Pandas

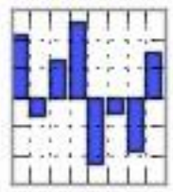
These slides are based on:

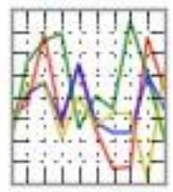
Pandas by Maik Röder

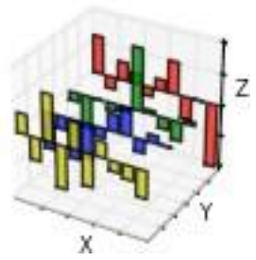
https://www.slideshare.net/maikroeder/pandas-16424935

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$







Maik Röder

Python Barcelona Meetup
7. February 2013

Python Consultant
maikroeder@gmail.com

Pandas

- Powerful and productive Python data analysis and management library
- **Pan**el **Da**ta **S**ystem

A.k.a. Excel for Python
(with a bit of databases)

- Open Sourced by AQR Capital Management, LLC in late 2009
- 30.000 lines of tested Python/Cython code
- Used in production in many companies

Pandas

- Rich data structures and functions to make working with structured data fast, easy, and expressive
- Built on top of Numpy with its high performance array-computing features
- flexible data manipulation capabilities of spreadsheets and relational databases
- Sophisticated indexing functionality
  - slice, dice, perform aggregations, select subsets of data

The ideal tool for data scientists

- Cleaning data
- Analyzing data
- Modeling data
- Organizing the results of the analysis into a form suitable for plotting or tabular display

Series

- one-dimensional array-like object

```
>>> s = Series((1,2,3,4,5))
```

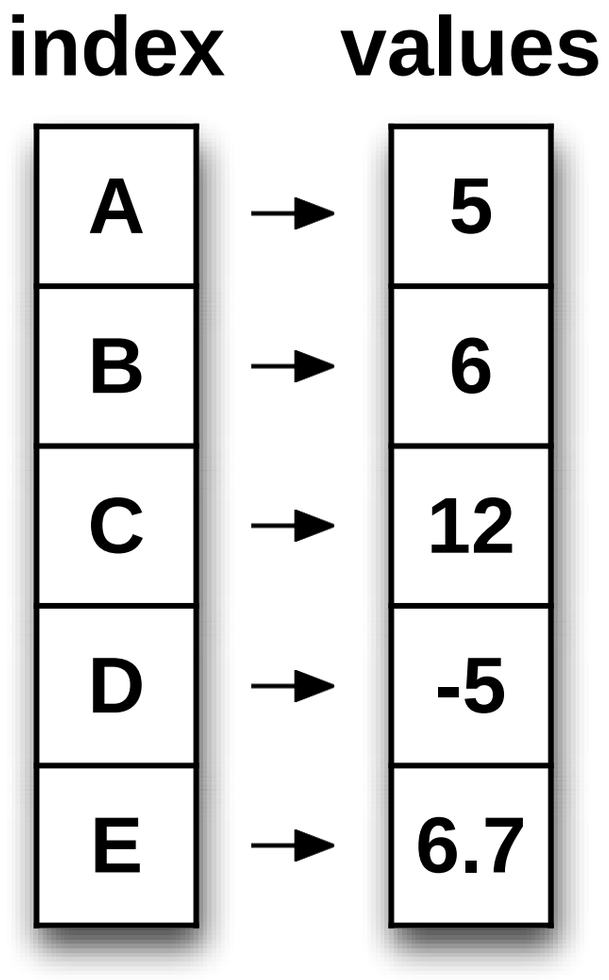
- Contains an array of data (of any Numpy data type)

```
>>> s.values
```

- Has an associated array of data labels, the *index* (Default index from 0 to N - 1)

```
>>> s.index
```

Series



- Subclass of `numpy.ndarray`
- Data: any type
- Index labels need not be ordered
- Duplicates are possible (but result in reduced functionality)

Series data structure

```
>>> import numpy
>>> randn = numpy.random.randn
>>> from pandas import *
>>> s = Series(randn(3),('a','b','c'))
>>> s
a   -0.889880
b    1.102135
c   -2.187296
>>> s.mean()
-0.65834710697853194
```

Series to/from dict

- Series to Python dict - **No more explicit order**

```
>>> dict(s)
{'a':  -0.88988001423312313,
 'c':  -2.1872960440695666,
 'b':  1.1021347373670938}
```

- Back to a Series with a **new Index from sorted dictionary keys**

```
>>> Series(dict(s))
a   -0.889880
b    1.102135
c   -2.187296
```

Reindexing labels

```
>>> s
a      -0.496848
b       0.607173
c      -1.570596
>>> s.index
Index([a, b, c],     dtype=object)
>>> s.reindex(['c','b','a'])
c      -1.570596
b       0.607173
a      -0.496848
```

Vectorization

```
>>> s + s
a   -1.779760
b    2.204269
c   -4.374592
```

- Series work with Numpy

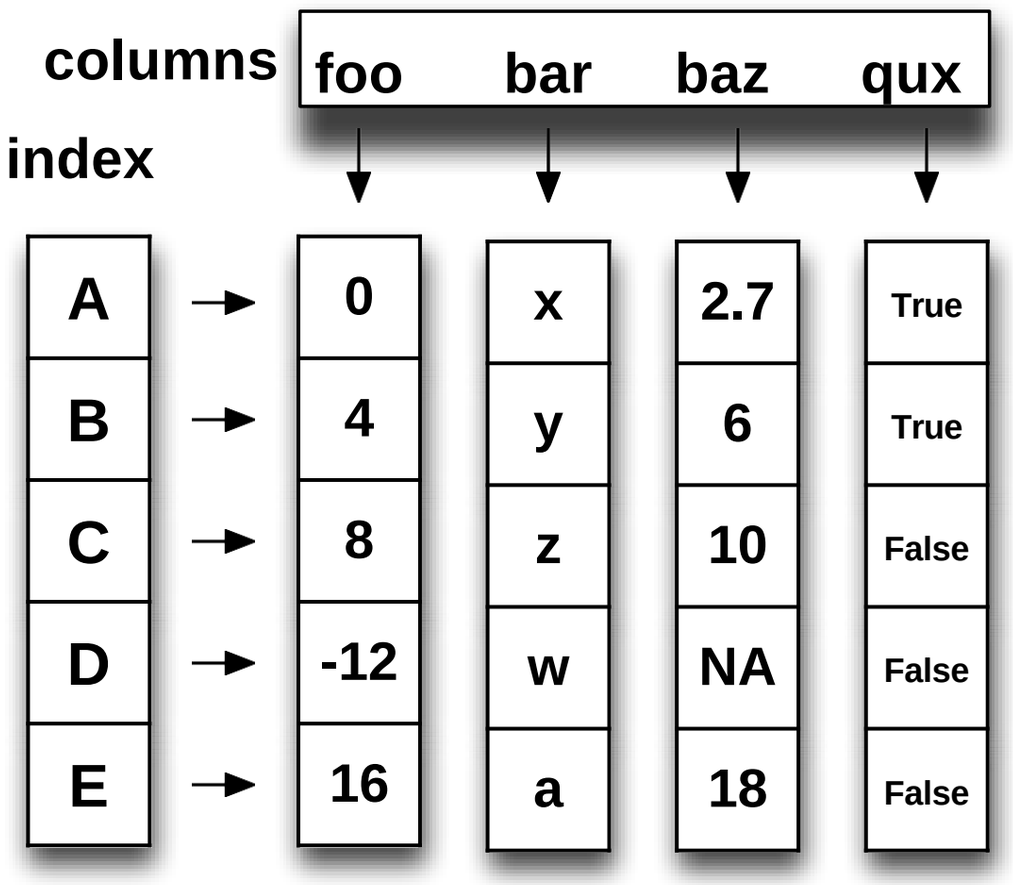
```
>>> numpy.exp(s)

a    0.410705
b    3.010586
c    0.112220
```

DataFrame

- Like data.frame in the statistical language/package R
- 2-dimensional tabular data structure
- Data manipulation with integrated indexing
- Support heterogeneous columns
- Homogeneous rows

DataFrame



- NumPy array-like
- Each column can have a different type
- Row and column index
- Size mutable: insert and delete columns

DataFrame

```
>>> d = {'one': s*s, 'two': s+s}
>>> DataFrame(d)
        one       two
a  0.791886 -1.779760
b  1.214701  2.204269
c  4.784264 -4.374592
>>> df.index
Index([a, b, c], dtype=object)
>>> df.columns
Index([one, two], dtype=objec)
```

Dataframe add column

- Add a third column

```
>>> df['three'] = s * 3
```

- It will share the existing index

```
>>> df
        one       two     three
a  0.791886 -1.779760 -2.669640
b  1.214701  2.204269  3.306404
c  4.784264 -4.374592 -6.561888
```

Access to columns

- Access by attribute

```
>>> df.one
   one
a  0.791886
b  1.214701
c  4.784264
```

- Access by dict like notation

```
>>> df['one']
   one
a  0.791886
b  1.214701
c  4.784264
```

Reindexing

```
>>>  df.reindex(['c','b','a'])
>>>  df
        one       two       three
c  4.784264 -4.374592 -6.561888
b  1.214701  2.204269  3.306404
a  0.791886 -1.779760 -2.669640
```

Drop entries from an axis

```
>>>  df.drop('c')
```

```
b  1.214701     2.204269     3.306404
a  0.791886    -1.779760    -2.669640
```

```
>>>  df.drop(['b','a'])
```

```
        one          two        three
c  4.784264    -4.374592    -6.561888
```

Descriptive statistics

```
>>> df.mean()
one          2.263617
two         -1.316694
three       -1.975041
```

- Also: count, sum, median, min, max, abs, prod, std, var, skew, kurt, quantile, cumsum, cumprod, cummax, cummin

Computational Tools

- Covariance

```
>>>  s1  =          Series(randn(1000))
>>>  s2  =          Series(randn(1000))
>>>  s1.cov(s2)
0.013973709323221539
```

- Also: pearson, kendall, spearman

I/O Operations

Read from

- csv, json, excel, html, SQL…

- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html#pandas.read_csv

Data alignment

- Binary operations are joins!
- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.align.html#pandas.Series.align

B	1
C	2
D	3
E	4

+

A	0
B	1
C	2
D	3

=

A	NA
B	2
C	4
D	6
E	NA

Data manipulations

- Join / merge – database-like syntax
- Get_dummies (one hot vector from categorical data)

Aggregated Statistics

- Group by
- Describe

Plotting

DataFrame.plot.plotType()
- hist, pie, density, box,…

This and much more...

- Group by: split-apply-combine
- Merge, join and aggregate
- Reshaping and Pivot Tables
- Time Series / Date functionality
- Plotting with matplotlib
- IO Tools (Text, CSV, HDF5, ...)
- Sparse data structures

Resources

- http://pypi.python.org/pypi/pandas
- http://code.google.com/p/pandas