小象学院
ChinaHadoop.cn

第十课（第28-30课时）

豆瓣书评数据案例

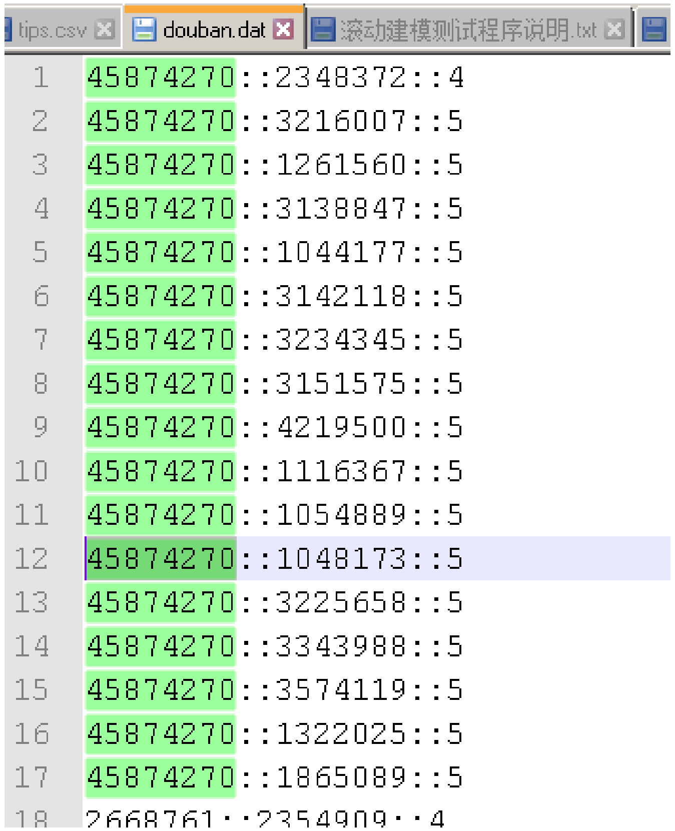
前情回顾

- **分析单个变量：各种方法**
- **分析多个变量：各种方法**
- **回归分析和广义线性模型：确认变量之间的关系---解释和预测**
- **分类分析：预测类别型因变量，有监督学习**
- **聚类分析：无监督学习，发现数据点之间的关系**
- **关联分析：无监督学习，频繁项集和关联规则**
- **基于重抽样：**
  - 统计量的显著性检验和区间估计
  - 增强训练效果和评价的稳定性
- **模型选择：**
  - 拟合度，查准率，查全率，ROC

大作业1：豆瓣书评数据（7月16日讲解）

小象学院 ChinaHadoop.cn

- **数据：**
  - 链接: http://pan.baidu.com/s/1o8fDDZO 密码: uzki
- **书的信息样例：https://book.douban.com/subject/1048173/**
- **目标：**
- **1.找出跟某一用户相似的若干读者**
- **2.找出跟某一本书有类似读者群的书**
- **3. 为读者划分类型**
- **4. 为书划分类型**
- **5. 为某一读者推荐他没有读过的书**
- **6. 使用关联分析进行推荐并且比较**



```
45874270::2348372::4
45874270::3216007::5
45874270::1261560::5
45874270::3138847::5
45874270::1044177::5
45874270::3142118::5
45874270::3234345::5
45874270::3151575::5
45874270::4219500::5
45874270::1116367::5
45874270::1054889::5
45874270::1048173::5
45874270::3225658::5
45874270::3343988::5
45874270::3574119::5
45874270::1322025::5
45874270::1865089::5
2668761::2354909::4
```

初步思路：

- **1. 提取特征：**
  - 读者：按照对书的评价（5分制，如无评价，对该书的评价为0）
  - 书籍：类似
- **2. 建立邻近性矩阵**
  - 参考文档向量，可以使用Jaccard或者余弦相似性（读者间，书籍间）
- **3. 使用K近邻算法找到与自己相似的若干对象**
- **4. 使用K-means聚类：读者、书**
- **5. 可使用协同过滤的方法（KNN）**

知识点复习：数据的邻近性

小象学院 ChinaHadoop.cn

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal	$d = \frac{\lvert p-q \rvert}{n-1}$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$
Interval or Ratio	$d = \lvert p - q \rvert$	$s = -d$, $s = \frac{1}{1+d}$ or $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$

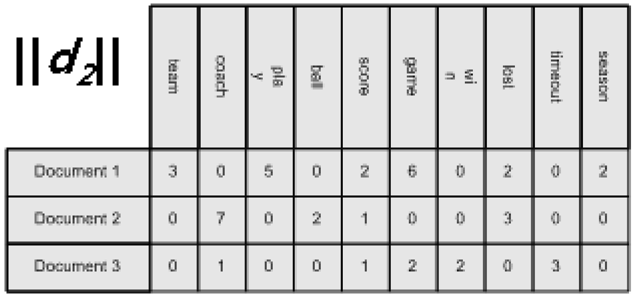
➢ **余弦相似性**

–如果d_1，d_2为两个文档向量

$$\cos(d_1, d_2) = (d_1 \bullet d_2) / \lVert d_1 \rVert \, \lVert d_2 \rVert$$

d_1 = 3 2 0 5 0 0 0 2 0 0

d_2 = 1 0 0 0 0 0 0 1 0 2



	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

数据分析

象）

知识点复习

➢ 聚类分析

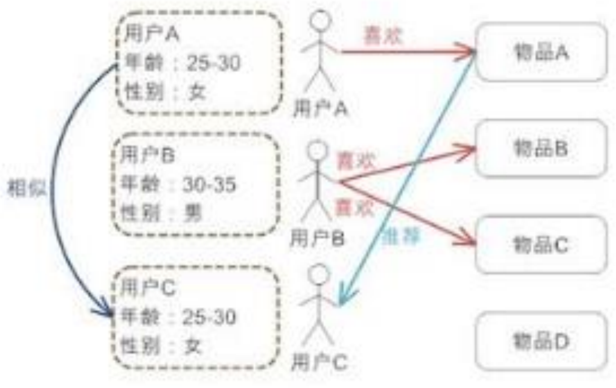
➢ K均值

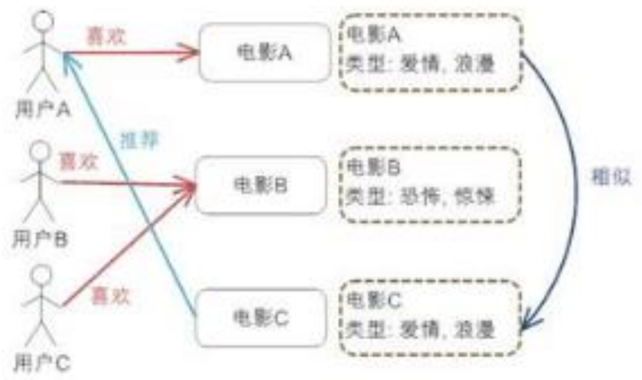
- 基于划分
- 每个簇有一个质心 centroid (center point)
- 每个点被分配给最近的质心
- 需指定簇的数量K
- 另：K中心点聚类（中心点必须是一个实际数据点）

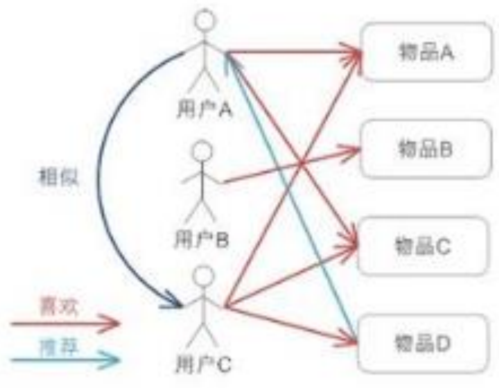
```
1: Select $K$ points as the initial centroids.
2: repeat
3:     Form $K$ clusters by assigning all points to the closest centroid.
4:     Recompute the centroid of each cluster.
5: until The centroids don't change
```

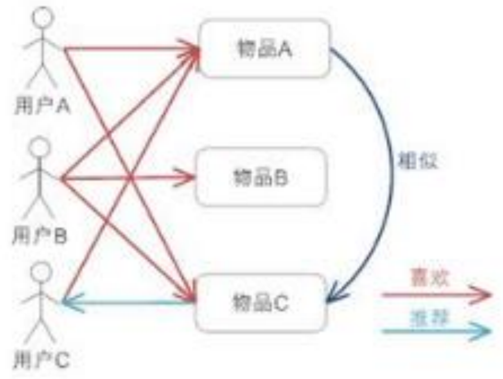
知识点复习

➢ **协同过滤**


用户A
年龄：25-30
性别：女
用户B
年龄：30-35
性别：男
用户C
年龄：25-30
性别：女
相似
用户A
用户B
用户C
喜欢
喜欢
喜欢
推荐
物品A
物品B
物品C
物品D


用户A
用户B
用户C
喜欢
喜欢
喜欢
推荐
电影A
电影B
电影C
电影A
类型：爱情，浪漫
电影B
类型：恐怖，惊悚
电影C
类型：爱情，浪漫
相似


用户A
用户B
用户C
相似
物品A
物品B
物品C
物品D
喜欢
推荐


用户A
用户B
用户C
物品A
物品B
物品C
相似
喜欢
推荐

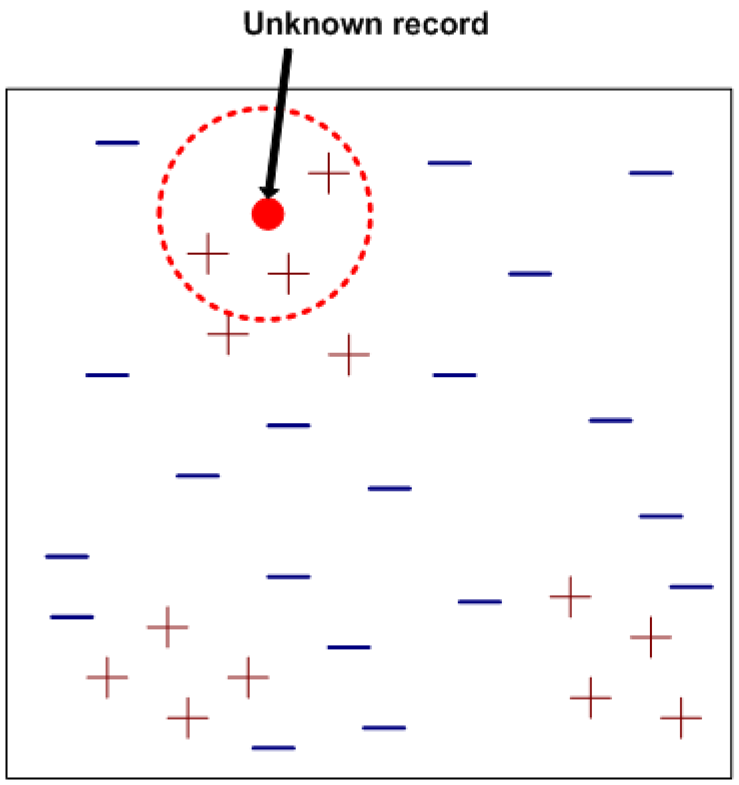
知识点复习

➢ 最邻近分类-KNN

➢ KNN-最邻近分类

- 定义数据集中记录之间的距离
- 定义参数k：临近数

- 分类时识别最近的k个紧邻
- 由k个近邻的类别对未知记录进行投票



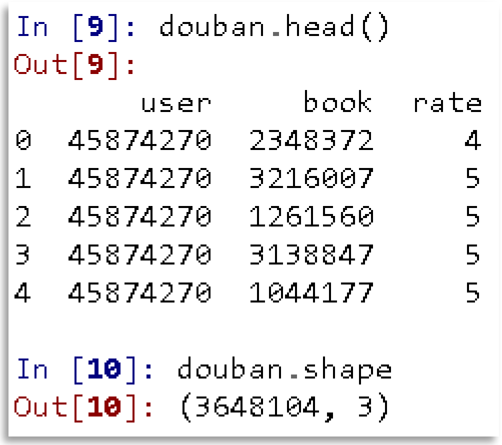
读入数据

➢ **确定路径**

- import os
- os.chdir(".. \\data ")
- !dir

➢ **读入数据**

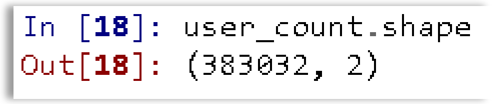
- import pandas as pd
- douban=pd.read_table("douban.dat",sep="::",names=["user","book","rate"])
- douban.head()
- douban.shape



```
In [9]: douban.head()
Out[9]:
       user     book  rate
0  45874270  2348372     4
1  45874270  3216007     5
2  45874270  1261560     5
3  45874270  3138847     5
4  45874270  1044177     5

In [10]: douban.shape
Out[10]: (3648104, 3)
```

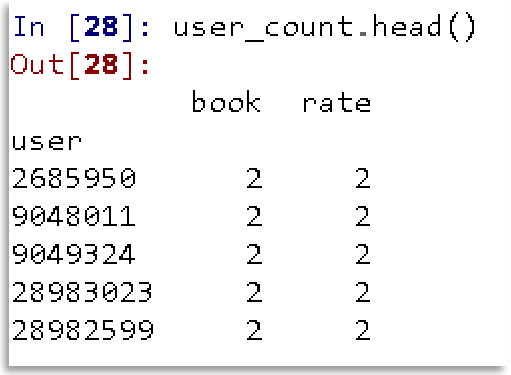
观察数据：读者

- **对读者的行为进行统计**
  - 总共多少个读者？
  - 每个读者评价了多少本书？
    - user_count=douban.groupby('user').count()
    - user_count.shape

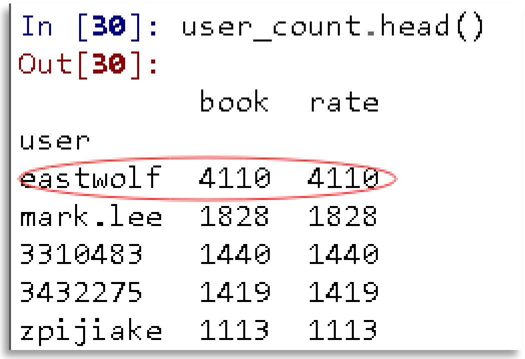


```
In [18]: user_count.shape
Out[18]: (383032, 2)
```

- user_count.head()
- user_count=user_count.sort('book',ascending=False)



```
In [28]: user_count.head()
Out[28]:
          book  rate
user
2685950      2     2
9048011      2     2
9049324      2     2
28983023     2     2
28982599     2     2
```

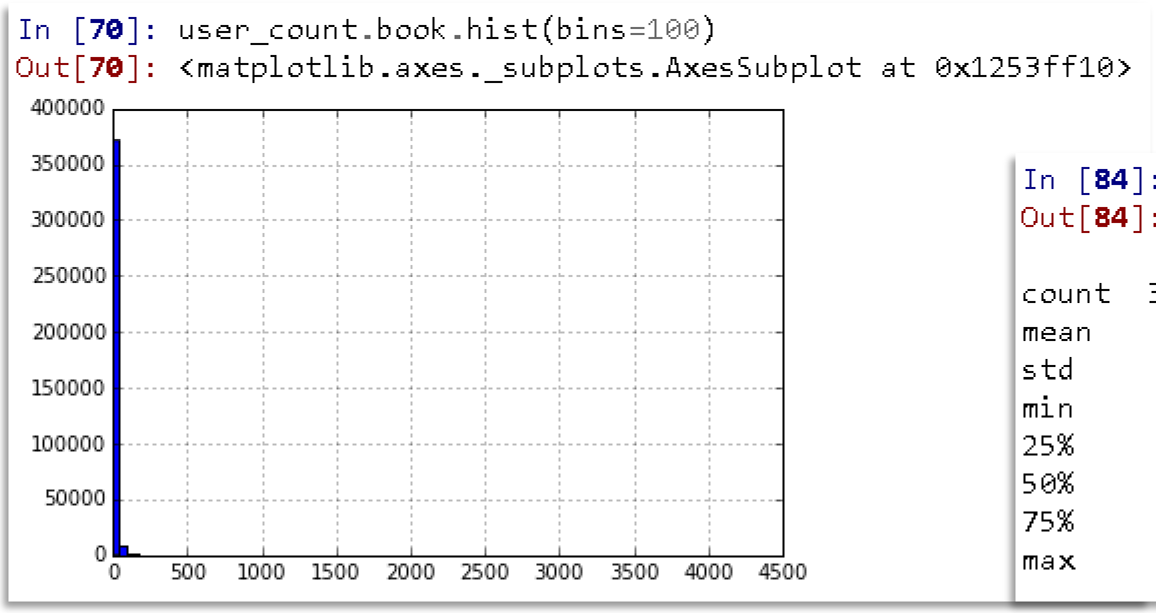


```
In [30]: user_count.head()
Out[30]:
          book  rate
user
eastwolf  4110  4110
mark.lee  1828  1828
3310483   1440  1440
3432275   1419  1419
zpijiake  1113  1113
```

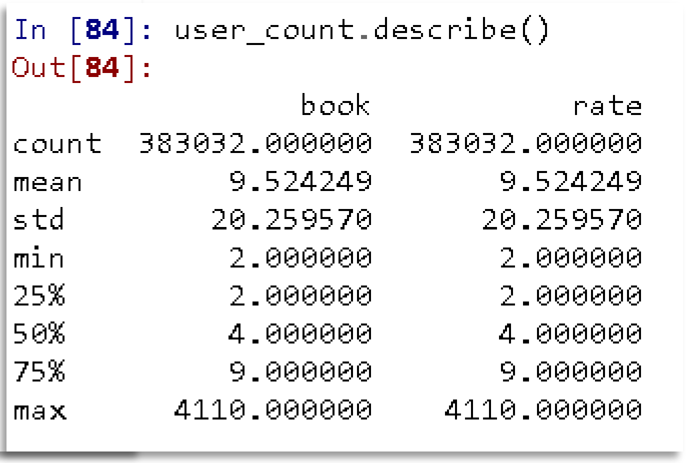
观察数据：读者

➢ **对读者评价书的数量进行统计**

- user_count.book.hist(bins=100)
- user_count.describe()



```
In [70]: user_count.book.hist(bins=100)
Out[70]: <matplotlib.axes._subplots.AxesSubplot at 0x1253ff10>
```

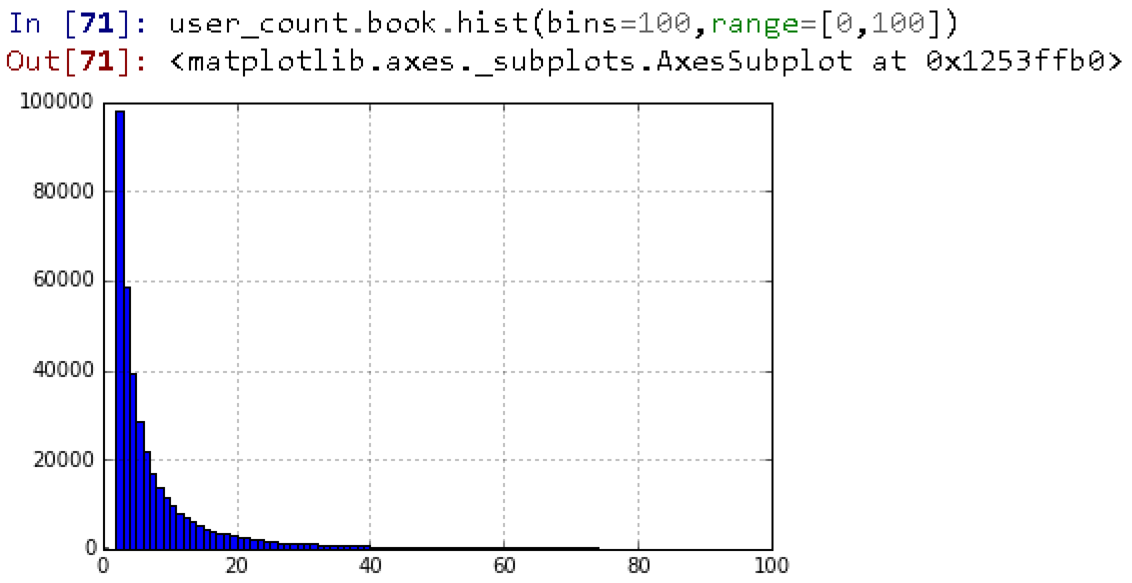


```
In [84]: user_count.describe()
Out[84]:
                book           rate
count  383032.000000  383032.000000
mean        9.524249       9.524249
std        20.259570      20.259570
min         2.000000       2.000000
25%         2.000000       2.000000
50%         4.000000       4.000000
75%         9.000000       9.000000
max      4110.000000    4110.000000
```

观察数据：读者

➢ **对读者评价书的数量进行统计**

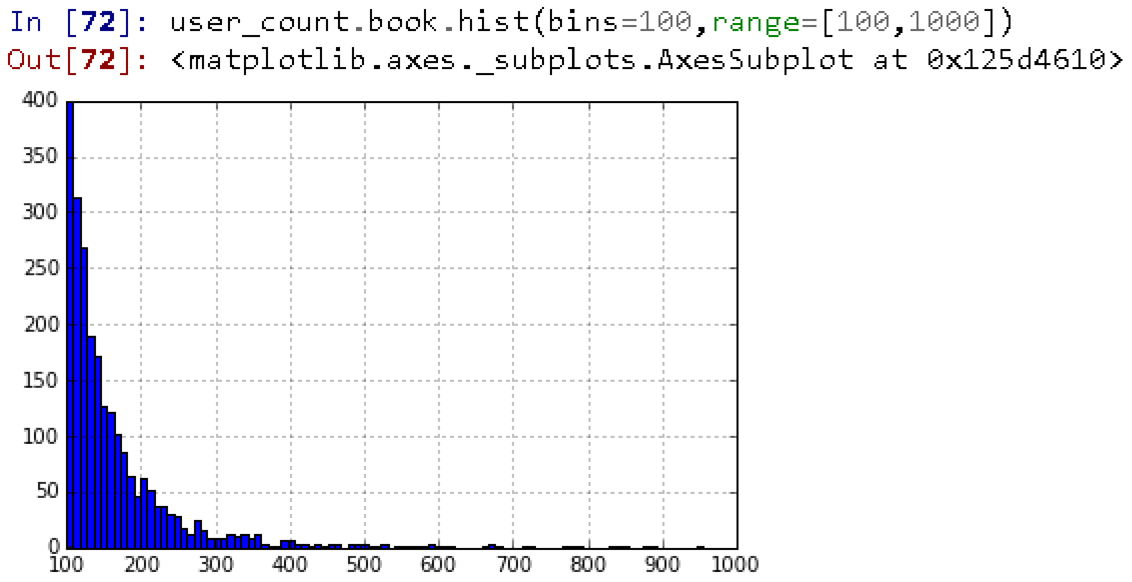
- user_count.book.hist(bins=100,range=[0,100])



```
In [71]: user_count.book.hist(bins=100,range=[0,100])
Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0x1253ffb0>
```

观察数据：读者

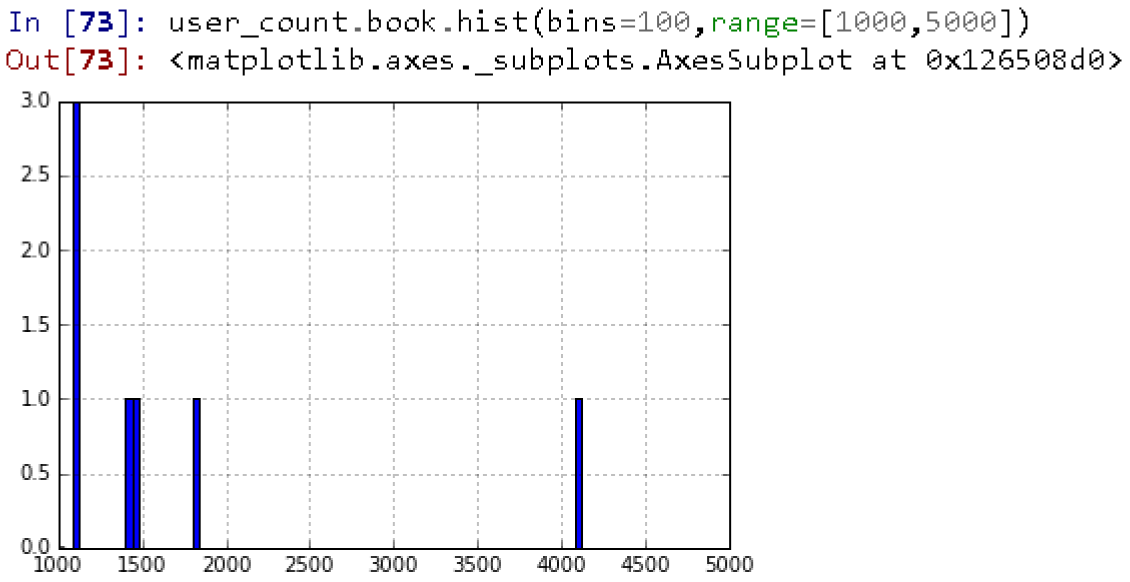
- **对读者评价书的数量进行统计**
  - user_count.book.hist(bins=100,range=[100,1000])



```
In [72]: user_count.book.hist(bins=100,range=[100,1000])
Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x125d4610>
```

观察数据：读者

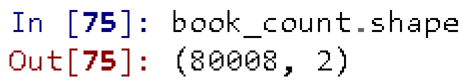
- 对读者评价书的数量进行统计
  - user_count.book.hist(bins=100,range=[1000,5000])



```
In [73]: user_count.book.hist(bins=100,range=[1000,5000])
Out[73]: <matplotlib.axes._subplots.AxesSubplot at 0x126508d0>
```

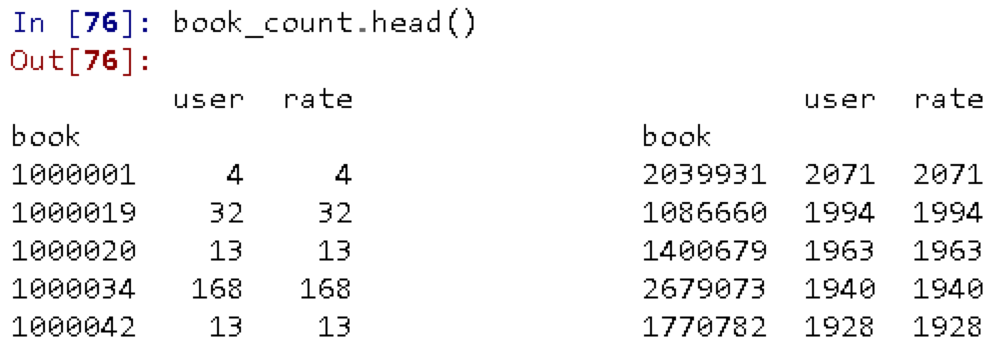
观察数据：书

- **对书的“行为“进行统计**
  - 总共多少本书？
  - 每本书被多少读者评价？
    - book_count=douban.groupby('book').count()
    - book_count.shape



```
In [75]: book_count.shape
Out[75]: (80008, 2)
```

- book_count.head()
- book_count=book_count.sort('user',ascending=False)

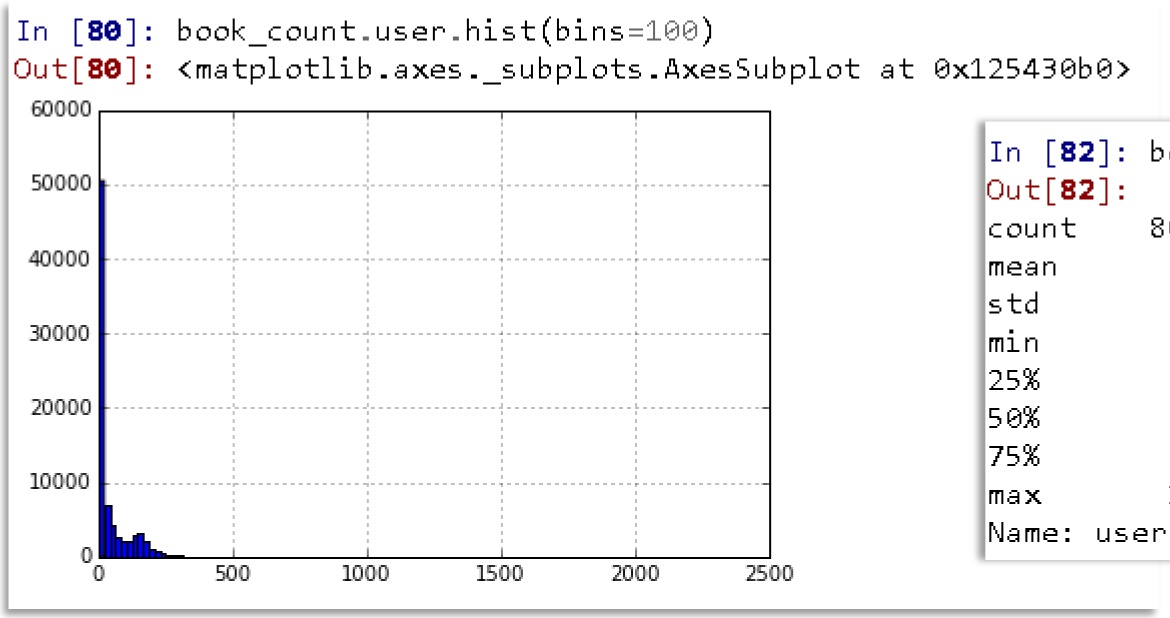


```
In [76]: book_count.head()
Out[76]:
         user  rate                     user  rate
book                           book
1000001     4     4            2039931  2071  2071
1000019    32    32            1086660  1994  1994
1000020    13    13            1400679  1963  1963
1000034   168   168            2679073  1940  1940
1000042    13    13            1770782  1928  1928
```

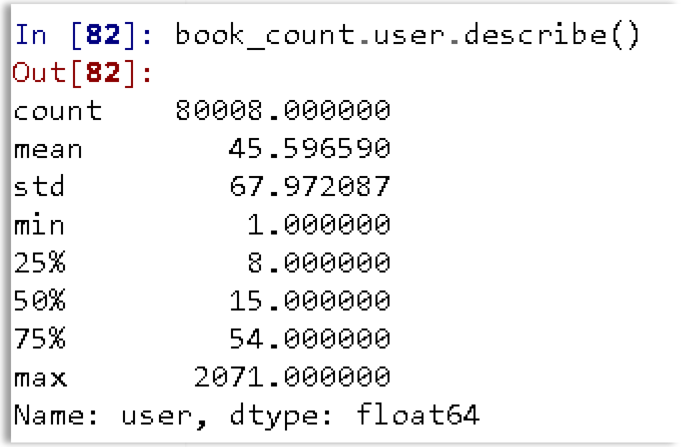
观察数据：书

➢ **对书的被评价数量进行统计**

- book_count.user.hist(bins=100)
- book_count.describe()



```
In [80]: book_count.user.hist(bins=100)
Out[80]: <matplotlib.axes._subplots.AxesSubplot at 0x125430b0>
```

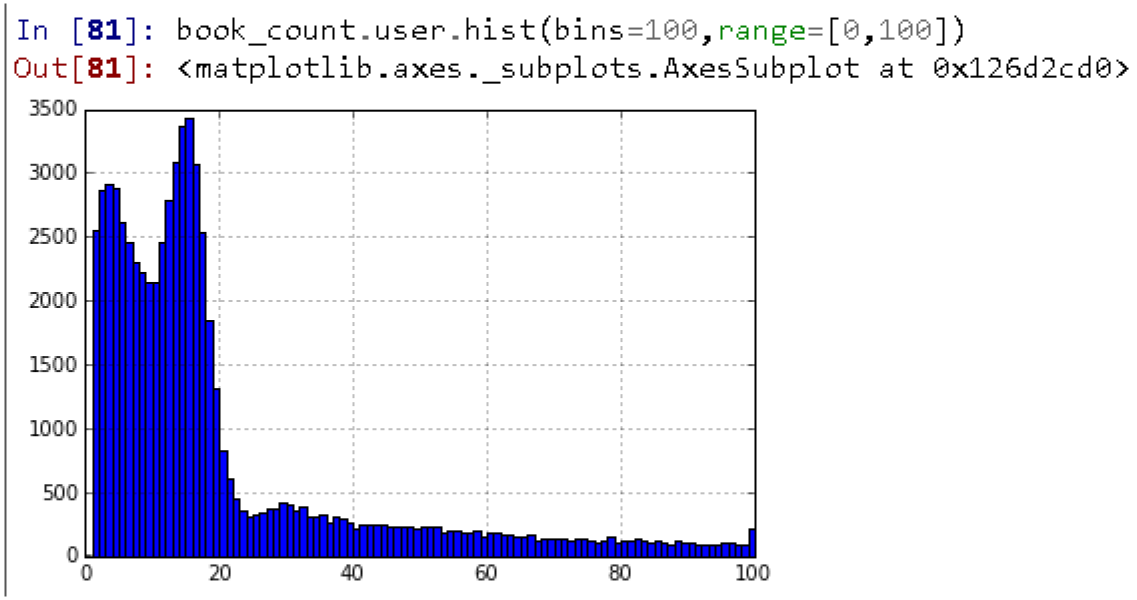


```
In [82]: book_count.user.describe()
Out[82]:
count    80008.000000
mean        45.596590
std         67.972087
min          1.000000
25%          8.000000
50%         15.000000
75%         54.000000
max       2071.000000
Name: user, dtype: float64
```

观察数据：书

➢ **对书的被评价数量进行统计**

- book_count.user.hist(bins=100,range=[0,100])



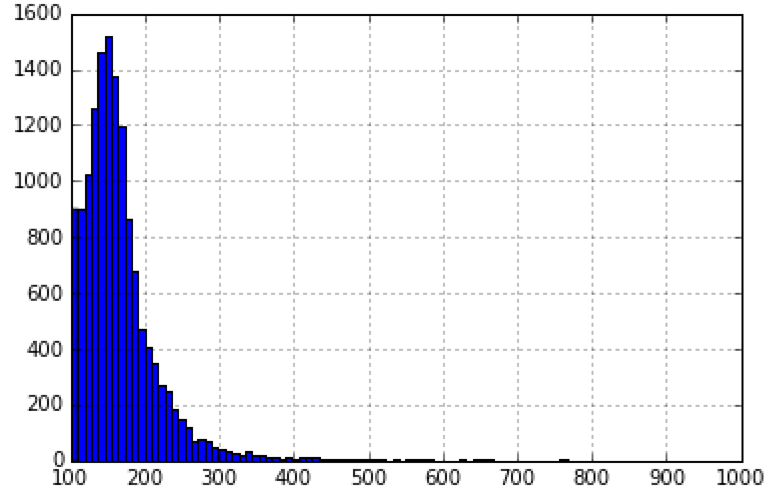
```
In [81]: book_count.user.hist(bins=100,range=[0,100])
Out[81]: <matplotlib.axes._subplots.AxesSubplot at 0x126d2cd0>
```

观察数据：书

➢ **对书的被评价数量进行统计**

- book_count.user.hist(bins=100,range=[100,1000])

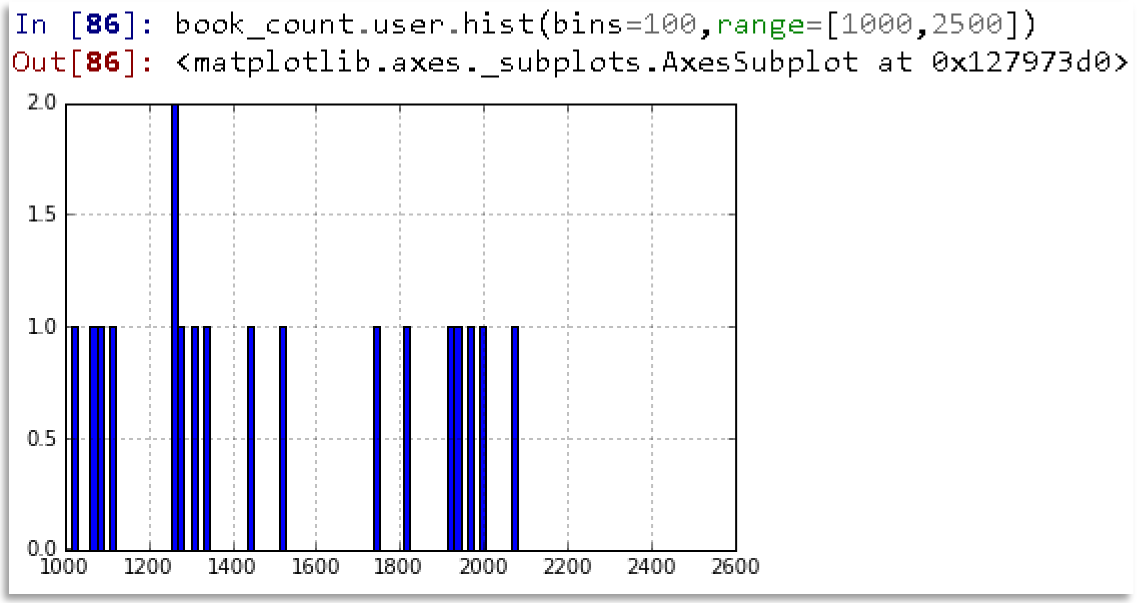
```
In [85]: book_count.user.hist(bins=100,range=[100,1000])
Out[85]: <matplotlib.axes._subplots.AxesSubplot at 0x127309b0>
```



观察数据：书

➢ **对书的被评价数量进行统计**

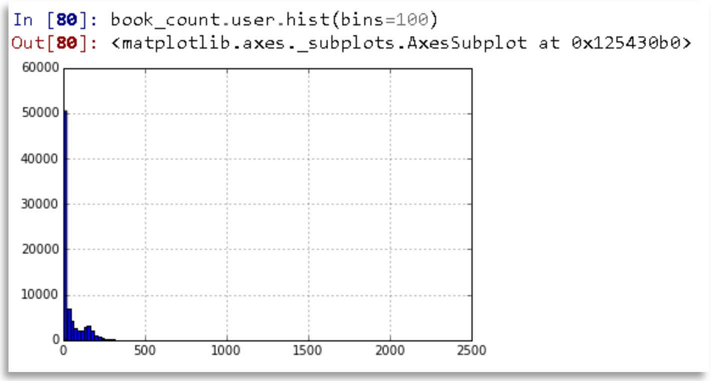
- book_count.user.hist(bins=100,range=[1000,2500])



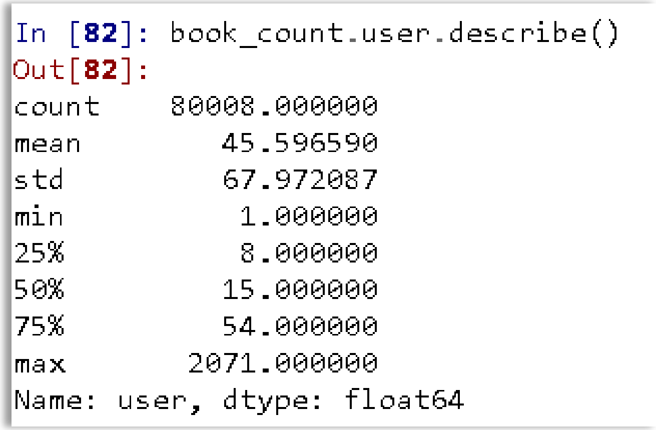
```
In [86]: book_count.user.hist(bins=100,range=[1000,2500])
Out[86]: <matplotlib.axes._subplots.AxesSubplot at 0x127973d0>
```

观察数据：书和读者的单维度分析

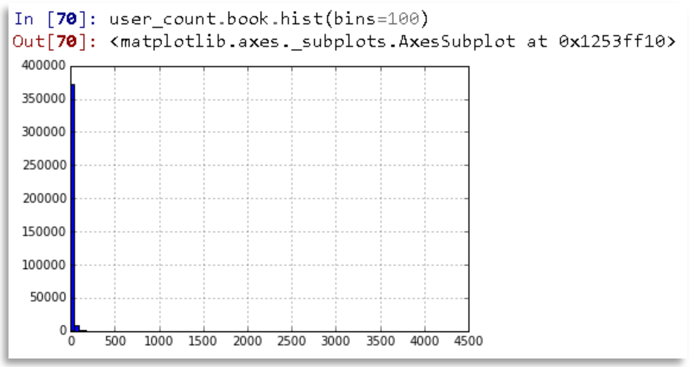
- 大部分读者（75%）评价的书少于10本（9本）
- 大部分书（75%）被评价的读着书少于55人（54人）
- 估计为帕累托分布：长尾效应



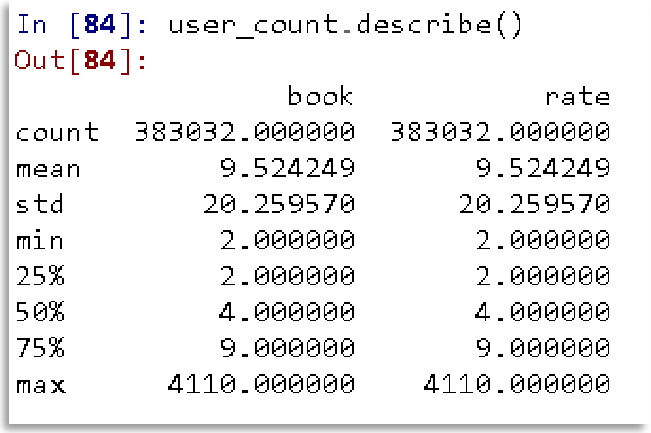
```
In [80]: book_count.user.hist(bins=100)
Out[80]: <matplotlib.axes._subplots.AxesSubplot at 0x125430b0>
```



```
In [82]: book_count.user.describe()
Out[82]:
count    80008.000000
mean        45.596590
std         67.972087
min          1.000000
25%          8.000000
50%         15.000000
75%         54.000000
max       2071.000000
Name: user, dtype: float64
```



```
In [70]: user_count.book.hist(bins=100)
Out[70]: <matplotlib.axes._subplots.AxesSubplot at 0x1253ff10>
```



```
In [84]: user_count.describe()
Out[84]:
                book           rate
count  383032.000000  383032.000000
mean        9.524249       9.524249
std        20.259570      20.259570
min         2.000000       2.000000
25%         2.000000       2.000000
50%         4.000000       4.000000
75%         9.000000       9.000000
max      4110.000000    4110.000000
```

观察数据：评分

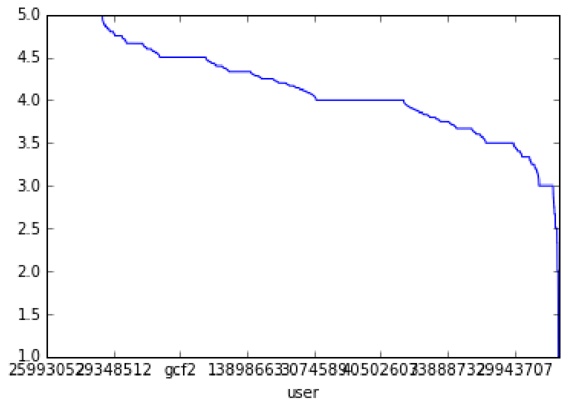
➢ **观察不同读者对书籍的评分**

- user_rate=douban.groupby('user').mean().sort('rate',ascending=False)
- user_rate.head()
- user_rate.describe()
- user_rate.rate.plot()
- user_rate.rate.hist()

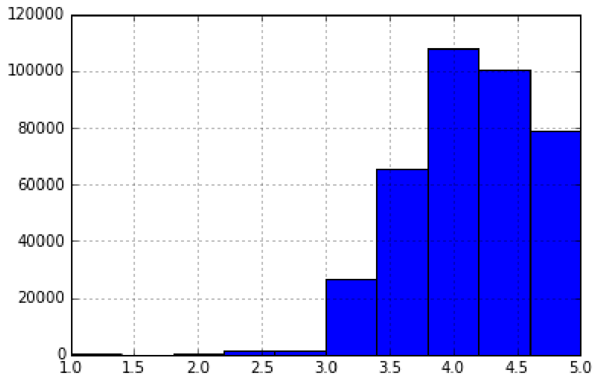
```
In [170]: user_rate.describe()
Out[170]:
                 book           rate
count   383032.000000  383032.000000
mean   2245810.790228       4.143778
std     768091.509089       0.536463
min    1000371.000000       1.000000
25%    1666521.660714       3.800000
50%    2116882.218750       4.125000
75%    2688075.500000       4.500000
max    5290775.500000       5.000000
```

```
In [130]: user_rate.head()
Out[130]:
                    book  rate
user
25993052  1077888.666667     5
2811970   2435402.000000     5
8099288   1765566.000000     5
28127007  1597994.600000     5
2812631   1981158.500000     5
```

```
In [131]: user_rate.rate.plot()
Out[131]: <matplotlib.axes._subplots.AxesSub
```



```
In [132]: user_rate.rate.hist()
Out[132]: <matplotlib.axes._subplots.AxesSubp
```



观察数据：评分

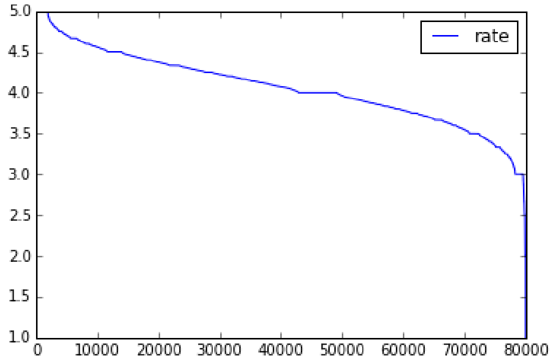
➢ **观察不同书的被所有读者的评分**

- book_rate=douban.groupby('book').mean().sort('rate',ascending=False)
- book_rate.head()
- book_rate.describe()
- book_rate.plot(use_index=False)
- book_rate.rate.hist()

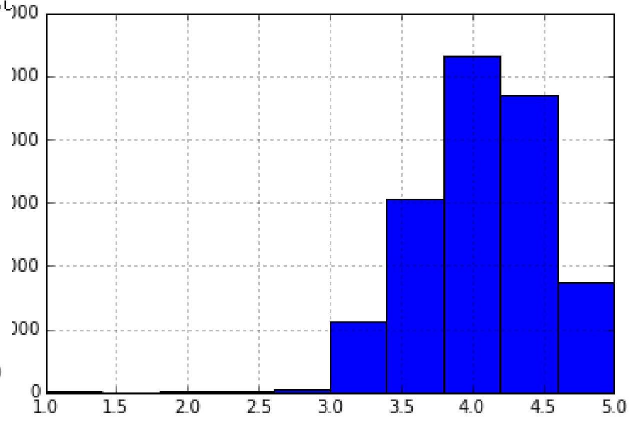
```
In [172]: book_rate.head()
Out[172]:
            rate
book
2179525       5
3686952       5
2683995       5
3689121       5
3689709       5
```

```
In [141]: book_rate.describe()
Out[141]:
               rate
count  80008.000000
mean       4.063981
std        0.444184
min        1.000000
25%        3.785714
50%        4.071429
75%        4.375000
max        5.000000
```

```
In [167]: book_rate.plot(use_index=False)
Out[167]: <matplotlib.axes._subplots.AxesSu
```

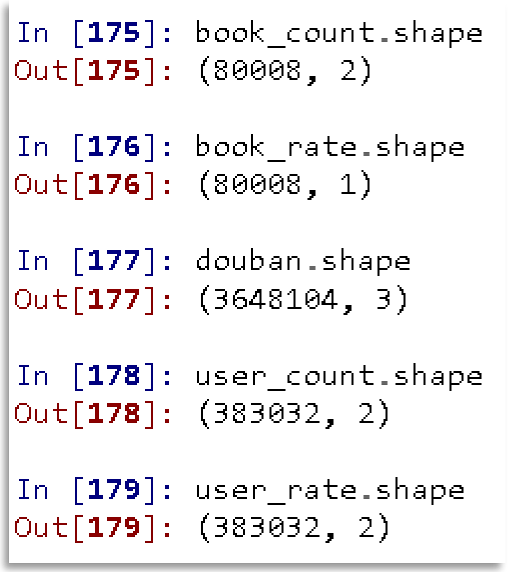


```
In [140]: book_rate.rate.hist()
:[140]: <matplotlib.axes._subplots.AxesSubpl
```



操作数据

- **目前查看的数据**
  - douban
    - user, book, rate
    - 原始的操作数据
  - user_count
    - user, book_count/rate_count
  - book_count
    - book,user_count/rate_count
  - user_rate
    - user, book_mean(dummy），rate_mean
  - book_rate
    - book,rate_mean
- **需要更方便的数据格式**

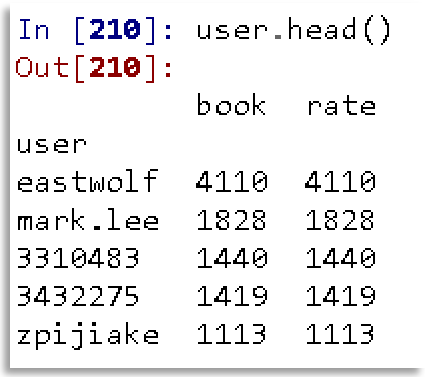


```
In [175]: book_count.shape
Out[175]: (80008, 2)

In [176]: book_rate.shape
Out[176]: (80008, 1)

In [177]: douban.shape
Out[177]: (3648104, 3)

In [178]: user_count.shape
Out[178]: (383032, 2)

In [179]: user_rate.shape
Out[179]: (383032, 2)
```

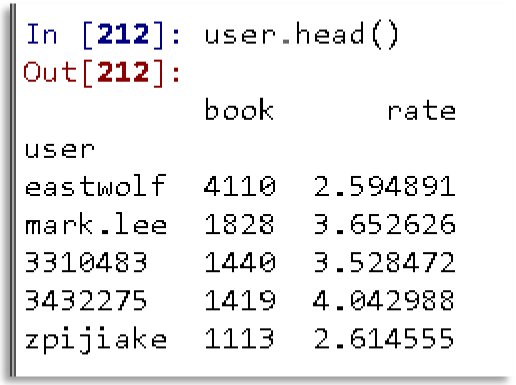
操作数据：用户

小象学院 ChinaHadoop.cn

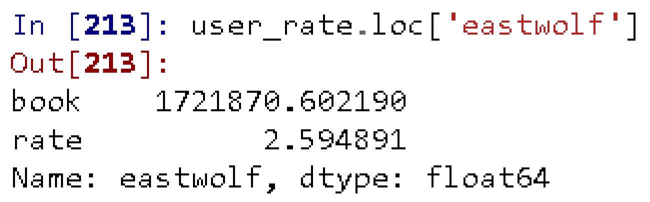
- 建立包含更全面信息的用户表
  - 用户，用户评价过的书籍数量，用户对所评价书籍的分数
    - user=user_count.copy()
    - user.head()
    - user['rate']=user_rate['rate']
    - user.head()
  - 检验一下：
    - user_rate.loc['eastwolf']



```
In [210]: user.head()
Out[210]:
           book  rate
user
eastwolf   4110  4110
mark.lee   1828  1828
3310483    1440  1440
3432275    1419  1419
zpijiake   1113  1113
```



```
In [212]: user.head()
Out[212]:
           book      rate
user
eastwolf   4110  2.594891
mark.lee   1828  3.652626
3310483    1440  3.528472
3432275    1419  4.042988
zpijiake   1113  2.614555
```



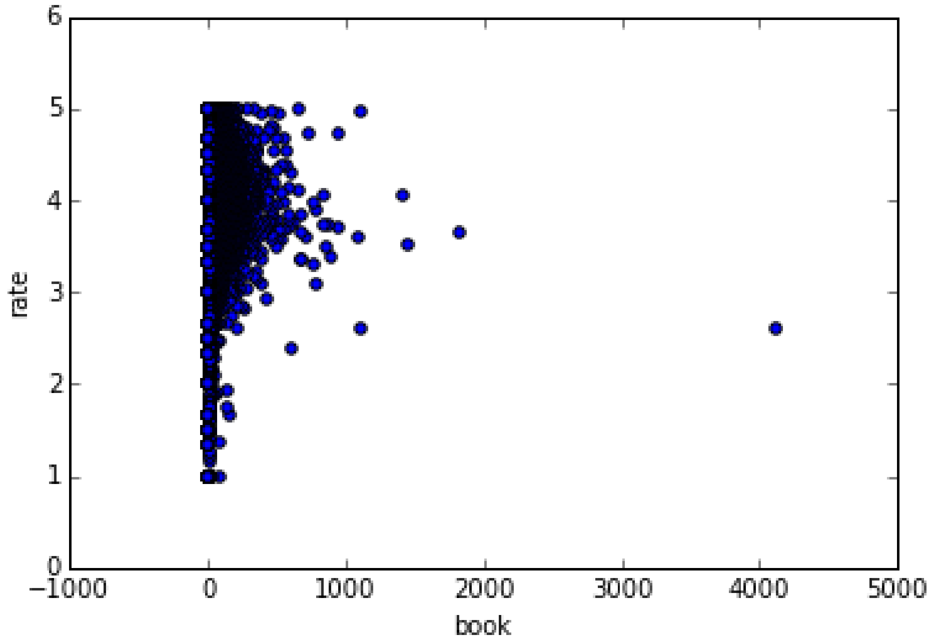
```
In [213]: user_rate.loc['eastwolf']
Out[213]:
book    1721870.602190
rate          2.594891
Name: eastwolf, dtype: float64
```

操作数据：进一步观察用户1

➢ **使用用户数据能做什么**

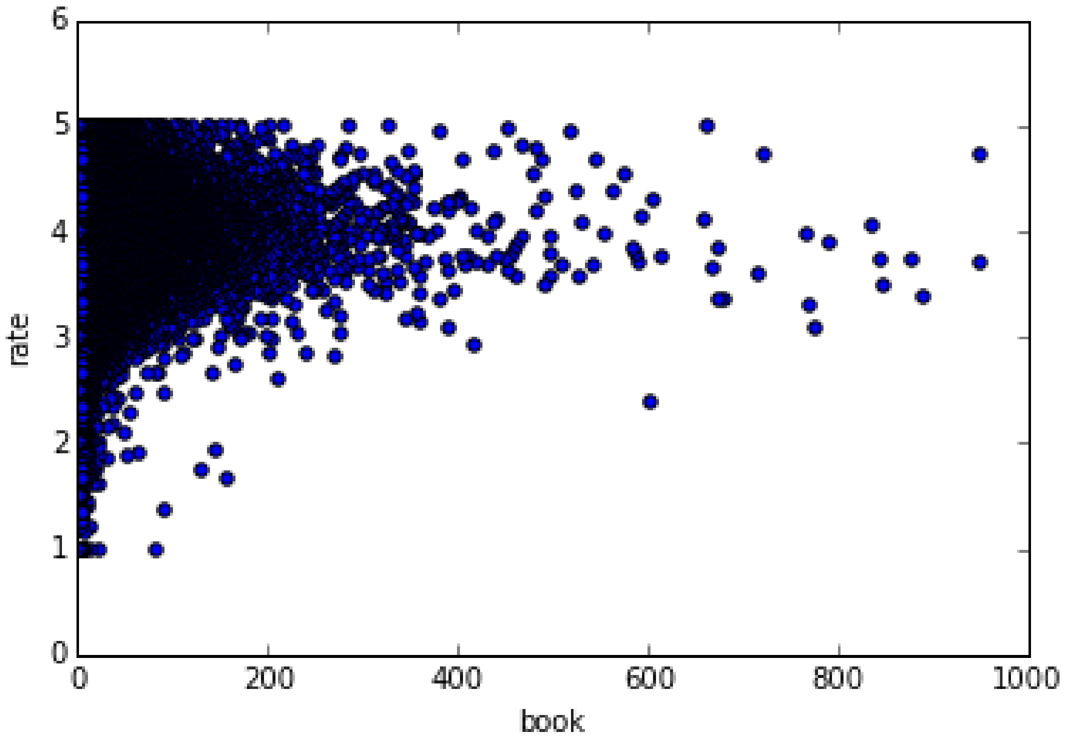
- 观察一下用户行为的趋势
- 是否存在不合理的地方？
  - user.plot(x='book',y='rate',kind='scatter')

```
In [214]: user.plot(x='book',y='rate',kind='sc
Out[214]: <matplotlib.axes._subplots.AxesSubpl
```



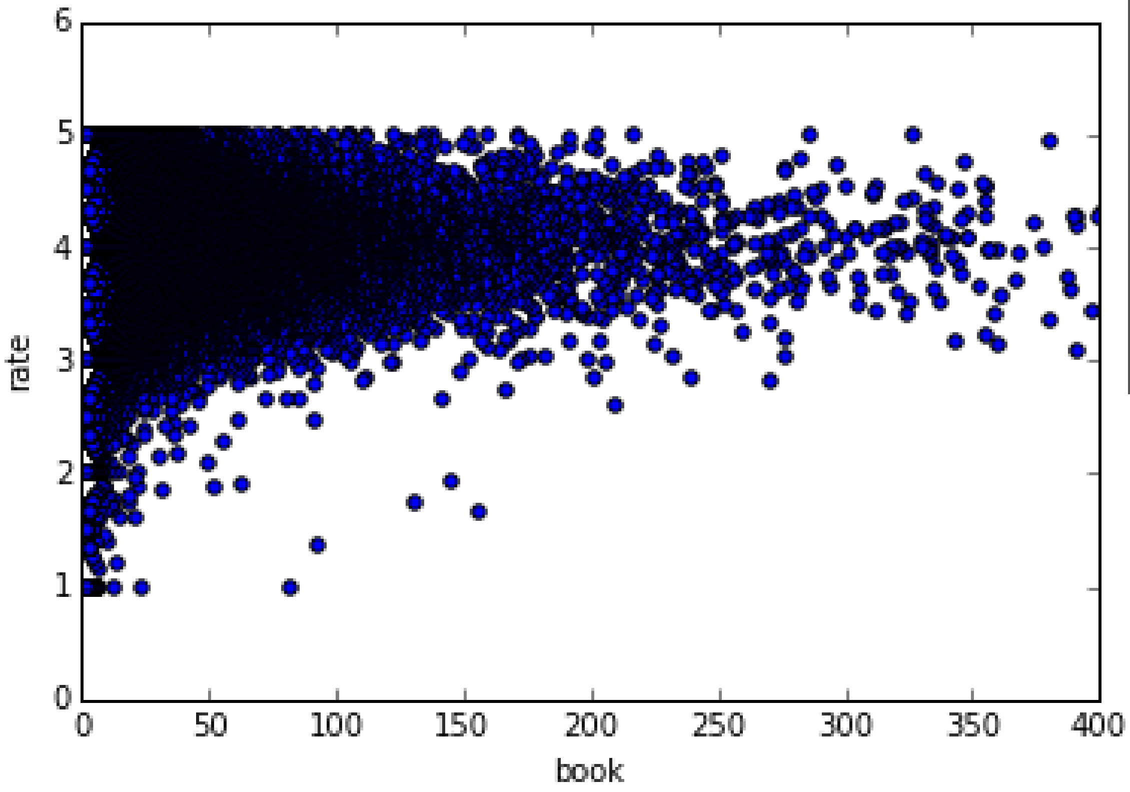
操作数据：进一步观察用户2

- **使用用户数据能做什么**
  - 观察一下用户行为的趋势
  - 是否存在不合理的地方？
    - user.plot(x='book',y='rate',kind='scatter',xlim=[0,1000])



操作数据：进一步观察用户3

- 使用用户数据能做什么
  - 观察一下用户行为的趋势
  - 是否存在不合理的地方？
    - user.plot(x='book',y='rate',kind='scatter',xlim=[0,400])



```
In [216]: douban.describe()
Out[216]:
                book            rate
count  3648104.000000  3648104.000000
mean   2176350.314101        4.090058
std    1083149.002175        0.835954
min    1000001.000000        1.000000
25%    1244300.000000        4.000000
50%    1890958.000000        4.000000
75%    3091356.000000        5.000000
max    5326850.000000        5.000000
In [217]: user.describe()
Out[217]:
                book           rate
count  383032.000000  383032.000000
mean        9.524249       4.143778
std        20.259570       0.536463
min         2.000000       1.000000
25%         2.000000       3.800000
50%         4.000000       4.125000
75%         9.000000       4.500000
max      4110.000000       5.000000
```

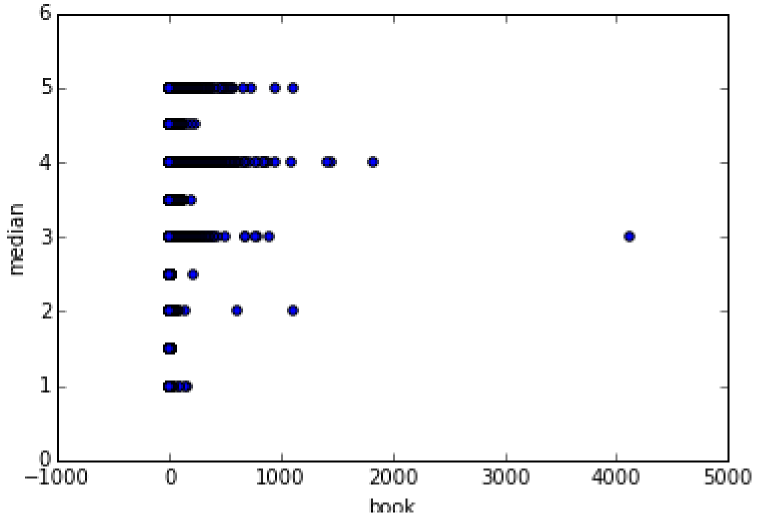
操作数据：继续补充用户数据

➢ **为用户表添加更多评价数据**

- user['min']=douban.groupby('user').min()['rate']
- user['max']=douban.groupby('user').max()['rate']
- user['median']=douban.groupby('user').median()['rate']
- user['std']=douban.groupby('user').std()['rate']
- user.head()
- user.plot(x='book',y='median',kind='scatter')

```
In [228]: user.head()
Out[228]:
           book      rate  min  max  median       std
user
eastwolf   4110  2.594891    1    5       3  0.820238
mark.lee   1828  3.652626    2    5       4  0.507426
3310483    1440  3.528472    1    5       4  0.649388
3432275    1419  4.042988    3    5       4  0.602286
zpijiake   1113  2.614555    1    5       2  1.584447
```

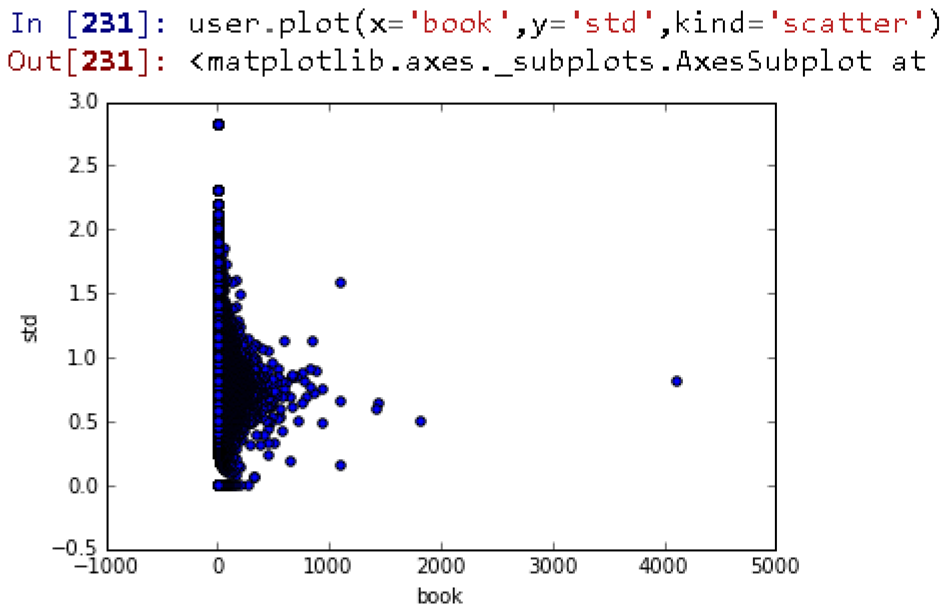
```
In [229]: user.plot(x='book',y='median',kind=
Out[229]: <matplotlib.axes._subplots.AxesSubp
```



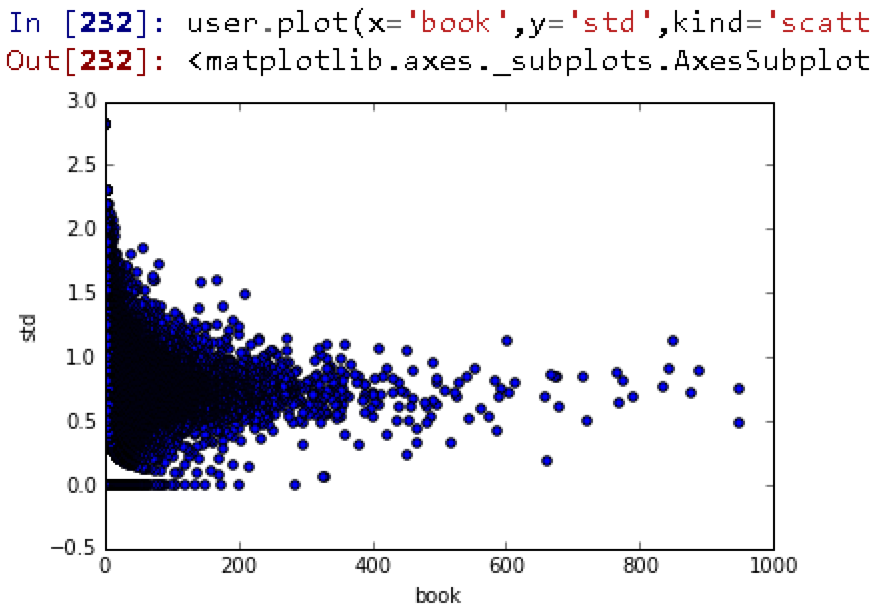
操作数据：继续补充用户数据

➢ **观察数据**

- user.plot(x='book',y='std',kind='scatter')
- user.plot(x='book',y='std',kind='scatter',xlim=[0,1000])



```
In [231]: user.plot(x='book',y='std',kind='scatter')
Out[231]: <matplotlib.axes._subplots.AxesSubplot at
```

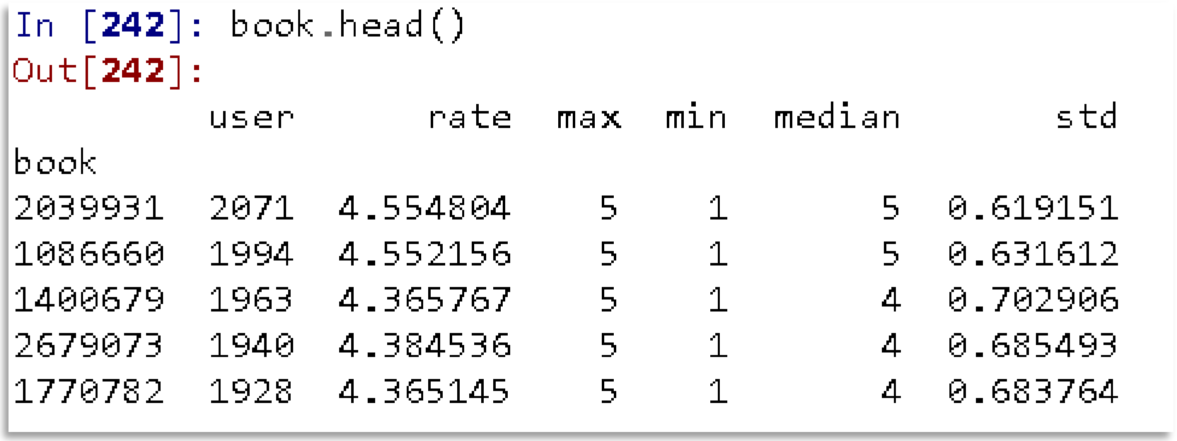


```
In [232]: user.plot(x='book',y='std',kind='scatt
Out[232]: <matplotlib.axes._subplots.AxesSubplot
```

操作数据：书籍数据

➢ **建立包含更全面信息的书籍表**

- book=book_count.copy()
- book['rate']=book_rate['rate']
- book['max']=douban.groupby('book').max()['rate']
- book['min']=douban.groupby('book').min()['rate']
- book['median']=douban.groupby('book').median()['rate']
- book['std']=douban.groupby('book').std()['rate']
- book.head()

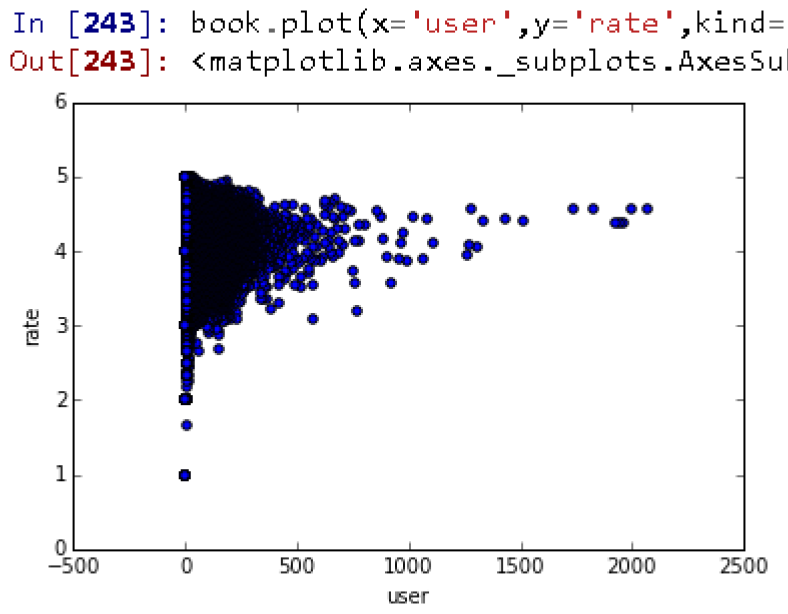


```
In [242]: book.head()
Out[242]:
         user      rate  max  min  median       std
book
2039931  2071  4.554804    5    1       5  0.619151
1086660  1994  4.552156    5    1       5  0.631612
1400679  1963  4.365767    5    1       4  0.702906
2679073  1940  4.384536    5    1       4  0.685493
1770782  1928  4.365145    5    1       4  0.683764
```

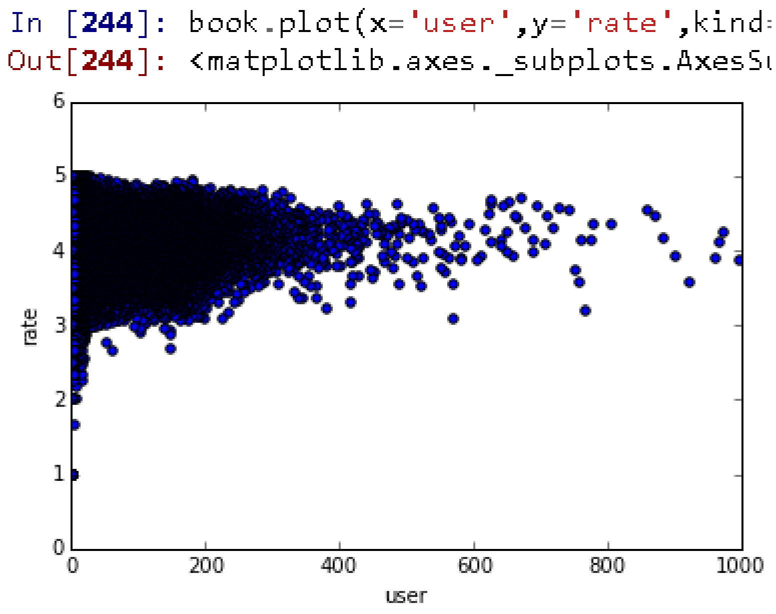
操作数据：书籍数据

➢ **进一步观察书籍数据**

- book.plot(x='user',y='rate',kind='scatter')
- book.plot(x='user',y='rate',kind='scatter',xlim=[0,1000])



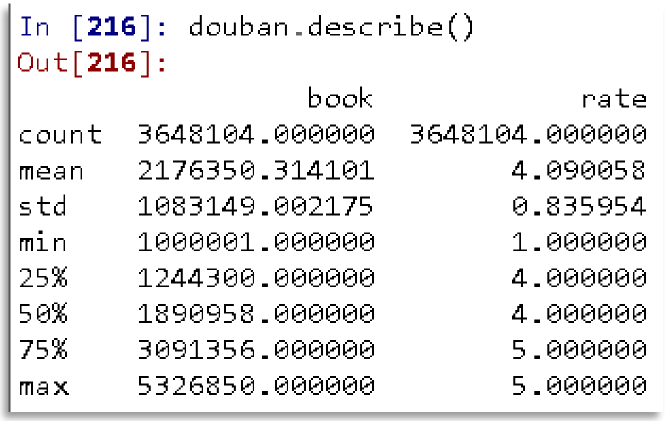
```
In [243]: book.plot(x='user',y='rate',kind='sc
Out[243]: <matplotlib.axes._subplots.AxesSubpl
```



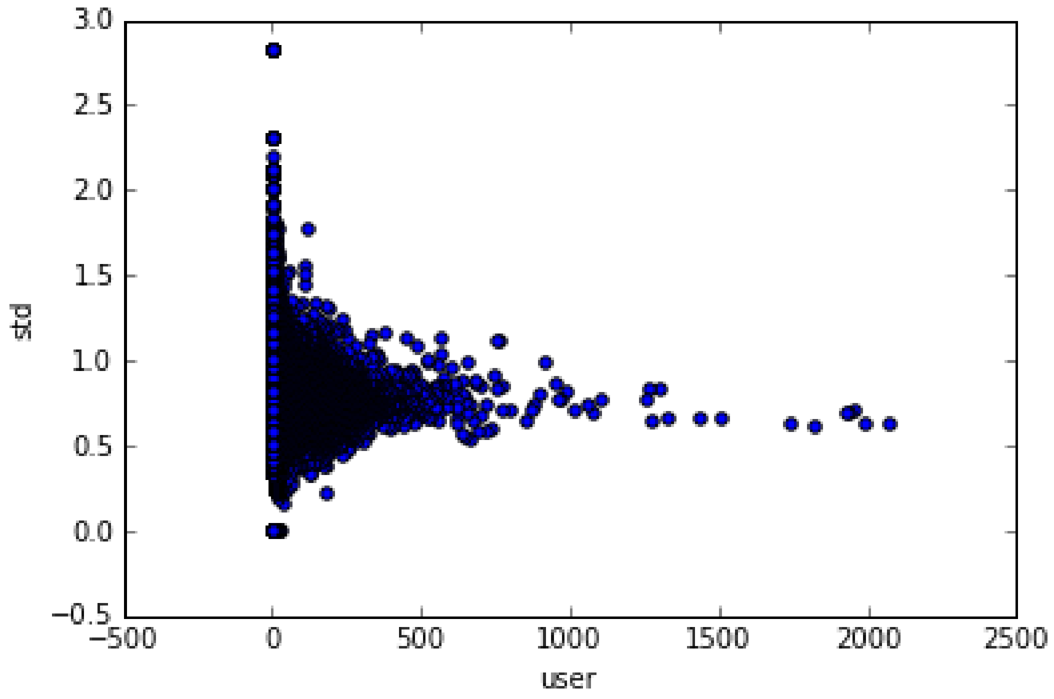
```
In [244]: book.plot(x='user',y='rate',kind=
Out[244]: <matplotlib.axes._subplots.AxesS
```

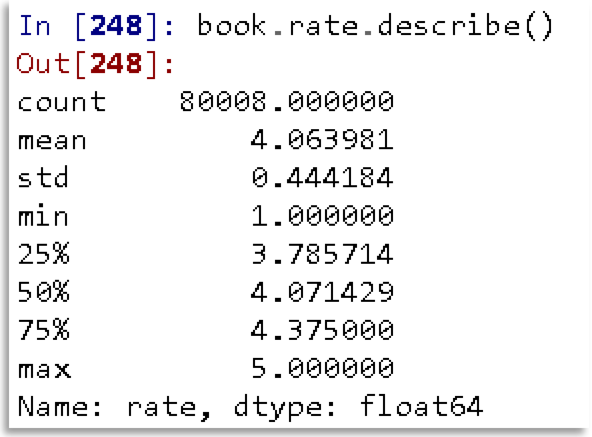
操作数据：书籍数据

- **进一步观察书籍数据**
  - book.plot(x='user',y='std',kind='scatter')
  - book.rate.describe()



```
In [216]: douban.describe()
Out[216]:
                 book            rate
count  3648104.000000  3648104.000000
mean   2176350.314101        4.090058
std    1083149.002175        0.835954
min    1000001.000000        1.000000
25%    1244300.000000        4.000000
50%    1890958.000000        4.000000
75%    3091356.000000        5.000000
max    5326850.000000        5.000000
```

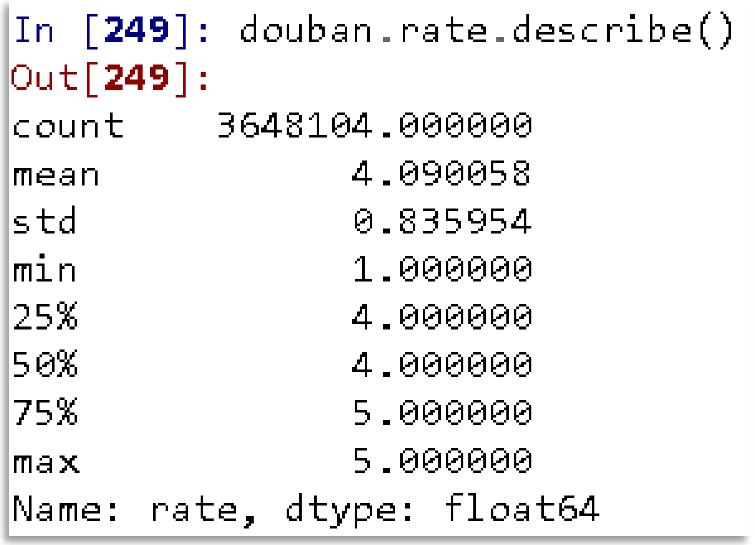




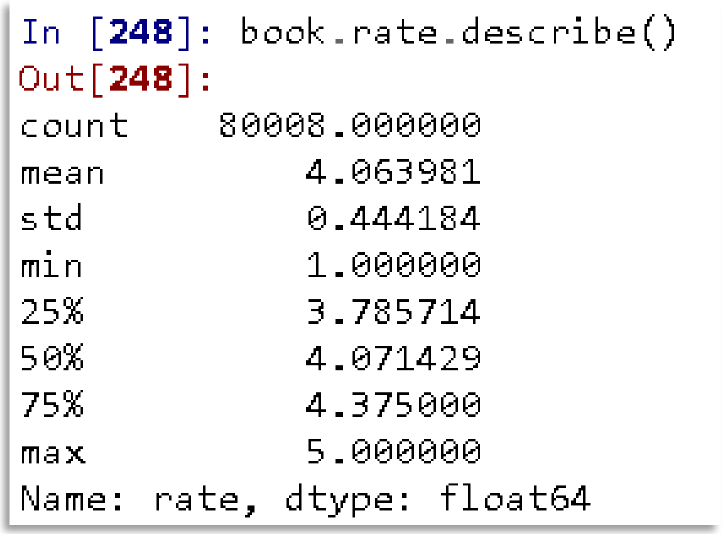
```
In [248]: book.rate.describe()
Out[248]:
count    80008.000000
mean         4.063981
std          0.444184
min          1.000000
25%          3.785714
50%          4.071429
75%          4.375000
max          5.000000
Name: rate, dtype: float64
```

操作数据

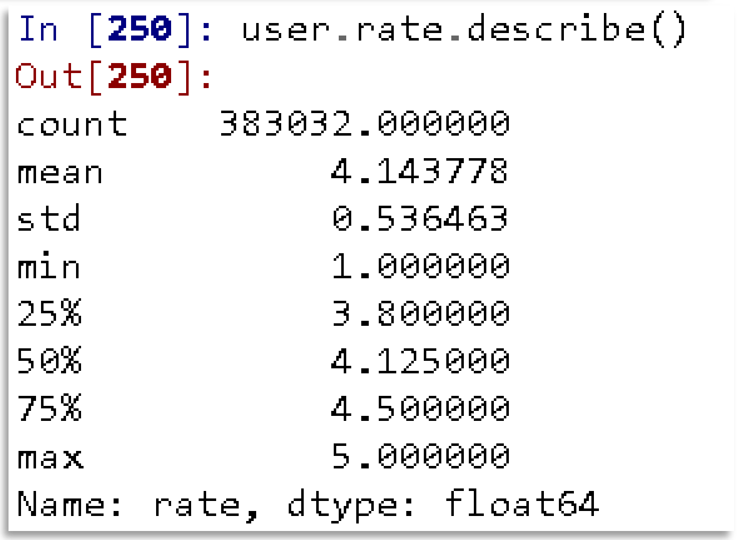
- ➢ **评价数据**
  - – 尝试解释



```
In [249]: douban.rate.describe()
Out[249]:
count    3648104.000000
mean           4.090058
std            0.835954
min            1.000000
25%            4.000000
50%            4.000000
75%            5.000000
max            5.000000
Name: rate, dtype: float64
```



```
In [248]: book.rate.describe()
Out[248]:
count    80008.000000
mean         4.063981
std          0.444184
min          1.000000
25%          3.785714
50%          4.071429
75%          4.375000
max          5.000000
Name: rate, dtype: float64
```



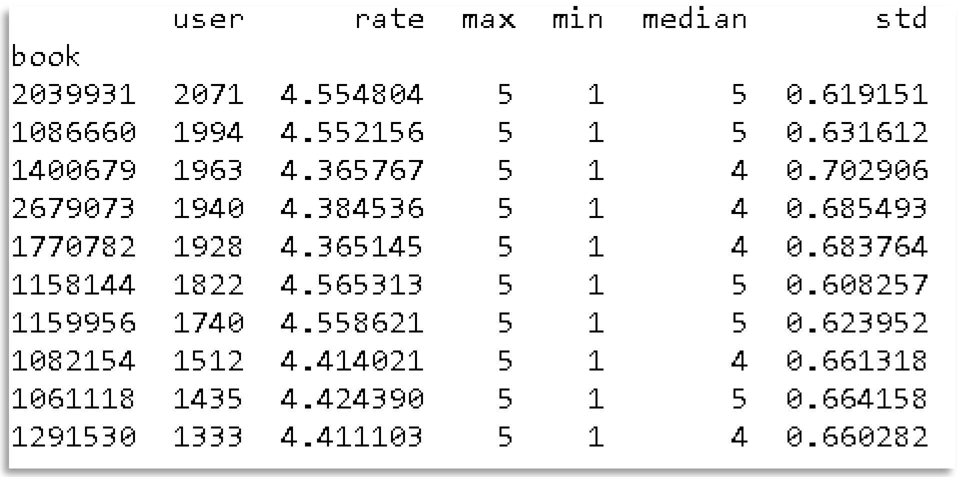
```
In [250]: user.rate.describe()
Out[250]:
count    383032.000000
mean          4.143778
std           0.536463
min           1.000000
25%           3.800000
50%           4.125000
75%           4.500000
max           5.000000
Name: rate, dtype: float64
```

发散一下：目前的数据能做什么？

- user
  - 作者阅读量排名
  - 作者读书兴趣分析：如果有书籍标签数据，可以做
  - 作者读书态度分析：是否更挑剔，或更容易满足
- book
  - 书的阅读量排名
  - 书的好评度排名
  - 书的信息相对容易获取：

发散一下：目前的数据能做什么？

- **书的阅读量排名**
  - https://book.douban.com/subject/1291530/



```
          user      rate  max  min  median       std
book
2039931   2071  4.554804    5    1       5  0.619151
1086660   1994  4.552156    5    1       5  0.631612
1400679   1963  4.365767    5    1       4  0.702906
2679073   1940  4.384536    5    1       4  0.685493
1770782   1928  4.365145    5    1       4  0.683764
1158144   1822  4.565313    5    1       5  0.608257
1159956   1740  4.558621    5    1       5  0.623952
1082154   1512  4.414021    5    1       4  0.661318
1061118   1435  4.424390    5    1       5  0.664158
1291530   1333  4.411103    5    1       4  0.660282
```

- 。。。

发散一下：目前的数据能做什么？

- book
  - 书籍的属性：
    - 书名-版本（出版社、版）-作者
    - 书名-品类（可能有多级）
    - 书名-标签
  - 缺失

回到原思路：

- **1. 提取特征：**
  - 读者：按照对书的评价（5分制，如无评价，对该书的评价为0）
  - 书籍：类似
- **2. 建立邻近性矩阵**
  - 参考文档向量，可以使用Jaccard或者余弦相似性（读者间，书籍间）
- **3. 使用K近邻算法找到与自己相似的若干对象**
- **4. 使用K-means聚类：读者、书**
- **5. 可使用协同过滤的方法（KNN）**

操作数据：

- 将“长格式”转化为“宽格式”
- 例：

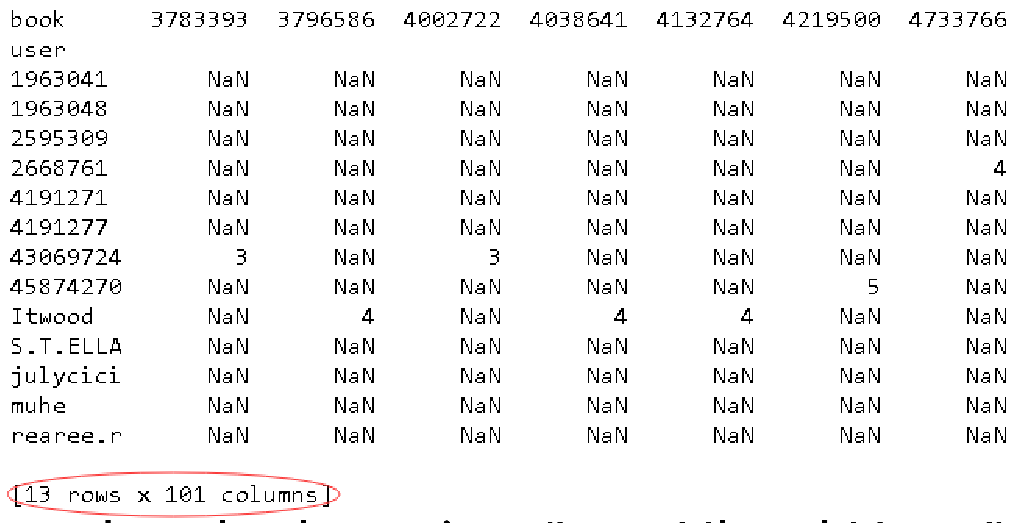
人员	物品	数量
甲	A	2
甲	B	3
乙	A	4
乙	C	5
丙	B	6

→

数量		物品		
		A	B	C
人员	甲	2	3	
	乙	4		5
	丙		6	

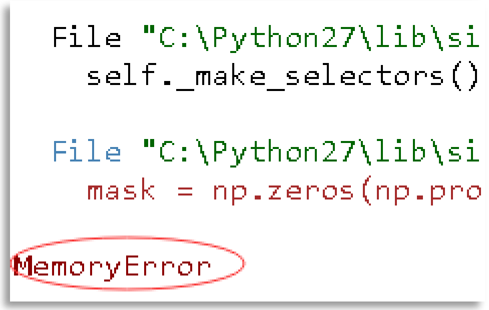
操作数据

- 将“长格式”转化为“宽格式”
  - df.pivot
  - dummy=douban.loc[0:100,['user','book','rate']]
  - dummy.pivot('user','book','rate')



```
book      3783393  3796586  4002722  4038641  4132764  4219500  4733766
user
1963041       NaN      NaN      NaN      NaN      NaN      NaN      NaN
1963048       NaN      NaN      NaN      NaN      NaN      NaN      NaN
2595309       NaN      NaN      NaN      NaN      NaN      NaN      NaN
2668761       NaN      NaN      NaN      NaN      NaN      NaN        4
4191271       NaN      NaN      NaN      NaN      NaN      NaN      NaN
4191277       NaN      NaN      NaN      NaN      NaN      NaN      NaN
43069724        3      NaN        3      NaN      NaN      NaN      NaN
45874270      NaN      NaN      NaN      NaN      NaN        5      NaN
Itwood        NaN        4      NaN        4        4      NaN      NaN
S.T.ELLA      NaN      NaN      NaN      NaN      NaN      NaN      NaN
julycici      NaN      NaN      NaN      NaN      NaN      NaN      NaN
muhe          NaN      NaN      NaN      NaN      NaN      NaN      NaN
rearee.r      NaN      NaN      NaN      NaN      NaN      NaN      NaN

[13 rows x 101 columns]
```

  - ubr=douban.pivot('user','book','rate')
    - 383032x80008



```
File "C:\Python27\lib\si
  self._make_selectors()

File "C:\Python27\lib\si
  mask = np.zeros(np.pro

MemoryError
```

- 问题来了：

操作数据：

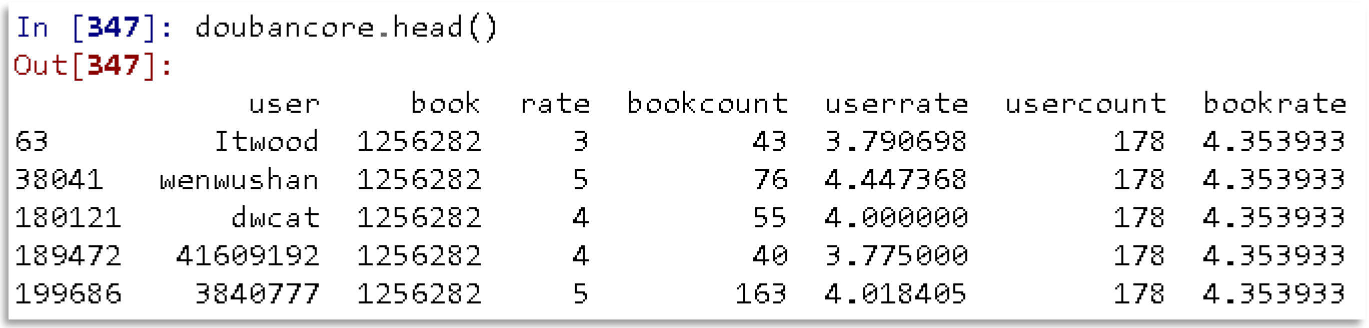
- **如果希望通过读者的读书习惯进行分类**
  - 由于数据的稀疏性，计算难以进行
  - 如果减少数据量，仍难以解决稀疏性

- **解决方案（仅供参考）**
  - 取评价书量最多的部分读者
  - 去被评价量最多的部分书籍
  - 假设各取前5%
    - 80008*0.05~4000
    - 383032*0.05~19152

操作数据：

➢ **选取核心读者和书籍**

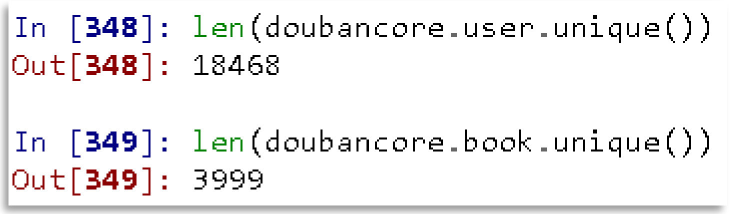
- usercore=user.iloc[0:19151,0:2]
- usercore.columns=['bookcount','userrate']
- bookcore=book.iloc[0:3999,0:2]
- bookcore.columns=['usercount','bookrate']
- doubancore1=pd.merge(douban,usercore,left_on='user', right_index=True)
- doubancore=pd.merge(doubancore1,bookcore,left_on='book',right_index=True)
- doubancore.head()



```
In [347]: doubancore.head()
Out[347]:
              user     book  rate  bookcount  userrate  usercount  bookrate
63         Itwood  1256282     3         43  3.790698        178  4.353933
38041   wenwushan  1256282     5         76  4.447368        178  4.353933
180121      dwcat  1256282     4         55  4.000000        178  4.353933
189472   41609192  1256282     4         40  3.775000        178  4.353933
199686    3840777  1256282     5        163  4.018405        178  4.353933
```

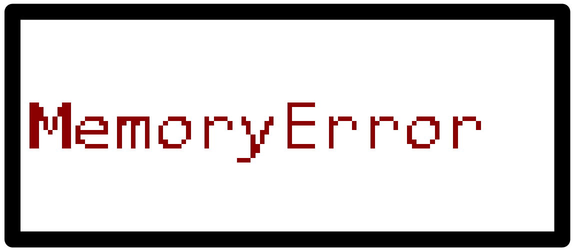
操作数据

- **选取核心读者和书籍**
  - len(doubancore.user.unique())
  - len(doubancore.book.unique())



```
In [348]: len(doubancore.user.unique())
Out[348]: 18468

In [349]: len(doubancore.book.unique())
Out[349]: 3999
```

  - ubrcore=doubancore.pivot('user','book','rate')

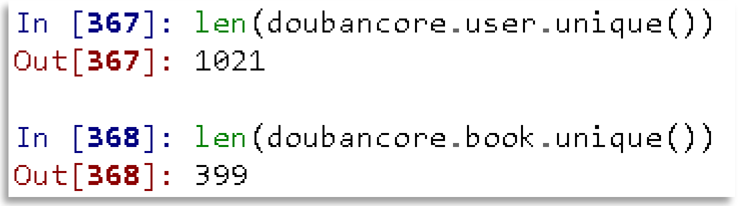


```
MemoryError
```

  - 还是过大，每个维度再缩小到1/10

操作数据

- **选取核心读者和书籍**
  - usercore=user.iloc[0:1915,0:2]
  - bookcore=book.iloc[0:399,0:2]
  - 。。。。。（略）
  - len(doubancore.user.unique())
  - len(doubancore.book.unique())



```
In [367]: len(doubancore.user.unique())
Out[367]: 1021

In [368]: len(doubancore.book.unique())
Out[368]: 399
```

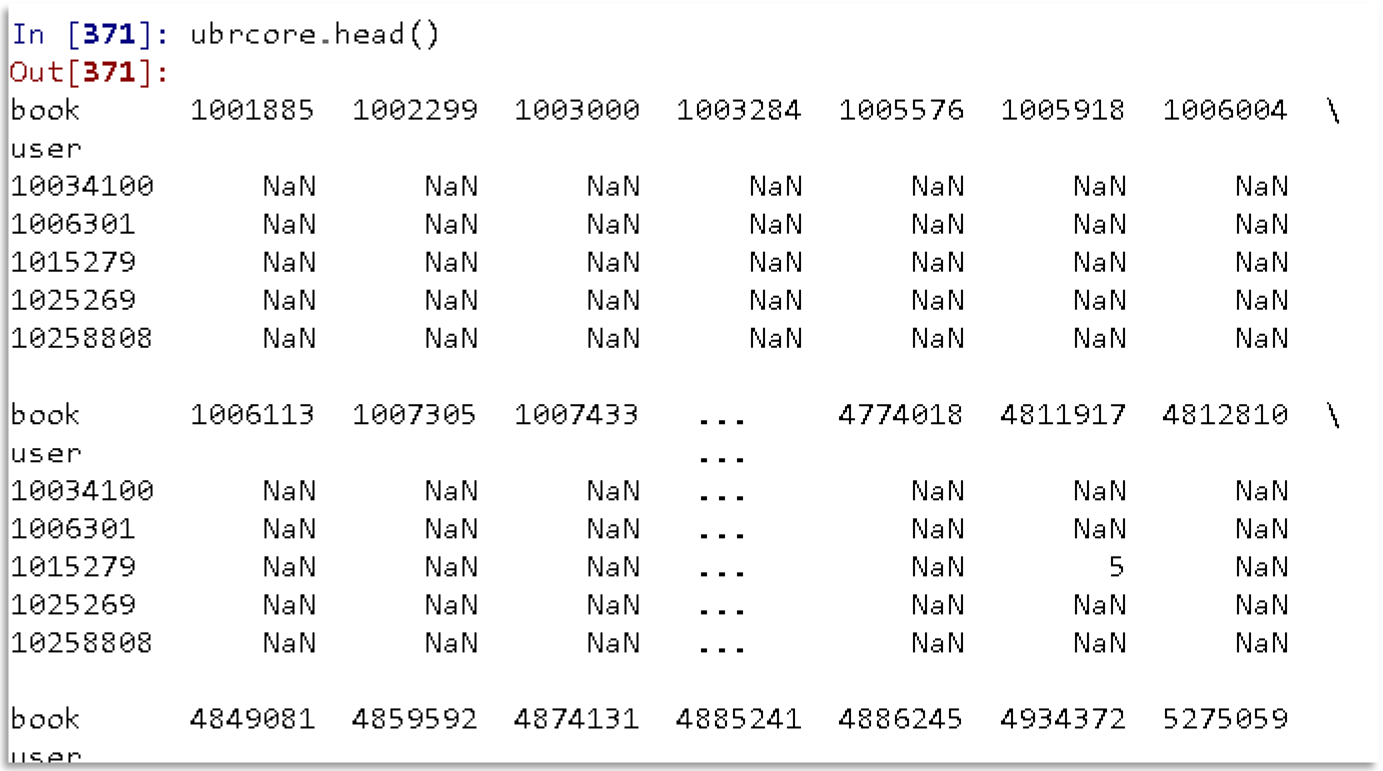
  - ubrcore=doubancore.pivot('user','book','rate')

- **注意：如何选取核心读者和书籍，可进化为两个参数，项目中应选取最佳参数**

操作数据

➢ **选取核心读者和书籍**

- ubrcore=doubancore.pivot('user','book','rate')

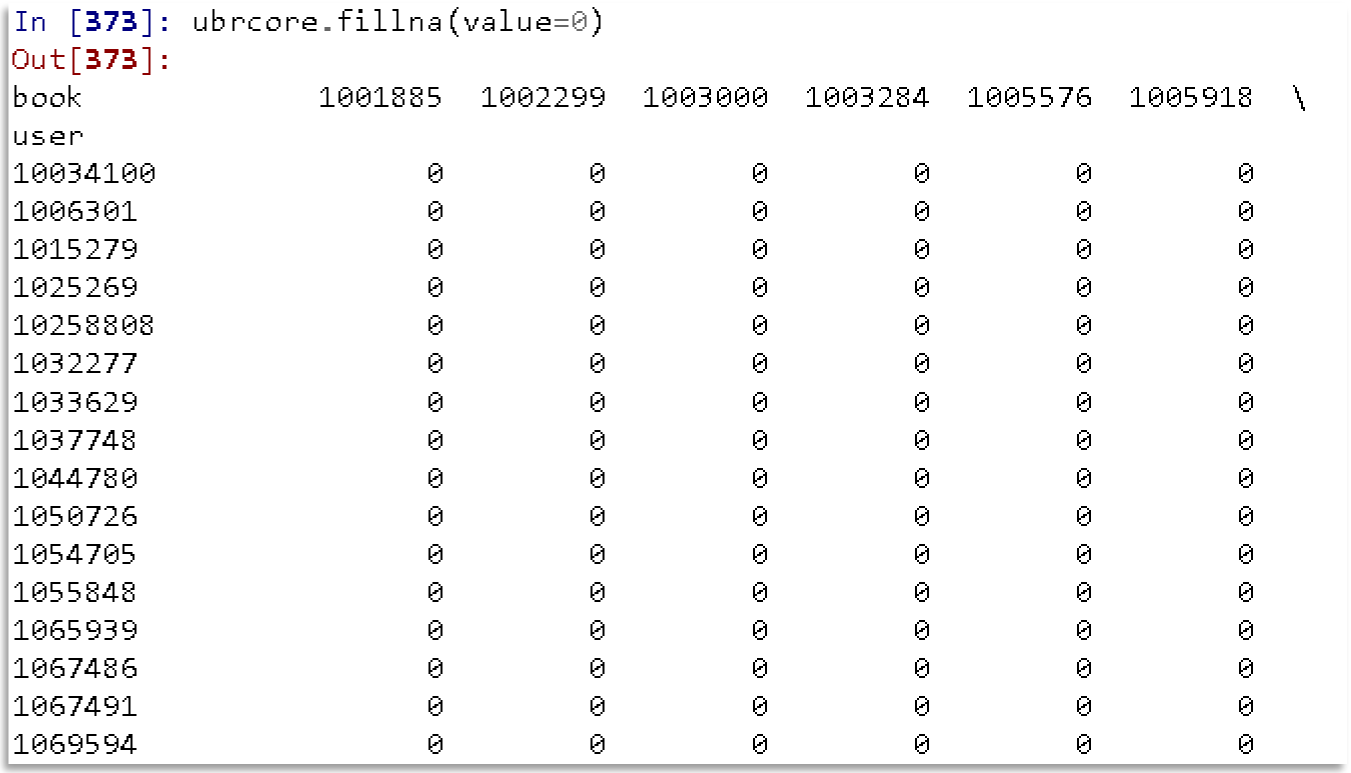


```
In [371]: ubrcore.head()
Out[371]:
book       1001885  1002299  1003000  1003284  1005576  1005918  1006004  \
user
10034100       NaN      NaN      NaN      NaN      NaN      NaN      NaN
1006301        NaN      NaN      NaN      NaN      NaN      NaN      NaN
1015279        NaN      NaN      NaN      NaN      NaN      NaN      NaN
1025269        NaN      NaN      NaN      NaN      NaN      NaN      NaN
10258808       NaN      NaN      NaN      NaN      NaN      NaN      NaN

book       1006113  1007305  1007433    ...     4774018  4811917  4812810  \
user                                    ...
10034100       NaN      NaN      NaN    ...         NaN      NaN      NaN
1006301        NaN      NaN      NaN    ...         NaN      NaN      NaN
1015279        NaN      NaN      NaN    ...         NaN        5      NaN
1025269        NaN      NaN      NaN    ...         NaN      NaN      NaN
10258808       NaN      NaN      NaN    ...         NaN      NaN      NaN

book       4849081  4859592  4874131  4885241  4886245  4934372  5275059
user
```

操作数据

➢ **选取核心读者和书籍**

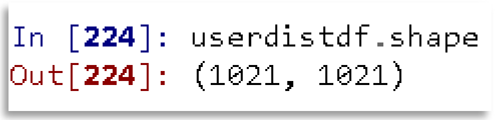
- ubrcore.fillna(value=0)



```
In [373]: ubrcore.fillna(value=0)
Out[373]:
book             1001885  1002299  1003000  1003284  1005576  1005918  \
user
10034100               0        0        0        0        0        0
1006301                0        0        0        0        0        0
1015279                0        0        0        0        0        0
1025269                0        0        0        0        0        0
10258808               0        0        0        0        0        0
1032277                0        0        0        0        0        0
1033629                0        0        0        0        0        0
1037748                0        0        0        0        0        0
1044780                0        0        0        0        0        0
1050726                0        0        0        0        0        0
1054705                0        0        0        0        0        0
1055848                0        0        0        0        0        0
1065939                0        0        0        0        0        0
1067486                0        0        0        0        0        0
1067491                0        0        0        0        0        0
1069594                0        0        0        0        0        0
```

计算邻近性矩阵

- 核心数据准备完毕
- 计算核心读者之间的距离或相似性
  - 使用余弦相似性
    - #calculate distance matrix
    - for i in range(m):
    -     for j in range(m):
    -         userdist[i,j]=np.dot(ubrcore.iloc[i,],ubrcore.iloc[j,]) \
    -         /np.sqrt(np.dot(ubrcore.iloc[i,],ubrcore.iloc[i,])\
    -         *np.dot(ubrcore.iloc[j,],ubrcore.iloc[j,]))
    - userdistdf=pd.DataFrame(userdist,index=list(ubrcore.index),columns=list(ubrcore.index))
    - userdistdf.to_csv('userdist.csv')
  - 计算相当长的时间



```
In [224]: userdistdf.shape
Out[224]: (1021, 1021)
```

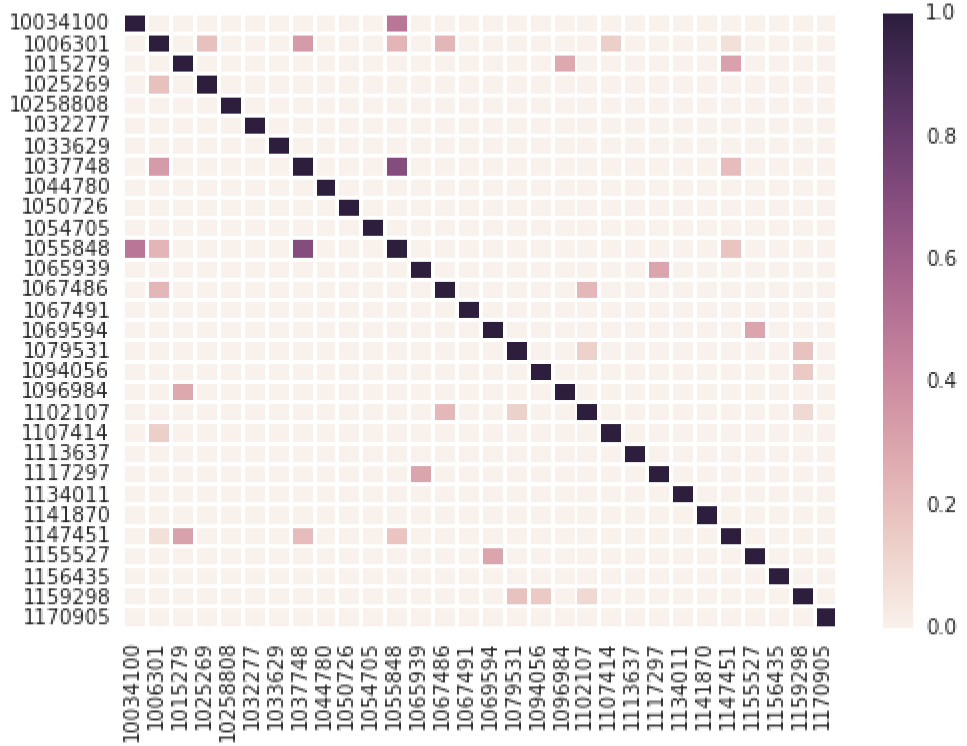
计算邻近性矩阵

小象学院 ChinaHadoop.cn

➢ **显示热力图**

- import seaborn as sns
- sns.set(style="white")
- sns.heatmap(userdistdf.iloc[0:30,0:30])

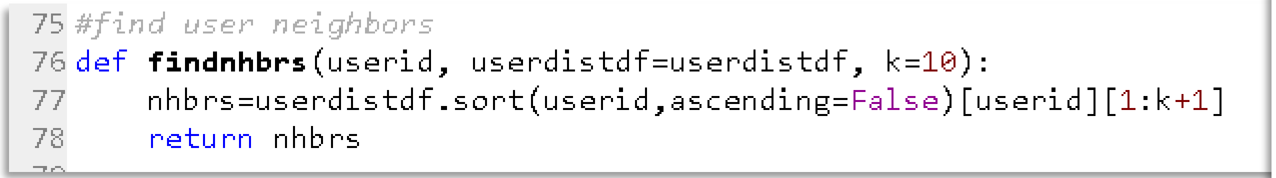
```
In [225]: sns.heatmap(userdistdf.iloc[0:30,0:30])
Out[225]: <matplotlib.axes._subplots.AxesSubplot at 0x1251
```



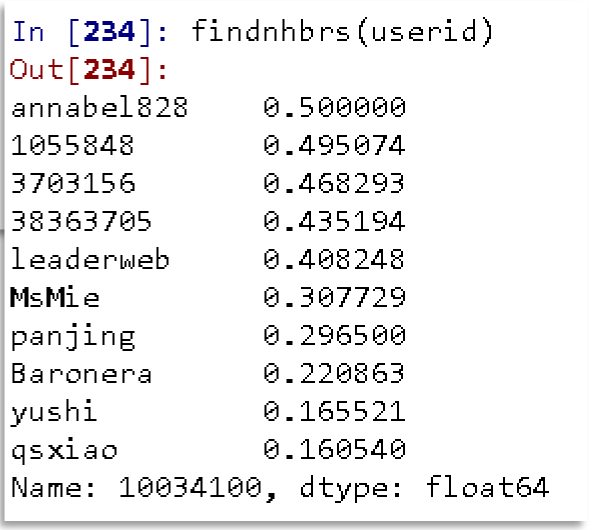
使用邻近性矩阵

➢ **进行相似性查找**

- #find user neighbors
- def findnhbrs(userid, userdistdf=userdistdf, k=10):
- nhbrs=userdistdf.sort(userid,ascending=False)[userid][1:k+1]
- return nhbrs



```
#find user neighbors
def findnhbrs(userid, userdistdf=userdistdf, k=10):
    nhbrs=userdistdf.sort(userid,ascending=False)[userid][1:k+1]
    return nhbrs
```

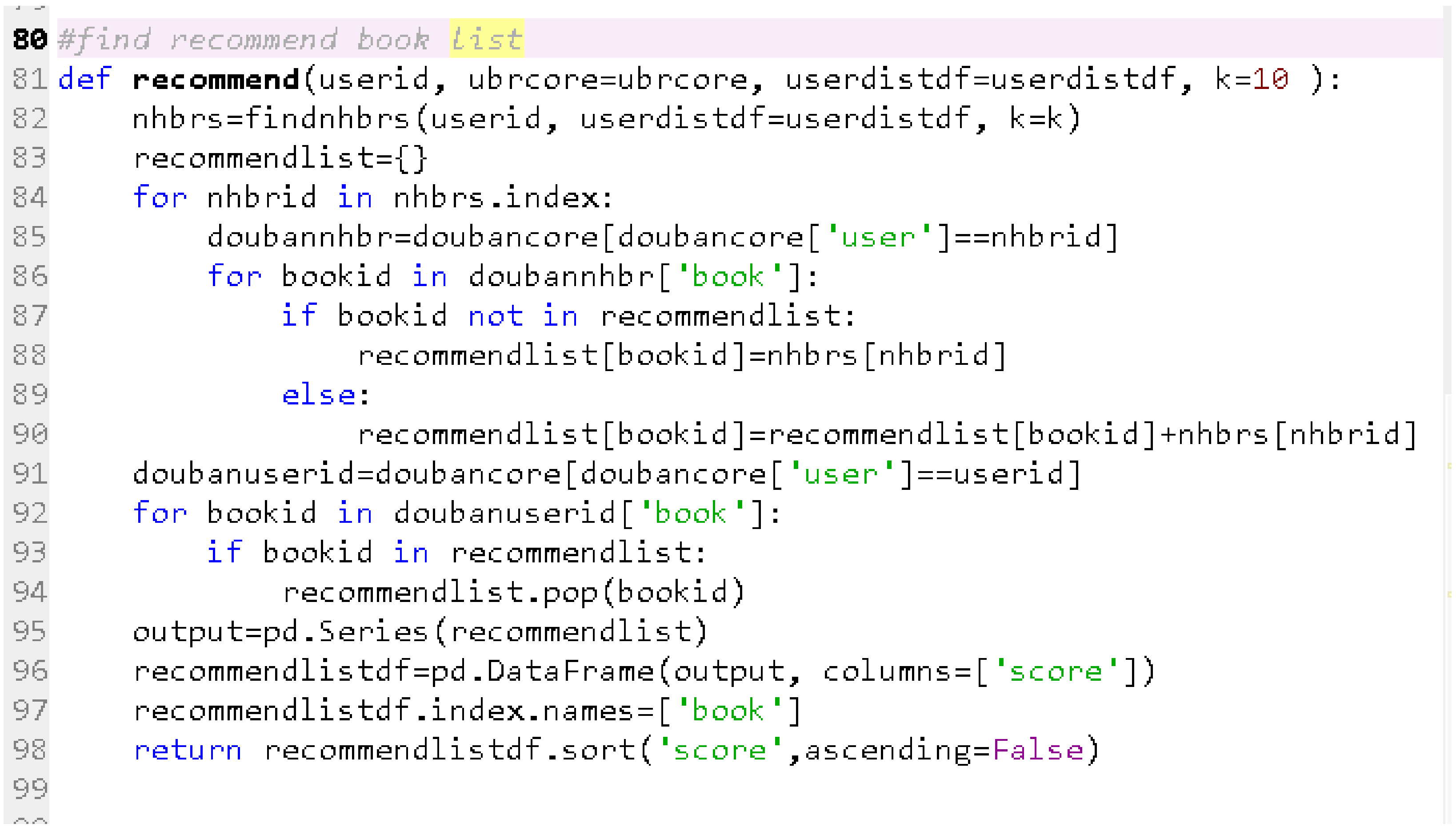


```
In [234]: findnhbrs(userid)
Out[234]:
annabel828     0.500000
1055848        0.495074
3703156        0.468293
38363705       0.435194
leaderweb      0.408248
MsMie          0.307729
panjing        0.296500
Baronera       0.220863
yushi          0.165521
qsxiao         0.160540
Name: 10034100, dtype: float64
```

协同过滤进行推荐

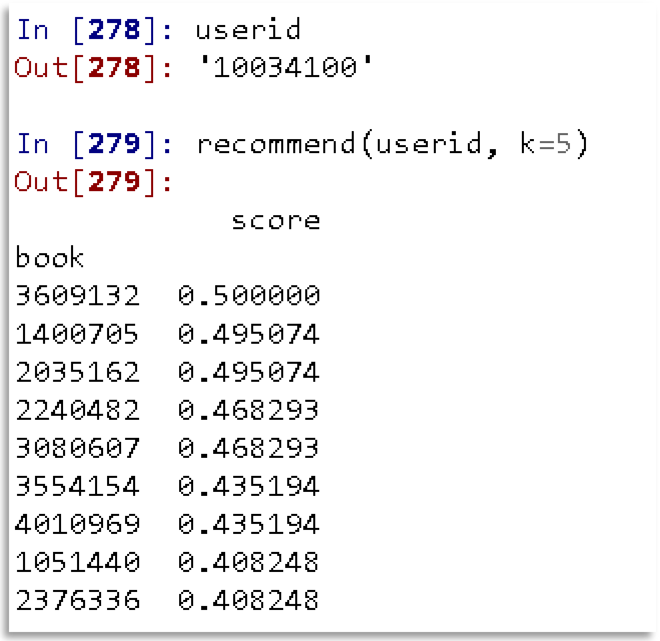
- **思路：**
  - 1. 给定用户，给定k值，通过邻近性矩阵找到k个近邻
  - 2. k个近邻用户所点评的书，作为推荐备选
    - 还可以通过评分过滤
    - 不过目前数据稀疏，暂不采用
  - 3. 使用与用户的邻近性值作为权重，作为推荐书籍的得分
    - 如果有多个近邻推荐同一本书，邻近性累加
  - 4. 排除用户本来就已经评价的书籍

协同过滤进行推荐

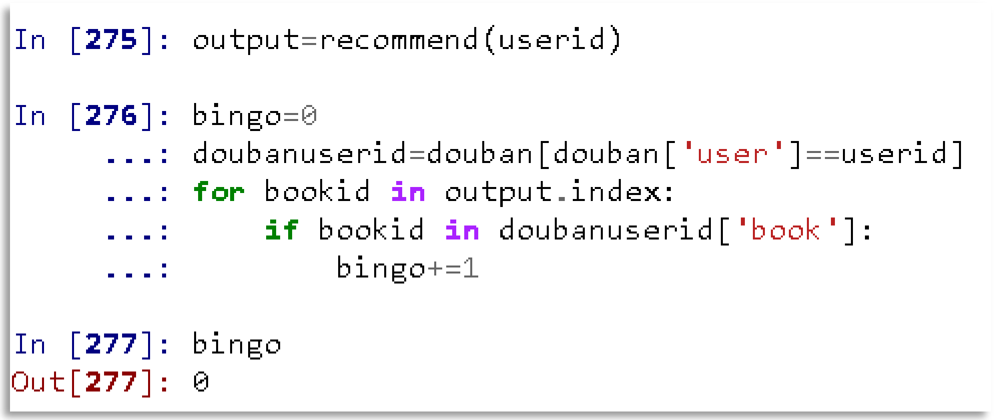


```
#find recommend book list
def recommend(userid, ubrcore=ubrcore, userdistdf=userdistdf, k=10 ):
    nhbrs=findnhbrs(userid, userdistdf=userdistdf, k=k)
    recommendlist={}
    for nhbrid in nhbrs.index:
        doubannhbr=doubancore[doubancore['user']==nhbrid]
        for bookid in doubannhbr['book']:
            if bookid not in recommendlist:
                recommendlist[bookid]=nhbrs[nhbrid]
            else:
                recommendlist[bookid]=recommendlist[bookid]+nhbrs[nhbrid]
    doubanuserid=doubancore[doubancore['user']==userid]
    for bookid in doubanuserid['book']:
        if bookid in recommendlist:
            recommendlist.pop(bookid)
    output=pd.Series(recommendlist)
    recommendlistdf=pd.DataFrame(output, columns=['score'])
    recommendlistdf.index.names=['book']
    return recommendlistdf.sort('score',ascending=False)
```

协同过滤进行推荐



```
In [278]: userid
Out[278]: '10034100'

In [279]: recommend(userid, k=5)
Out[279]:
                score
book
3609132  0.500000
1400705  0.495074
2035162  0.495074
2240482  0.468293
3080607  0.468293
3554154  0.435194
4010969  0.435194
1051440  0.408248
2376336  0.408248
```



```
In [275]: output=recommend(userid)

In [276]: bingo=0
     ...: doubanuserid=douban[douban['user']==userid]
     ...: for bookid in output.index:
     ...:     if bookid in doubanuserid['book']:
     ...:         bingo+=1

In [277]: bingo
Out[277]: 0
```

讨论

- 1. 同理可用作为书推荐读者
  - 邻近性矩阵
- 2. 通过排序切块似乎并不能很好的解决数据稀疏问题
  - 凝聚：外部信息，将图书归类
  - 聚类：计算量很大
- 3. 使用关联分析进行推荐
  - 数据量过大，此处略

联系我们：

- 新浪微博：ChinaHadoop
- 微信公号：ChinaHadoop
- 网站：http://chinahadoop.cn
- 问答社区：http://wenda.ChinaHadoop.cn