Pandas Practice

Yihong_Daily (/u/614228d3dcbc) + 关注
2017.04.22 14:54* 字数 873 阅读 1005 评论 0 喜欢 12
(/u/614228d3dcbc)

pandas是一个方便易用的Python数据处理库，数据科学家们的利器之一哦。

本文简要介绍pandas的一些常用方法。

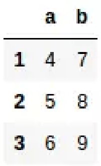
1 语法——创建DataFrames

```
import pandas as pd
```

Ex 1.1 由字典创建DataFrames

```
df = pd.DataFrame(
{"a":[4,5,6],    #每一列的数据
 "b":[7,8,9]},
index = [1,2,3]) #行索引
```

```
df
```

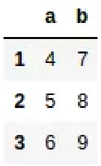


	a	b
1	4	7
2	5	8
3	6	9

Ex 1.2 由数组创建DataFrames

```
df = pd.DataFrame(
[[4,7],
[5,8],
[6,9]],
index = [1,2,3],
columns = ['a','b'])
```

```
df
```



	a	b
1	4	7
2	5	8
3	6	9

Ex 1.3 多重索引的DataFrames

```
df = pd.DataFrame(
{"a":[4,5,6],"b":[7,8,9]},
index = pd.MultiIndex.from_tuples(
[('d',1),('d',2),('e',2)]))
```

```
df
```

2 Reshaping Data ——改变数据集的布局

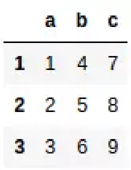
Ex 2.1 pd.melt

考虑一个 `DataFrame` , 某些列为ID变量 `id_vars` ， 其余列为测量的变量 `value_vars` ; 测量变量列被逆透视为行,最终除了ID列只剩下两列 `variable` 和 `value` 。这个函数起名为融化 `melt` ,名副其实——将许多列消融至两列，胖胖表变成瘦瘦表的视觉效果。

[使用场景]：适合用于将高维特征转化为 Event Log（ID，FeatX,x)

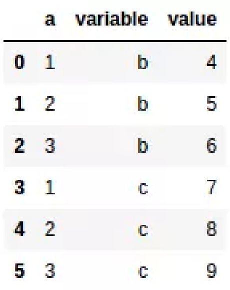
```
df = pd.DataFrame(
{"a":[1,2,3],"b":[4,5,6],"c":[7,8,9]},
index = [1,2,3])
```

```
df
```



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

```
pd.melt(df,id_vars=['a'])
```



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

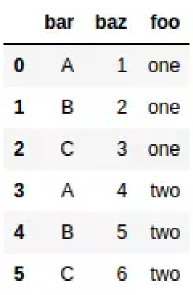
Ex 2.2 df.pivot

[使用场景]：与melt相反，pivot将行组织成紧凑的列，适合将Event log 转化为 高维特征矩阵。

考虑这么一个 `DataFrame`

```
df = pd.DataFrame({'foo': ['one','one','one','two','two','two'],
                            'bar': ['A', 'B', 'C', 'A', 'B', 'C'],
                            'baz': [1, 2, 3, 4, 5, 6]})
```

对foo列进行bar透视，即将foo这一列变成新的行索引，bar这一列变成列索引，这样我们就可以清晰地看见foo列和bar列的关系



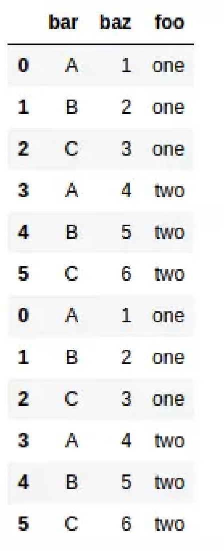
	bar	baz	foo
0	A	1	one
1	B	2	one
2	C	3	one
3	A	4	two
4	B	5	two
5	C	6	two

```
df.pivot(index = 'foo',columns = 'bar',values = 'baz')
```

Ex 2.3 pd.concat

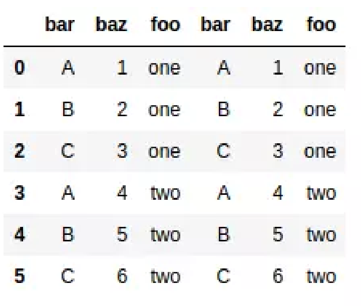
向DataFrame中添加行或列

```
pd.concat([df,df])#注意数组
```



	bar	baz	foo
0	A	1	one
1	B	2	one
2	C	3	one
3	A	4	two
4	B	5	two
5	C	6	two
0	A	1	one
1	B	2	one
2	C	3	one
3	A	4	two
4	B	5	two
5	C	6	two

```
pd.concat([df,df],axis = 1)
```



	bar	baz	foo	bar	baz	foo
0	A	1	one	A	1	one
1	B	2	one	B	2	one
2	C	3	one	C	3	one
3	A	4	two	A	4	two
4	B	5	two	B	5	two
5	C	6	two	C	6	two

Ex 2.4 df.unstack

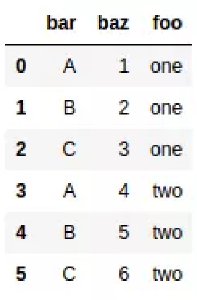
对于一个多索引的 `DataFrame` 来说，我们可以将某一个层级的索引“解”出来——变为列。

```
df.index.get_level_values(1)
new_df = df[['value']].unstack(level=-1).fillna(False) # level=-1选择最里面的索引层
new_df.columns = new_df.columns.get_level_values(1) # 取最里层列索引作为新的列
```

Ex 2.5 Else

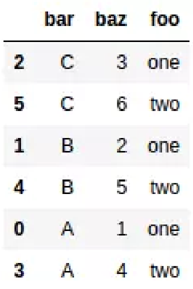
- 排序 df.sort_values
- 设置索引 df.reset_index 将索引置为行号，原索引变成列
- 调整索引 df.reindex(index=new_index),向原来的索引中加入或者删除项
- 重命名列 df.rename
- 丢弃列 df.drop(axis=1)

```
df
```



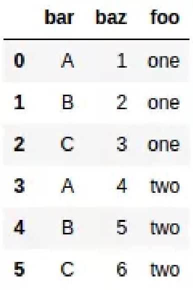
	bar	baz	foo
0	A	1	one
1	B	2	one
2	C	3	one
3	A	4	two
4	B	5	two
5	C	6	two

```
df.sort_values('bar',ascending=False)
```



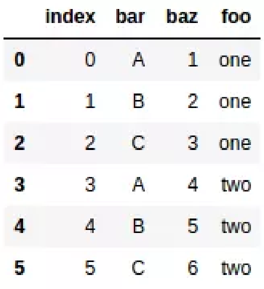
	bar	baz	foo
2	C	3	one
5	C	6	two
1	B	2	one
4	B	5	two
0	A	1	one
3	A	4	two

```
df.sort_index()
```



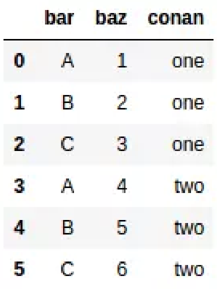
	bar	baz	foo
0	A	1	one
1	B	2	one
2	C	3	one
3	A	4	two
4	B	5	two
5	C	6	two

```
df.reset_index()
```



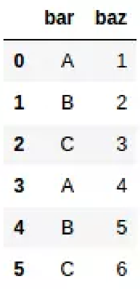
	index	bar	baz	foo
0	0	A	1	one
1	1	B	2	one
2	2	C	3	one
3	3	A	4	two
4	4	B	5	two
5	5	C	6	two

```
df.rename(columns = {"foo":"conan"})
```



	bar	baz	conan
0	A	1	one
1	B	2	one
2	C	3	one
3	A	4	two
4	B	5	two
5	C	6	two

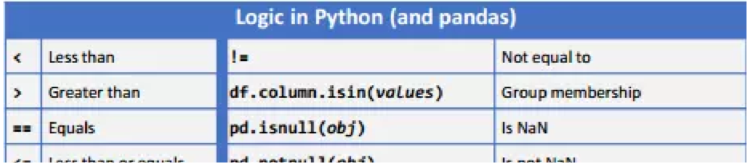
```
df.drop(['foo'],axis =1)
```



	bar	baz
0	A	1
1	B	2
2	C	3
3	A	4
4	B	5
5	C	6

3 Subset Observations ——行

- 头几行
- 尾几行
- 去重
- 逻辑准则
- 采样
- 按位置选择
- 按序选择（最大 最小）



Logic in Python (and pandas)

<	Less than	!=	Not equal to
>	Greater than	df.column.isin(*values*)	Group membership
==	Equals	pd.isnull(*obj*)	Is NaN
<=	Less than or equals	pd.notnull(*obj*)	Is not NaN

```
df.head(2)
```

```
   bar  baz  foo
0    A    1  one
1    B    2  one
```

```
df.tail(2)
```

```
   bar  baz  foo
4    B    5  two
5    C    6  two
```

```
df.drop_duplicates()
```

```
   bar  baz  foo
0    A    1  one
1    B    2  one
2    C    3  one
3    A    4  two
4    B    5  two
5    C    6  two
```

```
df.sample(frac = 0.2)
```

```
df.sample(2)
```

```
df.iloc[2:3]#与python数组类似
```

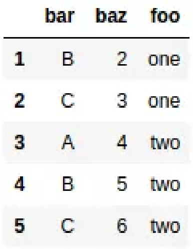
```
df.nlargest(2,'baz')
```

```
   bar  baz  foo
5    C    6  two
4    B    5  two
```

```
df.nsmallest(2,'baz')
```

```
   bar  baz  foo
0    A    1  one
1    B    2  one
```

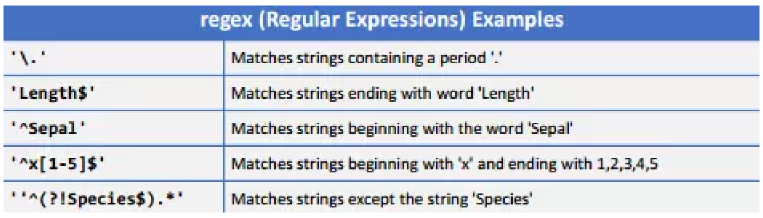
```
df[df.baz > 1]
```



	bar	baz	foo
1	B	2	one
2	C	3	one
3	A	4	two
4	B	5	two
5	C	6	two

4 Subset Variables——列

- 选择一列
- 选择多列
- 选择列名匹配给定正则表达式的列
- 按位置选列



regex (Regular Expressions) Examples	
'\.'	Matches strings containing a period '.'
'Length$'	Matches strings ending with word 'Length'
'^Sepal'	Matches strings beginning with the word 'Sepal'
'^x[1-5]$'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
''^(?!Species$).*'	Matches strings except the string 'Species'

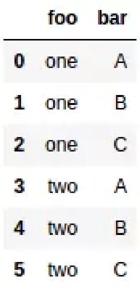
Ex 4.1 比较 iloc loc ix

- loc 只能处理index的label
- iloc 只能处理index的位置，因此只接受整数
- ix 试图像loc一样通过label处理index，失败时就如同iloc

```
df.foo
```

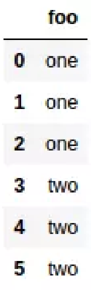
```
0    one
1    one
2    one
3    two
4    two
5    two
Name: foo, dtype: object
```

```
df[['foo','bar']]
```



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

```
df.filter(regex='o$')
```



	foo
0	one
1	one
2	one
3	two
4	two
5	two

```
df.loc[:,'bar':'foo']
```

	bar	baz	foo
0	A	1	one
1	B	2	one

```
df.iloc[:,[0,1]]
```

	bar	baz
0	A	1
1	B	2
2	C	3
3	A	4
4	B	5
5	C	6

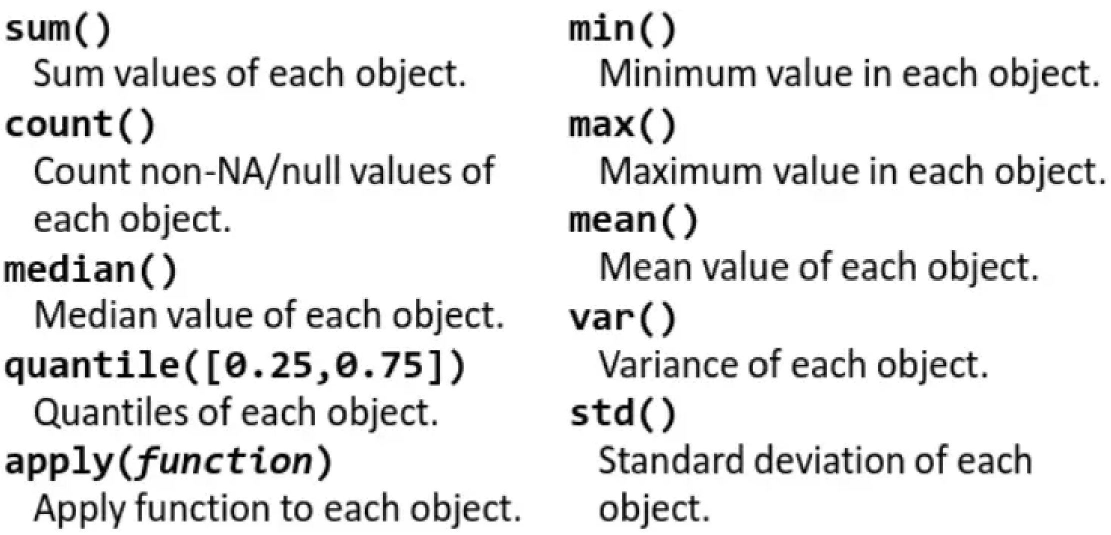
```
df.loc[df['baz'] > 1, ['foo','bar']]
```

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

5 Summarize Data

- 统计unique值的出现频率 df.column_name.value_counts()
- 每一列的描述性统计 df.decribe
- 常用summary functions
    - 处理各种pandas对象：DataFrame columns,Series,GroupBy,Expanding,Rolling
    - 每个group得到单独的一个值
    - 当应用到DataFrame时，返回Series

其中apply函数需要指定axis
* 0 or 'index': apply function to each column
* 1 or 'columns': apply function to each row



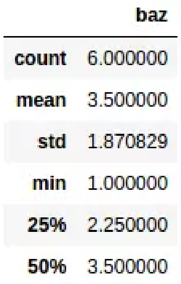
sum()
Sum values of each object.
count()
Count non-NA/null values of each object.
median()
Median value of each object.
quantile([0.25,0.75])
Quantiles of each object.
apply(*function*)
Apply function to each object.

min()
Minimum value in each object.
max()
Maximum value in each object.
mean()
Mean value of each object.
var()
Variance of each object.
std()
Standard deviation of each object.

```
df.foo.value_counts()
```

```
two    3
one    3
Name: foo, dtype: int64
```

```
df.describe()
```



	baz
count	6.000000
mean	3.500000
std	1.870829
min	1.000000
25%	2.250000
50%	3.500000

```
type(df.sum())
```

```
pandas.core.series.Series
```

```
df
```

image.png

```
def myfunc(row):
    #print(row)
    #print(type(row))
    print(row.foo)
    return"finished"

df.apply(myfunc,axis=1)#注意指定axis
```

```
one
one
one
two
two
two






0    finished
1    finished
2    finished
3    finished
4    finished
5    finished
dtype: object
```

6 缺失值

- dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
- fillna

7 创建新变量

- df.assign 给DataFrame添加新的一列，返回一个新的dataframe
- 也可直接df['new_col']
- pd.qcut(df.col,q,labels,precision)按分位数离散化数据
- df.clip 阈值化

```
df = df.assign(Area = lambda df:df.bar+df.foo)
```

```
df
```

```
df['Volumn'] = df.Area + df.foo
```

```
df
```

	bar	baz	foo	Area	Volumn
0	A	1	one	Aone	Aoneone
1	B	2	one	Bone	Boneone
2	C	3	one	Cone	Coneone
3	A	4	two	Atwo	Atwotwo
4	B	5	two	Btwo	Btwotwo
5	C	6	two	Ctwo	Ctwotwo

```
pd.cut(df.baz,3,retbins=True)#分成3类
```

```
(0    (0.995, 2.667]
 1    (0.995, 2.667]
 2    (2.667, 4.333]
 3    (2.667, 4.333]
 4        (4.333, 6]
 5        (4.333, 6]
 Name: baz, dtype: category
 Categories (3, object): [(0.995, 2.667] < (2.667, 4.333] < (4.333, 6]],
 array([ 0.995     ,  2.66666667,  4.33333333,  6.        ]))
```

```
df.baz.clip(lower=2,upper=3)
```

```
0    2
1    2
2    3
3    3
4    3
5    3
Name: baz, dtype: int64
```

8 Group Data

- df.groupby(by='col')按列分组
- df.groupby(level='ind')按某层级的索引分组

分组后返回的GroupBy Object

- size 求分组大小
- agg 使用函数对小组进行聚合
- 上述的summarize函数
- 应用到每个group,然后返回和原来的DataFrame一样大小的DataFrame
  - df.shift(1)
  - df.shift(-1)
  - rank(method='dense')返回排序后的rank值
  - rank(method='min')相等值取最小的rank值
  - ...
  - cumsum

- cummax
- cummin
- cumprod

```
df
```

	bar	baz	foo	Area	Volumn
0	A	1	one	Aone	Aoneone
1	B	2	one	Bone	Boneone
2	C	3	one	Cone	Coneone
3	A	4	two	Atwo	Atwotwo
4	B	5	two	Btwo	Btwotwo
5	C	6	two	Ctwo	Ctwotwo

```
df.groupby('foo').shift(1)#分组，每组都移动
```

	Area	Volumn	bar	baz
0	NaN	NaN	NaN	NaN
1	Aone	Aoneone	A	1.0
2	Bone	Boneone	B	2.0
3	NaN	NaN	NaN	NaN
4	Atwo	Atwotwo	A	4.0
5	Btwo	Btwotwo	B	5.0

```
df.groupby('foo').rank(method = 'dense')
```

	Area	Volumn	bar	baz
0	1.0	1.0	1.0	1.0
1	2.0	2.0	2.0	2.0
2	3.0	3.0	3.0	3.0
3	1.0	1.0	1.0	1.0
4	2.0	2.0	2.0	2.0
5	3.0	3.0	3.0	3.0

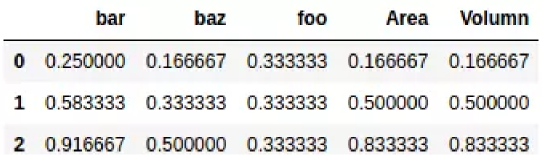
```
df.rank(method = 'dense')
```

	bar	baz	foo	Area	Volumn
0	1.0	1.0	1.0	1.0	1.0
1	2.0	2.0	1.0	3.0	3.0
2	3.0	3.0	1.0	5.0	5.0
3	1.0	4.0	2.0	2.0	2.0
4	2.0	5.0	2.0	4.0	4.0
5	3.0	6.0	2.0	6.0	6.0

```
df.rank(method = 'min')
```

	bar	baz	foo	Area	Volumn
0	1.0	1.0	1.0	1.0	1.0
1	3.0	2.0	1.0	3.0	3.0
2	5.0	3.0	1.0	5.0	5.0
3	1.0	4.0	4.0	2.0	2.0
4	3.0	5.0	4.0	4.0	4.0
5	5.0	6.0	4.0	6.0	6.0

```
df.rank(pct=True)
```



	bar	baz	foo	Area	Volumn
0	0.250000	0.166667	0.333333	0.166667	0.166667
1	0.583333	0.333333	0.333333	0.500000	0.500000
2	0.916667	0.500000	0.333333	0.833333	0.833333

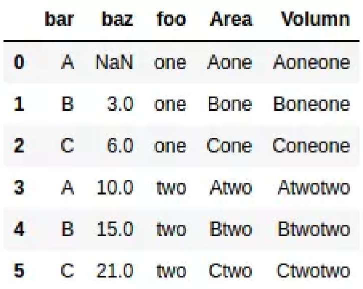
```
df.baz.rank(method = 'first')
```

```
0    1.0
1    2.0
2    3.0
3    4.0
4    5.0
5    6.0
Name: baz, dtype: float64
```

9 Windows

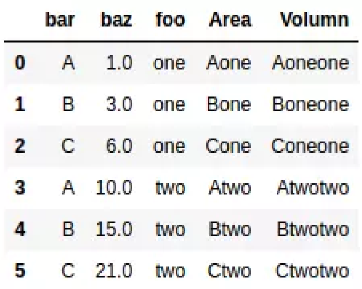
- expanding累积窗
- rolling滑动窗 moving average curve with variance as shades

```
df.expanding(2).sum()
```



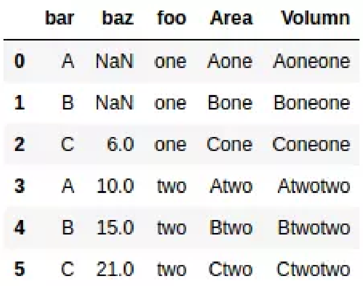
	bar	baz	foo	Area	Volumn
0	A	NaN	one	Aone	Aoneone
1	B	3.0	one	Bone	Boneone
2	C	6.0	one	Cone	Coneone
3	A	10.0	two	Atwo	Atwotwo
4	B	15.0	two	Btwo	Btwotwo
5	C	21.0	two	Ctwo	Ctwotwo

```
df.expanding(1).sum()
```



	bar	baz	foo	Area	Volumn
0	A	1.0	one	Aone	Aoneone
1	B	3.0	one	Bone	Boneone
2	C	6.0	one	Cone	Coneone
3	A	10.0	two	Atwo	Atwotwo
4	B	15.0	two	Btwo	Btwotwo
5	C	21.0	two	Ctwo	Ctwotwo

```
df.expanding(3).sum()
```



	bar	baz	foo	Area	Volumn
0	A	NaN	one	Aone	Aoneone
1	B	NaN	one	Bone	Boneone
2	C	6.0	one	Cone	Coneone
3	A	10.0	two	Atwo	Atwotwo
4	B	15.0	two	Btwo	Btwotwo
5	C	21.0	two	Ctwo	Ctwotwo

image.png

```
import numpy as np
df = pd.DataFrame(np.random.randn(1000,4),
                  index = pd.date_range('1/1/2000', periods=1000),
                  columns = ['A', 'B', 'C', 'D'])
```

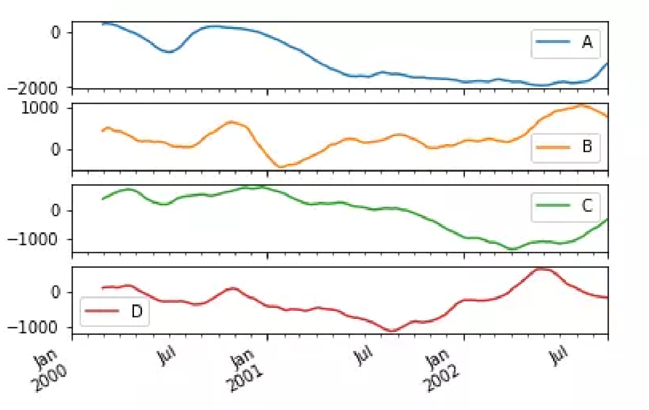
```
df = df.cumsum()
```

```
df.rolling(window=60).sum().plot(subplots=True)
```

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7fadbc4c5b70>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fadbc441e10>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fada8c15ef0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fada8bea048>], dtype=objec
```

```
import matplotlib.pyplot as plt
```

```
plt.show()
```

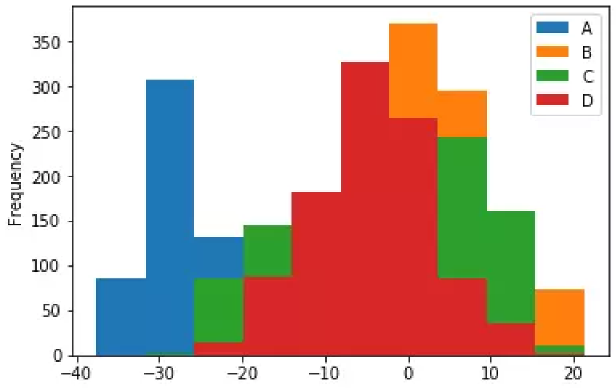


10 绘图

```
df.plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fada8a66518>
```

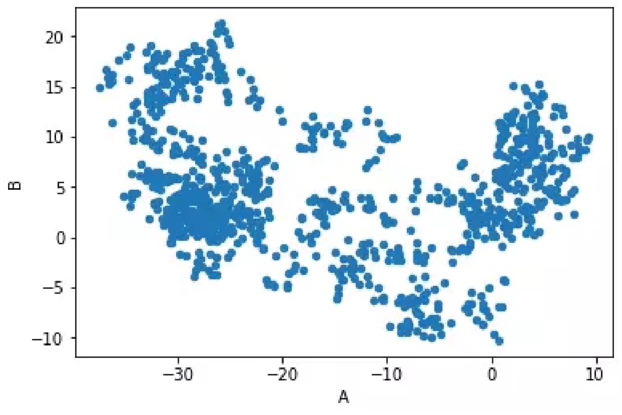
```
plt.show()
```



```
df.plot.scatter(x='A',y='B')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fada8905240>
```

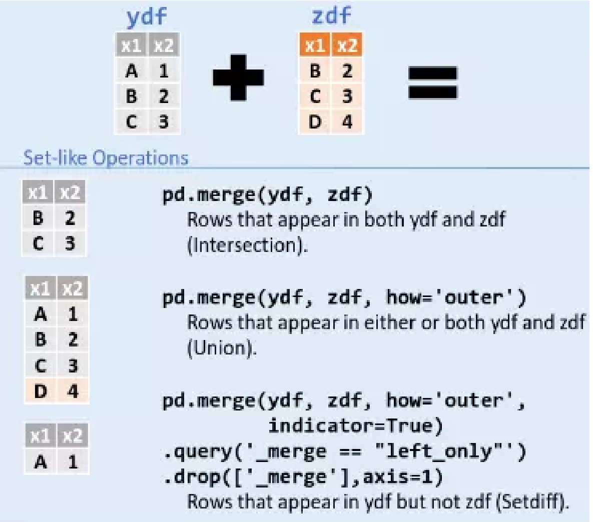
```
plt.show()
```



11 Combine Data Sets

- Join pd.merge(df1,df2,how='inner',on='key')
- 过滤 df1[~df1.x1.isin(df2.x1)]
- 集合操作

(/apps/redirect?utm_source=side banner-click)



ydf

x1	x2
A	1
B	2
C	3

+

zdf

x1	x2
B	2
C	3
D	4

=

Set-like Operations

x1	x2
B	2
C	3

```
pd.merge(ydf, zdf)
```

Rows that appear in both ydf and zdf (Intersection).

x1	x2
A	1
B	2
C	3
D	4

```
pd.merge(ydf, zdf, how='outer')
```

Rows that appear in either or both ydf and zdf (Union).

x1	x2
A	1

```
pd.merge(ydf, zdf, how='outer',
         indicator=True)
.query('_merge == "left_only"')
.drop(['_merge'],axis=1)
```

Rows that appear in ydf but not zdf (Setdiff).

Toolkits (/nb/11958194)

举报文章 © 著作权归作者所有

Yihong_Daily (/u/614228d3dcbc)

写了 21358 字，被 55 人关注，获得了 45 个喜欢

(/u/614228d3dcbc)

＋关注

Curiosity is the constant happiness. 轻微人来疯倾向，拖延症治愈中，属于健忘症高发人群。 业余跑者，...

喜欢 | 12

更多分享

登录 (/sign后发表评论source=desktop&utm_medium=not-signed-in-comment-fo

评论

智慧如你，不想发表一点想法 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-nocomments-text)咩~

被以下专题收入，发现更多相似内容

Python 运维 (/c/aa0b21cceb92?utm_source=desktop&utm_medium=notes-included-collection)

程序员 (/c/NEt52a?utm_source=desktop&utm_medium=notes-included-