

PROC TRANSPOSE: THE MYSTERY UNVEILED

Doug Shannon, Mercer Human Resource Consulting, Phoenix, AZ

ABSTRACT

SAS is great for data manipulation. One of the procedures often overlooked for doing this work is PROC TRANSPOSE. With many of my coworkers there is a great deal of confusion surrounding the procedure. What does it do? What exactly is it going to do to my data? This paper will answer these questions, along with, to going through some SAS options to make the procedure even more useful. A contrasting example of code, without using the transpose procedure, will be shared.

INTRODUCTION

The TRANSPOSE procedure has been a part of BASE SAS for many years. Yet, many are unsure of the procedure and its effects on data. The TRANSPOSE procedure transforms the data from rows to columns or from columns to rows depending on what you are trying to accomplish. There are many times when the TRANSPOSE procedure can cut down on the amount and complexity of coding.

SIMPLE EXAMPLE

Here is an example of the simplest TRANSPOSE. Given a dataset called "ONE" that contains one variable "Year" and labeled "Tax Year".

Year
2000
2001
2002

Dataset: ONE

If this dataset needed to be transposed, the following code could be used.

```
proc transpose data = one;  
run;
```

The resulting dataset would look like this.

NAME	LABEL	COL1	COL2	COL3
Year	Tax Year	2000	2001	2002

Dataset: DATA1

The dataset has been transposed in this example, but the resulting data layout is not as useful as it could be. The TRANSPOSE procedure has many options and portions of the syntax that can help make the resulting data more desirable.

TRANSPOSE SYNTAX

The syntax for the PROC TRANSPOSE is fairly straightforward, but there are some very important details which will help in getting the results you are expecting and making the procedure more robust.

VAR

This allows the user to select which variables to transpose. If no variables are listed, all numeric variables that are not part of the "BY" or "ID" statement will be transposed. In the previous example this was not necessary because there was only one variable and it was numeric. If there is more than one numeric variable, or if a character variable needs to be transposed, the VAR statement is needed.

BY

As with most procedures, the TRANSPOSE procedure can be done with a "BY" statement. This will cause the data set to be transposed within the certain combination of "BY" variables. This will be demonstrated under the Comparison Example section.

ID

This statement allows the user to use the values of a certain variable as the names for the variables transposed. It is important to note that if the combination of the "BY" statement variables and ID variables do not define a unique record then errors will occur.

TRANSPOSE OPTIONS

The TRANSPOSE procedure has many different options that make the procedure more useful.

OUT=

This option assigns an output dataset. If it is not assigned, the default is DATA1, DATA2, etc. writing to the first one that is not already being used. In the first example because no output dataset was assigned, it defaulted to DATA1.

LABEL=

This option lets the user assign a name to the variable that contains the label of the variable being transposed. The default is "_LABEL_". In the example this option was not used. So, the variable containing the label "Tax Year" was called "_LABEL_".

NAME=

This option allows the user to assign a variable name to the variable that contains the name of the variable being transposed. The default is "_NAME_". It is important not to drop this variable if you are transposing more than one variable. If it is dropped, it will be hard to identify which variable is represented in each observation.

PREFIX=

This option lets the user assign the prefix for the transposed variables. The default is COL, which would produce COL1, COL2, COL3, etc.

With these various options we could now change the code to the following.

```
proc transpose data = one prefix = Tax_Year  
out = Taxes(drop = _name_ _label_);  
run;
```

The result of this code would be a dataset Taxes.

Tax_Year1	Tax_Year2	Tax_Year3
2000	2001	2002

Dataset: Taxes

Though this is basically the same dataset as before, it is a little more presentable and the variables have much more meaningful names.

COMPARISON EXAMPLE

With any data manipulation problem there are many ways to accomplish the desired result. Two main methods exist (e.g., the procedure code method and the data step method). Here we will compare the two methods as they apply to the TRANSPOSE procedure. Suppose there is a dataset that looks like this.

Year	MCO	Members
2000	A	1,203
2001	A	1,400
2002	A	1,678
2000	B	956
2001	B	1,025
2002	B	975
2000	C	1,145
2001	C	1,235
2002	C	1,380
2000	D	515
2001	D	214

Dataset: MCO

The data really needs to be in this format.

Year	A	B	C	D
2000	1,203	956	1,145	515
2001	1,400	1,025	1,235	214
2002	1,678	975	1,380	

Which method would be the most appropriate to use?

SOLUTION 1 – The Data Step Method

The Data Step Method consists of using data steps in order to accomplish the desired result. The following code can be used to manipulate the data to get the rows into columns.

```
proc sort data = mco;
by year mco;
run;
```

```
data a(keep = year members rename =
(members = A))
b(keep = year members rename =
(members = B))
c(keep = year members rename =
(members = C))
```

```
d(keep = year members rename =
(members = D));
set mco;
if mco = 'A' then output a;
if mco = 'B' then output b;
if mco = 'C' then output c;
if mco = 'D' then output d;
run;

data mco2;
merge a b c d;
by year;
run;
```

Although this code is not too complex, it is somewhat lengthy. In addition, knowledge of the number of classes that exist in the variable “mco” and what the values are is needed to perform the manipulation. Many times these are not known. This makes the actual programming for this even longer, because some frequencies will need to be performed prior to the actual data manipulation.

SOLUTION 2 – The Procedure Method

The Procedure Method consists of using known SAS procedures (in this case the TRANSPOSE) to accomplish the desired result. The following code can be used to manipulate the data to get the rows into columns.

```
proc sort data = mco;
by year mco;
run;

proc transpose data = mco out = mco2(drop =
_name _label_);
by year;
var members;
id mco;
run;
```

This method is quick, direct, and very easy to use. The ID statement allows the user to make each of the values of the MCO variable identify which column to put the data and name the column that value.

The data can also easily be converted from columns back into rows by using the following code.

```
proc transpose data=mco2 name=mco out=
mco3;
by year;
var a b c d;
run;
```

ADVANCED EXAMPLE

The TRANSPOSE procedure can be used for more complex manipulation also. An example of what can be done follows.

Consider the following example of “Dataset TWO”:

Year	MCO	Employees	Dependents	Members
2000	A	789	414	1,203
2001	A	867	533	1,400
2002	A	890	788	1,678
2000	B	445	511	956
2001	B	478	547	1,025
2002	B	465	510	975
2000	C	578	567	1,145
2001	C	602	633	1,235
2002	C	679	701	1,380
2000	D	255	260	515
2001	D	160	54	214

Dataset: TWO

The following code could be created in order to produce a different type of summary.

```
proc sort data = two;
by year mco;
run;

proc transpose data = two name =
Member_Type out = mco4(drop = _label_);
by year;
var Employees Dependents Members;
id mco;
run;
```

The code would produce the following dataset results.

Year	Member_Type	A	B	C	D
2000	Employees	789	445	578	255
2000	Dependents	414	511	567	260
2000	Members	1,203	956	1,145	515
2001	Employees	867	478	602	160
2001	Dependents	533	547	633	54
2001	Members	1,400	1,025	1,235	214
2002	Employees	890	465	679	
2002	Dependents	788	510	701	
2002	Members	1,678	975	1,380	

Dataset: MCO4

By just adding a few more variables to the VAR statement and keeping track of where the data came from in the "NAME=" option, the resulting dataset was transposed and still maintained the meaning it had prior.

CONCLUSION

The TRANSPOSE procedure is very useful when used to transform data from rows to columns or columns to rows. Many are uncertain of its effects on their data, but by using the procedure along with the various options that are available, manipulating data can be made much easier.

CONTACT INFORMATION

Doug Shannon
Mercer Human Resource Consulting
3131 E. Camelback Rd. Suite 300
Phoenix, AZ 85016

Phone: (602) 522-8577
Fax: (602) 957-9573
Email: doug.shannon@mercer.com