1 pandas

pandas is a high-performance module that provides a comprehensive set of structures for working with data. pandas excels at handling structured data, such as data sets containing many variables, working with missing values and merging across multiple data sets.

While extremely useful, pandas is not an essential component of the Python scientific stack unlike NumPy, SciPy or matplotlib, and so while pandas doesn't make data analysis possible in Python, it makes it much easier. pandas also provides high-performance, robust methods for importing from and exporting to a wide range of formats.

1.1 Data Structures

pandas provides a set of data structures which include Series, DataFrames and Panels. Series are 1-dimensional arrays. DataFrames are collections of Series and so are 2-dimensional, and Panels are collections of DataFrames, and so are 3-dimensional. Note that the Panel type is not covered in this chapter.

head and tail

head() shows the first 5 rows of a series, and tail() shows the last 5 rows. An optional argument can be used to return a different number of entries, as in head(10).

isnull and notnull

- isnull() returns a Series with the same indices containing Boolean values indicating True for null values which include NaN and None, among others.
- notnull() returns the negation of isnull() – that is, True for non-null values, and False otherwise.

1.1.1 ix

ix is the indexing function and s.ix[0:2] is the same as s[0:2]. ix is more useful for DataFrames.

1.2 Statistical Function

pandas Series and DataFrame are derived from NumPy arrays and so the vast majority of simple statistical functions are available. This list includes sum, mean, std, var, skew, kurt, prod, median, quantile, abs, cumsum, and cumprod. DataFrame also supports cov and corr – the keyword argument axis determines the direction of the operation (0 for down columns, 1 for across rows). Novel statistical routines are described below.

count

count returns number of non-null values – that is, those which are not NaN or another null value such as None or NaT (not a time, for datetimes).

describe

describe provides a summary of the Series or DataFrame.

```
>>> state_gdp.describe()
gdp_2009 gdp_2010 gdp_2011 gdp_2012
count 51.000000 51.000000 51.000000 51.000000
mean 246866.980392 252840.666667 256995.647059 263327.313725
std 299134.165365 304446.797050 309689.475995 319842.518074
min 22108.000000 23341.000000 23639.000000 23912.000000
25% 64070.500000 65229.000000 65714.000000 66288.000000
50% 149843.000000 153839.000000 155390.000000 157272.000000
75% 307522.500000 318748.500000 327488.500000 337016.000000
max 1667152.000000 1672473.000000 1692301.000000 1751002.000000
gdp_growth_2009 gdp_growth_2010 gdp_growth_2011 gdp_growth_2012
count 51.000000 51.000000 51.000000 51.000000
mean 2.313725
2.462745 1.590196 2.103922
std 3.077663 1.886474 1.610497 1.948944
min 9.100000
1.700000
2.600000
0.100000
25% 3.900000
1.450000 0.900000 1.250000
50% 2.400000
2.300000 1.700000 1.900000
75% 1.050000
3.300000 2.200000 2.500000
max 7.700000 7.200000 7.800000 13.400000
```

value_counts

value_counts performs histogramming of a Series or DataFrame.

```
>>> state_gdp.region.value_counts()
SE 12
PL 7
NE 6
FW 6
MW 6
GL 5
RM 5
SW 4
dtype: int64
```

1.3 17.5 Graphics

pandas provides a set of useful plotting routines based on matplotlib which makes use of the structure of a DataFrame. Everything in pandas plot library is reproducible using matplotlib, although often at the cost of additional typing and code complexity (for example, axis labeling).

plot

plot is the main plotting method, and by default will produce a line graph of the data in a DataFrame. Calling plot on a DataFrame will plot all series using different colors and generate a legend. A number of keyword argument are available to affect the contents and appearance of the plot.

- style, a list of matplotlib styles, one for each series plotted. A dictionary using column names as keys and the line styles as values allows for further customization.
- title, a string containing the figure title.
- subplots, a Boolean indicating whether to plot using one subplot per series (True). The default it False.
- legend, a Boolean indicating whether to show a legend
- secondary_y, a Boolean indicating whether to plot a series on a secondary set of axis values. See the example below.
- ax, a matplotlib axis object to use for the plot. If no axis is provided, then a new axis is created.
- kind, a string, one of:
  - 'line', the default

- 'bar' to produce a bar chart. Can also use the keyword argument stacked=True to produce a stacked bar chart.
- 'barh' to produce a horizontal bar chart. Also support stacked=True.
- 'kde' or 'density' to produce a kernel density plot.

1.3.1 hist

hist produces a histogram plot, and is similar to producing a bar plot using the output of value_count.

1.3.2 boxplot

boxplot produces box plots of the series in a DataFrame.

1.3.3 scatter_plot

scatter_plot produce a scatter plot from two series in a DataFrame. Three inputs are required: the DataFrame, the column name for the x-axis data and the column name for the y-axis data. scatter_plot is located in pandas.tools.plotting.

1.3.4 scatter_matrix

scatter_matrix produces a n by n set of subplots where each subplot contains the bivariate scatter of two series. One input is required, the DataFrame. scatter_matrix is located in pandas.tools.plotting. By default, the diagonal elements are histograms, and the keyword argument diagonal='kde' produces a kernel density plot.

1.3.5 lag_plot

lag_plot produces a scatter plot of a series against its lagged value. The keyword argument lag chooses the lag used in the plot (default is 1).