pandas排序与合并

sort
rank
merge
concat

In [1]:
```
import pandas as pd
import numpy as np
```

sort

In [2]:
```
ser = pd.Series(list("CAB"), index=[2, 1, 3])
ser
```

Out[2]:
```
2    C
1    A
3    B
dtype: object
```

sort_index() 按照index排序

In [3]:
```
ser.sort_index()
```

Out[3]:
```
1    A
2    C
3    B
dtype: object
```

In [4]:
```
ser.sort_index(ascending=False)
```

Out[4]:
```
3    B
2    C
1    A
dtype: object
```

sort_values() 按照值的大小排序

In [5]:
```
ser.sort_values()
```

Out[5]:
```
1    A
3    B
2    C
dtype: object
```

In [6]:
```
ser.sort_values(ascending=False)
```

Out[6]:
```
2    C
3    B
1    A
dtype: object
```

数据中有None时，无论升序还是降序，None值都排到最后

In [7]:
```
ser1 = pd.Series(['E', None, 'C'])
ser1
```

Out[7]:
```
0       E
1    None
2       C
dtype: object
```

In [8]:
```
ser1.sort_values()
```

Out[8]:
```
2       C
0       E
1    None
dtype: object
```

In [9]:
```
ser1.sort_values(ascending=False)
```

Out[9]:
```
0       E
2       C
1    None
dtype: object
```

DataFrame的排序

In [10]:
```
arr = [[9, 4, 8],
       [4, 6, 7],
       [4, 5, 3]]
df = pd.DataFrame(arr, index=[0, 2, 1], columns=list('cab'))
df
```

Out[10]:

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

In [11]:
```
df.sort_index()          # 按index排序
```

Out[11]:

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

In [12]:
```
df.sort_index(axis=1)    # 按columns排序
```

Out[12]:

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

In [13]:
```
df
```

Out[13]:

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

In [14]:
```
df.sort_values(by='c')        # 按照columns的'c'排序
```

Out[14]:

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

In [15]:
```
df.sort_values(by=['c', 'a'])        # 按照columns的'c'排序, 'c'相同则按照'a'排序
```

Out[15]:

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

rank

In [16]:
```
df
```

Out[16]:

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

In [17]:
```
df.rank()        # 按每一列排序
```

Out[17]:

	c	a	b
0	3.0	1.0	3.0
2	1.5	3.0	2.0
1	1.5	2.0	1.0

1.5 是因为两个数相同，一个1，一个2，取平均值1.5

In [18]:
```
df.rank(method="first")        # 默认为average
```

Out[18]:

	c	a	b
0	3.0	1.0	3.0
2	1.0	3.0	2.0
1	2.0	2.0	1.0

first模式下，若有相同的值，先遇到的数字小

In [19]:
```
df.rank(method="max")
```

Out[19]:

	c	a	b
0	3.0	1.0	3.0
2	2.0	3.0	2.0
1	2.0	2.0	1.0

max模式下，若有相同的值，取最大的排序数。如上面，两个数相同，一个应该为1，一个为2，取大的2作为两个的排序

In [20]:
```
df.rank(method="min")
```

Out[20]:

	c	a	b
0	3.0	1.0	3.0
2	1.0	3.0	2.0
1	1.0	2.0	1.0

与max模式正好相反

In [21]:
```
df.rank(method="min", ascending=False)
```

Out[21]:

	c	a	b
0	1.0	3.0	1.0
2	2.0	1.0	2.0
1	2.0	2.0	3.0

merge

In [22]:
```
df1 = pd.DataFrame({
    'stu_no': ['s1','s2','s1','s3','s1','s1','s2','s4'],
    'score': np.random.randint(50, 100, size=8)
})
df1
```

Out[22]:

	stu_no	score
0	s1	60
1	s2	69
2	s1	66
3	s3	50
4	s1	50
5	s1	78
6	s2	68
7	s4	71

In [23]:
```
df2 = pd.DataFrame({
    'stu_no': ['s1','s2','s3','s5'],
    'name': ['张三','李四','王五','赵六']
})
df2
```

Out[23]:

	stu_no	name
0	s1	张三
1	s2	李四
2	s3	王五
3	s5	赵六

In [24]:
```
pd.merge(df1, df2, on="stu_no")            # on表示以'stu_no'为标准
```

Out[24]:

	stu_no	score	name
0	s1	60	张三
1	s1	66	张三
2	s1	50	张三
3	s1	78	张三
4	s2	69	李四
5	s2	68	李四
6	s3	50	王五

In [25]:
```
pd.merge(df1, df2, on="stu_no", how="left")    # left表示，以左边的df1为准，默认为inner
```

Out[25]:

	stu_no	score	name
0	s1	60	张三
1	s2	69	李四
2	s1	66	张三
3	s3	50	王五
4	s1	50	张三
5	s1	78	张三
6	s2	68	李四
7	s4	71	NaN

In [26]:
```
pd.merge(df1, df2, on="stu_no", how="right")
```

Out[26]:

	stu_no	score	name
0	s1	60.0	张三
1	s1	66.0	张三
2	s1	50.0	张三
3	s1	78.0	张三
4	s2	69.0	李四
5	s2	68.0	李四
6	s3	50.0	王五
7	s5	NaN	赵六

In [27]:
```
pd.merge(df1, df2, on="stu_no", how="outer")
```

Out[27]:

	stu_no	score	name
0	s1	60.0	张三
1	s1	66.0	张三
2	s1	50.0	张三
3	s1	78.0	张三
4	s2	69.0	李四
5	s2	68.0	李四
6	s3	50.0	王五
7	s4	71.0	NaN
8	s5	NaN	赵六

concat

In [28]:
```
df3 = pd.DataFrame(np.arange(4).reshape(2, -1))
df3
```

Out[28]:

	0	1
0	0	1
1	2	3

In [29]:
```
df4 = pd.DataFrame(np.zeros(4).reshape(2, -1))
df4
```

Out[29]:

	0	1
0	0.0	0.0
1	0.0	0.0

In [30]:
```
pd.concat([df3, df4])       # 一定要用中括号括起来          竖着连接
```

Out[30]:

	0	1
0	0.0	1.0
1	2.0	3.0
0	0.0	0.0
1	0.0	0.0

In [31]:
```
pd.concat([df3, df4], axis=1)       # 横着连接
```

Out[31]:

	0	1	0	1
0	0	1	0.0	0.0
1	2	3	0.0	0.0