Introduction

This paper will describe how to transform data sets using arrays. First, let us discuss what we mean by the term "transforming." You may want to create multiple observations from a single observation or vice versa. There are several possible reasons why you may want to do this. You may want to create multiple observations from a single observation to count frequencies or to allow for BY variable processing. You may also want to restructure SAS data sets for certain statistical analyses. Creating a single observation from multiple observations may make it easier for you to compute differences between values without resorting to LAG functions or to use the REPEATED statement in PROC GLM.

PROC TRANSFORM may come to mind as a solution to these transforming problems, but using arrays in a data step can be more flexible and allow you to have full control over the transformation process.

Example 1: Creating a New Data Set with Several Observations per Subject from a Data Set with One Observation per Subject

Suppose you may have a data set called DIAGNOSE, with the variables ID, DX1, DX2, and DX3. The DX variables represent three diagnosis codes. The observations in data set DIAGNOSE are:

```
DATA SET DIAGNOSE
ID DX1 DX2 DX3
01 3 4  
02 1 2 3
03 4 5  
04 7
```

As you can see, some subjects have only one diagnosis code, some two, and some all three. Suppose you want to count how many subjects have diagnosis 1, how many have diagnosis 2, and so on. You don't care if the diagnosis code is listed as DX1, DX2, or DX3. In the example here, you would have a frequency of one for diagnosis codes 1, 2, 5, and 7 and a frequency of two for diagnosis codes 3 and 4.

One way to accomplish this task is to transform the data set DIAGNOSE which has one observation per subject and three diagnosis variables, to a data set that has a single diagnosis variable and as many observations per subject as there are diagnoses for that subject. This new data set (call it NEW_DX) would look like the one shown next:

```
TRANSFORMED DATA SET (NEW_DX)
ID DX
01 3
01 4
02 1
02 2
02 3
03 4
03 5
04 7
```

It is now a simple job to count diagnosis codes using PROC FRE. on the single variable DX. Let us first write a SAS DATA step that accomplishes this task and does not use arrays. Here is the code:

```
*----------------------------------*
EXAMPLE 1A: CREATING MULTIPLE OBSERVATIONS FROM A SINGLE OBSERVATION WITHOUT USING AN ARRAY
*----------------------------------*
DATA NEW_DX;
SET DIAGNOSE;
DX = DX1;
IF DX = NE . THEN OUTPUT;
DX = DX2;
IF DX = NE . THEN OUTPUT;
DX = DX3;
IF DX = NE . THEN OUTPUT;
KEEP ID DX;
RUN;
```

As you read in each observation from data set DIAGNOSE, you create from one to three observations in the new data set NEW_DX. The
KEEP statement is needed since you only want the variables ID and DX in the new data set.

Notice the repetitive nature of the program and your array light bulb should turn on. Here is the program rewritten using arrays:

```sas
DATA NEW DX;
  SET DIAGNOSE;
  ARRAY DXARRAY[3] DX1 - DX3;
  DO I = 1 TO 3;
    DX = DXARRAY[I];
    IF DX NE . THEN OUTPUT;
  END;
  KEEP ID DX;
RUN;
```

In this program, you first create an array called DXARRAY which contains the three numeric variables DX1, DX2, and DX3. The two lines of code inside the DO loop are similar to the repeated lines in the non-array example with the variable names DX1, DX2, and DX3 replaced by the array elements. (For a more detailed discussion of array processing, please refer to the paper called Arrays, A Powerful Time Saver, in the section on Beginning Tutorials.)

To count the number of subjects with each diagnosis code, you can now use PROC FRE.

```sas
PROC FREQ DATA=NEW DX;
  TABLES DX / NOCUM;
RUN;
```

In this example, you only saved one line of SAS code. However, if there were more variables, DX1 to DX50 for example, the savings would be substantial.

Example 2: Another Example of Creating Multiple Observations from a Single Observation

Here is an example that is similar to Example 1. You start with a data set that contains an ID variable and three variables S1, S2, and S3 which represent a score at times 1, 2, and 3 respectively. The original data set called ONEPER, looks as follows:

```
DATA SET ONEPER
   ID S1 S2 S3
   01 3 4 5
   02 7 8 9
   03 6 5 4
```

You want to create a new data set called MANYPER, which looks like this:

```
DATA SET MANYPER
   ID TIME SCORE
   01 1 3
   01 2 4
   01 3 5
   02 1 7
   02 2 8
   02 3 9
   03 1 6
   03 2 5
   03 3 4
```

The program to transform data set ONEPER to data set MANYPER is similar to the program in Example 1 except that you need to create the TIME variable in the transformed data set. This is easily accomplished by naming the DO loop counter TIME as follows:

```sas
DATA MANYPER;
  SET ONEPER;
  ARRAY S[3];
  DO TIME = 1 TO 3;
    SCORE = S[TIME];
    OUTPUT;
  END;
  KEEP ID TIME SCORE;
RUN;
```

Notice that the ARRAY statement does not have a variable list. This was done to demonstrate another way of writing an array statement. When this list is omitted, the variable names default to the array name followed by the numbers from the lower bound to the upper bound. In this case, the statement

```sas
ARRAY S[3];
```

is equivalent to

```sas
ARRAY S[3] S1-S3;
```
Still going in the direction of creating multiple observations from a single observation, let us extend this program to include an additional dimension.

**Example 3: Going from One Observation per Subject to Many Observations per Subject Using Multidimensional Arrays**

Suppose you have a SAS data set (call it WT_ONE) that contains an ID and 6 weights on each subject in an experiment. The first 3 values represent weights at times 1, 2, and 3 under condition 1; the next 3 values represent weights at times 1, 2, and 3 under condition 2. To clarify this, suppose that data set WT_ONE contained two observations:

```
DATA SET WT_ONE
ID WT1 WT2 WT3 WT4 WT5 WT6
01 155 158 162 149 148 147
02 110 112 114 107 108 109
```

and you want a new data set called WT_MANY to look like this:

```
DATA SET WT_MANY
ID COND TIME WEIGHT
01 1 1 155
01 1 2 158
01 1 3 162
01 2 1 149
01 2 2 148
01 2 3 147
02 1 1 110
02 1 2 112
02 1 3 114
02 2 1 107
02 2 2 108
02 2 3 109
```

A convenient way to make this conversion would be to create a two-dimensional array with the first dimension representing condition and the second representing time. So, instead of having a one-dimensional array like this:

```
ARRAY WEIGHT[6] WT1-WT6;
```

you could create a two-dimensional array like this:

```
ARRAY WEIGHT[2,3] WT1-WT6;
```

The comma between the 2 and 3 separates the dimensions of the array. This is a 2 by 3 array. Array element WEIGHT[2,3], for example, would represent a subject's weight under condition 2 at time 3.

Let us use this array structure to create the new data set which contains 6 observations for each ID. Each observation is to contain the ID and one of the 6 weights, along with two new variables, COND and TIME which represent the condition and the time at which the weight was recorded. Here is the restructuring program:

```
*----------------------------------*
| EXAMPLE 3: USING A MULTIDIMEN- |
| SIGNAL ARRAY TO RESTRUCTURE A   |
| DATA SET                       |
*----------------------------------*
DATA WT_MANY;
SET WT_ONE;
ARRAY WTS [2,3] WT1-WT6;
DO COND = 1 TO 2;
  DO TIME = 1 TO 3;
    WEIGHT = WTS[COND,TIME];
    OUTPUT;
  END;
END;
DROP WT1-WT6;
RUN;
```

To cover all combinations of condition and time, you use "nested" DO loops, that is, a DO loop within a DO loop. Here's how it works: COND is first set to 1 by the outer loop. Next, TIME is set to 1, 2, and 3 while COND remains at 1. Each time a COND and TIME combination is selected, a WEIGHT is set equal to the appropriate array element and the observation is written out to the new data set.

**Example 4: Creating a Data Set with One Observation per Subject from a Data Set with Multiple Observations per Subject**

It's now time to reverse the transformation process. We will do the reverse of Example 2 to demonstrate how to create a single observation from multiple observations. This time, we will start with data set MANYPER and create data set ONEPER. First the program, then the explanation:
EXAMPLE 4A: CREATING A DATA SET WITH ONE OBSERVATION PER SUBJECT FROM A DATA SET WITH MULTIPLE OBSERVATIONS PER SUBJECT. (CAUTION, THIS PROGRAM WILL NOT WORK IF THERE ARE ANY MISSING TIME VALUES.)

PROC SORT DATA=MANYPER;
  BY ID TIME;
RUN;

DATA ONEPER;
  RETAIN 51-53;
  SET MANYPER;
  BY ID;
  S[TIME] = SCORE;
  IF LAST.ID THEN OUTPUT;
  KEEP ID 51-53;
RUN;

First you sort data set MANYPER to be sure that the observations are in ID and TIME order. In this example, data set MANYPER is already in the correct order but the SORT procedure makes the program more general. Next, you need to create an array containing the variables you want in the ONEPER data set, namely, S1, S2, and S3. You can “play computer” to see how this program works. The first observation in data set MANYPER is:

<table>
<thead>
<tr>
<th>ID</th>
<th>TIME</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Therefore S[TIME] will be S[1] which represents the variable S1 and is set to the value of SCORE which is 4. Since LAST.ID is false, the OUTPUT statement does not execute. However, the value of S1 is retained. In the next observation time is 2 and SCORE is 5, so the variable S2 is assigned a value of 5. Finally, the third and last observation is read for ID 01. S3 is set to the value of SCORE which is 6 and since LAST.ID is true, the first observation in data set ONEPER is written. Everything seems fine. Almost.

What if data set MANYPER did not have an observation at all three values of time for each ID? Use the data set MANYPER2 shown next to see what would happen:

DATA SET MANYPER2

<table>
<thead>
<tr>
<th>ID</th>
<th>TIME</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>01</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>01</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>02</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>02</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>03</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>03</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>03</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Notice that ID number 02 does not have an observation with TIME = 2. What will happen if you run program Example 4A? Since you retained the values of S1, S2, and S3 and never replaced the value of S2 for ID number 02, ID number 2 will be given the value of S2 from the previous subject! Not what you want. You must always be careful when you retain variables. To be sure that this will not happen, you need to set the values of S1, S2, and S3 to missing each time you encounter a new subject. This is easily accomplished by checking the value of FIRST.ID. The corrected program is shown next:

PROC SORT DATA=MANYPER2;
  BY ID TIME;
RUN;

DATA ONEPER;
  ARRAY S[3] 51-S3;
  RETAIN 51-S3;
  SET MANYPER2;
  BY ID;
  IF FIRST.ID THEN DO;
    S[I] = .;
  END;
  S[TIME] = SCORE;
  IF LAST.ID THEN OUTPUT;
  KEEP ID 51-S3;
RUN;

This program will now work correctly whether or not there are missing TIME values.

Example 5: Creating a Data Set with One Observation per Subject from a Data Set with Multiple Observations per Subject Using a Multidimensional Array
This example will be the reverse of Example 3. That is, you want to start from data set WT_MANY and wind up with data set WT_ONE. The solution to this problem is similar to Example 4 except we will use a multidimensional array. Instead of writing the program in two steps as we did in Examples 4A and 4B, we will present the general solution that will work whether or not there are any missing observations in the data set. Here is the program:

```
*----------------------------------* EXAMPLE 5: CREATING A DATA SET
| WITH ONE OBSERVATION PER        |
| SUBJECT FROM A DATA SET WITH    |
| MULTIPLE OBSERVATIONS PER       |
| SUBJECT USING A MULTIDIMEN-     |
| SIONAL ARRAY                   |
*----------------------------------*

PROC SORT DATA=WT_MANY;
  BY ID COND TIME;
RUN;

DATA WT_ONE;
ARRAY WT[2,3] WT1-WT6;
RETAIN WT1-WT6;
SET WT_MANY;
BY ID;
  IF FIRST.ID THEN
    DO I = 1 TO 2;
      DO J = 1 TO 3;
        WT[I,J] = .;
      END;
    END;
  WT[COND,TIME] = WEIGHT;
  IF LAST.ID THEN OUTPUT;
KEEP 1D WT1-WT6;
RUN;
PROC PRINT DATA=WT_ONE;
  TITLE 'WT_ONE AGAIN';
RUN;
```

Conclusion
You have seen how to restructure SAS data sets, going from one to many or from many to one observations per subject using arrays. You may want to keep these examples handy for the next time you have a restructuring job to be done.

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries, © indicated USA registration.