Pandas Learning

库的导入

```
# 导入numpy库并重命名为np
import numpy as np
# 导入pandas 库并重命名为pd
import pandas as pd
```

数据的导入

```
pd.read_csv(filename)   # 从csv导入
pd.read_table(filename) # 导入有分隔符的文本 (如TSV) 中的数据
pd.read_excel(filename) # 从excel导入
pd.read_sql(query, connection_object) # 导入SQL数据表/数据库中的数据
pd.read_json(json_string) # 导入JSON格式的字符，URL地址或者文件中的数据
pd.read_html(url) # 导入经过解析的URL地址中包含的数据框 (DataFrame) 数据
pd.read_clipboard() # 导入系统粘贴板里面的数据
pd.DataFrame(dict)  # 导入Python字典 (dict) 里面的数据，其中key是数据框的表头，value是数据框的内容。
```

数据的导出

```
df.to_csv(filename) # 将数据框 (DataFrame)中的数据导入csv格式的文件中
df.to_excel(filename) # 将数据框 (DataFrame)中的数据导入Excel格式的文件中
df.to_sql(table_name,connection_object) # 将数据框(DataFrame)中的数据导入SQL数据表/数据库中
df.to_json(filename) # 将数据框 (DataFrame)中的数据导入JSON格式的文件中
```

创建测试对象

```
pd.DataFrame(np.random.rand(5, 10)) # 创建一个5列10行的由随机浮点数组成的数据框 DataFrame
```

```
<tr style="text-align: right;">
  <th></th>
  <th>0</th>
  <th>1</th>
  <th>2</th>
  <th>3</th>
  <th>4</th>
  <th>5</th>
  <th>6</th>
  <th>7</th>
  <th>8</th>
  <th>9</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>0.016860</td>
  <td>0.855994</td>
  <td>0.992872</td>
  <td>0.652278</td>
  <td>0.517510</td>
  <td>0.742986</td>
  <td>0.452981</td>
  <td>0.568701</td>
  <td>0.795436</td>
  <td>0.622609</td>
</tr>
<tr>
  <th>1</th>
  <td>0.476801</td>
  <td>0.190823</td>
  <td>0.450436</td>
  <td>0.912401</td>
  <td>0.335651</td>
  <td>0.197766</td>
  <td>0.042523</td>
  <td>0.580323</td>
  <td>0.498982</td>
  <td>0.473128</td>
</tr>
<tr>
  <th>2</th>
  <td>0.029820</td>
  <td>0.886500</td>
  <td>0.902864</td>
  <td>0.465084</td>
  <td>0.380933</td>
  <td>0.033583</td>
```

```
  <td>0.928827</td>
  <td>0.501687</td>
  <td>0.857512</td>
  <td>0.671840</td>
</tr>
<tr>
  <th>3</th>
  <td>0.897254</td>
  <td>0.413717</td>
  <td>0.991061</td>
  <td>0.393033</td>
  <td>0.388630</td>
  <td>0.661025</td>
  <td>0.635417</td>
  <td>0.695609</td>
  <td>0.305378</td>
  <td>0.147508</td>
</tr>
<tr>
  <th>4</th>
  <td>0.573882</td>
  <td>0.786888</td>
  <td>0.177782</td>
  <td>0.864474</td>
  <td>0.594416</td>
  <td>0.765678</td>
  <td>0.217279</td>
  <td>0.446570</td>
  <td>0.930604</td>
  <td>0.686823</td>
</tr>
```

```
pd.Series(my_list) # 从一个可迭代的对象 my_list 中创建一个数据组
```

```
my_list = ['abc',123,'HelloWorld', 5.7]
pd.Series(my_list)
```

```
0           abc
1           123
2    HelloWorld
3           5.7
dtype: object
```

```
df = pd.DataFrame(np.random.rand(10, 5))
df.index = pd.date_range('2017/1/1', periods=df.shape[0])
df
```

```
<tr style="text-align: right;">
  <th></th>
  <th>0</th>
  <th>1</th>
  <th>2</th>
  <th>3</th>
  <th>4</th>
</tr>
```

```
<tr>
  <th>2017-01-01</th>
  <td>0.011762</td>
  <td>0.634116</td>
  <td>0.045220</td>
  <td>0.452117</td>
  <td>0.879969</td>
</tr>
<tr>
  <th>2017-01-02</th>
  <td>0.802262</td>
  <td>0.661908</td>
  <td>0.214822</td>
  <td>0.444259</td>
  <td>0.200370</td>
</tr>
<tr>
  <th>2017-01-03</th>
  <td>0.301050</td>
  <td>0.004534</td>
  <td>0.881042</td>
  <td>0.825632</td>
  <td>0.331118</td>
</tr>
<tr>
  <th>2017-01-04</th>
  <td>0.095324</td>
  <td>0.916430</td>
  <td>0.177795</td>
  <td>0.191502</td>
  <td>0.546973</td>
</tr>
<tr>
  <th>2017-01-05</th>
```

```
  <td>0.482868</td>
  <td>0.953719</td>
  <td>0.615461</td>
  <td>0.868984</td>
  <td>0.639286</td>
</tr>
<tr>
  <th>2017-01-06</th>
  <td>0.958404</td>
  <td>0.155357</td>
  <td>0.293012</td>
  <td>0.115218</td>
  <td>0.177846</td>
</tr>
<tr>
  <th>2017-01-07</th>
  <td>0.915488</td>
  <td>0.486922</td>
  <td>0.440474</td>
  <td>0.584764</td>
  <td>0.271243</td>
</tr>
<tr>
  <th>2017-01-08</th>
  <td>0.480413</td>
  <td>0.600622</td>
  <td>0.325212</td>
  <td>0.532259</td>
  <td>0.687718</td>
</tr>
<tr>
  <th>2017-01-09</th>
  <td>0.859887</td>

  <td>0.236677</td>
```

```
  <td>0.635073</td>
  <td>0.811840</td>
  <td>0.497289</td>
</tr>
<tr>
  <th>2017-01-10</th>
  <td>0.024623</td>
  <td>0.635122</td>
  <td>0.346393</td>
  <td>0.860260</td>
  <td>0.325502</td>
</tr>
```

数据的查看

```
df.head(n)  # 查看前n行的数据
df.tail(n)  # 查看后n行的数据
```

```
df = pd.DataFrame(np.random.rand(10, 5))
df.head(3)
```

```
<tr style="text-align: right;">
  <th></th>
  <th>0</th>
  <th>1</th>
  <th>2</th>
  <th>3</th>
  <th>4</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>0.996381</td>
  <td>0.440502</td>
  <td>0.583701</td>
  <td>0.120444</td>
  <td>0.241775</td>
</tr>
<tr>
  <th>1</th>
  <td>0.126877</td>
  <td>0.646841</td>
  <td>0.740163</td>
  <td>0.764182</td>
  <td>0.810129</td>
</tr>
<tr>
  <th>2</th>
  <td>0.254386</td>
  <td>0.451341</td>
  <td>0.288513</td>
  <td>0.515995</td>
  <td>0.146529</td>
</tr>
```

```
df = pd.DataFrame(np.random.rand(10, 5))
df.tail(3)
```

```
<tr style="text-align: right;">
  <th></th>
  <th>0</th>
  <th>1</th>
  <th>2</th>
  <th>3</th>
  <th>4</th>
</tr>
```

```
<tr>
  <th>7</th>
  <td>0.466316</td>
  <td>0.747013</td>
  <td>0.568442</td>
  <td>0.562552</td>
  <td>0.949529</td>
</tr>
<tr>
  <th>8</th>
  <td>0.243633</td>
  <td>0.605133</td>
  <td>0.114011</td>
  <td>0.898604</td>
  <td>0.024648</td>
</tr>
<tr>
  <th>9</th>
  <td>0.155605</td>
  <td>0.799580</td>
  <td>0.160883</td>
  <td>0.986743</td>
  <td>0.446114</td>
</tr>
```

查看数据的形状

```
df.shape # 查看数据的形状（行和宽）
```

查看数据的相关信息

```
df.info() # 查看数据的索引、数据类型及内存信息
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
0    10 non-null float64
1    10 non-null float64
2    10 non-null float64
3    10 non-null float64
4    10 non-null float64
dtypes: float64(5)
memory usage: 480.0 bytes
```

```
df.describe() # 对于数据类型为数值型的列，查询其描述性统计的内容
```

```
<tr style="text-align: right;">
  <th></th>
  <th>0</th>
  <th>1</th>
  <th>2</th>
  <th>3</th>
  <th>4</th>
</tr>
```

```
<tr>
  <th>count</th>
  <td>10.000000</td>
  <td>10.000000</td>
  <td>10.000000</td>
  <td>10.000000</td>
  <td>10.000000</td>
</tr>
<tr>
  <th>mean</th>
  <td>0.432296</td>
  <td>0.601318</td>
  <td>0.444123</td>
  <td>0.543053</td>
  <td>0.413944</td>
</tr>
<tr>
  <th>std</th>
  <td>0.268632</td>
  <td>0.216483</td>
  <td>0.298076</td>
  <td>0.284759</td>
  <td>0.345554</td>
</tr>
<tr>
  <th>min</th>
  <td>0.155605</td>
  <td>0.252712</td>
  <td>0.114011</td>
  <td>0.126172</td>
  <td>0.024648</td>
</tr>
<tr>
  <th>25%</th>
```

```
  <td>0.245740</td>
  <td>0.461876</td>
  <td>0.191090</td>
  <td>0.295224</td>
  <td>0.094655</td>
</tr>
<tr>
  <th>50%</th>
  <td>0.353927</td>
  <td>0.609304</td>
  <td>0.401004</td>
  <td>0.565072</td>
  <td>0.389053</td>
</tr>
<tr>
  <th>75%</th>
  <td>0.559964</td>
  <td>0.762463</td>
  <td>0.556721</td>
  <td>0.693000</td>
  <td>0.668621</td>
</tr>
<tr>
  <th>max</th>
  <td>0.985457</td>
  <td>0.932679</td>
  <td>0.990827</td>
  <td>0.986743</td>
  <td>0.949529</td>
</tr>
```

统计次数

```
s.value_counts(dropna=False) # 查询每个独特数据值出现次数统计
```

```
s = pd.Series([1,2,3,3,4,np.nan,5,5,5,6,7])
s.value_counts(dropna=False)
```

```
 5.0    3
 3.0    2
 7.0    1
 6.0    1
NaN     1
 4.0    1
 2.0    1
 1.0    1
dtype: int64
```

```
s = s.apply(lambda x: x+1)
s
```

```
0     3.0
1     4.0
2     5.0
3     5.0
4     6.0
5     NaN
6     7.0
7     7.0
8     7.0
9     8.0
10    9.0
dtype: float64
```

```
df.apply(pd.Series.value_counts) # 查询数据框 (Data Frame) 中每个列的独特数据值出现次数统计
```

数据选取

```
df[col] # 以数组 Series 的形式返回选取的列
```

```
df = pd.DataFrame(np.random.rand(5, 5), columns=list('ABCDE'))
df['C']
```

```
0    0.452717
1    0.407755
2    0.549391
3    0.759433
4    0.153871
Name: C, dtype: float64
```

```
df[[col1, col2]] # 选择多列
```

```
df = pd.DataFrame(np.random.rand(5, 5), columns=list('ABCDE'))
df[['C', 'D']]
```

```
<tr style="text-align: right;">
  <th></th>
  <th>C</th>
  <th>D</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>0.431885</td>
  <td>0.304796</td>
</tr>
<tr>
  <th>1</th>
  <td>0.028960</td>
  <td>0.187738</td>
</tr>
<tr>
  <th>2</th>
  <td>0.176520</td>
  <td>0.102980</td>
</tr>
<tr>
  <th>3</th>
  <td>0.370277</td>
  <td>0.098031</td>
</tr>
<tr>
  <th>4</th>
  <td>0.247122</td>
  <td>0.345735</td>
</tr>
```

```
s.iloc[0] # 按位置选取
```

```
s = pd.Series(np.array(['I', 'Love', 'China']))
s.iloc[0]
```

```
'I'
```

```
s.loc['index_one'] # 按索引选取
```

```
s = pd.Series(np.array(['I', 'Love', 'China']))
s.loc[0]
```

```
'I'
```

```
df.DataFrame[n, :] #选取第n行
```

```
df = pd.DataFrame(np.array([['I', 'Love', 'China'], ['I', 'Love', 'Data']]))
df.iloc[1, :]
```

```
0       I
1    Love
2    Data
Name: 1, dtype: object
```

```
df.iloc[0, 0] # 选取第一个元素
```

```
df = pd.DataFrame(np.random.rand(5, 5))
df
```

```
<tr style="text-align: right;">
  <th></th>
  <th>0</th>
  <th>1</th>
  <th>2</th>
  <th>3</th>
  <th>4</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>0.786709</td>
  <td>0.405902</td>
  <td>0.151383</td>
  <td>0.384778</td>
  <td>0.871664</td>
</tr>
<tr>
  <th>1</th>
  <td>0.491006</td>
  <td>0.774710</td>
  <td>0.388011</td>
  <td>0.758102</td>
  <td>0.762115</td>
</tr>
<tr>
  <th>2</th>
  <td>0.085647</td>
  <td>0.543243</td>
  <td>0.582565</td>
  <td>0.664243</td>
  <td>0.379896</td>
</tr>
<tr>
  <th>3</th>
  <td>0.806211</td>
  <td>0.794284</td>
  <td>0.968755</td>
  <td>0.883923</td>
  <td>0.354820</td>
</tr>
<tr>
  <th>4</th>
```

```
  <td>0.463902</td>
  <td>0.481756</td>
  <td>0.131181</td>
  <td>0.590878</td>
  <td>0.801769</td>
</tr>
```

```
df.iloc[0, 0]
```

```
0.78670886755075187
```

数据的清洗

```
df.columns = ['a', 'b'] # 对列名重新命名
```

```
df = pd.DataFrame({'A':np.array([1,np.nan,2,3,6,np.nan]),

'B':np.array([np.nan,4,np.nan,5,9,np.nan]),
                   'C':'foo'})
df
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>1.0</td>
  <td>NaN</td>
  <td>foo</td>
</tr>
<tr>
  <th>1</th>
  <td>NaN</td>
  <td>4.0</td>
  <td>foo</td>
</tr>
<tr>
  <th>2</th>
  <td>2.0</td>
  <td>NaN</td>
  <td>foo</td>
</tr>
<tr>
  <th>3</th>
  <td>3.0</td>
  <td>5.0</td>
  <td>foo</td>
</tr>
<tr>
  <th>4</th>
  <td>6.0</td>
  <td>9.0</td>
  <td>foo</td>
</tr>
<tr>
  <th>5</th>
  <td>NaN</td>
  <td>NaN</td>
```

```
  <td>foo</td>
</tr>
```

```
df.columns = ['a', 'b', 'c']
df
```

```
<tr style="text-align: right;">
  <th></th>
  <th>a</th>
  <th>b</th>
  <th>c</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>1.0</td>
  <td>NaN</td>
  <td>foo</td>
</tr>
<tr>
  <th>1</th>
  <td>NaN</td>
  <td>4.0</td>
  <td>foo</td>
</tr>
<tr>
  <th>2</th>
  <td>2.0</td>
  <td>NaN</td>
  <td>foo</td>
</tr>
<tr>
  <th>3</th>
  <td>3.0</td>
  <td>5.0</td>
  <td>foo</td>
</tr>
<tr>
  <th>4</th>
  <td>6.0</td>
  <td>9.0</td>
  <td>foo</td>
</tr>
<tr>
  <th>5</th>
  <td>NaN</td>
  <td>NaN</td>
```

```
  <td>foo</td>
</tr>
```

```
pd.isnull() # 检查数据中出现空值的情况，返回一个布尔型的列
pd.notnull() #相对应isnull 返回不是空值的情况
```

```
df = pd.DataFrame({'A':np.array([1,np.nan,2,3,6,np.nan]),

'B':np.array([np.nan,4,np.nan,5,9,np.nan]),
                   'C':'foo'})
df.isnull()
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>False</td>
  <td>True</td>
  <td>False</td>
</tr>
<tr>
  <th>1</th>
  <td>True</td>
  <td>False</td>
  <td>False</td>
</tr>
<tr>
  <th>2</th>
  <td>False</td>
  <td>True</td>
  <td>False</td>
</tr>
<tr>
  <th>3</th>
  <td>False</td>
  <td>False</td>
  <td>False</td>
</tr>
<tr>
  <th>4</th>
  <td>False</td>
  <td>False</td>
  <td>False</td>
</tr>
<tr>
  <th>5</th>
  <td>True</td>
  <td>True</td>
```

```
  <td>False</td>
</tr>
```

```
df.isnull().sum() # 对每一列的空值进行统计
```

```
A    2
B    3
C    0
dtype: int64
```

```
df.dropna(axis = 0, thresh=n) # 删除包含缺失值的行  axis = 1时删除列  # thresh = n移除空值超过(包括等于)n的行
```

```
df
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>1.0</td>
  <td>NaN</td>
  <td>foo</td>
</tr>
<tr>
  <th>1</th>
  <td>NaN</td>
  <td>4.0</td>
  <td>foo</td>
</tr>
<tr>
  <th>2</th>
  <td>2.0</td>
  <td>NaN</td>
  <td>foo</td>
</tr>
<tr>
  <th>3</th>
  <td>3.0</td>
  <td>5.0</td>
  <td>foo</td>
</tr>
<tr>
  <th>4</th>
  <td>6.0</td>
  <td>9.0</td>
  <td>foo</td>
</tr>
<tr>
  <th>5</th>
  <td>NaN</td>
  <td>NaN</td>
```

```
  <td>foo</td>
</tr>
```

```
df.dropna(axis = 0)
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
</tr>
```

```
<tr>
  <th>3</th>
  <td>3.0</td>
  <td>5.0</td>
  <td>foo</td>
</tr>
<tr>
  <th>4</th>
  <td>6.0</td>
  <td>9.0</td>
  <td>foo</td>
</tr>
```

```
df.dropna(axis = 1)
```

```
<tr style="text-align: right;">
  <th></th>
  <th>C</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>foo</td>
</tr>
<tr>
  <th>1</th>
  <td>foo</td>
</tr>
<tr>
  <th>2</th>
  <td>foo</td>
</tr>
<tr>
  <th>3</th>
  <td>foo</td>
</tr>
<tr>
  <th>4</th>
  <td>foo</td>
</tr>
<tr>
  <th>5</th>
  <td>foo</td>
</tr>
```

```
df.dropna(axis = 0, thresh = 2)
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>1.0</td>
  <td>NaN</td>
  <td>foo</td>
</tr>
<tr>
  <th>1</th>
  <td>NaN</td>
  <td>4.0</td>
  <td>foo</td>
</tr>
<tr>
  <th>2</th>
  <td>2.0</td>
  <td>NaN</td>
  <td>foo</td>
</tr>
<tr>
  <th>3</th>
  <td>3.0</td>
  <td>5.0</td>
  <td>foo</td>
</tr>
<tr>
  <th>4</th>
  <td>6.0</td>
  <td>9.0</td>
  <td>foo</td>
</tr>
```

```
df.fillna(df.mean()) # 用平均值来填充空值
```

```
s = pd.Series([1,3,5,np.nan,7,9,9])
s.fillna(s.mean())
```

```
0    1.000000
1    3.000000
2    5.000000
3    5.666667
4    7.000000
5    9.000000
6    9.000000
dtype: float64
```

```
s.astype(type) # 转换列的类型
```

```
s = pd.Series([1,3,5,np.nan,7,9,9])
s.fillna(s.mean()).astype(int)
```

```
0    1
1    3
2    5
3    5
4    7
5    9
6    9
dtype: int32
```

```
s.replace(1, 'one') # 将Series中的1替换为one
```

```
s = pd.Series([1,3,5,np.nan,7,9,9])
s.replace(1,'one')
```

```
0    one
1      3
2      5
3    NaN
4      7
5      9
6      9
dtype: object
```

```
s.replace([1,3],['one','three']) # 将数组(Series)中所有的1替换为'one', 所有的3替换为'three'
```

```
s = pd.Series([1,3,5,np.nan,7,9,9])
s.replace([1,3],['one','three'])
```

```
0      one
1    three
2        5
3      NaN
4        7
5        9
6        9
dtype: object
```

```
df.rename(columns=lambda x: x + 2) # 将全体列重命名
```

```
df = pd.DataFrame(np.random.rand(4,4))
df
```

```
<tr style="text-align: right;">
  <th></th>
  <th>0</th>
  <th>1</th>
  <th>2</th>
  <th>3</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>0.669102</td>
  <td>0.548996</td>
  <td>0.512802</td>
  <td>0.449220</td>
</tr>
<tr>
  <th>1</th>
  <td>0.108840</td>
  <td>0.974720</td>
  <td>0.665050</td>
  <td>0.271009</td>
</tr>
<tr>
  <th>2</th>
  <td>0.146804</td>
  <td>0.060744</td>
  <td>0.637770</td>
  <td>0.383380</td>
</tr>
<tr>
  <th>3</th>
  <td>0.108163</td>
  <td>0.893999</td>
  <td>0.216907</td>
  <td>0.730504</td>
</tr>
```

```
df.rename(columns=lambda x: x+ 2)
```

```
<tr style="text-align: right;">
  <th></th>
  <th>2</th>
  <th>3</th>
  <th>4</th>
  <th>5</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>0.669102</td>
  <td>0.548996</td>
  <td>0.512802</td>
  <td>0.449220</td>
</tr>
<tr>
  <th>1</th>
  <td>0.108840</td>
  <td>0.974720</td>
  <td>0.665050</td>
  <td>0.271009</td>
</tr>
<tr>
  <th>2</th>
  <td>0.146804</td>
  <td>0.060744</td>
  <td>0.637770</td>
  <td>0.383380</td>
</tr>
<tr>
  <th>3</th>
  <td>0.108163</td>
  <td>0.893999</td>
  <td>0.216907</td>
  <td>0.730504</td>
</tr>
```

```
df.rename(columns={'old_name': 'new_ name'}) # 将选择的列重命名
```

```
df = pd.DataFrame(np.random.rand(10,5),columns=list('ABCDE'))
df.rename(columns={'A':'New A', 'B':'New B'})
```

```
<tr style="text-align: right;">
  <th></th>
  <th>New A</th>
  <th>New B</th>
  <th>C</th>
  <th>D</th>
  <th>E</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>0.680941</td>
  <td>0.766561</td>
  <td>0.486226</td>
  <td>0.301537</td>
  <td>0.289970</td>
</tr>
<tr>
  <th>1</th>
  <td>0.917036</td>
  <td>0.100054</td>
  <td>0.464342</td>
  <td>0.181454</td>
  <td>0.933591</td>
</tr>
<tr>
  <th>2</th>
  <td>0.253549</td>
  <td>0.766181</td>
  <td>0.085607</td>
  <td>0.969627</td>
  <td>0.630674</td>
</tr>
<tr>
  <th>3</th>
  <td>0.377840</td>
  <td>0.909920</td>
  <td>0.214338</td>
  <td>0.011844</td>
  <td>0.392257</td>
</tr>
<tr>
  <th>4</th>
```

```
  <td>0.608564</td>
  <td>0.587614</td>
  <td>0.039867</td>
  <td>0.630492</td>
  <td>0.402101</td>
</tr>
<tr>
  <th>5</th>
  <td>0.361074</td>
  <td>0.937618</td>
  <td>0.787055</td>
  <td>0.054157</td>
  <td>0.300325</td>
</tr>
<tr>
  <th>6</th>
  <td>0.605472</td>
  <td>0.608429</td>
  <td>0.052152</td>
  <td>0.669343</td>
  <td>0.745648</td>
</tr>
<tr>
  <th>7</th>
  <td>0.660738</td>
  <td>0.158713</td>
  <td>0.352756</td>
  <td>0.028325</td>
  <td>0.195899</td>
</tr>
<tr>
  <th>8</th>
  <td>0.855695</td>

  <td>0.578177</td>
```

```
  <td>0.447043</td>
  <td>0.093923</td>
  <td>0.316234</td>
</tr>
<tr>
  <th>9</th>
  <td>0.337392</td>
  <td>0.645260</td>
  <td>0.140221</td>
  <td>0.616652</td>
  <td>0.727144</td>
</tr>
```

```
df.set_index('column_one') # 改变索引
```

```
df = pd.DataFrame(np.random.rand(10,5),columns=list('ABCDE'))
df.set_index('B')
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>C</th>
  <th>D</th>
  <th>E</th>
</tr>
<tr>
  <th>B</th>
  <th></th>
  <th></th>
  <th></th>
  <th></th>
</tr>
```

```
<tr>
  <th>0.824403</th>
  <td>0.013092</td>
  <td>0.055626</td>
  <td>0.895268</td>
  <td>0.837350</td>
</tr>
<tr>
  <th>0.310559</th>
  <td>0.689064</td>
  <td>0.541375</td>
  <td>0.275461</td>
  <td>0.808554</td>
</tr>
<tr>
  <th>0.533495</th>
  <td>0.072835</td>
  <td>0.563758</td>
  <td>0.695029</td>
  <td>0.524957</td>
</tr>
<tr>
  <th>0.541211</th>
  <td>0.820817</td>
  <td>0.591130</td>
  <td>0.268978</td>
  <td>0.546996</td>
</tr>
<tr>
  <th>0.286587</th>
  <td>0.936692</td>
  <td>0.343227</td>
  <td>0.383610</td>
  <td>0.811302</td>
```

```
</tr>
<tr>
  <th>0.878391</th>
  <td>0.938883</td>
  <td>0.636148</td>
  <td>0.776493</td>
  <td>0.025840</td>
</tr>
<tr>
  <th>0.156482</th>
  <td>0.918591</td>
  <td>0.030869</td>
  <td>0.235020</td>
  <td>0.096212</td>
</tr>
<tr>
  <th>0.857049</th>
  <td>0.613991</td>
  <td>0.810541</td>
  <td>0.917927</td>
  <td>0.921329</td>
</tr>
<tr>
  <th>0.713271</th>
  <td>0.949683</td>
  <td>0.811386</td>
  <td>0.920452</td>
  <td>0.213173</td>
</tr>
<tr>
  <th>0.686945</th>
  <td>0.522276</td>
  <td>0.881299</td>

  <td>0.936260</td>
```

```
  <td>0.030993</td>
</tr>
```

```
df.rename(index = lambda x: x+ 1) # 改变全体索引
```

```
df = pd.DataFrame(np.random.rand(10,5))
df.rename(index = lambda x: x+ 1)
```

```
<tr style="text-align: right;">
  <th></th>
  <th>0</th>
  <th>1</th>
  <th>2</th>
  <th>3</th>
  <th>4</th>
</tr>
```

```
<tr>
  <th>1</th>
  <td>0.382337</td>
  <td>0.185501</td>
  <td>0.457958</td>
  <td>0.009713</td>
  <td>0.628963</td>
</tr>
<tr>
  <th>2</th>
  <td>0.024175</td>
  <td>0.223274</td>
  <td>0.698171</td>
  <td>0.071715</td>
  <td>0.063272</td>
</tr>
<tr>
  <th>3</th>
  <td>0.913995</td>
  <td>0.713092</td>
  <td>0.269621</td>
  <td>0.575365</td>
  <td>0.805266</td>
</tr>
<tr>
  <th>4</th>
  <td>0.612708</td>
  <td>0.220953</td>
  <td>0.090858</td>
  <td>0.425472</td>
  <td>0.018996</td>
</tr>
<tr>
  <th>5</th>
```

```
  <td>0.045363</td>
  <td>0.153343</td>
  <td>0.730828</td>
  <td>0.323554</td>
  <td>0.364821</td>
</tr>
<tr>
  <th>6</th>
  <td>0.462096</td>
  <td>0.614072</td>
  <td>0.993130</td>
  <td>0.988894</td>
  <td>0.788648</td>
</tr>
<tr>
  <th>7</th>
  <td>0.887381</td>
  <td>0.802119</td>
  <td>0.191248</td>
  <td>0.980064</td>
  <td>0.628450</td>
</tr>
<tr>
  <th>8</th>
  <td>0.138270</td>
  <td>0.922870</td>
  <td>0.250827</td>
  <td>0.297472</td>
  <td>0.289915</td>
</tr>
<tr>
  <th>9</th>
  <td>0.258687</td>

  <td>0.807993</td>
```

```
  <td>0.930009</td>
  <td>0.811335</td>
  <td>0.609763</td>
</tr>
<tr>
  <th>10</th>
  <td>0.588020</td>
  <td>0.392127</td>
  <td>0.590799</td>
  <td>0.923180</td>
  <td>0.722801</td>
</tr>
```

数据的过滤(filter),排序(sort)和分组(groupby)

```
df[df[col] > 0.5] # 选取数据df中对应行的数值大于0.5的全部列  支持逻辑运算
```

```
df = pd.DataFrame(np.random.rand(10,5),columns=list('ABCDE'))
df
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
  <th>D</th>
  <th>E</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>0.051900</td>
  <td>0.548808</td>
  <td>0.744936</td>
  <td>0.848002</td>
  <td>0.299505</td>
</tr>
<tr>
  <th>1</th>
  <td>0.979053</td>
  <td>0.216078</td>
  <td>0.394286</td>
  <td>0.520654</td>
  <td>0.584194</td>
</tr>
<tr>
  <th>2</th>
  <td>0.185679</td>
  <td>0.453151</td>
  <td>0.839947</td>
  <td>0.730177</td>
  <td>0.392377</td>
</tr>
<tr>
  <th>3</th>
  <td>0.161267</td>
  <td>0.981833</td>
  <td>0.890858</td>
  <td>0.613972</td>
  <td>0.467528</td>
</tr>
<tr>
  <th>4</th>
```

```
  <td>0.091140</td>
  <td>0.369805</td>
  <td>0.600035</td>
  <td>0.372857</td>
  <td>0.897063</td>
</tr>
<tr>
  <th>5</th>
  <td>0.612195</td>
  <td>0.981150</td>
  <td>0.578304</td>
  <td>0.220064</td>
  <td>0.488182</td>
</tr>
<tr>
  <th>6</th>
  <td>0.898736</td>
  <td>0.626289</td>
  <td>0.788306</td>
  <td>0.747086</td>
  <td>0.386097</td>
</tr>
<tr>
  <th>7</th>
  <td>0.568531</td>
  <td>0.362593</td>
  <td>0.644950</td>
  <td>0.510410</td>
  <td>0.092556</td>
</tr>
<tr>
  <th>8</th>
  <td>0.872898</td>

  <td>0.771917</td>
```

```
  <td>0.853365</td>
  <td>0.227531</td>
  <td>0.045184</td>
</tr>
<tr>
  <th>9</th>
  <td>0.898296</td>
  <td>0.683850</td>
  <td>0.138142</td>
  <td>0.956854</td>
  <td>0.335476</td>
</tr>
```

```
df[df['A'] > 0.5]
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
  <th>D</th>
  <th>E</th>
</tr>
```

```
<tr>
  <th>1</th>
  <td>0.979053</td>
  <td>0.216078</td>
  <td>0.394286</td>
  <td>0.520654</td>
  <td>0.584194</td>
</tr>
<tr>
  <th>5</th>
  <td>0.612195</td>
  <td>0.981150</td>
  <td>0.578304</td>
  <td>0.220064</td>
  <td>0.488182</td>
</tr>
<tr>
  <th>6</th>
  <td>0.898736</td>
  <td>0.626289</td>
  <td>0.788306</td>
  <td>0.747086</td>
  <td>0.386097</td>
</tr>
<tr>
  <th>7</th>
  <td>0.568531</td>
  <td>0.362593</td>
  <td>0.644950</td>
  <td>0.510410</td>
  <td>0.092556</td>
</tr>
<tr>
  <th>8</th>
```

```
  <td>0.872898</td>
  <td>0.771917</td>
  <td>0.853365</td>
  <td>0.227531</td>
  <td>0.045184</td>
</tr>
<tr>
  <th>9</th>
  <td>0.898296</td>
  <td>0.683850</td>
  <td>0.138142</td>
  <td>0.956854</td>
  <td>0.335476</td>
</tr>
```

```
df[(df['A'] > 0.5) & (df['B'] < 0.7)]
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
  <th>D</th>
  <th>E</th>
</tr>
```

```
<tr>
  <th>1</th>
  <td>0.979053</td>
  <td>0.216078</td>
  <td>0.394286</td>
  <td>0.520654</td>
  <td>0.584194</td>
</tr>
<tr>
  <th>6</th>
  <td>0.898736</td>
  <td>0.626289</td>
  <td>0.788306</td>
  <td>0.747086</td>
  <td>0.386097</td>
</tr>
<tr>
  <th>7</th>
  <td>0.568531</td>
  <td>0.362593</td>
  <td>0.644950</td>
  <td>0.510410</td>
  <td>0.092556</td>
</tr>
<tr>
  <th>9</th>
  <td>0.898296</td>
  <td>0.683850</td>
  <td>0.138142</td>
  <td>0.956854</td>
  <td>0.335476</td>
</tr>
```

排序

```
df.sort_values(col, ascending=True) #按照列进行排序  # ascending: True 升序 False 降序
```

```
df = pd.DataFrame(np.random.rand(10,5),columns=list('ABCDE'))
df.sort_values('A', ascending=True)
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
  <th>D</th>
  <th>E</th>
</tr>
```

```
<tr>
  <th>3</th>
  <td>0.015834</td>
  <td>0.758417</td>
  <td>0.123415</td>
  <td>0.802403</td>
  <td>0.782450</td>
</tr>
<tr>
  <th>1</th>
  <td>0.068046</td>
  <td>0.373240</td>
  <td>0.414358</td>
  <td>0.105285</td>
  <td>0.759001</td>
</tr>
<tr>
  <th>7</th>
  <td>0.134238</td>
  <td>0.104416</td>
  <td>0.551595</td>
  <td>0.472277</td>
  <td>0.015997</td>
</tr>
<tr>
  <th>4</th>
  <td>0.236628</td>
  <td>0.391852</td>
  <td>0.390275</td>
  <td>0.904988</td>
  <td>0.650108</td>
</tr>
<tr>
  <th>8</th>
```

```
  <td>0.469382</td>
  <td>0.426359</td>
  <td>0.137109</td>
  <td>0.253183</td>
  <td>0.894667</td>
</tr>
<tr>
  <th>2</th>
  <td>0.508937</td>
  <td>0.443894</td>
  <td>0.147076</td>
  <td>0.149885</td>
  <td>0.434802</td>
</tr>
<tr>
  <th>0</th>
  <td>0.572640</td>
  <td>0.369032</td>
  <td>0.412343</td>
  <td>0.402019</td>
  <td>0.445365</td>
</tr>
<tr>
  <th>9</th>
  <td>0.663964</td>
  <td>0.533604</td>
  <td>0.217605</td>
  <td>0.602667</td>
  <td>0.637232</td>
</tr>
<tr>
  <th>6</th>
  <td>0.765109</td>

  <td>0.646277</td>
```

```
  <td>0.885381</td>
  <td>0.743307</td>
  <td>0.649711</td>
</tr>
<tr>
  <th>5</th>
  <td>0.962494</td>
  <td>0.650830</td>
  <td>0.754514</td>
  <td>0.578115</td>
  <td>0.659846</td>
</tr>
```

```
df.sort_values([col1,col2],ascending=[True,False]) # 按照数据框的列col1升序，col2降序的方式对数据框df做排序
```

```
df = pd.DataFrame(np.random.rand(10,5),columns=list('ABCDE'))
df.sort_values(['A','E'],ascending=[True,False])
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
  <th>D</th>
  <th>E</th>
</tr>
```

```
<tr>
  <th>1</th>
  <td>0.083526</td>
  <td>0.969267</td>
  <td>0.012550</td>
  <td>0.672851</td>
  <td>0.866501</td>
</tr>
<tr>
  <th>7</th>
  <td>0.107839</td>
  <td>0.383900</td>
  <td>0.982337</td>
  <td>0.390914</td>
  <td>0.308559</td>
</tr>
<tr>
  <th>0</th>
  <td>0.148742</td>
  <td>0.306888</td>
  <td>0.853949</td>
  <td>0.144650</td>
  <td>0.414474</td>
</tr>
<tr>
  <th>4</th>
  <td>0.190011</td>
  <td>0.794060</td>
  <td>0.514756</td>
  <td>0.272207</td>
  <td>0.086894</td>
</tr>
<tr>
  <th>6</th>
```

```
    <td>0.423982</td>
    <td>0.229873</td>
    <td>0.992318</td>
    <td>0.495706</td>
    <td>0.971735</td>
</tr>
<tr>
    <th>9</th>
    <td>0.532263</td>
    <td>0.106900</td>
    <td>0.528114</td>
    <td>0.456583</td>
    <td>0.362642</td>
</tr>
<tr>
    <th>2</th>
    <td>0.619678</td>
    <td>0.800373</td>
    <td>0.927766</td>
    <td>0.742667</td>
    <td>0.645809</td>
</tr>
<tr>
    <th>3</th>
    <td>0.815312</td>
    <td>0.920682</td>
    <td>0.833351</td>
    <td>0.266840</td>
    <td>0.132698</td>
</tr>
<tr>
    <th>5</th>
    <td>0.842293</td>

    <td>0.049499</td>
```

```
  <td>0.780198</td>
  <td>0.343752</td>
  <td>0.341800</td>
</tr>
<tr>
  <th>8</th>
  <td>0.932379</td>
  <td>0.398721</td>
  <td>0.080358</td>
  <td>0.200681</td>
  <td>0.549237</td>
</tr>
```

分组

```
df.groupby(col) # 按照某列对数据框df做分组 # 常与count进行连用，统计出各词的个数
```

```
df = pd.DataFrame({'A':np.array(['foo','foo','foo','foo','bar','bar']),
 'B':np.array(['one','one','two','two','three','three']),
'C':np.array(['small','medium','large','large','small','small']),
     'D':np.array([1,2,2,3,3,5])})
df
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
  <th>D</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>foo</td>
  <td>one</td>
  <td>small</td>
  <td>1</td>
</tr>
<tr>
  <th>1</th>
  <td>foo</td>
  <td>one</td>
  <td>medium</td>
  <td>2</td>
</tr>
<tr>
  <th>2</th>
  <td>foo</td>
  <td>two</td>
  <td>large</td>
  <td>2</td>
</tr>
<tr>
  <th>3</th>
  <td>foo</td>
  <td>two</td>
  <td>large</td>
  <td>3</td>
</tr>
<tr>
  <th>4</th>
  <td>bar</td>
  <td>three</td>
  <td>small</td>
  <td>3</td>
```

```
</tr>
<tr>
  <th>5</th>
  <td>bar</td>
  <td>three</td>
  <td>small</td>
  <td>5</td>
</tr>
```

```
df.groupby('B').count()
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>C</th>
  <th>D</th>
</tr>
<tr>
  <th>B</th>
  <th></th>
  <th></th>
  <th></th>
</tr>
```

```
<tr>
  <th>one</th>
  <td>2</td>
  <td>2</td>
  <td>2</td>
</tr>
<tr>
  <th>three</th>
  <td>2</td>
  <td>2</td>
  <td>2</td>
</tr>
<tr>
  <th>two</th>
  <td>2</td>
  <td>2</td>
  <td>2</td>
</tr>
```

```
df.groupby([col1,col2]) # 按照列col1和col2对数据框df做分组
```

```
df = pd.DataFrame({'A':np.array(['foo','foo','foo','foo','bar','bar']),
 'B':np.array(['one','one','two','two','three','three']),
'C':np.array(['small','medium','large','large','small','small']),
     'D':np.array([1,2,2,3,3,5])})
df.groupby(['A', 'B']).sum()
```

```
<tr style="text-align: right;">
  <th></th>
  <th></th>
  <th>D</th>
</tr>
<tr>
  <th>A</th>
  <th>B</th>
  <th></th>
</tr>
```

```
<tr>
  <th>bar</th>
  <th>three</th>
  <td>8</td>
</tr>
<tr>
  <th rowspan="2" valign="top">foo</th>
  <th>one</th>
  <td>3</td>
</tr>
<tr>
  <th>two</th>
  <td>5</td>
</tr>
```

```
df.groupby(col1)[col2].mean() # 按照列col1对数据框df做分组处理后，返回对应的col2的平均值
```

```
df = pd.DataFrame({'A':np.array(['foo','foo','foo','foo','bar','bar']),
 'B':np.array(['one','one','two','two','three','three']),
'C':np.array(['small','medium','large','large','small','small']),
     'D':np.array([1,2,2,3,3,5])})
df.groupby('A')['D'].mean()
```

```
A
bar    4
foo    2
Name: D, dtype: int32
```

```
df.pivot_table(index=col1,values=[col2,col3],aggfunc=mean) # 做透视表，索引为col1,针对的数值列为col2和col3，分组函数为平均值
```

```
df = pd.DataFrame({'A':np.array(['foo','foo','foo','foo','bar','bar']),

 'B':np.array(['one','one','two','two','three','three']),

'C':np.array(['small','medium','large','large','small','small']),
     'D':np.array([1,2,2,3,3,5])})
df.pivot_table(df, index=['A', 'B'], columns=['C'], aggfunc=np.sum)
```

```
<tr>
  <th></th>
  <th></th>
  <th colspan="3" halign="left">D</th>
</tr>
<tr>
  <th></th>
  <th>C</th>
  <th>large</th>
  <th>medium</th>
  <th>small</th>
</tr>
<tr>
  <th>A</th>
  <th>B</th>
  <th></th>
  <th></th>
  <th></th>
</tr>
```

```
<tr>
  <th>bar</th>
  <th>three</th>
  <td>NaN</td>
  <td>NaN</td>
  <td>8.0</td>
</tr>
<tr>
  <th rowspan="2" valign="top">foo</th>
  <th>one</th>
  <td>NaN</td>
  <td>2.0</td>
  <td>1.0</td>
</tr>
<tr>
  <th>two</th>
  <td>5.0</td>
  <td>NaN</td>
  <td>NaN</td>
</tr>
```

```
df =
pd.DataFrame({'A':np.array(['foo','foo','foo','foo','bar','b
ar']),

 'B':np.array(['one','one','two','two','three','three']),

'C':np.array(['small','medium','large','large','small','smal
l']),
     'D':np.array([1,2,2,3,3,5])})
df.groupby('A').agg(np.mean)
```

```
<tr style="text-align: right;">
  <th></th>
  <th>D</th>
</tr>
<tr>
  <th>A</th>
  <th></th>
</tr>
```

```
<tr>
  <th>bar</th>
  <td>4</td>
</tr>
<tr>
  <th>foo</th>
  <td>2</td>
</tr>
```

```
df.apply(np.mean, axis=0) # 对数据框df的每一列求平均值 axis: 0 对列名（横着的）进行处理  1对索引（竖着的）进行处理
```

```
df =
pd.DataFrame(np.random.rand(10,5),columns=list('ABCDE'))
df
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
  <th>D</th>
  <th>E</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>0.800902</td>
  <td>0.933676</td>
  <td>0.461338</td>
  <td>0.353398</td>
  <td>0.057885</td>
</tr>
<tr>
  <th>1</th>
  <td>0.988580</td>
  <td>0.681318</td>
  <td>0.533361</td>
  <td>0.486016</td>
  <td>0.220004</td>
</tr>
<tr>
  <th>2</th>
  <td>0.695034</td>
  <td>0.643920</td>
  <td>0.694040</td>
  <td>0.280063</td>
  <td>0.641867</td>
</tr>
<tr>
  <th>3</th>
  <td>0.925290</td>
  <td>0.084906</td>
  <td>0.120247</td>
  <td>0.880991</td>
  <td>0.399596</td>
</tr>
<tr>
  <th>4</th>
```

```
    <td>0.697742</td>
    <td>0.372860</td>
    <td>0.881456</td>
    <td>0.565627</td>
    <td>0.272549</td>
</tr>
<tr>
    <th>5</th>
    <td>0.614245</td>
    <td>0.658123</td>
    <td>0.797487</td>
    <td>0.609511</td>
    <td>0.544633</td>
</tr>
<tr>
    <th>6</th>
    <td>0.153517</td>
    <td>0.354870</td>
    <td>0.910838</td>
    <td>0.416895</td>
    <td>0.098821</td>
</tr>
<tr>
    <th>7</th>
    <td>0.088223</td>
    <td>0.501401</td>
    <td>0.702754</td>
    <td>0.334938</td>
    <td>0.182708</td>
</tr>
<tr>
    <th>8</th>
    <td>0.737348</td>

    <td>0.569340</td>
```

```
  <td>0.291342</td>
  <td>0.847058</td>
  <td>0.193331</td>
</tr>
<tr>
  <th>9</th>
  <td>0.083915</td>
  <td>0.396210</td>
  <td>0.589415</td>
  <td>0.806525</td>
  <td>0.598841</td>
</tr>
```

```
df.apply(np.mean, axis=0)
```

```
A    0.578480
B    0.519662
C    0.598228
D    0.558102
E    0.321024
dtype: float64
```

数据的连接(join)与组合(combine)

```
df1.append(df2) # 在数据框df2的末尾添加数据框df1，其中df1和df2的列数应该相等  列合并
```

```
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3']},
                   index=[0, 1, 2, 3])
df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],
                    'B': ['B4', 'B5', 'B6', 'B7'],
                    'C': ['C4', 'C5', 'C6', 'C7'],
                    'D': ['D4', 'D5', 'D6', 'D7']},
                   index=[4, 5, 6, 7])
df1.append(df2)
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
  <th>D</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>A0</td>
  <td>B0</td>
  <td>C0</td>
  <td>D0</td>
</tr>
<tr>
  <th>1</th>
  <td>A1</td>
  <td>B1</td>
  <td>C1</td>
  <td>D1</td>
</tr>
<tr>
  <th>2</th>
  <td>A2</td>
  <td>B2</td>
  <td>C2</td>
  <td>D2</td>
</tr>
<tr>
  <th>3</th>
  <td>A3</td>
  <td>B3</td>
  <td>C3</td>
  <td>D3</td>
</tr>
<tr>
  <th>4</th>
  <td>A4</td>
  <td>B4</td>
  <td>C4</td>
  <td>D4</td>
```

```
</tr>
<tr>
  <th>5</th>
  <td>A5</td>
  <td>B5</td>
  <td>C5</td>
  <td>D5</td>
</tr>
<tr>
  <th>6</th>
  <td>A6</td>
  <td>B6</td>
  <td>C6</td>
  <td>D6</td>
</tr>
<tr>
  <th>7</th>
  <td>A7</td>
  <td>B7</td>
  <td>C7</td>
  <td>D7</td>
</tr>
```

```
pd.concat([df1, df2], axis=1) # 在数据框df1的列最后添加数据框df2,其中df1和df2的行数应该相等  # 中括号可以换成圆括号  # axis: 0进行行合并  1进行列合并
```

```
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3']},
                   index=[0, 1, 2, 3])
df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],
                    'B': ['B4', 'B5', 'B6', 'B7'],
                    'C': ['C4', 'C5', 'C6', 'C7'],
                    'D': ['D4', 'D5', 'D6', 'D7']},
                   index=[4, 5, 6, 7])
pd.concat([df1, df2], axis=0)
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
  <th>D</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>A0</td>
  <td>B0</td>
  <td>C0</td>
  <td>D0</td>
</tr>
<tr>
  <th>1</th>
  <td>A1</td>
  <td>B1</td>
  <td>C1</td>
  <td>D1</td>
</tr>
<tr>
  <th>2</th>
  <td>A2</td>
  <td>B2</td>
  <td>C2</td>
  <td>D2</td>
</tr>
<tr>
  <th>3</th>
  <td>A3</td>
  <td>B3</td>
  <td>C3</td>
  <td>D3</td>
</tr>
<tr>
  <th>4</th>
  <td>A4</td>
  <td>B4</td>
  <td>C4</td>
  <td>D4</td>
```

```
</tr>
<tr>
  <th>5</th>
  <td>A5</td>
  <td>B5</td>
  <td>C5</td>
  <td>D5</td>
</tr>
<tr>
  <th>6</th>
  <td>A6</td>
  <td>B6</td>
  <td>C6</td>
  <td>D6</td>
</tr>
<tr>
  <th>7</th>
  <td>A7</td>
  <td>B7</td>
  <td>C7</td>
  <td>D7</td>
</tr>
```

```
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3']},
                   index=[0, 1, 2, 3])
df2 = pd.DataFrame({'E': ['A4', 'A5', 'A6', 'A7'],
                    'F': ['B4', 'B5', 'B6', 'B7'],
                    'G': ['C4', 'C5', 'C6', 'C7'],
                    'H': ['D4', 'D5', 'D6', 'D7']},
                   index=[0, 1, 2, 3])
pd.concat((df1, df2), axis=1)
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
  <th>D</th>
  <th>E</th>
  <th>F</th>
  <th>G</th>
  <th>H</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>A0</td>
  <td>B0</td>
  <td>C0</td>
  <td>D0</td>
  <td>A4</td>
  <td>B4</td>
  <td>C4</td>
  <td>D4</td>
</tr>
<tr>
  <th>1</th>
  <td>A1</td>
  <td>B1</td>
  <td>C1</td>
  <td>D1</td>
  <td>A5</td>
  <td>B5</td>
  <td>C5</td>
  <td>D5</td>
</tr>
<tr>
  <th>2</th>
  <td>A2</td>
  <td>B2</td>
  <td>C2</td>
  <td>D2</td>
  <td>A6</td>
  <td>B6</td>
  <td>C6</td>
  <td>D6</td>
</tr>

<tr>
```

```
  <th>3</th>
  <td>A3</td>
  <td>B3</td>
  <td>C3</td>
  <td>D3</td>
  <td>A7</td>
  <td>B7</td>
  <td>C7</td>
  <td>D7</td>
</tr>
```

```
df1.join(df2,on=col1,how='inner') # 对数据框df1和df2做内连接，其中连接的列为col1
```

```
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'key': ['K0', 'K1', 'K0', 'K1']})


df2 = pd.DataFrame({'C': ['C0', 'C1'],
                     'D': ['D0', 'D1']},
                    index=['K0', 'K1'])
df1.join(df2, on='key', how='inner')
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>key</th>
  <th>C</th>
  <th>D</th>
</tr>
```

```
<tr>
  <th>0</th>
  <td>A0</td>
  <td>B0</td>
  <td>K0</td>
  <td>C0</td>
  <td>D0</td>
</tr>
<tr>
  <th>2</th>
  <td>A2</td>
  <td>B2</td>
  <td>K0</td>
  <td>C0</td>
  <td>D0</td>
</tr>
<tr>
  <th>1</th>
  <td>A1</td>
  <td>B1</td>
  <td>K1</td>
  <td>C1</td>
  <td>D1</td>
</tr>
<tr>
  <th>3</th>
  <td>A3</td>
  <td>B3</td>
  <td>K1</td>
  <td>C1</td>
  <td>D1</td>
</tr>
```

数据的统计

```
df.mean() # 得到数据框df中每一列的平均值
```

```
df = pd.DataFrame(np.random.rand(10,5),columns=list('ABCDE'))
df.mean()
```

```
A    0.362158
B    0.432248
C    0.554478
D    0.331155
E    0.438283
dtype: float64
```

```
df.corr() # 得到数据框df中每一列与其他列的相关系数
```

```
df.corr()
```

```
<tr style="text-align: right;">
  <th></th>
  <th>A</th>
  <th>B</th>
  <th>C</th>
  <th>D</th>
  <th>E</th>
</tr>
```

```
<tr>
  <th>A</th>
  <td>1.000000</td>
  <td>-0.167715</td>
  <td>0.198216</td>
  <td>0.036939</td>
  <td>0.113714</td>
</tr>
<tr>
  <th>B</th>
  <td>-0.167715</td>
  <td>1.000000</td>
  <td>0.449789</td>
  <td>0.015883</td>
  <td>-0.236658</td>
</tr>
<tr>
  <th>C</th>
  <td>0.198216</td>
  <td>0.449789</td>
  <td>1.000000</td>
  <td>-0.296943</td>
  <td>0.386206</td>
</tr>
<tr>
  <th>D</th>
  <td>0.036939</td>
  <td>0.015883</td>
  <td>-0.296943</td>
  <td>1.000000</td>
  <td>-0.777327</td>
</tr>
<tr>
  <th>E</th>
```

```
  <td>0.113714</td>
  <td>-0.236658</td>
  <td>0.386206</td>
  <td>-0.777327</td>
  <td>1.000000</td>
</tr>
```

```
df.count() # 得到数据框df中每一列的非空值个数
```

```
df = pd.DataFrame(np.random.rand(10,5),columns=list('ABCDE'))
# df.loc[0][0] = np.nan
df.iloc[0, 0] = np.nan
df.count()
```

```
A     9
B    10
C    10
D    10
E    10
dtype: int64
```

```
df.max() # 得到数据框df中每一列的最大值
```

```
df = pd.DataFrame(np.random.rand(10,5),columns=list('ABCDE'))
df.max()
```

```
A    0.812708
B    0.886171
C    0.987035
D    0.977146
E    0.959625
dtype: float64
```

```
df.min() # 得到数据框df中每一列的最小值
```

```
df = pd.DataFrame(np.random.rand(10,5),columns=list('ABCDE'))
df.min()
```

```
A    0.128560
B    0.135905
C    0.167476
D    0.137062
E    0.050306
dtype: float64
```

```
df.median() # 得到数据框df中每一列的中位数
```

```
df = pd.DataFrame(np.random.rand(10,5),columns=list('ABCDE'))
df.median()
```

```
A    0.409373
B    0.597418
C    0.678203
D    0.705762
E    0.519713
dtype: float64
```

```
df.std() # 得到数据框df中每一列的标准差
```

```
df =
pd.DataFrame(np.random.rand(10,5),columns=list('ABCDE'))
df.std()
```

```
A    0.272847
B    0.254870
C    0.258956
D    0.301168
E    0.295753
dtype: float64
```