82.11 pandas的groupby函数

准备

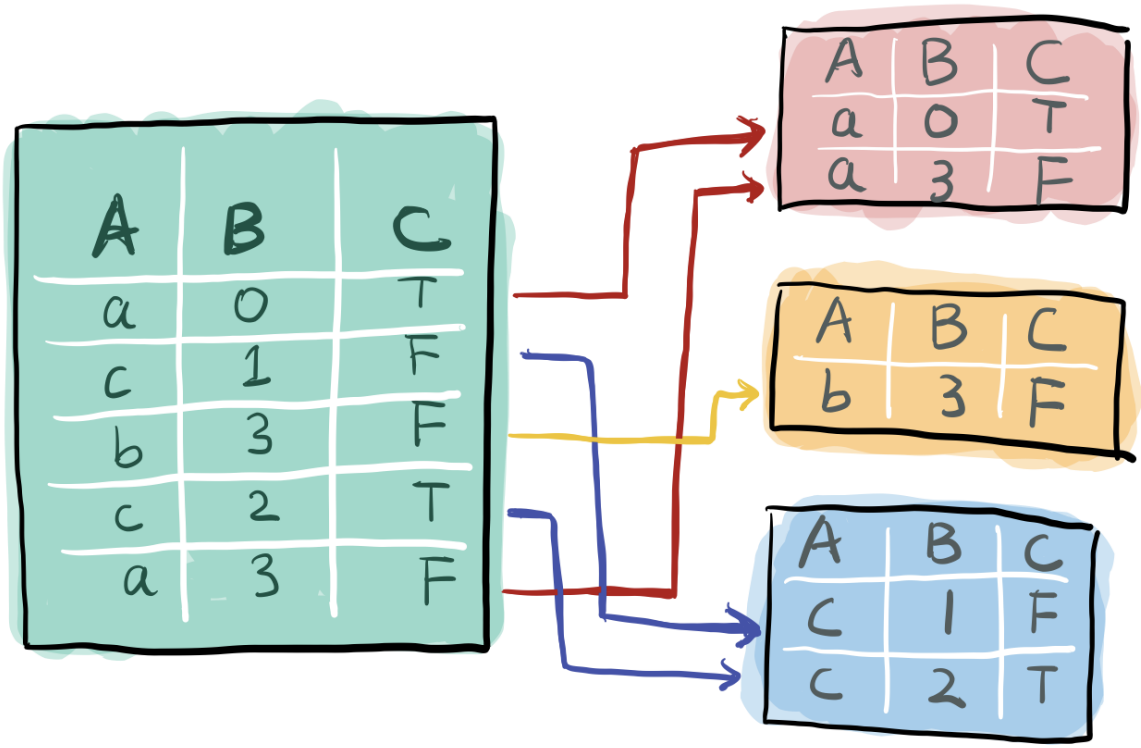
读入一个数据集.

```
# 读入一个数据集，我使用了美国警方击毙数据集.
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
plt.style.use('ggplot')
path = 'https://raw.githubusercontent.com/HoijanLai/dataset/master/Polic
data = pd.read_csv(path, encoding ='latin1')
data.sample(3)
```

	name	date	race	age	signs_of_mental_illness	flee
683	Tyrone Holman	09/09/15	B	37.0	True	Not fleein
1941	Michael Alan Altice	25/12/16	W	61.0	True	Not fleein
652	Manuel Soriano	27/08/15	H	29.0	False	Not fleein

什么是group by

groupby就是按xx分组. 比如, 将一个数据集按A进行分组, 效果是这样



我们尝试使用groupby来尝试实现这样的功能, 不过我们不用A列, 我们将用我们数据集里面的"种族"尝试分组:

```
data.groupby('race')
```

<pandas.core.groupby.DataFrameGroupBy object at 0x104fa2208>

这里我们得到了一个叫`DataFrameGroupBy`的东西, 虽然 pandas 不让我们直接看它长啥样, 但是你将它想象成上面那幅分组后的图(我手绘的)是完全没有问题的.

> *这篇稿主要介绍如何鼓捣这个`DataFrameGroupBy`, 这个`DataFrameGroupBy`主要的功能能是允许你在**不额外写循环的情况下, 快速对每一组数据进行操作***

基本操作

最基本的就是组内计数, 求和, 求均值, 求方差等
比如, 要求不同种族内， 被击毙人员年龄的均值:

```
    data.groupby('race')['age'].mean()
```

race
A 36.605263 # A种族的平均年龄
B 31.635468
H 32.995157
N 30.451613
O 33.071429
W 40.046980
Name: age, dtype: float64

上面我们求得了各个种族中被击毙的人员的平均年龄, 得到的是一个Series, 每一行对应了每一组的`mean`, 除此之外你还可以换成`std`, `median`, `min`, `max`这些基本的统计数据

上面age是连续属性, 我们还可以操作离散属性, 比如对不同取值的计数:
`.value_counts()`
以下尝试求不同种族内, 是否有精神异常迹象的分别有多少人

```
    data.groupby('race')['signs_of_mental_illness'].value_counts()
```

```
race  signs_of_mental_illness

A     False                       29
      True                        10
B     False                      523
```

```
      True                        95
H     False                      338
      True                        85
N     False                       23
      True                         8
O     False                       21
      True                         7
W     False                      819
      True                       382

Name: signs_of_mental_illness, dtype: int64
```

注: 这时, 组内操作的结果不是单个值, 是一个序列, 我们可以用`.unstack()`将它展开

```
data.groupby('race')['signs_of_mental_illness'].value_counts().unstack()
```

signs_of_mental_illness	False	True
race		
A	29	10
B	523	95
H	338	85
N	23	8
O	21	7
W	819	382

> *方法总结*
>
> - 首先通过groupby得到`DataFrameGroupBy`对象, 比如`data.groupby('race')`

> - 然后选择需要研究的列, 比如`['age']`, 这样我们就得到了一个`SeriesGroupby`, 它代表每一个组都有一个`Series`
> - 对`SeriesGroupby`进行操作, 比如`.mean()`, 相当于对每个组的`Series`求均值
>
> *注: 如果不选列, 那么第三步的操作会遍历所有列, pandas会对能成功操作的列进行操作, 最后返回的一个由操作成功的列组成的`DataFrame`*

更多基本操作

选择一个组
不细讲啦, 我自己觉得跟筛选数据差不多

可视化

> 这是我非常喜欢Groupby的一个地方, 它能够帮你很轻松地分组画图, 免去手写每个组的遍历的烦恼, 还能为你每个组分好颜色.

场景一: 不同种族中, 逃逸方式分别是如何分布的?

(属性A的不同分组中, 离散属性B的情况是怎么样的)

- 一种传统做法是:

1. 遍历每个组
2. 然后筛选不同组的数据
3. 逐个子集画条形图 (或者其他表示)

```
races = np.sort(data['race'].dropna().unique())
fig, axes = plt.subplots(1, len(races), figsize=(24, 4), sharey=True)
for ax, race in zip(axes, races):
    data[data['race']==race]['flee'].value_counts().sort_index().plot(ki
```

还不错, 但是使用Groupby能让我们直接免去循环, 而且不需要烦人的筛选, 一行就完美搞定

```
data.groupby('race')['flee'].value_counts().unstack().plot(kind='bar', f
```

> *方法总结*
>
> - 首先, 得到分组操作后的结果`data.groupby('race')['flee'].value_counts()`
> - 这里有一个之前介绍的`.unstack`操作, 这会让你得到一个`DateFrame`, 然后调用条形图, pandas就会遍历每一个组(`unstack`后为每一行), 然后作各组的条形图

场景二: 按不同逃逸类型分组, 组内的年龄分布是如何的?

(属性A的不同分组中, 连续属性B的情况是怎么样的)

```
data.groupby('flee')['age'].plot(kind='kde', legend=True, figsize=(20, 5
```

> *方法总结*
>
> 这里`data.groupby('flee')['age']`是一个`SeriesGroupby`对象, 顾名思义, 就是每一个组都有一个`Series`. 因为划分了不同逃逸类型的组, 每一组包含了组内的年龄数据, 所以直接`plot`相当于遍历了每一个逃逸类型, 然后分别画分布图.
>
> *pandas 会为不同组的作图分配颜色, 非常方便*

高级操作

场景三: 有时我们需要对组内不同列采取不同的操作

比如说, 我们按flee分组, 但是我们需要对每一组中的年龄求中位数, 对是否有精神问题求占比

这时我们可以这样做

```
data.groupby('race').agg({'age': np.median, 'signs_of_mental_illness': n
```

	age	signs_of_mental_illness
race		
A	35.0	0.256410
B	30.0	0.153722

H	31.0	0.200946
N	29.0	0.258065
O	29.5	0.250000
W	38.0	0.318068

> *方法总结*
> 这里我们操作的`data.groupby('race')`是一个`DataFrameGroupby`, 也就是说, 每一组都有一个`DataFrame`
>
> 我们把对这些`DataFrame`的操作计划写成了了一个字典`{'age': np.median, 'signs_of_mental_illness': np.mean}`, 然后进行`agg`, (aggragate, 合计)
>
> 然后我们得到了一个`DataFrame`, 每行对应一个组, 没列对应各组`DataFrame`的合计信息, 比如第二行第一列表示, 黑人被击毙者中, 年龄的中位数是30, 第二行第二列表示, 黑人被击毙者中, 有精神疾病表现的占15%

场景四: 我们需要同时求不同组内, 年龄的均值, 中位数, 方差

```
data.groupby('flee')['age'].agg([np.mean, np.median, np.std])
```

	mean	median	std
flee			
Car	33.911765	33.0	11.174234

Foot	30.972222	30.0	10.193900
Not fleeing	38.334753	36.0	13.527702
Other	33.239130	33.0	9.932043

> *方法总结*
>
> 现在我们对一个`SeriesGroupby`同时进行了多种操作. 相当于同时得到了这三行的结果:

```
data.groupby('flee')['age'].mean()
data.groupby('flee')['age'].median()
data.groupby('flee')['age'].std()
```

所以这其实是基本操作部分的进阶

场景五: 结合场景三和场景四可以吗?

答案是肯定的, 请看

```
data.groupby('flee').agg({'age': [np.median, np.mean], 'signs_of_mental_
```

	age		signs_of_mental_illness_mean
flee	**median**	**mean**	**mean**
Car	33.0	33.911765	0.114286
Foot	30.0	30.972222	0.115646

Not fleeing	36.0	38.334753	0.319174
Other	33.0	33.239130	0.072917

但是这里有一个问题, 这个列名分了很多层级, 我们可以进行重命名:

```
agg_df = data.groupby('flee').agg({'age': [np.median, np.mean], 'signs_o
agg_df.columns = ['_'.join(col).strip() for col in agg_df.columns.values]
agg_df
```

	age_median	age_mean	signs_of_mental_illness_mean
flee			
Car	33.0	33.911765	0.114286
Foot	30.0	30.972222	0.115646
Not fleeing	36.0	38.334753	0.319174
Other	33.0	33.239130	0.072917

> ***方法总结***
>
> 注意这里agg接受的不一定是`np.mean`这些函数, 你还可以进行自定义函数哦

总结

Groupby 可以简单总结为 split, apply, combine, 也就是说:

- **split** : 先将数据按一个属性分组 (*得到* `DataFrameGroupby` / `SeriesGroupby`)
- **apply** : 对每一组数据进行操作 (*取平均 取中值 取方差 或 自定义函数*)
- **combine**: 将操作后的结果结合起来 (*得到一个* `DataFrame` *或* `Series` *或可视化图像*)

希望看完本文你已经对groupby的使用有清晰的印象, 并充满信心, 如果你需要更细致的微操作, 多属性Groupby等, 可以进一步阅读文档