

Easy visualisation with Magics

The image displays a JupyterLab environment for visualizing meteorological data. On the left, a code cell shows the following Python code:

```
contour = magics.mcont(contour_automatic_setting = 'style_name',  
                      contour_style_name = One_style.value,  
                      legend = 'on')  
magics.plot(projection, data, contour, coast, legend, title)
```

The output of the code is a map showing a contour plot of divergence over Europe, with a color scale ranging from blue (negative) to red (positive). A 'Layer Management' panel is overlaid on the map, showing various data layers such as 'BASEMAP 1', 'BASEMAP 2', and several 'COMPOSITION_CO2' layers at different pressure levels (50HPA, 500HPA, 850HPA, SURFACE, 300HPA, 50HPA). The map is titled '2018-10-10T12:00:00.000Z'. To the right, a screenshot of the ECMWF website shows a search for 'magics mcont' and a result for 'Load predefined ECMWF contour setting', which includes the code snippet: `magics mcont contour_automatic_setting = 'ecmwf'`.

Milana Vuckovic - Sylvie Lamy-Thépaut



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

ECMWF @ECMWF · Sep 21
The 1-day Extreme Forecast Index (EFI) forecast showed anomalously high wind gusts for parts of Europe from #stormAli. In the south red areas are focused over land; computations can now identify when conditions will be highly unusual over land but less unusual over the sea.

Copernicus ECMWF @CopernicusECMWF · Oct 5
#Temperature highlights for September - #Copernicus #C3S. Most of Europe was warmer than average, esp Portugal & Spain. Iceland, Ireland & Scotland generally cooler than average. Globally it was around 0.4°C warmer than the average September. Read more bit.ly/2yg42LM

Search Google Maps
See travel times, traffic and nearby places

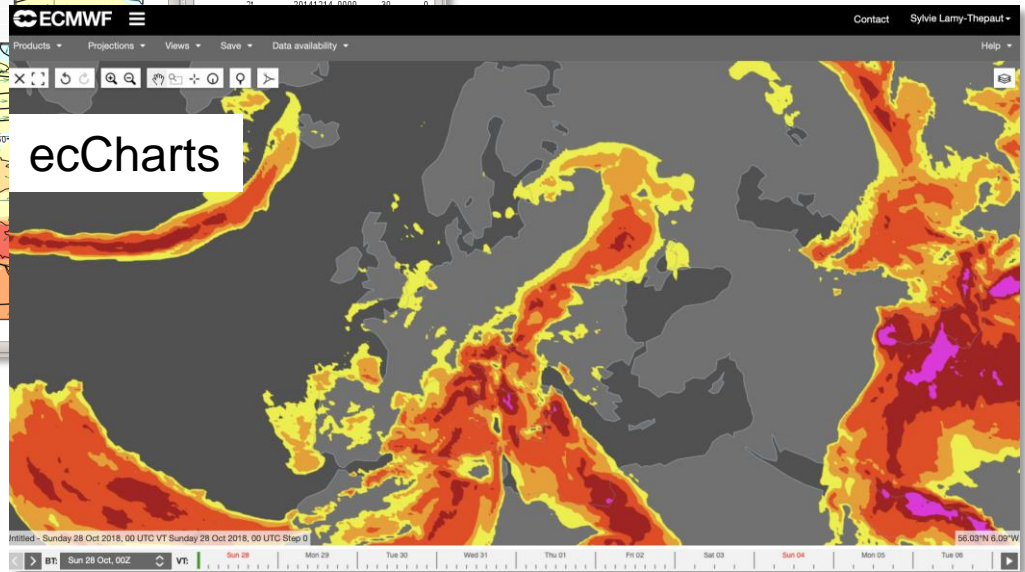
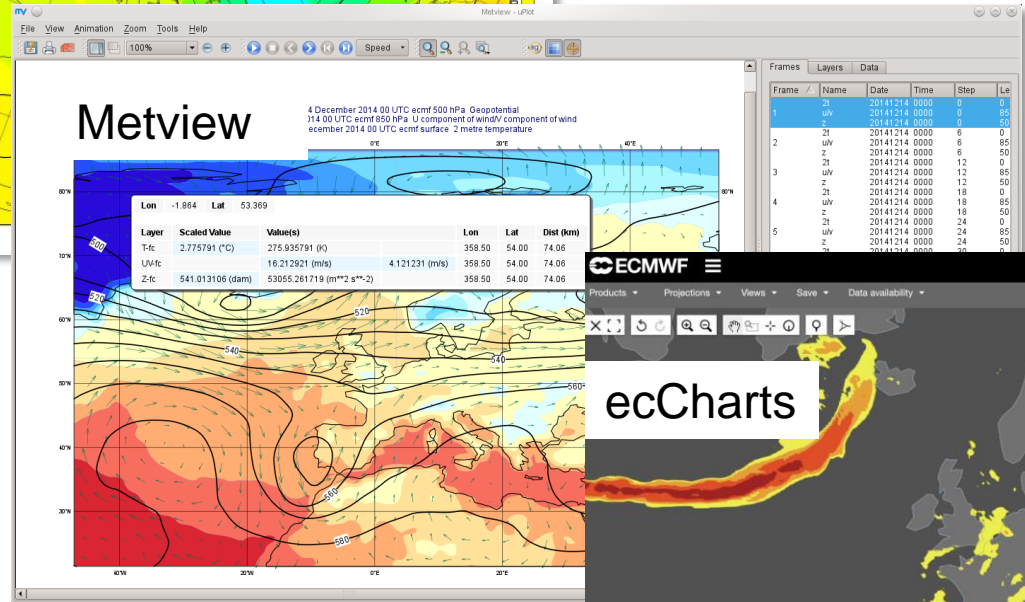
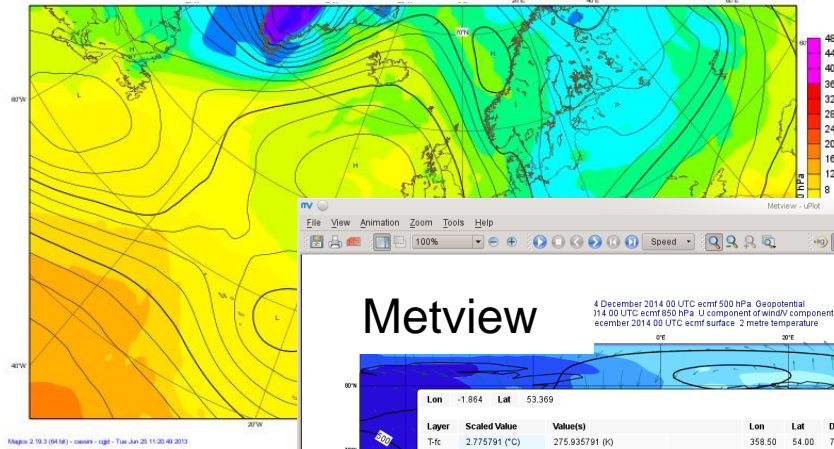
02 Plot Map

```
import cdttoolbox as ct  
  
ct.application(title='Plot Map')  
ct.input_dropdown('variable', value=[  
    '10m_temperature', '10m_u_component_of_wind',  
    '10m_v_component_of_wind',  
    'mean_sea_level_pressure', 'sea_surface_temperature',  
    ...  
)  
ct.input_dropdown('year', value=range(2008, 2018))  
ct.input_dropdown('month', value=range(1, 12))  
ct.output_figure()  
def plot_map(variable, year, month):  
    ...  
    Plot on a map the average over a given month.  
    ...  
dataset = ct.catalogue.retrieve(  
    'reanalysis-eras-single-levels',  
    {  
        'variable': variable,  
        'grid': ['3', '3'],  
        'product_type': 'reanalysis',  
        'year': [  
            '2008', '2009', '2010',  
            '2011', '2012', '2013',  
            '2014', '2015', '2016',  
            ...  
        ]  
    })
```

Mean Near Surface Air Temperature in 2008-01

How can we help ?

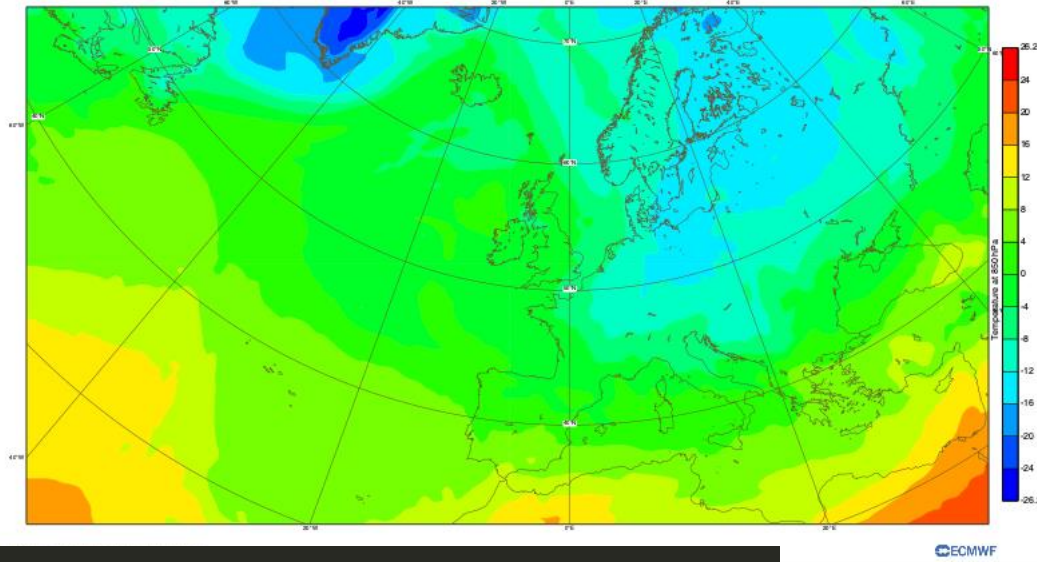
Magics



- 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
1
2 #importing Magics module
3 from Magics.macro import *
4
5 #Setting of the output file name
6 output = output(output_formats= ['png'],
7                 output_name_first_page_number= "off",
8                 output_name= "myplot")
9
10 #Setting the coordinates of the geographical area
11 europe = mmap(subpage_upper_right_longitude= 65.,
12              subpage_map_projection= "polar_stereographic",
13              subpage_map_vertical_longitude= 0.0,
14              subpage_lower_left_longitude= -37.27,
15              subpage_lower_left_latitude= 21.51,
16              subpage_upper_right_latitude= 51.28)
17
18 #Coastlines setting
19 coast = mcoast( map_grid_colour= "tan",
20               map_coastline_colour= "tan")
21
22 #import the t850 data
23 t850 = mgrib( grib_input_file_name = "./t850.grb",)
24
25 #Define the shading for t850
26 t850_contour = mcont(legend="on",
27                    contour_shade= "on",
28                    contour_hilo= "off",
29                    contour= "off",
30                    contour_label= "off",
31                    contour_shade_method= "area_fill",
32                    contour_level_selection_type= "interval",
33                    contour_interval= 4.,
34                    contour_shade_colour_method= "calculate",
35                    contour_shade_max_level_colour = "red",
36                    contour_shade_min_level_colour = "blue",
37                    contour_shade_colour_direction = "clockwise")
38
39 #To the plot
40 plot(output, europe, t850, t850_contour, coast, legend)
41
```

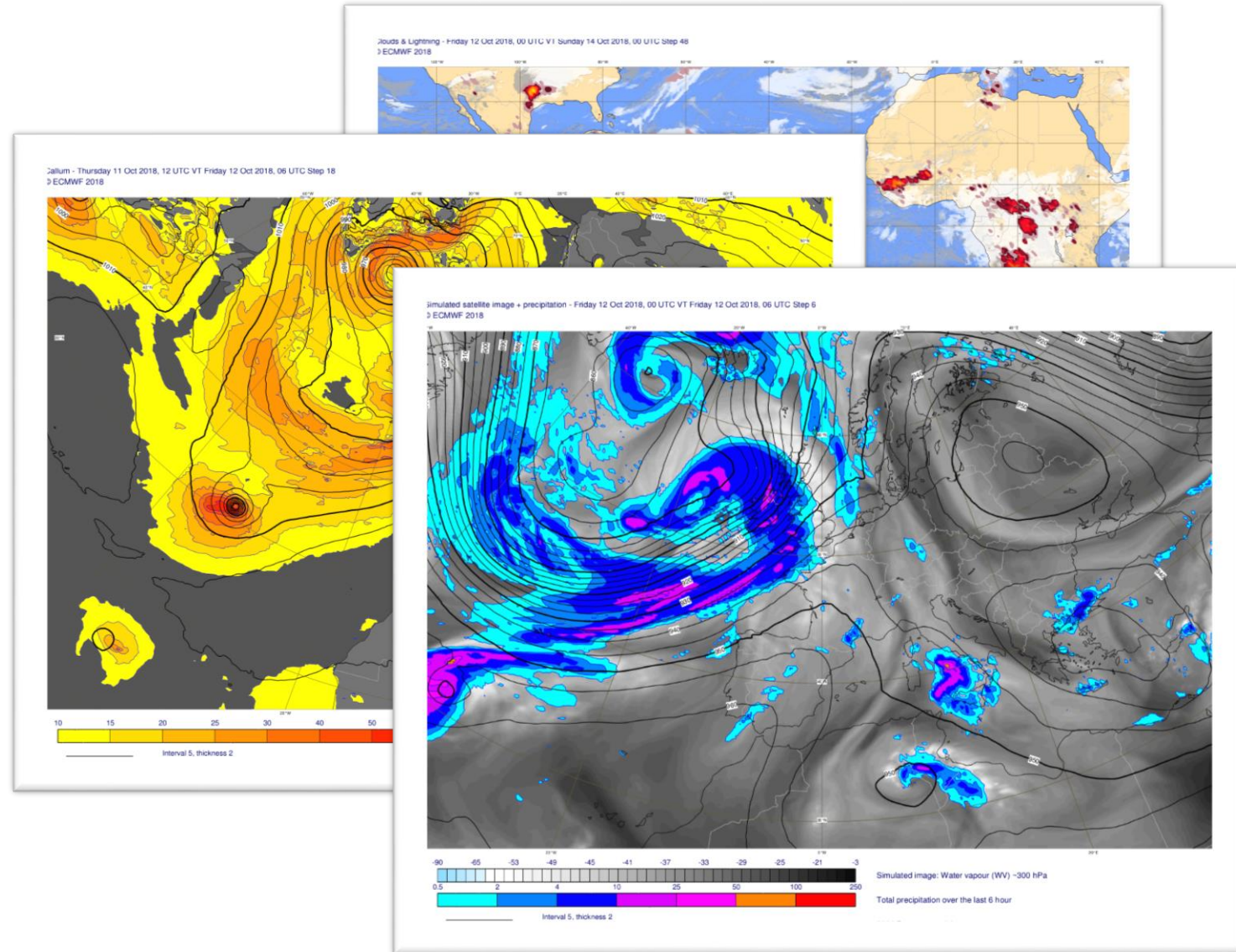


```
#Import the t850 data
t850 = mnetcdf( data = magics.mnetcdf(netcdf_filename = "2t.nc",
netcdf_value_variable = "t2m") )
```

```
#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 Z ( length=20/6b88 ) =====
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" : {
    "preferred_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_fM72t56l1st",
      "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",
    "preferred_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?

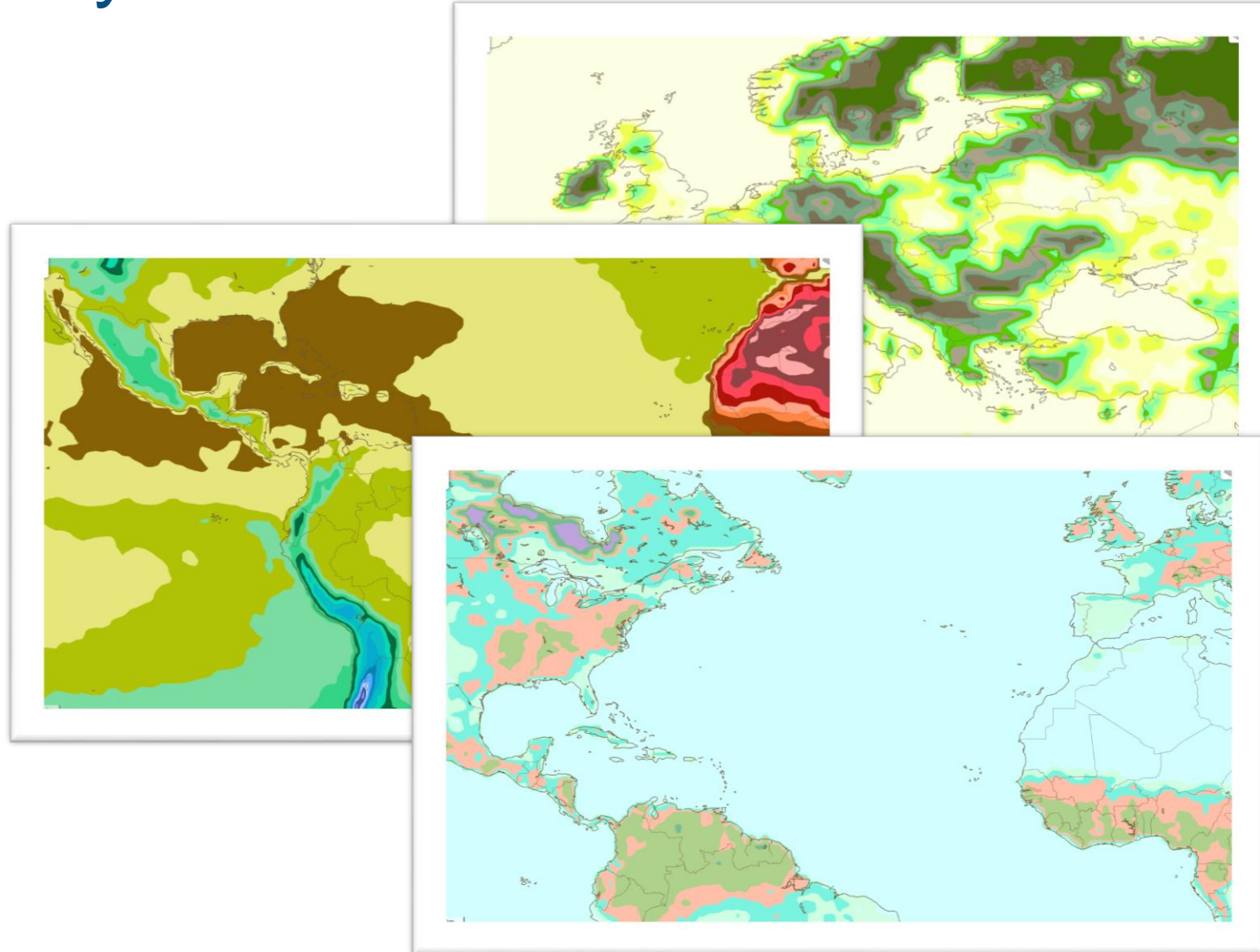
- Implemented new built in scaling in Magics, that works when units in file are different than preferred units in definition for style for parameter

But.....

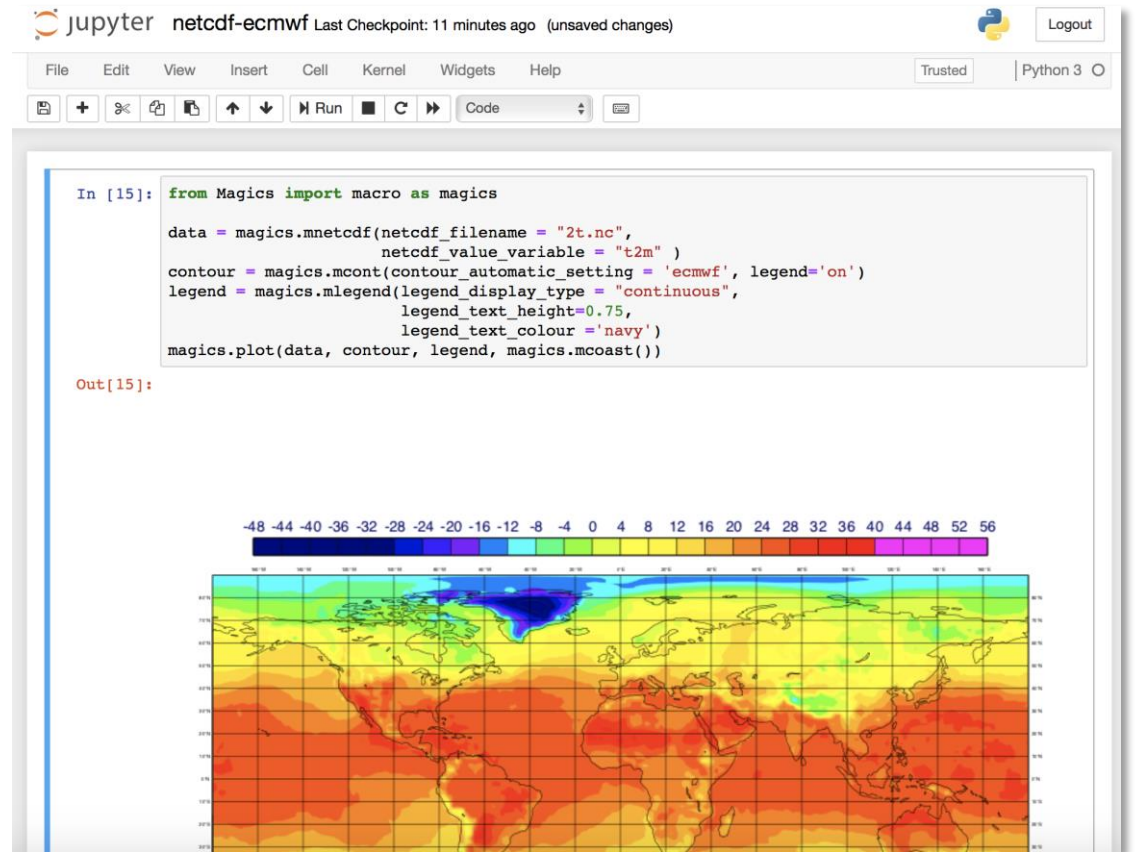
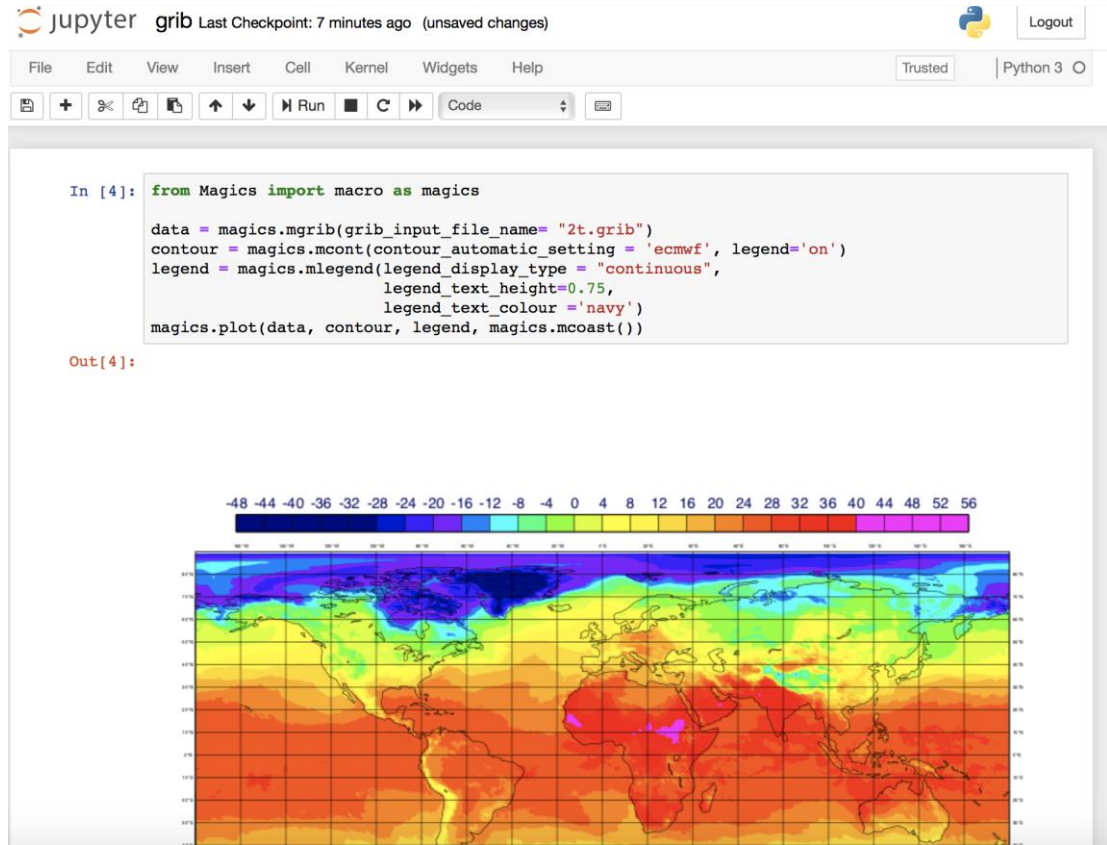
- Units are not always the same in grib and NetCDF

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

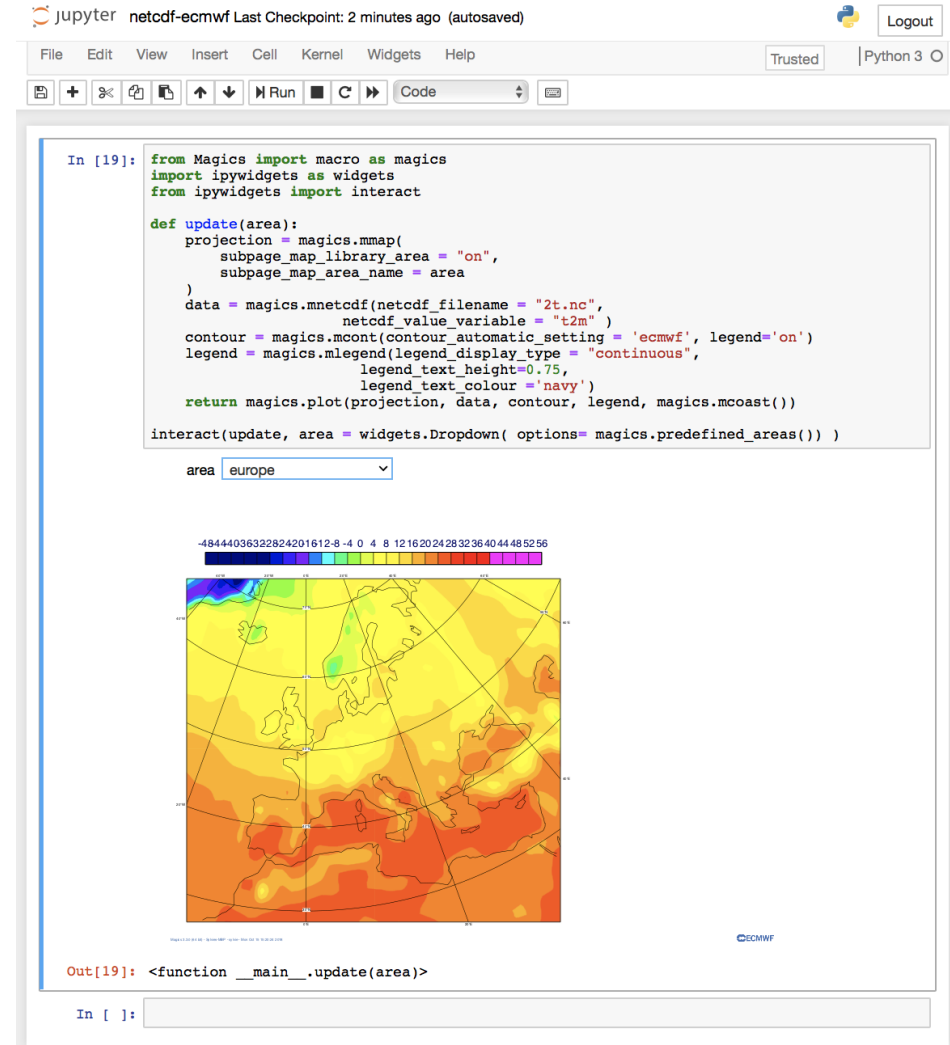


Better handling of NetCDF



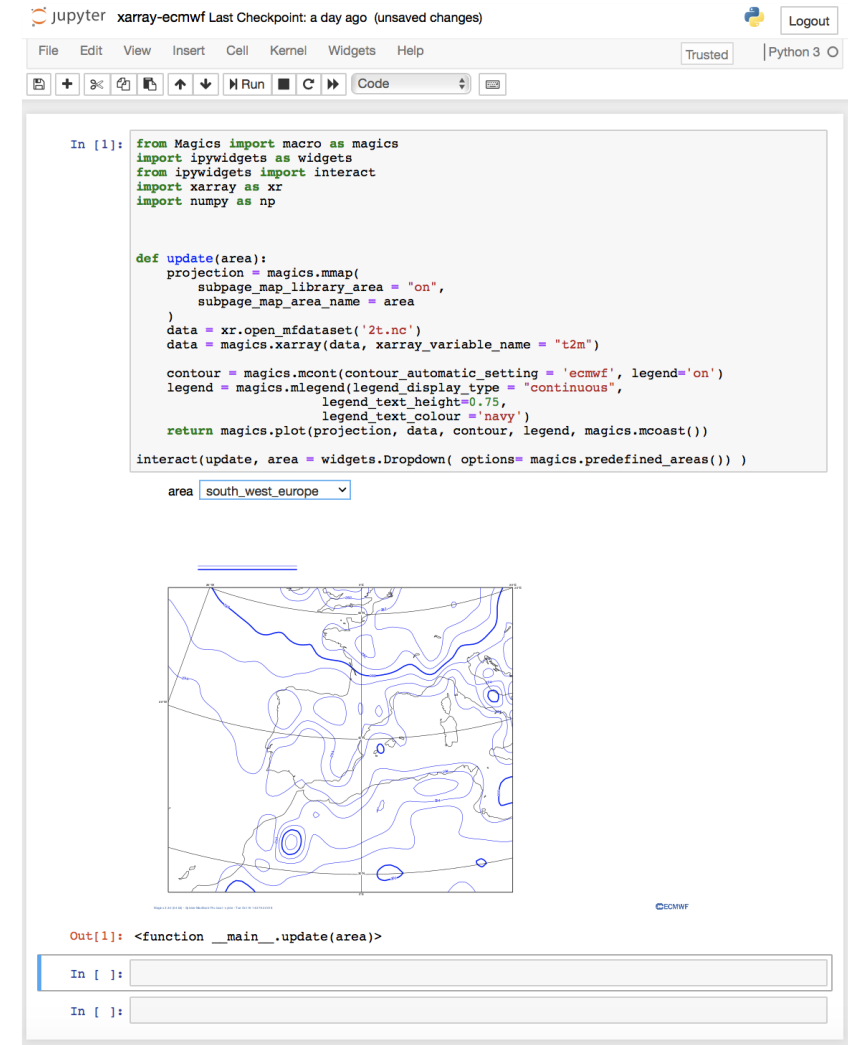
Better handling of NetCDF

- Automatic guess of the internal representation
- Automatic geo – referencing
- 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



The screenshot shows a Jupyter Notebook interface with the following content:

```
In [1]: from Magics import macro as magics
import ipywidgets as widgets
from ipywidgets import interact
import xarray as xr
import numpy as np

def update(area):
    projection = magics.mmap(
        subpage_map_library_area = "on",
        subpage_map_area_name = area
    )
    data = xr.open_mfdataset('2t.nc')
    data = magics.Xarray(data, xarray_variable_name = "t2m")
    contour = magics.mcont(contour_automatic_setting = 'ecmwf', legend='on')
    legend = magics.mlegend(legend_display_type = "continuous",
        legend_text_height=0.75,
        legend_text_colour = 'navy')
    return magics.plot(projection, data, contour, legend, magics.mcoast())

interact(update, area = widgets Dropdown( options= magics.predefined_areas() ) )
```

The notebook displays a contour plot of a weather dataset (likely temperature) over a geographical region. The plot shows blue contour lines on a white background, with a legend in the bottom right corner. The legend is titled "continuous" and has a "navy" color. The plot is titled "south_west_europe".

Out[1]: <function __main__.update(area)>

In []:

In []:

How to facilitate the use of Magics?

magics 0.9.2.dev0
pip install magics
Latest version
Last released: Sep 17, 2018

PUBLIC REPOSITORY
ecmwf/jupyter-notebook
Last pushed: a day ago

Repo Info Tags

Short Description
Jupyter Notebook image with ECMWF software installed

Docker Pull Command
docker pull ecmwf/jupyter-notebook

Full Description
Jupyter Notebook image with ECMWF software installed
This is a Docker container image based on [jupyter/base-notebook:6c85e4b43a26](#), including the following ECMWF software:

- Magics++ 3.3.0.1
- ecCodes 2.9.0
- Metview 5.2.1

Owner
ecmwf

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

ecmwf / notebook-examples
Unwatch 3 Star 0 Fork 1
Code Pull requests 0 Wiki Insights
Branch: master - Create new file Upload files Find file History
notebook-examples / visualisation /
sylvielamythepaut Update README.md Latest commit 0806a28 26 seconds ago
basic Adding netcdf example 19 minutes ago
gallery Simple shading thumbnail 2 days ago
images Cleaning 2 minutes ago
tutorials Contours shading notebook update 2 days ago
README.md Update README.md 26 seconds ago
magics.ipynb Adding visualisation examples 2 days ago
README.md
ECMWF Introduction to Magics
Magics is the latest generation of the ECMWF's meteorological plotting software. It offers an easy way to visualise data coded in meteorological formats such as GRIB, NetCDF and BUFR.
Its simple python interface allows users to quickly setup projection and geographical areas and apply easily a defined visualisation.
This gallery of tutorials and examples will help discover its functionality

- Easy visualisation of [GRIB data](#)
- Easy visualisation of [NetCDF data](#)
- Easy visualisation of xarray data

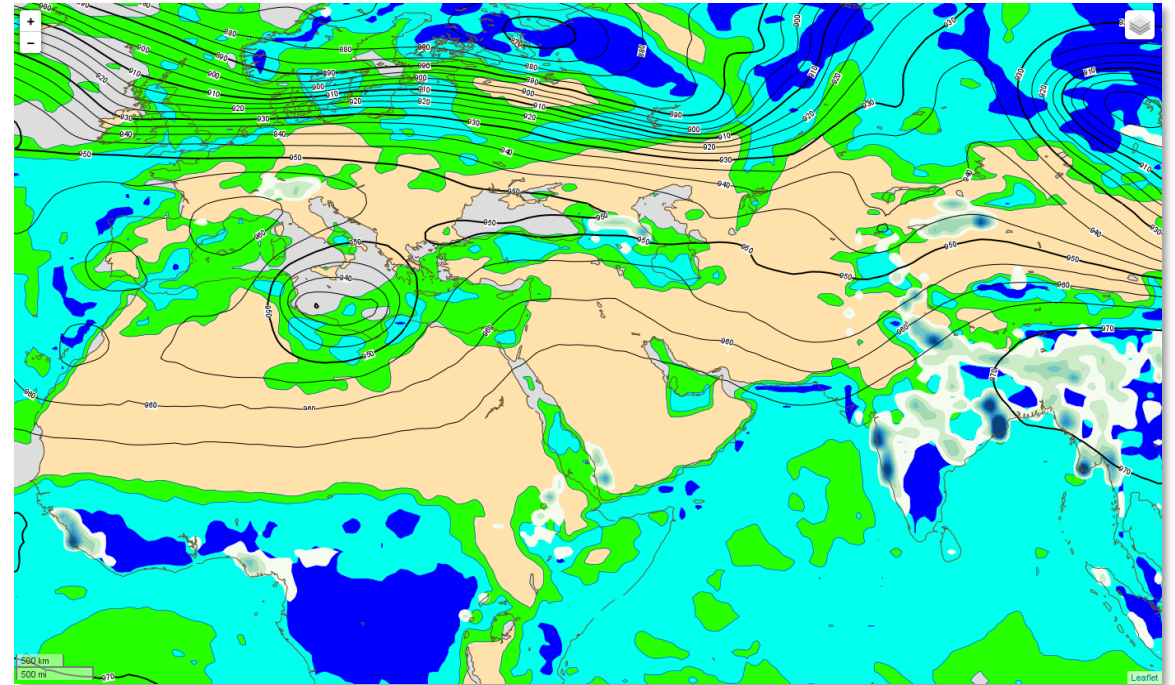
Tutorials

Setting the geographical view Customise coastlines Introduction to contouring

Gallery

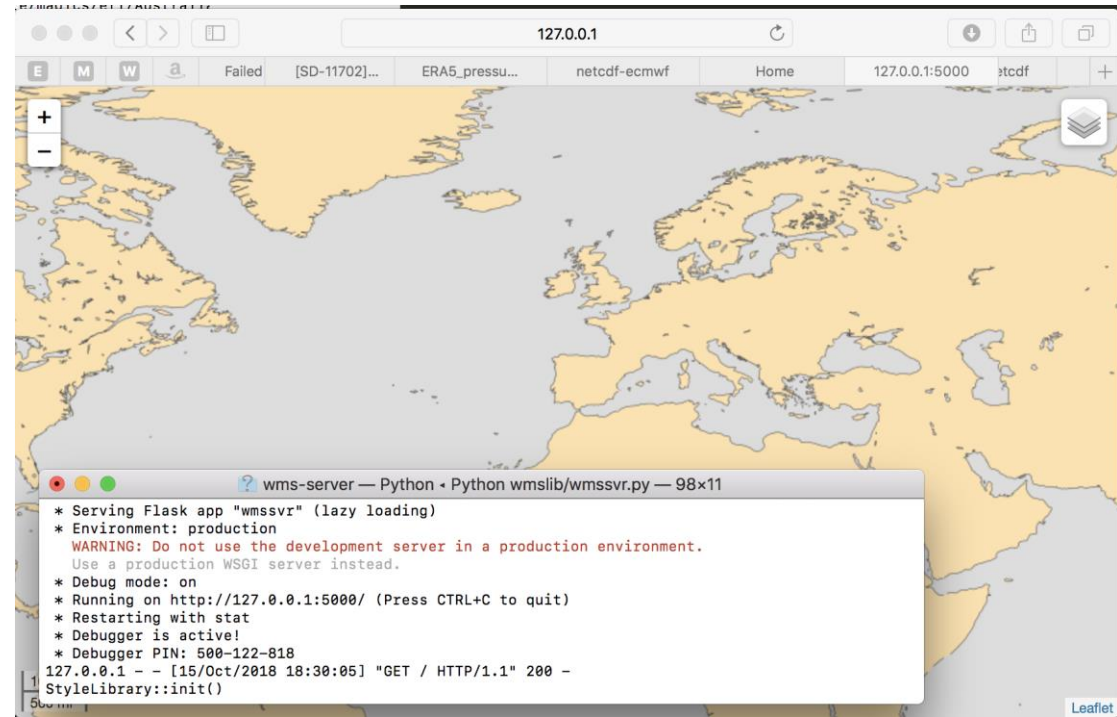
Another use of the automatic styling: The skinny WMS

- 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

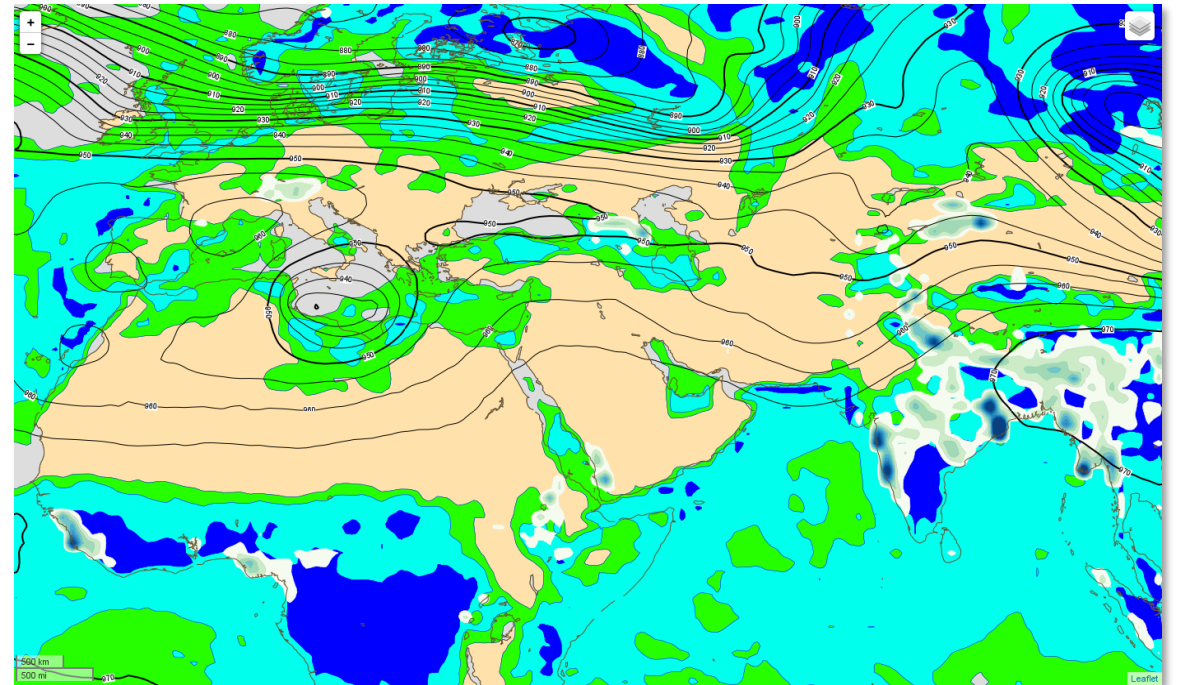
- The implementation :
 - Create a small web service to serve the 2 functions.
 - Package it in a container
 - Publish the container to a Docker registry



- 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.

