详解 Pandas 透视表（pivot_table）

2015/03/25 · 工具与框架, 框架 · 9 评论 · Pandas

本文由 伯乐在线 - PyPer 翻译，Daetalus 校稿。未经许可，禁止转载！
英文出处：pbpython。欢迎加入翻译组。

介绍

也许大多数人都有在Excel中使用数据透视表的经历，其实Pandas也提供了一个类似的功能，名为pivot_table。虽然pivot_table非常有用，但是我发现为了格式化输出我所需要的内容，经常需要记住它的使用语法。所以，本文将重点解释pandas中的函数 pivot_table，并教大家如何使用它来进行数据分析。

如果你对这个概念不熟悉，维基百科上对它做了详细的解释。顺便说一下，你知道微软为PivotTable（透视表）注册了商标吗？其实以前我也不知道。不用说，下面我将讨论的透视表并不是PivotTable。

作为一个额外的福利，我创建了一个总结pivot_table的简单备忘单。你可以在本文的最后找到它，我希望它能够对你有所帮助。如果它帮到了你，请告诉我。

数据

使用pandas中pivot_table的一个挑战是，你需要确保你理解你的数据，并清楚地知道你想通过透视表解决什么问题。其实，虽然pivot_table看起来只是一个简单的函数，但是它能够快速地对数据进行强大的分析。

一下“企业软件”、“资本设备”等），而管理者想更详细地了解它一整年的情况。

典型的问题包括：

- 本渠道收入是多少？
- 渠道的产品是什么？
- 谁在什么阶段有什么产品？
- 我们年底前结束交易的可能性有多大？

很多公司将会使用CRM工具或者其他销售使用的软件来跟踪此过程。虽然他们可能拥有有效的工具对数据进行分析，但肯定有人需要将数据导出到Excel，并使用一个透视表工具来总结这些数据。

使用Pandas透视表将是一个不错的选择，应为它有以下优点：

- 更快（一旦设置之后）
- 自行说明（通过查看代码，你将知道它做了什么）
- 易于生成报告或电子邮件
- 更灵活，因为你可以定义定制的聚合函数

Read in the data

首先，让我们搭建所需的环境。

如果你想跟随我继续下去，那么可以下载这个Excel文件。

Python

```
import pandas as pd
import numpy as np
```

版本提醒

因为Pivot_table API已经随着时间有所改变，所以为了使本文中示例代码能够正常工作，请确保你安装了最近版本的Pandas（>0.15）。本文示例还用到了category数据类型，而它也需要确保是最近版本。

首先，将我们销售渠道的数据读入到数据帧中。

Python

```
df = pd.read_excel("../in/sales-funnel.xlsx")
df.head()
```

	Account	Name	Rep	Manager	Product	Quantity	Price	Status
0	714466	Trantow-Barrows	Craig Booker	Debra Henley	CPU	1	30000	presented
1	714466	Trantow-Barrows	Craig Booker	Debra Henley	Software	1	10000	presented
2	714466	Trantow-Barrows	Craig Booker	Debra Henley	Maintenance	2	5000	pending
3	737550	Fritsch, Russel and Anderson	Craig Booker	Debra Henley	CPU	1	35000	declined
4	146832	Kiehn-Spinka	Daniel Hilton	Debra Henley	CPU	2	65000	won

其实，并不严格要求这样做，但这样做能够在分析数据的整个过程中，帮助我们保持所想要的顺序。

```python
df["Status"] = df["Status"].astype("category")
df["Status"].cat.set_categories(["won","pending","presented","declined"],inplace=True)
```

处理数据

既然我们建立数据透视表，我觉得最容易的方法就是一步一个脚印地进行。添加项目和检查每一步来验证你正一步一步得到期望的结果。为了查看什么样的外观最能满足你的需要，就不要害怕处理顺序和变量的繁琐。

最简单的透视表必须有一个数据帧和一个索引。在本例中，我们将使用“Name（名字）”列作为我们的索引。

```python
pd.pivot_table(df,index=["Name"])
```

	Account	Price	Quantity
Name			
Barton LLC	740150	35000	1.000000
Fritsch, Russel and Anderson	737550	35000	1.000000
Herman LLC	141962	65000	2.000000
Jerde-Hilpert	412290	5000	2.000000
Kassulke, Ondricka and Metz	307599	7000	3.000000
Keeling LLC	688981	100000	5.000000
Kiehn-Spinka	146832	65000	2.000000
Koepp Ltd	729833	35000	2.000000
Kulas Inc	218895	25000	1.500000
Purdy-Kunde	163416	30000	1.000000
Stokes LLC	239344	7500	1.000000
Trantow-Barrows	714466	15000	1.333333

此外，你也可以有多个索引。实际上，大多数的pivot_table参数可以通过列表获取多个值。

```python
pd.pivot_table(df,index=["Name","Rep","Manager"])
```

Name	Rep	Manager			
Barton LLC	John Smith	Debra Henley	740150	35000	1.000000
Fritsch, Russel and Anderson	Craig Booker	Debra Henley	737550	35000	1.000000
Herman LLC	Cedric Moss	Fred Anderson	141962	65000	2.000000
Jerde-Hilpert	John Smith	Debra Henley	412290	5000	2.000000
Kassulke, Ondricka and Metz	Wendy Yule	Fred Anderson	307599	7000	3.000000
Keeling LLC	Wendy Yule	Fred Anderson	688981	100000	5.000000
Kiehn-Spinka	Daniel Hilton	Debra Henley	146832	65000	2.000000
Koepp Ltd	Wendy Yule	Fred Anderson	729833	35000	2.000000
Kulas Inc	Daniel Hilton	Debra Henley	218895	25000	1.500000
Purdy-Kunde	Cedric Moss	Fred Anderson	163416	30000	1.000000
Stokes LLC	Cedric Moss	Fred Anderson	239344	7500	1.000000
Trantow-Barrows	Craig Booker	Debra Henley	714466	15000	1.333333

这样很有趣但并不是特别有用。我们可能想做的是通过将“Manager”和“Rep”设置为索引来查看结果。要实现它其实很简单，只需要改变索引就可以。

Python

```
1 pd.pivot_table(df,index=["Manager","Rep"])
```

		Account	Price	Quantity
Manager	Rep			
Debra Henley	Craig Booker	720237.0	20000.000000	1.250000
	Daniel Hilton	194874.0	38333.333333	1.666667
	John Smith	576220.0	20000.000000	1.500000
Fred Anderson	Cedric Moss	196016.5	27500.000000	1.250000
	Wendy Yule	614061.5	44250.000000	3.000000

可以看到，透视表比较智能，它已经开始通过将“Rep”列和“Manager”列进行对应分组，来实现数据聚合和总结。那么现在，就让我们共同看一下数据透视表可以为我们做些什么吧。

为此，“Account”和“Quantity”列对于我们来说并没什么用。所以，通过利用“values”域显式地定义我们关心的列，就可以实现移除那些不关心的列。

Python

```
1 pd.pivot_table(df,index=["Manager","Rep"],values=["Price"])
```

Manager	Rep	
Debra Henley	Craig Booker	20000
	Daniel Hilton	38333
	John Smith	20000
Fred Anderson	Cedric Moss	27500
	Wendy Yule	44250

“Price”列会自动计算数据的平均值，但是我们也可以对该列元素进行计数或求和。要添加这些功能，使用aggfunc和np.sum就很容易实现。

```python
pd.pivot_table(df,index=["Manager","Rep"],values=["Price"],aggfunc=np.sum)
```

		Price
Manager	Rep	
Debra Henley	Craig Booker	80000
	Daniel Hilton	115000
	John Smith	40000
Fred Anderson	Cedric Moss	110000
	Wendy Yule	177000

aggfunc可以包含很多函数，下面就让我们尝试一种方法，即使用numpy中的函数mean和len来进行计数。

```python
pd.pivot_table(df,index=["Manager","Rep"],values=["Price"],aggfunc=[np.mean,len])
```

		mean	len
		Price	Price
Manager	Rep		
Debra Henley	Craig Booker	20000	4
	Daniel Hilton	38333	3
	John Smith	20000	2
Fred Anderson	Cedric Moss	27500	4
	Wendy Yule	44250	4

如果我们想通过不同产品来分析销售情况，那么变量“columns”将允许我们定义一个或多个列。

列vs.值

性，变量“columns（列）”是可选的，它提供一种额外的方法来分割你所关心的实际值。然而，聚合函数aggfunc最后是被应用到了变量“values”中你所列举的项目上。

```python
pd.pivot_table(df,index=["Manager","Rep"],values=["Price"],
               columns=["Product"],aggfunc=[np.sum])
```

		sum			
		Price			
	Product	CPU	Maintenance	Monitor	Software
Manager	Rep				
Debra Henley	Craig Booker	65000	5000	NaN	10000
	Daniel Hilton	105000	NaN	NaN	10000
	John Smith	35000	5000	NaN	NaN
Fred Anderson	Cedric Moss	95000	5000	NaN	10000
	Wendy Yule	165000	7000	5000	NaN

然而，非数值（NaN）有点令人分心。如果想移除它们，我们可以使用“fill_value”将其设置为0。

```python
pd.pivot_table(df,index=["Manager","Rep"],values=["Price"],
               columns=["Product"],aggfunc=[np.sum],fill_value=0)
```

		sum			
		Price			
	Product	CPU	Maintenance	Monitor	Software
Manager	Rep				
Debra Henley	Craig Booker	65000	5000	0	10000
	Daniel Hilton	105000	0	0	10000
	John Smith	35000	5000	0	0
Fred Anderson	Cedric Moss	95000	5000	0	10000
	Wendy Yule	165000	7000	5000	0

其实，我觉得添加“Quantity”列将对我们有所帮助，所以将“Quantity”添加到“values”列表中。

```python
pd.pivot_table(df,index=["Manager","Rep"],values=["Price","Quantity"],
               columns=["Product"],aggfunc=[np.sum],fill_value=0)
```

		Price				Quantity			
	Product	CPU	Maintenance	Monitor	Software	CPU	Maintenance	Monitor	Software
Manager	Rep								
Debra Henley	Craig Booker	65000	5000	0	10000	2	2	0	1
	Daniel Hilton	105000	0	0	10000	4	0	0	1
	John Smith	35000	5000	0	0	1	2	0	0
Fred Anderson	Cedric Moss	95000	5000	0	10000	3	1	0	1
	Wendy Yule	165000	7000	5000	0	7	3	2	0

有趣的是，你可以将几个项目设置为索引来获得不同的可视化表示。下面的代码中，我们将“Product”从“columns”中移除，并添加到“index”变量中。

Python

```
pd.pivot_table(df,index=["Manager","Rep","Product"],
               values=["Price","Quantity"],aggfunc=[np.sum],fill_value=0)
```

			sum	
			Price	Quantity
Manager	Rep	Product		
Debra Henley	Craig Booker	CPU	65000	2
		Maintenance	5000	2
		Software	10000	1
	Daniel Hilton	CPU	105000	4
		Software	10000	1
	John Smith	CPU	35000	1
		Maintenance	5000	2
Fred Anderson	Cedric Moss	CPU	95000	3
		Maintenance	5000	1
		Software	10000	1
	Wendy Yule	CPU	165000	7
		Maintenance	7000	3
		Monitor	5000	2

对于这个数据集，这种显示方式看起来更有意义。不过，如果我想查看一些总和数据呢？“margins=True”就可以为我们实现这种功能。

```
pd.pivot_table(df,index=["Manager","Rep","Product"],
               values=["Price","Quantity"],
               aggfunc=[np.sum,np.mean],fill_value=0,margins=True)
```

			sum		mean	
			Price	Quantity	Price	Quantity
Manager	Rep	Product				
Debra Henley	Craig Booker	CPU	65000	2	32500.000000	1.000000
		Maintenance	5000	2	5000.000000	2.000000
		Software	10000	1	10000.000000	1.000000
	Daniel Hilton	CPU	105000	4	52500.000000	2.000000
		Software	10000	1	10000.000000	1.000000
	John Smith	CPU	35000	1	35000.000000	1.000000
		Maintenance	5000	2	5000.000000	2.000000
Fred Anderson	Cedric Moss	CPU	95000	3	47500.000000	1.500000
		Maintenance	5000	1	5000.000000	1.000000
		Software	10000	1	10000.000000	1.000000
	Wendy Yule	CPU	165000	7	82500.000000	3.500000
		Maintenance	7000	3	7000.000000	3.000000
		Monitor	5000	2	5000.000000	2.000000
All			522000	30	30705.882353	1.764706

下面，让我们以更高的管理者角度来分析此渠道。根据我们前面对category的定义，注意现在“Status”是如何排序的。

Python

```
pd.pivot_table(df,index=["Manager","Status"],values=["Price"],
               aggfunc=[np.sum],fill_value=0,margins=True)
```

		Price
Manager	Status	
Debra Henley	declined	70000
	pending	50000
	presented	50000
	won	65000
Fred Anderson	declined	65000
	pending	5000
	presented	45000
	won	172000
All		522000

一个很方便的特性是，为了对你选择的不同值执行不同的函数，你可以向aggfunc传递一个字典。不过，这样做有一个副作用，那就是必须将标签做的更加简洁才行。

Python

```
pd.pivot_table(df,index=["Manager","Status"],columns=["Product"],values=["Quantity","P
               aggfunc={"Quantity":len,"Price":np.sum},fill_value=0)
```

		Price				Quantity			
	Product	CPU	Maintenance	Monitor	Software	CPU	Maintenance	Monitor	Software
Manager	Status								
Debra Henley	declined	70000	0	0	0	2	0	0	0
	pending	40000	10000	0	0	1	2	0	0
	presented	30000	0	0	20000	1	0	0	2
	won	65000	0	0	0	1	0	0	0
Fred Anderson	declined	65000	0	0	0	1	0	0	0
	pending	0	5000	0	0	0	1	0	0
	presented	30000	0	5000	10000	1	0	1	1
	won	165000	7000	0	0	2	1	0	0

此外，你也可以提供一系列的聚合函数，并将它们应用到“values”中的每个元素上。

Python

```
table = pd.pivot_table(df,index=["Manager","Status"],columns=["Product"],values=["Quar
               aggfunc={"Quantity":len,"Price":[np.sum,np.mean]},fill_value=0)
table
```

		mean				sum				
	Product	CPU	Maintenance	Monitor	Software	CPU	Maintenance	Monitor	Software	
Manager	Status									
Debra Henley	declined	35000	0	0	0	70000	0	0	0	2
	pending	40000	5000	0	0	40000	10000	0	0	1
	presented	30000	0	0	10000	30000	0	0	20000	1
	won	65000	0	0	0	65000	0	0	0	1
Fred Anderson	declined	65000	0	0	0	65000	0	0	0	1
	pending	0	5000	0	0	0	5000	0	0	0
	presented	30000	0	5000	10000	30000	0	5000	10000	1
	won	82500	7000	0	0	165000	7000	0	0	2

也许，同一时间将这些东西全都放在一起会有点令人望而生畏，但是一旦你开始处理这些数据，并一步一步地添加新项目，你将能够领略到它是如何工作的。我一般的经验法则是，一旦你使用多个“grouby”，那么你需要评估此时使用透视表是否是一种好的选择。

高级透视表过滤

一旦你生成了需要的数据，那么数据将存在于数据帧中。所以，你可以使用自定义的标准数据帧函数来对其进行过滤。

如果你只想查看一个管理者（例如Debra Henley）的数据，可以这样：

Python

```
1 table.query('Manager == ["Debra Henley"]')
```

		Price								
		mean				sum				
	Product	CPU	Maintenance	Monitor	Software	CPU	Maintenance	Monitor	Software	
Manager	Status									
Debra Henley	declined	35000	0	0	0	70000	0	0	0	2
	pending	40000	5000	0	0	40000	10000	0	0	1
	presented	30000	0	0	10000	30000	0	0	20000	1
	won	65000	0	0	0	65000	0	0	0	1

我们可以查看所有的暂停（pending）和成功（won）的交易，代码如下所示：

Python

```
1 table.query('Status == ["pending","won"]')
```

		mean				sum				len
	Product	CPU	Maintenance	Monitor	Software	CPU	Maintenance	Monitor	Software	CP
Manager	Status									
Debra Henley	pending	40000	5000	0	0	40000	10000	0	0	1
	won	65000	0	0	0	65000	0	0	0	1
Fred Anderson	pending	0	5000	0	0	0	5000	0	0	0
	won	82500	7000	0	0	165000	7000	0	0	2

这是pivot_table中一个很强大的特性，所以一旦你得到了你所需要的pivot_table格式的数据，就不要忘了此时你就拥有了pandas的强大威力。

如果你想将其保存下来作为参考，那么这里提供完整的笔记：
http://nbviewer.ipython.org/url/pbpython.com/extras/Pandas-Pivot-Table-Explained.ipynb

数据下载地址：http://pbpython.com/extras/sales-funnel.xlsx

备忘单

为了试图总结所有这一切，我已经创建了一个备忘单，我希望它能够帮助你记住如何使用pandas的pivot_table。

	Account	Name	Rep	Manager	Product	Quantity	Price	Status
0	714466	Trantow-Barrows	Craig Booker	Debra Henley	CPU	1	30000	presented
1	714466	Trantow-Barrows	Craig Booker	Debra Henley	Software	1	10000	presented
2	714466	Trantow-Barrows	Craig Booker	Debra Henley	Maintenance	2	5000	pending
3	737550	Fritsch, Russel and Anderson	Craig Booker	Debra Henley	CPU	1	35000	declined
4	146832	Kiehn-Spinka	Daniel Hilton	Debra Henley	CPU	2	65000	won

```
pd.pivot_table(df,
    index=["Manager","Status"],
    columns=["Product"],
    aggfunc=[np.sum],
    values=["Price"],
    fill_value=0,
    margins=True,
    dropna=True)
```

Can also use a dictionary:
aggfunc={"Quantity":len,
"Price":[np.sum,np.mean]}

		sum				
		Price				
	Product	CPU	Maintenance	Monitor	Software	All
Manager	Status					
Debra Henley	declined	70000	0	0	0	70000
	pending	40000	10000	0	0	50000
	presented	30000	0	0	20000	50000
	won	65000	0	0	0	65000
Fred Anderson	declined	65000	0	0	0	65000
	pending	0	5000	0	0	5000
	presented	30000	0	5000	10000	45000
	won	165000	7000	0	0	172000
All		465000	22000	5000	30000	522000

pbpython.com

关于作者：PyPer

一名就读于羊城某高校的学生，主要关注 Python、PowerShell等脚本技术，以及SQL Server、MySQL、MongoDB、Redis等数据库，新浪微博：http://weibo.com/LIwianwpIO，微信公众号“NETEC”。