Paper BB135

PROC TRANSPOSE: Flip your Data 90° and Save Time

Rachel Straney, University of Central Florida

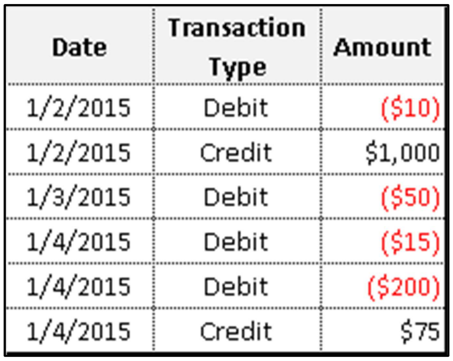
ABSTRACT

The process of transforming data from a vertical to horizontal structure is sometimes referred to as long-to-wide conversion, and is common in the analytical world. Although there is always more than one way to accomplish a task using SAS®, PROC TRANSPOSE is a staple procedure that should be in every programmer's tool box. This paper will guide the reader through some basic examples of PROC TRANSPOSE and share situations where it is most appropriately used.

INTRODUCTION

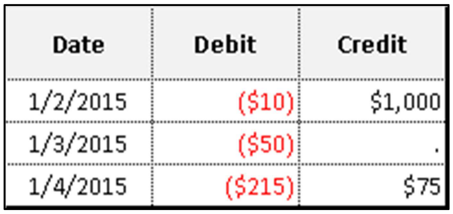
Often times the process of data collection is one of logging or listing information. This seems natural enough, since even the most non-data savvy of us tend organize information in lists: grocery lists, to-do lists, lists of transactions from an account and even lists of observed traffic flow at intersections. Data organized in this form is also known as data that is structured vertically. When analyzing data, however, lists and logs can prove challenging. Analysts and statisticians tend to think of data in terms of experimental units (unique observations) and the attributes that characterize them (variables), or horizontally structured data. And so begins the process of data manipulation, so that these lists and logs of data can be used in an analytical way. The process of transforming data from a vertical to horizontal structure is sometimes referred to as long-to-wide conversion. A visual representation of this process is displayed below using bank account transactions as an example. Whereas the vertical structure is simply a list of all transactions performed (including multiple transactions on the same day), the horizontal structure is organized such that each record is unique to the date of the transaction.

Vertical Structure



Date	Transaction Type	Amount
1/2/2015	Debit	($10)
1/2/2015	Credit	$1,000
1/3/2015	Debit	($50)
1/4/2015	Debit	($15)
1/4/2015	Debit	($200)
1/4/2015	Credit	$75

Horizontal Structure



Date	Debit	Credit
1/2/2015	($10)	$1,000
1/3/2015	($50)	.
1/4/2015	($215)	$75

The power of SAS is its ability to manipulate, structure and organize data efficiently and quickly. Although there is always more than one way to accomplish a task using SAS®, PROC TRANSPOSE is a staple procedure that should be in every programmer's tool box. It will save time and simplify steps in the analysis process that are very prevalent. This paper will guide the reader through some basic examples of PROC TRANSPOSE and share situations where it is most appropriately used.

PROC TRANSPOSE SYNTAX

General syntax of the PROC TRANSPOSE procedure is shown below which was referenced from the 9.4 SAS Procedures Guide:

```
PROC TRANSPOSE <DATA=input-data-set> <DELIMITER=delimiter> <LABEL=label> <LET>
<NAME=name> <OUT=output-data-set> <PREFIX=prefix> <SUFFIX=suffix>;
     BY <DESCENDING> variable-1 <<DESCENDING> variable-2 ...> NOTSORTED>;
     COPY variable(s);
     ID variable;
     IDLABEL variable;
     VAR variable(s);
```

To simplify the content of this paper we will only be discussing the following options and statements:

- Options listed in the PROC TRANSPOSE statement
  - **OUT**: controls the name of the restructured data set
  - **PREFIX**: controls the names of the newly created columns
- Statements used in the procedure
  - **BY**: variable(s) used in the creation of new records or rows*
  - **ID**: variable(s) used in the creation of new columns
  - **VAR**: variable(s) whose values are transposed to fit within the new structure

** Rows are not guaranteed to be unique after transposing data. Obtaining unique rows in the transposed data set will depend on how many variables are used and how the original data is structured, especially when character type variables are involved.*

There are many papers available on the TRANSPOSE procedure as well as documentation from SAS that are recommended if one would like learn more about the options and statements not covered in this paper.

DATA USED IN THE EXAMPLES

Three of the examples covered in the following sections demonstrate the use of the TRANSPOSE procedure along with the associated options and statements. To keep things simple and easy to follow, the data used in the examples are extremely basic and, most likely, are not representative of ‘real-world’ data one would be using for analysis. The last example, Example 4, uses a different data set that more accurately represents data used in practice. Whereas Examples 1-3 are shared to walk through the different components of the TRANSPOSE procedure, Example 4 is to put it all into context.

The SAS data steps below create the data to be used for Examples 1-3.

```
/*Data set with strictly numeric type variables*/
DATA TRNSP_NUM;
INPUT VAR_NUM_1 VAR_NUM_2 VAR_NUM_3 VAR_NUM_4;
CARDS;
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
;
RUN;

/*Data set with numeric and character type variables*/
DATA TRNSP_CHAR;
INPUT VAR_NUM_1 VAR_NUM_2 VAR_NUM_3 VAR_NUM_4 VAR_CHAR_1 $ VAR_CHAR_2 $;
CARDS;
1 2 3 4 A E
1 2 3 4 B E
1 2 3 4 C F
1 2 3 4 D F
;
RUN;
```

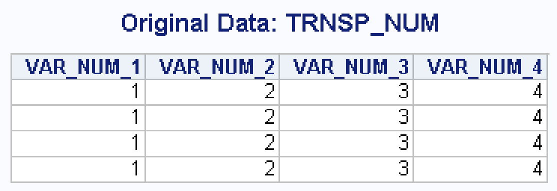
EXAMPLE 1: A BASIC PROC TRANSPOSE

The most basic TRANSPOSE procedure will literally transpose an entire dataset 90^{o}, swapping all rows for columns. One important feature of the TRANSPOSE procedure is that by default only numeric variables are used to transpose the data. If the data set only contains variables of a numeric type, then all rows in the newly created set will include all variables (columns). If character variables exist in the data set, then a basic PROC TRANSPOSE will not suffice. To successfully transpose data including both numeric and character type variables, more statements are needed and we will cover this in later examples.

Examples of a basic PROC TRANSPPOSE on two data sets, one with only numeric type variables and the other with numeric and character types are shown below. The options OUT and PREFIX are included.

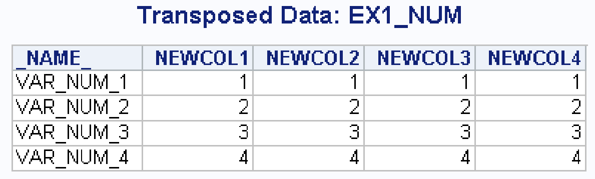
ONLY NUMERIC TYPE VARIABLES

```
PROC TRANSPOSE DATA=TRNSP_NUM
OUT=EX1_NUM PREFIX=NEWCOL;
RUN;

PROC PRINT DATA=TRNSP_NUM NOOBS;
TITLE 'Original Data: TRNSP_NUM';
RUN;

PROC PRINT DATA=EX1_NUM NOOBS;
TITLE 'Transposed Data: EX1_NUM';
RUN;
```



Original Data: TRNSP_NUM

VAR_NUM_1	VAR_NUM_2	VAR_NUM_3	VAR_NUM_4
1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4

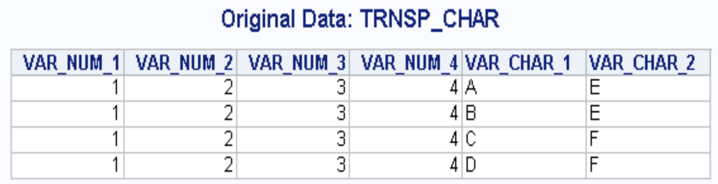


Transposed Data: EX1_NUM

NAME	NEWCOL1	NEWCOL2	NEWCOL3	NEWCOL4
VAR_NUM_1	1	1	1	1
VAR_NUM_2	2	2	2	2
VAR_NUM_3	3	3	3	3
VAR_NUM_4	4	4	4	4

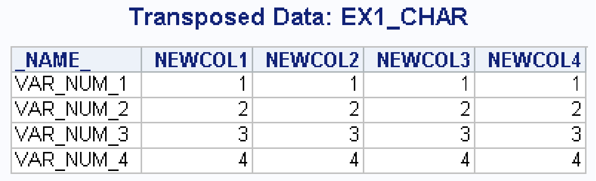
NUMERIC AND CHARACTER TYPE VARIABLES

```
PROC TRANSPOSE DATA=TRNSP_CHAR
OUT=EX1_CHAR PREFIX=NEWCOL;
RUN;

PROC PRINT DATA=TRNSP_CHAR NOOBS;
TITLE 'Original Data: TRNSP_CHAR';
RUN;

PROC PRINT DATA=EX1_CHAR NOOBS;
TITLE 'Transposed Data: EX1_CHAR';
RUN;
```



Original Data: TRNSP_CHAR

VAR_NUM_1	VAR_NUM_2	VAR_NUM_3	VAR_NUM_4	VAR_CHAR_1	VAR_CHAR_2
1	2	3	4	A	E
1	2	3	4	B	E
1	2	3	4	C	F
1	2	3	4	D	F



Transposed Data: EX1_CHAR

NAME	NEWCOL1	NEWCOL2	NEWCOL3	NEWCOL4
VAR_NUM_1	1	1	1	1
VAR_NUM_2	2	2	2	2
VAR_NUM_3	3	3	3	3
VAR_NUM_4	4	4	4	4

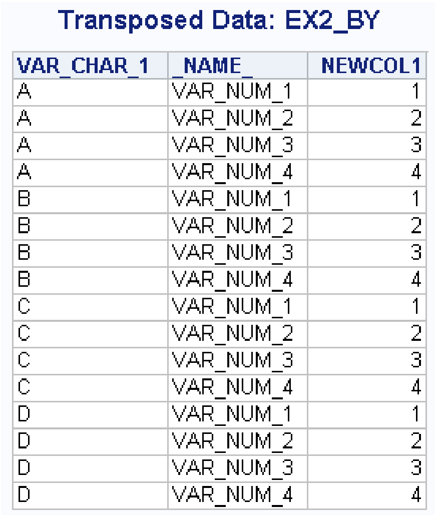
Notice that in both programs the resulting transposed sets, EX1_NUM and EX1_CHAR, are the same. This is due to the fact that, by default, only numeric type variables are used in a basic TRANSPOSE procedure. Furthermore, the options OUT and PREFIX have been included to name the transposed sets and to provide prefixes to the newly created variables.

The next two examples will only include the final printed data sets resulting from the TRANSPOSE procedures. The original data set used in both Example 2 and 3 is TRNSP_CHAR, which can be referenced from Example 1.

EXAMPLE 2: THE ROLES OF THE BY AND ID STATEMENTS

As mentioned previously, the use of the BY statement is used to control the records or rows of the transposed data set. The output data set EX2_BY (on the left below), which only references the BY statement, uses the variable VAR_CHAR_1 to create new records. Note that the program on the right references the ID statement as well as the BY statement, whereby VAR_CHAR_1 creates the rows and VAR_CHAR_2 creates new columns in the final transposed data set. As is typically done when using a BY statement in any SAS procedure, the data may be need to be sorted using a PROC SORT. This step has been skipped since the data are already in sorted order.

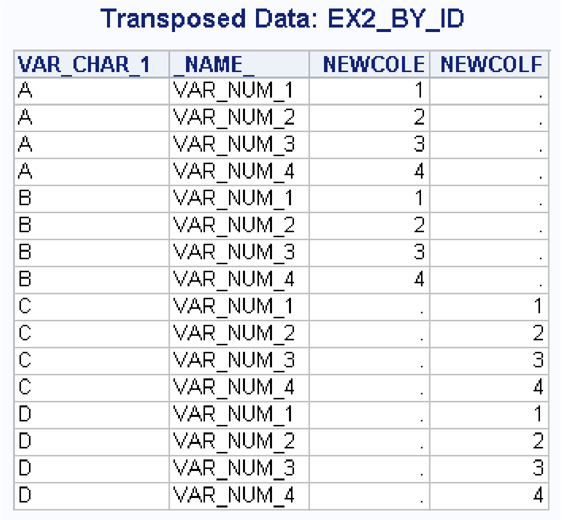
```
PROC TRANSPOSE DATA=TRNSP_CHAR
OUT=EX2_BY PREFIX=NEWCOL;
BY VAR_CHAR_1;
RUN;

PROC PRINT DATA=EX2_BY NOOBS;
TITLE 'Transposed Data: EX2_BY';
RUN;
```



Transposed Data: EX2_BY

VAR_CHAR_1	_NAME_	NEWCOL1
A	VAR_NUM_1	1
A	VAR_NUM_2	2
A	VAR_NUM_3	3
A	VAR_NUM_4	4
B	VAR_NUM_1	1
B	VAR_NUM_2	2
B	VAR_NUM_3	3
B	VAR_NUM_4	4
C	VAR_NUM_1	1
C	VAR_NUM_2	2
C	VAR_NUM_3	3
C	VAR_NUM_4	4
D	VAR_NUM_1	1
D	VAR_NUM_2	2
D	VAR_NUM_3	3
D	VAR_NUM_4	4

```
PROC TRANSPOSE DATA=TRNSP_CHAR
OUT=EX2_BY_ID PREFIX=NEWCOL;
BY VAR_CHAR_1; ID VAR_CHAR_2;
RUN;

PROC PRINT DATA=EX2_BY_ID NOOBS;
TITLE 'Transposed Data: EX2_BY_ID';
RUN;
```



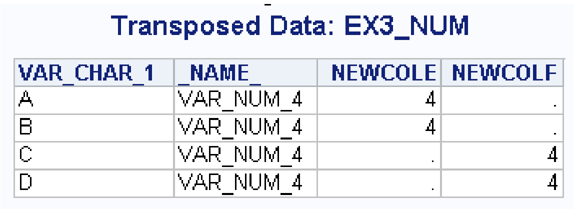
Transposed Data: EX2_BY_ID

VAR_CHAR_1	_NAME_	NEWCOLE	NEWCOLF
A	VAR_NUM_1	1	.
A	VAR_NUM_2	2	.
A	VAR_NUM_3	3	.
A	VAR_NUM_4	4	.
B	VAR_NUM_1	1	.
B	VAR_NUM_2	2	.
B	VAR_NUM_3	3	.
B	VAR_NUM_4	4	.
C	VAR_NUM_1	.	1
C	VAR_NUM_2	.	2
C	VAR_NUM_3	.	3
C	VAR_NUM_4	.	4
D	VAR_NUM_1	.	1
D	VAR_NUM_2	.	2
D	VAR_NUM_3	.	3
D	VAR_NUM_4	.	4

EXAMPLE 3: THE ROLE OF THE VAR STATEMENT

Example 3 builds on the last example to explain the use of the VAR statement. The program below on the left uses the numeric type variable, VAR_NUM_4, and the one on the right uses the character type variable VAR_CHAR_2. Overall, the structure of output data sets EX3_NUM and EX3_CHAR are the same, except for the content or values of the transposed records.

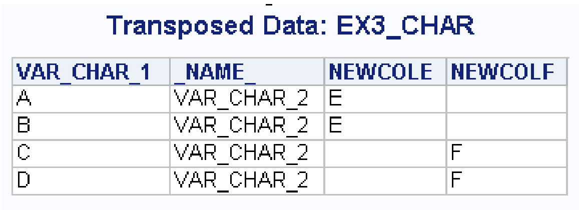
```
PROC TRANSPOSE DATA=TRNSP_CHAR
OUT=EX3_NUM PREFIX=NEWCOL;
BY VAR_CHAR_1;
ID VAR_CHAR_2;
VAR VAR_NUM_4;
RUN;

PROC PRINT DATA=EX3_NUM NOOBS;
TITLE 'Transposed Data: EX3_NUM';
RUN;
```



Transposed Data: EX3_NUM

VAR_CHAR_1	_NAME_	NEWCOLE	NEWCOLF
A	VAR_NUM_4	4	.
B	VAR_NUM_4	4	.
C	VAR_NUM_4	.	4
D	VAR_NUM_4	.	4

```
PROC TRANSPOSE DATA=TRNSP_CHAR
OUT=EX3_CHAR PREFIX=NEWCOL;
BY VAR_CHAR_1;
ID VAR_CHAR_2;
VAR VAR_CHAR_2;
RUN;

PROC PRINT DATA=EX3_CHAR NOOBS;
TITLE 'Transposed Data: EX3_CHAR';
RUN;
```



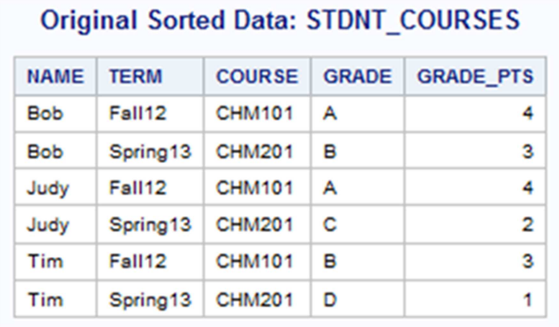
Transposed Data: EX3_CHAR

VAR_CHAR_1	_NAME_	NEWCOLE	NEWCOLF
A	VAR_CHAR_2	E	
B	VAR_CHAR_2	E	
C	VAR_CHAR_2		F
D	VAR_CHAR_2		F

EXAMPLE 4: PUTTING IT ALL TOGETHER IN A REAL WORLD EXAMPLE

The example here uses data collected on students who completed Chemistry I (CHM101) and Chemistry II (CHM201) courses over two semesters. The data includes the student's name, the term the course was completed, the course, final grade and final grade point earned. Suppose we wanted to calculate the difference in student grade points earned between semesters. Due to the vertical nature of the data, it is challenging to compute the difference in final grades from two separate rows. The program below takes advantage of PROC TRANSPOSE as well as PROC SQL to complete this task.

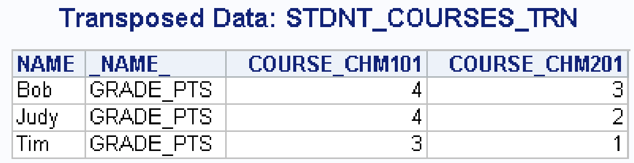
```
DATA STDNT_COURSES;
INPUT NAME $ TERM $ COURSE $ GRADE $ GRADE_PTS;
CARDS;
Judy Fall12 CHM101 A 4
Judy Spring13 CHM201 C 2

Bob Fall12 CHM101 A 4
Bob Spring13 CHM201 B 3

Tim Fall12 CHM101 B 3
Tim Spring13 CHM201 D 1
;
RUN;


PROC SORT DATA=STDNT_COURSES;
BY NAME;
RUN;

PROC PRINT DATA=STDNT_COURSES NOOBS;
TITLE 'Original Sorted Data: STDNT_COURSES';
RUN;
```



Original Sorted Data: STDNT_COURSES

NAME	TERM	COURSE	GRADE	GRADE_PTS
Bob	Fall12	CHM101	A	4
Bob	Spring13	CHM201	B	3
Judy	Fall12	CHM101	A	4
Judy	Spring13	CHM201	C	2
Tim	Fall12	CHM101	B	3
Tim	Spring13	CHM201	D	1

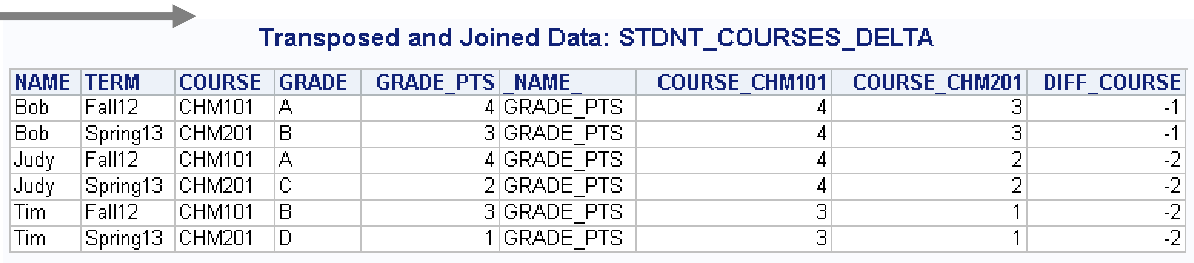
```
PROC TRANSPOSE DATA=STDNT_COURSES
OUT=STDNT_COURSES_TRN PREFIX=COURSE_;
BY NAME;
ID COURSE;
VAR GRADE_PTS;
RUN;

PROC PRINT DATA=STDNT_COURSES_TRN NOOBS;
TITLE 'Transposed Data: STDNT_COURSES_TRN';
RUN;
```



Transposed Data: STDNT_COURSES_TRN

NAME	_NAME_	COURSE_CHM101	COURSE_CHM201
Bob	GRADE_PTS	4	3
Judy	GRADE_PTS	4	2
Tim	GRADE_PTS	3	1

```
PROC SQL;
CREATE TABLE STDNT_COURSES_DELTA AS
SELECT a.*, b.*,
(COURSE_CHM201-COURSE_CHM101) AS DIFF_COURSE
FROM STDNT_COURSES AS a
LEFT JOIN
STDNT_COURSES_TRN AS b
ON a.NAME = b.NAME;
QUIT;

PROC PRINT DATA=STDNT_COURSES_DELTA NOOBS;
TITLE 'Transposed and Joined Data: STDNT_COURSES_DELTA';
RUN;
```



Transposed and Joined Data: STDNT_COURSES_DELTA

NAME	TERM	COURSE	GRADE	GRADE_PTS	_NAME_	COURSE_CHM101	COURSE_CHM201	DIFF_COURSE
Bob	Fall12	CHM101	A	4	GRADE_PTS	4	3	-1
Bob	Spring13	CHM201	B	3	GRADE_PTS	4	3	-1
Judy	Fall12	CHM101	A	4	GRADE_PTS	4	2	-2
Judy	Spring13	CHM201	C	2	GRADE_PTS	4	2	-2
Tim	Fall12	CHM101	B	3	GRADE_PTS	3	1	-2
Tim	Spring13	CHM201	D	1	GRADE_PTS	3	1	-2

CONCLUSION

PROC TRANPOSE is very versatile and can be extremely useful during data manipulation and preparation. In many cases, it can be used as an alternative to the SUMMAY and MEANS procedures to restructure data in a meaningful way. Although the examples used throughout this paper are restructuring data from vertical to horizontal, it should be noted that PROC TRANSPOSE can easily work the other way, organizing horizontal data into a vertical or list format. This procedure can save the analyst valuable time and energy in the data processing phase of any project.

REFERENCES

Li, Arthur X. "Simplifying Effective Data Transformation Via PROC TRANSPOSE." *Proceedings of the PharmaSUG 2012 Conference.* San Francisco, California.
Available at: http://www.pharmasug.org/proceedings/2012/TF/PharmaSUG-2012-TF03.pdf

SAS Institute Inc. 2015. *Base SAS® 9.4 Procedures Guide, Third Edition*. Cary, NC: SAS Institute Inc.
Available at: http://support.sas.com/documentation/cdl/en/proc/67916/PDF/default/proc.pdf

Stuelpner, Janet. "Proc Transpose or How to Turn It Around." *Proceedings of the SUGI 31 Conference.* San Francisco, California. Available at: http://www2.sas.com/proceedings/sugi31/234-31.pdf

Zdeb, Mike. 2012. "Long-to-Wide: PROC TRANSPOSE vs Arrays vs PROC SUMMARY." *Proceedings of the NESUG 2012 Conference.* Baltimore, Maryland. Available at: http://www.lexjansen.com/nesug/nesug12/ff/ff01.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Rachel Straney
University of Central Florida
12424 Research Parkway, Suite 225
Orlando, FL 32826
407-882-0280
rstraney@ucf.edu