UNDERSTANDING THE TRANSPOSE PROCEDURE

Susan P. Repole
ARC Professional Services Group

There are many times when a programmer will want to manipulate a SAS data set by creating observations from variables and/or variables from observations. This can be accomplished with Data Step processing but it can be coded faster and easier using PROC TRANSPOSE. This procedure can be used in very simple or very complex manipulations of data sets. It has also proven useful for increasing I/O efficiency in table look-up applications.

Why Use PROC TRANSPOSE?

Although Data Step processing can yield the same results, PROC TRANSPOSE has many advantages. Using the PROC TRANSPOSE will minimize coding and debugging time. The code will also be easier to maintain. To illustrate we have a data set made up of five years of enrollment figures for a fictional elementary school. (See Figure 1.)

The same results can be achieved with a very simple PROC TRANSPOSE as illustrated in Figure 4.

How Does PROC TRANSPOSE Work?

PROC TRANSPOSE shifts variables to observations and observations to variables. The procedure includes a number of options.
and statements which give you the power to customize this process in many ways. The example in Figure 4 uses the BY statement as well as the DATA=, OUT=, and NAME= options. The DATA= option names the input data set to the procedure. SAS will use the most recently created data set as input if one is not specified with the DATA= option. However it is good practice to specify the input data set explicitly to make the code more self-documenting.

The PROC TRANSPOSE procedure will not transpose a data set in place. It always creates a new data set as its output. If you do not select a name for this output data set with the OUT= option, SAS will select the name for you using the DATAn naming convention.

The NAME= option allows you to choose a variable name for a field which will contain the name of the transposed variables. If omitted SAS will name the variable _NAME_.

Using the data set from Figure 1, we can use PROC TRANSPOSE in its most basic form to see what happens without any related statements, and no options other than DATA= and OUT= which we have just discussed. Figure 5 contains the code for a basic transpose procedure as well as the resulting output.

```sas
PROC TRANSPOSE DATA=ENROLL
   OUT=ENRLTRAN;
RUN;
PROC PRINT DATA=ENRLTRAN;
   TITLE 'OUTPUT OF BASIC TRANSPOSE';
   RUN;
```

<table>
<thead>
<tr>
<th>OBS</th>
<th>NAME</th>
<th>COL1</th>
<th>COL2</th>
<th>COL3</th>
<th>COL4</th>
<th>COL5</th>
<th>COL6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>YR_88</td>
<td>48</td>
<td>52</td>
<td>48</td>
<td>47</td>
<td>38</td>
<td>35</td>
</tr>
<tr>
<td>2</td>
<td>YR_89</td>
<td>43</td>
<td>52</td>
<td>47</td>
<td>49</td>
<td>42</td>
<td>38</td>
</tr>
<tr>
<td>3</td>
<td>YR_90</td>
<td>44</td>
<td>48</td>
<td>50</td>
<td>46</td>
<td>47</td>
<td>44</td>
</tr>
<tr>
<td>4</td>
<td>YR_91</td>
<td>45</td>
<td>47</td>
<td>46</td>
<td>48</td>
<td>44</td>
<td>48</td>
</tr>
<tr>
<td>5</td>
<td>YR_92</td>
<td>48</td>
<td>49</td>
<td>49</td>
<td>48</td>
<td>48</td>
<td>48</td>
</tr>
</tbody>
</table>

**Figure 5**

Notice in the output in Figure 5, that SAS used the default variable name _NAME_ because the NAME= option was not used. SAS has also used the default for the transposed observations. The values of the original variable GRADE (1ST, 2ND, etc.) are represented as COL1 through COL6. The PREFIX= option can be used to give the transposed variable a more meaningful name. For instance, had the procedure included the option PREFIX=GRADE, the resulting data set would have had the variable names GRADE1 - GRADE6 which would more appropriately identify the variables.

The ID statement can be used to make more meaningful variable names. This statement will cause the values of the original GRADE variable to serve as variable names in the transposed data set. The code in Figure 6 illustrates this use of the ID statement.

```sas
PROC TRANSPOSE DATA=ENROLL
   ID GRADE;
   OUT=ENRLTRAN;
RUN;
PROC PRINT DATA=ENRLTRAN;
   TITLE 'TRANSPOSE USING ID';
   RUN;
```

<table>
<thead>
<tr>
<th>OBS</th>
<th>NAME</th>
<th>1ST</th>
<th>2ND</th>
<th>3RD</th>
<th>4TH</th>
<th>5TH</th>
<th>6TH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>YR_88</td>
<td>48</td>
<td>52</td>
<td>48</td>
<td>47</td>
<td>38</td>
<td>35</td>
</tr>
<tr>
<td>2</td>
<td>YR_89</td>
<td>43</td>
<td>52</td>
<td>47</td>
<td>49</td>
<td>42</td>
<td>38</td>
</tr>
<tr>
<td>3</td>
<td>YR_90</td>
<td>44</td>
<td>48</td>
<td>50</td>
<td>46</td>
<td>47</td>
<td>44</td>
</tr>
<tr>
<td>4</td>
<td>YR_91</td>
<td>45</td>
<td>47</td>
<td>46</td>
<td>48</td>
<td>44</td>
<td>48</td>
</tr>
<tr>
<td>5</td>
<td>YR_92</td>
<td>48</td>
<td>49</td>
<td>49</td>
<td>48</td>
<td>48</td>
<td>48</td>
</tr>
</tbody>
</table>

**Figure 6**

Because the values of the variable GRADE in the original data set began with a number, SAS inserted an underscore in front of the value so that it would be a legal SAS variable name. If the values had started with a character, they would have been left in their original form.

The IDLABEL statement can be used in conjunction with the ID statement to give the new variables a label. The label can be used just as any SAS variable label on an untransposed data set.

The IDLABEL statement creates labels for variables created by the transpose. The new
variables did not have labels before the transpose since they were observations in the original data set. What about labels which may have existed for variables which have been transposed to observations? If any variable which is to be transposed has a label, SAS will make the label a value in a variable called _LABEL_. If you would have a name you would rather use as the variable name for these label values, you can use the LABEL= option to let SAS know what name to assign.

An important statement associated with PROC TRANSPOSE is the VAR statement. If omitted, the transpose procedure will transpose all numeric variables in the data set. The exception to the automatic transposition is any variables which have been used in other statements in the transpose procedure. The COPY statement for example, allows you to copy a variable to the transposed data set without changing it. Any variable(s) used in the COPY statement would not be automatically transposed if the VAR statement were omitted.

In our example data set, all of the variables starting with ‘VR’ are numeric, therefore all were transposed because there was no VAR statement used. If you only want one or several of your numeric values transposed, you use the VAR statement to tell SAS which ones you want to use. Figure 7 contains a PROC TRANSPOSE on our example data set, using the VAR statement. The output is included in the figure.

```
PROC TRANSPOSE DATA=ENROLL OUT=ENRLTRAN;
   VAR YR_88 YR_90;
RUN;
   
PROC PRINT DATA=ENRLTRAN;
   TITLE 'TRANSPOSE WITH VAR STMT';
RUN;
```

Figure 7

The example in Figure 7 used the VAR statement to tell SAS that the only variables to be transposed were YR_88 and YR_90. The other variables were left out of the output data set, leaving it with only two observations as opposed to 5 observations when all numeric variables were transposed in Figure 6.

PROC TRANSPOSE can also transpose character variables. In order to transpose character variables, the VAR statement must be used. The variables will be transposed just as the numeric variables are. It is important to remember that all transposed variables are the same type (character or numeric) and length. Therefore if any character variables are included on the VAR statement, all of the values in your output data set will be character. If you are transposing your data in preparation for mathematical calculations, you would not want to have them all changed to character values before your computations. Figure 8 shows an example of transposition of a character and a numeric variable and the resulting output. The output looks as if the values of the numeric variables have been preserved as numeric. However, as the results of a PROC CONTENTS on the output data set reveal, that all the values in the data set are now character.

**PROC TRANSPOSE as an Efficiency Tool**

PROC TRANSPOSE can be used to reduce I/O (number of records read and written internally), when used as a tool for table look-up applications. The following example (taken from the NESUG '92 proceedings, see references below), illustrates the way in which PROC TRANSPOSE can be helpful in large table look-up problems.

In this example, we are working with two, large data sets. The first data set consists of two million observations with three variables each: a unique key, a start date, and an end date. Our task was to calculate the average temperature during the time between the start date and the end date (usually a thirty-day period) for each observation in the data set. The temperatures for each of the dates in the master data set are available on a secondary
PROC TRANSPOSE DATA=ENROLL
   OUT=ENRLTRAN;
   VAR GRADE YR_88;
RUN;
PROC PRINT DATA=ENRLTRAN;
   TITLE 'TRANSPOSE CHAR VARIABLE';
RUN;
PROC CONTENTS DATA=ENRLTRAN;
RUN;

TRANSPOSE CHAR VARIABLE
OBS _NAME_ COL1 COL2 ...
1 GRADE 1ST 48 52
2 YR_88 2ND 52 37

Figure 8

data set which consists of two variables: the date, and the temperature for that date. There are no missing values in the secondary data set. Therefore, for each of the two million observations in the primary data set, an average of thirty look-ups must be performed in order to complete the calculation.

Without PROC TRANSPOSE the best way we could find to minimize the number of records read and written internally was to use a data step employing the SET with POINT= option. The code in Figure 9 illustrates this solution.

It was discovered that PROC TRANSPOSE could help us to reduce the number of records which would have to be read and written to solve this application problem. We used PROC TRANSPOSE to change the secondary data set from many records, each with one temperature value, into a data set consisting of one record with all of the required temperatures as variable values. This single record could then be read only once into the master data set. The values are loaded into an array and retained. This allows the look-up to be easily performed as illustrated:

PROC TRANSPOSE DATA=LOOKUP
   OUT=TRANSPOS;
   ID DATE;
   VAR TEMP;
   FORMAT DATE J1!LI)ATE5.;
RUN;
DATA FINAL (KEEP=KEY MEANTEMP);
   SET MASTER;
   TOTAL = 0;
   DO OBS = (STARTDAY - '01JAN90'D+1)
               TO (ENDDAY - '01JAN90'D+1);
       SET LOOKUP POINT=OBS;
       TOTAL + TEMP;
   END;
   MEANTEMP = TOTAL / (ENDDAY - STARTDAY+1);
RUN;

Figure 9

This solution reduced I/O from 60,000,000 records to 2,000,001 - a considerable savings. The ability of PROC TRANSPOSE to change a data set with a large number of observations into a data set with only one observation (or change a many-variable data set to a few-variable data set), without a loss of data can be a helpful tool for efficient processing.

Conclusion

PROC TRANSPOSE is a very handy tool in the SAS System. It can be used to solve simple data manipulation problems quickly and easily. Yet it is powerful enough to solve complex problems in only a few lines of code. The more you use this procedure the more uses you may
find for it. Try it out using different combinations of options and statements to organize your data in a variety of ways!

REFERENCES


SAS is a registered trademark of SAS Institute Inc., Cary, NC, USA.