

Pandas数据特征分析

DV08

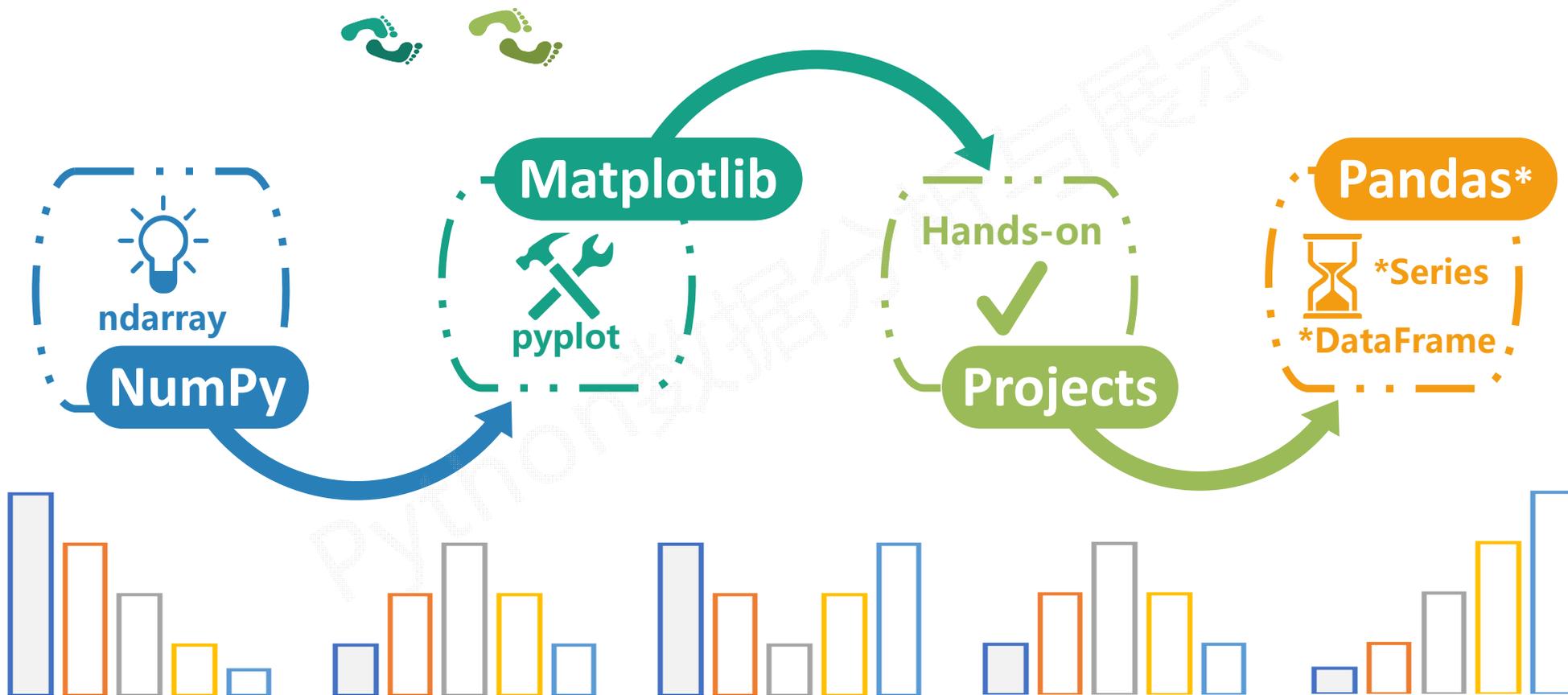


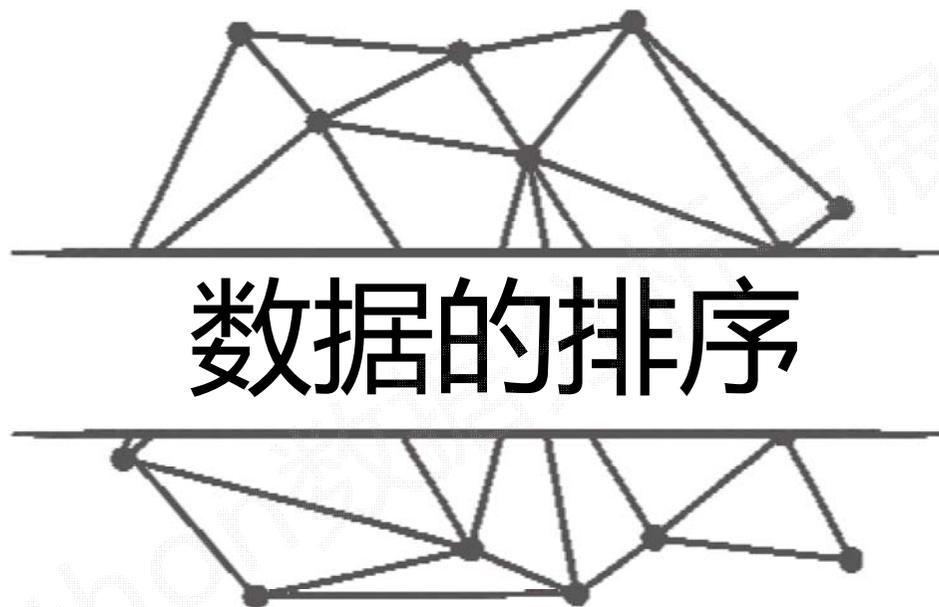
嵩天

www.python123.org

Python数据分析与展示

掌握表示、清洗、统计和展示数据的能力





Python 数据可视化展示

对一组数据的理解

3.1413 3.1404
3.1398 3.1401 3.1376
3.1349

一组数据

表达一个或多个含义

摘要



基本统计（含排序）

分布/累计统计

数据特征

相关性、周期性等

数据挖掘（形成知识）

数据形成有损特征的过程

Pandas库的数据排序

.sort_index()方法在指定轴上根据索引进行排序，默认升序

.sort_index(axis=0, ascending=True)

```
In [358]: import pandas as pd
```

```
In [359]: import numpy as np
```

```
In [360]: b = pd.DataFrame(np.arange(20).reshape(4,5), index=['c','a','d','b'])
```

```
In [361]: b
```

```
Out[361]:
```

	0	1	2	3	4
c	0	1	2	3	4
a	5	6	7	8	9
d	10	11	12	13	14
b	15	16	17	18	19

```
In [362]: b.sort_index()
```

```
Out[362]:
```

	0	1	2	3	4
a	5	6	7	8	9
b	15	16	17	18	19
c	0	1	2	3	4
d	10	11	12	13	14



```
In [363]: b.sort_index(ascending=False)
```

```
Out[363]:
```

	0	1	2	3	4
d	10	11	12	13	14
c	0	1	2	3	4
b	15	16	17	18	19
a	5	6	7	8	9

默认在axis=0轴上进行排序！

Pandas库的数据排序

`.sort_index(axis=0, ascending=True)`

```
In [358]: import pandas as pd
```

```
In [359]: import numpy as np
```

```
In [360]: b = pd.DataFrame(np.arange(20).reshape(4,5), index=['c','a','d','b'])
```

```
In [361]: b
```

```
Out[361]:
```

	0	1	2	3	4
c	0	1	2	3	4
a	5	6	7	8	9
d	10	11	12	13	14
b	15	16	17	18	19

```
In [366]: c = b.sort_index(axis=1, ascending=False)
```

```
In [367]: c
```

```
Out[367]:
```

	4	3	2	1	0
c	4	3	2	1	0
a	9	8	7	6	5
d	14	13	12	11	10
b	19	18	17	16	15

```
In [368]: c = c.sort_index()
```

```
In [369]: c
```

```
Out[369]:
```

	4	3	2	1	0
a	9	8	7	6	5
b	19	18	17	16	15
c	4	3	2	1	0
d	14	13	12	11	10



Pandas库的数据排序

.sort_values()方法在指定轴上根据数值进行排序，默认升序

```
Series.sort_values(axis=0, ascending=True)
```

```
DataFrame.sort_values(by, axis=0, ascending=True)
```

by : axis轴上的某个索引或索引列表

Pandas库的数据排序

`.sort_values(by, axis=0, ascending=True)`

```
In [358]: import pandas as pd
```

```
In [359]: import numpy as np
```

```
In [360]: b = pd.DataFrame(np.arange(20).reshape(4,5), index=['c','a','d','b'])
```

```
In [361]: b
```

```
Out[361]:
```

	0	1	2	3	4
c	0	1	2	3	4
a	5	6	7	8	9
d	10	11	12	13	14
b	15	16	17	18	19



```
In [386]: c = b.sort_values(2, ascending=False)
```

```
In [387]: c
```

```
Out[387]:
```

	0	1	2	3	4
b	15	16	17	18	19
d	10	11	12	13	14
a	5	6	7	8	9
c	0	1	2	3	4

```
In [388]: c = c.sort_values('a', axis=1, ascending=False)
```

```
In [389]: c
```

```
Out[389]:
```

	4	3	2	1	0
b	19	18	17	16	15
d	14	13	12	11	10
a	9	8	7	6	5
c	4	3	2	1	0

Pandas库的数据排序

NaN统一放到排序末尾

```
In [395]: import pandas as pd
```

```
In [396]: import numpy as np
```

```
In [397]: a = pd.DataFrame(np.arange(12).reshape(3,4), index=['a', 'b', 'c'])
```

```
In [398]: a
```

```
Out[398]:
```

```
   0  1  2  3
a  0  1  2  3
b  4  5  6  7
c  8  9 10 11
```

```
In [399]: b = pd.DataFrame(np.arange(20).reshape(4,5), index=['c', 'a', 'd', 'b'])
```

```
In [400]: b
```

```
Out[400]:
```

```
   0  1  2  3  4
c  0  1  2  3  4
a  5  6  7  8  9
d 10 11 12 13 14
b 15 16 17 18 19
```

```
In [401]: c = a + b
```

```
In [402]: c
```

```
Out[402]:
```

```
   0  1  2  3  4
a  5.0  7.0  9.0 11.0 NaN
b 19.0 21.0 23.0 25.0 NaN
c  8.0 10.0 12.0 14.0 NaN
d  NaN  NaN  NaN  NaN NaN
```

```
In [403]: c.sort_values(2, ascending = False)
```

```
Out[403]:
```

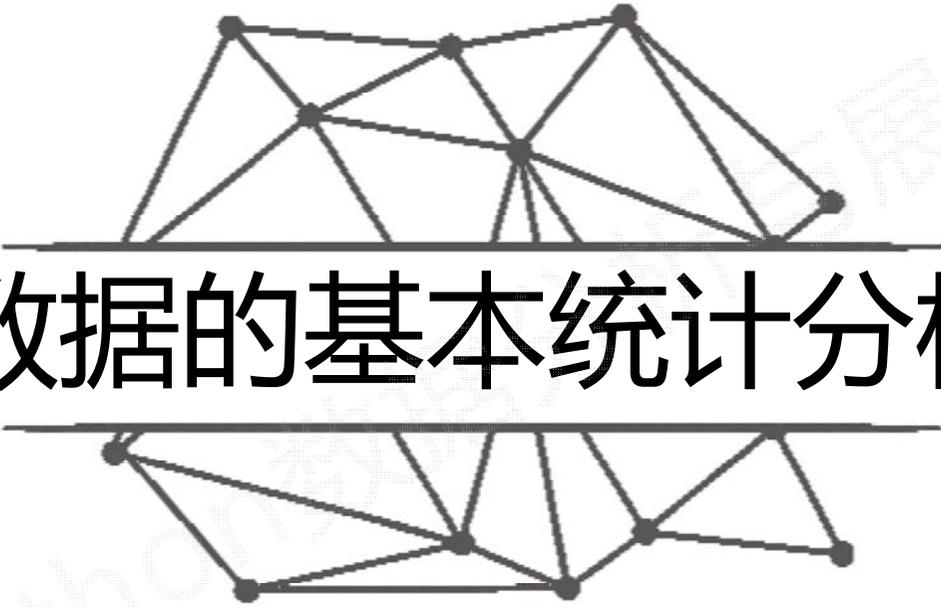
```
   0  1  2  3  4
b 19.0 21.0 23.0 25.0 NaN
c  8.0 10.0 12.0 14.0 NaN
a  5.0  7.0  9.0 11.0 NaN
d  NaN  NaN  NaN  NaN NaN
```

```
In [404]: c.sort_values(2, ascending = True)
```

```
Out[404]:
```

```
   0  1  2  3  4
a  5.0  7.0  9.0 11.0 NaN
c  8.0 10.0 12.0 14.0 NaN
b 19.0 21.0 23.0 25.0 NaN
d  NaN  NaN  NaN  NaN NaN
```





数据的基本统计分析

基本的统计分析函数

适用于Series和DataFrame类型

方法	说明
<code>.sum()</code>	计算数据的总和，按0轴计算，下同
<code>.count()</code>	非NaN值的数量
<code>.mean()</code> <code>.median()</code>	计算数据的算术平均值、算术中位数
<code>.var()</code> <code>.std()</code>	计算数据的方差、标准差
<code>.min()</code> <code>.max()</code>	计算数据的最小值、最大值

基本的统计分析函数

适用于Series类型

方法	说明
<code>.argmin()</code> <code>.argmax()</code>	计算数据最大值、最小值所在位置的索引位置（自动索引）
<code>.idxmin()</code> <code>.idxmax()</code>	计算数据最大值、最小值所在位置的索引（自定义索引）

基本的统计分析函数

适用于Series和DataFrame类型

方法	说明
<code>.describe()</code>	针对0轴（各列）的统计汇总

基本的统计分析函数

```
In [425]: import pandas as pd
```

```
In [426]: a = pd.Series([9, 8, 7, 6], index=['a','b','c','d'])
```

```
In [427]: a
```

```
Out[427]:
```

```
a    9  
b    8  
c    7  
d    6
```

```
dtype: int64
```

```
In [428]: a.describe()
```

```
Out[428]:
```

```
count    4.000000  
mean     7.500000  
std      1.290994  
min      6.000000  
25%     6.750000  
50%     7.500000  
75%     8.250000  
max      9.000000
```

```
dtype: float64
```



```
In [431]: type(a.describe())
```

```
Out[431]: pandas.core.series.Series
```

```
In [432]: a.describe()['count']
```

```
Out[432]: 4.0
```

```
In [433]: a.describe()['max']
```

```
Out[433]: 9.0
```

基本的统计分析函数

```
In [434]: import pandas as pd
```

```
In [435]: import numpy as np
```

```
In [436]: b = pd.DataFrame(np.arange(20).reshape(4,5), index=['c', 'a', 'd', 'b'])
```

```
In [437]: b.describe()
```

```
Out[437]:
```

	0	1	2	3	4
count	4.000000	4.000000	4.000000	4.000000	4.000000
mean	7.500000	8.500000	9.500000	10.500000	11.500000
std	6.454972	6.454972	6.454972	6.454972	6.454972
min	0.000000	1.000000	2.000000	3.000000	4.000000
25%	3.750000	4.750000	5.750000	6.750000	7.750000
50%	7.500000	8.500000	9.500000	10.500000	11.500000
75%	11.250000	12.250000	13.250000	14.250000	15.250000
max	15.000000	16.000000	17.000000	18.000000	19.000000



```
In [443]: type(b.describe())
```

```
Out[443]: pandas.core.frame.DataFrame
```

```
In [444]: b.describe().ix['max']
```

```
Out[444]:
```

```
0    15.0
```

```
1    16.0
```

```
2    17.0
```

```
3    18.0
```

```
4    19.0
```

```
Name: max, dtype: float64
```

loc替换ix!

```
In [445]: b.describe()[2]
```

```
Out[445]:
```

```
count    4.000000
```

```
mean     9.500000
```

```
std      6.454972
```

```
min      2.000000
```

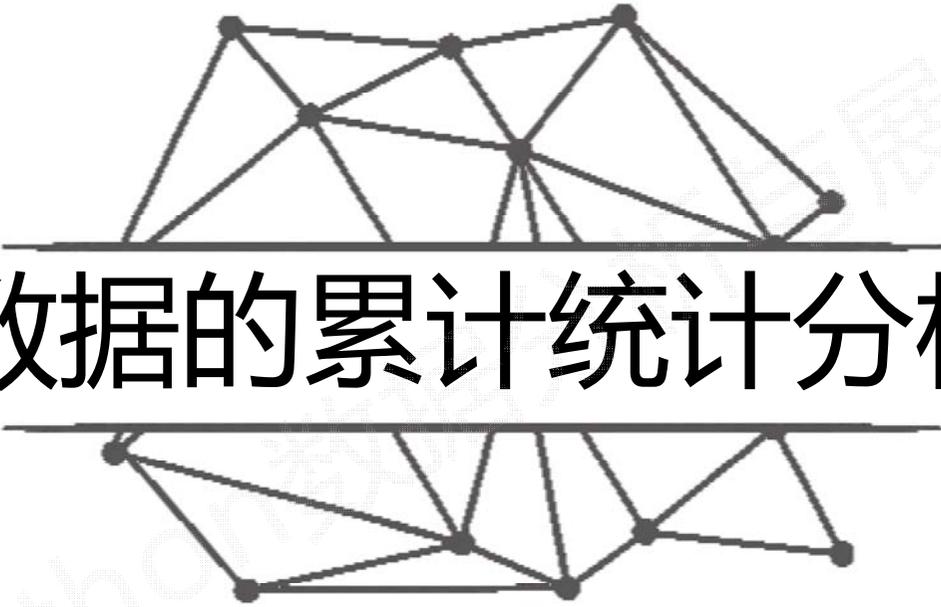
```
25%     5.750000
```

```
50%     9.500000
```

```
75%    13.250000
```

```
max     17.000000
```

```
Name: 2, dtype: float64
```



数据的累计统计分析

累计统计分析函数

适用于Series和DataFrame类型，累计计算

方法	说明
<code>.cumsum()</code>	依次给出前1、2、...、n个数的和
<code>.cumprod()</code>	依次给出前1、2、...、n个数的积
<code>.cummax()</code>	依次给出前1、2、...、n个数的最大值
<code>.cummin()</code>	依次给出前1、2、...、n个数的最小值

默认还是axis=0轴方向！

累计统计分析函数

```
In [448]: import pandas as pd
```

```
In [449]: import numpy as np
```

```
In [450]: b = pd.DataFrame(np.arange(20).reshape(4,5), index=['c','a','d','b'])
```

```
In [451]: b
```

```
Out[451]:
```

	0	1	2	3	4
c	0	1	2	3	4
a	5	6	7	8	9
d	10	11	12	13	14
b	15	16	17	18	19



```
In [452]: b.cumsum()
```

```
Out[452]:
```

	0	1	2	3	4
c	0	1	2	3	4
a	5	7	9	11	13
d	15	18	21	24	27
b	30	34	38	42	46

```
In [453]: b.cumprod()
```

```
Out[453]:
```

	0	1	2	3	4
c	0	1	2	3	4
a	0	6	14	24	36
d	0	66	168	312	504
b	0	1056	2856	5616	9576

```
In [454]: b.cummin()
```

```
Out[454]:
```

	0	1	2	3	4
c	0	1	2	3	4
a	0	1	2	3	4
d	0	1	2	3	4
b	0	1	2	3	4

```
In [455]: b.cummax()
```

```
Out[455]:
```

	0	1	2	3	4
c	0	1	2	3	4
a	5	6	7	8	9
d	10	11	12	13	14
b	15	16	17	18	19

累计统计分析函数

适用于Series和DataFrame类型，滚动计算（窗口计算）

w指定窗口大小

方法	说明
<code>.rolling(w).sum()</code>	依次计算相邻w个元素的和
<code>.rolling(w).mean()</code>	依次计算相邻w个元素的算术平均值
<code>.rolling(w).var()</code>	依次计算相邻w个元素的方差
<code>.rolling(w).std()</code>	依次计算相邻w个元素的标准差
<code>.rolling(w).min()</code> <code>.max()</code>	依次计算相邻w个元素的最小值和最大值

累计统计分析函数

```
In [448]: import pandas as pd
```

```
In [449]: import numpy as np
```

```
In [450]: b = pd.DataFrame(np.arange(20).reshape(4,5), index=['c', 'a', 'd', 'b'])
```

```
In [451]: b
```

```
Out[451]:
```

	0	1	2	3	4
c	0	1	2	3	4
a	5	6	7	8	9
d	10	11	12	13	14
b	15	16	17	18	19

```
In [465]: b.rolling(2).sum()
```

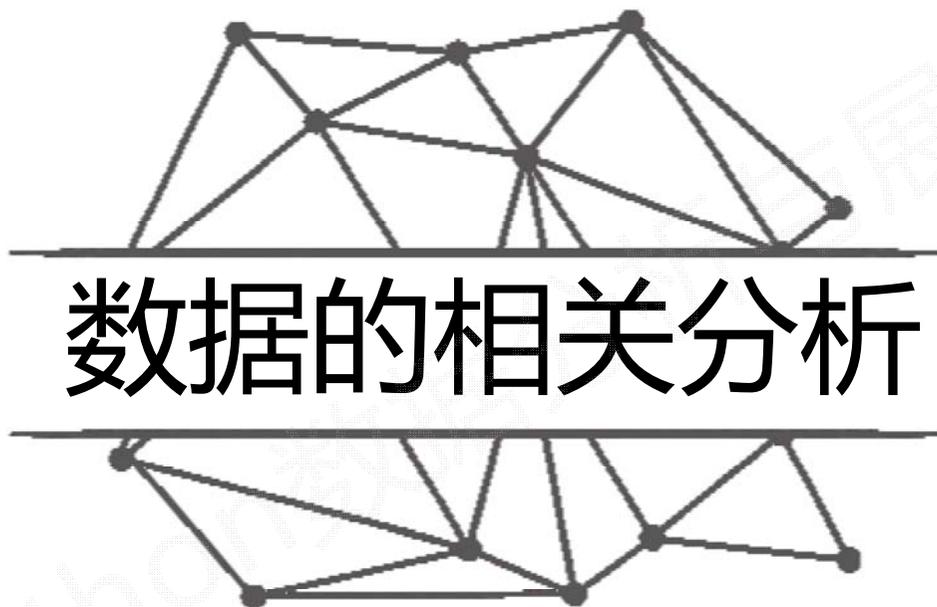
```
Out[465]:
```

	0	1	2	3	4
c	NaN	NaN	NaN	NaN	NaN
a	5.0	7.0	9.0	11.0	13.0
d	15.0	17.0	19.0	21.0	23.0
b	25.0	27.0	29.0	31.0	33.0

```
In [466]: b.rolling(3).sum()
```

```
Out[466]:
```

	0	1	2	3	4
c	NaN	NaN	NaN	NaN	NaN
a	NaN	NaN	NaN	NaN	NaN
d	15.0	18.0	21.0	24.0	27.0
b	30.0	33.0	36.0	39.0	42.0



数据的相关分析

Python 数据可视化展示

相关分析

两个事物，表示为 X 和 Y ，如何判断它们之间的存在相关性？

相关性

- X 增大， Y 增大，两个变量**正相关**
- X 增大， Y 减小，两个变量**负相关**
- X 增大， Y 无视，两个变量**不相关**

协方差

两个事物，表示为X和Y，如何判断它们之间的存在相关性？

$$\text{cov}(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{n - 1}$$

- 协方差 >0 ，X和Y正相关
- 协方差 <0 ，X和Y负相关
- 协方差 $=0$ ，X和Y独立无关

Pearson相关系数

两个事物，表示为X和Y，如何判断它们之间的存在相关性？

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

- 0.8-1.0 极强相关
- 0.6-0.8 强相关
- 0.4-0.6 中等程度相关
- 0.2-0.4 弱相关
- 0.0-0.2 极弱相关或无相关

r 取值范围[-1, 1]

相关分析函数

适用于Series和DataFrame类型

方法	说明
<code>.cov()</code>	计算协方差矩阵
<code>.corr()</code>	计算相关系数矩阵, Pearson、Spearman、Kendall等系数

实例：房价增幅与M2增幅的相关性

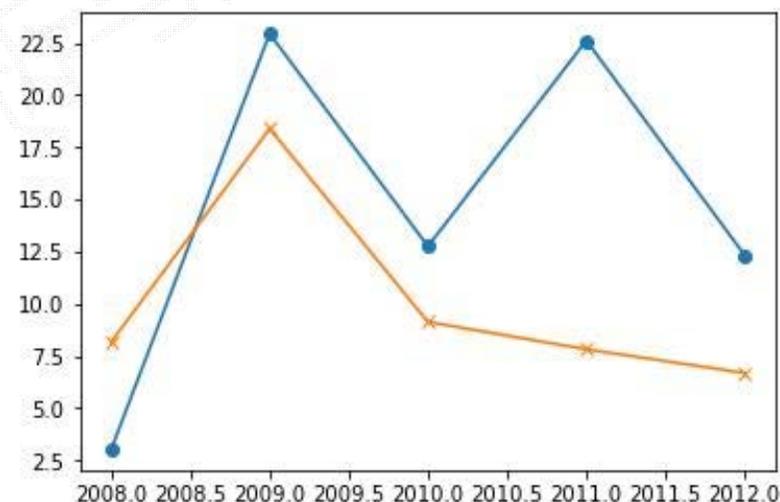
```
In [484]: import pandas as pd
```

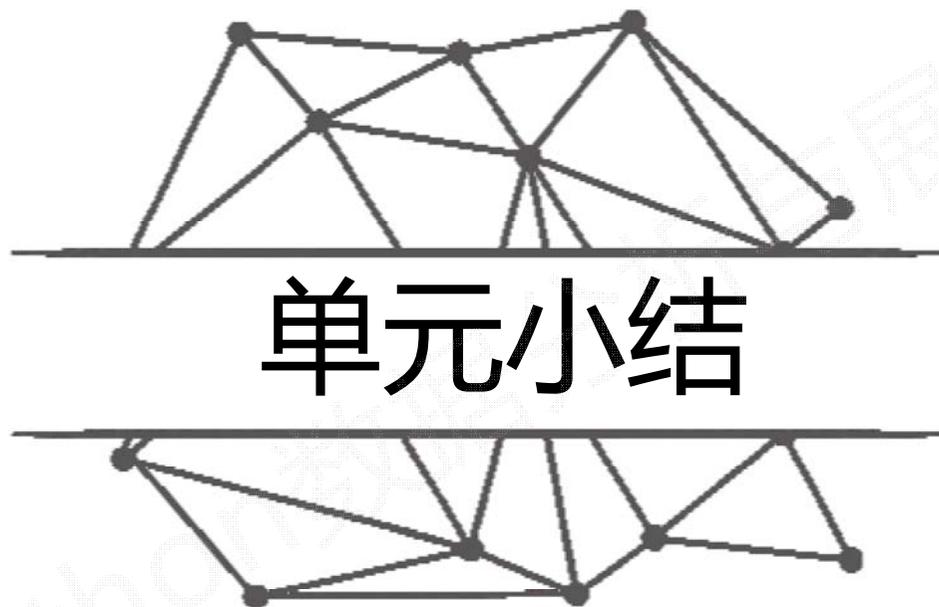
```
In [485]: hprice = pd.Series([3.04, 22.93, 12.75, 22.6, 12.33], index=['2008',  
'2009', '2010', '2011', '2012'])
```

```
In [486]: m2 = pd.Series([8.18, 18.38, 9.13, 7.82, 6.69], index=['2008', '2009',  
'2010', '2011', '2012'])
```

```
In [487]: hprice.corr(m2)
```

```
Out[487]: 0.5239439145220387
```





Pandas数据特征分析

一组数据的摘要

排序

`.sort_index()` `.sort_values()`

基本统计函数

`.describe()`

累计统计函数

`.cum*()` `.rolling().*()`

相关性分析

`.corr()` `.cov()`