



Data retrieval and preprocessing of Python

Python便捷数据获取与预处理

Dazhuang@NJU

Department of Computer Science and Technology
Department of University Basic Computer Teaching

数据处理过程





用Python玩转数据

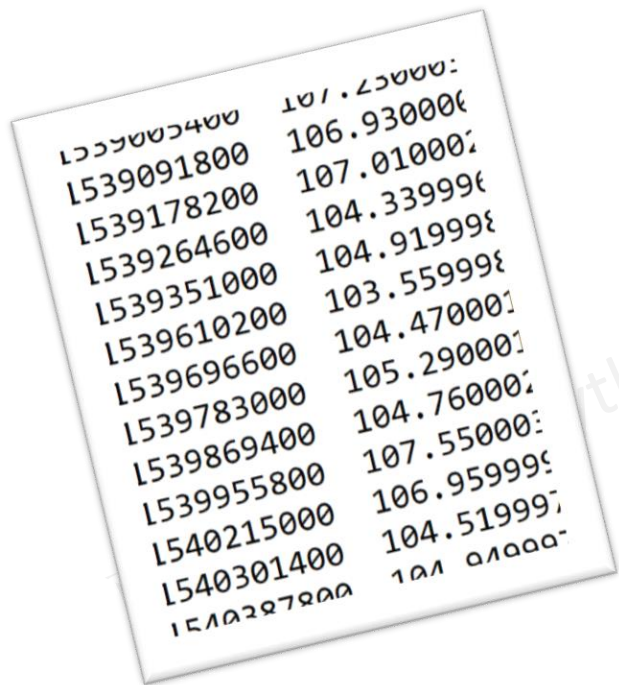
便捷数据获取

用Python获取数据

本地数据如何获取？

文件的打开，读写和关闭

- 文件打开
- 读文件
- 写文件
- 文件关闭

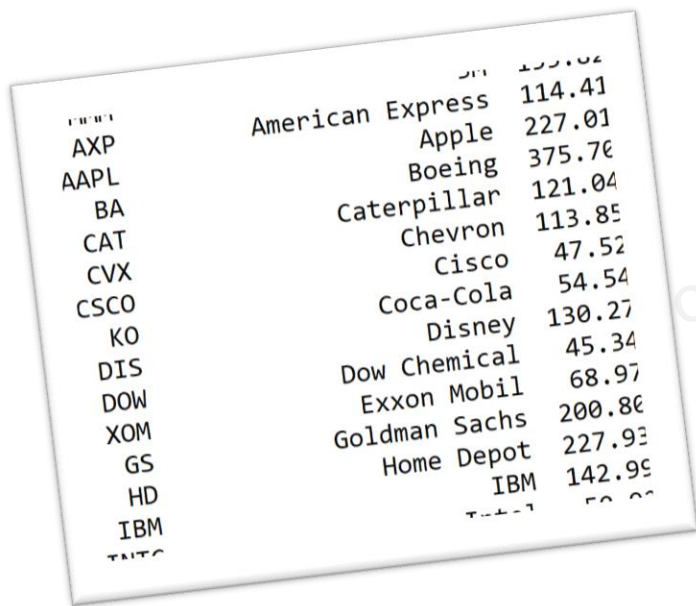


用Python获取数据

网络数据如何获取（爬取）？

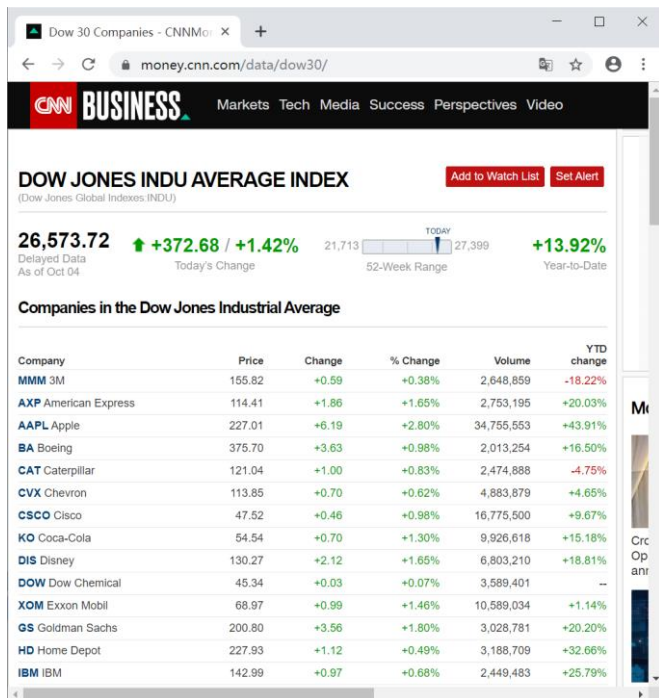
抓取网页，解析网页内容

- 抓取
 - urllib内建模块
 - urllib.request
 - Requests第三方库
 - Scrapy框架
- 解析
 - BeautifulSoup库
 - re模块

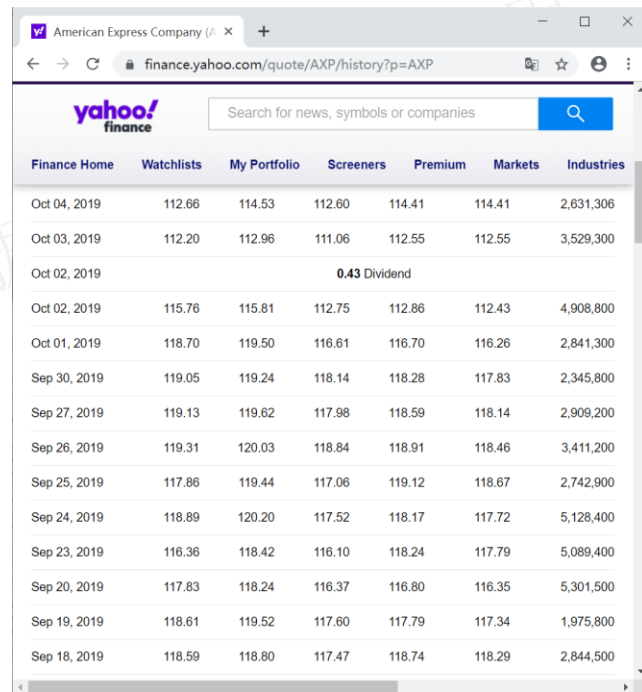


AXP	American Express	114.41
AAPL	Apple	227.01
BA	Boeing	375.70
CAT	Caterpillar	121.04
CVX	Chevron	113.85
CSCO	Cisco	47.52
KO	Coca-Cola	54.54
DIS	Disney	130.27
DOW	Dow Chemical	45.34
XOM	Exxon Mobil	68.97
GS	Goldman Sachs	200.80
HD	Home Depot	227.93
IBM	IBM	142.95

道指成分股数据



dji



quotes

数据形式

7

	code	name	price
0	MMM	3M	155.82
1	AXP	American Express	114.41
2	AAPL	Apple	227.01
3	BA	Boeing	375.70
4	CAT	Caterpillar	121.04
5	CVX	Chevron	113.85
6	CSCO	Cisco	47.52
7	KO	Coca-Cola	54.54
8	DIS	Disney	130.27
9	DOW	Dow Chemical	45.34
10	XOM	Exxon Mobil	68.97
11	GS	Goldman Sachs	200.80
12	HD	Home Depot	227.93
13	IBM	IBM	142.99
14	INTC	Intel	50.92
15	JNJ	Johnson & Johnson	133.66
16	JPM	JPMorgan Chase	114.62
17	MCD	McDonald's	211.69
18	MRK	Merck	85.00
19	MSFT	Microsoft	138.12
20	NKE	Nike	93.07
21	PFE	Pfizer	35.93
22	PG	Procter & Gamble	124.00
23	TRV	Travelers Companies Inc	144.96
24	UTX	United Technologies	133.21
25	UNH	UnitedHealth	219.80
26	VZ	Verizon	59.90
27	V	Visa	175.98
28	WMT	Wal-Mart	118.16
29	WBA	Walgreen	52.97

djidf

	close	date	high	low	open	volume
0	106.989998	1539005400	107.230003	105.570000	106.629997	2723400
1	106.660004	1539091800	106.930000	105.940002	106.300003	2604400
2	103.570000	1539178200	107.010002	103.519997	106.959999	4555600
3	101.580002	1539264600	104.339996	101.550003	103.220001	6069300
4	103.000000	1539351000	104.919998	101.709999	104.309998	4855900
5	102.620003	1539610200	103.559998	102.209999	102.849998	2780900
6	104.269997	1539696600	104.470001	102.690002	103.070000	3121000
7	104.339996	1539783000	105.290001	103.919998	104.330002	3792400
8	102.839996	1539869400	104.760002	102.290001	104.529999	4538200
9	106.730003	1539955800	107.550003	104.059998	104.059998	5726300
10	104.510002	1540215000	106.959999	104.449997	106.610001	5003100
11	104.379997	1540301400	104.519997	101.839996	102.410004	4223800
12	101.839996	1540387800	104.949997	101.510002	104.430000	4056700
13	103.599998	1540474200	104.169998	101.800003	102.480003	3378900
14	101.250000	1540560600	102.660004	100.139999	102.540001	5395700
15	101.190002	1540819800	103.250000	100.040001	102.470001	4238700
16	102.080002	1540906200	102.389999	100.410004	101.599998	3778200
17	102.730003	1540992600	103.709999	102.550003	103.059998	4511300
18	104.040001	1541079000	104.269997	103.019997	103.260002	2786800
19	103.709999	1541165400	105.050003	102.889999	104.930000	4322200

quotesdf

直接下载数据



是否能够简单方便并且快速的方式获得财经网站上公司股票的历史数据？

Time Period: Oct 05, 2018 - Oct 05, 2019 ▾

Show: Historical Prices ▾

Frequency: Daily ▾

Apply

Currency in USD

[Download Data](#)

Date	Open	High	Low	Close	Adj Close	Volume
2018/10/5	108.06	108.47	106.72	107.23	105.6803	2399800
2018/10/8	106.63	107.23	105.57	106.99	105.4437	2723400
2018/10/9	106.3	106.93	105.94	106.66	105.1185	2604400
2018/10/10	106.96	107.01	103.52	103.57	102.0732	4555600
2018/10/11	103.22	104.34	101.55	101.58	100.1119	6069300
2018/10/12	104.31	104.92	101.71	103	101.5114	4855900



```
# Filename: quotes_fromcsv.py
import pandas as pd
quotesdf = pd.read_csv('axp.csv')
print(quotesdf)
```


csv格式数据存取



将美国通用公司近一年来的股票基本信息存入文件stockAXP.csv中



```
# Filename: to_csv.py
import pandas as pd
...
quotes = retrieve_quotes_historical('AXP')
df = pd.DataFrame(quotes)
df.to_csv('stockAXP.csv')
```

excel数据存取

File

Filename: to_excel.py

...

```
quotes = retrieve_quotes_historical('AXP')
```

```
df = pd.DataFrame(quotes)
```

```
df.to_excel('stockAXP.xlsx', sheet_name = 'AXP')
```



File

Filename: read_excel.py

...

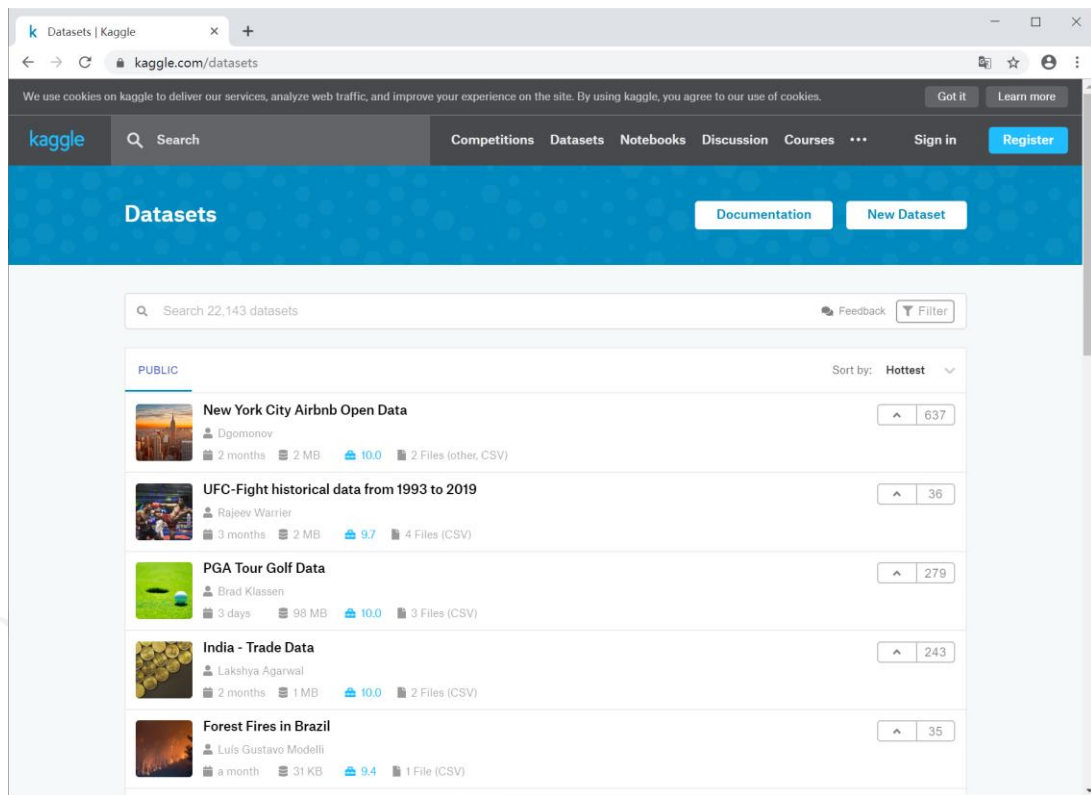
```
df = pd.read_excel('stockAXP.xlsx', index_col = 'date')
```

```
print(df['close'][:3])
```

```
0    106.989998
1    106.660004
2    103.570000
Name: close, dtype: float64
```

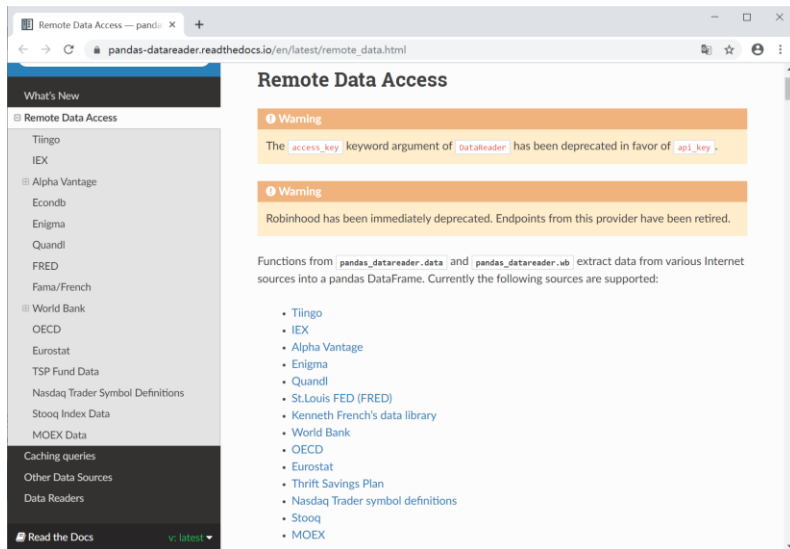
直接下载数据

11



API获取数据

12



The screenshot shows the pandas-datareader website. The left sidebar lists various data sources under 'Remote Data Access', including Tiingo, IEX, Alpha Vantage, Econdb, Enigma, Quandl, FRED, Fama/French, World Bank, OECD, Eurostat, TSP Fund Data, Nasdaq Trader Symbol Definitions, Stooq Index Data, MOEX Data, Caching queries, Other Data Sources, and Data Readers. The main content area, titled 'Remote Data Access', displays two warning messages: one about the deprecation of the 'access_key' keyword argument in favor of 'api_key', and another about the deprecation of Robinhood endpoints. Below the warnings, it states that functions from 'pandas_datareader.data' and 'pandas_datareader.web' extract data from various Internet sources into a pandas DataFrame, and lists the currently supported sources: Tiingo, IEX, Alpha Vantage, Enigma, Quandl, St.Louis FED (FRED), Kenneth French's data library, World Bank, OECD, Eurostat, Thrift Savings Plan, Nasdaq Trader symbol definitions, Stooq, and MOEX.

Source

```
>>> import pandas_datareader.data as web
>>> f = web.DataReader('AXP', 'stooq')
>>> f.head(5)
```

Date	Open	High	Low	Close	Volume
------	------	------	-----	-------	--------

2019-10-04	112.62	114.530	112.60	114.41	2753195
2019-10-03	112.52	112.955	111.06	112.55	3549232
2019-10-02	115.76	115.810	112.75	112.86	4931560
2019-10-01	118.70	119.500	116.61	116.70	2857528
2019-09-30	119.05	119.240	118.14	118.28	2353731

sklearn模块的datasets



```
>>> from sklearn import datasets
```

```
>>> iris = datasets.load_iris()
```

```
>>> iris.feature_names
```

```
['sepal length (cm)',
```

```
'sepal width (cm)',
```

```
'petal length (cm)',
```

```
'petal width (cm)']
```

```
>>> iris.data
```

```
array([[5.1, 3.5, 1.4, 0.2],
```

```
       [4.9, 3. , 1.4, 0.2],
```

```
       [4.7, 3.2, 1.3, 0.2],
```

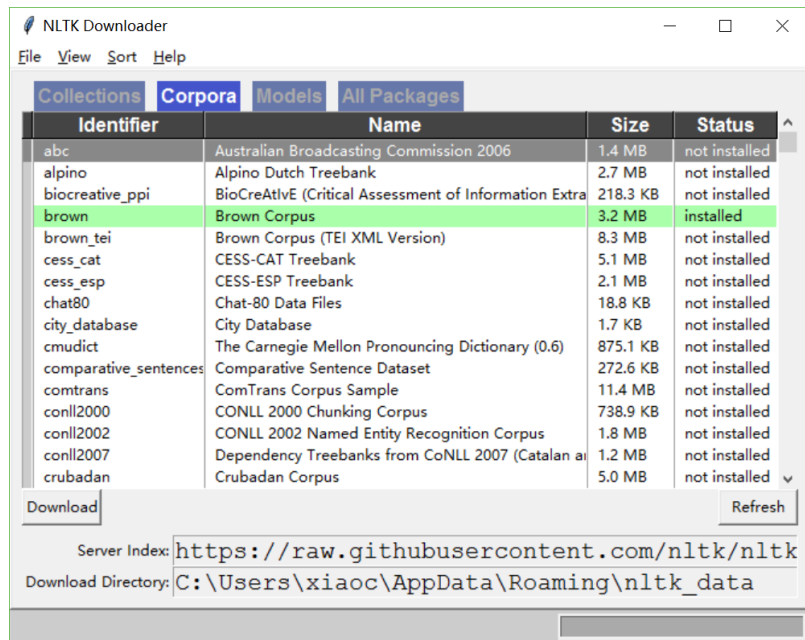
```
...
```

```
       [6.5, 3. , 5.2, 2. ],
```

```
       [6.2, 3.4, 5.4, 2.3],
```

```
       [5.9, 3. , 5.1, 1.8]])
```

NLTK语料库





```
>>> from nltk.corpus import gutenbergl brown
>>> import nltk
>>> print(gutenberg.fileids())
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-kjv.txt', 'blake-
poems.txt', 'bryant-stories.txt', 'burgess-busterbrown.txt', 'carroll-alice.txt',
'chesterton-ball.txt', 'chesterton-brown.txt', 'chesterton-thursday.txt', 'edgeworth-
parents.txt', 'melville-moby_dick.txt', 'milton-paradise.txt', 'shakespeare-caesar.txt',
'shakespeare-hamlet.txt', 'shakespeare-macbeth.txt', 'whitman-leaves.txt']
>>> texts = gutenberg.words('shakespeare-hamlet.txt')
>>> print(texts)
['I', 'The', 'Tragedie', 'of', 'Hamlet', 'by', ...]
```

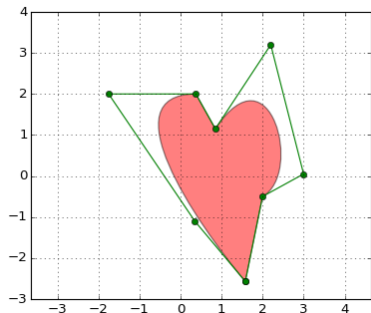
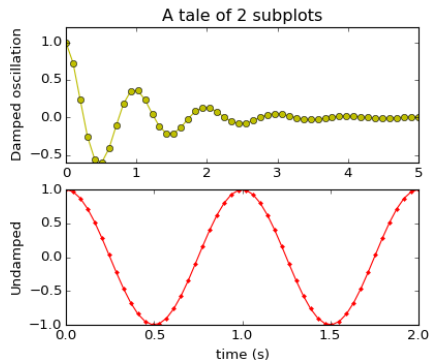
2

用Python玩转数据

PYTHON

绘图基础

Matplotlib绘图



- ## Matplotlib绘图

最著名Python绘图库,
主要用于二维绘图

- 画图质量高
- 方便快捷的绘图模块
 - 绘图API——pyplot模块

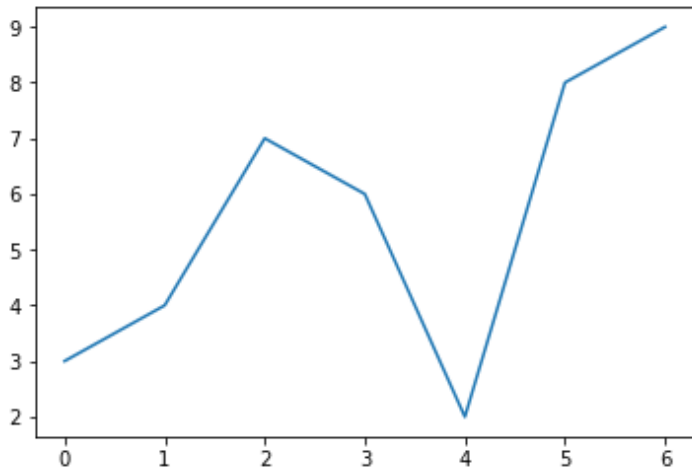
折线图

18



```
>>> import matplotlib.pyplot as plt  
>>> plt.plot([3, 4, 7, 6, 2, 8, 9])
```

```
plt.plot(range(7), [3, 4, 7, 6, 2, 8, 9])
```



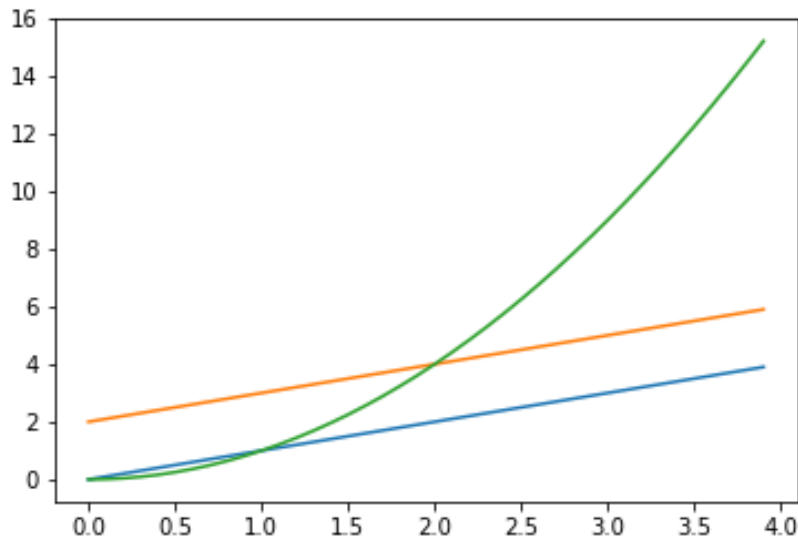
折线图-绘制多组数据

19

- NumPy数组也可以作为Matplotlib的参数
- 多组成对数据绘图

Source

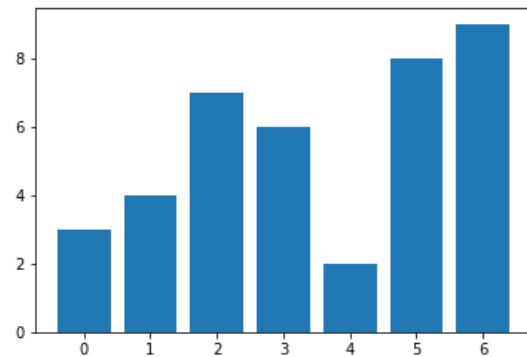
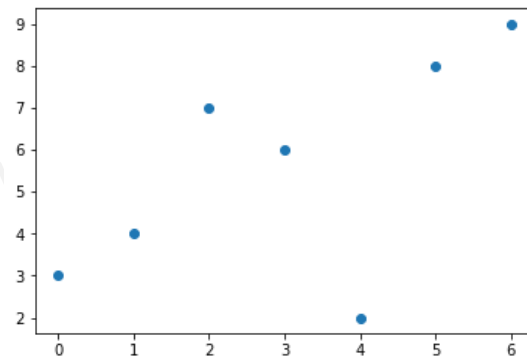
```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> t=np.arange(0., 4., 0.1)
>>> plt.plot(t, t, t, t+2, t, t**2)
```



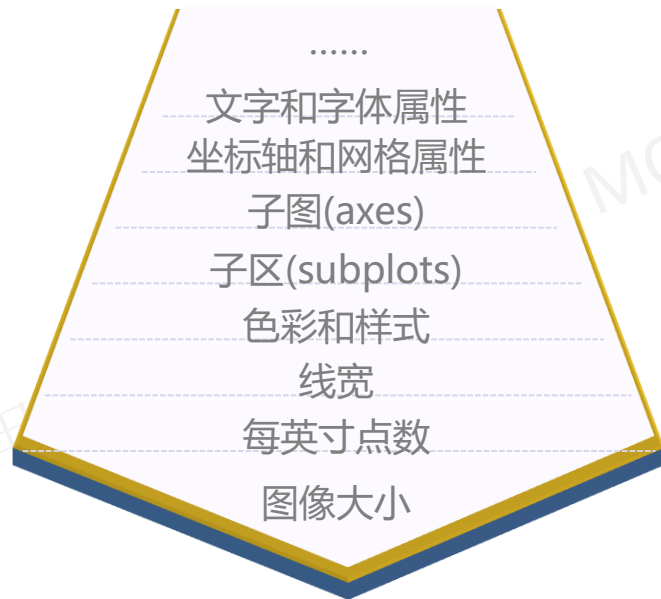
绘制其他类型的图

S_{ource}

```
>>> import matplotlib.pyplot as plt  
>>> plt.scatter(range(7), [3, 4, 7, 6, 2, 8, 9])  
>>> plt.bar(range(7), [3, 4, 7, 6, 2, 8, 9])
```



Matplotlib属性

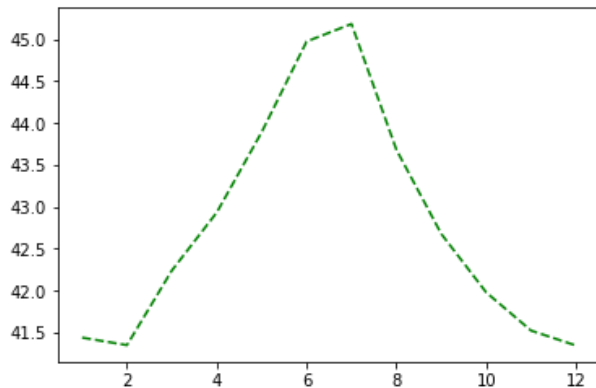


Matplotlib可以控制的默认属性

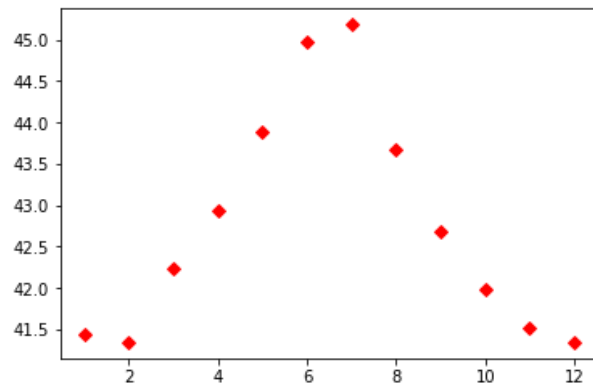
色彩和样式



绘图颜色
和线条类
型和样式
可以更改
吗?



```
plt.plot(x, y, 'g--')
```



```
plt.plot(x, y, 'rD')
```

色彩和样式

符号	颜色
b	blue
g	green
r	red
c	cyan
m	magenta
Y	yellow
k	black
w	white

线型	描述
'-'	solid
'--'	dashed
'-.'	dash_dot
':'	dotted
'None'	draw nothing
''	draw nothing
''	draw nothing

标记	描述
"o"	circle
"v"	triangle_down
"s"	square
"p"	pentagon
"*"	star
"h"	hexagon1
"+"	plus
"D"	diamond
...	...

其他属性

File

```
# Filename: multilines.py
```

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
plt.figure(figsize = (8, 6), dpi = 100)
```

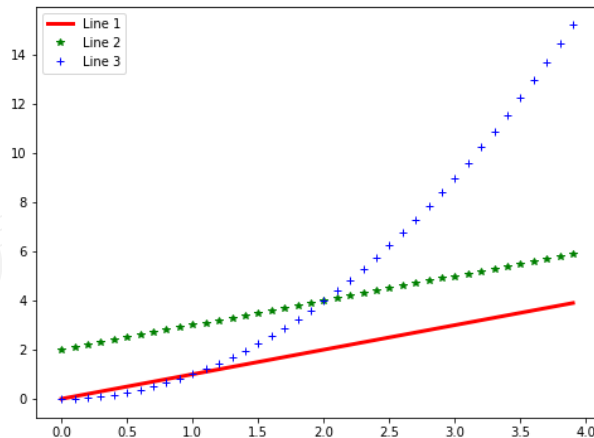
```
t = np.arange(0., 4., 0.1)
```

```
plt.plot(t, t, color='red', linestyle='-', linewidth=3, label='Line 1')
```

```
plt.plot(t, t+2, color='green', linestyle='', marker='*', linewidth=3, label='Line 2')
```

```
plt.plot(t, t**2, color='blue', linestyle='', marker='+', linewidth=3, label='Line 3')
```

```
plt.legend(loc = 'upper left')
```



加标题：图、横轴和纵轴



```
# Filename: title.py
```

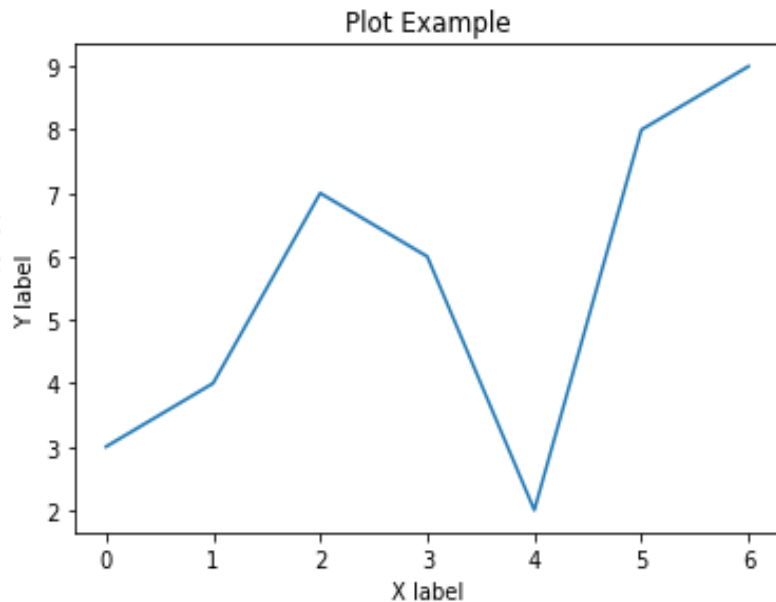
```
import matplotlib.pyplot as plt
```

```
plt.title('Plot Example')
```

```
plt.xlabel('X label')
```

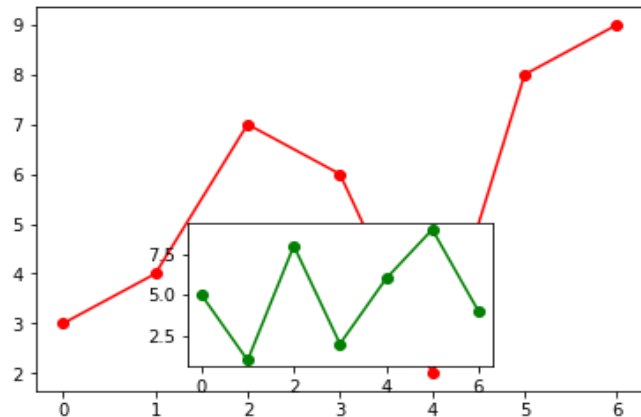
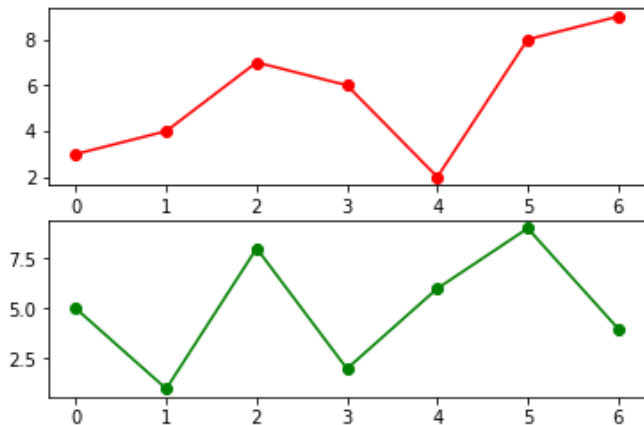
```
plt.ylabel('Y label')
```

```
plt.plot(range(7), [3, 4, 7, 6, 2, 8, 9])
```

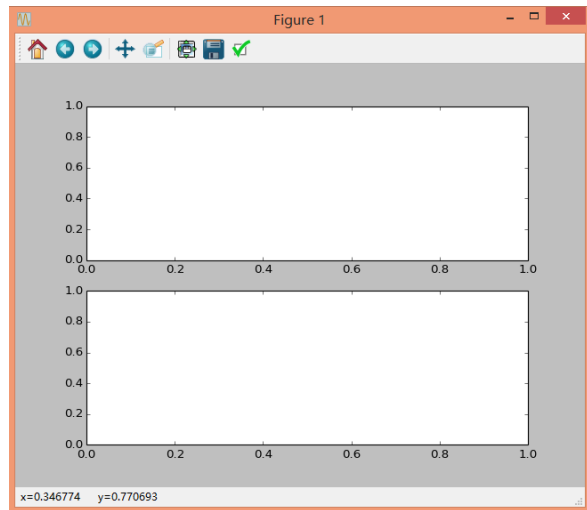


绘制子图

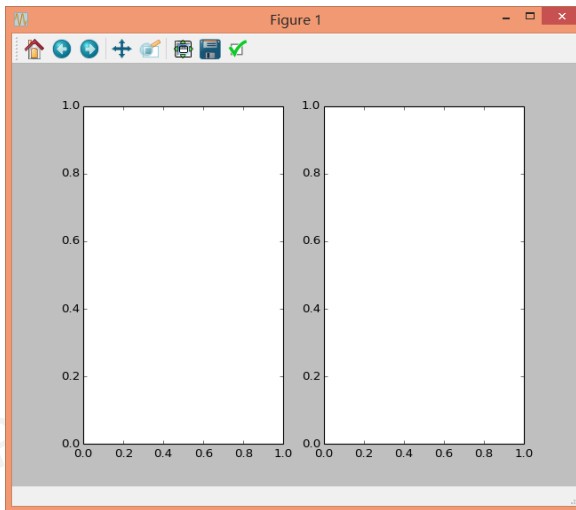
- 在Matplotlib中绘图在当前图形 (figure) 和当前坐标系 (axes) 中进行，默认在一个编号为1的figure中绘图，可以在一个图的多个区域分别绘图
- 使用subplot()/subplots()函数和axes()函数



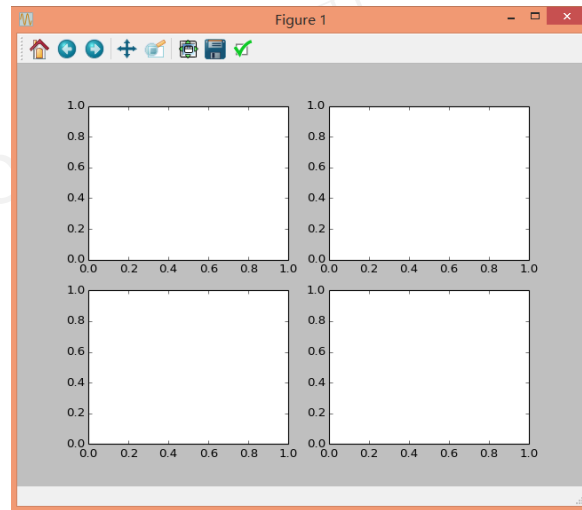
子图-subplot()



```
plt.subplot(211)  
plt.subplot(212)
```



```
plt.subplot(121)  
plt.subplot(122)
```



```
plt.subplot(221)  
plt.subplot(222)  
plt.subplot(223)  
plt.subplot(224)
```

子图-subplot()

File

Filename: subplot.py

import numpy as np

import matplotlib.pyplot as plt

x = np.linspace(-np.pi, np.pi, 300)

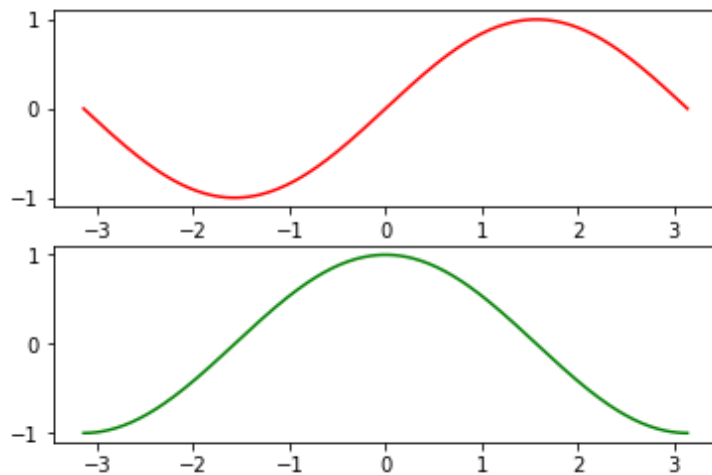
plt.figure(1) # 默认创建, 缺省

plt.subplot(211) # 第一个子图

plt.plot(x, np.sin(x), color = 'r')

plt.subplot(212) # 第二个子图

plt.plot(x, np.cos(x), color = 'g')



子图-subplots()

F_{ile}

Filename: subplots.py

import numpy as np

import matplotlib.pyplot as plt

x = np.linspace(-np.pi, np.pi, 300)

fig, (ax0, ax1) = plt.subplots(2, 1)

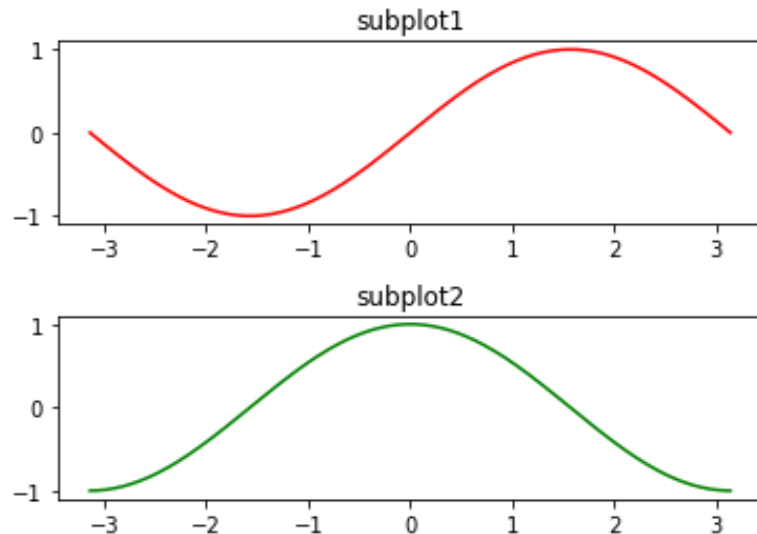
ax0.plot(x, np.sin(x), color = 'r')

ax0.set_title('subplot1')

plt.subplots_adjust(hspace = 0.5)

ax1.plot(x, np.cos(x), color = 'g')

ax1.set_title('subplot2')



子图-axes()

30

axes([left,bottom,width,height]) 参数范围为(0,1)

File

Filename: axes.py

import numpy as np

import matplotlib.pyplot as plt

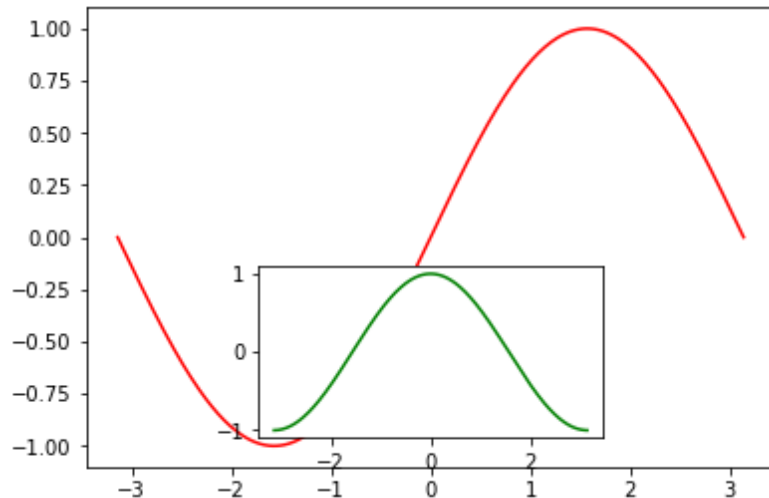
x = np.linspace(-np.pi, np.pi, 300)

plt.axes([.1, .1, 0.8, 0.8])

plt.plot(x, np.sin(x), color = 'r')

plt.axes([.3, .15, 0.4, 0.3])

plt.plot(x, np.cos(x), color = 'g')

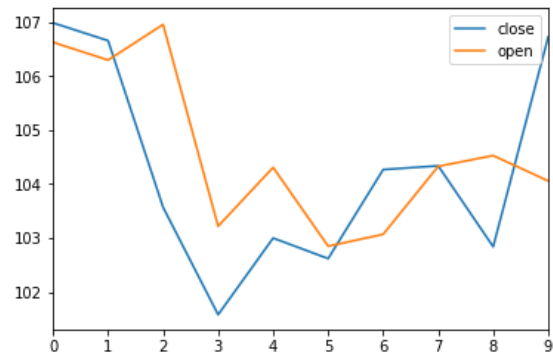
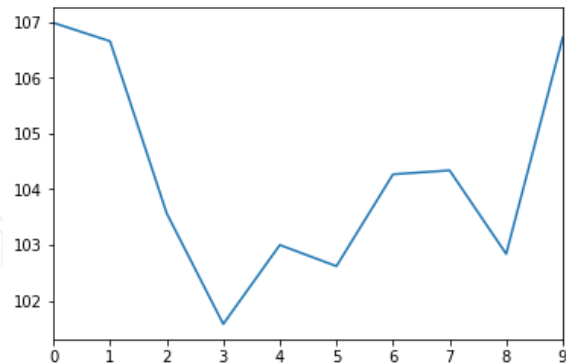


Source

```
>>> quotesdf.loc[:9, 'close'].plot()
```

Source

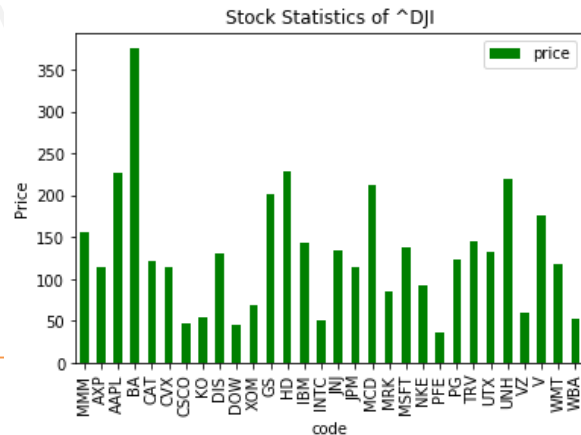
```
>>> quotesdf.loc[:9, ['close', 'open']].plot()
```



pandas绘图

Source

```
>>> ax = djidf.plot(kind = 'bar', x = 'code', y = 'price', color = 'g');  
ax.set(ylabel='Price', title = 'Stock Statistics of ^DJI')
```



用Python玩转数据

数据探索与预处理之数据清洗

数据探索

检查数据错误

了解数据分布特征和内在规律

数据预处理

数据清洗 Data cleaning

数据集成 Data integration

数据变换 Data transformation

数据规约 Data reduction

缺失值处理

如何处理?

- 删除
- 填充

缺失值填充	固定值
	均值, 中位数/众数
	上下数据
	插值函数
	最可能的值

缺失值处理—DataFrame

```
quotesdf_nan = pd.read_csv('AXP_NaN.csv', index_col = 'Date')
```

判断缺失值 `df.isnull()`

删除缺失行 `df.dropna()`

填充缺失行 `df.fillna()`

如何用均值填充?

```
quotesdf_nan.fillna(method='ffill', inplace = True)
```

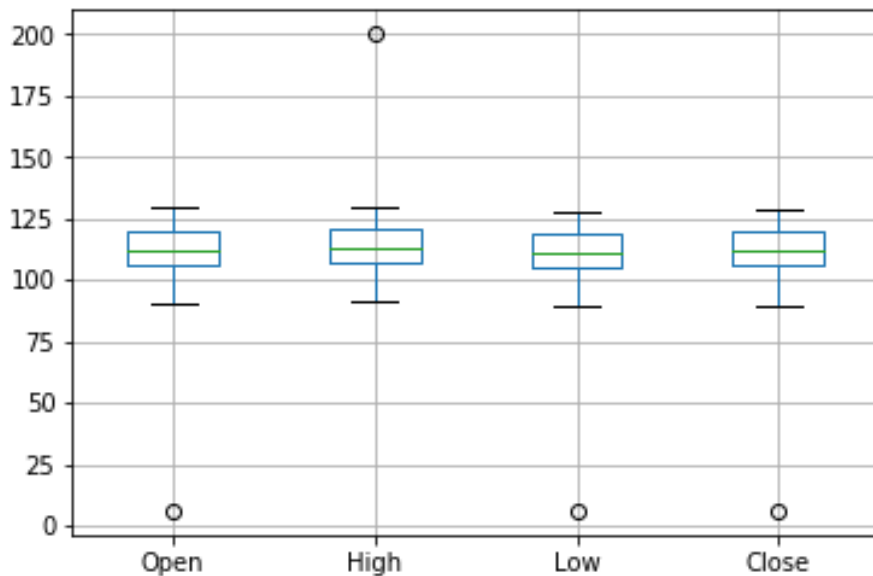
异常值处理

如何观察异常值？

- 简单统计
- 绘图
- 基于密度，最近邻和聚类等方法

如何处理？

- 同缺失值处理
- 局部均值(分箱)
- 不处理



用Python玩转数据

数据预处理之 数据变换



把数据变换成适合的形式

常见方式

规范化

连续属性离散化

特征二值化

数据规范化

解决哪些影响？

- 量纲不同
- 数值范围差异大

规范化常用方法

- 最小-最大规范化
- z-score规范化
- 小数定标规范化

波士顿房价数据集

```
>>> boston = datasets.load_boston()
>>> boston.feature_names
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
      'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
>>> boston.target
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, ..., ])
>>> boston_df = pd.DataFrame(boston.data[:, 4:7])
>>> boston_df.columns = boston.feature_names[4:7]
>>> boston_df
```

	NOX	RM	AGE
0	0.538	6.575	65.2
1	0.469	6.421	78.9
2	0.469	7.185	61.1
...			
504	0.573	6.794	89.3
505	0.573	6.030	80.8

4: NOX - 一氧化氮浓度 (每1000万份) nitric oxides concentration (parts per 10 million)
 5: RM - 平均每居民房数 average number of rooms per dwelling
 6: AGE - 在1940年之前建成的所有者占用单位的比例 proportion of owner-occupied units built prior to 1940
 MEDV - Median value of owner-occupied homes in \$1000's

最小-最大规范化

$$x' = \frac{x - \min}{\max - \min}$$

```
(df-df.min())/(df.max()-df.min())
```

问题:

- 若将来的数字超过min和max需重新定义
- 若某个数很大则规范化后值相近且均接近0

	NOX	RM	AGE
0	0.538	6.575	65.2
1	0.469	6.421	78.9
2	0.469	7.185	61.1
3	0.458	6.998	45.8
4	0.458	7.147	54.2
5	0.458	6.430	58.7

	NOX	RM	AGE
0	0.314815	0.577505	0.641607
1	0.172840	0.547998	0.782698
2	0.172840	0.694386	0.599382
3	0.150206	0.658555	0.441813
4	0.150206	0.687105	0.528321
5	0.150206	0.549722	0.574665

```
from sklearn import preprocessing
```

```
min_max_scaler = preprocessing.minmax_scale(df) # [0,1]
```



z-score规范化

$$x' = \frac{x - \bar{x}}{\sigma}$$

```
(df-df.mean())/df.std()
```

特征：

- 使用最多
- 处理后数据的均值为0，标准差为1

	NOX	RM	AGE
0	0.538	6.575	65.2
1	0.469	6.421	78.9
2	0.469	7.185	61.1
3	0.458	6.998	45.8
4	0.458	7.147	54.2
5	0.458	6.430	58.7

	NOX	RM	AGE
0	-0.144075	0.413263	-0.119895
1	-0.739530	0.194082	0.366803
2	-0.739530	1.281446	-0.265549
3	-0.834458	1.015298	-0.809088
4	-0.834458	1.227362	-0.510674
5	-0.834458	0.206892	-0.350810

z-score规范化

```
scaler = preprocessing.scale(df)
```

```
array([[ -0.14421743,  0.41367189, -0.12001342],  
       [ -0.74026221,  0.19427445,  0.36716642],  
       [ -0.74026221,  1.28271368, -0.26581176],  
       ...,  
       [ 0.15812412,  0.98496002,  0.79744934],  
       [ 0.15812412,  0.72567214,  0.73699637],  
       [ 0.15812412, -0.36276709,  0.43473151]])
```



小数定标规范化

$$x' = \frac{x}{10^j}$$

```
df/10**(np.ceil(np.log10(df.abs().max())))
```

特征:

- 移动小数点位置, 移动位数取决于属性绝对值的最大值
- 常见落在[-1, 1]之间

	NOX	RM	AGE
0	0.538	6.575	65.2
1	0.469	6.421	78.9
2	0.469	7.185	61.1
3	0.458	6.998	45.8
4	0.458	7.147	54.2
5	0.458	6.430	58.7

	NOX	RM	AGE
0	0.538	0.6575	0.652
1	0.469	0.6421	0.789
2	0.469	0.7185	0.611
3	0.458	0.6998	0.458
4	0.458	0.7147	0.542
5	0.458	0.6430	0.587

连续属性离散化

方法:

- 分箱 (binning) : 等宽法, 等频法
- 聚类

```
pd.cut(df.AGE, 5, labels = range(5))
pd.qcut(df.AGE, 5, labels = range(5))
```

0	65.2
1	78.9
2	61.1
3	45.8
4	54.2
5	58.7

0	3	0	1
1	3	1	2
2	2	2	1
3	2	3	1
4	2	4	1
5	2	5	1

特征二值化binarization

rating	Label
6	1
4	0
7	1
7	1
8	1
5	0
3	0
3	0
9	1
4	0
6	1
7	1
5	0
9	1
9	1
8	1
2	0
5	0
3	0
3	0



```
>>> from sklearn.preprocessing import Binarizer
```

```
>>> X = boston.target.reshape(-1,1)
```

```
>>> Binarizer(threshold = 20.0).fit_transform(X)
```




用Python玩转数据

数据预处理之 数据规约

目的:

- 对属性和数值进行规约获得一个比原数据集的小的多的规约表示, 但仍接近原数据的完整性, 在规约后数据集上挖掘可产生近乎相同的分析结果

数据规约

属性规约: 向前选择, 向后删除, 决策树, PCA

数值规约: 有参方法 (回归法, 对数线性模型), 无参法 (直方图, 聚类, 抽样)

S

```
>>> from sklearn.decomposition import PCA
>>> X = preprocessing.scale(boston.data)
>>> pca = PCA(n_components=5)
>>> pca.fit(X)
>>> pca.explained_variance_ratio_
array([0.47129606, 0.11025193, 0.0955859 , 0.06596732, 0.06421661])
```

数值规约—直方图

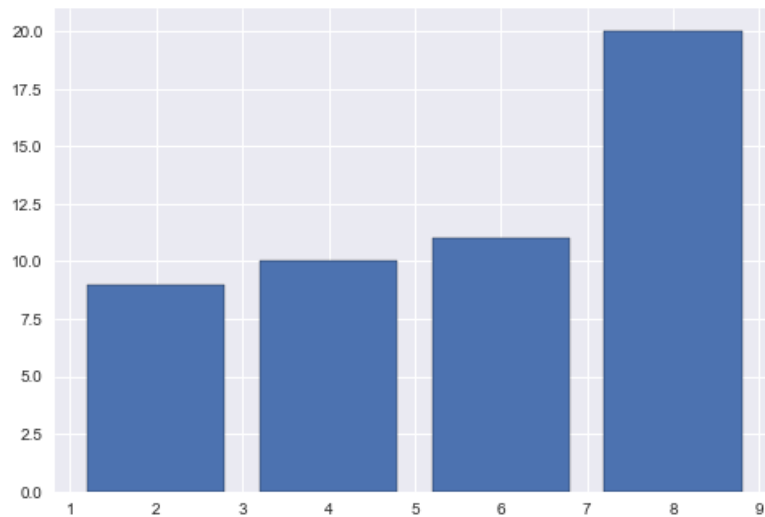
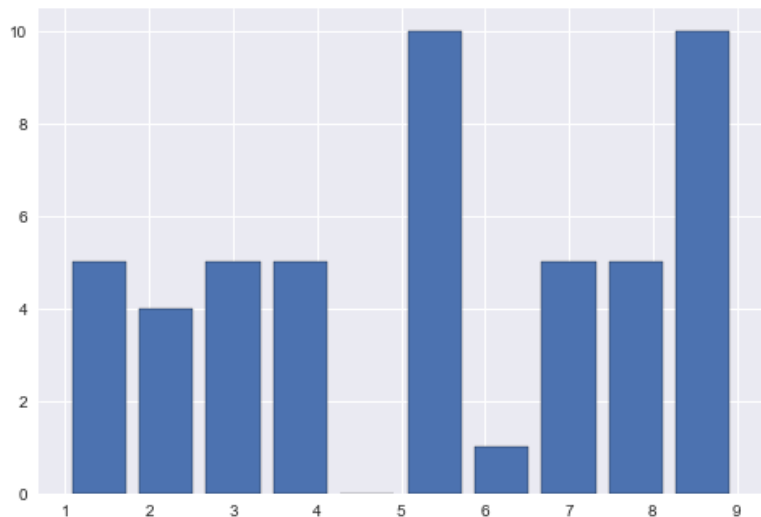
表现：

- 用分箱表示数据分布
- 每个箱子代表一个属性-频率对

```
array([4, 8, 9, 8, 7, 2, 8, 7, 5, 3, 1, 4, 5, 8, 7, 9, 5, 9, 9, 5, 9, 1, 9, 7, 1, 2, 9, 5, 5,  
5, 9, 4, 3, 5, 5, 4, 7, 4, 9, 8, 2, 6, 3, 5, 3, 2, 9, 1, 3, 1])
```

```
data = np.random.randint(1,10,50)
```

数值规约—直方图



```
array([4, 8, 9, 8, 7, 2, 8, 7, 5, 3, 1, 4, 5, 8, 7, 9, 5, 9, 9, 5, 9, 1, 9, 7, 1, 2, 9, 5, 5, 5, 9, 4, 3, 5, 5, 4, 7, 4, 9, 8, 2, 6, 3, 5, 3, 2, 9, 1, 3, 1])
```

```
plt.hist(data, bins=...)
```

抽 样	随机抽样：不放回
	随机抽样：放回
	聚类抽样
	分层抽样

特征列举：

- 不放回随机抽样：从原始数据集D的N个样本中抽取n个样本，每次抽到不同的数据
- 放回随机抽样：从原始数据集D的N个样本中抽取n个样本，抽取后记录它后放回，有可能抽到同样的数据
- 分层抽样：数据集D为划分成互不相交的部分即层，对每一层进行简单随机抽样获得最终结果

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
24	4.8	3.4	1.9	0.2
81	5.5	2.4	3.7	1.0
145	6.7	3.0	5.2	2.3
115	6.4	3.2	5.3	2.3
136	6.3	3.4	5.6	2.4
98	5.1	2.5	3.0	1.1
79	5.7	2.6	3.5	1.0
143	6.8	3.2	5.9	2.3
87	6.3	2.3	4.4	1.3
107	7.3	2.9	6.3	1.8

不放回：

```
iris_df.sample(n = 10)  
iris_df.sample(frac = 0.3)
```

有放回：

```
iris_df.sample(n = 10, replace = True)  
iris_df.sample(frac = 0.3, replace = True)
```

分层抽样

56



```
>>> A = iris_df[iris_df.target == 0].sample(frac = 0.3)
>>> B = iris_df[iris_df.target == 1].sample(frac = 0.2)
>>> A.append(B)
```



