psy-reg Documentation

Release 1.1.0

Philipp Sommer

Apr 09, 2018

Contents

Welcome to the psyplot plugin for visualizating and calculating regression plots. This package uses the scipy and statsmodels packages to evaluate your data, fit a regression to it and visualize it through the psy-simple plugin.

It's plot methods are the `linreg` and `densityreg` plot methods.

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

CHAPTER 1

Documentation

1.1 Installation

1.1.1 How to install

Installation using conda

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

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

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

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-reg
```

Note however, that you have to install scipy and statsmodels beforehand.

1.1.2 Dependencies

Besides the psyplot package, psy-reg uses the regression utilities from

- statsmodels: a python package for different statistical models
- scipy: The Python-based ecosystem of open-source software for mathematics, science, and engineering

1.1.3 Running the tests

First, clone out the github repository. And install psyplot, statsmodels and scipy.

After that, you can run:

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

or after having install pytest:

```
$ py.test
```

1.2 psyplot plot methods

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

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

In [2]: psy.plot.linreg
Out[2]: <psyplot.project.ProjectPlotter._register_plotter.<locals>.PlotMethod at 0x7fc9a05c6dd8>
```

linreg(*args, **kwargs)	Draw a fit from x to y
densityreg(*args, **kwargs)	Make a density plot and draw a fit from x to y of points

1.2.1 psyplot.project.plot.linreg

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

Draw a fit from x to y

This plotting method adds data arrays and plots them via *psy_reg.plotters.LinRegPlotter* plotters

To plot data from a netCDF file type:

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

Possible formatoptions are

axiscolor	*ci*	*color*	*coord*
error	*erroralpha*	*figtitle*	*figtitleprops*
figtitlesize	*figtitleweight*	*fit*	*fix*
grid	*id_color*	*ideal*	*labelprops*
labelsize	*labelweight*	*legend*	*legendlabels*
line_xlim	*linewidth*	*marker*	*markersize*
maskbetween	*maskgeq*	*maskgreater*	*maskleq*
maskless	*nboot*	*p0*	*param_bounds*
plot	*post*	*post_timing*	*sym_lims*
text	*ticksize*	*tickweight*	*tight*
title	*titleprops*	*titlesize*	*titleweight*
transpose	*xlabel*	*xlim*	*xrange*
xrotation	*xticklabels*	*xtickprops*	*xticks*
ylabel	*ylim*	*yrange*	*yrotation*
yticklabels	*ytickprops*	*yticks*	

Examples

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

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

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

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

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

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

1.2.2 psyplot.project.plot.densityreg

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

Make a density plot and draw a fit from x to y of points

This plotting method adds data arrays and plots them via *psy_reg.plotters.DensityRegPlotter* plotters

To plot data from a netCDF file type:

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

Possible formatoptions are

axiscolor	bins	bounds	cbar
cbarspacing	ci	clabel	clabelprops
clabelsize	clabelweight	cmap	color
coord	cticklabels	ctickprops	cticks
cticksize	ctickweight	datagrid	density
error	erroralpha	extend	figtitle
figtitleprops	figtitlesize	figtitleweight	fit
fix	grid	id_color	ideal
interp_bounds	labelprops	labelsize	labelweight
legend	legendlabels	levels	line_xlim
lineplot	linewidth	marker	markersize
maskbetween	maskgeq	maskgreater	maskleq
maskless	miss_color	nboot	normed
p0	param_bounds	plot	post
post_timing	precision	sym_lims	text
ticksize	tickweight	tight	title
titleprops	titlesize	titleweight	transpose
xlabel	xlim	xrange	xrotation
xticklabels	xtickprops	xticks	ylabel
ylim	yrange	yrotation	yticklabels
ytickprops	yticks		

Examples

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

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

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

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

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

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

1.3 Example Gallery

The examples provided in this section show you how to calculate and visualize a fit and access the data.

1.3.1 Creating and accessing a fit

This example shows you, how you can easily calculate and visualize a fit to your data

```
import numpy as np
import xarray as xr
import psyplot.project as psy
```

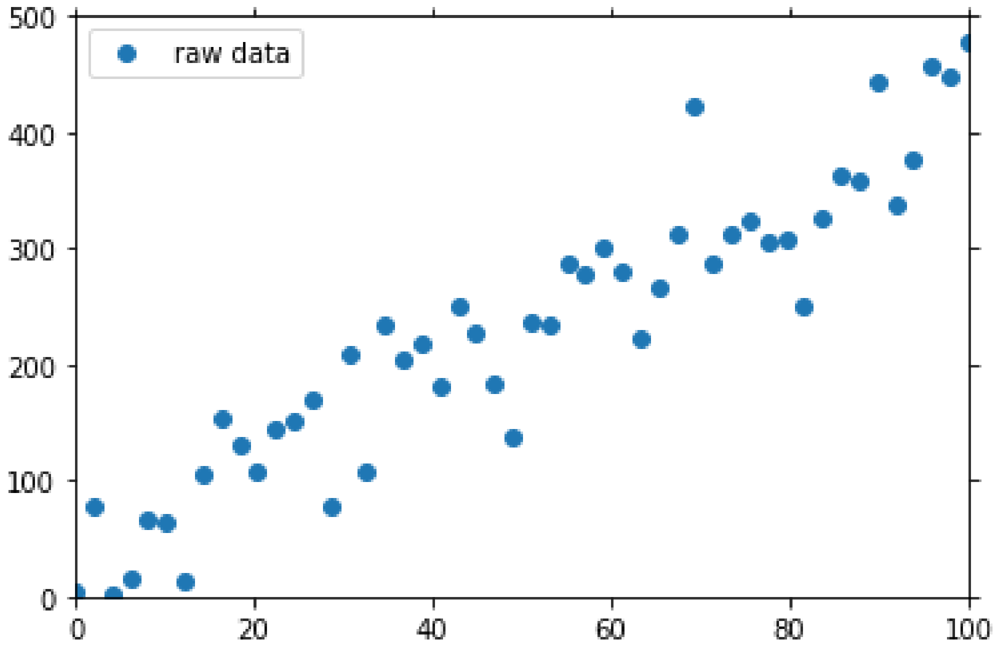
First we start with some example data to make a linear regression from the equation `y(x) = 4 * x + 30`

```
x = np.linspace(0, 100)
y = x * 4 + 30 + 50* np.random.normal(size=x.size)
ds = xr.Dataset({'x': xr.Variable(('experiment', ), x),
                 'y': xr.Variable(('experiment', ), y)})
ds
```

```
<xarray.Dataset>
Dimensions:  (experiment: 50)
Dimensions without coordinates: experiment
Data variables:
    x        (experiment) float64 0.0 2.041 4.082 6.122 8.163 10.2 12.24 ...
    y        (experiment) float64 3.264 77.61 2.623 14.89 65.1 64.44 13.2 ...
```

We can show this input data using the `lineplot` plot method from the psy-simple plugin:

```
raw = psy.plot.lineplot(
    ds, name='y', coord='x', linewidth=0, marker='o', legendlabels='raw data',
    legend='upper left')
```



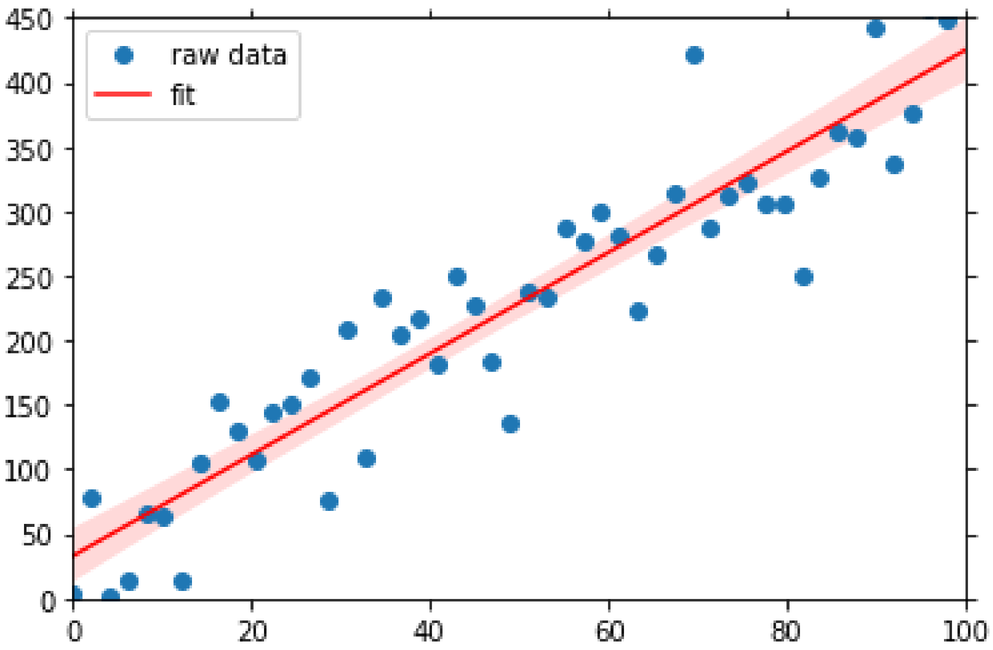
The visualization of the fit is straight forward using the `linreg` plot method:

```
fit = psy.plot.linreg(ds, ax=raw.plotters[0].ax, name='y', coord='x',
                      legendlabels='fit', color='red')
```

(continues on next page)

(continued from previous page)

```
fit.share(raw[0], keys='legend')
fit.show()
```



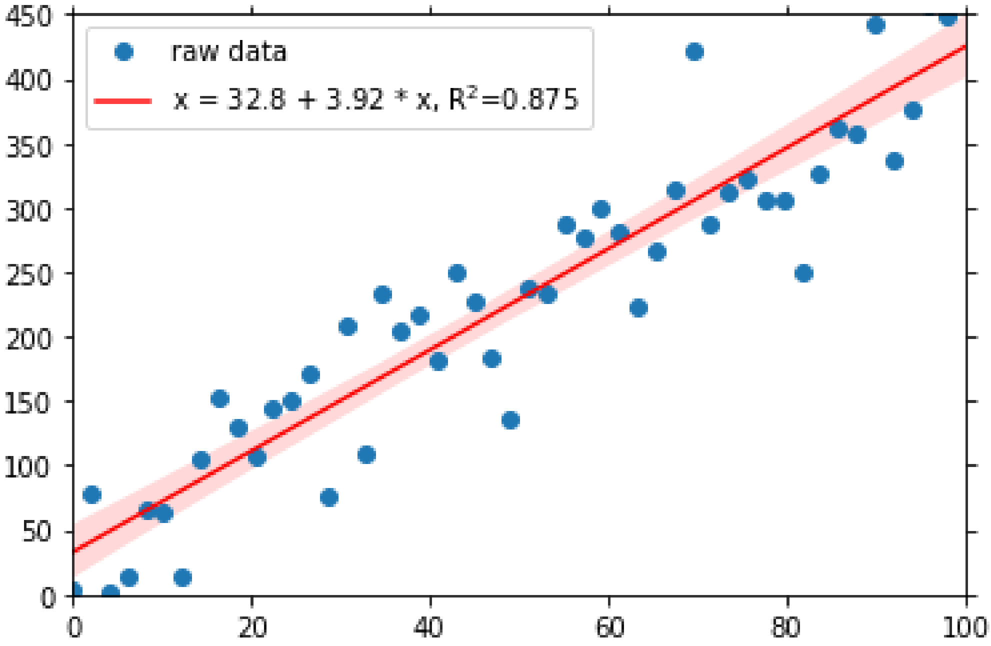
The shaded red area displays the 95% confidence interval. To access the data for the fit, just use the `plot_data` attribute:

```
data = fit[0].psy.plotter.plot_data
data[0]
```

```
<xarray.DataArray 'y' (variable: 3, x: 100)>
array([[ 32.79041 ,  36.750678,  40.710947, ..., 416.936429, 420.896698,
        424.856966],
       [ 12.779258,  17.024129,  21.277287, ..., 393.140343, 396.767974,
        400.359766],
       [ 54.296597,  57.930616,  61.625149, ..., 440.899034, 445.287076,
        449.631782]])
Coordinates:
  * x          (x) float64 0.0 1.01 2.02 3.03 4.04 5.051 6.061 7.071 8.081 ...
  * variable   (variable) <U7 'y' 'min_err' 'max_err'
Attributes:
    slope:      3.920665557503142
    intercept:  32.79041007460605
    rsquared:   0.8753991225130007
```

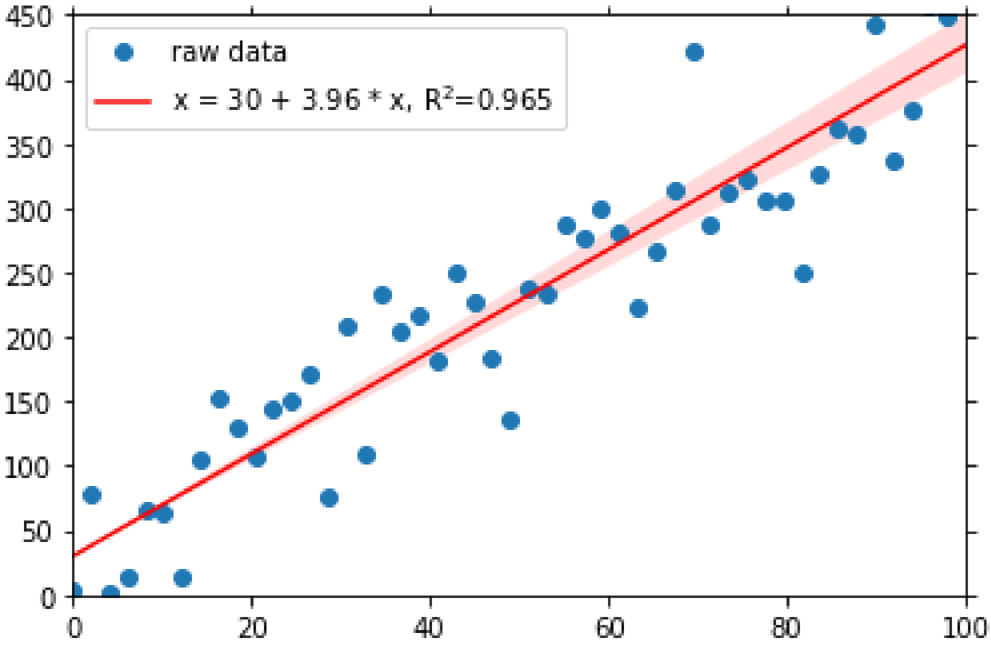
You see, that there are new attributes, `rsquared`, `intercept` and `slope`, the characteristics of the fit. As always with the dataset attributes in `psyplot`, you can visualize them, for example in the legend:

```
fit.update(legendlabels='%(yname)s = %(intercept).3g + %(slope).3g * %(xname)s, R$^2$=
↪%(rsquared).3g')
fit.show()
```



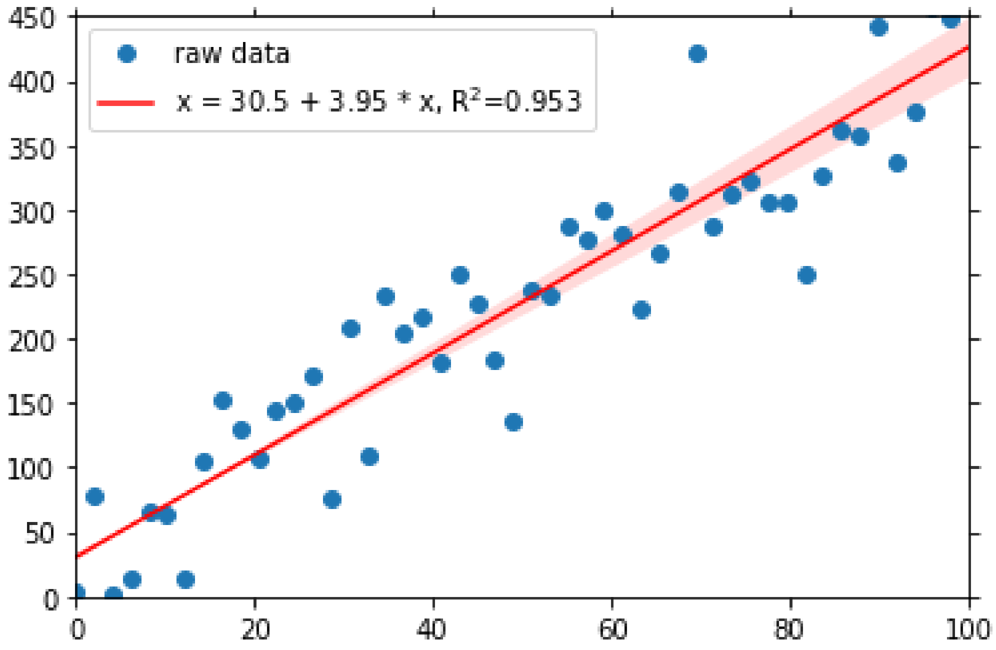
To improve the fit, we can also force the line to go through a given fix point. For example here, we know, that the fit crosses the y-line at 30:

```
fit.update(fix=30)
fit.show()
```



That works for any other point as well. E.g. we also know, that the line goes through `y = 4 * 10 + 30 = 70`:

```
fit.update(fix=[(10, 70)])
fit.show()
```



For more informations, look into the formatoptions of the `regression` group

```
fit.summaries('regression')
```

```
xrange
    Specify the range for the fit to use for the x-dimension
yrange
    Specify the range for the fit to use for the y-dimension
line_xlim
    Specify how wide the range for the plot should be
p0
    Initial parameters for the scipy.optimize.curve_fit() function
fit
    Choose the linear fitting method
fix
    Force the fit to go through a given point
nboot
    Set the number of bootstrap resamples for the confidence interval
ci
    Draw a confidence interval
ideal
    Draw an ideal line of the fit
```

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

1.3.2 Plot a fit over a density plot

Use the `densityreg` plot method to combine fits and their raw data.

This example uses artifical data to show you the capabilities of the `densityreg` plot method.

```
import numpy as np
import xarray as xr
import psyplot.project as psy
```

First we define our data which comes from multiple realizations of the underlying equation `sin(x)`

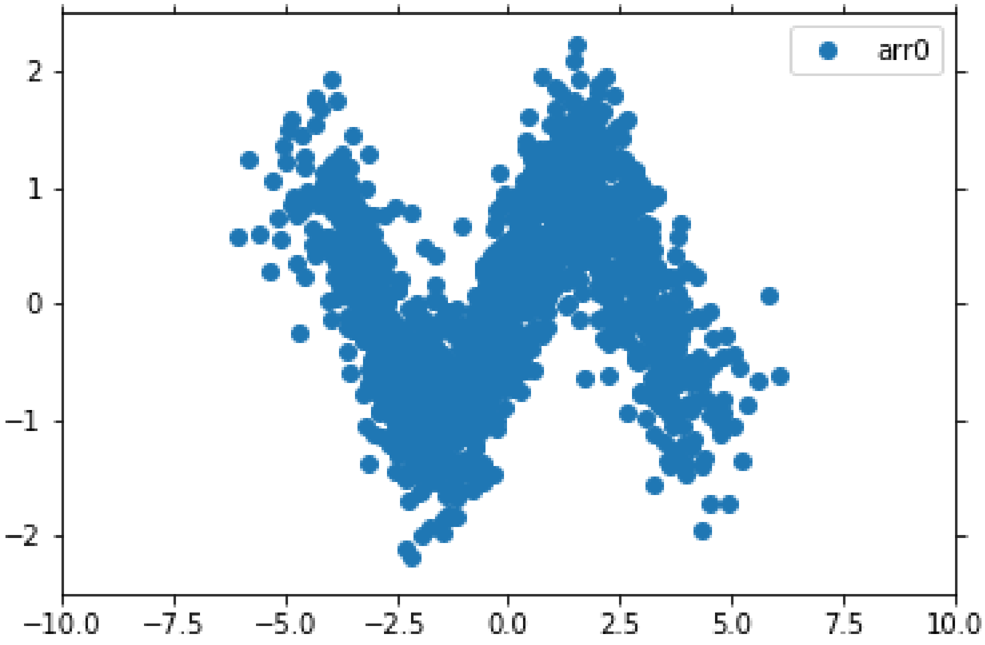
```
all_x = []
all_y = []
for i in range(30):
    deviation = np.abs(np.random.normal())
    all_x.append(np.linspace(-np.pi - deviation, np.pi + deviation))
    all_y.append(np.sin(all_x[-1]) + np.random.normal(scale=0.5, size=all_x[-1].size))
x = np.concatenate(all_x)
y = np.concatenate(all_y)
ds = xr.Dataset({'x': xr.Variable(('experiment', ), x),
                 'y': xr.Variable(('experiment', ), y)})
ds
```

```
<xarray.Dataset>
Dimensions:  (experiment: 1500)
Dimensions without coordinates: experiment
Data variables:
    x         (experiment) float64 -3.228 -3.096 -2.964 -2.833 -2.701 -2.569 ...
    y         (experiment) float64 0.5003 -0.1887 -0.01695 0.3151 -0.2478 ...
```

This dataset now contains the two variables `x` and `y`. A scatter plot of the data looks like

```
psy.plot.lineplot(ds, name='y', coord='x', marker='o', linewidth=0)
```

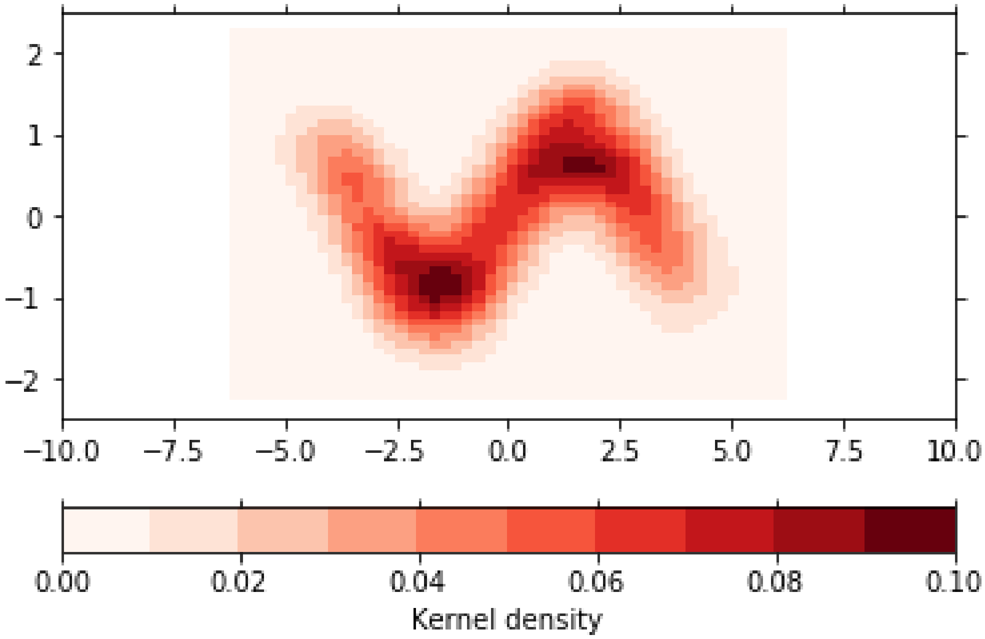
```
psyplot.project.Project([arr0: psyplot.data.InteractiveList([    arr0: 1-dim␣
↪DataArray of y, with (experiment)=(1500,), ])])
```



However, it is hard to see how many data points there are shown. Therefore this is a good candidate for a `density` plot:

```
psy.plot.density(ds, name='y', coord='x', cmap='Reds', bins=50, density='kde',
                 clabel='Kernel density')
```

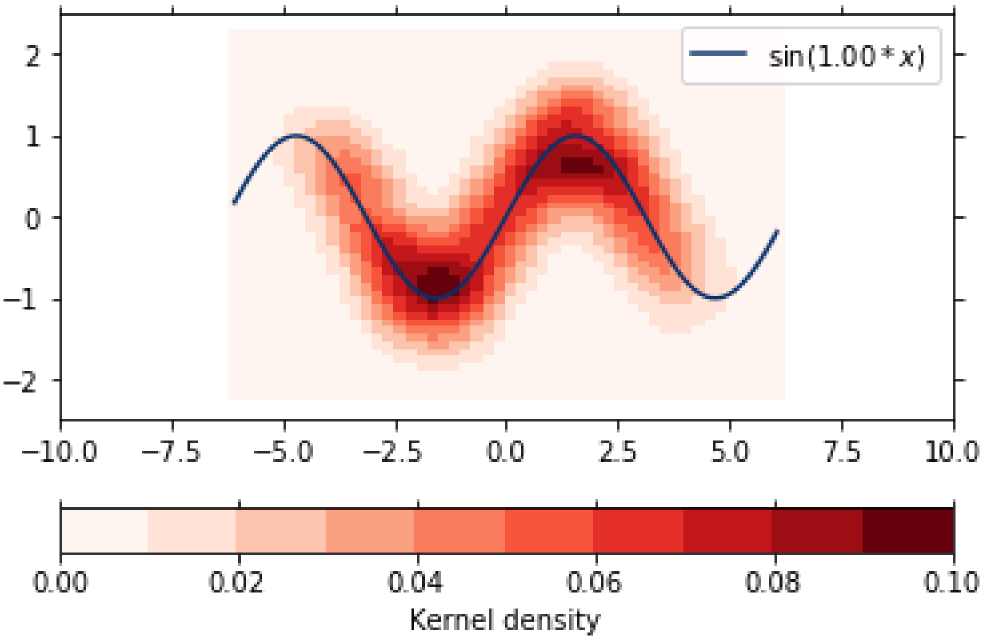
```
psyplot.project.Project([    arr1: 1-dim DataArray of y, with (experiment)=(1500,), ])
```



The densityreg plot method combines this plot with a fit through the data

```
psy.close('all')
psy.plot.densityreg(ds, name='y', coord='x', cmap='Reds', bins=50, density='kde',
                    clabel='Kernel density',
                    color='Blues_r', fit=lambda x, a: np.sin(a * x),
                    legendlabels='$\sin (%(a)1.2f * %(xname)s$)')
```

```
psyplot.project.Project([    arr0: 1-dim DataArray of y, with (experiment)=(1500,), ])
```



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

1.4 API Reference

psy-reg: Psyplot plugin for visualizing and calculating regression plots

This package contains the plotters for interactive visualization tasks with the abilities to calculate and visualize regression plots. This package uses statsmodels and scipy for it's calculations.

1.4.1 Submodules

psy_reg.plotters module

Module for fitting a linear model to the data

This module defines the *LinRegPlotter* and the *DensityRegPlotter* plotter classes that can be used to fit a linear model to the data and visualize it.

Formatoption classes

Ci(key[, plotter, index_in_list, ...])	Draw a confidence interval
FitPointDensity(key[, plotter, ...])	**param key** formatoption key in the *plotter*
FixPoint(key[, plotter, index_in_list, ...])	Force the fit to go through a given point
IdealLine(key[, plotter, index_in_list, ...])	Draw an ideal line of the fit
IdealLineColor(*args, **kwargs)	The colors of the ideal lines
InitialParameters(key[, plotter, ...])	Initial parameters for the `scipy.optimize.curve_fit()` function

Continued on next page

Table 2 – continued from previous page

`LinRegTranspose`(*args, **kwargs)	Switch x- and y-axes
`LinearRegressionFit`(*args, **kwargs)	Choose the linear fitting method
`LinearRegressionFitCombined`(*args, **kwargs)	Choose the linear fitting method
`NBoot`(key[, plotter, index_in_list, . . .])	Set the number of bootstrap resamples for the confidence interval
`ParameterBounds`(key[, plotter, . . .])	Parameter bounds for the function parameters
`XFitRange`(*args, **kwargs)	Specify the range for the fit to use for the x-dimension
`XLineRange`(*args, **kwargs)	Specify how wide the range for the plot should be
`YFitRange`(*args, **kwargs)	Specify the range for the fit to use for the y-dimension

Plotter classes

`DensityRegPlotter`([data, ax, auto_update, . . .])	A plotter that visualizes the density of points together with a linear
`LinRegPlotter`([data, ax, auto_update, . . .])	A plotter to visualize the fit on the data

Functions

`bootstrap`(x, y, func, n_boot[, random_seed])	Simple bootstrap algorithm used to estimate the confidence interval
`calc_ci`(a[, which, axis])	Return a quantile range from an array of values.

class `psy_reg.plotters.`**`Ci`**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: `psyplot.plotter.Formatoption`

Draw a confidence interval

Size of the confidence interval for the regression estimate. This will be drawn using translucent bands around the regression line. The confidence interval is estimated using a bootstrap; for large datasets, it may be advisable to avoid that computation by setting this parameter to None.

Possible types

Attributes

`dependencies`	list() -> new empty list
`fit`	fit Formatoption instance in the plotter
`fix`	fix Formatoption instance in the plotter
`group`	str(object='‘) -> str
`name`	str(object='‘) -> str
`nboot`	nboot Formatoption instance in the plotter
`priority`	int(x=0) -> integer
`transpose`	transpose Formatoption instance in the plotter

Methods

`initialize_plot`(*args, **kwargs)	Method that is called when the plot is made the first time

Continued on next page

Table 6 – continued from previous page

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

- *None* – Do not draw and calculate a confidence interval
- *float* – A quantile between 0 and 100

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 = ['transpose', 'fit', 'nboot', 'fix']

fit

fit Formatoption instance in the plotter

fix

fix Formatoption instance in the plotter

group = 'regression'

initialize_plot(**args*, ***kwargs*)

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

Parameters value – The value to use for the initialization

name = 'Draw a confidence interval'

nboot

nboot Formatoption instance in the plotter

priority = 30

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

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

Bases: psy_simple.plotters.ScalarCombinedBase, psy_simple.plotters.DensityPlotter, *psy_reg.plotters.LinRegPlotter*

A plotter that visualizes the density of points together with a linear regression

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

Data manipulation formatoptions

bins	Specify the bins of the 2D-Histogramm
density	
normed	Specify the normalization of the histogram
precision	Set the precision of the data
xrange	Specify the range of the histogram for the x-dimension
yrange	Specify the range of the histogram for the x-dimension
coord	Use an alternative variable as x-coordinate

Color coding formatoptions

cbar	Specify the position of the colorbars
color	Set the color coding
erroralpha	Set the alpha value for the error range
id_color	The colors of the ideal lines
bounds	Specify the boundaries of the colorbar
cbarspacing	Specify the spacing of the bounds in the colorbar
cmap	Specify the color map
ctickprops	Specify the font properties of the colorbar ticklabels
cticksize	Specify the font size of the colorbar ticklabels

Continued on next page

Table 8 – continued from previous page

`ctickweight`	Specify the fontweight of the colorbar ticklabels
`extend`	Draw arrows at the side of the colorbar
`levels`	The levels for the contour plot
`miss_color`	Set the color for missing values

Fitting formatoptions

`ci`	Draw a confidence interval
`fit`	Choose the linear fitting method
`fix`	Force the fit to go through a given point
`ideal`	Draw an ideal line of the fit
`line_xlim`	Specify how wide the range for the plot should be
`nboot`	Set the number of bootstrap resamples for the confidence interval
`p0`	Initial parameters for the `scipy.optimize.curve_fit()` function

Label formatoptions

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

Plot formatoptions

`error`	Visualize the error range
`lineplot`	Choose the line style of the plot
`plot`	Choose how to visualize a 2-dimensional scalar data field

Miscallaneous formatoptions

`legend`	Draw a legend
`legendlabels`	Set the labels of the arrays in the legend

Continued on next page

Table 12 – continued from previous page

param_bounds	Parameter bounds for the function parameters
ticksize	Change the ticksize of the ticklabels
tickweight	Change the fontweight of the ticks
datagrid	Show the grid of the data
interp_bounds	Interpolate grid cell boundaries for 2D plots
linewidth	Choose the width of the lines
marker	Choose the marker for points
markersize	Choose the size of the markers for points
sym_lims	Make x- and y-axis symmetric

Post processing formatoptions

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

Axes formatoptions

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

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

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

bins

Specify the bins of the 2D-Histogramm

This formatoption can be used to specify, how many bins to use. In other words, it determines the grid size of the resulting histogram or kde plot. If however you also set the *precision* formatoption keyword then the minimum of precision and the bins specified here will be used.

Possible types

- *int* – If 0, only use the bins specified by the `precision` keyword (raises an error if the `precision` is also set to 0), otherwise the number of bins to use
- *tuple (x, y) of int* – The bins for x and y explicitly

`cbar`

Specify the position of the colorbars

Possible types

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

Examples

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

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

`ci`

Draw a confidence interval

Size of the confidence interval for the regression estimate. This will be drawn using translucent bands around the regression line. The confidence interval is estimated using a bootstrap; for large datasets, it may be advisable to avoid that computation by setting this parameter to None.

Possible types

- *None* – Do not draw and calculate a confidence interval
- *float* – A quantile between 0 and 100

`clabel`

Show the colorbar label

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

- Strings like `'%Y'`, `'%b'`, etc. will be replaced using the `datetime.datetime.strftime()` method as long as the data has a time coordinate and this can be converted to a `datetime` object.

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

Possible types

str – The title for the set_label() method.

See also:

clabelsize, *clabelweight*, *clabelprops*

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

density

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`

`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`

`fit`

Choose the linear fitting method

This formatoption consists makes a linear fit of the data

Possible types

- *'fit'* – make a linear fit
- *'robust'* – make a robust linear fit
- *'poly<deg>'* – Make a polynomial fit of the order `'<deg>'`
- *function* – A callable function that takes an x-array and a y-array as input and can be used for the `scipy.optimize.curve_fit()` function
- *None* – make no fit

Notes

You can access the intercept, slope and rsquared by the correponding attribute. E.g.:

```
>>> plotter.update(
...     legendlabels='%(intercept)s + %(slope)s * x, '
...     '$R^2$=%(rsquared)s')
```

See also:

fix

fix

Force the fit to go through a given point

Possible types

- *None* – do not force the fit at all
- *float f* – make a linear fit forced through `(x, y) = (0, f)`
- *tuple (x', y')* – make a linear fit forced through `(x, y) = (x', y')`

See also:

fit

id_color

The colors of the ideal lines

Possible types

- *None* – Let it be determined by the color cycle of the *color* formatoption
- *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

See also:

ideal

ideal

Draw an ideal line of the fit

Possible types

- *None* – Don't draw an ideal line
- *list of floats* – The parameters for the line. If the *fit* formatoption is in `'robust'` or `'fit'`, then the first value corresponds to the interception, the second to the slope. Otherwise the list cornensponds to the parameters as used in the fit function of the lines

- *list of list of floats* – The same as above but with the specification for each array

See also:

id_color

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*

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`

line_xlim

Specify how wide the range for the plot should be

This formatoption specifies the range of the line to use

Possible types

- *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:

`xrange`

`lineplot`

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()`
- *str or list of str* – The line style string to use ([‘solid’ | ‘dashed’, ‘dashdot’, ‘dotted’ | (offset, on-off-dash-seq) | ‘-‘ | ‘–’ | ‘-.’ | ‘:’ | ‘None’ | ‘ ‘ | ‘’]).

`nboot`

Set the number of bootstrap resamples for the confidence interval

Parameters `int` – Number of bootstrap resamples used to estimate the `ci`. The default value attempts to balance time and stability; you may want to increase this value for “final” versions of plots.

See also:

`ci`

`normed`

Specify the normalization of the histogram

This formatoption can be used to normalize the histogram. It has no effect if the `density` formatoption is set to `'kde'`

Possible types

- *None* – Do not make any normalization
- *str* – One of

  counts To make the normalization based on the total number counts

  area To make the normalization basen on the total number of counts and area (the default behaviour of `numpy.histogram2d()`)

See also:

density

p0

Initial parameters for the `scipy.optimize.curve_fit()` function

This formatoptions can be used to set the initial parameters if the value of the *fit* formatoption is a callable function.

Note that the automatic estimation uses the boundaries of the *param_bounds* formatoption. This only works if the boundaries are given for each parameter and finite.

Possible types

- *'auto'* – The initial parameters are estimated automatically using the `from scipy.optimize.differential_evolution()` function
- *list of floats* – The initial parameters
- *list of list of floats or 'auto'* – A combination of the above types where each corresponds to one data array

param_bounds

Parameter bounds for the function parameters

This formatoption can be used to specify the boundaries for the parameters. It only has an effect if the value of the *fit* formatoption is a callable function.

These bounds will also be used by the *p0* formatoption to estimate the initial parameters.

Possible types

- *None* – Use open boundaries
- *list of tuples with length 2* – The boundaries for each of the parameters
- *list of tuples or None* – A combination of the above types where each corresponds to one data array

plot

Choose how to visualize a 2-dimensional scalar data field

Possible types

- *None* – Don't make any plotting

- *'mesh'* – Use the `matplotlib.pyplot.pcolormesh()` function to make the plot or the `matplotlib.pyplot.tripcolor()` for an unstructered grid
- *'tri'* – Use the `matplotlib.pyplot.tripcolor()` function to plot data on a triangular grid
- *'contourf'* – Make a filled contour plot using the `matplotlib.pyplot.contourf()` function or the `matplotlib.pyplot.tricontourf()` for triangular data. The levels for the contour plot are controlled by the *`levels`* formatoption
- *'tricontourf'* – Make a filled contour plot using the `matplotlib.pyplot.tricontourf()` function

`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

precision

Set the precision of the data

This formatoption can be used to specify the precision of the data which then will be the minimal bin width of the 2D histogram or the bandwith of the kernel size (if the `density` formatoption is set to `'kde'`)

Possible types

- *float* – If 0, this formatoption has no effect at all. Otherwise it is assumed to be the precision of the data
- *str* – One of `{'scott' | 'silverman'}`. If the `density` formatoption is set to `'kde'`, this describes the method how to calculate the bandwidth

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`*

`title`

Show the title

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

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

Possible types

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

Notes

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

See also:

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

`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

`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`

`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`

xrange

Specify the range of the histogram for the x-dimension

This formatoption specifies the minimum and maximum of the histogram in the x-dimension

Possible types

- *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

Notes

This formatoption always acts on the coordinate, no matter what the value of the `transpose` formatoption is

See also:

`yrange`

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`*

`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`

`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`

`yrange`
Specify the range of the histogram for the x-dimension

This formatoption specifies the minimum and maximum of the histogram in the x-dimension

Possible types

- *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

Notes

This formatoption always acts on the DataArray, no matter what the value of the `transpose` formatoption is

See also:

`xrange`

`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

`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.

`bounds`

Specify the boundaries of the colorbar

Possible types

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

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

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

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

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

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

---

Examples

Plot 11 bounds over the whole data range:

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

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

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

Plot logarithmic bounds:

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

See also:

cmap Specifies the colormap

cbarspacing
Specify the spacing of the bounds in the colorbar

Possible types

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

cmap
Specify the color map

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

Possible types

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

See also:

bounds specifies the boundaries of the colormap

ctickprops
Specify the font properties of the colorbar ticklabels

Possible types

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

See also:

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

cticksize
Specify the font size of the colorbar ticklabels

Possible types

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

See also:

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

ctickweight

Specify the fontweight of the colorbar ticklabels

Possible types

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

See also:

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

extend

Draw arrows at the side of the colorbar

Possible types

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

levels

The levels for the contour plot

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

Possible types

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

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

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

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

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

sym Same as minmax but symmetric around zero

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

`miss_color`
Set the color for missing values

Possible types

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

`clabelprops`
Properties of the Colorbar label

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

`clabel`, *`clabelsize`*, *`clabelweight`*

`clabelsize`
Set the size of the Colorbar label

Possible types

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

See also:

`clabel`, *`clabelweight`*, *`clabelprops`*

`clabelweight`
Set the fontweight of the Colorbar label

Possible types

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

See also:

`clabel`, *`clabelsize`*, *`clabelprops`*

figtitle

Plot a figure title

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

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

Possible types

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

Notes

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

See also:

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

figtitleprops

Properties of the figure title

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

`figtitle`, *`figtitlesize`*, *`figtitleweight`*

figtitlesize

Set the size of the figure title

Possible types

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

See also:

`figtitle`, `figtitleweight`, `figtitleprops`

`figtitleweight`

Set the fontweight of the figure title

Possible types

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

See also:

`figtitle`, `figtitlesize`, `figtitleprops`

`text`

Add text anywhere on the plot

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

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

Possible types

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

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

See also:

`title`, *`figtitle`*

`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`*

`datagrid`

Show the grid of the data

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

Possible types

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

`interp_bounds`
Interpolate grid cell boundaries for 2D plots

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

Possible types

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

`linewidth`
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

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!

`cticklabels`

Specify the colorbar ticklabels

Possible types

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

See also:

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

`cticks`

Specify the tick locations of the colorbar

Possible types

- *None* – use the default ticks

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

  data plot the ticks exactly where the data is.

  mid plot the ticks in the middle of the data.

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

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

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

  sym Same as minmax but symmetric around zero

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

See also:

`cticklabels`

`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`

class `psy_reg.plotters.`**`FitPointDensity`**(*key*, *plotter=None*, *index_in_list=None*, *additional_children=[]*, *additional_dependencies=[]*, ***kwargs*)

Bases: `psy_simple.plotters.PointDensity`

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.

Attributes

`bins`	bins Formatoption instance in the plotter

Continued on next page

Table 17 – continued from previous page

children	list() -> new empty list
coord	coord Formatoption instance in the plotter
line_xlim	line_xlim Formatoption instance in the plotter
normed	normed Formatoption instance in the plotter
precision	precision Formatoption instance in the plotter
xrange	xrange Formatoption instance in the plotter
yrange	yrange Formatoption instance in the plotter

bins
bins Formatoption instance in the plotter

children = ['line_xlim']

coord
coord Formatoption instance in the plotter

line_xlim
line_xlim Formatoption instance in the plotter

normed
normed Formatoption instance in the plotter

precision
precision Formatoption instance in the plotter

xrange
xrange Formatoption instance in the plotter

yrange
yrange Formatoption instance in the plotter

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

Force the fit to go through a given point

Possible types

Attributes

connections	list() -> new empty list
fit	fit Formatoption instance in the plotter
group	str(object=’‘) -> str
name	str(object=’‘) -> str
priority	int(x=0) -> integer

Methods

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

- *None* – do not force the fit at all
- *float f* – make a linear fit forced through `(x, y) = (0, f)`

- *tuple (x', y')* – make a linear fit forced through `(x, y) = (x', y')`

See also:

fit

Parameters

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

connections = ['fit']

fit
fit Formatoption instance in the plotter

group = 'regression'

name = 'Force the fit to go through a given point'

priority = 30

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

Parameters value – Value to update

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

Draw an ideal line of the fit

Possible types

Attributes

dependencies	list() -> new empty list
fit	fit Formatoption instance in the plotter
group	str(object='‘) -> str
id_color	id_color Formatoption instance in the plotter
plot	plot Formatoption instance in the plotter

Methods

initialize_plot(*args, **kwargs)	Method that is called when the plot is made the first time
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 an ideal line
- *list of floats* – The parameters for the line. If the `fit` formatoption is in `'robust'` or `'fit'`, then the first value corresponds to the interception, the second to the slope. Otherwise the list cornensponds to the parameters as used in the fit function of the lines
- *list of list of floats* – The same as above but with the specification for each array

See also:

`id_color`

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 = ['fit', 'id_color', 'plot']

fit

fit Formatoption instance in the plotter

group = 'regression'

id_color

id_color Formatoption instance in the plotter

initialize_plot(**args*, ***kwargs*)

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

Parameters value – The value to use for the initialization

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.

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

Parameters **value** – Value to update

class psy_reg.plotters.**IdealLineColor**(**args*, ***kwargs*)
Bases: `psy_simple.plotters.LineColors`

The colors of the ideal lines

Possible types

Attributes

color	color Formatoption instance in the plotter
dependencies	list() -> new empty list
ideal	ideal Formatoption instance in the plotter
parents	list() -> new empty list
priority	int(x=0) -> integer

Methods

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

- *None* – Let it be determined by the color cycle of the *color* formatoption
- *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

See also:

ideal

color
color Formatoption instance in the plotter

dependencies = ['color']

ideal
ideal Formatoption instance in the plotter

parents = ['ideal']

priority = 10

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

Parameters **value** – Value to update

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

Bases: psyplot.plotter.Formatoption

Initial parameters for the scipy.optimize.curve_fit() function

This formatoptions can be used to set the initial parameters if the value of the *fit* formatoption is a callable function.

Note that the automatic estimation uses the boundaries of the *param_bounds* formatoption. This only works if the boundaries are given for each parameter and finite.

Possible types

Attributes

connections	list() -> new empty list
data_dependent	bool(x) -> bool
dependencies	list() -> new empty list
fit	fit Formatoption instance in the plotter
group	str(object='') -> str
name	str(object='') -> str
param_bounds	param_bounds Formatoption instance in the plotter
priority	int(x=0) -> integer

Methods

p0([i])	
update(value)	Method that is call to update the formatoption on the axes

- *'auto'* – The initial parameters are estimated automatically using the from scipy.optimize.differential_evolution() function
- *list of floats* – The initial parameters
- *list of list of floats or 'auto'* – A combination of the above types where each corresponds to one data array

Parameters

- **key** (*str*) – formatoption key in the *plotter*
- **plotter** (*psyplot.plotter.Plotter*) – Plotter instance that holds this formatoption. If None, it is assumed that this instance serves as a descriptor.
- **index_in_list** (*int or None*) – The index that shall be used if the data is a psyplot.InteractiveList
- **additional_children** (*list or str*) – Additional children to use (see the children attribute)
- **additional_dependencies** (*list or str*) – Additional dependencies to use (see the *dependencies* attribute)

- **`**kwargs`** – Further keywords may be used to specify different names for children, dependencies and connection formatoptions that match the setup of the plotter. Hence, keywords may be anything of the `children`, *`dependencies`* and *`connections`* attributes, with values being the name of the new formatoption in this plotter.

`connections = ['fit']`

`data_dependent = True`

`dependencies = ['param_bounds']`

`fit`

fit Formatoption instance in the plotter

`group = 'regression'`

`name = 'Initial parameter values for the fit'`

`p0`(*i=None*)

`param_bounds`

param_bounds Formatoption instance in the plotter

`priority = 30`

`update`(*value*)

Method that is call to update the formatoption on the axes

Parameters **`value`** – Value to update

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

Bases: `psy_simple.plotters.LinePlotter`

A plotter to visualize the fit on the data

The most important formatoptions are the *`fit`* and *`ci`* formatoption. Otherwise this plotter behaves like the `psyplot.plotter.simple.LinePlotter` plotter class

Parameters

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

- **`**kwargs`** – Any formatoption key from the `formatoptions` attribute that shall be used

Attributes

`allowed_vars`	int(x=0) -> integer

Fitting formatoptions

`ci`	Draw a confidence interval
`fit`	Choose the linear fitting method
`fix`	Force the fit to go through a given point
`ideal`	Draw an ideal line of the fit
`line_xlim`	Specify how wide the range for the plot should be
`nboot`	Set the number of bootstrap resamples for the confidence interval
`p0`	Initial parameters for the `scipy.optimize.curve_fit()` function
`xrange`	Specify the range for the fit to use for the x-dimension
`yrange`	Specify the range for the fit to use for the y-dimension

Color coding formatoptions

`id_color`	The colors of the ideal lines
`color`	Set the color coding
`erroralpha`	Set the alpha value for the error range

Miscallaneous formatoptions

`param_bounds`	Parameter bounds for the function parameters
`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
`sym_lims`	Make x- and y-axis symmetric
`ticksize`	Change the ticksize of the ticklabels
`tickweight`	Change the fontweight of the ticks

Axes formatoptions

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

Label formatoptions

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
title	Show the title
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

Masking formatoptions

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

Axis tick formatoptions

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

Data manipulation formatoptions

coord	Use an alternative variable as x-coordinate

Plot formatoptions

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

Post processing formatoptions

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

`allowed_vars = 1`

ci

Draw a confidence interval

Size of the confidence interval for the regression estimate. This will be drawn using translucent bands around the regression line. The confidence interval is estimated using a bootstrap; for large datasets, it may be advisable to avoid that computation by setting this parameter to None.

Possible types

- *None* – Do not draw and calculate a confidence interval
- *float* – A quantile between 0 and 100

fit

Choose the linear fitting method

This formatoption consists makes a linear fit of the data

Possible types

- *'fit'* – make a linear fit
- *'robust'* – make a robust linear fit
- *'poly<deg>'* – Make a polynomial fit of the order `'<deg>'`
- *function* – A callable function that takes an x-array and a y-array as input and can be used for the `scipy.optimize.curve_fit()` function
- *None* – make no fit

Notes

You can access the intercept, slope and rsquared by the correponding attribute. E.g.:

```
>>> plotter.update(
...     legendlabels='%(intercept)s + %(slope)s * x, '
...     '$R^2$=%(rsquared)s')
```

See also:

fix

fix

Force the fit to go through a given point

Possible types

- *None* – do not force the fit at all
- *float f* – make a linear fit forced through `(x, y) = (0, f)`
- *tuple (x', y')* – make a linear fit forced through `(x, y) = (x', y')`

See also:

fit

id_color

The colors of the ideal lines

Possible types

- *None* – Let it be determined by the color cycle of the `color` formatoption
- *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

See also:

`ideal`

ideal

Draw an ideal line of the fit

Possible types

- *None* – Don't draw an ideal line
- *list of floats* – The parameters for the line. If the `fit` formatoption is in `'robust'` or `'fit'`, then the first value corresponds to the interception, the second to the slope. Otherwise the list cornensponds to the parameters as used in the fit function of the lines
- *list of list of floats* – The same as above but with the specification for each array

See also:

`id_color`

line_xlim

Specify how wide the range for the plot should be

This formatoption specifies the range of the line to use

Possible types

- *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:

xrange

nboot

Set the number of bootstrap resamples for the confidence interval

Parameters **int** – Number of bootstrap resamples used to estimate the `ci`. The default value attempts to balance time and stability; you may want to increase this value for "final" versions of plots.

See also:

ci

p0

Initial parameters for the `scipy.optimize.curve_fit()` function

This formatoptions can be used to set the initial parameters if the value of the *fit* formatoption is a callable function.

Note that the automatic estimation uses the boundaries of the *param_bounds* formatoption. This only works if the boundaries are given for each parameter and finite.

Possible types

- *'auto'* – The initial parameters are estimated automatically using the `from scipy.optimize.differential_evolution()` function
- *list of floats* – The initial parameters
- *list of list of floats or 'auto'* – A combination of the above types where each corresponds to one data array

param_bounds

Parameter bounds for the function parameters

This formatoption can be used to specify the boundaries for the parameters. It only has an effect if the value of the *fit* formatoption is a callable function.

These bounds will also be used by the *p0* formatoption to estimate the initial parameters.

Possible types

- *None* – Use open boundaries
- *list of tuples with length 2* – The boundaries for each of the parameters
- *list of tuples or None* – A combination of the above types where each corresponds to one data array

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

xrange

Specify the range for the fit to use for the x-dimension

This formatoption specifies the minimum and maximum of the fit in the x-dimension

Possible types

- *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

Notes

This formatoption always acts on the coordinate, no matter what the value of the `transpose` formatoption is

See also:

`yrange`, `line_xlim`

yrange

Specify the range for the fit to use for the y-dimension

This formatoption specifies the minimum and maximum of the fit in the y-dimension

Possible types

- *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

Notes

This formatoption always acts on the coordinate, no matter what the value of the *transpose* formatoption is

See also:

xrange

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*

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!

`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`

`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*

title

Show the title

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

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

Possible types

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

Notes

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

See also:

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

`titleprops`

Properties of the title

Specify the font properties of the figure title manually.

Possible types

dict – Items may be any valid text property

See also:

`title`, *`titlesize`*, *`titleweight`*

`titlesize`

Set the size of the title

Possible types

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

See also:

`title`, *`titleweight`*, *`titleprops`*

titleweight

Set the fontweight of the title

Possible types

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

See also:

title, *titlesize*, *titleprops*

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`

`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`

`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

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]
```

(continues on next page)

(continued from previous page)

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

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

`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()`
- *str or list of str* – The line style string to use ([‘solid’ | ‘dashed’, ‘dashdot’, ‘dotted’ | (offset, on-off-dash-seq) | ‘-‘ | ‘–’ | ‘-.’ | ‘:’ | ‘None’ | ‘ ‘ | ‘’]).

post

Apply your own postprocessing script

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

Possible types

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

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

Examples

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

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

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

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

See also:

`post_timing` Determine the timing of this formatoption

post_timing

Determine when to run the *`post`* formatoption

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

Possible types

- *'never'* – Never run post processing scripts
- *'always'* – Always run post processing scripts

- *'replot'* – Only run post processing scripts when the data changes or a replot is necessary

See also:

post The post processing formatoption

class psy_reg.plotters.**LinRegTranspose**(**args*, ***kwargs*)

Bases: psy_simple.plotters.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

Attributes

priority	int(x=0) -> integer

bool – If True, axes are switched

priority = 30

class psy_reg.plotters.**LinearRegressionFit**(**args*, ***kwargs*)

Bases: psyplot.plotter.Formatoption

Choose the linear fitting method

This formatoption consists makes a linear fit of the data

Possible types

Attributes

coord	coord Formatoption instance in the plotter
data_dependent	bool(x) -> bool
dependencies	list() -> new empty list
fix	fix Formatoption instance in the plotter
func_args	The arguments for the fit function if the method is
group	str(object='') -> str
line_xlim	line_xlim Formatoption instance in the plotter
name	str(object='') -> str
p0	p0 Formatoption instance in the plotter
param_bounds	param_bounds Formatoption instance in the plotter
priority	int(x=0) -> integer
transpose	transpose Formatoption instance in the plotter
xrange	xrange Formatoption instance in the plotter
yrange	yrange Formatoption instance in the plotter

Methods

get_kwargs(i)	Get the fitting kwargs for the line at index *i*
get_xline([i])	Get the x-data for the best fit line
get_xy(i, da)	
make_fit(i, x, y[, x_line])	
set_method(i)	
update(value)	Method that is call to update the formatoption on the axes

- *'fit'* – make a linear fit
- *'robust'* – make a robust linear fit
- *'poly<deg>'* – Make a polynomial fit of the order `'<deg>'`
- *function* – A callable function that takes an x-array and a y-array as input and can be used for the `scipy.optimize.curve_fit()` function
- *None* – make no fit

Notes

You can access the intercept, slope and rsquared by the correponding attribute. E.g.:

```
>>> plotter.update(
...     legendlabels='%(intercept)s + %(slope)s * x, '
...     '$R^2$=%(rsquared)s')
```

See also:

fix

coord
    coord Formatoption instance in the plotter

data_dependent = True

dependencies = ['transpose', 'fix', 'xrange', 'yrange', 'coord', 'line_xlim', 'p0', 'p

fix
    fix Formatoption instance in the plotter

func_args
    The arguments for the fit function if the `method` is 'curve_fit'

get_kwargs(*i*)
    Get the fitting kwargs for the line at index *i*

get_xline(*i=0*)
    Get the x-data for the best fit line

get_xy(*i, da*)

group = 'regression'

line_xlim
    line_xlim Formatoption instance in the plotter

make_fit(*i, x, y, x_line=None, **kwargs*)

name = 'Change the fit method'

p0
p0 Formatoption instance in the plotter

param_bounds
param_bounds Formatoption instance in the plotter

priority = 30

set_method(*i*)

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

xrange
xrange Formatoption instance in the plotter

yrange
yrange Formatoption instance in the plotter

class psy_reg.plotters.**LinearRegressionFitCombined**(**args*, ***kwargs*)
Bases: *psy_reg.plotters.LinearRegressionFit*

Choose the linear fitting method

This formatoption consists makes a linear fit of the data

Possible types

Attributes

coord	coord Formatoption instance in the plotter
fix	fix Formatoption instance in the plotter
line_xlim	line_xlim Formatoption instance in the plotter
p0	p0 Formatoption instance in the plotter
param_bounds	param_bounds Formatoption instance in the plotter
transpose	transpose Formatoption instance in the plotter
xrange	xrange Formatoption instance in the plotter
yrange	yrange Formatoption instance in the plotter

Methods

set_data(data[, i])	Reimplemented to change the *arr_name* attribute of the given array

- *'fit'* – make a linear fit
- *'robust'* – make a robust linear fit
- *'poly<deg>'* – Make a polynomial fit of the order `'<deg>'`
- *function* – A callable function that takes an x-array and a y-array as input and can be used for the `scipy.optimize.curve_fit()` function
- *None* – make no fit

Notes

You can access the intercept, slope and rsquared by the correponding attribute. E.g.:

```
>>> plotter.update(
...     legendlabels='%(intercept)s + %(slope)s * x, '
...     '$R^2$=%(rsquared)s')
```

See also:

fix

coord
coord Formatoption instance in the plotter

fix
fix Formatoption instance in the plotter

line_xlim
line_xlim Formatoption instance in the plotter

p0
p0 Formatoption instance in the plotter

param_bounds
param_bounds Formatoption instance in the plotter

set_data(*data*, *i=None*)
Reimplemented to change the *arr_name* attribute of the given array

transpose
transpose Formatoption instance in the plotter

xrange
xrange Formatoption instance in the plotter

yrange
yrange Formatoption instance in the plotter

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

Set the number of bootstrap resamples for the confidence interval

Parameters int – Number of bootstrap resamples used to estimate the ci. The default value attempts to balance time and stability; you may want to increase this value for “final” versions of plots.

Attributes

group	str(object=’‘) -> str
name	str(object=’‘) -> str
priority	int(x=0) -> integer

Methods

update(value)	Does nothing.

See also:

`ci`

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 = 'regression'

name = 'Set the bootstrapping number to calculate the confidence interval'

priority = 30

update(*value*)

Does nothing. The work is done by the `Ci` formatoption

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

Bases: `psyplot.plotter.Formatoption`

Parameter bounds for the function parameters

This formatoption can be used to specify the boundaries for the parameters. It only has an effect if the value of the `fit` formatoption is a callable function.

These bounds will also be used by the `p0` formatoption to estimate the initial parameters.

Possible types

Methods

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

- *None* – Use open boundaries
- *list of tuples with length 2* – The boundaries for each of the parameters
- *list of tuples or None* – A combination of the above types where each corresponds to one data array

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.

update(*value*)

Method that is call to update the formatoption on the axes

Parameters **value** – Value to update

class psy_reg.plotters.**XFitRange**(**args*, ***kwargs*)

Bases: psy_simple.plotters.Hist2DXRange

Specify the range for the fit to use for the x-dimension

This formatoption specifies the minimum and maximum of the fit in the x-dimension

Possible types

Attributes

coord	coord Formatoption instance in the plotter
group	str(object='') -> str
plot	plot Formatoption instance in the plotter
range	The range for each of the curves
transpose	transpose Formatoption instance in the plotter

Methods

set_limit(*args)	The method to set the minimum and maximum limit
update(value)	Method that is call to update the formatoption on the axes

- *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

Notes

This formatoption always acts on the coordinate, no matter what the value of the `transpose` formatoption is

See also:

`yrange`, `line_xlim`

`coord`
coord Formatoption instance in the plotter

`group = 'regression'`

`plot`
plot Formatoption instance in the plotter

`range`
The range for each of the curves

`set_limit`(**args*)
The method to set the minimum and maximum limit

Parameters

- **`min_val`** (`float`) – The value for the lower limit
- **`max_val`** (`float`) – The value for the upper limit

`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

class psy_reg.plotters.**XLineRange**(**args*, ***kwargs*)
Bases: `psy_reg.plotters.XFitRange`

Specify how wide the range for the plot should be

This formatoption specifies the range of the line to use

Possible types

Attributes

`coord`	coord Formatoption instance in the plotter
`plot`	plot Formatoption instance in the plotter
`transpose`	transpose Formatoption instance in the plotter

- *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:

`xrange`

`coord`
coord Formatoption instance in the plotter

`plot`
plot Formatoption instance in the plotter

`transpose`
transpose Formatoption instance in the plotter

class `psy_reg.plotters.`**`YFitRange`**(**args*, ***kwargs*)
Bases: `psy_simple.plotters.Hist2DYRange`

Specify the range for the fit to use for the y-dimension

This formatoption specifies the minimum and maximum of the fit in the y-dimension

Possible types

Attributes

`coord`	coord Formatoption instance in the plotter
`group`	str(object=’‘) -> str
`plot`	plot Formatoption instance in the plotter
`range`	The range for each of the curves
`transpose`	transpose Formatoption instance in the plotter

Methods

`set_limit`(*args)	The method to set the minimum and maximum limit
`update`(value)	Method that is call to update the formatoption on the axes

- *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

Notes

This formatoption always acts on the coordinate, no matter what the value of the `transpose` formatoption is

See also:

`xrange`

`coord`
coord Formatoption instance in the plotter

`group = 'regression'`

`plot`
plot Formatoption instance in the plotter

`range`
The range for each of the curves

`set_limit`(**args*)
The method to set the minimum and maximum limit

Parameters

- **`min_val`** (`float`) – The value for the lower limit
- **`max_val`** (`float`) – The value for the upper limit

`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

`psy_reg.plotters.`**`bootstrap`**(*x*, *y*, *func*, *n_boot*, *random_seed=None*, ***kwargs*)
Simple bootstrap algorithm used to estimate the confidence interval

This function is motivated by seaborns bootstrap algorithm `seaborn.algorithms.bootstrap()`

`psy_reg.plotters.`**`calc_ci`**(*a*, *which=95*, *axis=None*)
Return a quantile range from an array of values.

psy_reg.plugin module

psy-simple psyplot plugin

This module defines the rcParams for the psy-simple plugin

Classes

validate_list([dtype, length, listtype])	Validate a list of the specified *dtype*

Functions

get_versions([requirements])	
patch_prior_1_0(plotter_d, versions)	Patch psy_reg plotters for versions smaller than 1.0
validate_callable(val)	
validate_fit(val)	
validate_fix(val)	
validate_ideal(val)	
validate_iter(value)	Validate that the given value is an iterable
validate_line_xlim(val)	
validate_p0(value)	
validate_param_bounds(value)	
validate_stringlist(s)	Validate a list of strings

Data

patches	patches to apply when loading a project
rcParams	the RcParams for the psy-reg plugin

psy_reg.plugin.**get_versions**(*requirements=True*)

psy_reg.plugin.**patch_prior_1_0**(*plotter_d*, *versions*)

Patch psy_reg plotters for versions smaller than 1.0

Before psyplot 1.0.0, the plotters in the psy_reg package where part of the psyplot.plotter.linreg module. This has to be corrected

psy_reg.plugin.**patches = {('psyplot.plotter.linreg', 'DensityRegPlotter'): <function patcl**

patches to apply when loading a project

psy_reg.plugin.**rcParams**

the RcParams for the psy-reg plugin

psy_reg.plugin.**validate_callable**(*val*)

psy_reg.plugin.**validate_fit**(*val*)

psy_reg.plugin.**validate_fix**(*val*)

psy_reg.plugin.**validate_ideal**(*val*)

psy_reg.plugin.**validate_iter**(*value*)

Validate that the given value is an iterable

psy_reg.plugin.**validate_line_xlim**(*val*)

class psy_reg.plugin.**validate_list**(*dtype=None*, *length=None*, *listtype=<class 'list'>*)

Bases: object

Validate a list of the specified *dtype*

Parameters dtype (*object*) – A datatype (e.g. float) that shall be used for the conversion

Attributes

None	data type (e.g. `float`) used for the conversion

Initialization function

dtype = None

data type (e.g. `float`) used for the conversion

`psy_reg.plugin.`**`validate_p0`**(*value*)

`psy_reg.plugin.`**`validate_param_bounds`**(*value*)

`psy_reg.plugin.`**`validate_stringlist`**(*s*)

Validate a list of strings

Parameters **`val`** (*`iterable of strings`*) –

Returns list of str

Return type list

Raises `ValueError`

psy_reg.version module

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index