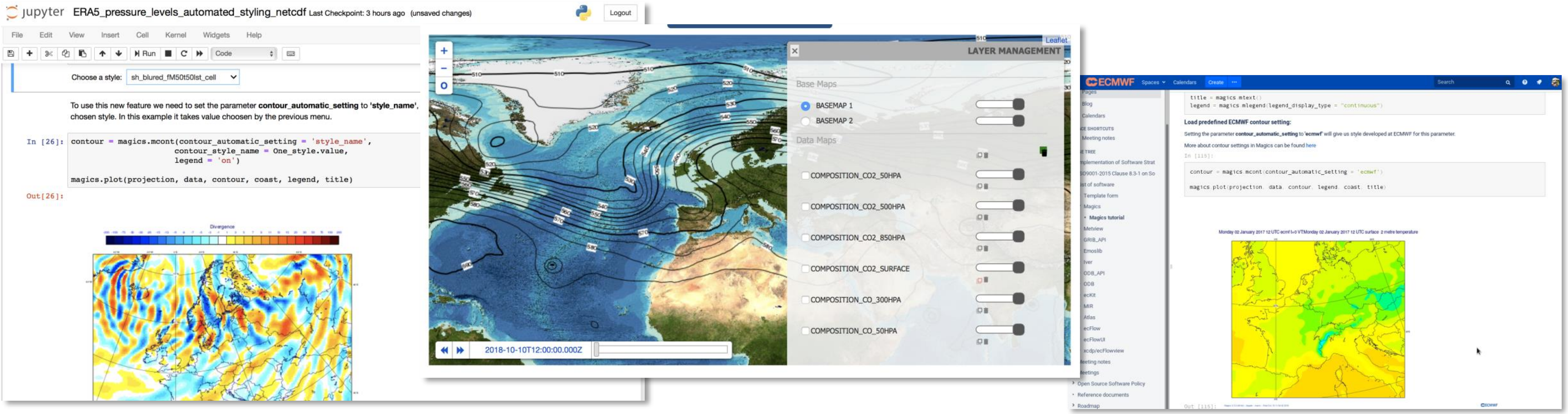
Easy visualisation with Magics



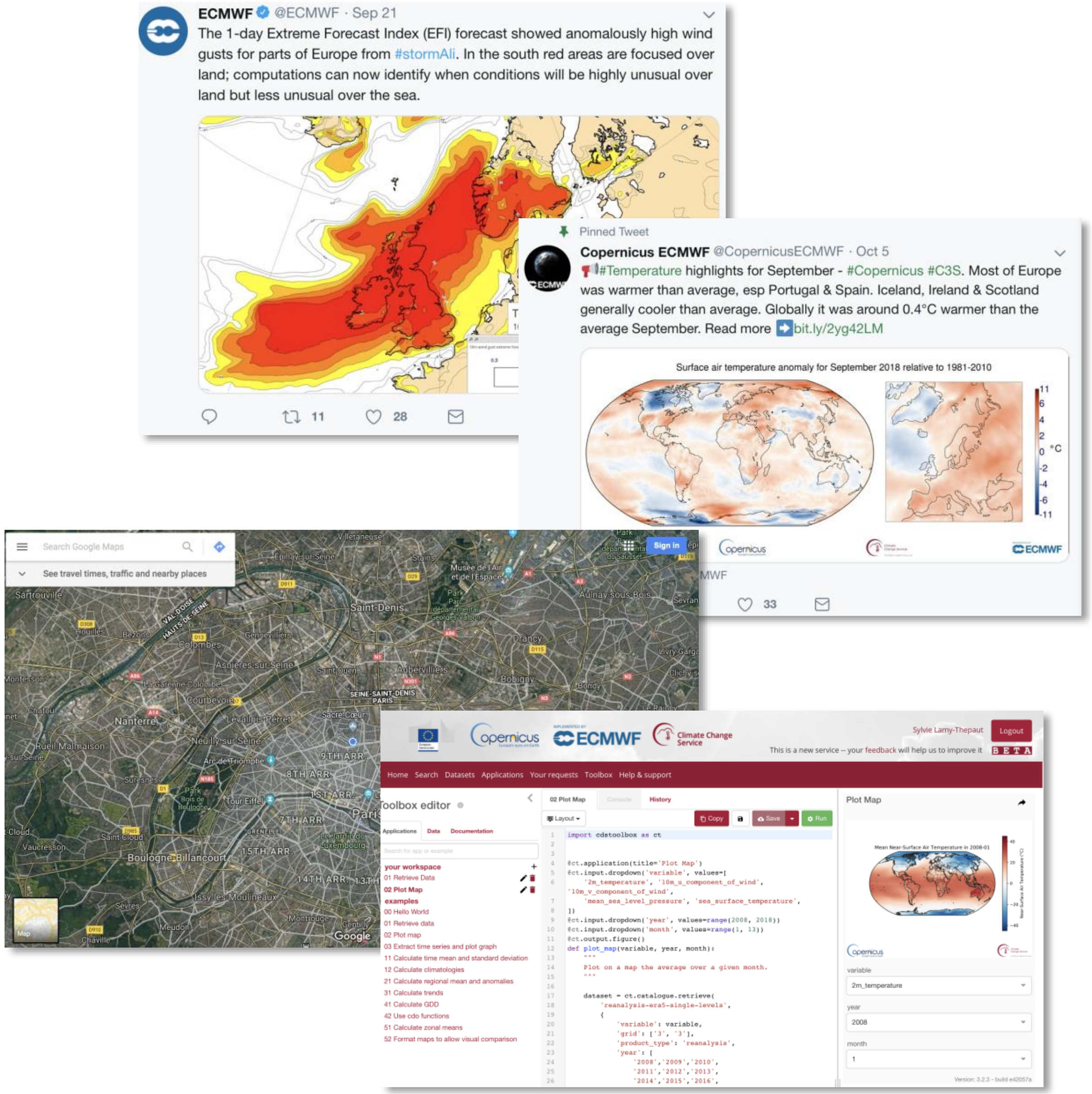
Milana Vuckovic - Sylvie Lamy-Thépaut

ECMWF

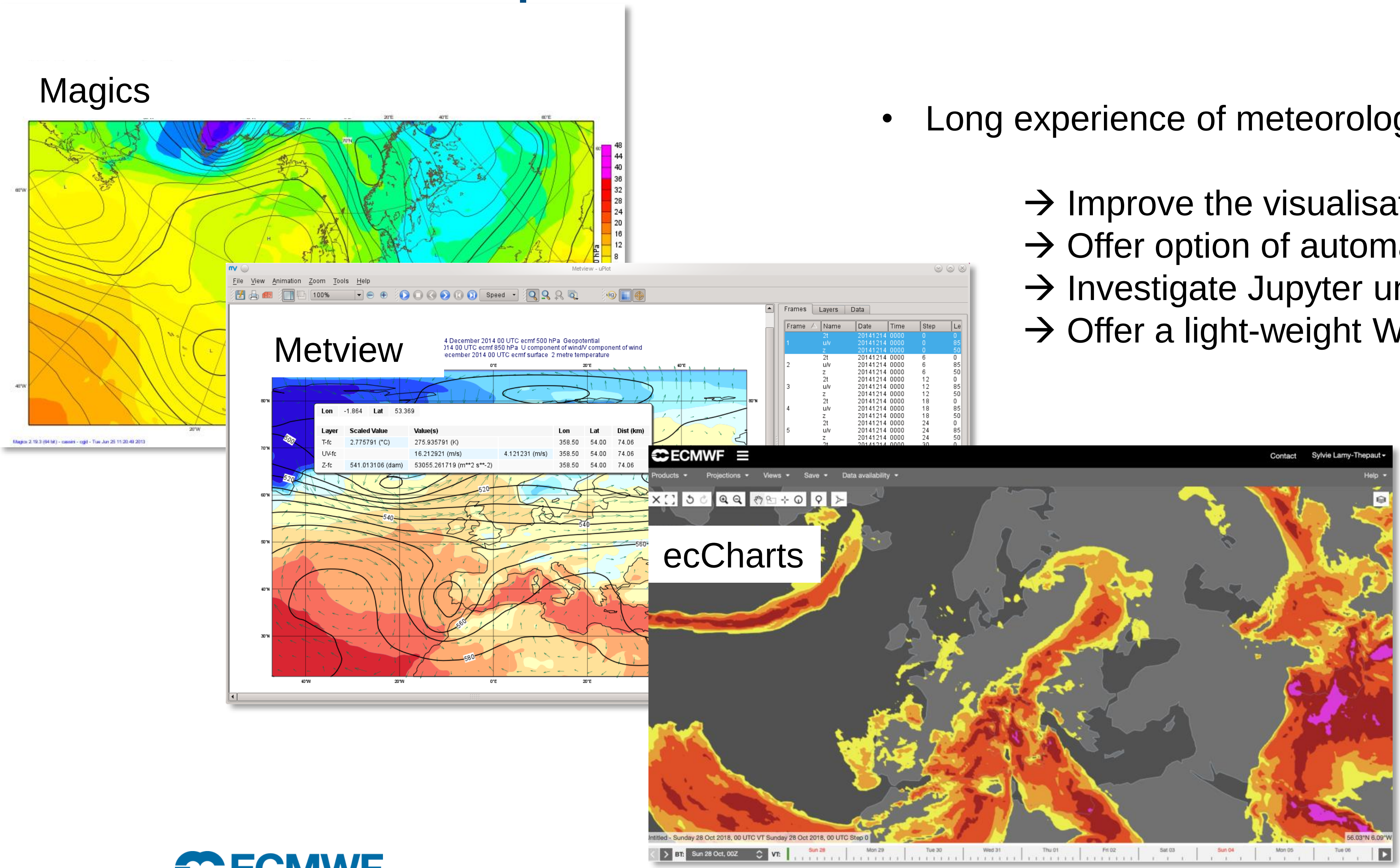
Motivation

Users want:

- An easy way to inspect meteorological data
- An easy way to share results of their work
- Interactive work with data
- Unified presentation of data



How can we help ?

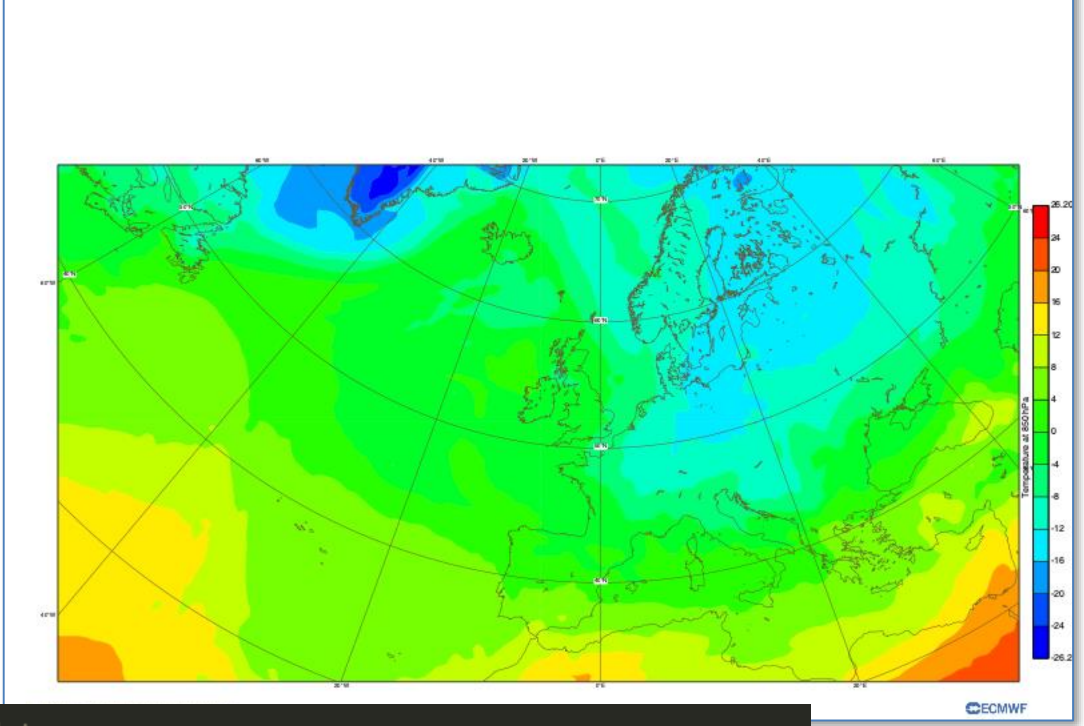


- Long experience of meteorological visualisation
  - → Improve the visualisation of NetCDF
  - → Offer option of automatic styling
  - → Investigate Jupyter universe
  - → Offer a light-weight WMS

Magics principles

shading.py

```python

#importing Magics module
from Magics.macro import *

#Setting of the output file name
output = output(output_formats= ['png'],
                output_name_first_page_number= "off",
                output_name= "myplot")

#Setting the coordinates of the geographical area
europe = mmap(subpage_upper_right_longitude= 65.,
              subpage_map_projection= "polar_stereographic",
              subpage_map_vertical_longitude= 0.0,
              subpage_lower_left_longitude= -37.27,
              subpage_lower_left_latitude= 21.51,
              subpage_upper_right_latitude= 51.28)

#Coastlines setting
coast = mcoast( map_grid_colour= "tan",
                map_coastline_colour= "tan")

#Import the t850 data
t850 =  mgrib( grib_input_file_name = "./t850.grb",)

#Define the shading for t850
t850_contour = mcont(legend="on",
             contour_shade= "on",
             contour_hilo= "off",
             contour= "off",
             contour_label= "off",
             contour_shade_method= "area_fill",
             contour_level_selection_type= "interval",
             contour_interval= 4.,
             contour_shade_colour_method= "calculate",
             contour_shade_max_level_colour = "red",
             contour_shade_min_level_colour = "blue",
             contour_shade_colour_direction = "clockwise")

#To the plot
plot(output, europe, t850, t850_contour, coast, legend)

```



```python
#Import the t850 data
t850 =  mnetcdf( data = magics.mnetcdf(netcdf_filename = "2t.nc",
                 netcdf_value_variable = "t2m") )
```

```python
#Define the shading for t850
t850_contour = mcont(contour_automatic_setting = 'ecmwf', legend='on')
```

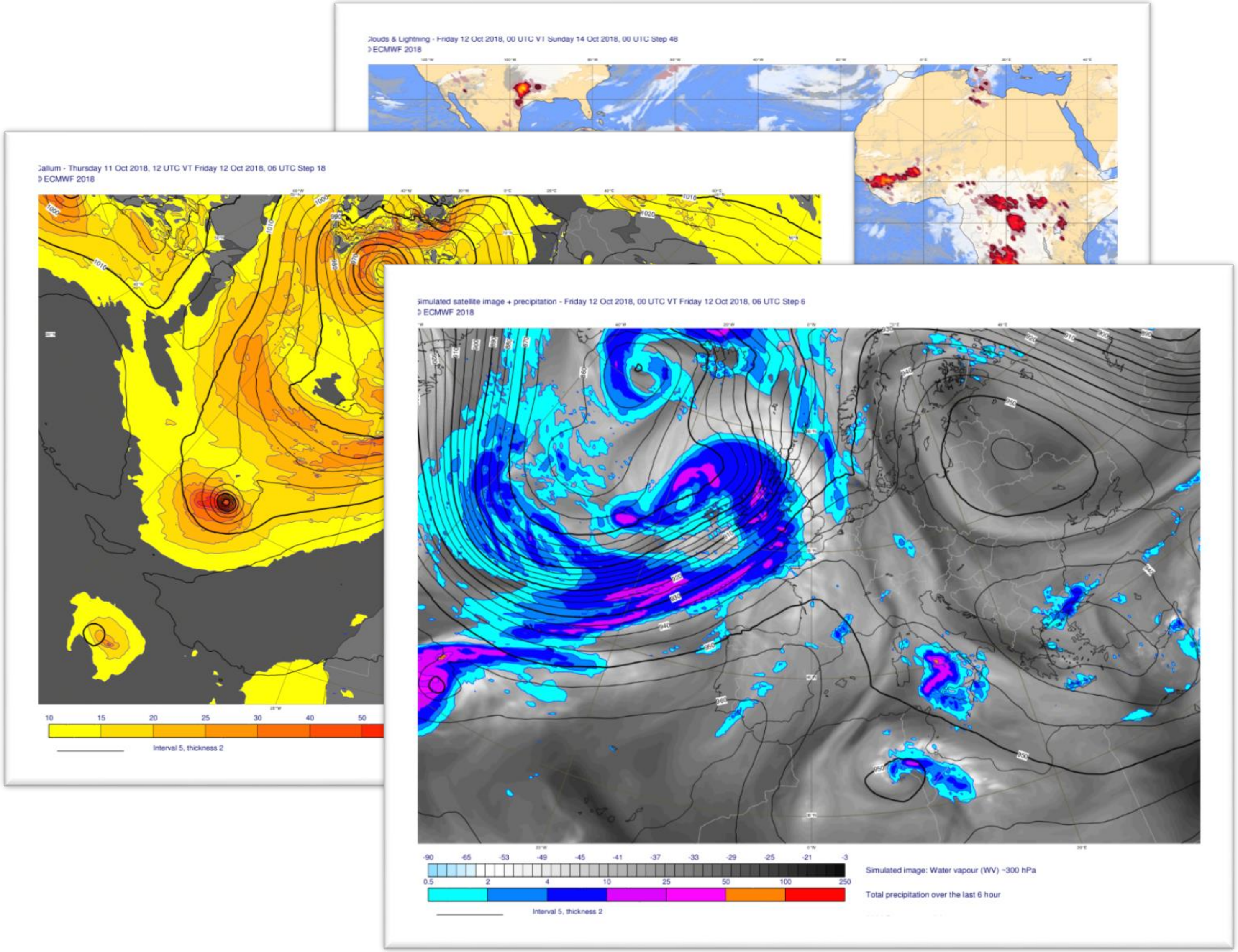
Automatic visualisation : where to start ? ecCharts !

- EcCharts products are used among many member states and their styles are recognizable for users

- There are already styles for over 250 meteorological parameters

- For most parameters there is more than one style

- Making reproducing ecCharts plots almost trivial



Teaching Magics to recognise data

Inspecting grib keys

```
#==============   MESSAGE 2 ( length=2076588 )            ==============
GRIB {
  editionNumber = 1;
  table2Version = 128;
  # European Centre for Medium-Range Weather Forecasts (common/c-1.table)
  centre = 98;
  generatingProcessIdentifier = 145;
  # Temperature  (K)  (grib1/2.98.128.table)
  indicatorOfParameter = 130;
  # Isobaric level pressure in hectoPascals  (hPa)  (grib1/local/ecmf/3.table , grib1/3.table)
  indicatorOfTypeOfLevel = 100;
  level = 250;
  # Forecast product valid at reference time + P1  (P1>0)  (grib1/local/ecmf/5.table , grib1/5.table)

  timeRangeIndicator = 0;
  # Unknown code table entry (grib1/0.ecmf.table)
  subCentre = 0;
  paramId = 130;
  #-READ ONLY- cfNameECMF = air_temperature;
  #-READ ONLY- cfName = air_temperature;
  #-READ ONLY- cfVarNameECMF = t;
  #-READ ONLY- cfVarName = t;
  #-READ ONLY- units = K;
  #-READ ONLY- nameECMF = Temperature;
  #-READ ONLY- name = Temperature;
  decimalScaleFactor = 0;
  dataDate = 20100202;
  dataTime = 0;
  # Hour (stepUnits.table)
  stepUnits = 1;
  stepRange = 0;
  startStep = 0;
  endStep = 0;
  #-READ ONLY- marsParam = 130.128;
  # MARS labelling or ensemble forecast data (grib1/localDefinitionNumber.98.table)
  localDefinitionNumber = 1;
  # ERA5 (mars/class.table)
  marsClass = 23;
  # Analysis (mars/type.table)
  marsType = 2;
  # Atmospheric model (mars/stream.table)
  marsStream = 1025;
  experimentVersionNumber = 0001;
  perturbationNumber = 0;
  numberOfForecastsInEnsemble = 0;
  shortName = t;
  GDSPresent = 1;
  bitmapPresent = 0;
  numberOfVerticalCoordinateValues = 0;
  Ni = 1440;
  Nj = 721;
  latitudeOfFirstGridPointInDegrees = 90;
```

-> Creating rules:

```
{
   "match" : {
         "prefered_units" : "C",
      "set" : [
         {
            "levelist" : ["250”],
            "paramId" : "130",
            "shortName" : "t",
            "levtype" : "pl"
         },
],
      "style" : "sh_all_fM64t52i4",
      "styles" : [
         "sh_all_fM64t52i4",
         "ct_red_i2_dash",
         "sh_gry_fM72t56lst",
         "sh_all_fM80t56i4_v2",
         "sh_all_fM50t58i2",
         "ct_red_i4_t3"
      ]
   }
}
```

-> Applying Magics definition

```
"sh_all_fM64t52i4" : {
        "contour" : "off",
         "contour_hilo" : "off",
        "contour_interval" : 4,
        "contour_label" : "off",
        "contour_level_selection_type" : "interval",
        "contour_line_thickness" : 3,
        "contour_shade" : "on",
        "contour_shade_colour_list" :
"rgb(0,0,0.1)/rgb(0.1,0,0.2)/…/red/magenta",
        "contour_shade_colour_method" : "list",
        "contour_shade_max_level" : 52,
        "contour_shade_method" : "area_fill",
        "contour_shade_min_level" : -72
    },
```

Teaching Magics to recognise data

NetCDF

```
netcdf pl {
dimensions:
	longitude = 360 ;
	latitude = 181 ;
	level = 3 ;
	time = 4 ;
variables:
	float longitude(longitude) ;
		longitude:units = "degrees_east" ;
		longitude:long_name = "longitude" ;
	float latitude(latitude) ;
		latitude:units = "degrees_north" ;
		latitude:long_name = "latitude" ;
	int level(level) ;
		level:units = "millibars" ;
		level:long_name = "pressure_level" ;
	int time(time) ;
		time:units = "hours since 1900-01-01 00:00:0.0" ;
		time:long_name = "time" ;
		time:calendar = "gregorian" ;
	short t(time, level, latitude, longitude) ;
		t:scale_factor = 0.00149840526246974 ;
		t:add_offset = 262.173239139654 ;
		t:_FillValue = -32767s ;
		t:missing_value = -32767s ;
		t:units = "K" ;
		t:long_name = "Temperature" ;
		t:standard_name = "air_temperature" ;
	short r(time, level, latitude, longitude) ;
		r:scale_factor = 0.00251813640893975 ;
		r:add_offset = 67.851697226809 ;
		r:_FillValue = -32767s ;
		r:missing_value = -32767s ;
		r:units = "%" ;
		r:long_name = "Relative humidity" ;
		r:standard_name = "relative_humidity" ;

// global attributes:
		:Conventions = "CF-1.6" ;
		:history = "2018-07-02 14:23:28 GMT by grib_to_netcdf-2.7.3: grib_to_netcdf pl.grib -o
 pl.nc" ;
}
```

```
{
   "match" : {
      "eccharts_layer" : "t250",
      "prefered_units" : "C",
      "set" : [
         {
            "levelist" : ["250"],
            "paramId" : "130",
            "shortName" : "t",
            "levtype" : "pl"
         },
         {
            "level" : [250]],
            "long_name" : "Temperature",
            "standard_name" : "air_temperature"
         }
      ],
      "style" : "sh_all_fM64t52i4",
      "styles" : [
         "sh_all_fM64t52i4",
       ]
   }
}
```

Units and Scaling

Why?

- Some styles in ecCharts require specific units (mm for precipitation, °C for temperature, hPa for MSLP)
- Some units are just more common than the original units in file

What we did?

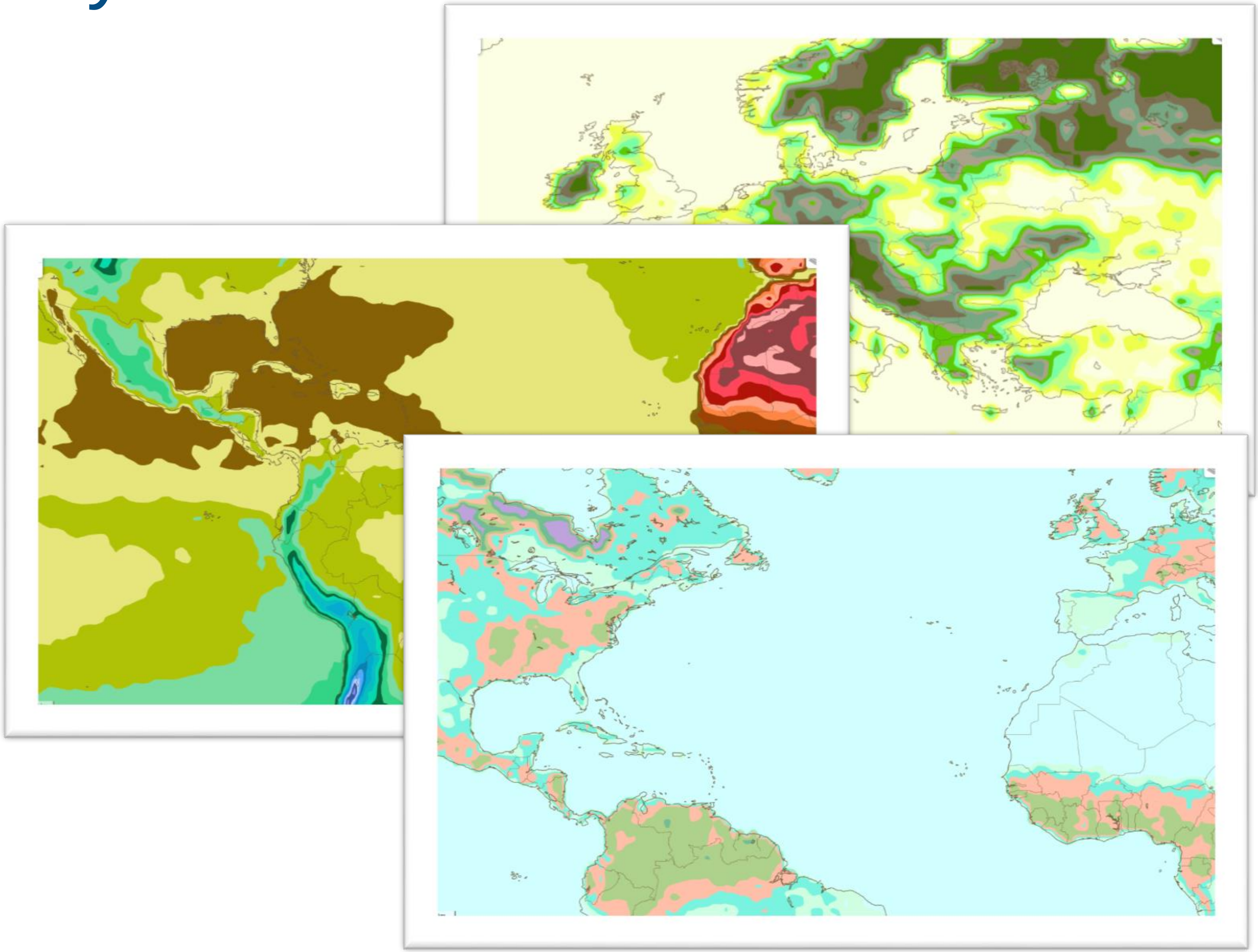
- Implemented new built in scaling in Magics, that works when units in file are different than preferred units in definition for style for parameter

But…..

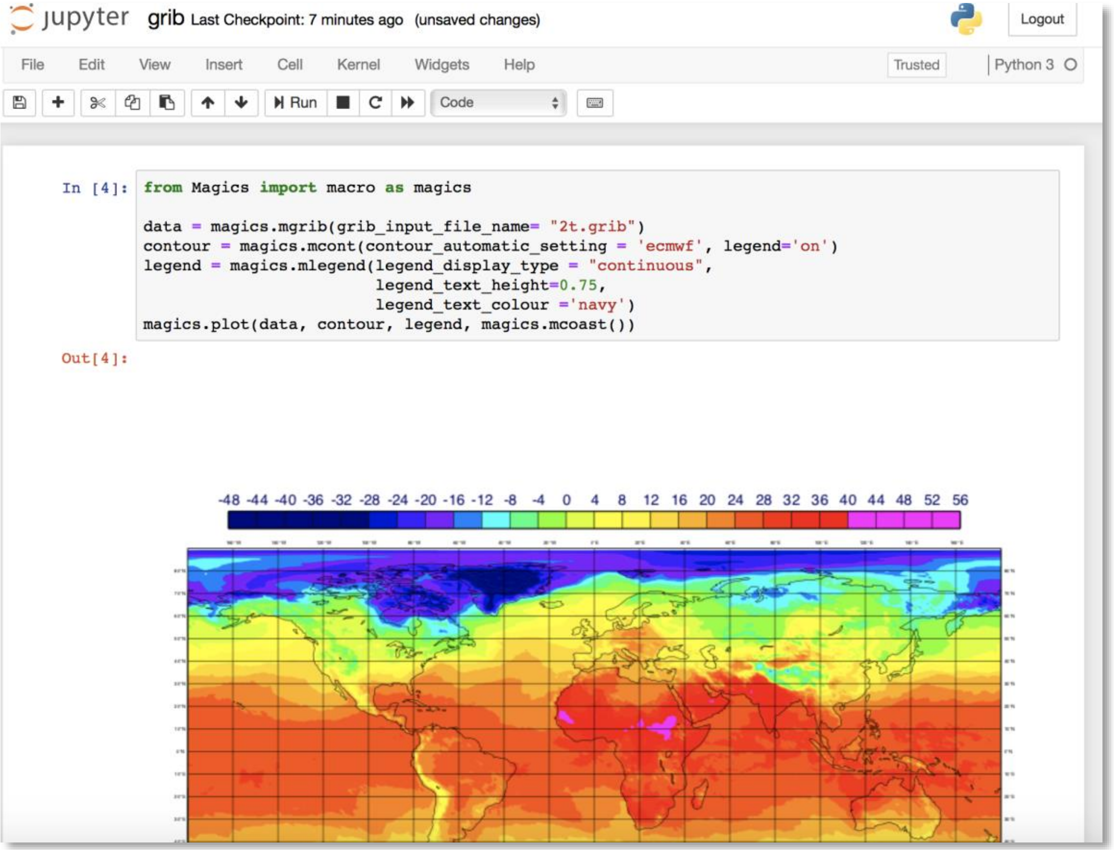
- Units are not always the same in grib and NetCDF

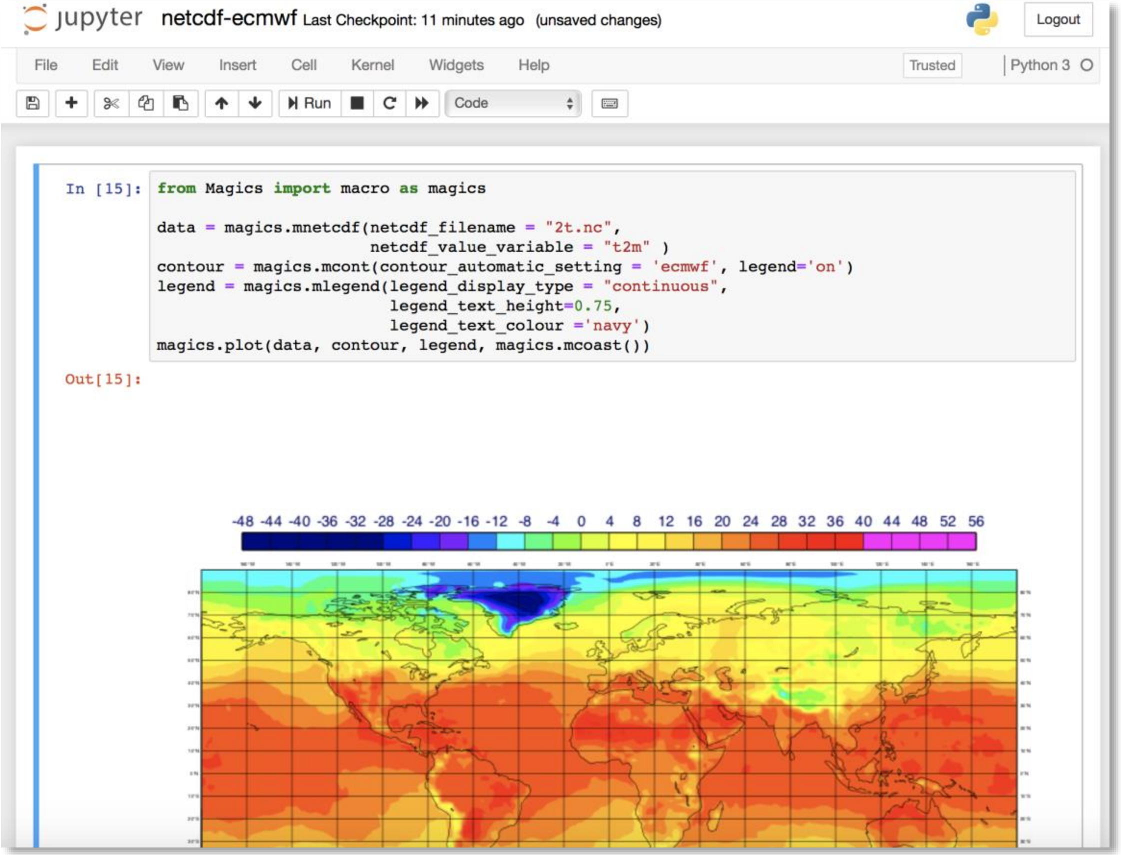
A solid framework for styles

- There are many meteorological parameters not present in ecCharts
- We started designing styles for most important ones
- Introduction of predefined palettes



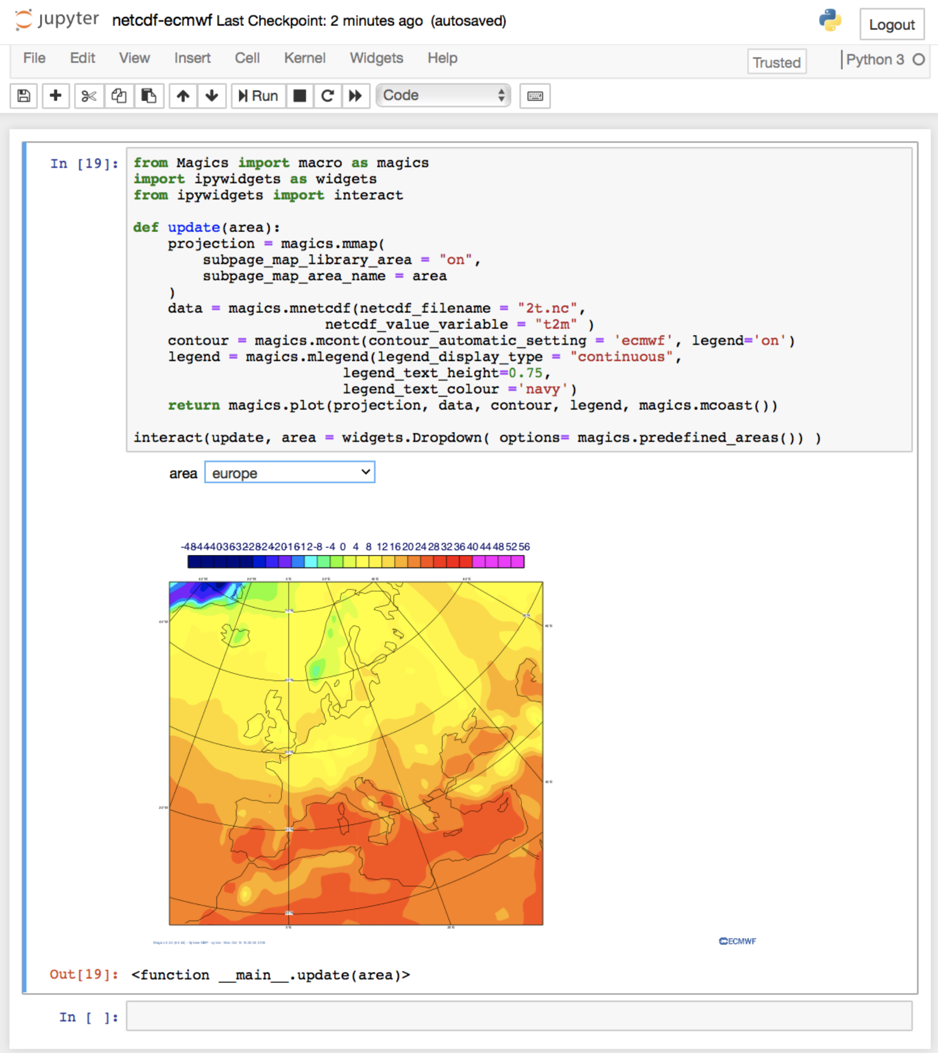
Better handling of NetCDF





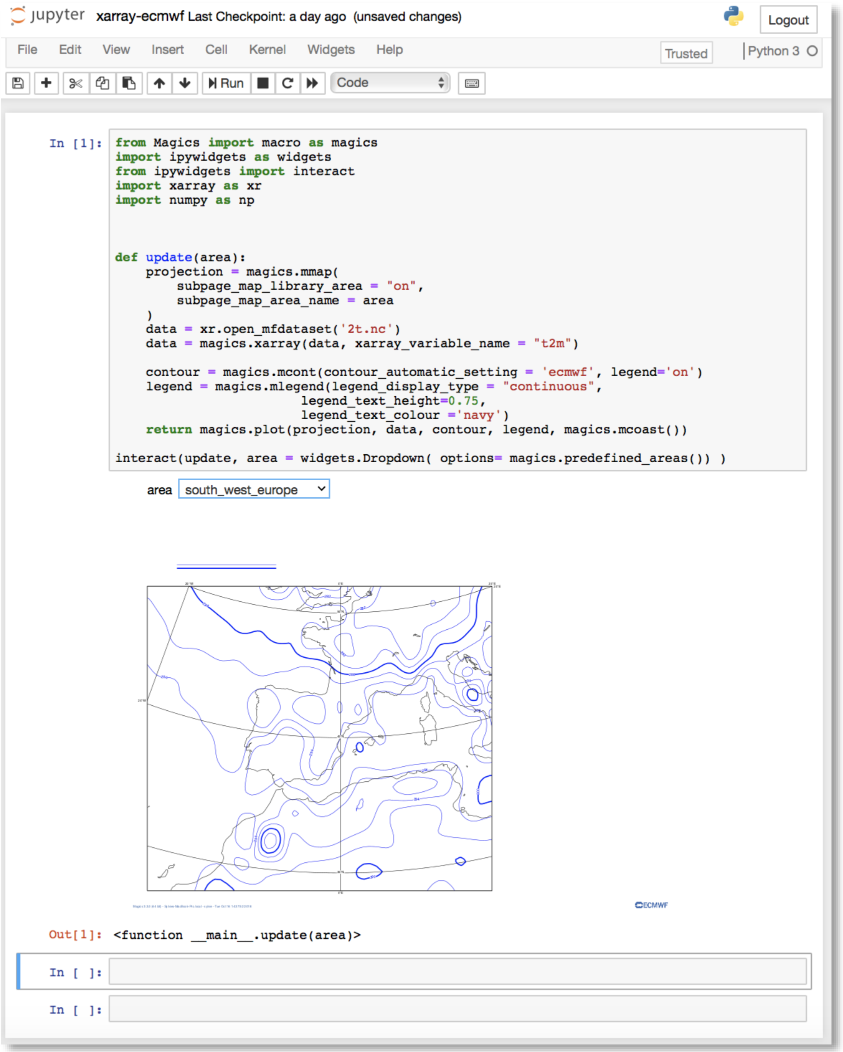
Better handling of NetCDF

- Automatic guess of the internal representation
- Automatic geo – referencement
- Scaling
- Automatic visualisation

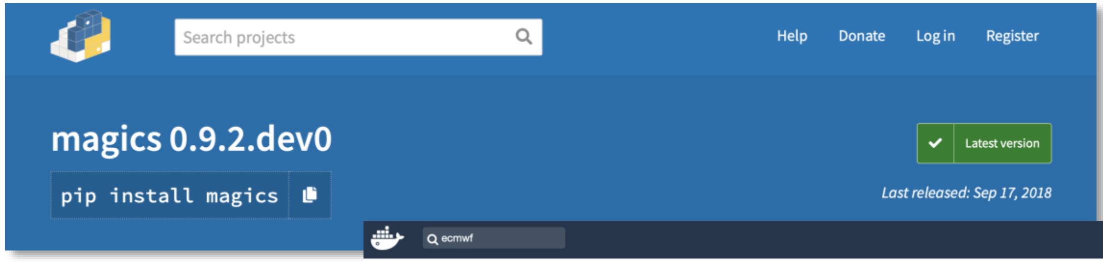


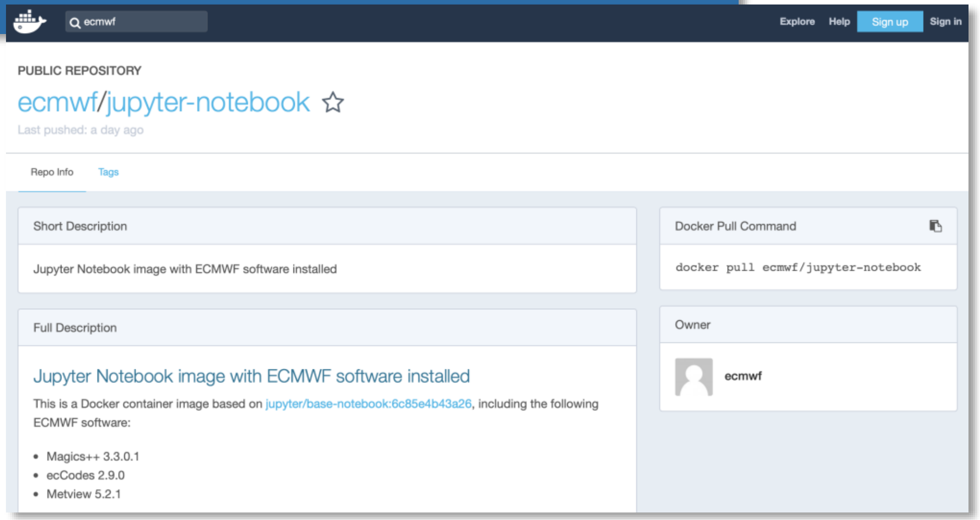
What about xarray ?

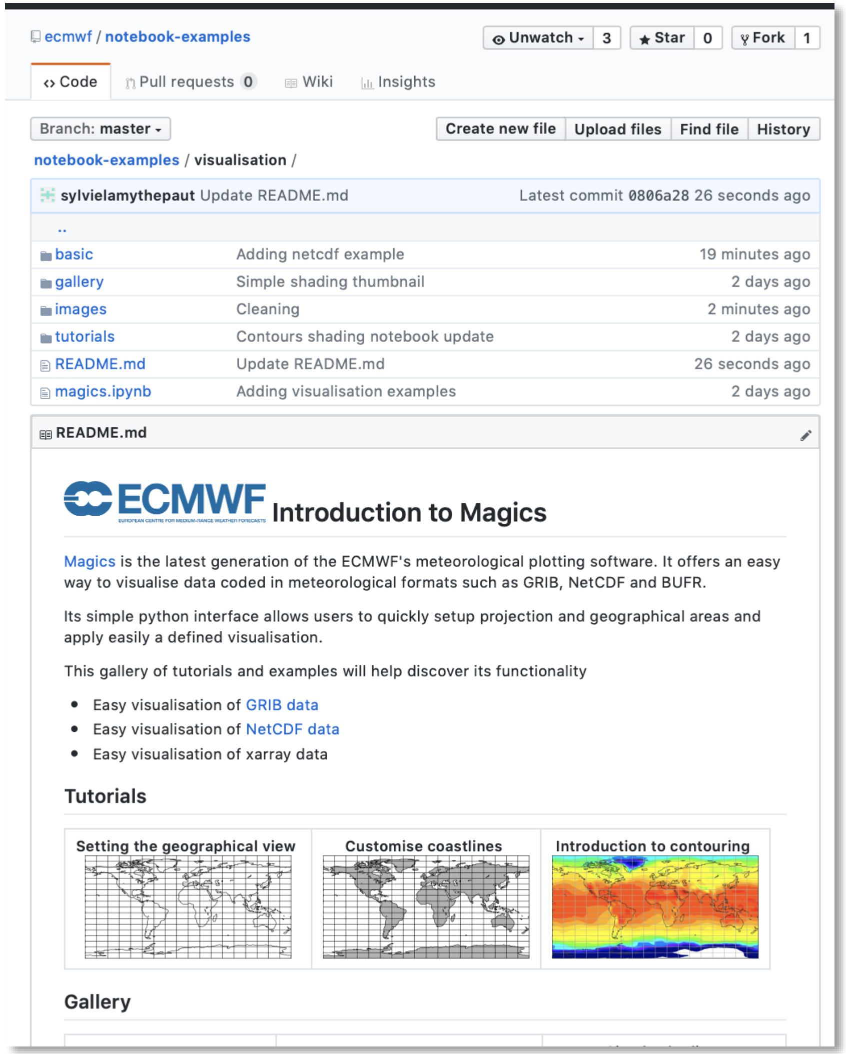
- xarray has become one of the most popular tools for working with data
- Both GRIB and NetCDF can be loaded as xarray dataset
- The metadata attached will be used to setup an automatic visualisation



How to facilitate the use of Magics?







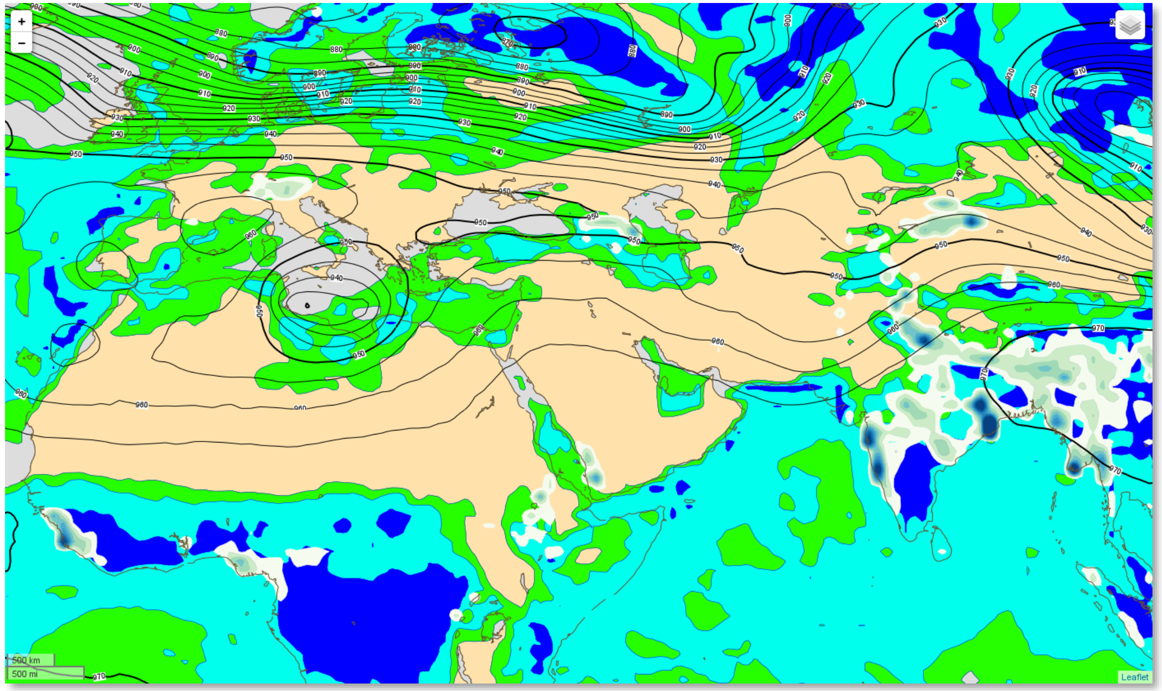
docker run --rm -p 8888:8888 -v "$PWD":/home/jovyan/work ecmwf/jupyter-notebook

Another use of the automatic styling: The skinny WMS

- The idea:
  - scan directory with NetCDF or grib data to collect:
    - Base time, steps and valid time
    - Relevant styles (detected by Magics)

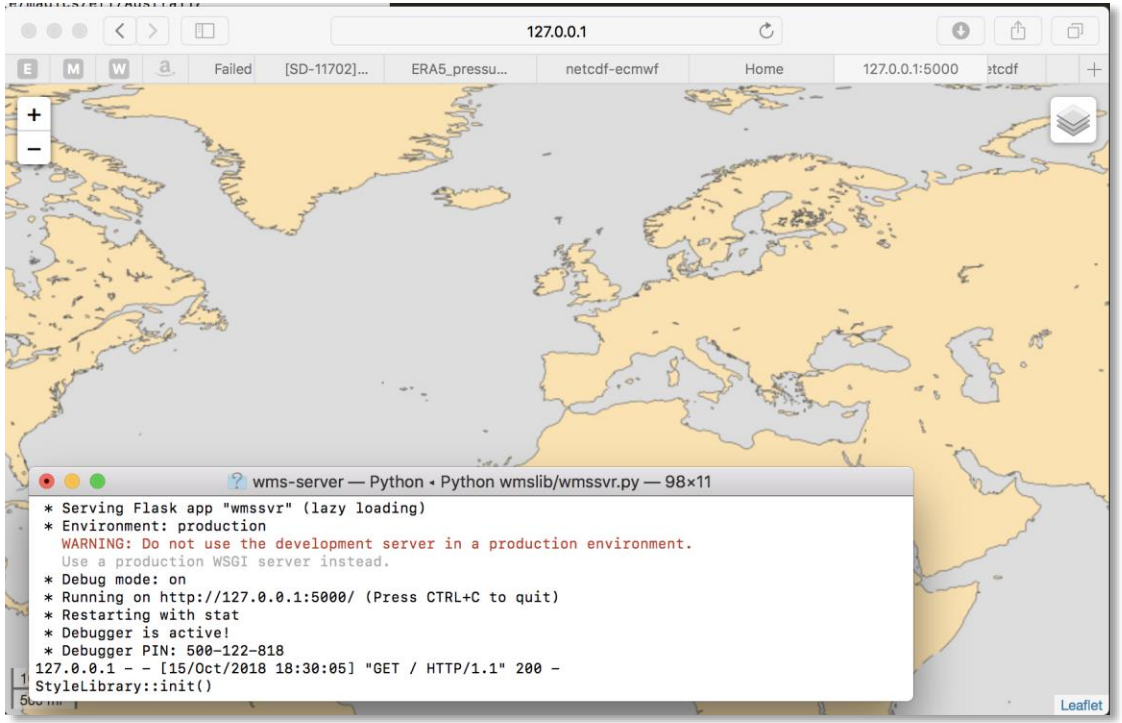
    → GetCapabilities
  - Call Magics to render the image (format+projection+data+style)

    → GetMap



"Skinny" WMS – our way to do it

- The implementation :
  - Create a small web service to serve the 2 functions.
  - Package it in a container
  - Publish the container to a Docker registry

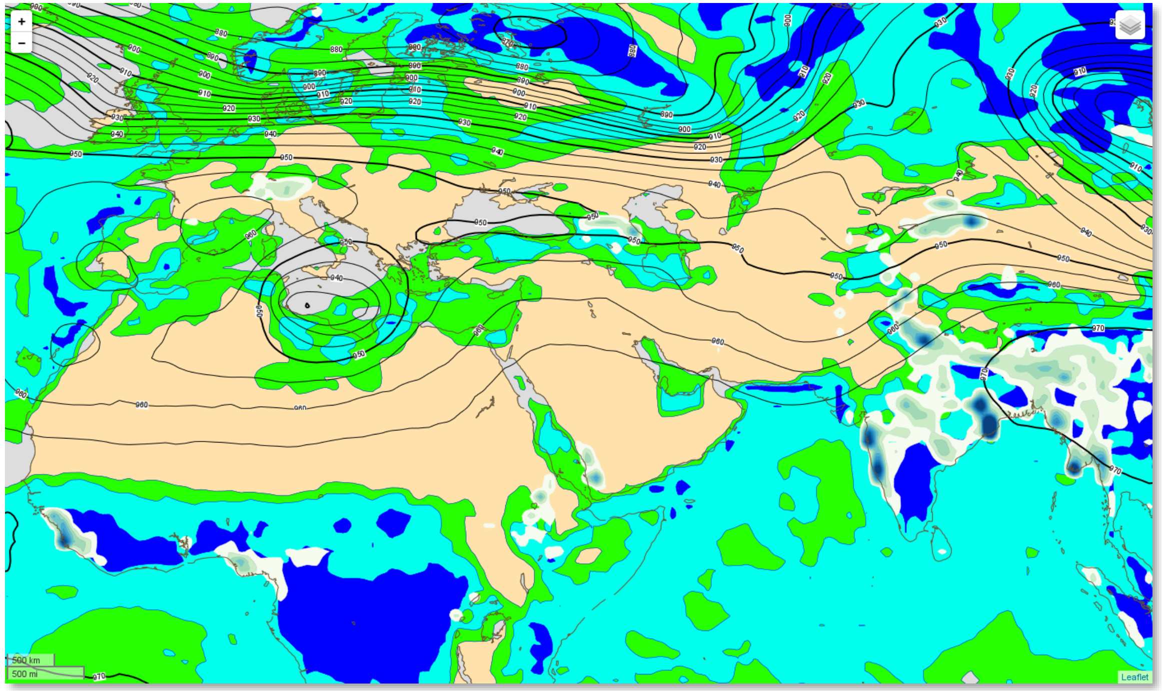


- To run:

  docker run -v /path/to/data-files:/data ecmwf/wms-server:1.4 /data

"Skinny" WMS – our way to do it

- Next steps:
  - Try more data types
  - Build more experience on GRIB and NetCDF metadata
  - Improve our support for projections.



Conclusions:

- Visualisation has always been important to understand data.

- We plan :
  - To create more rules for automatic styling
  - To keep a consistent approach on the visualisation
  - To improve our support of NetCDF
    - Automatic detection of the internal representation
    - Automatic styling
  - To improve Magics by using it in various contexts (ECMWF Data Portals, CDS toolbox)
  - To participate to python community and offer easy to use and reliable visualisation.