

阅读目录

- 1 引言
- 2 行 (列) 选取: `df[]`
- 3 区域选取
 - 3.1 `df.loc[]`
 - 3.2 `df.iloc[]`
 - 3.3 `df.ix[]`
- 4 单元格选取
 - 4.1 `df.at[]`
 - 4.2 `df.iat[]`
- 5 拓展与总结

1 引言

Pandas是作为Python数据分析著名的工具包，提供了多种数据选取的方法，方便实用。本文主要介绍Pandas的几种数据选取的方法。

Pandas中，数据主要保存为Dataframe和Series是数据结构，这两种数据结构数据选取的方式基本一致，本文主要以Dataframe为例进行介绍。

在Dataframe中选取数据大抵包括3中情况：

1) 行（列）选取（单维度选取）：`df[]`。这种情况一次只能选取行或者列，即一次选取中，只能为行或者列设置筛选条件（只能为一个维度设置筛选条件）。

2) 区域选取（多维选取）：`df.loc[]`，`df.iloc[]`，`df.ix[]`。这种方式可以同时为多个维度设置筛选条件。

3) 单元格选取（点选取）：`df.at[]`，`df.iat[]`。准确定位一个单元格。

接下来，我们以下面的数据为例，分别通过实例介绍这三种情况。

```
>>> import pandas as pd

>>> import numpy as np

>>> data = {'name': ['Joe', 'Mike', 'Jack', 'Rose', 'David', 'Marry', 'Wansi', 'Sidy', 'Jason', 'Even'],

            'age': [25, 32, 18, np.nan, 15, 20, 41, np.nan, 37, 32],
```

```
'gender': [1, 0, 1, 1, 0, 1, 0, 0, 1, 0],
```

```
'isMarried': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'yes', 'no', 'no']}]
```

```
>>> labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

```
>>> df = pd.DataFrame(data, index=labels)
```

```
>>> df
```

	name	age	gender	isMarried
a	Joe	25.0	1	yes
b	Mike	32.0	0	yes
c	Jack	18.0	1	no
d	Rose	NaN	1	yes
e	David	15.0	0	no
f	Marry	20.0	1	no
g	Wansi	41.0	0	no

h	Sidy	NaN	0	yes
i	Jason	37.0	1	no
j	Even	32.0	0	no



[回到顶部](#)

2 行 (列) 选取: `df[]`

行 (列) 选取是在单一维度上进行数据的选取，即以行为单位进行选取或者以列为单位进行选取。Dataframe对象的行有索引 (index)，默认情况下是[0,1,2,]的整数序列，也可以自定义添加额外的索引，例如上面的labels，（为区分默认索引和自定义的索引，在本文中将默认索引称为整数索引，自定义索引称为标签索引）。Dataframe对象的每一列都有列名，可以通过列名实现对列的选取。

1) 选取行

选取行的方式包括三种：整数索引切片、标签索引切片和布尔数组。

a) 整数索引切片：前闭后开

- 选取第一行：



```
>>> df[0:1]
```

```
   name  age  gender  isMarried
a  Joe  25.0      1         yes
```



- 选取前两行:

```
>>> df[0:2]
```

```
   name  age  gender  isMarried
a  Joe  25.0      1         yes
b  Mike 32.0      0         yes
```



b) 标签索引切片: 前闭后闭

- 选取第一行:



```
>>> df['a']
```

```
   name  age  gender  isMarried
a  Joe  25.0      1         yes
```



- 选取前两行:

```
>>> df['a':'b']
```

```
   name  age  gender  isMarried
a  Joe  25.0      1         yes
b  Mike 32.0      0         yes
```



注意：整数索引切片是前闭后开，标签索引切片是前闭后闭，这点尤其要注意。

c) 布尔数组

- 选取前三行



```
>>> df[[True,True,True,False,False,False,False,False,False,False]]
```

	name	age	gender	isMarried
a	Joe	25.0	1	yes
b	Mike	32.0	0	yes
c	Jack	18.0	1	no



- 选取所有age大于30的行

```
>>> df[[each>30 for each in df['age']]]
```

	name	age	gender	isMarried
b	Mike	32.0	0	yes
g	Wansi	41.0	0	no
i	Jason	37.0	1	no
j	Even	32.0	0	no





通过布尔数组的方式，又可以衍生出下面的选取方式：

- 选取所有age大于30的行



```
>>> df[df['age']>30]
```

	name	age	gender	isMarried
b	Mike	32.0	0	yes
g	Wansi	41.0	0	no
i	Jason	37.0	1	no
j	Even	32.0	0	no



- 选取所有age大于30，且isMarried为no的行



```
>>> df[(df['age']>30) & (df['isMarried']=='no')]
```



```
name    age  gender isMarried
g  Wansi  41.0      0        no
i  Jason  37.0      1        no
j   Even  32.0      0        no
```



- 选取出所有age为20或32的行

```
>>> df[(df['age']==20) | (df['age']==32)]
```

```
name    age  gender isMarried
b   Mike  32.0      0        yes
f  Marry  20.0      1        no
j   Even  32.0      0        no
```



注意：像上面这种通过多个布尔条件判断的情况，多个条件最好（一定）用括号括起来，否则非常容易出错。

2) 列选取

列选取方式也有三种：标签索引、标签列表、Callable对象

a) 标签索引：选取单个列

- 选取name列所有数据

```
>>> df['name']
```

```
a      Joe
```

```
b      Mike
```

```
c      Jack
```

```
d      Rose
```

```
e     David
```

```
f     Marry
```

```
g     Wansi
```

```
h      Sidy
```



```
i      Jason
```

```
j      Even
```

```
Name: name, dtype: object
```



b) 标签列表：选取多个列

- 选取name和age两列数据

```
>>> df[['name', 'age']]
```

```
   name  age
```

```
a    Joe  25.0
```

```
b   Mike  32.0
```

```
c   Jack  18.0
```

```
d   Rose   NaN
```

```
e  David  15.0
```



```
f  Marry  20.0
```

```
g  Wansi  41.0
```

```
h   Sidy   NaN
```

```
i  Jason  37.0
```

```
j   Even  32.0
```



c) callable对象

- 选取第一列

```
>>> df[lambda df: df.columns[0]]
```

```
a      Joe
```

```
b      Mike
```

```
c      Jack
```

```
d      Rose
```



```
e    David
```

```
f    Marry
```

```
g    Wansi
```

```
h     Sidy
```

```
i    Jason
```

```
j     Even
```

```
Name: name, dtype: object
```



[回到顶部](#)

3 区域选取

区域选取可以从多个维度（行和列）对数据进行筛选，可以通过`df.loc[]`，`df.iloc[]`，`df.ix[]`三种方法实现。采用`df.loc[]`，`df.iloc[]`，`df.ix[]`这三种方法进行数据选取时，方括号内必须有两个参数，第一个参数是对行的筛选条件，第二个参数是对列的筛选条件，两个参数用逗号隔开。**`df.loc[]`，`df.iloc[]`，`df.ix[]`的区别如下：**

`df.loc[]`只能使用标签索引，不能使用整数索引，通过便签索引切边进行筛选时，前闭后闭。

`df.iloc[]`只能使用整数索引，不能使用标签索引，通过整数索引切边进行筛选时，前闭后开。；

df.ix[]既可以使用标签索引，也可以使用整数索引。

下面分别通过实例演示这三种方法。

3.1 df.loc[]

1) 对行进行选取

- 选取索引为 'a' 的行：

```
>>> df.loc['a', :]  
  
name      Joe  
  
age       25  
  
gender     1  
  
isMarried  yes  
  
Name: a, dtype: object
```



- 选取索引为 'a' 或 'b' 或 'c' 的行



```
>>> df.loc[['a','b','c'], :]
```

	name	age	gender	isMarried
a	Joe	25.0	1	yes
b	Mike	32.0	0	yes
c	Jack	18.0	1	no



- 选取从 'a' 到 'd' 的所有行 (包括 'd' 行)

```
>>> df.loc['a':'d', :]
```

	name	age	gender	isMarried
a	Joe	25.0	1	yes
b	Mike	32.0	0	yes
c	Jack	18.0	1	no
d	Rose	NaN	1	yes



- 用布尔数组选取前3行

```
>>> df.loc[[True,True,True,False,False,False], :]
```

	name	age	gender	isMarried
a	Joe	25.0	1	yes
b	Mike	32.0	0	yes
c	Jack	18.0	1	no

- 选取所有age大于30的行

```
>>> df.loc[df['age']>30,:]
```

	name	age	gender	isMarried
b	Mike	32.0	0	yes
g	Wansi	41.0	0	no

i	Jason	37.0	1	no
j	Even	32.0	0	no



也可以使用下面两方法:

```
>>> df.loc[df.loc[:, 'age']>30, :]
```

	name	age	gender	isMarried
b	Mike	32.0	0	yes
g	Wansi	41.0	0	no
i	Jason	37.0	1	no
j	Even	32.0	0	no

```
>>> df.loc[df.iloc[:, 1]>30, :]
```

	name	age	gender	isMarried
--	------	-----	--------	-----------



b	Mike	32.0	0	yes
g	Wansi	41.0	0	no
i	Jason	37.0	1	no
j	Even	32.0	0	no



- 用callable对象选取age大于30的所有行

```
>>> df.loc[lambda df:df['age'] > 30, :]
```

	name	age	gender	isMarried
b	Mike	32.0	0	yes
g	Wansi	41.0	0	no
i	Jason	37.0	1	no
j	Even	32.0	0	no



2) 对列选取

- 输出所有人的姓名（选取name列）

```
>>> df.loc[:, 'name']
```

```
a      Joe
```

```
b      Mike
```

```
c      Jack
```

```
d      Rose
```

```
e      David
```

```
f      Marry
```

```
g      Wansi
```

```
h      Sidy
```

```
i      Jason
```

```
j      Even
```



```
Name: name, dtype: object
```



- 输出所有人的姓名和年龄 (选取name和age列)

```
>>> df.loc[:, 'name':'age']
```



	name	age
a	Joe	25.0
b	Mike	32.0
c	Jack	18.0
d	Rose	NaN
e	David	15.0
f	Marry	20.0
g	Wansi	41.0

```
h   Sidy   NaN
```

```
i   Jason  37.0
```

```
j   Even   32.0
```



- 输出所有人的姓名、年龄、婚否 (选取name、age、isMarried列)

```
>>> df.loc[:, ['name', 'age', 'isMarried']]
```

	name	age	isMarried
a	Joe	25.0	yes
b	Mike	32.0	yes
c	Jack	18.0	no
d	Rose	NaN	yes
e	David	15.0	no
f	Marry	20.0	no



```
g  Wansi  41.0      no
h   Sidy   NaN      yes
i  Jason  37.0      no
j   Even  32.0      no
```



- 用布尔数组的方式选取前3列

```
>>> df.loc[:, [True,True,True,False]]
```

```
   name  age  gender
a   Joe  25.0       1
b  Mike  32.0       0
c  Jack  18.0       1
d  Rose   NaN       1
```



```
e  David  15.0      0
f  Marry  20.0      1
g  Wansi  41.0      0
h   Sidy   NaN      0
i  Jason  37.0      1
j   Even  32.0      0
```



3) 同时对行和列进行筛选

- 输出年龄大于30的人的姓名和年龄

```
>>> df.loc[df['age']>30,['name','age']]
```

```
   name  age
b  Mike  32.0
g  Wansi  41.0
```



```
i   Jason   37.0
```

```
j   Even    32.0
```



- 输出行名为 'Mike' 或 'Marry' 的姓名和年龄

```
>>> df.loc[(df['name']=='Mike') | (df['name']=='Marry'), ['name', 'age']]
```

```
name    age
```

```
b   Mike   32.0
```

```
f   Marry  20.0
```



3.2 df.iloc[]

1) 行选取

- 选取第2行

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




```
name      Mike

age       32

gender    0

isMarried yes

Name: b, dtype: object
```



- 选取前3行

```
>>> df.iloc[:3, :]

   name  age  gender  isMarried
a  Joe  25.0      1        yes
b  Mike 32.0      0        yes
c  Jack 18.0      1         no
```



- 选取第2行、第4行、第6行

```
>>> df.iloc[[1,3,5],:]
```

	name	age	gender	isMarried
b	Mike	32.0	0	yes
d	Rose	NaN	1	yes
f	Marry	20.0	1	no



- 通过布尔数组选取前3行

```
>>> df.iloc[[True,True,True,False,False,False], :]
```

	name	age	gender	isMarried
a	Joe	25.0	1	yes
b	Mike	32.0	0	yes



```
c   Jack   18.0         1       no
```



2) 列选取

- 选取第2列

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

```
a      25.0
```

```
b      32.0
```

```
c      18.0
```

```
d      NaN
```

```
e      15.0
```

```
f      20.0
```

```
g      41.0
```

```
h      NaN
```



```
i    37.0
```

```
j    32.0
```

```
Name: age, dtype: float64
```



- 选取前3列

```
>>> df.iloc[:, 0:3]
```

	name	age	gender
a	Joe	25.0	1
b	Mike	32.0	0
c	Jack	18.0	1
d	Rose	NaN	1
e	David	15.0	0



```
f  Marry  20.0      1
g  Wansi  41.0      0
h   Sidy   NaN      0
i  Jason  37.0      1
j   Even  32.0      0
```

1 选取第1列、第3列、第4列



- 选取第1列、第3列和第4列

```
>>> df.iloc[:, [0,2,3]]

   name  gender  isMarried
a   Joe      1         yes
b  Mike      0         yes
c  Jack      1         no
```



d	Rose	1	yes
e	David	0	no
f	Marry	1	no
g	Wansi	0	no
h	Sidy	0	yes
i	Jason	1	no
j	Even	0	no



- 通过布尔数组选取前3列

```
>>> df.iloc[:, [True, True, True, False]]
```

	name	age	gender
a	Joe	25.0	1
b	Mike	32.0	0
c	Jack	18.0	1
d	Rose	NaN	1



```
e David 15.0 0
f Marry 20.0 1
g Wansi 41.0 0
h Sidy NaN 0
i Jason 37.0 1
j Even 32.0 0
```



3) 同时选取行和列

- 选取第2行的第1列、第3列、第4列

```
>>> df.iloc[1, [0,2,3]]
```

```
name      Mike
```

```
gender      0
```

```
isMarried  yes
```

```
Name: b, dtype: object
```



- 选取前3行的前3列

```
>>> df.iloc[:3, :3]
```

	name	age	gender
a	Joe	25.0	1
b	Mike	32.0	0
c	Jack	18.0	1

3.3 df.ix[]

df.ix[]既可以通过整数索引进行数据选取，也可以通过标签索引进行数据选取，换句话说，df.ix[]是df.loc[]和df.iloc[]的功能集合，且在同义词选取中，可以同时使用整数索引和标签索引。

- 选取第3行的name数据

```
>>> df.ix[2, 'name']
```

```
'Jack'
```

- 选取a行、c行的第1列，第2列和第4列数据


```
>>> df.ix[['a','c'], [0,1,3]]
```

	name	age	isMarried
a	Joe	25.0	yes
c	Jack	18.0	no



- 选取所有未婚者的姓名和年龄

```
>>> df.ix[df['isMarried']=='no',['name','age']]
```

	name	age
c	Jack	18.0
e	David	15.0
f	Marry	20.0
g	Wansi	41.0
i	Jason	37.0



```
j    Even    32.0
```



[回到顶部](#)

4 单元格选取

单元格选取包括df.at[]和df.iat[]两种方法。df.at[]和df.iat[]使用时必须输入两个参数，即行索引和列索引，其中df.at[]只能使用标签索引，df.iat[]只能使用整数索引。df.at[]和df.iat[]选取的都是单个单元格（单行单列），所以返回值都为基本数据类型。

4.1 df.at[]

- 选取b行的name列

```
>>> df.at['b', 'name']
```

```
'Mike'
```

4.2 df.iat[]

- 选取第2行第1列

```
>>> df.iat[1,0]
```

```
'Mike'
```

[回到顶部](#)

5 拓展与总结

1) 选取某一整行（多个整行）或某一整列（多个整列）数据时，可以用df[]、df.loc[]、df.iloc[]，此时df[]的方法书写要简单一些。

2) 进行区域选取时，如果只能用标签索引，则使用df.loc[]或df.ix[]，如果只能用整数索引，则用df.iloc[]或df.ix[]。不过我看到有资料说，不建议使用df.ix[]，因为df.loc[]和df.iloc[]更精确（有吗？我没理解精确在哪，望告知）。

3) 如果选取单元格，则df.at[]、df.iat[]、df.loc[]、df.iloc[]都可以，不过要注意参数。

4) 选取数据时，返回值存在以下情况：

- 如果返回值包括单行多列或多行单列时，返回值为Series对象；
- 如果返回值包括多行多列时，返回值为DataFrame对象；
- 如果返回值仅为一个单元格（单行单列）时，返回值为基本数据类型，例如str，int等。

5) df[]的方式只能选取行和列数据，不能精确到单元格，所以df[]的返回值一定DataFrame或Series对象。

6) 当使用DataFrame的默认索引（整数索引）时，整数索引即为标签索引。例如，使用上面的data实例化一个DataFrame对象：

```
>>> df2 = pd.DataFrame(data)
```

```
>>> df2.loc[1, 'name']
```

```
'Mike'
```

```
>>> df2.iloc[1,0]
```

```
'Mike'
```



分类: [python](#) , [数据分析](#)

标签: [python](#) , [数据分析](#)

posted @ 2019-03-06 19:51 奥辰 阅读(133506) 评论(7) 编辑 收藏 举报