

Jupyter... WMS ...

All we need is an easy way of visualising meteorological data ...

The image illustrates a workflow for visualizing meteorological data. On the left, a Jupyter Notebook interface shows a code cell with the following Python code:

```
In [26]: 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 of Europe showing a color-coded divergence field. In the center, a larger map of Europe displays pressure contours (isobars) over a satellite-style background. A 'LAYER MANAGEMENT' panel is overlaid on the right side of this map, showing a list of layers with toggle switches:

- Base Maps: BASEMAP 1 (checked), BASEMAP 2
- Data Maps: COMPOSITION_CO2_50HPA, COMPOSITION_CO2_500HPA, COMPOSITION_CO2_850HPA, COMPOSITION_CO2_SURFACE, COMPOSITION_CO_300HPA, COMPOSITION_CO_50HPA

On the right, a browser window displays the ECMWF website. It shows a search bar, navigation links, and a section titled 'Load predefined ECMWF contour setting:'. Below this, there is a code snippet:

```
contour = magics.mcont(contour_automatic_setting = 'ecmwf')  
magics.plot(projection, data, contour, legend, coast, title)
```

The browser also displays a map of Europe with a temperature field, titled 'Monday 02 January 2017 12 UTC: 2m surface 2-metre temperature'.

Milana Vuckovic - Sylvie Lamy-Thépaut –
Pierre Vernier – Carlos Valiente – Cihan Sahin

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

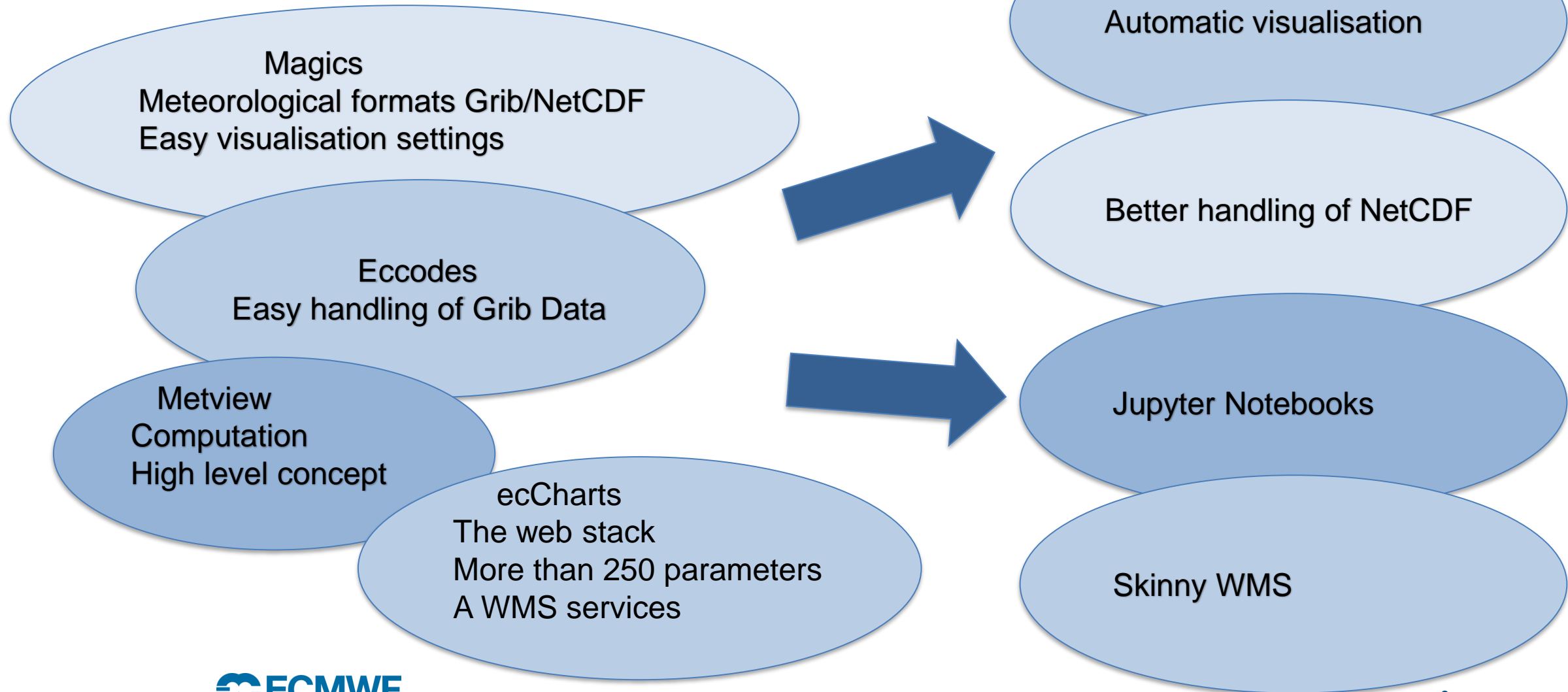
Search Google Maps
See travel times, traffic and nearby places

02 Plot Map

```
import cdtoolbox 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',  
            ...  
        ]  
    })
```

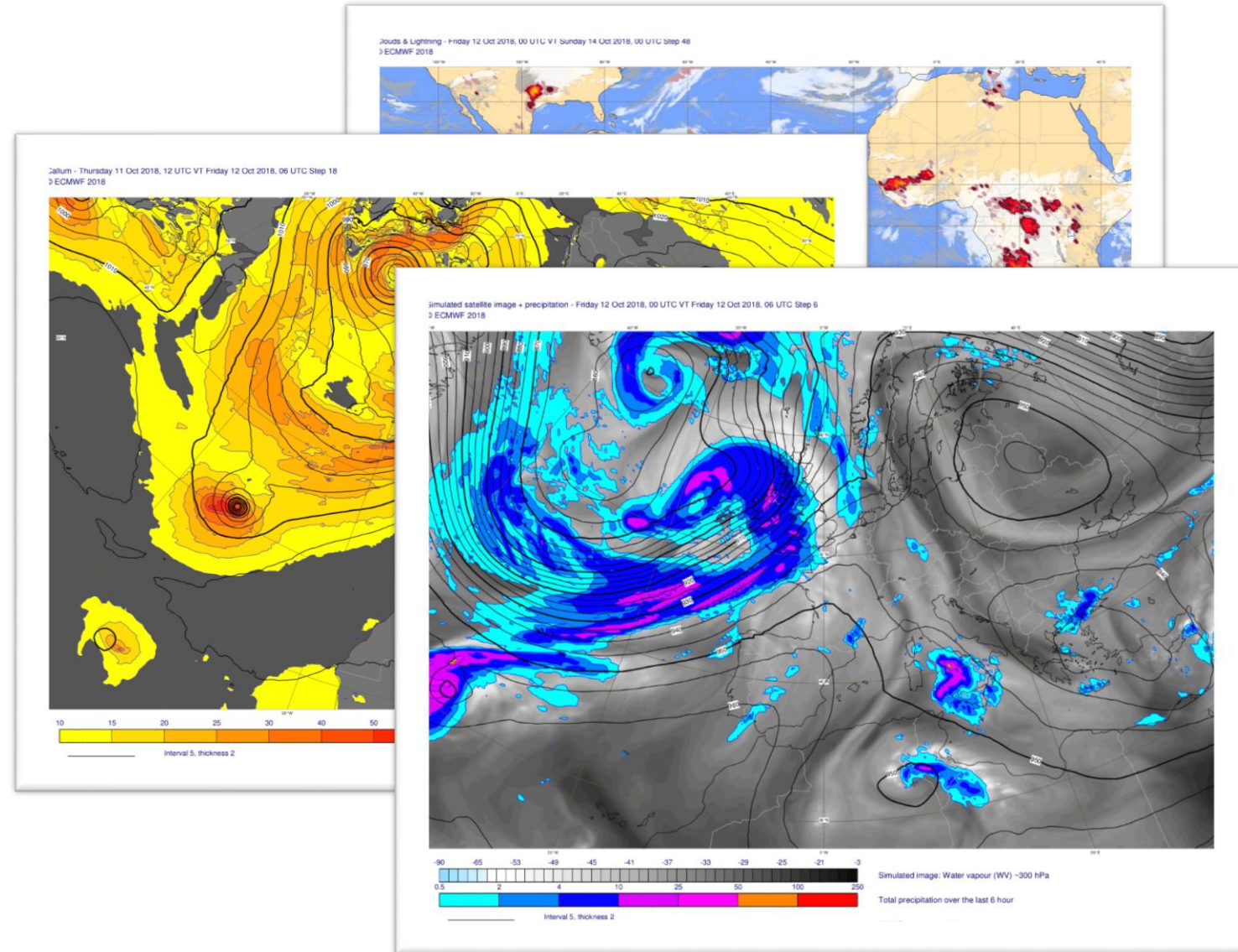
Mean Near Surface Air Temperature in 2008-01

How can we help ?



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

Temperature on different pressure levels example

```

{
  "match" : {
    "eccharts_layer" : "t500",
    "preferred_units" : "C",
    "set" : [
      {
        "levelist" : [
          "500",
          "450",
          "550"
        ],
        "paramId" : "130",
        "shortName" : "t",
        "levtype" : "pl"
      }
    ],
    "style" : "sh_all_fm52t48i4",
    "styles" : [
      "sh_all_fm52t48i4",
      "sh_all_fm64t52i4",
      "ct_red_i2_dash",
      "sh_all_fm52t48i4_light",
      "sh_gry_fm72t56lst",
      "sh_all_fm80t56i4_v2",
      "sh_all_fm50t58i2",
      "ct_red_i4_t3"
    ]
  }
}

```

← GRIB →

```

{
  "match" : {
    "eccharts_layer" : "t250",
    "preferred_units" : "C",
    "set" : [
      {
        "levelist" : [
          "250",
          "300",
          "350",
          "400"
        ],
        "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"
    ]
  }
}

```

GRIB →

```

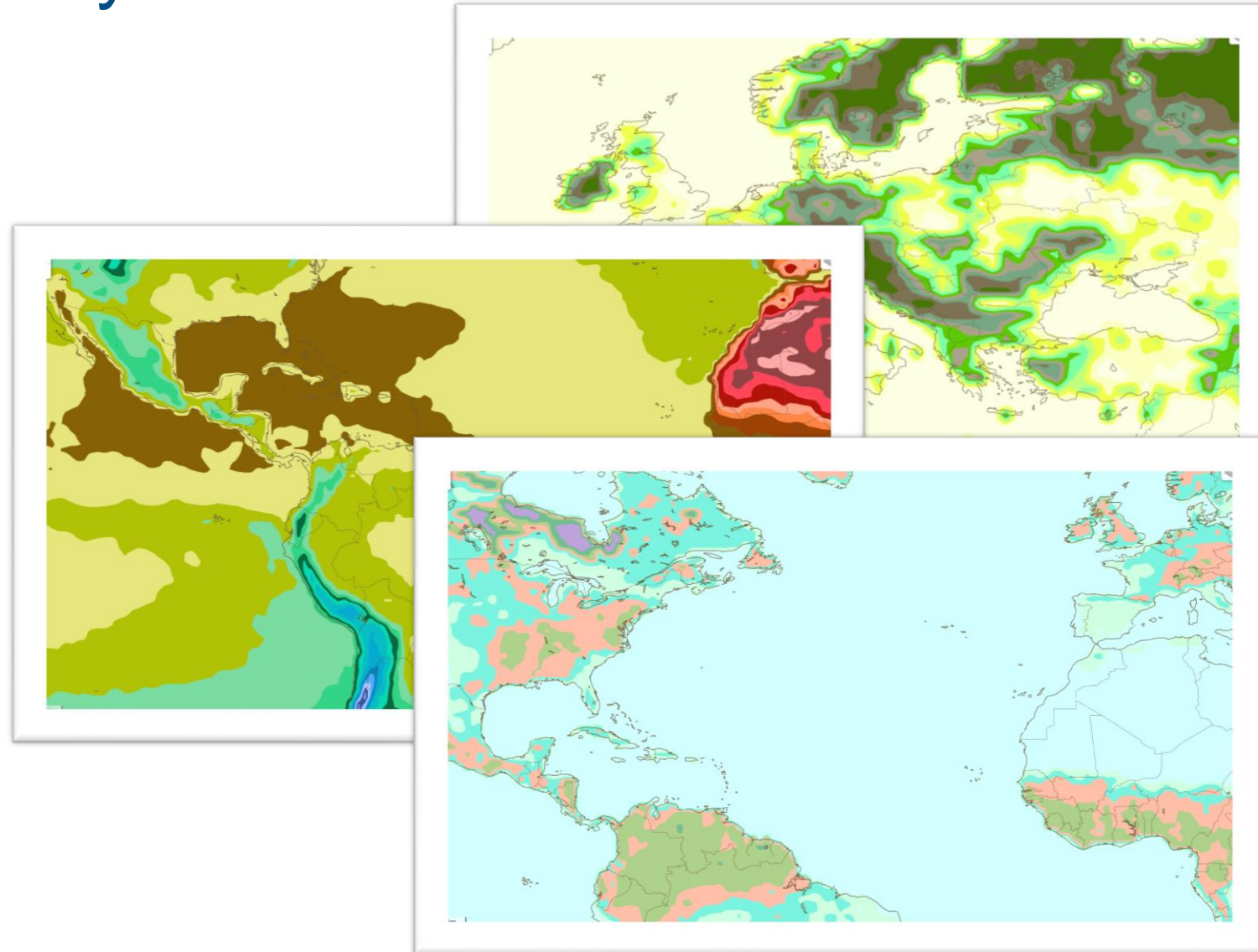
{
  "match" : {
    "eccharts_layer" : [
      "t925",
      "t850_field",
      "t800",
      "t700",
      "t600"
    ],
    "preferred_units" : "C",
    "set" : [
      {
        "levelist" : [
          "900",
          "925",
          "950",
          "975",
          "1000",
          "850",
          "825",
          "800",
          "600",
          "650",
          "700",
          "750",
          "775",
          "875"
        ],
        "paramId" : "130",
        "shortName" : "t",
        "levtype" : "pl"
      }
    ],
    "long_name" : "Temperature",
    "standard_name" : "air_temperature"
  },
  "style" : "sh_all_fm52t48i4",
  "styles" : [
    "sh_all_fm52t48i4",
    "sh_all_fm64t52i4",
    "ct_red_i2_dash",
    "sh_all_fm52t48i4_light",
    "sh_gry_fm72t56lst",
    "ct_red_i4_t3",
    "sh_all_fm80t56i4_v2",
    "sh_all_fm50t58i2",
    "transparent_zero_blue"
  ]
}

```

NetCDF →

A solid framework for styles

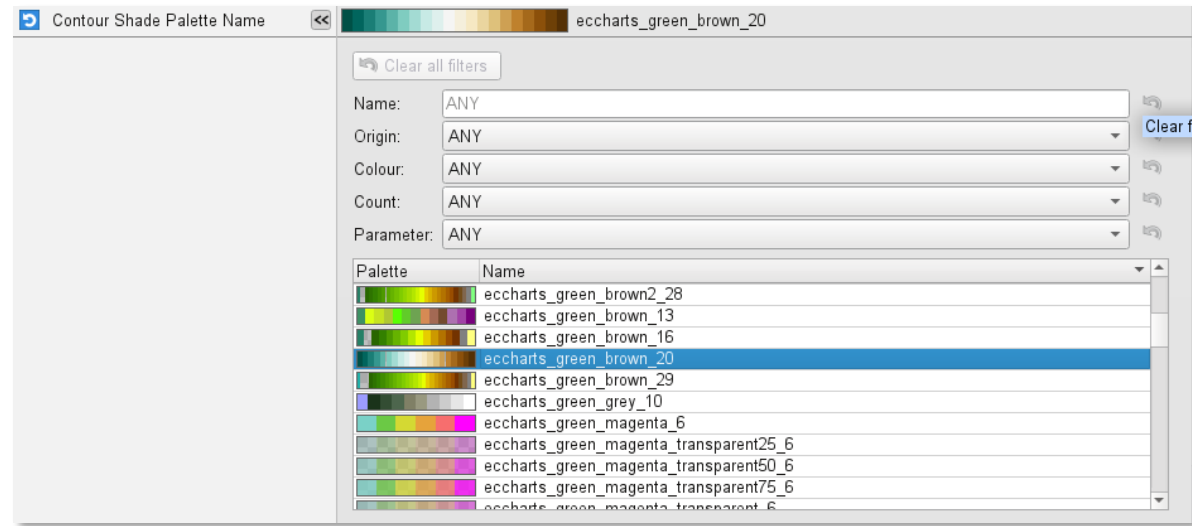
- There are many meteorological parameters not present in ecCharts
- We started designing styles for most important ones
- Introduction of predefined palettes



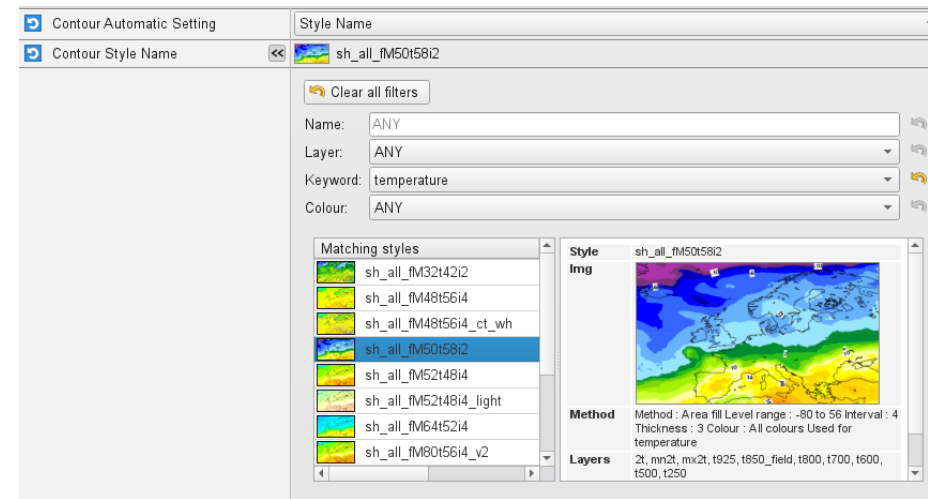
Building on top of the framework → Metview



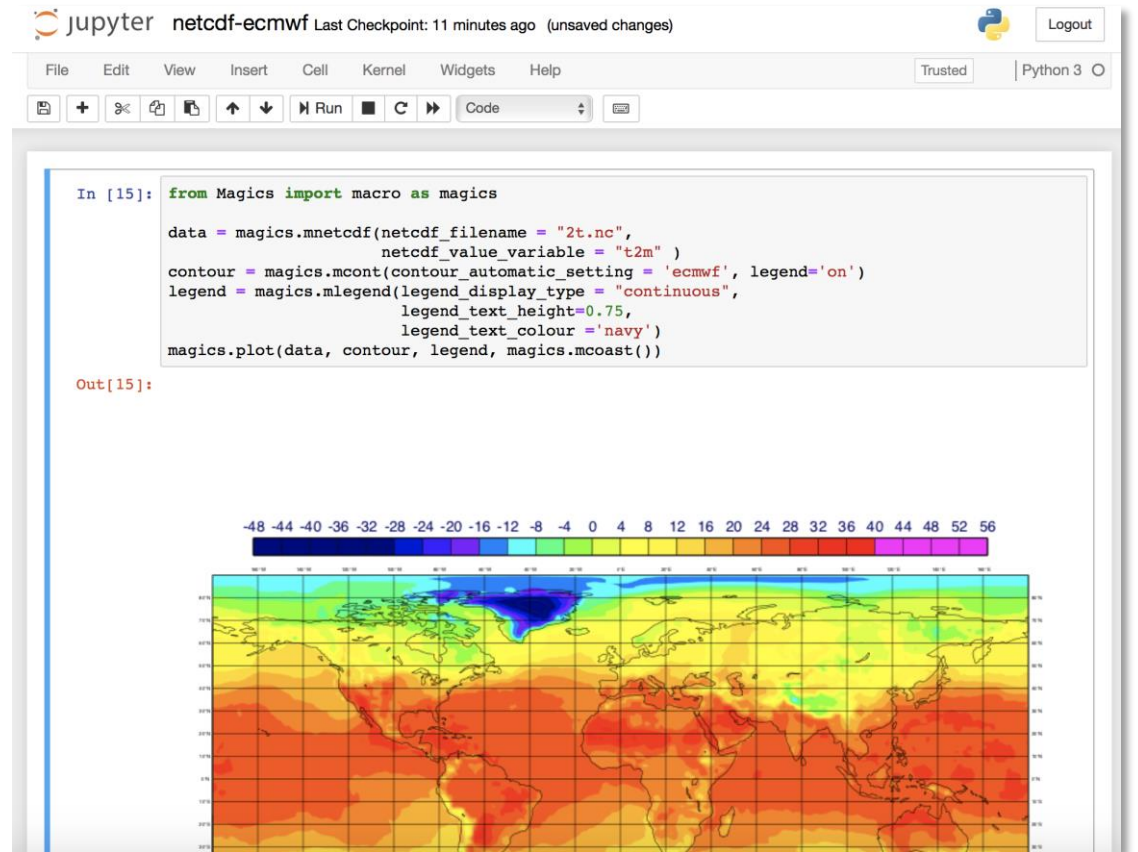
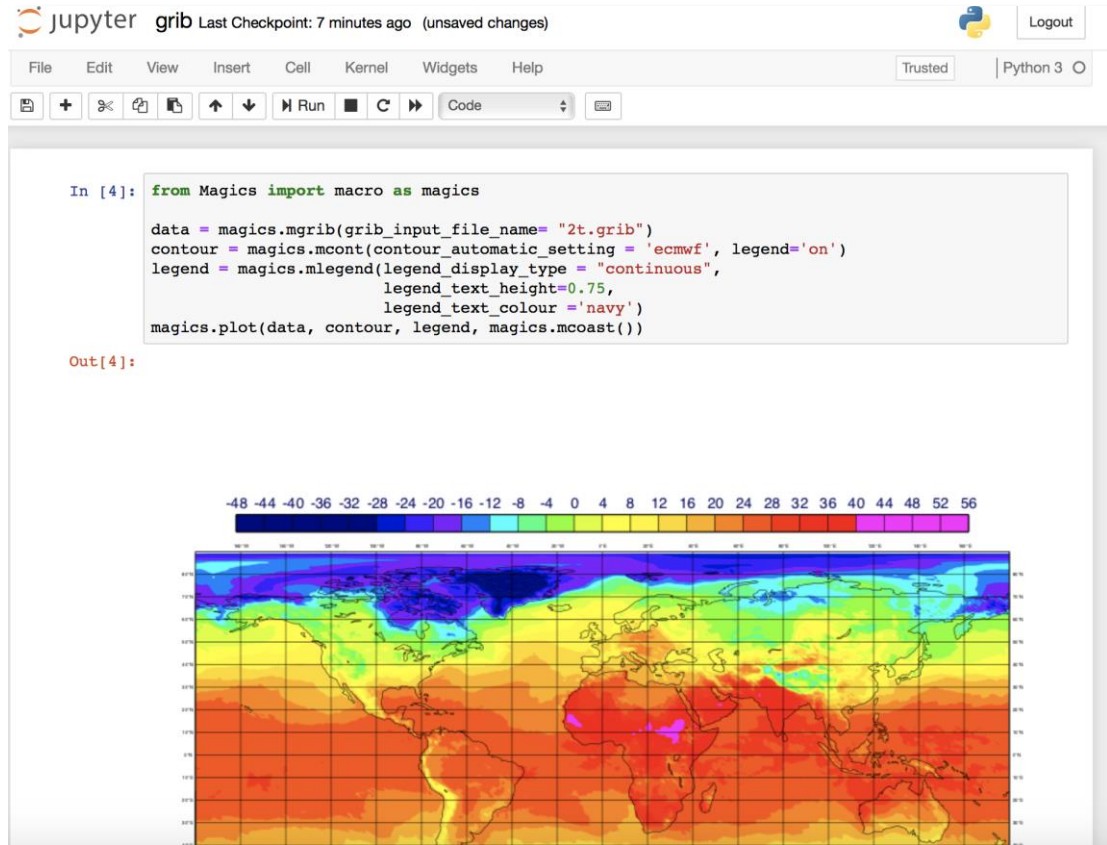
Palette selector



Style selector

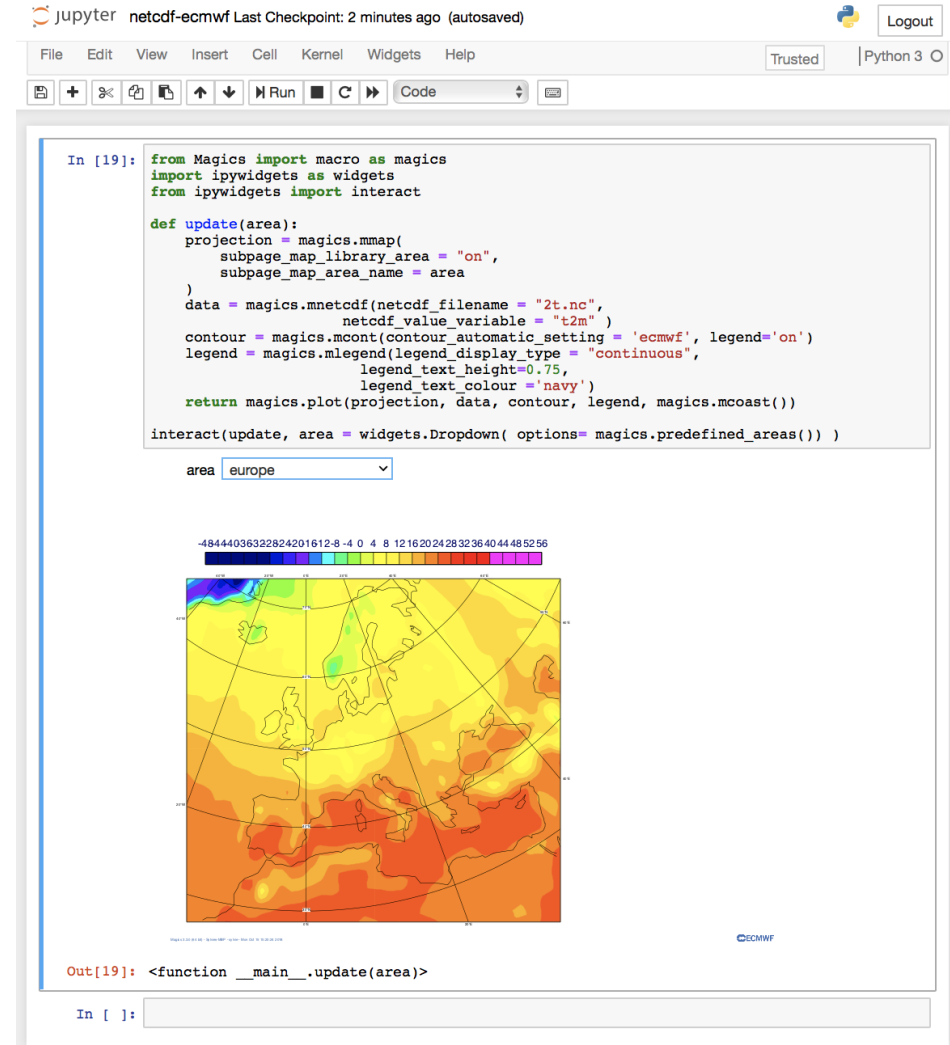


Better handling of NetCDF



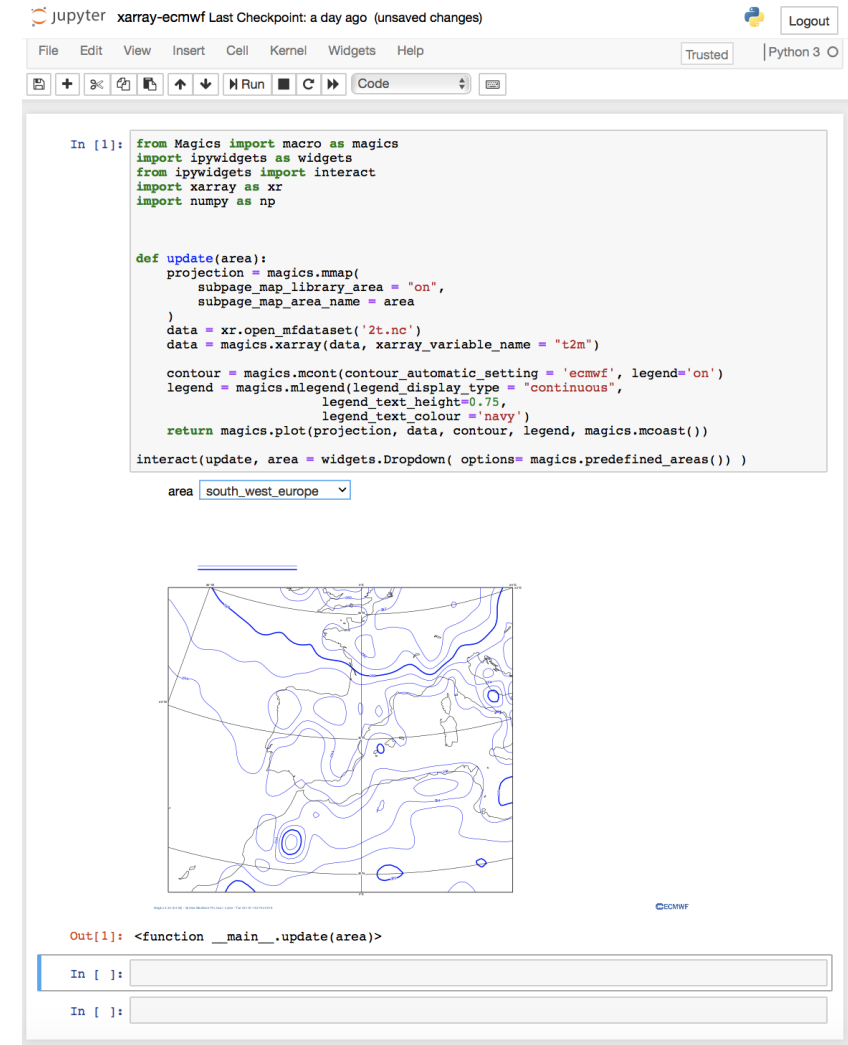
Better handling of NetCDF

- 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 could be used to setup an automatic visualization



Jupyter xarray-ecmwf Last Checkpoint: a day ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
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() ) )

area south_west_europe
```

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

In []:

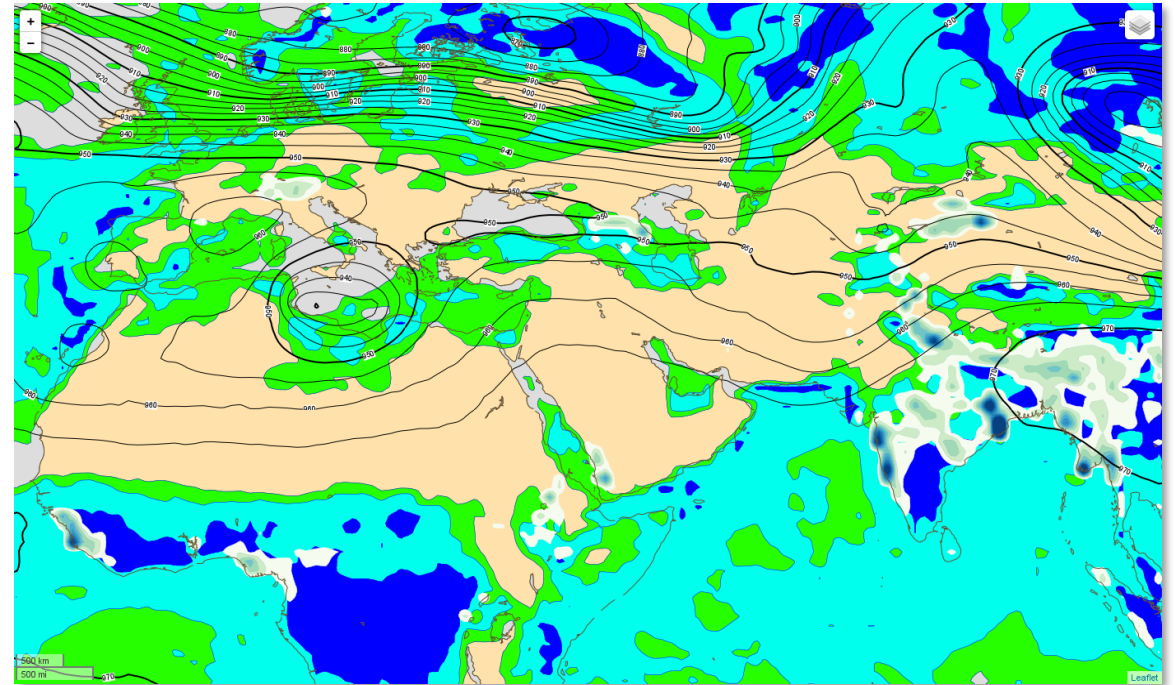
In []:

The next steps -> creating a WMS

- WMS is a popular service
 - The GetCapabilities to describe
 - The data: their availability, and available options for visualisation
 - The supported projections.
 - GetMap to get the selected data as graphical product with the selected style/projection
 - GetFeatureInfo to trigger further interactions on a geographical point.
- Many WMS clients out there, so users can keep working with their favorite tool (Open Layers, Leaflet, Qgis, Metview)
 - Most of them offer nice to way to browse the data to display, with all the common zoom and pan.
 - Tiling for performance and cachability

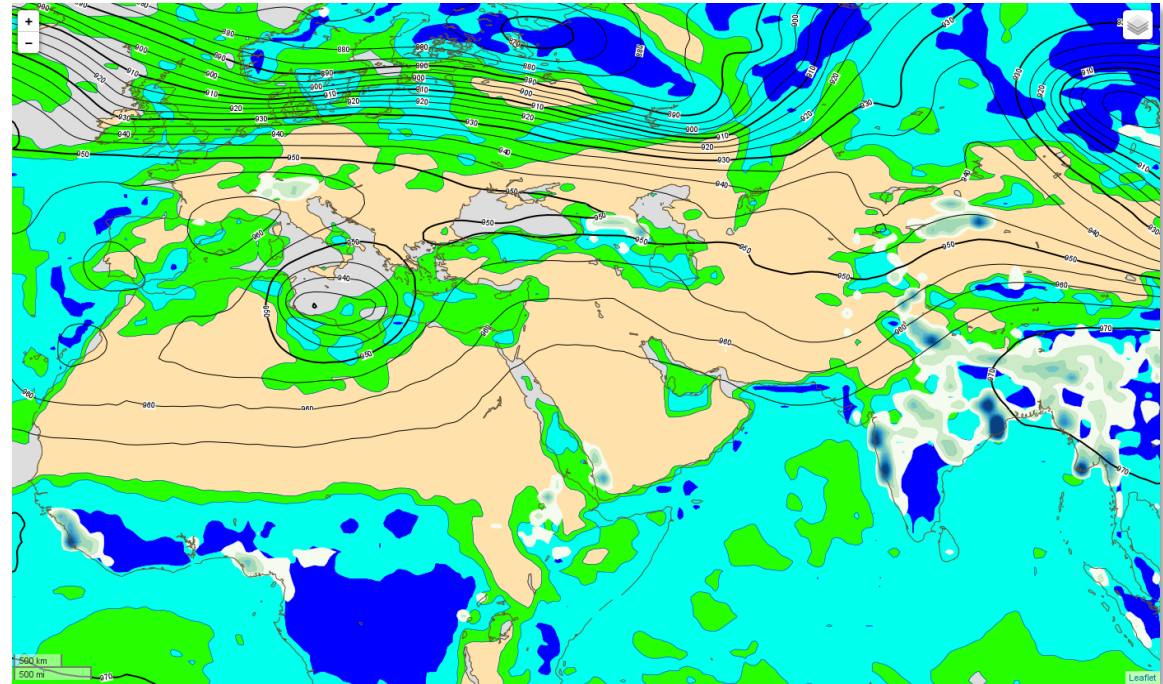
"Skinny" WMS – our way to do it

- 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

- A small demo:

The image shows a web browser window displaying a map of Europe. The browser's address bar shows the URL `127.0.0.1`. The map is a WMS service showing a background layer and several other layers. A legend on the right side of the map lists the following layers:

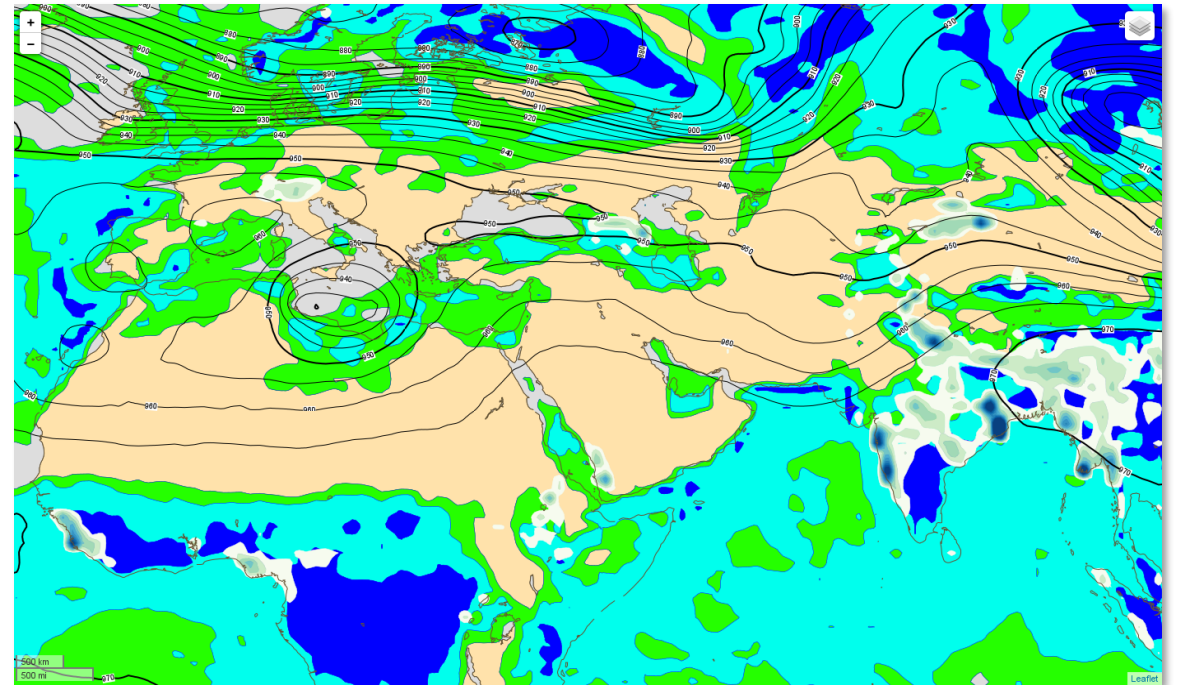
- background
- 10fg
- kx
- mx2t
- mwp
- sf
- foreground
- grid
- boundaries

The map shows a yellow background with blue and purple contour lines representing data. A terminal window is open in the foreground, showing the following output:

```
wms-server — Python • Python wmslib/wmssvr.py — 98x11
* Serving Flask app "wmssvr" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 500-122-818
127.0.0.1 -- [15/Oct/2018 18:30:05] "GET / HTTP/1.1" 200 -
StyleLibrary::init()
```

"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 Skinny WMS by using it in various contexts (ECMWF Data Portals, CDS toolbox)
 - To participate to python community and offer easy to use and reliable visualisation.

