Simplifying Effective Data Transformation Via PROC TRANSPOSE
Arthur X. Li, City of Hope Comprehensive Cancer Center, Duarte, CA

ABSTRACT

You can store data with repeated measures for each subject, either with repeated measures in columns (one observation per subject) or with repeated measures in rows (multiple observations per subject). Transforming data between formats is a common task because different statistical procedures require different data shapes. Experienced programmers often use ARRAY processing to reshape the data, which can be challenging for novice SAS® users. To avoid using complex programming techniques, you can also use the TRANSPOSE procedure to accomplish similar types of tasks. In this talk, PROC TRANSPOSE, along with its many options, will be presented through various simple and easy-to-follow examples.

INTRODUCTION

PROC TRANSPOSE is a flexible procedure that allows you to transpose one or more variables of all the observations in your entire data set or observations within each level of one or more variables. When transposing values of the variables for all the observations, data presented in rows from the input data is transposed into columns in the resulting data. For example, Dat1 (See Figure 1) contains the three English test scores for John and Mary. The scores are stored in three columns, E1 – E3, and two rows (for two observations) in Dat1. All the scores are presented in the form of a 2 X 3 matrix. To transpose the scores in Dat1, the scores in the rows need to be rotated to columns or scores in columns need to be rotated to rows. The dataset Dat1_Transpose1 is the transposed form of data set Dat1. Notice that all the scores are presented in the form of a 3 X 2 matrix in the transposed data.

You can also transpose Dat1 for each person. The values of E1 – E3 for each person/observation can also be considered as a group of scores, with each group being identified by the value of the NAME variable. The variable that is used to distinguish the groupings is called the BY-variable. The resulting transposed data set Dat1_Transpose2 is the transposed form of Dat1 by each level of the NAME variable. Variable TEST is used to distinguish the different scores.

Figure 1. SAS data sets, Dat1, Dat1_Transpose1, and Dat1_Transpose2.
To transpose data, you need to follow the syntax below. The six statements in the TRANSPOSE procedure, which includes PROC TRANPOSE, BY, COPY, ID, IDLABEL, and VAR statements, along with the eight options in the PROC TRANSPOSE statement, are used to apply different types of data transpositions and give the resulting data set a different appearance. In this paper, we will focus on the data transformation type and learn how to use these statements and/or options to perform the data transformation to achieve the results that we desired.

```
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-n>
    COPY variable(s);
    ID variable;
    IDLABEL variable;
    VAR variable(s);
```

TRANSPOSING AN ENTIRE DATA SET

THE DEFAULT FORMAT OF TRANPOSED DATA SETS

Program 1 starts with creating the data set `dat1` with an additional ID variable and labels E1 – E3 variables with English1 – English3. In the PROC TRANSPOSE statement, the OUT= option is used to specify the name of the transposed data set. Without using the OUT= option, PROC TRANSPOSE will create a data set that uses the DATAn naming convention.

By default, without specifying the names of the transposing variables, all the numeric variables from the input data set are transposed. In the transposed data set, `dat1_out1`, E1 – E3 is transposed to two variables with default variable names, COL1 and COL2. The names of the transposed variables from the input data set are stored under variable _NAME_. Since E1 – E3 have permanent labels from the input data set, these labels are stored under variable _LABEL_.

```
Program 1:
data dat1;
    input name $ id $ e1 - e3;
    label e1 = English1
        e2 = English2
        e3 = English3;
    datalines;
John A01 89 90 92
Mary A02 92 .  81
;
proc transpose data=dat1 out=dat1_out1;
run;

proc print data=dat1 label;
    title 'dat1 in the original form';
run;

proc print data=dat1_out1;
    title 'dat1 in transposed form wit OUT= option';
run;
```
Output from Program 1:

<table>
<thead>
<tr>
<th>Obs</th>
<th>name</th>
<th>id</th>
<th>English1</th>
<th>English2</th>
<th>English3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>A01</td>
<td>89</td>
<td>90</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>Mary</td>
<td>A02</td>
<td>92</td>
<td>.</td>
<td>81</td>
</tr>
</tbody>
</table>

dat1 in transposed form with OUT= option

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>NAME</em></th>
<th><em>LABEL</em></th>
<th>COL1</th>
<th>COL2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e1</td>
<td>English1</td>
<td>89</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>e2</td>
<td>English2</td>
<td>90</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>e3</td>
<td>English3</td>
<td>92</td>
<td>81</td>
</tr>
</tbody>
</table>

CONTROLING THE NAMES OF THE VARIABLES IN THE TRANPOSED DATA SET

All the variables in the transposed data set from Program 1 are assigned default variable names. You can provide the names of the transposed variables by utilizing some options in the PROC TRANSPOSE statement.

In Program 2 three additional options are added to the PROC TRANSPOSE statement. The NAME= option is used to specify the name of the variable in the transposed data set that contains the name of the variable that is being transposed. The LABEL= option is used to specify the name for the variable that contains the labels of the variables that are being transposed. The PREFIX= option is used to place a prefix in the transposed variable names. For example, since PREFIX = score_ is used in the PROC TRANSPOSE statement, the names of the transposed variables will be SCORE_1 and SCORE_2. You can also use the SUFFIX= option to attach a suffix in the transposed variable name.

The VAR statement is used in Program 2. Since the transposed variables were not specified, PROC TRANSPOSE will transpose all the numeric variables; thus, whether or not specifying var e1–e3 in Program 2 will yield the same result.

Program 2:

```sas
proc transpose data=dat1
   out=dat1_out2
   name=varname
   label=labelname
   prefix=score_
   var e1-e3;
run;

proc print data=dat1_out2;
   title 'dat1 in transposed form with controlled variable names';
run;
```

Output from Program 2:

<table>
<thead>
<tr>
<th>Obs</th>
<th>varname</th>
<th>labelname</th>
<th>score_1</th>
<th>score_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e1</td>
<td>English1</td>
<td>89</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>e2</td>
<td>English2</td>
<td>90</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>e3</td>
<td>English3</td>
<td>92</td>
<td>81</td>
</tr>
</tbody>
</table>
USING THE ID STATEMENT TO LABEL THE NAMES OF THE TRANSPOSED VARIABLES

In Program 2, the transposed variables are named SCORE_1 and SCORE_2. SCORE_1 contains the scores for John and SCORE_2 contains the scores for Mary. Instead of using SCORE_1 and SCORE_2, you can attach the name of the person to the transposed variable.

In Program 3, the ID statement is used to specify the variable from the input data set that contains the values to rename the transposed variables. Since the PREFIX= option is used, the name of the transposed variables are created by combining the value that is specified by the PREFIX= option and the values from the variable in the ID statement. Therefore, the names of the transposed variables are SCORE_JOHN and SCORE_MARY in the transposed data set. Without specifying the PREFIX= option, the names of the transposed variable will only be JOHN and MARY.

```
Program 3:
proc transpose data=dat1
   out=dat1_out3
   label=labelname
   name=varname
   prefix=score_
   var e1-e3;
   id name;
run;
proc print data=dat1_out3;
   title 'The use of ID statement';
run;
```

Output from Program 3:

```
The use of ID statement

<table>
<thead>
<tr>
<th>Obs</th>
<th>varname</th>
<th>labelname</th>
<th>score_</th>
<th>score_</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>John</td>
<td>Mary</td>
</tr>
<tr>
<td>1</td>
<td>e1</td>
<td>English1</td>
<td>89</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>e2</td>
<td>English2</td>
<td>90</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>e3</td>
<td>English3</td>
<td>92</td>
<td>81</td>
</tr>
</tbody>
</table>
```

In Program 4, two variables, NAME and ID, are used in the ID statement along with the DELIM= option in the PROC TRANSPOSE statement. The values that are created by concatenating the NAME and the ID variables (separated by the value that is specified by the DELIM= option) are used as the names of the transposed variables.

```
Program 4:
proc transpose data=dat1
   out=dat1_out4
   label=labelname
   name=varname
   delim=;
   var e1-e3;
   id name id;
run;
proc print data=dat1_out4;
   title 'The use of ID statement with more than one variable';
run;
```

Output from Program 4:

```
The use of ID statement with more than one variable

<table>
<thead>
<tr>
<th>Obs</th>
<th>varname</th>
<th>labelname</th>
<th>John_A01</th>
<th>Mary_A02</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e1</td>
<td>English1</td>
<td>89</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>e2</td>
<td>English2</td>
<td>90</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>e3</td>
<td>English3</td>
<td>92</td>
<td>81</td>
</tr>
</tbody>
</table>
```
Program 5 illustrates an alternative way to control the names of the transposed variables by adding the IDLABEL statement. The variable that is specified in the IDLABEL statement from the input data set contains the values to label the transposed variable. The variable that is specified in the IDLABEL statement can be either numeric or character. From the partial output from the CONTENTS procedure, you can see that the names of the transposed variables are SCORE_JOHN and SCORE_MARY, with A01 and A02 as their labels, respectively.

Program 5:

```sas
proc transpose data=dat1
  out=dat1_out5
  label=labelname
  name=varname
  prefix=score_;
  var e1-e3;
  id name;
  idlabel id;
run;

proc contents data=dat1_out5;
run;
```

Partial Output from Program 5:

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>labelname</td>
<td>Char</td>
<td>40</td>
<td>LABEL OF FORMER VARIABLE</td>
</tr>
<tr>
<td>3</td>
<td>score_John</td>
<td>Num</td>
<td>8</td>
<td>A01</td>
</tr>
<tr>
<td>4</td>
<td>score_Mary</td>
<td>Num</td>
<td>8</td>
<td>A02</td>
</tr>
<tr>
<td>1</td>
<td>varname</td>
<td>Char</td>
<td>8</td>
<td>NAME OF FORMER VARIABLE</td>
</tr>
</tbody>
</table>

### TRANSPONING BY-GROUPS

**THE DEFAULT FORMAT FOR TRANSPONING BY-GROUPS**

Program 6 transposes `dat1` by using `NAME` as the BY-variable. You can specify more than one variable in the BY statement. To use the BY statement in PROC TRANSPOSE, the data set must be previously sorted by using the same BY-variable. The BY-variable is not transposed. The number of observations in the transposed data set (6) equals to the number of BY-groups (2) times the number of variables that are transposed (3). The number of transposed variables equals to the number of the observations within each BY-group in the input data set. Thus, in this example, the number of transposed variables is one with a default name of COL1.

Program 6:

```sas
proc sort data=dat1 out=dat1_sort;
  by name;
run;

proc transpose data=dat1_sort out=dat1_out6 ;
  by name;
run;

proc print data=dat1_out6;
  title 'The default format of transposing by-groups';
run;
```
Output from Program 6:

<table>
<thead>
<tr>
<th>Obs</th>
<th>name</th>
<th><em>NAME</em></th>
<th><em>LABEL</em></th>
<th>COL1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>e1</td>
<td>English1</td>
<td>89</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>e2</td>
<td>English2</td>
<td>90</td>
</tr>
<tr>
<td>3</td>
<td>John</td>
<td>e3</td>
<td>English3</td>
<td>92</td>
</tr>
<tr>
<td>4</td>
<td>Mary</td>
<td>e1</td>
<td>English1</td>
<td>92</td>
</tr>
<tr>
<td>5</td>
<td>Mary</td>
<td>e2</td>
<td>English2</td>
<td>.</td>
</tr>
<tr>
<td>6</td>
<td>Mary</td>
<td>e3</td>
<td>English3</td>
<td>81</td>
</tr>
</tbody>
</table>

USE THE COPY STATEMENT TO COPY VARIABLES FROM THE INPUT DATA SET

You can use the COPY statement to copy one or more variables from the input data set directly to the transposed data set. For example, in Program 7, the COPY statement is used to copy the ID variable from the input data set. Since there are two observations from the input data set, the number of observations that will be copied will be two as well; SAS pads the missing values to the rest of the observations.

Program 7 also utilizes the data set option to make the appearance of the transposed data more appealing. The RENAME= option renames the default column names COL1 and _LABEL_ to SCORE and TEST, respectively. The DROP= option drops the variable _NAME_ and the WHERE= option is used to delete any observations with missing scores. Instead of using the RENAME= data set option to rename the _LABEL_ variable, you can also use the LABEL= option from the PROC TRANSPOSE statement to rename the _LABEL_ variable.

Program 7:
```
proc transpose data=dat1_sort
   out=dat1_out7 (rename=(col1=SCORE _label_=TEST)
                  drop=_name_
                  where=(score ne .));
   by name;
   copy id;
run;
```

Output from Program 7:

<table>
<thead>
<tr>
<th>Obs</th>
<th>name</th>
<th>id</th>
<th>TEST</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>A01</td>
<td>English1</td>
<td>89</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td></td>
<td>English2</td>
<td>90</td>
</tr>
<tr>
<td>3</td>
<td>John</td>
<td></td>
<td>English3</td>
<td>92</td>
</tr>
<tr>
<td>4</td>
<td>Mary</td>
<td>A02</td>
<td>English1</td>
<td>92</td>
</tr>
<tr>
<td>5</td>
<td>Mary</td>
<td></td>
<td>English3</td>
<td>81</td>
</tr>
</tbody>
</table>

SITUATIONS FOR USING THE ID STATEMENT FOR TRANSPOSING BY-GROUPS

The ID statement can be used to specify the variable from the input data set that contains the values to rename the transposed variables. In Program 7, the resulting transposed value yields one column. If you want to use the ID variable as the variable in the ID statement (see program 8 below), PROC TRANSPOSE will transpose the data set, but the result might not be the one that you expected. Notice that the transposed values now occupy two columns, with A01 and A02 as their variable names. The problem is that you are using the ID variable, which contains two values to name the transposed variable that was supposed to occupy only one column.
Program 8:
proc transpose data=dat1_sort
   out=dat1_out8 (drop=_name_)
   label=TEST;
   by name;
   id id;
run;
proc print data=dat1_out8;
title 'incorrect way to use the ID statement';
run;

Output from Program 8:

<table>
<thead>
<tr>
<th>Obs</th>
<th>name</th>
<th>TEST</th>
<th>A01</th>
<th>A02</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>English1</td>
<td>89</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>English2</td>
<td>90</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>John</td>
<td>English3</td>
<td>92</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>Mary</td>
<td>English1</td>
<td>92</td>
<td>81</td>
</tr>
<tr>
<td>5</td>
<td>Mary</td>
<td>English2</td>
<td>.</td>
<td>92</td>
</tr>
<tr>
<td>6</td>
<td>Mary</td>
<td>English3</td>
<td>.</td>
<td>81</td>
</tr>
</tbody>
</table>

Program 9 illustrates a situation where the ID statement is necessary in order to transpose data correctly. PROC TRANSPOSE in program 9 transposes one variable, SCORE, by using the variable NAME as the BY-variable. The resulting transposed data set has two observations, which equals the number of BY-groups (2) times the number of variables that are transposed (1). The problem with the transposed data set is that the third test score (81) for Mary is placed in the location for the second test score.

Program 9:
data dat2;
   input name $ id $ exam score;
datalines;
John A01 1 89
John A01 2 90
John A01 3 92
Mary A02 1 92
Mary A02 3 81
;
proc sort data=dat2 out=dat2_sort;
   by name;
run;
proc transpose data=dat2_sort out=dat2_out1;
   var score;
   by name;
run;
proc print data=dat2_out1;
   title 'Incorrect way to transpose - ID statement is not used';
run;

Output from Program 9:

<table>
<thead>
<tr>
<th>Obs</th>
<th>name</th>
<th><em>NAME</em></th>
<th>COL1</th>
<th>COL2</th>
<th>COL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>score</td>
<td>89</td>
<td>90</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>Mary</td>
<td>score</td>
<td>92</td>
<td>81</td>
<td>.</td>
</tr>
</tbody>
</table>
Program 10 fixes the problem in Program 9 by using the variable EXAM in the ID statement. In addition, the PREFIX= option is also used to add "TEST_" as the prefix for transposed variable names.

Program 10:

```sas
proc transpose data=dat2_sort
   out=dat2_out2 (drop=_name_)
   prefix=test_
 var score;
 by name;
 id exam;
run;
```

```sas
proc print data=dat2_out2;
 title 'Correct way to transpose - ID statement is not used';
run;
```

**Output from Program 10:**

<table>
<thead>
<tr>
<th>Obs</th>
<th>name</th>
<th>test_1</th>
<th>test_2</th>
<th>test_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>89</td>
<td>90</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>Mary</td>
<td>92</td>
<td>.</td>
<td>81</td>
</tr>
</tbody>
</table>

**HANDLING DUPLICATES BY USING THE LET OPTION**

Consider the example in Program 11. There are double entries of the scores for the third test. PROC TRANSPOSE in Program 11 attempts to transpose `dat3` by using both the BY and ID statements. The ID statement uses the EXAM variable, which is not unique; hence, Program 11 fails to transpose `dat3` and generates an error message in the log (see log from Program 11). Without using the ID statement, PROC TRANSPOSE will be able to transpose `dat3`, but the results might not be what you intended because it will transpose the variable SCORE into four columns.

Program 11:

```sas
data dat3;
   input name $ id $ exam score;
   datalines;
   John A01 1 89
   John A01 2 90
   John A01 3 92
   John A01 3 95
   Mary A02 1 92
   Mary A02 3 81
   Mary A02 3 85
;```

```sas
proc transpose data=dat3
   out=dat3_out1 (drop=_name_)
   prefix=test_
 var score;
 by name;
 id exam;
run;
```
Log from Program 11:

```
proc transpose data=dat3
  out=dat3_out1 (drop=_name_)
  prefix=test_
var score;
by name;
id exam;
run;
```

ERROR: The ID value "test_3" occurs twice in the same BY group.
NOTE: The above message was for the following BY group:
      name=John
ERROR: The ID value "test_3" occurs twice in the same BY group.
NOTE: The above message was for the following BY group:
      name=Mary
ERROR: All BY groups were bad.
NOTE: The SAS System stopped processing this step because of errors.
NOTE: There were 7 observations read from the data set WORK.DAT3.
WARNING: The data set WORK.DAT3_OUT1 may be incomplete. When this step was stopped there were 0 observations and 0 variables.
WARNING: Data set WORK.DAT3_OUT1 was not replaced because this step was stopped.

NOTE: PROCEDURE TRANSPOSE used (Total process time):
    real time           0.03 seconds
    cpu time            0.03 seconds

For situations with duplicated records, you may want to keep only one record, such as keeping the largest or the smallest of the duplicated entries. The LET option from the PROC TRANSPOSE statement allows you to keep the last occurrence of a particular ID value within either the entire data set or a BY group.

Program 12 transposes dat3 by keeping the largest value of each EXAM within each group of NAME variable. Thus, it is necessary to sort the data by NAME first, followed by EXAM, and then SCORE in ascending order. Since the LET option only keeps the last occurrence of the ID value, PROC TRANSPOSE correctly transposes data with only the largest score within each EXAM. SAS detected the duplicated values that occurred in "test_3" in the same BY group; a WARNING message is generated in the log.

Program 12:
```
proc sort data=dat3 out=dat3_sort1;
  by name exam score;
run;

proc transpose data=dat3_sort1
  out=dat3_out1 (drop=_name_)
  prefix=test_
  let;
var score;
by name;
id exam;
run;

proc print data=dat3_out1;
  title 'Keep the maximum score';
run;
```
Log from Program 12:

```sas
277 proc transpose data=dat3_sort1
278      out=dat3_out1 (drop=_name_)
279      prefix=test_
280   let;
281   var score;
282   by name;
283   id exam;
284 run;
```

**WARNING:** The ID value "test_3" occurs twice in the same BY group.
**NOTE:** The above message was for the following BY group:
  name=John

**WARNING:** The ID value "test_3" occurs twice in the same BY group.
**NOTE:** The above message was for the following BY group:
  name=Mary

**NOTE:** There were 7 observations read from the data set WORK.DAT3_SORT1.

**NOTE:** The data set WORK.DAT3_OUT1 has 2 observations and 4 variables.

**NOTE:** PROCEDURE TRANSPOSE used (Total process time):
  real time   0.04 seconds
  cpu time    0.01 seconds

Output from Program 12:

```
Keep the maximum score

<table>
<thead>
<tr>
<th>Obs</th>
<th>name</th>
<th>test_1</th>
<th>test_2</th>
<th>test_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>89</td>
<td>90</td>
<td>95</td>
</tr>
<tr>
<td>2</td>
<td>Mary</td>
<td>92</td>
<td>.</td>
<td>85</td>
</tr>
</tbody>
</table>
```

If you want to keep the smallest SCORE instead of the largest in the transposed data, all you need to do is sort NAME and EXAM in ascending order and then sort SCORE in descending order. Program 13 illustrates how to keep the smallest SCORE of each EXAM with each BY variable.

**Program 13:**
```sas
proc sort data=dat3 out=dat3_sort2;
by name exam descending score;
run;

proc transpose data=dat3_sort2
   out=dat3_out2 (drop=_name_)
   prefix=test_
   let;
   var score;
   by name;
   id exam;
run;

proc print data=dat3_out2;
title 'Keep the minimum score';
run;
```

Output from Program 13:

```
Keep the minimum score

<table>
<thead>
<tr>
<th>Obs</th>
<th>name</th>
<th>test_1</th>
<th>test_2</th>
<th>test_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>89</td>
<td>90</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>Mary</td>
<td>92</td>
<td>.</td>
<td>81</td>
</tr>
</tbody>
</table>
```
SITUATIONS FOR TRANSPOSING DATA MORE THAN ONCE

In some applications, simply transposing data once will not produce the desired results. For example, to transpose dat4 to dat4_transpose (See Figure 2), you need to use PROC TRANSPOSE twice.

```
Dat4:
<table>
<thead>
<tr>
<th>Name</th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
</table>
 1    John 89 90 92 78 89 90 |
 2    Mary 92 . 81 76 91 89 |

Dat4 transpose:
<table>
<thead>
<tr>
<th>Test_num</th>
<th>John_e</th>
<th>John_m</th>
<th>Mary_e</th>
<th>Mary_m</th>
</tr>
</thead>
</table>
 1    1    89   78   92   76    |
 2    2    90   89   .    91    |
 3    3    92   90   81   89    |
```

Figure 2. SAS data sets, Dat4 and Dat4_Transpose.

Program 14a transposes dat4 by variable NAME. In the next step, you need to transpose COL1 from dat4_out1 into three rows. Before performing a second transposing, you need to sort the data by the test number and NAME. For example, the first observation (John, E1) should be followed by the 4th, 7th, and 10th rows. You also need to create a variable that contains the test number, which is the last character of the _NAME_ variable in dat4_out1.

Program 14a:
```
data dat4;
  input name $ e1 - e3 m1 - m3;
datalines;
John 89 90 92 78 89 90
Mary 92 . 81 76 91 89
;
proc sort data=dat4 out=dat4_sort1;
  by name;
run;
proc transpose data=dat4_sort1 out=dat4_out1;
  by name;
run;
proc print data=dat4_out1;
  title 'First use of PROC TRANSPOSE for dat4';
run;
```

Output from Program 14a:
```
Obs name   _NAME_  COL1
  |      |        |    |
 1  John e1   89  |
 2  John e2   90  |
 3  John e3   92  |
 4  John m1   78  |
 5  John m2   89  |
 6  John m3   90  |
 7  Mary e1   92  |
 8  Mary e2   .   |
 9  Mary e3   81  |
10 Mary m1   76  |
11 Mary m2   91  |
12 Mary m3   89  |
```
Program 14b uses the SUBSTR function to create the TEST_NUM and CLASS variables by taking the last and first characters of the _NAME_ variable.

Program 14b:

```sas
data dat4_out1a;
  set dat4_out1;
  test_num=substr(_name_,2);
  class=substr(_name_,1,1);
run;
```

```sas
proc print data=dat4_out1a;
  title 'Creating TEST_NUM and CLASS variables';
run;
```

Output from Program 14b:

<table>
<thead>
<tr>
<th>Obs</th>
<th>name</th>
<th><em>NAME</em></th>
<th>COL1</th>
<th>test_num</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>e1</td>
<td>89</td>
<td>1</td>
<td>e</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>e2</td>
<td>90</td>
<td>2</td>
<td>e</td>
</tr>
<tr>
<td>3</td>
<td>John</td>
<td>e3</td>
<td>92</td>
<td>3</td>
<td>e</td>
</tr>
<tr>
<td>4</td>
<td>John</td>
<td>m1</td>
<td>78</td>
<td>1</td>
<td>m</td>
</tr>
<tr>
<td>5</td>
<td>John</td>
<td>m2</td>
<td>89</td>
<td>2</td>
<td>m</td>
</tr>
<tr>
<td>6</td>
<td>John</td>
<td>m3</td>
<td>90</td>
<td>3</td>
<td>m</td>
</tr>
<tr>
<td>7</td>
<td>Mary</td>
<td>e1</td>
<td>92</td>
<td>1</td>
<td>e</td>
</tr>
<tr>
<td>8</td>
<td>Mary</td>
<td>e2</td>
<td>.</td>
<td>2</td>
<td>e</td>
</tr>
<tr>
<td>9</td>
<td>Mary</td>
<td>e3</td>
<td>81</td>
<td>3</td>
<td>e</td>
</tr>
<tr>
<td>10</td>
<td>Mary</td>
<td>m1</td>
<td>76</td>
<td>1</td>
<td>m</td>
</tr>
<tr>
<td>11</td>
<td>Mary</td>
<td>m2</td>
<td>91</td>
<td>2</td>
<td>m</td>
</tr>
<tr>
<td>12</td>
<td>Mary</td>
<td>m3</td>
<td>89</td>
<td>3</td>
<td>m</td>
</tr>
</tbody>
</table>

Program 14c sorts the data by TEST_NUM and NAME. Notice that the test scores in COL1 have the desired order.

Program 14c:

```sas
proc sort data=dat4_out1a out=dat4_sort2;
  by test_num name;
run;
```

```sas
proc print data=dat4_sort2;
  title 'Sort data by TEST_NUM and NAME';
run;
```

Output from Program 14c:

<table>
<thead>
<tr>
<th>Obs</th>
<th>name</th>
<th><em>NAME</em></th>
<th>COL1</th>
<th>test_num</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>e1</td>
<td>89</td>
<td>1</td>
<td>e</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>m1</td>
<td>78</td>
<td>1</td>
<td>m</td>
</tr>
<tr>
<td>3</td>
<td>Mary</td>
<td>e1</td>
<td>92</td>
<td>1</td>
<td>e</td>
</tr>
<tr>
<td>4</td>
<td>Mary</td>
<td>m1</td>
<td>76</td>
<td>1</td>
<td>m</td>
</tr>
<tr>
<td>5</td>
<td>John</td>
<td>e2</td>
<td>90</td>
<td>2</td>
<td>e</td>
</tr>
<tr>
<td>6</td>
<td>John</td>
<td>m2</td>
<td>89</td>
<td>2</td>
<td>m</td>
</tr>
<tr>
<td>7</td>
<td>Mary</td>
<td>e2</td>
<td>.</td>
<td>2</td>
<td>e</td>
</tr>
<tr>
<td>8</td>
<td>Mary</td>
<td>m2</td>
<td>91</td>
<td>2</td>
<td>m</td>
</tr>
<tr>
<td>9</td>
<td>John</td>
<td>e3</td>
<td>92</td>
<td>3</td>
<td>e</td>
</tr>
<tr>
<td>10</td>
<td>John</td>
<td>m3</td>
<td>90</td>
<td>3</td>
<td>m</td>
</tr>
<tr>
<td>11</td>
<td>Mary</td>
<td>e3</td>
<td>81</td>
<td>3</td>
<td>e</td>
</tr>
<tr>
<td>12</td>
<td>Mary</td>
<td>m3</td>
<td>89</td>
<td>3</td>
<td>m</td>
</tr>
</tbody>
</table>
PROC TRANSPOSE in Program 14d transposes COL1 by variable TEST and uses NAME and CLASS as the ID variables. The names of the transposed variables are separated by the underscore from the DELIMITER= option.

Program 14d:
```
proc transpose data=dat4_sort2
  out=dat4_out2(drop=_name_)
  delimiter=;
  by test_num;
  var col1;
  id name class;
run;
```

```
proc print data=dat4_out2;
  title 'Second use of PROC TRANSPOSE for dat4';
run;
```

Output from Program 14d:
```
<table>
<thead>
<tr>
<th>Obs</th>
<th>test_num</th>
<th>John_e</th>
<th>John_m</th>
<th>Mary_e</th>
<th>Mary_m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>89</td>
<td>78</td>
<td>92</td>
<td>76</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>90</td>
<td>89</td>
<td>.</td>
<td>91</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>92</td>
<td>90</td>
<td>81</td>
<td>89</td>
</tr>
</tbody>
</table>
```

CONCLUSION
PROC TRANSPOSE is a powerful procedure to perform data transposition. In addition to grasping the syntax, more importantly, you need to know when best to utilize different options and statements to achieve the desired results.

ACKNOWLEDGMENTS
I would like to thank Jerry Leonard, Technical Support Analyst from SAS Technical Support, for his valuable programming suggestions and insight.

CONTACT INFORMATION
Arthur Li
City of Hope Comprehensive Cancer Center
Division of Information Science
1500 East Duarte Road
Duarte, CA 91010 - 3000
Work Phone: (626) 256-4673 ext. 65121
Fax: (626) 471-7106
E-mail: xueli@coh.org

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.