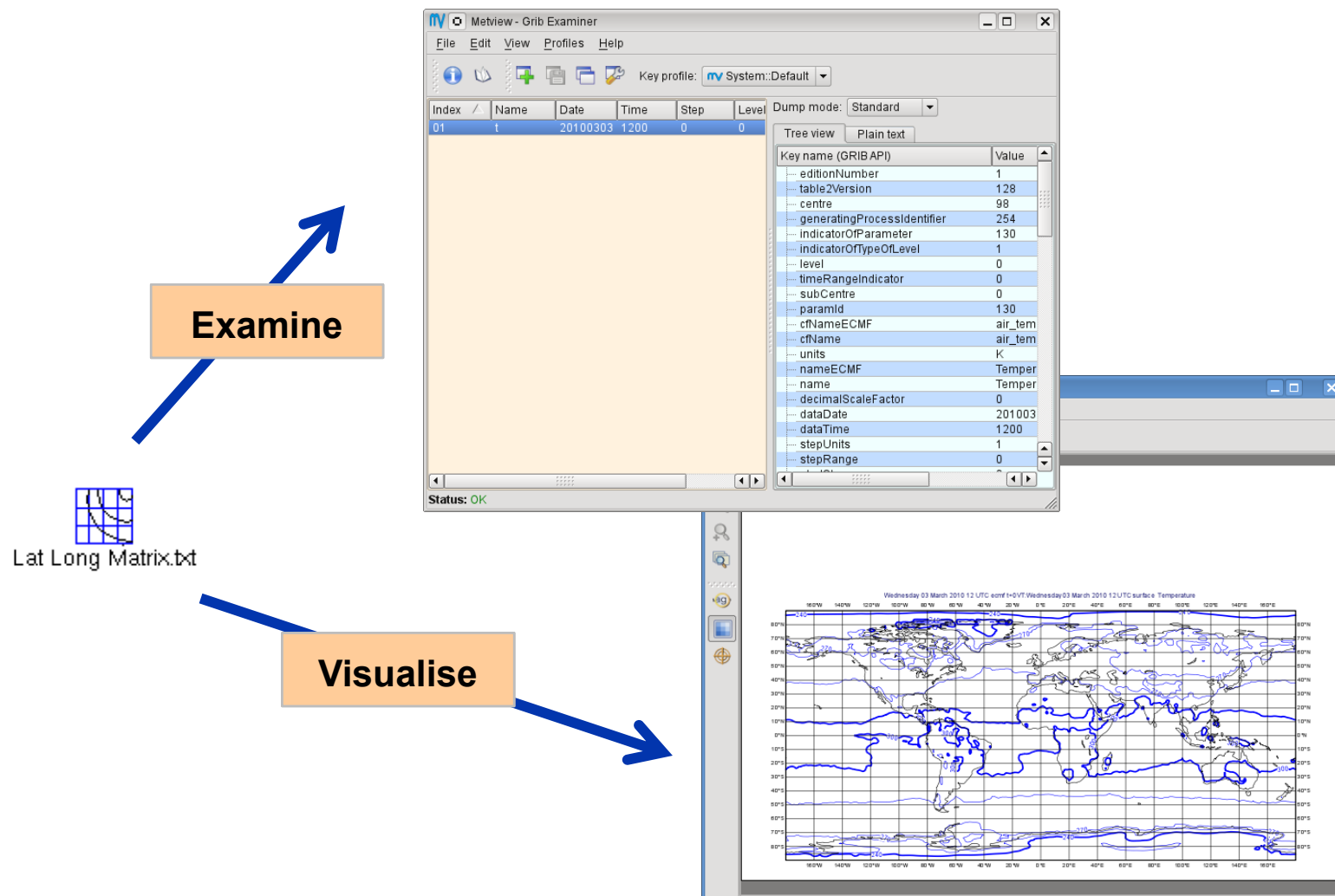
- Metview's ASCII format for gridded data
- Turned into GRIB internally
- Can be edited as a text file

```

#LLMATRIX
DATE=20100303.5
NORTH=90
WEST=0
NLAT=91
NLON=180
GRID=2/2
CENTRE=98
PARAM=130
TABLE2=128
MISSING=-9999
#DATA
239.044082642 239.044082642 239.044082642 239.044082642 239.044082642
239.442520142 239.485488892 239.532363892 239.583145142 239.637832642
239.438613892 239.348770142 239.317520142 239.325332642 239.372207642
240.747207642 240.598770142 240.520645142 240.497207642 240.352676392
243.215957642 243.575332642 244.196426392 244.161270142 244.028457642
258.817520142 275.290176392 275.680801392 275.708145142 275.993301392
274.165176392 274.223770142 274.403457642 275.712051392 276.005020142
272.407363892 272.719863892 273.399551392 274.524551392 275.649551392

```

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**



```
execute
visualize
examine
save
```

Message list

Index	Typ	Sut	C	Ssc	Date	Time	Lat1
1	0	1	98	1	2012-05-15	00:00	36.15
2	0	1	98	1	2012-05-15	00:00	35.85
3	0	1	98	1	2012-05-15	00:00	41.97
4	0	1	98	1	2012-05-15	00:00	54.18
5	0	1	98	1	2012-05-15	00:00	54.52
6	0	1	98	1	2012-05-15	00:00	51.43
7	0	1	98	1	2012-05-15	00:00	51.13
8	0	1	98	1	2012-05-15	00:00	50.37
9	0	1	98	1	2012-05-15	00:00	50.05
10	0	1	98	1	2012-05-15	00:00	48.68
11	0	1	98	1	2012-05-15	00:00	49.5
12	0	1	98	1	2012-05-15	00:00	54.38
13	0	1	98	1	2012-05-15	00:00	53.72
14	0	1	98	1	2012-05-15	00:00	54.18
15	0	1	98	1	2012-05-15	00:00	54.18
16	0	1	98	1	2012-05-15	00:00	53.03
17	0	1	98	1	2012-05-15	00:00	52.13
18	0	1	98	1	2012-05-15	00:00	52.38
19	0	1	98	1	2012-05-15	00:00	52.57
20	0	1	98	1	2012-05-15	00:00	51.18
21	0	1	98	1	2012-05-15	00:00	51.8
22	0	1	98	1	2012-05-15	00:00	51.8
23	0	1	98	1	2012-05-15	00:00	51.8
24	0	1	98	1	2012-05-15	00:00	51.8

Dumps for all the sections in the message

Index	Descriptor	Name	Value	Units
0	01001	Wmo Block Number	8	NUMERIC
1	01002	Wmo Station Number	495	NUMERIC
2	02001	Type Of Station	1	
3	04001	Year	2012	
4	04002	Month	5	
10	07001	Height Of Station (See Note 1)	0	M
11	10004	Pressure	101460	PA
12	10051	Pressure Reduced To Mean Sea Level	101540	PA
13	10061	3-Hour Pressure Change	-10	PA
14	10063	Characteristic Of Pressure Tendency	8	CODE TABL
15	11011	Wind Direction At 10 M	0	DEGREE TR
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	[Missing]	%
20	20001	Horizontal Visibility	20000	M
21	20003	Present Weather (See Note 1)	2	CODE TABL
22	20004	Past Weather (1) (See Note 2)	1	CODE TABL
23	20005	Past Weather (2) (See Note 2)	1	CODE TABL
24	20010	Cloud Cover (Total)	10	%

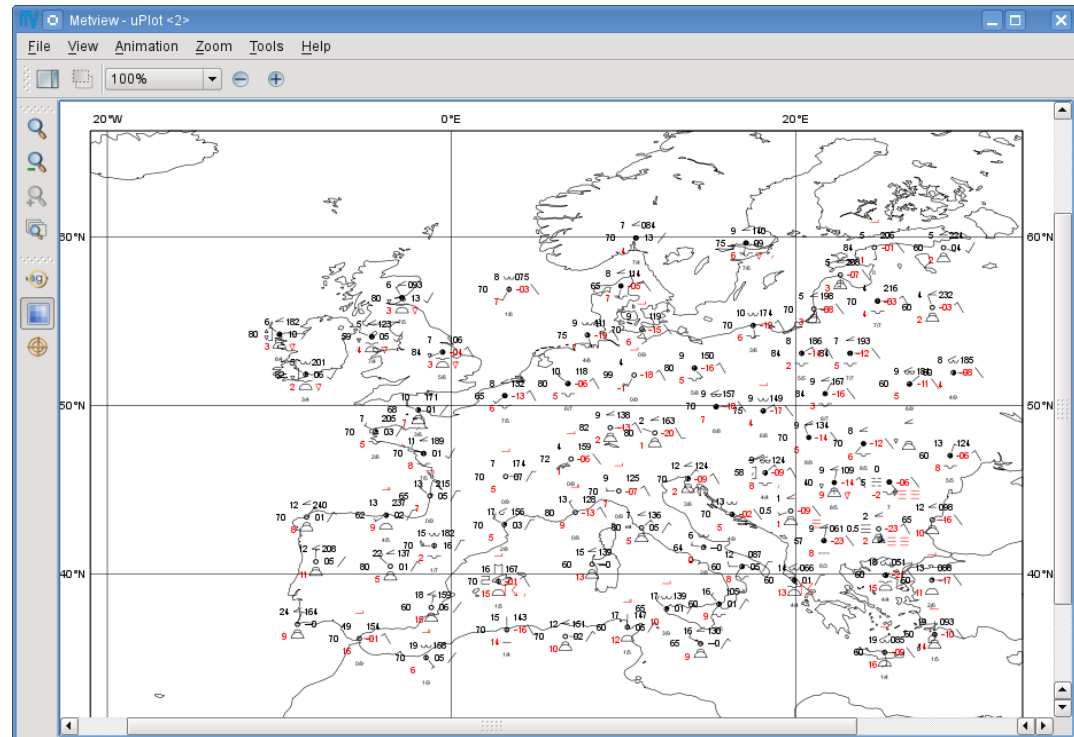
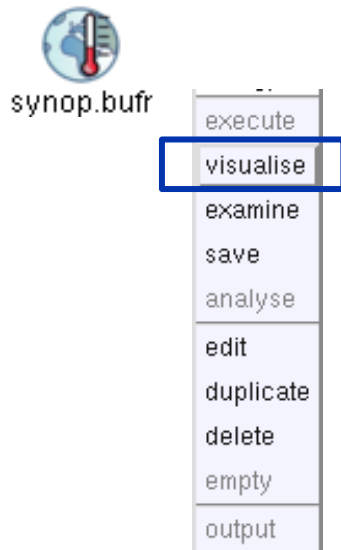
Log

```
Task: Generating BUFR data dump for message: 1 and for subset: 1
Method: BUFRX
Status: OK

Task: Generating BUFR bitmap dump for message: 1 and for subset: 1
Method: BUFRX
Status: OK
```


BUFR Plotting

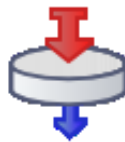
- We can directly visualise BUFR files with conventional observations (e.g. SYNOP)



BUFR: Accessing data

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

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

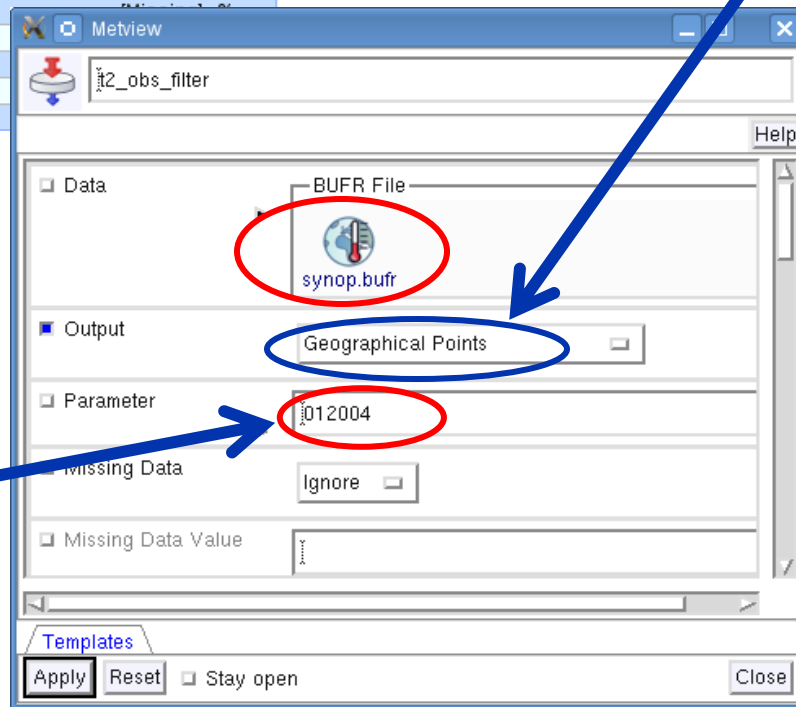


- It can perform filtering according to parameter, level, area, time, channel etc.

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**



Parameters are defined by their BUFR descriptors

We can save the result into a file



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
```

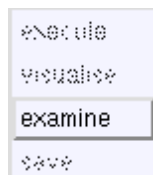
```
41.97    21.65    0       20120515 0000    282.4
```

Geopoints Examiner

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



t2.gpt



Metview - Geopoints Examiner

File Edit View Help

File: /home/graphics/cgr/metview/webinar_data/t2.gpt
 Permissions: -rwx-x--- Owner: cgr Group: graphics Size: 28KB Modified: 2013-11-29 09:45
 Format: Traditional
 Total number of points: 660

Meta data

Go to row: 1

Index	Lat_y	Lon_x	Level	Date	Time	Value
1	36.15	-5.35	0	20120515	0	292.4
2	35.85	14.48	0	20120515	0	288.8
3	41.97	21.65	0	20120515	0	282.4
4	54.18	7.9	0	20120515	0	282.5
5	54.53	9.55	0	20120515	0	279.5
6	54.53	11.07	0	20120515	0	282.7
7	53.63	9.98	0	20120515	0	282.7
8	54.1	13.4	0	20120515	0	281.6
9	53.05	8.8	0	20120515	0	281.2
10	52.47	9.68	0	20120515	0	284
11	52.22	14.12	0	20120515	0	282.5
12	51.3	6.77	0	20120515	0	283.3
13	51.43	12.23	0	20120515	0	281.4
14	51.13	13.75	0	20120515	0	279.4
15	50.37	6.87	0	20120515	0	282.1
16	50.05	8.6	0	20120515	0	279.2
17	48.68	9.23	0	20120515	0	282.1
18	49.5	11.05	0	20120515	0	279.3
19	54.38	10.15	0	20120515	0	282.1
20	53.72	7.15	0	20120515	0	282
21	54.18	12.08	0	20120515	0	284.6
22	53.03	14	0	20120515	0	281.6
23	52.13	7.7	0	20120515	0	283
24	52.38	13.07	0	20120515	0	280.9
25	52.57	13.32	0	20120515	0	282.8
26	51.18	8.48	0	20120515	0	279.7
27	51.8	10.62	0	20120515	0	277.7
28	51.17	14.95	0	20120515	0	282.1

Status: OK

Geopoints Plotting

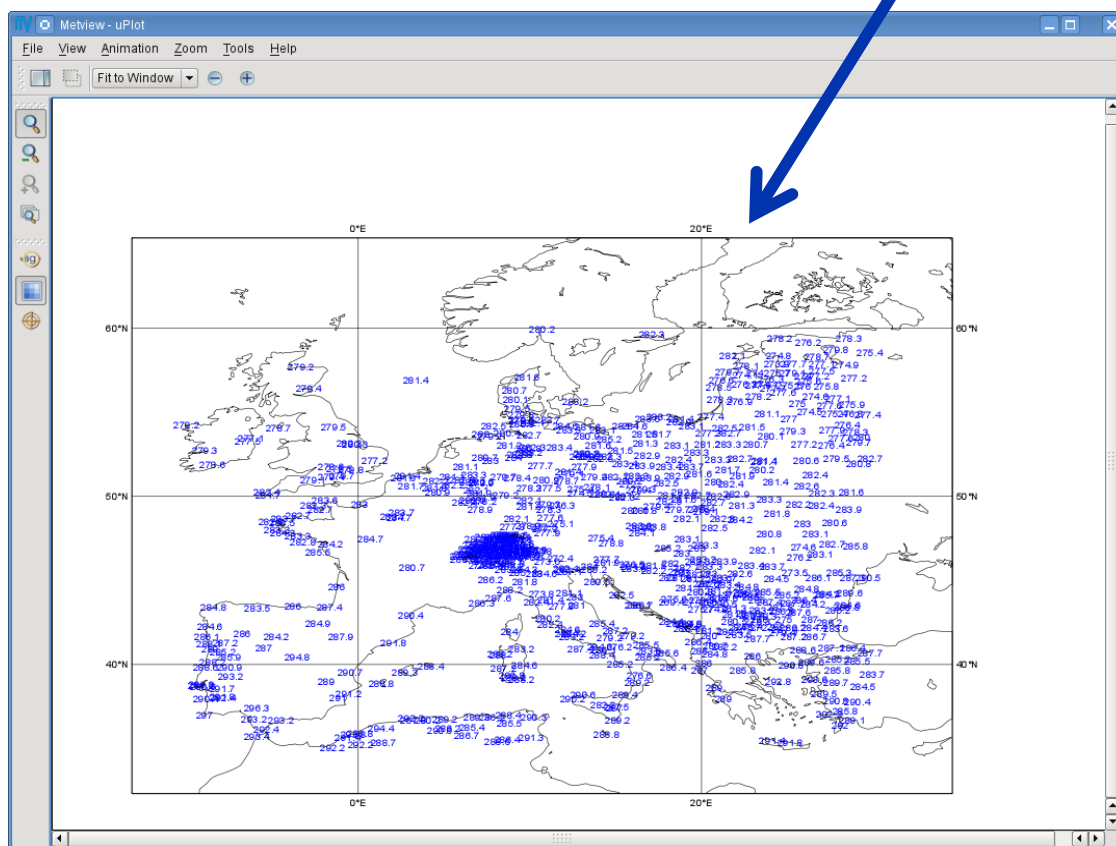
- We can directly visualise Geopoints
- It is based on symbol plotting

By default the numbers are plotted to the map



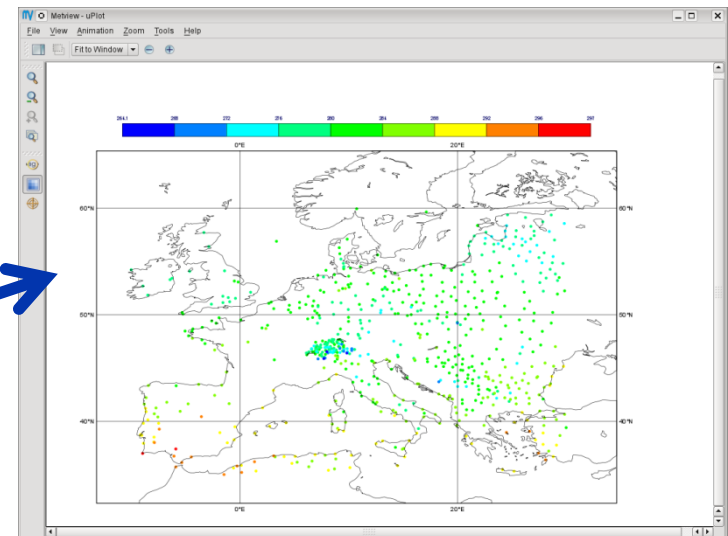
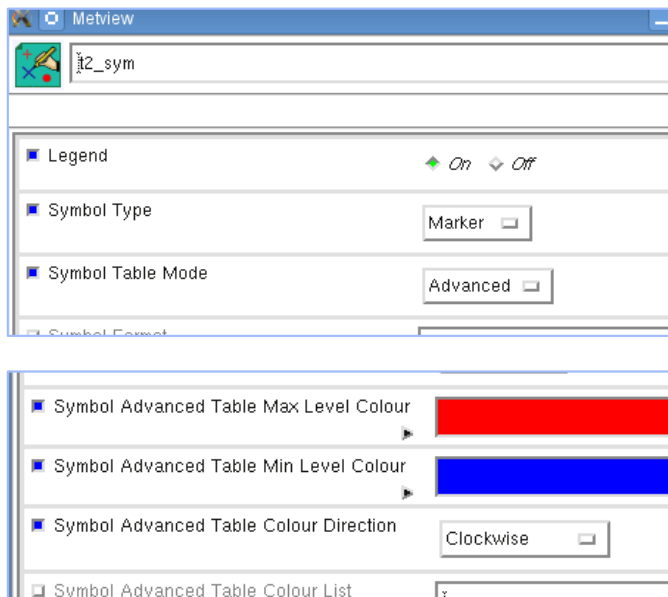
t2.gpt

t2.gpt
execute
visualise
examine
save
analyse
edit
duplicate
delete
empty
output



Customisation with Symbol Plotting

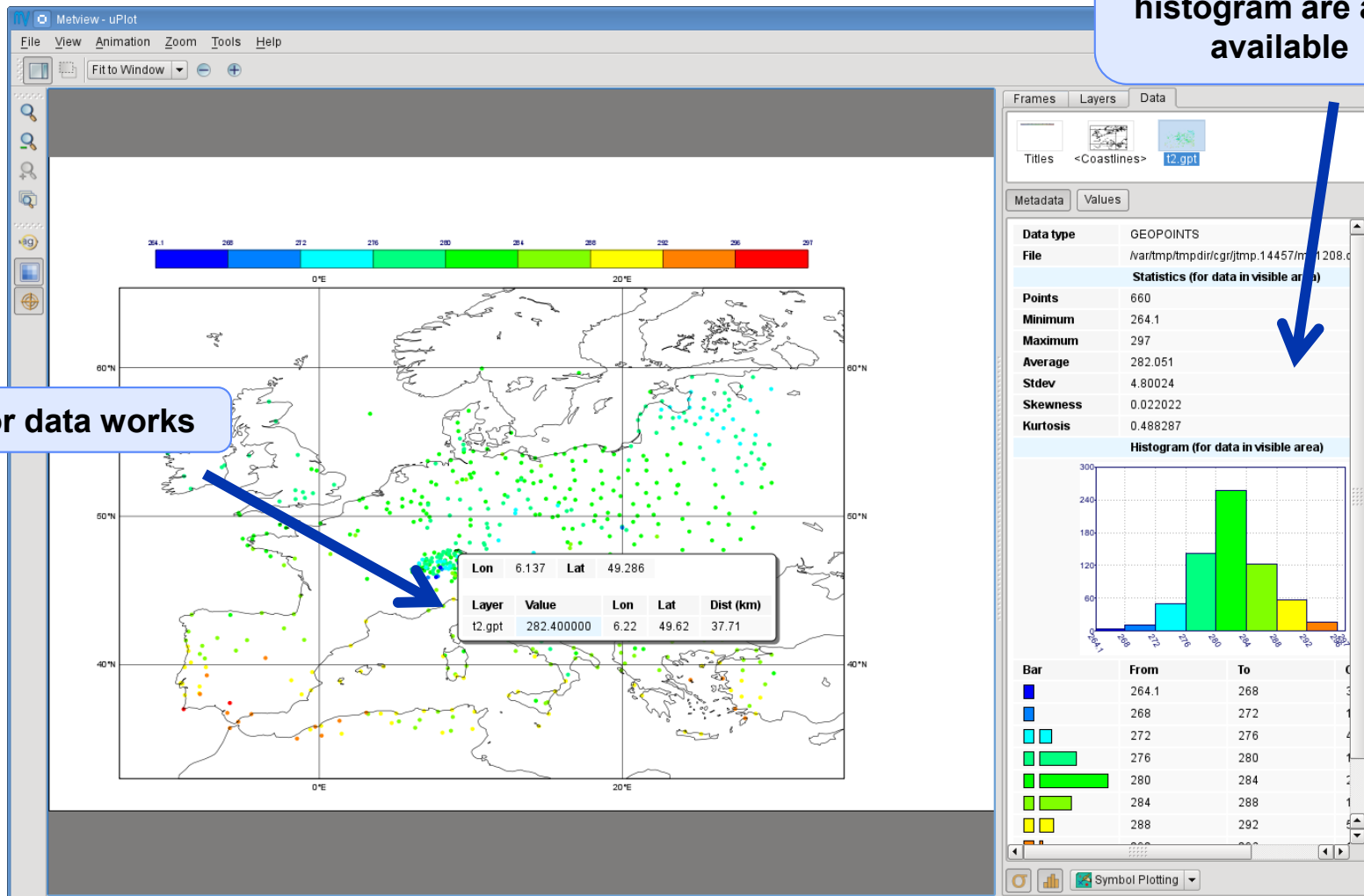
- The **Symbol Plotting** icon offers a large number of options for plot customisation
- We can use the **Advanced Table Mode** to define a nice colour palette between the min and max colours (just like for Contouring)



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

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

The result is
another
Geopoints


grib_minus_geo.mv

```
1 #Metview Macro
2
3 #Read t2 field forecast (96h) from the GRIB file
4 g=read(
5     source: "fc_surf.grib",
6     param: "2t",
7     step: 96
8 )
9
10 #Read observations from geopoints
11 gpt=read("t2.gpt")
12
13 #Compute the difference
14 res=g-gpt
15
16 #Return results
17 return res
18
```

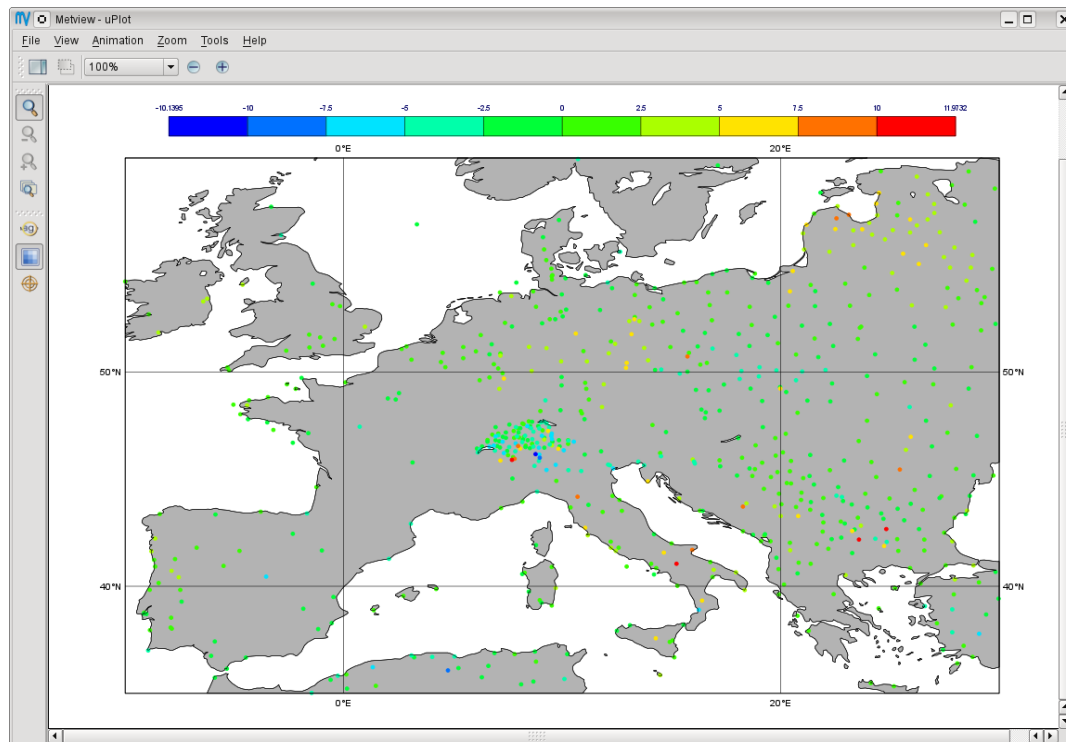
Compute difference between GRIB and Geopoints


grib_minus_geo.mv

execute
visualise
examine
save
analyse
edit



Forecast minus
observation
differences



Geopoints to GRIB

Example: interpolate T2 observations onto a grid then apply contouring

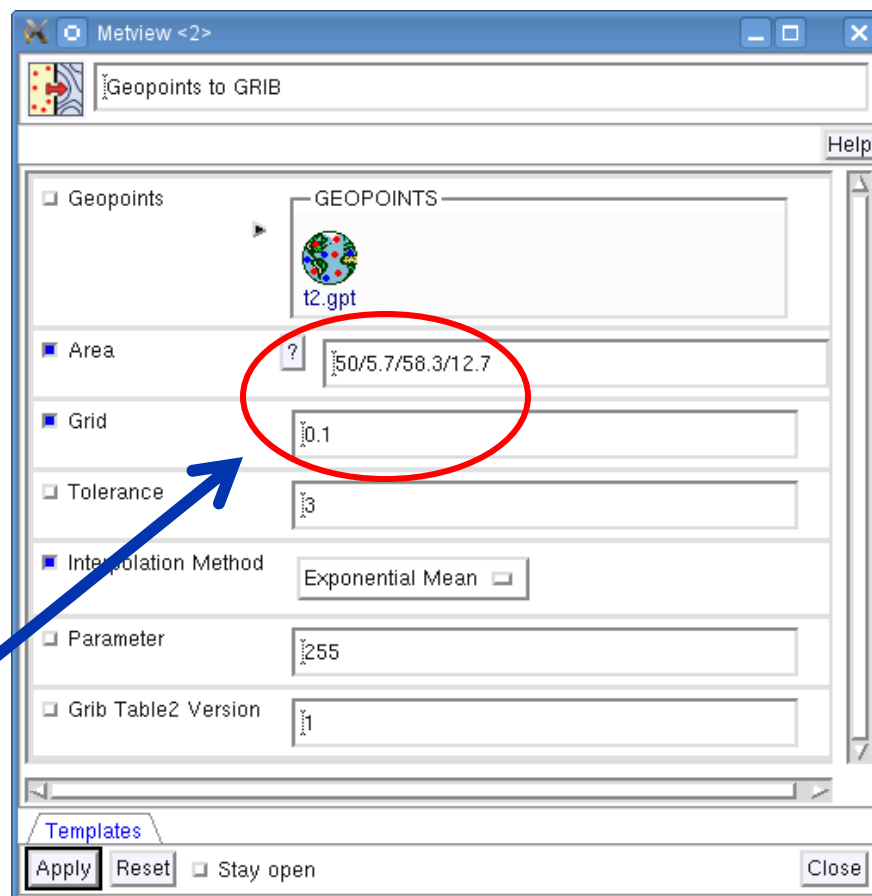
- We need to use the **Geopoints to GRIB** icon



Geopoints to GRIB

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

The grid definition



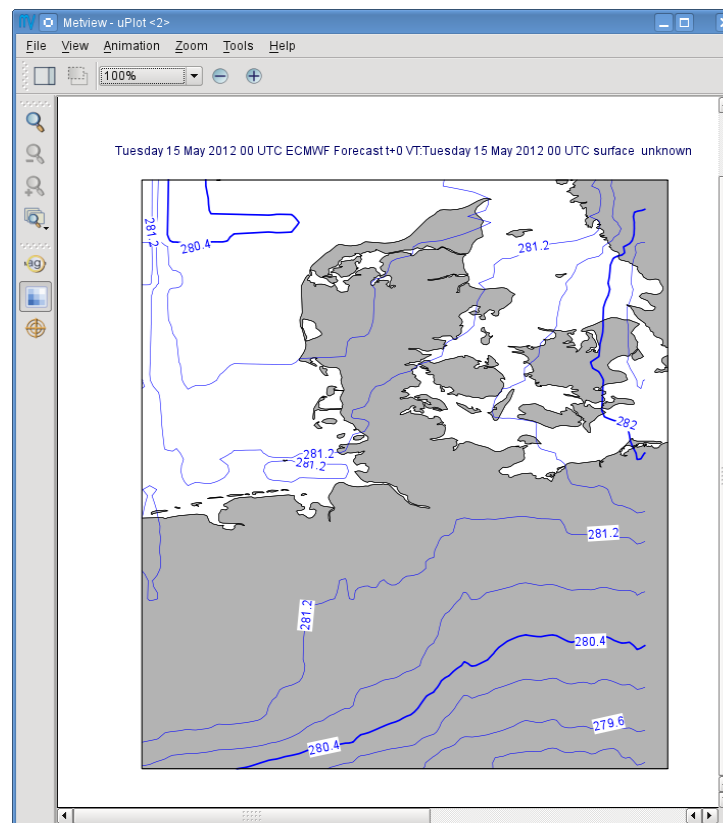
Geopoints to GRIB


Geopoints to GRIB

execute
visualise
examine
save
analyse
edit



**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**

The image shows two windows of the NetCDF Examiner application. The left window displays a tree view of metadata for the file 'fc_surf.nc'. A blue arrow points from a callout box labeled 'Tree view of metadata' to this tree view. A context menu is open over the file, with 'examine' highlighted. The right window shows the 'ncdump' output for the same file, with a blue arrow pointing from a callout box labeled 'ncdump' to the output text.

Tree view of metadata

Parameters	Values
Variables	
longitude	
latitude	
time	
v2t	
Type	short
Dimensions	(time, latitude, longitude)
Attributes	
scale_factor	0.0017198166180648
add_offset	263.466318436418
_FillValue	-32767s
missing_value	-32767s
units	K
long_name	2 metre temperature
Data values	
msl	
tp	
v10u	
v10v	
Dimensions	
longitude	240
latitude	121
time	9

ncdump

```

netcdf fc_surf {
dimensions:
    longitude = 240 ;
    latitude = 121 ;
    time = 9 ;
variables:
    float longitude(longitude) ;
        longitude:units = "degrees_east" ;
        longitude:long_name = "longitude" ;
    float latitude(latitude) ;
        latitude:units = "degrees_north" ;
        latitude:long_name = "latitude" ;
    int time(time) ;
        time:units = "hours since 1900-01-01 00:00:0.0" ;
        time:long_name = "time" ;
    short v2t(time, latitude, longitude) ;
        v2t:scale_factor = 0.0017198166180648 ;
        v2t:add_offset = 263.466318436418 ;
        v2t:_FillValue = -32767s ;
        v2t:missing_value = -32767s ;
        v2t:units = "K" ;
        v2t:long_name = "2 metre temperature" ;
    short msl(time, latitude, longitude) ;
        msl:scale_factor = 0.162427708177559 ;

```

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

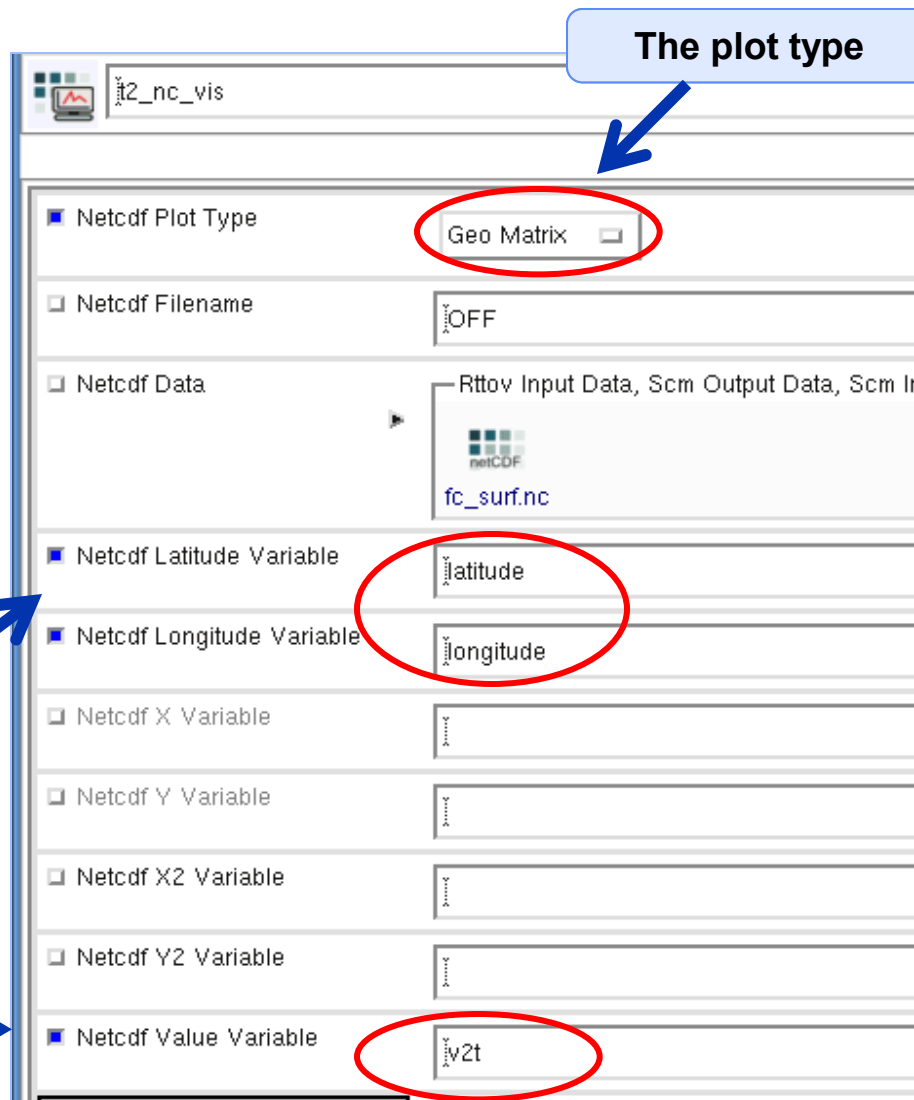
Example:
plot T2 from
file
fc_surf.nc



**Latitude and
longitude variables**

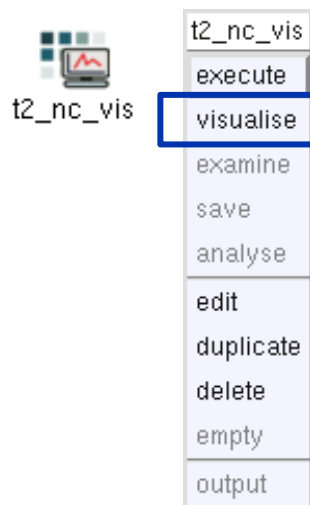
**The variable to
plot (T2 is called
v2t in our file)**

The plot type

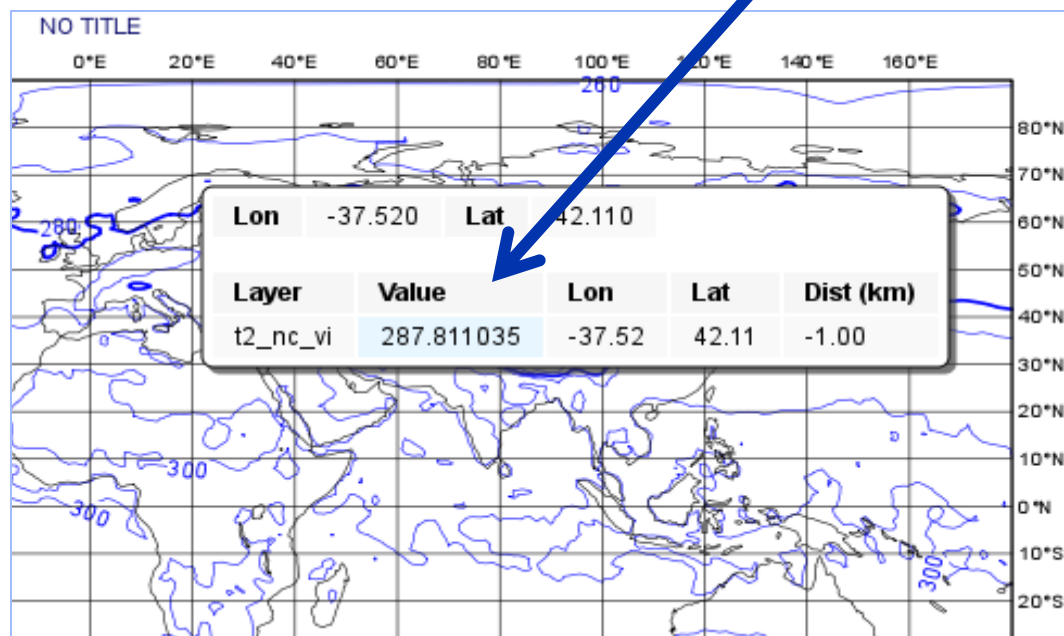


<input checked="" type="checkbox"/> Netcdf Plot Type	Geo Matrix
<input type="checkbox"/> Netcdf Filename	OFF
<input type="checkbox"/> Netcdf Data	Rttov Input Data, Scm Output Data, Scm In netCDF fc_surf.nc
<input checked="" type="checkbox"/> Netcdf Latitude Variable	latitude
<input checked="" type="checkbox"/> Netcdf Longitude Variable	longitude
<input type="checkbox"/> Netcdf X Variable	
<input type="checkbox"/> Netcdf Y Variable	
<input type="checkbox"/> Netcdf X2 Variable	
<input type="checkbox"/> Netcdf Y2 Variable	
<input checked="" type="checkbox"/> Netcdf Value Variable	v2t

NetCDF: Plotting



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



NetCDF: Macro Usage



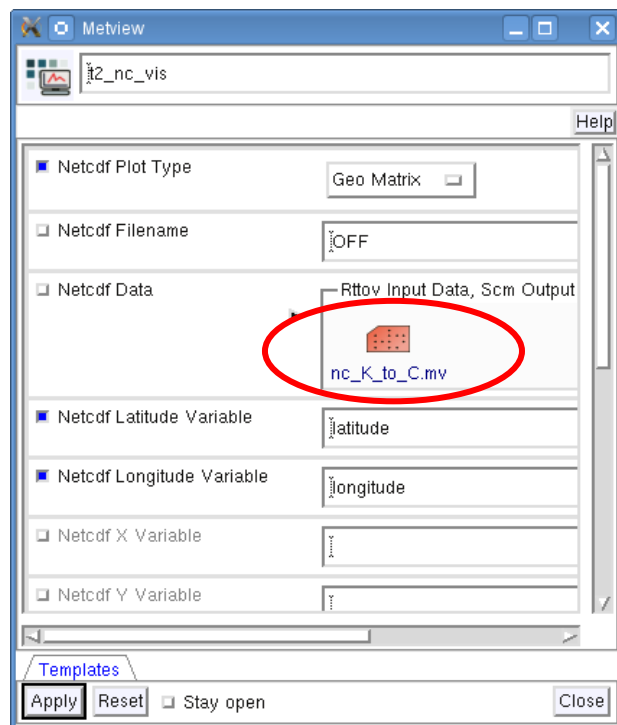
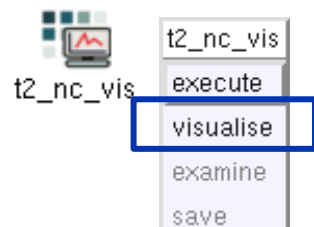
nc_K_to_C.mv

Example: convert values of T2 from Kelvin to Celsius

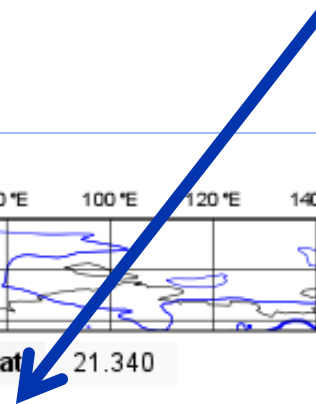
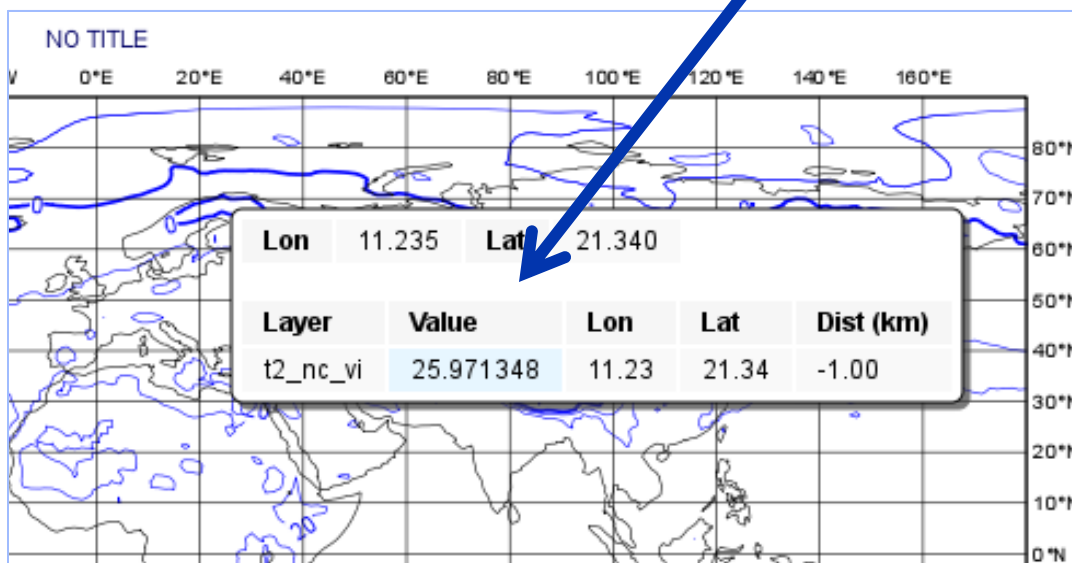
- The NetCDF macro interface is based on the **current variable** concept: all operations are only valid to the currently selected NetCDF variable!

```
1 #Metview Macro
2
3 #Read netcdf file
4 nc=read("fc_surf.nc")
5
6 #Get the list of netcdf variables
7 var_list = variables(nc)
8
9 #Find index for t2
10 idx=find(var_list,"v2t")
11
12 #Set the current variable to t2
13 setcurrent(nc,idx)
14
15 #Change the values of the current variable
16 nc = nc - 273.16
17
18 #Return results
19 return nc
```

NetCDF: Plotting the modified data



Now we have values in Celsius



ASCII Table Data

	A	B	C
1			
2			
3			

- 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

	A	B	C
1			
2			
3			

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



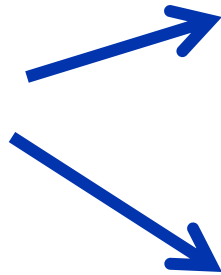
- It defines the way columns are used for plotting

Plotting Table Data

Example: plot the forecast values from file `data.csv`



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



The plot type

Metview

csv_map_vis

Table Plot Type Geo Points

Table Filename OFF

Table Data Notes, GEOPOINTS, Table
data.csv

Table Variable Identifier Type Index

Table Longitude Variable 2

Table Latitude Variable 1

Table X Component Variable

Table Y Component Variable

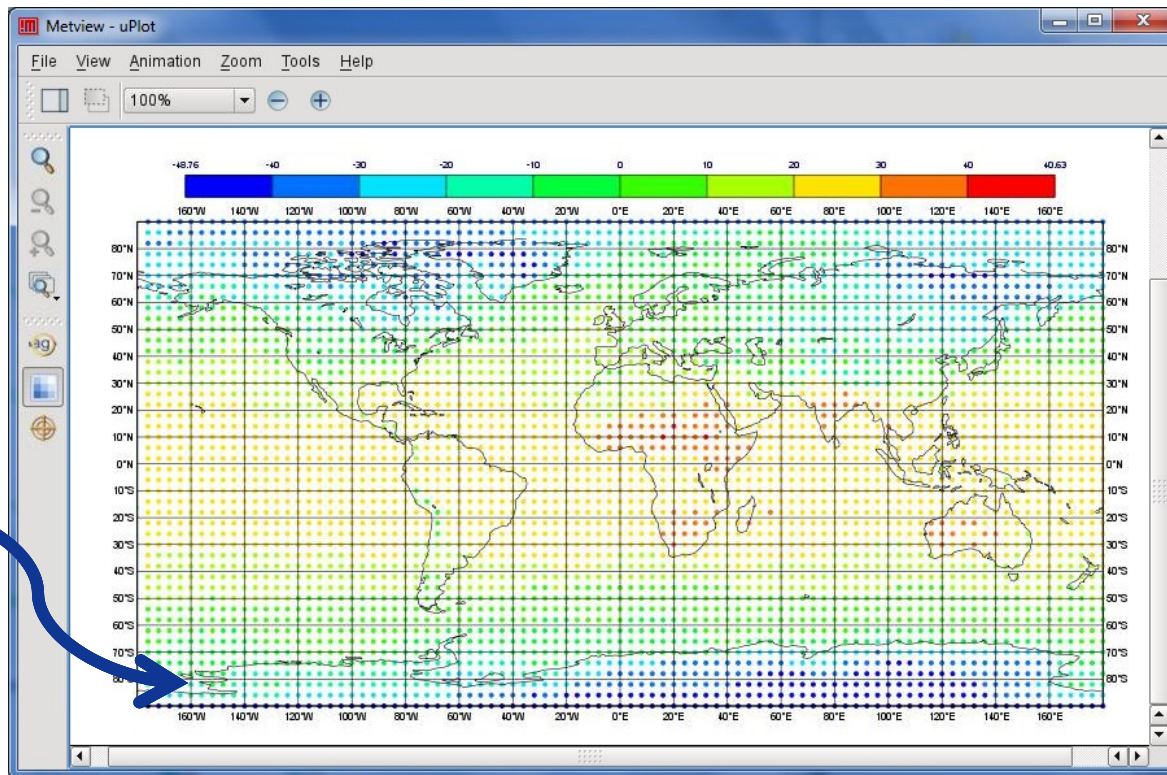
Table Value Variable 3

Plotting Table data


csv_map_vis

- execute
- visualise**
- examine
- save
- analyse


t_syn



Scatterplots from Table data

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

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

The plot type

Table Plot Type: Xy Points

Table Filenane: OFF

Table Data: Notes, GEOPOINTS, Table
data.csv

Table X Type: Number

Table Y Type: Number

Table Variable Identifier Type: Index

Table X Variable: 3

Table Y Variable: 4

Table Value Variable: 5

Table Binning: Binning

Scatterplots from Table data


csv_xy_vis

- execute
- visualise
- examine
- save
- analyse


t_diff

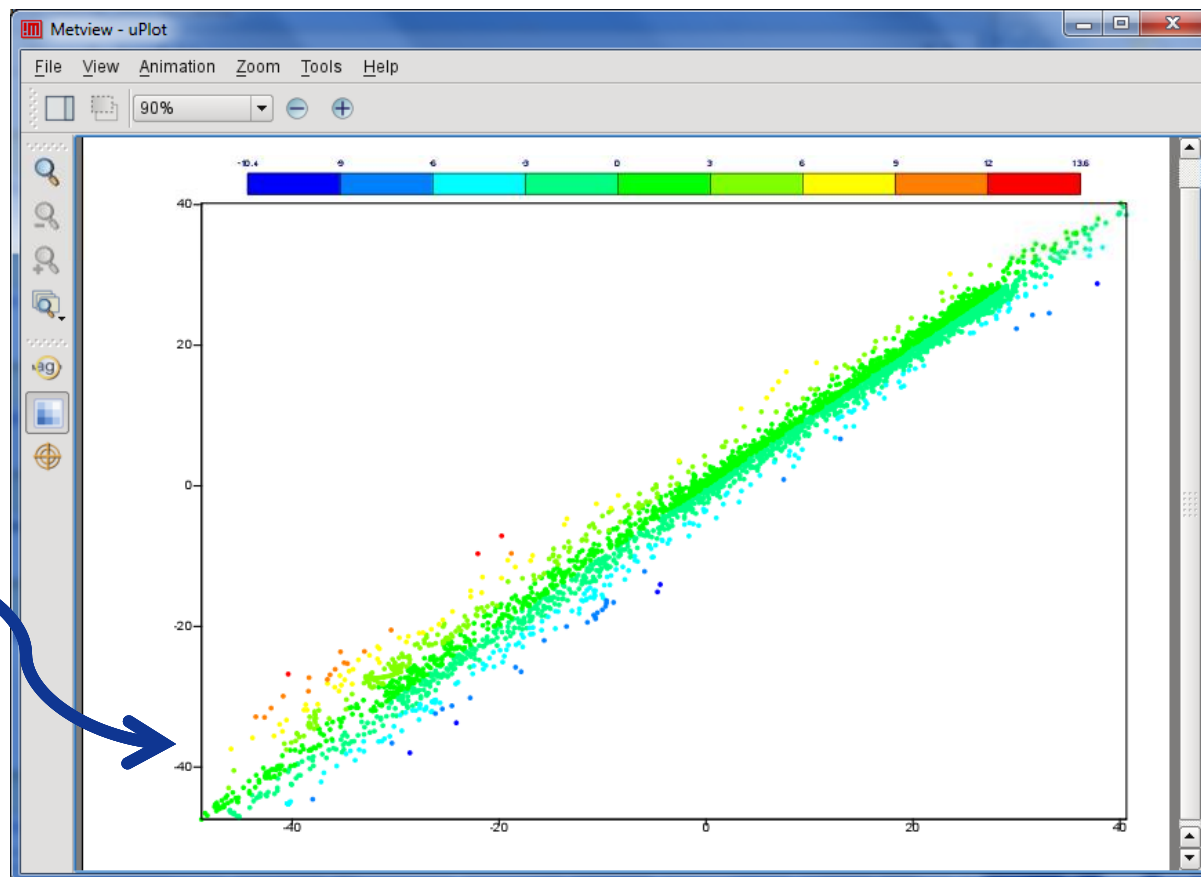


Table Data: macro usage

Example: compute the mean of the forecast-analysis values (5th column) from file `data.csv`

```
1 #Metview Macro
2
3 #Read csv file
4 t=read_table(table_filename: "data.csv")
5
6 #Read the fc-an column into a vector
7 v = values(t, "fc-an")
8 #could be v=values(t,5) as well
9 #since fc-an is the fifth column
10
11 #Print mean
12 print("mean=", mean(v))
```

The output of
the macro



```
mean=-0.0265241545894
```

For more information ...

email us:

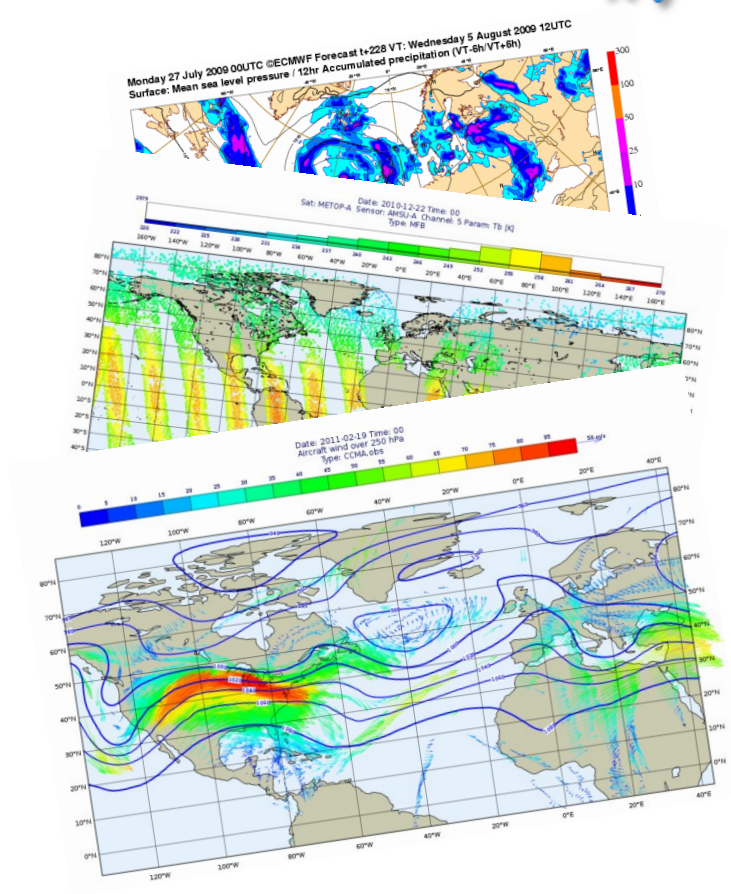
🖱 **Metview:** metview@ecmwf.int

visit our web pages:

🖱 <https://software.ecmwf.int/metview>

➤ **Documentation and tutorials**

➤ **Download the virtual machine**



Thursday, 5th December, 9:30 AM UTC: Q&A

www.hipchat.com/gRuxxenIY