pandas學習筆記3—數據重塑圖解Pivot, Pivot-Table, Stack and Unstack

文章目錄

數據重塑圖解—Pivot, Pivot-Table, Stack and Unstack

引言

Pandas是python中常用的數據分析軟件庫，它提供了DataFrames和Series的工具，這使得numpy和matplotlib可以更加便捷地讀取轉換數據。

數據重塑表示轉換一個表格或者向量的結構，使其適合於進一步的分析。Pandas擁有一些其他軟件不具備的重塑功能，這對初學者來說可能會比較棘手。

本文中我將舉例說明Pandas中一些常用的重塑函數，並結合圖表進行闡述。

Pivot

pivot函數用於創建一個新的派生表，該函數有三個參數：index, columns和values。你需要在原始表中指定這三個參數所對定的列名，接下來pivot函數會創建一個新的表格，其中行索引和列索引都是唯一標示值，表格中的數值由原始表中參數value對應的數據所表示。

是不是感覺有點難以理解呢？看完下面這個例子你就明白了，假設給定下面這個表格：

其中item表示商品名稱，USD表示商品的美元價格，EU表示歐元價格，CType表示每個客戶對應的類別。

ix	Item	CType	USD	EU
0	Item0	Gold	1$	1€
1	Item0	Bronze	2$	2€
2	Item1	Gold	3$	3€
3	Item1	Silver	4$	4€

下述代碼片段用於創建DataFrame，需要注意的是本文中所有的代碼片段均需要導入以下模塊：

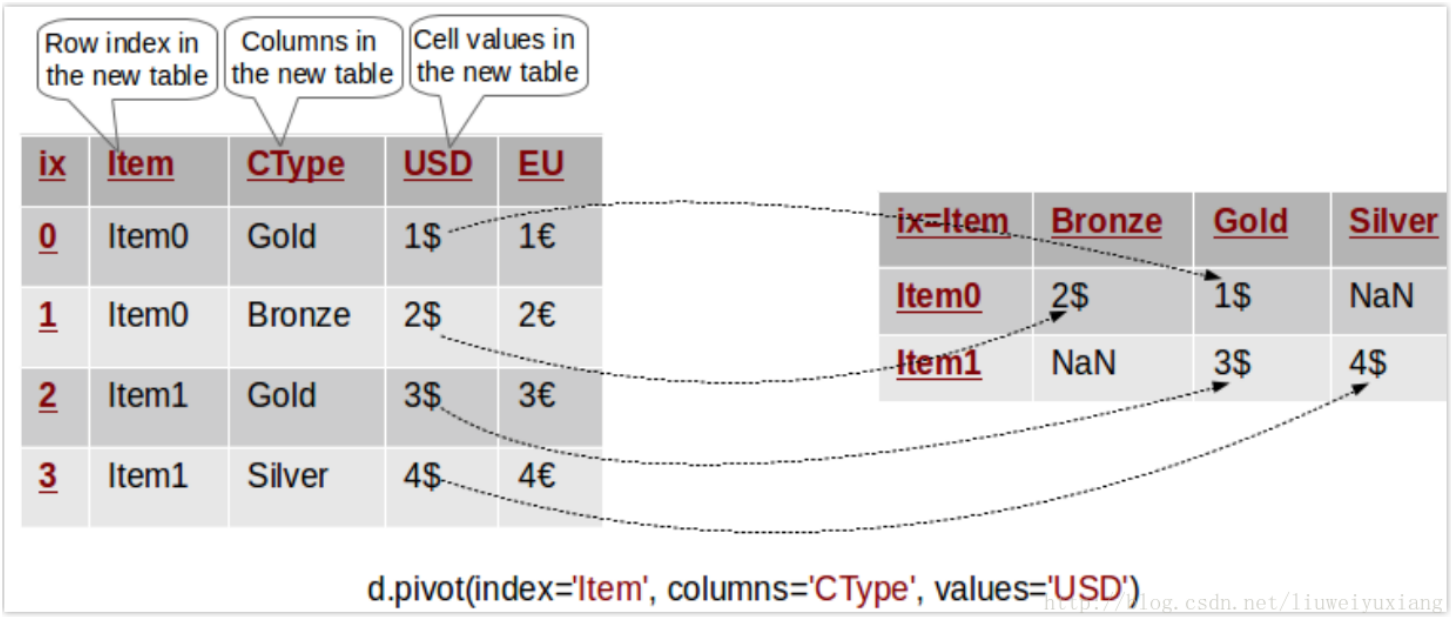
```
from collections import OrderedDict

from pandas import DataFrame

import pandas as pd

import numpy as np

table = OrderDict((

("Item",['Item0','Item0','Item1','Item1']),

("CType",['Gold','Bronze','Gold','Silver']),

("USD",['1$','2$','3$','4$']),

("EU",['1€','2€','3€','4€'])

))

d = DataFrame(table)
```

在這個表格中，我們很難觀測到商品的美元價格在不同的客戶中是如何變化的。此時我們傾向於重塑表格，使得所有的價格信息都按行排列：

```
p = d.pivot(index='Item', columns='CType', values='USD')
```

上述命令創建了一個新的表格，其中列索引是d.CType 中的唯一值，行索引是d.Item 中的唯一值，表格中的數值由d.USD 來填充。下圖形像地展示了這個過程：

換句話說，原始表中的USD數據已經被轉移到新表中，其中行列索引分別由Item和CType所表示，無法找到原始數據的用NaN所表示。



下述代碼介紹瞭如何分別從原始表和新表中查詢數據：

```
# Original DataFrame: Access the USD cost of Item0 for Gold customers

print(d[(d.Item=='Item0') & (d.CType=='Gold')].USD.values)

# Pivoted DataFrame: Access the USD cost of Item0 for Gold customers

print(p[p.index=='Item0'].Gold.values)
```

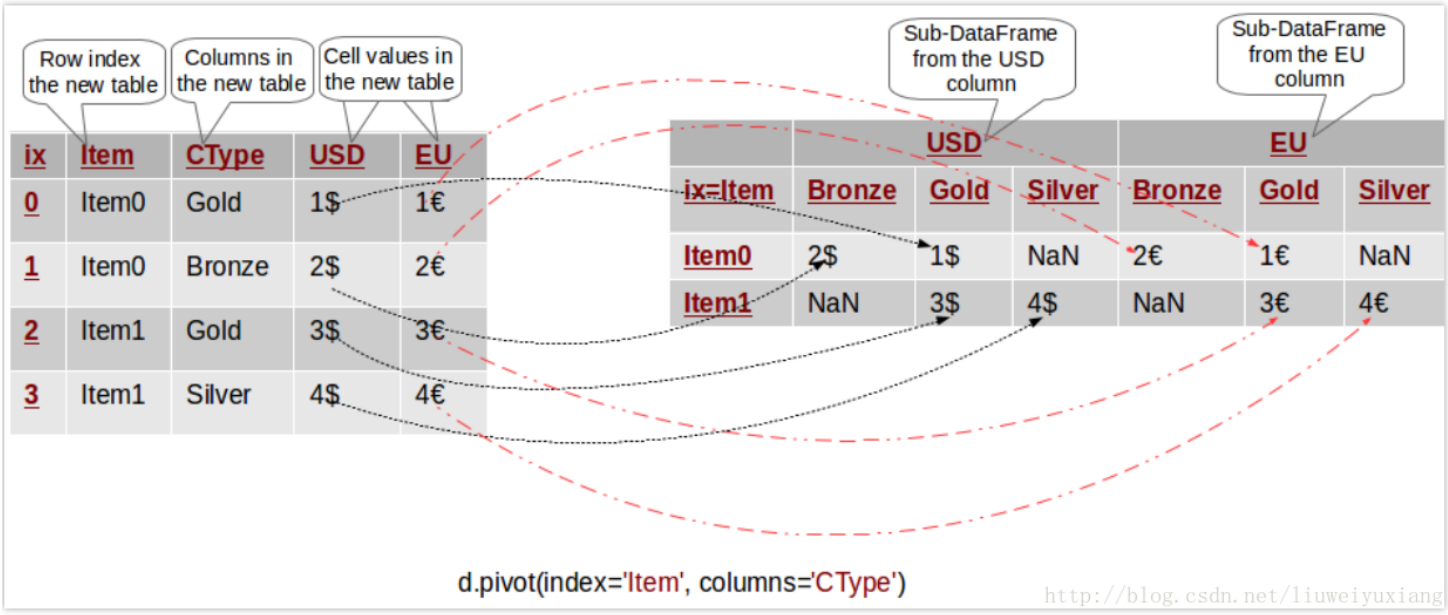
需要注意的是，該數據透視表中沒有包含歐元價格的任何信息。事實上，數據透視表是原始表格的簡化版本，它只包含我們所關心的變量信息。

現在我們對上述案例進行拓展，我們想將每個商品的歐元價格信息也納入數據透視表中(Pivoting By Multiple Columns)。這非常容易實現——我們只需將values 參數刪掉即可：

```
p = d.pivot(index='Item', columns='CType')
```

此時，Pandas會在新表格中創建一個分層列索引。你可以將分層索引想像成一個樹形索引，每個行/列索引都由從最頂層的索引到底部索引的路徑所組成。最頂層的索引由pivot函數中沒有定義的參數所組成——比如本例中的USD 和EU，第二層索引表示對應列中的所有唯一值。下圖形像地展示了該過程：

我們可以利用分層索引從原始表中過濾出某個變量的數據。比如p.USD將返回只包含USD 數據的數據透視表，p.USD.Bronze將上述透視表中的第一列篩選出來。

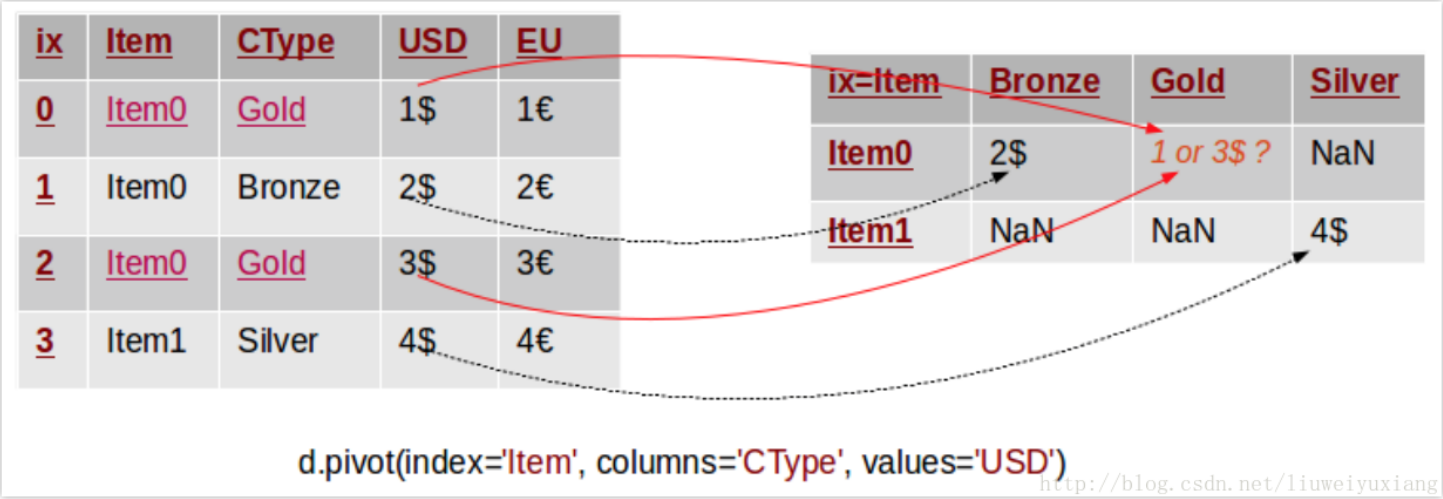



```
# Original DataFrame: Access the USD cost of Item0 for Gold customers

print(d[(d.Item=='Item0')&(d.CType=='Gold')].USD.values)

# Pivoted DataFrame: p.USD gives a "sub-DataFrame" with the USD values only

print(p.USD[p.USD.index=='Item0'].Gold.values)
```

常見錯誤

從上文的描述中我們可以看出：pivot方法至少需要兩個參數—— index 和columns。那麼如果原始數據集中存在重複條目時，重塑過程將會發生什麼問題呢？pivot函數如何確定數據透視表中的數值呢？下圖形像地展示了這個問題：

在這個案例中，原始數據集中存在重複條目，此時pivot函數無法確定數據透視表中的數值，它會返回一個錯誤信息：ValueError: Index contains duplicate entries, cannot reshape




```
table = OrderDict((

("Item",['Item0','Item0','Item0','Item1']),

('CType',['Gold','Bronze','Gold','Silver']),

('USD',['1$','2$','3$','4$']),

('EU',['1€','2€','3€','4€'])

))

d = DataFrame(table)

p = d.pivot(index='Item', columns='CType', values='USD')
```

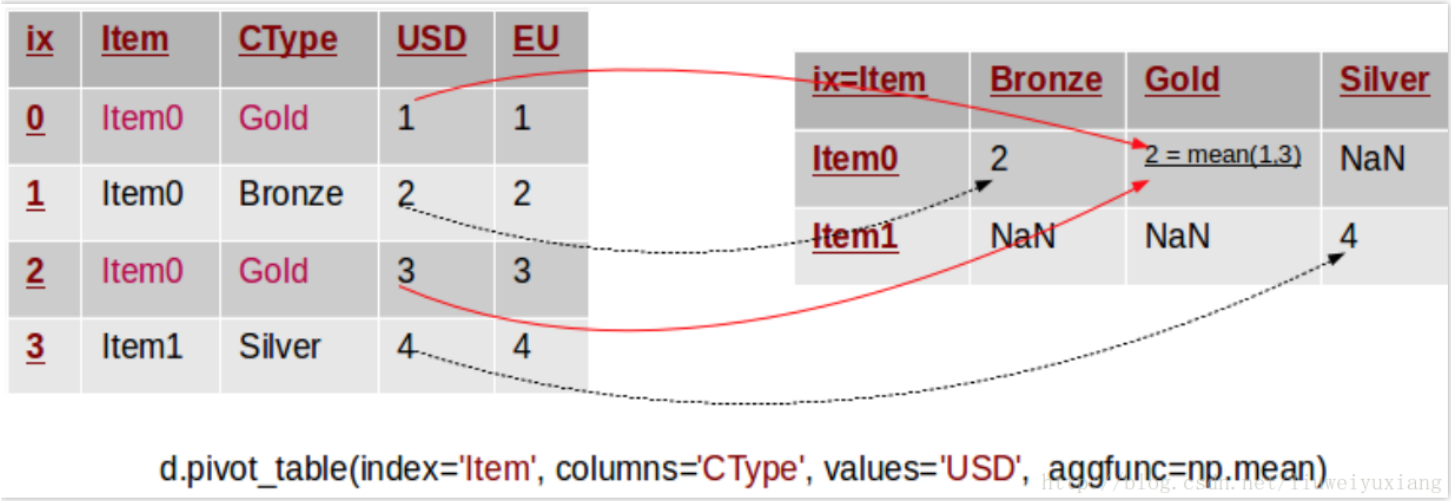
因此，我們在調用pivot方法前需要保證數據集中不存在重複條目，否則我們需要調用另外一個方法——pivot_table。

Pivot Table

pivot_table方法可以用來解決上述問題，與pivot相比，該方法可以匯總多個重複條目的數據。換句話說，在前面的例子中，我們可以用均值、中位數或者其他匯總函數來計算重複條目的數值。下圖形像地展示了這個過程：

注意，在這個例子中，我們移除了數據集中的美元和歐元符號。原始數據集中存在兩行重複條目，我們利用樣本均值來填充數據透視表中的數據。pivot_table方法需要傳遞一個新的參數aggfunc，該參數用於指明轉換

時所需的匯總函數。



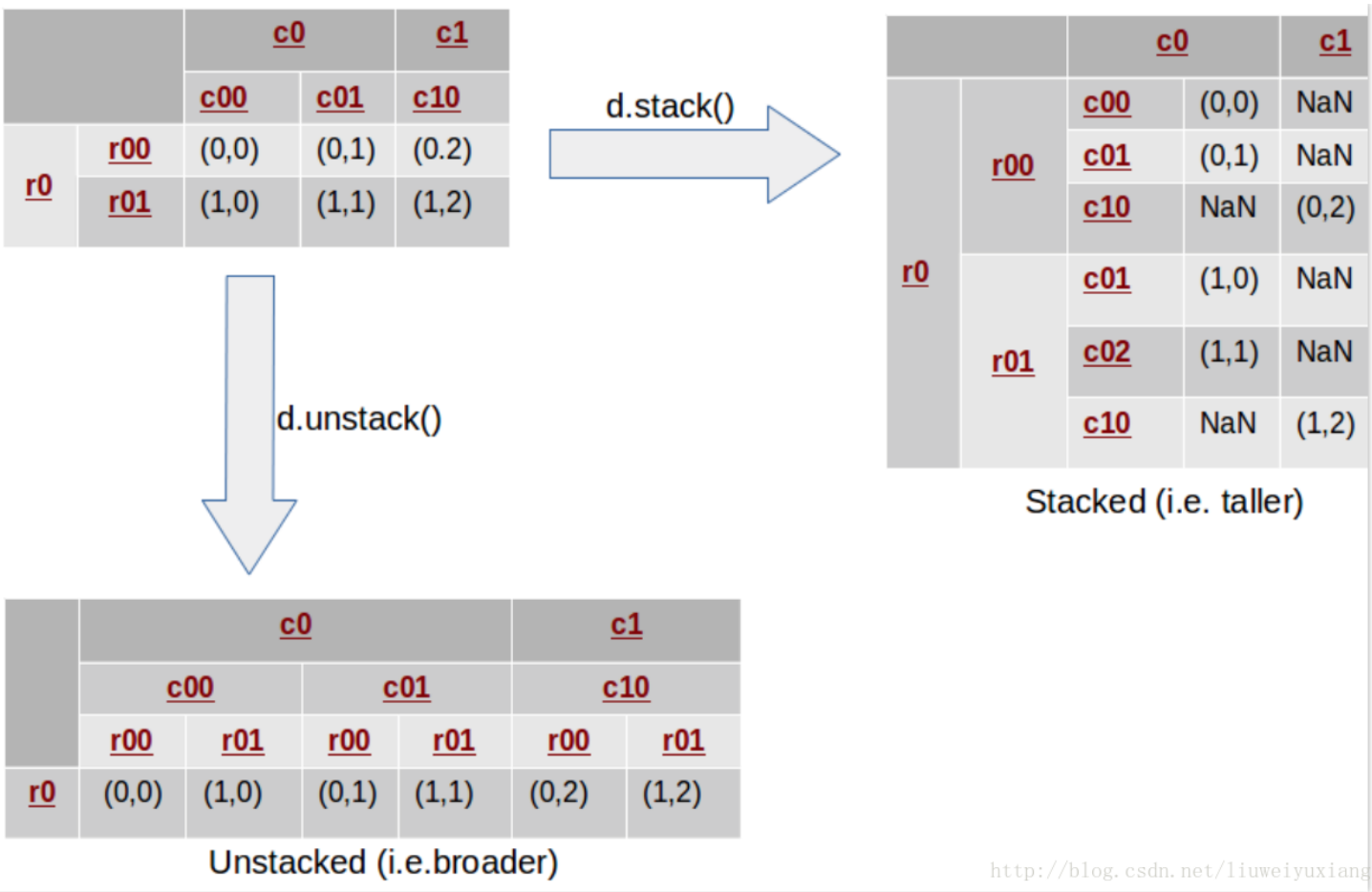

```
table = OrderDict((

('Item',['Item0','Item0','Item0','Item1']),

('CType',['Gold','Bronze','Gold','Silver']),

('USD',[1,2,3,4]),

('EU',[1.1,2.2,3.3,4.4])

))

d = DataFrame(table)

p=d.pivot_table(index='Item',columns='CType',values='USD', aggfunc=np.mean)
```

從本質上來說，pivot_table方法是pivot的通用版，該方法可以匯總重複條目的數據。

Stack/Unstack

實際上，軸向旋轉(pivot)運算是堆疊(stack)過程的特例。首先假設原始數據集中的行列索引中均為層次索引。stack 過程表示將數據集的列旋轉為行，同理unstack 過程表示將數據的行旋轉為列。下圖形像地展示了該過程：

在這個例子中，我們看到原始數據集中的行列索引都由二級分層索引組成。堆疊過程主要是將最內層的列索引轉換成最內層的行索引，然後再重新安排單元格中的數據。相反地，unstack過程是講最內層的行索引移到最內層的列索引中。



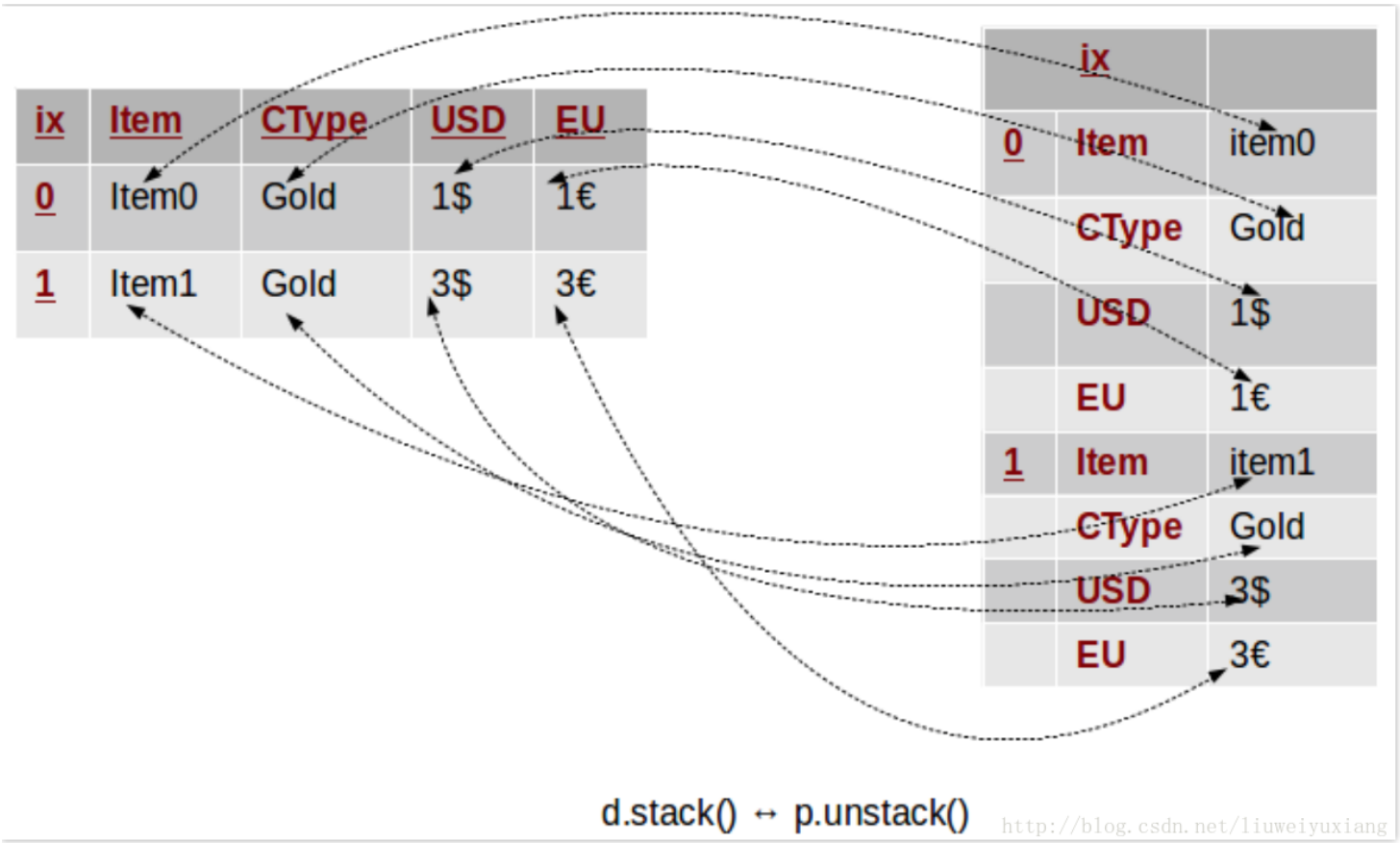
因此，我們可以發現stack使得數據集變得更長，unstack使得數據集變得更寬。

```
# Row Multi-Index

row_idx_arr = list(zip(['r0','r0'],['r-00','r-01']))

row_idx = pd.MultiIndex.from_tuples(row_idx_arr)

# Column Multi-Index

col_idx_arr = lis(zip(['c0','c0','c1'], ['c-00','c-01','c-10']))

col_idx = pd.MultiIndex.from_tuples(col_idx_arr)

# Create the DataFrame

d = DataFrame(np.arange(6).reshape(2,3),index=row_idx,  columns=col_idx)

d = d.applymap(lambda x: (x // 3, x % 3))

# Stack/Unstack

s = d.stack()

```

```
u = d.unstack()
```

事實上，Pandas允許我們利用stack/unstack 處理任一等級的索引。因此雖然默認設定處理最內層的索引，但是在上述的例子中，我們也可以處理最外層的索引。

Stacking 和Unstacking 也可以運用到單層索引的數據集中，如下圖所示：



參考文獻

1 Pandas中的數據重塑(reshape)功能

2 Reshaping in Pandas - Pivot, Pivot-Table, Stack and Unstack explained with Pictures

3. python pandas stack和unstack函數stack和unstack這篇的講解不錯