(continued from previous page)

```
In [73]: df['b'] = df['b'] = df['b'] + np.arange(1000)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-73-2214dd14e63e> in <module>
----> 1 df['b'] = df['b'] = df['b'] + np.arange(1000)

NameError: name 'df' is not defined

In [74]: df['z'] = np.random.uniform(0, 3, 1000)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-74-a84ae78b3d08> in <module>
----> 1 df['z'] = np.random.uniform(0, 3, 1000)

NameError: name 'df' is not defined

In [75]: df.plot.hexbin(x='a', y='b', C='z', reduce_C_function=np.max, gridsize=25)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-75-76020970b51d> in <module>
----> 1 df.plot.hexbin(x='a', y='b', C='z', reduce_C_function=np.max, gridsize=25)

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

See the `hexbin` method and the matplotlib hexbin documentation for more.

Pie plot

You can create a pie plot with `DataFrame.plot.pie()` or `Series.plot.pie()`. If your data includes any `NaN`, they will be automatically filled with 0. A `ValueError` will be raised if there are any negative values in your data.

```
In [76]: series = pd.Series(3 * np.random.rand(4),
   ....:                    index=['a', 'b', 'c', 'd'], name='series')
   ....:
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-76-ede9c1f3bcce> in <module>
----> 1 series = pd.Series(3 * np.random.rand(4),
      2                    index=['a', 'b', 'c', 'd'], name='series')

NameError: name 'pd' is not defined

In [77]: series.plot.pie(figsize=(6, 6))
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-77-e4c2454e0a19> in <module>
----> 1 series.plot.pie(figsize=(6, 6))

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

For pie plots it's best to use square figures, i.e. a figure aspect ratio 1. You can create the figure with equal width and

height, or force the aspect ratio to be equal after plotting by calling `ax.set_aspect('equal')` on the returned `axes` object.

Note that pie plot with `DataFrame` requires that you either specify a target column by the `y` argument or `subplots=True`. When `y` is specified, pie plot of selected column will be drawn. If `subplots=True` is specified, pie plots for each column are drawn as subplots. A legend will be drawn in each pie plots by default; specify `legend=False` to hide it.

```
In [78]: df = pd.DataFrame(3 * np.random.rand(4, 2),
   ....:                   index=['a', 'b', 'c', 'd'], columns=['x', 'y'])
   ....:
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-78-753e3aa96aef> in <module>
----> 1 df = pd.DataFrame(3 * np.random.rand(4, 2),
      2                   index=['a', 'b', 'c', 'd'], columns=['x', 'y'])

NameError: name 'pd' is not defined

In [79]: df.plot.pie(subplots=True, figsize=(8, 4))
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-79-bd4770dabaff> in <module>
----> 1 df.plot.pie(subplots=True, figsize=(8, 4))

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

You can use the `labels` and `colors` keywords to specify the labels and colors of each wedge.

Warning: Most pandas plots use the `label` and `color` arguments (note the lack of "s" on those). To be consistent with `matplotlib.pyplot.pie()` you must use `labels` and `colors`.

If you want to hide wedge labels, specify `labels=None`. If `fontsize` is specified, the value will be applied to wedge labels. Also, other keywords supported by `matplotlib.pyplot.pie()` can be used.

```
In [80]: series.plot.pie(labels=['AA', 'BB', 'CC', 'DD'], colors=['r', 'g', 'b', 'c'],
   ....:                 autopct='%.2f', fontsize=20, figsize=(6, 6))
   ....:
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-80-f6a8e8e24c35> in <module>
----> 1 series.plot.pie(labels=['AA', 'BB', 'CC', 'DD'], colors=['r', 'g', 'b', 'c'],
      2                 autopct='%.2f', fontsize=20, figsize=(6, 6))

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

If you pass values whose sum total is less than 1.0, matplotlib draws a semicircle.

```
In [81]: series = pd.Series([0.1] * 4, index=['a', 'b', 'c', 'd'], name='series2')
---------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
NameError                                 Traceback (most recent call last)
<ipython-input-81-80a435a1151e> in <module>
----> 1 series = pd.Series([0.1] * 4, index=['a', 'b', 'c', 'd'], name='series2')

NameError: name 'pd' is not defined

In [82]: series.plot.pie(figsize=(6, 6))
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-82-e4c2454e0a19> in <module>
----> 1 series.plot.pie(figsize=(6, 6))

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

See the matplotlib pie documentation for more.

2.11.3 Plotting with missing data

Pandas tries to be pragmatic about plotting `DataFrames` or `Series` that contain missing data. Missing values are dropped, left out, or filled depending on the plot type.

Plot Type	NaN Handling
Line	Leave gaps at NaNs
Line (stacked)	Fill 0's
Bar	Fill 0's
Scatter	Drop NaNs
Histogram	Drop NaNs (column-wise)
Box	Drop NaNs (column-wise)
Area	Fill 0's
KDE	Drop NaNs (column-wise)
Hexbin	Drop NaNs
Pie	Fill 0's

If any of these defaults are not what you want, or if you want to be explicit about how missing values are handled, consider using *`fillna()`* or *`dropna()`* before plotting.

2.11.4 Plotting Tools

These functions can be imported from `pandas.plotting` and take a `Series` or `DataFrame` as an argument.

Scatter matrix plot

You can create a scatter plot matrix using the `scatter_matrix` method in `pandas.plotting`:

```
In [83]: from pandas.plotting import scatter_matrix

In [84]: df = pd.DataFrame(np.random.randn(1000, 4), columns=['a', 'b', 'c', 'd'])
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-84-d09bb7ca2382> in <module>
----> 1 df = pd.DataFrame(np.random.randn(1000, 4), columns=['a', 'b', 'c', 'd'])

NameError: name 'pd' is not defined

In [85]: scatter_matrix(df, alpha=0.2, figsize=(6, 6), diagonal='kde')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-85-be6b017bf310> in <module>
----> 1 scatter_matrix(df, alpha=0.2, figsize=(6, 6), diagonal='kde')

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

Density plot

You can create density plots using the `Series.plot.kde()` and `DataFrame.plot.kde()` methods.

```
In [86]: ser = pd.Series(np.random.randn(1000))
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-86-58c851be3369> in <module>
----> 1 ser = pd.Series(np.random.randn(1000))

NameError: name 'pd' is not defined

In [87]: ser.plot.kde()
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-87-270b2425f93c> in <module>
----> 1 ser.plot.kde()

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

Andrews curves

Andrews curves allow one to plot multivariate data as a large number of curves that are created using the attributes of samples as coefficients for Fourier series, see the Wikipedia entry for more information. By coloring these curves differently for each class it is possible to visualize data clustering. Curves belonging to samples of the same class will usually be closer together and form larger structures.

Note: The "Iris" dataset is available here.

```
In [88]: from pandas.plotting import andrews_curves

In [89]: data = pd.read_csv('data/iris.data')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-89-ea4716c8ea20> in <module>
----> 1 data = pd.read_csv('data/iris.data')

NameError: name 'pd' is not defined

In [90]: plt.figure()
Out[90]: <Figure size 640x480 with 0 Axes>

In [91]: andrews_curves(data, 'Name')
---------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
NameError                                 Traceback (most recent call last)
<ipython-input-91-fc5acbcde18a> in <module>
----> 1 andrews_curves(data, 'Name')

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

Parallel coordinates

Parallel coordinates is a plotting technique for plotting multivariate data, see the Wikipedia entry for an introduction. Parallel coordinates allows one to see clusters in data and to estimate other statistics visually. Using parallel coordinates points are represented as connected line segments. Each vertical line represents one attribute. One set of connected line segments represents one data point. Points that tend to cluster will appear closer together.

```
In [92]: from pandas.plotting import parallel_coordinates

In [93]: data = pd.read_csv('data/iris.data')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-93-ea4716c8ea20> in <module>
----> 1 data = pd.read_csv('data/iris.data')

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

(continues on next page)

(continued from previous page)

```
In [94]: plt.figure()
Out[94]: <Figure size 640x480 with 0 Axes>

In [95]: parallel_coordinates(data, 'Name')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-95-7d09ca821842> in <module>
----> 1 parallel_coordinates(data, 'Name')

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

Lag plot

Lag plots are used to check if a data set or time series is random. Random data should not exhibit any structure in the lag plot. Non-random structure implies that the underlying data are not random. The `lag` argument may be passed, and when `lag=1` the plot is essentially `data[:-1]` vs. `data[1:]`.

```
In [96]: from pandas.plotting import lag_plot

In [97]: plt.figure()
Out[97]: <Figure size 640x480 with 0 Axes>
```

(continues on next page)

(continued from previous page)

```
In [98]: spacing = np.linspace(-99 * np.pi, 99 * np.pi, num=1000)

In [99]: data = pd.Series(0.1 * np.random.rand(1000) + 0.9 * np.sin(spacing))
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-99-a1dee79fc325> in <module>
----> 1 data = pd.Series(0.1 * np.random.rand(1000) + 0.9 * np.sin(spacing))

NameError: name 'pd' is not defined

In [100]: lag_plot(data)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-100-76d4c87cecfc> in <module>
----> 1 lag_plot(data)

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

Autocorrelation plot

Autocorrelation plots are often used for checking randomness in time series. This is done by computing autocorrelations for data values at varying time lags. If time series is random, such autocorrelations should be near zero for any and all time-lag separations. If time series is non-random then one or more of the autocorrelations will be significantly non-zero. The horizontal lines displayed in the plot correspond to 95% and 99% confidence bands. The dashed line is 99% confidence band. See the Wikipedia entry for more about autocorrelation plots.

```
In [101]: from pandas.plotting import autocorrelation_plot

In [102]: plt.figure()
Out[102]: <Figure size 640x480 with 0 Axes>

In [103]: spacing = np.linspace(-9 * np.pi, 9 * np.pi, num=1000)

In [104]: data = pd.Series(0.7 * np.random.rand(1000) + 0.3 * np.sin(spacing))
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-104-8a50b1acf632> in <module>
----> 1 data = pd.Series(0.7 * np.random.rand(1000) + 0.3 * np.sin(spacing))

NameError: name 'pd' is not defined

In [105]: autocorrelation_plot(data)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-105-eccad460986f> in <module>
----> 1 autocorrelation_plot(data)

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

Bootstrap plot

Bootstrap plots are used to visually assess the uncertainty of a statistic, such as mean, median, midrange, etc. A random subset of a specified size is selected from a data set, the statistic in question is computed for this subset and the process is repeated a specified number of times. Resulting plots and histograms are what constitutes the bootstrap plot.

```
In [106]: from pandas.plotting import bootstrap_plot

In [107]: data = pd.Series(np.random.rand(1000))
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-107-a21ce4cd1aac> in <module>
----> 1 data = pd.Series(np.random.rand(1000))

NameError: name 'pd' is not defined

In [108]: bootstrap_plot(data, size=50, samples=500, color='grey')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-108-126f9a3d5653> in <module>
----> 1 bootstrap_plot(data, size=50, samples=500, color='grey')

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

RadViz

RadViz is a way of visualizing multi-variate data. It is based on a simple spring tension minimization algorithm. Basically you set up a bunch of points in a plane. In our case they are equally spaced on a unit circle. Each point represents a single attribute. You then pretend that each sample in the data set is attached to each of these points by a spring, the stiffness of which is proportional to the numerical value of that attribute (they are normalized to unit interval). The point in the plane, where our sample settles to (where the forces acting on our sample are at an equilibrium) is where a dot representing our sample will be drawn. Depending on which class that sample belongs it will be colored differently. See the R package Radviz for more information.

Note: The "Iris" dataset is available here.

```
In [109]: from pandas.plotting import radviz

In [110]: data = pd.read_csv('data/iris.data')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-110-ea4716c8ea20> in <module>
----> 1 data = pd.read_csv('data/iris.data')

NameError: name 'pd' is not defined

In [111]: plt.figure()
Out[111]: <Figure size 640x480 with 0 Axes>
```

(continues on next page)

(continued from previous page)

```
In [112]: radviz(data, 'Name')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-112-1720a88f3922> in <module>
----> 1 radviz(data, 'Name')

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

2.11.5 Plot Formatting

Setting the plot style

From version 1.5 and up, matplotlib offers a range of pre-configured plotting styles. Setting the style can be used to easily give plots the general look that you want. Setting the style is as easy as calling `matplotlib.style.use(my_plot_style)` before creating your plot. For example you could write `matplotlib.style.use('ggplot')` for ggplot-style plots.

You can see the various available style names at `matplotlib.style.available` and it's very easy to try them out.

General plot style arguments

Most plotting methods have a set of keyword arguments that control the layout and formatting of the returned plot:

```
In [113]: plt.figure();

In [114]: ts.plot(style='k--', label='Series');
```

For each kind of plot (e.g. *line*, *bar*, *scatter*) any additional arguments keywords are passed along to the corresponding matplotlib function (`ax.plot()`, `ax.bar()`, `ax.scatter()`). These can be used to control additional styling, beyond what pandas provides.

Controlling the legend

You may set the `legend` argument to `False` to hide the legend, which is shown by default.

```
In [115]: df = pd.DataFrame(np.random.randn(1000, 4),
   .....:                   index=ts.index, columns=list('ABCD'))
   .....:
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-115-ae243d2a43b5> in <module>
----> 1 df = pd.DataFrame(np.random.randn(1000, 4),
      2                   index=ts.index, columns=list('ABCD'))
```

(continues on next page)

(continued from previous page)

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

In [116]: df = df.cumsum()
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-116-08208d45ae16> in <module>
----> 1 df = df.cumsum()

NameError: name 'df' is not defined

In [117]: df.plot(legend=False)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-117-c885e70fbb28> in <module>
----> 1 df.plot(legend=False)

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

Scales

You may pass `logy` to get a log-scale Y axis.

```
In [118]: ts = pd.Series(np.random.randn(1000),
   .....:                index=pd.date_range('1/1/2000', periods=1000))
   .....:
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-118-00eeb137fb11> in <module>
----> 1 ts = pd.Series(np.random.randn(1000),
      2                index=pd.date_range('1/1/2000', periods=1000))

NameError: name 'pd' is not defined

In [119]: ts = np.exp(ts.cumsum())
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-119-a60c32c780a6> in <module>
----> 1 ts = np.exp(ts.cumsum())

NameError: name 'ts' is not defined

In [120]: ts.plot(logy=True)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-120-9e595842ea79> in <module>
----> 1 ts.plot(logy=True)

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

See also the `logx` and `loglog` keyword arguments.

Plotting on a secondary y-axis

To plot data on a secondary y-axis, use the `secondary_y` keyword:

```
In [121]: df['A'].plot()
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-121-142941d65816> in <module>
----> 1 df['A'].plot()

NameError: name 'df' is not defined

In [122]: df['B'].plot(secondary_y=True, style='g')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-122-d13b0146b561> in <module>
----> 1 df['B'].plot(secondary_y=True, style='g')

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

To plot some columns in a `DataFrame`, give the column names to the `secondary_y` keyword:

```
In [123]: plt.figure()
Out[123]: <Figure size 640x480 with 0 Axes>

In [124]: ax = df.plot(secondary_y=['A', 'B'])
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-124-c7f4eaf8c12b> in <module>
----> 1 ax = df.plot(secondary_y=['A', 'B'])

NameError: name 'df' is not defined

In [125]: ax.set_ylabel('CD scale')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-125-0396311d12a3> in <module>
----> 1 ax.set_ylabel('CD scale')

NameError: name 'ax' is not defined

In [126]: ax.right_ax.set_ylabel('AB scale')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-126-5ddf1e3892e0> in <module>
```

(continues on next page)

(continued from previous page)

```
----> 1 ax.right_ax.set_ylabel('AB scale')

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

Note that the columns plotted on the secondary y-axis is automatically marked with "(right)" in the legend. To turn off the automatic marking, use the `mark_right=False` keyword:

```
In [127]: plt.figure()
Out[127]: <Figure size 640x480 with 0 Axes>

In [128]: df.plot(secondary_y=['A', 'B'], mark_right=False)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-128-cfe006c313fe> in <module>
----> 1 df.plot(secondary_y=['A', 'B'], mark_right=False)

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

Custom formatters for timeseries plots

Changed in version 1.0.0.

Pandas provides custom formatters for timeseries plots. These change the formatting of the axis labels for dates and times. By default, the custom formatters are applied only to plots created by pandas with `DataFrame.plot()` or `Series.plot()`. To have them apply to all plots, including those made by matplotlib, set the option `pd.options.plotting.matplotlib.register_converters = True` or use *pandas.plotting.register_matplotlib_converters()*.

Suppressing tick resolution adjustment

pandas includes automatic tick resolution adjustment for regular frequency time-series data. For limited cases where pandas cannot infer the frequency information (e.g., in an externally created `twinx`), you can choose to suppress this behavior for alignment purposes.

Here is the default behavior, notice how the x-axis tick labeling is performed:

```
In [129]: plt.figure()
Out[129]: <Figure size 640x480 with 0 Axes>

In [130]: df['A'].plot()
-----------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
NameError                                 Traceback (most recent call last)
<ipython-input-130-142941d65816> in <module>
----> 1 df['A'].plot()

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

Using the `x_compat` parameter, you can suppress this behavior:

```
In [131]: plt.figure()
Out[131]: <Figure size 640x480 with 0 Axes>

In [132]: df['A'].plot(x_compat=True)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-132-3060a6ce70ed> in <module>
----> 1 df['A'].plot(x_compat=True)

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

If you have more than one plot that needs to be suppressed, the `use` method in `pandas.plotting.plot_params` can be used in a *with statement*:

```
In [133]: plt.figure()
Out[133]: <Figure size 640x480 with 0 Axes>

In [134]: with pd.plotting.plot_params.use('x_compat', True):
   .....:     df['A'].plot(color='r')
   .....:     df['B'].plot(color='g')
   .....:     df['C'].plot(color='b')
   .....:
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-134-b939e52d1f0a> in <module>
----> 1 with pd.plotting.plot_params.use('x_compat', True):
      2     df['A'].plot(color='r')
      3     df['B'].plot(color='g')
      4     df['C'].plot(color='b')

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

Automatic date tick adjustment

`TimedeltaIndex` now uses the native matplotlib tick locator methods, it is useful to call the automatic date tick adjustment from matplotlib for figures whose ticklabels overlap.

See the `autofmt_xdate` method and the matplotlib documentation for more.