

To produce a stacked bar plot, pass stacked=True:

```
In [22]: df2.plot.bar(stacked=True);
```



Histograms

Histograms can be drawn by using the DataFrame.plot.hist() and Series.plot.hist() methods.

```
In [24]: df4 = pd.DataFrame({'a': np.random.randn(1000) + 1, 'b': np.random.
\rightarrowrandn(1000),
                               'c': np.random.randn(1000) - 1}, columns=['a', 'b', 'c'])
   . . . . :
   . . . . :
                                            Traceback (most recent call last)
NameError
<ipython-input-24-3b054428c392> in <module>
---> 1 df4 = pd.DataFrame({'a': np.random.randn(1000) + 1, 'b': np.random.
\hookrightarrowrandn(1000),
      2
                             'c': np.random.randn(1000) - 1}, columns=['a', 'b', 'c'])
NameError: name 'pd' is not defined
In [25]: plt.figure();
In [26]: df4.plot.hist(alpha=0.5)
NameError
                                            Traceback (most recent call last)
<ipython-input-26-d12a7608cec9> in <module>
----> 1 df4.plot.hist(alpha=0.5)
```

(continues on next page)

(continued from previous page)

```
NameError: name 'df4' is not defined
```

A histogram can be stacked using stacked=True. Bin size can be changed using the bins keyword.

You can pass other keywords supported by matplotlib hist. For example, horizontal and cumulative histograms can be drawn by orientation='horizontal' and cumulative=True.

See the hist method and the matplotlib hist documentation for more.

The existing interface ${\tt DataFrame.hist}$ to plot histogram still can be used.

DataFrame.hist() plots the histograms of the columns on multiple subplots:

The by keyword can be specified to plot grouped histograms:

Box plots

Boxplot can be drawn calling Series.plot.box() and DataFrame.plot.box(), or DataFrame.boxplot() to visualize the distribution of values within each column.

For instance, here is a boxplot representing five trials of 10 observations of a uniform random variable on [0,1).

Boxplot can be colorized by passing color keyword. You can pass a dict whose keys are boxes, whiskers, medians and caps. If some keys are missing in the dict, default colors are used for the corresponding artists. Also, boxplot has sym keyword to specify fliers style.

When you pass other type of arguments via color keyword, it will be directly passed to matplotlib for all the boxes, whiskers, medians and caps colorization.

The colors are applied to every boxes to be drawn. If you want more complicated colorization, you can get each drawn artists by passing *return_type*.

Also, you can pass other keywords supported by matplotlib boxplot. For example, horizontal and custom-positioned boxplot can be drawn by vert=False and positions keywords.

See the boxplot method and the matplotlib boxplot documentation for more.

The existing interface DataFrame.boxplot to plot boxplot still can be used.

You can create a stratified boxplot using the by keyword argument to create groupings. For instance,

```
In [45]: df = pd.DataFrame(np.random.rand(10, 2), columns=['Col1', 'Col2'])
          ______
                               Traceback (most recent call last)
<ipython-input-45-da722611cdbb> in <module>
----> 1 df = pd.DataFrame(np.random.rand(10, 2), columns=['Col1', 'Col2'])
NameError: name 'pd' is not defined
Traceback (most recent call last)
NameError
<ipython-input-46-b2bbda782b40> in <module>
NameError: name 'pd' is not defined
In [47]: plt.figure();
In [48]: bp = df.boxplot(by='X')
NameError
                               Traceback (most recent call last)
<ipython-input-48-8598e842a6ba> in <module>
----> 1 bp = df.boxplot(by='X')
```

(continues on next page)

(continued from previous page)

```
NameError: name 'df' is not defined
```

You can also pass a subset of columns to plot, as well as group by multiple columns:

(continues on next page)

(continued from previous page)

In boxplot, the return type can be controlled by the return_type, keyword. The valid choices are {"axes", "dict", "both", None}. Faceting, created by DataFrame.boxplot with the by keyword, will affect the output type as well:

return_type=	Faceted	Output type
None	No	axes
None	Yes	2-D ndarray of axes
'axes'	No	axes
'axes'	Yes	Series of axes
'dict'	No	dict of artists
'dict'	Yes	Series of dicts of artists
'both'	No	namedtuple
'both'	Yes	Series of namedtuples

Groupby.boxplot always returns a Series of return_type.

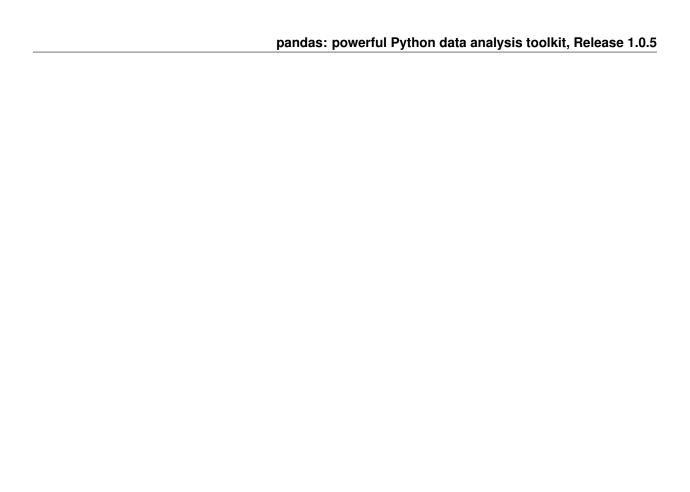
```
In [54]: np.random.seed(1234)
In [55]: df_box = pd.DataFrame(np.random.randn(50, 2))
                                      Traceback (most recent call last)
NameError
<ipython-input-55-043b0e16e969> in <module>
----> 1 df_box = pd.DataFrame(np.random.randn(50, 2))
NameError: name 'pd' is not defined
In [56]: df_box['g'] = np.random.choice(['A', 'B'], size=50)
_____
NameError
                                       Traceback (most recent call last)
<ipython-input-56-e39101f788cc> in <module>
----> 1 df_box['g'] = np.random.choice(['A', 'B'], size=50)
NameError: name 'df_box' is not defined
In [57]: df_box.loc[df_box['g'] == 'B', 1] += 3
        _____
NameError
                                      Traceback (most recent call last)
<ipython-input-57-996ee2e7f114> in <module>
----> 1 df_box.loc[df_box['g'] == 'B', 1] += 3
NameError: name 'df_box' is not defined
In [58]: bp = df_box.boxplot(by='g')
                                      Traceback (most recent call last)
<ipython-input-58-8fc769e009a9> in <module>
----> 1 bp = df_box.boxplot(by='g')
NameError: name 'df_box' is not defined
```

The subplots above are split by the numeric columns first, then the value of the g column. Below the subplots are first split by the value of g, then by the numeric columns.

Area plot

You can create area plots with Series.plot.area() and DataFrame.plot.area(). Area plots are stacked by default. To produce stacked area plot, each column must be either all positive or all negative values.

When input data contains *NaN*, it will be automatically filled by 0. If you want to drop or fill by different values, use dataframe.dropna() or dataframe.fillna() before calling *plot*.



To produce an unstacked plot, pass stacked=False. Alpha value is set to 0.5 unless otherwise specified:

```
In [62]: df.plot.area(stacked=False);
```

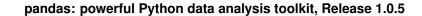
Scatter plot

Scatter plot can be drawn by using the DataFrame.plot.scatter() method. Scatter plot requires numeric columns for the x and y axes. These can be specified by the x and y keywords.

To plot multiple column groups in a single axes, repeat plot method specifying target ax. It is recommended to specify color and label keywords to distinguish each groups.

```
In [65]: ax = df.plot.scatter(x='a', y='b', color='DarkBlue', label='Group 1');
In [66]: df.plot.scatter(x='c', y='d', color='DarkGreen', label='Group 2', ax=ax);
```

		oolkit, Release 1.0.5		
The keyword a may be	given on the name of a		and maint	
		column to provide colors for	each point:	
	given as the name of a c		each point:	
			each point:	



You can pass other keywords supported by matplotlib scatter. The example below shows a bubble chart using a column of the DataFrame as the bubble size.

```
In [68]: df.plot.scatter(x='a', y='b', s=df['c'] * 200);
```

See the scatter method and the matplotlib scatter documentation for more.

Hexagonal bin plot

You can create hexagonal bin plots with <code>DataFrame.plot.hexbin()</code>. Hexbin plots can be a useful alternative to scatter plots if your data are too dense to plot each point individually.

(continues on next page)

(continued from previous page)

```
NameError

Traceback (most recent call last)

<ipython-input-71-48fcf967aa91> in <module>
----> 1 df.plot.hexbin(x='a', y='b', gridsize=25)

NameError: name 'df' is not defined
```

A useful keyword argument is gridsize; it controls the number of hexagons in the x-direction, and defaults to 100. A larger gridsize means more, smaller bins.

By default, a histogram of the counts around each (x, y) point is computed. You can specify alternative aggregations by passing values to the C and reduce_C_function arguments. C specifies the value at each (x, y) point and reduce_C_function is a function of one argument that reduces all the values in a bin to a single number (e.g. mean, max, sum, std). In this example the positions are given by columns a and b, while the value is given by column z. The bins are aggregated with NumPy's max function.

(continues on next page)