psy-maps Documentation

Release 1.1.0

Philipp Sommer

Apr 09, 2018

Contents

1 Documentation **3**

1.1 Installation . . . 3
1.1.1 How to install . . . 3
Installation using conda . . . 3
Installation using pip . . . 3
1.1.2 Running the tests . . . 3
1.2 psyplot plot methods . . . 4
1.2.1 psyplot.project.plot.mapplot . . . 4
1.2.2 psyplot.project.plot.mapvector . . . 5
1.2.3 psyplot.project.plot.mapcombined . . . 6
1.3 Example Gallery . . . 7
1.3.1 Visualizing circumpolar data . . . 7
Note . . . 10
References . . . 11
1.3.2 Basic data visualization on a map . . . 11
Visualizing scalar fields . . . 11
Visualizing vector data . . . 15
Visualizing combined scalar and vector data . . . 17
Summary . . . 18
1.3.3 Visualizing unstructured data . . . 20
1.4 API Reference . . . 22
1.4.1 Submodules . . . 22
psy_maps.boxes module . . . 22
psy_maps.plotters module . . . 22
Possible types . . . 23
Possible types . . . 27
Notes . . . 27
Possible types . . . 28
Possible types . . . 32
Possible types . . . 32
Possible types . . . 32
Possible types . . . 33
Possible types . . . 33
Possible types . . . 34
Possible types . . . 34
Notes . . . 34

Possible types . . . 35
Notes . . . 35
Possible types . . . 35
Possible types . . . 36
Possible types . . . 36
Possible types . . . 36
Possible types . . . 36
Possible types . . . 37
Possible types . . . 37
Possible types . . . 37
Notes . . . 37
Possible types . . . 38
Possible types . . . 38
Notes . . . 38
Possible types . . . 38
Possible types . . . 39
Possible types . . . 39
Possible types . . . 40
Possible types . . . 40
Possible types . . . 40
Possible types . . . 41
Possible types . . . 42
Possible types . . . 42
Possible types . . . 42
Possible types . . . 42
Possible types . . . 43
Notes . . . 43
Possible types . . . 43
Possible types . . . 44
Possible types . . . 44
Possible types . . . 44
Possible types . . . 45
Notes . . . 45
Possible types . . . 46
Possible types . . . 46
Possible types . . . 46
Possible types . . . 47
Possible types . . . 47
Possible types . . . 47
Possible types . . . 47
Possible types . . . 48
Possible types . . . 48
Possible types . . . 49
Possible types . . . 49
Possible types . . . 49
Possible types . . . 50
Possible types . . . 50
Possible types . . . 50
Possible types . . . 51
Possible types . . . 51
Possible types . . . 51
Possible types . . . 52
Possible types . . . 52
Possible types . . . 53

Possible types . . . 53
Possible types . . . 53
Possible types . . . 53
Possible types . . . 54
Possible types . . . 54
Possible types . . . 54
Possible types . . . 54
Possible types . . . 55
Possible types . . . 55
Possible types . . . 55
Possible types . . . 55
Possible types . . . 56
Possible types . . . 59
Possible types . . . 59
Possible types . . . 59
Possible types . . . 60
Notes . . . 60
Possible types . . . 60
Possible types . . . 61
Possible types . . . 61
Possible types . . . 61
Possible types . . . 61
Possible types . . . 62
Notes . . . 62
Possible types . . . 62
Possible types . . . 62
Notes . . . 63
Possible types . . . 63
Possible types . . . 63
Possible types . . . 64
Possible types . . . 64
Possible types . . . 65
Possible types . . . 65
Possible types . . . 66
Possible types . . . 67
Possible types . . . 67
Possible types . . . 67
Possible types . . . 67
Possible types . . . 68
Possible types . . . 68
Possible types . . . 68
Possible types . . . 69
Possible types . . . 69
Possible types . . . 69
Possible types . . . 69
Possible types . . . 70
Notes . . . 70
Possible types . . . 70
Possible types . . . 70
Possible types . . . 71
Possible types . . . 71
Possible types . . . 72
Notes . . . 72
Possible types . . . 72

Possible types . . . 73
Possible types . . . 73
Possible types . . . 73
Possible types . . . 73
Possible types . . . 74
Possible types . . . 74
Possible types . . . 74
Possible types . . . 74
Possible types . . . 74
Possible types . . . 75
Possible types . . . 75
Possible types . . . 76
Possible types . . . 76
Possible types . . . 78
Possible types . . . 79
Possible types . . . 80
Possible types . . . 82
Notes . . . 83
Possible types . . . 83
Possible types . . . 84
Notes . . . 85
Notes . . . 86
Notes . . . 86
Possible types . . . 87
Possible types . . . 88
Possible types . . . 88
Notes . . . 88
Possible types . . . 89
Possible types . . . 93
Notes . . . 93
Possible types . . . 93
Possible types . . . 93
Possible types . . . 94
Possible types . . . 94
Possible types . . . 94
Possible types . . . 94
Notes . . . 95
Possible types . . . 95
Possible types . . . 95
Notes . . . 95
Possible types . . . 96
Possible types . . . 96
Possible types . . . 96
Possible types . . . 97
Possible types . . . 97
Possible types . . . 98
Possible types . . . 99
Possible types . . . 99
Possible types . . . 99
Possible types . . . 99
Possible types . . . 100
Possible types . . . 101
Possible types . . . 101
Possible types . . . 101

Possible types . . . 101
Possible types . . . 102
Possible types . . . 102
Possible types . . . 102
Possible types . . . 102
Possible types . . . 103
Possible types . . . 104
Possible types . . . 104
Possible types . . . 105
Possible types . . . 107
Possible types . . . 108
Possible types . . . 109
Possible types . . . 110
Possible types . . . 114
Possible types . . . 114
Possible types . . . 114
Possible types . . . 115
Possible types . . . 115
Possible types . . . 116
Possible types . . . 116
Possible types . . . 116
Possible types . . . 117
Possible types . . . 117
Possible types . . . 117
Possible types . . . 117
Possible types . . . 117
Notes . . . 118
Possible types . . . 118
Notes . . . 118
Possible types . . . 118
Possible types . . . 119
Possible types . . . 119
Possible types . . . 119
Possible types . . . 120
Possible types . . . 120
Possible types . . . 120
Notes . . . 120
Possible types . . . 121
Possible types . . . 121
Notes . . . 121
Possible types . . . 121
Possible types . . . 122
Possible types . . . 122
Possible types . . . 123
Possible types . . . 123
Possible types . . . 124
Possible types . . . 124
Possible types . . . 125
Possible types . . . 125
Possible types . . . 125
Notes . . . 126
Possible types . . . 126
Possible types . . . 126
Possible types . . . 126

Possible types . . . 127
Possible types . . . 128
Notes . . . 128
Possible types . . . 128
Possible types . . . 128
Possible types . . . 129
Possible types . . . 129
Possible types . . . 129
Possible types . . . 129
Possible types . . . 129
Possible types . . . 130
Possible types . . . 130
Possible types . . . 130
Possible types . . . 130
Possible types . . . 131
Possible types . . . 132
Possible types . . . 132
Possible types . . . 134
References . . . 135
psy_maps.plugin module . . . 135
psy_maps.version module . . . 136

2 Indices and tables **137**

Python Module Index **139**

Welcome to the psyplot plugin for visualizations on a map. This package uses the cartopy package to project the plots that are made with the psy-simple plugin to an earth-referenced grid. It's main plot methods are the `mapplot` and `mapvector` plot methods which can plot rectangular and triangular 2-dimensional data.

See the *psyplot plot methods* and *Example Gallery* for more information.

CHAPTER 1

Documentation

1.1 Installation

1.1.1 How to install

Installation using conda

We highly recommend to use conda for installing psy-maps.

After downloading the installer from anaconda, you can install psy-maps simply via:

```
$ conda install -c conda-forge psy-maps
```

Installation using pip

If you do not want to use conda for managing your python packages, you can also use the python package manager `pip` and install via:

```
$ pip install psy-maps
```

Note however, that you have to install cartopy beforehand.

1.1.2 Running the tests

First, clone out the github repository. First you have to

- either checkout the reference figures via:

  ```
  $ git submodule update --init `python tests/get_ref_dir.py`
  ```

- or create the reference figures via:

```
$ python setup.py test -a "--ref"
```

After that, you can run:

```
$ python setup.py test
```

or after having install pytest:

```
$ py.test
```

1.2 psyplot plot methods

This plugin defines the following new plot methods for the `psyplot.project.ProjectPlotter` class. They can, for example, be accessed through

```
In [1]: import psyplot.project as psy

In [2]: psy.plot.mapplot
Out[2]: <psyplot.project.ProjectPlotter._register_plotter.<locals>.PlotMethod at
↪0x7f4fd02e7748>
```

mapplot(*args, **kwargs)	Plot a 2D scalar field on a map
mapvector(*args, **kwargs)	Plot a 2D vector field on a map
mapcombined(*args, **kwargs)	Plot a 2D scalar field with an overlying vector field on a map

1.2.1 psyplot.project.plot.mapplot

plot.**mapplot**(**args*, ***kwargs*)

Plot a 2D scalar field on a map

This plotting method adds data arrays and plots them via *psy_maps.plotters.FieldPlotter* plotters

To plot data from a netCDF file type:

```
>>> psy.plot.mapplot(filename, name=['my_variable'], ...)
```

Possible formatoptions are

bounds	*cbar*	*cbarspacing*	*clabel*
clabelprops	*clabelsize*	*clabelweight*	*clat*
clip	*clon*	*cmap*	*cticklabels*
ctickprops	*cticks*	*cticksize*	*ctickweight*
datagrid	*extend*	*figtitle*	*figtitleprops*
figtitlesize	*figtitleweight*	*grid_color*	*grid_labels*
grid_labelsize	*grid_settings*	*interp_bounds*	*levels*
lonlatbox	*lsm*	*map_extent*	*maskbetween*
maskgeq	*maskgreater*	*maskleq*	*maskless*
miss_color	*plot*	*post*	*post_timing*
projection	*stock_img*	*text*	*tight*
title	*titleprops*	*titlesize*	*titleweight*
transform	*xgrid*	*ygrid*	

Examples

To explore the formatoptions and their documentations, use the `keys`, `summaries` and `docs` methods. For example:

```
>>> import psyplot.project as psy

# show the keys corresponding to a group or multiple
# formatopions
>>> psy.plot.mapplot.keys('labels')

# show the summaries of a group of formatoptions or of a
# formatoption
>>> psy.plot.mapplot.summaries('title')

# show the full documentation
>>> psy.plot.mapplot.docs('plot')

# or access the documentation via the attribute
>>> psy.plot.mapplot.plot
```

1.2.2 psyplot.project.plot.mapvector

`plot.`**`mapvector`**(**args*, ***kwargs*)

Plot a 2D vector field on a map

This plotting method adds data arrays and plots them via *psy_maps.plotters.VectorPlotter* plotters

To plot data from a netCDF file type:

```
>>> psy.plot.mapvector(filename, name=[['u_var', 'v_var']], ...)
```

Possible formatoptions are

arrowsize	*arrowstyle*	*bounds*	*cbar*
cbarspacing	*clabel*	*clabelprops*	*clabelsize*
clabelweight	*clat*	*clip*	*clon*
cmap	*color*	*cticklabels*	*ctickprops*
cticks	*cticksize*	*ctickweight*	*datagrid*
density	*extend*	*figtitle*	*figtitleprops*
figtitlesize	*figtitleweight*	*grid_color*	*grid_labels*
grid_labelsize	*grid_settings*	*linewidth*	*lonlatbox*
lsm	*map_extent*	*maskbetween*	*maskgeq*
maskgreater	*maskleq*	*maskless*	*plot*
post	*post_timing*	*projection*	*stock_img*
text	*tight*	*title*	*titleprops*
titlesize	*titleweight*	*transform*	*xgrid*
ygrid			

Examples

To explore the formatoptions and their documentations, use the `keys`, `summaries` and `docs` methods. For example:

```
>>> import psyplot.project as psy

# show the keys corresponding to a group or multiple
# formatopions
>>> psy.plot.mapvector.keys('labels')

# show the summaries of a group of formatoptions or of a
# formatoption
>>> psy.plot.mapvector.summaries('title')

# show the full documentation
>>> psy.plot.mapvector.docs('plot')

# or access the documentation via the attribute
>>> psy.plot.mapvector.plot
```

1.2.3 psyplot.project.plot.mapcombined

`plot.`**`mapcombined`**(**args*, ***kwargs*)

Plot a 2D scalar field with an overlying vector field on a map

This plotting method adds data arrays and plots them via *psy_maps.plotters.CombinedPlotter* plotters

To plot data from a netCDF file type:

```
>>> psy.plot.mapcombined(filename, name=[['my_variable', ['u_var', 'v_var']]], ...
↪)
```

Possible formatoptions are

arrowsize	*arrowstyle*	*bounds*	*cbar*
cbarspacing	*clabel*	*clabelprops*	*clabelsize*
clabelweight	*clat*	*clip*	*clon*
cmap	*color*	*cticklabels*	*ctickprops*
cticks	*cticksize*	*ctickweight*	*datagrid*
density	*extend*	*figtitle*	*figtitleprops*
figtitlesize	*figtitleweight*	*grid_color*	*grid_labels*
grid_labelsize	*grid_settings*	*interp_bounds*	*levels*
linewidth	*lonlatbox*	*lsm*	*map_extent*
maskbetween	*maskgeq*	*maskgreater*	*maskleq*
maskless	*miss_color*	*plot*	*post*
post_timing	*projection*	*stock_img*	*text*
tight	*title*	*titleprops*	*titlesize*
titleweight	*transform*	*vbounds*	*vcbar*
vcbarspacing	*vclabel*	*vclabelprops*	*vclabelsize*
vclabelweight	*vcmap*	*vcticklabels*	*vctickprops*
vcticks	*vcticksize*	*vctickweight*	*vplot*
xgrid	*ygrid*		

Examples

To explore the formatoptions and their documentations, use the `keys`, `summaries` and `docs` methods. For example:

```
>>> import psyplot.project as psy

# show the keys corresponding to a group or multiple
# formatopions
>>> psy.plot.mapcombined.keys('labels')

# show the summaries of a group of formatoptions or of a
# formatoption
>>> psy.plot.mapcombined.summaries('title')

# show the full documentation
>>> psy.plot.mapcombined.docs('plot')

# or access the documentation via the attribute
>>> psy.plot.mapcombined.plot
```

1.3 Example Gallery

The examples provided in this section show you how to visualize data on a map using the `psy-maps` module and how to interact with the created plots.

1.3.1 Visualizing circumpolar data

Demo script to show how a circumpolar mesh can be visualized

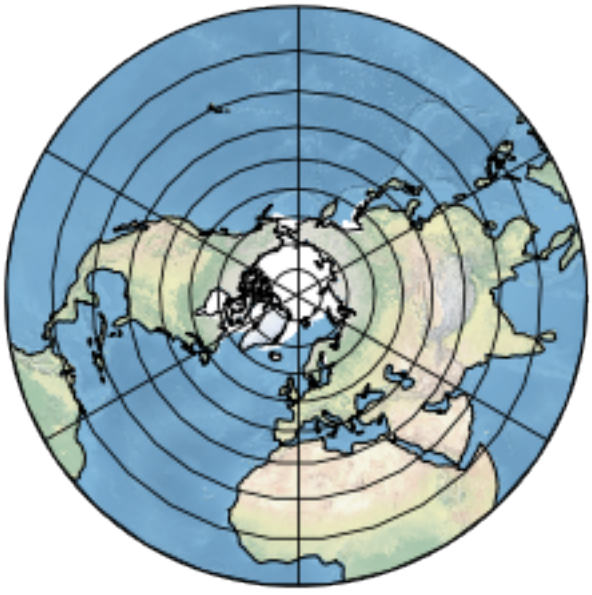
This example requires the psy-maps plugin and the file `'G10010_SIBT1850_v1.1._2013-01-15_circumpolar.nc'` which contains one variable for the sea ice concentration in the arctis. This file is based on Walsh et al., 2015 and has been remapped to a circumpolar grid using Climate Data Operators (CDO, 2015).

```
import psyplot.project as psy
import matplotlib.colors as mcol
import numpy as np
        # we show the figures after they are drawn or updated. This is useful for the
# visualization in the ipython notebook
psy.rcParams['auto_show'] = True
```

Usually, netCDF files contain one-dimensional coordinates, one for the longitude and one for the latitude. Circumpolar grids, however, are defined using 2D coordinates. The visualization using `psyplot` is however straight forward.

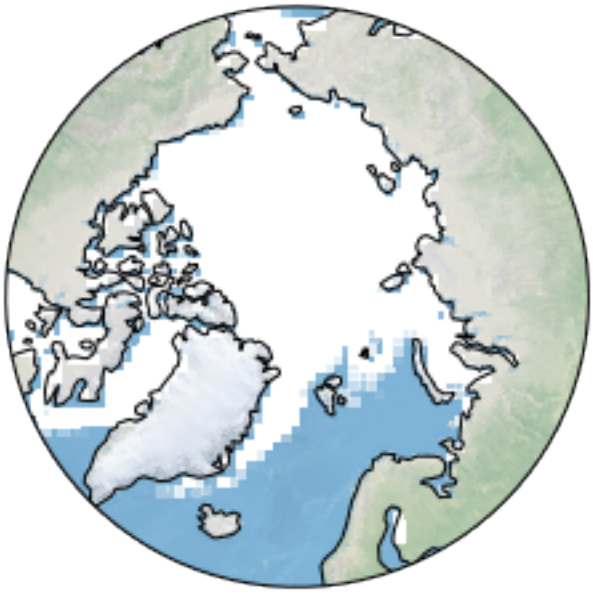
The file we are plotting here contains a variable for the sea ice concentration (0 - the grid cell contains no ice, 1 - fully ice covered). Therefore we use a colormap that reflects this behaviour. It is white but it's visibility transparency (the alpha value) increases for larger concentration. Furthermore we use a `'northpole'` projection (see Cartopy's projection list) to display it

```
colors = np.ones((100, 4))   # all white
# increase the alpha values from 0 to 1
colors[50:, -1] = np.linspace(0, 1, 50)
colors[:50, -1] = 0
cmap = mcol.LinearSegmentedColormap.from_list('white', colors, 100)
sp = psy.plot.mapplot('G10010_SIBT1850_v1.1._2013-01-15_circumpolar.nc',
                      projection='northpole', cmap=cmap,
                      # mask all values below 0
                      maskless=0.0,
                      # do not show the colorbar
                      cbar=False,
                      # plot a Natural Earth shaded relief raster on the map
                      stock_img=True)
```



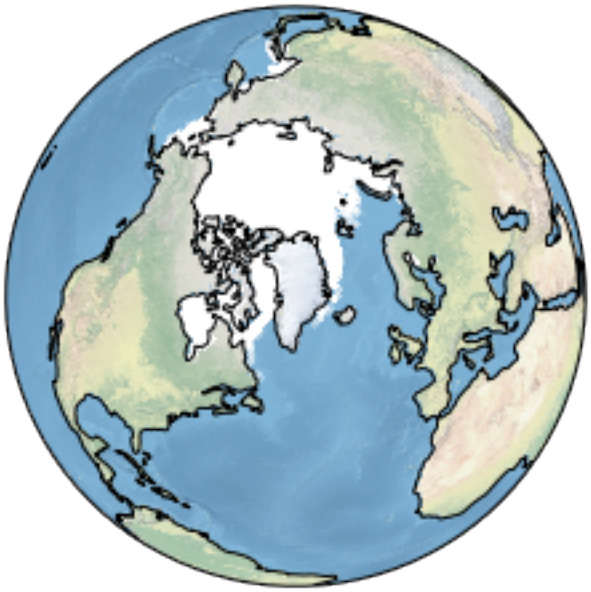
This plot now shows the entire northern hemisphere. We are however only interested in the arctic, so we adapt our lonlatbox

```
sp.update(lonlatbox=[-180, 180, 60, 90],  # lonmin, lonmax, latmin, latmax
          # disable the grid
          xgrid=False, ygrid=False)
```



We can also use the `clon` and `clat` formatoptions to focus on Greenland. Here, we might also want to change the projection since the *northpole* projection implies `clat=0`

```
sp.update(clon='Greenland', clat='Greenland', projection='ortho', lonlatbox=None)
```



Despite the beautiness of these plots, there are a few things to notice when it comes to circumpolar plots

1. Usually, psy-maps interpolates the boundaries for 1D-plots (see the `interp_bounds` formatoption), which is by default disabled for two-dimensional coordinates

2. As stated above, circumpolar grids have two dimensional coordinates. Those coordinates have to be specified in the `coordinates` attribute of the visualized netCDF variable (see the CF Conventions on Alternative Coordinates). It is important here that the dataset has not been opened using the `xarray.open_dataset` method, since this will delete the `coordinates` attribute. Instead, use the `psyplot.project.open_dataset` function which also interpretes the coordinates but does not modify the variable attributes.
3. Unfortunately, the CF-Conventions do not specify the order of the coordinates. So the coordinate attribute could be 'longitude latitude', 'latitude longitude', 'x y' or 'y x' or anything else. This causes troubles for the visualization since we do not know always automatically, what is the x- and what is the y-coordinate (except, the `axis` attribute has been specified). If we cannot tell it, we look whether there is a `lon` in the coordinate name and if yes, we assume that this is the x-coordinate. If you want to be 100% sure, create the decoder for the data array by yourself and give the x- and y-names explicitly

```
from psyplot.data import CFDecoder
ds = psy.open_dataset('netcdf-file.nc')
decoder = CFDecoder(x={'x-coordinate-name'}, y={'y-coordinate-name'})
sp = psy.plot.mapplot(fname, decoder=decoder)
```

   For more information, see the get_x and get_y methods of the CFDecoder class

To sum it all up: 1. by default, circumpolar plots are slightly shifted and the last column and row is not visualized (due to matplotlib) 2. Do not use the `xarray.open_dataset` method, it will delete the `coordinates` attribute from the variable 3. Be aware, that a coordinate listed in the `coordinates` meta attribute that contains a `lon` in the name is associated with the x-coordinate.

```
psy.close('all')
```

Note

To highlight the differences between `xarray.open_dataset` and `psyplot.project.open_dataset` just look at the `Attributes` section of the two variables in the variables below

```
import xarray as xr
print('Opened by xarray.open_dataset')
print(xr.open_dataset('G10010_SIBT1850_v1.1._2013-01-15_circumpolar.nc')['seaice_conc
↪'])
print('Opened by psyplot.project.open_dataset')
print(psy.open_dataset('G10010_SIBT1850_v1.1._2013-01-15_circumpolar.nc')['seaice_conc
↪'])
```

```
Opened by xarray.open_dataset
<xarray.DataArray 'seaice_conc' (y: 180, x: 180)>
array([[-1., -1., -1., ..., -1., -1., -1.],
       [-1., -1., -1., ..., -1., -1., -1.],
       [-1., -1., -1., ..., -1., -1., -1.],
       ...,
       [-1., -1., -1., ..., -1., -1., -1.],
       [-1., -1., -1., ..., -1., -1., -1.],
       [-1., -1., -1., ..., -1., -1., -1.]], dtype=float32)
Coordinates:
    lon      (y, x) float64 ...
    lat      (y, x) float64 ...
Dimensions without coordinates: y, x
Attributes:
    standard_name:  Sea_Ice_Concentration
    long_name:      Sea_Ice_Concentration
```

(continues on next page)

(continued from previous page)

```
    units:          Percent
    short_name:     concentration
Opened by psyplot.project.open_dataset
<xarray.DataArray 'seaice_conc' (y: 180, x: 180)>
array([[-1., -1., -1., ..., -1., -1., -1.],
       [-1., -1., -1., ..., -1., -1., -1.],
       [-1., -1., -1., ..., -1., -1., -1.],
       ...,
       [-1., -1., -1., ..., -1., -1., -1.],
       [-1., -1., -1., ..., -1., -1., -1.],
       [-1., -1., -1., ..., -1., -1., -1.]], dtype=float32)
Coordinates:
    lon      (y, x) float64 ...
    lat      (y, x) float64 ...
Dimensions without coordinates: y, x
Attributes:
    standard_name:  Sea_Ice_Concentration
    long_name:      Sea_Ice_Concentration
    units:          Percent
    short_name:     concentration
```

References

- CDO 2015: Climate Data Operators. Available at: http://www.mpimet.mpg.de/cdo
- Walsh, J. E., W. L. Chapman, and F. Fetterer. 2015. Gridded Monthly Sea Ice Extent and Concentration, 1850 Onward, Version 1. Boulder, Colorado USA. NSIDC: National Snow and Ice Data Center. doi: http://dx.doi.org/10.7265/N5833PZ5. Accessed 26.09.2017.

1.3.2 Basic data visualization on a map

Demo script to show all basic plot types on the map.

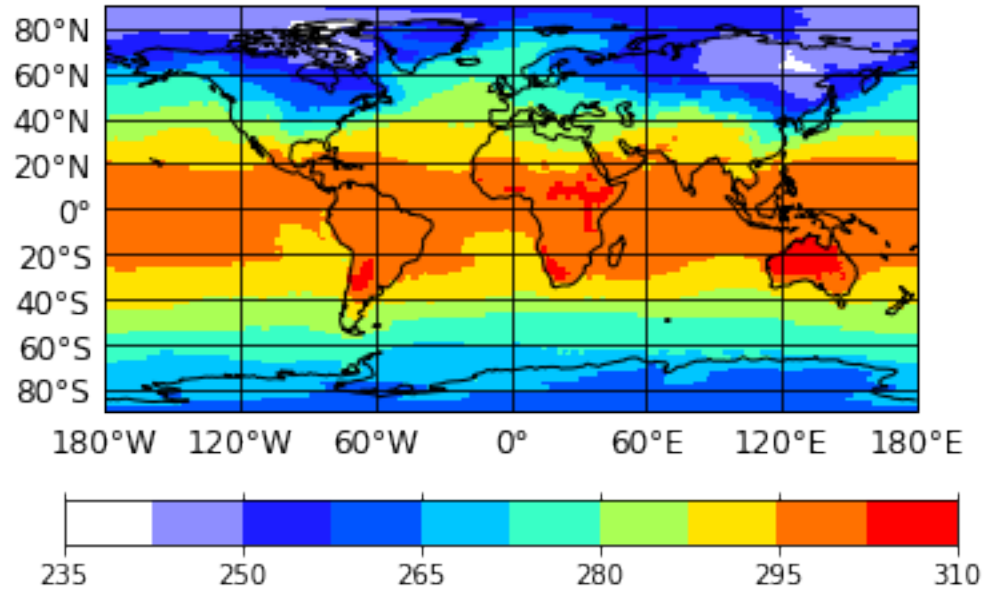
This example requires the psy-maps plugin and the file `'demo.nc'` which contains one variable for the temperature, one for zonal and one for the meridional wind direction.

```
import psyplot.project as psy
        # we show the figures after they are drawn or updated. This is useful for the
# visualization in the ipython notebook
psy.rcParams['auto_show'] = True
```

Visualizing scalar fields

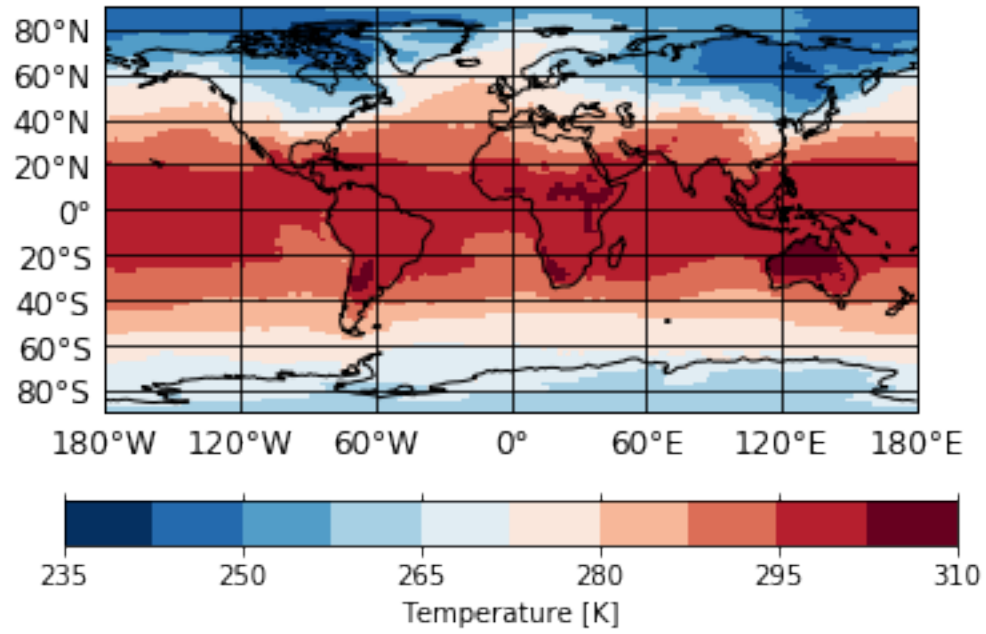
The *mapplot* method visualizes scalar data on a map.

```
maps = psy.plot.mapplot('demo.nc', name='t2m')
```



To show the colorbar label we can use the *clabel* formatoption keyword and use one of the predefined labels. Furthermore we can use the *cmap* formatoption to see one of the many available colormaps

```
maps.update(clabel='{desc}', cmap='RdBu_r')
```

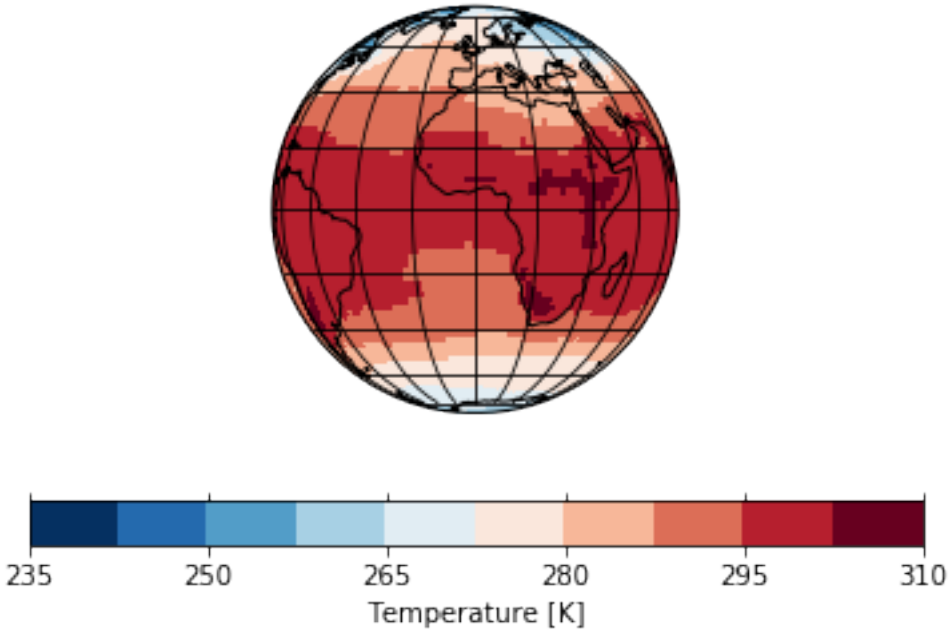


Especially useful formatoption keywords are

- projection: To modify the projection on which we draw
- lonlatbox: To select only a specific slice
- xgrid and ygrid: to disable, enable or modify the latitude-longitude grid

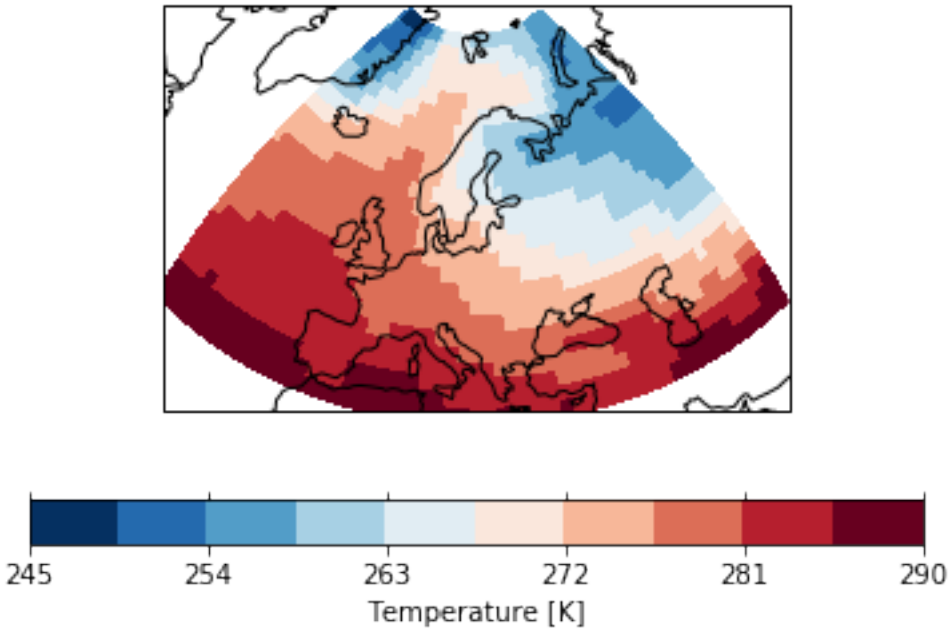
To use an orthogonal projection, we change the projection keyword to

```
maps.update(projection='ortho')
```



To focus on Europe and disable the latitude-longitude grid, we can set

```
maps.update(lonlatbox='Europe', xgrid=False, ygrid=False)
```



There are many more formatoption keys that you can explore in the online-documentation or via

```
psy.plot.mapplot.keys(grouped=True)
```

****

Color coding formatoptions
****
bounds	cbar	cbarspacing	cmap
ctickprops	cticksize	ctickweight	extend
levels	miss_color		

****
Label formatoptions
****
clabel	clabelprops	clabelsize	clabelweight
figtitle	figtitleprops	figtitlesize	figtitleweight
text	title	titleprops	titlesize
titleweight			

****
Miscallaneous formatoptions
****
clat	clip	clon	datagrid
grid_color	grid_labels	grid_labelsize	grid_settings
interp_bounds	lonlatbox	lsm	map_extent
projection	stock_img	transform	xgrid
ygrid			

****
Axis tick formatoptions
****
cticklabels	cticks

****
Masking formatoptions
****
maskbetween	maskgeq	maskgreater	maskleq
maskless			

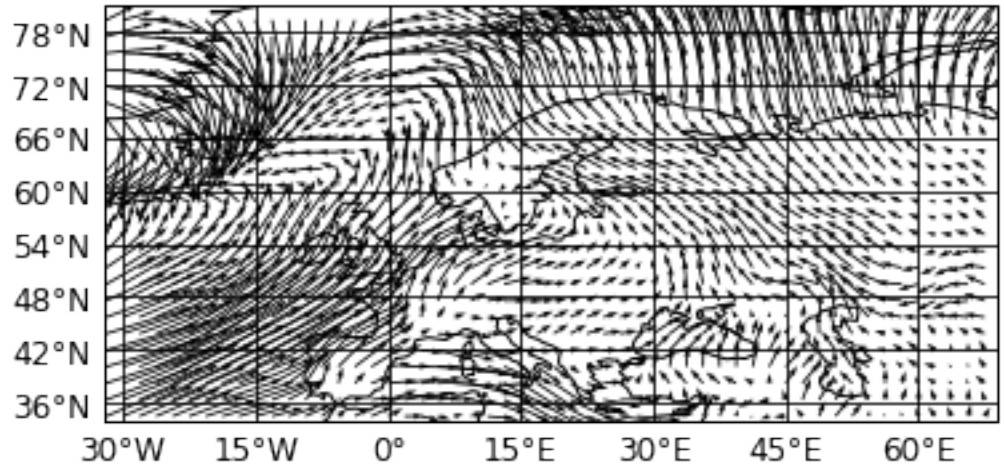
```
**************
Plot formatoptions
**************
+------+
| plot |
+------+

************************
Post processing formatoptions
************************
+-------------+-------------+
| post        | post_timing |
+-------------+-------------+

**************
Axes formatoptions
**************
+-------+
| tight |
+-------+
```

```
psy.close('all')  # we close the project because we create other figures below
```

Visualizing vector data

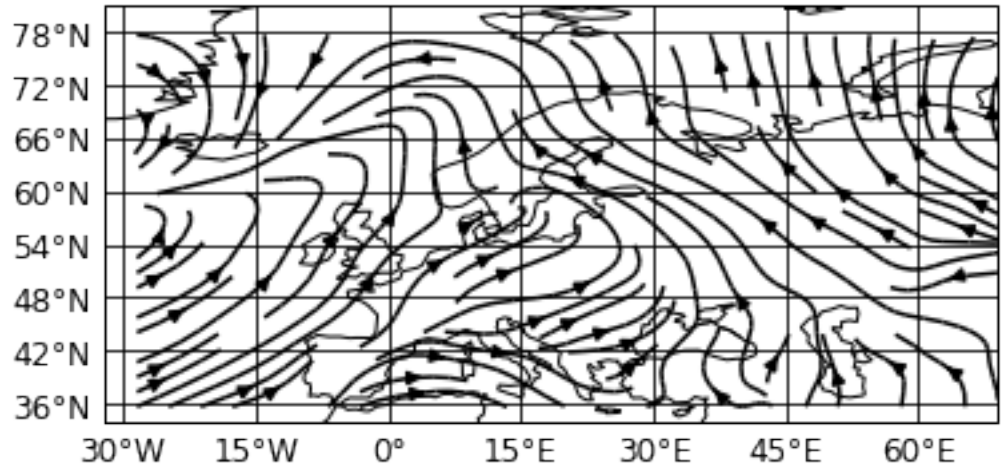
The *mapvector* method can visualize vectorized data on a map. But note that it needs a list in a list list to make the plot, where the first variable (here `'u'`) is the wind component in the x- and the second (here `'v'`) the wind component in the y-direction.

```
mapvectors = psy.plot.mapvector('demo.nc', name=[['u', 'v']], lonlatbox='Europe',
                                arrowsize=100)
```



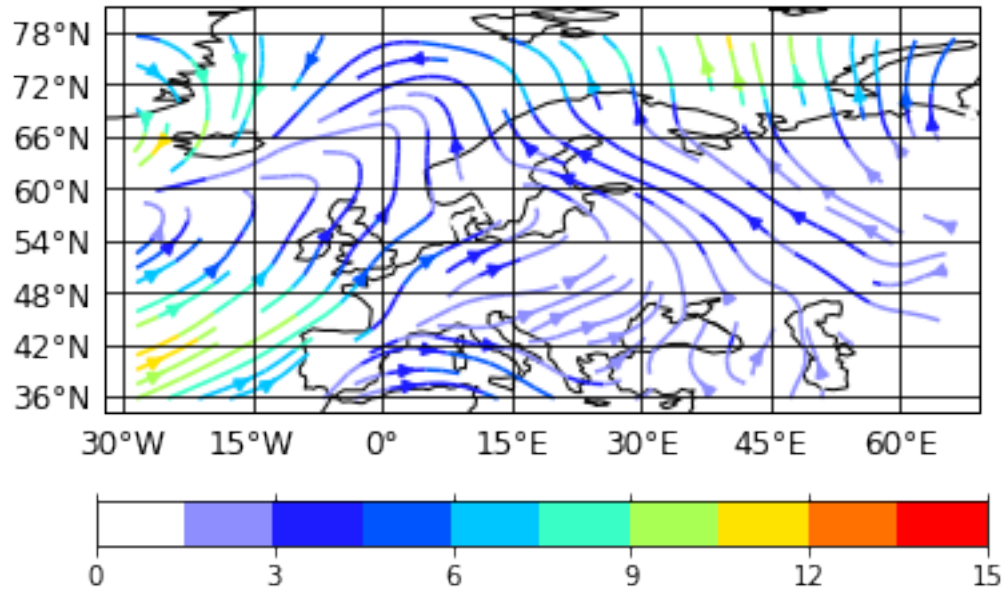
The plotter supports all formatoptions that the *mapplot* method supports. The *plot* formatoption furthermore supplies the `'stream'` value in order to make a streamplot

```
mapvectors.update(plot='stream', arrowsize=None)
```



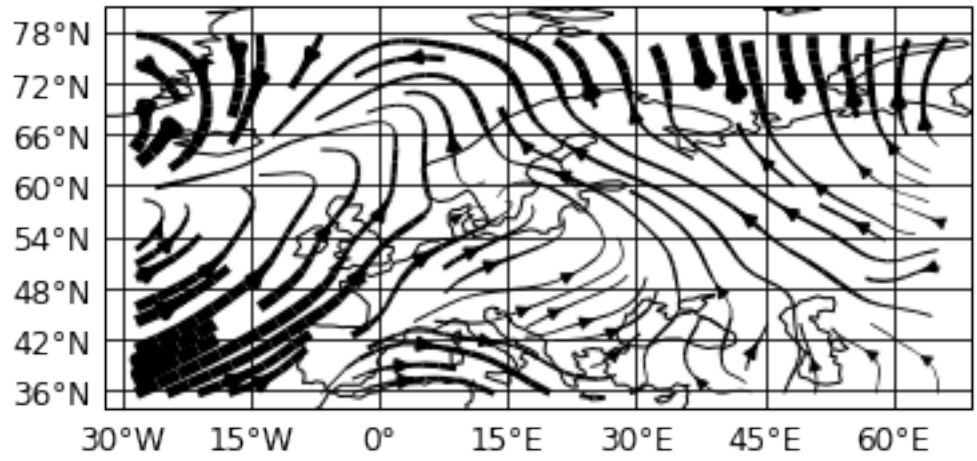
and we have two possibities to visualize the strength of the wind, either via the color coding

```
mapvectors.update(color='absolute')
```



or via the linewidth

```
mapvectors.update(color='k', linewidth=['absolute', 0.5])
```



The second number for the linewidth scales the linewidth of the arrows, where the default number is 1.0

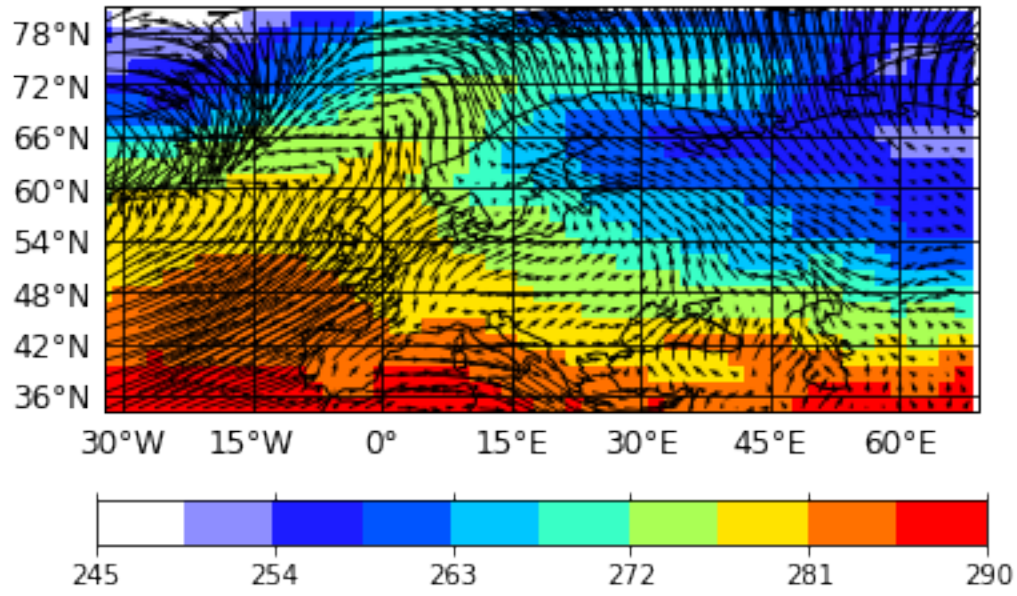
```
psy.close('all')
```

Visualizing combined scalar and vector data

The *mapcombined* method can visualize a scalar field (here temperature) with overlayed vector field. This method needs 3 variables: one for the scalar field and two for the wind fields. The calling format is

```
psy.plot.mapcombined(filename, name=[['<scalar variable name>', ['<x-vector>', '<y-
↪vector>']]])
```

```
maps = psy.plot.mapcombined('demo.nc', name=[['t2m', ['u', 'v']]], lonlatbox='Europe',
                            arrowsize=100)
```



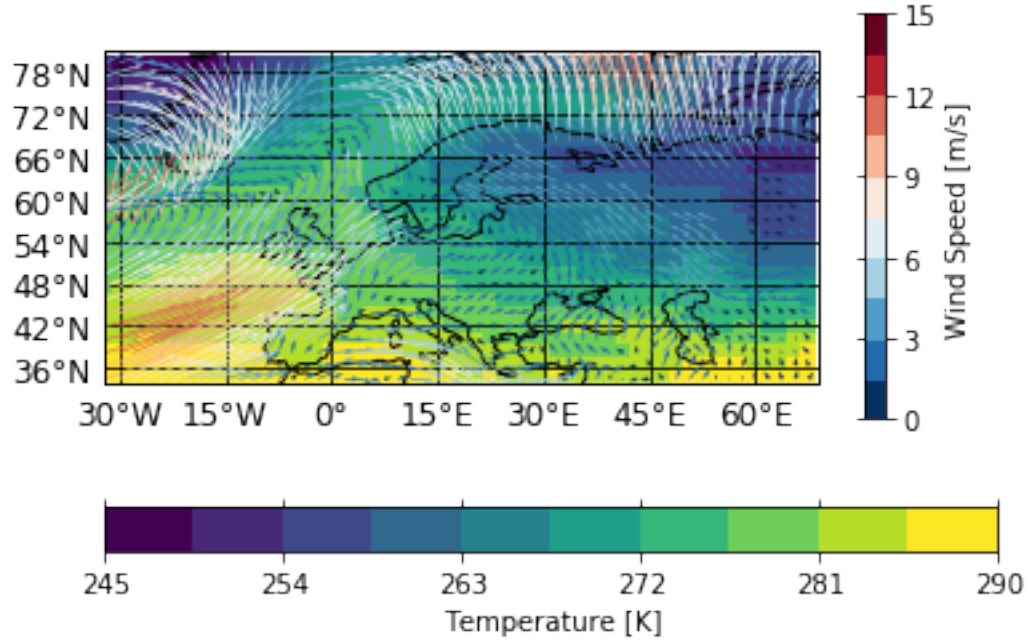
We can also modify the color coding etc. here, but all the formatoptions that affect the vector color coding start with `'v'`

```
psy.plot.mapcombined.keys('colors')
```

```
+--------------+--------------+--------------+--------------+
| color        | vcbar        | vcbarspacing | vcmap        |
+--------------+--------------+--------------+--------------+
| vbounds      | vcticksize   | vctickweight | vctickprops  |
+--------------+--------------+--------------+--------------+
| cbar         | bounds       | levels       | miss_color   |
+--------------+--------------+--------------+--------------+
| cmap         | extend       | cbarspacing  | cticksize    |
+--------------+--------------+--------------+--------------+
| ctickweight  | ctickprops   |              |              |
+--------------+--------------+--------------+--------------+
```

For example, let's modify the wind vector plots color coding and place a colorbar on the right side

```
maps.update(color='absolute', cmap='viridis', vcmap='RdBu_r', vcbar='r',
            clabel='{desc}', vclabel='Wind Speed [%(units)s]')
```



Summary

To sum it all up:

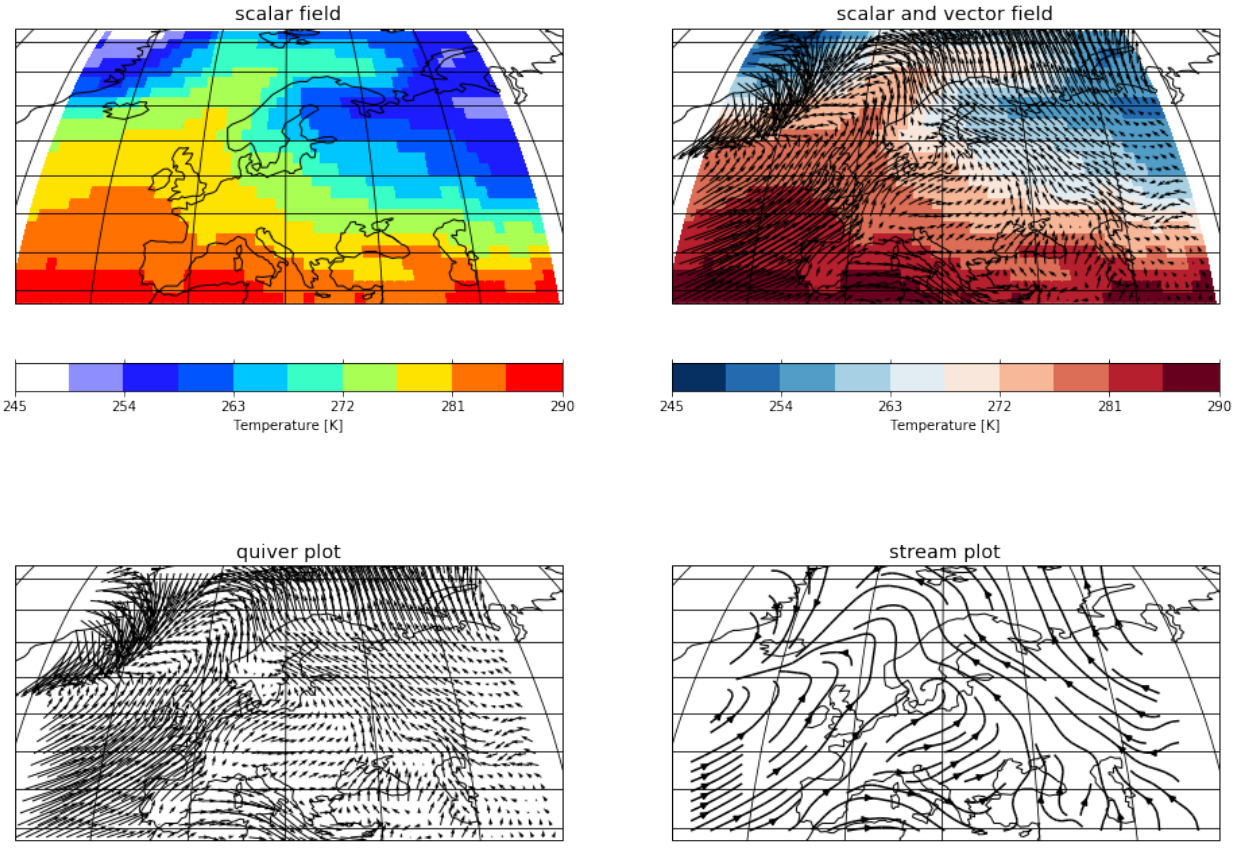
- The *mapplot* method visualizes scalar fields
- The *mapvector* method visualizes vector fiels
- The *mapcombined* method visualizes scalar and vector fields

```
# create the subplots
axes = psy.multiple_subplots(2, 2, n=4, for_maps=True)
# disable the automatic showing of the figures
psy.rcParams['auto_show'] = False
```

(continues on next page)

(continued from previous page)

```
# create plots for the scalar fields
maps = psy.plot.mapplot('demo.nc', name='t2m', clabel='{desc}', ax=axes[0],
                        title='scalar field')
# create plots for scalar and vector fields
combined = psy.plot.mapcombined(
    'demo.nc', name=[['t2m', ['u', 'v']]], clabel='{desc}', arrowsize=100,
    cmap='RdBu_r', ax=axes[1], title='scalar and vector field')
# create two plots for vector field
mapvectors = psy.plot.mapvector('demo.nc', name=[['u', 'v'], ['u', 'v']],
                                ax=axes[2:])
# where one of them shall be a stream plot
mapvectors[0].psy.update(arrowsize=100, title='quiver plot')
mapvectors[1].psy.update(plot='stream', title='stream plot')
# now update all to a robin projection
p = psy.gcp(True)
with p.no_auto_update:
    p.update(projection='robin', titlesize='x-large')
    # and the one with the wind fields to focus on Europe
    p[1:].update(lonlatbox='Europe')
    p.start_update()
```




```
psy.close('all')
```

1.3.3 Visualizing unstructured data

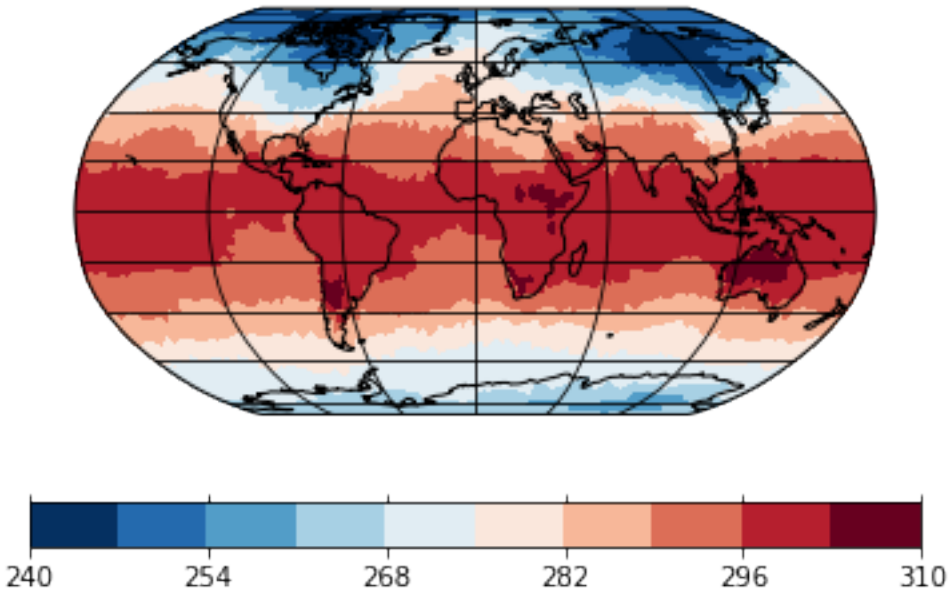
Demo script to show all basic plot types on the map.

This example requires the psy-maps plugin and the file `'icon_grid_demo.nc'` which contains one variable for the temperature, one for zonal and one for the meridional wind direction.

```
import psyplot.project as psy
```

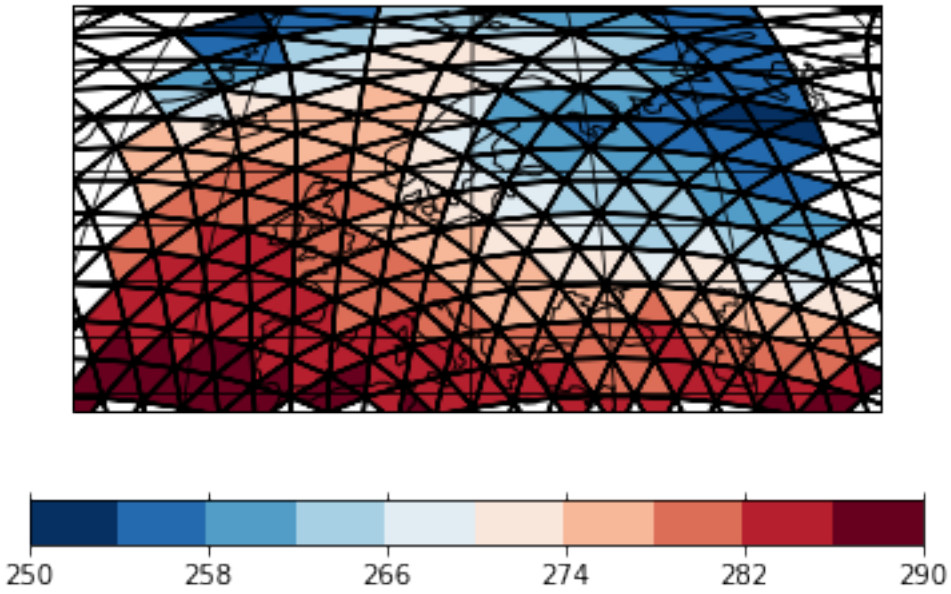
The visualization works the same way as for a usual rectangular grid. We furthermore choose a robinson projection and a colormap ranging from blue to red.

```
maps = psy.plot.mapplot('icon_grid_demo.nc', name='t2m', projection='robin',
                        cmap='RdBu_r')
```



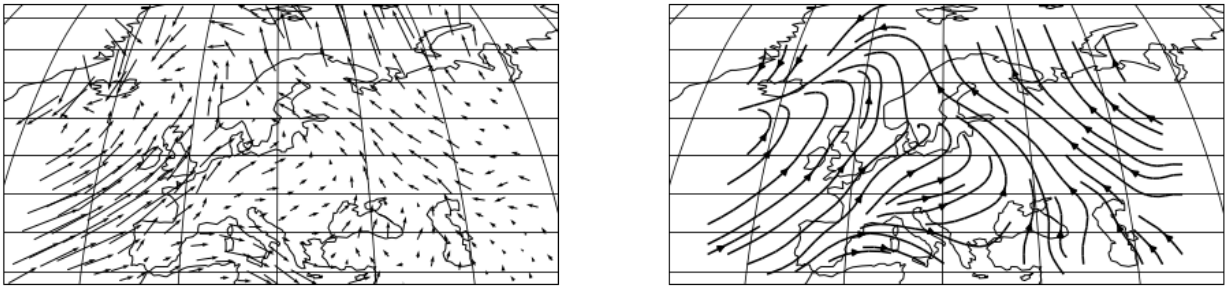
To display the data grid, we can use the `datagrid` formatoption. We furthermore restrict ourselves to `Europe` for this visualization.

```
maps.update(lonlatbox='Europe', datagrid='k-')
maps.show()
```



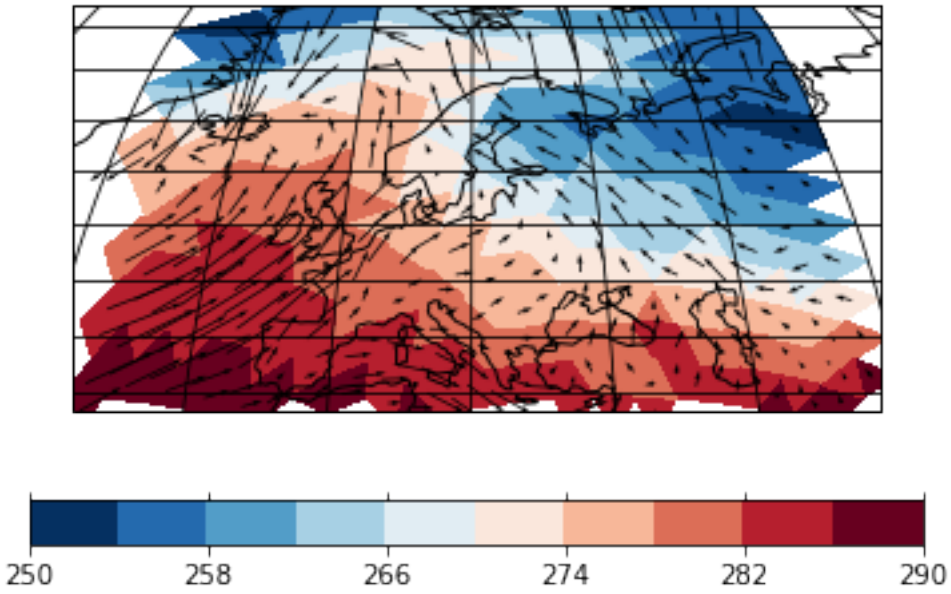
The same works for vector data

```
vectors = psy.plot.mapvector(
    'icon_grid_demo.nc', name=[['u', 'v']] * 2, projection='robin',
    ax=(1, 2), lonlatbox='Europe')
vectors.plotters[0].update(arrowsize=100)
vectors.plotters[1].update(plot='stream')
```



And combined scalar and vector fields

```
combined = psy.plot.mapcombined(
    'icon_grid_demo.nc', name=[['t2m', ['u', 'v']]], projection='robin',
    lonlatbox='Europe', arrowsize=100, cmap='RdBu_r')
```




```
psy.close('all')
```

1.4 API Reference

psy-maps: The psyplot plugin for visualizations on a map

This package contains the plotters for interactive visualization tasks on a map with the psyplot visualization framework. The package uses cartopy for projecting and displaying the data

1.4.1 Submodules

psy_maps.boxes module

Module defining a dictionary containing longitude-latitude boundary boxes for all countries and continents covered by the Vmap0 dataset

Data

lonlatboxes	dict. lonlatboxes for different countries and continents

psy_maps.boxes.**lonlatboxes = {'Afghanistan': [60.0, 74.0, 29.0, 38.0], 'Africa': [-26.0,**
dict. lonlatboxes for different countries and continents

psy_maps.plotters module

Formatoption classes

BoxBase(key[, plotter, index_in_list, ...])	Abstract base class for specifying a longitude-latitude box
CenterLat(key[, plotter, index_in_list, ...])	Set the center latitude of the plot

Continued on next page

Table 3 – continued from previous page

CenterLon(key[, plotter, index_in_list, ...])	Set the center longitude of the plot
ClipAxes(key[, plotter, index_in_list, ...])	Clip the part outside the latitudes of the map extent
CombinedMapVectorPlot(*args, **kwargs)	Choose the vector plot type
GridBase(*args, **kwargs)	Abstract base class for x- and y- grid lines
GridColor(key[, plotter, index_in_list, ...])	Set the color of the grid
GridLabelSize(key[, plotter, index_in_list, ...])	Modify the size of the grid tick labels
GridLabels(key[, plotter, index_in_list, ...])	Display the labels of the grid
GridSettings(key[, plotter, index_in_list, ...])	Modify the settings of the grid explicitly
LSM(key[, plotter, index_in_list, ...])	Draw the continents
LonLatBox(key[, plotter, index_in_list, ...])	Set the longitude-latitude box of the data shown
MapDataGrid(key[, plotter, index_in_list, ...])	Show the grid of the data
MapDensity(*args, **kwargs)	Change the density of the arrows
MapExtent(key[, plotter, index_in_list, ...])	Set the extent of the map
MapPlot2D(*args, **kwargs)	Choose how to visualize a 2-dimensional scalar data field
MapVectorColor(*args, **kwargs)	Set the color for the arrows
MapVectorPlot(*args, **kwargs)	Choose the vector plot type
Projection(*args, **kwargs)	Specify the projection for the plot
ProjectionBase(key[, plotter, ...])	Base class for formatoptions that uses cartopy.crs.CRS instances
StockImage(key[, plotter, index_in_list, ...])	Display a stock image on the map
Transform(key[, plotter, index_in_list, ...])	Specify the coordinate system of the data
XGrid(*args, **kwargs)	Draw vertical grid lines (meridians)
YGrid(*args, **kwargs)	Draw horizontal grid lines (parallels)

Plotter classes

CombinedPlotter([data, ax, auto_update, ...])	Combined 2D plotter and vector plotter on a map
FieldPlotter([data, ax, auto_update, ...])	Plotter for 2D scalar fields on a map
MapPlotter([data, ax, auto_update, project, ...])	Base plotter for visualizing data on a map
VectorPlotter([data, ax, auto_update, ...])	Plotter for visualizing 2-dimensional vector data on a map

Functions

degree_format()	
format_lats(x, pos)	
format_lons(x, pos)	
shiftdata(lonsin, datain, lon_0)	Shift longitudes (and optionally data) so that they match map projection region.

class psy_maps.plotters.**BoxBase**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: psyplot.plotter.Formatoption

Abstract base class for specifying a longitude-latitude box

Possible types

Methods

lola_from_pattern(s)	Calculate the longitude-latitude box based upon a pattern

- *str* – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)
- *[lonmin, lonmax, latmin, latmax]* – The surrounding longitude-latitude that shall be used. Values can be either a float or a string as above

See also:

LonLatBox, *MapExtent*

Parameters

- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a `psyplot.InteractiveList`
- **additional_children** (*list or str*) – Additional children to use (see the `children` attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the `dependencies` attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the `children`, `dependencies` and `connections` attributes, with values being the name of the new formatoption in this plotter.

lola_from_pattern(*s*)

Calculate the longitude-latitude box based upon a pattern

This method uses the psyplot.rcParams `'extents.boxes'` item to find longitude that match *s* and takes the surrounding box.

Parameters **s** (*str*) – The pattern to use for the keys in the `psyplot.plotter.maps.lonlatboxes` dictionary and the `'extents.boxes'` item in the `psyplot.rcParams`

Returns float – The surrounding longitude-latitude box of all items in `psyplot.rcParams['extents.boxes']` whose key match *s* if there was any match. Otherwise None is returned

Return type lonmin, lonmax, latmin, latmax or None

class psy_maps.plotters.**CenterLat**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: *psy_maps.plotters.BoxBase*

Set the center latitude of the plot

Parameters

- **None** – Let the *lonlatbox* formatoption determine the center

- **float** – Specifiy the center manually
- **str** – A pattern that matches any of the keys in the psyplot.rcParams 'extents.boxes' item (contains user-defined longitude-latitude boxes) or the psyplot.plotter.boxes.lonlatboxes dictionary (contains longitude-latitude boxes of different countries and continents)
- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a psyplot.InteractiveList
- **additional_children** (*list or str*) – Additional children to use (see the children attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the *dependencies* attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the children, *dependencies* and connections attributes, with values being the name of the new formatoption in this plotter.

Attributes

dependencies	list() -> new empty list
lonlatbox	lonlatbox Formatoption instance in the plotter
name	str(object='') -> str
priority	int(x=0) -> integer
requires_clearing	bool(x) -> bool

Methods

update(value)	Method that is call to update the formatoption on the axes

dependencies = ['lonlatbox']

lonlatbox
lonlatbox Formatoption instance in the plotter

name = 'Latitude of the center of the plot'

priority = 30

requires_clearing = True

update(*value*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

class psy_maps.plotters.**CenterLon**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: *psy_maps.plotters.BoxBase*

Set the center longitude of the plot

Parameters

- **None** – Let the *lonlatbox* formatoption determine the center
- **float** – Specifiy the center manually
- **str** – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)
- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a `psyplot.InteractiveList`
- **additional_children** (*list or str*) – Additional children to use (see the `children` attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the *dependencies* attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the `children`, *dependencies* and `connections` attributes, with values being the name of the new formatoption in this plotter.

Attributes

dependencies	list() -> new empty list
lonlatbox	lonlatbox Formatoption instance in the plotter
name	str(object='') -> str
priority	int(x=0) -> integer
requires_clearing	bool(x) -> bool

Methods

update(value)	Method that is call to update the formatoption on the axes

dependencies = ['lonlatbox']

lonlatbox
lonlatbox Formatoption instance in the plotter

name = 'Longitude of the center of the plot'

priority = 30

requires_clearing = True

update(*value*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

class psy_maps.plotters.**ClipAxes**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: psyplot.plotter.Formatoption

Clip the part outside the latitudes of the map extent

Possible types

Attributes

connections	list() -> new empty list
lonlatbox	lonlatbox Formatoption instance in the plotter
map_extent	map_extent Formatoption instance in the plotter
priority	int(x=0) -> integer

Methods

draw_circle()	
remove()	Method to remove the effects of this formatoption
update(value)	Method that is call to update the formatoption on the axes

- *None* – Clip if all longitudes are shown (i.e. the extent goes from -180 to 180) and the projection is orthographic or stereographic
- *bool* – True, clip, else, don't

Notes

If the plot is clipped. You might need to update with *replot=True*!

Parameters

- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a psyplot.InteractiveList
- **additional_children** (*list or str*) – Additional children to use (see the children attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the dependencies attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the children, dependencies and *connections* attributes, with values being the name of the new formatoption in this plotter.

connections = ['lonlatbox', 'map_extent']

draw_circle()

lonlatbox
lonlatbox Formatoption instance in the plotter

map_extent
map_extent Formatoption instance in the plotter

priority = 20

remove()
Method to remove the effects of this formatoption

This method is called when the axes is cleared due to a formatoption with `requires_clearing` set to True. You don't necessarily have to implement this formatoption if your plot results are removed by the usual `matplotlib.axes.Axes.clear()` method.

update(*value*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

class psy_maps.plotters.**CombinedMapVectorPlot**(**args*, ***kwargs*)
Bases: *psy_maps.plotters.MapVectorPlot*

Choose the vector plot type

Possible types

Attributes

arrowsize	arrowsize Formatoption instance in the plotter
arrowstyle	arrowstyle Formatoption instance in the plotter
bounds	bounds Formatoption instance in the plotter
clat	clat Formatoption instance in the plotter
clip	clip Formatoption instance in the plotter
clon	clon Formatoption instance in the plotter
cmap	cmap Formatoption instance in the plotter
color	color Formatoption instance in the plotter
data_dependent	bool(x) -> bool
density	density Formatoption instance in the plotter
linewidth	linewidth Formatoption instance in the plotter
lonlatbox	lonlatbox Formatoption instance in the plotter
transform	transform Formatoption instance in the plotter
transpose	transpose Formatoption instance in the plotter

Methods

update(*args, **kwargs)	Method that is call to update the formatoption on the axes

str – Plot types can be either

quiver to make a quiver plot

stream to make a stream plot

arrowsize
arrowsize Formatoption instance in the plotter

arrowstyle
arrowstyle Formatoption instance in the plotter

bounds
bounds Formatoption instance in the plotter

clat
clat Formatoption instance in the plotter

clip
clip Formatoption instance in the plotter

clon
clon Formatoption instance in the plotter

cmap
cmap Formatoption instance in the plotter

color
color Formatoption instance in the plotter

data_dependent = True

density
density Formatoption instance in the plotter

linewidth
linewidth Formatoption instance in the plotter

lonlatbox
lonlatbox Formatoption instance in the plotter

transform
transform Formatoption instance in the plotter

transpose
transpose Formatoption instance in the plotter

update(**args*, ***kwargs*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

class psy_maps.plotters.**CombinedPlotter**(*data=None*, *ax=None*, *auto_update=None*, *project=None*, *draw=None*, *make_plot=True*, *clear=False*, *enable_post=False*, ***kwargs*)

Bases: psy_simple.plotters.CombinedBase, *psy_maps.plotters.FieldPlotter*, *psy_maps.plotters.VectorPlotter*

Combined 2D plotter and vector plotter on a map

See also:

psyplot.plotter.simple.CombinedSimplePlotter for a simple version of this class

FieldPlotter, *VectorPlotter*

Vector plot formatoptions

density	Change the density of the arrows
arrowsize	Change the size of the arrows
arrowstyle	Change the style of the arrows

Plot formatoptions

plot	Choose how to visualize a 2-dimensional scalar data field
vplot	Choose the vector plot type

Miscallaneous formatoptions

xgrid	Draw vertical grid lines (meridians)
ygrid	Draw horizontal grid lines (parallels)
clat	Set the center latitude of the plot
clip	Clip the part outside the latitudes of the map extent
clon	Set the center longitude of the plot
datagrid	Show the grid of the data
grid_color	Set the color of the grid
grid_labels	Display the labels of the grid
grid_labelsize	Modify the size of the grid tick labels
grid_settings	Modify the settings of the grid explicitly
interp_bounds	Interpolate grid cell boundaries for 2D plots
linewidth	Change the linewidth of the arrows
lonlatbox	Set the longitude-latitude box of the data shown
lsm	Draw the continents
map_extent	Set the extent of the map
projection	Specify the projection for the plot
stock_img	Display a stock image on the map
transform	Specify the coordinate system of the data

Axis tick formatoptions

cticklabels	Specify the colorbar ticklabels
cticks	Specify the tick locations of the colorbar
vcticklabels	Specify the colorbar ticklabels
vcticks	Specify the tick locations of the vector colorbar

Label formatoptions

clabel	Show the colorbar label
clabelprops	Properties of the Colorbar label
clabelsize	Set the size of the Colorbar label
clabelweight	Set the fontweight of the Colorbar label
figtitle	Plot a figure title
figtitleprops	Properties of the figure title
figtitlesize	Set the size of the figure title
figtitleweight	Set the fontweight of the figure title
text	Add text anywhere on the plot
title	Show the title
titleprops	Properties of the title
titlesize	Set the size of the title
titleweight	Set the fontweight of the title
vclabel	Show the colorbar label of the vector plot
vclabelprops	Properties of the Vector colorbar label

Continued on next page

Table 19 – continued from previous page

vclabelsize	Set the size of the Vector colorbar label
vclabelweight	Set the fontweight of the Vector colorbar label

Axes formatoptions

tight	Automatically adjust the plots.

Color coding formatoptions

bounds	Specify the boundaries of the colorbar
cbar	Specify the position of the colorbars
cbarspacing	Specify the spacing of the bounds in the colorbar
cmap	Specify the color map
color	Set the color for the arrows
ctickprops	Specify the font properties of the colorbar ticklabels
cticksize	Specify the font size of the colorbar ticklabels
ctickweight	Specify the fontweight of the colorbar ticklabels
extend	Draw arrows at the side of the colorbar
levels	The levels for the contour plot
miss_color	Set the color for missing values
vbounds	Specify the boundaries of the vector colorbar
vcbar	Specify the position of the vector plot colorbars
vcbarspacing	Specify the spacing of the bounds in the colorbar
vcmap	Specify the color map
vctickprops	Specify the font properties of the colorbar ticklabels
vcticksize	Specify the font size of the colorbar ticklabels
vctickweight	Specify the fontweight of the colorbar ticklabels

Masking formatoptions

maskbetween	Mask data points between two numbers
maskgeq	Mask data points greater than or equal to a number
maskgreater	Mask data points greater than a number
maskleq	Mask data points smaller than or equal to a number
maskless	Mask data points smaller than a number

Post processing formatoptions

post	Apply your own postprocessing script
post_timing	Determine when to run the `post` formatoption

Parameters

- **data** (*InteractiveArray or ArrayList, optional*) – Data object that shall be visualized. If given and *plot* is True, the `initialize_plot()` method is called at the end. Otherwise you can call this method later by yourself
- **ax** (*matplotlib.axes.Axes*) – Matplotlib Axes to plot on. If None, a new one will be created as soon as the `initialize_plot()` method is called
- **auto_update** (*bool*) – Default: None. A boolean indicating whether this list shall auto-

matically update the contained arrays when calling the `update()` method or not. See also the `no_auto_update` attribute. If None, the value from the `'lists.auto_update'` key in the `psyplot.rcParams` dictionary is used.

- **draw** (*bool or None*) – Boolean to control whether the figure of this array shall be drawn at the end. If None, it defaults to the *'auto_draw'* parameter in the `psyplot.rcParams` dictionary
- **make_plot** (*bool*) – If True, and *data* is not None, the plot is initialized. Otherwise only the framework between plotter and data is set up
- **clear** (*bool*) – If True, the axes is cleared first
- **enable_post** (*bool*) – If True, the *post* formatoption is enabled and post processing scripts are allowed
- ****kwargs** – Any formatoption key from the `formatoptions` attribute that shall be used

density
Change the density of the arrows

Possible types

- *float* – Scales the density of the arrows in x- and y-direction (1.0 means no scaling)
- *tuple (x, y)* – Defines the scaling in x- and y-direction manually

plot
Choose how to visualize a 2-dimensional scalar data field

Possible types

- *None* – Don't make any plotting
- *'mesh'* – Use the `matplotlib.pyplot.pcolormesh()` function to make the plot or the `matplotlib.pyplot.tripcolor()` for an unstructered grid
- *'tri'* – Use the `matplotlib.pyplot.tripcolor()` function to plot data on a triangular grid
- *'contourf'* – Make a filled contour plot using the `matplotlib.pyplot.contourf()` function or the `matplotlib.pyplot.tricontourf()` for triangular data. The levels for the contour plot are controlled by the *levels* formatoption
- *'tricontourf'* – Make a filled contour plot using the `matplotlib.pyplot.tricontourf()` function

vplot
Choose the vector plot type

Possible types

str – Plot types can be either

quiver to make a quiver plot

stream to make a stream plot

xgrid
Draw vertical grid lines (meridians)

This formatoption specifies at which longitudes to draw the meridians.

Possible types

- *None* – Don't draw gridlines (same as `False`)
- *bool* – True: draw gridlines and determine position automatically False: don't draw gridlines
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero

- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.

See also:

ygrid, *grid_color*, *grid_labels*

ygrid
Draw horizontal grid lines (parallels)

This formatoption specifies at which latitudes to draw the parallels.

Possible types

- *None* – Don't draw gridlines (same as `False`)
- *bool* – True: draw gridlines and determine position automatically False: don't draw gridlines
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

minmax Uses the minimum as minimal tick and maximum as maximal tick

sym Same as minmax but symmetric around zero

- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.

See also:

xgrid, *grid_color*, *grid_labels*

arrowsize
Change the size of the arrows

Possible types

- *None* – make no scaling
- *float* – Factor scaling the size of the arrows

See also:

arrowstyle, *linewidth*, *density*, *color*

arrowstyle
Change the style of the arrows

Possible types

str – Any arrow style string (see `FancyArrowPatch`)

Notes

This formatoption only has an effect for stream plots

See also:

arrowsize, *linewidth*, *density*, *color*

clat
Set the center latitude of the plot

Parameters

- **None** – Let the *lonlatbox* formatoption determine the center
- **float** – Specifiy the center manually

- **str** – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)

clip
Clip the part outside the latitudes of the map extent

Possible types

- *None* – Clip if all longitudes are shown (i.e. the extent goes from -180 to 180) and the projection is orthographic or stereographic
- *bool* – True, clip, else, don't

Notes

If the plot is clipped. You might need to update with *replot=True*!

clon
Set the center longitude of the plot

Parameters

- **None** – Let the *lonlatbox* formatoption determine the center
- **float** – Specifiy the center manually
- **str** – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)

datagrid
Show the grid of the data

This formatoption shows the grid of the data (without labels)

Possible types

- *None* – Don't show the data grid
- *str* – A linestyle in the form `'k-'`, where `'k'` is the color and `'-'` the linestyle.
- *dict* – any keyword arguments that are passed to the plotting function (`matplotlib.pyplot.triplot()` for triangular grids and `matplotlib.pyplot.hlines()` for rectilinear grids)

See also:

xgrid, *ygrid*

grid_color
Set the color of the grid

Possible types

- *None* – Choose the default line color
- *color* – Any valid color for matplotlib (see the `matplotlib.pyplot.plot()` documentation)

See also:

grid_settings, *grid_labels*, *grid_labelsize*, *xgrid*, *ygrid*

grid_labels
Display the labels of the grid

Possible types

- *None* – Grid labels are draw if possible
- *bool* – If True, labels are drawn and if this is not possible, a warning is raised

See also:

grid_color, *grid_settings*, *grid_labelsize*, *xgrid*, *ygrid*

grid_labelsize
Modify the size of the grid tick labels

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

grid_color, *grid_labels*, *xgrid*, *ygrid*, *grid_settings*

grid_settings
Modify the settings of the grid explicitly

Possible types

dict – Items may be any key-value-pair of the `matplotlib.collections.LineCollection` class

See also:

grid_color, *grid_labels*, *grid_labelsize*, *xgrid*, *ygrid*

interp_bounds
Interpolate grid cell boundaries for 2D plots

This formatoption can be used to tell enable and disable the interpolation of grid cell boundaries. Usually, netCDF files only contain the centered coordinates. In this case, we interpolate the boundaries between the grid cell centers.

Possible types

- *None* – Interpolate the boundaries, except for circumpolar grids
- *bool* – If True (the default), the grid cell boundaries are inter- and extrapolated. Otherwise, if False, the coordinate centers are used and the default behaviour of matplotlib cuts of the most outer row and column of the 2D-data. Note that this results in a slight shift of the data

linewidth
Change the linewidth of the arrows

Possible types

- *float* – give the linewidth explicitly
- *string {'absolute', 'u', 'v'}* – Strings may define how the formatoption is calculated. Possible strings are
  - **absolute**: for the absolute wind speed
  - **u**: for the u component
  - **v**: for the v component
- *tuple (string, float)* – *string* may be one of the above strings, *float* may be a scaling factor
- *2D-array* – The values determine the linewidth for each plotted arrow. Note that the shape has to match the one of u and v.

See also:

arrowsize, *arrowstyle*, *density*, *color*

lonlatbox
Set the longitude-latitude box of the data shown

This formatoption extracts the data that matches the specified box.

Possible types

- *None* – Use the full data
- *str* – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)
- *[lonmin, lonmax, latmin, latmax]* – The surrounding longitude-latitude that shall be used. Values can be either a float or a string as above

Notes

- For only specifying the region of the plot, see the *map_extent* formatoption
- If the coordinates are two-dimensional (e.g. for a circumpolar grid), than the data is not extracted but values outside the specified longitude-latitude box are set to NaN

See also:

map_extent

lsm
Draw the continents

Possible types

- *bool* – True: draw the continents with a line width of 1 False: don't draw the continents
- *float* – Specifies the linewidth of the continents
- *str* – The resolution of the land-sea mask (see the `cartopy.mpl.geoaxes.GeoAxesSubplot.coastlines()` method. Usually one of `('110m', '50m', '10m')`.
- *list [str or bool, float]* – The resolution and the linewidth

map_extent
Set the extent of the map

Possible types

- *None* – The map extent is specified by the data (i.e. by the *lonlatbox* formatoption)
- *'global'* – The whole globe is shown
- *str* – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)
- *[lonmin, lonmax, latmin, latmax]* – The surrounding longitude-latitude that shall be used. Values can be either a float or a string as above

Notes

This formatoption sets the extent of the plot. For choosing the region for the data, see the *lonlatbox* formatoption

See also:

lonlatbox

projection
Specify the projection for the plot

This formatoption defines the projection of the plot

Possible types

- *cartopy.crs.CRS* – A cartopy projection instance (e.g. `cartopy.crs.PlateCarree`)
- *str* – A string specifies the projection instance to use. The centered longitude and latitude are determined by the *clon* and *clat* formatoptions. Possible strings are (each standing for the specified projection)

cyl	`cartopy.crs.PlateCarree`
robin	`cartopy.crs.Robinson`
moll	`cartopy.crs.Mollweide`
geo	`cartopy.crs.Geostationary`
northpole	`cartopy.crs.NorthPolarStereo`
southpole	`cartopy.crs.SouthPolarStereo`
ortho	`cartopy.crs.Orthographic`
stereo	`cartopy.crs.Stereographic`
near	`cartopy.crs.NearsidePerspective`

Warning: An update of the projection clears the axes!

`stock_img`
Display a stock image on the map

This formatoption uses the `cartopy.mpl.geoaxes.GeoAxes.stock_img()` method to display a downsampled version of the Natural Earth shaded relief raster on the map

Possible types

bool – If True, the image is displayed

`transform`
Specify the coordinate system of the data

This formatoption defines the coordinate system of the data (usually we expect a simple latitude longitude coordinate system)

Possible types

- *cartopy.crs.CRS* – A cartopy projection instance (e.g. `cartopy.crs.PlateCarree`)
- *str* – A string specifies the projection instance to use. The centered longitude and latitude are determined by the *`clon`* and *`clat`* formatoptions. Possible strings are (each standing for the specified projection)

cyl	`cartopy.crs.PlateCarree`
robin	`cartopy.crs.Robinson`
moll	`cartopy.crs.Mollweide`
geo	`cartopy.crs.Geostationary`
northpole	`cartopy.crs.NorthPolarStereo`
southpole	`cartopy.crs.SouthPolarStereo`
ortho	`cartopy.crs.Orthographic`
stereo	`cartopy.crs.Stereographic`
near	`cartopy.crs.NearsidePerspective`

`cticklabels`
Specify the colorbar ticklabels

Possible types

- *str* – A formatstring like `'%Y'` for plotting the year (in the case that time is shown on the axis) or '%i' for integers
- *array* – An array of strings to use for the ticklabels

See also:

cticks, *cticksize*, *ctickweight*, *ctickprops*, *vcticks*, *vcticksize*, *vctickweight*, *vctickprops*

cticks
Specify the tick locations of the colorbar

Possible types

- *None* – use the default ticks
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero

  bounds let the *bounds* keyword determine the ticks. An additional integer *i* may be specified to only use every i-th bound as a tick (see also *int* below)
- *int* – Specifies how many ticks to use with the `'bounds'` option. I.e. if integer `i`, then this is the same as `['bounds', i]`.

See also:

ticklabels

vcticklabels
Specify the colorbar ticklabels

Possible types

- *str* – A formatstring like `'%Y'` for plotting the year (in the case that time is shown on the axis) or '%i' for integers
- *array* – An array of strings to use for the ticklabels

See also:

cticks, *cticksize*, *ctickweight*, *ctickprops*, *vcticks*, *vcticksize*, *vctickweight*, *vctickprops*

vcticks
Specify the tick locations of the vector colorbar

Possible types

- *None* – use the default ticks
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero

  bounds let the *bounds* keyword determine the ticks. An additional integer *i* may be specified to only use every i-th bound as a tick (see also *int* below)

- *int* – Specifies how many ticks to use with the `'bounds'` option. I.e. if integer `i`, then this is the same as `['bounds', i]`.

See also:

cticklabels, *vcticklabels*

clabel
Show the colorbar label

Set the label of the colorbar. You can insert any meta key from the `xarray.DataArray.attrs` via a string like `'%(key)s'`. Furthermore there are some special cases:

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.
- `'%(x)s'`, `'%(y)s'`, `'%(z)s'`, `'%(t)s'` will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)
- any attribute of one of the above coordinates is inserted via `axis + key` (e.g. the name of the x-coordinate can be inserted via `'%(xname)s'`).
- Labels defined in the `psyplot.rcParams 'texts.labels'` key are also replaced when enclosed by '{}'. The standard labels are
  - tinfo: `%H:%M`

- dtinfo: `%B %d, %Y. %H:%M`
- dinfo: `%B %d, %Y`
- desc: `%(long_name)s [%(units)s]`
- sdesc: `%(name)s [%(units)s]`

Possible types

str – The title for the `set_label()` method.

See also:

clabelsize, *clabelweight*, *clabelprops*

clabelprops
Properties of the Colorbar label

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

clabel, *clabelsize*, *clabelweight*

clabelsize
Set the size of the Colorbar label

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

clabel, *clabelweight*, *clabelprops*

clabelweight
Set the fontweight of the Colorbar label

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

clabel, *clabelsize*, *clabelprops*

figtitle
Plot a figure title

Set the title of the figure. You can insert any meta key from the xarray.DataArray.attrs via a string like '%(key)s'. Furthermore there are some special cases:

- Strings like '%Y', '%b', etc. will be replaced using the datetime.datetime.strftime() method as long as the data has a time coordinate and this can be converted to a datetime object.
- '%(x)s', '%(y)s', '%(z)s', '%(t)s' will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)
- any attribute of one of the above coordinates is inserted via axis + key (e.g. the name of the x-coordinate can be inserted via '%(xname)s').
- Labels defined in the psyplot.rcParams 'texts.labels' key are also replaced when enclosed by '{}'. The standard labels are
  - tinfo: %H:%M
  - dtinfo: %B %d, %Y. %H:%M
  - dinfo: %B %d, %Y
  - desc: %(long_name)s [%(units)s]
  - sdesc: %(name)s [%(units)s]

Possible types

str – The title for the suptitle() function

Notes

- If the plotter is part of a psyplot.project.Project and multiple plotters of this project are on the same figure, the replacement attributes (see above) are joined by a delimiter. If the delimiter attribute of this Figtitle instance is not None, it will be used. Otherwise the rcParams['texts.delimiter'] item is used.
- This is the title of the whole figure! For the title of this specific subplot, see the *title* formatoption.

See also:

title, *figtitlesize*, *figtitleweight*, *figtitleprops*

figtitleprops
Properties of the figure title

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

figtitle, *figtitlesize*, *figtitleweight*

figtitlesize
Set the size of the figure title

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

figtitle, *figtitleweight*, *figtitleprops*

figtitleweight
Set the fontweight of the figure title

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

figtitle, *figtitlesize*, *figtitleprops*

text
Add text anywhere on the plot

This formatoption draws a text on the specified position on the figure. You can insert any meta key from the `xarray.DataArray.attrs` via a string like `'%(key)s'`. Furthermore there are some special cases:

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.
- `'%(x)s'`, `'%(y)s'`, `'%(z)s'`, `'%(t)s'` will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)
- any attribute of one of the above coordinates is inserted via `axis + key` (e.g. the name of the x-coordinate can be inserted via `'%(xname)s'`).
- Labels defined in the `psyplot.rcParams 'texts.labels'` key are also replaced when enclosed by '{}'. The standard labels are
  - tinfo: `%H:%M`
  - dtinfo: `%B %d, %Y. %H:%M`
  - dinfo: `%B %d, %Y`
  - desc: `%(long_name)s [%(units)s]`
  - sdesc: `%(name)s [%(units)s]`

Possible types

- *str* – If string s: this will be used as (1., 1., s, {'ha': 'right'}) (i.e. a string in the upper right corner of the axes).

- *tuple or list of tuples (x,y,s[,coord.-system][,options])* – Each tuple defines a text instance on the plot. 0<=x, y<=1 are the coordinates. The coord.-system can be either the data coordinates (default, `'data'`) or the axes coordinates (`'axes'`) or the figure coordinates ('fig'). The string s finally is the text. options may be a dictionary to specify format the appearence (e.g. `'color'`, `'fontweight'`, `'fontsize'`, etc., see `matplotlib.text.Text` for possible keys). To remove one single text from the plot, set (x,y,'[, coord.-system]) for the text at position (x,y)
- *empty list* – remove all texts from the plot

See also:

title, *figtitle*

title
Show the title

Set the title of the plot. You can insert any meta key from the `xarray.DataArray.attrs` via a string like `'%(key)s'`. Furthermore there are some special cases:

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.
- `'%(x)s'`, `'%(y)s'`, `'%(z)s'`, `'%(t)s'` will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)
- any attribute of one of the above coordinates is inserted via `axis + key` (e.g. the name of the x-coordinate can be inserted via `'%(xname)s'`).
- Labels defined in the `psyplot.rcParams 'texts.labels'` key are also replaced when enclosed by '{}'. The standard labels are
  - tinfo: `%H:%M`
  - dtinfo: `%B %d, %Y. %H:%M`
  - dinfo: `%B %d, %Y`
  - desc: `%(long_name)s [%(units)s]`
  - sdesc: `%(name)s [%(units)s]`

Possible types

str – The title for the `title()` function.

Notes

This is the title of this specific subplot! For the title of the whole figure, see the *figtitle* formatoption.

See also:

figtitle, *titlesize*, *titleweight*, *titleprops*

titleprops
Properties of the title

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

title, *titlesize*, *titleweight*

titlesize
Set the size of the title

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

title, *titleweight*, *titleprops*

titleweight
Set the fontweight of the title

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

title, *titlesize*, *titleprops*

vclabel
Show the colorbar label of the vector plot

Set the label of the colorbar. You can insert any meta key from the `xarray.DataArray.attrs` via a string like `'%(key)s'`. Furthermore there are some special cases:

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.
- `'%(x)s'`, `'%(y)s'`, `'%(z)s'`, `'%(t)s'` will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)
- any attribute of one of the above coordinates is inserted via `axis + key` (e.g. the name of the x-coordinate can be inserted via `'%(xname)s'`).
- Labels defined in the `psyplot.rcParams 'texts.labels'` key are also replaced when enclosed by '{}'. The standard labels are
  - tinfo: `%H:%M`
  - dtinfo: `%B %d, %Y. %H:%M`
  - dinfo: `%B %d, %Y`
  - desc: `%(long_name)s [%(units)s]`

– sdesc: `%(name)s [%(units)s]`

Possible types

str – The title for the `set_label()` method.

See also:

vclabelsize, *vclabelweight*, *vclabelprops*

vclabelprops
Properties of the Vector colorbar label

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

vclabel, *vclabelsize*, *vclabelweight*

vclabelsize
Set the size of the Vector colorbar label

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

vclabel, *vclabelweight*, *vclabelprops*

vclabelweight
Set the fontweight of the Vector colorbar label

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

vclabel, *vclabelsize*, *vclabelprops*

tight
Automatically adjust the plots.

If set to True, the plots are automatically adjusted to fit to the figure limitations via the `matplotlib.pyplot.tight_layout()` function.

Possible types

bool – True for automatic adjustment

Warning: There is no update method to undo what happend after this formatoption is set to True!

bounds
Specify the boundaries of the colorbar

Possible types

- *None* – make no normalization
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero
- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.
- *matplotlib.colors.Normalize* – A matplotlib normalization instance

Examples

Plot 11 bounds over the whole data range:

```
>>> plotter.update(bounds='rounded')
```

Plot 7 ticks over the whole data range where the maximal and minimal tick matches the data maximum and minimum:

```
>>> plotter.update(bounds=['minmax', 7])
```

Plot logarithmic bounds:

```
>>> from matplotlib.colors import LogNorm
>>> plotter.update(bounds=LogNorm())
```

See also:

cmap Specifies the colormap

cbar
Specify the position of the colorbars

Possible types

- *bool* – True: defaults to ‘b’ False: Don’t draw any colorbar
- *str* – The string can be a combination of one of the following strings: {‘fr’, ‘fb’, ‘fl’, ‘ft’, ‘b’, ‘r’, ‘sv’, ‘sh’}
  - ‘b’, ‘r’ stand for bottom and right of the axes
  - ‘fr’, ‘fb’, ‘fl’, ‘ft’ stand for bottom, right, left and top of the figure
  - ‘sv’ and ‘sh’ stand for a vertical or horizontal colorbar in a separate figure
- *list* – A containing one of the above positions

Examples

Draw a colorbar at the bottom and left of the axes:

```
>>> plotter.update(cbar='bl')
```

cbarspacing
Specify the spacing of the bounds in the colorbar

Possible types

str {‘uniform’, ‘proportional’} – if `'uniform'`, every color has exactly the same width in the colorbar, if `'proportional'`, the size is chosen according to the data

cmap
Specify the color map

This formatoption specifies the color coding of the data via a `matplotlib.colors.Colormap`

Possible types

- *str* – Strings may be any valid colormap name suitable for the `matplotlib.cm.get_cmap()` function or one of the color lists defined in the ‘colors.cmaps’ key of the `psyplot.rcParams` dictionary (including their reversed color maps given via the ‘_r’ extension).
- *matplotlib.colors.Colormap* – The colormap instance to use

See also:

bounds specifies the boundaries of the colormap

color
Set the color for the arrows

This formatoption can be used to set a single color for the vectors or define the color coding

Possible types

- *float* – Determines the greyness
- *color* – Defines the same color for all arrows. The string can be either a html hex string (e.g. '#eeefff'), a single letter (e.g. 'b': blue, 'g': green, 'r': red, 'c': cyan, 'm': magenta, 'y': yellow, 'k': black, 'w': white) or any other color
- *string {'absolute', 'u', 'v'}* – Strings may define how the formatoption is calculated. Possible strings are
  - **absolute**: for the absolute wind speed
  - **u**: for the u component
  - **v**: for the v component
- *2D-array* – The values determine the color for each plotted arrow. Note that the shape has to match the one of u and v.

See also:

arrowsize, *arrowstyle*, *density*, *linewidth*

ctickprops
Specify the font properties of the colorbar ticklabels

Possible types

dict – Items may be anything of the `matplotlib.pyplot.tick_params()` function

See also:

cticksize, *ctickweight*, *cticklabels*, *cticks*, *vcticksize*, *vctickweight*, *vcticklabels*, *vcticks*

cticksize
Specify the font size of the colorbar ticklabels

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

ctickweight, *ctickprops*, *cticklabels*, *cticks*, *vctickweight*, *vctickprops*, *vcticklabels*, *vcticks*

ctickweight
Specify the fontweight of the colorbar ticklabels

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

cticksize, *ctickprops*, *cticklabels*, *cticks*, *vcticksize*, *vctickprops*, *vcticklabels*, *vcticks*

extend
Draw arrows at the side of the colorbar

Possible types

str {'neither', 'both', 'min' or 'max'} – If not 'neither', make pointed end(s) for out-of-range values

levels
The levels for the contour plot

This formatoption sets the levels for the filled contour plot and only has an effect if the *plot* Formatoption is set to `'contourf'`

Possible types

- *None* – Use the settings from the *bounds* formatoption and if this does not specify boundaries, use 11
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero

- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.

miss_color
Set the color for missing values

Possible types

- *None* – Use the default from the colormap
- *string, tuple.* – Defines the color of the grid.

vbounds
Specify the boundaries of the vector colorbar

Possible types

- *None* – make no normalization
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero
- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.
- *matplotlib.colors.Normalize* – A matplotlib normalization instance

---

Examples

Plot 11 bounds over the whole data range:

```
>>> plotter.update(bounds='rounded')
```

Plot 7 ticks over the whole data range where the maximal and minimal tick matches the data maximum and minimum:

```
>>> plotter.update(bounds=['minmax', 7])
```

Plot logarithmic bounds:

```
>>> from matplotlib.colors import LogNorm
>>> plotter.update(bounds=LogNorm())
```

See also:

cmap Specifies the colormap

vcbar
Specify the position of the vector plot colorbars

Possible types

- *bool* – True: defaults to 'b' False: Don't draw any colorbar
- *str* – The string can be a combination of one of the following strings: {'fr', 'fb', 'fl', 'ft', 'b', 'r', 'sv', 'sh'}
  - 'b', 'r' stand for bottom and right of the axes
  - 'fr', 'fb', 'fl', 'ft' stand for bottom, right, left and top of the figure
  - 'sv' and 'sh' stand for a vertical or horizontal colorbar in a separate figure
- *list* – A containing one of the above positions

vcbarspacing
Specify the spacing of the bounds in the colorbar

Possible types

str {'uniform', 'proportional'} – if `'uniform'`, every color has exactly the same width in the colorbar, if `'proportional'`, the size is chosen according to the data

vcmap
Specify the color map

This formatoption specifies the color coding of the data via a `matplotlib.colors.Colormap`

Possible types

- *str* – Strings may be any valid colormap name suitable for the `matplotlib.cm.get_cmap()` function or one of the color lists defined in the 'colors.cmaps' key of the `psyplot.rcParams` dictionary (including their reversed color maps given via the '_r' extension).
- *matplotlib.colors.Colormap* – The colormap instance to use

See also:

bounds specifies the boundaries of the colormap

vctickprops
Specify the font properties of the colorbar ticklabels

Possible types

dict – Items may be anything of the `matplotlib.pyplot.tick_params()` function

See also:

cticksize, *ctickweight*, *cticklabels*, *cticks*, *vcticksize*, *vctickweight*, *vcticklabels*, *vcticks*

vcticksize
Specify the font size of the colorbar ticklabels

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

ctickweight, *ctickprops*, *cticklabels*, *cticks*, *vctickweight*, *vctickprops*, *vcticklabels*, *vcticks*

vctickweight
Specify the fontweight of the colorbar ticklabels

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

cticksize, *ctickprops*, *cticklabels*, *cticks*, *vcticksize*, *vctickprops*, *vcticklabels*, *vcticks*

maskbetween
Mask data points between two numbers

Possible types

float – The floating number to mask above

See also:

maskless, *maskleq*, *maskgreater*, *maskgeq*

maskgeq
Mask data points greater than or equal to a number

Possible types

float – The floating number to mask above

See also:

maskless, *maskleq*, *maskgreater*, *maskbetween*

maskgreater
Mask data points greater than a number

Possible types

float – The floating number to mask above

See also:

maskless, *maskleq*, *maskgeq*, *maskbetween*

maskleq
Mask data points smaller than or equal to a number

Possible types

float – The floating number to mask below

See also:

maskless, *maskgreater*, *maskgeq*, *maskbetween*

maskless
Mask data points smaller than a number

Possible types

float – The floating number to mask below

See also:

maskleq, *maskgreater*, *maskgeq*, *maskbetween*

post
Apply your own postprocessing script

This formatoption let's you apply your own post processing script. Just enter the script as a string and it will be executed. The formatoption will be made available via the `self` variable

Possible types

- *None* – Don't do anything
- *str* – The post processing script as string

Note: This formatoption uses the built-in `exec()` function to compile the script. Since this poses a security risk when loading psyplot projects, it is by default disabled through the `Plotter.enable_post` attribute. If you are sure that you can trust the script in this formatoption, set this attribute of the corresponding `Plotter` to `True`

Examples

Assume, you want to manually add the mean of the data to the title of the matplotlib axes. You can simply do this via

```
from psyplot.plotter import Plotter
from xarray import DataArray
plotter = Plotter(DataArray([1, 2, 3]))
# enable the post formatoption
plotter.enable_post = True
plotter.update(post="self.ax.set_title(str(self.data.mean()))")
plotter.ax.get_title()
'2.0'
```

By default, the `post` formatoption is only ran, when it is explicitly updated. However, you can use the *post_timing* formatoption, to run it automatically. E.g. for running it after every update of the plotter, you can set

```
plotter.update(post_timing='always')
```

See also:

post_timing Determine the timing of this formatoption

post_timing
Determine when to run the *post* formatoption

This formatoption determines, whether the *post* formatoption should be run never, after replot or after every update.

Possible types

- *'never'* – Never run post processing scripts
- *'always'* – Always run post processing scripts
- *'replot'* – Only run post processing scripts when the data changes or a replot is necessary

See also:

post The post processing formatoption

class psy_maps.plotters.**FieldPlotter**(*data=None*, *ax=None*, *auto_update=None*, *project=None*, *draw=None*, *make_plot=True*, *clear=False*, *enable_post=False*, ***kwargs*)

Bases: psy_simple.plotters.Simple2DBase, *psy_maps.plotters.MapPlotter*, psy_simple.base.BasePlotter

Plotter for 2D scalar fields on a map

Parameters

- **data** (*InteractiveArray or ArrayList, optional*) – Data object that shall be visualized. If given and *plot* is True, the `initialize_plot()` method is called at the end. Otherwise you can call this method later by yourself
- **ax** (*matplotlib.axes.Axes*) – Matplotlib Axes to plot on. If None, a new one will be created as soon as the `initialize_plot()` method is called
- **auto_update** (*bool*) – Default: None. A boolean indicating whether this list shall automatically update the contained arrays when calling the `update()` method or not. See also

the `no_auto_update` attribute. If None, the value from the `'lists.auto_update'` key in the `psyplot.rcParams` dictionary is used.

- **draw** (*bool or None*) – Boolean to control whether the figure of this array shall be drawn at the end. If None, it defaults to the *'auto_draw'* parameter in the `psyplot.rcParams` dictionary
- **make_plot** (*bool*) – If True, and *data* is not None, the plot is initialized. Otherwise only the framework between plotter and data is set up
- **clear** (*bool*) – If True, the axes is cleared first
- **enable_post** (*bool*) – If True, the *post* formatoption is enabled and post processing scripts are allowed
- ****kwargs** – Any formatoption key from the `formatoptions` attribute that shall be used

Miscallaneous formatoptions

interp_bounds	Interpolate grid cell boundaries for 2D plots
clat	Set the center latitude of the plot
clip	Clip the part outside the latitudes of the map extent
clon	Set the center longitude of the plot
datagrid	Show the grid of the data
grid_color	Set the color of the grid
grid_labels	Display the labels of the grid
grid_labelsize	Modify the size of the grid tick labels
grid_settings	Modify the settings of the grid explicitly
lonlatbox	Set the longitude-latitude box of the data shown
lsm	Draw the continents
map_extent	Set the extent of the map
projection	Specify the projection for the plot
stock_img	Display a stock image on the map
transform	Specify the coordinate system of the data
xgrid	Draw vertical grid lines (meridians)
ygrid	Draw horizontal grid lines (parallels)

Color coding formatoptions

levels	The levels for the contour plot
bounds	Specify the boundaries of the colorbar
cbar	Specify the position of the colorbars
cbarspacing	Specify the spacing of the bounds in the colorbar
cmap	Specify the color map
ctickprops	Specify the font properties of the colorbar ticklabels
cticksize	Specify the font size of the colorbar ticklabels
ctickweight	Specify the fontweight of the colorbar ticklabels
extend	Draw arrows at the side of the colorbar
miss_color	Set the color for missing values

Plot formatoptions

plot	Choose how to visualize a 2-dimensional scalar data field

Label formatoptions

clabel	Show the colorbar label
clabelprops	Properties of the Colorbar label
clabelsize	Set the size of the Colorbar label
clabelweight	Set the fontweight of the Colorbar label
figtitle	Plot a figure title
figtitleprops	Properties of the figure title
figtitlesize	Set the size of the figure title
figtitleweight	Set the fontweight of the figure title
text	Add text anywhere on the plot
title	Show the title
titleprops	Properties of the title
titlesize	Set the size of the title
titleweight	Set the fontweight of the title

Axes formatoptions

tight	Automatically adjust the plots.

Masking formatoptions

maskbetween	Mask data points between two numbers
maskgeq	Mask data points greater than or equal to a number
maskgreater	Mask data points greater than a number
maskleq	Mask data points smaller than or equal to a number
maskless	Mask data points smaller than a number

Axis tick formatoptions

cticklabels	Specify the colorbar ticklabels
cticks	Specify the tick locations of the colorbar

Post processing formatoptions

post	Apply your own postprocessing script
post_timing	Determine when to run the `post` formatoption

`interp_bounds`

Interpolate grid cell boundaries for 2D plots

This formatoption can be used to tell enable and disable the interpolation of grid cell boundaries. Usually, netCDF files only contain the centered coordinates. In this case, we interpolate the boundaries between the grid cell centers.

Possible types

- *None* – Interpolate the boundaries, except for circumpolar grids
- *bool* – If True (the default), the grid cell boundaries are inter- and extrapolated. Otherwise, if False, the coordinate centers are used and the default behaviour of matplotlib cuts of the most outer row and column of the 2D-data. Note that this results in a slight shift of the data

levels
The levels for the contour plot

This formatoption sets the levels for the filled contour plot and only has an effect if the *plot* Formatoption is set to `'contourf'`

Possible types

- *None* – Use the settings from the *bounds* formatoption and if this does not specify boundaries, use 11
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero

- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.

plot
Choose how to visualize a 2-dimensional scalar data field

Possible types

- *None* – Don't make any plotting
- *'mesh'* – Use the `matplotlib.pyplot.pcolormesh()` function to make the plot or the `matplotlib.pyplot.tripcolor()` for an unstructered grid
- *'tri'* – Use the `matplotlib.pyplot.tripcolor()` function to plot data on a triangular grid
- *'contourf'* – Make a filled contour plot using the `matplotlib.pyplot.contourf()` function or the `matplotlib.pyplot.tricontourf()` for triangular data. The levels for the contour plot are controlled by the *levels* formatoption

- *'tricontourf'* – Make a filled contour plot using the `matplotlib.pyplot.tricontourf()` function

clat
Set the center latitude of the plot

Parameters

- **None** – Let the *lonlatbox* formatoption determine the center
- **float** – Specifiy the center manually
- **str** – A pattern that matches any of the keys in the `psyplot.rcParams` `'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)

clip
Clip the part outside the latitudes of the map extent

Possible types

- *None* – Clip if all longitudes are shown (i.e. the extent goes from -180 to 180) and the projection is orthographic or stereographic
- *bool* – True, clip, else, don't

Notes

If the plot is clipped. You might need to update with *replot=True*!

clon
Set the center longitude of the plot

Parameters

- **None** – Let the *lonlatbox* formatoption determine the center
- **float** – Specifiy the center manually
- **str** – A pattern that matches any of the keys in the `psyplot.rcParams` `'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)

datagrid
Show the grid of the data

This formatoption shows the grid of the data (without labels)

Possible types

- *None* – Don't show the data grid
- *str* – A linestyle in the form `'k-'`, where `'k'` is the color and `'-'` the linestyle.
- *dict* – any keyword arguments that are passed to the plotting function (`matplotlib.pyplot.triplot()` for triangular grids and `matplotlib.pyplot.hlines()` for rectilinear grids)

See also:

xgrid, *ygrid*

grid_color
Set the color of the grid

Possible types

- *None* – Choose the default line color
- *color* – Any valid color for matplotlib (see the `matplotlib.pyplot.plot()` documentation)

See also:

grid_settings, *grid_labels*, *grid_labelsize*, *xgrid*, *ygrid*

grid_labels
Display the labels of the grid

Possible types

- *None* – Grid labels are draw if possible
- *bool* – If True, labels are drawn and if this is not possible, a warning is raised

See also:

grid_color, *grid_settings*, *grid_labelsize*, *xgrid*, *ygrid*

grid_labelsize
Modify the size of the grid tick labels

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

grid_color, *grid_labels*, *xgrid*, *ygrid*, *grid_settings*

grid_settings
Modify the settings of the grid explicitly

Possible types

dict – Items may be any key-value-pair of the `matplotlib.collections.LineCollection` class

See also:

grid_color, *grid_labels*, *grid_labelsize*, *xgrid*, *ygrid*

lonlatbox
Set the longitude-latitude box of the data shown

This formatoption extracts the data that matches the specified box.

Possible types

- *None* – Use the full data
- *str* – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)
- *[lonmin, lonmax, latmin, latmax]* – The surrounding longitude-latitude that shall be used. Values can be either a float or a string as above

Notes

- For only specifying the region of the plot, see the *map_extent* formatoption
- If the coordinates are two-dimensional (e.g. for a circumpolar grid), than the data is not extracted but values outside the specified longitude-latitude box are set to NaN

See also:

map_extent

lsm
Draw the continents

Possible types

- *bool* – True: draw the continents with a line width of 1 False: don't draw the continents
- *float* – Specifies the linewidth of the continents
- *str* – The resolution of the land-sea mask (see the `cartopy.mpl.geoaxes.GeoAxesSubplot.coastlines()` method. Usually one of `('110m', '50m', '10m')`.
- *list [str or bool, float]* – The resolution and the linewidth

map_extent
Set the extent of the map

Possible types

- *None* – The map extent is specified by the data (i.e. by the *lonlatbox* formatoption)
- *'global'* – The whole globe is shown
- *str* – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)
- *[lonmin, lonmax, latmin, latmax]* – The surrounding longitude-latitude that shall be used. Values can be either a float or a string as above

Notes

This formatoption sets the extent of the plot. For choosing the region for the data, see the *lonlatbox* formatoption

See also:

lonlatbox

projection
Specify the projection for the plot

This formatoption defines the projection of the plot

Possible types

- *cartopy.crs.CRS* – A cartopy projection instance (e.g. cartopy.crs.PlateCarree)
- *str* – A string specifies the projection instance to use. The centered longitude and latitude are determined by the *clon* and *clat* formatoptions. Possible strings are (each standing for the specified projection)

cyl	cartopy.crs.PlateCarree
robin	cartopy.crs.Robinson
moll	cartopy.crs.Mollweide
geo	cartopy.crs.Geostationary
northpole	cartopy.crs.NorthPolarStereo
southpole	cartopy.crs.SouthPolarStereo
ortho	cartopy.crs.Orthographic
stereo	cartopy.crs.Stereographic
near	cartopy.crs.NearsidePerspective

Warning: An update of the projection clears the axes!

stock_img
Display a stock image on the map

This formatoption uses the cartopy.mpl.geoaxes.GeoAxes.stock_img() method to display a downsampled version of the Natural Earth shaded relief raster on the map

Possible types

bool – If True, the image is displayed

transform
Specify the coordinate system of the data

This formatoption defines the coordinate system of the data (usually we expect a simple latitude longitude coordinate system)

Possible types

- *cartopy.crs.CRS* – A cartopy projection instance (e.g. `cartopy.crs.PlateCarree`)
- *str* – A string specifies the projection instance to use. The centered longitude and latitude are determined by the *clon* and *clat* formatoptions. Possible strings are (each standing for the specified projection)

cyl	`cartopy.crs.PlateCarree`
robin	`cartopy.crs.Robinson`
moll	`cartopy.crs.Mollweide`
geo	`cartopy.crs.Geostationary`
northpole	`cartopy.crs.NorthPolarStereo`
southpole	`cartopy.crs.SouthPolarStereo`
ortho	`cartopy.crs.Orthographic`
stereo	`cartopy.crs.Stereographic`
near	`cartopy.crs.NearsidePerspective`

xgrid

Draw vertical grid lines (meridians)

This formatoption specifies at which longitudes to draw the meridians.

Possible types

- *None* – Don't draw gridlines (same as `False`)
- *bool* – True: draw gridlines and determine position automatically False: don't draw gridlines
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero

- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.

See also:

ygrid, *grid_color*, *grid_labels*

ygrid
Draw horizontal grid lines (parallels)

This formatoption specifies at which latitudes to draw the parallels.

Possible types

- *None* – Don't draw gridlines (same as `False`)
- *bool* – True: draw gridlines and determine position automatically False: don't draw gridlines
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero

- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.

See also:

xgrid, *grid_color*, *grid_labels*

bounds
Specify the boundaries of the colorbar

Possible types

- *None* – make no normalization
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

minmax Uses the minimum as minimal tick and maximum as maximal tick

sym Same as minmax but symmetric around zero

- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.
- *matplotlib.colors.Normalize* – A matplotlib normalization instance

---

Examples

Plot 11 bounds over the whole data range:

```
>>> plotter.update(bounds='rounded')
```

Plot 7 ticks over the whole data range where the maximal and minimal tick matches the data maximum and minimum:

```
>>> plotter.update(bounds=['minmax', 7])
```

Plot logarithmic bounds:

```
>>> from matplotlib.colors import LogNorm
>>> plotter.update(bounds=LogNorm())
```

See also:

cmap Specifies the colormap

cbar

Specify the position of the colorbars

Possible types

- *bool* – True: defaults to 'b' False: Don't draw any colorbar
- *str* – The string can be a combination of one of the following strings: {'fr', 'fb', 'fl', 'ft', 'b', 'r', 'sv', 'sh'}
  - 'b', 'r' stand for bottom and right of the axes
  - 'fr', 'fb', 'fl', 'ft' stand for bottom, right, left and top of the figure
  - 'sv' and 'sh' stand for a vertical or horizontal colorbar in a separate figure
- *list* – A containing one of the above positions

Examples

Draw a colorbar at the bottom and left of the axes:

```
>>> plotter.update(cbar='bl')
```

---

cbarspacing
Specify the spacing of the bounds in the colorbar

Possible types

str {'uniform', 'proportional'} – if `'uniform'`, every color has exactly the same width in the colorbar, if `'proportional'`, the size is chosen according to the data

cmap
Specify the color map

This formatoption specifies the color coding of the data via a `matplotlib.colors.Colormap`

Possible types

- *str* – Strings may be any valid colormap name suitable for the `matplotlib.cm.get_cmap()` function or one of the color lists defined in the 'colors.cmaps' key of the `psyplot.rcParams` dictionary (including their reversed color maps given via the '_r' extension).
- *matplotlib.colors.Colormap* – The colormap instance to use

See also:

bounds specifies the boundaries of the colormap

ctickprops
Specify the font properties of the colorbar ticklabels

Possible types

dict – Items may be anything of the `matplotlib.pyplot.tick_params()` function

See also:

cticksize, *ctickweight*, *cticklabels*, *cticks*, `vcticksize`, `vctickweight`, `vcticklabels`, `vcticks`

cticksize
Specify the font size of the colorbar ticklabels

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

ctickweight, *ctickprops*, *cticklabels*, *cticks*, `vctickweight`, `vctickprops`, `vcticklabels`, `vcticks`

ctickweight
Specify the fontweight of the colorbar ticklabels

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

cticksize, *ctickprops*, *cticklabels*, *cticks*, vcticksize, vctickprops, vcticklabels, vcticks

extend
Draw arrows at the side of the colorbar

Possible types

str {'neither', 'both', 'min' or 'max'} – If not 'neither', make pointed end(s) for out-of-range values

miss_color
Set the color for missing values

Possible types

- *None* – Use the default from the colormap
- *string, tuple.* – Defines the color of the grid.

clabel
Show the colorbar label

Set the label of the colorbar. You can insert any meta key from the `xarray.DataArray.attrs` via a string like `'%(key)s'`. Furthermore there are some special cases:

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.
- `'%(x)s'`, `'%(y)s'`, `'%(z)s'`, `'%(t)s'` will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)
- any attribute of one of the above coordinates is inserted via `axis + key` (e.g. the name of the x-coordinate can be inserted via `'%(xname)s'`).
- Labels defined in the `psyplot.rcParams 'texts.labels'` key are also replaced when enclosed by '{}'. The standard labels are
  - tinfo: `%H:%M`
  - dtinfo: `%B %d, %Y. %H:%M`
  - dinfo: `%B %d, %Y`
  - desc: `%(long_name)s [%(units)s]`
  - sdesc: `%(name)s [%(units)s]`

Possible types

str – The title for the `set_label()` method.

See also:

clabelsize, *clabelweight*, *clabelprops*

clabelprops
Properties of the Colorbar label

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

clabel, *clabelsize*, *clabelweight*

clabelsize
Set the size of the Colorbar label

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

clabel, *clabelweight*, *clabelprops*

clabelweight
Set the fontweight of the Colorbar label

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

clabel, *clabelsize*, *clabelprops*

figtitle
Plot a figure title

Set the title of the figure. You can insert any meta key from the `xarray.DataArray.attrs` via a string like `'%(key)s'`. Furthermore there are some special cases:

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.
- `'%(x)s'`, `'%(y)s'`, `'%(z)s'`, `'%(t)s'` will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)

- any attribute of one of the above coordinates is inserted via `axis + key` (e.g. the name of the x-coordinate can be inserted via `'%(xname)s'`).
- Labels defined in the `psyplot.rcParams 'texts.labels'` key are also replaced when enclosed by '{}'. The standard labels are
  - tinfo: `%H:%M`
  - dtinfo: `%B %d, %Y. %H:%M`
  - dinfo: `%B %d, %Y`
  - desc: `%(long_name)s [%(units)s]`
  - sdesc: `%(name)s [%(units)s]`

Possible types

str – The title for the `suptitle()` function

Notes

- If the plotter is part of a `psyplot.project.Project` and multiple plotters of this project are on the same figure, the replacement attributes (see above) are joined by a delimiter. If the `delimiter` attribute of this `Figtitle` instance is not None, it will be used. Otherwise the rcParams['texts.delimiter'] item is used.
- This is the title of the whole figure! For the title of this specific subplot, see the *title* formatoption.

See also:

title, *figtitlesize*, *figtitleweight*, *figtitleprops*

figtitleprops
Properties of the figure title

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

figtitle, *figtitlesize*, *figtitleweight*

figtitlesize
Set the size of the figure title

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

figtitle, *figtitleweight*, *figtitleprops*

figtitleweight
Set the fontweight of the figure title

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

figtitle, *figtitlesize*, *figtitleprops*

text
Add text anywhere on the plot

This formatoption draws a text on the specified position on the figure. You can insert any meta key from the `xarray.DataArray.attrs` via a string like `'%(key)s'`. Furthermore there are some special cases:

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.
- `'%(x)s'`, `'%(y)s'`, `'%(z)s'`, `'%(t)s'` will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)
- any attribute of one of the above coordinates is inserted via `axis + key` (e.g. the name of the x-coordinate can be inserted via `'%(xname)s'`).
- Labels defined in the `psyplot.rcParams 'texts.labels'` key are also replaced when enclosed by '{}'. The standard labels are
  - tinfo: `%H:%M`
  - dtinfo: `%B %d, %Y. %H:%M`
  - dinfo: `%B %d, %Y`
  - desc: `%(long_name)s [%(units)s]`
  - sdesc: `%(name)s [%(units)s]`

Possible types

- *str* – If string s: this will be used as (1., 1., s, {'ha': 'right'}) (i.e. a string in the upper right corner of the axes).
- *tuple or list of tuples (x,y,s[,coord.-system][,options])* – Each tuple defines a text instance on the plot. 0<=x, y<=1 are the coordinates. The coord.-system can be either the data coordinates (default, `'data'`) or the axes coordinates (`'axes'`) or the figure coordinates ('fig'). The string s finally is the text. options may be a dictionary to specify format the appearence (e.g. `'color'`, `'fontweight'`, `'fontsize'`, etc., see `matplotlib.text.Text` for possible keys). To remove one single text from the plot, set (x,y,'[, coord.-system]) for the text at position (x,y)
- *empty list* – remove all texts from the plot

See also:

title, *figtitle*

title
Show the title

Set the title of the plot. You can insert any meta key from the `xarray.DataArray.attrs` via a string like `'%(key)s'`. Furthermore there are some special cases:

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.
- `'%(x)s'`, `'%(y)s'`, `'%(z)s'`, `'%(t)s'` will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)
- any attribute of one of the above coordinates is inserted via `axis + key` (e.g. the name of the x-coordinate can be inserted via `'%(xname)s'`).
- Labels defined in the `psyplot.rcParams 'texts.labels'` key are also replaced when enclosed by '{}'. The standard labels are
  - tinfo: `%H:%M`
  - dtinfo: `%B %d, %Y. %H:%M`
  - dinfo: `%B %d, %Y`
  - desc: `%(long_name)s [%(units)s]`
  - sdesc: `%(name)s [%(units)s]`

Possible types

str – The title for the `title()` function.

Notes

This is the title of this specific subplot! For the title of the whole figure, see the *figtitle* formatoption.

See also:

figtitle, *titlesize*, *titleweight*, *titleprops*

titleprops
Properties of the title

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

title, *titlesize*, *titleweight*

titlesize
Set the size of the title

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

title, *titleweight*, *titleprops*

titleweight
Set the fontweight of the title

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

title, *titlesize*, *titleprops*

tight
Automatically adjust the plots.

If set to True, the plots are automatically adjusted to fit to the figure limitations via the `matplotlib.pyplot.tight_layout()` function.

Possible types

bool – True for automatic adjustment

> **Warning:** There is no update method to undo what happend after this formatoption is set to True!

maskbetween
Mask data points between two numbers

Possible types

float – The floating number to mask above

See also:

maskless, *maskleq*, *maskgreater*, *maskgeq*

maskgeq
Mask data points greater than or equal to a number

Possible types

float – The floating number to mask above

See also:

maskless, *maskleq*, *maskgreater*, *maskbetween*

maskgreater
Mask data points greater than a number

Possible types

float – The floating number to mask above

See also:

maskless, *maskleq*, *maskgeq*, *maskbetween*

maskleq
Mask data points smaller than or equal to a number

Possible types

float – The floating number to mask below

See also:

maskless, *maskgreater*, *maskgeq*, *maskbetween*

maskless
Mask data points smaller than a number

Possible types

float – The floating number to mask below

See also:

maskleq, *maskgreater*, *maskgeq*, *maskbetween*

cticklabels
Specify the colorbar ticklabels

Possible types

- *str* – A formatstring like `'%Y'` for plotting the year (in the case that time is shown on the axis) or '%i' for integers
- *array* – An array of strings to use for the ticklabels

See also:

cticks, *ctickslize*, *ctickweight*, *ctickprops*, `vcticks`, `vcticksize`, `vctickweight`, `vctickprops`

cticks
Specify the tick locations of the colorbar

Possible types

- *None* – use the default ticks
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero

  bounds let the *bounds* keyword determine the ticks. An additional integer *i* may be specified to only use every i-th bound as a tick (see also *int* below)

- *int* – Specifies how many ticks to use with the `'bounds'` option. I.e. if integer `i`, then this is the same as `['bounds', i]`.

See also:

cticklabels

post
Apply your own postprocessing script

This formatoption let's you apply your own post processing script. Just enter the script as a string and it will be executed. The formatoption will be made available via the `self` variable

Possible types

- *None* – Don't do anything
- *str* – The post processing script as string

Note: This formatoption uses the built-in `exec()` function to compile the script. Since this poses a security risk when loading psyplot projects, it is by default disabled through the `Plotter.enable_post` attribute. If you are sure that you can trust the script in this formatoption, set this attribute of the corresponding `Plotter` to `True`

Examples

Assume, you want to manually add the mean of the data to the title of the matplotlib axes. You can simply do this via

```
from psyplot.plotter import Plotter
from xarray import DataArray
plotter = Plotter(DataArray([1, 2, 3]))
# enable the post formatoption
plotter.enable_post = True
plotter.update(post="self.ax.set_title(str(self.data.mean()))")
plotter.ax.get_title()
'2.0'
```

By default, the `post` formatoption is only ran, when it is explicitly updated. However, you can use the *post_timing* formatoption, to run it automatically. E.g. for running it after every update of the plotter, you can set

```
plotter.update(post_timing='always')
```

See also:

post_timing Determine the timing of this formatoption

post_timing
Determine when to run the *post* formatoption

This formatoption determines, whether the *post* formatoption should be run never, after replot or after every update.

Possible types

- *'never'* – Never run post processing scripts
- *'always'* – Always run post processing scripts
- *'replot'* – Only run post processing scripts when the data changes or a replot is necessary

See also:

post The post processing formatoption

class psy_maps.plotters.**GridBase**(**args*, ***kwargs*)
Bases: psy_simple.plotters.DataTicksCalculator

Abstract base class for x- and y- grid lines

Possible types

Attributes

axis	The axis string

Continued on next page

Table 32 – continued from previous page

connections	list() -> new empty list
dependencies	list() -> new empty list
grid_color	grid_color Formatoption instance in the plotter
grid_labels	grid_labels Formatoption instance in the plotter
grid_settings	grid_settings Formatoption instance in the plotter
lonlatbox	lonlatbox Formatoption instance in the plotter
map_extent	map_extent Formatoption instance in the plotter
plot	plot Formatoption instance in the plotter
projection	projection Formatoption instance in the plotter
transform	transform Formatoption instance in the plotter

Methods

get_kwargs(loc)	
remove()	Method to remove the effects of this formatoption
update(value)	Method that is call to update the formatoption on the axes

- *None* – Don't draw gridlines (same as `False`)
- *bool* – True: draw gridlines and determine position automatically False: don't draw gridlines
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero

- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.

See also:

grid_color, *grid_labels*

axis
The axis string

connections = ['plot']

dependencies = ['transform', 'grid_labels', 'grid_color', 'grid_settings', 'projection

get_kwargs(*loc*)

grid_color
grid_color Formatoption instance in the plotter

grid_labels
grid_labels Formatoption instance in the plotter

grid_settings
grid_settings Formatoption instance in the plotter

lonlatbox
lonlatbox Formatoption instance in the plotter

map_extent
map_extent Formatoption instance in the plotter

plot
plot Formatoption instance in the plotter

projection
projection Formatoption instance in the plotter

remove()
Method to remove the effects of this formatoption

This method is called when the axes is cleared due to a formatoption with `requires_clearing` set to True. You don't necessarily have to implement this formatoption if your plot results are removed by the usual `matplotlib.axes.Axes.clear()` method.

transform
transform Formatoption instance in the plotter

update(*value*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

class psy_maps.plotters.**GridColor**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: psyplot.plotter.Formatoption

Set the color of the grid

Possible types

Attributes

connections	list() -> new empty list
name	str(object='') -> str
xgrid	xgrid Formatoption instance in the plotter
ygrid	ygrid Formatoption instance in the plotter

Methods

update(value)	Method that is call to update the formatoption on the axes

- *None* – Choose the default line color

- *color* – Any valid color for matplotlib (see the `matplotlib.pyplot.plot()` documentation)

See also:

`grid_settings`, `grid_labels`, `grid_labelsize`, *`xgrid`*, *`ygrid`*

Parameters

- **key** (*`str`*) – formatoption key in the *plotter*
- **plotter** (*`psyplot.plotter.Plotter`*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*`int or None`*) – The index that shall be used if the data is a `psyplot.InteractiveList`
- **additional_children** (*`list or str`*) – Additional children to use (see the `children` attribute)
- **additional_dependencies** (*`list or str`*) – Additional dependencies to use (see the `dependencies` attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the `children`, `dependencies` and *`connections`* attributes, with values being the name of the new formatoption in this plotter.

connections = ['xgrid', 'ygrid']

name = 'Color of the latitude-longitude grid'

update(*value*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

xgrid
xgrid Formatoption instance in the plotter

ygrid
ygrid Formatoption instance in the plotter

class psy_maps.plotters.**GridLabelSize**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: `psyplot.plotter.Formatoption`

Modify the size of the grid tick labels

Possible types

Attributes

`dependencies`	list() -> new empty list
`name`	str(object='') -> str
`xgrid`	xgrid Formatoption instance in the plotter
`ygrid`	ygrid Formatoption instance in the plotter

Methods

update(value)	Method that is call to update the formatoption on the axes

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

grid_color, grid_labels, *xgrid*, *ygrid*, grid_settings

Parameters

- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a psyplot.InteractiveList
- **additional_children** (*list or str*) – Additional children to use (see the children attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the *dependencies* attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the children, *dependencies* and connections attributes, with values being the name of the new formatoption in this plotter.

dependencies = ['xgrid', 'ygrid']

name = 'Label size of the latitude-longitude grid'

update(*value*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

xgrid
xgrid Formatoption instance in the plotter

ygrid
ygrid Formatoption instance in the plotter

class psy_maps.plotters.**GridLabels**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)
Bases: psyplot.plotter.Formatoption

Display the labels of the grid

Possible types

Attributes

connections	list() -> new empty list
dependencies	list() -> new empty list
name	str(object='') -> str
projection	projection Formatoption instance in the plotter
transform	transform Formatoption instance in the plotter
xgrid	xgrid Formatoption instance in the plotter
ygrid	ygrid Formatoption instance in the plotter

Methods

update(value)	Method that is call to update the formatoption on the axes

- *None* – Grid labels are draw if possible
- *bool* – If True, labels are drawn and if this is not possible, a warning is raised

See also:

`grid_color`, `grid_settings`, `grid_labelsize`, *xgrid*, *ygrid*

Parameters

- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a `psyplot.InteractiveList`
- **additional_children** (*list or str*) – Additional children to use (see the `children` attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the *dependencies* attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the `children`, *dependencies* and *connections* attributes, with values being the name of the new formatoption in this plotter.

connections = ['xgrid', 'ygrid']

dependencies = ['projection', 'transform']

name = 'Labels of the latitude-longitude grid'

projection
projection Formatoption instance in the plotter

transform
transform Formatoption instance in the plotter

update(*value*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

xgrid
xgrid Formatoption instance in the plotter

ygrid
ygrid Formatoption instance in the plotter

class psy_maps.plotters.**GridSettings**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: psyplot.plotter.DictFormatoption

Modify the settings of the grid explicitly

Possible types

Attributes

children	list() -> new empty list
connections	list() -> new empty list
grid_color	grid_color Formatoption instance in the plotter
grid_labels	grid_labels Formatoption instance in the plotter
name	str(object='') -> str
xgrid	xgrid Formatoption instance in the plotter
ygrid	ygrid Formatoption instance in the plotter

Methods

set_value(value[, validate, todefault])	Set (and validate) the value in the plotter
update(value)	Method that is call to update the formatoption on the axes

dict – Items may be any key-value-pair of the matplotlib.collections.LineCollection class

See also:

grid_color, *grid_labels*, grid_labelsize, *xgrid*, *ygrid*

Parameters

- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a psyplot.InteractiveList
- **additional_children** (*list or str*) – Additional children to use (see the *children* attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the dependencies attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the *children*, dependencies and *connections* attributes, with values being the name of the new formatoption in this plotter.

children = ['grid_labels', 'grid_color']

connections = ['xgrid', 'ygrid']

grid_color
grid_color Formatoption instance in the plotter

grid_labels
grid_labels Formatoption instance in the plotter

name = 'Line properties of the latitude-longitude grid'

set_value(*value*, *validate=True*, *todefault=False*)
Set (and validate) the value in the plotter

Parameters

- **value** – Value to set
- **validate** (*bool*) – if True, validate the *value* before it is set
- **todefault** (*bool*) – True if the value is updated to the default value

Notes

- If the current value in the plotter is None, then it will be set with the given *value*, otherwise the current value in the plotter is updated
- If the value is an empty dictionary, the value in the plotter is cleared

update(*value*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

xgrid
xgrid Formatoption instance in the plotter

ygrid
ygrid Formatoption instance in the plotter

class psy_maps.plotters.**LSM**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)
Bases: psyplot.plotter.Formatoption

Draw the continents

Possible types

Attributes

name	str(object='') -> str

Methods

remove()	Method to remove the effects of this formatoption
update(value)	Method that is call to update the formatoption on the axes

- *bool* – True: draw the continents with a line width of 1 False: don't draw the continents
- *float* – Specifies the linewidth of the continents
- *str* – The resolution of the land-sea mask (see the `cartopy.mpl.geoaxes.GeoAxesSubplot.coastlines()` method. Usually one of `('110m', '50m', '10m')`.
- *list [str or bool, float]* – The resolution and the linewidth

Parameters

- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a `psyplot.InteractiveList`
- **additional_children** (*list or str*) – Additional children to use (see the `children` attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the `dependencies` attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the `children`, `dependencies` and `connections` attributes, with values being the name of the new formatoption in this plotter.

lsm = None

name = 'Land-Sea mask'

remove()
Method to remove the effects of this formatoption

This method is called when the axes is cleared due to a formatoption with `requires_clearing` set to True. You don't necessarily have to implement this formatoption if your plot results are removed by the usual `matplotlib.axes.Axes.clear()` method.

update(*value*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

class psy_maps.plotters.**LonLatBox**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)
Bases: *psy_maps.plotters.BoxBase*

Set the longitude-latitude box of the data shown

This formatoption extracts the data that matches the specified box.

Possible types

Methods

calc_lonlatbox(lon, lat)	

Continued on next page

Table 44 – continued from previous page

data_dependent(data[, set_data])	bool(x) -> bool
mask_outside(data, lon, lat, lonmin, lonmax, ...)	
shiftdata(lonsin, datain, lon_0)	Shift the data such that it matches the region we want to show
to_degree([units])	Converts arrays with radian units to degree
update(value)	Method that is call to update the formatoption on the axes
update_array(value, data, decoder[, base_var])	Update the given *data* array

Attributes

dependencies	list() -> new empty list
lonlatbox_transformed	
name	str(object='') -> str
priority	int(x=0) -> integer
requires_clearing	bool(x) -> bool
transform	transform Formatoption instance in the plotter

- *None* – Use the full data
- *str* – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)
- *[lonmin, lonmax, latmin, latmax]* – The surrounding longitude-latitude that shall be used. Values can be either a float or a string as above

Notes

- For only specifying the region of the plot, see the `map_extent` formatoption
- If the coordinates are two-dimensional (e.g. for a circumpolar grid), than the data is not extracted but values outside the specified longitude-latitude box are set to NaN

See also:

`map_extent`

Parameters

- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a `psyplot.InteractiveList`
- **additional_children** (*list or str*) – Additional children to use (see the `children` attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the *dependencies* attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords

may be anything of the `children`, *dependencies* and `connections` attributes, with values being the name of the new formatoption in this plotter.

calc_lonlatbox(*lon*, *lat*)

data_dependent(*data*, *set_data=True*)
bool(x) -> bool

Returns True when the argument x is true, False otherwise. The builtins True and False are the only two instances of the class bool. The class bool is a subclass of the class int, and cannot be subclassed.

dependencies = ['transform']

lonlatbox_transformed

mask_outside(*data*, *lon*, *lat*, *lonmin*, *lonmax*, *latmin*, *latmax*, *is_unstructured=False*)

name = 'Longitude-Latitude box of the data'

priority = 30

requires_clearing = True

shiftdata(*lonsin*, *datain*, *lon_0*)
Shift the data such that it matches the region we want to show

Parameters **%(shiftdata.parameters)s** –

Notes

datain can also be multiple fields stored in a three-dimensional array. Then we shift all fields along the first dimension

to_degree(*units=None*, **args*)
Converts arrays with radian units to degree

Parameters

- **units** (*str*) – if `'radian'`, the arrays in `*args` will be converted
- ***args** – numpy arrays

Returns returns the arrays provided with `*args`

Return type list of np.ndarray

Notes

if *units* is `'radian'`, a copy of the array will be returned

transform
transform Formatoption instance in the plotter

update(*value*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

update_array(*value*, *data*, *decoder*, *base_var=None*)
Update the given *data* array

class psy_maps.plotters.**MapDataGrid**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: psy_simple.plotters.DataGrid

Show the grid of the data

This formatoption shows the grid of the data (without labels)

Possible types

Attributes

transform	transform Formatoption instance in the plotter
triangles	The matplotlib.tri.Triangulation instance containing the

- *None* – Don't show the data grid
- *str* – A linestyle in the form `'k-'`, where `'k'` is the color and `'-'` the linestyle.
- *dict* – any keyword arguments that are passed to the plotting function (matplotlib.pyplot.triplot() for triangular grids and matplotlib.pyplot.hlines() for rectilinear grids)

See also:

xgrid, ygrid

Parameters

- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a psyplot.InteractiveList
- **additional_children** (*list or str*) – Additional children to use (see the children attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the dependencies attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the children, dependencies and connections attributes, with values being the name of the new formatoption in this plotter.

transform
transform Formatoption instance in the plotter

triangles
The matplotlib.tri.Triangulation instance containing the spatial informations

class psy_maps.plotters.**MapDensity**(**args*, ***kwargs*)

Bases: psy_simple.plotters.Density

Change the density of the arrows

Possible types

Attributes

plot	plot Formatoption instance in the plotter

- *float* – Scales the density of the arrows in x- and y-direction (1.0 means no scaling)
- *tuple (x, y)* – Defines the scaling in x- and y-direction manually

plot
plot Formatoption instance in the plotter

class psy_maps.plotters.**MapExtent**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: *psy_maps.plotters.BoxBase*

Set the extent of the map

Possible types

Attributes

dependencies	list() -> new empty list
lonlatbox	lonlatbox Formatoption instance in the plotter
name	str(object='') -> str
plot	plot Formatoption instance in the plotter
priority	int(x=0) -> integer
update_after_plot	bool(x) -> bool
vplot	vplot Formatoption instance in the plotter

Methods

update(value)	Method that is call to update the formatoption on the axes

- *None* – The map extent is specified by the data (i.e. by the *lonlatbox* formatoption)
- *'global'* – The whole globe is shown
- *str* – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)
- *[lonmin, lonmax, latmin, latmax]* – The surrounding longitude-latitude that shall be used. Values can be either a float or a string as above

Notes

This formatoption sets the extent of the plot. For choosing the region for the data, see the *lonlatbox* formatoption

See also:

lonlatbox

Parameters

- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a psyplot.InteractiveList
- **additional_children** (*list or str*) – Additional children to use (see the children attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the *dependencies* attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the children, *dependencies* and connections attributes, with values being the name of the new formatoption in this plotter.

dependencies = ['lonlatbox', 'plot', 'vplot']

lonlatbox
lonlatbox Formatoption instance in the plotter

name = 'Longitude-Latitude box of the plot'

plot
plot Formatoption instance in the plotter

priority = 10

update(*value*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

update_after_plot = True

vplot
vplot Formatoption instance in the plotter

class psy_maps.plotters.**MapPlot2D**(**args*, ***kwargs*)
Bases: psy_simple.plotters.Plot2D

Choose how to visualize a 2-dimensional scalar data field

Possible types

Methods

add2format_coord(x, y)	Additional information for the format_coord()
remove(*args, **kwargs)	Method to remove the effects of this formatoption

Attributes

array	The (masked) data array that is plotted
bounds	bounds Formatoption instance in the plotter
clip	clip Formatoption instance in the plotter
cmap	cmap Formatoption instance in the plotter
connections	list() -> new empty list
data_dependent	bool(x) -> bool
dependencies	list() -> new empty list
interp_bounds	interp_bounds Formatoption instance in the plotter
levels	levels Formatoption instance in the plotter
lonlatbox	lonlatbox Formatoption instance in the plotter
transform	transform Formatoption instance in the plotter

- *None* – Don't make any plotting
- *'mesh'* – Use the `matplotlib.pyplot.pcolormesh()` function to make the plot or the `matplotlib.pyplot.tripcolor()` for an unstructered grid
- *'tri'* – Use the `matplotlib.pyplot.tripcolor()` function to plot data on a triangular grid
- *'contourf'* – Make a filled contour plot using the `matplotlib.pyplot.contourf()` function or the `matplotlib.pyplot.tricontourf()` for triangular data. The levels for the contour plot are controlled by the *levels* formatoption
- *'tricontourf'* – Make a filled contour plot using the `matplotlib.pyplot.tricontourf()` function

add2format_coord(*x*, *y*)
Additional information for the `format_coord()`

array
The (masked) data array that is plotted

bounds
bounds Formatoption instance in the plotter

clip
clip Formatoption instance in the plotter

cmap
cmap Formatoption instance in the plotter

connections = ['transform', 'lonlatbox']

data_dependent = True

dependencies = ['levels', 'interp_bounds', 'clip']

interp_bounds
interp_bounds Formatoption instance in the plotter

levels
levels Formatoption instance in the plotter

lonlatbox
lonlatbox Formatoption instance in the plotter

remove(**args*, ***kwargs*)
Method to remove the effects of this formatoption

This method is called when the axes is cleared due to a formatoption with `requires_clearing` set to True. You don't necessarily have to implement this formatoption if your plot results are removed by the usual `matplotlib.axes.Axes.clear()` method.

transform
transform Formatoption instance in the plotter

class psy_maps.plotters.**MapPlotter**(*data=None*, *ax=None*, *auto_update=None*, *project=None*, *draw=None*, *make_plot=True*, *clear=False*, *enable_post=False*, ***kwargs*)

Bases: `psy_simple.plotters.Base2D`

Base plotter for visualizing data on a map

Parameters

- **data** (*InteractiveArray or ArrayList, optional*) – Data object that shall be visualized. If given and *plot* is True, the `initialize_plot()` method is called at the end. Otherwise you can call this method later by yourself
- **ax** (*matplotlib.axes.Axes*) – Matplotlib Axes to plot on. If None, a new one will be created as soon as the `initialize_plot()` method is called
- **auto_update** (*bool*) – Default: None. A boolean indicating whether this list shall automatically update the contained arrays when calling the `update()` method or not. See also the `no_auto_update` attribute. If None, the value from the `'lists.auto_update'` key in the `psyplot.rcParams` dictionary is used.
- **draw** (*bool or None*) – Boolean to control whether the figure of this array shall be drawn at the end. If None, it defaults to the *'auto_draw'* parameter in the `psyplot.rcParams` dictionary
- **make_plot** (*bool*) – If True, and *data* is not None, the plot is initialized. Otherwise only the framework between plotter and data is set up
- **clear** (*bool*) – If True, the axes is cleared first
- **enable_post** (*bool*) – If True, the *post* formatoption is enabled and post processing scripts are allowed
- ****kwargs*** – Any formatoption key from the `formatoptions` attribute that shall be used

Attributes

ax	Axes instance of the plot
convert_radian	Boolean that is True if triangles with units in radian should be

Miscallaneous formatoptions

clat	Set the center latitude of the plot
clip	Clip the part outside the latitudes of the map extent
clon	Set the center longitude of the plot
datagrid	Show the grid of the data
grid_color	Set the color of the grid
grid_labels	Display the labels of the grid
grid_labelsize	Modify the size of the grid tick labels
grid_settings	Modify the settings of the grid explicitly
lonlatbox	Set the longitude-latitude box of the data shown

Continued on next page

Table 53 – continued from previous page

lsm	Draw the continents
map_extent	Set the extent of the map
projection	Specify the projection for the plot
stock_img	Display a stock image on the map
transform	Specify the coordinate system of the data
xgrid	Draw vertical grid lines (meridians)
ygrid	Draw horizontal grid lines (parallels)

Label formatoptions

clabel	Show the colorbar label
clabelprops	Properties of the Colorbar label
clabelsize	Set the size of the Colorbar label
clabelweight	Set the fontweight of the Colorbar label

Color coding formatoptions

bounds	Specify the boundaries of the colorbar
cbar	Specify the position of the colorbars
cbarspacing	Specify the spacing of the bounds in the colorbar
cmap	Specify the color map
ctickprops	Specify the font properties of the colorbar ticklabels
cticksize	Specify the font size of the colorbar ticklabels
ctickweight	Specify the fontweight of the colorbar ticklabels
extend	Draw arrows at the side of the colorbar

Axis tick formatoptions

cticklabels	Specify the colorbar ticklabels
cticks	Specify the tick locations of the colorbar

Post processing formatoptions

post	Apply your own postprocessing script
post_timing	Determine when to run the `post` formatoption

ax
Axes instance of the plot

clat
Set the center latitude of the plot

Parameters

- **None** – Let the *lonlatbox* formatoption determine the center
- **float** – Specifiy the center manually
- **str** – A pattern that matches any of the keys in the `psyplot.rcParams` `'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)

clip
Clip the part outside the latitudes of the map extent

Possible types

- *None* – Clip if all longitudes are shown (i.e. the extent goes from -180 to 180) and the projection is orthographic or stereographic
- *bool* – True, clip, else, don't

Notes

If the plot is clipped. You might need to update with *replot*=*True*!

clon
Set the center longitude of the plot

Parameters

- **None** – Let the *lonlatbox* formatoption determine the center
- **float** – Specifiy the center manually
- **str** – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)

convert_radian = True
Boolean that is True if triangles with units in radian should be converted to degrees

datagrid
Show the grid of the data

This formatoption shows the grid of the data (without labels)

Possible types

- *None* – Don't show the data grid
- *str* – A linestyle in the form `'k-'`, where `'k'` is the color and `'-'` the linestyle.
- *dict* – any keyword arguments that are passed to the plotting function (`matplotlib.pyplot.triplot()` for triangular grids and `matplotlib.pyplot.hlines()` for rectilinear grids)

See also:

xgrid, *ygrid*

grid_color
Set the color of the grid

Possible types

- *None* – Choose the default line color
- *color* – Any valid color for matplotlib (see the `matplotlib.pyplot.plot()` documentation)

See also:

grid_settings, *grid_labels*, *grid_labelsize*, *xgrid*, *ygrid*

grid_labels
Display the labels of the grid

Possible types

- *None* – Grid labels are draw if possible
- *bool* – If True, labels are drawn and if this is not possible, a warning is raised

See also:

grid_color, *grid_settings*, *grid_labelsize*, *xgrid*, *ygrid*

grid_labelsize
Modify the size of the grid tick labels

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

grid_color, *grid_labels*, *xgrid*, *ygrid*, *grid_settings*

grid_settings
Modify the settings of the grid explicitly

Possible types

dict – Items may be any key-value-pair of the `matplotlib.collections.LineCollection` class

See also:

grid_color, *grid_labels*, *grid_labelsize*, *xgrid*, *ygrid*

lonlatbox
Set the longitude-latitude box of the data shown

This formatoption extracts the data that matches the specified box.

Possible types

- *None* – Use the full data
- *str* – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)
- *[lonmin, lonmax, latmin, latmax]* – The surrounding longitude-latitude that shall be used. Values can be either a float or a string as above

Notes

- For only specifying the region of the plot, see the *map_extent* formatoption
- If the coordinates are two-dimensional (e.g. for a circumpolar grid), than the data is not extracted but values outside the specified longitude-latitude box are set to NaN

See also:

map_extent

lsm
Draw the continents

Possible types

- *bool* – True: draw the continents with a line width of 1 False: don't draw the continents
- *float* – Specifies the linewidth of the continents
- *str* – The resolution of the land-sea mask (see the `cartopy.mpl.geoaxes.GeoAxesSubplot.coastlines()` method. Usually one of `('110m', '50m', '10m')`.
- *list [str or bool, float]* – The resolution and the linewidth

map_extent
Set the extent of the map

Possible types

- *None* – The map extent is specified by the data (i.e. by the *lonlatbox* formatoption)
- *'global'* – The whole globe is shown
- *str* – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)
- *[lonmin, lonmax, latmin, latmax]* – The surrounding longitude-latitude that shall be used. Values can be either a float or a string as above

Notes

This formatoption sets the extent of the plot. For choosing the region for the data, see the *lonlatbox* formatoption

See also:

lonlatbox

projection
Specify the projection for the plot

This formatoption defines the projection of the plot

Possible types

- *cartopy.crs.CRS* – A cartopy projection instance (e.g. `cartopy.crs.PlateCarree`)
- *str* – A string specifies the projection instance to use. The centered longitude and latitude are determined by the *clon* and *clat* formatoptions. Possible strings are (each standing for the specified projection)

cyl	`cartopy.crs.PlateCarree`
robin	`cartopy.crs.Robinson`
moll	`cartopy.crs.Mollweide`
geo	`cartopy.crs.Geostationary`
northpole	`cartopy.crs.NorthPolarStereo`
southpole	`cartopy.crs.SouthPolarStereo`
ortho	`cartopy.crs.Orthographic`
stereo	`cartopy.crs.Stereographic`
near	`cartopy.crs.NearsidePerspective`

Warning: An update of the projection clears the axes!

stock_img
Display a stock image on the map

This formatoption uses the `cartopy.mpl.geoaxes.GeoAxes.stock_img()` method to display a downsampled version of the Natural Earth shaded relief raster on the map

Possible types

bool – If True, the image is displayed

transform
Specify the coordinate system of the data

This formatoption defines the coordinate system of the data (usually we expect a simple latitude longitude coordinate system)

Possible types

- *cartopy.crs.CRS* – A cartopy projection instance (e.g. `cartopy.crs.PlateCarree`)
- *str* – A string specifies the projection instance to use. The centered longitude and latitude are determined by the *clon* and *clat* formatoptions. Possible strings are (each standing for the specified projection)

cyl	`cartopy.crs.PlateCarree`
robin	`cartopy.crs.Robinson`
moll	`cartopy.crs.Mollweide`
geo	`cartopy.crs.Geostationary`
northpole	`cartopy.crs.NorthPolarStereo`
southpole	`cartopy.crs.SouthPolarStereo`
ortho	`cartopy.crs.Orthographic`
stereo	`cartopy.crs.Stereographic`
near	`cartopy.crs.NearsidePerspective`

xgrid

Draw vertical grid lines (meridians)

This formatoption specifies at which longitudes to draw the meridians.

Possible types

- *None* – Don't draw gridlines (same as `False`)
- *bool* – True: draw gridlines and determine position automatically False: don't draw gridlines
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero
- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.

See also:

ygrid, *grid_color*, *grid_labels*

ygrid

Draw horizontal grid lines (parallels)

This formatoption specifies at which latitudes to draw the parallels.

Possible types

- *None* – Don't draw gridlines (same as `False`)

- *bool* – True: draw gridlines and determine position automatically False: don't draw gridlines
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero

- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.

See also:

xgrid, *grid_color*, *grid_labels*

clabel

Show the colorbar label

Set the label of the colorbar. You can insert any meta key from the `xarray.DataArray.attrs` via a string like `'%(key)s'`. Furthermore there are some special cases:

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.
- `'%(x)s'`, `'%(y)s'`, `'%(z)s'`, `'%(t)s'` will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)
- any attribute of one of the above coordinates is inserted via `axis + key` (e.g. the name of the x-coordinate can be inserted via `'%(xname)s'`).
- Labels defined in the `psyplot.rcParams 'texts.labels'` key are also replaced when enclosed by '{}'. The standard labels are
  - tinfo: `%H:%M`
  - dtinfo: `%B %d, %Y. %H:%M`
  - dinfo: `%B %d, %Y`
  - desc: `%(long_name)s [%(units)s]`
  - sdesc: `%(name)s [%(units)s]`

Possible types

str – The title for the `set_label()` method.

See also:

clabelsize, *clabelweight*, *clabelprops*

clabelprops
Properties of the Colorbar label

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

clabel, *clabelsize*, *clabelweight*

clabelsize
Set the size of the Colorbar label

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

clabel, *clabelweight*, *clabelprops*

clabelweight
Set the fontweight of the Colorbar label

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

clabel, *clabelsize*, *clabelprops*

bounds
Specify the boundaries of the colorbar

Possible types

- *None* – make no normalization
- *numeric array* – specifies the ticks manually
- *str or list [str, . . .]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

mid plot the ticks in the middle of the data.

rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

minmax Uses the minimum as minimal tick and maximum as maximal tick

sym Same as minmax but symmetric around zero

- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.
- *matplotlib.colors.Normalize* – A matplotlib normalization instance

---

Examples

Plot 11 bounds over the whole data range:

```
>>> plotter.update(bounds='rounded')
```

Plot 7 ticks over the whole data range where the maximal and minimal tick matches the data maximum and minimum:

```
>>> plotter.update(bounds=['minmax', 7])
```

Plot logarithmic bounds:

```
>>> from matplotlib.colors import LogNorm
>>> plotter.update(bounds=LogNorm())
```

See also:

cmap Specifies the colormap

cbar

Specify the position of the colorbars

Possible types

- *bool* – True: defaults to ‘b’ False: Don’t draw any colorbar
- *str* – The string can be a combination of one of the following strings: {‘fr’, ‘fb’, ‘fl’, ‘ft’, ‘b’, ‘r’, ‘sv’, ‘sh’}
  - ‘b’, ‘r’ stand for bottom and right of the axes
  - ‘fr’, ‘fb’, ‘fl’, ‘ft’ stand for bottom, right, left and top of the figure
  - ‘sv’ and ‘sh’ stand for a vertical or horizontal colorbar in a separate figure
- *list* – A containing one of the above positions

Examples

Draw a colorbar at the bottom and left of the axes:

```
>>> plotter.update(cbar='bl')
```

cbarspacing
Specify the spacing of the bounds in the colorbar

Possible types

str {'uniform', 'proportional'} – if `'uniform'`, every color has exactly the same width in the colorbar, if `'proportional'`, the size is chosen according to the data

cmap
Specify the color map

This formatoption specifies the color coding of the data via a `matplotlib.colors.Colormap`

Possible types

- *str* – Strings may be any valid colormap name suitable for the `matplotlib.cm.get_cmap()` function or one of the color lists defined in the 'colors.cmaps' key of the `psyplot.rcParams` dictionary (including their reversed color maps given via the '_r' extension).
- *matplotlib.colors.Colormap* – The colormap instance to use

See also:

bounds specifies the boundaries of the colormap

ctickprops
Specify the font properties of the colorbar ticklabels

Possible types

dict – Items may be anything of the `matplotlib.pyplot.tick_params()` function

See also:

cticksize, *ctickweight*, *cticklabels*, *cticks*, `vcticksize`, `vctickweight`, `vcticklabels`, `vcticks`

cticksize
Specify the font size of the colorbar ticklabels

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

ctickweight, *ctickprops*, *cticklabels*, *cticks*, vctickweight, vctickprops, vcticklabels, vcticks

ctickweight
Specify the fontweight of the colorbar ticklabels

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

cticksize, *ctickprops*, *cticklabels*, *cticks*, vcticksize, vctickprops, vcticklabels, vcticks

extend
Draw arrows at the side of the colorbar

Possible types

str {'neither', 'both', 'min' or 'max'} – If not 'neither', make pointed end(s) for out-of-range values

cticklabels
Specify the colorbar ticklabels

Possible types

- *str* – A formatstring like `'%Y'` for plotting the year (in the case that time is shown on the axis) or '%i' for integers
- *array* – An array of strings to use for the ticklabels

See also:

cticks, *cticksize*, *ctickweight*, *ctickprops*, vcticks, vcticksize, vctickweight, vctickprops

cticks
Specify the tick locations of the colorbar

Possible types

- *None* – use the default ticks
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

mid plot the ticks in the middle of the data.

rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

minmax Uses the minimum as minimal tick and maximum as maximal tick

sym Same as minmax but symmetric around zero

bounds let the *bounds* keyword determine the ticks. An additional integer *i* may be specified to only use every i-th bound as a tick (see also *int* below)

- *int* – Specifies how many ticks to use with the `'bounds'` option. I.e. if integer `i`, then this is the same as `['bounds', i]`.

See also:

cticklabels

post

Apply your own postprocessing script

This formatoption let's you apply your own post processing script. Just enter the script as a string and it will be executed. The formatoption will be made available via the `self` variable

Possible types

- *None* – Don't do anything
- *str* – The post processing script as string

Note: This formatoption uses the built-in `exec()` function to compile the script. Since this poses a security risk when loading psyplot projects, it is by default disabled through the `Plotter.enable_post` attribute. If you are sure that you can trust the script in this formatoption, set this attribute of the corresponding `Plotter` to `True`

Examples

Assume, you want to manually add the mean of the data to the title of the matplotlib axes. You can simply do this via

```
from psyplot.plotter import Plotter
from xarray import DataArray
plotter = Plotter(DataArray([1, 2, 3]))
# enable the post formatoption
plotter.enable_post = True
plotter.update(post="self.ax.set_title(str(self.data.mean()))")
plotter.ax.get_title()
'2.0'
```

By default, the `post` formatoption is only ran, when it is explicitly updated. However, you can use the *post_timing* formatoption, to run it automatically. E.g. for running it after every update of the plotter, you can set

```
plotter.update(post_timing='always')
```

See also:

post_timing Determine the timing of this formatoption

post_timing
Determine when to run the *post* formatoption

This formatoption determines, whether the *post* formatoption should be run never, after replot or after every update.

Possible types

- *'never'* – Never run post processing scripts
- *'always'* – Always run post processing scripts
- *'replot'* – Only run post processing scripts when the data changes or a replot is necessary

See also:

post The post processing formatoption

class psy_maps.plotters.**MapVectorColor**(**args*, ***kwargs*)
Bases: psy_simple.plotters.VectorColor

Set the color for the arrows

This formatoption can be used to set a single color for the vectors or define the color coding

Possible types

Attributes

bounds	bounds Formatoption instance in the plotter
cmap	cmap Formatoption instance in the plotter
plot	plot Formatoption instance in the plotter
transpose	transpose Formatoption instance in the plotter

- *float* – Determines the greyness
- *color* – Defines the same color for all arrows. The string can be either a html hex string (e.g. '#eeefff'), a single letter (e.g. 'b': blue, 'g': green, 'r': red, 'c': cyan, 'm': magenta, 'y': yellow, 'k': black, 'w': white) or any other color
- *string {'absolute', 'u', 'v'}* – Strings may define how the formatoption is calculated. Possible strings are
  - **absolute**: for the absolute wind speed
  - **u**: for the u component

  - **v**: for the v component
- *2D-array* – The values determine the color for each plotted arrow. Note that the shape has to match the one of u and v.

See also:

`arrowsize`, `arrowstyle`, `density`, `linewidth`

bounds
bounds Formatoption instance in the plotter

cmap
cmap Formatoption instance in the plotter

plot
plot Formatoption instance in the plotter

transpose
transpose Formatoption instance in the plotter

class psy_maps.plotters.**MapVectorPlot**(**args*, ***kwargs*)
Bases: psy_simple.plotters.VectorPlot

Choose the vector plot type

Possible types

Methods

add2format_coord(x, y)	Additional information for the `format_coord()`
set_value(value, *args, **kwargs)	Set (and validate) the value in the plotter.

Attributes

arrowsize	arrowsize Formatoption instance in the plotter
arrowstyle	arrowstyle Formatoption instance in the plotter
bounds	bounds Formatoption instance in the plotter
clat	clat Formatoption instance in the plotter
clip	clip Formatoption instance in the plotter
clon	clon Formatoption instance in the plotter
cmap	cmap Formatoption instance in the plotter
color	color Formatoption instance in the plotter
data_dependent	bool(x) -> bool
density	density Formatoption instance in the plotter
dependencies	list() -> new empty list
linewidth	linewidth Formatoption instance in the plotter
lonlatbox	lonlatbox Formatoption instance in the plotter
transform	transform Formatoption instance in the plotter
transpose	transpose Formatoption instance in the plotter

str – Plot types can be either

quiver to make a quiver plot

stream to make a stream plot

add2format_coord(*x*, *y*)
Additional information for the `format_coord()`

arrowsize
arrowsize Formatoption instance in the plotter

arrowstyle
arrowstyle Formatoption instance in the plotter

bounds
bounds Formatoption instance in the plotter

clat
clat Formatoption instance in the plotter

clip
clip Formatoption instance in the plotter

clon
clon Formatoption instance in the plotter

cmap
cmap Formatoption instance in the plotter

color
color Formatoption instance in the plotter

data_dependent = True

density
density Formatoption instance in the plotter

dependencies = ['lonlatbox', 'transform', 'clon', 'clat', 'clip']

linewidth
linewidth Formatoption instance in the plotter

lonlatbox
lonlatbox Formatoption instance in the plotter

set_value(*value*, **args*, ***kwargs*)
Set (and validate) the value in the plotter. This method is called by the plotter when it attempts to change the value of the formatoption.

Parameters

- **value** – Value to set
- **validate** (*bool*) – if True, validate the *value* before it is set
- **todefault** (*bool*) – True if the value is updated to the default value

transform
transform Formatoption instance in the plotter

transpose
transpose Formatoption instance in the plotter

class psy_maps.plotters.**Projection**(**args*, ***kwargs*)
Bases: *psy_maps.plotters.ProjectionBase*

Specify the projection for the plot

This formatoption defines the projection of the plot

Possible types

Attributes

clat	clat Formatoption instance in the plotter
clon	clon Formatoption instance in the plotter
dependencies	list() -> new empty list
name	str(object='') -> str
priority	int(x=0) -> integer
requires_clearing	an update of this formatoption requires that the axes is cleared

Methods

initialize_plot(value[, clear])	Initialize the plot and set the projection for the axes
update(value)	Update the formatoption

- *cartopy.crs.CRS* – A cartopy projection instance (e.g. `cartopy.crs.PlateCarree`)
- *str* – A string specifies the projection instance to use. The centered longitude and latitude are determined by the *clon* and *clat* formatoptions. Possible strings are (each standing for the specified projection)

cyl	`cartopy.crs.PlateCarree`
robin	`cartopy.crs.Robinson`
moll	`cartopy.crs.Mollweide`
geo	`cartopy.crs.Geostationary`
northpole	`cartopy.crs.NorthPolarStereo`
southpole	`cartopy.crs.SouthPolarStereo`
ortho	`cartopy.crs.Orthographic`
stereo	`cartopy.crs.Stereographic`
near	`cartopy.crs.NearsidePerspective`

Warning: An update of the projection clears the axes!

clat
clat Formatoption instance in the plotter

clon
clon Formatoption instance in the plotter

dependencies = ['clon', 'clat']

initialize_plot(*value*, *clear=True*)
Initialize the plot and set the projection for the axes

name = 'Projection of the plot'

priority = 30

requires_clearing = True
an update of this formatoption requires that the axes is cleared

update(*value*)
Update the formatoption

Since this formatoption requires clearing, this method does nothing. Everything is done in the *initialize_plot()* method.

class psy_maps.plotters.**ProjectionBase**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: psyplot.plotter.Formatoption

Base class for formatoptions that uses cartopy.crs.CRS instances

Possible types

Methods

get_kwargs(value[, clon, clat])	
set_projection(value, *args, **kwargs)	

Attributes

projection_kwargs	dict() -> new empty dictionary
projections	dict() -> new empty dictionary

- *cartopy.crs.CRS* – A cartopy projection instance (e.g. cartopy.crs.PlateCarree)
- *str* – A string specifies the projection instance to use. The centered longitude and latitude are determined by the clon and clat formatoptions. Possible strings are (each standing for the specified projection)

cyl	cartopy.crs.PlateCarree
robin	cartopy.crs.Robinson
moll	cartopy.crs.Mollweide
geo	cartopy.crs.Geostationary
northpole	cartopy.crs.NorthPolarStereo
southpole	cartopy.crs.SouthPolarStereo
ortho	cartopy.crs.Orthographic
stereo	cartopy.crs.Stereographic
near	cartopy.crs.NearsidePerspective

Parameters

- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a psyplot.InteractiveList
- **additional_children** (*list or str*) – Additional children to use (see the children attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the dependencies attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords

may be anything of the `children`, `dependencies` and `connections` attributes, with values being the name of the new formatoption in this plotter.

get_kwargs(*value*, *clon=None*, *clat=None*)

projection_kwargs = {'cyl': ['central_longitude'], 'geo': ['central_longitude'], 'mo

projections = {'cyl': <class 'cartopy.crs.PlateCarree'>, 'geo': <class 'cartopy.crs.

set_projection(*value*, **args*, ***kwargs*)

class psy_maps.plotters.**StockImage**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: `psyplot.plotter.Formatoption`

Display a stock image on the map

This formatoption uses the `cartopy.mpl.geoaxes.GeoAxes.stock_img()` method to display a downsampled version of the Natural Earth shaded relief raster on the map

Possible types

Attributes

connections	list() -> new empty list
name	str(object='') -> str
plot	plot Formatoption instance in the plotter
priority	int(x=0) -> integer

Methods

remove()	Method to remove the effects of this formatoption
update(value)	Method that is call to update the formatoption on the axes

bool – If True, the image is displayed

Parameters

- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a `psyplot.InteractiveList`
- **additional_children** (*list or str*) – Additional children to use (see the `children` attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the `dependencies` attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the `children`, `dependencies` and *connections* attributes, with values being the name of the new formatoption in this plotter.

connections = ['plot']

image = None

name = 'Display Natural Earth shaded relief raster'

plot
plot Formatoption instance in the plotter

priority = 10

remove()
Method to remove the effects of this formatoption

This method is called when the axes is cleared due to a formatoption with `requires_clearing` set to True. You don't necessarily have to implement this formatoption if your plot results are removed by the usual `matplotlib.axes.Axes.clear()` method.

update(*value*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

class psy_maps.plotters.**Transform**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)
Bases: *psy_maps.plotters.ProjectionBase*

Specify the coordinate system of the data

This formatoption defines the coordinate system of the data (usually we expect a simple latitude longitude coordinate system)

Possible types

Attributes

connections	list() -> new empty list
name	str(object='') -> str
plot	plot Formatoption instance in the plotter
priority	int(x=0) -> integer
vplot	vplot Formatoption instance in the plotter

Methods

update(value)	Method that is call to update the formatoption on the axes

- *cartopy.crs.CRS* – A cartopy projection instance (e.g. `cartopy.crs.PlateCarree`)
- *str* – A string specifies the projection instance to use. The centered longitude and latitude are determined by the `clon` and `clat` formatoptions. Possible strings are (each standing for the specified projection)

cyl	cartopy.crs.PlateCarree
robin	cartopy.crs.Robinson
moll	cartopy.crs.Mollweide
geo	cartopy.crs.Geostationary
northpole	cartopy.crs.NorthPolarStereo
southpole	cartopy.crs.SouthPolarStereo
ortho	cartopy.crs.Orthographic
stereo	cartopy.crs.Stereographic
near	cartopy.crs.NearsidePerspective

Parameters

- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a psyplot.InteractiveList
- **additional_children** (*list or str*) – Additional children to use (see the children attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the dependencies attribute)
- ****kwargs** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the children, dependencies and *connections* attributes, with values being the name of the new formatoption in this plotter.

connections = ['plot', 'vplot']

name = 'Coordinate system of the data'

plot
plot Formatoption instance in the plotter

priority = 30

update(*value*)
Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

vplot
vplot Formatoption instance in the plotter

class psy_maps.plotters.**VectorPlotter**(*data=None*, *ax=None*, *auto_update=None*, *project=None*, *draw=None*, *make_plot=True*, *clear=False*, *enable_post=False*, ***kwargs*)

Bases: *psy_maps.plotters.MapPlotter*, psy_simple.plotters.BaseVectorPlotter, psy_simple.base.BasePlotter

Plotter for visualizing 2-dimensional vector data on a map

See also:

psyplot.plotter.simple.SimpleVectorPlotter for a simple version of drawing vector data

FieldPlotter for plotting scaler fields

CombinedPlotter for combined scalar and vector fields

Color coding formatoptions

color	Set the color for the arrows
bounds	Specify the boundaries of the vector colorbar
cbar	Specify the position of the vector plot colorbars
cbarspacing	Specify the spacing of the bounds in the colorbar
cmap	Specify the color map
ctickprops	Specify the font properties of the colorbar ticklabels
cticksize	Specify the font size of the colorbar ticklabels
ctickweight	Specify the fontweight of the colorbar ticklabels
extend	Draw arrows at the side of the colorbar

Vector plot formatoptions

density	Change the density of the arrows
arrowsize	Change the size of the arrows
arrowstyle	Change the style of the arrows

Plot formatoptions

plot	Choose the vector plot type

Miscallaneous formatoptions

clat	Set the center latitude of the plot
clip	Clip the part outside the latitudes of the map extent
clon	Set the center longitude of the plot
datagrid	Show the grid of the data
grid_color	Set the color of the grid
grid_labels	Display the labels of the grid
grid_labelsize	Modify the size of the grid tick labels
grid_settings	Modify the settings of the grid explicitly
linewidth	Change the linewidth of the arrows
lonlatbox	Set the longitude-latitude box of the data shown
lsm	Draw the continents
map_extent	Set the extent of the map
projection	Specify the projection for the plot
stock_img	Display a stock image on the map
transform	Specify the coordinate system of the data
xgrid	Draw vertical grid lines (meridians)
ygrid	Draw horizontal grid lines (parallels)

Label formatoptions

clabel	Show the colorbar label
clabelprops	Properties of the Colorbar label
clabelsize	Set the size of the Colorbar label
clabelweight	Set the fontweight of the Colorbar label

Continued on next page

Table 73 – continued from previous page

figtitle	Plot a figure title
figtitleprops	Properties of the figure title
figtitlesize	Set the size of the figure title
figtitleweight	Set the fontweight of the figure title
text	Add text anywhere on the plot
title	Show the title
titleprops	Properties of the title
titlesize	Set the size of the title
titleweight	Set the fontweight of the title

Axes formatoptions

tight	Automatically adjust the plots.

Masking formatoptions

maskbetween	Mask data points between two numbers
maskgeq	Mask data points greater than or equal to a number
maskgreater	Mask data points greater than a number
maskleq	Mask data points smaller than or equal to a number
maskless	Mask data points smaller than a number

Axis tick formatoptions

cticklabels	Specify the colorbar ticklabels
cticks	Specify the tick locations of the vector colorbar

Post processing formatoptions

post	Apply your own postprocessing script
post_timing	Determine when to run the `post` formatoption

Parameters

- **data** (*InteractiveArray or ArrayList, optional*) – Data object that shall be visualized. If given and *plot* is True, the `initialize_plot()` method is called at the end. Otherwise you can call this method later by yourself
- **ax** (*matplotlib.axes.Axes*) – Matplotlib Axes to plot on. If None, a new one will be created as soon as the `initialize_plot()` method is called
- **auto_update** (*bool*) – Default: None. A boolean indicating whether this list shall automatically update the contained arrays when calling the `update()` method or not. See also the `no_auto_update` attribute. If None, the value from the `'lists.auto_update'` key in the `psyplot.rcParams` dictionary is used.
- **draw** (*bool or None*) – Boolean to control whether the figure of this array shall be drawn at the end. If None, it defaults to the *'auto_draw'* parameter in the `psyplot.rcParams` dictionary
- **make_plot** (*bool*) – If True, and *data* is not None, the plot is initialized. Otherwise only the framework between plotter and data is set up

- **clear** (*bool*) – If True, the axes is cleared first
- **enable_post** (*bool*) – If True, the *post* formatoption is enabled and post processing scripts are allowed
- ****kwargs** – Any formatoption key from the `formatoptions` attribute that shall be used

color
Set the color for the arrows

This formatoption can be used to set a single color for the vectors or define the color coding

Possible types

- *float* – Determines the greyness
- *color* – Defines the same color for all arrows. The string can be either a html hex string (e.g. '#eeefff'), a single letter (e.g. 'b': blue, 'g': green, 'r': red, 'c': cyan, 'm': magenta, 'y': yellow, 'k': black, 'w': white) or any other color
- *string {'absolute', 'u', 'v'}* – Strings may define how the formatoption is calculated. Possible strings are
  - **absolute**: for the absolute wind speed
  - **u**: for the u component
  - **v**: for the v component
- *2D-array* – The values determine the color for each plotted arrow. Note that the shape has to match the one of u and v.

See also:

arrowsize, *arrowstyle*, *density*, *linewidth*

density
Change the density of the arrows

Possible types

- *float* – Scales the density of the arrows in x- and y-direction (1.0 means no scaling)
- *tuple (x, y)* – Defines the scaling in x- and y-direction manually

plot
Choose the vector plot type

Possible types

str – Plot types can be either

quiver to make a quiver plot

stream to make a stream plot

bounds
Specify the boundaries of the vector colorbar

Possible types

- *None* – make no normalization
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero
- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.
- *matplotlib.colors.Normalize* – A matplotlib normalization instance

---

Examples

Plot 11 bounds over the whole data range:

```
>>> plotter.update(bounds='rounded')
```

Plot 7 ticks over the whole data range where the maximal and minimal tick matches the data maximum and minimum:

```
>>> plotter.update(bounds=['minmax', 7])
```

Plot logarithmic bounds:

```
>>> from matplotlib.colors import LogNorm
>>> plotter.update(bounds=LogNorm())
```

See also:

cmap Specifies the colormap

cbar

Specify the position of the vector plot colorbars

Possible types

- *bool* – True: defaults to 'b' False: Don't draw any colorbar

- *str* – The string can be a combination of one of the following strings: {'fr', 'fb', 'fl', 'ft', 'b', 'r', 'sv', 'sh'}
  - 'b', 'r' stand for bottom and right of the axes
  - 'fr', 'fb', 'fl', 'ft' stand for bottom, right, left and top of the figure
  - 'sv' and 'sh' stand for a vertical or horizontal colorbar in a separate figure
- *list* – A containing one of the above positions

cbarspacing
Specify the spacing of the bounds in the colorbar

Possible types

str {'uniform', 'proportional'} – if `'uniform'`, every color has exactly the same width in the colorbar, if `'proportional'`, the size is chosen according to the data

cmap
Specify the color map

This formatoption specifies the color coding of the data via a `matplotlib.colors.Colormap`

Possible types

- *str* – Strings may be any valid colormap name suitable for the `matplotlib.cm.get_cmap()` function or one of the color lists defined in the 'colors.cmaps' key of the `psyplot.rcParams` dictionary (including their reversed color maps given via the '_r' extension).
- *matplotlib.colors.Colormap* – The colormap instance to use

See also:

bounds specifies the boundaries of the colormap

ctickprops
Specify the font properties of the colorbar ticklabels

Possible types

dict – Items may be anything of the `matplotlib.pyplot.tick_params()` function

See also:

ctickssize, *ctickweight*, *cticklabels*, *cticks*, vcticksize, vctickweight, vcticklabels, vcticks

cticksize
Specify the font size of the colorbar ticklabels

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

ctickweight, *ctickprops*, *cticklabels*, *cticks*, vctickweight, vctickprops, vcticklabels, vcticks

ctickweight
Specify the fontweight of the colorbar ticklabels

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

cticksize, *ctickprops*, *cticklabels*, *cticks*, vcticksize, vctickprops, vcticklabels, vcticks

extend
Draw arrows at the side of the colorbar

Possible types

str {'neither', 'both', 'min' or 'max'} – If not 'neither', make pointed end(s) for out-of-range values

arrowsize
Change the size of the arrows

Possible types

- *None* – make no scaling
- *float* – Factor scaling the size of the arrows

See also:

arrowstyle, *linewidth*, *density*, *color*

arrowstyle
Change the style of the arrows

Possible types

str – Any arrow style string (see FancyArrowPatch)

Notes

This formatoption only has an effect for stream plots

See also:

arrowsize, *linewidth*, *density*, *color*

clat
Set the center latitude of the plot

Parameters

- **None** – Let the *lonlatbox* formatoption determine the center
- **float** – Specifiy the center manually
- **str** – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)

clip
Clip the part outside the latitudes of the map extent

Possible types

- *None* – Clip if all longitudes are shown (i.e. the extent goes from -180 to 180) and the projection is orthographic or stereographic
- *bool* – True, clip, else, don't

Notes

If the plot is clipped. You might need to update with *replot=True*!

clon
Set the center longitude of the plot

Parameters

- **None** – Let the *lonlatbox* formatoption determine the center
- **float** – Specifiy the center manually
- **str** – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)

datagrid
Show the grid of the data

This formatoption shows the grid of the data (without labels)

Possible types

- *None* – Don't show the data grid

- *str* – A linestyle in the form `'k-'`, where `'k'` is the color and `'-'` the linestyle.
- *dict* – any keyword arguments that are passed to the plotting function (`matplotlib.pyplot.triplot()` for triangular grids and `matplotlib.pyplot.hlines()` for rectilinear grids)

See also:

xgrid, *ygrid*

grid_color
Set the color of the grid

Possible types

- *None* – Choose the default line color
- *color* – Any valid color for matplotlib (see the `matplotlib.pyplot.plot()` documentation)

See also:

grid_settings, *grid_labels*, *grid_labelsize*, *xgrid*, *ygrid*

grid_labels
Display the labels of the grid

Possible types

- *None* – Grid labels are draw if possible
- *bool* – If True, labels are drawn and if this is not possible, a warning is raised

See also:

grid_color, *grid_settings*, *grid_labelsize*, *xgrid*, *ygrid*

grid_labelsize
Modify the size of the grid tick labels

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

grid_color, *grid_labels*, *xgrid*, *ygrid*, *grid_settings*

grid_settings
Modify the settings of the grid explicitly

Possible types

dict – Items may be any key-value-pair of the `matplotlib.collections.LineCollection` class

See also:

grid_color, *grid_labels*, *grid_labelsize*, *xgrid*, *ygrid*

linewidth
Change the linewidth of the arrows

Possible types

- *float* – give the linewidth explicitly
- *string {'absolute', 'u', 'v'}* – Strings may define how the formatoption is calculated. Possible strings are
  - **absolute**: for the absolute wind speed
  - **u**: for the u component
  - **v**: for the v component
- *tuple (string, float)* – *string* may be one of the above strings, *float* may be a scaling factor
- *2D-array* – The values determine the linewidth for each plotted arrow. Note that the shape has to match the one of u and v.

See also:

arrowsize, *arrowstyle*, *density*, *color*

lonlatbox
Set the longitude-latitude box of the data shown

This formatoption extracts the data that matches the specified box.

Possible types

- *None* – Use the full data
- *str* – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)
- *[lonmin, lonmax, latmin, latmax]* – The surrounding longitude-latitude that shall be used. Values can be either a float or a string as above

Notes

- For only specifying the region of the plot, see the *map_extent* formatoption
- If the coordinates are two-dimensional (e.g. for a circumpolar grid), than the data is not extracted but values outside the specified longitude-latitude box are set to NaN

See also:

map_extent

lsm
Draw the continents

Possible types

- *bool* – True: draw the continents with a line width of 1 False: don't draw the continents
- *float* – Specifies the linewidth of the continents
- *str* – The resolution of the land-sea mask (see the `cartopy.mpl.geoaxes.GeoAxesSubplot.coastlines()` method. Usually one of `('110m', '50m', '10m')`.
- *list [str or bool, float]* – The resolution and the linewidth

map_extent
Set the extent of the map

Possible types

- *None* – The map extent is specified by the data (i.e. by the *lonlatbox* formatoption)
- *'global'* – The whole globe is shown
- *str* – A pattern that matches any of the keys in the `psyplot.rcParams 'extents.boxes'` item (contains user-defined longitude-latitude boxes) or the `psyplot.plotter.boxes.lonlatboxes` dictionary (contains longitude-latitude boxes of different countries and continents)
- *[lonmin, lonmax, latmin, latmax]* – The surrounding longitude-latitude that shall be used. Values can be either a float or a string as above

Notes

This formatoption sets the extent of the plot. For choosing the region for the data, see the *lonlatbox* formatoption

See also:

lonlatbox

projection
Specify the projection for the plot

This formatoption defines the projection of the plot

Possible types

- *cartopy.crs.CRS* – A cartopy projection instance (e.g. `cartopy.crs.PlateCarree`)
- *str* – A string specifies the projection instance to use. The centered longitude and latitude are determined by the *clon* and *clat* formatoptions. Possible strings are (each standing for the specified projection)

cyl	`cartopy.crs.PlateCarree`
robin	`cartopy.crs.Robinson`
moll	`cartopy.crs.Mollweide`
geo	`cartopy.crs.Geostationary`
northpole	`cartopy.crs.NorthPolarStereo`
southpole	`cartopy.crs.SouthPolarStereo`
ortho	`cartopy.crs.Orthographic`
stereo	`cartopy.crs.Stereographic`
near	`cartopy.crs.NearsidePerspective`

Warning: An update of the projection clears the axes!

stock_img
Display a stock image on the map

This formatoption uses the `cartopy.mpl.geoaxes.GeoAxes.stock_img()` method to display a downsampled version of the Natural Earth shaded relief raster on the map

Possible types

bool – If True, the image is displayed

transform
Specify the coordinate system of the data

This formatoption defines the coordinate system of the data (usually we expect a simple latitude longitude coordinate system)

Possible types

- *cartopy.crs.CRS* – A cartopy projection instance (e.g. `cartopy.crs.PlateCarree`)
- *str* – A string specifies the projection instance to use. The centered longitude and latitude are determined by the *clon* and *clat* formatoptions. Possible strings are (each standing for the specified projection)

cyl	`cartopy.crs.PlateCarree`
robin	`cartopy.crs.Robinson`
moll	`cartopy.crs.Mollweide`
geo	`cartopy.crs.Geostationary`
northpole	`cartopy.crs.NorthPolarStereo`
southpole	`cartopy.crs.SouthPolarStereo`
ortho	`cartopy.crs.Orthographic`
stereo	`cartopy.crs.Stereographic`
near	`cartopy.crs.NearsidePerspective`

xgrid
Draw vertical grid lines (meridians)

This formatoption specifies at which longitudes to draw the meridians.

Possible types

- *None* – Don't draw gridlines (same as `False`)
- *bool* – True: draw gridlines and determine position automatically False: don't draw gridlines
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero
- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.

See also:

ygrid, *grid_color*, *grid_labels*

ygrid

Draw horizontal grid lines (parallels)

This formatoption specifies at which latitudes to draw the parallels.

Possible types

- *None* – Don't draw gridlines (same as `False`)
- *bool* – True: draw gridlines and determine position automatically False: don't draw gridlines
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

minmax Uses the minimum as minimal tick and maximum as maximal tick

sym Same as minmax but symmetric around zero

- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.

See also:

xgrid, *grid_color*, *grid_labels*

clabel
Show the colorbar label

Set the label of the colorbar. You can insert any meta key from the `xarray.DataArray.attrs` via a string like `'%(key)s'`. Furthermore there are some special cases:

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.
- `'%(x)s'`, `'%(y)s'`, `'%(z)s'`, `'%(t)s'` will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)
- any attribute of one of the above coordinates is inserted via `axis + key` (e.g. the name of the x-coordinate can be inserted via `'%(xname)s'`).
- Labels defined in the `psyplot.rcParams 'texts.labels'` key are also replaced when enclosed by '{}'. The standard labels are
  - tinfo: `%H:%M`
  - dtinfo: `%B %d, %Y. %H:%M`
  - dinfo: `%B %d, %Y`
  - desc: `%(long_name)s [%(units)s]`
  - sdesc: `%(name)s [%(units)s]`

Possible types

str – The title for the `set_label()` method.

See also:

clabelsize, *clabelweight*, *clabelprops*

clabelprops
Properties of the Colorbar label

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

clabel, *clabelsize*, *clabelweight*

clabelsize
Set the size of the Colorbar label

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be ‘xx-small’, ‘x-small’, ‘small’, ‘medium’, ‘large’, ‘x-large’, ‘xx-large’.

See also:

clabel, *clabelweight*, *clabelprops*

clabelweight
Set the fontweight of the Colorbar label

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of ‘ultralight’, ‘light’, ‘normal’, ‘regular’, ‘book’, ‘medium’, ‘roman’, ‘semibold’, ‘demibold’, ‘demi’, ‘bold’, ‘heavy’, ‘extra bold’, ‘black’.

See also:

clabel, *clabelsize*, *clabelprops*

figtitle
Plot a figure title

Set the title of the figure. You can insert any meta key from the `xarray.DataArray.attrs` via a string like `'%(key)s'`. Furthermore there are some special cases:

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.
- `'%(x)s'`, `'%(y)s'`, `'%(z)s'`, `'%(t)s'` will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)
- any attribute of one of the above coordinates is inserted via `axis + key` (e.g. the name of the x-coordinate can be inserted via `'%(xname)s'`).
- Labels defined in the `psyplot.rcParams 'texts.labels'` key are also replaced when enclosed by ‘{}’. The standard labels are
  - tinfo: `%H:%M`
  - dtinfo: `%B %d, %Y. %H:%M`
  - dinfo: `%B %d, %Y`
  - desc: `%(long_name)s [%(units)s]`
  - sdesc: `%(name)s [%(units)s]`

Possible types

str – The title for the `suptitle()` function

Notes

- If the plotter is part of a `psyplot.project.Project` and multiple plotters of this project are on the same figure, the replacement attributes (see above) are joined by a delimiter. If the `delimiter` attribute of this `Figtitle` instance is not None, it will be used. Otherwise the rcParams['texts.delimiter'] item is used.
- This is the title of the whole figure! For the title of this specific subplot, see the *title* formatoption.

See also:

title, *figtitlesize*, *figtitleweight*, *figtitleprops*

figtitleprops
Properties of the figure title

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

figtitle, *figtitlesize*, *figtitleweight*

figtitlesize
Set the size of the figure title

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'.

See also:

figtitle, *figtitleweight*, *figtitleprops*

figtitleweight
Set the fontweight of the figure title

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

figtitle, *figtitlesize*, *figtitleprops*

text
Add text anywhere on the plot

This formatoption draws a text on the specified position on the figure. You can insert any meta key from the `xarray.DataArray.attrs` via a string like `'%(key)s'`. Furthermore there are some special cases:

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.
- `'%(x)s'`, `'%(y)s'`, `'%(z)s'`, `'%(t)s'` will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)
- any attribute of one of the above coordinates is inserted via `axis + key` (e.g. the name of the x-coordinate can be inserted via `'%(xname)s'`).
- Labels defined in the `psyplot.rcParams 'texts.labels'` key are also replaced when enclosed by '{}'. The standard labels are
  - tinfo: `%H:%M`
  - dtinfo: `%B %d, %Y. %H:%M`
  - dinfo: `%B %d, %Y`
  - desc: `%(long_name)s [%(units)s]`
  - sdesc: `%(name)s [%(units)s]`

Possible types

- *str* – If string s: this will be used as (1., 1., s, {'ha': 'right'}) (i.e. a string in the upper right corner of the axes).
- *tuple or list of tuples (x,y,s[,coord.-system][,options])* – Each tuple defines a text instance on the plot. 0<=x, y<=1 are the coordinates. The coord.-system can be either the data coordinates (default, `'data'`) or the axes coordinates (`'axes'`) or the figure coordinates ('fig'). The string s finally is the text. options may be a dictionary to specify format the appearence (e.g. `'color'`, `'fontweight'`, `'fontsize'`, etc., see `matplotlib.text.Text` for possible keys). To remove one single text from the plot, set (x,y,'[, coord.-system]) for the text at position (x,y)
- *empty list* – remove all texts from the plot

See also:

title, *figtitle*

title

Show the title

Set the title of the plot. You can insert any meta key from the `xarray.DataArray.attrs` via a string like `'%(key)s'`. Furthermore there are some special cases:

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.
- `'%(x)s'`, `'%(y)s'`, `'%(z)s'`, `'%(t)s'` will be replaced by the value of the x-, y-, z- or time coordinate (as long as this coordinate is one-dimensional in the data)
- any attribute of one of the above coordinates is inserted via `axis + key` (e.g. the name of the x-coordinate can be inserted via `'%(xname)s'`).
- Labels defined in the `psyplot.rcParams 'texts.labels'` key are also replaced when enclosed by '{}'. The standard labels are

- tinfo: `%H:%M`
- dtinfo: `%B %d, %Y. %H:%M`
- dinfo: `%B %d, %Y`
- desc: `%(long_name)s [%(units)s]`
- sdesc: `%(name)s [%(units)s]`

Possible types

str – The title for the `title()` function.

Notes

This is the title of this specific subplot! For the title of the whole figure, see the *figtitle* formatoption.

See also:

figtitle, *titlesize*, *titleweight*, *titleprops*

titleprops
Properties of the title

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

title, *titlesize*, *titleweight*

titlesize
Set the size of the title

Possible types

- *float* – The absolute font size in points (e.g., 12)
- *string* – Strings might be ‘xx-small’, ‘x-small’, ‘small’, ‘medium’, ‘large’, ‘x-large’, ‘xx-large’.

See also:

title, *titleweight*, *titleprops*

titleweight
Set the fontweight of the title

Possible types

- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

title, *titlesize*, *titleprops*

tight
Automatically adjust the plots.

If set to True, the plots are automatically adjusted to fit to the figure limitations via the `matplotlib.pyplot.tight_layout()` function.

Possible types

bool – True for automatic adjustment

Warning: There is no update method to undo what happend after this formatoption is set to True!

maskbetween
Mask data points between two numbers

Possible types

float – The floating number to mask above

See also:

maskless, *maskleq*, *maskgreater*, *maskgeq*

maskgeq
Mask data points greater than or equal to a number

Possible types

float – The floating number to mask above

See also:

maskless, *maskleq*, *maskgreater*, *maskbetween*

maskgreater
Mask data points greater than a number

Possible types

float – The floating number to mask above

See also:

maskless, *maskleq*, *maskgeq*, *maskbetween*

maskleq
Mask data points smaller than or equal to a number

Possible types

float – The floating number to mask below

See also:

maskless, *maskgreater*, *maskgeq*, *maskbetween*

maskless
Mask data points smaller than a number

Possible types

float – The floating number to mask below

See also:

maskleq, *maskgreater*, *maskgeq*, *maskbetween*

cticklabels
Specify the colorbar ticklabels

Possible types

- *str* – A formatstring like `'%Y'` for plotting the year (in the case that time is shown on the axis) or '%i' for integers
- *array* – An array of strings to use for the ticklabels

See also:

cticks, *cticksize*, *ctickweight*, *ctickprops*, `vcticks`, `vcticksize`, `vctickweight`, `vctickprops`

cticks
Specify the tick locations of the vector colorbar

Possible types

- *None* – use the default ticks
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

minmax Uses the minimum as minimal tick and maximum as maximal tick

sym Same as minmax but symmetric around zero

bounds let the *bounds* keyword determine the ticks. An additional integer *i* may be specified to only use every i-th bound as a tick (see also *int* below)

- *int* – Specifies how many ticks to use with the `'bounds'` option. I.e. if integer `i`, then this is the same as `['bounds', i]`.

See also:

cticklabels, vcticklabels

post

Apply your own postprocessing script

This formatoption let's you apply your own post processing script. Just enter the script as a string and it will be executed. The formatoption will be made available via the `self` variable

Possible types

- *None* – Don't do anything
- *str* – The post processing script as string

Note: This formatoption uses the built-in `exec()` function to compile the script. Since this poses a security risk when loading psyplot projects, it is by default disabled through the `Plotter.enable_post` attribute. If you are sure that you can trust the script in this formatoption, set this attribute of the corresponding `Plotter` to `True`

Examples

Assume, you want to manually add the mean of the data to the title of the matplotlib axes. You can simply do this via

```
from psyplot.plotter import Plotter
from xarray import DataArray
plotter = Plotter(DataArray([1, 2, 3]))
# enable the post formatoption
plotter.enable_post = True
plotter.update(post="self.ax.set_title(str(self.data.mean()))")
plotter.ax.get_title()
'2.0'
```

By default, the `post` formatoption is only ran, when it is explicitly updated. However, you can use the *post_timing* formatoption, to run it automatically. E.g. for running it after every update of the plotter, you can set

```
plotter.update(post_timing='always')
```

See also:

post_timing Determine the timing of this formatoption

post_timing
Determine when to run the *post* formatoption

This formatoption determines, whether the *post* formatoption should be run never, after replot or after every update.

Possible types

- *'never'* – Never run post processing scripts
- *'always'* – Always run post processing scripts
- *'replot'* – Only run post processing scripts when the data changes or a replot is necessary

See also:

post The post processing formatoption

class psy_maps.plotters.**XGrid**(**args*, ***kwargs*)
Bases: *psy_maps.plotters.GridBase*

Draw vertical grid lines (meridians)

This formatoption specifies at which longitudes to draw the meridians.

Possible types

Attributes

array	The numpy array of the data
axis	str(object='') -> str
clon	clon Formatoption instance in the plotter
dependencies	list() -> new empty list
grid_color	grid_color Formatoption instance in the plotter
grid_labels	grid_labels Formatoption instance in the plotter
grid_settings	grid_settings Formatoption instance in the plotter
lonlatbox	lonlatbox Formatoption instance in the plotter
map_extent	map_extent Formatoption instance in the plotter
name	str(object='') -> str
plot	plot Formatoption instance in the plotter
projection	projection Formatoption instance in the plotter
transform	transform Formatoption instance in the plotter

- *None* – Don't draw gridlines (same as `False`)
- *bool* – True: draw gridlines and determine position automatically False: don't draw gridlines

- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero

- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.

See also:

ygrid, *grid_color*, *grid_labels*

array
The numpy array of the data

axis = 'x'

clon
clon Formatoption instance in the plotter

dependencies = ['transform', 'grid_labels', 'grid_color', 'grid_settings', 'projection

grid_color
grid_color Formatoption instance in the plotter

grid_labels
grid_labels Formatoption instance in the plotter

grid_settings
grid_settings Formatoption instance in the plotter

lonlatbox
lonlatbox Formatoption instance in the plotter

map_extent
map_extent Formatoption instance in the plotter

name = 'Meridians'

plot
plot Formatoption instance in the plotter

projection
projection Formatoption instance in the plotter

transform
transform Formatoption instance in the plotter

class psy_maps.plotters.**YGrid**(**args*, ***kwargs*)
Bases: *psy_maps.plotters.GridBase*

Draw horizontal grid lines (parallels)

This formatoption specifies at which latitudes to draw the parallels.

Possible types

Attributes

array	The numpy array of the data
axis	str(object='') -> str
grid_color	grid_color Formatoption instance in the plotter
grid_labels	grid_labels Formatoption instance in the plotter
grid_settings	grid_settings Formatoption instance in the plotter
lonlatbox	lonlatbox Formatoption instance in the plotter
map_extent	map_extent Formatoption instance in the plotter
name	str(object='') -> str
plot	plot Formatoption instance in the plotter
projection	projection Formatoption instance in the plotter
transform	transform Formatoption instance in the plotter

- *None* – Don't draw gridlines (same as `False`)
- *bool* – True: draw gridlines and determine position automatically False: don't draw gridlines
- *numeric array* – specifies the ticks manually
- *str or list [str, ...]* – Automatically determine the ticks corresponding to the data. The given string determines how the ticks are calculated. If not a single string but a list, the second value determines the number of ticks (see below). A string can be one of the following:

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

  rounded Sets the minimum and maximum of the ticks to the rounded data minimum or maximum. Ticks are rounded to the next 0.5 value with to the difference between data max- and minimum. The minimal tick will always be lower or equal than the data minimum, the maximal tick will always be higher or equal than the data maximum.

  roundedsym Same as *rounded* above but the ticks are chose such that they are symmetric around zero

  minmax Uses the minimum as minimal tick and maximum as maximal tick

  sym Same as minmax but symmetric around zero

- *int* – Specifies how many ticks to use with the `'rounded'` option. I.e. if integer `i`, then this is the same as `['rounded', i]`.

See also:

xgrid, *grid_color*, *grid_labels*

array
The numpy array of the data

axis = 'y'

grid_color
grid_color Formatoption instance in the plotter

grid_labels
grid_labels Formatoption instance in the plotter

grid_settings
grid_settings Formatoption instance in the plotter

lonlatbox
lonlatbox Formatoption instance in the plotter

map_extent
map_extent Formatoption instance in the plotter

name = 'Parallels'

plot
plot Formatoption instance in the plotter

projection
projection Formatoption instance in the plotter

transform
transform Formatoption instance in the plotter

psy_maps.plotters.**degree_format**()

psy_maps.plotters.**format_lats**(*x*, *pos*)

psy_maps.plotters.**format_lons**(*x*, *pos*)

psy_maps.plotters.**shiftdata**(*lonsin*, *datain*, *lon_0*)
Shift longitudes (and optionally data) so that they match map projection region. Only valid for cylindrical/pseudo-cylindrical global projections and data on regular lat/lon grids. longitudes and data can be 1-d or 2-d, if 2-d it is assumed longitudes are 2nd (rightmost) dimension.

Parameters

- **lonsin** – original 1-d or 2-d longitudes.
- **datain** – original 1-d or 2-d data
- **lon_0** – center of map projection region

References

This function is copied and taken from the mpl_toolkits.basemap.Basemap class. The only difference is that we do not mask values outside the map projection region

psy_maps.plugin module

psy-simple psyplot plugin

This module defines the rcParams for the psy-simple plugin

Classes

ProjectionValidator(key, valid[, ignorecase])	valid is a list of legal strings

Functions

get_versions([requirements])	
patch_prior_1_0(plotter_d, versions)	Patch psy_maps plotters for versions smaller than 1.0
validate_dict_yaml(s)	
validate_grid(val)	
validate_lonlatbox(value)	
validate_lsm(val)	

Data

patches	patches to apply when loading a project
rcParams	the RcParams for the psy-simple plugin

class psy_maps.plugin.**ProjectionValidator**(*key*, *valid*, *ignorecase=False*)
Bases: matplotlib.rcsetup.ValidateInStrings

valid is a list of legal strings

psy_maps.plugin.**get_versions**(*requirements=True*)

psy_maps.plugin.**patch_prior_1_0**(*plotter_d*, *versions*)
Patch psy_maps plotters for versions smaller than 1.0

Before psyplot 1.0.0, the plotters in the psy_maps package where part of the psyplot.plotter.maps module. This has to be corrected

psy_maps.plugin.**patches = {('psyplot.plotter.maps', 'CombinedPlotter'): <function patch_pr**
patches to apply when loading a project

psy_maps.plugin.**rcParams**
the RcParams for the psy-simple plugin

psy_maps.plugin.**validate_dict_yaml**(*s*)

psy_maps.plugin.**validate_grid**(*val*)

psy_maps.plugin.**validate_lonlatbox**(*value*)

psy_maps.plugin.**validate_lsm**(*val*)

psy_maps.version module

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

psy_maps, 22
psy_maps.boxes, 22
psy_maps.plotters, 22
psy_maps.plugin, 135
psy_maps.version, 136

Index

A

add2format_coord() (psy_maps.plotters.MapPlot2D method), 90
add2format_coord() (psy_maps.plotters.MapVectorPlot method), 105
array (psy_maps.plotters.MapPlot2D attribute), 90
array (psy_maps.plotters.XGrid attribute), 133
array (psy_maps.plotters.YGrid attribute), 134
arrowsize (psy_maps.plotters.CombinedMapVectorPlot attribute), 28
arrowsize (psy_maps.plotters.CombinedPlotter attribute), 34
arrowsize (psy_maps.plotters.MapVectorPlot attribute), 106
arrowsize (psy_maps.plotters.VectorPlotter attribute), 117
arrowstyle (psy_maps.plotters.CombinedMapVectorPlot attribute), 28
arrowstyle (psy_maps.plotters.CombinedPlotter attribute), 34
arrowstyle (psy_maps.plotters.MapVectorPlot attribute), 106
arrowstyle (psy_maps.plotters.VectorPlotter attribute), 117
ax (psy_maps.plotters.MapPlotter attribute), 92
axis (psy_maps.plotters.GridBase attribute), 77
axis (psy_maps.plotters.XGrid attribute), 133
axis (psy_maps.plotters.YGrid attribute), 134

B

bounds (psy_maps.plotters.CombinedMapVectorPlot attribute), 28
bounds (psy_maps.plotters.CombinedPlotter attribute), 48
bounds (psy_maps.plotters.FieldPlotter attribute), 65
bounds (psy_maps.plotters.MapPlot2D attribute), 90
bounds (psy_maps.plotters.MapPlotter attribute), 99
bounds (psy_maps.plotters.MapVectorColor attribute), 105
bounds (psy_maps.plotters.MapVectorPlot attribute), 106
bounds (psy_maps.plotters.VectorPlotter attribute), 114
BoxBase (class in psy_maps.plotters), 23

C

calc_lonlatbox() (psy_maps.plotters.LonLatBox method), 86
cbar (psy_maps.plotters.CombinedPlotter attribute), 49
cbar (psy_maps.plotters.FieldPlotter attribute), 66
cbar (psy_maps.plotters.MapPlotter attribute), 100
cbar (psy_maps.plotters.VectorPlotter attribute), 115
cbarspacing (psy_maps.plotters.CombinedPlotter attribute), 49
cbarspacing (psy_maps.plotters.FieldPlotter attribute), 66
cbarspacing (psy_maps.plotters.MapPlotter attribute), 101
cbarspacing (psy_maps.plotters.VectorPlotter attribute), 116
CenterLat (class in psy_maps.plotters), 24
CenterLon (class in psy_maps.plotters), 25
children (psy_maps.plotters.GridSettings attribute), 82
clabel (psy_maps.plotters.CombinedPlotter attribute), 41
clabel (psy_maps.plotters.FieldPlotter attribute), 68
clabel (psy_maps.plotters.MapPlotter attribute), 98
clabel (psy_maps.plotters.VectorPlotter attribute), 124
clabelprops (psy_maps.plotters.CombinedPlotter attribute), 42
clabelprops (psy_maps.plotters.FieldPlotter attribute), 69
clabelprops (psy_maps.plotters.MapPlotter attribute), 99
clabelprops (psy_maps.plotters.VectorPlotter attribute), 124
clabelsize (psy_maps.plotters.CombinedPlotter attribute), 42
clabelsize (psy_maps.plotters.FieldPlotter attribute), 69
clabelsize (psy_maps.plotters.MapPlotter attribute), 99
clabelsize (psy_maps.plotters.VectorPlotter attribute), 124
clabelweight (psy_maps.plotters.CombinedPlotter attribute), 42
clabelweight (psy_maps.plotters.FieldPlotter attribute), 69

clabelweight (psy_maps.plotters.MapPlotter attribute), 99
clabelweight (psy_maps.plotters.VectorPlotter attribute), 125
clat (psy_maps.plotters.CombinedMapVectorPlot attribute), 29
clat (psy_maps.plotters.CombinedPlotter attribute), 34
clat (psy_maps.plotters.FieldPlotter attribute), 60
clat (psy_maps.plotters.MapPlotter attribute), 92
clat (psy_maps.plotters.MapVectorPlot attribute), 106
clat (psy_maps.plotters.Projection attribute), 107
clat (psy_maps.plotters.VectorPlotter attribute), 118
clip (psy_maps.plotters.CombinedMapVectorPlot attribute), 29
clip (psy_maps.plotters.CombinedPlotter attribute), 35
clip (psy_maps.plotters.FieldPlotter attribute), 60
clip (psy_maps.plotters.MapPlot2D attribute), 90
clip (psy_maps.plotters.MapPlotter attribute), 92
clip (psy_maps.plotters.MapVectorPlot attribute), 106
clip (psy_maps.plotters.VectorPlotter attribute), 118
ClipAxes (class in psy_maps.plotters), 26
clon (psy_maps.plotters.CombinedMapVectorPlot attribute), 29
clon (psy_maps.plotters.CombinedPlotter attribute), 35
clon (psy_maps.plotters.FieldPlotter attribute), 60
clon (psy_maps.plotters.MapPlotter attribute), 93
clon (psy_maps.plotters.MapVectorPlot attribute), 106
clon (psy_maps.plotters.Projection attribute), 107
clon (psy_maps.plotters.VectorPlotter attribute), 118
clon (psy_maps.plotters.XGrid attribute), 133
cmap (psy_maps.plotters.CombinedMapVectorPlot attribute), 29
cmap (psy_maps.plotters.CombinedPlotter attribute), 49
cmap (psy_maps.plotters.FieldPlotter attribute), 67
cmap (psy_maps.plotters.MapPlot2D attribute), 90
cmap (psy_maps.plotters.MapPlotter attribute), 101
cmap (psy_maps.plotters.MapVectorColor attribute), 105
cmap (psy_maps.plotters.MapVectorPlot attribute), 106
cmap (psy_maps.plotters.VectorPlotter attribute), 116
color (psy_maps.plotters.CombinedMapVectorPlot attribute), 29
color (psy_maps.plotters.CombinedPlotter attribute), 49
color (psy_maps.plotters.MapVectorPlot attribute), 106
color (psy_maps.plotters.VectorPlotter attribute), 114
CombinedMapVectorPlot (class in psy_maps.plotters), 28
CombinedPlotter (class in psy_maps.plotters), 29
connections (psy_maps.plotters.ClipAxes attribute), 27
connections (psy_maps.plotters.GridBase attribute), 77
connections (psy_maps.plotters.GridColor attribute), 79
connections (psy_maps.plotters.GridLabels attribute), 81
connections (psy_maps.plotters.GridSettings attribute), 83
connections (psy_maps.plotters.MapPlot2D attribute), 90
connections (psy_maps.plotters.StockImage attribute), 109
connections (psy_maps.plotters.Transform attribute), 111
convert_radian (psy_maps.plotters.MapPlotter attribute), 93
cticklabels (psy_maps.plotters.CombinedPlotter attribute), 39
cticklabels (psy_maps.plotters.FieldPlotter attribute), 74
cticklabels (psy_maps.plotters.MapPlotter attribute), 102
cticklabels (psy_maps.plotters.VectorPlotter attribute), 130
ctickprops (psy_maps.plotters.CombinedPlotter attribute), 50
ctickprops (psy_maps.plotters.FieldPlotter attribute), 67
ctickprops (psy_maps.plotters.MapPlotter attribute), 101
ctickprops (psy_maps.plotters.VectorPlotter attribute), 116
cticks (psy_maps.plotters.CombinedPlotter attribute), 40
cticks (psy_maps.plotters.FieldPlotter attribute), 74
cticks (psy_maps.plotters.MapPlotter attribute), 102
cticks (psy_maps.plotters.VectorPlotter attribute), 130
cticksize (psy_maps.plotters.CombinedPlotter attribute), 50
cticksize (psy_maps.plotters.FieldPlotter attribute), 67
cticksize (psy_maps.plotters.MapPlotter attribute), 101
cticksize (psy_maps.plotters.VectorPlotter attribute), 116
ctickweight (psy_maps.plotters.CombinedPlotter attribute), 50
ctickweight (psy_maps.plotters.FieldPlotter attribute), 67
ctickweight (psy_maps.plotters.MapPlotter attribute), 102
ctickweight (psy_maps.plotters.VectorPlotter attribute), 117

D

data_dependent (psy_maps.plotters.CombinedMapVectorPlot attribute), 29
data_dependent (psy_maps.plotters.MapPlot2D attribute), 90
data_dependent (psy_maps.plotters.MapVectorPlot attribute), 106
data_dependent() (psy_maps.plotters.LonLatBox method), 86
datagrid (psy_maps.plotters.CombinedPlotter attribute), 35
datagrid (psy_maps.plotters.FieldPlotter attribute), 60
datagrid (psy_maps.plotters.MapPlotter attribute), 93
datagrid (psy_maps.plotters.VectorPlotter attribute), 118
degree_format() (in module psy_maps.plotters), 135
density (psy_maps.plotters.CombinedMapVectorPlot attribute), 29
density (psy_maps.plotters.CombinedPlotter attribute), 32
density (psy_maps.plotters.MapVectorPlot attribute), 106
density (psy_maps.plotters.VectorPlotter attribute), 114
dependencies (psy_maps.plotters.CenterLat attribute), 25
dependencies (psy_maps.plotters.CenterLon attribute), 26

dependencies (psy_maps.plotters.GridBase attribute), 77
dependencies (psy_maps.plotters.GridLabels attribute), 81
dependencies (psy_maps.plotters.GridLabelSize attribute), 80
dependencies (psy_maps.plotters.LonLatBox attribute), 86
dependencies (psy_maps.plotters.MapExtent attribute), 89
dependencies (psy_maps.plotters.MapPlot2D attribute), 90
dependencies (psy_maps.plotters.MapVectorPlot attribute), 106
dependencies (psy_maps.plotters.Projection attribute), 107
dependencies (psy_maps.plotters.XGrid attribute), 133
draw_circle() (psy_maps.plotters.ClipAxes method), 27

E

extend (psy_maps.plotters.CombinedPlotter attribute), 51
extend (psy_maps.plotters.FieldPlotter attribute), 68
extend (psy_maps.plotters.MapPlotter attribute), 102
extend (psy_maps.plotters.VectorPlotter attribute), 117

F

FieldPlotter (class in psy_maps.plotters), 56
figtitle (psy_maps.plotters.CombinedPlotter attribute), 42
figtitle (psy_maps.plotters.FieldPlotter attribute), 69
figtitle (psy_maps.plotters.VectorPlotter attribute), 125
figtitleprops (psy_maps.plotters.CombinedPlotter attribute), 43
figtitleprops (psy_maps.plotters.FieldPlotter attribute), 70
figtitleprops (psy_maps.plotters.VectorPlotter attribute), 126
figtitlesize (psy_maps.plotters.CombinedPlotter attribute), 43
figtitlesize (psy_maps.plotters.FieldPlotter attribute), 70
figtitlesize (psy_maps.plotters.VectorPlotter attribute), 126
figtitleweight (psy_maps.plotters.CombinedPlotter attribute), 44
figtitleweight (psy_maps.plotters.FieldPlotter attribute), 71
figtitleweight (psy_maps.plotters.VectorPlotter attribute), 126
format_lats() (in module psy_maps.plotters), 135
format_lons() (in module psy_maps.plotters), 135

G

get_kwargs() (psy_maps.plotters.GridBase method), 77
get_kwargs() (psy_maps.plotters.ProjectionBase method), 109
get_versions() (in module psy_maps.plugin), 136
grid_color (psy_maps.plotters.CombinedPlotter attribute), 35
grid_color (psy_maps.plotters.FieldPlotter attribute), 61
grid_color (psy_maps.plotters.GridBase attribute), 77
grid_color (psy_maps.plotters.GridSettings attribute), 83
grid_color (psy_maps.plotters.MapPlotter attribute), 93
grid_color (psy_maps.plotters.VectorPlotter attribute), 119
grid_color (psy_maps.plotters.XGrid attribute), 133
grid_color (psy_maps.plotters.YGrid attribute), 134
grid_labels (psy_maps.plotters.CombinedPlotter attribute), 36
grid_labels (psy_maps.plotters.FieldPlotter attribute), 61
grid_labels (psy_maps.plotters.GridBase attribute), 78
grid_labels (psy_maps.plotters.GridSettings attribute), 83
grid_labels (psy_maps.plotters.MapPlotter attribute), 94
grid_labels (psy_maps.plotters.VectorPlotter attribute), 119
grid_labels (psy_maps.plotters.XGrid attribute), 133
grid_labels (psy_maps.plotters.YGrid attribute), 135
grid_labelsize (psy_maps.plotters.CombinedPlotter attribute), 36
grid_labelsize (psy_maps.plotters.FieldPlotter attribute), 61
grid_labelsize (psy_maps.plotters.MapPlotter attribute), 94
grid_labelsize (psy_maps.plotters.VectorPlotter attribute), 119
grid_settings (psy_maps.plotters.CombinedPlotter attribute), 36
grid_settings (psy_maps.plotters.FieldPlotter attribute), 61
grid_settings (psy_maps.plotters.GridBase attribute), 78
grid_settings (psy_maps.plotters.MapPlotter attribute), 94
grid_settings (psy_maps.plotters.VectorPlotter attribute), 119
grid_settings (psy_maps.plotters.XGrid attribute), 133
grid_settings (psy_maps.plotters.YGrid attribute), 135
GridBase (class in psy_maps.plotters), 76
GridColor (class in psy_maps.plotters), 78
GridLabels (class in psy_maps.plotters), 80
GridLabelSize (class in psy_maps.plotters), 79
GridSettings (class in psy_maps.plotters), 82

I

image (psy_maps.plotters.StockImage attribute), 110
initialize_plot() (psy_maps.plotters.Projection method), 107
interp_bounds (psy_maps.plotters.CombinedPlotter attribute), 36
interp_bounds (psy_maps.plotters.FieldPlotter attribute), 58
interp_bounds (psy_maps.plotters.MapPlot2D attribute), 90

L

levels (psy_maps.plotters.CombinedPlotter attribute), 51
levels (psy_maps.plotters.FieldPlotter attribute), 59
levels (psy_maps.plotters.MapPlot2D attribute), 90
linewidth (psy_maps.plotters.CombinedMapVectorPlot attribute), 29
linewidth (psy_maps.plotters.CombinedPlotter attribute), 37
linewidth (psy_maps.plotters.MapVectorPlot attribute), 106
linewidth (psy_maps.plotters.VectorPlotter attribute), 120
lola_from_pattern() (psy_maps.plotters.BoxBase method), 24
LonLatBox (class in psy_maps.plotters), 84
lonlatbox (psy_maps.plotters.CenterLat attribute), 25
lonlatbox (psy_maps.plotters.CenterLon attribute), 26
lonlatbox (psy_maps.plotters.ClipAxes attribute), 27
lonlatbox (psy_maps.plotters.CombinedMapVectorPlot attribute), 29
lonlatbox (psy_maps.plotters.CombinedPlotter attribute), 37
lonlatbox (psy_maps.plotters.FieldPlotter attribute), 61
lonlatbox (psy_maps.plotters.GridBase attribute), 78
lonlatbox (psy_maps.plotters.MapExtent attribute), 89
lonlatbox (psy_maps.plotters.MapPlot2D attribute), 90
lonlatbox (psy_maps.plotters.MapPlotter attribute), 94
lonlatbox (psy_maps.plotters.MapVectorPlot attribute), 106
lonlatbox (psy_maps.plotters.VectorPlotter attribute), 120
lonlatbox (psy_maps.plotters.XGrid attribute), 133
lonlatbox (psy_maps.plotters.YGrid attribute), 135
lonlatbox_transformed (psy_maps.plotters.LonLatBox attribute), 86
lonlatboxes (in module psy_maps.boxes), 22
LSM (class in psy_maps.plotters), 83
lsm (psy_maps.plotters.CombinedPlotter attribute), 38
lsm (psy_maps.plotters.FieldPlotter attribute), 62
lsm (psy_maps.plotters.LSM attribute), 84
lsm (psy_maps.plotters.MapPlotter attribute), 95
lsm (psy_maps.plotters.VectorPlotter attribute), 121

M

map_extent (psy_maps.plotters.ClipAxes attribute), 27
map_extent (psy_maps.plotters.CombinedPlotter attribute), 38
map_extent (psy_maps.plotters.FieldPlotter attribute), 62
map_extent (psy_maps.plotters.GridBase attribute), 78
map_extent (psy_maps.plotters.MapPlotter attribute), 95
map_extent (psy_maps.plotters.VectorPlotter attribute), 121
map_extent (psy_maps.plotters.XGrid attribute), 133
map_extent (psy_maps.plotters.YGrid attribute), 135
mapcombined (psyplot.project.plot attribute), 6
MapDataGrid (class in psy_maps.plotters), 86
MapDensity (class in psy_maps.plotters), 87
MapExtent (class in psy_maps.plotters), 88
mapplot (psyplot.project.plot attribute), 4
MapPlot2D (class in psy_maps.plotters), 89
MapPlotter (class in psy_maps.plotters), 91
mapvector (psyplot.project.plot attribute), 5
MapVectorColor (class in psy_maps.plotters), 104
MapVectorPlot (class in psy_maps.plotters), 105
mask_outside() (psy_maps.plotters.LonLatBox method), 86
maskbetween (psy_maps.plotters.CombinedPlotter attribute), 54
maskbetween (psy_maps.plotters.FieldPlotter attribute), 73
maskbetween (psy_maps.plotters.VectorPlotter attribute), 129
maskgeq (psy_maps.plotters.CombinedPlotter attribute), 54
maskgeq (psy_maps.plotters.FieldPlotter attribute), 73
maskgeq (psy_maps.plotters.VectorPlotter attribute), 129
maskgreater (psy_maps.plotters.CombinedPlotter attribute), 54
maskgreater (psy_maps.plotters.FieldPlotter attribute), 74
maskgreater (psy_maps.plotters.VectorPlotter attribute), 129
maskleq (psy_maps.plotters.CombinedPlotter attribute), 55
maskleq (psy_maps.plotters.FieldPlotter attribute), 74
maskleq (psy_maps.plotters.VectorPlotter attribute), 130
maskless (psy_maps.plotters.CombinedPlotter attribute), 55
maskless (psy_maps.plotters.FieldPlotter attribute), 74
maskless (psy_maps.plotters.VectorPlotter attribute), 130
miss_color (psy_maps.plotters.CombinedPlotter attribute), 51
miss_color (psy_maps.plotters.FieldPlotter attribute), 68

N

name (psy_maps.plotters.CenterLat attribute), 25
name (psy_maps.plotters.CenterLon attribute), 26
name (psy_maps.plotters.GridColor attribute), 79
name (psy_maps.plotters.GridLabels attribute), 81
name (psy_maps.plotters.GridLabelSize attribute), 80
name (psy_maps.plotters.GridSettings attribute), 83
name (psy_maps.plotters.LonLatBox attribute), 86
name (psy_maps.plotters.LSM attribute), 84
name (psy_maps.plotters.MapExtent attribute), 89
name (psy_maps.plotters.Projection attribute), 107
name (psy_maps.plotters.StockImage attribute), 110
name (psy_maps.plotters.Transform attribute), 111
name (psy_maps.plotters.XGrid attribute), 133
name (psy_maps.plotters.YGrid attribute), 135

P

patch_prior_1_0() (in module psy_maps.plugin), 136
patches (in module psy_maps.plugin), 136
plot (psy_maps.plotters.CombinedPlotter attribute), 32
plot (psy_maps.plotters.FieldPlotter attribute), 59
plot (psy_maps.plotters.GridBase attribute), 78
plot (psy_maps.plotters.MapDensity attribute), 88
plot (psy_maps.plotters.MapExtent attribute), 89
plot (psy_maps.plotters.MapVectorColor attribute), 105
plot (psy_maps.plotters.StockImage attribute), 110
plot (psy_maps.plotters.Transform attribute), 111
plot (psy_maps.plotters.VectorPlotter attribute), 114
plot (psy_maps.plotters.XGrid attribute), 133
plot (psy_maps.plotters.YGrid attribute), 135
post (psy_maps.plotters.CombinedPlotter attribute), 55
post (psy_maps.plotters.FieldPlotter attribute), 75
post (psy_maps.plotters.MapPlotter attribute), 103
post (psy_maps.plotters.VectorPlotter attribute), 131
post_timing (psy_maps.plotters.CombinedPlotter attribute), 56
post_timing (psy_maps.plotters.FieldPlotter attribute), 76
post_timing (psy_maps.plotters.MapPlotter attribute), 104
post_timing (psy_maps.plotters.VectorPlotter attribute), 132
priority (psy_maps.plotters.CenterLat attribute), 25
priority (psy_maps.plotters.CenterLon attribute), 26
priority (psy_maps.plotters.ClipAxes attribute), 28
priority (psy_maps.plotters.LonLatBox attribute), 86
priority (psy_maps.plotters.MapExtent attribute), 89
priority (psy_maps.plotters.Projection attribute), 107
priority (psy_maps.plotters.StockImage attribute), 110
priority (psy_maps.plotters.Transform attribute), 111
Projection (class in psy_maps.plotters), 106
projection (psy_maps.plotters.CombinedPlotter attribute), 38
projection (psy_maps.plotters.FieldPlotter attribute), 63
projection (psy_maps.plotters.GridBase attribute), 78
projection (psy_maps.plotters.GridLabels attribute), 81
projection (psy_maps.plotters.MapPlotter attribute), 95
projection (psy_maps.plotters.VectorPlotter attribute), 121
projection (psy_maps.plotters.XGrid attribute), 133
projection (psy_maps.plotters.YGrid attribute), 135
projection_kwargs (psy_maps.plotters.ProjectionBase attribute), 109
ProjectionBase (class in psy_maps.plotters), 108
projections (psy_maps.plotters.ProjectionBase attribute), 109
ProjectionValidator (class in psy_maps.plugin), 136
psy_maps (module), 22
psy_maps.boxes (module), 22
psy_maps.plotters (module), 22
psy_maps.plugin (module), 135
psy_maps.version (module), 136

R

rcParams (in module psy_maps.plugin), 136
remove() (psy_maps.plotters.ClipAxes method), 28
remove() (psy_maps.plotters.GridBase method), 78
remove() (psy_maps.plotters.LSM method), 84
remove() (psy_maps.plotters.MapPlot2D method), 90
remove() (psy_maps.plotters.StockImage method), 110
requires_clearing (psy_maps.plotters.CenterLat attribute), 25
requires_clearing (psy_maps.plotters.CenterLon attribute), 26
requires_clearing (psy_maps.plotters.LonLatBox attribute), 86
requires_clearing (psy_maps.plotters.Projection attribute), 107

S

set_projection() (psy_maps.plotters.ProjectionBase method), 109
set_value() (psy_maps.plotters.GridSettings method), 83
set_value() (psy_maps.plotters.MapVectorPlot method), 106
shiftdata() (in module psy_maps.plotters), 135
shiftdata() (psy_maps.plotters.LonLatBox method), 86
stock_img (psy_maps.plotters.CombinedPlotter attribute), 39
stock_img (psy_maps.plotters.FieldPlotter attribute), 63
stock_img (psy_maps.plotters.MapPlotter attribute), 96
stock_img (psy_maps.plotters.VectorPlotter attribute), 122
StockImage (class in psy_maps.plotters), 109

T

text (psy_maps.plotters.CombinedPlotter attribute), 44
text (psy_maps.plotters.FieldPlotter attribute), 71
text (psy_maps.plotters.VectorPlotter attribute), 126
tight (psy_maps.plotters.CombinedPlotter attribute), 47
tight (psy_maps.plotters.FieldPlotter attribute), 73
tight (psy_maps.plotters.VectorPlotter attribute), 129
title (psy_maps.plotters.CombinedPlotter attribute), 45
title (psy_maps.plotters.FieldPlotter attribute), 71
title (psy_maps.plotters.VectorPlotter attribute), 127
titleprops (psy_maps.plotters.CombinedPlotter attribute), 45
titleprops (psy_maps.plotters.FieldPlotter attribute), 72
titleprops (psy_maps.plotters.VectorPlotter attribute), 128
titlesize (psy_maps.plotters.CombinedPlotter attribute), 46
titlesize (psy_maps.plotters.FieldPlotter attribute), 72
titlesize (psy_maps.plotters.VectorPlotter attribute), 128
titleweight (psy_maps.plotters.CombinedPlotter attribute), 46

titleweight (psy_maps.plotters.FieldPlotter attribute), 73
titleweight (psy_maps.plotters.VectorPlotter attribute), 128
to_degree() (psy_maps.plotters.LonLatBox method), 86
Transform (class in psy_maps.plotters), 110
transform (psy_maps.plotters.CombinedMapVectorPlot attribute), 29
transform (psy_maps.plotters.CombinedPlotter attribute), 39
transform (psy_maps.plotters.FieldPlotter attribute), 63
transform (psy_maps.plotters.GridBase attribute), 78
transform (psy_maps.plotters.GridLabels attribute), 81
transform (psy_maps.plotters.LonLatBox attribute), 86
transform (psy_maps.plotters.MapDataGrid attribute), 87
transform (psy_maps.plotters.MapPlot2D attribute), 91
transform (psy_maps.plotters.MapPlotter attribute), 96
transform (psy_maps.plotters.MapVectorPlot attribute), 106
transform (psy_maps.plotters.VectorPlotter attribute), 122
transform (psy_maps.plotters.XGrid attribute), 133
transform (psy_maps.plotters.YGrid attribute), 135
transpose (psy_maps.plotters.CombinedMapVectorPlot attribute), 29
transpose (psy_maps.plotters.MapVectorColor attribute), 105
transpose (psy_maps.plotters.MapVectorPlot attribute), 106
triangles (psy_maps.plotters.MapDataGrid attribute), 87

U

update() (psy_maps.plotters.CenterLat method), 25
update() (psy_maps.plotters.CenterLon method), 26
update() (psy_maps.plotters.ClipAxes method), 28
update() (psy_maps.plotters.CombinedMapVectorPlot method), 29
update() (psy_maps.plotters.GridBase method), 78
update() (psy_maps.plotters.GridColor method), 79
update() (psy_maps.plotters.GridLabels method), 81
update() (psy_maps.plotters.GridLabelSize method), 80
update() (psy_maps.plotters.GridSettings method), 83
update() (psy_maps.plotters.LonLatBox method), 86
update() (psy_maps.plotters.LSM method), 84
update() (psy_maps.plotters.MapExtent method), 89
update() (psy_maps.plotters.Projection method), 107
update() (psy_maps.plotters.StockImage method), 110
update() (psy_maps.plotters.Transform method), 111
update_after_plot (psy_maps.plotters.MapExtent attribute), 89
update_array() (psy_maps.plotters.LonLatBox method), 86

V

validate_dict_yaml() (in module psy_maps.plugin), 136
validate_grid() (in module psy_maps.plugin), 136
validate_lonlatbox() (in module psy_maps.plugin), 136
validate_lsm() (in module psy_maps.plugin), 136
vbounds (psy_maps.plotters.CombinedPlotter attribute), 52
vcbar (psy_maps.plotters.CombinedPlotter attribute), 53
vcbarspacing (psy_maps.plotters.CombinedPlotter attribute), 53
vclabel (psy_maps.plotters.CombinedPlotter attribute), 46
vclabelprops (psy_maps.plotters.CombinedPlotter attribute), 47
vclabelsize (psy_maps.plotters.CombinedPlotter attribute), 47
vclabelweight (psy_maps.plotters.CombinedPlotter attribute), 47
vcmap (psy_maps.plotters.CombinedPlotter attribute), 53
vcticklabels (psy_maps.plotters.CombinedPlotter attribute), 40
vctickprops (psy_maps.plotters.CombinedPlotter attribute), 53
vcticks (psy_maps.plotters.CombinedPlotter attribute), 41
vcticksize (psy_maps.plotters.CombinedPlotter attribute), 54
vctickweight (psy_maps.plotters.CombinedPlotter attribute), 54
VectorPlotter (class in psy_maps.plotters), 111
vplot (psy_maps.plotters.CombinedPlotter attribute), 32
vplot (psy_maps.plotters.MapExtent attribute), 89
vplot (psy_maps.plotters.Transform attribute), 111

X

XGrid (class in psy_maps.plotters), 132
xgrid (psy_maps.plotters.CombinedPlotter attribute), 32
xgrid (psy_maps.plotters.FieldPlotter attribute), 64
xgrid (psy_maps.plotters.GridColor attribute), 79
xgrid (psy_maps.plotters.GridLabels attribute), 81
xgrid (psy_maps.plotters.GridLabelSize attribute), 80
xgrid (psy_maps.plotters.GridSettings attribute), 83
xgrid (psy_maps.plotters.MapPlotter attribute), 97
xgrid (psy_maps.plotters.VectorPlotter attribute), 122

Y

YGrid (class in psy_maps.plotters), 133
ygrid (psy_maps.plotters.CombinedPlotter attribute), 33
ygrid (psy_maps.plotters.FieldPlotter attribute), 64
ygrid (psy_maps.plotters.GridColor attribute), 79
ygrid (psy_maps.plotters.GridLabels attribute), 82
ygrid (psy_maps.plotters.GridLabelSize attribute), 80
ygrid (psy_maps.plotters.GridSettings attribute), 83
ygrid (psy_maps.plotters.MapPlotter attribute), 97
ygrid (psy_maps.plotters.VectorPlotter attribute), 123