**Lesson6 Pandas-groupby¶**

* 数据的分组和聚合
  + pandas groupby 方法
  + pandas agg 方法
  + pandas apply 方法

**groupby=拆分数据(split)+应用某个函数(apply)+汇总计算结果(aggregate)¶**

* 主要思想是分拆-应用-汇总
* 将数据根据某些条件分拆为几个子数据，然后在每个子数据上进行计算从而得到所要的结果
* 对于一些简单的计算，比如最大值最小值的计算，我们可以直接使用groupby之后采用相应的内置方法
* 对于一些更为复杂的计算，我们需要自己定义函数然后应用到拆分后的子数据上。根据具体要求来决定使用agg方法还是apply方法

In [3]:

**import** **pandas** **as** **pd**

iris=pd.read\_csv("/Users/dengsudden/Documents/python\_course/iris.csv")

iris.head()

Out[3]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **sepallength** | **sepalwidth** | **petallength** | **petalwidth** | **species** |
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

In [4]:

iris.species.value\_counts()

Out[4]:

Iris-virginica 50

Iris-versicolor 50

Iris-setosa 50

Name: species, dtype: int64

**对于一些简单的计算，比如最大值最小值的计算，我们可以直接使用groupby之后采用相应的内置方法¶**

In [7]:

*#按品种划分，取出每个品种的花萼、花瓣长宽最大值*

iris.groupby('species').max()

Out[7]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **sepallength** | **sepalwidth** | **petallength** | **petalwidth** |
| **species** |  |  |  |  |
| **Iris-setosa** | 5.8 | 4.4 | 1.9 | 0.6 |
| **Iris-versicolor** | 7.0 | 3.4 | 5.1 | 1.8 |
| **Iris-virginica** | 7.9 | 3.8 | 6.9 | 2.5 |

In [10]:

iris.groupby("species").mean()

Out[10]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **sepallength** | **sepalwidth** | **petallength** | **petalwidth** |
| **species** |  |  |  |  |
| **Iris-setosa** | 5.006 | 3.418 | 1.464 | 0.244 |
| **Iris-versicolor** | 5.936 | 2.770 | 4.260 | 1.326 |
| **Iris-virginica** | 6.588 | 2.974 | 5.552 | 2.026 |

In [15]:

*# 用size查看每个分组的大小*

print(iris.groupby("species").size())

*#value\_counts()也可以实现*

iris.species.value\_counts()

species

Iris-setosa 50

Iris-versicolor 50

Iris-virginica 50

dtype: int64

Out[15]:

Iris-virginica 50

Iris-versicolor 50

Iris-setosa 50

Name: species, dtype: int64

**对于一些更为复杂的计算，我们需要自己定义函数然后应用到拆分后的子数据上。根据具体要求来决定使用agg方法还是apply方法¶**

* agg:自定义一个函数来进行计算, 传入一个数组做参数，返回一个标量的结果.agg方法将一个函数使用在一个数列上，然后返回一个标量的值。
* apply:是一个更一般化的方法, 将一个数据分拆-应用-汇总

**agg( )¶**

In [20]:

*# 计算各品种各属性数值的跨度范围*

*#自定义一个函数，用于计算输入数组的最大最小值之差*

**def** range\_iris(arr):

**return** arr.max()-arr.min()

*#将自定义函数，用agg()方法应用到各分组*

iris.groupby("species").agg(range\_iris)

Out[20]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **sepallength** | **sepalwidth** | **petallength** | **petalwidth** |
| **species** |  |  |  |  |
| **Iris-setosa** | 1.5 | 2.1 | 0.9 | 0.5 |
| **Iris-versicolor** | 2.1 | 1.4 | 2.1 | 0.8 |
| **Iris-virginica** | 3.0 | 1.6 | 2.4 | 1.1 |

In [21]:

*#agg()中，可以同时应用多个函数（自定义的、内置的均可）*

*#只需将函数名放入一个列表，再将列表放入agg()中*

*#内置函数需要使用引号“”*

iris.groupby("species").agg(["max","min",range\_iris])

Out[21]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **sepallength** | | | **sepalwidth** | | | **petallength** | | | **petalwidth** | | |
|  | **max** | **min** | **range\_iris** | **max** | **min** | **range\_iris** | **max** | **min** | **range\_iris** | **max** | **min** | **range\_iris** |
| **species** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Iris-setosa** | 5.8 | 4.3 | 1.5 | 4.4 | 2.3 | 2.1 | 1.9 | 1.0 | 0.9 | 0.6 | 0.1 | 0.5 |
| **Iris-versicolor** | 7.0 | 4.9 | 2.1 | 3.4 | 2.0 | 1.4 | 5.1 | 3.0 | 2.1 | 1.8 | 1.0 | 0.8 |
| **Iris-virginica** | 7.9 | 4.9 | 3.0 | 3.8 | 2.2 | 1.6 | 6.9 | 4.5 | 2.4 | 2.5 | 1.4 | 1.1 |

In [25]:

*#agg()中，可以针对不同列应用不同的函数（参考字典写法{key1:value1,key2:value2}）*

iris.groupby("species").agg({"petallength":["mean","min"],"petalwidth":["mean",range\_iris]})

Out[25]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **petallength** | | **petalwidth** | |
|  | **mean** | **min** | **mean** | **range\_iris** |
| **species** |  |  |  |  |
| **Iris-setosa** | 1.464 | 1.0 | 0.244 | 0.5 |
| **Iris-versicolor** | 4.260 | 3.0 | 1.326 | 0.8 |
| **Iris-virginica** | 5.552 | 4.5 | 2.026 | 1.1 |

**apply()¶**

In [31]:

*# 提取每个品种前n个观测值作为一个样本*

*#自定义函数，返回df的前n个值*

**def** first\_n(df,n):

**return** df[0:n]

*#将自定义函数应用到各分组中，此处使用apply()，而不是agg()*

*#因为agg方法必须生成一个标量数值，如果提取的样本数大于1，agg方法就不再适用？？*

iris.groupby("species").apply(first\_n,3)

Out[31]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **sepallength** | **sepalwidth** | **petallength** | **petalwidth** |
| **species** |  |  |  |  |
| **Iris-setosa** | 5.006 | 3.418 | 1.464 | 0.244 |
| **Iris-versicolor** | 5.936 | 2.770 | 4.260 | 1.326 |
| **Iris-virginica** | 6.588 | 2.974 | 5.552 | 2.026 |

**agg apply区别¶**

经过前节的代码实操，发现agg与apply需要拿出来对比说明

* 两者都是作用在groupby之后的对象上的
* agg将一维数组简化为标量值
* apply更一般化，可以处理多维数组，比如取分组前n个值，排序等等

网友见解：这两个函数都是作用在groupby对象上的，也就是分完组的对象上的，分完组之后针对某一组，如果值是一维数组，在利用完特定的函数之后，能做到简化的话，agg就能调用，反之，如果自定义的函数是排序，或者像是书中278页所定义的top这一类的函数，当然是agg所不能解决的，这时候用apply就可以解决，因为apply更一般化。

In [38]:

*#计算鸢尾花各品种的各个属性均值、最大值、最小值*

iris.groupby("species").agg(["mean","max","min"])

Out[38]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **sepallength** | | | **sepalwidth** | | | **petallength** | | | **petalwidth** | | |
|  | **mean** | **max** | **min** | **mean** | **max** | **min** | **mean** | **max** | **min** | **mean** | **max** | **min** |
| **species** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Iris-setosa** | 5.006 | 5.8 | 4.3 | 3.418 | 4.4 | 2.3 | 1.464 | 1.9 | 1.0 | 0.244 | 0.6 | 0.1 |
| **Iris-versicolor** | 5.936 | 7.0 | 4.9 | 2.770 | 3.4 | 2.0 | 4.260 | 5.1 | 3.0 | 1.326 | 1.8 | 1.0 |
| **Iris-virginica** | 6.588 | 7.9 | 4.9 | 2.974 | 3.8 | 2.2 | 5.552 | 6.9 | 4.5 | 2.026 | 2.5 | 1.4 |

In [44]:

*#计算鸢尾花各品种花萼长度(sepal\_length)大于6cm的数据个数*

*#自定义函数，找出数组中sepallength>6的条目，并返回满足条件的条目个数*

**def** big6(arr):

arr6=arr[arr.sepallength>6]

**return** arr6.sepallength.count()

*#对分组数据使用自定义函数*

iris.groupby("species").apply(big6)

Out[44]:

species

Iris-setosa 0

Iris-versicolor 20

Iris-virginica 41

dtype: int64