Visualizing PROC TRANSPOSE

Daniel Boisvert, Genzyme Corporation, Cambridge MA
Shafi Chowdhury, Shafi Consultancy, London England

Abstract

PROC TRANSPOSE continues to confuse programmers. The ability to effectively transpose a data set is very important when working with different data structures and different data standards. This paper will provide a non technical approach to understanding the transpose procedure by showing the programmer how to visualize the expected output. PROC TRANSPOSE will be deconstructed into three simple movements: what goes up, what goes down, and what goes into the middle. Programmers who have a hard time fully grasping PROC TRANSPOSE will benefit from this paper.

Introduction

How do I know I need to transpose?

If you ever catch yourself saying, "It'd be a lot easier if these observations were next to each other", or "It'd be a lot easier if these variables were observations one underneath each other", you need to transpose. Transposes are limited to these two types, and a combination of these two. "It'd be a lot easier if these observations were next to each other", describes an "up" transpose. "Up" transposes can be thought of changing rows of a dataset to columns, or changing a vertical data structure to a horizontal data structure. "It'd be a lot easier if these variables were observations one underneath each other", describes a "down" transpose. "Down" transposes can be though of as changing columns to rows, or changing a horizontal data structure to a vertical data structure.

After having determined whether your transpose is an "Up" or "Down", you need to ask yourself four questions.

1. What should stay the same?
2. What goes up?
3. What goes down?
4. What goes into the middle

Answering these questions will lead us to build our PROC TRANSPOSE syntax.

Visualization

Once you have answered these questions you will be able to easily visualize the transpose. In practice, print out a page of your dataset and draw the movements as I describe in the following example. If while drawing, you realize you cannot visualize any of the movements or the outcome of your transpose, you shouldn't be transposing. This will save you a lot of time. However, if you can visualize it, you'll know that the transpose is valid, and you'll also know what to expect as output. In the following example I will draw the movements on a dataset to exemplify how this process works.

Example

Using the following data set (Figure 1), imagine that we wanted to calculate (VAL1+VAL3)/VAL2 for every patient at each visit for both SOMEVAL and SOMEVAL2. It'd be difficult, right? This is a classic example for an "Up and Down" transpose. By transposing VERTVAR up, and SOMEVAL and SOMEVAL2 down, it will be easier to perform our calculation. But, before we get ahead of ourselves, let's go back to the questions and look at them in more detail.

(Figure 1)

```
 USUBJID     VISIT   VERTVAR     SOMEVAL     SOMEVAL2

    1          1       VAL1      48.6038      88.6349
    1          1       VAL2       5.2829      56.6006
    1          1       VAL3       8.6694      28.9123
    1          2       VAL1      39.2996      53.0151
    1          2       VAL2      97.3495      16.6469
    1          2       VAL3      30.7158      74.9157
    1          3       VAL1      68.0895      29.7941
    1          3       VAL2       8.7368      57.3068
    1          3       VAL3      95.5423      61.5186
    2          1       VAL1      15.9667      68.7324
    2          1       VAL2      81.9658      55.4981
    2          1       VAL3      18.1096      96.8633
    2          2       VAL1       7.8062      93.1806
    2          2       VAL2      16.8828      30.9221
    2          2       VAL3      69.9848      51.4851
    2          3       VAL1       5.5580      30.0224
    2          3       VAL2      40.8118      84.3604
    2          3       VAL3      33.0362      90.6971
```

Question 1: What should stay the same?
Answer 1: The BY statement
In your data set, there should be variables that you want to keep in your output data set. Variables that need to be kept in their original structure should be specified in the BY statement. If you are doing an "Up" transpose, these variables need to identify unique values of your "Up" variable. In Figure 1, these variables are USUBJID and VISIT. We can think of the rest of the transpose as rotating the data set around these variables.
Syntax 1:

```
PROC TRANSPOSE DATA=fig_1 OUT=tranfig_1;
  BY usubjid visit;
```

Question 2: What goes up?
Answer 2: The ID statement
We've already determined that there is an "Up" element to this transpose, and that VERTVAR is going up. But what does this mean? This means that the individual values of VERTVAR are to be turned into variables themselves. One variable for each unique value of VERTVAR will be created.
Syntax 2:

```
 PROC TRANSPOSE DATA=fig_1 OUT=tranfig_1;
```

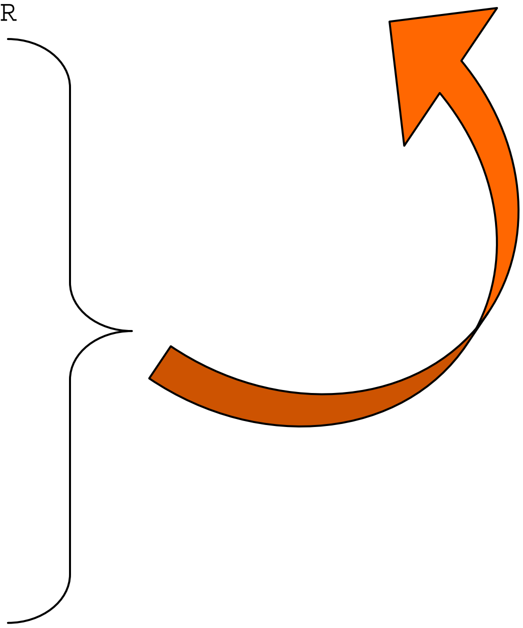
```
   BY usubjid visit;
   ID vertvar;
```

Visualization – Up Movement:

(Figure 2)

```
USUBJID    VISIT    VERTVAR

     1         1       VAL1
     1         1       VAL2
     1         1       VAL3
     1         2       VAL1
     1         2       VAL2
     1         2       VAL3
     1         3       VAL1
     1         3       VAL2
     1         3       VAL3
     2         1       VAL1
     2         1       VAL2
     2         1       VAL3
     2         2       VAL1
     2         2       VAL2
     2         2       VAL3
     2         3       VAL1
     2         3       VAL2
     2         3       VAL3
```



(Figure 3)

```
USUBJID    VISIT    VAL1    VAL2     VAL3

     1         1
     1         2
     1         3
     2         1
     2         2
     2         3
```

Summary 2: Figure 2 shows the data set before the transpose, and illustrates, by way of the arrow, how the values of the ID variable move up to create new variables. In Figure 3, we see the three new variables that have hence been created (VAL1,VAL2, VAL3). Also, note that the number of observations in the data set has been decreased now that VAL1, VAL2, VAL3 do not each need a separate record.

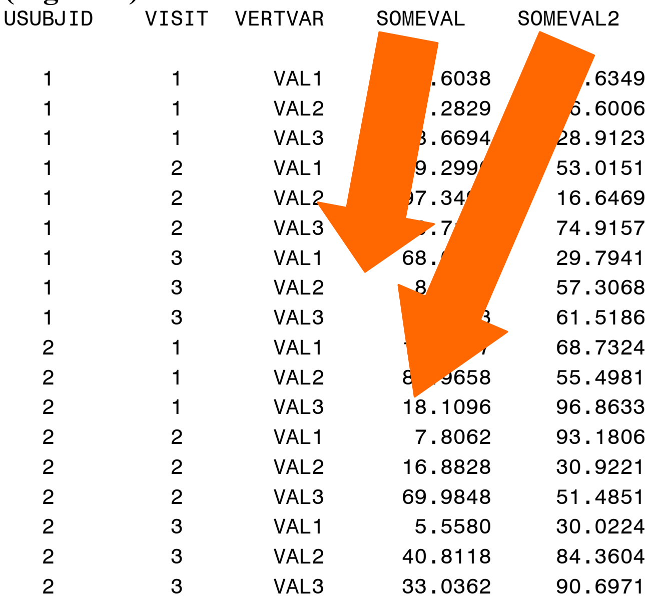
Question 3: What goes down?
Answer 3: The VAR statement
The VAR statement performs two distinct movements: Down, and Middle. Here we will concentrate on what goes down. As mentioned before, SOMEVAL and SOMEVAL2 are our "Down". Note that it is not the values that move down, but rather the variable name. (S-O-M-E-V-A-L). SOMEVAL and SOMEVAL2, become the values of the newly created variable _NAME_. When transposing down like this, it is a good practice to rename _NAME_ into something more meaningful. To do this, we use the NAME= option.

Syntax 3:

```
PROC TRANSPOSE DATA=fig_1 OUT=tranfig_1 NAME=downvar;
  BY usubjid visit;
  ID vertvar;
  VAR someval someval2;
RUN;
```

Visualization – Down Movement:

(Figure 4)



```
USUBJID     VISIT   VERTVAR     SOMEVAL     SOMEVAL2

    1          1        VAL1       .6038        .6349
    1          1        VAL2       .2829       6.6006
    1          1        VAL3      3.6694      28.9123
    1          2        VAL1      9.299       53.0151
    1          2        VAL2     97.34        16.6469
    1          2        VAL3       .7         74.9157
    1          3        VAL1     68.          29.7941
    1          3        VAL2      8           57.3068
    1          3        VAL3           3      61.5186
    2          1        VAL1     7       7    68.7324
    2          1        VAL2     8  .9658     55.4981
    2          1        VAL3     18.1096      96.8633
    2          2        VAL1      7.8062      93.1806
    2          2        VAL2     16.8828      30.9221
    2          2        VAL3     69.9848      51.4851
    2          3        VAL1      5.5580      30.0224
    2          3        VAL2     40.8118      84.3604
    2          3        VAL3     33.0362      90.6971
```

(Figure 5)

```
USUBJID     VISIT      DOWNVAR        VAL1        VAL2        VAL3

     1          1      SOMEVAL
     1          1      SOMEVAL2
     1          2      SOMEVAL
     1          2      SOMEVAL2
     1          3      SOMEVAL
     1          3      SOMEVAL2
     2          1      SOMEVAL
     2          1      SOMEVAL2
     2          2      SOMEVAL
     2          2      SOMEVAL2
     2          3      SOMEVAL
     2          3      SOMEVAL2
```

Summary 3:

Figure 4 shows the data set in its original state before any transposing has occurred, while Figure 5 illustrates what happens with the combination of the ID and VAR statements. In real time, all of these movements occur at the same time, but here I have slowed that time down to take a better look at it. In Figure 5, the number of observations has doubled from Figure 3; now that we have 2 observations per USUBJID, per VISIT. This is the final structure of our transposed data set.

Question 4: What goes into the middle?

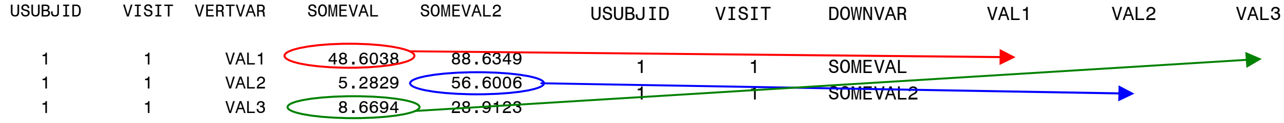
Answer 4: The VAR statement
Now that we have the final structure of our output data set, we need only to fill in the values of the newly created variables. As mentioned, the VAR statement has two movements. Middle, is the second of these. Using the same syntax as above, the values of SOMEVAL and SOMEVAL2 are pushed into the new variables VAL1, VAL2, VAL3.

Syntax 4:

```
PROC TRANSPOSE DATA=fig_1 OUT=tranfig_1;
  BY usubjid visit;
  ID vertvar;
  VAR someval;
RUN;
```

Visualization – Middle Movement

(Figure 6)



Original Data Set

```
USUBJID   VISIT  VERTVAR    SOMEVAL    SOMEVAL2

   1        1       VAL1     48.6038    88.6349
   1        1       VAL2      5.2829    56.6006
   1        1       VAL3      8.6694    28.9123
```

New Data Set

```
USUBJID    VISIT     DOWNVAR        VAL1       VAL2       VAL3

   1         1       SOMEVAL
   1         1       SOMEVAL2
```

(Figure 7)

```
USUBJID     VISIT     DOWNVAR         VAL1        VAL2        VAL3

    1          1      SOMEVAL       48.6038      5.2829      8.6694
    1          1      SOMEVAL2      88.6349     56.6006     28.9123
```

Summary 3 : Et voila, our transpose is complete. The values of SOMEVAL and SOMEVAL2 have moved into the middle of our newly created variables (from the ID statement) while our BY variables (USUBJID, VISIT) have stayed put. It is important to note that the integrity of our data set is still intact. The values of the cross between SOMEVAL and VAL1 for USUBJID=1 VISIT=1 is the same in the starting data set (Figure 1) as it is in the output data set (Figure 7). With our data in this new form we can easily do our calculation of (VAL1+VAL3)/VAL2.

Conclusion

Planning, drawing and visualizing your transpose will help you understand the inner workings of PROC TRANSPOSE. Before transposing, answering the above questions will lead you towards consistent and correct transpositions. Harnessing the power of PROC TRANSPOSE will allow you to rethink data structures and lead to more efficient programming.

Aknowledgements

Special thanks to Shafi Chowdhury for teaching me all of this.

Contact Information

You comments and questions are very much encouraged.

Daniel Boisvert
Email: daniel.boisvert (at) genzyme (dot) com

Shafi Chowdhury
Email: Shafi@shaficonsultancy.com
Website: www.shaficonsultancy.com