USING THE **TRANSPOSE** PROCEDURE TO EFFECTIVELY PRESENT SURVEY RESULTS

Emily Passino, Department of Personnel, State of Tennessee
Karen Montefiori, Department of Finance and Administration, State of Tennessee

INTRODUCTION

In the course of administering Employee Feedback Surveys to over 35,000 employees in 33 different state agencies, our office must generate individual summaries of the data for all levels of each organization. Each work group supervisor receives a report summarizing how his or her employees responded; division managers receive reports summarizing the division's work groups; and each agency head receives an agency-wide summary.

The point of the reports is to stimulate constructive problem-solving at each organizational level. The survey covers a wide variety of work issues, including such sensitive areas as teamwork and supervisory effectiveness. The readers vary from those with advanced degrees to those without a high school diploma.

SAS® software capabilities allowed us to meet the challenge to mass produce thousands of individual reports which would enable readers of widely varying sophistication to zero in on those aspects of the data which would be most fruitful to discuss. Specifically, PROC TRANSPOSE proved to be the key to generating results in "plain English," to highlight key survey results which would then be expanded upon in later sections of the report.

FEATURES

The two key features of PROC TRANSPOSE which make it the procedure of choice for this application are (1) its ability to transpose and re-transpose a data set, and (2) the fact that it preserves labels attached to variables throughout the manipulations.

PROC TRANSPOSE rearranges a SAS data set so that observations become variables and/or variables become observations. Such a transposition allows one to easily compare the results of a series of related questions.

If, for example, you want to be able to report a series of means in descending order, transposing these separate variables into a single variable allows a simple PROC SORT to order the results from "most satisfied" to "least satisfied." Thus manipulated, the single variable can be transposed back again to multiple variables, this time with the desired new information. In the example here, we can now report what this group of employees is most and least satisfied with. Attaching labels to the means before they are transposed allows these results to be reported in the desired "plain English," as shown in Figure 1.

Getting from separate mean values to phrases reporting what employees are "most" and "least" satisfied with is rather straight forward. As with any new procedure, however, when first trying to use PROC TRANSPOSE it is wise to watch what is happening to the data at each step using PROC PRINT. Figure 2 illustrates the general changes in the data which occur with transposition. Options are available with PROC TRANSPOSE to help you track what the data look like at each step along the way, and to retrieve the desired values.

SPECIFICATIONS

Two PROC TRANSPOSE statements were used for this application:

```
PROC TRANSPOSE options;
  VAR variables;
```

The options found to be useful were:

DATA = *SASdataset*
to name the set of related questions to be transposed, which had been previously defined and labeled; or to name the data set containing the transposed observations

PREFIX = *name*
which was used both to construct new variable names when transposing from observations to variables, and to create a variable name when transposing from variables to observations

OUT = *SASdataset*
which was used to specify a name for the data set being created at that step of the process.

Options not used. For our purposes, we found that using the SAS generated names of __NAME__ and __LABEL__, both automatically created with the procedure, worked just as well as supplying our own substitutes. Finally, we had no need for the option LET, which can handle duplicate values of an ID variable.

FIGURE 1: EXAMPLE OF PRINTED OUTPUT

EXAMPLE "A": Reporting "Most" and "Least"

With respect to the job itself, people report the highest satisfaction with:
FELLOW EMPLOYEES, and VARIETY OF SKILLS ABILITIES.

They report the least satisfaction with:
OPPORTUNITY FOR GROWTH, DEV, and CONTROL OVER WORK.

Regarding their employment, they are most satisfied with:
ASSIGNED WORKING HOURS, and JOB SECURITY.

They are least satisfied with:
PAY, and PROMOTIONAL OPPORTUNITIES.

EXAMPLE "B": Reporting "Top Three"

When asked to choose which topics would be most important to discuss in their feedback meetings, these employees felt the following should be at the top of the list:
IMPROVING WORK GROUP MORALE,
CUTTING DOWN ON WASTE, DUPLICATION, and
SUPERVISION, MANAGEMENT.

EXAMPLE "C": Reporting By Category

Where are improvements needed? Responses to this series of questions showed that, on the whole, these employees would recommend the following:

Little or No Improvement Necessary	Some Improvement Helpful	Much Improvement Needed
CLEAR CHAIN-OF-COMMAND	EQUIPMENT,MATERIALS,SUPPLIES	MANAGEABLE WORK LOAD
APPROPRIATE TRAINING	CLEAR POLICIES,PROCEDURES	MANAGEABLE PRESSURE, STRESS
EFFECTIVE SUPERVISION	SUPPORT FROM HIGHER MANAGEMENT	TIMELY, RELIABLE INFORMATION
	WELL-DEFINED JOB RESPNSBLTIES,GOALS	AUTHORITY TO CARRY OUT JOB
	COORDNTN WITH OTHER WORK GROUPS	

FIGURE 2: OVERVIEW OF HOW THE DATA CHANGE

STAGE 1: *Before Transposition*

OBS	VAR1	VAR2	VAR3 . . .	VARn
1	Value1	Value2	Value3. . .	Valuen

STAGE 2: *After First Transposition*

OBS	_NAME_	_LABEL_	GROUP1
1	VAR1	Label1	Value1
2	VAR2	Label2	Value2
3	VAR3	Label3	Value3
.	.	.	.
.	.	.	.
n	VARn	Labeln	Valuen

STAGE 3: *After Second Transposition*

OBS	NEWVAR1	NEWVAR2	NEWVAR3 . . .	NEWVARn
1	Label1	Label2	Label3 . . .	Labeln

EXAMPLE "A": Reporting "Most" and "Least"

For our report, we wanted to highlight which job facets employees reported being the most and the least satisfied with. Within the broad dimension of "job satisfaction," we inquired about 16 areas, seven related to satisfaction with employment, and nine related to satisfaction with the job itself. The following general steps were taken to summarize and report these outcomes, in the manner shown in Figure 1, Part "A."

```
*******CODE BEGINS*******

(1)  PROC UNIVARIATE;
      VAR Z1-Z16;
       OUTPUT OUT = WGMEANS
        MEAN = MEAN1-MEAN16;

(2)  DATA PHRASE1;
        SET WGMEANS;
      LABEL MEAN1 = 'PAY';
      LABEL MEAN2 = 'PROMOTIONAL '
                    'OPPORTUNITIES';
      LABEL MEAN3 = 'JOB SECURITY';
             etc.


(3)  PROC TRANSPOSE DATA = PHRASE1
        PREFIX = GROUP  OUT = EMPSAT;
        VAR MEAN1 - MEAN7;
     PROC SORT;
        BY GROUP1;

     PROC TRANSPOSE DATA = PHRASE1
        PREFIX = GROUP  OUT = JOBSAT;
        VAR MEAN8 - MEAN16;
     PROC SORT;
        BY GROUP1;

(4)  DATA BOTHSATS; SET EMPSAT JOBSAT;


(5)  PROC TRANSPOSE DATA = BOTHSATS
       PREFIX = SAT OUT = JOBSATS;
        VAR  _LABEL_;

*******CODE ENDS*******
```

(1) PROC UNIVARIATE was used to produce a temporary SAS dataset (WGMEANS) containing the 16 mean values for these questions.

NOTES: PROC UNIVARIATE was chosen because we also needed to have median values calculated for a different set of questions. Means were given separate new names to preserve a distinction between this dataset and the original, still needed elsewhere.

(2) Labels were attached to each of the means with separate LABEL statements.

NOTES: At this point (Stage 1 in Figure 2) we now have one observation consisting of 16 separate, laheled variables, MEAN1-MEAN16.

(3) The two subsets of job satisfaction means were then separately transposed and sorted.

NOTES: See Stage 2, Figure 2. Since the data being transposed (PHRASE1) contain variables which are labeled, the labels themselves become a variable ("_LABEL_") in the transposed data set.

The PREFIX option is used to name the newly created variable, GROUP1. It can then be used to sort the transposed dataset. Note that the same variable name (GROUP1) is used in the two transpositions so that the two temporary datasets (EMPSAT and JOBSAT) can be joined together in the next step.

(4) The two subsets of transposed data were then joined.

NOTES: The data here are still arranged as shown in Stage 2. We now have a list of the 16 job satisfaction items (DATA = BOTHSATS), sorted and combined so that the first seven are always related to employment satisfaction, in ascending order from "most" to "least." The next nine are then related to satisfaction with the job itself, again in ascending order.

(5) The variables were once again transposed, now back to a single observation (Stage 3 in Figure 2).

NOTES: With the prefix "SAT," the new variables constructed from this transposition will be the 16 label-phrases named SAT1, SAT2, ... SAT16. Because of the way the data were structured in the dataset "BOTHSATS," we know that we can select the first two variables, SAT1, and SAT2, to retrieve the phrases describing areas of employment the employees are most satisfied with. The sixth and seventh variables, SAT6 and SAT7, will reflect areas of employment they are least satisfied with. Similarly, SAT8 and SAT9 will head the list of factors associated with the job itself, while SAT15 and SAT16 will identify areas of least satisfaction with the job itself.

For the report itself, we have used the following PUT statements to generate the first sentences shown in Figure 1, Part "A."

```
*******CODE BEGINS*******

PUT #32 @23 'With respect to the job '
 'itself, people report the highest '
 'satisfaction with:'/// @23 'They report '
 'the least satisfaction with:';
PUT #33 @25 SAT8 ', and ' SAT9 '.';
PUT #36 @25 SAT15 ', and ' SAT16 '.';
PUT #38 @23 'Regarding their employment, '
 'they are most satisfied with:' /// @23
 'They are least satisfied with:';
PUT #39 @25 SAT1 ', and ' SAT2 '.';
PUT #42 @25 SAT6 ', and ' SAT7 '.';

*******CODE ENDS*******
```

EXAMPLE "B": Reporting "Top Three"

One survey question presented a list of nine possible areas each employee might consider as an area worthy of discussion/problem-solving. Employees could mark as many of the nine as they wished, but were encouraged to choose only three. To report which three areas were of the most concern to a particular group, a sequence of steps similar to the above was performed. This time, however, instead of sorting by means, the data are sorted by frequency. The portion of the report to be produced here is shown as Part "B" in Figure 1.

```
*******CODE BEGINS*******

(1) DATA PAGE1;
       SET PAGE1 END = LASTOBS;
    ARRAY Q{9} Q1-Q9;
    ARRAY A{9} A1-A9;
    RETAIN A1-A9;
       DO I = 1 TO 9;
          IF Q{I}= 1 THEN A{I}+ 1;
       END;
    IF LASTOBS NE 1 THEN RETURN;

(2) DATA PHRASE2;
      SET PAGE1;
      LABEL A1= 'IMPROVING QUALITY OF '
        'WORK';
      LABEL A2= 'CUTTING DOWN ON WASTE, '
        'DUPLICATION';

                    etc.

(3) PROC TRANSPOSE DATA = PHRASE2
      PREFIX = CONCERN  OUT = PRTYLIST;
       VAR = A1-A9;
    PROC SORT;
          BY DESCENDING CONCERN1;

(4) PROC TRANSPOSE DATA=PRTYLIST
       PREFIX = TOPS  OUT = TOP3;
       VAR  _LABEL_;

*******CODE ENDS*******
```

(1) First, the frequencies with which each response was chosen were calculated.

(2) Next, labels were attached with separate LABEL statements.

(3) Data were now ready to be transposed and sorted.

NOTES: With this first transposition, the data have been transposed from one observation with nine variables, to nine observations with one variable: CONCERN1.

The values of the variable CONCERN1 represent the frequencies for each observation. Sorting in DESCENDING order places the most frequently marked concerns at the top of the list.

(4) Once again, the separate observations are transposed back to a single observation.

NOTE: Here we have named the variables newly created by this second transposition by using the option PREFIX to name them TOPS1 through TOPS9. For the report, we needed only the top three, TOPS1, TOPS2, and TOPS3.

Once again, PUT statements are used to generate the sentence shown as Part "B" of Figure 1, as follows:

```
*******CODE BEGINS*******

PUT #49 @53'PRIORITIES AND '
 'RECOMMENDATIONS'// @12'When asked '
 'to choose which topics would be most '
 'important to discuss in their feedback '
 'meetings, these employees '/ @12 'felt '
 'the following should be at the top of '
 'the list:';

       PUT #53 @25 TOPS1 ',';
       PUT #54 @25 TOPS2 ', and ';
       PUT #55 @25 TOPS3 '.';

*******CODE ENDS*******
```

EXAMPLE "C": Reporting By Category

The third and last example of using PROC TRANSPOSE shows how this procedure allowed us to sort aggregate employee responses into three broad categories (Part "C," Figure 1). We began with questions about the adequacy of 12 resources which employees could describe as needing "little or no improvement," "some improvement" or "much improvement." The goal was to report how the group as a whole regarded each resource.

```
*******CODE BEGINS*******

(1) PROC UNIVARIATE;
       VAR Y1-Y12;
           OUTPUT OUT = WGMEDS
       MEDIAN = MED1-MED12;

(2) DATA PHRASE3;
       SET WGMEDS;
    LABEL MED1 = 'EQUIPMENT, MATERIALS, '
       'SUPPLIES';
    LABEL MED2 = 'CLEAR POLICIES, '
       'PROCEDURES';
                    etc.

(3) PROC TRANSPOSE DATA = PHRASE3
       PREFIX = GROUP  OUT = RECLIST;
       VAR MED1 - MED12;

(4) DATA RECLIST; SET RECLIST;
       IF 1.0 LE GROUP1 LE 1.5 THEN REC = 1;
       IF 1.6 LE GROUP1 LE 2.5 THEN REC = 2;
       IF 2.6 LE GROUP1 LE 3.0 THEN REC = 3;
```

```
(5) DATA A; SET REGLIST; IF REC = 3;
      PROC TRANSPOSE DATA = A
        PREFIX = LITTLE  OUT = FIRSTCOL;
         VAR  _LABEL_;

    DATA B; SET REGLIST; IF REC = 2;
      PROC TRANSPOSE DATA = B
       PREFIX = SOME  OUT = NEXTCOL;
         VAR  _LABEL_;

    DATA C; SET REGLIST; IF REC = 1;
      PROC TRANSPOSE DATA = C
       PREFIX = LOTS  OUT = LASTCOL;
         VAR  _LABEL_;

(6) DATA ABC;
     MERGE FIRSTCOL NEXTCOL LASTCOL;
```

*********CODE ENDS*********

(1) Once again we began with PROC UNIVARIATE, this time to calculate medians.

NOTES: Medians were found to be the best measure of central tendency to use with these questions, after trials with real survey data. In part this may have been because these questions had only three possible responses.

(2) Labels were then attached using LABEL statements, allowing a maximum of 35 characters so that three side-by-side columns could be produced.

(3) The first transposition was then produced.

From one observation of 12 medians, we now have twelve observations of the variable, GROUP1, along with the automatically created variables, _NAME_ and _LABEL_. This time, instead of sorting by frequency or by mean value, we used the median values to determine how the group rated each resource.

(4) Based on median values, we created a new variable to indicate whether a resource was judged to need little, some, or much improvement.

NOTES: Median values of the variable, GROUP1 are used as the basis of defining the new variable, REC. REC has three possible values. Note that the data are still in the "Stage 2" configuration, now with 12 observations and a fourth variable, REC.

(5) Based on the values of REC, three subsets of data are created to be transposed with an identifying prefix.

NOTES: The first set of statements selects all of the observations in the data set RECLIST in which REC=3. When transposed, these observations become new variables, potentially LITTLE1 through LITTLE12. Thus, whatever questions had a median value of 2.6 to 3.0 can be reported, as phrases, in the first column of the table, shown in Part "C," Figure 1.

The next statements follow the same logic to select those questions/phrases to be slotted into the middle and last columns.

Just as one might expect, some groups of employees rated the 12 resources so that there were a few in each column, while others may have lumped them all as two or even one category. The procedure outlined above is capable of handling 0 observations in a subset. If this happens, the SAS system just returns a message that that particular data set has "0" observations.

(6) The three transposed data sets, each with one observation, are now merged into one dataset with one observation.

At this point, we now have one observation with 12 variables. The variables may be LITTLE1 through LITTLE12, with no 'SOME' or 'LOTS' variables, or they may be any other possible interesting mixture.

We are now ready to program the PUT statements to generate the "table" shown in Figure 1.

*********CODE BEGINS*********

```
PUT @58 @12 'Where are Improvements '
 'needed?  Responses to this series of '
 'questions showed that, on the whole,'
 'these employees' / @12 'would '
 'recommend the following:'// @12
 'Little or No Improvement'/ @17
 'Necessary' @49 'Some Improvement '
 'Helpful '@86 'Much 'Improvement '
 'Needed '/ @12 108*'-';

PUT #64 @12 LITTLE1/ @12 LITTLE2/
 @12 LITTLE3/ @12 LITTLE4/ @12 LITTLE5/
 @12 LITTLE6/ @12 LITTLE7/ @12 LITTLE 8/
 @12 LITTLE9/ @12 LITTLE10/ @12 LITTLE11/
 @12 LITTLE12;
PUT #64 @49 SOME1/ @49 SOME2/ @49 SOME3/
 @49 SOME4/ @49 SOME5/ @49 SOME6/
 @49 SOME7/ @49 SOME8/ @49 SOME9/
 @49 SOME10/ @49 SOME11/ @49 SOME12;
PUT #64 @86 LOTS1/ @86 LOTS2/ @86 LOTS3/
 @86 LOTS4/ @86 LOTS5/ @86 LOTS6/
 @86 LOTS7/ @86 LOTS8/ @86 LOTS9/
 @86 LOTS10/ @86 LOTS11/ @86 LOTS12;
```

*********CODE ENDS*********

NOTES: Because any group could have any 12 of the 36 variables listed in the PUT statements, all 36 must be listed. This means that in the DATA step, an OPTIONS statement must be used to prevent SAS's "missing data" symbol, '.', from peppering the page. Other options such as 'N = PS' (to free the pointer) and 'NOTITLES' will be needed as well. Thus, the following statements will be needed:

```
DATA _NULL_; SET ABC;
    OPTIONS MISSING = ' ';
FILE PRINT N=PS  PS = 90 NOTITLES;
```

Finally, the "extra" variables here will mean that on each SAS log you can expect 24 "NOTES" that 24 variables are uninitialized. Again, this will not affect the SAS processing.

COMMENTS

Why the second transposition back to one observation?

It is entirely possible to use the data set created with the first PROC TRANSPOSE. For example, to report the categorized resource questions in Example "C," the code shown below works. Remember, here we are dealing with "Stage 2" data: 12 observations with 3 variables.

```
*******CODE BEGINS*******

IF _N_ = 1 THEN DO;
   LINE1 = 63; LINE2 = 63; LINE3 = 63;
END;
IF 1.0 LE GROUP1 LE 1.5 THEN DO;
   LINE1 + 1
   PUT #LINE @12 _LABEL_;
END;
IF 1.6 LE GROUP1 LE 2.5 THEN DO;
   LINE2 + 1;
   PUT #LINE @49 _LABEL_;
END;
IF 2.6 LE GROUP LE 3.0 THEN DO;
   LINE3 + 1;
   PUT #LINE @86 _LABEL_;
END;

*******CODE ENDS*******
```

For our purposes, since the data set used for the remaining 20 some odd pages of the report was a single observation, we chose to have the entire data set formed as a single observation.

CONCLUSION

The problem we originally set out to solve was how to translate complex survey results into plain English. The program to be developed had to return character values instead of intimidating tables and numbers. Furthermore, whatever solution we developed had to be capable of "automatically" generating thousands of individualized reports.

The TRANSPOSE procedure proved to be a simple, yet elegant, solution to achieve our goal. In the process of working with this procedure to understand how it was manipulating the data set, we came to appreciate three features which will be useful for other applications.

First, the capability of treating variables as observations through PROC TRANSPOSE enables one to compare, sort, and redefine related values from a whole new perspective. In our "old" way of looking at separate mean values, for example, each cell value was located in a horizontal row. Comparing these values "side-by-side" could only be done through complex and cumbersome data definitions. Once we discovered just how the transposition was operating, that these values now became located in a vertical column and could be treated as if they were observations, many new research and reporting options were opened to us.

Secondly, the TRANSPOSE procedure makes it possible to process variable labels as observations. As we all work toward trying to have our output more "reader friendly," this possibility appears to have a multitude of applications.

Finally, the VAR statement in the TRANSPOSE procedure provides a streamlined method of conditionally processing a data set. The statement here works just the same as the selecting VAR statements with such procedures as FREQ and UNIVARIATE. Rather than separately subsetting the input dataset with KEEP or IF statements prior to the operation, particular subsets of the input data set are specified as the operation is invoked. The statement's value for transpositions means that one can retain the basic shape of the input data set, transposing only the portion of interest.

Our "discoveries" outlined above may well seem elementary for anyone who understands the TRANSPOSE procedure. They are offered for those like ourselves who start with a particular application need, stumble upon a SAS procedure which appears to offer a solution, and struggle through understanding how it works in order to get the procedure to accomplish the original objective. As we cautioned earlier in the paper, it pays to watch what is happening to your data set at each step using PROC PRINT. This not only ensures that what you think is happening is really happening, but it also may enable you to take further advantage of some built-in capabilities of the procedures.

ACKNOWLEDGEMENTS

SAS® is the registered trademark of SAS Institute Inc., Cary, N.C., U.S.A.