目录

pandas 精要笔记

cuihu

2020 年 8 月 31 日

1 封面

这本笔记是根据下边这本书来得来的：

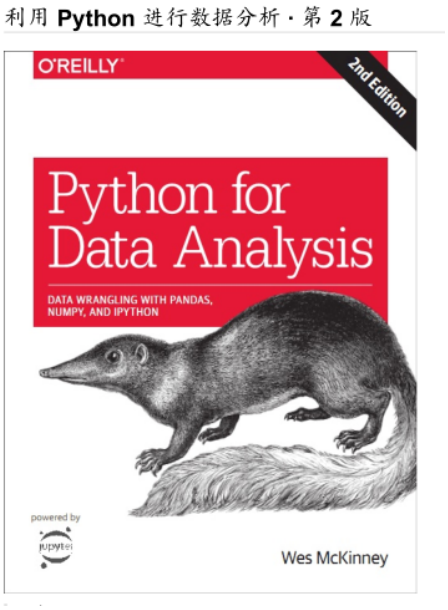
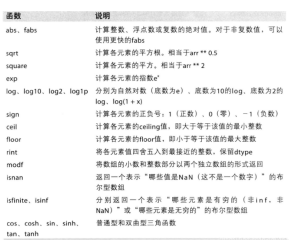


图 1: 书籍封面

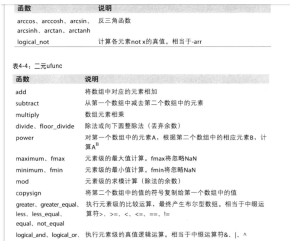
2 numpy 基础

2.1 创建 ndarray



函数	说明
abs、fabs	计算整数、浮点数或复数的绝对值。对于非复数值，可以使用更快的fabs
sqrt	计算各元素的平方根。相当于arr ** 0.5
square	计算各元素的平方。相当于arr ** 2
exp	计算各元素的指数e^x
log、log10、log2、log1p	分别为自然对数（底数为e）、底数为10的log、底数为2的log、log(1 + x)
sign	计算各元素的正负号：1（正数）、0（零）、-1（负数）
ceil	计算各元素的ceiling值，即大于等于该值的最小整数
floor	计算各元素的floor值，即小于等于该值的最大整数
rint	将各元素值四舍五入到最接近的整数，保留dtype
modf	将数组的小数和整数部分以两个独立数组的形式返回
isnan	返回一个表示“哪些值是NaN（这不是一个数字）”的布尔型数组
isfinite、isinf	分别返回一个表示“哪些元素是有穷的（非inf，非NaN）”或“哪些元素是无穷的”的布尔型数组
cos、cosh、sin、sinh、tan、tanh	普通型和双曲型三角函数

(a) 1



函数	说明
arccos、arccosh、arcsin、arcsinh、arctan、arctanh	反三角函数
logical_not	计算各元素not x的真值，相当于-arr

表4-4：二元ufunc

函数	说明
add	将数组中对应的元素相加
subtract	从第一个数组中减去第二个数组中的元素
multiply	数组元素相乘
divide、floor_divide	除法或向下圆整除法（丢弃余数）
power	对第一个数组中的元素A，根据第二个数组中的相应元素B，计算A^B
maximum、fmax	元素级的最大值计算，fmax将忽略NaN
minimum、fmin	元素级的最小值计算，fmin将忽略NaN
mod	元素级的求模计算（除法的余数）
copysign	将第二个数组中的值的符号复制给第一个数组中的值
greater、greater_equal、less、less_equal、equal、not_equal	执行元素级的比较运算，最终产生布尔型数组。相当于中缀运算符>、>=、<、<=、==、!=
logical_and、logical_or、	执行元素级的真值逻辑运算，相当于中缀运算符&、|、^

(b) 2

图 2: 基础函数

```
arr1.shape\\array
asarray
arange
ones,ones_like
zeros,zeros_like
empty. empty_like.
eye, identity
```

数组转置和轴对称：

```
arr.T
arr.transpose
arr.swapaxes
```

2.2 利用数组进行数据处理

np.meshgrid 接收两个一维数组，产生两个二维矩阵。对应于所有的（x,y）对，产生的两个矩阵一个对第一个数组进行行重复，一个以列进行重复。两个矩阵凉凉对应后，就是所谓的网格点坐标对军阵。

```
points = np.arange(-5,5,0.01)
xs,ys = np.meshgrid(points, points)
ys
```

输出：

```
ys:
array([[-5. , -5. , -5. , ..., -5. , -5. , -5. ],
[-4.99, -4.99, -4.99, ..., -4.99, -4.99, -4.99],
```

```
[-4.98, -4.98, -4.98, ..., -4.98, -4.98, -4.98],
...,
[ 4.97, 4.97, 4.97, ..., 4.97, 4.97, 4.97],
[ 4.98, 4.98, 4.98, ..., 4.98, 4.98, 4.98],
[ 4.99, 4.99, 4.99, ..., 4.99, 4.99, 4.99]])
xs:
array([[-5. , -4.99, -4.98, ..., 4.97, 4.98, 4.99],
[-5. , -4.99, -4.98, ..., 4.97, 4.98, 4.99],
[-5. , -4.99, -4.98, ..., 4.97, 4.98, 4.99],
...,
[-5. , -4.99, -4.98, ..., 4.97, 4.98, 4.99],
[-5. , -4.99, -4.98, ..., 4.97, 4.98, 4.99],
[-5. , -4.99, -4.98, ..., 4.97, 4.98, 4.99]])
z = np.sqrt(xs**2 + ys**2)
输出:
array([[7.07106781, 7.06400028, 7.05693985, ..., 7.04988652,
    7.05693985,
7.06400028],
[7.06400028, 7.05692568, 7.04985815, ..., 7.04279774, 7.04985815,
7.05692568],
[7.05693985, 7.04985815, 7.04278354, ..., 7.03571603, 7.04278354,
7.04985815],
...,
[7.04988652, 7.04279774, 7.03571603, ..., 7.0286414 , 7.03571603,
7.04279774],
[7.05693985, 7.04985815, 7.04278354, ..., 7.03571603, 7.04278354,
7.04985815],
[7.06400028, 7.05692568, 7.04985815, ..., 7.04279774, 7.04985815,
7.05692568]])

画图:
plt.imshow(z,cmap=plt.cm.gray); plt.colorbar()
plt.title('Image plot of $\sqrt{x^2+y^2}$ for a grid of values')
```

2.3 将条件逻辑表述为数组运算

numpy.where(condition,x,y) 函数是三元表达式 x if condition else y 的矢量化版本。

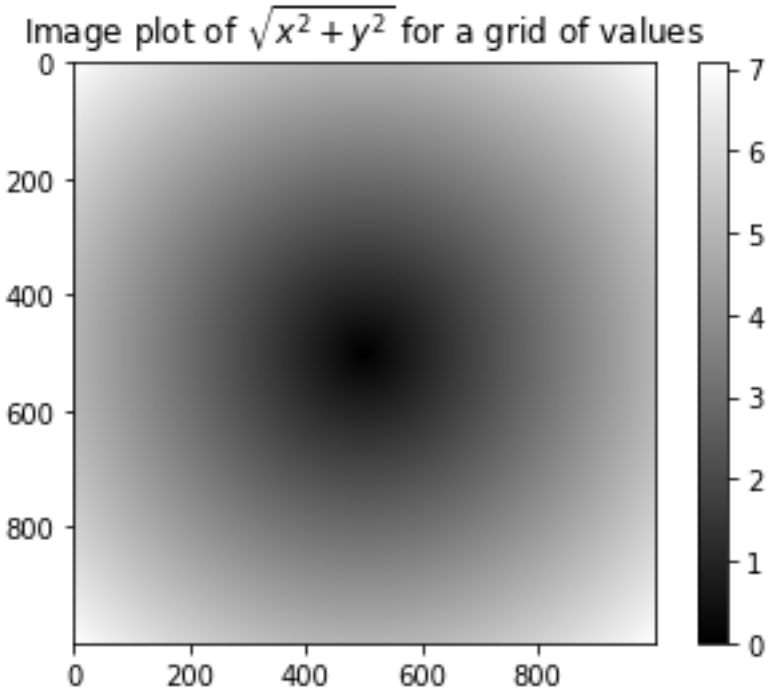


图 3: meshgird 灰度图

```
xarr = np.array([1.1, 1.2, 1.3, 1.4, 1.5])
yarr = np.array([2.1, 2.2, 2.3, 2.4, 2.5])
cond = np.array([True, False, True, True, False])
#我们想要根据cond的值选取xarr和yarr，当cond的true，选取xarr的值，否则选取yarr的值。
result = [(x if c else y) for x, y, c in zip(xarr, yarr, cond)]
结果:
[1.1, 2.2, 1.3, 1.4, 2.5]
```

当我们使用多维数组的时候，上述处理方法不是很快，要快速处理的话，我们使用如下方法：

```
d = zip(xarr, yarr, cond)
result = np.where(cond, xarr, yarr)
result:
array([1.1, 2.2, 1.3, 1.4, 2.5])

from numpy.random import randn
arr = randn(4,4)
arr:
array([[-1.00815132, 0.02094386, -1.40737364, 0.57612992],
[-0.27886028, 0.76804402, -0.28459415, -1.75665045],
[-0.39905925, -2.3398309 , 0.26511772, -0.37557294],
[ 0.44804335, 3.08441085, 0.7388182 , -0.5724736 ]])
np.where(arr > 0, 2, -2) #
    条件后边还可以直接跟标量。如果为矢量的话，那么维度就需要跟arr有相一致。
array([[-2, 2, -2, 2],
[-2, 2, -2, -2],
[-2, -2, 2, -2],
[ 2, 2, 2, -2]])
```

2.4 数学和统计方法

聚合运算:*aggregation*,约简 *reduction sum,mean,max,min,std,np.cumsum,np.cumprod....cumsum,cumprod* 都是不产生聚合效果的。

2.5 用于布尔型数组的方法

表4-5：基本数组统计方法

方法	说明
sum	对数组中全部或某轴向的元素求和。零长度的数组的sum为0
mean	算术平均数。零长度的数组的mean为NaN
std、var	分别为标准差和方差，自由度可调（默认为n）
min、max	最大值和最小值
argmin、argmax	分别为最大和最小元素的索引

表4-5：基本数组统计方法（续）

方法	说明
cumsum	所有元素的累计和
cumprod	所有元素的累计积

图 4: 各种聚合和约简单函数

```
arr = randn(100)
(arr > 0)
输出:
array([ True, False, True, False, True, False, True, False, False,
False, True, True, False, True, True, True, True, False,
False, True, False, False, False, True, True, True, True,
False, True, False, False, False, True, False, True, True,
False, True, False, False, False, True, True, False, True,
False, True, False, False, False, True, True, True, False,
False, False, False, False, False, False, True, False, False,
False, False, False, True, True, True, False, True, False,
False, True, True, False, True, True, True, False, False,
False, True, False, True, False, False, True, True, False,
False, False, False, True, True, True, False, True, False,
True])
统计ture的个数:
(arr > 0).sum()
默认将True当作1，将false当作0 来进行统计:
46
```

np.any(),np.all,array.all(),array.any() 都是即可以根据轴 axis 来进行统计，也可以进行总体统计。

2.6 排序

arr.sort(axis=???) 根据轴进行排序。

np.unique() 类似于 python 的 *set()* 函数。

np.in1d(ar1,ar2) 测试集合 1 的成员是否再集合 2 中。

```
values = np.array([6,0,3,2,5,6])
np.in1d(values, [2,3,5]) # 测试values的成员是否在当前集合中。
输出:
array([False, False, True, True, True, False])
```

表4-6：数组的集合运算

方法	说明
unique(x)	计算x中的唯一元素，并返回有序结果
intersect1d(x, y)	计算x和y中的公共元素，并返回有序结果
union1d(x, y)	计算x和y的并集，并返回有序结果
in1d(x, y)	得到一个表示“x的元素是否包含于y”的布尔型数组
setdiff1d(x, y)	集合的差，即元素在x中且不在y中
setxor1d(x, y)	集合的对称差，即存在于一个数组中但不同时存在于两个数组中的元素[译注2]

图 5: 数组集合运算

2.7 用于数组的文件输入输出

```
arr = np.arange(10)
数组的保存于读取:
np.save('some_array',arr)
np.load('some_array.npy')
多个数组保存于读取:
np.savez('some_array.npz',a=arr,b=arr) # 将两个数组压缩到一起存储。
arch = np.load('some_array.npz')
arch['a']
```

2.8 存取文本文件

```
read_csv, rea_table
arr = np.loadtxt('array_ex.txt',delimiter=',')
```

当然这些都是针对纯数组文件的。

2.9 线性代数

```
x = np.array([[1.,2.,3.],[4.,5.,6.]])
y = np.array([[6.,23.],[-1,7],[8,9]])
输出:
[[1. 2. 3.]
[4. 5. 6.]]
[[ 6. 23.]
[-1. 7.]
[ 8. 9.]]
x.dot(y),np.dot(x,y)
分解，行列式，求逆运算，方程组的解。
from numpy.linalg import inv,qr
mat = x.T.dot(x)
inv(mat)
mat.dot(inv(mat))
计算矩阵的qr分解:
q,r = qr(mat)
因为q是正交的，所以q.dot(q.T) 为单位矩阵。
因为r为对角的，所以
    inv(r).dot(r)也是单位。从而应用q,r,就可以计算多元方程组。
ax = b

a= qr

x = inv(r).dot(q.T).dot(b)
```

2.10 随机数生成

随机高斯分布生成：

```
samples = np.random.normal(size=(4,4))
```

2.11 范例，随机漫步

函数	说明
diag	以一维数组的形式返回方阵的对角线（或非对角线）元素，或将一维数组转换为方阵（非对角线元素为0）
dot	矩阵乘法
trace	计算对角线元素的和
det	计算矩阵行列式
eig	计算方阵的本征值和本征向量
inv	计算方阵的逆
pinv	计算矩阵的Moore-Penrose伪逆
qr	计算QR分解
svd	计算奇异值分解（SVD）
solve	解线性方程组Ax = b，其中A为一个方阵
lstsq	计算Ax = b的最小二乘解

图 6: 常用线性函数

函数	说明
seed	确定随机数生成器的种子
permutation	返回一个序列的随机排列或返回一个随机排列的范围
shuffle	对一个序列就地随机排列
rand	产生均匀分布的样本值
randint	从给定的上下限范围内随机选取整数
randn	产生正态分布（平均值为0，标准差为1）的样本值，类似于MATLAB接口
binomial	产生二项分布的样本值
normal	产生正态（高斯）分布的样本值
beta	产生Beta分布的样本值

图 7: 常用的 random 函数 1

函数	说明
chisquare	产生卡方分布的样本值
gamma	产生Gamma分布的样本值
uniform	产生在[0, 1)中均匀分布的样本值

图 8: 常用的 random 函数 2

```
import random
position = 0
walk = [position]
steps = 1000
for i in range(steps):
step = 1 if random.randint(0,1) else -1
position += step
walk.append(position)
```

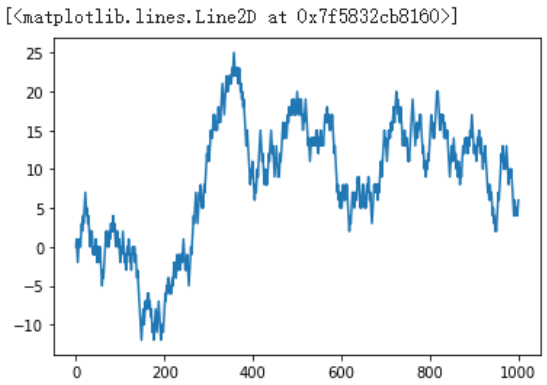


图 9: 随机漫步效果

用 np 的方法来实现：

```
nsteps = 1000
产生两个数0 1 :
draws = np.random.randint(0,2,size=nsteps)
判断方向:
steps = np.where(draws >0,1,-1)
进行走动:
walk = steps.cumsum()
判断最远和最近:
walk.min()
walk.max()
画出图像:
plt.plot([x for x in range(nsteps)],walk)
第一次离0点距离大于10:
```

```
a = np.abs(walk) >= 10
找出第一次的坐标:
a.argmax()
```

2.12 一次模拟多个随机漫步

```
nwalks = 5000
nsteps = 1000
draws = np.random.randint(0,2,size=(nwalks, nsteps))
steps = np.where(draws > 0, 1, -1)
walks = steps.cumsum(1) #在列方向进行累加。
walks.max()
walks.min()
hist30 = (np.abs(walks)>= 30).any(1)
多少次随机漫步穿过了30
hist30.sum()
找出那些穿过30的随机漫步，第一次穿过30的索引。
crossing_times = (np.abs(walks[hist30]) >=30).argmax(1)
对他们求平均值。
```

3 pandas 入门

3.1 Series Class

```
from pandas import Series, DataFrame
import pandas as pd
obj = Series([4,7,-5,3])
obj.values
obj.index
obj2 = Series([4,7,5,3], index=['a', 'b', 'c', 'd'])
结果:
a   4
b   7
c   5
d   3
dtype: int64
```

```
obj2['d'] 3
字典和series转换:
sdata = {'ohi':3500, 'texa':7100, 'orgen':1900, 'utah':4000}
obj3 = Series(sdata)
:
ohi     3500
texa    7100
orgen   1900
utah    4000
dtype: int64
列表当作index:
states = ['aiaang','ohi', 'orgen', 'utah']
obj4 = Series(sdata, index=states)
判断series各个值是否为null:
pd.isnull(obj4)
pd.notnull()
series相加: pandas会根据索引自动进行配对, 没有配对成功的结果做nan

series的name属性:
obj4.name = 'population'
obj4.index.name = 'state'
::
state
aiaang      NaN
ohi      3500.0
orgen    1900.0
utah     4000.0
Name: population, dtype: float64
```

3.2 DataFrame Class

```
import numpy as np
import pandas as pd
from pandas import Series
from pandas import DataFrame
# 传入由等长列表组成的字典, 来构建DataFrame
```

```
data = {'state': ['Ohio','Ohio','Ohio','Nevada','Nevada'],
'year': [2000, 2001, 2002, 2001, 2002],
'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
frame = DataFrame(data)
state year pop
0 Ohio 2000 1.5
1 Ohio 2001 1.7
2 Ohio 2002 3.6
3 Nevada 2001 2.4
4 Nevada 2002 2.9
```

可以向上边一样，之际由字典来构成一个 dataframe：字典 key 自动当成 columns。索引自动生成。也可以自动改变索引:`DataFrame(data, columns=['year', 'state', 'pop'])`

自己设定 index，columns 名字：

```
frame2 = DataFrame(data, columns=['year', 'state', 'pop', 'debt'], index=['one', 'two', 'three', 'four', 'five'])
```

---

df:列名列表：

```
  frame2.columns
Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

dataframe可以转置：

```
frame3 = DataFrame(pop)
frame3
Nevada Ohio
2001 2.4 1.7
2002 2.9 3.6
2000 NaN 1.5
frame3.T
2001 2002 2000
Nevada 2.4 2.9 NaN
Ohio 1.7 3.6 1.5

frame3.index.name = 'year'
frame3.columns.name = 'state'
```

Table 5-1. Possible data inputs to DataFrame constructor

Type	Notes
2D ndarray	A matrix of data, passing optional row and column labels
dict of arrays, lists, or tuples	Each sequence becomes a column in the DataFrame; all sequences must be the same length
NumPy structured/record array	Treated as the "dict of arrays" case
dict of Series	Each value becomes a column; indexes from each Series are unioned together to form the result's row index if no explicit index is passed
dict of dicts	Each inner dict becomes a column; keys are unioned to form the row index as in the "dict of Series" case
List of dicts or Series	Each item becomes a row in the DataFrame; union of dict keys or Series indexes become the DataFrame's column labels
List of lists or tuples	Treated as the "2D ndarray" case
Another DataFrame	The DataFrame's indexes are used unless different ones are passed
NumPy MaskedArray	Like the "2D ndarray" case except masked values become NA/missing in the DataFrame result

图 10: 所有可以传给 dataframe 的值

3.3 Index Objects 索引对象

比如：`frame3.columns` 就是 index object 索引对象。
然后可以判断这个索引对象内是否含有某个对象：'Ohio' in frame3.columns
: true

3.4 Essential Functionality

3.5 reindexing 索引重建

```
obj = pd.Series([4.5, 7.2, -5.3, 3.6],index=['d', 'b', 'a', 'c'])
obj2 = obj.reindex(['a','b','c','d','e'])
obj3 = pd.Series(['blue', 'purple', 'yellow'],index=[0,2,4])
obj3.reindex(range(6), method='ffill') #range boject
```

3.6 Dropping Entries from an Axis

Dropping Entries from an Axis
根据轴来删除条目
obj.drop(labels=None, axis=0, index=None, columns=None, level=None,

inplace=False, errors='raise')

```
obj= pd.Series(np.arange(5.0),index=['a','b','c','d','e'])
new_obj = obj.drop('a')
data = pd.DataFrame(np.arange(16).reshape(4,4), index=['Ohio',
    'Colorada', 'Utah', 'New York'],
columns=['one', 'two', 'three', 'four'])
data.drop(['Colorada', 'Ohio'])
data.drop(['one','two'], axis= 1)
data.drop(columns=['one', 'two'])
data.drop(index=['Ohio'], columns=['two'], inplace=True)
```

3.7 Indexing, Selection, and Filtering 索引、选择和筛选

3.7.1 indexing

```
Series:
  obj = pd.Series(np.arange(4.), index=['a', 'b', 'c', 'd'])
  obj[['a','c']]
  obj[[1,3]]
  obj[obj < 2]
  obj['a':'c']
  obj['a':'c'] = 100
DataFrame:
  data = pd.DataFrame(np.arange(16).reshape(4,4), index=['Ohio',
      'Colorada', 'Utah', 'NewYork'],
  columns= ['one', 'two', 'trhee', 'four'])
  data
  data.loc['Ohio']
  data['two']
  data[['trhee','two']]
  data[:2]
  data[:1]
  data[data['trhee'] > 5 ]
  data < 5
  data[data < 5] = 0 # like numpy case.
```

3.7.2 Selection with loc and iloc (loc,iloc)

loc 用索引名字符，iloc 用索引位置：

```
Selection with loc and iloc (loc,iloc)
data.iloc[2,[3,0,1]]
data.iloc[2]
data.loc[:'Utah', 'two']
data.loc['Utah', 'two']
data.iloc[:,:3]
one  two  trhee
Ohio 0  0  0
Colorada 0 5 6
Utah 8  9  10
NewYork 12 13 14
```

3.7.3 Integer Indexes 整数索引

```
ser = pd.Series(np.arange(6.0))
ser.loc[:1]
ser2.iloc[:2]
```

3.8 Arithmetic and Data Alignment. 算法和数据对齐

意思就是，在进行运算的时候，dataframe 和 Series 可以自动进行对齐，对不齐的部分结果为 0.

3.9 Arithmetic methods with fill values 填充值的算数方法

一般来说直接用“+-*/”方法，但是没法对不对齐的部分进行填充。但是如果用相应的函数运算的话，那么可以进行一些非对齐 NAN 填充。
用着写函数来对 dataframe 进行运算的时候,可以传一个对应的`fill_value`参数。

3.10 Operations between DataFrame and Series

默认 serie 和 dataframe 加减，自动进行行广播。

Table 5-2. Some Index methods and properties

Method	Description
`append`	Concatenate with additional Index objects, producing a new Index
`difference`	Compute set difference as an Index
`intersection`	Compute set intersection
`union`	Compute set union
`isin`	Compute boolean array indicating whether each value is contained in the passed collection
`delete`	Compute new Index with element at index `i` deleted
`drop`	Compute new Index by deleting passed values
`insert`	Compute new Index by inserting element at index `i`
`is_monotonic`	Returns `True` if each element is greater than or equal to the previous element
`is_unique`	Returns `True` if the Index has no duplicate values
`unique`	Compute the array of unique values in the Index

图 11: index 对象的各种方法和属性

Table 5-4. Indexing options with DataFrame

Type	Notes
`df[val]`	Select single column or sequence of columns from the DataFrame; special case conveniences: boolean array (filter rows), slice (slice rows), or boolean DataFrame (set values based on some criterion)
`df.loc[val]`	Selects single row or subset of rows from the DataFrame by label
`df.loc[:, val]`	Selects single column or subset of columns by label
`df.loc[val1, val2]`	Select both rows and columns by label
`df.iloc[where]`	Selects single row or subset of rows from the DataFrame by integer position

Type	Notes
`df.iloc[:, where]`	Selects single column or subset of columns by integer position
`df.iloc[where_i, where_j]`	Select both rows and columns by integer position
`df.at[label_i, label_j]`	Select a single scalar value by row and column label
`df.iat[i, j]`	Select a single scalar value by row and column position (integers)
`reindex` method	Select either rows or columns by labels
`get_value, set_value` methods	Select single value by row and column label

图 12: DataFrame 索引

Method	Description
add, radd	Methods for addition (+)
sub, rsub	Methods for subtraction (-)
div, rdiv	Methods for division (/)
floordiv, rfloordiv	Methods for floor division (//)
mul, rmul	Methods for multiplication (*)
pow, rpow	Methods for exponentiation (**)

图 13: 相应的计算函数

```
frame = df(np.arange(12.0).reshape((4,3)),
columns= list('bde'),
index=['Utah', 'Ohio', 'Texas', 'Oregon'])
series = frame.iloc[0]

frame - series
    进行行广播，当然无法自动对齐，也就是没有对应列的进行NAN赋值。
b    d    e
Utah   0.0  1.0  2.0
Ohio   3.0  4.0  5.0
Texas  6.0  7.0  8.0
Oregon 9.0 10.0 11.0

series2 = pd.Series(range(3), index= list('bef'))

frame + series2
b  d  e  f
Utah 0.0  NaN  3.0  NaN
Ohio 3.0  NaN  6.0  NaN
Texas 6.0 NaN  9.0  NaN
Oregon  9.0  NaN  12.0 NaN
```

3.11 Function Application and Mapping 函数应用与映射

```
frame = df(np.random.randn(4,3), columns=list('bde'), index=['Utah',
    'Ohio', 'Texas', 'Oregon'])

np.abs(frame)

默认取列：因为一般将列当作特征，行当作样本行：
f = lambda x: x.max() - x.min()
frame.apply(f)

frame.apply(f, axis = 'columns')
frame.apply(f, axis= 1)
传递的参数不是表格的单个元素，而是整个表格的列或者行，或者整个表格：
def f(x):
  return pd.Series([x.min(), x.max()], index=['min', 'max'])
frame.apply(f)
表格的形状不改变，只是改变了表格元素对应值：用map和applymap:
format = lambda x: '%2f' % x
frame.applymap(format)

frame['e'].map(format)
```

3.12 Sorting and Ranking

```
Signature: frame.sort_index(axis=0, level=None, ascending=True,
    inplace=False, kind='quicksort', na_position='last',
    sort_remaining=True, ignore_index: bool=False)

Docstring: Sort object by labels (along an axis).

Signature: obj.sort_values(axis=0, ascending=True, inplace=False,
    kind='quicksort', na_position='last', ignore_index=False)

Docstring: Sort by the values.

还有一种：obj.rank(method='first')
```

排出的是对应数据的位次。

3.13 Axis Indexes with Duplicate Labels 具有重复标签的轴索隐

obj = sr(range(5), index=list('aabbc'))

3.14 Summarizing and Computing Descriptive Statistics 汇总计算描述性统计

意思就是像：sum,mean,maxa,min 之类的都是汇总计算描述性统计。

```
df1.mean(axis= 1,skipna= False) #不能忽略NA\\
df1.idxmax()\\
df1.cumsum()
df1.describe()
```

3.15 Correlation and Covariance 相关以及协方差

```
import pandas_datareader.data as web
从雅虎网站读取股票收盘价格信息:
all_data = {ticker:web.get_data_yahoo(ticker) for ticker in ['AAPL',
    'IBM', 'MSFT', 'GOOG']}
price = DF({ticker: data['Adj Close'] for ticker, data in
    all_data.items()}) #收盘价
volume = DF({ticker: data['Volume'] for ticker,data in
    all_data.items()}) #成交量。

变换量百分比:
returns = price.pct_change()

AAPL IBM  MSFT GOOG
Date
2020-08-03 0.025198 0.011144 0.056241 -0.005739
2020-08-04 0.006678 0.012308 -0.015009 -0.006430
```

count	Number of non-NA values
describe	Compute set of summary statistics for Series or each DataFrame columr
min, max	Compute minimum and maximum values
argmin, argmax	Compute index locations (integers) at which minimum or maximum va
idxmin, idxmax	Compute index labels at which minimum or maximum value obtained,
quantile	Compute sample quantile ranging from 0 to 1
sum	Sum of values
mean	Mean of values
median	Arithmetic median (50% quantile) of values
mad	Mean absolute deviation from mean value
prod	Product of all values
var	Sample variance of values
std	Sample standard deviation of values
skew	Sample skewness (third moment) of values
kurt	Sample kurtosis (fourth moment) of values
cumsum	Cumulative sum of values
cummin, cummax	Cumulative minimum or maximum of values, respectively
cumprod	Cumulative product of values
diff	Compute first arithmetic difference (useful for time series)
pct_change	Compute percent changes

分位数

图 14: 各种统计函数

```
2020-08-05 0.003625 -0.003099 -0.001641 0.005898
2020-08-06 0.034889 0.005341 0.016014 0.017976
2020-08-07 -0.022736 0.003775 -0.017888 -0.003740
计算不同股票涨跌协方差，相关性：
returns['IBM'].corr(returns['GOOG'])

returns['IBM'].cov(returns['GOOG'])
产生协方差和相关矩阵：
returns.cov()
returns.corr()
returns.corrwith(returns.IBM)
计算价格涨跌和成交量的关系：
returns.corrwith(volume)
```

3.16 Unique Values, Value Counts, and Membership

Method	Description
isin	Compute boolean array indicating whether each Series value is contained in the passed sequence of values
match	Compute integer indices for each value in an array into another array of distinct values; helpful for data alignment and join-type operations
unique	Compute array of unique values in a Series, returned in the order observed
value_counts	Return a Series containing unique values as its index and frequencies as its values, ordered count in descending order

图 15: unique,valuecount,membership

4 Data Loading, Storage, and File Formats

pandas features a number of functions for reading tabular data as a DataFrame object. Table 6-1 summarizes some of them, though `read_csv` and `read_table` are likely the ones you' ll use the most.

4.1 Reading and Writing Data in Text Format

indexing: can treat one or more columns as the returened DataFrame
type inference an data conversion:

Function	Description
read_csv	Load delimited data from a file, URL, or file-like object; use comma as default delimiter
read_table	Load delimited data from a file, URL, or file-like object; use tab ('\t') as default delimiter
read_fwf	Read data in fixed-width column format (i.e., no delimiters)
read_clipboard	Version of read_table that reads data from the clipboard; useful for converting tables from web pages
read_excel	Read tabular data from an Excel XLS or XLSX file
read_hdf	Read HDF5 files written by pandas
read_html	Read all tables found in the given HTML document
read_json	Read data from a JSON (JavaScript Object Notation) string representation
read_msgpack	Read pandas data encoded using the MessagePack binary format
read_pickle	Read an arbitrary object stored in Python pickle format

图 16: parsingfunction in pandas

detectime parsing

iterating

unclean data issues

```
df = pd.read_csv('/content/sample_data/ex1.csv')
pd.read_table('/content/sample_data/ex1.csv', delimiter=',')
pd.read_csv('/content/sample_data/ex2.csv', header=None)
pd.read_csv('/content/sample_data/ex2.csv', names=list('abcde'))
names = ['a','b','c','d','message']
pd.read_csv('/content/sample_data/ex2.csv',names=names,
    index_col='message')# 用names中的message列做为index。

parsed = pd.read_csv('/content/sample_data/csv_mindex.csv',
    index_col=['key1','key2']) #用两列做二级索引。

list(open('/content/sample_data/ex3.txt'))
result = pd.read_table('/content/sample_data/ex3.txt',sep='\s+')

pd.read_csv('/content/sample_data/ex4.csv', skiprows=[0,2,3])

result = pd.read_csv('/content/sample_data/ex5.csv', na_values=['NULL'])
result = pd.read_csv('/content/sample_data/ex5.csv',
    na_values={'message':['foo', 'NA'], 'something':['two']})
```

```
pd.read_csv('ex6.csv', nrows=5)

chunker = pd.read_csv('ex6.csv', chunksize=1000) # chunker is iter
tot = Series([])
for piece in chunker:
  tot =tot.add(piece['key'].value_counts(), fill_value=0)
      #每个块1000个，每次统计1000个，将最后的结果相加，就是总体的。

tot = tot.sort_values(ascending=False)
```

4.2 Writing Data to Text Format

```
  import sys
  data.to_csv(sys.stdout, sep="|")
  data.to_csv(sys.stdout, sep='|', na_rep='NULL')
  存储后没有索引:
  data.to_csv(sys.stdout, index=False,header=False, sep='|',
      na_rep='null')
  有设置列号:
  data.to_csv(sys.stdout, index=False, columns=['a','b'])
  设置时间索引:
  dates = pd.date_range('1/1/2000', periods=7)
  ts = pd.Series(np.arange(7), index=dates)
  ts.to_csv(sys.stdout)
```

4.3 Working with Delimited Formats 重点

```
import csv后读取:
  f = open('ex7.csv')
  reader = csv.reader(f)
  for line in reader:
    print(line)
reader方法:
  with open('ex7.csv') as f:
    lines = list(csv.reader(f))
```

Argument	Description
path	String indicating filesystem location, URL, or file-like object
sep or delimiter	Character sequence or regular expression to use to split fields in each row
header	Row number to use as column names; defaults to 0 (first row), but should be None if there is no header row
index_col	Column numbers or names to use as the row index in the result; can be a single name/number or a list of them for a hierarchical index
names	List of column names for result, combine with header=None

(a) 1

Argument	Description
skiprows	Number of rows at beginning of file to ignore or list of row numbers (starting from 0) to skip.
na_values	Sequence of values to replace with NA.
comment	Character(s) to split comments off the end of lines.
parse_dates	Attempt to parse data to datetime; False by default. If True, will attempt to parse all columns. Otherwise can specify a list of column numbers or name to parse. If element of list is tuple or list, will combine multiple columns together and parse to date (e.g., if date/time split across two columns).
keep_date_col	If joining columns to parse date, keep the joined columns; False by default.
converters	Dict containing column number of name mapping to functions (e.g., {'foo': f} would apply the function f to all values in the 'foo' column).
dayfirst	When parsing potentially ambiguous dates, treat as international format (e.g., 7/6/2012 -> June 7, 2012); False by default.
date_parser	Function to use to parse dates.
nrows	Number of rows to read from beginning of file.
iterator	Return a TextParser object for reading file piecemeal.
chunksize	For iteration, size of file chunks.
skip_footer	Number of lines to ignore at end of file.
verbose	Print various parser output information, like the number of missing values placed in non-numeric columns.
encoding	Text encoding for Unicode (e.g., 'utf-8' for UTF-8 encoded text).
squeeze	If the parsed data only contains one column, return a Series.
thousands	Separator for thousands (e.g., ',' or '.').

(b) 2

图 17: read_csv 的参数

```
  header,values = lines[0], lines[1:]
  data_dict = {h:v for h,v in zip(header,zip(*values))}
      ######################
  data_dict
```

自定以分隔spliter:

```
#simple subclass csv.Dialect
class my_dialect(csv.Dialect):
  lineterminator = '\n'
  delimiter = ';'
  quotechar = '"'
  quoting = csv.QUOTE_MINIMAL
f = open('ex7.csv')
reader = csv.reader(f, dialect=my_dialect)
```

4.4 JSON Data

读取:

```
  import json
  result = json.loads(obj)
  result
```

装载（载入）:

```
  asjson = json.dumps(result)
```

因为json 是字典形式的, 所以天生就可以转换成dataframe

用字典的key来做columns:

```
siblings = DF(result['siblings'], columns= ['age','name','pets'])
  age  name pets
0  30 Scott [Zeus, Zuko]
1  38 Katie [Sixes, Stache, Cisco]
```

pandas.read_json 可以自动的将 JSON 数据按特定顺序排列成 dataframe

```
data = pd.read_json('example.json')
```

data.to_json() 也可以将 dataframe 转换成为 json

4.5 XML and HTML : Web Scraping

```
读取银行倒闭信息:
  tables = pd.read_html('fdic_failed_bank_list.html')
  时间信息转换标准时间格式:
  cl = pd.to_datetime(x['Closing Date'])
  从是按列, 抽取年份:
  cl.dt.year
  统计倒闭数量:
  cl.dt.year.value_counts()
```

4.6 Parsing XML with lxml.objectify

分层，嵌套的文件形式。

```
from lxml import objectify
path = 'Performance_MNR.xml'
parsed = objectify.parse(open(path)) ##############important way.
root = parsed.getroot()

data = []
skip_fields = ['PARENT_SEQ', 'INDICATOR_SEQ','DESIRED_CHANGE',
    'DECIMAL_PLACES']

for elt in root.INDICATOR:
  el_data = {}
  for child in elt.getchildren():
    if child.tag in skip_fields:
      continue
    el_data[child.tag] = child.pyval #child.tag标签和pyval
        值, 存储。作为一个字典。
  data.append(el_data)
```

4.7 Binary Data Formats

pandas 有两个函数：to_pickle and read_pickle 可以将数据简单的存储为二进制文件。

```
frame = pd.read_csv('ex1.csv')
print(frame)
frame.to_pickle('ex1_frame_pickle')
pd.read_pickle('ex1_frame_pickle')

a  b  c  d  message
0  1  2  3  4  hello
1  5  6  7  8  world
2  9  10 11 12 foo
```

4.8 Using HDF5 Format

```
frame = pd.DataFrame({'a':np.random.randn(100)})
store = pd.HDFStore('mydata.h5')
store['obj1'] = frame
obj2为。
store.put('obj2',frame,format='table')
frame.to_hdf('mydata.h5','obj3',format='table')
简便的方法:
pd.read_hdf('mydata.h5', 'obj3', where=['index > 10 and index <
    20'],mode='r+')
```

4.9 Reading Microsoft Excel Files

```
xlsx = pd.ExcelFile('ex1.xlsx')
pd.read_excel(xlsx,'Sheet1')

简便的方法:
frame = pd.read_excel('ex1.xlsx', 'Sheet1')
写入:
writer = pd.ExcelWriter('ex2.xlsx')
frame.to_excel(writer, 'Sheet1')
writer.save()
简便的写法:
frame.to_excel('ex2.xlsx','Sheet2')
```

4.10 Interacting with Web APIs

requests.get() 网址信息，得到.json() 对象。所以可以直接于 daframe 转换。

```
import requests
url = 'https://api.github.com/repos/pandas-dev/pandas/issues'
resp = requests.get(url)
issues = DF(data, columns=['number', 'title', 'labels',
    'state','url'])
issues.head()
```

4.11 Interacting with Databases

与数据库对象进行互动：

```
import sqlite3
query = '''
CREATE TABLE test
(a VARCHAR(20), b varchar(20), c real, d integer);'''
创建和连接数据库:
con = sqlite3.connect('mydata.sqlite')
执行创建表:
con.execute(query)
con.commit()

data = [('Atlanta', 'Georgia', 1.25, 6),
('Tallahassee', 'Florida', 2.6, 3),
('Sacramento', 'California', 1.7, 5)]
stmt = "INSERT INTO test VALUES(?, ?, ?, ?)"
多次执行命令:
将data插入到test表格:
con.executemany(stmt, data)
con.commit()

读取数据库:
cursor = con.execute('select * from test')
rows = cursor.fetchall()
```

魔术方法：

```
import sqlalchemy as sqla

db = sqla.create_engine('sqlite:///mydata.sqlite')
pd.read_sql('select * from test',db)
```

5 数据清洗与准备

5.1 Handling Missing Data

```
string_data.isnull()
丢弃缺失数据：
data[data.notnull()]
data.dropna(axis, how,thresh, subset,inplace)
（可以按行丢弃或者列丢弃）

Filling In Missing Data 填充
```

Argument	Description
dropna	Filter axis labels based on whether values for each label have missing data, with varying thresholds for how much missing data to tolerate.
fillna	Fill in missing data with some value or using an interpolation method such as 'ffill' or 'bfill'.
isnull	Return boolean values indicating which values are missing/NA.
notnull	Negation of isnull.

图 18: 缺失数据函数

```
df.fillna(value=none, method=none, axis=none,inplace, limite,downcast)
value 可以指定填充的值，也可以指定相应的填充索引列或者行。
```

5.2 Data Transformation

5.2.1 数据冗余

duplicated(subset, keep:first,last) 标记冗余数据的位置，只有一个会标记为 true，其他为 false 表示冗余。

Argument	Description
value	Scalar value or dict-like object to use to fill missing values
method	Interpolation; by default 'ffill' if function called with no other arguments
axis	Axis to fill on; default axis=0
inplace	Modify the calling object without producing a copy
limit	For forward and backward filling, maximum number of consecutive periods to fill

图 19: 数据填充函数

```
data.duplicated()
data.drop_duplicates() 丢球冗余数据。
data.drop_duplicates(['x', 'y'], keep='first'/'last')
    丢弃x列的冗余数据，其他列冗余不冗余不管。
```

5.2.2 Transforming Data Using a Function or Mapping

```
data:
food ounces
0  bacon 4.0
1  pulled pork 3.0
2  bacon 12.0
3  Pastrami 6.0
4  corned beef 7.5
5  Bacon 8.0
6  pastrami 3.0
7  honey ham  5.0
8  nova lox 6.0

  meat_to_animal = {
  'bacon':'pig',
  'pulled pork':'pig',
  'pastrami':'cow',
  'corned beef':'cow',
  'honey ham':'pig',
  'nova lox':'salmon'
```

```
    }
    lowercased = data['food'].str.lower()

data['animal'] = lowercased.map(meat_to_animal)
data['meat2'] = data['food'].map(lambda x: meat_to_animal[x.lower()])
```

5.2.3 Replicing Values

数据值的替换。关键字：replace

```
data.replace(-999, np.nan)
data.replace([-999,-1000], ['xx','yy'])
data.replace({-999:'xxxx', -1000:'yyyy'})
```

5.2.4 Renaming Axis Indexes

可以用 map

```
tansform = lambda x:x[:4].upper()
data.index.map(tansform)
data.index = data.index.map(lambda x:x[:4].upper())
data.columns = data.columns.map(lambda x:x[:].upper())
k = data.rename(index = str.title, columns= str.lower)
data.rename(index={'OHIO':'fuck'}, columns={'ONE':111})
```

5.2.5 Discretization and Binning

```
cats = pd.cut(ages, bins)
cats.codes
cats.categories
pd.value_counts(cats)
```

5.2.6 检测和过滤异常

```
col[np.abs(col) > 3]
data[(np.abs(data) >3).any(axis = 1)]
```

```
data[np.abs(data) >3] = np.sign(data)*3
```

5.2.7 排列和随机抽样

```
df = pd.DataFrame(np.arange(5*4).reshape((5,4)))
从1-5随机抽取:
sampler = np.random.permutation(5)
按行随机抽取:
df.take(sampler)
df.iloc[sampler]

pandas里也有随机:
df.sample(n=2) 随机抽两行。

可以重复抽取:
choices = Series([5, 7, -1, 6, 4])
draws = choices.sample(n=10, replace= True)
```

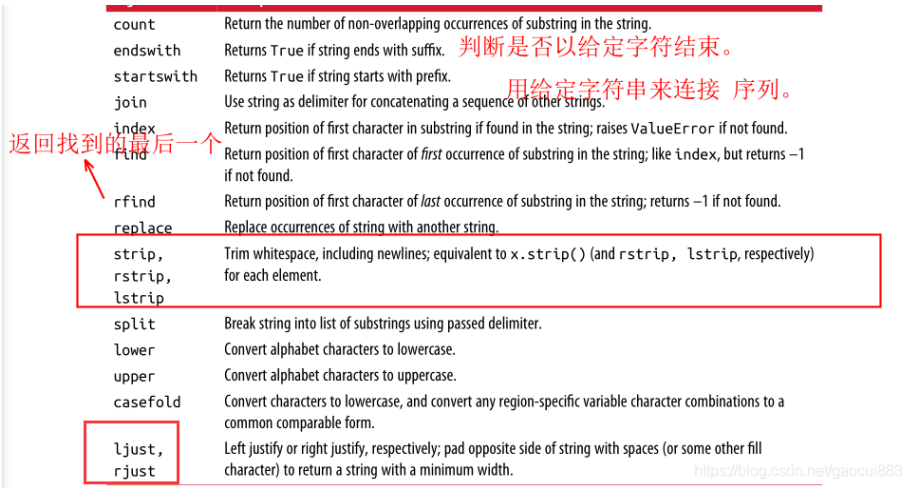
5.2.8 指示器和哑变量

```
df = pd.DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'b'],
'data1': range(6)})
pd.get_dummies(df['key'],prefix='key')
a  b  c
0  0  1  0
1  0  1  0
2  1  0  0
3  0  0  1
4  1  0  0
5  0  1  0
   df_with_dummy = df[['data1']].join(dummies)


不用get_dummies的例子:
mnames = ['movie_id', 'title', 'genres']
movies = pd.read_table('movies.dat', sep='::',header=None, names=mnames)
```

```
# 我们现在想做的工作是什么？？？
#
    我们需要制作一个dataframe，用来显示这些不同的电影的风格，纵坐标是电影，横坐标风格，1表示该电影是此风格

# 首先找出一共的风格数，并对其进行统计

all_genres = []
for x in movies.genres:
all_genres.extend(x.split('|'))
    #将各个风格进行分割，并且添加到all——genres中genres =
    pd.unique(all_genres)
from pandas import DataFrame as DF
zero_matrix = np.zeros((len(movies), len(genres)))
dummies = DF(zero_matrix, columns = genres)

# 下一步就是根据电影中的gernes 进行填写即可。

for i,gen in enumerate(movies.genres):
   indices = dummies.columns.get_indexer(gen.split('|'))
       #得到电影的风格索引。
   #根据这个索引进行填写为1.
   dummies.iloc[i, indices] = 1.0

movies_windic = movies.join((dummies.add_prefix('Genre_')))
```

5.3 string 相关操作

```
   str.split()
   reslut2 = "::".join(pieces)
   val.index(',')
   val.find(',')
   val2 = val.replace(",", '::::')
   val.replace(",","") # 用来删除逗号。
   val.rjust(14, '&')
   val.ljust(15, "*")
   data.str.contains('gmail')
```



count	Return the number of non-overlapping occurrences of substring in the string.
endswith	Returns True if string ends with suffix.
startswith	Returns True if string starts with prefix.
join	Use string as delimiter for concatenating a sequence of other strings.
index	Return position of first character in substring if found in the string; raises ValueError if not found.
rfind	Return position of first character of *first* occurrence of substring in the string; like index, but returns –1 if not found.
rfind	Return position of first character of *last* occurrence of substring in the string; returns –1 if not found.
replace	Replace occurrences of string with another string.
strip, rstrip, lstrip	Trim whitespace, including newlines; equivalent to x.strip() (and rstrip, lstrip, respectively) for each element.
split	Break string into list of substrings using passed delimiter.
lower	Convert alphabet characters to lowercase.
upper	Convert alphabet characters to uppercase.
casefold	Convert characters to lowercase, and convert any region-specific variable character combinations to a common comparable form.
ljust, rjust	Left justify or right justify, respectively; pad opposite side of string with spaces (or some other fill character) to return a string with a minimum width.

图 20: str 相关操作

cat	Concatenate strings element-wise with optional delimiter
contains	Return boolean array if each string contains pattern/regex
count	Count occurrences of pattern
extract	Use a regular expression with groups to extract one or more strings from a Series of strinc will be a DataFrame with one column per group
endswith	Equivalent to x.endswith(pattern) for each element
startswith	Equivalent to x.startswith(pattern) for each element
findall	Compute list of all occurrences of pattern/regex for each string
get	Index into each element (retrieve *i*-th element)
isalnum	Equivalent to built-in str.alnum
isalpha	Equivalent to built-in str.isalpha
isdecimal	Equivalent to built-in str.isdecimal
isdigit	Equivalent to built-in str.isdigit
islower	Equivalent to built-in str.islower
isnumeric	Equivalent to built-in str.isnumeric
isupper	Equivalent to built-in str.isupper
join	Join strings in each element of the Series with passed separator
len	Compute length of each string
lower, upper	Convert cases; equivalent to x.lower() or x.upper() for each element

图 21: str 和正则表达相关

6 数据的整理 join，combine，reshpae

6.1 多重索引

```
data = Series(np.random.randn(9), index = [['a', 'a', 'a', 'b', 'b',
    'c', 'c', 'd', 'd'],
[1, 2, 3, 1, 3, 1, 2, 2, 3]])

data.unstack()
data.unstack().stack()

frame = DataFrame(np.arange(12).reshape((4, 3)),
index=[['a', 'a', 'b', 'b'], [1, 2, 1, 2]],
columns=[['Ohio', 'Ohio', 'Colorado'],
['Green', 'Red', 'Green']])


交换索引级别:
swaplevel
x.swaplevel('state','color',axis=1)
frame.swaplevel('key1','key2',axis=0)
排序:
frame.sort_index(level=1,axis=0,ascending=True, inplace= False,
    kind='quicksort')

frame.sort_index(level=0,axis=0,ascending=True, inplace= False,
    kind='quicksort')

按索引级别分层统计:

frame.sum(level='key2')
```

6.2 setindex,resetincex

用列c做index索引:

```
frame2 = frame.set_index(['c'])
```

多层索引:

```
   frame2 = frame.set_index(['c','d'])
```
成为索引的同时丢弃原来：
```
  frame3 = frame.set_index(['c','d'], drop=False)
```
默认将所有的index转换成index：
```
  frame4 = frame2.reset_index()
```
指定转换：
```
  frame4 = frame2.reset_index(level=1, drop= False)
```

6.3 三大链接神奇：merge,jion,concat

merge 是行方向，关键字相关链接，也就是根据索引或者列进行链接。join 简单的将新的属性加入到原来的列表中即可。没有配对的设定为 NAn

`'inner'`	Use only the key combinations observed in both tables
`'left'`	Use all key combinations found in the left table
`'right'`	Use all key combinations found in the right table
`'output'`	Use all key combinations observed in both tables together

Table 8-2. merge function arguments

Argument	Description
`left`	DataFrame to be merged on the left side.
`right`	DataFrame to be merged on the right side.
`how`	One of `'inner'`, `'outer'`, `'left'`, or `'right'`; defaults to `'inner'`.
`on`	Column names to join on. Must be found in both DataFrame objects. If not specified and no other join keys given, will use the intersection of the column names in `left` and `right` as the join keys.
`left_on`	Columns in `left` DataFrame to use as join keys.
`right_on`	Analogous to `left_on` for `left` DataFrame.
`left_index`	Use row index in `left` as its join key (or keys, if a MultiIndex).
`right_index`	Analogous to `left_index`.
`sort`	Sort merged data lexicographically by join keys; `True` by default (disable to get better performance in some cases on large datasets).
`suffixes`	Tuple of string values to append to column names in case of overlap; defaults to `('_x', '_y')` (e.g., if `'data'` in both DataFrame objects, would appear as `'data_x'` and `'data_y'` in result).
`copy`	If `False`, avoid copying data into resulting data structure in some exceptional cases; by default always copies.
`indicator`	Adds a special column `_merge` that indicates the source of each row; values will be `'left_only'`,

图 22: merge 相关参数

concat 列方向，竖直方向链接不同的表。

```
pd.merge(df1,df2) # 默认对相同的名字的coloumn进行链接。
# 也可对链接的名进行指定。
pd.merge(df1,df2, on= 'key')

df3 = pd.DataFrame({'lkey': ['b', 'b', 'a', 'c', 'a', 'a', 'b'],
'data1': range(7)})

df4 = pd.DataFrame({'rkey': ['a', 'b', 'd'],
'data2': range(3)})

pd.merge(df3,df4, left_on='lkey',right_on='rkey', how = 'inner')

上边都是在一个列的基础上进行链接，也可以指定两个列：
pd.merge(left,right, how='outer', on=['key1', 'key2'])

如果非链接的多个列相同，那么可以设定后缀。
pd.merge(left,right, how='outer', on=['key1'], suffixes=('_mother',
    '_dady'))

按index和列名链接：
pd.merge(left1,right1, left_on='key', right_on= None, left_index=
    False, right_index= True, how='outer')

concat:

pd.concat([s1,s2,s3], axis=1)
链接后还可以设定高级索引：
pd.concat([df1,df2], axis=1, keys=['level1', 'level2'])
# 也可以传入字典，字典key 默认作为多层索引。
pd.concat({'level1':df1, 'level2':df2}, axis=1)
pd.concat([df1,df2], ignore_index=True, axis=0, keys=['df1','df2'])
# 我们看到一个特别的情况， keys 不起作用了，这是因为 ignore_index.
```

6.4 相同类型的表格的合并

```
a = Series([np.nan, 2.5, np.nan, 3.5, 4.5, np.nan], index =['f', 'e',
    'd', 'c', 'b', 'a'])
```

```
b = Series(np.arange(len(a), dtype= np.float64), index=['f', 'e',
    'd', 'c', 'b', 'a'])

np.where(pd.isnull(a), b, a) # pd.isnull 是条件，真选b，假选a
等价于:
b[:-2].combine_first(a[2:])
dataframe类似:
df1 = DataFrame({'a': [1., np.nan, 5., np.nan],
'b': [np.nan, 2., np.nan, 6.],
'c': range(2, 18, 4)})

df2 = DataFrame({'a': [5., 4., np.nan, 3., 7.],
'b': [np.nan, 3., 4., 6., 8.]})
df1.combine_first(df2)
```

6.5 reshpae,pivoting

```
result = data.stack()
行列进行交换:
df.unstack('state').stack('side')
透视表:
pivoted = ldata.pivot('date', 'item', 'value')
```

7 绘图和可视化

7.1 初级教程

```
    fig = plt.figure()
    ax1 = fig.add_subplot(2,2,1)
    ax2 = fig.add_subplot(2,2,2)
    ax3 = fig.add_subplot(2,2,3)

    plt.plot(np.random.randn(50).cumsum(), 'k--')
        #而这样的，则会默认在最后一个figure的最后一个subplot进行绘图。
```

```
_ = ax1.hist(np.random.randn(100), bins=20, color='k', alpha=0.3)
    # alpha ??? 透明度。
ax2.scatter(np.arange(30), np.arange(30)+3*np.random.randn(30))

调整子图间距:
subplots_adjust(left=None, bottom=None, right=None, top=None,
    wspace=None, hspace=None)


fig,axes = plt.subplots(2,1)

axes[0].plot(np.random.randn(20),np.random.randn(20),'g--') #
    绿色，虚线。折线。
# more explictily as
axes[1].plot(np.random.randn(30),np.random.randn(30),
    linestyle='--', color='r')。

from numpy.random import randn
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.plot(randn(1000).cumsum(),'c',label='one')

ax.plot(randn(1000).cumsum(), 'k--', label='two')
ax.plot(randn(1000).cumsum(), 'r.', label='three')
ax.legend(loc='best')
```

7.2 Annotations and Drawing on a Subplot

```
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
data = pd.read_csv('spx.csv', index_col=0, parse_dates = True) #
    index_col 哪一列来做索引。parse_dates
spx = data['SPX']
spx.plot(ax=ax, style='k-')

crisis_data = [(datetime(2007, 10, 11), 'Peak of bull market'),
(datetime(2008, 3, 12), 'Bear Stearns Fails'),
```

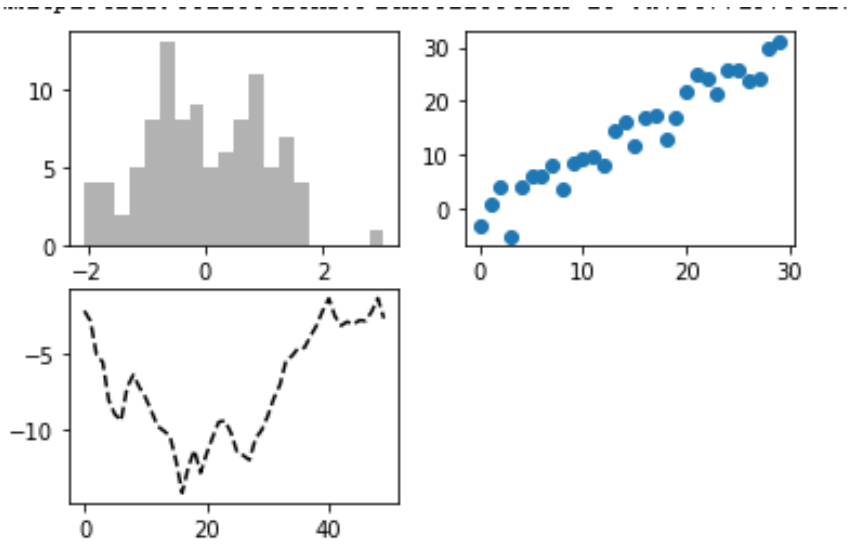


图 23: fig-plot1

```
Using matplotlib backend: agg
<matplotlib.legend.Legend at 0x7f477c940160>
```

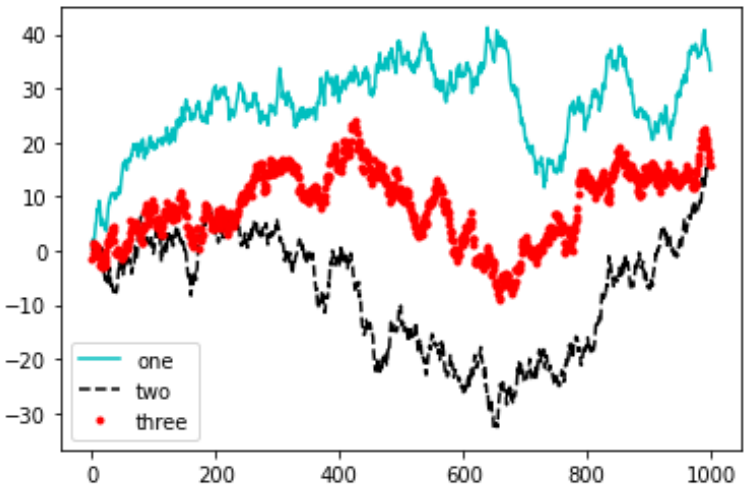


图 24: fig-plot2

```
  (datetime(2008, 9, 15), 'Lehman Bankruptcy')]

  for date,label in crisis_data:
    ax.annotate(label, xy=(date, spx.asof(date)+75), #spx.asof(date)
        ????
    xytext = (date,spx.asof(date)+225),
    arrowprops = dict(facecolor='black', headwidth=4,
        width=2,headlength=4),
    horizontalalignment='left',#水平对齐
    verticalalignment='top') # 垂直对齐。
  ax.set_xlim(['1/1/2007','1/1/2011'])
  ax.set_ylim([600, 1800])
  ax.set_title('Important dates in the 2008-2009 financial crisis')
```

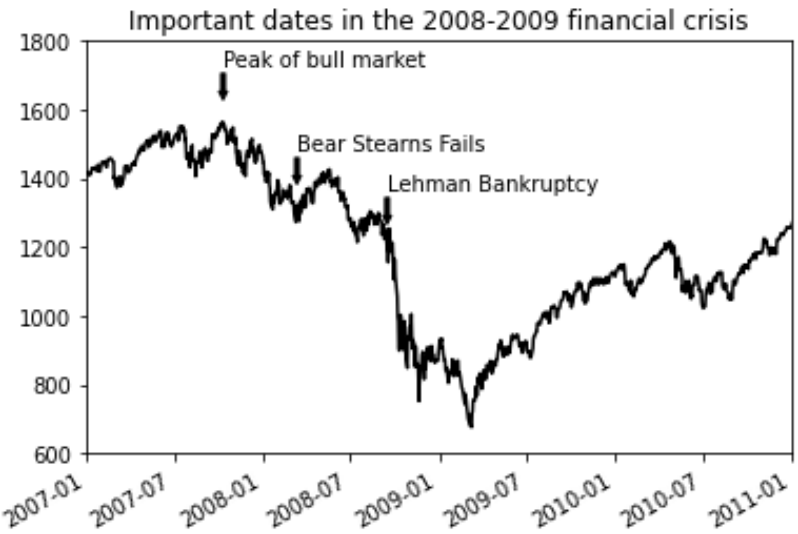


图 25: fig-plot3

添加图形对象：

```
  rect = plt.Rectangle(xy = (0.2,0.75), width= 0.4, height= 0.5,
      alpha=0.3)
  circ = plt.Circle(xy=(0.7,0.2), radius=0.18, color='r', alpha=0.3)
  pgon = plt.Polygon(xy=[[0.15, 0.15], [0.35, 0.4], [0.2, 0.6]],
      color='c', alpha = 0.5)
```

```
ax.add_patch(rect)
ax.add_patch(circ)
ax.add_patch(pgon)
```

7.3 保存图片

Table 9-2. Figure.savefig options

Argument	Description
`fname`	String containing a filepath or a Python file-like object. The figure format is inferred from the file extension (e.g., `.pdf` for PDF or `.png` for PNG)
`dpi`	The figure resolution in dots per inch; defaults to 100 out of the box but can be configured
`facecolor, edgecolor`	The color of the figure background outside of the subplots; `'w'` (white), by default
`format`	The explicit file format to use (`'png'`, `'pdf'`, `'svg'`, `'ps'`, `'eps'`, ...)
`bbox_inches`	The portion of the figure to save; if `'tight'` is passed, will attempt to trim the empty space around the figure

图 26: fig-plot4

```
plt.savefig(buffer)
```

7.4 用 pandas 和 seaborn 绘图

series 和 pandas 都可以直接用来绘图：

```
# dataframe 默认会将每列化成线性图。

# .plot() 还有很多其他options: use_index, xticks,xlim y_ticks,ylim.


df = DataFrame(np.random.randn(10,4).cumsum(0),
    columns=['a','b','c','d'], index=np.arange(0,100,10))
df.plot.bar()
堆叠bar图:
df.plot.bar(stacked = True)

seaborn:
需要指定对应data,x,y
sbn.barplot(x='tip_pct', y='day', data=tips, orient='h')
```

Table 9-3. Series.plot method arguments

Argument	Description
`label`	Label for plot legend
`ax`	matplotlib subplot object to plot on; if nothing passed, uses active matplotlib subplot
`style`	Style string, like `'ko--'`, to be passed to matplotlib
`alpha`	The plot fill opacity (from 0 to 1)
`kind`	Can be `'area'`, `'bar'`, `'barh'`, `'density'`, `'hist'`, `'kde'`, `'line'`, `'pie'`
`logy`	Use logarithmic scaling on the y-axis
`use_index`	Use the object index for tick labels
`rot`	Rotation of tick labels (0 through 360)
`xticks`	Values to use for x-axis ticks
`yticks`	Values to use for y-axis ticks
`xlim`	x-axis limits (e.g., `[0, 10]`)
`ylim`	y-axis limits
`grid`	Display axis grid (on by default)

各种图

可以关闭轴网格显示

图 27: fig-plot5

Table 9-4. DataFrame-specific plot arguments

Argument	Description
`subplots`	Plot each DataFrame column in a separate subplot
`sharex`	If `subplots=True`, share the same x-axis, linking ticks and limits
`sharey`	If `subplots=True`, share the same y-axis
`figsize`	Size of figure to create as tuple
`title`	Plot title as string
`legend`	Add a subplot legend (`True` by default)
`sort_columns`	Plot columns in alphabetical order; by default uses existing column order

图 28: fig-plot6

```
Histograms and Density Plots直方图和密度图:
   tips['tip_pct'].plot.hist(bins=30)
   tips['tip_pct'].plot.density()
   sbn.distplot(values, bins=100, color='r')
Scatter or Point Plots:
    sbn.regplot('m1','unemp', data=trans_data)
    sbn.pairplot(trans_data, diag_kind='kde', plot_kws={'alpha':0.5})
```

7.5 多维数据的绘图

```
   tips.head():
   total_bill tip  smoker  day   time size tip_pct
   0  16.99 1.01 No Sun Dinner  2  0.063204
   1  10.34 1.66 No Sun Dinner  3  0.191244
   2  21.01 3.50 No Sun Dinner  3  0.199886
   3  23.68 3.31 No Sun Dinner  2  0.162494
   4  24.59 3.61 No Sun Dinner  4  0.172069

sbn.factorplot(x='day', y='tip_pct', hue='time', col='smoker',
     kind='bar', data=tips[tips.tip_pct < 1])

sbn.factorplot(x='day', y='tip_pct', row='time',
col='smoker',
kind='bar', data=tips[tips.tip_pct < 1])

sbn.factorplot(x='tip_pct', y='day', kind='box',
data=tips[tips.tip_pct < 0.5])

sbn.factorplot ======sbn.catplot()
```

8 Data Aggregation and Group Operations

内容:对 pandas 对象进行 split, 计算统计信息,应用 groupby,pivot_table, 计算分位数分析信息。

```
/usr/local/lib/python3.6/dist-packages/seaborn/categorical.py:3666: UserWarning: The `factorplot` function has b
  warnings.warn(msg)
<seaborn.axisgrid.FacetGrid at 0x7ff3bb82b0f0>
```

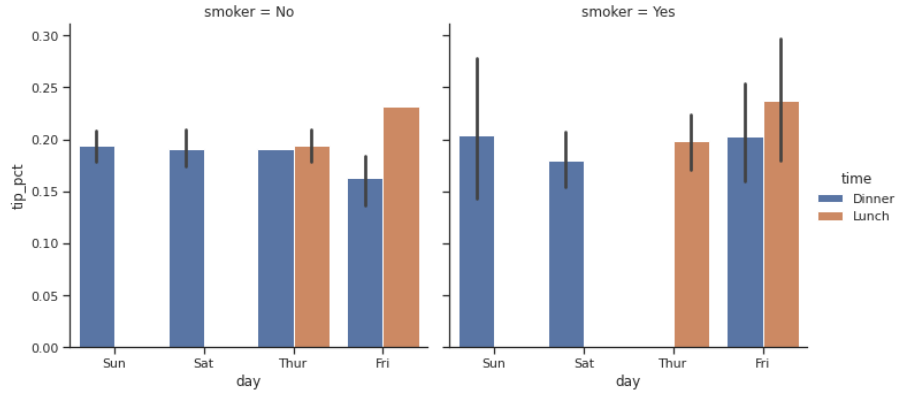


图 29: fig-plot7

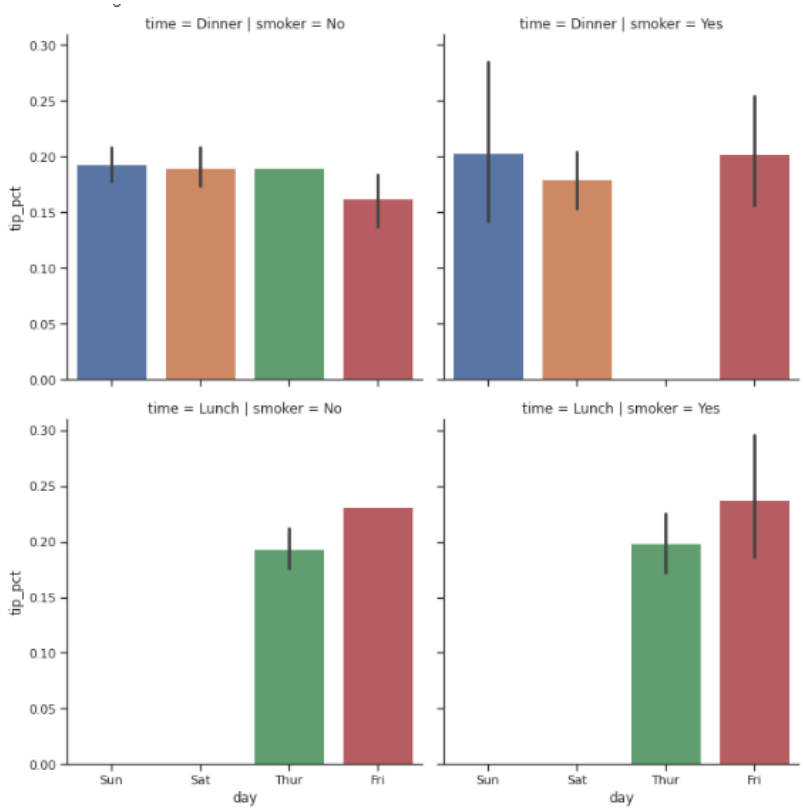


图 30: fig-plot8

```
<seaborn.axisgrid.FacetGrid at 0x7ff3bafeeac8>
```

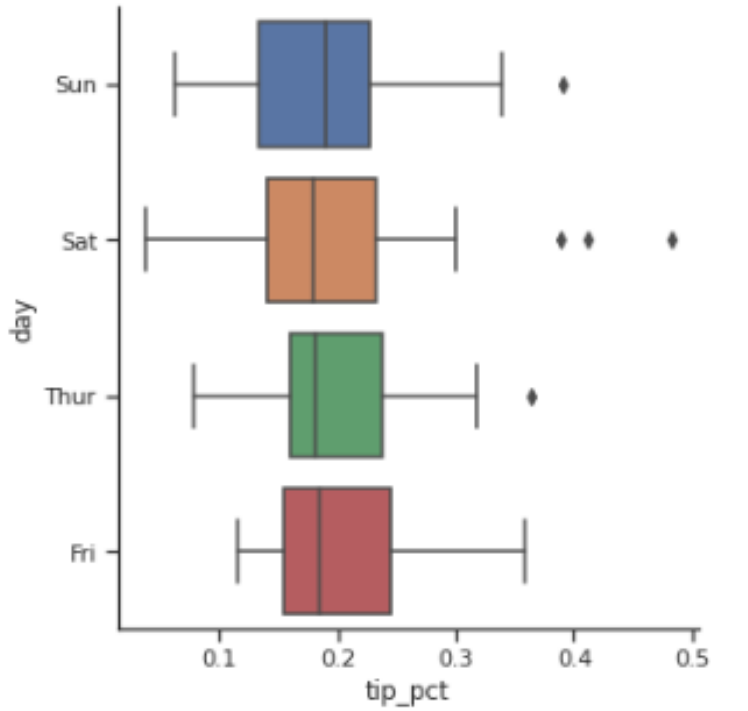


图 31: fig-plot9

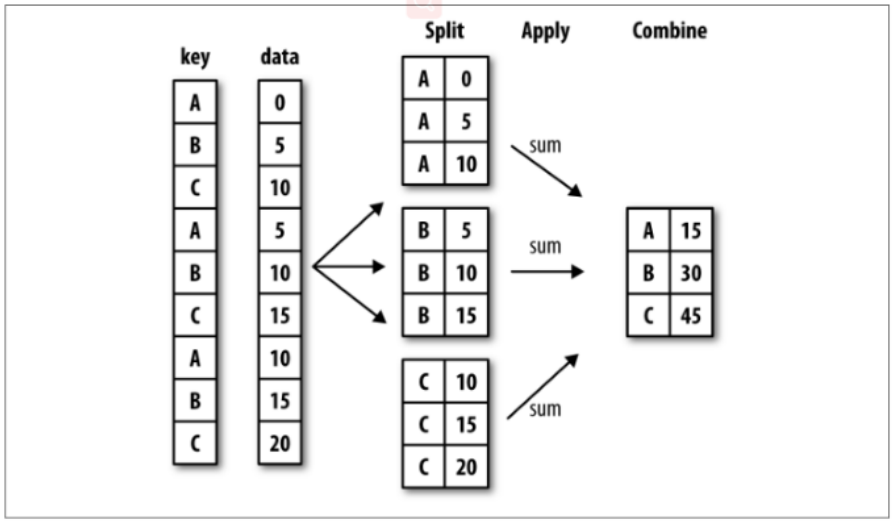


图 32: fig-plot10

```
grouped = df['data1'].groupby(df['key1'])
grouped
grouped.mean()
df['data1'].groupby([df['key1'],df['key2']]).mean()
key1 key2
a    one    0.296851
two    0.339220
b    one   -0.959031
two   -0.288006

df.groupby([df['key1'],df['key2']]).mean()
df.groupby(['key1','key2']).size()
可以迭代性:
for name, group in df.groupby(['key1']):
print(name)
print(group)


for (k1,k2), group in df.groupby(['key1','key2']):
print((k1,k2))
```

```
print(group)

pieces = dict(list(df.groupby('key1')))

注意以下两个的不同:
df.groupby('key1')['data1']
df.groupby('key1')[['data1']]

df.groupby(['key1','key2'])['data2'].mean()

mapping是一个字典:
by_column = people.groupby(mapping, axis=1)

series_column = people.groupby(map_series, axis=1).sum()

应用函数:
people.groupby(len).sum() #默认调用index，返回的数值用作新的index。
people.groupby([len, key_list]).min()

hier_df.groupby(level='cty', axis=1).mean()

def peaktopeak(arr):
return arr.max()-arr.min()

grouped.agg(peaktopeak) # 注意agg或者aggregate
grouped.describe()
grouped = tips.groupby(['day', 'smoker'])
grouped_pct = grouped['tip_pct']
grouped_pct

grouped_pct.aggregate('mean')
grouped_pct.agg(['mean', 'std', peaktopeak])
# 如果我们要顺便改掉结果名呢？？？

# 传递一个元组的列表。
# 元组的第一个元素就是新的列名，第二个参数就是函数名。

grouped_pct.aggregate([('fuck1','mean'), ('fuck2','std'), ('fuck3',
    peaktopeak)])
```

```
# 如果是个DataFrame 呢。也就多个列。

functions = ['count', 'mean', 'max']
result = grouped['tip_pct', 'total_bill'].agg(functions) #
    对两列使用多个函数。
grouped.aggregate({'tip':[('fuck1','max'),('fuck2','min'),('fuck3','mean')],
    'size':['sum','std']})

def top(df, n=5, column='tip_pct'):
  return df.sort_values(by=column)[-n:] #找出最大的五个。
tips.groupby('smoker').apply(top) # 典型的split, apply, merge
tips.groupby('smoker').apply(top, n=3, column='tip') #
    这里的n和column都是top的参数。
a = tips.groupby(['smoker','day']).apply(top, n=1, column='tip')
```

8.1 中位数和桶分析

```
  frame = DataFrame({'data1':np.random.randn(1000),
      'data2':np.random.randn(1000)})
  quartiles = pd.cut(frame.data1, 4)
  quartiles.head()

# 然后计算每部分的最大，最小，均值，以及数据个数。

def get_stats(group):
  return {'min': group.min(),
  'max':group.max(),
  'mean':group.mean(),
  'count':group.count()}

grouped = frame.data2.groupby(quartiles)
x = grouped.apply(get_stats)
x.unstack()
根据group的数统计来填入缺失数据:
data.groupby(group_key).apply(lambda x: x.fillna(x.mean()))
```

```
# 当然，如果提前准备好了相应的groupby对应的要填充的值也可以
fill_values = {'East':0.55555, 'West':-88888}

data.groupby(group_key).apply(lambda x : x.fillna(fill_values[x.name]))
    #x.name


# Hearts, Spades, Clubs, Diamonds #红桃，黑桃。梅花，方片。
suits = ['H', 'S', 'C', 'D']

card_val = (list(range(1, 11)) + [10] * 3) * 4

base_names = ['A'] + list(range(2, 11)) + ['J', 'K', 'Q']

cards = []

for suit in ['H', 'S', 'C', 'D']:

   cards.extend(str(num) + suit for num in base_names)
       #对每个红桃对应每个数字。作为索引。

deck = pd.Series(card_val, index=cards)


def draw(deck, n=5):
   return deck.sample(n)
每个花色抽两张:
get_suit = lambda card: card[-1]
deck.groupby(get_suit).apply(draw, n=2)


   加权取均值:
   get_wavg = lambda x : np.average(x.data, weights= x.weights, axis= 0)

   grouped.apply(get_wavg)


   spx_corr = lambda x: x.corrwith(x['SPX']) #
       其他列对spx列的贡献，或者相关度。
```

```
rets = close_px.pct_change().dropna()
get_year = lambda x: x.year
by_year = rets.groupby(get_year)#默认取索引，也就是年份。
by_year.apply(spx_corr) #以年份来计算对指数收益的贡献。
```

8.2 透视表

```
tips.pivot_table?

重要参数:
values 要统计的透视的信息。
index : 纵向的指标。
columns:  横向指标。
aggfun: 统计透视函数。
fill_value: 要替换na为什么值。

# Signature: tips.pivot_table(values=None, index=None, columns=None,
    aggfunc='mean',
# fill_value=None, margins=False, dropna=True, margins_name='All',
    observed=False) -> 'DataFrame'
# Docstring:
# Create a spreadsheet-style pivot table as a DataFrame.

交叉表:
交叉表是透视表的特例，计算频率。

# pd.crosstab?
# Signature: pd.crosstab(index, columns, values=None, rownames=None,
    colnames=None, aggfunc=None, margins=False, margins_name:
    str='All', dropna: bool=True, normalize=False) -> 'DataFrame'
两个或者多个交叉信息。
主要用来统计个数。
```

9 时间序列

时间索引类型:

时间戳。

时间段。

时间间隔。

累计时间间隔。

```
now = datetime.now()
now.year
delta = datetime(2011,1,5) - datetime(2010,2,3,12,8,3)
from datetime import timedelta
start = datetime(2011,1,7)

x = start + timedelta(13)
print(x)

y = start - 2*timedelta(12)
print(y)

x.isocalendar() # 返回year,week,day.
x.weekday() # 星期4. 0 - 6 星期1到天。
x.ctime() # 返回一个日期字符串。
x.strftime('%Y-%m-%d')


stamp = datetime(2011, 1, 3)
转化成特定格式的字符串:
stamp.strftime('%Y-%m-%d')

# 来识别特定格式的字符串时间序列。
# 关键字strptime str pass time

value = '2011-01-03'

datetime.strptime(value, '%Y-%m-%d')

datestrs = ['7/6/2011', '8/6/2011']

[datetime.strptime(x, '%m-%d-%Y') for x in datestrs]

[datetime.strptime(x, '%m/%d/%Y') for x in datestrs]
```

```
# 第三方库， dateutil 的时间解析库parser.parse
    可以识别大部分人类能够识别的时间字符串格式。

value = '2011-01-03'

from dateutil.parser import parse

value_date = parse(value)

另一种，pandas自带的， pd.to_datetime()

datestrs = ['2011-07-06', '2011-08-06']

import pandas as pd

pd.to_datetime(datestrs)

时间段:
pd.date_range(start=None, end=None, periods=None, freq=None, tz=None,
    normalize=False, name=None, closed=None, **kwargs) ->
    pandas.core.indexes.datetimes.DatetimeIndex

ts.truncate(after='1/9/2011') #截断，裁剪，丢弃的意思。
Signature: ts.truncate(before=None, after=None, axis=None, copy:
    bool=True) -> ~FrameOrSeries
```

9.1 Date Ranges, Frequencies, and Shifting

```
index = pd.date_range('2012.04.01', '2012.06.01')
from pandas.tseries.offsets import Hour, Minute

fourhour = Hour(4)
pd.date_range('2000.1.1', '2000.1.3', freq=fourhour)
pd.date_range('2000.1.1', '2000.1.3', freq='4H')
rng = pd.date_range('2012.1.1', '2012.9.1', freq='WOM-3FRI')#
    每个月第三个星期五。
```

Type	Description
%Y	Four-digit year
%y	Two-digit year
%m	Two-digit month [01, 12]
%d	Two-digit day [01, 31]
%H	Hour (24-hour clock) [00, 23]
%I	Hour (12-hour clock) [01, 12]
%M	Two-digit minute [00, 59]
%S	Second [00, 61] (seconds 60, 61 account for leap seconds)
%w	Weekday as integer [0 (Sunday), 6]

图 33: fig-t1

Type	Description
%U	Week number of the year [00, 53]; Sunday is considered the first day of the week, and days before the first Sunday of the year are "week 0"
%W	Week number of the year [00, 53]; Monday is considered the first day of the week, and days before the first Monday of the year are "week 0"
%z	UTC time zone offset as +HHMM or -HHMM; empty if time zone naive
%F	Shortcut for %Y-%m-%d (e.g., 2012-4-18)
%D	Shortcut for %m/%d/%y (e.g., 04/18/12)

Table 11-3. Locale-specific date formatting

Type	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Full date and time (e.g., 'Tue 01 May 2012 04:20:57 PM')
%p	Locale equivalent of AM or PM
%x	Locale-appropriate formatted date (e.g., in the United States, May 1, 2012 yields '05/01/2012')
%X	Locale-appropriate time (e.g., '04:24:12 PM')

图 34: fig-plott2

Alias	Offset type	Description
D	Day	Calendar daily
B	BusinessDay	Business daily
H	Hour	Hourly
T or min	Minute	Minutely
S	Second	Secondly
L or ms	Milli	Millisecond (1/1,000 of 1 second)
U	Micro	Microsecond (1/1,000,000 of 1 second)
M	MonthEnd	Last calendar day of month
BM	BusinessMonthEnd	Last business day (weekday) of month
MS	MonthBegin	First calendar day of month
BMS	BusinessMonthBegin	First weekday of month
W-MON, W-TUE, ...	Week	Weekly on given day of week (MON, TUE, WED, THU, FRI, SAT, or SUN)
WOM-1MON, WOM-2MON, ...	WeekOfMonth	Generate weekly dates in the first, second, third, or fourth week of the month (e.g., WOM-3FRI for the third Friday of each month)
Q-JAN, Q-FEB, ...	QuarterEnd	Quarterly dates anchored on last calendar day of each month, for year ending in indicated month (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC)
BQ-JAN, BQ-FEB, ...	BusinessQuarterEnd	Quarterly dates anchored on last weekday day of each month, for year ending in indicated month
QS-JAN, QS-FEB, ...	QuarterBegin	Quarterly dates anchored on first calendar day of each month, for year ending in indicated month
BQS-JAN, BQS-FEB, ...	BusinessQuarterBegin	Quarterly dates anchored on first weekday day of each month, for year ending in indicated month
A-JAN, A-FEB, ...	YearEnd	Annual dates anchored on last calendar day of given month (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC)
BA-JAN, BA-FEB, ...	BusinessYearEnd	Annual dates anchored on last weekday of given month
AS-JAN, AS-FEB, ...	YearBegin	Annual dates anchored on first day of given month

图 35: 基础时间频率

9.2 shifting time

```
ts = pd.Series(np.random.randn(4), index=pd.date_range('1/1/2000',
    periods=4, freq='M'))
ts.shift(2)
(ts-ts.shift(1))/ts.shift(1) # 可以很方便的计算两天的增长百分比。

ts.shift(2,freq='M')
ts.shift(2, freq='D')
offset = MonthEnd()
offset.rollforward(NOW) #显式的前滚。

offset.rollback(NOW)

ts = pd.Series(np.random.randn(20),
index=pd.date_range('1/15/2000', periods=20, freq='4d'))

ts.groupby(offset.rollforward).mean()
```

9.3 时区设定

```
import pytz
pytz.common_timezones[-5:]
tz = pytz.timezone('America/New_York')



tz = pytz.timezone('America/New_York')
rng = pd.date_range('3/9/2012 9:30', periods=6, freq='D')
ts = pd.Series(np.random.randn(len(rng)), index=rng)
ts_utc = ts.tz_localize('UTC')

# 当然，也可以在生成的时候指定tz属性。
pd.date_range('3/9/2011 9:30', periods=10, freq='D', tz='UTC')

 将之前的是按转换为美国纽约时间。
x = ts_utc.tz_convert('America/New_York')
```

```
ts.index.tz_localize('Asia/Shanghai')

stamp = pd.Timestamp('2011-03-12 04:00')
stamp_utc = stamp.tz_localize('utc')
stamp_utc.tz_convert('America/New_York')
```

9.4 时间区间

```
rng = pd.period_range('2001-01-01', '2002-05-01', freq='M')
Signature: pd.period_range(start=None, end=None, periods=None,
    freq=None, name=None) -> pandas.core.indexes.period.PeriodIndex
Docstring:
Return a fixed frequency PeriodIndex.

pd.period_range(start='2017-01-01', end='2018-01-01', freq='M')
```

9.5 时间区间转换 p.asfreq

```
p.asfreq?
Docstring:
Convert Period to desired frequency, at the start or end of the
    interval.

p.asfreq('M', how='start')
p.asfreq('M', how='S') # s e 都是start和end的缩写。

rng = pd.period_range('2006', '2009', freq='A-DEC')
ts = pd.Series(np.random.randn(len(rng)), index=rng)

ts.asfreq('M', how='S')
```

9.6 季度周期

```
p = pd.Period('2012Q4', freq='Q-JAN')
```

```
p4pm = (p.asfreq('B', 'e') -1).asfreq('T','s') + 16*60 #
    表示分钟。s表示start
rng = pd.period_range('2011Q3','2012Q4', freq='Q-JAN')
PeriodIndex(['2011Q3', '2011Q4', '2012Q1', '2012Q2', '2012Q3',
    '2012Q4'], dtype='period[Q-JAN]', freq='Q-JAN')

ts = pd.Series(np.arange(len(rng)), index=rng)
new_rng = (rng.asfreq('B','e') -1).asfreq('T','s')+16*60
ts.index = new_rng.to_timestamp()


pts = ts.to_period()
ts2.to_period('M').to_timestamp(how='start') #可以转回去。
将不同列的时间信息组合起来:
index = pd.PeriodIndex(year=data.year, quarter=data.quarter,
    freq='Q-DEC')
```

9.7 时间采样，上下混合采样等

```
  rng = pd.date_range('2000-01-01', periods=100, freq='D')
  ts = pd.Series(np.random.randn(len(rng)), index=rng)
  ts.resample?
  Signature: ts.resample(rule, axis=0, closed: Union[str,
      NoneType]=None, label: Union[str, NoneType]=None,
  convention: str='start', kind: Union[str, NoneType]=None,
  loffset=None, base: int=0, on=None, level=None)

  一般采样后都进行统计 resample().sum()/fill()/apply()/asfreq()/
  # 计算一个bins内的first（open), last(close), maximum(high)

   minimal(low)四个数据。
  # 公式: ohlc

  ts.resample('M').ohlc()
  annual_frame.resample('Q-DEC').ffill()
  annual_frame.resample('Q-DEC', convention='end').ffill()
```

9.8 滑动窗口 rolling()

```
  close_px_all = pd.read_csv('stock_px_2.csv', parse_dates=True,
      index_col=0)
  close_px = close_px_all[['AAPL', 'MSFT', 'XOM']]
  close_px = close_px.resample('B').ffill()


  close_px.AAPL.plot()
  close_px.AAPL.rolling(250).mean().plot() #
      滑动窗口的平均。类似于groupby 和 resample ,调用后可以调用统计函数。

  # 一个问题，在开始阶段，我们数据可能少于window periods 怎么设置？？？
  # min_periods 这个参数可以

  appl_std250 = close_px.AAPL.rolling(250, min_periods=10).std()

  close_px.AAPL.rolling(window, min_periods=None, center=False,
      win_type=None, on=None, axis=0, closed=None)


  # expanding 从窗口从小到大，知道最后充满整个序列。

  expending_mean = appl_std250.expanding().mean()

close_px:
  AAPL MSFT XOM
  2003-01-02 7.40 21.11 29.22
  2003-01-03 7.45 21.14 29.24
  2003-01-06 7.45 21.52 29.96
  2003-01-07 7.43 21.93 28.95
  2003-01-08 7.28 21.31 28.83
  ...  ...  ...  ...
  2011-10-10 388.81 26.94 76.28
  2011-10-11 400.29 27.00 76.27
  2011-10-12 402.19 26.96 77.16
  2011-10-13 408.43 27.18 76.37
  2011-10-14 422.00 27.27 78.11
close_px.rolling(60).mean().plot(logy=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d3992ec18>
```

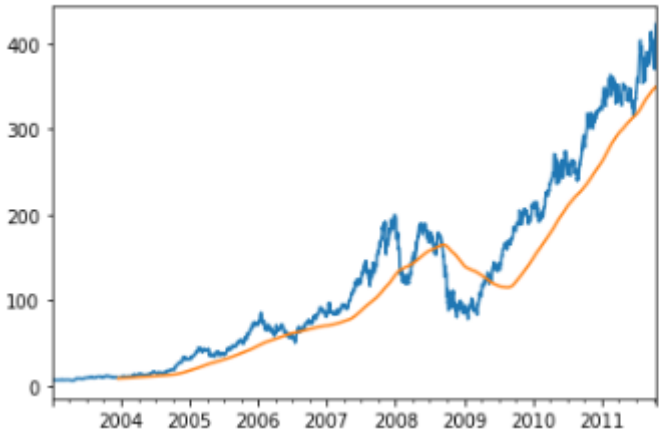
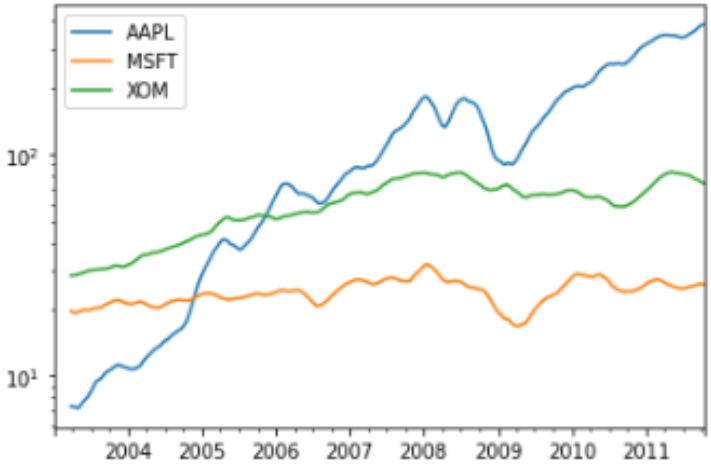


图 36: 时间滑动窗口

9.9 指数加权函数

```
appl_px = close_px.AAPL['2006':'2007']

ma60 = appl_px.rolling(30, min_periods=20).mean()

ma60x = appl_px.rolling(30, min_periods=20)

max60gaussian = appl_px.rolling(30, min_period=20,
    win_tpye='guassian').mean(std = 3)

ewma60 = appl_px.ewm(span=30).mean() #指数加权。



#
    我们之前都是用的mean,sum函数，其实也可以自定义函数，要注意的就是，函数是从一个list返回一个标量即可。

from scipy.stats import percentileofscore
score_at_2percent = lambda x: percentileofscore(x, 0.02)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d37f54550>
```




```
1 close_px.expanding().mean().plot(logy=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d37ca41d0>
```

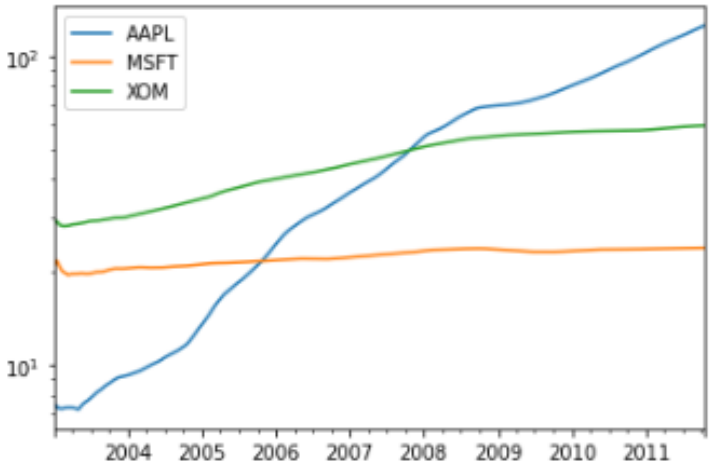


图 37: 画图对象是 DATAFRAME 对象的时候

```
reslult = returns.AAPL.rolling(250).apply(score_at_2percent)
reslult.plot()
```

10 pandas 高级部分

10.1 分类类类型 categorical

更好的性能和内存使用，在分类数据或者机器学习的应用中

```
# 关键词:
# Categorical
# astype('category') 将类型转换为pandas.Categorical 对象。
#
# 属性:  values.categories && values.codes
# # 直接建立Categorical 对象。 pandas.Categorical() 类似于
    dataframe,series.

# # 也可以指定 相应的代码。
# categories = [xxx]
# codes = [xx]
# my_cats_2 = pd.Categorical.from_codes(codes, categories, ordered =
    Ture or False)


fruit_cat = df.fruit.astype('category')

# frome codes 建立。
categories = ['foo', 'bar', 'baz']
codes = [0,1,2,0,0,1]

my_cat = pd.Categorical.from_codes(codes, categories, ordered=True)
my_cat
```

10.2 categorical 类型的计算效率

```
pd.qcut 产生的 categorical对象。
```

```
# 可以直接用 .codes && categories 属性访问。
# 可以设置label 来修改输出的结果。
# bins = pd.qcut(list, 4, label=['x1','x2','x3','x4'])
# 然后对bins 进行
    groupby，统计相应的值。因为groupby可以直接传入一个维度或者长度于对应列数相同的列表或者series.
# 性能提升： 当用字符串存储的时候，和用 categorical
    对象存储的时候，所占用的内存大小完全不一样。
# 当然，在这个基础上用groupby等运算的时候，所用的时间也不一样。


draws = np.random.randn(1000)
# 上边的1000个数，进行qcut，根据大小范围，四份。
bins = pd.qcut(draws, 4)

# 从结果看出，bins拥有categories的属性。
# 我们来用categories && codes来访问

bins.categories


用bins来对draws进行groupby并进行一些统计输出
bins = pd.Series(bins, name='quartile')

resluts = pd.Series(draws).groupby(bins).agg(['count', 'min', 'max',
    'mean']).reset_index()

# 然后我们可以查看，categorical 对象和 字符对象占用内存对比。

N = 10000000
draws = pd.Series(np.random.randn(N))
labels = pd.Series(['foo', 'bar', 'baz', 'qux'] * (N // 4))

# 将series 转换成categorical

categories = labels.astype('category')

labels.memory_usage()/categories.memory_usage()
7.999756807782151
%time _=labels.astype('category')
```

```
CPU times: user 377 ms, sys: 4.52 ms, total: 382 ms
Wall time: 383 ms


# cat 提供了对访问codes categories 的方法。

# cat.set_categories() 来扩展类别对象，虽然只是观察到了有限个。
# 比如没有观察到的类比依然会统计信息统计到，只是个数为0而已。
    value_counts...

# 大型数据集中，很多类别没有使用，想要去掉这些没有观察到的类别，使用：
    remove_unused_categories.


# 哑变量： 转换： pandas.get_dummies(series)
cat_s2.cat.remove_unused_categories()
cat_s2 = cat_s.cat.add_categories(['e'])
pd.get_dummies(cat_s)
```

11 groupby 更加高级的用法

```
df = pd.DataFrame({'key': ['a', 'b', 'c'] * 4,
'value': np.arange(12.)})

g = df.groupby('key').value

g.transform(lambda x: x.mean())
g.transform('mean')
g.apply(lambda x:x*2)====
g.transform(lambda x:x.rank(ascending=False))

normalized = (df['value'] - g.transform('mean'))/g.transform('std')

时间序列数据，resample类似于groupby,但是当dataframe的时候，需要使用TimeGrouper

# 想要对df2 的key和time都进行groupby操作，该如何进行？
df2.groupby('key').resample('5min', on='time').sum()
```

```
     value
key  time
a  2017-05-20 00:00:00 30.0
2017-05-20 00:05:00 105.0
2017-05-20 00:10:00 180.0
b  2017-05-20 00:00:00 35.0
2017-05-20 00:05:00 110.0
2017-05-20 00:10:00 185.0
c  2017-05-20 00:00:00 40.0
2017-05-20 00:05:00 115.0
2017-05-20 00:10:00 190.0
```

```
df2.groupby('key').resample('5min', on='time').sum().reset_index()
```