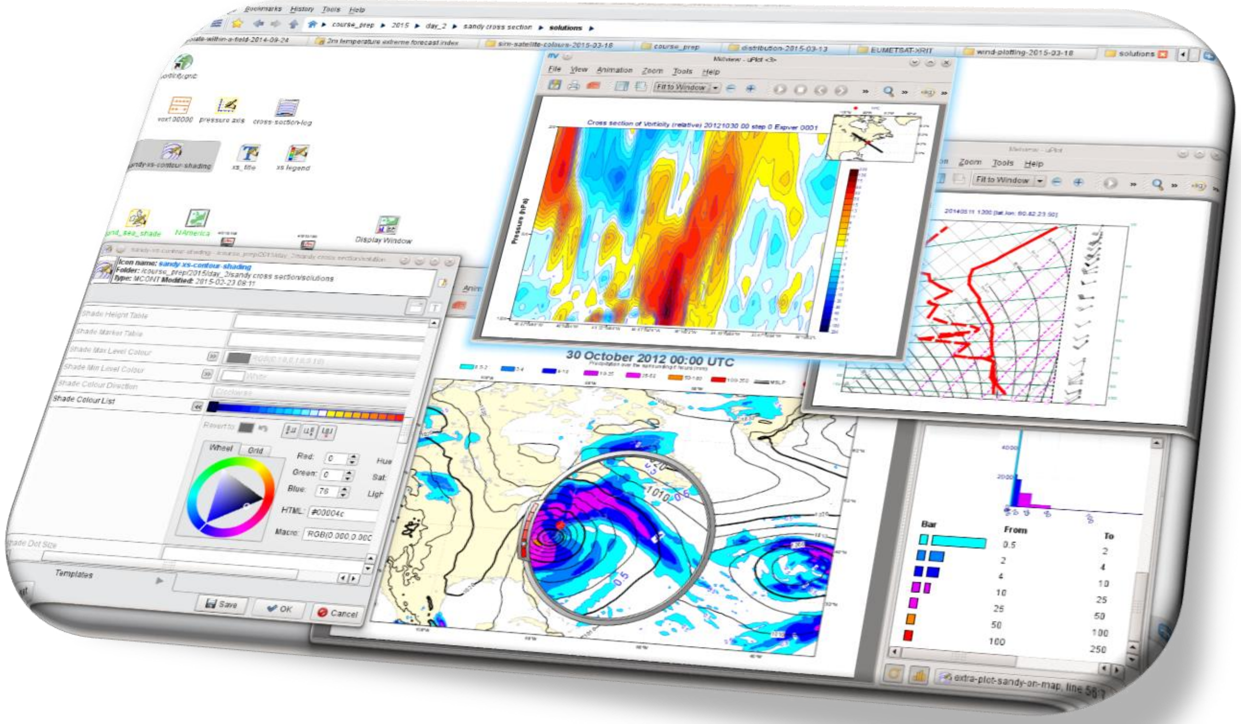
Metview’s new Python interface

Workshop on developing
Python frameworks for
earth system sciences.
ECMWF, 2018

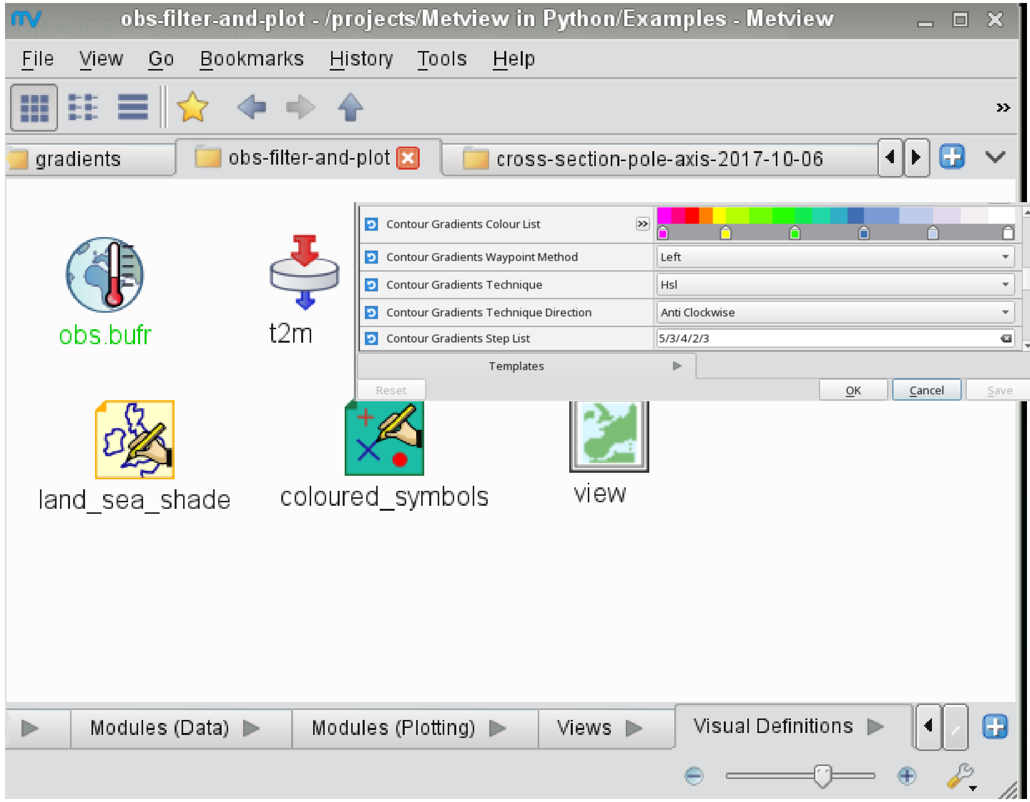
Iain Russell
Development Section, ECMWF

Thanks to
Sándor Kertész
Fernando Ii
Stephan Siemen
Martin Janousek

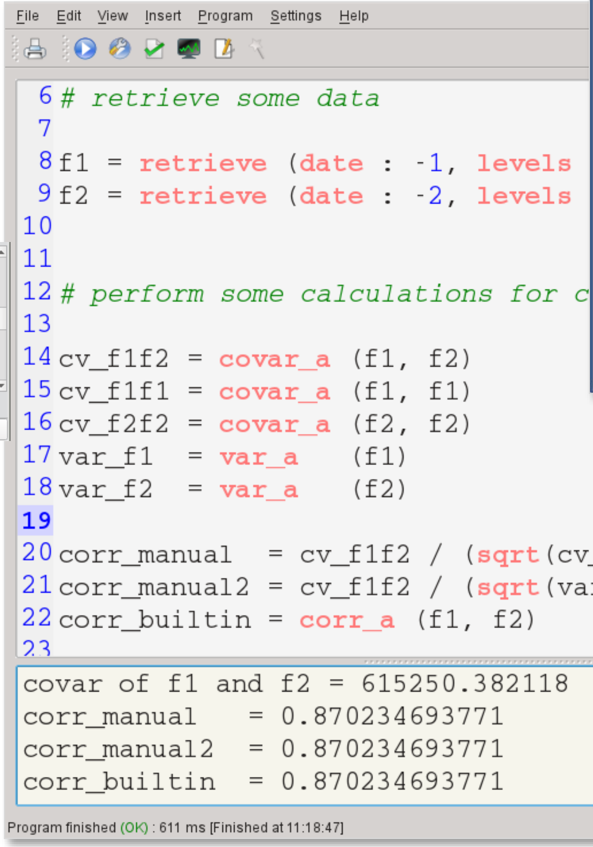


Cross section of Vorticity (relative) 20121030 00 step 0 Expver 0001
30 October 2012 00:00 UTC
Display Window
Templates
Save
OK
Cancel

What is Metview?



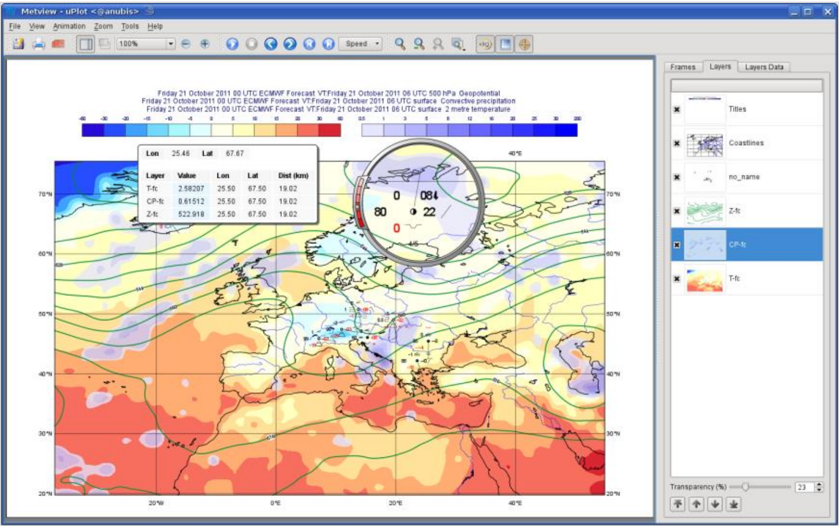
Contour Gradients Colour List	
Contour Gradients Waypoint Method	Left
Contour Gradients Technique	Hsl
Contour Gradients Technique Direction	Anti Clockwise
Contour Gradients Step List	5/3/4/2/3



```
# retrieve some data

f1 = retrieve (date : -1, levels
f2 = retrieve (date : -2, levels


# perform some calculations for c

cv_f1f2 = covar_a (f1, f2)
cv_f1f1 = covar_a (f1, f1)
cv_f2f2 = covar_a (f2, f2)
var_f1  = var_a   (f1)
var_f2  = var_a   (f2)

corr_manual  = cv_f1f2 / (sqrt(cv_f1f
corr_manual2 = cv_f1f2 / (sqrt(var_f1
corr_builtin = corr_a (f1, f2)
```

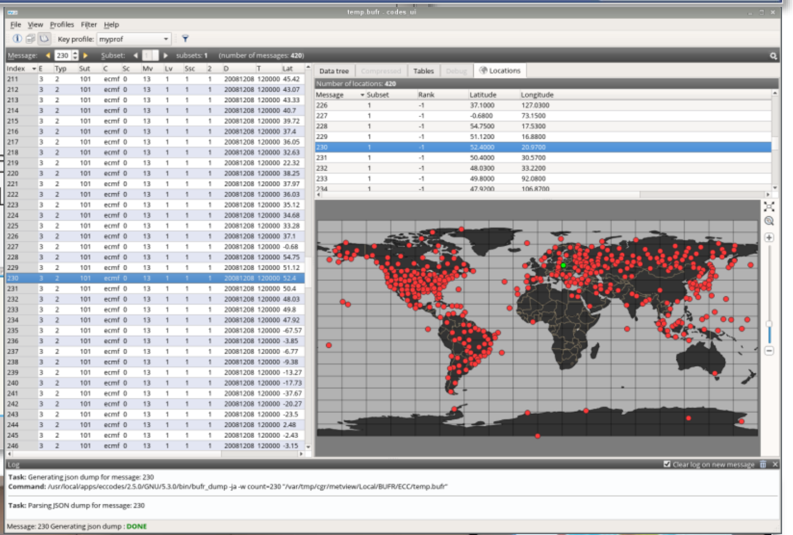
```
covar of f1 and f2 = 615250.382118
corr_manual   = 0.870234693771
corr_manual2  = 0.870234693771
corr_builtin  = 0.870234693771
```

Program finished (OK) : 611 ms [Finished at 11:18:47]



Friday 21 October 2011 00 UTC ECMWF Forecast VT:Friday 21 October 2011 06 UTC 500 hPa Geopotential
Friday 21 October 2011 00 UTC ECMWF Forecast VT:Friday 21 October 2011 06 UTC surface Convective precipitation
Friday 21 October 2011 00 UTC ECMWF Forecast VT:Friday 21 October 2011 06 UTC surface 2 metre temperature

Lon	25.46	Lat	67.67	
Layer	Value	Lon	Lat	Dist (km)
T-fc	2.58207	25.50	67.50	19.02
CP-fc	0.61512	25.50	67.50	19.02
Z-fc	522.918	25.50	67.50	19.02



Message	Subset	Rank	Latitude	Longitude
226	1	-1	37.1000	127.0300
227	1	-1	-0.6800	73.1500
228	1	-1	54.7500	17.5300
229	1	-1	51.1200	16.8800
230	1	-1	52.4000	20.9700
231	1	-1	50.4000	30.5700
232	1	-1	48.0300	33.2200
233	1	-1	49.8000	92.0800

Task: Generating json dump for message: 230
Command: /usr/local/apps/eccodes/2.5.0/GNU/5.3.0/bin/bufr_dump -ja -w count=230 "/var/tmp/cgr/metview/Local/BUFR/ECC/temp.bufr"

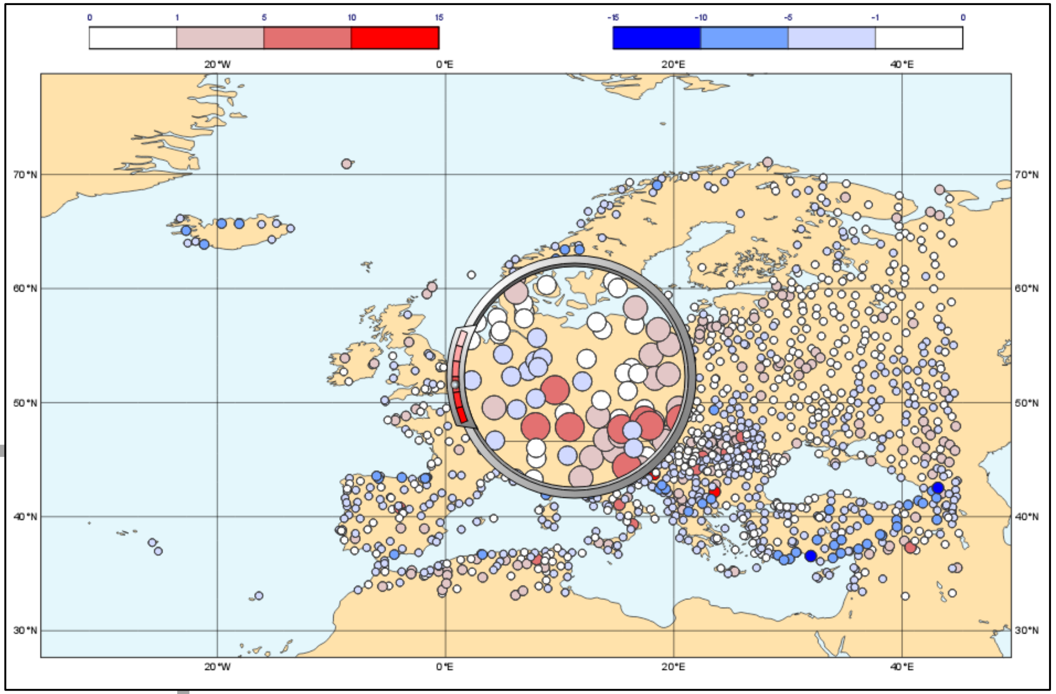
Task: Parsing JSON dump for message: 230

Message: 230 Generating json dump : **DONE**

- Data access, examination, manipulation, visualisation
- UNIX, Open Source under Apache Licence 2.0
- Metview is a co-operation project with INPE (Brazil)

High-level data processing with Metview Macro

- We already had the Macro language…



Forecast – observation difference:

```
forecast = retrieve (…) # GRIB from MARS
obs      = retrieve (…) # BUFR from MARS
t2m_obs  = obsfilter(data:      obs,
                     parameter: 'airTemperatureAt2M',
                     output:    'geopoints')
plot(forecast – t2m_obs)
```

Why create a Python interface to Metview?

- Enable Metview to work seamlessly within the Python eco-system
  - Bring Metview’s data processing and interactive data inspection tools into Python sessions ; interact with Python data structures
  - Use existing solutions where possible (e.g. for multi-dimensional data arrays, data models)
- Enable Metview to be a component of the Copernicus Climate Data Store Toolbox

- Python 3 from the start!

python™

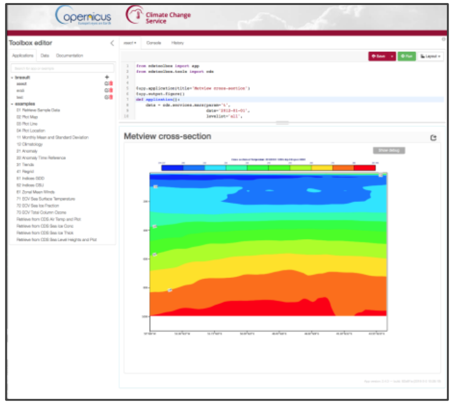
xarray

pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

NumPy

jupyter

Climate Change Service



```
from cdstoolbox import app
from cdstoolbox.tools import cds


@app.application(title='Metview cross-section')
@app.output.figure()
def application():
    data = cds.services.mars(param='t',
                             date='2012-01-01',
                             levelist='all',
```

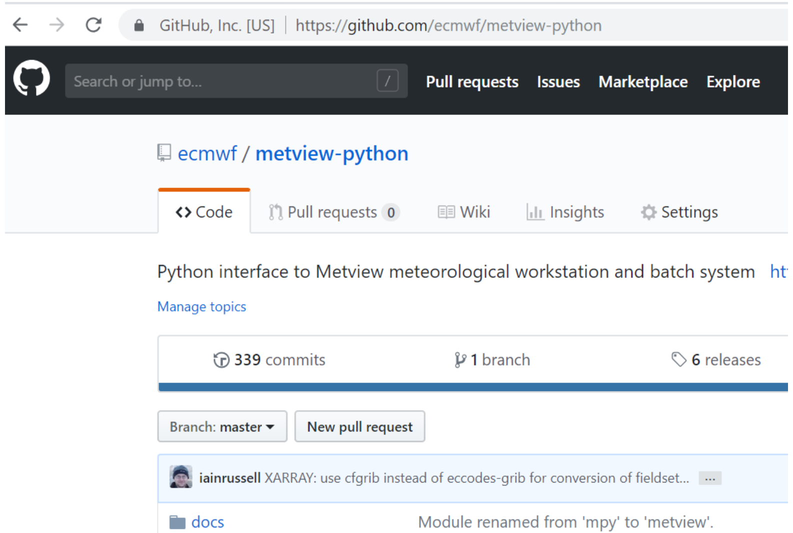
Metview cross-section

Current Status

- Beta release 0.9.0 (developed with B-Open)

- Available on github and PyPi
  - https://github.com/ecmwf/metview-python
  - `pip install metview`
    `python3 -m metview selfcheck`

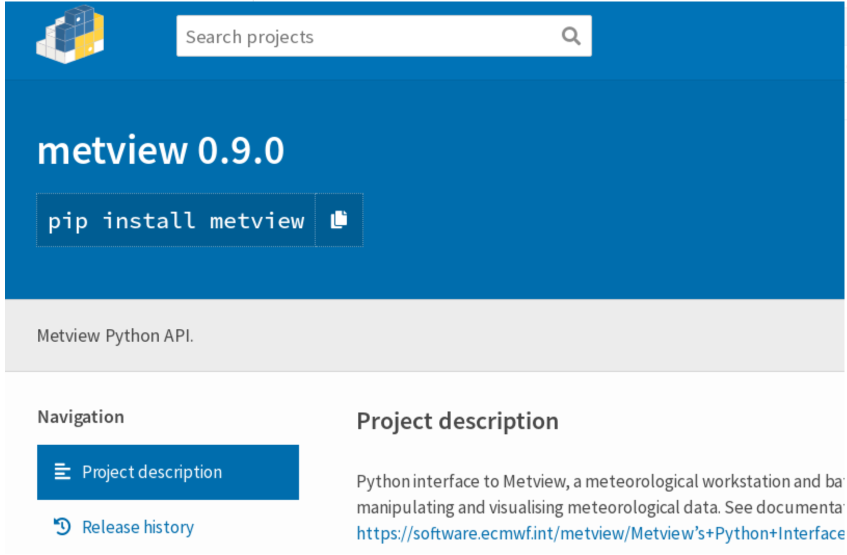
- Python layer only – this still requires the Metview binaries to be installed too



Python interface to Metview meteorological workstation and batch system ht

iainrussell XARRAY: use cfgrib instead of eccodes-grib for conversion of fieldset...

docs   Module renamed from 'mpy' to 'metview'.



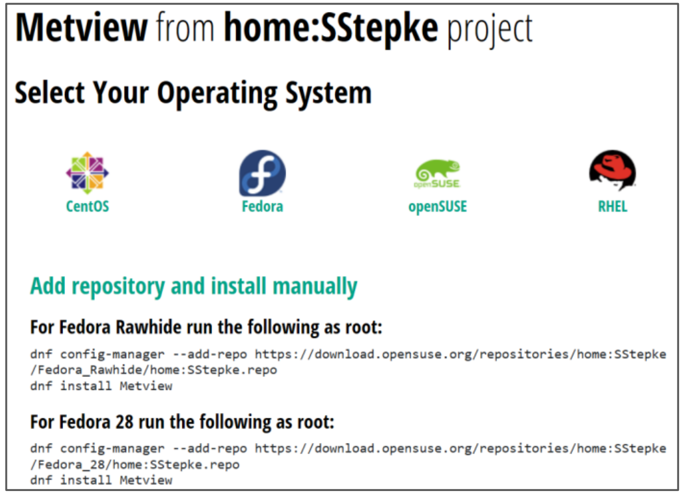
metview 0.9.0

```
pip install metview
```

Metview Python API.

Project description

Python interface to Metview, a meteorological workstation and ba
manipulating and visualising meteorological data. See documenta
https://software.ecmwf.int/metview/Metview's+Python+Interface


Metview from **home:SStepke** project

Select Your Operating System

Add repository and install manually

For Fedora Rawhide run the following as root:

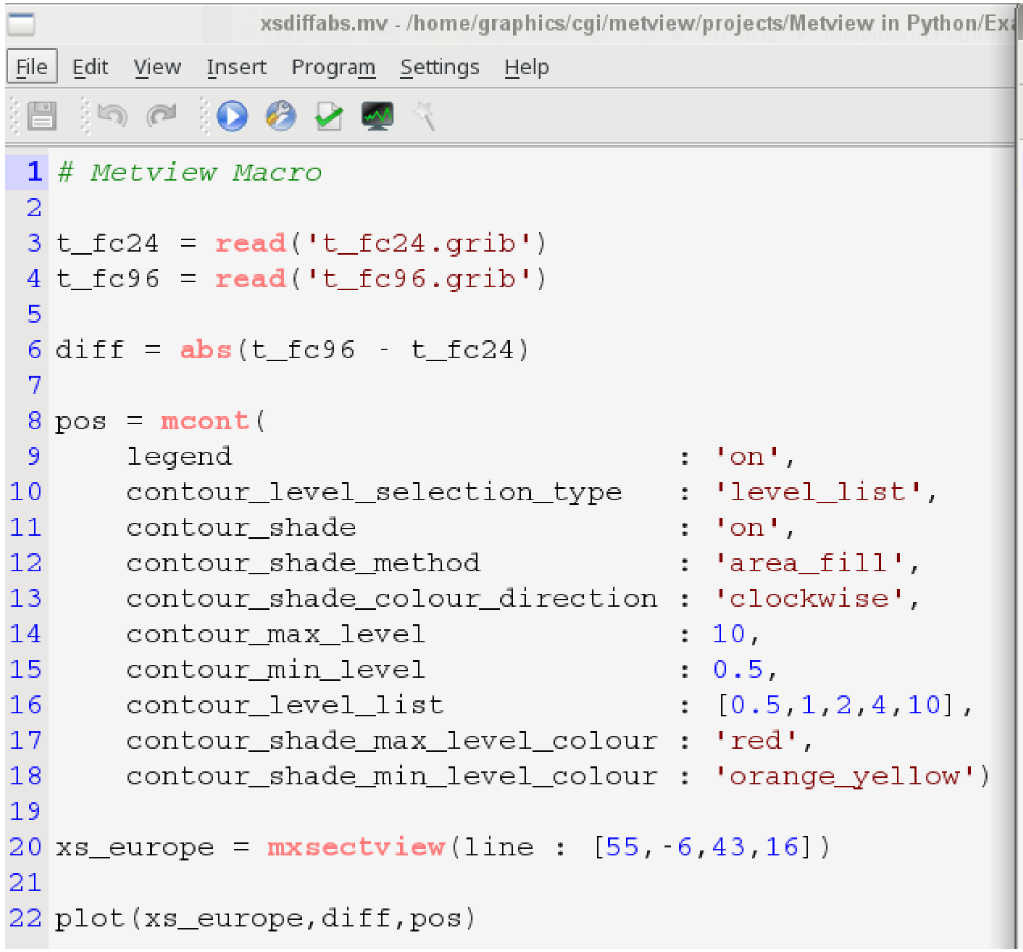
```
dnf config-manager --add-repo https://download.opensuse.org/repositories/home:SStepke
/Fedora_Rawhide/home:SStepke.repo
dnf install Metview
```

For Fedora 28 run the following as root:

```
dnf config-manager --add-repo https://download.opensuse.org/repositories/home:SStepke
/Fedora_28/home:SStepke.repo
dnf install Metview
```

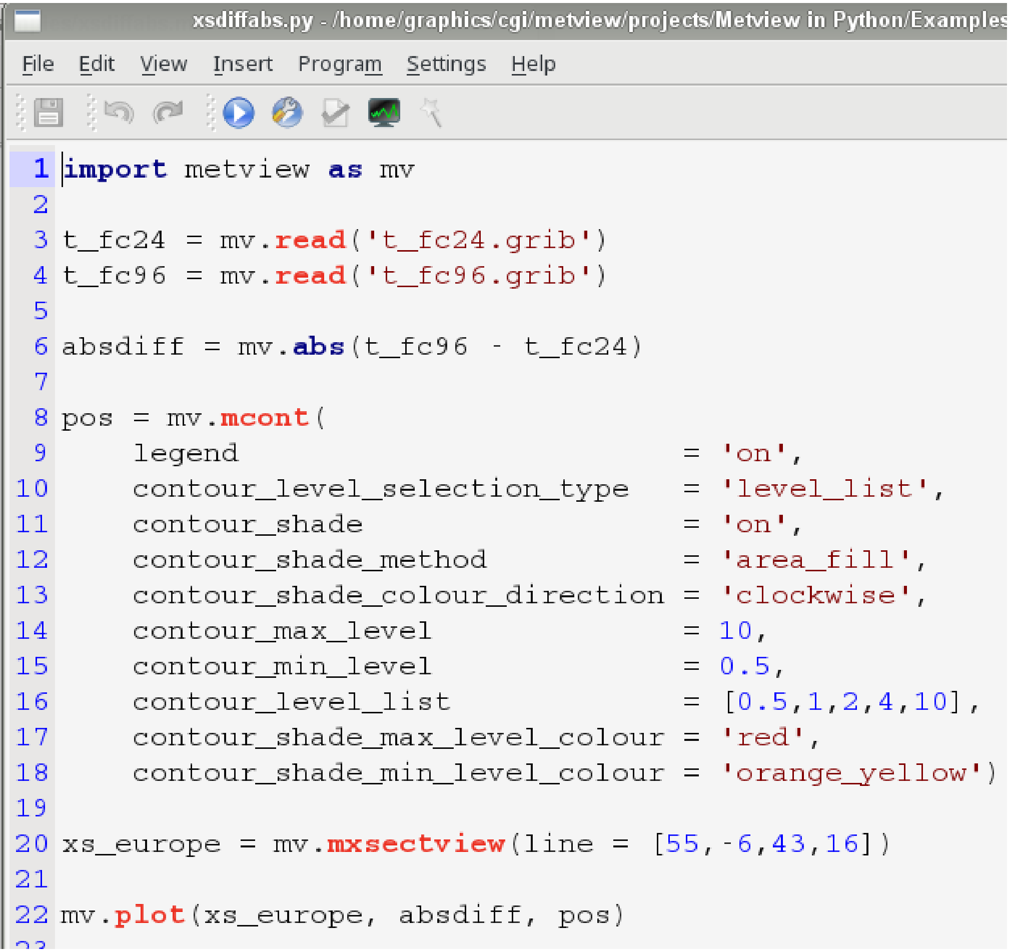
Macro / Python comparison

Macro


xsdiffabs.mv - /home/graphics/cgi/metview/projects/Metview in Python/Ex
File
Edit
View
Insert
Program
Settings
Help

```
# Metview Macro

t_fc24 = read('t_fc24.grib')
t_fc96 = read('t_fc96.grib')

diff = abs(t_fc96 - t_fc24)

pos = mcont(
    legend                            : 'on',
    contour_level_selection_type      : 'level_list',
    contour_shade                     : 'on',
    contour_shade_method              : 'area_fill',
    contour_shade_colour_direction    : 'clockwise',
    contour_max_level                 : 10,
    contour_min_level                 : 0.5,
    contour_level_list                : [0.5,1,2,4,10],
    contour_shade_max_level_colour    : 'red',
    contour_shade_min_level_colour    : 'orange_yellow')

xs_europe = mxsectview(line : [55,-6,43,16])

plot(xs_europe,diff,pos)
```

Python

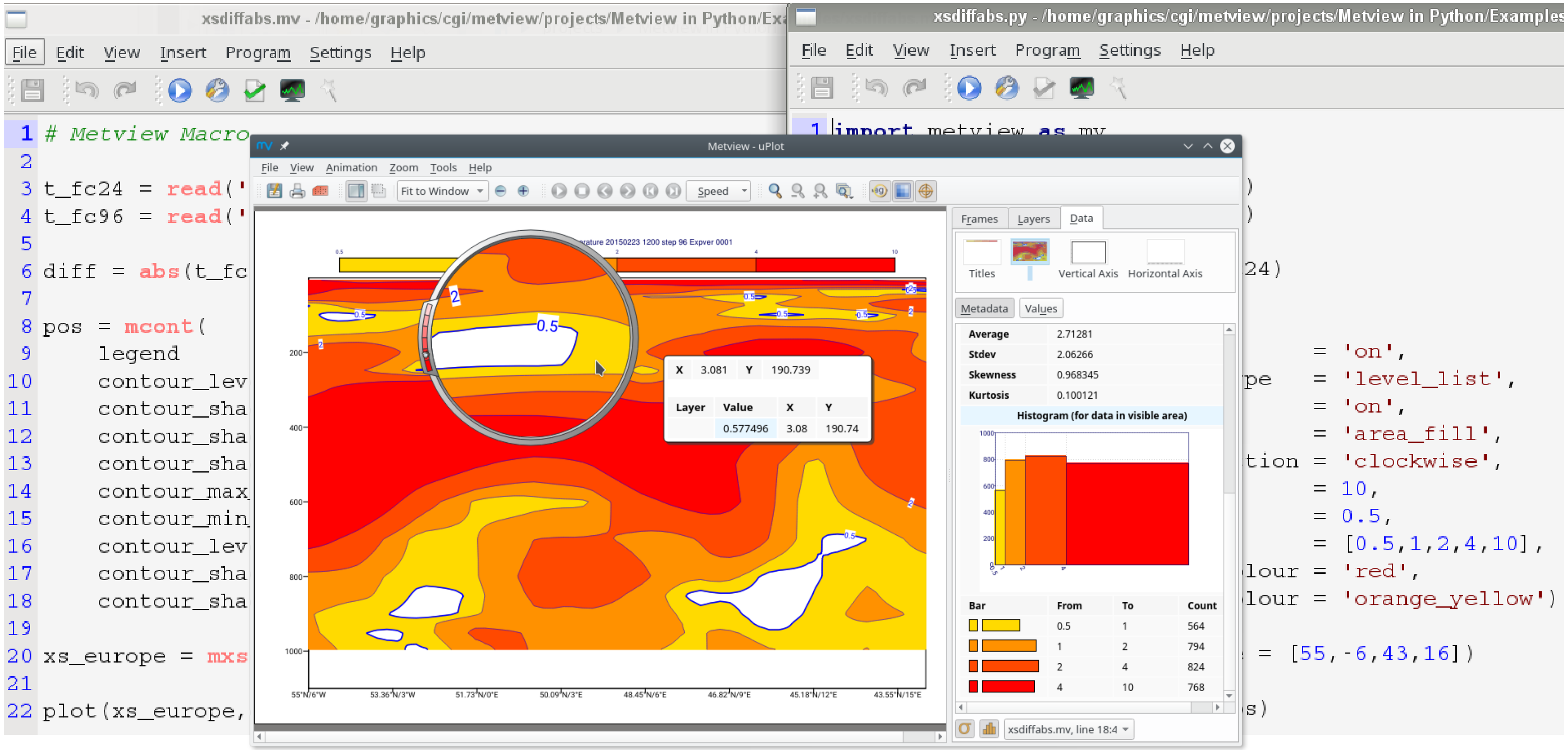

xsdiffabs.py - /home/graphics/cgi/metview/projects/Metview in Python/Examples
File
Edit
View
Insert
Program
Settings
Help

```
import metview as mv

t_fc24 = mv.read('t_fc24.grib')
t_fc96 = mv.read('t_fc96.grib')

absdiff = mv.abs(t_fc96 - t_fc24)

pos = mv.mcont(
    legend                         = 'on',
    contour_level_selection_type   = 'level_list',
    contour_shade                  = 'on',
    contour_shade_method           = 'area_fill',
    contour_shade_colour_direction = 'clockwise',
    contour_max_level              = 10,
    contour_min_level              = 0.5,
    contour_level_list             = [0.5,1,2,4,10],
    contour_shade_max_level_colour = 'red',
    contour_shade_min_level_colour = 'orange_yellow')

xs_europe = mv.mxsectview(line = [55,-6,43,16])

mv.plot(xs_europe, absdiff, pos)
```

Macro / Python comparison

Macro

Python



xsdiffabs.mv - /home/graphics/cgi/metview/projects/Metview in Python/Ex

```
# Metview Macro

t_fc24 = read('
t_fc96 = read('

diff = abs(t_fc

pos = mcont(
     legend
     contour_lev
     contour_sha
     contour_sha
     contour_sha
     contour_max
     contour_min
     contour_lev
     contour_sha
     contour_sha

xs_europe = mxs

plot(xs_europe,
```

xsdiffabs.py - /home/graphics/cgi/metview/projects/Metview in Python/Examples

```
import metview as mv
                                                )
                                                )

                                                24)

                                                = 'on',
                                             pe = 'level_list',
                                                = 'on',
                                                = 'area_fill',
                                           tion = 'clockwise',
                                                = 10,
                                                = 0.5,
                                                = [0.5,1,2,4,10],
                                           lour = 'red',
                                           lour = 'orange_yellow')

                                             = [55,-6,43,16])

                                           s)
```

Metview - uPlot

X	3.081	Y	190.739

Layer	Value	X	Y
	0.577496	3.08	190.74

Average	2.71281
Stdev	2.06266
Skewness	0.968345
Kurtosis	0.100121

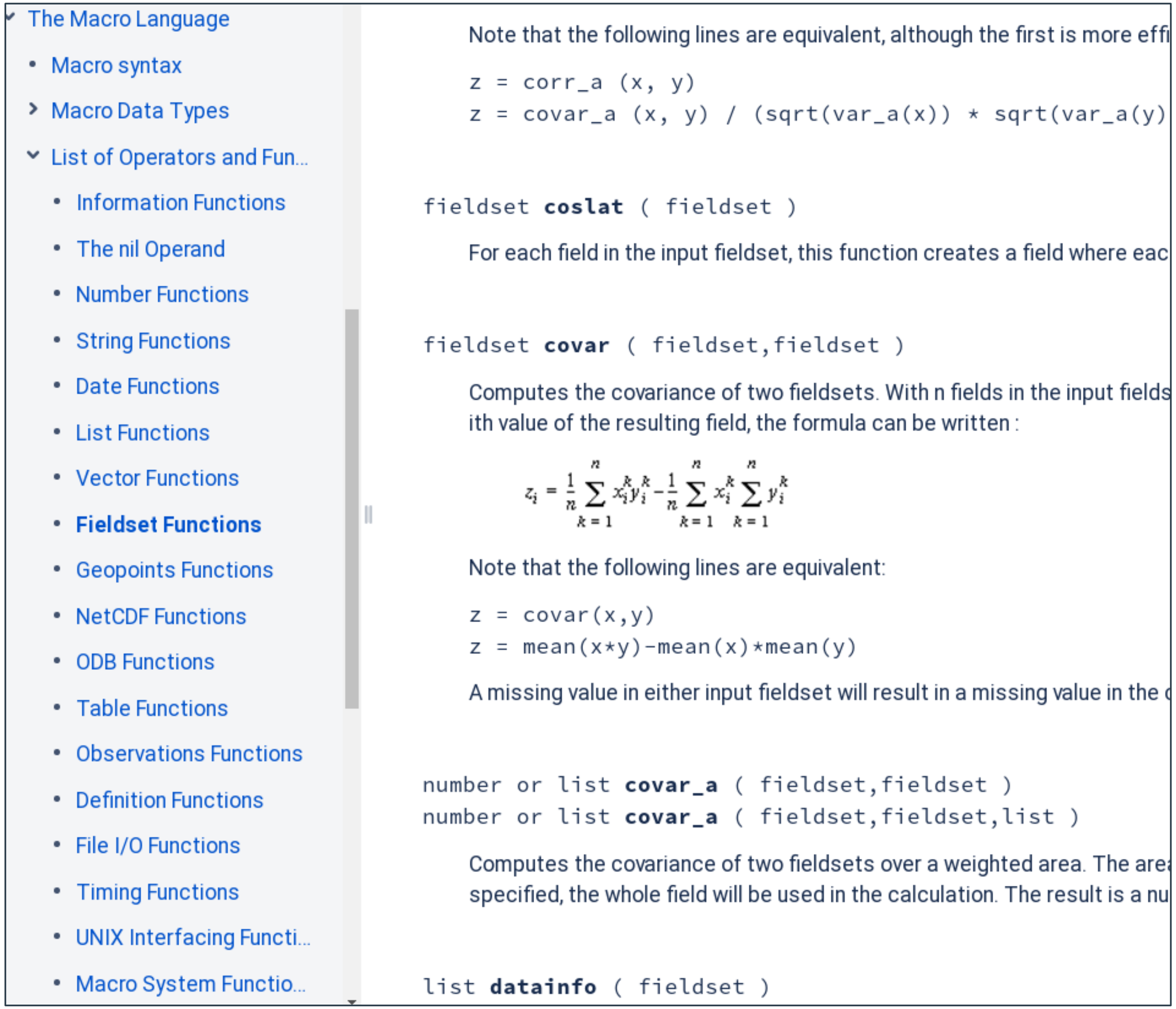
Histogram (for data in visible area)

Bar	From	To	Count
🟨	0.5	1	564
🟧	1	2	794
🟧	2	4	824
🟥	4	10	768

xsdiffabs.mv, line 18:4

Data types (1)

- All Metview Macro functions can be called from Python, e.g. mv.covar(f1, f2)
- Data types returned are either standard Python types (numbers, lists, strings, datetimes), numpy arrays …
- … or thin class wrappers around more complex objects such as fieldsets, geopoints or ODB



Note that the following lines are equivalent, although the first is more effi

```
z = corr_a (x, y)
z = covar_a (x, y) / (sqrt(var_a(x)) * sqrt(var_a(y)
```

```
fieldset coslat ( fieldset )
```

For each field in the input fieldset, this function creates a field where eac

```
fieldset covar ( fieldset,fieldset )
```

Computes the covariance of two fieldsets. With n fields in the input fields ith value of the resulting field, the formula can be written :

$$z_i = \frac{1}{n}\sum_{k=1}^{n} x_i^k y_i^k - \frac{1}{n}\sum_{k=1}^{n} x_i^k \sum_{k=1}^{n} y_i^k$$

Note that the following lines are equivalent:

```
z = covar(x,y)
z = mean(x*y)-mean(x)*mean(y)
```

A missing value in either input fieldset will result in a missing value in the

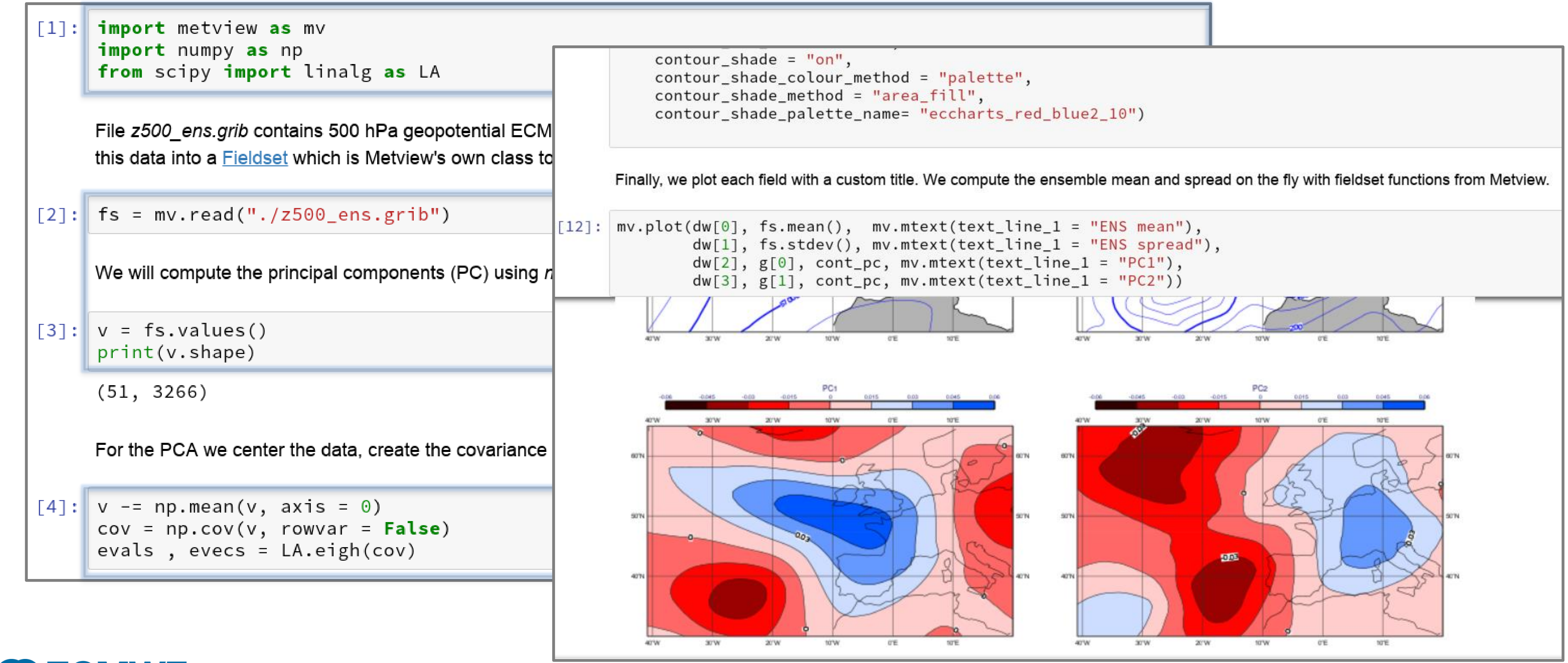
```
number or list covar_a ( fieldset,fieldset )
number or list covar_a ( fieldset,fieldset,list )
```

Computes the covariance of two fieldsets over a weighted area. The are specified, the whole field will be used in the calculation. The result is a nu

```
list datainfo ( fieldset )
```

Data types (2)

- Can extract numpy arrays from most Metview data types
- Example: compute and plot principal components of ensemble forecasts stored in GRIB



```
[1]: import metview as mv
     import numpy as np
     from scipy import linalg as LA
```

File *z500_ens.grib* contains 500 hPa geopotential ECM
this data into a Fieldset which is Metview's own class to

```
[2]: fs = mv.read("./z500_ens.grib")
```

We will compute the principal components (PC) using n

```
[3]: v = fs.values()
     print(v.shape)

     (51, 3266)
```

For the PCA we center the data, create the covariance

```
[4]: v -= np.mean(v, axis = 0)
     cov = np.cov(v, rowvar = False)
     evals , evecs = LA.eigh(cov)
```

```
         contour_shade = "on",
         contour_shade_colour_method = "palette",
         contour_shade_method = "area_fill",
         contour_shade_palette_name= "eccharts_red_blue2_10")
```

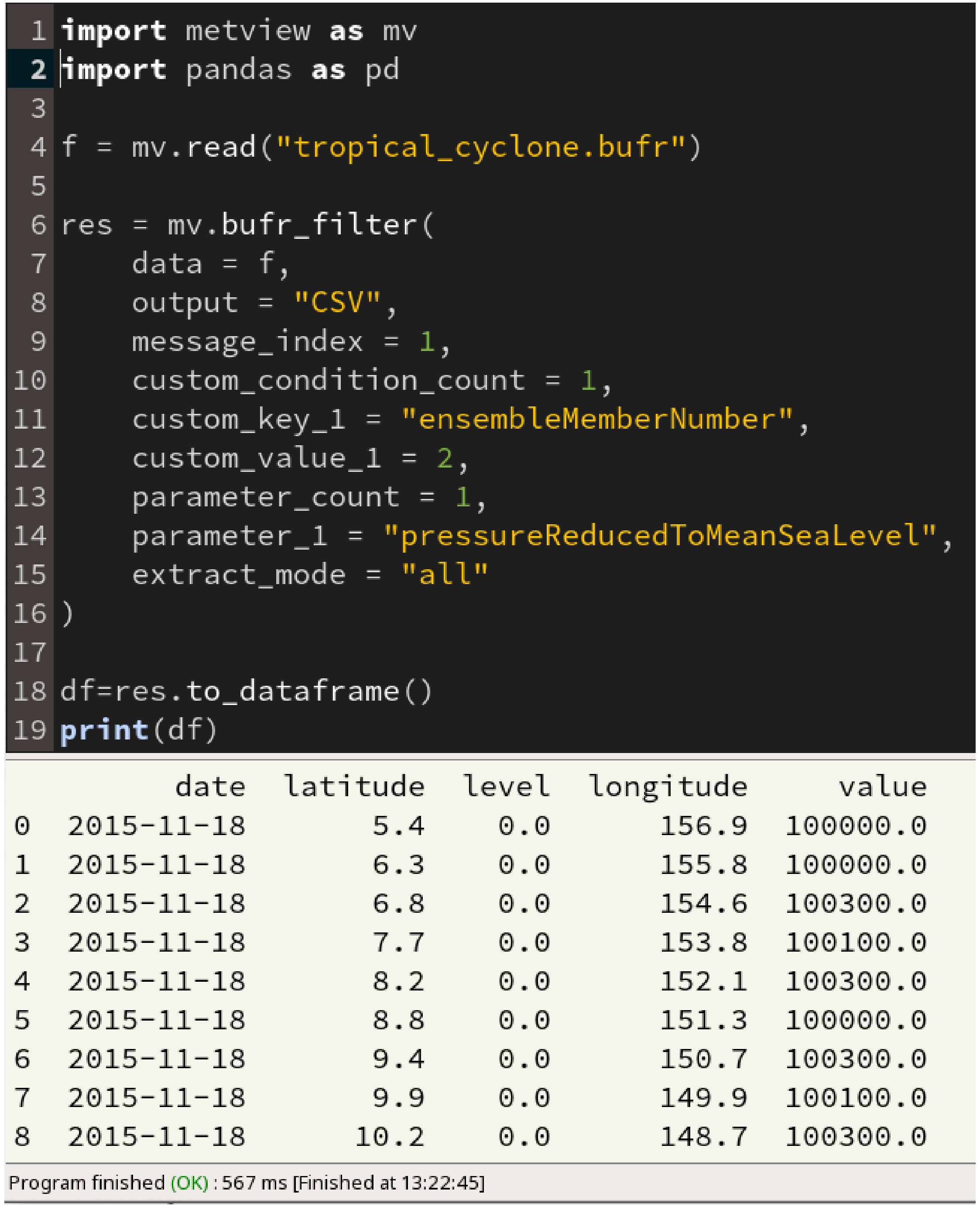
Finally, we plot each field with a custom title. We compute the ensemble mean and spread on the fly with fieldset functions from Metview.

```
[12]: mv.plot(dw[0], fs.mean(),  mv.mtext(text_line_1 = "ENS mean"),
              dw[1], fs.stdev(), mv.mtext(text_line_1 = "ENS spread"),
              dw[2], g[0], cont_pc, mv.mtext(text_line_1 = "PC1"),
              dw[3], g[1], cont_pc, mv.mtext(text_line_1 = "PC2"))
```

Data types (3)

- Can also export Metview Geopoints (and BUFR via the filter), ODB and Table data types to **pandas** Dataframes (table-like format, common in scientific data processing)

- Example:
  - run a filter on a BUFR file containing tropical cyclone tracks
  - Convert to pandas dataframe



```
import metview as mv
import pandas as pd

f = mv.read("tropical_cyclone.bufr")

res = mv.bufr_filter(
    data = f,
    output = "CSV",
    message_index = 1,
    custom_condition_count = 1,
    custom_key_1 = "ensembleMemberNumber",
    custom_value_1 = 2,
    parameter_count = 1,
    parameter_1 = "pressureReducedToMeanSeaLevel",
    extract_mode = "all"
)

df=res.to_dataframe()
print(df)
```

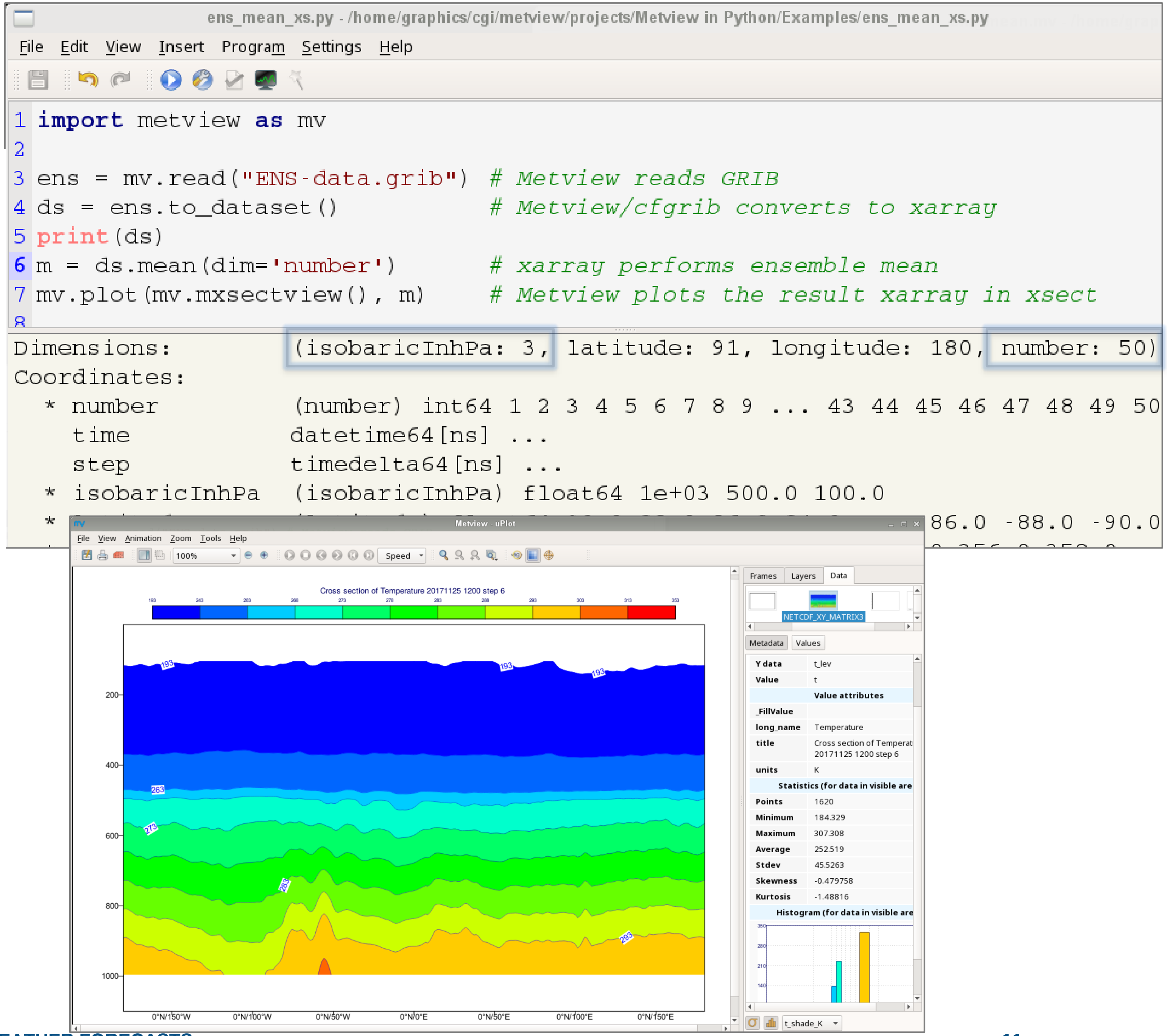
```
         date  latitude  level  longitude     value
0  2015-11-18       5.4    0.0      156.9  100000.0
1  2015-11-18       6.3    0.0      155.8  100000.0
2  2015-11-18       6.8    0.0      154.6  100300.0
3  2015-11-18       7.7    0.0      153.8  100100.0
4  2015-11-18       8.2    0.0      152.1  100300.0
5  2015-11-18       8.8    0.0      151.3  100000.0
6  2015-11-18       9.4    0.0      150.7  100300.0
7  2015-11-18       9.9    0.0      149.9  100100.0
8  2015-11-18      10.2    0.0      148.7  100300.0
```

```
Program finished (OK) : 567 ms [Finished at 13:22:45]
```

Data types (4)

xarray

- Can also export Metview Fieldsets to **xarray** Datasets
  - Provides data in the Common Data Model used by netCDF and the CDS
- Uses the **cfgrib** package developed by B-Open, available on github and PyPi
- Can also pass *some* xarray datasets into Metview functions



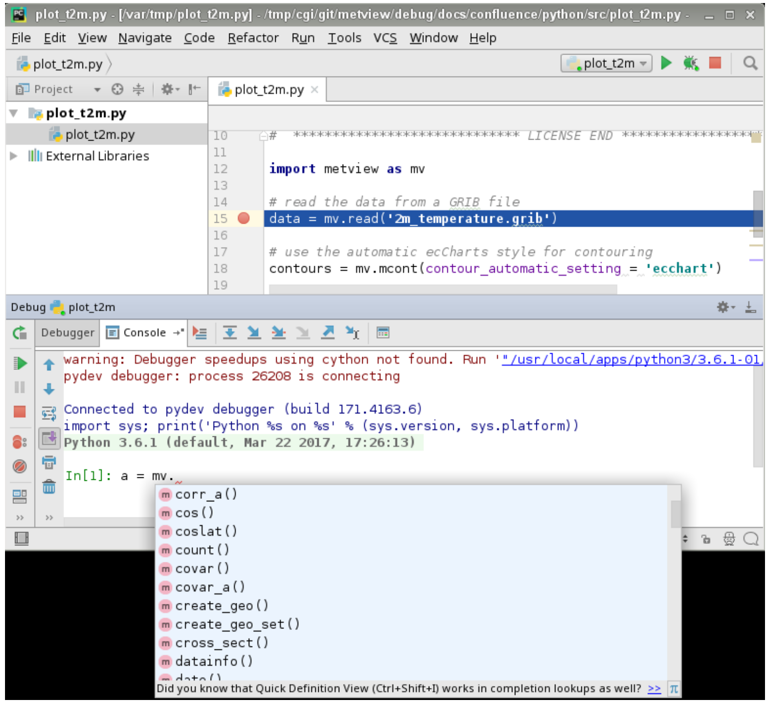
```
import metview as mv

ens = mv.read("ENS-data.grib")  # Metview reads GRIB
ds = ens.to_dataset()           # Metview/cfgrib converts to xarray
print(ds)
m = ds.mean(dim='number')       # xarray performs ensemble mean
mv.plot(mv.mxsectview(), m)     # Metview plots the result xarray in xsect
```

```
Dimensions:          (isobaricInhPa: 3, latitude: 91, longitude: 180, number: 50)
Coordinates:
  * number           (number) int64 1 2 3 4 5 6 7 8 9 ... 43 44 45 46 47 48 49 50
    time             datetime64[ns] ...
    step             timedelta64[ns] ...
  * isobaricInhPa    (isobaricInhPa) float64 1e+03 500.0 100.0
  *                                                          86.0 -88.0 -90.0
```

Y data	t_lev
Value	t
	Value attributes
_FillValue	
long_name	Temperature
title	Cross section of Temperat 20171125 1200 step 6
units	K
	Statistics (for data in visible are
Points	1620
Minimum	184.329
Maximum	307.308
Average	252.519
Stdev	45.5263
Skewness	-0.479758
Kurtosis	-1.48816
	Histogram (for data in visible are

Running Metview from an IDE

- From command line or IDE, e.g. Jupyter, PyCharm – can provide code completion and debugging facilities
- We can improve the amount of information we supply to IDEs

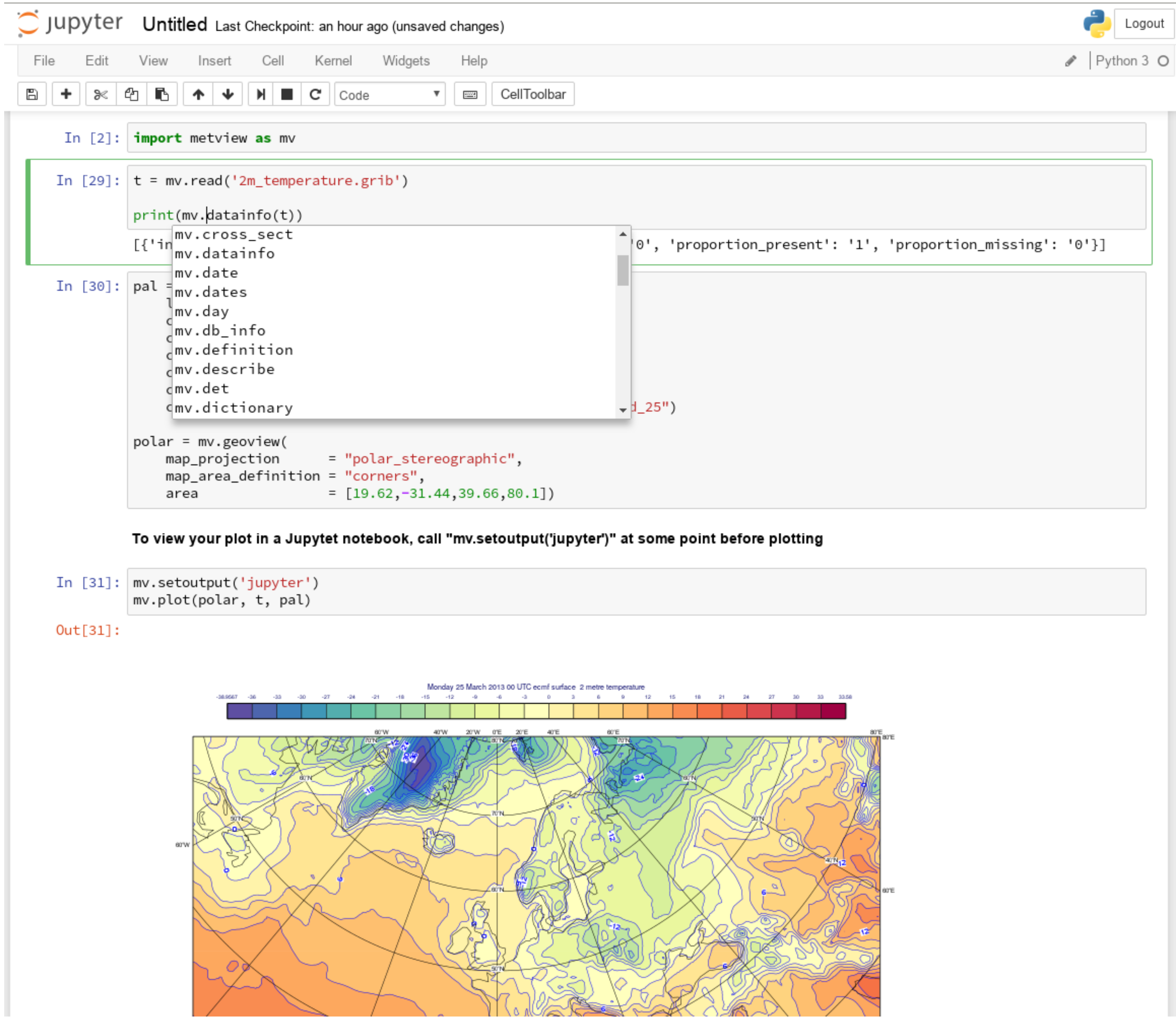

plot_t2m.py - [/var/tmp/plot_t2m.py] - /tmp/cgi/git/metview/debug/docs/confluence/python/src/plot_t2m.py -
File Edit View Navigate Code Refactor Run Tools VCS Window Help
plot_t2m.py
plot_t2m
Project
plot_t2m.py
plot_t2m.py
External Libraries
plot_t2m.py
Debug plot_t2m
Debugger
Console

```
#   ****************************** LICENSE END ******************

import metview as mv

# read the data from a GRIB file
data = mv.read('2m_temperature.grib')

# use the automatic ecCharts style for contouring
contours = mv.mcont(contour_automatic_setting = 'ecchart')
```

```
warning: Debugger speedups using cython not found. Run '"/usr/local/apps/python3/3.6.1-01,
pydev debugger: process 26208 is connecting

Connected to pydev debugger (build 171.4163.6)
import sys; print('Python %s on %s' % (sys.version, sys.platform))
Python 3.6.1 (default, Mar 22 2017, 17:26:13)

In[1]: a = mv.
```

```
m corr_a()
m cos()
m coslat()
m count()
m covar()
m covar_a()
m create_geo()
m create_geo_set()
m cross_sect()
m datainfo()
m date()
```

Did you know that Quick Definition View (Ctrl+Shift+I) works in completion lookups as well? >>

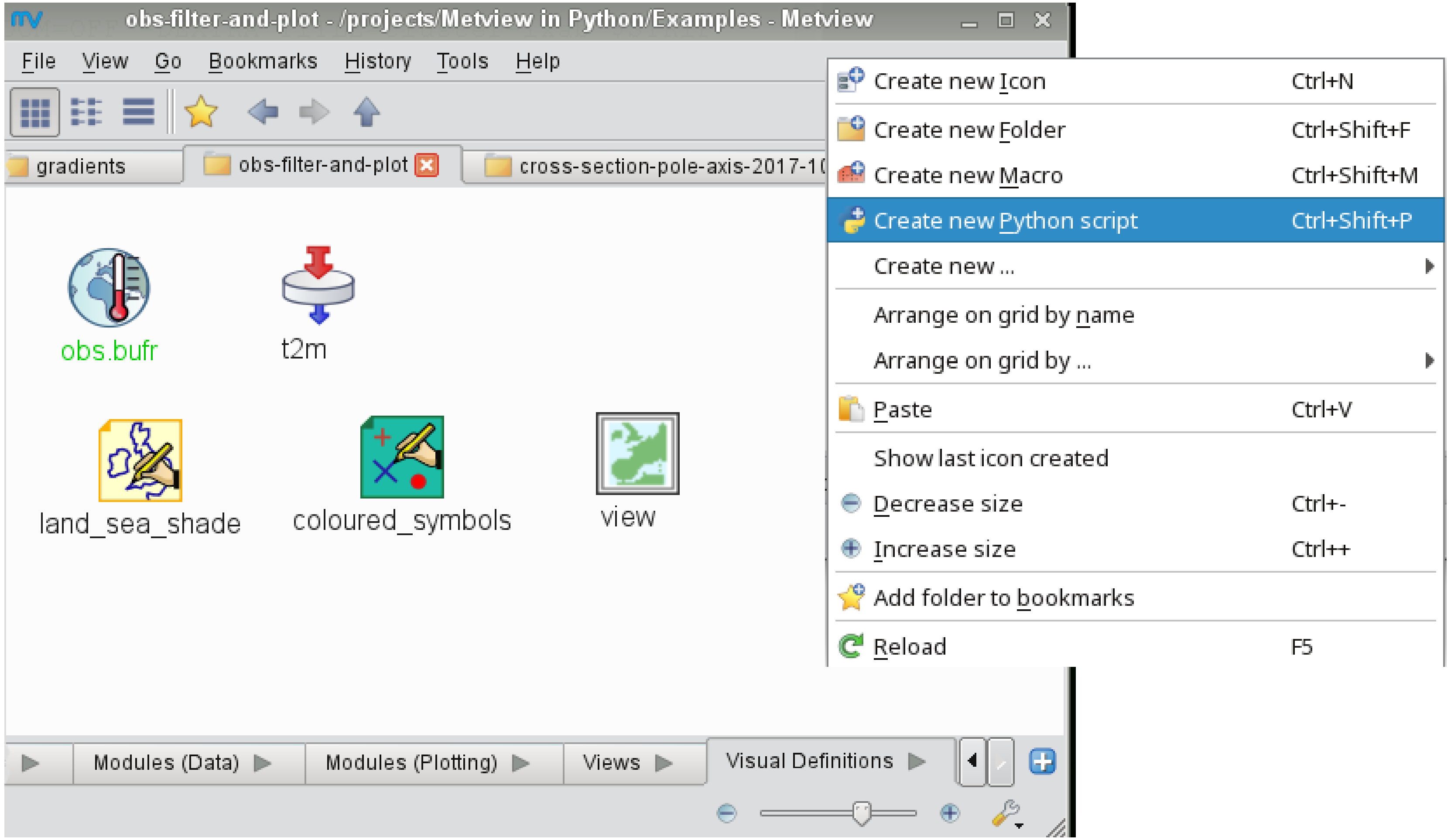

jupyter
Untitled Last Checkpoint: an hour ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Python 3
Code
CellToolbar
Monday 25 March 2013 00 UTC ecmf surface 2 metre temperature

```
In [2]:  import metview as mv

In [29]: t = mv.read('2m_temperature.grib')

         print(mv.datainfo(t))
         mv.cross_sect
         mv.datainfo
         mv.date
         mv.dates
         mv.day
         mv.db_info
         mv.definition
         mv.describe
         mv.det
         mv.dictionary
         [{'in                                                '0', 'proportion_present': '1', 'proportion_missing': '0'}]

In [30]: pal =
             l
             c
             c
             c
             c
             c
             c                                                d_25")

         polar = mv.geoview(
             map_projection       = "polar_stereographic",
             map_area_definition = "corners",
             area                 = [19.62,-31.44,39.66,80.1])
```

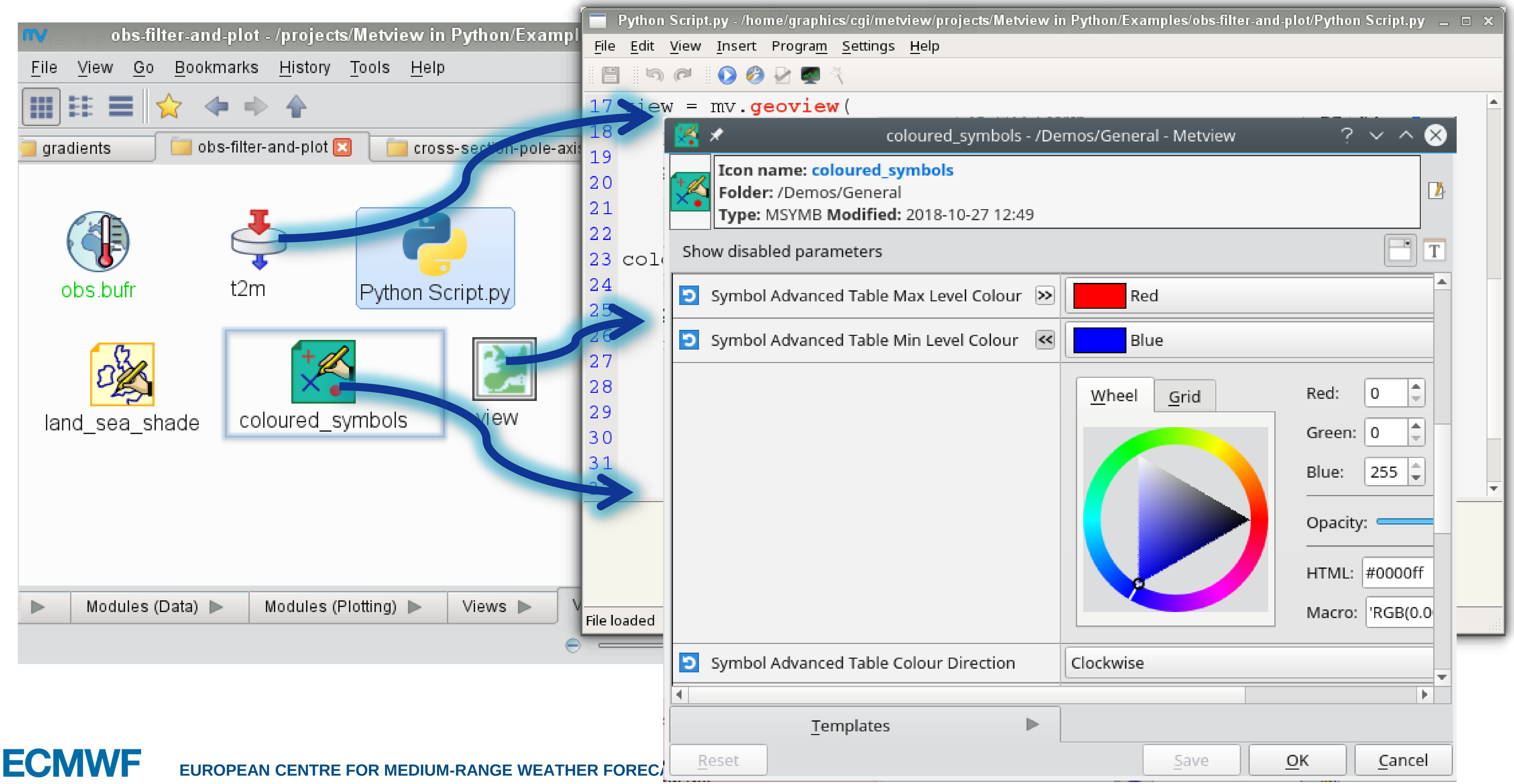
To view your plot in a Jupytet notebook, call "mv.setoutput('jupyter')" at some point before plotting

```
In [31]: mv.setoutput('jupyter')
         mv.plot(polar, t, pal)

Out[31]:
```

Running Python scripts from a Metview session (1)



Running Python scripts from a Metview session (2)



obs-filter-and-plot - /projects/Metview in Python/Exampl
File View Go Bookmarks History Tools Help
gradients
obs-filter-and-plot
cross-section-pole-axis
obs.bufr
t2m
Python Script.py
land_sea_shade
coloured_symbols
view
Modules (Data)
Modules (Plotting)
Views
Python Script.py - /home/graphics/cgi/metview/projects/Metview in Python/Examples/obs-filter-and-plot/Python Script.py
File Edit View Insert Program Settings Help
File loaded
coloured_symbols - /Demos/General - Metview
Show disabled parameters
Wheel
Grid
Templates
Reset
Save
OK
Cancel

```
view = mv.geoview(
col
```

Icon name: coloured_symbols
Folder: /Demos/General
Type: MSYMB Modified: 2018-10-27 12:49

Symbol Advanced Table Max Level Colour: 🟥 Red

Symbol Advanced Table Min Level Colour: 🟦 Blue

Red: 0

Green: 0

Blue: 255

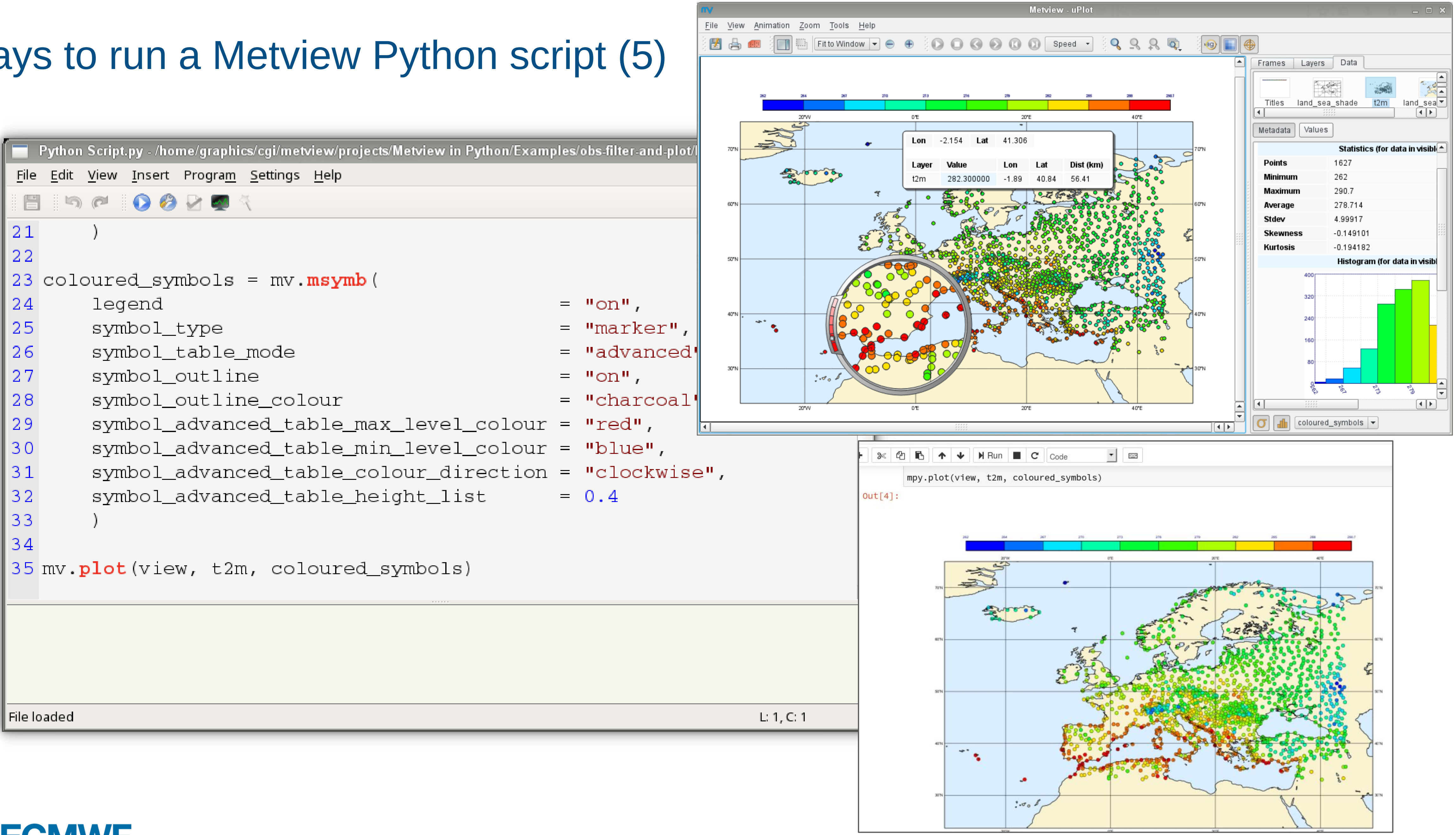
Opacity:

HTML: #0000ff

Macro: 'RGB(0.0

Symbol Advanced Table Colour Direction: Clockwise

Ways to run a Metview Python script (5)



Python Script.py - /home/graphics/cgi/metview/projects/Metview in Python/Examples/obs-filter-and-plot/

```
    )

coloured_symbols = mv.msymb(
    legend                                = "on",
    symbol_type                           = "marker",
    symbol_table_mode                     = "advanced"
    symbol_outline                        = "on",
    symbol_outline_colour                 = "charcoal"
    symbol_advanced_table_max_level_colour = "red",
    symbol_advanced_table_min_level_colour = "blue",
    symbol_advanced_table_colour_direction = "clockwise",
    symbol_advanced_table_height_list     = 0.4
    )

mv.plot(view, t2m, coloured_symbols)
```

File loaded L: 1, C: 1

Metview - uPlot

Lon	-2.154	Lat	41.306	
Layer	Value	Lon	Lat	Dist (km)
t2m	282.300000	-1.89	40.84	56.41

Statistics (for data in visibl

Points	1627
Minimum	262
Maximum	290.7
Average	278.714
Stdev	4.99917
Skewness	-0.149101
Kurtosis	-0.194182

Histogram (for data in visibl

```
mpy.plot(view, t2m, coloured_symbols)
```

Out[4]:

Implementation details

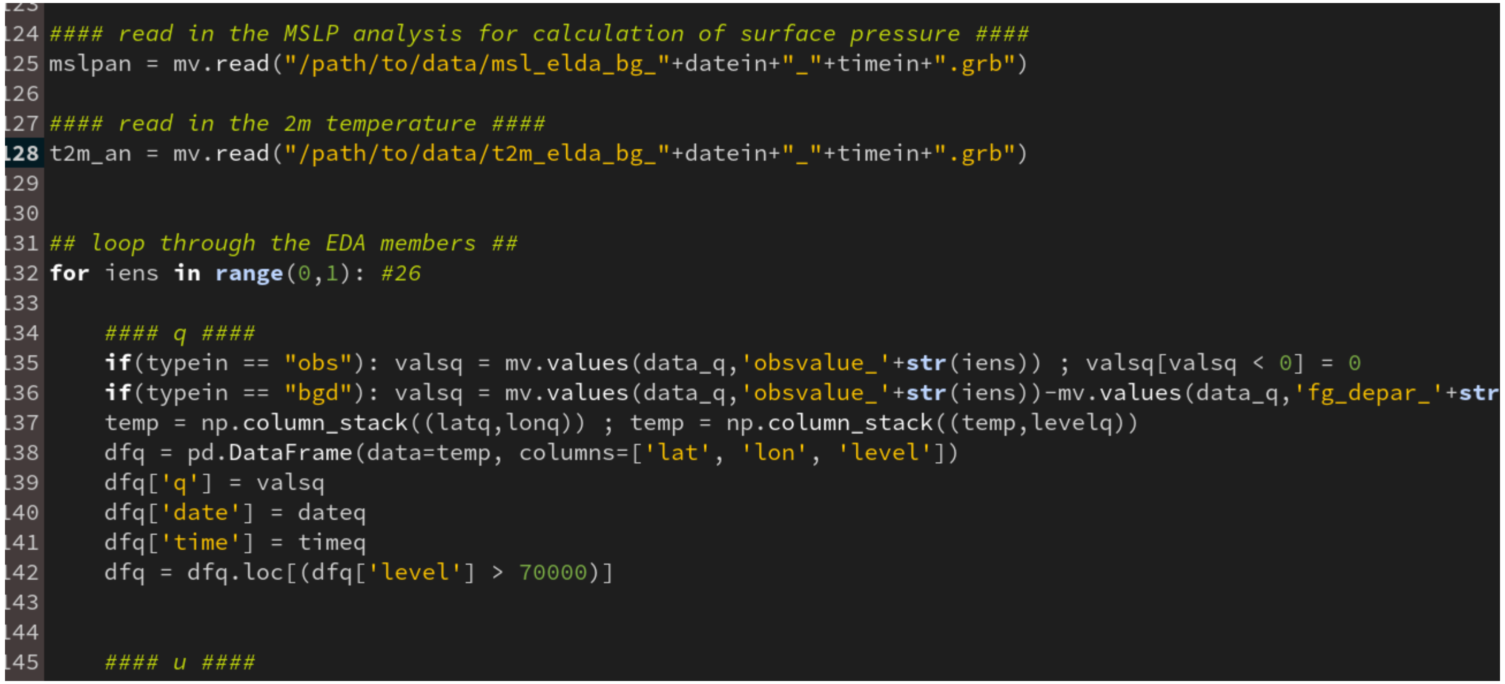
- We use the **cffi** package to bridge C++/Python
  - Links to a shared library of Metview functions
  - Some of these functions call other Metview services (e.g. Cross Section, uPlot)
  - Metview now needs to be context-aware because Macro uses 1-based indexing, Python uses 0-based indexing

- Metview binaries vs Python layer independence
  - Try to keep the interface functions as generic as possible
  - The Python layer queries the binaries for the list of available functions
  - New data types in Metview will require a little code in the Python layer

- Faster import
  - we noticed that importing some modules was quite slow (e.g. IPython for detecting the Jupyter environment), so we only import when actually needed

Feedback

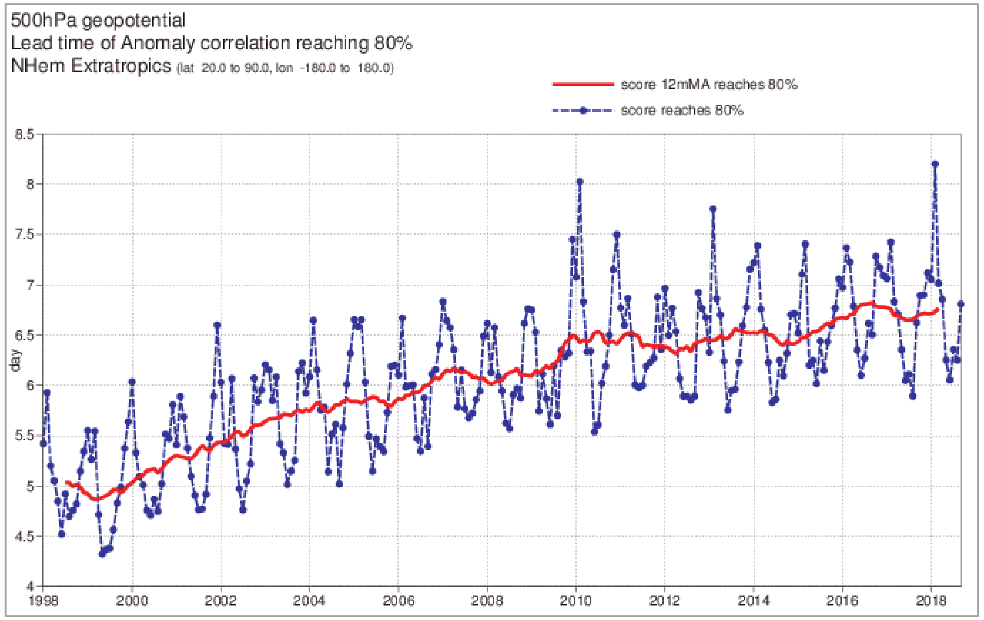
- Feedback has so far been very positive
- We have some enthusiastic users: “It combines all the power of Metview with all the power of Python!” (internal user to Iain, Oct 2018)
- As is often the case, we only hear from users if they encounter a problem, but log files suggest quite a lot of activity



```
#### read in the MSLP analysis for calculation of surface pressure ####
mslpan = mv.read("/path/to/data/msl_elda_bg_"+datein+"_"+timein+".grb")

#### read in the 2m temperature ####
t2m_an = mv.read("/path/to/data/t2m_elda_bg_"+datein+"_"+timein+".grb")


## loop through the EDA members ##
for iens in range(0,1): #26

    #### q ####
    if(typein == "obs"): valsq = mv.values(data_q,'obsvalue_'+str(iens)) ; valsq[valsq < 0] = 0
    if(typein == "bgd"): valsq = mv.values(data_q,'obsvalue_'+str(iens))-mv.values(data_q,'fg_depar_'+str
    temp = np.column_stack((latq,lonq)) ; temp = np.column_stack((temp,levelq))
    dfq = pd.DataFrame(data=temp, columns=['lat', 'lon', 'level'])
    dfq['q'] = valsq
    dfq['date'] = dateq
    dfq['time'] = timeq
    dfq = dfq.loc[(dfq['level'] > 70000)]


    #### u ####
```

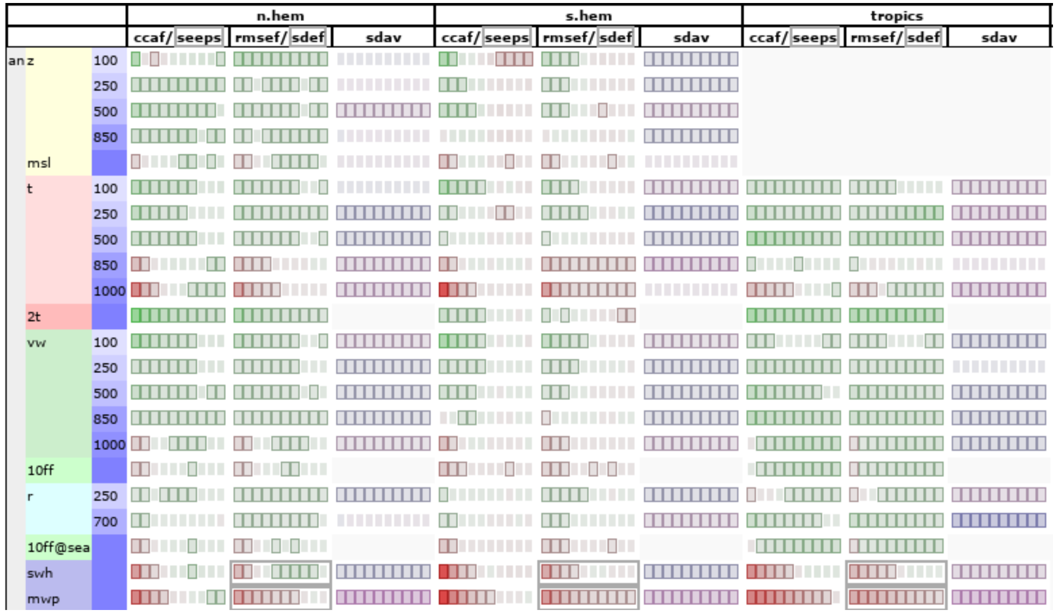
Reaching out – forecast verification toolbox (1)

- One of the major verification packages at ECMWF has been using Python for > 10 years
- Provides a simple interface to describe which statistics of what forecasts are to be computed
  - data format details, etc are hidden
- The package has been successful, with some shortcomings:
  - interfacing to low-level data decoding packages and consequently implementing own geographical and meteorological algorithms
  - lack of flexibility for newly emerged verification and diagnostic techniques and requirements



Reaching out – forecast verification toolbox (2)

- The solution: a new verification toolbox built on Metview’s Python layer
  - replace the data interfacing and manipulation layer with Metview
  - take the opportunity to improve other layers
  - involve developers of other verification packages to broaden the scope of the toolbox
  - repack the user interface layer to fully use the toolbox (to support the existing users)



Future

- Advertise beta version more widely to get more feedback
- Release version 1.0.0 – end of 2018 / early 2019?

- Provide tools for automatic translation from Macro to Python
- Improve information available for IDEs (e.g. function descriptions)

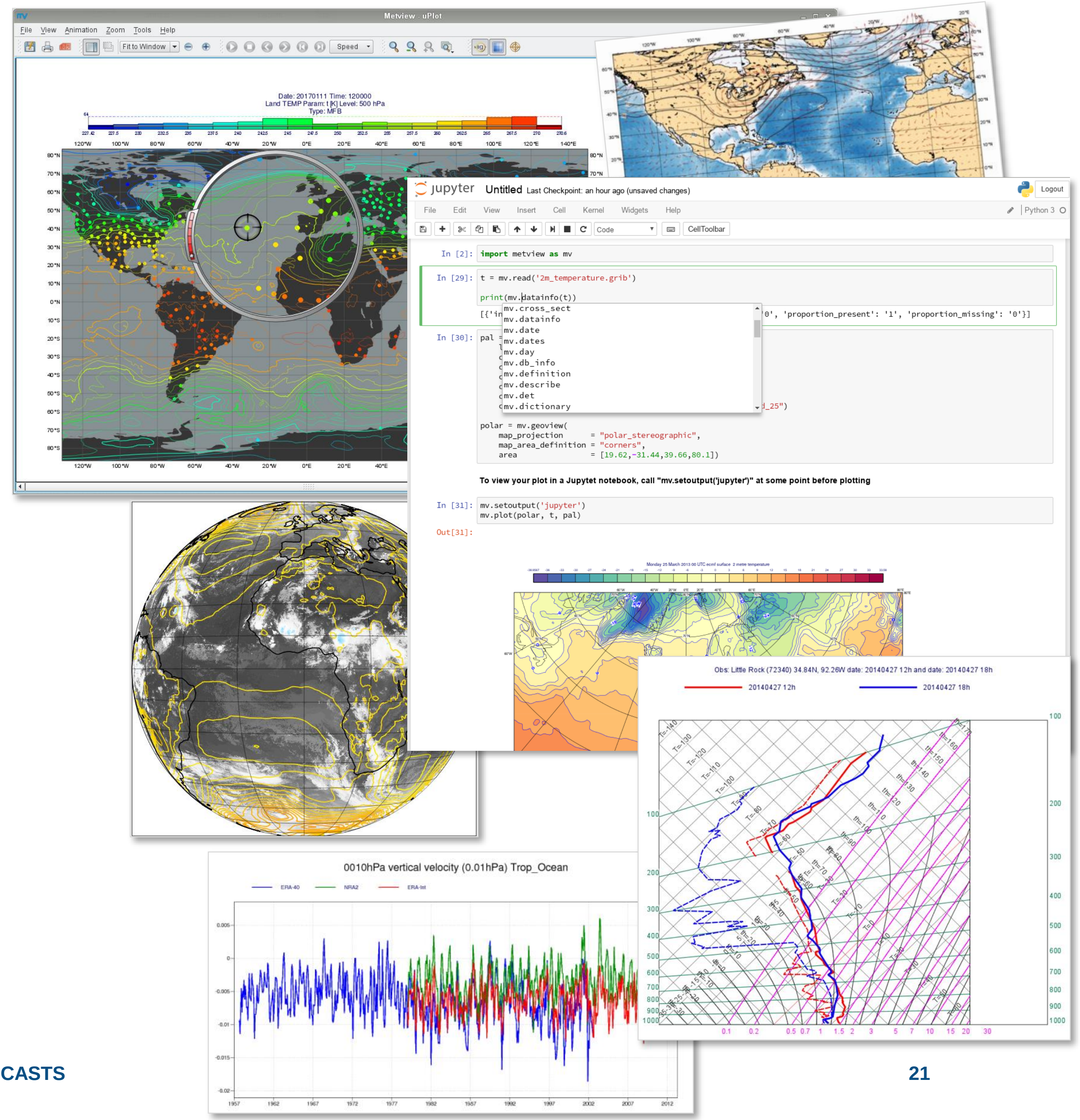
- Investigate **conda** for packaging Metview's binaries and the Python layer together

- Plenty more we want to do!

For more information…

- Email us:
  - Developers: metview@ecmwf.int
  - Support: software.support@ecmwf.int
- Visit our web pages:
  - http://confluence.ecmwf.int/metview
- Download (Metview source, binaries)
- Documentation and tutorials available
- Metview articles in ECMWF newsletters
- e-Learning material
- Download Metview's Python interface:
  - pip install metview
  - https://github.com/ecmwf/metview-python

Questions?


Metview - uPlot
File View Animation Zoom Tools Help
Fit to Window
Speed
Date: 20170111 Time: 120000
Land TEMP Param: t [K] Level: 500 hPa
Type: MFB
jupyter Untitled Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Python 3
Code
CellToolbar

```
In [2]: import metview as mv

In [29]: t = mv.read('2m_temperature.grib')

         print(mv.datainfo(t))
         [{'in                                   '0', 'proportion_present': '1', 'proportion_missing': '0'}]

In [30]: pal =
                                                  d_25")

         polar = mv.geoview(
             map_projection      = "polar_stereographic",
             map_area_definition = "corners",
             area                = [19.62,-31.44,39.66,80.1])
```

```
mv.cross_sect
mv.datainfo
mv.date
mv.dates
mv.day
mv.db_info
mv.definition
mv.describe
mv.det
mv.dictionary
```

To view your plot in a Jupytet notebook, call "mv.setoutput('jupyter')" at some point before plotting

```
In [31]: mv.setoutput('jupyter')
         mv.plot(polar, t, pal)

Out[31]:
```

Monday 25 March 2013 00 UTC ecmf surface 2 metre temperature
0010hPa vertical velocity (0.01hPa) Trop_Ocean
ERA-40
NRA2
ERA-Int
Obs: Little Rock (72340) 34.84N, 92.26W date: 20140427 12h and date: 20140427 18h
20140427 12h
20140427 18h