plot_utils

Release v0.6.14

Jian Shi

May 01, 2023

CONTENTS

Welcome! This is a Python module that contains some useful data visualization functions.

CHAPTER

ONE

INSTALLATION

Recommended method (in the terminal or command window, execute the following command):

```
pip install git+https://github.com/jsh9/python-plot-utils@v0.6.14
```

For other installation alternatives, see the installation guide.

CHAPTER

TWO

DEPENDENCIES

- Python 2.7 or 3.5+
- matplotlib 1.5.0+, or 2.0.0+ (Version 2.1.0+ is strongly recommended.)
- numpy: 1.11.0+
- scipy: 0.19.0+
- pandas: 0.20.0+
- cycler: 0.10.0+
- matplotlib/basemap: 1.0.7 (only if you want to plot the two choropleth maps)
- PIL (only if you want to use the `trim_img()` function)

CHAPTER

THREE

API DOCUMENTATION

1. Visualizing one column of data

3.1 Pie chart

plot_utils.**piechart**(*target_array*, *class_names=None*, *dropna=False*, *top_n=None*, *sort_by='counts'*, *fig=None*, *ax=None*, *figsize=(3, 3)*, *dpi=100*, *colors=None*, *display='percent'*, *title=None*, *fontsize=None*, *verbose=True*, ***piechart_kwargs*)

Plot a pie chart demonstrating proportions of different categories within an array.

Parameters

- **target_array** (*array_like*) – An array containing categorical values (could have more than two categories). Target value can be numeric or texts.
- **class_names** (*sequence of str*) – Names of different classes. The order should correspond to that in the target_array. For example, if target_array has 0 and 1 then class_names should be ['0', '1']; and if target_array has "pos" and "neg", then class_names should be ['neg','pos'] (i.e., alphabetical). If None, values of the categories will be used as names. If [], then no class names are displayed.
- **dropna** (*bool*) – Whether to drop NaN values or not. If `False`, they show up as 'N/A'.
- **top_n** (*int*) – An integer between 1 and the number of unique categories in `target_array`. Useful for preventing plotting too many unique categories (very slow). If None, plot all categories.
- **sort_by** (*{'counts', 'name'}*) – An option to control whether the pie slices are arranged by the counts of each unique categories, or by the names of those categories.
- **fig** (matplotlib.figure.Figure or `None`) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or `None`) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of `fig` (if not `None`) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of `fig` (if not `None`) will override this parameter.
- **colors** (list or `None`) – A list of colors (can be RGB values, hex strings, or color names) to be used for each class. The length can be longer or shorter than the number of classes. If longer, only the first few colors are used; if shorter, colors are wrapped around. If `None`, automatically use the Pastel2 color map (8 colors total).
- **display** ({'percent', 'count', 'both', `None`}) – An option of what to show on top of each pie slices: percentage of each class, or count of each class, or both percentage

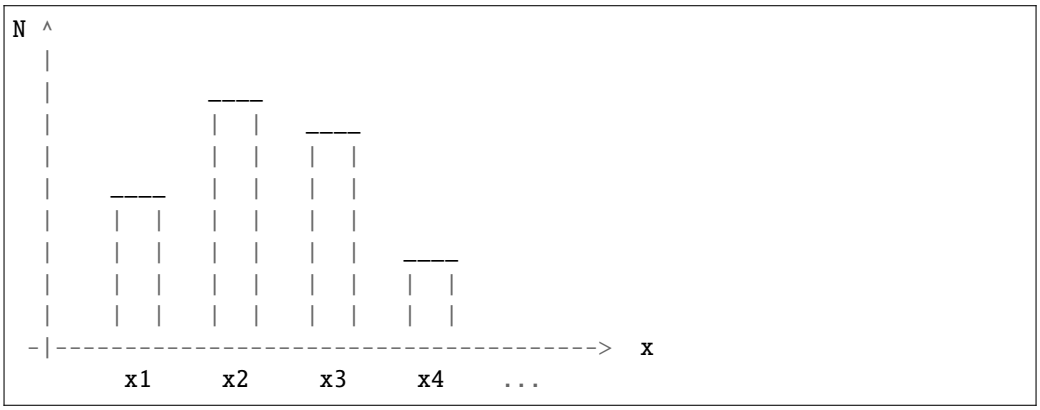
and count, or nothing.
- **title** (str or None) – The text to be shown on the top of the pie chart.
- **fontsize** (*scalar or tuple/list of two scalars*) – Font size. If scalar, both the class names and the percentages are set to the specified size. If tuple of two scalars, the first value sets the font size of class names, and the last value sets the font size of the percentages.
- **verbose** (*bool*) – Whether or to show a "Plotting more than 100 slices; please be patient" message when the number of categories exceeds 100.
- ****piechart_kwargs** – Keyword arguments to be passed to matplotlib.pyplot.pie function, except for "colors", "labels"and "autopct" because this subroutine redefines these three arguments. (See https://matplotlib.org/api/_as_gen/matplotlib.pyplot.pie.html)

Returns
- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

3.2 Discrete histogram

plot_utils.**discrete_histogram**(*x*, *fig=None*, *ax=None*, *figsize=(5, 3)*, *dpi=100*, *color=None*, *alpha=None*, *rot=0*, *logy=False*, *title=None*, *xlabel=None*, *ylabel='Number of occurrences'*, *show_xticklabel=True*)

Plot a discrete histogram based on the given data **x**, such as below:



In the figure, N is the number of occurences for x1, x2, x3, x4, etc. And x1, x2, x3, x4, etc. are the discrete values within **x**.

Parameters
- **x** (*list, numpy.ndarray, pandas.Series, or dict*) – Data to be visualized. If **x** is a list, numpy arrary, or pandas Series, the content of **x** is analyzed and counts of **x**'s values are plotted. If **x** is a dict, then **x**'s keys are treated as discrete values and **x**'s values are treated as counts.
- **fig** (matplotlib.figure.Figure or None) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or None) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of fig (if not None) will override this parameter.

- **dpi** (*float*) – Figure resolution. The dpi of `fig` (if not `None`) will override this parameter.
- **color** (str, list<float>, or `None`) – Color of bar. If `None`, the default color (muted blue) is used.
- **alpha** (float or `None`) – Opacity of bar. If `None`, the default value (1.0) is used.
- **rot** (*float or int*) – Rotation angle (degrees) of X axis label. Default = 0 (up-right label).
- **logy** (*bool*) – Whether or not to use log scale for the Y axis.
- **title** (*str*) – The title of the plot.
- **xlabel** (*str*) – The X axis label.
- **ylabel** (*str*) – The Y axis label.
- **show_xticklabel** (*bool*) – Whether or not to show the X tick labels (the names of the classes).

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.
- **value_count** (*pandas.Series*) – The counts of each discrete values within `x` (if `x` is an array) with each values sorted in ascending order, or the pandas Series generated from `x` (if `x` is a dict).

Notes

References:

http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.plot.html http://pandas.pydata.org/pandas-docs/version/0.18.1/visualization.html#bar-plots

See also:

plot_ranking
Plot bars showing the ranking of the data

2. Visualizing two columns of data

3.3 Bin and mean

`plot_utils.`**bin_and_mean**(*xdata*, *ydata*, *bins=10*, *distribution='normal'*, *show_fig=True*, *fig=None*, *ax=None*, *figsize=None*, *dpi=100*, *show_bins=True*, *raw_data_label='raw data'*, *mean_data_label='average'*, *xlabel=None*, *ylabel=None*, *logx=False*, *logy=False*, *grid_on=True*, *error_bounds=True*, *err_bound_type='shade'*, *legend_on=True*, *subsamp_thres=None*, *show_stats=True*, *show_SE=False*, *err_bound_shade_opacity=0.5*)

Calculate the "bin-and-mean" results and optionally show the "bin-and-mean" plot.

A "bin-and-mean" plot is a more salient way to show the dependency of `ydata` on `xdata`. The data points (`xdata`, `ydata`) are divided into different bins according to the values in `xdata` (via `bins`), and within each bin, the mean values of x and y are calculated, and treated as the representative x and y values.

"Bin-and-mean" is preferred when data points are highly skewed (e.g., a lot of data points for when x is small, but very few for large x). The data points when x is large are usually not noises,

and could be even more valuable (think of the case where x is earthquake magnitude and y is the related economic loss). If we want to study the relationship between economic loss and earthquake magnitude, we need to bin-and-mean raw data and draw conclusions from the mean data points.

The theory that enables this method is the assumption that the data points with similar x values follow the same distribution. Naively, we assume the data points are normally distributed, then y_mean is the arithmetic mean of the data points within a bin. We also often assume the data points follow log-normal distribution (if we want to assert that y values are all positive), then y_mean is the expected value of the log-normal distribution, while x_mean for any bins are still just the arithmetic mean.

Notes

(a) **For log-normal distribution, the expective value of y is:**
E(Y) = exp(mu + (1/2)*sigma^2)
and the variance is:
Var(Y) = [exp(sigma^2) - 1] * exp(2*mu + sigma^2)

where mu and sigma are the two parameters of the distribution.

(b) Knowing E(Y) and Var(Y), mu and sigma can be back-calculated:

```
                    ___________________
mu = ln[ E(Y) / V 1 + Var(Y)/E^2(Y)  ]


          ___________________________
sigma = V ln[ 1 + Var(Y)/E^2(Y) ]
```

(Reference: https://en.wikipedia.org/wiki/Log-normal_distribution)

Parameters

- **xdata** (*list, numpy.ndarray, or pandas.Series*) – X data.
- **ydata** (*list, numpy.ndarray, or pandas.Series*) – Y data.
- **bins** (*int, list, numpy.ndarray, or pandas.Series*) – Number of bins (an integer), or an array representing the actual bin edges. If `bins` means bin edges, the edges are inclusive on the lower bound, e.g., a value 2 shall fall into the bin [2, 3), but not the bin [1, 2). Note that the binning is done according to the X values.
- **distribution** (*{'normal', 'lognormal'}*) – Specifies which distribution the Y values within a bin follow. Use 'lognormal' if you want to assert all positive Y values. Only supports normal and log-normal distributions at this time.
- **show_fig** (*bool*) – Whether or not to show a bin-and-mean plot.
- **fig** (matplotlib.figure.Figure or `None`) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or `None`) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of `fig` (if not `None`) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of `fig` (if not `None`) will override this parameter.
- **show_bins** (*bool*) – Whether or not to show the bin edges as vertical lines on the plots.
- **raw_data_label** (*str*) – The label name of the raw data to be shown in the legend (such as "raw data"). It has no effects if `show_legend` is `False`.
- **mean_data_label** (*str*) – The label name of the mean data to be shown in the legend (such as "averaged data"). It has no effects if `show_legend` is `False`.

- **xlabel** (str or None) – X axis label. If None and xdata is a pandas Series, use xdata's "name" attribute as xlabel.
- **ylabel** (str of None) – Y axis label. If None and ydata is a pandas Series, use ydata's "name" attribute as ylabel.
- **logx** (*bool*) – Whether or not to show the X axis in log scale.
- **logy** (*bool*) – Whether or not to show the Y axis in log scale.
- **grid_on** (*bool*) – Whether or not to show grids on the plot.
- **error_bounds** (*bool*) – Whether or not to show error bounds of each bin.
- **err_bound_type** (*{'shade', 'bar'}*) – Type of error bound: shaded area or error bars. It has no effects if error_bounds is set to False.
- **legend_on** (*bool*) – Whether or not to show a legend.
- **subsamp_thres** (*int*) – A positive integer that defines the number of data points in each bin to show in the scatter plot. The smaller this number, the faster the plotting process. If larger than the number of data points in a bin, then all data points from that bin are plotted. If None, then all data points from all bins are plotted.
- **show_stats** (*bool*) – Whether or not to show R^2 scores, correlation coefficients of the raw data and the binned averages on the plot.
- **show_SE** (*bool*) – If True, show the standard error of y_mean (orange dots) of each bin as the shaded area beneath the mean value lines. If False, show the standard deviation of raw Y values (gray dots) within each bin.
- **err_bound_shade_opacity** (*float*) – The opacity of the shaded area representing the error bound. 0 means completely transparent, and 1 means completely opaque. It has no effect if error_bound_type is 'bar'.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function. None, if show_fig is set to False.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function. None, if show_fig is set to False.
- **x_mean** (*numpy.ndarray*) – Mean X values of each data bin (in terms of X values).
- **y_mean** (*numpy.ndarray*) – Mean Y values of each data bin (in terms of X values).
- **y_std** (*numpy.ndarray*) – Standard deviation of Y values or each data bin (in terms of X values).
- **y_SE** (*numpy.ndarray*) – Standard error of y_mean. It describes how far y_mean is from the population mean (or the "true mean value") within each bin, which is a different concept from y_std. See https://en.wikipedia.org/wiki/Standard_error#Standard_error_of_mean_versus_standard_deviation for further information.
- **stats_** (*tuple<float>*) – A tuple in the order of (r2_score_raw, corr_coeff_raw, r2_score_binned, corr_coeff_binned), which are the R^2 score and correlation coefficient of the raw data (xdata and ydata) and the binned averages (x_mean and y_mean).

3.4 Category means

plot_utils.**category_means**(*categorical_array*, *continuous_array*, *fig=None*, *ax=None*, *figsize=None*, *dpi=100*, *title=None*, *xlabel=None*, *ylabel=None*, *rot=0*, *dropna=False*, *show_stats=True*, *sort_by='name'*, *vert=True*, *plot_violins=True*, ***extra_kwargs*)

Summarize the mean values of entries of continuous_array corresponding to each distinct category in categorical_array, and show a violin plot to visualize it. The violin plot will show the distribution of values in continuous_array corresponding to each category in categorical_array.

Also, a one-way ANOVA test (H0: different categories in `categorical_array` yield the same average values in `continuous_array`) is performed, and F statistics and p-value are returned.

Parameters

- **categorical_array** (*list, numpy.ndarray, or pandas.Series*) – An vector of categorical values.
- **continuous_array** (*list, numpy.ndarray, or pandas.Series*) – The target variable whose values correspond to the values in x. Must have the same length as x. It is natural that y contains continuous values, but if y contains categorical values (expressed as integers, not strings), this function should also work.
- **fig** (matplotlib.figure.Figure or `None`) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or `None`) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of `fig` (if not `None`) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of `fig` (if not `None`) will override this parameter.
- **title** (*str*) – The title of the violin plot, usually the name of ``categorical_array`
- **xlabel** (*str*) – The label for the x axis (i.e., categories) of the violin plot. If `None` and `categorical_array` is a pandas Series, use the 'name' attribute of `categorical_array` as xlabel.
- **ylabel** (*str*) – The label for the y axis (i.e., average `continuous_array` values) of the violin plot. If `None` and `continuous_array` is a pandas Series, use the 'name' attribute of `continuous_array` as ylabel.
- **rot** (*float*) – The rotation (in degrees) of the x axis labels.
- **dropna** (*bool*) – Whether or not to exclude N/A records in the data.
- **show_stats** (*bool*) – Whether or not to show the statistical test results (F statistics and p-value) on the figure.
- **sort_by** (*{'name', 'mean', 'median', None}*) – Option to arrange the different categories in *categorical_array* in the violin plot. `None` means no sorting, i.e., using the hashed order of the category names; 'mean' and 'median' mean sorting the violins according to the mean/median values of each category; 'name' means sorting the violins according to the category names.
- **vert** (*bool*) – Whether to show the violins as vertical.
- **plot_violins** (*bool*) – If `True`, use violin plots to illustrate the distribution of groups. Otherwise, use multi-histogram (hist_multi()).
- ****extra_kwargs** – Keyword arguments to be passed to plt.violinplot() or hist_multi(). (https://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.violinplot.html) Note that this subroutine overrides the default behavior of violinplot: showmeans is overriden to True and showextrema to False.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.
- **mean_values** (*dict*) – A dictionary whose keys are the categories in x, and their corresponding values are the mean values in y.
- **F_test_result** (*tuple<float>*) – A tuple in the order of (F_stat, p_value), where F_stat is the computed F-value of the one-way ANOVA test, and p_value is the associated p-value from the F-distribution.

3.5 Positive rate

plot_utils.**positive_rate**(*categorical_array*, *two_classes_array*, *fig=None*, *ax=None*, *figsize=None*, *dpi=100*, *barh=True*, *top_n=None*, *dropna=False*, *xlabel=None*, *ylabel=None*, *show_stats=True*)

Calculate the proportions of the different categories in `categorical_array` that fall into class "1" (or `True`) in `two_classes_array`, and optionally show a figure.

Also, a Pearson's chi-squared test is performed to test the independence between `categorical_array` and `two_classes_array`. The chi-squared statistics, p-value, and degree-of-freedom are returned.

Parameters

- **categorical_array** (*list, numpy.ndarray, or pandas.Series*) – An array of categorical values.
- **two_class_array** (*list, numpy.ndarray, or pandas.Series*) – The target variable containing two classes. Each value in this parameter correspond to a value in `categorical_array` (at the same index). It must have the same length as `categorical_array`. The second unique value in this parameter will be considered as the positive class (for example, "True" in [True, False, True], or "3" in [1, 1, 3, 3, 1]).
- **fig** (matplotlib.figure.Figure or `None`) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or `None`) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of `fig` (if not `None`) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of `fig` (if not `None`) will override this parameter.
- **barh** (*bool*) – Whether or not to show the bars as horizontal (otherwise, vertical).
- **top_n** (*int*) – Only shows `top_n` categories (ranked by their positive rate) in the figure. Useful when there are too many categories. If `None`, show all categories.
- **dropna** (*bool*) – If `True`, ignore entries (in both arrays) where there are missing values in at least one array. If `False`, the missing values are treated as a new category: "N/A".
- **xlabel** (*str*) – X axes label.
- **ylabel** (*str*) – Y axes label.
- **show_stats** (*bool*) – Whether or not to show the statistical test results (chi2 statistics and p-value) on the figure.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.
- **pos_rate** (*pandas.Series*) – The positive rate of each categories in x
- **chi2_results** (*tuple<float>*) – A tuple in the order of (chi2, p_value, degree_of_freedom)

3.6 Contingency table

plot_utils.**contingency_table**(*array_horizontal*, *array_vertical*, *fig=None*, *ax=None*, *figsize='auto'*, *dpi=100*, *color_map='auto'*, *xlabel=None*, *ylabel=None*, *dropna=False*, *rot=45*, *normalize=True*, *symm_cbar=True*, *show_stats=True*)

Calculate and visualize the contingency table from two categorical arrays. Also perform a Pearson's chi-squared test to evaluate whether the two arrays are independent.

Parameters

- **array_horizontal** (*list, numpy.ndarray, or pandas.Series*) – Array to show as the horizontal margin in the contigency table (i.e., its categories are the column headers).
- **array_vertical** (*list, numpy.ndarray, or pandas.Series*) – Array to show as the vertical margin in the contigency table (i.e., its categories are the row names).
- **fig** (matplotlib.figure.Figure or None) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or None) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of fig (if not None) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of fig (if not None) will override this parameter.
- **color_map** (*str or matplotlib.colors.Colormap*) – The color scheme specifications. Valid names are listed in https://matplotlib.org/users/colormaps.html. If relative_color is True, use diverging color maps (e.g., PiYG, PRGn, BrBG, PuOr, RdGy, RdBu, RdYlBu, RdYlGn, Spectral, coolwarm, bwr, seismic). Otherwise, use sequential color maps (e.g., viridis, jet).
- **xlabel** (*str*) – The label for the horizontal axis. If None and array_horizontal is a pandas Series, use the 'name' attribute of array_horizontal as xlabel.
- **ylabel** (*str*) – The label for the vertical axis. If None and array_vertical is a pandas Series, use the 'name' attribute of array_vertical as ylabel.
- **dropna** (*bool*) – If True, ignore entries (in both arrays) where there are missing values in at least one array. If False, the missing values are treated as a new category: "N/A".
- **rot** (*float or 'vertical' or 'horizontal'*) – The rotation of the x axis labels (in degrees).
- **normalize** (*bool*) – If True, plot the contingency table as the relative difference between the observed and the expected (i.e., (obs. - exp.)/exp.). If False, plot the original "observed frequency".
- **symm_cbar** (*bool*) – If True, the limits of the color bar are symmetric. Otherwise, the limits are the natural minimum/maximum of the table to be plotted. It has no effect if "normalize" is set to False.
- **show_stats** (*bool*) – Whether or not to show the statistical test results (chi2 statistics and p-value) on the figure.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.
- **chi2_results** (*tuple<float>*) – A tuple in the order of (chi2, p_value, degree_of_freedom).
- **correlation_metrics** (*tuple<float>*) – A tuple in the order of (phi coef., coeff. of

contingency, Cramer's V).

3.7 Scatter plots of two columns

plot_utils.**scatter_plot_two_cols**(*X*, *two_columns*, *fig=None*, *ax=None*, *figsize=(3, 3)*, *dpi=100*, *alpha=0.5*, *color=None*, *grid_on=True*, *logx=False*, *logy=False*)

Produce scatter plots of two of the columns in `X` (the data matrix). The correlation between the two columns are shown on top of the plot.

Parameters

- **X** (*pandas.DataFrame*) – The dataset. Currently only supports pandas dataframe.
- **two_columns** (*[str, str] or [int, int]*) – The names or indices of the two columns within `X`. Must be a list of length 2. The elements must either be both integers, or both strings.
- **fig** (matplotlib.figure.Figure or `None`) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or `None`) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of `fig` (if not `None`) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of `fig` (if not `None`) will override this parameter.
- **alpha** (*float*) – Opacity of the scatter points.
- **color** (str, list<float>, tuple<float>, or `None`) – Color of the scatter points. If `None`, default matplotlib color palette will be used.
- **grid_on** (*bool*) – Whether or not to show grids on the plot.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

3. Visualizing multiple columns of data

3.8 3D histograms

plot_utils.**histogram3d**(*X*, *bins=10*, *fig=None*, *ax=None*, *figsize=(8, 4)*, *dpi=100*, *elev=30*, *azim=5*, *alpha=0.6*, *data_labels=None*, *plot_legend=True*, *plot_xlabel=False*, *color=None*, *dx_factor=0.4*, *dy_factor=0.8*, *ylabel='Data'*, *zlabel='Counts'*, ***legend_kwargs*)

Plot 3D histograms. 3D histograms are best used to compare the distribution of more than one set of data.

Parameters

- **X** (*numpy.ndarray, list<list<float>>, pandas.Series, pandas.DataFrame*) –

  Input data. `X` can be:

  (a) a 2D numpy array, where each row is one data set;
  (b) a 1D numpy array, containing only one set of data;
  (c) a list of lists, e.g., [[1,2,3],[2,3,4,5],[2,4]], where each element corresponds to a data set (can have different lengths);

(d) a list of 1D numpy arrays. [Note: Robustness is not guaranteed for X being a list of
2D numpy arrays.]
(5) a pandas Series, which is treated as a 1D numpy array; (5) a pandas DataFrame, where each column is one data set.

- **bins** (*int, list, numpy.ndarray, or pandas.Series*) –
Bin specifications. Can be:
(a) An integer, which indicates number of bins;
(b) An array or list, which specifies bin edges. [Note: If an integer is used, the widths of bars across data
sets may be different. Thus array/list is recommended.]
- **fig** (matplotlib.figure.Figure or `None`) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or `None`) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of `fig` (if not `None`) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of `fig` (if not `None`) will override this parameter.
- **elev** (*float*) – Elevation of the 3D view point.
- **azim** (*float*) – Azimuth angle of the 3D view point (unit: degree).
- **alpha** (*float*) – Opacity of bars
- **data_labels** (*list of str*) – Names of different datasets, e.g., ['Simulation', 'Measurement']. If not provided, generic names ['Dataset #1', 'Dataset #2', ...] are used. The data_labels are only shown when either plot_legend or plot_xlabel is `True`. If not provided, and X is a pandas DataFrame/Series, data_labels will be overridden by the column names (or name) of `X`.
- **plot_legend** (*bool*) – Whether to show legends or not.
- **plot_xlabel** (*str*) – Whether to show data_labels of each data set on their respective x axis position or not.
- **color** (*list<list>, or tuple<tuples>*) – Colors of each distributions. Needs to be at least the same length as the number of data series in `X`. Can be RGB colors, HEX colors, or valid color names in Python. If `None`, get_colors(N=N, color_scheme='tab10') will be queried.
- **dx_factor** (*float*) – Width factor of 3D bars in x direction.
- **dy_factor** (*float*) – Width factor of 3D bars in y direction. For example, if `dy_factor` is 0.9, there will be a small gap between bars in y direction.
- **ylabel** (*str*) – Label of Y axes.
- **zlabel** (*str*) – Labels of Z axes.

Returns

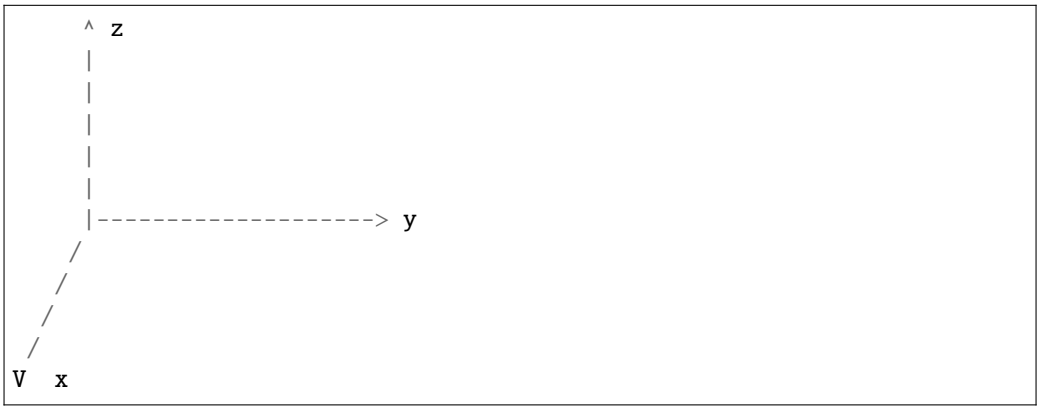
- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

Notes

x direction :
Across data sets (i.e., if we have three datasets, the bars will occupy three different x values).

y direction :
Within dataset.

Illustration:



3.9 Multiple histograms

plot_utils.**hist_multi**(*X*, *bins=10*, *fig=None*, *ax=None*, *figsize=None*, *dpi=100*, *nan_warning=False*, *showmeans=True*, *showmedians=False*, *vert=True*, *data_names=[]*, *rot=45*, *name_ax_label=None*, *data_ax_label=None*, *sort_by=None*, *title=None*, *show_vals=True*, *show_pct_diff=False*, *baseline_data_index=0*, *legend_loc='best'*, *show_counts_on_data_ax=True*, ***extra_kwargs*)

Generate multiple histograms, one for each data set within `X`.

Parameters

- **X** (*pandas.DataFrame, pandas.Series, numpy.ndarray, or dict*) – The data to be visualized. It can be of the following types:
  - **pandas.DataFrame:**
    - Each column contains a set of data
  - **pandas.Series:**
    - Contains only one set of data
  - **numpy.ndarray:**
    - 1D numpy array: only one set of data
    - 2D numpy array: each column contains a set of data
    - Higher dimensional numpy array: not allowed
  - **dict:**
    - Each key-value pair is one set of data

  - **list of lists:**
    * Each sub-list is a data set

  Note that the NaN values in the data are implicitly excluded.

- **bins** (*int or sequence or str*) – If an integer is given, the whole range of data (i.e., all the numbers within `X`) is divided into `bins` segments. If sequence or str, they will be passed to the `bins` argument of `matplotlib.pyplot.hist()`.
- **fig** (matplotlib.figure.Figure or `None`) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or `None`) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of `fig` (if not `None`) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of `fig` (if not `None`) will override this parameter.
- **nan_warning** (*bool*) – Whether to show a warning if there are NaN values in the data.
- **showmeans** (*bool*) – Whether to show the mean values of each data group.
- **showmedians** (*bool*) – Whether to show the median values of each data group.
- **vert** (*bool*) – Whether to show the "base" of the histograms as vertical.
- **data_names** (list<str>, `[]`, or `None`) – The names of each data set, to be shown as the axis tick label of each data set. If `[]` or `None`, it will be determined automatically. If `X` is a:
  - **numpy.ndarray:**
    * data_names = ['data_0', 'data_1', 'data_2', ...]
  - **pandas.Series:**
    * data_names = X.name
  - **pd.DataFrame:**
    * data_names = list(X.columns)
  - **dict:**
    * data_names = list(X.keys())
- **rot** (*float*) – The rotation (in degrees) of the data_names when shown as the tick labels. If vert is False, rot has no effect.
- **name_ax_label** (*str*) – The label of the "name axis". ("Name axis" is the axis along which different violins are presented.)
- **data_ax_label** (*str*) – The labels of the "data axis". ("Data axis" is the axis along which the data values are presented.)

- **sort_by** ({'name', 'mean', 'median', None}) – Option to sort the different data groups in X in the violin plot. None means no sorting, keeping the violin plot order as provided; 'mean' and 'median' mean sorting the violins according to the mean/median values of each data group; 'name' means sorting the violins according to the names of the groups.
- **title** (*str*) – The title of the plot.
- **show_vals** (*bool*) – Whether to show mean and/or median values along the mean/median bars. Only effective if showmeans and/or showmedians are turned on.
- **show_pct_diff** (*bool*) – Whether to show percent difference of mean and/or median values between different data sets. Only effective when show_vals is set to True.
- **baseline_data_index** (*int*) – Which data set is considered the "baseline" when showing percent differences.
- **legend_loc** (*str*) – The location specification for the legend.
- **show_counts_on_data_ax** (*bool*) – Whether to show counts besides the histograms.
- ****extra_kwargs** (*dict*) – Other keyword arguments to be passed to matplotlib.pyplot.bar().

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

3.10 Violin plot

plot_utils.**violin_plot**(*X*, *fig=None*, *ax=None*, *figsize=None*, *dpi=100*, *nan_warning=False*, *showmeans=True*, *showextrema=False*, *showmedians=False*, *vert=True*, *data_names=[]*, *rot=45*, *name_ax_label=None*, *data_ax_label=None*, *sort_by=None*, *title=None*, ***violinplot_kwargs*)

Generate violin plots for each data set within X.

Parameters

- **X** (*pandas.DataFrame, pandas.Series, numpy.ndarray, or dict*) – The data to be visualized. It can be of the following types:
  - **pandas.DataFrame:**
    - Each column contains a set of data
  - **pandas.Series:**
    - Contains only one set of data
  - **numpy.ndarray:**
    - 1D numpy array: only one set of data

        * 2D numpy array: each column contains a set of data
        * Higher dimensional numpy array: not allowed
      - **dict:**
        * Each key-value pair is one set of data
      - **list of lists:**
        * Each sub-list is a data set

    Note that the NaN values in the data are implicitly excluded.

- **fig** (matplotlib.figure.Figure or None) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or None) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of fig (if not None) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of fig (if not None) will override this parameter.
- **nan_warning** (*bool*) – Whether to show a warning if there are NaN values in the data.
- **showmeans** (*bool*) – Whether to show the mean values of each data group.
- **showextrema** (*bool*) – Whether to show the extrema of each data group.
- **showmedians** (*bool*) – Whether to show the median values of each data group.
- **vert** (*bool*) – Whether to show the violins as vertical.
- **data_names** (list<str>, [], or None) – The names of each data set, to be shown as the axis tick label of each data set. If [] or None, it will be determined automatically. If X is a:
  - **numpy.ndarray:**
    * data_names = [‘data_0’, ‘data_1’, ‘data_2’, . . .]
  - **pandas.Series:**
    * data_names = X.name
  - **pd.DataFrame:**
    * data_names = list(X.columns)
  - **dict:**
    * data_names = list(X.keys())
- **rot** (*float*) – The rotation (in degrees) of the data_names when shown as the tick labels. If vert is False, rot has no effect.
- **name_ax_label** (*str*) – The label of the “name axis”. (“Name axis” is the axis along which different violins are presented.)

- **data_ax_label** (*str*) – The labels of the "data axis". ("Data axis" is the axis along which the data values are presented.)
- **sort_by** ({'name', 'mean', 'median', None}) – Option to sort the different data groups in X in the violin plot. None means no sorting, keeping the violin plot order as provided; 'mean' and 'median' mean sorting the violins according to the mean/median values of each data group; 'name' means sorting the violins according to the names of the groups.
- **title** (*str*) – The title of the plot.
- ****violinplot_kwargs** (*dict*) – Other keyword arguments to be passed to matplotlib.pyplot.violinplot().

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

3.11 Correlation matrix

plot_utils.**correlation_matrix**(*X*, *color_map='RdBu_r'*, *fig=None*, *ax=None*, *figsize=None*, *dpi=100*, *variable_names=None*, *rot=45*, *scatter_plots=False*)

Plot correlation matrix of a dataset X, whose columns are different variables (or a sample of a certain random variable).

Parameters

- **X** (*numpy.ndarray or pandas.DataFrame*) – The data set.
- **color_map** (*str or matplotlib.colors.Colormap*) – The color scheme to show high, low, negative high correlations. Valid names are listed in https://matplotlib.org/users/colormaps.html. Using diverging color maps is recommended: PiYG, PRGn, BrBG, PuOr, RdGy, RdBu, RdYlBu, RdYlGn, Spectral, coolwarm, bwr, seismic.
- **fig** (matplotlib.figure.Figure or None) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or None) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of fig (if not None) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of fig (if not None) will override this parameter.
- **variable_names** (*list<str>*) – Names of the variables in X. If X is a pandas DataFrame, this argument is not needed: column names of X is automatically used as variable names. If X is a numpy array, and this argument is not provided, then X's column indices are used. The length of variable_names should match the number of columns in X; if not, a warning will be thrown (not error).

- **rot** (*float*) – The rotation of the x axis labels, in degrees.
- **scatter_plots** (*bool*) – Whether or not to show the scatter plots of pairs of variables.

Returns

- **correlations** (*pandas.DataFrame*) – The correlation matrix.
- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

3.12 Missing values

plot_utils.**missing_value_counts**(*X*, *fig=None*, *ax=None*, *figsize=None*, *dpi=100*, *rot=45*)

Visualize the number of missing values in each column of **X**.

Parameters

- **X** (*pandas.DataFrame or pandas.Series*) – Input data set whose every row is an observation and every column is a variable.
- **fig** (matplotlib.figure.Figure or None) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or None) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of fig (if not None) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of fig (if not None) will override this parameter.
- **rot** (*float*) – Rotation (in degrees) of the x axis labels.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.
- **null_counts** (*pandas.Series*) – A pandas Series whose every element is the number of missing values corresponding to each column of **X**.

4. Map plotting

3.13 Choropleth maps (state and county levels)

plot_utils.**choropleth_map_county**(*data_per_county*, *fig=None*, *ax=None*, *figsize=(10, 7)*, *dpi=100*, *vmin=None*, *vmax=None*, *unit=''*, *cmap='OrRd'*, *map_title='USA county map'*, *fontsize=14*, *cmap_midpoint=None*, *shapefile_dir=None*)

Generate a choropleth map of the US (including Alaska and Hawaii), on a county level.

According to Wikipedia, a choropleth map is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable being displayed on the map, such as population density or per-capita income.

Parameters

- **data_per_county** (*dict or pandas.Series or pandas.DataFrame*) – Numerical data of each county, to be plotted onto the map. Acceptable data types include:
  - **pandas Series: Index should be valid county identifiers (i.e.,** 5-digit county FIPS codes)
  - **pandas DataFrame: The dataframe can have only one column (with** the index being valid county identifiers), two columns (with one of the column named 'state', 'State', or 'FIPS_code', and containing county identifiers).
  - **dictionary: with keys being valid county identifiers, and values** being the numerical values to be visualized
- **fig** (matplotlib.figure.Figure or `None`) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or `None`) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of `fig` (if not `None`) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of `fig` (if not `None`) will override this parameter.
- **vmin** (*float*) – Minimum value to be shown on the map. If `vmin` is larger than the actual minimum value in the data, some of the data values will be "clipped". This is useful if there are extreme values in the data and you do not want those values to complete skew the color distribution.
- **vmax** (*float*) – Maximum value to be shown on the map. Similar to `vmin`.
- **map_title** (*str*) – Title of the map, to be shown on the top of the map.
- **unit** (*str*) – Unit of the numerical (for example, "population per km^2"), to be shown on the right side of the color bar.

- **cmap** (*str or <matplotlib.colors.Colormap>*) – Color map name. Suggested names: 'hot_r', 'summer_r', and 'RdYlBu' for plotting deviation maps.
- **fontsize** (*scalar*) – Font size of all the texts on the map.
- **cmap_midpoint** (*float*) – A numerical value that specifies the "deviation point". For example, if your data ranges from -200 to 1000, and you want negative values to appear blue-ish, and positive values to appear red-ish, then you can set `cmap_midpoint` to 0.0. If `None`, then the "deviation point" will be the median value of the data values.
- **shapefile_dir** (*str*) – Directory where shape files are stored. Shape files (state level and county level) should be organized as follows:

  `shapefile_dir`/usa_states/st99_d00.(...)
  `shapefile_dir`/usa_counties/cb_2016_us_county_500k.(...)

  If `None`, the shapefile directory within this library will be used.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

References

This function is based partly on an example in the Basemap repository (https://github.com/matplotlib/basemap/blob/master/examples/fillstates.py) as well as a modification on Stack Overflow (https://stackoverflow.com/questions/39742305).

`plot_utils.`**choropleth_map_state**(*data_per_state*, *fig=None*, *ax=None*, *figsize=(10, 7)*, *dpi=100*, *vmin=None*, *vmax=None*, *map_title='USA map'*, *unit=''*, *cmap='OrRd'*, *fontsize=14*, *cmap_midpoint=None*, *shapefile_dir=None*)

Generate a choropleth map of the US (including Alaska and Hawaii), on a state level.

According to Wikipedia, a choropleth map is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable being displayed on the map, such as population density or per-capita income.

Parameters

- **data_per_state** (*dict or pandas.Series or pandas.DataFrame*) – Numerical data of each state, to be plotted onto the map. Acceptable data types include:
  - **pandas Series: Index should be valid state identifiers (i.e.,** state full name, abbreviation, or FIPS code)
  - **pandas DataFrame: The dataframe can have only one column (with** the index being valid state identifiers), two columns (with one of the column named 'state', 'State', or 'FIPS_code', and containing state identifiers).
  - **dictionary: with keys being valid state identifiers, and values** being the numerical values to be visualized

- **fig** (matplotlib.figure.Figure or None) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or None) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of fig (if not None) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of fig (if not None) will override this parameter.
- **vmin** (*float*) – Minimum value to be shown on the map. If vmin is larger than the actual minimum value in the data, some of the data values will be "clipped". This is useful if there are extreme values in the data and you do not want those values to complete skew the color distribution.
- **vmax** (*float*) – Maximum value to be shown on the map. Similar to vmin.
- **map_title** (*str*) – Title of the map, to be shown on the top of the map.
- **unit** (*str*) – Unit of the numerical (for example, "population per km^2"), to be shown on the right side of the color bar.
- **cmap** (*str or matplotlib.colors.Colormap*) – Color map name. Suggested names: 'hot_r', 'summer_r', and 'RdYlBu' for plotting deviation maps.
- **fontsize** (*float*) – Font size of all the texts on the map.
- **cmap_midpoint** (*float*) – A numerical value that specifies the "deviation point". For example, if your data ranges from -200 to 1000, and you want negative values to appear blue-ish, and positive values to appear red-ish, then you can set cmap_midpoint to 0.0. If None, then the "deviation point" will be the median value of the data values.
- **shapefile_dir** (*str*) – Directory where shape files are stored. Shape files (state level and county level) should be organized as follows:

  shapefile_dir/usa_states/st99_d00.(...)
  shapefile_dir/usa_counties/cb_2016_us_county_500k.(...)

  If None, the shapefile directory within this library will be used.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

References

This function is based partly on an example in the Basemap repository (https://github.com/matplotlib/basemap/blob/master/examples/fillstates.py) as well as a modification on Stack Overflow (https://stackoverflow.com/questions/39742305).

5. Time series plotting

3.14 Plot time series

plot_utils.**plot_multiple_timeseries**(*multiple_time_series*, *show_legend=True*, *fig=None*, *ax=None*, *figsize=(10, 3)*, *dpi=100*, *ncol_legend=5*, ***kwargs*)

Plot multiple time series.

Note that setting keyword arguments such as `color` or `linestyle` will force all time series to have the same color or line style. So we recommend letting this function generate distinguishable line specifications (color/ linestyle/linewidth combinations) by itself. (Although the more time series, the less the distinguishability. 240 time series or less is recommended.)

Parameters

- **multiple_time_series** (*pandas.DataFrame or pandas.Series*) – If it is a pandas DataFrame, its index is the date, and each column is a different time series. If it is a pandas Series, it will be internally converted into a 1-column pandas DataFrame.
- **fig** (matplotlib.figure.Figure or `None`) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or `None`) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of `fig` (if not `None`) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of `fig` (if not `None`) will override this parameter.
- **ncol_legend** (*int*) – Number of columns of the legend.
- ****kwargs** – Other keyword arguments to be passed to *plot_timeseries()*, such as color, marker, fontsize, alpha, etc.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

See also:

plot_timeseries()
Plot a single set of time series.

plot_utils.**plot_timeseries**(*time_series*, *date_fmt=None*, *fig=None*, *ax=None*, *figsize=(10, 3)*, *dpi=100*, *xlabel='Time'*, *ylabel=None*, *label=None*, *color=None*, *lw=2*, *ls=None*, *marker=None*, *fontsize=12*, *xgrid_on=True*, *ygrid_on=True*, *title=None*, *zorder=None*, *alpha=1.0*, *month_grid_width=None*)

Plot time series (i.e., values a function of dates).

You can plot multiple time series by supplying a multi-column pandas Dataframe, but you cannot use custom line specifications (colors, width, and styles) for each time series. It is recommended to use *plot_multiple_timeseries()* in stead.

Parameters

- **time_series** (*pandas.Series or pandas.DataFrame*) – A pandas Series, with index being date; or a pandas DataFrame, with index being date, and each column being a different time series.
- **date_fmt** (*str*) – Date format specifier, e.g., '%Y-%m' or '%d/%m/%y'.
- **fig** (matplotlib.figure.Figure or None) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or None) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of fig (if not None) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of fig (if not None) will override this parameter.
- **xlabel** (*str*) – Label of X axis. Usually "Time" or "Date".
- **ylabel** (*str*) – Label of Y axis. Usually the meaning of the data, e.g., "Gas price [$]".
- **label** (*str*) – Label of data, for plotting legends.
- **color** (*list<float> or str*) – Color of line. If None, let Python decide for itself.
- **xgrid_on** (*bool*) – Whether or not to show vertical grid lines (default: True).
- **ygrid_on** (*bool*) – Whether or not to show horizontal grid lines (default: True).
- **title** (*str*) – Figure title (optional).
- **zorder** (*float*) – Set the zorder for lines. Higher zorder are drawn on top.
- **alpha** (*float*) – Opacity of the line.
- **month_grid_width** (*float*) – the on-figure "horizontal width" that each time interval occupies. This value determines how X axis labels are displayed (e.g., smaller width leads to date labels being displayed with 90 deg rotation). Do not change this unless you really know what you are doing.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

See also:

plot_multiple_timeseries()
Plot multiple time series, with the ability to specify different line specifications for each line.

3.15 Plot multiple time series

plot_utils.**plot_multiple_timeseries**(*multiple_time_series*, *show_legend=True*, *fig=None*, *ax=None*, *figsize=(10, 3)*, *dpi=100*, *ncol_legend=5*, ***kwargs*)

Plot multiple time series.

Note that setting keyword arguments such as `color` or `linestyle` will force all time series to have the same color or line style. So we recommend letting this function generate distinguishable line specifications (color/ linestyle/linewidth combinations) by itself. (Although the more time series, the less the distinguishability. 240 time series or less is recommended.)

Parameters

- **multiple_time_series** (*pandas.DataFrame or pandas.Series*) – If it is a pandas DataFrame, its index is the date, and each column is a different time series. If it is a pandas Series, it will be internally converted into a 1-column pandas DataFrame.
- **fig** (matplotlib.figure.Figure or `None`) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or `None`) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of `fig` (if not `None`) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of `fig` (if not `None`) will override this parameter.
- **ncol_legend** (*int*) – Number of columns of the legend.
- ****kwargs** – Other keyword arguments to be passed to *plot_timeseries()*, such as color, marker, fontsize, alpha, etc.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

See also:

plot_timeseries()
Plot a single set of time series.

3.16 Plot time series with filled error bounds

plot_utils.**fill_timeseries**(*time_series*, *upper_bound*, *lower_bound*, *date_fmt=None*, *fig=None*, *ax=None*, *figsize=(10, 3)*, *dpi=100*, *xlabel='Time'*, *ylabel=None*, *line_label=None*, *shade_label=None*, *color='orange'*, *lw=3*, *ls='-'*, *fontsize=12*, *title=None*, *xgrid_on=True*, *ygrid_on=True*)

Plot time series as a line and then plot the upper and lower bounds as shaded areas.

Parameters

- **time_series** (*pandas.Series*) – A pandas Series, with index being date.
- **upper_bound** (*pandas.Series*) – Upper bounds of the time series, must have the same length as `time_series`.
- **lower_bound** (*pandas.Series*) – Lower bounds of the time series, must have the same length as `time_series`.
- **date_fmt** (*str*) – Date format specifier, e.g., '%Y-%m' or '%d/%m/%y'.
- **fig** (matplotlib.figure.Figure or `None`) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or `None`) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of `fig` (if not `None`) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of `fig` (if not `None`) will override this parameter.
- **xlabel** (*str*) – Label of X axis. Usually "Time" or "Date".
- **ylabel** (*str*) – Label of Y axis. Usually the meaning of the data (e.g., "Gas price [$]").
- **line_label** (*str*) – Label of the line, for plotting legends.
- **shade_label** (*str*) – Label of the shade, for plotting legends.
- **color** (*str or list or tuple*) – Color of line. If None, let Python decide for itself.
- **lw** (*scalar*) – Line width of the line that represents time_series.
- **ls** (*str*) – Line style of the line that represents time_series.
- **fontsize** (*scalar*) – Font size of the texts in the figure.
- **title** (*str*) – Figure title.
- **xgrid_on** (*bool*) – Whether or not to show vertical grid lines (default: `True`).

- **ygrid_on** (*bool*) – Whether or not to show horizontal grid lines (default: `True`).

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

6. Miscellaneous

3.17 Get colors

`plot_utils.`**`get_colors`**(*N=None*, *color_scheme='tab10'*)

Return a list of N distinguisable colors. When N is larger than the color scheme capacity, the color cycle is wrapped around.

What does each color_scheme look like?

https://matplotlib.org/mpl_examples/color/colormaps_reference_04.png https://matplotlib.org/users/dflt_style_changes.html#colors-color-cycles-and-color-maps https://github.com/vega/vega/wiki/Scales#scale-range-literals https://www.mathworks.com/help/matlab/graphics_transition/why-are-plot-lines-different-colors.html

Parameters

- **N** (int or `None`) – Number of qualitative colors desired. If None, returns all the colors in the specified color scheme.
- **color_scheme** (*str or {8.3, 8.4}*) – Color scheme specifier. Valid specifiers are: (1) Matplotlib qualitative color map names:

  'Pastel1' 'Pastel2' 'Paired' 'Accent' 'Dark2' 'Set1' 'Set2' 'Set3' 'tab10' 'tab20' 'tab20b' 'tab20c' (https://matplotlib.org/mpl_examples/color/colormaps_reference_04.png)

  (a) 'tab10_muted': A set of 10 colors that are the muted version of "tab10"

  (b) '8.3' and '8.4': old and new MATLAB color scheme Old: https://www.mathworks.com/help/matlab/graphics_transition/transition_colororder_old.png New: https://www.mathworks.com/help/matlab/graphics_transition/transition_colororder.png

  (c) 'rgbcmyk': old default Matplotlib color palette (v1.5 and earlier)

  (d) 'bw' (or 'bw3'), 'bw4', and 'bw5' Black-and-white (grayscale colors in 3, 4, and 5 levels)

Returns

colors – A list of colors (as RGB, or color name, or hex)

Return type

list<list<float>>, list<str>

3.18 Get line specifications

plot_utils.**get_linespecs**(*color_scheme='tab10'*, *n_linestyle=4*, *range_linewidth=[1, 2, 3]*, *priority='color'*)

Return a list of distinguishable line specifications (color, line style, and line width combinations).

Parameters

- **color_scheme** (*str or {8.3, 8.4}*) – Color scheme specifier. See documentation of `get_colors()` for valid specifiers.
- **n_linestyle** (*{1, 2, 3, 4}*) – Number of different line styles to use. There are only four available line stylies in Matplotlib: (1) - (2) – (3) -. and (4) .. For example, if you use 2, you choose only - and –
- **range_linewidth** (*list, numpy.ndarray, or pandas.Series*) – The range of different line width values to use.
- **priority** (*{'color', 'linestyle', 'linewidth'}*) – Which one of the three line specification aspects (i.e., color, line style, or line width) should change first in the resulting list of line specifications.

Returns

style_cycle_list –

A list whose every element is a dictionary that looks like this:
{'color': '#1f77b4', 'ls': '-', 'lw': 1}.

Each element can then be passed as keyword arguments to `matplotlib.pyplot.plot()` or other similar functions.

Return type
list<dict>

Example

```
>>> import plot_utils as pu
>>> import matplotlib.pyplot as plt
>>> plt.plot([0,1], [0,1], **pu.get_linespecs()[53])
```

3.19 Demonstrating `get_linespecs()`

plot_utils.**linespecs_demo**(*line_specs*, *horizontal_plot=False*)

Demonstrate line specifications generated by :func:~`get_linespecs()`.

3.19.1 Parameter

line_spec
[list<dict>] A list of line specifications. It can be the returned value of *get_linespecs()*.

horizontal_plot
[bool] Whether or not to demonstrate the line specifications in a horizontal plot.

returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

3.20 Two classes for querying colors

class plot_utils.**Color**(*color*, *is_rgb_normalized=True*)

A class that defines a color.

Parameters

- **color** (*str or <tuple> or <list>*) – The color information to initialize the Color object. Can be a list or tuple of 3 elements (i.e., the RGB information), or a HEX string such as "#00FF00", or XKCD color names (https://xkcd.com/color/rgb/) or X11 color names (http://cng.seas.rochester.edu/CNG/docs/x11color.html).
- **is_rgb_normalized** (*bool*) – Whether or not the input information (if RGB) contains the normalized values (such as [0, 0.5, 0.5]). This parameter has no effect if the input is not RGB.

as_hex()

Exports the color object as HEX values.

Returns
hex_val – HEX value.

Return type
str

as_rgb(*normalize=True*)

Export thes color as RGB values.

3.20.1 Parameter

normalize
[bool] Whether or not to return the normalized (between 0 and 1) RGB.

returns
rgb_val – RGB values in three numbers.

rtype
tuple<float>

as_rgba(*alpha=1.0*)

Exports the color object as RGBA values. The R, G, and B values are always normalized (between 0 and 1).

3.20.2 Parameter

alpha
[float] The transparency (0 being completely transparent and 1 opaque).

returns
rgba_val – RGBA values in four numbers.

rtype
tuple<float>

show()

Shows color as a square patch.

class plot_utils.**Multiple_Colors**(*colors*, *is_rgb_normalized=True*)

A class that defines multiple colors.

Parameters

- **colors** (*list*) – A list of color information to initialize the Multiple_Colors object. The list elements can be:
  - a list or tuple of 3 elements (i.e., the RGB information)
  - a HEX string such as "#00FF00"
  - an XKCD color name (https://xkcd.com/color/rgb/)
  - an X11 color name (http://cng.seas.rochester.edu/CNG/docs/x11color.html)

  Different elements of colors do not need to be of the same type.
- **is_rgb_normalized** (*bool*) – Whether or not the input information (if RGB) contains the normalized values (such as [0, 0.5, 0.5]). This parameter has no effect if the input is not RGB.

as_hex()

Exports the colors as a list of HEX values

Returns
hex_list – A list of HEX colors

Return type
list<str>

`as_rgb`(*normalize=True*)

Exports the colors as a list of RGB values

3.20.3 Parameter

normalize
[bool] Whether or not to return the normalized (between 0 and 1) RGB.

returns
rgb_list – A list of list: each sub-list represents a RGB color in three numbers.

rtype
list<list<float>>

`as_rgba`(*alpha=1.0*)

Exports the colors as a list of RGBA values

3.20.4 Parameter

alpha
[float] The transparency (0 being completely transparent and 1 opaque).

returns
rgba_list – A list of list: each sub-list represents a RGBA color in four numbers.

rtype
list<list<float>>

`show`(*vertical=False*, *text=None*)

Shows the colors as square patches

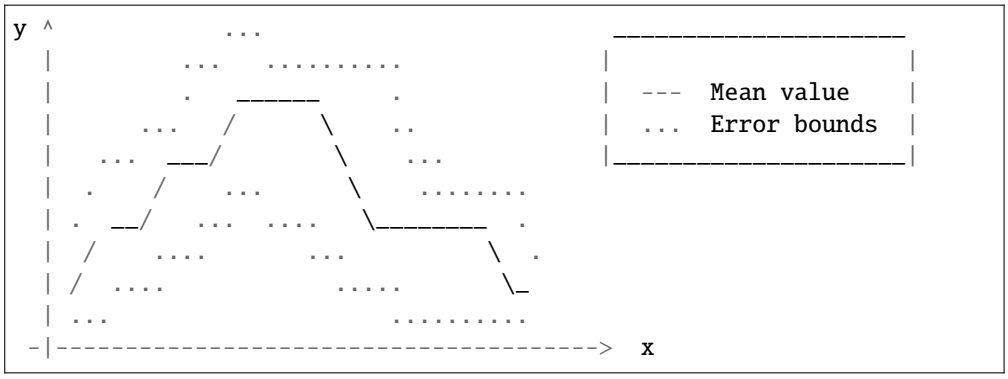
Parameters

- **vertical** (*bool*) – Whether or not to show the patches vertically
- **text** (*str*) – The text to show next to the colors

3.21 Plot lines with upper/lower bounds

`plot_utils.plot_with_error_bounds`(*x*, *y*, *upper_bound*, *lower_bound*, *fig=None*, *ax=None*, *figsize=None*, *dpi=100*, *line_color=[0.4, 0.4, 0.4]*, *shade_color=[0.7, 0.7, 0.7]*, *shade_alpha=0.5*, *linewidth=2.0*, *legend_loc='best'*, *line_label='Data'*, *shade_label='$\\mathregular{\\pm}$STD'*, *logx=False*, *logy=False*, *grid_on=True*)

Plot a graph with one line and its upper and lower bounds, with areas between bounds shaded. The effect is similar to this illustration below:



Parameters

- **x** (*list, numpy.ndarray, or pandas.Series*) – X data points to be plotted as a line.
- **y** (*list, numpy.ndarray, or pandas.Series*) – Y data points to be plotted as a line.
- **fig** (matplotlib.figure.Figure or None) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or None) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of fig (if not None) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of fig (if not None) will override this parameter.
- **upper_bound** (*list, numpy.ndarray, or pandas.Series*) – Upper bound of the Y values.
- **lower_bound** (*list, numpy.ndarray, or pandas.Series*) – Lower bound of the Y values.
- **line_color** (*str, list, or tuple*) – Color of the line.
- **shade_color** (*str, list, or tuple*) – Color of the underlying shades.
- **shade_alpha** (*float*) – Opacity of the shades.
- **linewidth** (*float*) – Width of the line.
- **legend_loc** (*int, str*) – Location of the legend, to be passed directly to plt.legend().
- **line_label** (*str*) – Label of the line, to be used in the legend.
- **shade_label** (*str*) – Label of the shades, to be used in the legend.
- **logx** (*bool*) – Whether or not to show the X axis in log scale.
- **logy** (*bool*) – Whether or not to show the Y axis in log scale.
- **grid_on** (*bool*) – Whether or not to show grids on the plot.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

3.22 Trim images

plot_utils.**trim_img**(*files*, *white_margin=True*, *pad_width=20*, *pad_color='w'*, *inplace=False*, *verbose=True*, *show_old_img=False*, *show_new_img=False*, *forcibly_overwrite=False*, *resize=False*, *resize_ratio=1.0*)

Trim the margins of image file(s) on the hard drive, and (optionally) add padded margins of a specified width and color.

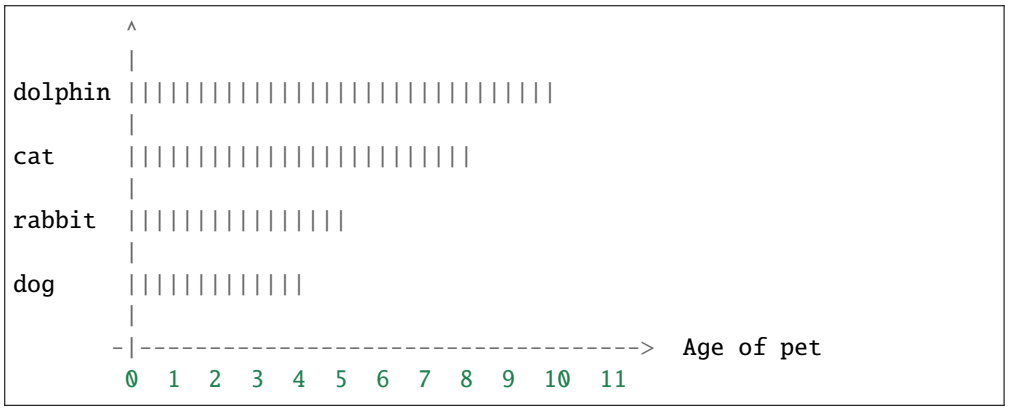
Parameters

- **files** (*str or list<str> or tuple<str>*) – A file name (as Python str) or several file names (as Python list or tuple) to be trimmed.
- **white_margin** (*bool*) – Whether to treat white color as the margin to be trimmed. If `True`, white image margins will be trimmed. If `False`, black image margins will be trimmed.
- **pad_width** (*float*) – The amount of white margins to be padded (unit: pixels).
- **pad_color** (*str or tuple<float> or list<float>*) – The color of the padded margin. Valid `pad_color` values are color names recognizable by matplotlib: https://matplotlib.org/tutorials/colors/colors.html
- **inplace** (*bool*) – Whether or not to replace the existing figure file with the trimmed content.
- **verbose** (*bool*) – Whether or not to print the progress onto the console.
- **show_old_img** (*bool*) – Whether or not to show the old figure in the console.
- **show_new_img** (*bool*) – Whether or not to show the trimmed figure in the console.
- **forcibly_overwrite** (*bool*) – Whether or not to overwrite an image on the hard drive with the same name. Only applicable when `inplace` is `False`.
- **resize** (*bool*) – Whether to resize the padded image
- **resize_ratio** (*float*) – The image resizing ratio. It has no effect if ``resize` is false. For example, if it's 0.5, it means resizing to 50% of the original width and height.

3.23 Plot ranking

plot_utils.**plot_ranking**(*ranking*, *fig=None*, *ax=None*, *figsize='auto'*, *dpi=100*, *barh=True*, *top_n=None*, *score_ax_label=None*, *name_ax_label=None*, *invert_name_ax=False*, *grid_on=True*)

Plot rankings as a bar plot (in descending order), such as:



Parameters

- **ranking** (*dict or pandas.Series*) –

  The ranking information, for example:
  {'rabbit': 5, 'cat': 8, 'dog': 4, 'dolphin': 10}

  It does not need to be sorted externally.

- **fig** (matplotlib.figure.Figure or None) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or None) – Axes object. If None, a new axes will be created.
- **figsize** (*(float, float)*) – Figure size in inches, as a tuple of two numbers. The figure size of fig (if not None) will override this parameter.
- **dpi** (*float*) – Figure resolution. The dpi of fig (if not None) will override this parameter.
- **barh** (*bool*) – Whether or not to show the bars as horizontal (otherwise, vertical)
- **top_n** (*int*) – If None, show all categories. top_n > 0 means showing the highest top_n categories. top_n < 0 means showing the lowest |``top_n``| categories.
- **score_ax_label** (*str*) – Label of the score axis (e.g., "Age of pet").
- **name_ax_label** (*str*) – Label of the "category name" axis (e.g., "Pet name").
- **invert_name_ax** (*bool*) – Whether to invert the "category name" axis. For example, if invert_name_ax is False, then higher values are shown on the top if barh is True.
- **grid_on** (*bool*) – Whether or not to show grids on the plot.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

3.24 Visualize CV scores

plot_utils.**visualize_cv_scores**(*fig=None*, *ax=None*, *dpi=100*, *n_folds=5*, *cv_scores=None*, *box_height=0.6*, *box_width=0.9*, *gap_frac=0.05*, *metric_name='AUC'*, *avg_cv_score=None*, *no_holdout_set=False*, *holdout_score=None*, *fontsize=9*, *flip_yaxis=True*)

Visualize K-fold cross-validation scores as well as hold-out set performance in an intuitive way.

Parameters

- **fig** (matplotlib.figure.Figure or `None`) – Figure object. If None, a new figure will be created.
- **ax** (matplotlib.axes._subplots.AxesSubplot or `None`) – Axes object. If None, a new axes will be created.
- **dpi** (*float*) – Figure resolution. The dpi of `fig` (if not `None`) will override this parameter.
- **n_folds** (*int*) – Number of CV folds.
- **cv_scores** (list<float> or `None`) – The validation score of each fold. If `None`, no scores will be shown on the small boxes.
- **box_height** (*float*) – The height of the the small box, in inches.
- **box_width** (*float*) – The width of the small box, in inches.
- **gap_frac** (*float*) – How much gap should there be between each small box.
- **metric_name** (*str*) – The name of the metric to be shown in the figure.
- **avg_cv_score** (float or `None`) – The average cross-validation score. If `None` (recommended), it will be calculated by numpy.mean(cv_scores).
- **no_holdout_set** (*bool*) – If `False`, the hold-out data set will be visualized alongside the training data set. This parameter supersedes `holdout_score`.
- **holdout_score** (float or `None`) – The performance on the hold-out data set. If `no_holdout_set` is `True`, this parameter has no effect.
- **fontsize** (*float*) – The font size of all the texts.
- **flip_yaxis** (*bool*) – If `True`, everything will be flipped upside down. This parameter is for diagnosis and and debugging purpose only. It is recommended to leave it as `True`.

Returns

- **fig** (*matplotlib.figure.Figure*) – The figure object being created or being passed into this function.
- **ax** (*matplotlib.axes._subplots.AxesSubplot*) – The axes object being created or being passed into this function.

2. Other helper functions

3.25 Convert FIPS codes to state names

plot_utils.**_convert_FIPS_to_state_name**(*dict1*)

Convert state FIPS codes such as '01' and '45' into full state names.

3.25.1 Parameter

dict1

[dict] A dictionary whose keys are 2-digit FIPS codes of state names.

returns

dict3 – A dictionary whose keys are state abbreviations. Its values of each state come from `dict`.

rtype

dict

3.26 Translate between full state names and abbreviations

plot_utils.**_translate_state_abbrev**(*dict1*, *abbrev_to_full=True*)

Convert state full names into state abbreviations, or the other way. Overseas territories (except Puerto Rico) cannot be converted.

Robustness is not guaranteed: if invalide state names (full or abbreviated) exist in dict1, a KeyError will be raised.

Parameters

- **dict1** (*dict*) – A mapping between state name and some data, e.g., {'AK': 1, 'AL': 2, ...}
- **abbrev_to_full** (*bool*) – If `True`, translate {'AK': 1, 'AL': 2, ...} into {'Alaska': 1, 'Alabama': 2, ...}. If `False`, the opposite way.

Returns

dict2 – The converted dictionary

Return type

dict

3.27 Find axes limits

plot_utils.**_find_axes_lim**(*data_limit*, *tick_base_unit*, *direction='upper'*)

Return a "whole" number to be used as the upper or lower limit of axes.

For example, if the maximum x value of the data is 921.5, and you would like the upper x_limit to be a multiple of 50, then this function returns 950.

Parameters

- **data_limit** (*float, int, list<float>, list<int>, tuple<float>, tuple<int>*) –

  The upper and/or lower limit(s) of data.

  (a) If a tuple (or list) of two elements is provided, then the upper and lower axis limits are automatically determined. (The order of the two elements does not matter.)

  (b) If a float or an int is provided, then the axis limit is determined based on the `direction` provided.

- **tick_base_unit** (*float*) – For example, if you want your axis limit(s) to be a multiple of 20 (such as 80, 120, 2020, etc.), then use 20.

- **direction** (*{'upper', 'lower'}*) – The direction of the limit to be found. For example, if the maximum of the data is 127, and `tick_base_unit` is 50, then a `direction` of lower yields a result of 100. This parameter is effective only when `data_limit` is a scalar.

Returns

axes_lim – If `data_limit` is a list/tuple of length 2, return a list: [min_limit, max_limit] (always ordered no matter what the order of `data_limit` is). If `data_limit` is a scalar, return the axis limit according to `direction`.

Return type

list<float> or float

CHAPTER

FOUR

GALLERY

See here.

CHAPTER

FIVE

EXAMPLES

Examples are presented as Jupyter notebooks here.

CHAPTER

SIX

COPYRIGHT AND LICENSE

CHAPTER

SEVEN

GITHUB REPOSITORY

https://github.com/jsh9/python-plot-utils

Bug reports and/or suggestions are welcome!

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX