15. Nhóm dữ liệu (phần 1)

Groupby

Với "groupby " chúng ta đang đề cập đến một quá trình liên quan đến một hoặc nhiều bước sau:

(1) Chia dữ liệu thành các nhóm dựa trên một số tiêu chí

(2) Áp dụng một số tính toán cho từng nhóm một cách độc lập

(3) Kết hợp các kết quả vào một cấu trúc dữ liệu

Trong số này, bước phân chia nhóm (1) là đơn giản nhất. Trên thực tế, trong nhiều trường hợp, bạn có thể chia tập dữ liệu thành các nhóm và làm điều gì đó với các nhóm đó. Trong bước (2), mong muốn có thể là một trong những điều sau:

Tổng hợp (**Aggregation/Reduction)**: phân tích thống kê trong một nhóm. Ví dụ:

Tính sum, mean, std, min, max của nhóm dữ liệu

Tính toán số lượng nhóm (size/count)

Chuyển đổi (**Transformation**): thực hiện một số tính toán nhóm cụ thể và trả về một kiểu giống chỉ mục. Vài ví dụ:

Chuẩn hoá dữ liệu (zscore) trong nhóm

Fill các giá trị NAs bằng một giá trị từ các nhóm.

Lọc (**Filtration**): loại bỏ một số nhóm, tính toán theo nhóm nhằm đánh giá Đúng hay Sai. Vài ví dụ:

Loại bỏ dữ liệu thuộc nhóm chỉ với một vài thành viên

Lọc dữ liệu dựa trên sum hay mean của nhóm

Hoặc là kết hợp của các điều trên: GroupBy sẽ kiểm tra kết quả của các bước và cố gắng trả lại một kết quả hợp lý hợp lý nếu nó không phù hợp với một trong hai loại trên.

Bài học gồm:

+ groupdby

+ Tổng hợp (**Aggregation/Reduction)**

Cú pháp trong pandas tự nhiên hơn cách làm trong sql.

Chẳng hạn trong sql, groupby như sau.

```
SELECT Column1, Column2, mean(Column3), sum(Column4)
FROM SomeTable
GROUP BY Column1, Column2
```

Ví dụ, groupby trong dataframe.

```
>>> import pandas as pd

>>> import numpy as np
>>> df = pd.DataFrame({'A' : ['foo', 'bar', 'foo', 'bar'],'B' : ['one', 'one', 'two', 'three'],'C' : np.random.randn(4), 'D' : np.random.randn(4)
>>> df
     A      B         C         D
0  foo    one  0.064892  1.436126
1  bar    one  1.119634 -1.452583
2  foo    two  0.369738  1.437522
```

```
3  bar  three  0.114924  0.849737
>>> grouped = df.groupby(['A', 'B'])
>>> grouped.last()
                C         D
A   B
bar one    1.119634 -1.452583
    three  0.114924  0.849737
foo one    0.064892  1.436126
    two    0.369738  1.437522
>>> grouped.count()
          C  D
A   B
bar one    1  1
    three  1  1
foo one    1  1
    two    1  1
>>> grouped.sum()
                C         D
A   B
bar one    1.119634 -1.452583
    three  0.114924  0.849737
foo one    0.064892  1.436126
    two    0.369738  1.437522
>>>
```

TRICK. Chú ý với kiểu dữ liệu string (categorical data) ở cột A và cột B. Ta cần chuyển A và B về kiểu category type để có được những lợi ích sau: Tốn ít bộ nhớ, tăng tốc các operations đặc biệt là trong groupby().

```
>>> df.A.unique()

array(['foo', 'bar'], dtype=object)

>>> df.B.unique()

array(['one', 'two', 'three'], dtype=object)

>>> df.A = df.A.astype('category')

>>> df.B = df.B.astype('category')
```

Ví dụ, groupby trong Series

```
>>> s = pd.Series(list(df['C']), df['A'])

>>> s

A

foo    0.064892

bar    1.119634

foo    0.369738

bar    0.114924

dtype: float64

>>> grouped = s.groupby(level=0)

>>> grouped.first()

A

bar    1.119634

foo    0.064892

dtype: float64

>>> grouped.count()
```

```
A
bar    2
foo    2
dtype: int64
>>> grouped.sum()
A
bar    1.234558
foo    0.434630
dtype: float64
>>>
```

Theo mặc định, các keys được sắp xếp trong quá trình groupby. Tuy nhiên bạn có thể không sắp xếp qua phép gán sort = False để tăng tốc:

```
>>> df.groupby(['A','B']).sum()
                C         D
A   B
bar one    1.119634 -1.452583
    three  0.114924  0.849737
foo one    0.064892  1.436126
    two    0.369738  1.437522
>>> df.groupby(['A','B'],sort=False).sum()
                C         D
A   B
foo one    0.064892  1.436126
bar one    1.119634 -1.452583
foo two    0.369738  1.437522
bar three  0.114924  0.849737
>>>
```

Ta có thể nhận kết quả của groupby là kiểu dictionary qua thuộc tính “groups”

```
>>> df.groupby('A').groups
{'foo': Int64Index([0, 2], dtype='int64'), 'bar': Int64Index([1, 3], dtype='int64')}
>>>
```

Duyệt qua toàn bộ grouped

```
>>> grouped = df.groupby(['A','B'])
>>> for name, group in grouped:
...     print name
...     print group
```

```
...
('bar', 'one')
   A   B        C        D
1  bar  one  1.119634 -1.452583
('bar', 'three')
   A      B        C        D
3  bar  three  0.114924  0.849737
('foo', 'one')
   A   B        C        D
0  foo  one  0.064892  1.436126
('foo', 'two')
   A   B        C        D
2  foo  two  0.369738  1.437522
>>>
```

Phương thức get_group() giúp ta có được giá trị của một nhóm. Ví dụ

```
>>> grouped.get_group(('foo','two'),None)
   A   B        C        D
2  foo  two  0.369738  1.437522
>>>
```

Aggregation

Các phép “aggregation/reduction” có sẵn như sum(), count(),mean() ngoài ra ta có thể tự định nghĩa nhiều “aggregation” riêng cho dữ liệu của mình.

Tạo một dataframe để minh họa cho bài học.

```
>>> import pandas as pd
>>> import numpy as np
>>> df = pd.DataFrame({'A' : ['foo', 'bar', 'foo', 'bar'],'B' : ['one', 'one', 'two', 'three'],'C' : np.random.randn(4), 'D' : np.random.randn(4)
>>> df
   A      B        C        D
0  foo    one  0.791518 -0.184029
1  bar    one  0.268664 -0.891467
2  foo    two  0.652872  1.045733
3  bar  three  0.650932 -1.657792
>>>
```

Ví dụ sau: so sánh kết quả của “aggregate” sử dụng hàm có sẵn np.sum và một hàm tự viết để tính tổng trong một group.

```
>>> grouped.aggregate(np.sum)
                C         D
A   B
bar one   -0.312775  0.873260
    three -0.229364 -0.177659
foo one   -1.270082 -1.648492
    two   -0.605489  0.885898
```

Một ví dụ cho việc sử dụng custom aggregation.

```
>>> grouped.aggregate(lambda series: reduce(lambda x, y: x + y, series.tolist()))
                C         D
A   B
bar one   -0.312775  0.873260
    three -0.229364 -0.177659
foo one   -1.270082 -1.648492
    two   -0.605489  0.885898
>>>
```

Trong quá trình “aggregation” ta có thể áp dụng cùng lúc nhiều hàm.

```
>>> grouped['C'].aggregate([np.sum, np.mean, np.std])
              sum      mean  std
A   B
bar one   -0.312775 -0.312775  NaN
    three -0.229364 -0.229364  NaN
foo one   -1.270082 -1.270082  NaN
    two   -0.605489 -0.605489  NaN
>>>
```

Ta có thể áp dụng từng hàm cho từng cột trong quá trình “aggregation”

```
>>> grouped.agg({'C' : np.sum,'D' : np.mean})
                C         D
A   B
bar one   -0.312775  0.873260
    three -0.229364 -0.177659
foo one   -1.270082 -1.648492
    two   -0.605489  0.885898
>>>
```

Kết luận

Trong bài này, bạn đọc đã có thể học được cách xác định và chia dataframe theo các nhóm để tiến hành tổng hợp (**Aggregation/Reduction**) hoặc phân tích thêm (ví dụ như tính sum, std, min, max ...). Bạn đọc sẽ được học cách chuyển đổi và lọc dữ liệu, bao gồm cách phát hiện các giá trị ngoại vi (outliers) và xử lý các giá trị bị thiếu ở bài tiếp theo.