

pandas类库API帮助文档：

<https://pandas.pydata.org/pandas-docs/stable/reference/frame.html>

数据创建

df=pd.DataFrame(np.random.rand(10,5)) # 创建一个5列10行的由随机浮点数组成的数据框 DataFrame 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'}, index=[4, 5, 6, 7]) # 创建一个5列10行的指定值组成的数据框 DataFrame,索引从4~7  
pd.Series(my\_list) # 从一个可迭代的对象 my\_list 中创建一个数据组 df.index = pd.date\_range('2017/1/1',  
periods=df.shape[0]) # 添加一个日期索引 index

数据的查看与检查

df.head(n) # 查看数据框的前n行 df.tail(n) # 查看数据框的最后n行 df.shape # 查看数据框的行数与列数  
df.info() # 查看数据框 (DataFrame) 的索引、数据类型及内存信息 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 s = pd.Series([1,2,3,3,4,np.nan,5,5,5,6,7]) s.value\_counts(dropna=False) # 查  
询每个独特数据值出现次数统计 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 DataFrame.nunique(self, axis=0, dropna=True)

数据的选取

df[col] # 以数组 Series 的形式返回选取的列 df[[col1, col2]] # 以新的数据框(DataFrame)的形式返回选取的列  
s = pd.Series(np.array(['I','Love','Data'])) s.iloc[0] # 按照位置选取 s.loc[] # 按照索引选取 df.iloc[0,:] # 选取  
第一行 df.iloc[0,0] # 选取第一行的第一个元素

数据的清洗（更改）

df.columns = ['a','b','c','d','e'] # 重命名数据框的列名称 df.rename(columns=lambda x: x + 2) # 将全体列  
重命名 df.rename(columns={'old\_name': 'new\_name'}) # 将选择的列重命名 df.set\_index('column\_one') #  
改变索引见下 df.rename(index = lambda x: x+ 1) # 改变全体索引 pd.isnull() # 检查数据中空值出现的  
情况，并返回一个由布尔值(True,Fale)组成的列 pd.notnull() # 检查数据中非空值出现的情况，并返回一个由布尔  
值(True,False)组成的列 df.dropna() # 移除数据框 DataFrame 中包含空值的行 df.dropna(axis=1) # 移除数  
据框 DataFrame 中包含空值的列 df.fillna(x) # 将数据框 DataFrame 中的所有空值替换为 x  
s.fillna(s.mean()) # 将所有空值替换为平均值 s.astype(float) # 将数组(Series)的格式转化为浮点数  
s.replace(1,'one') # 将数组(Series)中的所有1替换为'one' 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'

set\_index

DataFrame可以通过set\_index方法，可以设置单索引和复合索引。  
DataFrame.set\_index(keys, drop=True, append=False, inplace=False, verify\_integrity=False)  
append添加新索引， drop为False， inplace为True时， 索引将会还原为列

In [307]: data Out[307]: a b c d 0 bar one z 1.0 1 bar two y 2.0 2 foo one x 3.0  
3 foo two w 4.0 In [308]: indexed1 = data.set\_index('c') In [309]: indexed1 Out[309]: a b d  
c z bar one 1.0 y bar two 2.0 x foo one 3.0 w foo two 4.0 In [310]:  
indexed2 = data.set\_index(['a', 'b']) In [311]: indexed2 Out[311]: c d a b bar  
one z 1.0 two y 2.0 foo one x 3.0 two w 4.0  
reset\_index  
data.reset\_index()

数据的过滤(filter),排序(sort)和分组(groupby)

df[df[col] > 0.5] # 选取数据框df中对应行的数值大于0.5的全部列 df[(df[col] > 0.5) & (df[col] < 0.7)] # 选取  
数据框df中对应行的数值大于0.5，并且小于0.7的全部列 df.sort\_values(col1) # 按照数据框的列col1升序  
(ascending)的方式对数据框df做排序 df.sort\_values(col2,ascending=False) # 按照数据框的列col2降序  
(descending)的方式对数据框df做排序 df.sort\_values([col1,col2],ascending=[True,False]) # 先按照数据框的  
列col1升序，再col2降序的方式对数据框df做排序 df.groupby(col) # 按照某列对数据框df做分组  
df.groupby('A').count() #####按分组进行=》统计 df.groupby([col1,col2]) # 按照列col1和col2对数据框df  
做分组 df.groupby(['B','C']).sum() #####同上分组，对结果的数值列求和 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.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

数据透视表：

```
[5]: import pandas as pd
import numpy as np
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])})

[6]: df

[6]:
```

	A	B	C	D
0	foo	one	small	1
1	foo	one	medium	2
2	foo	two	large	2
3	foo	two	large	3
4	bar	three	small	3
5	bar	three	small	5

```
[7]: df.pivot_table(df,index=['A','B'],
                    columns='C',aggfunc=np.sum)

[7]:
```

		D		
		C large	medium	small
A	B			
bar	three	NaN	NaN	8.0
foo	one	NaN	2.0	1.0
	two	5.0	NaN	NaN

df.groupby(col1).agg(np.mean) #对所有数值类型的列进行函数求值

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

[10]:
```

	A	B	C	D
0	foo	one	1	1
1	foo	one	2	2
2	foo	two	2	2
3	foo	two	3	3
4	bar	three	3	3
5	bar	three	5	5

```
[11]: df.groupby('A').agg(np.mean) #对所有数值类型的列进行函数求值

[11]:
```

	A	C	D
bar	4	4	
foo	2	2	

##=====所有列都是数值类型是前提： df.apply(np.mean) # 对数据框df的每一列求平均值

df.apply(np.max,axis=1) # 对数据框df的每一行求最大值

数据（不是DataFrame）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的列数必须相等 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的行数可以不相等

```
[32]: df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                        'B': ['B0', 'B1', 'B2', 'B3'],
                        'C': ['C0', 'C1', 'C2', 'C3'],
                        'D': ['D0', 'D1', 'D2', 'D3']})
df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7', 'A8'],
                        'B': ['B4', 'B5', 'B6', 'B7', 'B8'],
                        'C': ['C4', 'C5', 'C6', 'C7', 'C8'],
                        'D': ['D4', 'D5', 'D6', 'D7', 'D8'],
                        'E': ['E4', 'E5', 'E6', 'E7', 'E8']})

df1
#pd.concat([df1,df2],axis=1)

[32]:
```

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

```
[33]: df2

[33]:
```

	A	B	C	D	E
0	A4	B4	C4	D4	D4
1	A5	B5	C5	D5	D5
2	A6	B6	C6	D6	D6
3	A7	B7	C7	D7	D7
4	GG	GG	GG	GG	GG

```
[34]: pd.concat([df1,df2],axis=1)

[34]:
```

	A	B	C	D	A	B	C	D	E
0	A0	B0	C0	D0	A4	B4	C4	D4	D4
1	A1	B1	C1	D1	A5	B5	C5	D5	D5
2	A2	B2	C2	D2	A6	B6	C6	D6	D6
3	A3	B3	C3	D3	A7	B7	C7	D7	D7
4	NaN	NaN	NaN	NaN	GG	GG	GG	GG	GG

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 #如果要使用键列进  
行联接，则需要将key设置为df和other的索引。加入的DataFrame将具有键作为其索引。 >>>

df.set\_index('key').join(other.set\_index('key')) A B key K0 A0 B0 K1 A1 B1 K2 A2 B2 K3 A3 NaN K4 A4  
NaN K5 A5 NaN

数据的统计

df.describe() # 得到数据框df每一列的描述性统计 df.mean() # 得到数据框df中每一列的平均值 df.corr() # 得  
到数据框df中每一列与其他列的相关系数 df.count() # 得到数据框df中每一列的非空值个数 df.median() # 得  
到数据框df中每一列的中位数 df.std() # 得到数据框df中每一列的标准差 #其他。。。 略 df['diff'].sum()