13. Tổ chức lại bảng dữ liệu (phần 1)

Dữ liệu mà ta gặp phải thường không ở dạng (shape) mà chúng ta mong muốn. Các thao tác reshaping dữ liệu nhằm mục đích khiến nó dễ dàng để quan sát mối quan hệ, tương quan của các cặp dữ liệu và đó là lí do cho bài học này. Có vô vàn các nhu cầu chuyển đổi qua lại giữa cột và hàng.

Bài học gồm có:

+ pivot()

+ stack()

+ unstack()

.pivot(index=None, columns=None, values=None)

Ví dụ cửa hàng tôi bán 3 sản phẩm và số lượng sản phẩm mỗi loại theo ngày khác nhau như dữ liệu sau đây:

```
>>> import pandas as pd
>>> df = pd.DataFrame([[2011,'Iphone',10000],[2011,'SS',30000],[2011,'LG',123000],\
[2012,'SS',134500],[2012,'LG',90000],[2012,'Iphone',23400],[2013,'Iphone',56000],\
[2013,'LG',234000],[2013,'SS',1234567]], columns=['year','product','turnover'])
>>> df
   year product  turnover
0  2011  Iphone     10000
1  2011      SS     30000
2  2011      LG    123000
3  2012      SS    134500
4  2012      LG     90000
5  2012  Iphone     23400
6  2013  Iphone     56000
7  2013      LG    234000
8  2013      SS   1234567
>>>
```

Rõ ràng khi tôi muốn so sánh số liệu bán hàng của mỗi sản phẩm qua các năm thì cách biểu diễn trên là không tốt. Như vậy mong muốn chuyển đổi dữ liệu trên thành dạng như sau:

```
product  Iphone      LG       SS
year
2011      10000  123000    30000
2012      23400   90000   134500
2013      56000  234000  1234567
```

Pandas đã cung cấp một phương thức giúp ta thay đổi shape của dữ liệu như trên đó là ".pivot(index=None, columns=None, values=None)". Lưu ý nho nhỏ ở đây là nếu values không được chỉ định rõ ràng, tất cả các cột còn lại sẽ được sử dụng và kết quả sẽ có các cột được lập chỉ mục theo thứ bậc. Với ví dụ trên ta làm như sau:

```
>>> df.pivot(index='year', columns='product',values = 'turnover')
product  Iphone      LG       SS
year
2011      10000  123000    30000
2012      23400   90000   134500
2013      56000  234000  1234567
>>>
```

Và chúng ta cũng đã học được cách tạo ra multi-level index sử dụng set_index() từ các bài trước đó.

```
>>> df.set_index(['year','product'], inplace=True)
              turnover
year product
2011 Iphone      10000
     SS          30000
     LG         123000
2012 SS         134500
     LG          90000
     Iphone      23400
2013 Iphone      56000
     LG         234000
     SS        1234567
>>>
```

.unstack(level=-1, fill_value=None)” & .stack(level=-1, dropna=True)

Giả sử đầu vào dữ liệu lại thuộc dạng multi-level index như bảng trên thì phương thức .pivot() không thể rearrange các cột của một dataframe. Như vậy bài toán hiện tại là cần phải chuyển một vài index level về column level. Để làm được điều này thì ta có thể sử dụng phương thức “.unstack(level=-1, fill_value=None)”.

Level nhận giá trị default là -1 ứng với last level.

```
>>> df.unstack(level='year')
        turnover
year        2011    2012     2013
product
Iphone     10000   23400    56000
LG        123000   90000   234000
SS         30000  134500  1234567
>>> df.unstack(level='product')
        turnover
product   Iphone      LG       SS
year
2011       10000  123000    30000
2012       23400   90000   134500
2013       56000  234000  1234567
>>>
```

Dữ liệu được chuyển đổi từ “long dataframe” thành “wider dataframe”.

Và ngược với bài toán cần đến unstack là bài toán cần chuyển một vài column level thành index level khiến dữ liệu từ “wide dataframe” thành dạng “longer dataframe”.

Ví dụ cần chuyển “wide dataframe” sau thành dạng “longer dataframe”.

```
>>> df_wide = df.unstack(level="product")
>>> df_wide
        turnover
product   Iphone       SS      LG
year
2011       10000    30000  123000
2012       23400   134500   90000
2013       56000  1234567  234000
```

```
>>> df_long = df_wide.stack(level="product")
>>> df_long
             turnover
year product
2011 Iphone      10000
     LG         123000
     SS          30000
2012 Iphone      23400
     LG          90000
     SS         134500
2013 Iphone      56000
     LG         234000
     SS        1234567
>>>
```

.swaplevel(i=-2, j=-1, axis=0)

Như vậy unstack() và stack() giúp ta chuyển đổi qua lại giữa multi-level index và multi-level column. Ngoài ra ta còn một chuyển đổi khác đó là hoán đổi vị trí của các index levels, lúc đó ta cần dùng đến swaplevel(). Ví dụ với dataframe như sau:

```
>>> df = pd.DataFrame([[2011,'Iphone',10000],[2011,'SS',30000],[2011,'LG',123000],\
[2012,'SS',134500],[2012,'LG',90000],[2012,'Iphone',23400],[2013,'Iphone',56000],\
[2013,'LG',234000],[2013,'SS',1234567]], columns=['year','product','turnover'])
>>> df = df.set_index(['product','year'])
>>> df
              turnover
product year
Iphone  2011     10000
SS      2011     30000
LG      2011    123000
SS      2012    134500
LG      2012     90000
Iphone  2012     23400
        2013     56000
LG      2013    234000
SS      2013   1234567
>>>
```

Cần hoán đổi vị trí của hai index levels là “year” và “product” ta làm như sau:

```
>>> df.swaplevel(0,1)
             turnover
year product
2011 Iphone      10000
     SS          30000
     LG         123000
2012 SS         134500
     LG          90000
     Iphone      23400
2013 Iphone      56000
     LG         234000
     SS        1234567
>>>
```

Hoặc và kết hợp với sort_index()

```
>>> df.swaplevel("year","product").sort_index()
             turnover
year product
2011 Iphone      10000
     LG         123000
     SS          30000
```

```
2012 Iphone      23400
    LG          90000
    SS         134500
2013 Iphone      56000
    LG         234000
    SS        1234567
>>>
```

Kết luận

Trong bài học này chúng tôi đã đề cập đến một số phương thức được hỗ trợ bởi pandas với mục đích bố trí lại dữ liệu để thuận tiện cho việc quan sát và kết xuất thông tin cho người sử dụng. Các bạn có thể thay đổi shape của dữ liệu với hàm .pivot(), chuyển đổi qua lại giữa multi-level index và multi-level column với .unstack() và .stack() hay hoán đổi vị trí của các index levels với .swaplevel(). Chúng ta sẽ tiếp tục với bài toán về rearranging và reshapeing dữ liệu trong bài học tiếp theo.