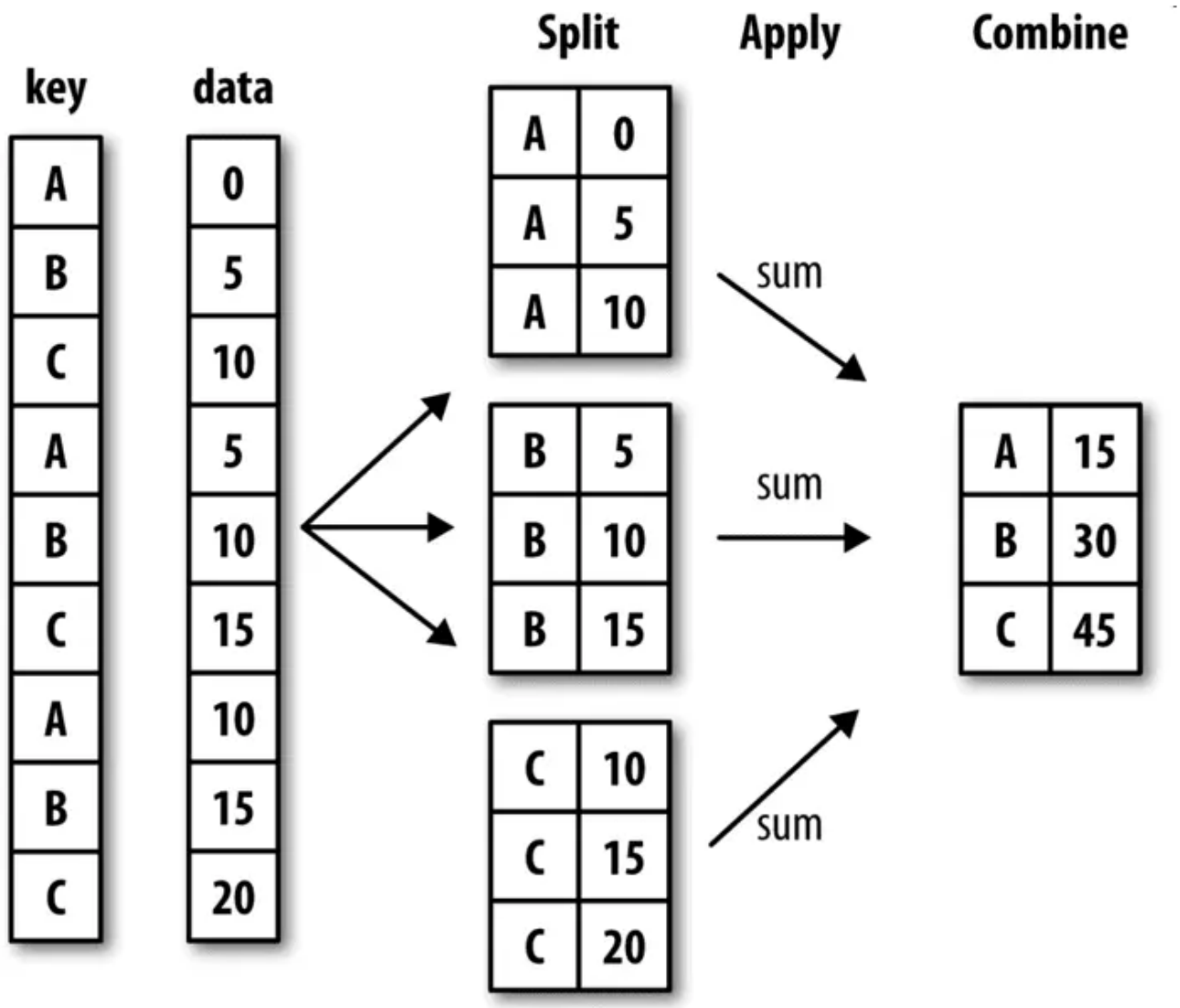
数据聚合和分组操作

Groupby机制




```
df
Out[1403]:
  key1 key2  data1  data2
0    a  one      0      5
1    a  two      1      6
2    b  one      2      7
3    b  two      3      8
4    a  one      4      9

df.groupby('key1').mean()
Out[1408]:
         data1     data2
```

```
key1
a     1.666667  6.666667
b     2.500000  7.500000

df.groupby(['key1','key2']).mean()
Out[1409]:
           data1  data2
key1 key2
a    one       2      7
     two       1      6
b    one       2      7
     two       3      8

#遍历各分组
for name,group in df.groupby('key1'):
    print(name)
    print(group)
a
  key1 key2  data1  data2
0    a  one      0      5
1    a  two      1      6
4    a  one      4      9
b
  key1 key2  data1  data2
2    b  one      2      7
3    b  two      3      8

for name,group in df.groupby(['key1','key2']):
    print(name)
    print(group)
('a', 'one')
  key1 key2  data1  data2
0    a  one      0      5
4    a  one      4      9
('a', 'two')
  key1 key2  data1  data2
1    a  two      1      6
('b', 'one')
  key1 key2  data1  data2
2    b  one      2      7
('b', 'two')
  key1 key2  data1  data2
3    b  two      3      8

#使用字典分组
mapping
Out[1422]: {'a': 'red', 'b': 'blue', 'c': 'red', 'd': 'red', 'e': 'blue'}
people
Out[1423]:
         a     b     c   d   e
Joe      0   1.0   2.0   3   4
Steve    5   6.0   7.0   8   9
Wes     10  11.0  12.0  13  14
```

```
Jim     15   NaN   NaN  18  19
Travis  20  21.0  22.0  23  24

people.groupby(mapping,axis = 'columns').count()
Out[1426]:
        blue  red
Joe        2    3
Steve      2    3
Wes        2    3
Jim        1    2
Travis     2    3

people.groupby(mapping,axis = 'columns').sum()
Out[1427]:
        blue   red
Joe      5.0   5.0
Steve   15.0  20.0
Wes     25.0  35.0
Jim     19.0  33.0
Travis  45.0  65.0

#使用函数分组
key_list
Out[1429]: ['one', 'one', 'one', 'two', 'two']
people.groupby([len,key_list]).min()
Out[1430]:
        a     b     c   d   e
3 one   0   1.0   2.0   3   4
  two  15   NaN   NaN  18  19
5 one   5   6.0   7.0   8   9
6 two  20  21.0  22.0  23  24

#根据索引层级分组
df
Out[1432]:
city   US          JP
tenor   1   3   5   1   3
0       0   1   2   3   4
1       5   6   7   8   9
2      10  11  12  13  14
3      15  16  17  18  19
df.groupby(level = 'city',axis = 1).count()
Out[1433]:
city  JP  US
0      2   3
1      2   3
2      2   3
3      2   3
```

数据聚合

```
df
Out[1466]:
  key1 key2  data1  data2
0    a  one      0      5
1    a  two      1      6
2    b  one      2      7
3    b  two      3      8
4    a  one      4      9

df.groupby('key1').agg('mean')
Out[1467]:
         data1     data2
key1
a     1.666667  6.666667
b     2.500000  7.500000

df.groupby('key1').agg(['mean','max','min','count'])
Out[1468]:
         data1                   data2
          mean max min count      mean max min count
key1
a     1.666667   4   0     3  6.666667   9   5     3
b     2.500000   3   2     2  7.500000   8   7     2

#传入元组自定义列名
df.groupby(['key1','key2']).agg([('max_value','max'),('cnt','count')])
Out[1471]:
              data1         data2
          max_value cnt max_value cnt
key1 key2
a    one          4   2         9   2
     two          1   1         6   1
b    one          2   1         7   1
     two          3   1         8   1

#使用有列名与函数对应的字典将不同的函数应用到一个或多个列上
df.groupby('key1').agg({'data1':'mean','data2':'sum'})
Out[1475]:
         data1  data2
key1
a     1.666667     20
b     2.500000     15

df.groupby('key1').agg({'data1':['mean','max','min'],'data2':['sum','std','var']})
Out[1477]:
         data1         data2
          mean max min   sum       std       var
key1
a     1.666667   4   0    20  2.081666  4.333333
b     2.500000   3   2    15  0.707107  0.500000

#返回不带行索引的聚合函数 as_index = False
df.groupby(['key1','key2'],as_index = False).sum()
```

```
Out[1478]:
  key1 key2  data1  data2
0    a  one      4     14
1    a  two      1      6
2    b  one      2      7
3    b  two      3      8
```

应用：通用拆分-应用-联合

```
tips = sns.load_datasets('tips')
def top3(df,n = 3,column = 'tip_pct'):
    return df.sort_values(by = column)[-n:]
top3(tips)
Out[1486]:
     total_bill   tip     sex smoker  day    time  size   tip_pct
67         3.07  1.00  Female    Yes  Sat  Dinner     1  0.325733
178        9.60  4.00  Female    Yes  Sun  Dinner     2  0.416667
172        7.25  5.15    Male    Yes  Sun  Dinner     2  0.710345

#先按照smoker进行分组，再调用apply
tips.groupby('smoker').apply(top3)
Out[1489]:
            total_bill   tip     sex smoker   day    time  size   tip_pct
smoker
Yes    67         3.07  1.00  Female    Yes   Sat  Dinner     1  0.325733
       178        9.60  4.00  Female    Yes   Sun  Dinner     2  0.416667
       172        7.25  5.15    Male    Yes   Sun  Dinner     2  0.710345
No     51        10.29  2.60  Female     No   Sun  Dinner     2  0.252672
       149        7.51  2.00    Male     No  Thur   Lunch     2  0.266312
       232       11.61  3.39    Male     No   Sat  Dinner     2  0.291990

#设定关键字
tips.groupby(['smoker','day']).apply(top3,n=1,column = 'total_bill')
Out[1496]:
                 total_bill    tip     sex smoker   day    time  size   tip_pct
smoker day
Yes    Thur 197       43.11   5.00  Female    Yes  Thur   Lunch     4  0.115982
       Fri  95        40.17   4.73    Male    Yes   Fri  Dinner     4  0.117750
       Sat  170       50.81  10.00    Male    Yes   Sat  Dinner     3  0.196812
       Sun  182       45.35   3.50    Male    Yes   Sun  Dinner     3  0.077178
No     Thur 142       41.19   5.00    Male     No  Thur   Lunch     5  0.121389
       Fri  94        22.75   3.25  Female     No   Fri  Dinner     2  0.142857
       Sat  212       48.33   9.00    Male     No   Sat  Dinner     4  0.186220
       Sun  156       48.17   5.00    Male     No   Sun  Dinner     6  0.103799

#压缩分组键
tips.groupby(['smoker','day'],group_keys = False).apply(top3,n=1,column = 'total_bill')
Out[1497]:
     total_bill    tip     sex smoker   day    time  size   tip_pct
197       43.11   5.00  Female    Yes  Thur   Lunch     4  0.115982
95        40.17   4.73    Male    Yes   Fri  Dinner     4  0.117750
```

```
170       50.81  10.00    Male    Yes   Sat  Dinner     3  0.196812
182       45.35   3.50    Male    Yes   Sun  Dinner     3  0.077178
142       41.19   5.00    Male     No  Thur   Lunch     5  0.121389
94        22.75   3.25  Female     No   Fri  Dinner     2  0.142857
212       48.33   9.00    Male     No   Sat  Dinner     4  0.186220
156       48.17   5.00    Male     No   Sun  Dinner     6  0.103799


#桶分析与分位数
frame = pd.DataFrame({'data1':np.random.randn(1000),'data2':np.random.randn(1000)})
quartiles = pd.cut(frame.data1,4)
quartiles[:10]
Out[1507]:
0    (-1.479, 0.14]
1    (-1.479, 0.14]
2    (1.759, 3.378]
3     (0.14, 1.759]
4    (-1.479, 0.14]
5     (0.14, 1.759]
6    (-1.479, 0.14]
7     (0.14, 1.759]
8     (0.14, 1.759]
9    (-1.479, 0.14]
Name: data1, dtype: category
Categories (4, interval[float64]): [(-3.105, -1.479] < (-1.479, 0.14] < (0.14, 1.759] <
(1.759, 3.378]]

def get_stats(group):
    return {'count':group.count(),'max':group.max(),'min':group.min(),'mean':group.mean()}
                                                                                    #
等长桶
frame.data2.groupby(quartiles).apply(get_stats).unstack()
Out[1509]:
                 count       max      mean       min
data1
(-3.105, -1.479]   79.0  1.928149 -0.137689 -2.118819
(-1.479, 0.14]    460.0  3.221790 -0.034230 -3.040561
(0.14, 1.759]     414.0  3.474453  0.033618 -2.756049
(1.759, 3.378]     47.0  2.098449 -0.087062 -1.615950

 #等深桶
quartiles = pd.qcut(frame.data1,10,labels = False)
frame.data2.groupby(quartiles).apply(get_stats).unstack()
Out[1511]:
       count       max      mean       min
data1
0      100.0  1.928149 -0.080553 -2.118819
1      100.0  2.424224 -0.017541 -3.040561
2      100.0  2.045359 -0.152340 -2.645968
3      100.0  3.221790  0.184596 -2.293599
4      100.0  2.197459 -0.131215 -2.614145
5      100.0  2.050473 -0.149522 -2.453174
6      100.0  2.342420  0.075888 -2.315995
```

```
7      100.0  2.903131 -0.007657 -1.806430
8      100.0  3.474453  0.054125 -2.756049
9      100.0  2.291237  0.056243 -2.249208


 #使用指定分组值填充缺失值
 data
Out[1525]:
Ohio          1.0
New York      2.0
Vermont       NaN
Florida       4.0
Oregon        NaN
Nevada        6.0
California    7.0
Idaho         NaN
dtype: float64
group_key
Out[1526]: ['West', 'West', 'West', 'West', 'East', 'East', 'East', 'East']

data.groupby(group_key).mean()
Out[1524]:
East    6.500000
West    2.333333
dtype: float64

fill_mean = lambda g:g.fillna(g.mean())
data.groupby(group_key).apply(fill_mean)
Out[1523]:
Ohio          1.000000
New York      2.000000
Vermont       2.333333
Florida       4.000000
Oregon        6.500000
Nevada        6.000000
California    7.000000
Idaho         6.500000
dtype: float64

#分组加权平均
df
Out[1530]:
  category  data    weight
0        a     0  0.518266
1        a     1  0.044373
2        a     2  0.322689
3        a     3  0.573515
4        b     4  0.907809
5        b     5  0.450028
6        b     6  0.967870
7        b     7  0.505708

get_wavg = lambda g:np.average(g['data'],weights = g['weight'])
```

```
df.groupby('category').apply(get_wavg)
Out[1532]:
category
a    1.652197
b    5.378424
dtype: float64
```

数据透视表

```
tips.head()
Out[1543]:
   total_bill   tip     sex smoker  day    time  size   tip_pct
0       16.99  1.01  Female     No  Sun  Dinner     2  0.059447
1       10.34  1.66    Male     No  Sun  Dinner     3  0.160542
2       21.01  3.50    Male     No  Sun  Dinner     3  0.166587
3       23.68  3.31    Male     No  Sun  Dinner     2  0.139780
4       24.59  3.61  Female     No  Sun  Dinner     4  0.146808

tips.pivot_table(index = ['day','smoker'])
Out[1539]:
                size       tip   tip_pct  total_bill
day  smoker
Thur Yes     2.352941  3.030000  0.163863   19.190588
     No      2.488889  2.673778  0.160298   17.113111
Fri  Yes     2.066667  2.714000  0.174783   16.813333
     No      2.250000  2.812500  0.151650   18.420000
Sat  Yes     2.476190  2.875476  0.147906   21.276667
     No      2.555556  3.102889  0.158048   19.661778
Sun  Yes     2.578947  3.516842  0.187250   24.120000
     No      2.929825  3.167895  0.160113   20.506667

tips.pivot_table(['tip_pct','size'],index = ['time','day'],columns = 'smoker')
Out[1541]:
                size             tip_pct
smoker           Yes        No       Yes        No
time   day
Lunch  Thur  2.352941  2.500000  0.163863  0.160311
       Fri   1.833333  3.000000  0.188937  0.187735
Dinner Thur       NaN  2.000000       NaN  0.159744
       Fri   2.222222  2.000000  0.165347  0.139622
       Sat   2.476190  2.555556  0.147906  0.158048
       Sun   2.578947  2.929825  0.187250  0.160113

#传入margins = True 扩充表来包含部分总计
tips.pivot_table(['tip_pct','size'],index = ['time','day'],columns = 'smoker',margins =
True)
Out[1542]:
                size                     tip_pct
smoker           Yes        No       All       Yes        No       All
time   day
Lunch  Thur  2.352941  2.500000  2.459016  0.163863  0.160311  0.161301
```

```
       Fri   1.833333  3.000000  2.000000  0.188937  0.187735  0.188765
Dinner Thur       NaN  2.000000  2.000000       NaN  0.159744  0.159744
       Fri   2.222222  2.000000  2.166667  0.165347  0.139622  0.158916
       Sat   2.476190  2.555556  2.517241  0.147906  0.158048  0.153152
       Sun   2.578947  2.929825  2.842105  0.187250  0.160113  0.166897
All          2.408602  2.668874  2.569672  0.163196  0.159328  0.160803

#交叉表

pd.crosstab([tips.time,tips.day],tips.smoker,margins = True)
Out[1544]:
smoker       Yes   No  All
time   day
Lunch  Thur   17   44   61
       Fri     6    1    7
Dinner Thur    0    1    1
       Fri     9    3   12
       Sat    42   45   87
       Sun    19   57   76
All           93  151  244
```