Paper 060-2009

Learn the Basics of Proc Transpose

Douglas Zirbel, Wells Fargo and Co., St. Louis, Missouri

ABSTRACT

PROC TRANSPOSE is a powerful yet underutilized PROC in the Base SAS® toolset. This paper presents an easy before-and-after approach to learning PROC TRANSPOSE. It contains three sample SAS® input files, a set of basic PROC TRANSPOSE variations, and their output results. There is a Summary Sheet at the end of the paper as well for later reference. By going through these few exercises, you should be ready to use PROC TRANSPOSE whenever you need it.

INTRODUCTION

Here are two situations programmers sometimes face. One – data for a single subject is stored in multiple rows. For example, balance amounts for a single customer is found in 4 different, "narrow" rows, each row containing the balance for one account:

Cust	account	balance
Smith	checking	$1,000.00
Smith	savings	$4,000.00
Smith	mortgage	$150,000.00
Smith	credit_card	$500.00
Jones	checking	$973.78
Jones	savings	$2,613.44
Jones	mortgage	.
Jones	credit_card	$140.48

For your purposes, it would be easier to accomplish your task if all accounts and balances were "wide" -- on 1 line:

Cust	checking	savings	mortgage	credit_card
Smith	$1,000.00	$4,000.00	$150,000.00	$500.00
Jones	$973.78	$2,613.44	.	$140.48

Another situation is the opposite: "wide" data

Patient	visit	HR	PR	QT	QRS	QTCb	Comments
Smith	1	80	200	250	300	310	Normal ECG
Jones	1	70	190	220	290	300	Sinus bradycardia

that you would like to have in a narrow file:

Patient	visit	measurement	value
Smith	1	HR	80
Smith	1	PR	200
Smith	1	QT	250
Smith	1	QRS	300
Smith	1	QTCb	310
Smith	1	Comments	Normal ECG
Jones	1	HR	70
Jones	1	PR	190
Jones	etc…		

If you are not familiar with the behavior of Proc Transpose, but you are fluent in the Data Step, you will figure out a way to handle these situations. Yet, Proc Transpose will usually deliver your output with far less programming time than the Data Step approach.

The following is a) a step-by-step tutorial on Proc Transpose basics, and b) a quick reference sheet to which you can return for help later. There are three lessons. Each lesson starts with a small input SAS file followed by several Proc Transpose variations and their resulting output, and a few notes of explanation.

You can comprehend and become adept with Proc Transpose in about 30 minutes of practice with this paper, a SAS session, and SAS documentation at hand. The code to create the input datasets is provided in the appendix. Do the exercises yourself. Answer the questions. Let's begin.

LESSON 1: NARROW FILE AND RESULTING TRANSPOSED WIDE FILES

Small, narrow input file

*** File: narrow_file1			
Obs	**pet_ owner**	**pet**	**population**
1	Mr. Black	dog	2
2	Mr. Black	bird	1
3	Mrs. Green	fish	5
4	Mr. White	cat	3

A) Simple transpose

```
proc transpose data=work.narrow_file1
               out=work.narrow_file1_transp_default;
run;
```

Result

*** File: narrow_file1_transp_default					
Obs	**_NAME_**	**COL1**	**COL2**	**COL3**	**COL4**
1	population	2	1	5	3

Question: do you see how data in rows is transposed to become data in columns?

Note: unless you tell it otherwise, only numeric variables are transposed. Go back and look at the input file – the other, non-numeric, variables, including pet_owner and pet, are ignored. The input file could just as easily have contained only 1 column, "population".

Question: without looking at anything else, can you tell what COL1 represents?

B) PREFIX option

```
proc transpose data=work.narrow_file1
               out=work.narrow_file1_transp_prefix
               prefix=pet_count;
run;
```

Result

*** File: narrow_file1_transp_prefix					
Obs	**_NAME_**	**pet_count1**	**pet_count2**	**pet_count3**	**pet_count4**
1	population	2	1	5	3

Question: what did the prefix option do?

C) NAME option

```
proc transpose data=work.narrow_file1
               out=work.narrow_file1_transp_prefix_name
               name=column_that_was_transposed
               prefix=pet_count;
run;
```

Result

*** File: narrow_file1_transp_prefix_name					
Obs	**column_ that_was_ transposed**	**pet_ count1**	**pet_ count2**	**pet_ count3**	**pet_ count4**
1	population	2	1	5	3

Question: what did the name option do?

D) ID statement

```
proc transpose data=work.narrow_file1
               out=work.narrow_file1_transp_id
               name=column_that_was_transposed;
    id pet;
run;
```

Result

*** File: narrow_file1_transp_id					
Obs	**column_ that_was_ transposed**	**dog**	**bird**	**fish**	**cat**
1	population	2	1	5	3

Question: what did the ID statement do?

Question: why was the prefix option not needed?

E) VAR statement

```
proc transpose data=work.narrow_file1
    out=work.narrow_file1_transp_var;
    var pet population;
run;
```

Result

*** File: narrow_file1_transp_var					
Obs	_NAME_	COL1	COL2	COL3	COL4
1	pet	dog	bird	fish	cat
2	population	2	1	5	3

Question: Transpose transposes numeric vars by default; why was a numeric and a character variable transposed here?

F) VAR and ID statements

```
proc transpose data=work.narrow_file1
               out=work.narrow_file1_transp_id_var
               name=column_that_was_transposed;
    var pet population;
    id pet;
run;
```

Result

*** File: narrow_file1_transp_id_var

Obs	column_ that_was_ transposed	dog	bird	fish	cat
1	pet	dog	bird	fish	cat
2	population	2	1	5	3

Question: what's the difference between this output and the preceding output?

LESSON 2: NARROW FILE (MANY ROWS) AND RESULTING TRANSPOSED WIDE FILES

Longer, narrow input file

*** File: narrow_file2

Obs	pet_ owner	pet	population
1	Mr. Black	dog	2
2	Mr. Black	cat	1
3	Mrs. Brown	dog	1
4	Mrs. Brown	cat	0
5	Mrs. Green	fish	5
6	Mr. White	fish	7
7	Mr. White	dog	1
8	Mr. White	cat	3

G) Simple transpose

```
proc transpose data=work.narrow_file2
               out=work.narrow_file2_transp_default;
run;
```

Result

*** File: narrow_file2_transp_default

Obs	_NAME_	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8
1	population	2	1	1	0	5	7	1	3

Question: without looking at anything else, can you tell what the COLs represent?

H) VAR statement

```
proc transpose data=work.narrow_file2
                out=work.narrow_file2_transp_var
               name=column_that_was_transposed;
    var pet population;
run;
```

Result

*** File: narrow_file2_transp_var

Obs	column_ that_was_ transposed	COL1	COL2	COL3	COL4	COL	COL6	COL7	COL8
1	pet	dog	cat	dog	cat	fish	fish	dog	cat
2	population	2	1	1	0	5	7	1	3

I) BY statement

```
proc sort data=work.narrow_file2
          out= work.sorted_narrow_file2;
    by pet_owner;
run;

proc transpose data=work.sorted_narrow_file2
               out=work.narrow_file2_transp_by
               name=column_that_was_transposed;
    by pet_owner;
run;
```

Result

*** File: narrow_file2_transp_by

Obs	pet_owner	column_ that_was_ transposed	COL1	COL2	COL3
1	Mr. Black	population	2	1	.
2	Mr. White	population	7	1	3
3	Mrs. Brown	population	1	0	.
4	Mrs. Green	population	5	.	.

Question: Can you tell what the COLs represent? Hint: it may not be what you expect. Take a look at the input file

J) BY and ID statements

```
proc transpose data=work.sorted_narrow_file2
                out=work.narrow_file2_transp_id_by
               name=column_that_was_transposed;
    by pet_owner;
    id pet;
run;
```

Result

*** File: narrow_file2_transp_id_by

Obs	pet_owner	column_ that_was_ transposed	dog	cat	fish
1	Mr. Black	population	2	1	.
2	Mr. White	population	1	3	7
3	Mrs. Brown	population	1	0	.
4	Mrs. Green	population	.	.	5

Question: what happened to the transposed columns?

LESSON 3: WIDE FILE AND RESULTING TRANSPOSED NARROW FILES

Wide input file

*** File: wide_file3					
Obs	pet_ owner	cat	dog	fish	bird
1	Mr. Black	1	2	.	0
2	Mr. Brown	0	1	0	1
3	Mrs. Green	.	0	5	.
4	Mrs. White	3	1	7	2

K) Simple transpose

```
proc transpose data=work.wide_file3
               out=work.wide_file3_transp_default;
run;
```

Result

*** File: wide_file3_transp_default					
Obs	_NAME_	COL1	COL2	COL3	COL4
1	cat	1	0	.	3
2	dog	2	1	0	1
3	fish	.	0	5	7
4	bird	0	1	.	2

L) NAME and PREFIX options

```
proc transpose data=work.wide_file3
               out=work.wide_file3_transp_name_prefix
               name=column_that_was_transposed
               prefix=pet_count;
run;
```

Result

*** File: wide_file3_transp_name_prefix					
Obs	column_that_was_transposed	pet_ count1	pet_ count2	pet_ count3	pet_ count4
1	cat	1	0	.	3
2	dog	2	1	0	1
3	fish	.	0	5	7
4	bird	0	1	.	2

M) ID statement

```
proc transpose data=work.wide_file3
               out=work.wide_file3_transp_id
               name=column_that_was_transposed;
    id pet_owner;
run;
```

Result

*** File: wide_file3_transp_id					
Obs	column_that_was_transposed	Mr__ Black	Mrs__ Brown	Mrs__ Green	Mr__ White
1	cat	1	0	.	3
2	dog	2	1	0	1
3	fish	.	0	5	7
4	bird	0	1	.	2

Quick Reference Sheet – Proc Transpose

- Proc Transpose changes multiple *values* in rows (for a column) into *columns*, and can also change multiple *columns'* values into multiple *rows* values for a single column
- It knows how many columns to create for your output file based on the maximum number of values in a column to be transposed (to do this with a Data Step is tedious)
- **ID** statement names the column in the input file whose row values provide the column names in the output file. There should only be one variable in an ID statement. Also, the column used for the ID statement cannot have any duplicate values. For example, Mr. Black and Mr. White cannot both have cats. If this is the case, one solution is to create a different input dataset by using Proc Means to sum the values so that there is only one row for each pet. What if the values of ID are numeric and can't be used as SAS column names? Then use the prefix= option with the ID statement to create variables like "mile140", "mile150", etc.
- **VAR** statement specifies which variables' values are to be transposed; can be character and/or numeric variables; if VAR is omitted, Transpose transposes all numeric vars
- **BY** statement names row-identification variable(s) whose values are not transposed; it requires a preliminary Proc Sort
- Transpose includes some default variables in the output dataset such as **_NAME_**, **_LABEL_**. You can override them with statement options or drop them in a dataset drop option
- **Prefix** option provides a prefix to the transposed column names instead of COL1, COL2, etc
- **Name** option provides the name for an output file column which tells which input variables were transposed
- **Transposing two times** – it is sometimes necessary to transpose an input file two or more times, then merge the output files together due to the only-1-ID-per-transpose limitation.

CONTACT INFORMATION

Douglas Zirbel
Wells Fargo & Co.
St. Louis, MO 63105
douglas.w.zirbel@wellsfargo.com

REFERENCES

Excellent introductions to Proc Transpose
Stuelpner, Janet, *The TRANSPOSE Procedure or How To Turn It Around*, SUGI 31, paper 234.
Tilanus, Erik W., *Turning the data around: PROC TRANSPOSE and alternative approaches*, SAS Global Forum 2007, paper 046.

Specialized or advanced discussions of Proc Transpose
Izrael, David and Russo, David, *Transforming Multiple-Record Data Into Single-Record Format When the Number of Variables Is Large*, SUGI 24, paper 210.
Leighton, Ralph W., *Some Uses (and Handy Abuses) Of Proc Transpose*, SUGI 27, paper 17.
Worden, Jeanina, *Skinny to Fat: In Search of the "Ideal" Dataset*, SAS Global Forum 2007, paper 162.

Please note also several discussions and examples at the SAS web site, www.support.sas.com.

APPENDIX

Code used to generate the tables

```
*****************************************************;
* Create file1 input file;
*****************************************************;
data work.narrow_file1;
    infile cards;
    length pet_owner $10 pet $4 population 4;
    input  pet_owner $1-10 pet $ population;
cards;
Mr. Black  dog  2
Mr. Black  bird  1
Mrs. Green fish 5
Mr. White  cat 3
;
run;


*****************************************************;
* Create file2 input file;
*****************************************************;
data work.narrow_file2;
    infile cards;
    length pet_owner $10 pet $4 population 4;
    input  pet_owner $1-10 pet $ population;
cards;
Mr. Black  dog  2
Mr. Black cat  1
Mrs. Brown dog 1
Mrs. Brown cat 0
Mrs. Green fish 5
Mr. White fish 7
Mr. White dog 1
Mr. White cat 3
;
run;

*****************************************************;
* Create file3 input file;
*****************************************************;
data work.wide_file3;
    infile cards missover;
    length pet_owner $10 cat 4 dog 4 fish 4 bird 4;
    input  pet_owner $1-10 cat dog fish bird;
cards;
Mr. Black   1 2 . 0
Mrs. Brown 0 1 0 1
Mrs. Green . 0 5
Mr. White 3 1 7 2
;
run;
```