1 Documentation

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 unstructured data ......................... 7
    Visualizing UGRID data ................................... 8
    Visualizing scalar and vector ICON data ................ 10
    The handling unstructured grids ......................... 14
      Interpreting the UGRID Conventions ................... 14
      Interpreting the CF Conventions ....................... 16
      Identification of unstructured variables ............. 17
    Acknowledgement ......................................... 18
  1.3.2 Basic data visualization on a map ..................... 18
    Visualizing scalar fields ................................ 18
    Visualizing vector data .................................. 22
    Visualizing combined scalar and vector data .......... 23
    Summary .................................................. 25
  1.3.3 Visualizing circumpolar data ......................... 26
    Note ..................................................... 29

1.4 Contributing to psy-maps .................................. 30

1.5 API Reference ................................................ 30
  1.5.1 Submodules ............................................ 30
    psy_maps.boxes module .................................. 30
    psy_maps.plotters module ................................ 31
    psy_maps.plugin module ................................ 142
    psy_maps.version module ................................ 143

2 Indices and tables ........................................... 145
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 maplot and mapvector plot methods which can plot rectangular and triangular 2-dimensional data.

See the psyplot plot methods and Example Gallery for more information.
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 0x7f17a240>
```

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mapplot</code></td>
<td>Plot a 2D scalar field on a map</td>
</tr>
<tr>
<td><code>mapvector</code></td>
<td>Plot a 2D vector field on a map</td>
</tr>
<tr>
<td><code>mapcombined</code></td>
<td>Plot a 2D scalar field with an overlying vector field on a map</td>
</tr>
</tbody>
</table>

### 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
Examples

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

```python
>>> import psyplot.project as psy
# show the keys corresponding to a group or multiple
# formatoptions
>>> 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:

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

Possible format options are
Examples

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

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

# show the keys corresponding to a group or multiple formatoptions
>>> 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:

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

Possible formatoptions are

<table>
<thead>
<tr>
<th>arrowsize</th>
<th>arrowstyle</th>
<th>bounds</th>
<th>cbar</th>
</tr>
</thead>
<tbody>
<tr>
<td>cbarspacing</td>
<td>clabel</td>
<td>clabelprops</td>
<td>clabelsize</td>
</tr>
<tr>
<td>clabelweight</td>
<td>clat</td>
<td>clip</td>
<td>clon</td>
</tr>
<tr>
<td>cmap</td>
<td>color</td>
<td>cticklabels</td>
<td>ctickprops</td>
</tr>
<tr>
<td>cticks</td>
<td>cticksize</td>
<td>ctickweight</td>
<td>datagrid</td>
</tr>
<tr>
<td>density</td>
<td>extend</td>
<td>figtitle</td>
<td>figtitleprops</td>
</tr>
<tr>
<td>figtitlesize</td>
<td>figtitleweight</td>
<td>grid_color</td>
<td>grid_labels</td>
</tr>
<tr>
<td>grid_labelsize</td>
<td>grid_settings</td>
<td>linewidth</td>
<td>lonlatbox</td>
</tr>
<tr>
<td>lsm</td>
<td>map_extent</td>
<td>maskbetween</td>
<td>maskgeq</td>
</tr>
<tr>
<td>maskgreater</td>
<td>maskleg</td>
<td>maskless</td>
<td>plot</td>
</tr>
<tr>
<td>post</td>
<td>post_timing</td>
<td>projection</td>
<td>stock_img</td>
</tr>
<tr>
<td>text</td>
<td>tight</td>
<td>title</td>
<td>titleprops</td>
</tr>
<tr>
<td>titlesize</td>
<td>titleweight</td>
<td>transform</td>
<td>xgrid</td>
</tr>
</tbody>
</table>
Examples

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

```python
>>> import psyplot.project as psy
# show the keys corresponding to a group or multiple # formatoptions
>>> 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 unstructured data

Example visualization of unstructured ICON and UGRID data.
Here we show, how the psy-maps plugin can visualize the unstructured grid of the Earth System Model ICON from the Max-Planck-Institute for Meteorology in Hamburg, Germany, and data following the unstructured grids (UGRID). The visualization works the same as for normal rectilinear grids. Internally, however, the coordinates are interpreted in a completely different way (see below for a detailed explanation).

```python
import logging
logging.captureWarnings(True)
logging.getLogger('py.warnings').setLevel(logging.ERROR)
import psyplot.project as psy
import matplotlib as mpl
psy.rcParams['plotter.maps.xgrid'] = False
psy.rcParams['plotter.maps.ygrid'] = False
mpl.rcParams['figure.figsize'] = [10., 8.]
```

### Visualizing UGRID data

A widely accepted approach for unstructured grids are the so-called UGRID Conventions. For a demonstration, we use the 'ugrid_demo.nc' netCDF file that contains sea surface height of a Tsunami simulation. We use the load parameter, to directly load it into memory. That speeds up the plotting of the data.

psyplot automatically recognizes the UGRID conventions and adapts it’s plotting algorithm for displaying the data. For this simulation, let’s focus on Japan. We use the maskleq keyword here to mask the land surface and display a stock_img on the continents.

```python
tsunami = psy.plot.mapplot(
    'ugrid_demo.nc', name='Mesh2_height', load=True,
    maskleq=0, lonlatbox='Japan', cmap='Blues',
    clabel='{desc}', stock_img=True, lsm='50m')
```
To visualize the unstructured grid, we can use the `datagrid` format option. It expects a string or the line properties.

```python
tsunami.docs('datagrid')
```

datagrid
========
Show the grid of the data

This format option 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

1.3. Example Gallery
Visualizing scalar and vector ICON data

This section 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.

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

```python
tsunami.update(datagrid={'c': 'k', 'lw': 0.1})
tsunami.show()
```
We can again zoom in to Europe and use the `datagrid` format option to display the triangular grid.

```python
maps.update(lonlatbox='Europe', datagrid=dict(color='k', linewidth=0.2))
maps.show()
```
The same works for vector data

```python
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

```python
combined = psy.plot.mapcombined('icon_grid_demo.nc', name=['t2m', ['u', 'v']], projection='robin', lonlatbox='Europe', arrowsize=100, cmap='RdBu_r', datagrid={'c': 'k', 'lw': 0.1})
```
The `mapplot` plot method does also not care about the shape of the grid cells. Therefore it can also visualize the ICON edge grid:

```python
maps = psy.plot.mapplot('icon_grid_demo.nc', name='t2m_edge', projection='robin',
                         cmap='Wistia', datagrid=dict(c='b', lw=0.2))
```
The handling unstructured grids

In this section, we go a bit into detail into how psyplot interpretes unstructured grids. If your data is an ICON file or follows the UGRID Conventions, you might want to skip this section, because then psyplot can handle your data automatically.

Interpreting the UGRID Conventions

The best way to specify unstructured grids is to follow the unstructured grids (UGRID) conventions. Variables that follow these conventions are then interpreted by the UGridDecoder class. If one variable has a mesh attribute, psyplot assumes that it follows the UGRID conventions.

Furthermore, psyplot interpretes the location attribute. Attribute can either live on the edge, face or the node of the grid cell. In case of a variable the lives on the node, the data is assumed to be triangular and we plot the data using a Delaunay triangulation.

```python
ds = tsunami[0].psy.base
ds
```

![Image of a world map with color gradient representing data.](https://via.placeholder.com/150)
<table>
<thead>
<tr>
<th>Data variables:</th>
<th>Dimensions without coordinates: Three, Two, nMesh2_edge, nMesh2_node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mesh2_edge_nodes (nMesh2_edge, Two) int32 1 2 0 0 1 1 3 2 3 3 4 4 2 3 ...</td>
<td></td>
</tr>
<tr>
<td>Mesh2_height (time, nMesh2_face) float32 4069.1465 3689.5886 ...</td>
<td></td>
</tr>
<tr>
<td>Mesh2_bathy (time, nMesh2_face) float32 6930.8535 7310.4116 ...</td>
<td></td>
</tr>
<tr>
<td>Mesh2_m_x (time, nMesh2_face) float32 -3.3569866e-10 ...</td>
<td></td>
</tr>
<tr>
<td>Mesh2_m_y (time, nMesh2_face) float32 1.0910523e-10 ...</td>
<td></td>
</tr>
<tr>
<td>Mesh2_u_x (time, nMesh2_face) float32 -8.249855e-14 ...</td>
<td></td>
</tr>
<tr>
<td>Mesh2_u_y (time, nMesh2_face) float32 2.6812806e-14 ...</td>
<td></td>
</tr>
<tr>
<td>Mesh2_level (time, nMesh2_face) int16 7 8 9 10 11 11 11 10 9 8 8 9 ...</td>
<td></td>
</tr>
</tbody>
</table>

Attributes:
- title: netCDF output from StormFlash2d
- institution: University of Hamburg, KlimaCampus
- contact: None
- source: None
- references: None
- comment: None
- Conventions: UGRID-0.9
- creation_date: 2014-10-03 15:29:40 02:00
- modification_date: 2014-10-03 15:29:40 02:00

```
pass
```
Interpreting the CF Conventions

However, there is also another way that follows more closely the standard CF Conventions. This is also the way, that ICON uses, namely the netCDF attributes `coordinates` and `bounds`. These two attributes are decoded CFDecoder (namely the `DataArray.psy.decoder` attribute) and used for the visualization.

To explain it a bit more, we can look into the `icon_grid_demo.nc` file:

```python
ds = maps[0].psy.base
ds
```

This dataset contains two grid definitions, one for variables living on the face of one grid cell (namely `t2m`, `u`, `v`) and one for a variable living on the edges of a grid cell (`t2m_edge`). Which grid is chosen, used depends on the `coordinates` attribute of the specific variable:

```python
print('t2m:', ds.t2m.encoding['coordinates'])
print('t2m_edge:', ds.t2m_edge.encoding['coordinates'])
```

```python
t2m: clat clon
t2m_edge: elat elon
```

The variables mentioned in these coordinates do then have a `bounds` attribute to the variable with the lat-lon information of the vortices for each grid cell:

```python
print(ds.clat.bounds)
ds[ds.clat.bounds]
```
clat_bnds

```
<xarray.DataArray 'clat_bnds' (ncells: 5120, vertices: 3)>
array([[ 0.962634, 0.895414, 0.895414],
       [ 0.962634, 0.958974, 0.895414],
       [ 0.895414, 0.828942, 0.895414],
       ...
       [-0.684167, -0.69024 , -0.755691],
       [-0.755691, -0.818119, -0.746385],
       [-0.746385, -0.674147, -0.684167]])
Coordinates:
  clon      (ncells) float64 ...
  clon_bnds (ncells, vertices) float64 ...
  clat      (ncells) float64 ...
  clat_bnds (ncells, vertices) float64 0.9626 0.8954 0.8954 0.9626 0.959 ...
Dimensions without coordinates: ncells, vertices
```

```
print(ds.elat.bounds)
ds[ds.elat.bounds]
```

elat_bnds

```
<xarray.DataArray 'elat_bnds' (edge: 480, no: 4)>
array([[ 0.809014, 0.737659, 0.809014, 0.918438],
       [ 0.809014, 0.99046 , 1.107149, 0.918438],
       [ 1.107149, 0.99046 , 0.809014, 0.918438],
       ...
       [-0.809014, -0.87296 , -0.740435, -0.688687],
       [-0.740435, -0.59862 , -0.530217, -0.688687],
       [-0.530217, -0.646124, -0.809014, -0.688687]], dtype=float32)
Coordinates:
  elon      (edge) float32 ...
  elon_bnds (edge, no) float32 ...
  elat      (edge) float32 ...
  elat_bnds (edge, no) float32 0.8090137 0.7376585 0.8090137 0.9184382 ...
  * edge     (edge) int64  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 ...
Dimensions without coordinates: no
```

**Identification of unstructured variables**

Psyplot automatically detects, whether the variable is unstructured based on the above mentioned `bounds` coordinate. If the length of the second dimension (here `vertices` or `no`) is larger than 2, then it assumes an unstructured variable.

Alternatively, it assumes that the variable is unstructured if the optional `CDI_grid_type` netCDF attribute or the `grid_type` attribute is equal to 'unstructured':

```
ds.t2m.CDI_grid_type
```

```
'unstructured'
```

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

### 1.3. Example Gallery
Acknowledgement

Thanks [@Try2Code](https://github.com/Try2Code) for providing the ICON data and thanks to Stefan Vater and the Research Group for Numerical Methods in Geosciences from the University of Hamburg for providing the Tsunami simulation file.

### 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.

```python
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.

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

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

```python
maps.update(clabel='{desc}', cmap='RdBu_r')
```
Especially useful format option 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

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

To focus on Europe and disable the latitude-longitude grid, we can set
There are many more formatoption keys that you can explore in the online-documentation or via

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

**************
Color coding formatoptions
**************

<table>
<thead>
<tr>
<th>bounds</th>
<th>cbar</th>
<th>cbarSpacing</th>
<th>cmap</th>
</tr>
</thead>
<tbody>
<tr>
<td>ctickprops</td>
<td>cticksize</td>
<td>ctickweight</td>
<td>extend</td>
</tr>
<tr>
<td>levels</td>
<td>miss_color</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**************
Label formatoptions
**************

<table>
<thead>
<tr>
<th>clabel</th>
<th>clabelprops</th>
<th>clabelsize</th>
<th>clabelweight</th>
</tr>
</thead>
<tbody>
<tr>
<td>figtitle</td>
<td>figtitleprops</td>
<td>figtitlesize</td>
<td>figtitleweight</td>
</tr>
<tr>
<td>text</td>
<td>title</td>
<td>titleprops</td>
<td>titlesize</td>
</tr>
<tr>
<td></td>
<td>titleweight</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**************
Miscellaneous formatoptions
**************
Axis tick formatoptions

Masking formatoptions

Plot formatoptions

Post processing formatoptions

Axes formatoptions
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.

```python
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

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

and we have two possibilities 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>']])
```

1.3. Example Gallery
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 format options that affect the vector color coding start with 'v'.

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

<table>
<thead>
<tr>
<th>color</th>
<th>vcbar</th>
<th>vcbarspacing</th>
<th>vcmap</th>
</tr>
</thead>
<tbody>
<tr>
<td>vbounds</td>
<td>vcticksize</td>
<td>vctickweight</td>
<td>vctickprops</td>
</tr>
<tr>
<td>cbar</td>
<td>bounds</td>
<td>levels</td>
<td>miss_color</td>
</tr>
<tr>
<td>cmap</td>
<td>extend</td>
<td>cbarspacing</td>
<td>cticksize</td>
</tr>
<tr>
<td>ctickweight</td>
<td>ctickprops</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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', vcbarspacing=100, clabel='(desc)', vclabel='Wind Speed [$\text{m/s}$]')
```
Summary

To sum it all up:

- The `mapplot` method visualizes scalar fields
- The `mapvector` method visualizes vector fields
- The `mapcombined` method visualizes scalar and vector fields

```python
# 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
# 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()
```
1.3.3 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).

```python
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` format options to focus on Greenland. Here, we might also want to change the projection since the `northpole` projection implies `clat=0`.

```python
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` format option), 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.

```python
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.

```python
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.

```python
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'])
```

(continues on next page)
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

1.4 Contributing to psy-maps

First off, thanks for taking the time to contribute!
Please see the section about contributing to psyplot for details.

1.5 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.

1.5.1 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

<table>
<thead>
<tr>
<th>lonlatboxes</th>
<th>dict. lonlatboxes for different countries and continents</th>
</tr>
</thead>
<tbody>
<tr>
<td>psy_maps.boxes.lonlatboxes =</td>
<td>{'Afghanistan': [60.0, 74.0, 29.0, 38.0], 'Africa': [-26.0,</td>
</tr>
</tbody>
</table>
Formatoption classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BoxBase</code></td>
<td>Abstract base class for specifying a longitude-latitude box</td>
</tr>
<tr>
<td><code>CenterLat</code></td>
<td>Set the center latitude of the plot</td>
</tr>
<tr>
<td><code>CenterLon</code></td>
<td>Set the center longitude of the plot</td>
</tr>
<tr>
<td><code>ClipAxes</code></td>
<td>Clip the part outside the latitudes of the map extent</td>
</tr>
<tr>
<td><code>CombinedMapVectorPlot</code></td>
<td>Choose the vector plot type</td>
</tr>
<tr>
<td><code>GridBase</code></td>
<td>Abstract base class for x- and y- grid lines</td>
</tr>
<tr>
<td><code>GridColor</code></td>
<td>Set the color of the grid</td>
</tr>
<tr>
<td><code>GridLabelSize</code></td>
<td>Modify the size of the grid tick labels</td>
</tr>
<tr>
<td><code>GridLabels</code></td>
<td>Display the labels of the grid</td>
</tr>
<tr>
<td><code>GridSettings</code></td>
<td>Modify the settings of the grid explicitly</td>
</tr>
<tr>
<td><code>LSM</code></td>
<td>Draw the continents</td>
</tr>
<tr>
<td><code>LonLatBox</code></td>
<td>Set the longitude-latitude box of the data shown</td>
</tr>
<tr>
<td><code>MapDataGrid</code></td>
<td>Show the grid of the data</td>
</tr>
<tr>
<td><code>MapDensity</code></td>
<td>Change the density of the arrows</td>
</tr>
<tr>
<td><code>MapExtent</code></td>
<td>Set the extent of the map</td>
</tr>
<tr>
<td><code>MapPlot2D</code></td>
<td>Choose how to visualize a 2-dimensional scalar data field</td>
</tr>
<tr>
<td><code>MapVectorColor</code></td>
<td>Set the color for the arrows</td>
</tr>
<tr>
<td><code>MapVectorPlot</code></td>
<td>Choose the vector plot type</td>
</tr>
<tr>
<td><code>ProjectionBase</code></td>
<td>Base class for formatoptions that uses cartopy.crs.CRS instances</td>
</tr>
<tr>
<td><code>StockImage</code></td>
<td>Display a stock image on the map</td>
</tr>
<tr>
<td><code>Transform</code></td>
<td>Specify the coordinate system of the data</td>
</tr>
<tr>
<td><code>XGrid</code></td>
<td>Draw vertical grid lines (meridians)</td>
</tr>
<tr>
<td><code>YGrid</code></td>
<td>Draw horizontal grid lines (parallels)</td>
</tr>
</tbody>
</table>

Plotter classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CombinedPlotter</code></td>
<td>Combined 2D plotter and vector plotter on a map</td>
</tr>
<tr>
<td><code>FieldPlotter</code></td>
<td>Plotter for 2D scalar fields on a map</td>
</tr>
<tr>
<td><code>MapPlotter</code></td>
<td>Base plotter for visualizing data on a map</td>
</tr>
<tr>
<td><code>VectorPlotter</code></td>
<td>Plotter for visualizing 2-dimensional vector data on a map</td>
</tr>
</tbody>
</table>

Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>degree_format()</code></td>
<td></td>
</tr>
<tr>
<td><code>format_lats(x, pos)</code></td>
<td></td>
</tr>
<tr>
<td><code>format_lons(x, pos)</code></td>
<td></td>
</tr>
<tr>
<td><code>shiftdata(lonsin, datain, lon_0)</code></td>
<td>Shift longitudes (and optionally data) so that they match map projection region.</td>
</tr>
</tbody>
</table>

```python
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 – Specify 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
lonlatbox
name
priority
requires_clearing

Methods

update(value)

dependencies = ['lonlatbox']
lonlatbox
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

Set the center longitude of the plot

**Parameters**

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

<table>
<thead>
<tr>
<th>dependencies</th>
<th>Built-in mutable sequence.</th>
</tr>
</thead>
<tbody>
<tr>
<td>lonlatbox</td>
<td>lonlatbox Formatoption instance in the plotter</td>
</tr>
<tr>
<td>name</td>
<td>str(object='') -&gt; str</td>
</tr>
<tr>
<td>priority</td>
<td>int([x]) -&gt; integer</td>
</tr>
<tr>
<td>requires_clearing</td>
<td>bool(x) -&gt; bool</td>
</tr>
</tbody>
</table>

---

**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
Built-in mutable sequence.
lonlatbox
lonlatbox Formatoption instance in the plotter
map_extent
map_extent Formatoption instance in the plotter
priority
int([x]) -> 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.

```python
collections = ['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.plotter.CombinedMapVectorPlot(*args, **kwargs)
Bases: psy_maps.plotter.MapVectorPlot
Choose the vector plot type

**Possible types**

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrowsize</td>
<td>arrowsize Formatoption instance in the plotter</td>
</tr>
<tr>
<td>arrowstyle</td>
<td>arrowstyle Formatoption instance in the plotter</td>
</tr>
<tr>
<td>bounds</td>
<td>bounds Formatoption instance in the plotter</td>
</tr>
<tr>
<td>clat</td>
<td>clat Formatoption instance in the plotter</td>
</tr>
<tr>
<td>clip</td>
<td>clip Formatoption instance in the plotter</td>
</tr>
<tr>
<td>clon</td>
<td>clon Formatoption instance in the plotter</td>
</tr>
<tr>
<td>cmap</td>
<td>cmap Formatoption instance in the plotter</td>
</tr>
<tr>
<td>color</td>
<td>color Formatoption instance in the plotter</td>
</tr>
<tr>
<td>data_dependent</td>
<td>bool(x) -&gt; bool</td>
</tr>
<tr>
<td>density</td>
<td>density Formatoption instance in the plotter</td>
</tr>
<tr>
<td>linewidth</td>
<td>linewidth Formatoption instance in the plotter</td>
</tr>
<tr>
<td>lonlatbox</td>
<td>lonlatbox Formatoption instance in the plotter</td>
</tr>
<tr>
<td>transform</td>
<td>transform Formatoption instance in the plotter</td>
</tr>
<tr>
<td>transpose</td>
<td>transpose Formatoption instance in the plotter</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>update(*args, **kwargs)</td>
<td>Method that is call to update the formatoption on the axes</td>
</tr>
</tbody>
</table>

`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**=False,  **make_plot**=True,  **clear**=False,  **enable_post**=False,  ****kwargs)


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

1.5. API Reference
### psy-maps Documentation, Release 1.2.0

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>density</td>
<td>Change the density of the arrows</td>
</tr>
<tr>
<td>arrowsize</td>
<td>Change the size of the arrows</td>
</tr>
<tr>
<td>arrowstyle</td>
<td>Change the style of the arrows</td>
</tr>
</tbody>
</table>

### Plot format options

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot</td>
<td>Choose how to visualize a 2-dimensional scalar data field</td>
</tr>
<tr>
<td>vplot</td>
<td>Choose the vector plot type</td>
</tr>
</tbody>
</table>

### Miscellaneous format options

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

### Post processing format options

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>post</td>
<td>Apply your own postprocessing script</td>
</tr>
<tr>
<td>post_timing</td>
<td>Determine when to run the post format option</td>
</tr>
</tbody>
</table>

### Label format options

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clabel</td>
<td>Show the colorbar label</td>
</tr>
<tr>
<td>clabelprops</td>
<td>Properties of the Colorbar label</td>
</tr>
<tr>
<td>clabelsize</td>
<td>Set the size of the Colorbar label</td>
</tr>
<tr>
<td>clabelweight</td>
<td>Set the fontweight of the Colorbar label</td>
</tr>
<tr>
<td>figtitle</td>
<td>Plot a figure title</td>
</tr>
<tr>
<td>figtitleprops</td>
<td>Properties of the figure title</td>
</tr>
<tr>
<td>figtitlesize</td>
<td>Set the size of the figure title</td>
</tr>
<tr>
<td>figtitlweight</td>
<td>Set the fontweight of the figure title</td>
</tr>
<tr>
<td>text</td>
<td>Add text anywhere on the plot</td>
</tr>
<tr>
<td>title</td>
<td>Show the title</td>
</tr>
<tr>
<td>titleprops</td>
<td>Properties of the title</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>titlesize</td>
<td>Set the size of the title</td>
</tr>
<tr>
<td>titleweight</td>
<td>Set the fontweight of the title</td>
</tr>
<tr>
<td>vclabel</td>
<td>Show the colorbar label of the vector plot</td>
</tr>
<tr>
<td>vclabelprops</td>
<td>Properties of the Vector colorbar label</td>
</tr>
<tr>
<td>vclabelsize</td>
<td>Set the size of the Vector colorbar label</td>
</tr>
<tr>
<td>vclabelweight</td>
<td>Set the fontweight of the Vector colorbar label</td>
</tr>
</tbody>
</table>

**Color coding formatoptions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bounds</td>
<td>Specify the boundaries of the colorbar</td>
</tr>
<tr>
<td>cbar</td>
<td>Specify the position of the colorbars</td>
</tr>
<tr>
<td>cbarspacing</td>
<td>Specify the spacing of the bounds in the colorbar</td>
</tr>
<tr>
<td>cmap</td>
<td>Specify the color map</td>
</tr>
<tr>
<td>color</td>
<td>Set the color for the arrows</td>
</tr>
<tr>
<td>ctickprops</td>
<td>Specify the font properties of the colorbar ticklabels</td>
</tr>
<tr>
<td>cticksize</td>
<td>Specify the font size of the colorbar ticklabels</td>
</tr>
<tr>
<td>ctickweight</td>
<td>Specify the fontweight of the colorbar ticklabels</td>
</tr>
<tr>
<td>extend</td>
<td>Draw arrows at the side of the colorbar</td>
</tr>
<tr>
<td>levels</td>
<td>The levels for the contour plot</td>
</tr>
<tr>
<td>miss_color</td>
<td>Set the color for missing values</td>
</tr>
<tr>
<td>vbounds</td>
<td>Specify the boundaries of the vector colorbar</td>
</tr>
<tr>
<td>vcbar</td>
<td>Specify the position of the vector plot colorbars</td>
</tr>
<tr>
<td>vcBarspacing</td>
<td>Specify the spacing of the bounds in the colorbar</td>
</tr>
<tr>
<td>vcmap</td>
<td>Specify the color map</td>
</tr>
<tr>
<td>vctickprops</td>
<td>Specify the font properties of the colorbar ticklabels</td>
</tr>
<tr>
<td>vcticksize</td>
<td>Specify the font size of the colorbar ticklabels</td>
</tr>
<tr>
<td>vctickweight</td>
<td>Specify the fontweight of the colorbar ticklabels</td>
</tr>
</tbody>
</table>

**Masking formatoptions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maskbetween</td>
<td>Mask data points between two numbers</td>
</tr>
<tr>
<td>maskgeq</td>
<td>Mask data points greater than or equal to a number</td>
</tr>
<tr>
<td>maskgreater</td>
<td>Mask data points greater than a number</td>
</tr>
<tr>
<td>maskleg</td>
<td>Mask data points smaller than or equal to a number</td>
</tr>
<tr>
<td>maskless</td>
<td>Mask data points smaller than a number</td>
</tr>
</tbody>
</table>

**Axes formatoptions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tight</td>
<td>Automatically adjust the plots.</td>
</tr>
</tbody>
</table>

**Axis tick formatoptions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cticklabels</td>
<td>Specify the colorbar ticklabels</td>
</tr>
<tr>
<td>cticks</td>
<td>Specify the tick locations of the colorbar</td>
</tr>
<tr>
<td>vcticklabels</td>
<td>Specify the colorbar ticklabels</td>
</tr>
<tr>
<td>vcticks</td>
<td>Specify the tick locations of the vector colorbar</td>
</tr>
</tbody>
</table>

**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 unstructured 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

\texttt{str} – Plot types can be either

\texttt{quiver} to make a quiver plot

\texttt{stream} to make a stream plot

\texttt{xgrid}

Draw vertical grid lines (meridians)

This format option specifies at which longitudes to draw the meridians.

Possible types

\begin{itemize}
  \item \textit{None} – Don’t draw gridlines (same as \texttt{False})
  \item \texttt{bool} – True: draw gridlines and determine position automatically
                  False: don’t draw gridlines
  \item \texttt{numeric array} – specifies the ticks manually
  \item \texttt{str or list \texttt{[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:
    \begin{itemize}
      \item \texttt{data} plot the ticks exactly where the data is.
      \item \texttt{mid} plot the ticks in the middle of the data.
      \item \texttt{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.
      \item \texttt{rounded\texttt{sym}} Same as \texttt{rounded} above but the ticks are chose such that they are symmetric around zero
      \item \texttt{minmax} Uses the minimum as minimal tick and maximum as maximal tick
      \item \texttt{sym} Same as \texttt{minmax} but symmetric around zero
      \item \texttt{int} – Specifies how many ticks to use with the \texttt{"rounded\'} option. I.e. if integer \texttt{i}, then this is the same as \texttt{[\texttt{"rounded\'}, \texttt{i}]}.\end{itemize}
  \end{itemize}

See also:

\texttt{ygrid}, \texttt{grid\_color}, \texttt{grid\_labels}

\texttt{ygrid}

Draw horizontal grid lines (parallels)

This format option specifies at which latitudes to draw the parallels.

Possible types

\begin{itemize}
  \item \textit{None} – Don’t draw gridlines (same as \texttt{False})
  \item \texttt{bool} – True: draw gridlines and determine position automatically
                  False: don’t draw gridlines
  \item \texttt{numeric array} – specifies the ticks manually
\end{itemize}
• \textit{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:

\texttt{data}  plot the ticks exactly where the data is.
\texttt{mid}  plot the ticks in the middle of the data.
\texttt{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.
\texttt{roundedsym} Same as \texttt{rounded} above but the ticks are chose such that they are symmetric around zero
\texttt{minmax} Uses the minimum as minimal tick and maximum as maximal tick
\texttt{sym} Same as \texttt{minmax} but symmetric around zero

• \texttt{int}  – Specifies how many ticks to use with the \texttt{'rounded'} option. I.e. if integer \(i\), then this is the same as \texttt{[\texttt{\textquoteleft rounded\textquoteleft}, \texttt{i}]}.

\textbf{See also:}
\texttt{xgrid, grid_color, grid_labels}

\textbf{arrowsize}
Change the size of the arrows

\textbf{Possible types}

• \texttt{None}  – make no scaling
• \texttt{float}  – Factor scaling the size of the arrows

\textbf{See also:}
\texttt{arrowstyle, linewidth, density, color}

\textbf{arrowstyle}
Change the style of the arrows

\textbf{Possible types}

\textit{str}  – Any arrow style string (see \texttt{FancyArrowPatch})

\textbf{Notes}

This format option only has an effect for stream plots

\textbf{See also:}
\texttt{arrowsize, linewidth, density, color}

\textbf{clat}
Set the center latitude of the plot

\textbf{Parameters}
**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** – Specify 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)

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 format option 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 con-
  tinents)

• [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 deter-
  mined by the clon and clat formatoptions. Possible strings are (each standing for the specified
  projection)
stock_img
Display a stock image on the map

This format option uses the `cartopy.mpl.geoaxes.GeoAxes.stock_img()` method to display a downscaled 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 format option 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` format options. Possible strings are (each standing for the specified projection):

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cyl</td>
<td><code>cartopy.crs.PlateCarree</code></td>
</tr>
<tr>
<td>robin</td>
<td><code>cartopy.crs.Robinson</code></td>
</tr>
<tr>
<td>moll</td>
<td><code>cartopy.crs.Mollweide</code></td>
</tr>
<tr>
<td>geo</td>
<td><code>cartopy.crs.Geostationary</code></td>
</tr>
<tr>
<td>northpole</td>
<td><code>cartopy.crs.NorthPolarStereo</code></td>
</tr>
<tr>
<td>southpole</td>
<td><code>cartopy.crs.SouthPolarStereo</code></td>
</tr>
<tr>
<td>ortho</td>
<td><code>cartopy.crs.Orthographic</code></td>
</tr>
<tr>
<td>stereo</td>
<td><code>cartopy.crs.Stereographic</code></td>
</tr>
<tr>
<td>near</td>
<td><code>cartopy.crs.NearsisdePerspective</code></td>
</tr>
</tbody>
</table>

**Warning:** An update of the projection clears the axes!

post
Apply your own postprocessing script

This format option lets you apply your own post processing script. Just enter the script as a string and it will be executed. The format option will be made available via the `self` variable.
Possible types

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

**Note:** This format option 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 format option, 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

```python
from psyplot.plotter import Plotter
from xarray import DataArray
plotter = Plotter(DataArray([1, 2, 3]))
# enable the post format option
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` format option is only ran, when it is explicitly updated. However, you can use the `post_timing` format option, to run it automatically. E.g. for running it after every update of the plotter, you can set

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

See also:

- **post** The post processing format option
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)

See also:

`clabel, clabelweight, clabelprops`
clabelweight
Set the fontweight of the Colorbar label

Possible types

- float – a float between 0 and 1000

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 rc-Params['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)

See also:
figtitle, figtitleweight, figtitleprops

**figtitleweight**
Set the fontweight of the figure title

**Possible types**

- float – a float between 0 and 1000

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
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,[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)

See also:

`title, titleweight, titleprops`

titleweight
Set the fontweight of the title

Possible types

- `float` – a float between 0 and 1000

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 [%s]
  – sdesc: %(name)s [%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)

See also:

vclabel, vclabelweight, vclabelprops

vclabelweight
Set the fontweight of the Vector colorbar label

Possible types

• float – a float between 0 and 1000
See also:

| vclabel, vclabelsize, vclabelprops |

**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:

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

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

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

Plot logarithmic bounds:

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

See also:

| cmap | Specifies the colormap |
**cbars**
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:

```python
>>> plotter.update(cobar='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 format option 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 format option 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)

See also:

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

**ctickweight**

Specify the font weight of the colorbar ticklabels

Possible types

- **float** – a float between 0 and 1000
See also:

\texttt{cticksize}, \texttt{ctickprops}, \texttt{cticklabels}, \texttt{cticks}, \texttt{vcticksize}, \texttt{vctickprops}, \texttt{vcticklabels}, \texttt{vcticks}

\texttt{extend}

Draw arrows at the side of the colorbar

\textbf{Possible types}

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

\textbf{levels}

The levels for the contour plot

This format option sets the levels for the filled contour plot and only has an effect if the \texttt{plot} Format option is set to 'contourf'

\textbf{Possible types}

• \texttt{None} – Use the settings from the \texttt{bounds} format option and if this does not specify boundaries, use

11

• \texttt{numeric array} – specifies the ticks manually

• \texttt{str or list [str, \ldots]} – 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:

\texttt{data} plot the ticks exactly where the data is.

\texttt{mid} plot the ticks in the middle of the data.

\texttt{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. |

\texttt{roundedsym} Same as \texttt{rounded} above but the ticks are chose such that they are symmetric around zero

\texttt{minmax} Uses the minimum as minimal tick and maximum as maximal tick

\texttt{sym} Same as minmax but symmetric around zero

• \texttt{int} – Specifies how many ticks to use with the 'rounded' option. I.e. if integer i, then this is the same as ['rounded', i].

\textbf{miss_color}

Set the color for missing values

\textbf{Possible types}

• \texttt{None} – Use the default from the colormap

• \texttt{str, tuple} – Defines the color of the grid.

\textbf{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:

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

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

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

Plot logarithmic bounds:

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

See also:

- cmap Specify 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 format option 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 `psycopplot.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**

**vcticksiz**
Specify the font size of the colorbar ticklabels

**Possible types**

• **float** – The absolute font size in points (e.g., 12)

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

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 above

See also:

maskless, maskleq, maskgeq, maskbetween
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

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 happened after this format option is set to True!

ticklabels
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

ticks
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**

**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 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.

rounded_sym 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 1, then this is the same as ['bounds', i].

See also:

cticklabels, vcticklabels

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


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

Miscellaneous formatoptions

<table>
<thead>
<tr>
<th>interp_bounds</th>
<th>Interpolate grid cell boundaries for 2D plots</th>
</tr>
</thead>
<tbody>
<tr>
<td>clat</td>
<td>Set the center latitude of the plot</td>
</tr>
</tbody>
</table>

Continued on next page
Table 24 – continued from previous page

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clip</td>
<td>Clip the part outside the latitudes of the map extent</td>
</tr>
<tr>
<td>clon</td>
<td>Set the center longitude of the plot</td>
</tr>
<tr>
<td>datagrid</td>
<td>Show the grid of the data</td>
</tr>
<tr>
<td>grid_color</td>
<td>Set the color of the grid</td>
</tr>
<tr>
<td>grid_labels</td>
<td>Display the labels of the grid</td>
</tr>
<tr>
<td>grid_labelsize</td>
<td>Modify the size of the grid tick labels</td>
</tr>
<tr>
<td>grid_settings</td>
<td>Modify the settings of the grid explicitly</td>
</tr>
<tr>
<td>lonlatbbox</td>
<td>Set the longitude-latitude box of the data shown</td>
</tr>
<tr>
<td>lsm</td>
<td>Draw the continents</td>
</tr>
<tr>
<td>map_extent</td>
<td>Set the extent of the map</td>
</tr>
<tr>
<td>projection</td>
<td>Specify the projection for the plot</td>
</tr>
<tr>
<td>stock_img</td>
<td>Display a stock image on the map</td>
</tr>
<tr>
<td>transform</td>
<td>Specify the coordinate system of the data</td>
</tr>
<tr>
<td>xgrid</td>
<td>Draw vertical grid lines (meridians)</td>
</tr>
<tr>
<td>ygrid</td>
<td>Draw horizontal grid lines (parallels)</td>
</tr>
</tbody>
</table>

**Color coding formatoptions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>levels</td>
<td>The levels for the contour plot</td>
</tr>
<tr>
<td>bounds</td>
<td>Specify the boundaries of the colorbar</td>
</tr>
<tr>
<td>cbar</td>
<td>Specify the position of the colorbars</td>
</tr>
<tr>
<td>cbar.spacing</td>
<td>Specify the spacing of the bounds in the colorbar</td>
</tr>
<tr>
<td>cmap</td>
<td>Specify the color map</td>
</tr>
<tr>
<td>ctickprops</td>
<td>Specify the font properties of the colorbar ticklabels</td>
</tr>
<tr>
<td>cticksize</td>
<td>Specify the font size of the colorbar ticklabels</td>
</tr>
<tr>
<td>ctickweight</td>
<td>Specify the fontweight of the colorbar ticklabels</td>
</tr>
<tr>
<td>extend</td>
<td>Draw arrows at the side of the colorbar</td>
</tr>
<tr>
<td>miss_color</td>
<td>Set the color for missing values</td>
</tr>
</tbody>
</table>

**Plot formatoptions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot</td>
<td>Choose how to visualize a 2-dimensional scalar data field</td>
</tr>
</tbody>
</table>

**Post processing formatoptions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>post</td>
<td>Apply your own postprocessing script</td>
</tr>
<tr>
<td>post_timing</td>
<td>Determine when to run the post formatoption</td>
</tr>
</tbody>
</table>

**Label formatoptions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clabel</td>
<td>Show the colorbar label</td>
</tr>
<tr>
<td>clabelprops</td>
<td>Properties of the Colorbar label</td>
</tr>
<tr>
<td>clabelsize</td>
<td>Set the size of the Colorbar label</td>
</tr>
<tr>
<td>clabelweight</td>
<td>Set the fontweight of the Colorbar label</td>
</tr>
<tr>
<td>figtitle</td>
<td>Plot a figure title</td>
</tr>
<tr>
<td>figtitleprops</td>
<td>Properties of the figure title</td>
</tr>
<tr>
<td>figtitlesize</td>
<td>Set the size of the figure title</td>
</tr>
<tr>
<td>figtitleweight</td>
<td>Set the fontweight of the figure title</td>
</tr>
</tbody>
</table>
Table 28 – continued from previous page

<table>
<thead>
<tr>
<th>text</th>
<th>Add text anywhere on the plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>Show the title</td>
</tr>
<tr>
<td>titleprops</td>
<td>Properties of the title</td>
</tr>
<tr>
<td>titlesize</td>
<td>Set the size of the title</td>
</tr>
<tr>
<td>titleweight</td>
<td>Set the fontweight of the title</td>
</tr>
</tbody>
</table>

Masking formatoptions

<table>
<thead>
<tr>
<th>maskbetween</th>
<th>Mask data points between two numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>maskgeq</td>
<td>Mask data points greater than or equal to a number</td>
</tr>
<tr>
<td>maskgreater</td>
<td>Mask data points greater than a number</td>
</tr>
<tr>
<td>maskleq</td>
<td>Mask data points smaller than or equal to a number</td>
</tr>
<tr>
<td>maskless</td>
<td>Mask data points smaller than a number</td>
</tr>
</tbody>
</table>

Axes formatoptions

| tight               | Automatically adjust the plots. |

Axis tick formatoptions

<table>
<thead>
<tr>
<th>cticklabels</th>
<th>Specify the colorbar ticklabels</th>
</tr>
</thead>
<tbody>
<tr>
<td>ticks</td>
<td>Specify the tick locations of the colorbar</td>
</tr>
</tbody>
</table>

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
- numeric array – specifies the ticks manually
• \texttt{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:

\begin{itemize}
  \item \texttt{data} plot the ticks exactly where the data is.
  \item \texttt{mid} plot the ticks in the middle of the data.
  \item \texttt{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.
  \item \texttt{roundedsym} Same as \texttt{rounded} above but the ticks are chose such that they are symmetric around zero
  \item \texttt{minmax} Uses the minimum as minimal tick and maximum as maximal tick
  \item \texttt{sym} Same as \texttt{minmax} but symmetric around zero
  \item \texttt{int} – Specifies how many ticks to use with the \texttt{'rounded'} option. I.e. if integer \texttt{i}, then this is the same as \texttt{['rounded', i]}.
\end{itemize}

\textbf{plot}
Choose how to visualize a 2-dimensional scalar data field

\textbf{Possible types}

\begin{itemize}
  \item \texttt{None} – Don’t make any plotting
  \item \texttt{'mesh'} – Use the \texttt{matplotlib.pyplot.pcolormesh()} function to make the plot or the \texttt{matplotlib.pyplot.tripcolor()} for an unstructured grid
  \item \texttt{'tri'} – Use the \texttt{matplotlib.pyplot.tripcolor()} function to plot data on a triangular grid
  \item \texttt{'contourf'} – Make a filled contour plot using the \texttt{matplotlib.pyplot.contourf()} function or the \texttt{matplotlib.pyplot.tricontourf()} for triangular data. The levels for the contour plot are controlled by the \texttt{levels} format option
  \item \texttt{'tricontourf'} – Make a filled contour plot using the \texttt{matplotlib.pyplot.tricontourf()} function
\end{itemize}

\textbf{clat}
Set the center latitude of the plot

\textbf{Parameters}

\begin{itemize}
  \item \texttt{None} – Let the \texttt{lonlatbox} format option determine the center
  \item \texttt{float} – Specify the center manually
  \item \texttt{str} – A pattern that matches any of the keys in the \texttt{psyplot.rcParams 'extents, boxes'} item (contains user-defined longitude-latitude boxes) or the \texttt{psyplot.plotter.boxes.lonlatboxes} dictionary (contains longitude-latitude boxes of different countries and continents)
\end{itemize}

\textbf{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* – Specify 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)

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 con-
  tinents)

• [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 deter-
  mined by the clon and clat formatoptions. Possible strings are (each standing for the specified
  projection)
### stock_img

Display a stock image on the map

This format option 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 format option 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` format options. 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.NearsidedPerspective`

### xgrid

Draw vertical grid lines (meridians)

This format option 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 format option 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

  See also:

  - **ygrid, grid_color, grid_labels**

  **ygrid**

  Draw horizontal grid lines (parallels)

  This format option specifies at which latitudes to draw the parallels.
**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

**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

•  \textit{bool} – True: defaults to ‘b’ False: Don’t draw any colorbar
•  \textit{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
•  \textit{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

\textit{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 \texttt{matplotlib.colors.Colormap}

Possible types

•  \textit{str} – Strings may be any valid colormap name suitable for the \texttt{matplotlib.cm.get_cmap()} function or one of the color lists defined in the ‘colors.cmaps’ key of the \texttt{psyplot.rcParams} dictionary (including their reversed color maps given via the ‘\_r’ extension).

•  \texttt{matplotlib.colors.Colormap} – The colormap instance to use

See also:

\texttt{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)


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


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.

**post**

Apply your own postprocessing script

This format option lets you apply your own post processing script. Just enter the script as a string and it will be executed. The format option will be made available via the `self` variable

**Possible types**

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

**Note:** This format option 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 format option, 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

```python
from psyplot.plotter import Plotter
from xarray import DataArray
plotter = Plotter(DataArray([1, 2, 3]))
# enable the post format option
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` format option is only ran, when it is explicitly updated. However, you can use the `post_timing` format option, to run it automatically. E.g. for running it after every update of the plotter, you can set

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

**See also:**

`post_timing` Determine the timing of this format option

**post_timing**

Determine when to run the `post` format option

This format option determines, whether the `post` format option 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 format option

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.strptime() 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
  – info: %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)

See also:
clabel, clabelweight, clabelprops

clabelweight
Set the fontweight of the Colorbar label

Possible types

- float – a float between 0 and 1000

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 `rc-Params['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)

See also:

`figtitle, figtitleweight, figtitleprops`

`figtitleweight`

Set the fontweight of the figure title

Possible types

- `float` – a float between 0 and 1000

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.strptime()` 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,[,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.strptime()` 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]
---

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)

See also:

title, titleweight, titleprops

titleweight

Set the fontweight of the title

Possible types

- float – a float between 0 and 1000

See also:

title, titlesize, titleprops

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, maskgeq`

**maskgreater**

Mask data points greater than a number

**Possible types**

- `float` – The floating number to mask above

  **See also:**

  `maskless, maskleq, maskgreater, maskgeq`

**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`

**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 happened after this format option is set to True!

clicklabels
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].
class psy_maps.plotters.GridBase(*args, **kwargs)

Bases: psy_simple.plotters.DataTicksCalculator

Abstract base class for x- and y- grid lines

**Possible types**

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis</td>
<td>The axis string</td>
</tr>
<tr>
<td>connections</td>
<td>Built-in mutable sequence.</td>
</tr>
<tr>
<td>dependencies</td>
<td>Built-in mutable sequence.</td>
</tr>
<tr>
<td>grid_color</td>
<td>grid_color Formatoption instance in the plotter</td>
</tr>
<tr>
<td>grid_labels</td>
<td>grid_labels Formatoption instance in the plotter</td>
</tr>
<tr>
<td>grid_settings</td>
<td>grid_settings Formatoption instance in the plotter</td>
</tr>
<tr>
<td>lonlatbox</td>
<td>lonlatbox Formatoption instance in the plotter</td>
</tr>
<tr>
<td>map_extent</td>
<td>map_extent Formatoption instance in the plotter</td>
</tr>
<tr>
<td>plot</td>
<td>plot Formatoption instance in the plotter</td>
</tr>
<tr>
<td>projection</td>
<td>projection Formatoption instance in the plotter</td>
</tr>
<tr>
<td>transform</td>
<td>transform Formatoption instance in the plotter</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_kwargs(loc)</td>
<td></td>
</tr>
<tr>
<td>remove()</td>
<td>Method to remove the effects of this formatoption</td>
</tr>
<tr>
<td>update(value)</td>
<td>Method that is call to update the formatoption on the axes</td>
</tr>
</tbody>
</table>

- *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', 'lonlatbox', 'map_extent']`  

- `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 = ['xgrid', 'ygrid']
name = 'Color of the latitude-longitude grid'

**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

---

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

Bases: psyplot.plotter.Formatoption

Modify the size of the grid tick labels
Possible types

Attributes

<table>
<thead>
<tr>
<th>dependencies</th>
<th>Built-in mutable sequence.</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>str(object='') -&gt; str</td>
</tr>
<tr>
<td>xgrid</td>
<td>xgrid Formatoption instance in the plotter</td>
</tr>
<tr>
<td>ygrid</td>
<td>ygrid Formatoption instance in the plotter</td>
</tr>
</tbody>
</table>

Methods

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

- float – The absolute font size in points (e.g., 12)

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**
  - Built-in mutable sequence.

- **dependencies**
  - Built-in mutable sequence.

- **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 -- Built-in mutable sequence.
    connections -- Built-in mutable sequence.
    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.

```python
children = ['grid_labels', 'grid_color']
connections = ['xgrid', 'ygrid']
```

```python
grid_color
    grid_color Formatoption instance in the plotter
```

```python
grid_labels
    grid_labels Formatoption instance in the plotter
```

```python
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

```python
xgrid
    xgrid Formatoption instance in the plotter
```

```python
ygrid
    ygrid Formatoption instance in the plotter
```

```python
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.

```python
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)
- 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 Built-in mutable sequence.
- lonlatbox_transformed
- name str(object='') -> str
- priority int([x]) -> 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

```python
class psy_maps.plotters.MapDataGrid(*args, **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**

<table>
<thead>
<tr>
<th><strong>transform</strong></th>
<th>transform Formatoption instance in the plotter</th>
</tr>
</thead>
</table>

**Methods**

<table>
<thead>
<tr>
<th><strong>update</strong> <em>(value)</em></th>
<th>Method that is call to update the formatoption on the axes</th>
</tr>
</thead>
</table>

- **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

```python
Parameters % (Formatoption.parameters)s –
```

**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

```python
class psy_maps.plotters.MapDensity(*args, **kwargs)
Bases: psy_simple.plotters.Density

Change the density of the arrows
```
Possible types

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>plot</strong></td>
<td>plot Formatoption instance in the plotter</td>
</tr>
<tr>
<td><strong>dependencies</strong></td>
<td>Built-in mutable sequence.</td>
</tr>
<tr>
<td><strong>lonlatbox</strong></td>
<td>lonlatbox Formatoption instance in the plotter</td>
</tr>
<tr>
<td><strong>name</strong></td>
<td>str(object='') -&gt; str</td>
</tr>
<tr>
<td><strong>plot</strong></td>
<td>plot Formatoption instance in the plotter</td>
</tr>
<tr>
<td><strong>priority</strong></td>
<td>int([x]) -&gt; integer</td>
</tr>
<tr>
<td><strong>update_after_plot</strong></td>
<td>bool(x) -&gt; bool</td>
</tr>
<tr>
<td><strong>vplot</strong></td>
<td>vplot Formatoption instance in the plotter</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>update(value)</strong></td>
<td>Method that is call to update the formatoption on the axes</td>
</tr>
</tbody>
</table>

- **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**
• None – Don’t make any plotting

• ‘mesh’ – Use the matplotlib.pyplot.pcolormesh() function to make the plot or the matplotlib.pyplot.tripcolor() function to make the plot or the matplotlib.pyplot.tripcolor() function to plot data on an unstructured grid

• ‘tri’ – Use the matplotlib.pyplot.tripcolor() function to plot data on a triangular grid

• ‘contour’ – Make a filled contour plot using the matplotlib.pyplot.contourf() function or the matplotlib.pyplot.tricontourf() function 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

cell_nodes_x
  The unstructured x-boundaries with shape (N, m) where m > 2

clip
  clip Formatoption instance in the plotter

cmap
  cmap Formatoption instance in the plotter

connections
  Built-in mutable sequence.

data_dependent
  bool(x) -> bool

dependencies
  Built-in mutable sequence.

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
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=False, 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

<table>
<thead>
<tr>
<th>ax</th>
<th>Axes instance of the plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>convert_radian</td>
<td>Boolean that is True if triangles with units in radian should be</td>
</tr>
</tbody>
</table>

Miscellaneous formatoptions

<table>
<thead>
<tr>
<th>clat</th>
<th>Set the center latitude of the plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>clip</td>
<td>Clip the part outside the latitudes of the map extent</td>
</tr>
<tr>
<td>clon</td>
<td>Set the center longitude of the plot</td>
</tr>
<tr>
<td>datagrid</td>
<td>Show the grid of the data</td>
</tr>
<tr>
<td>grid_color</td>
<td>Set the color of the grid</td>
</tr>
<tr>
<td>grid_labels</td>
<td>Display the labels of the grid</td>
</tr>
</tbody>
</table>

Continued on next page
Table 54 – continued from previous page

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>grid_labelsize</strong></td>
<td>Modify the size of the grid tick labels</td>
</tr>
<tr>
<td><strong>grid_settings</strong></td>
<td>Modify the settings of the grid explicitly</td>
</tr>
<tr>
<td><strong>lonlatbox</strong></td>
<td>Set the longitude-latitude box of the data shown</td>
</tr>
<tr>
<td><strong>lsm</strong></td>
<td>Draw the continents</td>
</tr>
<tr>
<td><strong>map_extent</strong></td>
<td>Set the extent of the map</td>
</tr>
<tr>
<td><strong>projection</strong></td>
<td>Specify the projection for the plot</td>
</tr>
<tr>
<td><strong>stock_img</strong></td>
<td>Display a stock image on the map</td>
</tr>
<tr>
<td><strong>transform</strong></td>
<td>Specify the coordinate system of the data</td>
</tr>
<tr>
<td><strong>xgrid</strong></td>
<td>Draw vertical grid lines (meridians)</td>
</tr>
<tr>
<td><strong>ygrid</strong></td>
<td>Draw horizontal grid lines (parallels)</td>
</tr>
</tbody>
</table>

**Post processing formatoptions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>post</strong></td>
<td>Apply your own postprocessing script</td>
</tr>
<tr>
<td><strong>post_timing</strong></td>
<td>Determine when to run the <strong>post</strong> formatoption</td>
</tr>
</tbody>
</table>

**Color coding formatoptions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bounds</strong></td>
<td>Specify the boundaries of the colorbar</td>
</tr>
<tr>
<td><strong>cbar</strong></td>
<td>Specify the position of the colorbars</td>
</tr>
<tr>
<td><strong>cbarspacing</strong></td>
<td>Specify the spacing of the bounds in the colorbar</td>
</tr>
<tr>
<td><strong>cmap</strong></td>
<td>Specify the color map</td>
</tr>
<tr>
<td><strong>ctickprops</strong></td>
<td>Specify the font properties of the colorbar ticklabels</td>
</tr>
<tr>
<td><strong>cticksize</strong></td>
<td>Specify the font size of the colorbar ticklabels</td>
</tr>
<tr>
<td><strong>ctickweight</strong></td>
<td>Specify the fontweight of the colorbar ticklabels</td>
</tr>
<tr>
<td><strong>extend</strong></td>
<td>Draw arrows at the side of the colorbar</td>
</tr>
</tbody>
</table>

**Label formatoptions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>clabel</strong></td>
<td>Show the colorbar label</td>
</tr>
<tr>
<td><strong>clabelprops</strong></td>
<td>Properties of the Colorbar label</td>
</tr>
<tr>
<td><strong>clabelsize</strong></td>
<td>Set the size of the Colorbar label</td>
</tr>
<tr>
<td><strong>clabelweight</strong></td>
<td>Set the fontweight of the Colorbar label</td>
</tr>
</tbody>
</table>

**Axis tick formatoptions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cticklabels</strong></td>
<td>Specify the colorbar ticklabels</td>
</tr>
<tr>
<td><strong>cticks</strong></td>
<td>Specify the tick locations of the colorbar</td>
</tr>
</tbody>
</table>

**ax**

Axes instance of the plot

**clat**

Set the center latitude of the plot

**Parameters**

- **None** – Let the **lonlatbox** formatoption determine the center
- **float** – Specify the center manually
- **str** – A pattern that matches any of the keys in the **psyplot.rcParams** 'extents'.

1.5. API Reference 99
boxes’ item (contains user-defined longitude-latitude boxes) or the psyplot.
plotter.boxes.lonlatboxes dictionary (contains longitude-latitude boxes of dif-
ferent 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* – Specify 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 dif-
ferent 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)

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` format option
- 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` format option)
- *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 format option sets the extent of the plot. For choosing the region for the data, see the `lonlatbox` format option.

See also:

`lonlatbox`

**projection**

Specify the projection for the plot.

This format option 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` format options. 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 format option 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 format option 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):

<table>
<thead>
<tr>
<th>String</th>
<th>Projection</th>
</tr>
</thead>
<tbody>
<tr>
<td>cyl</td>
<td>cartopy.crs.PlateCarree</td>
</tr>
<tr>
<td>robin</td>
<td>cartopy.crs.Robinson</td>
</tr>
<tr>
<td>moll</td>
<td>cartopy.crs.Mollweide</td>
</tr>
<tr>
<td>geo</td>
<td>cartopy.crs.Geostationary</td>
</tr>
<tr>
<td>northpole</td>
<td>cartopy.crs.NorthPolarStereo</td>
</tr>
<tr>
<td>southpole</td>
<td>cartopy.crs.SouthPolarStereo</td>
</tr>
<tr>
<td>ortho</td>
<td>cartopy.crs.Orthographic</td>
</tr>
<tr>
<td>stereo</td>
<td>cartopy.crs.Stereographic</td>
</tr>
<tr>
<td>near</td>
<td>cartopy.crs.NearsidePerspective</td>
</tr>
</tbody>
</table>

**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

**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

**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:

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

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

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

Plot logarithmic bounds:

```python
>>> 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**

1.5. API Reference
Draw a colorbar at the bottom and left of the axes:

```python
>>> 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 format option 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

clickprops
Specify the font properties of the colorbar ticklabels

**Possible types**

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

See also:

`csticksize, cstickweight, cticklabels, cticks, vcticksize, vctickweight, vcticklabels, vcticks`

cclicksize
Specify the font size of the colorbar ticklabels

**Possible types**

- `float` – The absolute font size in points (e.g., 12)

See also:

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

Specify the font weight of the colorbar ticklabels

Possible types

- float – a float between 0 and 1000

See also:

cticksize, tickprops, cticklabels, ticks, vcticks, vcticksize, vctickprops, vcticklabels, vticks

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

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)

See also:
clabel, clabelweight, clabelprops

clabelweight
Set the fontweight of the Colorbar label

Possible types

• float – a float between 0 and 1000

See also:
clabel, clabelsize, clabelprops

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, vticksize, vtickweight, vtickprops

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`

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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bounds</code></td>
<td>bounds Formatoption instance in the plotter</td>
</tr>
<tr>
<td><code>cmap</code></td>
<td>cmap Formatoption instance in the plotter</td>
</tr>
<tr>
<td><code>plot</code></td>
<td>plot Formatoption instance in the plotter</td>
</tr>
<tr>
<td><code>transpose</code></td>
<td>transpose Formatoption instance in the plotter</td>
</tr>
</tbody>
</table>

- **float** – Determines the greyness
- **color** – Defines the same color for all arrows. The string can be either a html hex string (e.g. ‘#eefff’), 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add2format_coord(x, y)</td>
<td>Additional information for the format_coord()</td>
</tr>
<tr>
<td>set_value(value, *args, **kwargs)</td>
<td>Set (and validate) the value in the plotter.</td>
</tr>
</tbody>
</table>

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrowsize</td>
<td>arrowsize Formatoption instance in the plotter</td>
</tr>
<tr>
<td>arrowstyle</td>
<td>arrowstyle Formatoption instance in the plotter</td>
</tr>
<tr>
<td>bounds</td>
<td>bounds Formatoption instance in the plotter</td>
</tr>
<tr>
<td>clat</td>
<td>clat Formatoption instance in the plotter</td>
</tr>
<tr>
<td>clip</td>
<td>clip Formatoption instance in the plotter</td>
</tr>
<tr>
<td>clon</td>
<td>clon Formatoption instance in the plotter</td>
</tr>
<tr>
<td>cmap</td>
<td>cmap Formatoption instance in the plotter</td>
</tr>
<tr>
<td>color</td>
<td>color Formatoption instance in the plotter</td>
</tr>
<tr>
<td>data_dependent</td>
<td>bool(x) -&gt; bool</td>
</tr>
<tr>
<td>density</td>
<td>density Formatoption instance in the plotter</td>
</tr>
<tr>
<td>dependencies</td>
<td>Built-in mutable sequence.</td>
</tr>
<tr>
<td>linewidth</td>
<td>linewidth Formatoption instance in the plotter</td>
</tr>
<tr>
<td>lonlatbox</td>
<td>lonlatbox Formatoption instance in the plotter</td>
</tr>
<tr>
<td>transform</td>
<td>transform Formatoption instance in the plotter</td>
</tr>
<tr>
<td>transpose</td>
<td>transpose Formatoption instance in the plotter</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clat</td>
<td>clat Formatoption instance in the plotter</td>
</tr>
<tr>
<td>clon</td>
<td>clon Formatoption instance in the plotter</td>
</tr>
<tr>
<td>connections</td>
<td>Built-in mutable sequence.</td>
</tr>
<tr>
<td>dependencies</td>
<td>Built-in mutable sequence.</td>
</tr>
<tr>
<td>name</td>
<td>str(object='') -&gt; str</td>
</tr>
<tr>
<td>priority</td>
<td>int([x]) -&gt; integer</td>
</tr>
<tr>
<td>requires_clearing</td>
<td>an update of this formatoption requires that the axes is cleared</td>
</tr>
<tr>
<td>transform</td>
<td>transform Formatoption instance in the plotter</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialize_plot(value[, clear])</td>
<td>Initialize the plot and set the projection for the axes</td>
</tr>
<tr>
<td>update(value)</td>
<td>Update the formatoption</td>
</tr>
</tbody>
</table>

- `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)

<table>
<thead>
<tr>
<th>String</th>
<th>Projection Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>cyl</td>
<td><code>cartopy.crs.PlateCarree</code></td>
</tr>
<tr>
<td>robin</td>
<td><code>cartopy.crs.Robinson</code></td>
</tr>
<tr>
<td>moll</td>
<td><code>cartopy.crs.Mollweide</code></td>
</tr>
<tr>
<td>geo</td>
<td><code>cartopy.crs.Geostationary</code></td>
</tr>
<tr>
<td>northpole</td>
<td><code>cartopy.crs.NorthPolarStereo</code></td>
</tr>
<tr>
<td>southpole</td>
<td><code>cartopy.crs.SouthPolarStereo</code></td>
</tr>
<tr>
<td>ortho</td>
<td><code>cartopy.crs.Orthographic</code></td>
</tr>
<tr>
<td>stereo</td>
<td><code>cartopy.crs.Stereographic</code></td>
</tr>
<tr>
<td>near</td>
<td><code>cartopy.crs.NearsidesPerspective</code></td>
</tr>
</tbody>
</table>

**Warning:** An update of the projection clears the axes!

clat
- clat Formatoption instance in the plotter

clon
- clon Formatoption instance in the plotter

connections = ['transform']
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


```python
requires_clearing = True

transform

transform Formatoption instance in the plotter

update(value)

Update the formatoption

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

```python
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 |
| `orthogonal` | cartopy.crs.Orthographic |
| `stereo` | cartopy.crs.Stereographic |
| `near` | cartopy.crs.NearsidesPerspective |

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.

```python
def get_kwargs(value, clon=None, clat=None):
    projection_kwargs = {'cyl': ['central_longitude'], 'geo': ['central_longitude'], 'moll': ['central_longitude'], ...
    projections = {'cyl': <class 'cartopy.crs.PlateCarree'>, 'geo': <class 'cartopy.crs.Geostationary'>, ...
```

```python
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 downsampling version of the Natural Earth shaded relief raster on the map

### Possible types

**Attributes**

<table>
<thead>
<tr>
<th>connections</th>
<th>Built-in mutable sequence.</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>str(object='') -&gt; str</td>
</tr>
<tr>
<td>plot</td>
<td>plot Formatoption instance in the plotter</td>
</tr>
<tr>
<td>priority</td>
<td>int([]) -&gt; integer</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>remove()</th>
<th>Method to remove the effects of this formatoption</th>
</tr>
</thead>
<tbody>
<tr>
<td>update(value)</td>
<td>Method that is call to update the formatoption on the axes</td>
</tr>
</tbody>
</table>

`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 formats 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.

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

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**

<table>
<thead>
<tr>
<th>connections</th>
<th>Built-in mutable sequence.</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>str(object=')') -&gt; str</td>
</tr>
<tr>
<td>plot</td>
<td>plot Formatoption instance in the plotter</td>
</tr>
<tr>
<td>priority</td>
<td>int([x]) -&gt; integer</td>
</tr>
<tr>
<td>vplot</td>
<td>vplot Formatoption instance in the plotter</td>
</tr>
</tbody>
</table>

**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):

<table>
<thead>
<tr>
<th>Formatoption</th>
<th>Cartopy CRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>cyl</td>
<td>cartopy.crs.PlateCarree</td>
</tr>
<tr>
<td>robin</td>
<td>cartopy.crs.Robinson</td>
</tr>
<tr>
<td>moll</td>
<td>cartopy.crs.Mollweide</td>
</tr>
<tr>
<td>geo</td>
<td>cartopy.crs.Geostationary</td>
</tr>
<tr>
<td>northpole</td>
<td>cartopy.crs.NorthPolarStereo</td>
</tr>
<tr>
<td>southpole</td>
<td>cartopy.crs.SouthPolarStereo</td>
</tr>
<tr>
<td>ortho</td>
<td>cartopy.crs.Orthographic</td>
</tr>
<tr>
<td>stereo</td>
<td>cartopy.crs.Stereographic</td>
</tr>
<tr>
<td>near</td>
<td>cartopy.crs.NearsidePerspective</td>
</tr>
</tbody>
</table>

### 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=False, 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 scalar fields

*CombinedPlotter* for combined scalar and vector fields

### Color coding format options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>Set the color for the arrows</td>
</tr>
<tr>
<td>bounds</td>
<td>Specify the boundaries of the vector colorbar</td>
</tr>
<tr>
<td>cbar</td>
<td>Specify the position of the vector plot colorbars</td>
</tr>
<tr>
<td>cbarspacing</td>
<td>Specify the spacing of the bounds in the colorbar</td>
</tr>
<tr>
<td>cmap</td>
<td>Specify the color map</td>
</tr>
<tr>
<td>ctipprops</td>
<td>Specify the font properties of the colorbar ticklabels</td>
</tr>
<tr>
<td>ctipsize</td>
<td>Specify the font size of the colorbar ticklabels</td>
</tr>
<tr>
<td>ctipweight</td>
<td>Specify the fontweight of the colorbar ticklabels</td>
</tr>
<tr>
<td>extend</td>
<td>Draw arrows at the side of the colorbar</td>
</tr>
</tbody>
</table>

### Vector plot format options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>density</td>
<td>Change the density of the arrows</td>
</tr>
<tr>
<td>arrowsize</td>
<td>Change the size of the arrows</td>
</tr>
<tr>
<td>arrowstyle</td>
<td>Change the style of the arrows</td>
</tr>
</tbody>
</table>

### Plot format options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot</td>
<td>Choose the vector plot type</td>
</tr>
</tbody>
</table>

### Post processing format options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>post</td>
<td>Apply your own postprocessing script</td>
</tr>
<tr>
<td>post_timing</td>
<td>Determine when to run the post formatoption</td>
</tr>
</tbody>
</table>

### Miscellaneous format options

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clabel</td>
<td>Show the colorbar label</td>
</tr>
<tr>
<td>clabelprops</td>
<td>Properties of the Colorbar label</td>
</tr>
<tr>
<td>clabelsize</td>
<td>Set the size of the Colorbar label</td>
</tr>
<tr>
<td>clabelweight</td>
<td>Set the fontweight of the Colorbar label</td>
</tr>
<tr>
<td>figtitle</td>
<td>Plot a figure title</td>
</tr>
<tr>
<td>figtitleprops</td>
<td>Properties of the figure title</td>
</tr>
<tr>
<td>figtitlesize</td>
<td>Set the size of the figure title</td>
</tr>
<tr>
<td>figtitleweight</td>
<td>Set the fontweight of the figure title</td>
</tr>
<tr>
<td>text</td>
<td>Add text anywhere on the plot</td>
</tr>
<tr>
<td>title</td>
<td>Show the title</td>
</tr>
<tr>
<td>titleprops</td>
<td>Properties of the title</td>
</tr>
<tr>
<td>titlesize</td>
<td>Set the size of the title</td>
</tr>
<tr>
<td>titleweight</td>
<td>Set the fontweight of the title</td>
</tr>
</tbody>
</table>

Masking format options

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maskbetween</td>
<td>Mask data points between two numbers</td>
</tr>
<tr>
<td>maskgeq</td>
<td>Mask data points greater than or equal to a number</td>
</tr>
<tr>
<td>maskgreater</td>
<td>Mask data points greater than a number</td>
</tr>
<tr>
<td>maskleq</td>
<td>Mask data points smaller than or equal to a number</td>
</tr>
<tr>
<td>maskless</td>
<td>Mask data points smaller than a number</td>
</tr>
</tbody>
</table>

Axes format options

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tight</td>
<td>Automatically adjust the plots.</td>
</tr>
</tbody>
</table>

Axis tick format options

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cticklabels</td>
<td>Specify the colorbar ticklabels</td>
</tr>
<tr>
<td>ticks</td>
<td>Specify the tick locations of the vector colorbar</td>
</tr>
</tbody>
</table>

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. ‘#eefff’), 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:

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

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

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

Plot logarithmic bounds:

```python
>>> 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 format option 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`

**cticks**
Specify the font size of the colorbar ticklabels

**Possible types**

- **float** – The absolute font size in points (e.g., 12)

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

See also:

*cticks, cticksize, ctickprops, cticklabels, vcticks, 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*
**post**

Apply your own postprocessing script

This format option lets you apply your own post processing script. Just enter the script as a string and it will be executed. The format option will be made available via the `self` variable

**Possible types**

- `None` – Don’t do anything
- `str` – The post processing script as string

**Note:** This format option 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 format option, 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

```python
from psyplot.plotter import Plotter
from xarray import DataArray
plotter = Plotter(DataArray([1, 2, 3]))
# enable the post format option
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` format option is only ran, when it is explicitly updated. However, you can use the `post_timing` format option, to run it automatically. E.g. for running it after every update of the plotter, you can set

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

**See also:**

- `post Timing` Determine the timing of this format option

**post Timing**

Determine when to run the `post` format option

This format option determines, whether the `post` format option 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

clat
Set the center latitude of the plot

Parameters

• None – Let the lonlatbox formatoption determine the center
• float – Specify 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 dif-
ferent 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 – Specify 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 dif-
ferent 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)

See also:

`grid_color, grid_labels, grid_labelsize, xgrid, ygrid`

**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)
stock_img

Display a stock image on the map

This format option 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 format option 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` format options. Possible strings are (each standing for the specified projection)

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Cartopy CRS Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>cyl</td>
<td>cartopy.crs.PlateCarree</td>
</tr>
<tr>
<td>robin</td>
<td>cartopy.crs.Robinson</td>
</tr>
<tr>
<td>moll</td>
<td>cartopy.crs.Mollweide</td>
</tr>
<tr>
<td>geo</td>
<td>cartopy.crs.Geostationary</td>
</tr>
<tr>
<td>northpole</td>
<td>cartopy.crs.NorthPolarStereo</td>
</tr>
<tr>
<td>southpole</td>
<td>cartopy.crs.SouthPolarStereo</td>
</tr>
<tr>
<td>ortho</td>
<td>cartopy.crs.Orthographic</td>
</tr>
<tr>
<td>stereo</td>
<td>cartopy.crs.Stereographic</td>
</tr>
<tr>
<td>near</td>
<td>cartopy.crs.NearsidePerspective</td>
</tr>
</tbody>
</table>

**xgrid**

Draw vertical grid lines (meridians)

This format option 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 format option 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)

See also:

`clabel, clabelweight, clabelprops`

`clabelweight`
Set the fontweight of the Colorbar label

Possible types

- **float** – a float between 0 and 1000

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.strptime()` 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
  - dinfo: %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 `rc-Params['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

**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,[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)

See also:

title, titleweight, titleprops

titleweight

Set the fontweight of the title

Possible types

• float – a float between 0 and 1000

See also:

title, titlesize, titleprops

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`

**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 happened after this format option is set to True!

clicklabels
Specify the colorbar ticklabels

Possible types

• str – A format string 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

clicks
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 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:

\textit{cticklabels}, \textit{vcticklabels}

class psy\_maps.plotters.XGrid(*args, **kwargs)
Bases: \textit{psy\_maps.plotters.GridBase}

Draw vertical grid lines (meridians)

This \texttt{formatoption} specifies at which longitudes to draw the meridians.

**Possible types**

**Attributes**

\begin{array}{ll}
\textbf{array} & \text{The numpy array of the data} \\
\textbf{axis} & \text{str(object=’)’} \rightarrow \text{str} \\
\textbf{clon} & \text{clon Formatoption instance in the plotter} \\
\textbf{dependencies} & \text{Built-in mutable sequence.} \\
\textbf{grid\_color} & \text{grid\_color Formatoption instance in the plotter} \\
\textbf{grid\_labels} & \text{grid\_labels Formatoption instance in the plotter} \\
\textbf{grid\_settings} & \text{grid\_settings Formatoption instance in the plotter} \\
\textbf{lonlatbox} & \text{lonlatbox Formatoption instance in the plotter} \\
\textbf{map\_extent} & \text{map\_extent Formatoption instance in the plotter} \\
\textbf{name} & \text{str(object=’)’} \rightarrow \text{str} \\
\textbf{plot} & \text{plot Formatoption instance in the plotter} \\
\textbf{projection} & \text{projection Formatoption instance in the plotter} \\
\textbf{transform} & \text{transform Formatoption instance in the plotter}
\end{array}

- \textit{None} – Don’t draw gridlines (same as \texttt{False})
- \textit{bool} – True: draw gridlines and determine position automatically \texttt{False}: don’t draw gridlines
- \textit{numeric array} – specifies the ticks manually
- \text{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:
  - \textbf{data} plot the ticks exactly where the data is.
  - \textbf{mid} plot the ticks in the middle of the data.
  - \textbf{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.
  - \textbf{roundedsym} Same as \textit{rounded} above but the ticks are chose such that they are symmetric around zero
  - \textbf{minmax} Uses the minimum as minimal tick and maximum as maximal tick
  - \textbf{sym} Same as minmax but symmetric around zero
- \textit{int} – Specifies how many ticks to use with the \texttt{’rounded’} option. I.e. if integer \texttt{i}, then this is the same as \texttt{[’rounded’, i]}.

See also:

\textit{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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>The numpy array of the data</td>
</tr>
<tr>
<td>axis</td>
<td>str(object='') -&gt; str</td>
</tr>
<tr>
<td>grid_color</td>
<td>grid_color Formatoption instance in the plotter</td>
</tr>
<tr>
<td>grid_labels</td>
<td>grid_labels Formatoption instance in the plotter</td>
</tr>
<tr>
<td>grid_settings</td>
<td>grid_settings Formatoption instance in the plotter</td>
</tr>
<tr>
<td>lonlatbox</td>
<td>lonlatbox Formatoption instance in the plotter</td>
</tr>
<tr>
<td>map_extent</td>
<td>map_extent Formatoption instance in the plotter</td>
</tr>
<tr>
<td>name</td>
<td>str(object='') -&gt; str</td>
</tr>
<tr>
<td>plot</td>
<td>plot Formatoption instance in the plotter</td>
</tr>
<tr>
<td>projection</td>
<td>projection Formatoption instance in the plotter</td>
</tr>
<tr>
<td>transform</td>
<td>transform Formatoption instance in the plotter</td>
</tr>
</tbody>
</table>

• 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 Documentation, Release 1.2.0

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

```python
ProjectionValidator(key, valid[, ignorecase])
```
valid is a list of legal strings

Functions

```python
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

```python
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

class psy_maps.plugin.get_versions(requirements=True)
class psy_maps.plugin.patch_prior_1_0(plotter_d, versions)
Patch psyMaps 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

```python
psy_maps.plugin.patches = {('psyplot.plotter.maps', 'CombinedPlotter'): <function patch_prior_1_0>, ('psyplot.plotter.maps', 'MapPlotter'): <function patch_prior_1_0>, ('psyplot.plotter.maps', 'VectorPlotter'): <function patch_prior_1_0},
```

142 Chapter 1. Documentation
patches to apply when loading a project

```python
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, 30
psy_maps.boxes, 30
psy_maps.plotters, 31
psy_maps.plugin, 142
psy_maps.version, 143
clabelprops (psy_maps.plotters.VectorPlotter attribute), 132
clabelsize (psy_maps.plotters.CombinedPlotter attribute), 49
clabelsize (psy_maps.plotters.FieldPlotter attribute), 77
clabelsize (psy_maps.plotters.MapPlotter attribute), 110
clabelsize (psy_maps.plotters.VectorPlotter attribute), 132
clabelweight (psy_maps.plotters.CombinedPlotter attribute), 49
clabelweight (psy_maps.plotters.FieldPlotter attribute), 78
clabelweight (psy_maps.plotters.MapPlotter attribute), 110
clabelweight (psy_maps.plotters.VectorPlotter attribute), 133
clat (psy_maps.plotters.CombinedMapVectorPlot attribute), 37
clat (psy_maps.plotters.CombinedPlotter attribute), 42
clat (psy_maps.plotters.FieldPlotter attribute), 67
clat (psy_maps.plotters.MapPlotter attribute), 99
clat (psy_maps.plotters.MapVectorPlot attribute), 113
clat (psy_maps.plotters.Projection attribute), 144
clat (psy_maps.plotters.VectorPlotter attribute), 126
clip (psy_maps.plotters.CombinedMapVectorPlot attribute), 37
clip (psy_maps.plotters.CombinedPlotter attribute), 43
clip (psy_maps.plotters.FieldPlotter attribute), 67
clip (psy_maps.plotters.MapPlot2D attribute), 97
clip (psy_maps.plotters.MapPlotter attribute), 100
clip (psy_maps.plotters.MapVectorPlot attribute), 113
clip (psy_maps.plotters.VectorPlotter attribute), 126
ClipAxes (class in psy_maps.plotters), 35
clon (psy_maps.plotters.CombinedMapVectorPlot attribute), 37
clon (psy_maps.plotters.CombinedPlotter attribute), 43
clon (psy_maps.plotters.FieldPlotter attribute), 68
clon (psy_maps.plotters.MapPlotter attribute), 100
clon (psy_maps.plotters.MapVectorPlot attribute), 113
clon (psy_maps.plotters.Projection attribute), 144
clon (psy_maps.plotters.VectorPlotter attribute), 126
clon (psy_maps.plotters.XGrid attribute), 140
cmap (psy_maps.plotters.CombinedMapVectorPlot attribute), 37
cmap (psy_maps.plotters.CombinedPlotter attribute), 56
cmap (psy_maps.plotters.FieldPlotter attribute), 74
cmap (psy_maps.plotters.MapPlot2D attribute), 97
cmap (psy_maps.plotters.MapPlotter attribute), 108
cmap (psy_maps.plotters.MapVectorColor attribute), 112
cmap (psy_maps.plotters.MapVectorPlot attribute), 113
cmap (psy_maps.plotters.VectorPlotter attribute), 123
color (psy_maps.plotters.CombinedMapVectorPlot attribute), 37
color (psy_maps.plotters.CombinedPlotter attribute), 56
color (psy_maps.plotters.MapVectorPlot attribute), 113
color (psy_maps.plotters.VectorPlotter attribute), 121
CombinedMapVectorPlot (class in psy_maps.plotters), 36
CombinedPlotter (class in psy_maps.plotters), 37
connections (psy_maps.plotters.ClipAxes attribute), 36
connections (psy_maps.plotters.GridBase attribute), 85
connections (psy_maps.plotters.GridColor attribute), 86
connections (psy_maps.plotters.GridLabels attribute), 88
connections (psy_maps.plotters.GridSettings attribute), 90
connections (psy_maps.plotters.MapPlot2D attribute), 97
connections (psy_maps.plotters.Projection attribute), 114
connections (psy_maps.plotters.StockImage attribute), 117
connections (psy_maps.plotters.Transform attribute), 118
convert_radian (psy_maps.plotters.MapPlotter attribute), 100
cticklabels (psy_maps.plotters.CombinedPlotter attribute), 62
cticklabels (psy_maps.plotters.FieldPlotter attribute), 83
cticklabels (psy_maps.plotters.MapPlotter attribute), 110
cticklabels (psy_maps.plotters.VectorPlotter attribute), 138
ctickprops (psy_maps.plotters.CombinedPlotter attribute), 57
ctickprops (psy_maps.plotters.FieldPlotter attribute), 74
ctickprops (psy_maps.plotters.MapPlotter attribute), 108
ctickprops (psy_maps.plotters.VectorPlotter attribute), 123
cticks (psy_maps.plotters.CombinedPlotter attribute), 62
cticks (psy_maps.plotters.FieldPlotter attribute), 83
cticks (psy_maps.plotters.MapPlotter attribute), 110
cticks (psy_maps.plotters.VectorPlotter attribute), 138
cticksize (psy_maps.plotters.CombinedPlotter attribute), 57
cticksize (psy_maps.plotters.FieldPlotter attribute), 75
cticksize (psy_maps.plotters.MapPlotter attribute), 108
cticksize (psy_maps.plotters.VectorPlotter attribute), 123
ctickweight (psy_maps.plotters.CombinedPlotter attribute), 57
ctickweight (psy_maps.plotters.FieldPlotter attribute), 75
ctickweight (psy_maps.plotters.MapPlotter attribute), 108
ctickweight (psy_maps.plotters.VectorPlotter attribute), 124
data_dependent (psy_maps.plotters.CombinedMapVectorPlot attribute), 37
data_dependent (psy_maps.plotters.MapPlot2D attribute), 97
data_dependent (psy_maps.plotters.MapVectorPlot attribute), 113
data_dependent (psy_maps.plotters.VectorPlotter attribute), 121
dependencies (psy_maps.plotters.CenterLat attribute), 33
dependencies (psy_maps.plotters.CenterLon attribute), 34
dependencies (psy_maps.plotters.GridBase attribute), 85
dependencies (psy_maps.plotters.GridLabels attribute), 88
dependencies (psy_maps.plotters.GridLabelSize attribute), 87
dependencies (psy_maps.plotters.LonLatBox attribute), 93
dependencies (psy_maps.plotters.MapExtent attribute), 96
dependencies (psy_maps.plotters.MapPlot2D attribute), 97
dependencies (psy_maps.plotters.MapVectorPlot attribute), 113
dependencies (psy_maps.plotters.Projection attribute), 114
dependencies (psy_maps.plotters.XGrid attribute), 140
draw_circle () (psy_maps.plotters.ClipAxes method), 36
extend (psy_maps.plotters.CombinedPlotter attribute), 58
extend (psy_maps.plotters.FieldPlotter attribute), 75
extend (psy_maps.plotters.MapPlotter attribute), 109
extend (psy_maps.plotters.VectorPlotter attribute), 124
FieldPlotter (class in psy_maps.plotters), 64
figtitle (psy_maps.plotters.CombinedPlotter attribute), 50
figtitle (psy_maps.plotters.FieldPlotter attribute), 78
figtitle (psy_maps.plotters.VectorPlotter attribute), 133
figtitleprops (psy_maps.plotters.CombinedPlotter attribute), 50
figtitleprops (psy_maps.plotters.FieldPlotter attribute), 79
figtitleprops (psy_maps.plotters.VectorPlotter attribute), 134
figtitlesize (psy_maps.plotters.CombinedPlotter attribute), 51
figtitlesize (psy_maps.plotters.FieldPlotter attribute), 79
figtitlesize (psy_maps.plotters.VectorPlotter attribute), 134
figtitleweight (psy_maps.plotters.CombinedPlotter attribute), 51
figtitleweight (psy_maps.plotters.FieldPlotter attribute), 79
figtitleweight (psy_maps.plotters.VectorPlotter attribute), 134
format_lats () (in module psy_maps.plotters), 141
format_lons () (in module psy_maps.plotters), 141
get_kwargs () (psy_maps.plotters.GridBase method), 85
get_kwargs () (psy_maps.plotters.ProjectionBase method), 116
get_versions () (in module psy_maps.plugin), 142

Index 151
grid_color (psy_maps.plotters.CombinedPlotter attribute), 43
grid_color (psy_maps.plotters.FieldPlotter attribute), 68
grid_color (psy_maps.plotters.GridBase attribute), 85
grid_color (psy_maps.plotters.GridSettings attribute), 90
grid_color (psy_maps.plotters.MapPlotter attribute), 100
grid_color (psy_maps.plotters.VectorPlotter attribute), 127
grid_color (psy_maps.plotters.XGrid attribute), 140
grid_color (psy_maps.plotters.YGrid attribute), 141
grid_labels (psy_maps.plotters.CombinedPlotter attribute), 44
grid_labels (psy_maps.plotters.FieldPlotter attribute), 68
grid_labels (psy_maps.plotters.GridBase attribute), 85
grid_labels (psy_maps.plotters.GridSettings attribute), 90
grid_labels (psy_maps.plotters.MapPlotter attribute), 101
grid_labels (psy_maps.plotters.VectorPlotter attribute), 127
grid_labels (psy_maps.plotters.XGrid attribute), 140
grid_labels (psy_maps.plotters.YGrid attribute), 141
grid_labelsize (psy_maps.plotters.CombinedPlotter attribute), 44
grid_labelsize (psy_maps.plotters.FieldPlotter attribute), 68
grid_labelsize (psy_maps.plotters.GridBase attribute), 85
grid_labelsize (psy_maps.plotters.GridSettings attribute), 90
grid_labelsize (psy_maps.plotters.MapPlotter attribute), 101
grid_labelsize (psy_maps.plotters.VectorPlotter attribute), 127
grid_labelsize (psy_maps.plotters.XGrid attribute), 140
grid_labelsize (psy_maps.plotters.YGrid attribute), 141
GridLabelSize (class in psy_maps.plotters), 86
GridSettings (class in psy_maps.plotters), 89

image (psy_maps.plotters.StockImage attribute), 117
initialize_plot () (psy_maps.plotters.Projection method), 114
interp_bounds (psy_maps.plotters.CombinedPlotter attribute), 44
interp_bounds (psy_maps.plotters.FieldPlotter attribute), 66
interp_bounds (psy_maps.plotters.MapPlot2D attribute), 97

levels (psy_maps.plotters.CombinedPlotter attribute), 58
levels (psy_maps.plotters.FieldPlotter attribute), 66
levels (psy_maps.plotters.MapPlot2D attribute), 97
linewidth (psy_maps.plotters.CombinedMapVectorPlot attribute), 37
linewidth (psy_maps.plotters.CombinedPlotter attribute), 45
linewidth (psy_maps.plotters.MapVectorPlot attribute), 113
linewidth (psy_maps.plotters.VectorPlotter attribute), 127
lola_from_pattern () (psy_maps.plotters.BoxBase method), 32
LonLatBox (class in psy_maps.plotters), 91
lonlatbox (psy_maps.plotters.CenterLat attribute), 33
lonlatbox (psy_maps.plotters.CenterLon attribute), 34
lonlatbox (psy_maps.plotters.ClipAxes attribute), 36
lonlatbox (psy_maps.plotters.CombinedMapVectorPlot attribute), 37
lonlatbox (psy_maps.plotters.CombinedPlotter attribute), 45
lonlatbox (psy_maps.plotters.FieldPlotter attribute), 69
lonlatbox (psy_maps.plotters.GridBase attribute), 85
lonlatbox (psy_maps.plotters.MapExtent attribute), 96
lonlatbox (psy_maps.plotters.MapPlot2D attribute), 97
lonlatbox (psy_maps.plotters.MapPlotter attribute), 101
lonlatbox (psy_maps.plotters.MapVectorPlot attribute), 113
lonlatbox (psy_maps.plotters.VectorPlotter attribute), 128
lonlatbox (psy_maps.plotters.XGrid attribute), 140
lonlatbox (psy_maps.plotters.YGrid attribute), 141
lonlatbox_transformed

(map_extent (psy_maps.plotters.ClipAxes attribute), 93)
lonlatboxes (in module psy_maps.boxes), 30
LSM (class in psy_maps.plotters), 90
lsm (psy_maps.plotters.CombinedPlotter attribute), 46
lsm (psy_maps.plotters.FieldPlotter attribute), 70
lsm (psy_maps.plotters.LSM attribute), 91
lsm (psy_maps.plotters.MapPlotter attribute), 102
lsm (psy_maps.plotters.VectorPlotter attribute), 128

M
map_extent (psy_maps.plotters.ClipAxes attribute), 36
map_extent (psy_maps.plotters.CombinedPlotter attribute), 46
map_extent (psy_maps.plotters.FieldPlotter attribute), 70
map_extent (psy_maps.plotters.GridBase attribute), 85
map_extent (psy_maps.plotters.MapPlotter attribute), 102
map_extent (psy_maps.plotters.VectorPlotter attribute), 129
map_extent (psy_maps.plotters.XGrid attribute), 140
map_extent (psy_maps.plotters.YGrid attribute), 141
mapcombined (psyplot.project.plot attribute), 6
MapDataGrid (class in psy_maps.plotters), 94
MapDensity (class in psy_maps.plotters), 94
MapExtent (class in psy_maps.plotters), 95
mapplot (psyplot.project.plot attribute), 4
MapPlot2D (class in psy_maps.plotters), 96
MapPlotter (class in psy_maps.plotters), 98
mapvector (psyplot.project.plot attribute), 5
MapVectorColor (class in psy_maps.plotters), 111
MapVectorPlot (class in psy_maps.plotters), 112
mask_outside () (psy_maps.plotters.LonLatBox method), 93
maskbetween (psy_maps.plotters.CombinedPlotter attribute), 61
maskbetween (psy_maps.plotters.FieldPlotter attribute), 81
maskbetween (psy_maps.plotters.VectorPlotter attribute), 136
maskgeq (psy_maps.plotters.CombinedPlotter attribute), 61
maskgeq (psy_maps.plotters.FieldPlotter attribute), 82
maskgeq (psy_maps.plotters.VectorPlotter attribute), 137
maskgreater (psy_maps.plotters.CombinedPlotter attribute), 61
maskgreater (psy_maps.plotters.FieldPlotter attribute), 82
maskgreater (psy_maps.plotters.VectorPlotter attribute), 137
maskleq (psy_maps.plotters.CombinedPlotter attribute), 137
maskleq (psy_maps.plotters.FieldPlotter attribute), 61
maskleq (psy_maps.plotters.VectorPlotter attribute), 82
maskless (psy_maps.plotters.CombinedPlotter attribute), 62
maskless (psy_maps.plotters.FieldPlotter attribute), 82
maskless (psy_maps.plotters.VectorPlotter attribute), 137
miss_color (psy_maps.plotters.CombinedPlotter attribute), 58
miss_color (psy_maps.plotters.FieldPlotter attribute), 75

N
name (psy_maps.plotters.CenterLat attribute), 33
name (psy_maps.plotters.CenterLon attribute), 34
name (psy_maps.plotters.GridColor attribute), 86
name (psy_maps.plotters.GridLabels attribute), 88
name (psy_maps.plotters.GridLabelSize attribute), 87
name (psy_maps.plotters.GridSettings attribute), 90
name (psy_maps.plotters.LonLatBox attribute), 93
name (psy_maps.plotters.LSM attribute), 91
name (psy_maps.plotters.MapExtent attribute), 96
name (psy_maps.plotters.Projection attribute), 114
name (psy_maps.plotters.StockImage attribute), 117
name (psy_maps.plotters.Transform attribute), 118
name (psy_maps.plotters.XGrid attribute), 140
name (psy_maps.plotters.YGrid attribute), 141

P
patch_prior_1_0 () (in module psy_maps.plugin), 142
patches (in module psy_maps.plugin), 142
plot (psy_maps.plotters.CombinedPlotter attribute), 40
plot (psy_maps.plotters.FieldPlotter attribute), 67
plot (psy_maps.plotters.GridBase attribute), 85
plot (psy_maps.plotters.MapDensity attribute), 95
plot (psy_maps.plotters.MapExtent attribute), 96
plot (psy_maps.plotters.MapVectorColor attribute), 112
plot (psy_maps.plotters.StockImage attribute), 117
plot (psy_maps.plotters.Transform attribute), 118
plot (psy_maps.plotters.VectorPlotter attribute), 121
plot (psy_maps.plotters.XGrid attribute), 140
plot (psy_maps.plotters.YGrid attribute), 141
post (psy_maps.plotters.CombinedPlotter attribute), 47
post (psy_maps.plotters.FieldPlotter attribute), 76
post (psy_maps.plotters.MapPlotter attribute), 105
post (psy_maps.plotters.VectorPlotter attribute), 124
post_timing (psy_maps.plotters.CombinedPlotter attribute), 48
post_timing (psy_maps.plotters.FieldPlotter attribute), 76
post_timing (psy_maps.plotters.MapPlotter attribute), 106
post_timing (psy_maps.plotters.VectorPlotter attribute), 125
priority (psy_maps.plotters.CenterLat attribute), 33
priority (psy_maps.plotters.CenterLon attribute), 34
priority (psy_maps.plotters.ClipAxes attribute), 36
priority (psy_maps.plotters.LonLatBox attribute), 93
priority (psy_maps.plotters.MapExtent attribute), 96
priority (psy_maps.plotters.Projection attribute), 114
priority (psy_maps.plotters.StockImage attribute), 117
priority (psy_maps.plotters.Transform attribute), 118
projection (class in psy_maps.plotters), 113
projection (psy_maps.plotters.CombinedPlotter attribute), 46
projection (psy_maps.plotters.FieldPlotter attribute), 70
projection (psy_maps.plotters.GridBase attribute), 85
projection (psy_maps.plotters.GridLabels attribute), 89
projection (psy_maps.plotters.MapPlotter attribute), 103
projection (psy_maps.plotters.VectorPlotter attribute), 129
projection (psy_maps.plotters.XGrid attribute), 140
projection (psy_maps.plotters.YGrid attribute), 141
projection_kwargs (psy_maps.plotters.ProjectionBase attribute), 116
ProjectionBase (class in psy_maps.plotters), 115
projections (psy_maps.plotters.ProjectionBase attribute), 116
ProjectionValidator (class in psy_maps.plugin), 142
psy_maps (module), 30
psy_maps.boxes (module), 30
psy_maps.plotters (module), 31
psy_maps.plugin (module), 142
psy_maps.version (module), 143

requires_clearing (psy_maps.plotters.CenterLat attribute), 33
requires_clearing (psy_maps.plotters.CenterLon attribute), 35
requires_clearing (psy_maps.plotters.LonLatBox attribute), 93
requires_clearing (psy_maps.plotters.Projection attribute), 114

S
set_projection() (psy_maps.plotters.ProjectionBase method), 116
set_value() (psy_maps.plotters.GridSettings method), 90
set_value() (psy_maps.plotters.MapVectorPlot method), 113
shiftdata() (in module psy_maps.plotters), 141
shiftdata() (psy_maps.plotters.LonLatBox method), 93
stock_img (psy_maps.plotters.CombinedPlotter attribute), 47
stock_img (psy_maps.plotters.FieldPlotter attribute), 71
stock_img (psy_maps.plotters.MapPlotter attribute), 103
stock_img (psy_maps.plotters.VectorPlotter attribute), 130
StockImage (class in psy_maps.plotters), 116

T
text (psy_maps.plotters.CombinedPlotter attribute), 51
text (psy_maps.plotters.FieldPlotter attribute), 79
text (psy_maps.plotters.VectorPlotter attribute), 134
tight (psy_maps.plotters.CombinedPlotter attribute), 62
tight (psy_maps.plotters.FieldPlotter attribute), 82
tight (psy_maps.plotters.VectorPlotter attribute), 137
title (psy_maps.plotters.CombinedPlotter attribute), 52
title (psy_maps.plotters.FieldPlotter attribute), 80
title (psy_maps.plotters.VectorPlotter attribute), 135
titleprops (psy_maps.plotters.CombinedPlotter attribute), 53
titleprops (psy_maps.plotters.FieldPlotter attribute), 81
titleprops (psy_maps.plotters.VectorPlotter attribute), 136
titlesize (psy_maps.plotters.CombinedPlotter attribute), 53
titlesize (psy_maps.plotters.FieldPlotter attribute), 81
titlesize (psy_maps.plotters.VectorPlotter attribute), 136

R
cParams (in module psy_maps.plugin), 143
remove() (psy_maps.plotters.ClipAxes method), 36
remove() (psy_maps.plotters.GridAxes method), 85
remove() (psy_maps.plotters.LSM method), 91
remove() (psy_maps.plotters.MapPlot2D method), 97
remove() (psy_maps.plotters.StockImage method), 117

Index
titleweight (psy_maps.plotters.CombinedPlotter
attribute), 53
titleweight (psy_maps.plotters.FieldPlotter
attribute), 81
titleweight (psy_maps.plotters.VectorPlotter
attribute), 136
to_degree() (psy_maps.plotters.LonLatBox method), 93
Transform (class in psy_maps.plotters), 117
transform (psy_maps.plotters.CombinedMapVectorPlot
attribute), 37
transform (psy_maps.plotters.FieldPlotter
attribute), 47
transform (psy_maps.plotters.FieldPlotter
attribute), 71
transform (psy_maps.plotters.GridBase attribute), 85
transform (psy_maps.plotters.GridLabels attribute), 89
transform (psy_maps.plotters.LonLatBox attribute), 94
transform (psy_maps.plotters.MapDataGrid
attribute), 94
transform (psy_maps.plotters.MapPlot2D attribute), 98
transform (psy_maps.plotters.MapPlotter attribute), 103
transform (psy_maps.plotters.MapVectorPlot
attribute), 113
transform (psy_maps.plotters.Projection attribute), 115
transform (psy_maps.plotters.VectorPlotter attribute), 130
transform (psy_maps.plotters.XGrid attribute), 140
transform (psy_maps.plotters.YGrid attribute), 141
transpose (psy_maps.plotters.CombinedMapVectorPlot
attribute), 37
transpose (psy_maps.plotters.MapVectorColor
attribute), 112
transpose (psy_maps.plotters.MapVectorPlot
attribute), 113
update() (psy_maps.plotters.LSM method), 91
update() (psy_maps.plotters.MapDataGrid method), 94
update() (psy_maps.plotters.MapExtent method), 96
update() (psy_maps.plotters.Projection method), 115
update() (psy_maps.plotters.StockImage method), 117
update() (psy_maps.plotters.Transform method), 118
update_after_plot (psy_maps.plotters.MapExtent
attribute), 96
update_array() (psy_maps.plotters.LonLatBox
method), 94
validate_dict_yaml() (in module psy_maps.plugin), 143
validate_grid() (in module psy_maps.plugin), 143
validate_lonlatbox() (in module psy_maps.plugin), 143
vbounds (psy_maps.plotters.CombinedPlotter
attribute), 58
vcbarspacing (psy_maps.plotters.CombinedPlotter
attribute), 59
vcbar (psy_maps.plotters.CombinedPlotter attribute), 59
vcbarwidth (psy_maps.plotters.CombinedPlotter
attribute), 59
vclabel (psy_maps.plotters.CombinedPlotter
attribute), 60
vclabelprops (psy_maps.plotters.CombinedPlotter
attribute), 53
vclabelsize (psy_maps.plotters.CombinedPlotter
attribute), 64
vclabelweight (psy_maps.plotters.CombinedPlotter
attribute), 54
vcmap (psy_maps.plotters.CombinedPlotter attribute), 60
vcticklabels (psy_maps.plotters.CombinedPlotter
attribute), 63
vctickprops (psy_maps.plotters.CombinedPlotter
attribute), 63
vcticks (psy_maps.plotters.CombinedPlotter
attribute), 60
vcticksize (psy_maps.plotters.CombinedPlotter
attribute), 63
vctickwidth (psy_maps.plotters.CombinedPlotter
attribute), 60
VectorPlotter (class in psy_maps.plotters), 118
vplot (psy_maps.plotters.CombinedPlotter attribute), 40
vplot (psy_maps.plotters.MapExtent attribute), 96
vplot (psy_maps.plotters.Transform attribute), 118
X
XGrid (class in psy_maps.plotters), 139
xgrid (psy_maps.plotters.CombinedPlotter attribute), 41
xgrid (psy_maps.plotters.FieldPlotter attribute), 71
xgrid (psy_maps.plotters.GridColor attribute), 86
xgrid (psy_maps.plotters.GridLabels attribute), 89
xgrid (psy_maps.plotters.GridLabelSize attribute), 87
xgrid (psy_maps.plotters.GridSettings attribute), 90
xgrid (psy_maps.plotters.MapPlotter attribute), 104
xgrid (psy_maps.plotters.VectorPlotter attribute), 130

Y
YGrid (class in psy_maps.plotters), 140
ygrid (psy_maps.plotters.CombinedPlotter attribute), 41
ygrid (psy_maps.plotters.FieldPlotter attribute), 72
ygrid (psy_maps.plotters.GridColor attribute), 86
ygrid (psy_maps.plotters.GridLabels attribute), 89
ygrid (psy_maps.plotters.GridLabelSize attribute), 87
ygrid (psy_maps.plotters.GridSettings attribute), 90
ygrid (psy_maps.plotters.MapPlotter attribute), 104
ygrid (psy_maps.plotters.VectorPlotter attribute), 131