

# groupby后对层级索引levels的处理

23 不论如何未来很美好 阅读数 4449

博主原创文章，转载请附上此地址。 [https://blog.csdn.net/qq\\_36523839/article/details/80468063](https://blog.csdn.net/qq_36523839/article/details/80468063)  
刚开始学习pandas的时候没有太多的操作关于groupby，仅仅是简单的count、sum、size等等，没有更深入的利用groupby后的数据进行处理。近来数据处理的时间，所以这里记录以及复习一下：（以下皆是个人实践后的理解）

举下面的问题：一张数据表中有三列（动物物种、物种品种、品种价格），选出每个物种从大到小品种的前两种，最后只需要品种和价格这两列。

```
{ 'df1': ['cat', 'cat', 'dog', 'cat', 'dog', 'dog'], 'df2': [2,3,4,1,3,1], 'df3': [100,200,100,300,200,200]}
```

[https://blog.csdn.net/qq\\_36523839](https://blog.csdn.net/qq_36523839)

以上这张表是我们后面需要处理的数据表（物种和品种）

创建pandas类型时可以预先定义；使用groupby后也会生成（索引）

（根据df1物种分类，再根据df2品种排序后 如下图）

```
df3
0    100
1    200
2    100
3    300
4    200
5    200
```

groupby分类后的cat、dog便是level，以及后面的一列原始位置索引也是level

那么，我们该如何对它进行处理，如何完成上面的实例呢？（可能你拿到这样的层级数据，不会操作，不知道如何提取其中的信息）

import pandas as pd，以及创建原始数据：

```
s = pd.DataFrame({'df1': ['cat', 'cat', 'dog', 'cat', 'dog', 'dog'], 'df2': [2,3,4,1,3,1], 'df3': [100,200,100,300,200,200]})
```

df1

分类，并且使用apply调用sort\_df2函数对品种进行排序：

```
(data):
    = data.sort_values(by='df2',ascending=False)      #df2: 品种列 ascending: 排序方式
    rn data
    groupby(df['df1']).apply(sort_df2)      #groupby 以及apply 的结合使用
```

二张图

x) #看看groupby后的行索引什么样

```
[[ 'cat', 'dog'], [0, 1, 2, 3, 4, 5]],
[[0, 0, 0, 1, 1, 1], [1, 0, 3, 2, 4, 5]],
['df1', None]]
```

与层级标签（这里两列）， labels标签（分类，位置）

一层级标签的第一列（也就是cat、dog）

```
index.levels[0] #取出第一级标签:
```

完成从中选出（物种前两个品种以及它的价格）， 很简单的操作：

```
els:
group = group.loc[i]          #选出i 标签物种的所有品种
group = mid_group.iloc[:2,:]  #我们只取排序后的品种的前两种（要注意这里使用iloc，它与loc的区别）
cnt = len(mid_group)         #为了防止循环长度错误，所以我们还是需要计算长度，因为如果真正数据不足2条还是不报错
for j in range(cnt):         #现在在每个物种cat、dog 中操作
    value = mid_group.iloc[j,:] #我们选出该物种的第j 条所有信息df1、df2、df3
    value_pro = (value['df2'],value['df3']) #然后只取df2、df3，将它们放到元组中
    values.append(value_pro)
```

我们看看结果：

#此时在列表中保存了上面提取的元组信息，我们可以使用pandas再次转换它们为DataFrame，也可以做其它操作

```
), (4, 100), (3, 200)]

. csdn. net/qq_36523839
```

形象，但是还是有逻辑欠缺的地方，不过不重要，看懂了上面的例子，基本上就能了解和处理层级数据了。当然这里的数据简单，只是为了更好的理解，真正的与复杂的层级结构，这时需要能够更灵活的处理，如果你有更好的理解和建议，可以回复。

-----更新（增加对两层索引的操作）-----


一列df4表示动物的大小特征


```
e({'df1':['cat','cat','dog','cat','dog','dog'],'df2':[2,3,4,1,3,1],'df3':[100,200,100,300,200,200],'df4':['大','中','小','巨大','小'],'
```


```
df4
大
中
小
巨大
小
中
```


列来分类，再对两层的层级索引操作：


```
oupby(['df1','df4']).size()
```


1

5



















五个特征的动物数量，现在来取得其中的值：

```
up.index)
.loc[['cat','df4']]
```

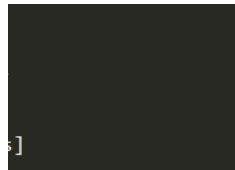
息，从中我们可以看到两层索引对应的levels有两中，然后我们根据loc测试选出cat类的df4这一列（也可以填大、中、巨大选出一

```
df1[['cat', 'dog'], ['中', '大', '小', '巨大']],
df2[['0', 0, 0, 1, 1], [0, 1, 3, 0, 2]],
df3[['df1', 'df4']]

https://blog.csdn.net/qq_36523839
```

的信息，当然也可以选出dog种类，那么如何得出(cat,巨大，1)这样的——对应的数据呢？

```
f_group.index.levels[0]          #获得第一层的分类cat、dog
for i in range(len(df1_name)):    #循环遍历第一层
    df_group.loc[[df1_name[i], 'df4']] #这里是选出第一层的所有信息
    df_level_ch = pd.DataFrame(df_level) #由于上面得到是Series我们需要将它转换为DataFrame才能更好的操作
    for j in range(len(df_level_ch)): #开始对第二层进行遍历
        df_level_ch.ix[j].name #由于是DataFrame所以可以取每一行的name值('cat','大')
        df_level_ch.values[j][0] #获取对应数量，由于是嵌套列表，所以我们逐层获取
    print(a,b)
```



还是很简单的。这只是其中的一个例子，如果遇到需要其他的操作，可以根据这个例子来随机变换。

法，但是个人觉得数据量过大，就不是很好，暂时没有更好的方法，如果那位朋友有其他操作，可以分享一下。

直销 非洲柚木 精品家具柚木 高档柚木原木 防腐木柚木 柚木销售

“group.index” 会报错，同理无法访问 “group.index.levels”

ValueError: Cannot access attribute 'index' of 'DataFrameGroupBy' objects, try using the 'apply' method

查看回复(4)