CHAPTER 39

# The TRANSPOSE Procedure

## Overview

The TRANSPOSE procedure creates an output data set by restructuring the values in a SAS data set, transposing selected variables into observations. The TRANSPOSE procedure can often eliminate the need to write a lengthy DATA step to achieve the same result. Further, the output data set can be used in subsequent DATA or PROC steps for analysis, reporting, or further data manipulation.

PROC TRANSPOSE does not produce printed output. To print the output data set from the PROC TRANSPOSE step, use PROC PRINT, PROC REPORT, or another SAS reporting tool.

A *transposed variable* is a variable the procedure creates by transposing the values of an observation in the input data set into values of a variable in the output data set.

Output 39.1 on page 1270 illustrates a simple transposition. In the input data set, each *variable* represents the scores from one tester. In the output data set, each *observation* now represents the scores from one tester. Each value of _NAME_ is the name of a variable in the input data set that the procedure transposed. Thus, the value of _NAME_ identifies the source of each observation in the output data set. For example, the values in the first observation in the output data set come from the values of the variable Tester1 in the input data set. The statements that produce the output follow.

```
proc print data=proclib.product noobs;
   title 'The Input Data Set';
run;

proc transpose data=proclib.product
               out=proclib.product_transposed;
run;

proc print data=proclib.product_transposed noobs;
   title 'The Output Data Set';
run;
```

**Output 39.1** A Simple Transposition

```
                          The Input Data Set                              1

                Tester1    Tester2    Tester3    Tester4

                   22         25         21         21
                   15         19         18         17
                   17         19         19         19
                   20         19         16         19
                   14         15         13         13
                   15         17         18         19
                   10         11          9         10
                   22         24         23         21
```

```
                         The Output Data Set                              2

    _NAME_     COL1     COL2     COL3     COL4     COL5     COL6     COL7     COL8

    Tester1     22       15       17       20       14       15       10       22
    Tester2     25       19       19       19       15       17       11       24
    Tester3     21       18       19       16       13       18        9       23
    Tester4     21       17       19       19       13       19       10       21
```

Output 39.2 on page 1270 is a more complex example that uses BY groups. The input data set represents measurements of fish weight and length at two lakes. The statements that create the output data set

- □ transpose only the variables that contain the length measurements
- □ create six BY groups, one for each lake and date
- □ use a data set option to name the transposed variable.

**Output 39.2** A Transposition with BY Groups

```
                              Input Data Set                              1

     L
     o                  L      W      L      W      L      W      L      W
     c                  e      e      e      e      e      e      e      e
     a                  n      i      n      i      n      i      n      i
     t            D     g      g      g      g      g      g      g      g
     i            a     t      h      t      h      t      h      t      h
     o            t     h      t      h      t      h      t      h      t
     n            e     1      1      2      2      3      3      4      4

  Cole Pond    02JUN95    31    0.25    32    0.30    32    0.25    33    0.30
  Cole Pond    03JUL95    33    0.32    34    0.41    37    0.48    32    0.28
  Cole Pond    04AUG95    29    0.23    30    0.25    34    0.47    32    0.30
  Eagle Lake   02JUN95    32    0.35    32    0.25    33    0.30     .      .
  Eagle Lake   03JUL95    30    0.20    36    0.45     .      .      .      .
  Eagle Lake   04AUG95    33    0.30    33    0.28    34    0.42     .      .
```

```
                 Fish Length Data for Each Location and Date               2

                Location        Date    _NAME_     Measurement

               Cole Pond     02JUN95    Length1         31
               Cole Pond     02JUN95    Length2         32
               Cole Pond     02JUN95    Length3         32
               Cole Pond     02JUN95    Length4         33
               Cole Pond     03JUL95    Length1         33
               Cole Pond     03JUL95    Length2         34
               Cole Pond     03JUL95    Length3         37
               Cole Pond     03JUL95    Length4         32
               Cole Pond     04AUG95    Length1         29
               Cole Pond     04AUG95    Length2         30
               Cole Pond     04AUG95    Length3         34
               Cole Pond     04AUG95    Length4         32
               Eagle Lake    02JUN95    Length1         32
               Eagle Lake    02JUN95    Length2         32
               Eagle Lake    02JUN95    Length3         33
               Eagle Lake    02JUN95    Length4          .
               Eagle Lake    03JUL95    Length1         30
               Eagle Lake    03JUL95    Length2         36
               Eagle Lake    03JUL95    Length3          .
               Eagle Lake    03JUL95    Length4          .
               Eagle Lake    04AUG95    Length1         33
               Eagle Lake    04AUG95    Length2         33
               Eagle Lake    04AUG95    Length3         34
               Eagle Lake    04AUG95    Length4          .
```

For a complete explanation of the SAS program that produces Output 39.2 on page 1270, see Example 4 on page 1282.

## Procedure Syntax

**Tip:** Does not support the Output Delivery System

**Reminder:** You can use the ATTRIB, FORMAT, LABEL, and WHERE statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," for details. You can also use any global statements as well. See Chapter 2, "Fundamental Concepts for Using Base SAS Procedures," for a list.

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

| To do this | Use this statement |
|---|---|
| Transpose each BY group | BY |
| Copy variables directly without transposing them | COPY |
| Specify a variable whose values name the transposed variables | ID |
| Create labels for the transposed variables | IDLABEL |
| List the variables to transpose | VAR |

## PROC TRANSPOSE Statement

**Reminder:** You can use data set options with the DATA= and OUT= options. See Chapter 2, "Fundamental Concepts for Using Base SAS Procedures," for a list.

**PROC TRANSPOSE** <DATA=*input-data-set*> <LABEL=*label*> <LET>
  <NAME=*name*> <OUT=*output-data-set*> <PREFIX=*prefix*>;

### Options

**DATA=** ***input-data-set***
names the SAS data set to transpose.

**Default:** most recently created SAS data set

**LABEL=** ***label***
specifies a name for the variable in the output data set that contains the label of the variable that is being transposed to create the current observation.

**Default:** _LABEL_

**LET**
allows duplicate values of an ID variable. PROC TRANSPOSE transposes the observation containing the last occurrence of a particular ID value within the data set or BY group.

**Featured in:** Example 5 on page 1284

**NAME=** ***name***
specifies the name for the variable in the output data set that contains the name of the variable being transposed to create the current observation.

**Default:** _NAME_

**Featured in:** Example 2 on page 1280

**OUT=** ***output-data-set***
names the output data set. If *output-data-set* does not exist, PROC TRANSPOSE creates it using the DATA*n* naming convention.

**Default:** DATA*n*

**Featured in:** Example 1 on page 1278

**PREFIX=** ***prefix***
specifies a prefix to use in constructing names for transposed variables in the output data set. For example, if PREFIX=VAR, the names of the variables are VAR1, VAR2, . . . ,VAR*n*.

**Interaction:** when you use PREFIX= with an ID statement, the value prefixes to the ID value.

**Featured in:** Example 2 on page 1280

## BY Statement

**Defines BY groups.**

**Main discussion:** "BY" on page 68

**Featured in:** Example 4 on page 1282

**Restriction:** You cannot use PROC TRANSPOSE with a BY statement or an ID statement with an engine that supports concurrent access if another user is updating the data set at the same time.

### Required Arguments

***variable***
specifies the variable that PROC TRANSPOSE uses to form BY groups. You can specify more than one variable. If you do not use the NOTSORTED option in the BY statement, the observations must be either sorted by all the variables that you specify, or they must be indexed appropriately. Variables in a BY statement are called *BY variables*.

### Options

**DESCENDING**
specifies that the data set is sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

**NOTSORTED**
specifies that observations are not necessarily sorted in alphabetic or numeric order. The data are grouped in another way, for example, chronological order.

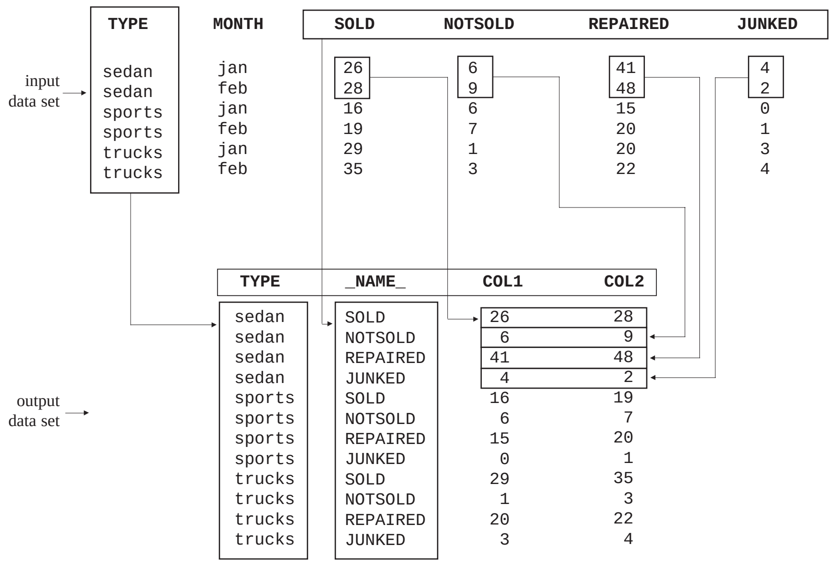
The requirement for ordering or indexing observations according to the values of BY variables is suspended for BY-group processing when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. The procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same values for the BY variables are not contiguous, the procedure treats each contiguous set as a separate BY group.

## Transpositions with BY Groups

PROC TRANSPOSE does not transpose BY groups. Instead, for each BY group, PROC TRANSPOSE creates one observation for each variable that it transposes.

Figure 39.1 on page 1274 shows what happens when you transpose a data set with BY groups. TYPE is the BY variable, and SOLD, NOTSOLD, REPAIRED, and JUNKED are the variables to transpose.

**Figure 39.1** Transposition with BY Groups



| TYPE | MONTH | SOLD | NOTSOLD | REPAIRED | JUNKED |
|---|---|---|---|---|---|
| sedan | jan | 26 | 6 | 41 | 4 |
| sedan | feb | 28 | 9 | 48 | 2 |
| sports | jan | 16 | 6 | 15 | 0 |
| sports | feb | 19 | 7 | 20 | 1 |
| trucks | jan | 29 | 1 | 20 | 3 |
| trucks | feb | 35 | 3 | 22 | 4 |

| TYPE | _NAME_ | COL1 | COL2 |
|---|---|---|---|
| sedan | SOLD | 26 | 28 |
| sedan | NOTSOLD | 6 | 9 |
| sedan | REPAIRED | 41 | 48 |
| sedan | JUNKED | 4 | 2 |
| sports | SOLD | 16 | 19 |
| sports | NOTSOLD | 6 | 7 |
| sports | REPAIRED | 15 | 20 |
| sports | JUNKED | 0 | 1 |
| trucks | SOLD | 29 | 35 |
| trucks | NOTSOLD | 1 | 3 |
| trucks | REPAIRED | 20 | 22 |
| trucks | JUNKED | 3 | 4 |

- □ The number of observations in the output data set (12) is the number of BY groups (3) multiplied by the number of variables that are transposed (4).
- □ The BY variable is not transposed.
- □ _NAME_ contains the name of the variable in the input data set that was transposed to create the current observation in the output data set. You can use the NAME= option to specify another name for the _NAME_ variable.
- □ The maximum number of observations in any BY group in the input data set is two; therefore, the output data set contains two variables, COL1 and COL2. COL1 and COL2 contain the values of SOLD, NOTSOLD, REPAIRED, and JUNKED.

*Note:* If a BY group in the input data set has more observations than other BY groups, PROC TRANSPOSE assigns missing values in the output data set to the variables that have no corresponding input observations. △

## COPY Statement

**Copies variables directly from the input data set to the output data set without transposing them.**

**Featured in:** Example 6 on page 1286

**COPY** *variable(s)*;

### Required Argument

***variable(s)***
names one or more variables that the COPY statement copies directly from the input data set to the output data set without transposing them.

### Details

Because the COPY statement copies variables directly to the output data set, the number of observations in the output data set is equal to the number of observations in the input data set.

The procedure pads the output data set with missing values if the number of observations in the input data set and the number of variables it transposes are not equal.

## ID Statement

**Specifies a variable in the input data set whose formatted values name the transposed variables in the output data set.**

**Featured in:** Example 2 on page 1280

**Restriction:** You cannot use PROC TRANSPOSE with an ID statement or a BY statement with an engine that supports concurrent access if another user is updating the data set at the same time.

**ID** *variable*;

### Required Argument

***variable***
names the variable whose formatted values name the transposed variables.

### Duplicate ID Values

Typically, each formatted ID value occurs only once in the input data set or, if you use a BY statement, only once within a BY group. Duplicate values cause PROC TRANSPOSE to issue a warning message and stop. However, if you use the LET option in the PROC TRANSPOSE statement, the procedure issues a warning message about duplicate ID values and transposes the observation containing the last occurrence of the duplicate ID value.

### Making Variable Names Out of Numeric Values

When you use a numeric variable as an ID variable, PROC TRANSPOSE changes the formatted ID value into a valid SAS name.

However, SAS variable names cannot begin with a number. Thus, when the first character of the formatted value is numeric, the procedure prefixes an underscore to the value, truncating the last character of an 32-character value. Any remaining invalid characters are replaced by underscores. The procedure truncates to 32 characters any ID value that is longer than 32 characters when it uses that value to name a transposed variable.

If the formatted value looks like a numeric constant, PROC TRANSPOSE changes the characters '+', '–', and '.' to 'P','N', and 'D', respectively. If the formatted value has characters that are not numerics, PROC TRANSPOSE changes the characters '+', '–', and '.' to underscores.

*Note:* If the value of the VALIDVARNAME system option is V6, PROC TRANSPOSE truncates transposed variable names to eight characters. △

### Missing Values

If you use an ID variable that contains a missing value, PROC TRANSPOSE writes an error message to the log. The procedure does not transpose observations that have a missing value for the ID variable.

## IDLABEL Statement

**Creates labels for the transposed variables.**

**Restriction:** Must appear after an ID statement.

**Featured in:** Example 3 on page 1281

**IDLABEL** *variable*;

### Required Argument

***variable***
names the variable whose values the procedure uses to label the variables that the ID statement names. *variable* can be character or numeric.

*Note:* To see the effect of the IDLABEL statement, print the output data set with the PRINT procedure using the LABEL option, or print the contents of the output data set using the CONTENTS statement in the DATASETS procedure. △

## VAR Statement

**Lists the variables to transpose.**

**Featured in:** Example 4 on page 1282 and Example 6 on page 1286

**VAR** *variable(s)*;

### Required Argument

***variable(s)***
names one or more variables to transpose.

### Details

- □ If you omit the VAR statement, the TRANSPOSE procedure transposes all numeric variables in the input data set that are not listed in another statement.
- □ You must list character variables in a VAR statement if you want to transpose them.

## Results

### Output Data Set

The TRANSPOSE procedure always produces an output data set, regardless of whether you specify the OUT= option in the PROC TRANSPOSE statement. PROC TRANSPOSE does not print the output data set. Use PROC PRINT, PROC REPORT or some other SAS reporting tool to print the output data set.

The output data set contains the following variables:

- □ variables that result from transposing the values of each variable into an observation.
- □ a variable that PROC TRANSPOSE creates to identify the source of the values in each observation in the output data set. This variable is a character variable whose values are the names of the variables transposed from the input data set. By default, PROC TRANSPOSE names this variable _NAME_. To override the default name, use the NAME= option. The label for the _NAME_ variable is NAME OF FORMER VARIABLE.
- □ variables that PROC TRANSPOSE copies from the input data set when you use either the BY or COPY statement. These variables have the same names and values as they do in the input data set.
- □ a character variable whose values are the variable labels of the variables being transposed (if any of the variables the procedure is transposing have labels). Specify the name of the variable with the LABEL= option. The default is **`_LABEL_`**.

*Note:* If the value of the LABEL= option or the NAME= option is the same as a variable that appears in a BY or COPY statement, the output data set does not contain a variable whose values are the names or labels of the transposed variables. △

### Attributes of Transposed Variables

- All transposed variables are the same type and length.
- If all variables that the procedure is transposing are numeric, the transposed variables are numeric. Thus, if the numeric variable has a character string as a formatted value, its unformatted numeric value is transposed.
- If any variable that the procedure is transposing is character, all transposed variables are character. Thus, if you are transposing a numeric variable that has a character string as a formatted value, the formatted value is transposed.
- The length of the transposed variables is equal to the length of the longest variable being transposed.

### Names of Transposed Variables

PROC TRANSPOSE names transposed variables using the following rules:

1. An ID statement specifies a variable in the input data set whose formatted values become names for the transposed variables.
2. The PREFIX= option specifies a prefix to use in constructing the names of transposed variables.
3. If you do not use an ID statement or the PREFIX= option, PROC TRANSPOSE looks for an input variable called _NAME_ from which to get the names of the transposed variables.
4. If you do not use an ID statement or the PREFIX= option, and the input data set does not contain a variable named _NAME_, PROC TRANSPOSE assigns the names COL1, COL2, . . . , COL*n* to the transposed variables.

## Examples

## Example 1: Performing a Simple Transposition

**Procedure features:**

PROC TRANSPOSE statement option:

OUT=

This example performs a default transposition and uses no subordinate statements.

### Program

```
options nodate pageno=1 linesize=80 pagesize=40;
```

The data set SCORE contains students' names, their identification numbers, and their grades on two tests and a final exam.

```
data score;
   input Student $9. +1 StudentID $ Section $ Test1 Test2 Final;
   datalines;
Capalleti 0545 1  94 91 87
Dubose    1252 2  51 65 91
Engles    1167 1  95 97 97
Grant     1230 2  63 75 80
Krupski   2527 2  80 76 71
Lundsford 4860 1  92 40 86
Mcbane    0674 1  75 78 72
;
```

PROC TRANSPOSE transposes only the numeric variables, Test1, Test2, and Final because no VAR statement appears and none of the numeric variables appear in another statement. OUT= puts the result of the transposition in the SCORE_TRANSPOSED data set.

```
proc transpose data=score out=score_transposed;
run;
```

PROC PRINT prints the output data set.

```
proc print data=score_transposed noobs;
    title 'Student Test Scores in Variables';
run;
```

## Output

In the output data set SCORE_TRANSPOSED, variables COL1 through COL7 contain the individual scores for the students. Each observation contains all the scores for one test. The _NAME_ variable contains the names of the variables from the input data set that were transposed.

```
                   Student Test Scores in Variables                    1

        _NAME_    COL1    COL2    COL3    COL4    COL5    COL6    COL7

        Test1      94      51      95      63      80      92      75
        Test2      91      65      97      75      76      40      78
        Final      87      91      97      80      71      86      72
```

## Example 2: Naming Transposed Variables

**Procedure features:**

PROC TRANSPOSE statement options:

NAME=

PREFIX=

ID statement

**Data set:** SCORE on page 1279

This example uses the values of a variable and a user-supplied value to name transposed variables.

### Program

```
options nodate pageno=1 linesize=80 pagesize=40;
```

PROC TRANSPOSE transposes only the numeric variables, Test1, Test2, and Final because no VAR statement appears. OUT= puts the result of the transposition in the IDNUMBER data set. NAME= specifies Test as the name for the variable that contains the names of the variables in the input data set that the procedure transposes. The procedure names the transposed variables by using the value from PREFIX=, sn, and the value of the ID variable StudentID

```
proc transpose data=score out=idnumber name=Test
     prefix=sn;
    id studentid;
run;
```

PROC PRINT prints the data set.

```
proc print data=idnumber noobs;
   title 'Student Test Scores';
run;
```

### Output

The output data set, IDNUMBER

```
                         Student Test Scores                              1

Test     sn0545    sn1252    sn1167    sn1230    sn2527    sn4860    sn0674

Test1      94        51        95        63        80        92        75
Test2      91        65        97        75        76        40        78
Final      87        91        97        80        71        86        72
```

## Example 3: Labeling Transposed Variables

**Procedure features:**
PROC TRANSPOSE statement option:
PREFIX=
IDLABEL statement

**Data set:** SCORE on page 1279

This example uses the values of the variable in the IDLABEL statement to label transposed variables.

### Program

```
options nodate pageno=1 linesize=80 pagesize=40;
```

PROC TRANSPOSE transposes only the numeric variables, Test1, Test2, and Final because no VAR statement appears. OUT= puts the result of the transposition in the IDLABEL data set. NAME= specifies Test as the name for the variable that contains the names of the variables in the input data set that the procedure transposes. The procedure names the transposed variables by using the value from PREFIX=, sn, and the value of the ID variable StudentID.

```
proc transpose data=score out=idlabel name=Test
     prefix=sn;
   id studentid;
```

PROC TRANSPOSE uses the values of the variable Student to label the transposed variables. The procedure provides

```
NAME OF FORMER VARIABLE
```

as the label for the _NAME_ variable.

```
     idlabel student;
run;
```

PROC PRINT prints the output data set and uses the variable labels as column headers. The LABEL option causes PROC PRINT to print variable labels for column headers.

```
  proc print data=idlabel label noobs;
     title 'Student Test Scores';
  run;
```

### Output

The output data set, IDLABEL

```
                              Student Test Scores                              1

NAME OF
 FORMER
VARIABLE   Capalleti   Dubose   Engles   Grant   Krupski   Lundsford   Mcbane

 Test1         94        51       95       63       80         92         75
 Test2         91        65       97       75       76         40         78
 Final         87        91       97       80       71         86         72
```

## Example 4: Transposing BY Groups

**Procedure features:**
BY statement
VAR statement
**Other features:** Data set option:
RENAME=

This example illustrates transposing BY groups and selecting variables to transpose.

### Program

```
options nodate pageno=1 linesize=80 pagesize=40;
```

The input data represent length and weight measurements of fish caught at two ponds on three separate days. The data are sorted by Location and Date.

```
data fishdata;
   infile datalines missover;
   input Location & $10. Date date7.
         Length1 Weight1 Length2 Weight2 Length3 Weight3
         Length4 Weight4;
   format date date7.;
   datalines;
Cole Pond   2JUN95 31 .25 32 .3  32 .25 33 .3
Cole Pond   3JUL95 33 .32 34 .41 37 .48 32 .28
Cole Pond   4AUG95 29 .23 30 .25 34 .47 32 .3
Eagle Lake  2JUN95 32 .35 32 .25 33 .30
Eagle Lake  3JUL95 30 .20 36 .45
Eagle Lake  4AUG95 33 .30 33 .28 34 .42
;
```

OUT= puts the result of the transposition in the FISHLENGTH data set. RENAME= renames COL1 in the output data set to Measurement.

```
proc transpose data=fishdata
     out=fishlength(rename=(col1=Measurement));
```

PROC TRANSPOSE transposes only the Length1-Length4 variables because they appear in the VAR statement.

```
   var length1-length4;
```

The BY statement creates BY groups for each unique combination of values of Location and Date. The procedure does not transpose the BY variables.

```
   by location date;
run;
```

PROC PRINT prints the output data set.

```
proc print data=fishlength noobs;
   title 'Fish Length Data for Each Location and Date';
run;
```

## Output

The output data set, FISHLENGTH. For each BY group in the original data set, PROC TRANSPOSE creates four observations, one for each variable it is transposing. Missing values appear for the variable Measurement (renamed from COL1) when the variables being transposed have no value in the input data set for that BY group. Several observations have a missing value for Measurement. For example, in the last observation, a missing value appears because there was no value for Length4 on 04AUG95 at Eagle Lake in the input data.

```
              Fish Length Data for Each Location and Date                 1

             Location        Date    _NAME_     Measurement

            Cole Pond     02JUN95    Length1         31
            Cole Pond     02JUN95    Length2         32
            Cole Pond     02JUN95    Length3         32
            Cole Pond     02JUN95    Length4         33
            Cole Pond     03JUL95    Length1         33
            Cole Pond     03JUL95    Length2         34
            Cole Pond     03JUL95    Length3         37
            Cole Pond     03JUL95    Length4         32
            Cole Pond     04AUG95    Length1         29
            Cole Pond     04AUG95    Length2         30
            Cole Pond     04AUG95    Length3         34
            Cole Pond     04AUG95    Length4         32
            Eagle Lake    02JUN95    Length1         32
            Eagle Lake    02JUN95    Length2         32
            Eagle Lake    02JUN95    Length3         33
            Eagle Lake    02JUN95    Length4          .
            Eagle Lake    03JUL95    Length1         30
            Eagle Lake    03JUL95    Length2         36
            Eagle Lake    03JUL95    Length3          .
            Eagle Lake    03JUL95    Length4          .
            Eagle Lake    04AUG95    Length1         33
            Eagle Lake    04AUG95    Length2         33
            Eagle Lake    04AUG95    Length3         34
            Eagle Lake    04AUG95    Length4          .
```

## Example 5: Naming Transposed Variables When the ID Variable Has Duplicate Values

**Procedure features:**

PROC TRANSPOSE statement option:

LET

This example shows how to use values of a variable (ID) to name transposed variables even when the ID variable has duplicate values.

### Program

```
options nodate pageno=1 linesize=64 pagesize=40;
```

STOCKS contains stock prices for two competing kite manufacturers. The prices are recorded three times a day: at opening, at noon, and at closing, on two days. Notice that the input data set contains duplicate values for the Date variable.

```
data stocks;
    input Company $14. Date $ Time $ Price;
    datalines;
Horizon Kites jun11 opening 29
Horizon Kites jun11 noon    27
Horizon Kites jun11 closing 27
Horizon Kites jun12 opening 27
Horizon Kites jun12 noon    28
Horizon Kites jun12 closing 30
SkyHi Kites   jun11 opening 43
SkyHi Kites   jun11 noon    43
SkyHi Kites   jun11 closing 44
SkyHi Kites   jun12 opening 44
SkyHi Kites   jun12 noon    45
SkyHi Kites   jun12 closing 45
;
```

LET transposes only the last observation for each BY group. PROC TRANSPOSE transposes only the Price variable. OUT= puts the result of the transposition in the CLOSE data set.

```
proc transpose data=stocks out=close let;
```

The BY statement creates two BY groups, one for each company.

```
   by company;
```

The values of Date are used as names for the transposed variables.

```
   id date;
 run;
```

PROC PRINT prints the output data set.

```
proc print data=close noobs;
   title 'Closing Prices for Horizon Kites and SkyHi Kites';
run;
```

## Output

The output data set, CLOSE

```
        Closing Prices for Horizon Kites and SkyHi Kites       1

              Company       _NAME_    jun11    jun12

           Horizon Kites    Price       27       30
           SkyHi Kites      Price       44       45
```

## Example 6: Transposing Data for Statistical Analysis

**Procedure features:**

COPY statement

VAR statement

This example arranges data to make them suitable for either a multivariate or univariate repeated-measures analysis.

The data are from Chapter 8, "Repeated-Measures Analysis of Variance" in *SAS System for Linear Models, Third Edition.*

### Program 1

```
options nodate pageno=1 linesize=80 pagesize=40;
```

The data represent the results of an exercise therapy study of three weight-lifting programs: CONT is control, RI is a program in which the number of repetitions are increased, and WI is a program in which the weight is increased.

```
data weights;
   input Program $ s1-s7;
   datalines;
CONT  85 85 86 85 87 86 87
CONT  80 79 79 78 78 79 78
CONT  78 77 77 77 76 76 77
CONT  84 84 85 84 83 84 85
CONT  80 81 80 80 79 79 80
RI    79 79 79 80 80 78 80
RI    83 83 85 85 86 87 87
RI    81 83 82 82 83 83 82
RI    81 81 81 82 82 83 81
RI    80 81 82 82 82 84 86
WI    84 85 84 83 83 83 84
WI    74 75 75 76 75 76 76
WI    83 84 82 81 83 83 82
WI    86 87 87 87 87 87 86
WI    82 83 84 85 84 85 86
;
```

The DATA step rearranges WEIGHTS to create the data set SPLIT. The DATA step transposes the strength values and creates two new variables: Time and Subject. SPLIT contains one observation for each repeated measure. SPLIT can be used in a PROC GLM step for a univariate repeated-measures analysis.

```
data split;
   set weights;
   array s{7} s1-s7;
   Subject + 1;
   do Time=1 to 7;
      Strength=s{time};
      output;
   end;
   drop s1-s7;
run;
```

PROC PRINT prints the data set. The OBS= data set option limits the printing to the first 15 observations. SPLIT has 105 observations.

```
proc print data=split(obs=15) noobs;
   title 'SPLIT Data Set';
   title2 'First 15 Observations Only';
run;
```

## Output 1

```
                         SPLIT Data Set                            1
                   First 15 Observations Only

             Program    Subject    Time    Strength

              CONT         1         1        85
              CONT         1         2        85
              CONT         1         3        86
              CONT         1         4        85
              CONT         1         5        87
              CONT         1         6        86
              CONT         1         7        87
              CONT         2         1        80
              CONT         2         2        79
              CONT         2         3        79
              CONT         2         4        78
              CONT         2         5        78
              CONT         2         6        79
              CONT         2         7        78
              CONT         3         1        78
```

## Program 2

```
options nodate pageno=1 linesize=80 pagesize=40;
```

PROC TRANSPOSE transposes SPLIT to create TOTSPLIT. The TOTSPLIT data set contains the same variables as SPLIT and a variable for each strength measurement (Str1-Str7). TOTSPLIT can be used for either a multivariate repeated-measures analysis or for a univariate repeated-measures analysis.

```
proc transpose data=split out=totsplit prefix=Str;
```

The variables in the BY and COPY statements are not transposed. TOTSPLIT contains the variables Program, Subject, Time, and Strength with the same values that are in SPLIT. The BY statement creates the first observation in each BY group, which contains the transposed values of Strength. The COPY statement creates the other observations in each BY group by copying the values of Time and Strength without transposing them.

```
   by program subject;
   copy time strength;
```

The VAR statement specifies the Strength variable as the only variable to be transposed.

```
   var strength;
run;
```

PROC PRINT prints the output data set.

```
proc print data=totsplit(obs=15) noobs;
   title  'TOTSPLIT Data Set';
   title2 'First 15 Observations Only';
run;
```

## Output 2

The variables in TOTSPLIT with missing values are used only in a multivariate repeated–measures analysis. The missing values do not preclude this data set from being used in a repeated-measures analysis because the MODEL statement in PROC GLM ignores observations with missing values.

```
                           TOTSPLIT Data Set                              1
                        First 15 Observations Only

Program Subject Time Strength  _NAME_  Str1 Str2 Str3 Str4 Str5 Str6 Str7

  CONT       1      1      85    Strength  85   85   86   85   87   86   87
  CONT       1      2      85               .    .    .    .    .    .    .
  CONT       1      3      86               .    .    .    .    .    .    .
  CONT       1      4      85               .    .    .    .    .    .    .
  CONT       1      5      87               .    .    .    .    .    .    .
  CONT       1      6      86               .    .    .    .    .    .    .
  CONT       1      7      87               .    .    .    .    .    .    .
  CONT       2      1      80    Strength  80   79   79   78   78   79   78
  CONT       2      2      79               .    .    .    .    .    .    .
  CONT       2      3      79               .    .    .    .    .    .    .
  CONT       2      4      78               .    .    .    .    .    .    .
  CONT       2      5      78               .    .    .    .    .    .    .
  CONT       2      6      79               .    .    .    .    .    .    .
  CONT       2      7      78               .    .    .    .    .    .    .
  CONT       3      1      78    Strength  78   77   77   77   76   76   77
```

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Procedures Guide, Version 8,* Cary, NC: SAS Institute Inc., 1999. 1729 pp.

**SAS® Procedures Guide, Version 8**

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1–58025–482–9

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999