home | search | examples | gallery | docs » The Matplotlib API » previous | next | modules | index

matplotlib pyplot

matplotlib.pyplot

Provides a MATLAB-like plotting framework.

pylab combines pyplot with numpy into a single namespace. This is convenient for interactive work, but for programming it is recommended that the namespaces be kept separate, e.g.:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 5, 0.1);
y = np.sin(x)
plt.plot(x, y)
```

matplotlib.pyplot.**acorr**(*x*, *hold=None*, ***kwargs*)

call signature:

```
acorr(x, normed=True, detrend=mlab.detrend_none,
usevlines=True,
      maxlags=10, **kwargs)
```

Plot the autocorrelation of *x*. If *normed* = *True*, normalize the data by the autocorrelation at 0-th lag. *x* is detrended by the *detrend* callable (default no normalization).

Data are plotted as `plot(lags, c, **kwargs)`

Return value is a tuple (*lags*, *c*, *line*) where:

- *lags* are a length 2*maxlags+1 lag vector
- *c* is the 2*maxlags+1 auto correlation vector
- *line* is a `Line2D` instance returned by `plot()`

The default *linestyle* is None and the default *marker* is `'o'`, though these can be overridden with keyword args. The cross correlation is performed with `numpy.correlate()` with *mode* = 2.

If *usevlines* is *True*, `vlines()` rather than `plot()` is used to draw vertical lines from the origin to the acorr. Otherwise, the plot style is determined by the kwargs, which are `Line2D` properties.

maxlags is a positive integer detailing the number of lags to show.

The default value of *None* will return all $2imes\mathrm{len}(x) - 1$ lags.

The return value is a tuple (*lags*, *c*, *linecol*, *b*) where

- *linecol* is the `LineCollection`
- *b* is the *x*-axis.

See also

`plot()` or `vlines()` For documentation on valid kwargs.

Example:

`xcorr()` above, and `acorr()` below.

Example:

[source code, hires.png, pdf]

Table Of Contents

Previous topic

matplotlib path

Next topic

matplotlib nxutils

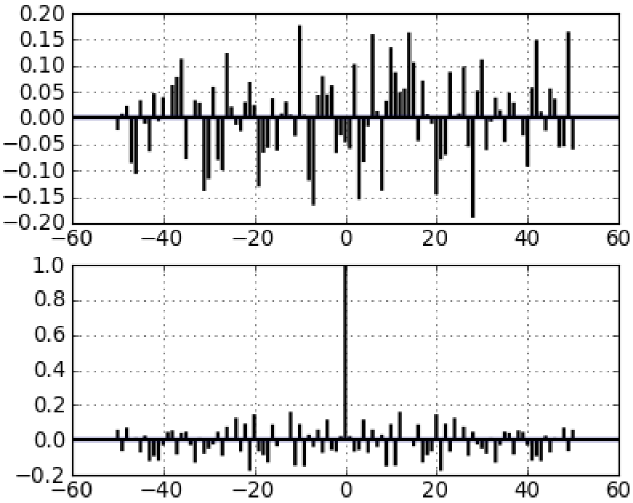
This Page

Show Source

Quick search

Go

Enter search terms or a module, class or function name.



Additional kwargs: hold = [True|False] overrides default hold state

matplotlib.pyplot.**annotate**(**args*, ***kwargs*)

call signature:

```
annotate(s, xy, xytext=None, xycoords='data',
         textcoords='data', arrowprops=None, **kwargs)
```

Keyword arguments:

Annotate the *x*, *y* point *xy* with text *s* at *x*, *y* location *xytext*. (If *xytext* = *None*, defaults to *xy*, and if *textcoords* = *None*, defaults to *xycoords*).

arrowprops, if not *None*, is a dictionary of line properties (see `matplotlib.lines.Line2D`) for the arrow that connects annotation to the point.

If the dictionary has a key *arrowstyle*, a FancyArrowPatch instance is created with the given dictionary and is drawn. Otherwise, a YAArow patch instance is created and drawn. Valid keys for YAArow are

Key	Description
width	the width of the arrow in points
frac	the fraction of the arrow length occupied by the head
headwidth	the width of the base of the arrow head in points
shrink	oftentimes it is convenient to have the arrowtip and base a bit away from the text and point being annotated. If *d* is the distance between the text and annotated point, shrink will shorten the arrow so the tip and base are shink percent of the distance *d* away from the endpoints. ie, `shrink=0.05 is 5%`
?	any key for `matplotlib.patches.polygon`

Valid keys for FancyArrowPatch are

Key	Description
arrowstyle	the arrow style

connectionstyle	the connection style
relpos	default is (0.5, 0.5)
patchA	default is bounding box of the text
patchB	default is None
shrinkA	default is 2 points
shrinkB	default is 2 points
mutation_scale	default is text size (in points)
mutation_aspect	default is 1.
?	any key for `matplotlib.patches.PathPatch`

xycoords and *textcoords* are strings that indicate the coordinates of *xy* and *xytext*.

Property	Description
'figure points'	points from the lower left corner of the figure
'figure pixels'	pixels from the lower left corner of the figure
'figure fraction'	0,0 is lower left of figure and 1,1 is upper, right
'axes points'	points from lower left corner of axes
'axes pixels'	pixels from lower left corner of axes
'axes fraction'	0,1 is lower left of axes and 1,1 is upper right
'data'	use the coordinate system of the object being annotated (default)
'offset points'	Specify an offset (in points) from the *xy* value
'polar'	you can specify *theta*, *r* for the annotation, even in cartesian plots. Note that if you are using a polar axes, you do not need to specify polar for the coordinate system since that is the native "data" coordinate system.

If a 'points' or 'pixels' option is specified, values will be added to the bottom-left and if negative, values will be subtracted from the top-right. Eg:

```
# 10 points to the right of the left border of the axes and
# 5 points below the top border
xy=(10,-5), xycoords='axes points'
```

You may use an instance of `Transform` or `Artist`. See *Annotating Axes* for more details.

The *annotation_clip* attribute contols the visibility of the annotation when it goes outside the axes area. If True, the annotation will only be drawn when the *xy* is inside the axes. If False, the annotation will always be drawn regardless of its position. The default is *None*, which behave as True only if *xycoords* is"data".

Additional kwargs are Text properties:

Property	Description
`agg_filter`	unknown

alpha	float (0.0 transparent through 1.0 opaque)
animated	[True | False]
axes	an Axes instance
backgroundcolor	any matplotlib color
bbox	rectangle prop dict
clip_box	a matplotlib.transforms.Bbox instance
clip_on	[True | False]
clip_path	[(Path, Transform) | Patch | None]
color	any matplotlib color
contains	a callable function
family or fontfamily or fontname or name	[FONTNAME | ‘serif’ | ‘sans-serif’ | ‘cursive’ | ‘fantasy’ | ‘monospace’]
figure	a matplotlib.figure.Figure instance
fontproperties or font_properties	a matplotlib.font_manager.FontProperties instance
gid	an id string
horizontalalignment or ha	[‘center’ | ‘right’ | ‘left’]
label	any string
linespacing	float (multiple of font size)
lod	[True | False]
multialignment	[‘left’ | ‘right’ | ‘center’]
path_effects	unknown
picker	[None|float|boolean|callable]
position	(x,y)
rasterized	[True | False | None]
rotation	[angle in degrees | ‘vertical’ | ‘horizontal’]
rotation_mode	unknown
size or fontsize	[size in points | ‘xx-small’ | ‘x-small’ | ‘small’ | ‘medium’ | ‘large’ | ‘x-large’ | ‘xx-large’]
snap	unknown
stretch or fontstretch	[a numeric value in range 0-1000 | ‘ultra-condensed’ | ‘extra-condensed’ | ‘condensed’ | ‘semi-condensed’ | ‘normal’ | ‘semi-expanded’ | ‘expanded’ | ‘extra-expanded’ | ‘ultra-expanded’]
style or fontstyle	[‘normal’ | ‘italic’ | ‘oblique’]
text	string or anything printable with ‘%s’ conversion.
transform	Transform instance
url	a url string
variant or fontvariant	[‘normal’ | ‘small-caps’]
verticalalignment or va or ma	[‘center’ | ‘top’ | ‘bottom’ | ‘baseline’]