05分组聚合

Pandas使用groupby实现分组统计

类似`SQL`：

`select city,max(temperature) from city_weather group by city;`

`groupby`：先对数据分组，然后在每个分组上应用聚合函数、转换函数

> *本次演示：*
> *1、分组使用聚合函数做数据统计*
> *2、遍历groupby的结果理解执行流程*
> *3、实例分组探索天气数据*

```
# 定义一个df
df = pd.DataFrame({'A': ['foo', 'bar', 'foo', 'bar', 'foo', 'bar', 'foo', 'foo'],
                   'B': ['one', 'one', 'two', 'three', 'two', 'two', 'one', 'three'],
                   'C': np.random.randn(8),
                   'D': np.random.randn(8)})
```

一、分组使用聚合函数做数据统计

1、单个列groupby，查询所有数据列的统计

```
df.groupby('A').sum()
#可以看到输出结果中的'A'变成了索引，并且'B'列不是数值，所以被忽略
```

2、多个列groupby，查询所有数据列的统计

```
df.groupby(['A','B']).mean()
# ('A','B')成对变成了二级索引
df.groupby(['A','B'], as_index=False).mean()
# 去除二级索引
```

3、同时查看多种数据统计

```
df.groupby('A').agg([np.sum, np.mean, np.std])
# 列变成了多级索引
```

4、查看单列的结果数据统计

```
# 方法1：预过滤，性能更好
df.groupby('A')['C'].agg([np.sum, np.mean, np.std])
# 方法2
df.groupby('A').agg([np.sum, np.mean, np.std])['C']
```

5、不同列使用不同的聚合函数

```
df.groupby('A').agg({"C":np.sum, "D":np.mean})
```

二、遍历groupby的结果理解执行流程

for循环可以直接遍历每个group

1、遍历单个列聚合的分组

```
g = df.groupby('A')
g
# 获取单个分组的数据
g.get_group('bar')
```

2、遍历多个列聚合的分组

```
g = df.groupby(['A', 'B'])
for name,group in g:
    print(name)
    print(group)
    print()
# 可以看到，是一个2个元素的tuple，代表不同的列
g.get_group(('foo', 'one'))
# 可以直接查询group后的某几列，生成Series或者子DataFrame
g['C']
for name, group in g['C']:
    print(name)
    print(group)
    print(type(group))
    print()
# 其实所有的聚合统计，都是在dataframe和series上进行的；
```

三、实例分组探索天气数据

```
fpath = "res/beijing_tianqi_2018.csv"
df = pd.read_csv(fpath)
# 替换掉温度的后缀℃
df.loc[:, "bWendu"] = df["bWendu"].str.replace("℃",
"").astype('int32')
df.loc[:, "yWendu"] = df["yWendu"].str.replace("℃",
"").astype('int32')
```

```
df.head()

# 1.新增一列为月份
df['month'] = df['ymd'].str[:7]
df.head()

# 2.查看每个月的最高温度
data = df.groupby('month')['bWendu'].max()
data
type(data)
data.plot()  # 画图

# 3.查看每个月的最高温度、最低温度、平均空气质量指数
group_data = df.groupby('month').agg({"bWendu":np.max,
"yWendu":np.min, "aqi":np.mean})
group_data
group_data.plot()
```

四、分组聚合后不同列的数据统计

电影评分数据集（UserID，MovieID，Rating，Timestamp）

聚合后单列-单指标统计：每个**MovieID**的平均评分

```
df.groupby("MovieID")["Rating"].mean()
```

聚合后单列-多指标统计：每个**MoiveID**的最高评分、最低评分、平均评分

```
df.groupby("MovieID")["Rating"].agg(mean="mean", max="max", min=np.min)
df.groupby("MovieID").agg({"Rating":['mean', 'max', np.min]})
```

聚合后多列-多指标统计：每个**MoiveID**的评分人数，最高评分、最低评分、平均评分

```
df.groupby("MovieID").agg( rating_mean=("Rating", "mean"), user_count=
("UserID", lambda x : x.nunique())
df.groupby("MovieID").agg( {"Rating": ['mean', 'min', 'max'], "UserID":
lambda x :x.nunique()})
df.groupby("MovieID").apply( lambda x: pd.Series( {"min":
x["Rating"].min(), "mean": x["Rating"].mean()}))
```

记忆：agg(新列名=函数)、agg(新列名=(原列名，函数))、agg({"原列名"：函数/列表})
agg函数的两种形式，等号代表“把结果赋值给新列”，字典/元组代表“对这个列运用这些函数”

官网文档：https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.core.groupby.DataFrameGroupBy.agg.html

1.读取数据

```
df = pd.read_csv(
    "res/ratings.dat",
    sep="::",
    engine='python',
    names="UserID::MovieID::Rating::Timestamp".split("::")
)
```

2.聚合后单列-单指标统计

```
# 每个MovieID的平均评分
result = df.groupby("MovieID")["Rating"].mean()
result.head()
type(result)
```

3.聚合后单列-多指标统计

每个MoiveID的最高评分、最低评分、平均评分

```
# 方法1：agg函数传入多个结果列名=函数名形式
result = df.groupby("MovieID")["Rating"].agg(
    mean="mean", max="max", min=np.min
)
result.head()

# 方法2：agg函数传入字典，key是column名，value是函数列表
# 每个MoiveID的最高评分、最低评分、平均评分
result = df.groupby("MovieID").agg(
    {"Rating":['mean', 'max', np.min]}
)
result.head()
# 去除二级索引
result.columns = ['age_mean', 'age_min', 'age_max']
result.head()
```

4.聚合后多列-多指标统计

每个MoiveID的评分人数，最高评分、最低评分、平均评分

```
# 方法1：agg函数传入字典，key是原列名，value是原列名和函数元组
result = df.groupby("MovieID").agg(
        rating_mean=("Rating", "mean"),
        rating_min=("Rating", "min"),
        rating_max=("Rating", "max"),
        user_count=("UserID", lambda x : x.nunique())
)
result.head()

# 方法2：agg函数传入字典，key是原列名，value是函数列表
result = df.groupby("MovieID").agg(
```

```
    {
        "Rating": ['mean', 'min', 'max'],
        "UserID": lambda x :x.nunique()
    }
)
result.head() # 结果是二级索引
# 去除二级索引
result.columns = ["rating_mean",
"rating_min","rating_max","user_count"]
result.head()

# 方法3：使用groupby之后apply对每个子df单独统计
def agg_func(x):
    """注意，这个x是子DF"""

    # 这个Series会变成一行，字典KEY是列名
    return pd.Series({
        "rating_mean": x["Rating"].mean(),
        "rating_min": x["Rating"].min(),
        "rating_max": x["Rating"].max(),
        "user_count": x["UserID"].nunique()
    })

result = df.groupby("MovieID").apply(agg_func)
result.head()

```