psy-strat Documentation

Release 0.1.0

Philipp Sommer

Sep 19, 2019

CONTENTS

Welcome to the psyplot plugin for stratigraphic visualization. This package defines the `stratplot()` function that visualizes a dataframe in a stratigraphic plot.

Additionally, this plugin interfaces with the psyplot GUI package to allow you an interactive manipulation of the stratigraphic plot.

See the *Example Gallery* for more information.

CHAPTER

ONE

DOCUMENTATION

1.1 Installation

1.1.1 How to install

Installation using conda

We highly recommend to use conda for installing psy-strat. After downloading the installer from anaconda, you can install psy-strat simply via:

```
$ conda install -c chilipp psy-strat
```

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-strat
```

1.1.2 Running the tests

First, clone out the github repository. Then run:

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

or after having install pytest:

```
$ py.test
```

1.2 Example Gallery

psy-strat visualizes stratigraphic data on bar diagrams, area diagrams, line diagrams and stacked diagrams. The examples in this gallery show you the effects of the different parameters of the *psy_strat.stratplot.stratplot()* function.

1.2.1 Complex pollen diagram

This is an example of bars, stacked, lines and area plots in one single diagram.

We display pollen count data from the site Hoya del Castillo in Spain collected by *Davis and Stevenson (2007)* obtained from the European pollen database (EPD). We extended this data by two additional columns: summer (JJA) and winter (DJF) temperature.

We start by importing the necessary libraries. We use `pandas` to read and manage the pollen data and `stratplot` for it's visualization.

```
import psyplot.project as psy
import pandas as pd
from psy_strat.stratplot import stratplot
import matplotlib as mpl
```

Adjusting the figure size and dpi improves the readability of the plots in this notebook.

```
    mpl.rcParams['figure.figsize'] = (16, 10)
mpl.rcParams['figure.dpi'] = 150
```

The data is stored as a comma-separated text file that can be loaded into a `pandas.DataFrame` using pandas `pandas.read_csv` function:

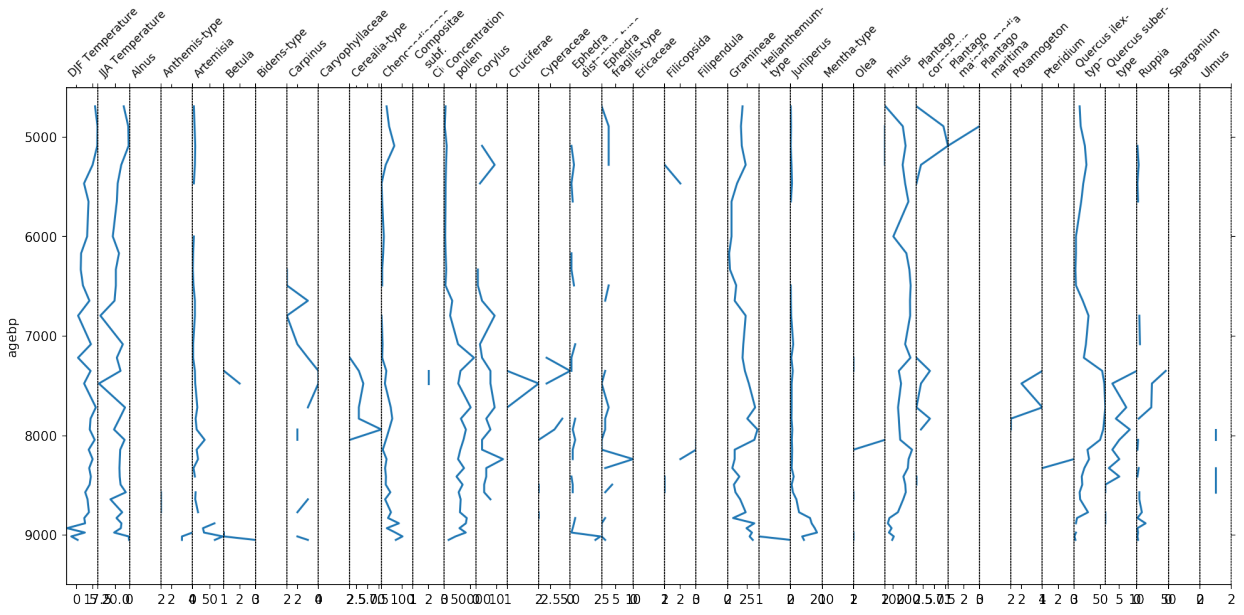
```
df = pd.read_csv('pollen-data.csv', index_col='agebp')
print(df.shape)
df.head(5)
```

```
(34, 37)
```

This data contains 34 samples and 37 columns. Note that we chose the DataFrame to be indexed by the `'agebp'` column. This will be the vertical axis of our stratigraphic diagram that is shared between all the variables.

Now we can display this dataframe using `stratplot`:

```
sp, groupers = stratplot(df)
```



You see now one plot for each column in the above dataframe. However, this figure is not very informative. The x-axis labels are hardly readable and the taxa all have different scalings. We can significantly improve this plot by grouping the variables together.

Luckily, the EPD comes with a mapping from taxon name to group names. This mapping is stored in the tab separated file `epd-groups.tsv`:

```
groups = pd.read_csv('epd-groups.tsv', delimiter='\t', index_col=0)
groups.head(5)
```

Using this data, we can group the columns in our DataFrame using the following function:

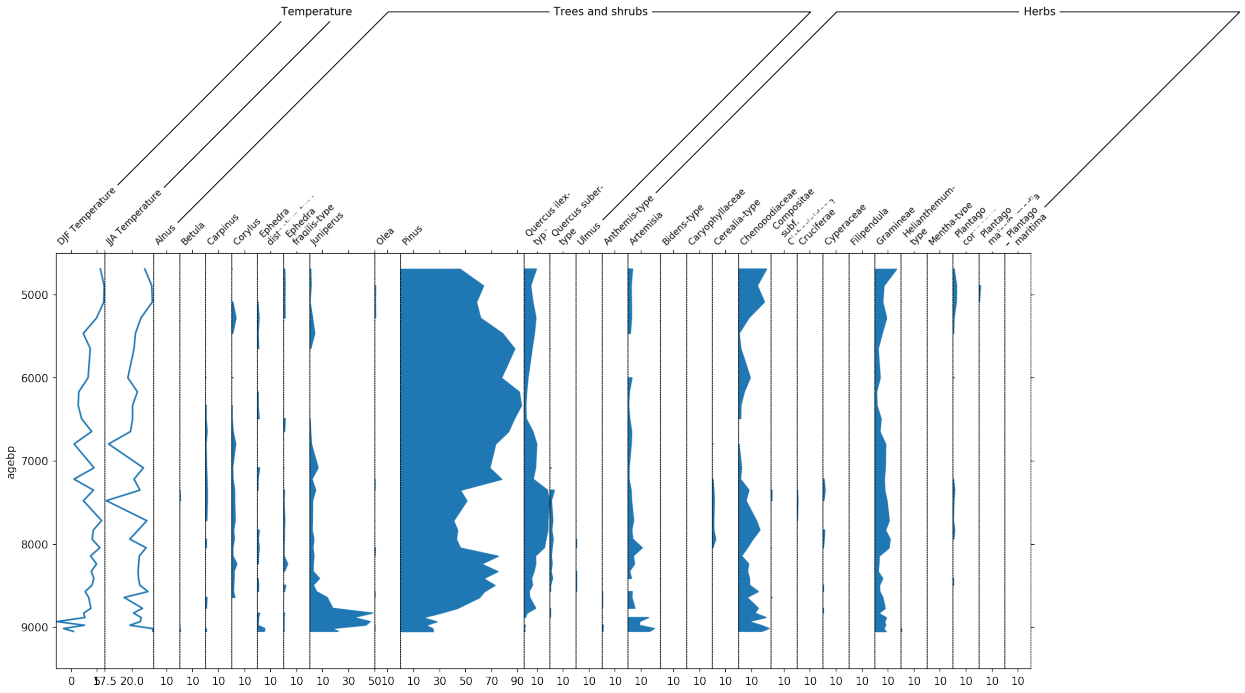
```
def grouper(col):
    if 'Temperature' in col:
        return 'Temperature'
    else:
        return groups.groupname.loc[col]
```

And have a look into how this functions groups our data:

```
group2taxon = pd.DataFrame.from_dict(df.groupby(grouper, axis=1).groups, orient='index
↪').T
group2taxon.fillna('')
```

For our pollen diagram, we are actually only interested in Temperature, Herbs, Trees and shrubs. Therefore we can exclude the other groups from our plot using the *exclude* parameter of `stratplot`. Additionally we transform the pollen counts into percentages to get a better scaling of the diagram. Since *Trees and shrubs* and *Herbs* should both be considered when calculating the percentages, we additionally put them into a larger *Pollen* group.

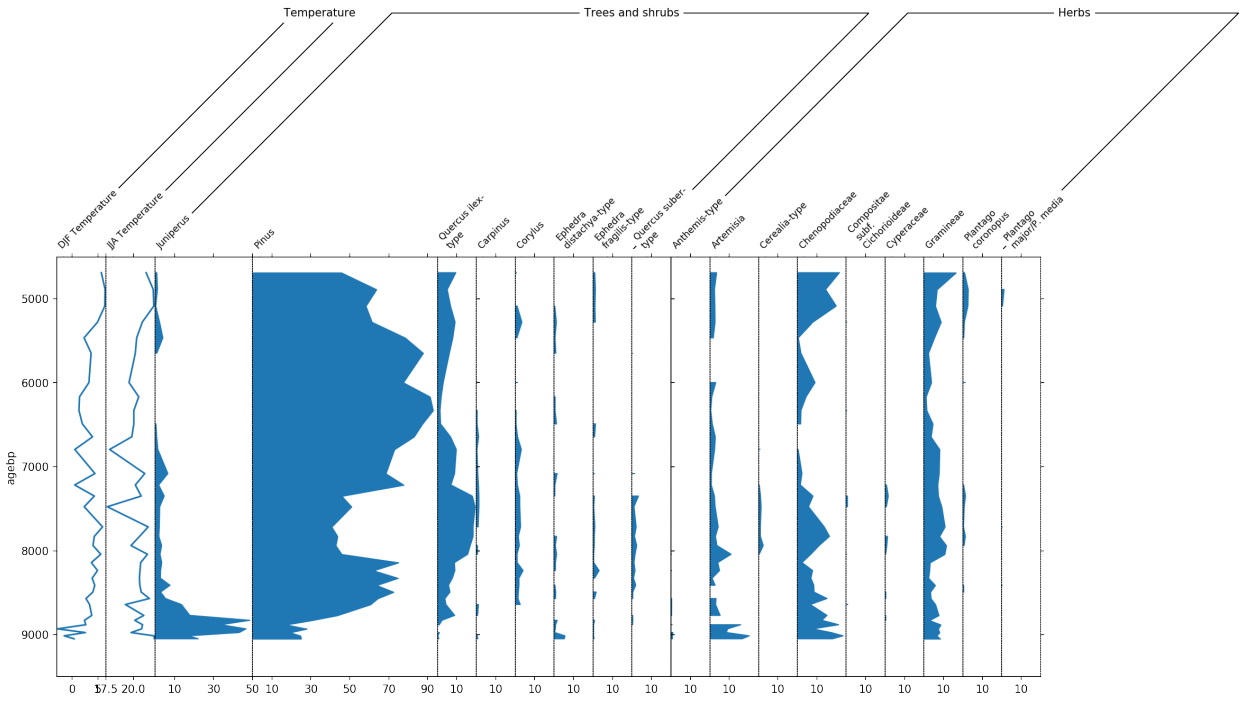
```
sp, groupers = stratplot(
    df, grouper,
    widths={'Temperature': 0.1, 'Pollen': 0.9},
    percentages=['Pollen'],
    subgroups={'Pollen': ['Trees and shrubs', 'Herbs']},
    exclude=['Aquatics', 'Dwarf shrubs', 'Helophytes', 'Nonpollen',
             'Vascular cryptogams (Pteridophytes)'])
```



This diagram already looks much better, however there are still to many taxa in there that have only very little amount of data. Therefore we use the *thresh* parameter to set a threshold of 1%. Every taxon now that is never above 1% will not be displayed.

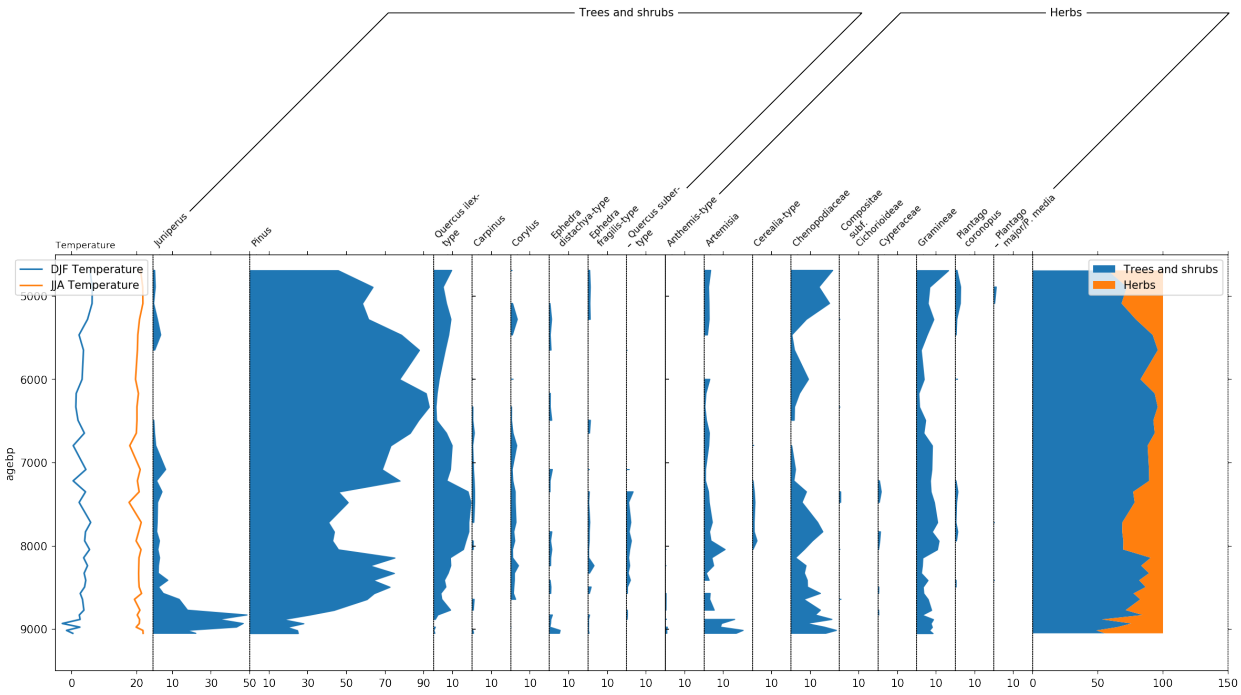
Additionally we apply a new order to the columns such that we put Juniperus, Pinus and Quercus ilex-type, and then the other trees and shrubs to the left. Note that, if you run this in the psyplot GUI, you can change the order of the variables more easily without scripting. However, it is also not so difficult using the `reorder` method of the grouper for the *pollen* variables.

```
sp, groupers = stratplot(
    df, grouper,
    thresh=1.0,
    widths={'Temperature': 0.1, 'Pollen': 0.9},
    percentages=['Pollen'],
    subgroups={'Pollen': ['Trees and shrubs', 'Herbs']},
    exclude=['Aquatics', 'Dwarf shrubs', 'Helophytes', 'Nonpollen',
             'Vascular cryptogams (Pteridophytes)'])

# apply a new order where we first display Juniperus, Pinus and Quercus, and then the␣
↪rest
pollen_grouper = groupers[1]
first_taxa = ['Juniperus', 'Pinus', 'Quercus ilex-type']
remaining_trees = group2taxon['Trees and shrubs'][
    ~group2taxon['Trees and shrubs'].isin(first_taxa)].dropna().tolist()

neworder = first_taxa + remaining_trees
pollen_grouper.reorder(neworder)
```



Finally, we can display the temperature columns in one single diagram because they share the same units. This is done using the *all_in_one* parameter. Additionally we can include a sum of the different pollen subgroups using the *summed* parameter.

```
sp, groupers = stratplot(
    df, grouper,
    thresh=1.0,
    all_in_one=['Temperature'],
    summed=['Trees and shrubs', 'Herbs'],
    widths={'Temperature': 0.1, 'Pollen': 0.9},
    percentages=['Pollen'],
    subgroups={'Pollen': ['Trees and shrubs', 'Herbs']},
    exclude=['Aquatics', 'Dwarf shrubs', 'Helophytes', 'Nonpollen',
             'Vascular cryptogams (Pteridophytes)'])

# apply a new order where we first display Juniperus, Pinus and Quercus, and then the␣
↪rest
pollen_grouper = groupers[1]
first_taxa = ['Juniperus', 'Pinus', 'Quercus ilex-type']
remaining_trees = group2taxon['Trees and shrubs'][
    ~group2taxon['Trees and shrubs'].isin(first_taxa)].dropna().tolist()

neworder = first_taxa + remaining_trees
pollen_grouper.reorder(neworder)
```



Last but not least, let's talk a bit about the final layout. To better distinguish herbs from Trees and shrubs, we can display this group using a bar plot by making use of the *use_bars* parameter of `stratplot`. Furthermore we decrease the size of the groupers and increase the height of the plot using the `trunc_height` parameter.

Additionally we can use the psyplot framework to do some changes to the colors, etc..:

- display trees in green
- exaggerate the trees by a factor of 4
- highlight low pollen occurences below 1% with a +
- change the JJA temperature curve to red
- change the legendlabels for temperature
- change x- and y-label for temperature

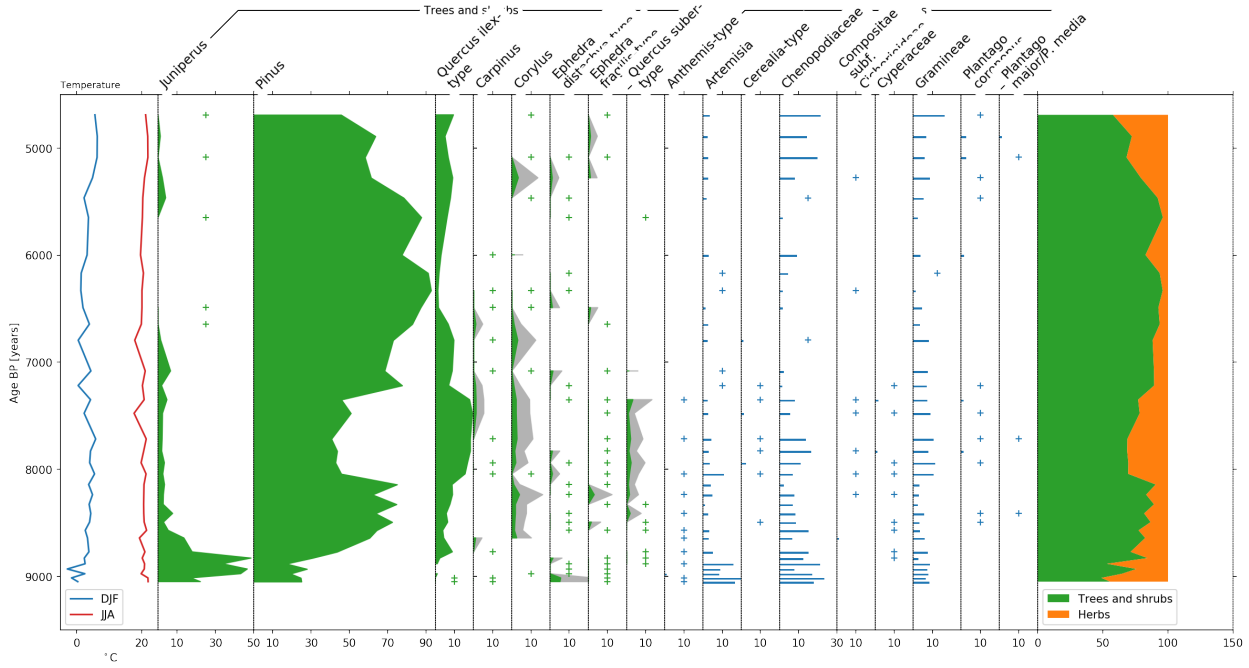
This modifications make the plot look much nicer!

```
sp, groupers = stratplot(
    df, grouper,
    thresh=1.0,
    trunc_height=0.1,
    use_bars=['Herbs'],
    all_in_one=['Temperature'],
    summed=['Trees and shrubs', 'Herbs'],
    widths={'Temperature': 0.1, 'Pollen': 0.9},
    percentages=['Pollen'],
    calculate_percentages=True,
    subgroups={'Pollen': ['Trees and shrubs', 'Herbs']},
    exclude=['Aquatics', 'Dwarf shrubs', 'Helophytes', 'Nonpollen',
             'Vascular cryptogams (Pteridophytes)'])

# apply a new order where we first display Juniperus, Pinus and Quercus, and then the␣
↪rest
```

(continues on next page)

(continued from previous page)

```
pollen_grouper = groupers[1]
first_taxa = ['Juniperus', 'Pinus', 'Quercus ilex-type']
remaining_trees = group2taxon['Trees and shrubs'][
    ~group2taxon['Trees and shrubs'].isin(first_taxa)].dropna().tolist()

neworder = first_taxa + remaining_trees
pollen_grouper.reorder(neworder)


# -- psyplot update
blue = '#1f77b4'
orange = '#ff7f0e'
green = '#2ca02c'
red = '#d62728'

# change the color of trees and shrubs to green
sp(group='Trees and shrubs').update(color=[green])

# exaggerate the trees with low counts by a factor of 4
sp(name=remaining_trees).update(exag='areax', exag_factor=4)

# mark small taxon occurences below 1% with a +
sp(maingroup='Pollen').update(occurences=1.0)

# change the color of JJA temperature to red, shorten legend labels and
# change the x- and y-label
sp(group='Temperature').update(color=[blue, red], legendlabels=['DJF', 'JJA'],
                               ylabel='Age BP [years]', xlabel='$^\circ$C',
                               legend={'loc': 'lower left'})

# change the color of the summed trees and shrubs to green and put the legend
# on the bottom
sp(group='Summed').update(color=[green, orange], legend={'loc': 'lower left'})
```



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

References

- Davis, B.A. and Stevenson, A.C., **2007**. The 8.2 ka event and Early–Mid Holocene forests, fires and flooding in the Central Ebro Desert, NE Spain. *Quaternary Science Reviews*, 26(13-14), pp.1695-1712.

1.3 API Reference

psy-strat: A psyplot plugin for stratigraphic plots

Describe your plugin here and do whatever you want. It should at least have a __version__ attribute to specify the version of the package

1.3.1 Submodules

psy_strat.plotters module

plotters module of the psy-strat psyplot plugin

This module defines the plotters for the psy-strat package.

Formatoption classes

AxesGrouper(key[, plotter, index_in_list, ...])	Group several axes through a bar
AxisLineStyle(key[, plotter, index_in_list, ...])	Set the linestyle the x- and y-axes
ExagFactor(key[, plotter, index_in_list, ...])	The exaggerations factor
ExagPlot(*args, **kwargs)	Choose the line style of the plot
LeftTitle(key[, plotter, index_in_list, ...])	Show the title
MeasurementLines(key[, plotter, ...])	Draw lines at the measurement locations
OccurenceMarker(key[, plotter, ...])	Specify the marker for occurences
OccurencePlot(key[, plotter, index_in_list, ...])	Specify the value to use for occurences in the plot
Occurences(key[, plotter, index_in_list, ...])	Specify the range for occurences
TitleLoc(key[, plotter, index_in_list, ...])	Specify the position of the axes title
TitleWrap(key[, plotter, index_in_list, ...])	Wrap the title automatically

Plotter classes

BarStratPlotter([data, ax, auto_update, ...])	A bar plotter for stratigraphic diagrams
StratPlotter([data, ax, auto_update, ...])	A plotter for stratigraphic diagrams

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

Bases: psy_simple.base.TextBase, psyplot.plotter.Formatoption

Group several axes through a bar

This formatoption groups several plots on the same row by drawing a bar over them

Possible types

Attributes

annotations	Built-in mutable sequence.
dependencies	Built-in mutable sequence.
name	str(object='') -> str
texts	Built-in mutable sequence.
title	title Formatoption instance in the plotter
titleprops	titleprops Formatoption instance in the plotter

Methods

create_annotations()	Annotate from the left to the right axes
create_text(value)	
initialize_plot(value)	Method that is called when the plot is made the first time
onresize(event)	
remove([annotation, text])	Method to remove the effects of this formatoption
set_params(value)	Set the parameters for the annotation and the text
update(value)	Method that is call to update the formatoption on the axes

- *None* – To not do anything
- tuple (float `y`, str `s`) – A tuple of length 2, where the first parameter `0<=y<=1` determines the distance of the bar to the top y-axis and the second is the title of the group. *y* must be given relative to the axes height.

  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.

annotations = []

create_annotations()
Annotate from the left to the right axes

create_text(*value*)

dependencies = ['titleprops', 'title']

initialize_plot(*value*)
Method that is called when the plot is made the first time

Parameters **value** – The value to use for the initialization

name = 'Group the axes'

onresize(*event*)

remove(*annotation=True*, *text=True*)
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.

set_params(*value*)
Set the parameters for the annotation and the text

texts = []

property title
title Formatoption instance in the plotter

property titleprops
titleprops Formatoption instance in the plotter

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

Parameters **value** – Value to update

value2share = None

class psy_strat.plotters.**AxisLineStyle**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)
Bases: `psyplot.plotter.DictFormatoption`

Set the linestyle the x- and y-axes

This formatoption sets the linestyle of the left, right, bottom and top axis.

Possible types

Attributes

group	str(object='') -> str
name	str(object='') -> str
value2pickle	Return the current axis colors

Methods

initialize_plot(value)	Method that is called when the plot is made the first time
update(value)	Method that is call to update the formatoption on the axes

dict – Keys may be one of {'right', 'left', 'bottom', 'top'}, the values can be any valid linestyle or None to use the default style. The line style string can be one of (['solid' | 'dashed', 'dashdot', 'dotted' | (offset, on-off-dash-

seq) | '-' | '–' | '-.' | ':' | 'None' | ' ' | '']).

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.

group = 'axes'

initialize_plot(*value*)

Method that is called when the plot is made the first time

Parameters **value** – The value to use for the initialization

name = 'Linestyle of x- and y-axes'

update(*value*)

Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

property value2pickle

Return the current axis colors

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

Bases: psy_simple.plotters.BarPlotter

A bar plotter for stratigraphic diagrams

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

Axes formatoptions

axislinestyle	Set the linestyle the x- and y-axes
axiscolor	Color the x- and y-axes
grid	Display the grid
tight	Automatically adjust the plots.
transpose	Switch x- and y-axes
xlim	Set the x-axis limits
ylim	Set the y-axis limits

Plot formatoptions

exag	Choose the line style of the plot
plot	Choose how to make the bar plot

Color coding formatoptions

exag_color	Set the color coding
color	Set the color coding

Miscallaneous formatoptions

exag_factor	The exaggerations factor
hlines	Draw lines at the measurement locations
occurence_marker	Specify the marker for occurences
occurence_value	Specify the value to use for occurences in the plot
occurences	Specify the range for occurences
alpha	Specify the transparency (alpha)
categorical	The location of each bar
legend	Draw a legend
legendlabels	Set the labels of the arrays in the legend
sym_lims	Make x- and y-axis symmetric
ticksize	Change the ticksize of the ticklabels
tickweight	Change the fontweight of the ticks
widths	Specify the widths of the bars

Label formatoptions

grouper	Group several axes through a bar
grouperprops	Properties of the grouper
groupersize	Set the size of the grouper
grouperweight	Set the fontweight of the grouper

Continued on next page

Table 11 – continued from previous page

title	Show the title
title_loc	Specify the position of the axes title
title_wrap	Wrap the title automatically
figtitle	Plot a figure title
figtitleprops	Properties of the figure title
figtitlesize	Set the size of the figure title
figtitleweight	Set the fontweight of the figure title
labelprops	Set the font properties of both, x- and y-label
labelsize	Set the size of both, x- and y-label
labelweight	Set the font size of both, x- and y-label
text	Add text anywhere on the plot
titleprops	Properties of the title
titlesize	Set the size of the title
titleweight	Set the fontweight of the title
xlabel	Set the x-axis label
ylabel	Set the y-axis label

Axis tick formatoptions

xrotation	Rotate the x-axis ticks
xticklabels	Modify the x-axis ticklabels
xtickprops	Specify the x-axis tick parameters
xticks	Modify the x-axis ticks
yrotation	Rotate the y-axis ticks
yticklabels	Modify the y-axis ticklabels
ytickprops	Specify the y-axis tick parameters
yticks	Modify the y-axis ticks

Masking formatoptions

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

Post processing formatoptions

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

Data manipulation formatoptions

coord	Use an alternative variable as x-coordinate

`axislinestyle`
Set the linestyle the x- and y-axes

This formatoption sets the linestyle of the left, right, bottom and top axis.

Possible types

dict – Keys may be one of {'right', 'left', 'bottom', 'top'}, the values can be any valid linestyle or None to use the default style. The line style string can be one of (['solid' | 'dashed', 'dashdot', 'dotted' | (offset, on-off-dash-seq) | '-' | '–' | '-.' | ':' | 'None' | ' ' | '']).

exag
Choose the line style of the plot

Possible types

- *None* – Don't make any plotting
- `'area'` – To make an area plot (filled between y=0 and y), see `matplotlib.pyplot.fill_between()`
- `'areax'` – To make a transposed area plot (filled between x=0 and x), see `matplotlib.pyplot.fill_betweenx()`
- `'stacked'` – Make a stacked plot
- *str or list of str* – The line style string to use (['solid' | 'dashed', 'dashdot', 'dotted' | (offset, on-off-dash-seq) | '-' | '–' | '-.' | ':' | 'None' | ' ' | '']).

exag_color
Set the color coding

This formatoptions sets the color of the lines, bars, etc.

Possible types

- *None* – to use the axes color_cycle
- *iterable* – (e.g. list) to specify the colors manually
- *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* – to automatically choose the colors according to the number of lines, etc. from the given colormap

exag_factor
The exaggerations factor

Possible types

float – The factor by how much the data should be exaggerated

See also:

exag_color, *exag*

grouper
Group several axes through a bar

This formatoption groups several plots on the same row by drawing a bar over them

Possible types

- *None* – To not do anything
- tuple (float `y`, str `s`) – A tuple of length 2, where the first parameter `0<=y<=1` determines the distance of the bar to the top y-axis and the second is the title of the group. *y* must be given relative to the axes height.

grouperprops
Properties of the grouper

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

grouper, *groupersize*, *grouperweight*

groupersize
Set the size of the grouper

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:

grouper, *grouperweight*, *grouperprops*

grouperweight
Set the fontweight of the grouper

Possible types

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

See also:

grouper, *groupersize*, *grouperprops*

hlines
Draw lines at the measurement locations

Possible types

- *None* – Don't draw any lines
- *color* – The color of the lines

occurence_marker
Specify the marker for occurences

This formatoption can be used to define the marker style for occurences.

Possible types

- *None* – Use the mean of the axes limits
- *float* – Specify the x-value for an occurence
- *list of floats* – Specify the x-value for an occurence for each array explicitly

See also:

occurences, *occurence_value*

occurence_value
Specify the value to use for occurences in the plot

This formatoption can be used to define where the occurence marker should be placed.

Possible types

- *None* – Use the mean of the axes limits
- *float* – Specify the x-value for an occurence
- *list of floats* – Specify the x-value for an occurence for each array explicitly

See also:

occurences, *occurence_marker*

occurences
Specify the range for occurences

This formatoption can be used to specify a minimum and a maximum value. The parts of the data that fall into this range will be considered as an occurence, set to 0 and marked by the *occurence_value* formatoption

Possible types

- *None* – Do not mark anything as an occurence
- *float* – Anything below the given number will be considered as an occurence
- tuple of floats `(vmin, vmax)` – The minimum and maximum value. Anything between *vmin* and *vmax* will be marked as a occurence

See also:

occurence_marker, *occurence_value*

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:

title_loc The location of the title

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

title_loc
Specify the position of the axes title

Parameters **str** – The position the axes title

center In the center of the axes (the standard way)

left At the left corner

right At the right corner

title_wrap
Wrap the title automatically

This formatoption wraps the title using `textwrap.wrap()`.

Possible types

int – If 0, the title will not be rapped, otherwise it will be wrapped after the given number of characters

Notes

This wraps the title after a certain amount of characters. For wrapping the text automatically before it leaves the plot area, use `titleprops=dict(wrap=True)`

axiscolor
Color the x- and y-axes

This formatoption colors the left, right, bottom and top axis bar.

Possible types

dict – Keys may be one of {'right', 'left', 'bottom', 'top'}, the values can be any valid color or None.

Notes

The following color abbreviations are supported:

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

In addition, you can specify colors in many weird and wonderful ways, including full names (`'green'`), hex strings (`'#008000'`), RGB or RGBA tuples (`(0,1,0,1)`) or grayscale intensities as a string (`'0.8'`).

grid
Display the grid

Show the grid on the plot with the specified color.

Possible types

- *None* – If the grid is currently shown, it will not be displayed any longer. If the grid is not shown, it will be drawn
- *bool* – If True, the grid is displayed with the automatic settings (usually black)
- *string, tuple.* – Defines the color of the grid.

Notes

The following color abbreviations are supported:

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

In addition, you can specify colors in many weird and wonderful ways, including full names (`'green'`), hex strings (`'#008000'`), RGB or RGBA tuples (`(0,1,0,1)`) or grayscale intensities as a string (`'0.8'`).

tight
Automatically adjust the plots.

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

Possible types

bool – True for automatic adjustment

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

transpose
Switch x- and y-axes

By default, one-dimensional arrays have the dimension on the x-axis and two dimensional arrays have the first dimension on the y and the second on the x-axis. You can set this formatoption to True to change this behaviour

Possible types

bool – If True, axes are switched

xlim
Set the x-axis limits

Possible types

- *None* – To not change the current limits
- *str or list [str, str] or [[str, float], [str, float]]* – Automatically determine the ticks corresponding to the data. The given string determines how the limits are calculated. The float determines the percentile to use A string can be one of the following:

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

roundedsym Same as *rounded* above but the limits are chosen such that they are symmetric around zero

minmax Uses the minimum and maximum

sym Same as minmax but symmetric around zero

- *tuple (xmin, xmax)* – *xmin* is the smaller value, *xmax* the larger. Any of those values can be None or one of the strings (or lists) above to use the corresponding value here

See also:

ylim

ylim
Set the y-axis limits

Possible types

- *None* – To not change the current limits
- *str or list [str, str] or [[str, float], [str, float]]* – Automatically determine the ticks corresponding to the data. The given string determines how the limits are calculated. The float determines the percentile to use A string can be one of the following:

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

  roundedsym Same as *rounded* above but the limits are chosen such that they are symmetric around zero

  minmax Uses the minimum and maximum

  sym Same as minmax but symmetric around zero

- *tuple (xmin, xmax)* – *xmin* is the smaller value, *xmax* the larger. Any of those values can be None or one of the strings (or lists) above to use the corresponding value here

See also:

xlim

plot
Choose how to make the bar plot

Possible types

- *None* – Don't make any plotting
- *'bar'* – Create a usual bar plot with the bars side-by-side
- *'stacked'* – Create stacked plot

color
Set the color coding

This formatoptions sets the color of the lines, bars, etc.

Possible types

- *None* – to use the axes color_cycle
- *iterable* – (e.g. list) to specify the colors manually
- *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* – to automatically choose the colors according to the number of lines, etc. from the given colormap

alpha
Specify the transparency (alpha)

Possible types

float – A value between 0 (opaque) and 1 invisible

categorical
The location of each bar

Possible types

- *None* – If None, use a categorical plotting if the widths are `'equal'`, otherwise, not
- *bool* – If True, use a categorical plotting

See also:

widths

legend
Draw a legend

This formatoption determines where and if to draw the legend. It uses the `labels` formatoption to determine the labels.

Possible types

- *bool* – Draw a legend or not
- *str or int* – Specifies where to plot the legend (i.e. the location)
- *dict* – Give the keywords for the `matplotlib.pyplot.legend()` function

See also:

`labels`

legendlabels
Set the labels of the arrays in the legend

This formatoption specifies the labels for each array in the legend. 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* – A single string that shall be used for all arrays.
- *list of str* – Same as a single string but specified for each array

See also:

legend

sym_lims
Make x- and y-axis symmetric

Possible types

- *None* – No symmetric type
- *'min'* – Use the minimum of x- and y-limits
- *'max'* – Use the maximum of x- and y-limits
- *[str, str]* – A combination, `None`, `'min'` and `'max'` specific for minimum and maximum limit

ticksize
Change the ticksize of the ticklabels

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *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:

tickweight, *xtickprops*, *ytickprops*

tickweight
Change the fontweight of the ticks

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

ticksize, *xtickprops*, *ytickprops*

widths
Specify the widths of the bars

Possible types

- *'equal'* – Each bar will have the same width (the default)
- *'data'* – Each bar will have the width as specified by the boundaries
- *float* – The width for each bar

See also:

categorical

figtitle
Plot a figure title

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

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

Possible types

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

Notes

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

See also:

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

figtitleprops
Properties of the figure title

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

figtitle, *figtitlesize*, *figtitleweight*

figtitlesize
Set the size of the figure title

Possible types

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

See also:

figtitle, *figtitleweight*, *figtitleprops*

figtitleweight
Set the fontweight of the figure title

Possible types

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

See also:

figtitle, *figtitlesize*, *figtitleprops*

labelprops
Set the font properties of both, x- and y-label

Possible types

- *dict* – A dictionary with the keys `'x'` and (or) `'y'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is used for the x- and y-axis. The values in the dictionary can be one types below.
- *dict* – Items may be any valid text property

See also:

xlabel, *ylabel*, *labelsize*, *labelweight*

labelsize
Set the size of both, x- and y-label

Possible types

- *dict* – A dictionary with the keys `'x'` and (or) `'y'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is used for the x- and y-axis. The values in the dictionary can be one types below.
- *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:

xlabel, *ylabel*, *labelweight*, *labelprops*

labelweight
Set the font size of both, x- and y-label

Possible types

- *dict* – A dictionary with the keys `'x'` and (or) `'y'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is used for the x- and y-axis. The values in the dictionary can be one types below.
- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

xlabel, *ylabel*, *labelsize*, *labelprops*

text
Add text anywhere on the plot

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

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

Possible types

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

See also:

title, *figtitle*

titleprops
Properties of the title

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

title, *titlesize*, *titleweight*

titlesize
Set the size of the title

Possible types

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

See also:

title, *titleweight*, *titleprops*

titleweight
Set the fontweight of the title

Possible types

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

See also:

title, *titlesize*, *titleprops*

xlabel
Set the x-axis label

Set the label for the x-axis. 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 text for the `xlabel()` function.

See also:

`xlabelsize`, `xlabelweight`, `xlabelprops`

ylabel
Set the y-axis label

Set the label for the y-axis. 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 text for the `ylabel()` function.

See also:

`ylabelsize`, `ylabelweight`, `ylabelprops`

xrotation
Rotate the x-axis ticks

Possible types

float – The rotation angle in degrees

See also:

yrotation

xticklabels
Modify the x-axis ticklabels

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *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:

xticks, *ticksize*, *tickweight*, *xtickprops*, *yticklabels*

xtickprops
Specify the x-axis tick parameters

This formatoption can be used to make a detailed change of the ticks parameters on the x-axis.

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *dict* – Items may be anything of the `matplotlib.pyplot.tick_params()` function

See also:

xticks, *yticks*, *ticksize*, *tickweight*, *ytickprops*

xticks
Modify the x-axis ticks

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *None* – use the default ticks
- *int* – for an integer *i*, only every *i-th* tick of the default ticks are used
- *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

  hour draw ticks every hour

  day draw ticks every day

week draw ticks every week

month, monthend, monthbegin draw ticks in the middle, at the end or at the beginning of each month

year, yearend, yearbegin draw ticks in the middle, at the end or at the beginning of each year

For data, mid, hour, day, week, month, etc., the optional second value can be an integer i determining that every i-th data point shall be used (by default, it is set to 1). For rounded, roundedsym, minmax and sym, the second value determines the total number of ticks (defaults to 11).

Examples

Plot 11 ticks over the whole data range:

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

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

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

Plot ticks every year and minor ticks every month:

```
>>> plotter.update(xticks={'major': 'year', 'minor': 'month'})
```

See also:

xticklabels, *ticksize*, *tickweight*, *xtickprops*, *yticks*

yrotation
Rotate the y-axis ticks

Possible types

float – The rotation angle in degrees

See also:

xrotation

yticklabels
Modify the y-axis ticklabels

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *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:

yticks, *ticksize*, *tickweight*, *ytickprops*, *xticklabels*

ytickprops
Specify the y-axis tick parameters

This formatoption can be used to make a detailed change of the ticks parameters of the y-axis.

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *dict* – Items may be anything of the `matplotlib.pyplot.tick_params()` function

See also:

xticks, *yticks*, *ticksize*, *tickweight*, *xtickprops*

yticks
Modify the y-axis ticks

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *None* – use the default ticks
- *int* – for an integer *i*, only every *i-th* tick of the default ticks are used
- *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

  hour draw ticks every hour

  day draw ticks every day

week draw ticks every week

month, monthend, monthbegin draw ticks in the middle, at the end or at the beginning of each month

year, yearend, yearbegin draw ticks in the middle, at the end or at the beginning of each year

For data, mid, hour, day, week, month, etc., the optional second value can be an integer i determining that every i-th data point shall be used (by default, it is set to 1). For rounded, roundedsym, minmax and sym, the second value determines the total number of ticks (defaults to 11).

See also:

yticklabels, *ticksize*, *tickweight*, *ytickprops*

xticks for possible examples

maskbetween
Mask data points between two numbers

Possible types

float – The floating number to mask above

See also:

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

maskgeq
Mask data points greater than or equal to a number

Possible types

float – The floating number to mask above

See also:

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

maskgreater
Mask data points greater than a number

Possible types

float – The floating number to mask above

See also:

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

maskleq
Mask data points smaller than or equal to a number

Possible types

float – The floating number to mask below

See also:

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

maskless
Mask data points smaller than a number

Possible types

float – The floating number to mask below

See also:

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

post
Apply your own postprocessing script

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

Possible types

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

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

Examples

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

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

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

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

See also:

post_timing Determine the timing of this formatoption

post_timing
Determine when to run the *post* formatoption

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

Possible types

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

See also:

post The post processing formatoption

coord
Use an alternative variable as x-coordinate

This formatoption let's you specify another variable in the base dataset of the data array in case you want to use this as the x-coordinate instead of the raw data

Possible types

- *None* – Use the default
- *str* – The name of the variable to use in the base dataset
- *xarray.DataArray* – An alternative variable with the same shape as the displayed array

---

Examples

To see the difference, we create a simple test dataset:

```
>>> import xarray as xr

>>> import numpy as np

>>> import psyplot.project as psy

>>> ds = xr.Dataset({
...     'temp': xr.Variable(('time', ), np.arange(5)),
...     'std': xr.Variable(('time', ), np.arange(5, 10))})
>>> ds
<xarray.Dataset>
Dimensions:  (time: 5)
Coordinates:
  * time     (time) int64 0 1 2 3 4
Data variables:
    temp     (time) int64 0 1 2 3 4
    std      (time) int64 5 6 7 8 9
```

If we create a plot with it, we get the `'time'` dimension on the x-axis:

```
>>> plotter = psy.plot.lineplot(ds, name=['temp']).plotters[0]

>>> plotter.plot_data[0].dims
('time',)
```

If we however set the `'coord'` keyword, we get:

```
>>> plotter = psy.plot.lineplot(
...     ds, name=['temp'], coord='std').plotters[0]

>>> plotter.plot_data[0].dims
('std',)
```

and `'std'` is plotted on the x-axis.

---

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

Bases: psyplot.plotter.Formatoption

The exaggerations factor

Possible types

Attributes

name	str(object='') -> str
priority	int([x]) -> integer

Methods

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

float – The factor by how much the data should be exaggerated

See also:

exag_color, exag

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.

name = 'Exaggeration factor'

priority = 20

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

Parameters **value** – Value to update

class psy_strat.plotters.**ExagPlot**(**args*, ***kwargs*)
Bases: psy_simple.plotters.LinePlot

Choose the line style of the plot

Possible types

Attributes

color	color Formatoption instance in the plotter
data_dependent	bool(x) -> bool
dependencies	Built-in mutable sequence.
exag_factor	exag_factor Formatoption instance in the plotter
marker	marker Formatoption instance in the plotter
transpose	transpose Formatoption instance in the plotter

Methods

plot_arr(arr, *args, **kwargs)

- *None* – Don't make any plotting
- `'area'` – To make an area plot (filled between y=0 and y), see `matplotlib.pyplot.fill_between()`
- `'areax'` – To make a transposed area plot (filled between x=0 and x), see `matplotlib.pyplot.fill_betweenx()`
- `'stacked'` – Make a stacked plot
- *str or list of str* – The line style string to use (['solid' | 'dashed', 'dashdot', 'dotted' | (offset, on-off-dash-seq) | '-' | '–' | '-.' | ':' | 'None' | ' ' | '']).

property color
color Formatoption instance in the plotter

data_dependent = True

dependencies = ['exag_factor']

property exag_factor
exag_factor Formatoption instance in the plotter

property marker
marker Formatoption instance in the plotter

plot_arr(*arr*, **args*, ***kwargs*)

property transpose
transpose Formatoption instance in the plotter

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

Bases: psy_simple.base.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]`

Attributes

dependencies	Built-in mutable sequence.
title_loc	title_loc Formatoption instance in the plotter

Methods

initialize_plot(value)	Method that is called when the plot is made the first time
update(value)	Method that is call to update the formatoption on the axes

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:

title_loc The location of the title

figtitle, titlesize, titleweight, titleprops

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 = ['title_loc']

initialize_plot(*value*)
Method that is called when the plot is made the first time

Parameters **value** – The value to use for the initialization

property title_loc
title_loc 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_strat.plotters.**MeasurementLines**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: psyplot.plotter.Formatoption

Draw lines at the measurement locations

Possible types

Attributes

dependencies	Built-in mutable sequence.
plot	plot Formatoption instance in the plotter
transpose	transpose Formatoption instance in the plotter
xlim	xlim Formatoption instance in the plotter

Methods

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

- *None* – Don't draw any lines
- *color* – The color of the lines

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.

artists = None

default = None

dependencies = ['transpose', 'xlim', 'plot']

property plot
plot 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.

property transpose
transpose Formatoption instance in the plotter

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

Parameters **value** – Value to update

property xlim
xlim Formatoption instance in the plotter

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

Bases: psyplot.plotter.Formatoption

Specify the marker for occurences

This formatoption can be used to define the marker style for occurences.

Possible types

Attributes

name	str(object='') -> str

Methods

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

- *None* – Use the mean of the axes limits
- *float* – Specify the x-value for an occurence
- *list of floats* – Specify the x-value for an occurence for each array explicitly

See also:

`occurences`, `occurence_value`

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.

name = 'Marker for the occurences'

update(*value*)

Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

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

Bases: psyplot.plotter.Formatoption

Specify the value to use for occurences in the plot

This formatoption can be used to define where the occurence marker should be placed.

Possible types

Attributes

color	color Formatoption instance in the plotter
dependencies	Built-in mutable sequence.
name	str(object='') -> str
occurence_marker	occurence_marker Formatoption instance in the plotter
occurences	occurences Formatoption instance in the plotter
transpose	transpose Formatoption instance in the plotter
xlim	xlim Formatoption instance in the plotter

Methods

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

- *None* – Use the mean of the axes limits
- *float* – Specify the x-value for an occurence
- *list of floats* – Specify the x-value for an occurence for each array explicitly

See also:

occurences, *occurence_marker*

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.

property color
color Formatoption instance in the plotter

dependencies = ['occurences', 'xlim', 'occurence_marker', 'color', 'transpose']

name = 'Occurence plot value'

property occurence_marker
occurence_marker Formatoption instance in the plotter

property occurences
occurences 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.

property transpose
transpose Formatoption instance in the plotter

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

Parameters **value** – Value to update

property xlim
xlim Formatoption instance in the plotter

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

Bases: `psyplot.plotter.Formatoption`

Specify the range for occurences

This formatoption can be used to specify a minimum and a maximum value. The parts of the data that fall into this range will be considered as an occurence, set to 0 and marked by the `occurence_value` formatoption

Possible types

Attributes

children	Built-in mutable sequence.
maskgeq	maskgeq Formatoption instance in the plotter
maskgreater	maskgreater Formatoption instance in the plotter
maskleq	maskleq Formatoption instance in the plotter
maskless	maskless Formatoption instance in the plotter
name	str(object='') -> str
priority	int([x]) -> integer

Methods

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

- *None* – Do not mark anything as an occurence
- *float* – Anything below the given number will be considered as an occurence
- tuple of floats `(vmin, vmax)` – The minimum and maximum value. Anything between *vmin* and *vmax* will be marked as a occurence

See also:

`occurence_marker`, `occurence_value`

Parameters

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

children = ['maskless', 'maskleq', 'maskgreater', 'maskgeq']

property maskgeq
maskgeq Formatoption instance in the plotter

property maskgreater
maskgreater Formatoption instance in the plotter

property maskleq
maskleq Formatoption instance in the plotter

property maskless
maskless Formatoption instance in the plotter

name = 'Occurences range'

priority = 30

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

Parameters **value** – Value to update

class psy_strat.plotters.**StratPlotter**(*data=None*, *ax=None*, *auto_update=None*, *project=None*, *draw=False*, *make_plot=True*, *clear=False*, *enable_post=False*, ***kwargs*)
Bases: `psy_simple.plotters.LinePlotter`

A plotter for stratigraphic diagrams

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

Axes formatoptions

axislinestyle	Set the linestyle the x- and y-axes
axiscolor	Color the x- and y-axes
grid	Display the grid
tight	Automatically adjust the plots.
transpose	Switch x- and y-axes
xlim	Set the x-axis limits
ylim	Set the y-axis limits

Plot formatoptions

exag	Choose the line style of the plot
error	Visualize the error range
plot	Choose the line style of the plot

Color coding formatoptions

exag_color	Set the color coding
color	Set the color coding
erroralpha	Set the alpha value for the error range

Miscallaneous formatoptions

exag_factor	The exaggerations factor
hlines	Draw lines at the measurement locations
occurence_marker	Specify the marker for occurences
occurence_value	Specify the value to use for occurences in the plot
occurences	Specify the range for occurences
legend	Draw a legend
legendlabels	Set the labels of the arrays in the legend
linewidth	Choose the width of the lines
marker	Choose the marker for points
markersize	Choose the size of the markers for points

Continued on next page

Table 33 – continued from previous page

sym_lims	Make x- and y-axis symmetric
ticksize	Change the ticksize of the ticklabels
tickweight	Change the fontweight of the ticks

Label formatoptions

grouper	Group several axes through a bar
grouperprops	Properties of the grouper
groupersize	Set the size of the grouper
grouperweight	Set the fontweight of the grouper
title	Show the title
title_loc	Specify the position of the axes title
title_wrap	Wrap the title automatically
figtitle	Plot a figure title
figtitleprops	Properties of the figure title
figtitlesize	Set the size of the figure title
figtitleweight	Set the fontweight of the figure title
labelprops	Set the font properties of both, x- and y-label
labelsize	Set the size of both, x- and y-label
labelweight	Set the font size of both, x- and y-label
text	Add text anywhere on the plot
titleprops	Properties of the title
titlesize	Set the size of the title
titleweight	Set the fontweight of the title
xlabel	Set the x-axis label
ylabel	Set the y-axis label

Axis tick formatoptions

xrotation	Rotate the x-axis ticks
xticklabels	Modify the x-axis ticklabels
xtickprops	Specify the x-axis tick parameters
xticks	Modify the x-axis ticks
yrotation	Rotate the y-axis ticks
yticklabels	Modify the y-axis ticklabels
ytickprops	Specify the y-axis tick parameters
yticks	Modify the y-axis ticks

Masking formatoptions

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

Post processing formatoptions

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

Data manipulation formatoptions

coord	Use an alternative variable as x-coordinate

axislinestyle
Set the linestyle the x- and y-axes

This formatoption sets the linestyle of the left, right, bottom and top axis.

Possible types

dict – Keys may be one of {'right', 'left', 'bottom', 'top'}, the values can be any valid linestyle or None to use the default style. The line style string can be one of (['solid' | 'dashed', 'dashdot', 'dotted' | (offset, on-off-dash-seq) | '-' | '–' | '-.' | ':' | 'None' | ' ' | '']).

exag
Choose the line style of the plot

Possible types

- *None* – Don't make any plotting
- `'area'` – To make an area plot (filled between y=0 and y), see `matplotlib.pyplot.fill_between()`
- `'areax'` – To make a transposed area plot (filled between x=0 and x), see `matplotlib.pyplot.fill_betweenx()`
- `'stacked'` – Make a stacked plot
- *str or list of str* – The line style string to use (['solid' | 'dashed', 'dashdot', 'dotted' | (offset, on-off-dash-seq) | '-' | '–' | '-.' | ':' | 'None' | ' ' | '']).

exag_color
Set the color coding

This formatoptions sets the color of the lines, bars, etc.

Possible types

- *None* – to use the axes color_cycle
- *iterable* – (e.g. list) to specify the colors manually
- *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* – to automatically choose the colors according to the number of lines, etc. from the given colormap

exag_factor
The exaggerations factor

Possible types

float – The factor by how much the data should be exaggerated

See also:

exag_color, *exag*

grouper
Group several axes through a bar

This formatoption groups several plots on the same row by drawing a bar over them

Possible types

- *None* – To not do anything
- tuple (float `y`, str `s`) – A tuple of length 2, where the first parameter `0<=y<=1` determines the distance of the bar to the top y-axis and the second is the title of the group. *y* must be given relative to the axes height.

grouperprops
Properties of the grouper

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

grouper, *groupersize*, *grouperweight*

groupersize
Set the size of the grouper

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:

grouper, *grouperweight*, *grouperprops*

grouperweight
Set the fontweight of the grouper

Possible types

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

See also:

grouper, *groupersize*, *grouperprops*

hlines
Draw lines at the measurement locations

Possible types

- *None* – Don’t draw any lines
- *color* – The color of the lines

occurence_marker
Specify the marker for occurences

This formatoption can be used to define the marker style for occurences.

Possible types

- *None* – Use the mean of the axes limits
- *float* – Specify the x-value for an occurence
- *list of floats* – Specify the x-value for an occurence for each array explicitly

See also:

occurences, *occurence_value*

occurence_value
Specify the value to use for occurences in the plot

This formatoption can be used to define where the occurence marker should be placed.

Possible types

- *None* – Use the mean of the axes limits
- *float* – Specify the x-value for an occurence
- *list of floats* – Specify the x-value for an occurence for each array explicitly

See also:

occurences, *occurence_marker*

occurences
Specify the range for occurences

This formatoption can be used to specify a minimum and a maximum value. The parts of the data that fall into this range will be considered as an occurence, set to 0 and marked by the *occurence_value* formatoption

Possible types

- *None* – Do not mark anything as an occurence
- *float* – Anything below the given number will be considered as an occurence
- tuple of floats `(vmin, vmax)` – The minimum and maximum value. Anything between *vmin* and *vmax* will be marked as a occurence

See also:

occurence_marker, *occurence_value*

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:

title_loc The location of the title

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

title_loc
Specify the position of the axes title

Parameters **str** – The position the axes title

center In the center of the axes (the standard way)

left At the left corner

right At the right corner

title_wrap
Wrap the title automatically

This formatoption wraps the title using `textwrap.wrap()`.

Possible types

int – If 0, the title will not be rapped, otherwise it will be wrapped after the given number of characters

Notes

This wraps the title after a certain amount of characters. For wrapping the text automatically before it leaves the plot area, use `titleprops=dict(wrap=True)`

axiscolor
Color the x- and y-axes

This formatoption colors the left, right, bottom and top axis bar.

Possible types

dict – Keys may be one of {'right', 'left', 'bottom', 'top'}, the values can be any valid color or None.

Notes

The following color abbreviations are supported:

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

In addition, you can specify colors in many weird and wonderful ways, including full names (`'green'`), hex strings (`'#008000'`), RGB or RGBA tuples (`(0,1,0,1)`) or grayscale intensities as a string (`'0.8'`).

grid
Display the grid

Show the grid on the plot with the specified color.

Possible types

- *None* – If the grid is currently shown, it will not be displayed any longer. If the grid is not shown, it will be drawn
- *bool* – If True, the grid is displayed with the automatic settings (usually black)
- *string, tuple.* – Defines the color of the grid.

Notes

The following color abbreviations are supported:

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

In addition, you can specify colors in many weird and wonderful ways, including full names (`'green'`), hex strings (`'#008000'`), RGB or RGBA tuples (`(0,1,0,1)`) or grayscale intensities as a string (`'0.8'`).

tight

Automatically adjust the plots.

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

Possible types

bool – True for automatic adjustment

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

transpose

Switch x- and y-axes

By default, one-dimensional arrays have the dimension on the x-axis and two dimensional arrays have the first dimension on the y and the second on the x-axis. You can set this formatoption to True to change this behaviour

Possible types

bool – If True, axes are switched

xlim

Set the x-axis limits

Possible types

- *None* – To not change the current limits
- *str or list [str, str] or [[str, float], [str, float]]* – Automatically determine the ticks corresponding to the data. The given string determines how the limits are calculated. The float determines the percentile to use A string can be one of the following:

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

  roundedsym Same as *rounded* above but the limits are chosen such that they are symmetric around zero

  minmax Uses the minimum and maximum

  sym Same as minmax but symmetric around zero
- *tuple (xmin, xmax)* – *xmin* is the smaller value, *xmax* the larger. Any of those values can be None or one of the strings (or lists) above to use the corresponding value here

See also:

ylim

ylim
Set the y-axis limits

Possible types

- *None* – To not change the current limits
- *str or list [str, str] or [[str, float], [str, float]]* – Automatically determine the ticks corresponding to the data. The given string determines how the limits are calculated. The float determines the percentile to use A string can be one of the following:

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

  roundedsym Same as *rounded* above but the limits are chosen such that they are symmetric around zero

  minmax Uses the minimum and maximum

  sym Same as minmax but symmetric around zero
- *tuple (xmin, xmax)* – *xmin* is the smaller value, *xmax* the larger. Any of those values can be None or one of the strings (or lists) above to use the corresponding value here

See also:

xlim

error
Visualize the error range

This formatoption visualizes the error range. For this, you must provide a two-dimensional data array as input. The first dimension might be either of length

- 2 to provide the deviation from minimum and maximum error range from the data
- 3 to provide the minimum and maximum error range explicitly

Possible types

- *None* – No errors are visualized
- *'fill'* – The area between min- and max-error is filled with the same color as the line and the alpha is determined by the `fillalpha` attribute

---

Examples

Assume you have the standard deviation stored in the `'std'`-variable and the data in the `'data'` variable. Then you can visualize the standard deviation simply via:

```
>>> psy.plot.lineplot(input_ds, name=[['data', 'std']])
```

On the other hand, assume you want to visualize the area between the 25th and 75th percentile (stored in the variables `'p25'` and `'p75'`):

```
>>> psy.plot.lineplot(input_ds, name=[['data', 'p25', 'p75']])
```

See also:

erroralpha

plot
Choose the line style of the plot

Possible types

- *None* – Don't make any plotting
- `'area'` – To make an area plot (filled between y=0 and y), see `matplotlib.pyplot.fill_between()`
- `'areax'` – To make a transposed area plot (filled between x=0 and x), see `matplotlib.pyplot.fill_betweenx()`
- `'stacked'` – Make a stacked plot
- *str or list of str* – The line style string to use ([‘solid’ | ‘dashed’, ‘dashdot’, ‘dotted’ | (offset, on-off-dash-seq) | ‘-‘ | ‘–’ | ‘-.’ | ‘:’ | ‘None’ | ‘ ‘ | ‘’]).

color
Set the color coding

This formatoptions sets the color of the lines, bars, etc.

Possible types

- *None* – to use the axes color_cycle
- *iterable* – (e.g. list) to specify the colors manually

- *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* – to automatically choose the colors according to the number of lines, etc. from the given colormap

erroralpha
Set the alpha value for the error range

This formatoption can be used to set the alpha value (opacity) for the *error* formatoption

Possible types

float – A float between 0 and 1

See also:

error

legend
Draw a legend

This formatoption determines where and if to draw the legend. It uses the `labels` formatoption to determine the labels.

Possible types

- *bool* – Draw a legend or not
- *str or int* – Specifies where to plot the legend (i.e. the location)
- *dict* – Give the keywords for the `matplotlib.pyplot.legend()` function

See also:

`labels`

legendlabels
Set the labels of the arrays in the legend

This formatoption specifies the labels for each array in the legend. 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* – A single string that shall be used for all arrays.
- *list of str* – Same as a single string but specified for each array

See also:

legend

linewidth
Choose the width of the lines

Possible types

- *None* – Use the default from matplotlibs rcParams
- *float* – The width of the lines

marker
Choose the marker for points

Possible types

- *None* – Use the default from matplotlibs rcParams
- *str* – A valid symbol for the matplotlib markers (see `matplotlib.markers`)

markersize
Choose the size of the markers for points

Possible types

- *None* – Use the default from matplotlibs rcParams
- *float* – The size of the marker

sym_lims
Make x- and y-axis symmetric

Possible types

- *None* – No symmetric type
- *'min'* – Use the minimum of x- and y-limits
- *'max'* – Use the maximum of x- and y-limits
- *[str, str]* – A combination, `None`, `'min'` and `'max'` specific for minimum and maximum limit

ticksize
Change the ticksize of the ticklabels

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *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:

tickweight, *xtickprops*, *ytickprops*

tickweight
Change the fontweight of the ticks

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

ticksize, *xtickprops*, *ytickprops*

figtitle
Plot a figure title

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

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

Possible types

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

Notes

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

See also:

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

figtitleprops
Properties of the figure title

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

figtitle, *figtitlesize*, *figtitleweight*

figtitlesize
Set the size of the figure title

Possible types

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

See also:

figtitle, *figtitleweight*, *figtitleprops*

figtitleweight
Set the fontweight of the figure title

Possible types

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

See also:

figtitle, *figtitlesize*, *figtitleprops*

labelprops
Set the font properties of both, x- and y-label

Possible types

- *dict* – A dictionary with the keys `'x'` and (or) `'y'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is used for the x- and y-axis. The values in the dictionary can be one types below.
- *dict* – Items may be any valid text property

See also:

xlabel, *ylabel*, *labelsize*, *labelweight*

labelsize
Set the size of both, x- and y-label

Possible types

- *dict* – A dictionary with the keys `'x'` and (or) `'y'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is used for the x- and y-axis. The values in the dictionary can be one types below.
- *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:

xlabel, *ylabel*, *labelweight*, *labelprops*

labelweight
Set the font size of both, x- and y-label

Possible types

- *dict* – A dictionary with the keys `'x'` and (or) `'y'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is used for the x- and y-axis. The values in the dictionary can be one types below.
- *float* – a float between 0 and 1000
- *string* – Possible strings are one of 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'.

See also:

xlabel, *ylabel*, *labelsize*, *labelprops*

text
Add text anywhere on the plot

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

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

Possible types

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

See also:

title, *figtitle*

titleprops
Properties of the title

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

title, *titlesize*, *titleweight*

titlesize
Set the size of the title

Possible types

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

See also:

title, *titleweight*, *titleprops*

titleweight
Set the fontweight of the title

Possible types

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

See also:

title, *titlesize*, *titleprops*

xlabel
Set the x-axis label

Set the label for the x-axis. 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 text for the `xlabel()` function.

See also:

`xlabelsize`, `xlabelweight`, `xlabelprops`

ylabel
Set the y-axis label

Set the label for the y-axis. 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 text for the `ylabel()` function.

See also:

`ylabelsize`, `ylabelweight`, `ylabelprops`

xrotation
Rotate the x-axis ticks

Possible types

float – The rotation angle in degrees

See also:

yrotation

xticklabels
Modify the x-axis ticklabels

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *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:

xticks, *ticksize*, *tickweight*, *xtickprops*, *yticklabels*

xtickprops
Specify the x-axis tick parameters

This formatoption can be used to make a detailed change of the ticks parameters on the x-axis.

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *dict* – Items may be anything of the `matplotlib.pyplot.tick_params()` function

See also:

xticks, *yticks*, *ticksize*, *tickweight*, *ytickprops*

xticks
Modify the x-axis ticks

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *None* – use the default ticks
- *int* – for an integer *i*, only every *i-th* tick of the default ticks are used
- *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

  hour draw ticks every hour

  day draw ticks every day

week draw ticks every week

month, monthend, monthbegin draw ticks in the middle, at the end or at the beginning of each month

year, yearend, yearbegin draw ticks in the middle, at the end or at the beginning of each year

For data, mid, hour, day, week, month, etc., the optional second value can be an integer i determining that every i-th data point shall be used (by default, it is set to 1). For rounded, roundedsym, minmax and sym, the second value determines the total number of ticks (defaults to 11).

Examples

Plot 11 ticks over the whole data range:

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

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

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

Plot ticks every year and minor ticks every month:

```
>>> plotter.update(xticks={'major': 'year', 'minor': 'month'})
```

See also:

xticklabels, *ticksize*, *tickweight*, *xtickprops*, *yticks*

yrotation
Rotate the y-axis ticks

Possible types

float – The rotation angle in degrees

See also:

xrotation

yticklabels
Modify the y-axis ticklabels

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *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:

yticks, *ticksize*, *tickweight*, *ytickprops*, *xticklabels*

ytickprops
Specify the y-axis tick parameters

This formatoption can be used to make a detailed change of the ticks parameters of the y-axis.

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *dict* – Items may be anything of the `matplotlib.pyplot.tick_params()` function

See also:

xticks, *yticks*, *ticksize*, *tickweight*, *xtickprops*

yticks
Modify the y-axis ticks

Possible types

- *dict* – A dictionary with the keys `'minor'` and (or) `'major'` to specify which ticks are managed. If the given value is not a dictionary with those keys, it is put into a dictionary with the key determined by the rcParams `'ticks.which'` key (usually `'major'`). The values in the dictionary can be one types below.
- *None* – use the default ticks
- *int* – for an integer *i*, only every *i-th* tick of the default ticks are used
- *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

  hour draw ticks every hour

  day draw ticks every day

week draw ticks every week

month, monthend, monthbegin draw ticks in the middle, at the end or at the beginning of each month

year, yearend, yearbegin draw ticks in the middle, at the end or at the beginning of each year

For data, mid, hour, day, week, month, etc., the optional second value can be an integer i determining that every i-th data point shall be used (by default, it is set to 1). For rounded, roundedsym, minmax and sym, the second value determines the total number of ticks (defaults to 11).

See also:

yticklabels, *ticksize*, *tickweight*, *ytickprops*

xticks for possible examples

maskbetween
Mask data points between two numbers

Possible types

float – The floating number to mask above

See also:

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

maskgeq
Mask data points greater than or equal to a number

Possible types

float – The floating number to mask above

See also:

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

maskgreater
Mask data points greater than a number

Possible types

float – The floating number to mask above

See also:

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

maskleq
Mask data points smaller than or equal to a number

Possible types

float – The floating number to mask below

See also:

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

maskless
Mask data points smaller than a number

Possible types

float – The floating number to mask below

See also:

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

post
Apply your own postprocessing script

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

Possible types

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

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

Examples

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

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

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

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

See also:

post_timing Determine the timing of this formatoption

post_timing
Determine when to run the *post* formatoption

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

Possible types

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

See also:

post The post processing formatoption

coord
Use an alternative variable as x-coordinate

This formatoption let's you specify another variable in the base dataset of the data array in case you want to use this as the x-coordinate instead of the raw data

Possible types

- *None* – Use the default
- *str* – The name of the variable to use in the base dataset
- *xarray.DataArray* – An alternative variable with the same shape as the displayed array

---

Examples

To see the difference, we create a simple test dataset:

```
>>> import xarray as xr

>>> import numpy as np

>>> import psyplot.project as psy

>>> ds = xr.Dataset({
...     'temp': xr.Variable(('time', ), np.arange(5)),
...     'std': xr.Variable(('time', ), np.arange(5, 10))})
>>> ds
<xarray.Dataset>
Dimensions:  (time: 5)
Coordinates:
  * time     (time) int64 0 1 2 3 4
Data variables:
    temp     (time) int64 0 1 2 3 4
    std      (time) int64 5 6 7 8 9
```

If we create a plot with it, we get the `'time'` dimension on the x-axis:

```
>>> plotter = psy.plot.lineplot(ds, name=['temp']).plotters[0]

>>> plotter.plot_data[0].dims
('time',)
```

If we however set the `'coord'` keyword, we get:

```
>>> plotter = psy.plot.lineplot(
...     ds, name=['temp'], coord='std').plotters[0]

>>> plotter.plot_data[0].dims
('std',)
```

and `'std'` is plotted on the x-axis.

---

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

Bases: psyplot.plotter.Formatoption

Specify the position of the axes title

Parameters

- **str** – The position the axes title

  center In the center of the axes (the standard way)

  left At the left corner

  right At the right corner

- **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

group	str(object='') -> str
name	str(object='') -> str

Methods

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

group = 'labels'

name = 'Location of the axes title'

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

Parameters **value** – Value to update

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

Wrap the title automatically

This formatoption wraps the title using `textwrap.wrap()`.

Possible types

Attributes

dependencies	Built-in mutable sequence.
group	str(object='') -> str
name	str(object='') -> str
title	title Formatoption instance in the plotter

Methods

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

int – If 0, the title will not be rapped, otherwise it will be wrapped after the given number of characters

Notes

This wraps the title after a certain amount of characters. For wrapping the text automatically before it leaves the plot area, use `titleprops=dict(wrap=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.

dependencies = ['title']

group = 'labels'

name = 'Wrap the title'

property title
title Formatoption instance in the plotter

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

Parameters **value** – Value to update

psy_strat.plugin module

psy-strat psyplot plugin

This module defines the rcParams for the psy-strat plugin. This module will be imported when psyplot is imported. What is should contain is:

- an rcParams variable as instance of `psyplot.config.rcsetup.RcParams` that describes the configuration of your plugin
- a get_versions function that returns the version of your plugin and the ones from its requirements

Warning: Because of recursion issues, You have to load the psyplot module before loading this module! In other words, you have to type

```
import psyplot
import psy_strat.plugin
```

Functions

get_versions([requirements])	Get the versions of psy-strat and it's requirements
validate_axislinestyle(value)	Validate a dictionary containing axiscolor definitions
validate_grouper(value)	Validate the grouper formatoption
validate_hlines(value)	Validate the hlines formatoption

psy_strat.plugin.**get_versions**(*requirements=True*)
Get the versions of psy-strat and it's requirements

Parameters **requirements** (*bool*) – If True, the requirements are imported and it's versions are included

psy_strat.plugin.**validate_axislinestyle**(*value*)
Validate a dictionary containing axiscolor definitions

Parameters **value** (*dict*) – a mapping from 'left', 'right', 'bottom', 'top' to the linestyle

Returns

Return type dict

Raises **ValueError** –

psy_strat.plugin.**validate_grouper**(*value*)
Validate the grouper formatoption

Parameters **value** (tuple (float y1, str s)) – A tuple of length 2, where the first parameter 0<=y1<=1 determines the distance of the bar to the top y-axis and the second is the title of the group

psy_strat.plugin.**validate_hlines**(*value*)
Validate the hlines formatoption

Parameters **value** (*object*) – Either None, True or a color

psy_strat.strat_widget module

Module for a widget for stratigraphic plots

This module defines the *StratPlotsWidget* class that can be used to manage stratigraphic plots. It is designed as a plugin for the psyplot_gui.main.MainWindow class

Classes

GrouperItem(grouper, tree, *args, **kwargs)	An item whose contents is filled by a StratGrouper
StratPlotsWidget(*args, **kwargs)	A widget for managing the stratigraphic plots from the psy-strat package

Functions

get_stratplots_widgets([mainwindow])	Get the StraditizerWidgets from the psyplot GUI mainwindow

class psy_strat.strat_widget.**GrouperItem**(*grouper*, *tree*, **args*, ***kwargs*)
Bases: PyQt5.QtWidgets.QTreeWidgetItem

An item whose contents is filled by a StratGrouper

This item is automatically initialized by an instance of psy_strat.stratplot.StratGrouper **Methods**

add_array_children()	
show_or_hide_func(name)	Create a function that displays or hides the plot for an array

add_array_children()

show_or_hide_func(*name*)
Create a function that displays or hides the plot for an array

class psy_strat.strat_widget.**StratPlotsWidget**(**args*, ***kwargs*)
Bases: PyQt5.QtWidgets.QWidget, psyplot_gui.common.DockMixin

A widget for managing the stratigraphic plots from the psy-strat package **Methods**

add_tree(groupers[, title])	Add a new QTreeWidget to the `tabs` widget
move_selected_children(child, col)	
update_trees_from_project(project)	Add a new QTreeWidget from the main project

Attributes

dock_position	Display the dock widget at the right side of the GUI
hidden	True if there is no `psy_strat.plotter.StratPlotter` in the
stratplotter_cls	The *psy_strat.plotters.StratPlotter* class if it's module has already been imported.
title	The title of the widget

add_tree(*groupers*, *title=None*)
Add a new QTreeWidget to the `tabs` widget

dock_position = 2
Display the dock widget at the right side of the GUI

property hidden
True if there is no `psy_strat.plotter.StratPlotter` in the current main project

move_selected_children(*child*, *col*)

property stratplotter_cls
The *psy_strat.plotters.StratPlotter* class if it's module has already been imported. Otherwise None.

title = 'Stratigraphic plots'
The title of the widget

update_trees_from_project(*project*)
Add a new QTreeWidget from the main project

`psy_strat.strat_widget.`**get_stratplots_widgets**(*mainwindow=None*)
Get the `StraditizerWidgets` from the psyplot GUI mainwindow

psy_strat.stratplot module

Module to create new stratigraphic plots

This module defines the *stratplot()* function that can be used to create stratigraphic plots such as pollen diagrams

Classes

StackedGroup(arrays[, bbox, use_weakref, group])	A grouper for stacked plots
StratAllInOne(arrays[, bbox, use_weakref, group])	A *StratGroup* for single plots
StratGroup(arrays[, bbox, use_weakref, group])	Base class for visualizing stratigraphic plots
StratPercentages(arrays[, bbox, ...])	A *StratGroup* for percentages plots

Functions

stratplot(df[, group_func, formatoptions, ...])	Visualize a dataframe as a stratigraphic plot

class psy_strat.stratplot.**StackedGroup**(*arrays*, *bbox=None*, *use_weakref=True*, *group=None*)

Bases: *psy_strat.stratplot.StratAllInOne*

A grouper for stacked plots

Parameters

- **arrays** (*list of xarray.DataArray*) – The data arrays that are plotted by this *StratGroup* instance
- **bbox** (*matplotlib.transforms.Bbox*) – The bounding box for the axes
- **use_weakref** (*bool*) – If True, only weak references are used
- **group** (*str*) – The groupname of this grouper. If not given, it will be taken from the `'maingroup'` attribute of the first array

Attributes

bar_default_fmt	dict() -> new empty dictionary
default_fmt	dict() -> new empty dictionary

bar_default_fmt = {'categorical': False, 'legend': True, 'plot': 'stacked', 'title'

default_fmt = {'legend': True, 'plot': 'stacked', 'title': '%(group)s', 'titleprops

class psy_strat.stratplot.**StratAllInOne**(*arrays*, *bbox=None*, *use_weakref=True*, *group=None*)

Bases: *psy_strat.stratplot.StratGroup*

A *StratGroup* for single plots

Parameters

- **arrays** (*list of xarray.DataArray*) – The data arrays that are plotted by this *StratGroup* instance
- **bbox** (*matplotlib.transforms.Bbox*) – The bounding box for the axes
- **use_weakref** (*bool*) – If True, only weak references are used
- **group** (*str*) – The groupname of this grouper. If not given, it will be taken from the `'maingroup'` attribute of the first array

Attributes

arrays	The arrays managed by this *StratGroup*.
bar_default_fmt	dict() -> new empty dictionary
default_fmt	dict() -> new empty dictionary

Methods

from_dataset(fig, bbox, ds, variables[, ...])	Create *StratGroup* while creating a stratigraphic plot
group_plots(height)	Reimplemented to do nothing because all variables are in one axes
hide_array(name)	Hide the variable of the given *name*
is_visible(arr)	Check if the given *arr* is shown
reorder(names)	Reorder the plot objects

Continued on next page

Table 53 – continued from previous page

show_array(name)	Show the variable of the given *name*

property arrays
The arrays managed by this *StratGroup*. One array for each variable

bar_default_fmt = {'categorical': False, 'legend': True, 'title': '%(group)s', 'tit

default_fmt = {'legend': True, 'title': '%(group)s', 'titleprops': {}, 'ytickprops'

classmethod from_dataset(*fig*, *bbox*, *ds*, *variables*, *fmt=None*, *project=None*, *ax0=None*, *use_bars=False*, *group=None*)
Create *StratGroup* while creating a stratigraphic plot

Create a stratigraphic plot within the given *bbox* of *fig*.

Parameters

- **fig** (*matplotlib.figure.Figure*) – The figure to plot in
- **bbox** (*matplotlib.transforms.Bbox*) – The bounding box for the newly created axes
- **ds** (*xarray.Dataset*) – The dataset
- **variables** (*list*) – The variables that shall be plot in the given *ds*
- **project** (*psyplot.project.Project*) – The mother project. If given, only weak references are stored in the returned *StratGroup* and each array is appended to the *project*.
- **ax0** (*matplotlib.axes.Axes*) – The first subplot to share the y-axis with
- **use_bars** (*bool*) – Whether to use a bar plot or a line/area plot

Returns The newly created instance with the arrays

Return type *StratGroup*

group_plots(*height*)
Reimplemented to do nothing because all variables are in one axes

hide_array(*name*)
Hide the variable of the given *name*

Parameters **name** (*str*) – The variable name

is_visible(*arr*)
Check if the given *arr* is shown

reorder(*names*)
Reorder the plot objects

Parameters **mapping** (*dict*) – A mapping from the new index to the old one

show_array(*name*)
Show the variable of the given *name*

Parameters **name** (*str*) – The variable name

class psy_strat.stratplot.**StratGroup**(*arrays*, *bbox=None*, *use_weakref=True*, *group=None*)
Bases: object

Base class for visualizing stratigraphic plots

Parameters

- **arrays** (*list of xarray.DataArray*) – The data arrays that are plotted by this *StratGroup* instance
- **bbox** (*matplotlib.transforms.Bbox*) – The bounding box for the axes
- **use_weakref** (*bool*) – If True, only weak references are used
- **group** (*str*) – The groupname of this grouper. If not given, it will be taken from the `'maingroup'` attribute of the first array

Attributes

all_arrays	All variables of this group in the dataset
arr_names	
arrays	The arrays managed by this *StratGroup*.
axes	
bar_default_fmt	dict() -> new empty dictionary
default_fmt	The default formatoptions for the plots
figure	The figure that contains the plots
plotter_arrays	The data objects that contain the plotters
plotters	The plotters of the `arrays`

Methods

from_dataset(fig, bbox, ds, variables[, ...])	Create *StratGroup* while creating a stratigraphic plot
group_plots([height])	Group the variables visually
hide_array(name)	Hide the variable of the given *name*
is_visible(arr)	Check if the given *arr* is shown
reorder(names)	Reorder the plot objects
resize_axes(axes)	Resize the axes in this group
show_array(name)	Show the variable of the given *name*

property all_arrays
All variables of this group in the dataset

property arr_names

property arrays
The arrays managed by this *StratGroup*. One array for each variable

property axes

bar_default_fmt = {'categorical': False, 'ytickprops': {'labelleft': False, 'left':

default_fmt = {'ytickprops': {'labelleft': False, 'left': False}}
The default formatoptions for the plots

property figure
The figure that contains the plots

classmethod from_dataset(*fig*, *bbox*, *ds*, *variables*, *fmt=None*, *project=None*, *ax0=None*, *use_bars=False*, *group=None*)
Create *StratGroup* while creating a stratigraphic plot

Create a stratigraphic plot within the given *bbox* of *fig*.

Parameters

- **fig** (*matplotlib.figure.Figure*) – The figure to plot in
- **bbox** (*matplotlib.transforms.Bbox*) – The bounding box for the newly created axes
- **ds** (*xarray.Dataset*) – The dataset
- **variables** (*list*) – The variables that shall be plot in the given *ds*
- **project** (*psyplot.project.Project*) – The mother project. If given, only weak references are stored in the returned *StratGroup* and each array is appended to the *project*.
- **ax0** (*matplotlib.axes.Axes*) – The first subplot to share the y-axis with
- **use_bars** (*bool*) – Whether to use a bar plot or a line/area plot

Returns The newly created instance with the arrays

Return type *StratGroup*

group_plots(*height=None*)
Group the variables visually

Parameters **height** (*float*) – The height of the grouper. If not specified, the previous *grouper_height* attribute will be used

grouper_height = None

hide_array(*name*)
Hide the variable of the given *name*

Parameters **name** (*str*) – The variable name

is_visible(*arr*)
Check if the given *arr* is shown

property plotter_arrays
The data objects that contain the plotters

property plotters
The plotters of the *arrays*

reorder(*names*)
Reorder the plot objects

Parameters **names** (*list of str*) – The variable names that should be the first

resize_axes(*axes*)
Resize the axes in this group

show_array(*name*)
Show the variable of the given *name*

Parameters **name** (*str*) – The variable name

class psy_strat.stratplot.**StratPercentages**(*arrays*, *bbox=None*, *use_weakref=True*, *group=None*)

Bases: *psy_strat.stratplot.StratGroup*

A *StratGroup* for percentages plots

Parameters

- **arrays** (*list of xarray.DataArray*) – The data arrays that are plotted by this *StratGroup* instance

- **bbox** (*matplotlib.transforms.Bbox*) – The bounding box for the axes
- **use_weakref** (*bool*) – If True, only weak references are used
- **group** (*str*) – The groupname of this grouper. If not given, it will be taken from the `'maingroup'` attribute of the first array

Attributes

bar_default_fmt	dict() -> new empty dictionary
default_fmt	dict() -> new empty dictionary

Methods

resize_axes(axes)	Resize the axes in this group

bar_default_fmt = {'categorical': False, 'xlim': (0, 'rounded'), 'xticks': array([1

default_fmt = {'plot': 'areax', 'xlim': (0, 'rounded'), 'xticks': array([10, 30, 50

resize_axes(*axes*)
Resize the axes in this group

psy_strat.stratplot.**stratplot**(*df*, *group_func=None*, *formatoptions=None*, *ax=None*, *thresh=0.01*, *percentages=[]*, *exclude=[]*, *widths=None*, *calculate_percentages=True*, *min_percentage=20.0*, *trunc_height=0.3*, *fig=None*, *all_in_one=[]*, *stacked=[]*, *summed=[]*, *use_bars=False*, *subgroups={}*)
Visualize a dataframe as a stratigraphic plot

This functions takes a `pandas.DataFrame` and transforms it to a stratigraphic plot. The columns in the DataFrame may be grouped together using the *group_func* and the widths per group should then be specified. This function uses matplotlib axes for each subdiagram that all share a common vertical axes, the index of *df*. The variables are managed in the order of occurence in the input *df* but, however, are grouped together depending on the *group_func*.

The default is to plot every variable in *df* into separete line plots that line up vertically. You can use the *percentages* parameter for area plots, the *all_in_one* parameter for groups that should be all in one single plot (i.e. axes) and the *stacked* parameter for stacked plots.

Parameters

- **df** (*pandas.DataFrame*) – The dataframe containing the data to plot.
- **group_func** (*function*) – A function that groups the columns in the input *df* together. It must accept the name of a column and return the corresponding group name:

```
def group_func(col_name: str):
    return "name of it's group"
```

  If this parameter is not specified, each column will be assigned to the *'nogroup'* group that can then be used in the other parameters, such as *formatoptions* and *percentages*. Each group may also be divided into *subgroups* (see below), in this case, the *group_func* should return the corresponding subgroup.

- **formatoptions** (*dict*) – The formatoption for each group. Depending on the chosen plot method, this contains the formatoptions for the psyplot plotter.

- **ax** (*matplotlib.axes.Axes*) – The matplotlib axes to plot on. New axes will be created that cover all the space of the given axes. If this parameter is not specified and *fig* is None, a new matplotlib figure is created with a new matplotlib axes.
- **thresh** (*float*) – A minimum number between 0 and 100 (by default 1%) that a *percentages* column has to fullfil in order to be included in the plot. If a variable is always below this threshold, it will not be included in the figure
- **percentages** (*list of str or bool*) – The group names (see *group_func*) that represent percentage values. This variables will be visualized using an area plot and can be rescaled to sum up to 100% using the *calculate_percentages* parameter. This parameter can also be set to True if all groups shall be considered as percentage data
- **exclude** (*list of str*) – Either group names of column names in *df* that should be excluded in the plot
- **widths** (*dict*) – A mapping from group name to it's relative width in the plot. The values of this mapping should some up to 1, e.g.:

```
widths = {'group1': 0.3, 'group2': 0.5, 'group3': 0.2}
```

- **calculate_percentages** (*bool or list of str*) – If True, rescale the groups mentioned in the *percentages* parameter to sum up to 100%. In case of a list of str, this parameter represents the group (or variable) names that shall be used for the normalization
- **min_percentage** (*float*) – The minimum percentage (between 0 and 100) that should be covered by variables displaying *percentages* data. Each plot in one of the *percentages* groups will have at least have a xlim from 0 to *min_percentage*
- **trunc_height** (*float*) – A float between 0 and 1. The fraction of the *ax* that should be reserved for the group titles.
- **fig** (*matplotlib.Figure*) – The matplotlib figure to draw the plot on. If neither *ax* nor *fig* is specified, a new figure will be created.
- **all_in_one** (*list of str*) – The groups mentioned in this parameter will all be plotted in one single axes whereas the default is to plot each variable in a separate plot
- **stacked** (*list of str*) – The groups mentioned in this parameter will all be plotted in one single axes, stacked onto each other
- **summed** (*list of str*) – The groups (or subgroups) mentioned in this parameter will be summed and an extra plot will be appended to the right of the stratigraphic diagram
- **use_bars** (*list of str or bool*) – The variables specified in this parameter (or all variables if *use_bars* is `True`) will be visualized by a bar diagram, instead of a line or area plot.
- **subgroups** (*dict*) – A mapping from group name to a list of subgroups, e.g.:

```
subgroups = {'Pollen': ['Trees', 'Shrubs']}
```

  to divide an overarching group into subgroups.

Returns

- *psyplot.project.Project* – The newly created psyplot subproject that contains the displayed data
- list of *StratGroup* – The groupers that manage the different variables. There is one grouper per group

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

M

N

O

P

R

S

T

U

V

W

X

Y