

SYMPLIFY your Data Set Transposition with SYMPUT, and Make it Data-Driven Too!

Christianna S. Williams, University of North Carolina at Chapel Hill

ABSTRACT

I frequently need to convert a SAS® data set with multiple observations per subject into one with a single observation per subject with multiple variables to hold information that had been stored as single variables – in other words “de-normalize” the data. This is often the case with longitudinal data, where information is collected on individuals repeatedly over time. To successfully complete this transposition, one needs to know how many records there are for each subject in the input data to determine how many variables are needed in the output data. This can be complicated when there are variable numbers of records among subjects and/or this number of records can change over time. Basically, to prevent loss of information in the transposed file, one needs to know the maximum number of observations that any one subject has in the input data set. In this paper, I will present a small tidbit of code that determines this maximum and then uses the CALL SYMPUT routine to direct the construction of the output data set. The beauty of this method is that it will work even if that maximum number changes from one run of the data to the next.

INTRODUCTION

Imagine that you have a data set that has multiple and variable numbers of observation per subject – in the health field these might be hospital admissions, clinic visits, diagnoses, or medications. In a business application, they might be monthly banking transactions per customer or products per vendor. Good database practice would generally dictate that these data are maintained in a normalized fashion – i.e. with minimal redundancy of key information. However, sometimes – for a particular report or to speed certain types of processing, one needs to put the data in a “de-normalized” form. Let’s consider some progressively more complex examples.

EXAMPLE 1 – Balanced data, one variable to transpose

In this example, all subjects have the same number of observations in the input file (i.e. the data are balanced), and there is only one variable that needs to be transposed:

DATA TESTV1		
ID	VISIT	PULSE
01	1	72
01	2	88
02	1	64
02	2	63
03	1	78
03	2	70

The code* to convert this data set to one observation per ID is simple, either with PROC TRANSPOSE or a DATA step. Note that for the DATA step method, the VISIT variable is used as an array index, and I need to know how many observations (visits) each ID has; hence, for this very simple case TRANSPOSE is probably preferred.

PROC TRANSPOSE METHOD:

```
PROC TRANSPOSE DATA=testv1 OUT=testh1 (DROP=_name_) PREFIX=hr;
BY id ;
VAR pulse ;
RUN;
```

(*NOTE: Here and in code examples throughout the paper, SAS keywords are in all caps to readily distinguish them from user-defined names, which are in lower case.)

DATA STEP METHOD:

```

DATA testh1_b (KEEP = id hr1 hr2);
SET testv1 ;

BY id ;
ARRAY hr{2} hr1 hr2 ;
RETAIN hr1 hr2 ;
hr{visit} = pulse ;
IF LAST.id THEN OUTPUT ;

RUN;

```

With either method the result is the desired “horizontal” data set:

TESTH1 (and TESTH1_B)		
ID	HR1	HR2
01	72	88
02	64	63
03	78	70

EXAMPLE 2 – Unbalanced data, one variable to transpose

The same basic code will even work if the data are unbalanced (i.e. not all ID's have the same number of observations) – here we add a third visit only for ID 02.

DATA TESTV2		
ID	VISIT	PULSE
01	1	72
01	2	88
02	1	64
02	2	63
02	3	65
03	1	78
03	2	70

No changes at all are needed for the PROC TRANSPOSE method (except the input and output data set names):

```

PROC TRANSPOSE DATA=testv2 OUT=testh2 (DROP=_name_) PREFIX=hr;
BY id ;
VAR pulse ;
RUN;

```

The Data Step Method gets a bit uglier, and it is important to initialize (or “clean out”) the array for the first observation of each BY group:

```

DATA testh2_b ;
SET testv2 ;

BY id ;
ARRAY hr{3} hr1-hr3 ;
RETAIN hr1-hr3 .;

IF FIRST.id THEN
  DO i = 1 TO 3;
    hr{i} = . ;
  END;

hr{visit} = pulse ;
IF LAST.id THEN OUTPUT ;

KEEP id hr1-hr3 ;
RUN;

```

But, again, either method will do the trick. Here is the resulting data set:

DATA TESTH2 (& TESTH2_B)			
ID	HR1	HR2	HR3
01	72	88	.
02	64	63	65
03	78	70	.

The reason I bother showing the Data Step method is that it is more expandable when multiple variables need to be transposed, (and is where the tip on which this paper is based comes in!)

EXAMPLE 3 – Unbalanced Data, Multiple Variables to transpose

In this case, we not only have pulse measured at each visit but systolic and diastolic blood pressure as well. Here's the starting point:

DATA TESTV3				
ID	VISIT	PULSE	SYSTOLIC	DIASTOLIC
01	1	72	80	184
01	2	88	84	177
02	1	64	79	152
02	2	63	73	144
02	3	65	71	160
03	1	78	72	191
03	2	70	79	183

In this case a single TRANSPOSE won't give the desired result, which is still a data set with a single observation for each ID, and variables for each vital sign for each visit. But, I can get it with the DATA STEP, which is almost identical to the one for Example 2:

```
DATA testh3 ;
  SET testv3 ;

  BY id ;
  ARRAY hr{3} hr1-hr3 ;
  ARRAY sbp{3} sbp1-sbp3 ;
  ARRAY dbp{3} dbp1-dbp3 ;
  RETAIN hr1-hr3 sbp1-sbp3 dbp1-dbp3;

  IF FIRST.id THEN
    DO i = 1 TO 3;
      hr{i} = . ;
      sbp{i} = . ;
      dbp{i} = . ;
    END;

  hr{visit} = pulse ;
  sbp{visit} = systolic;
  dbp{visit} = diastolic;

  IF LAST.id THEN OUTPUT ;

  KEEP id hr1-hr3 sbp1-sbp3 dbp1-dbp3;
  RUN;
```

DATA TESTH3									
ID	HR1	HR2	HR3	SBP1	SBP2	SBP3	DBP1	DBP2	DBP3
01	72	88	.	80	84	.	184	177	.
02	64	63	65	79	73	71	152	144	160
03	78	70	.	72	79	.	191	183	.

EXAMPLE 4 – Maximum number of visits per ID is unknown

Let's say we have the data set very similar to the one used in Example 3, but we don't know what the maximum number of visits is in the data – or it could change as more data are collected, but you need to do the same processing. We'll also throw in the complication that the VISIT count variable doesn't exist – instead we have a date – another variable we want to transpose. Finally, here's where SYMPUT comes in.

DATA TESTV4				
ID	VISITDT	PULSE	SYSTOLIC	DIASTOLIC
01	07FEB2004	72	80	184
01	25APR2004	88	84	177
02	09NOV2003	64	65	144
02	31DEC2003	63	71	160
02	17MAR2004	65	79	172
03	06JUN2003	78	72	191
03	14JAN2004	70	79	183

The first step is to determine the maximum number of visits per ID, and store that information in a macro variable:

```
DATA testV4b ;
    SET testv4 END=lastobs;

    BY id ;
    IF FIRST.id THEN visit = 0;
    RETAIN visit maxvis ;

    visit + 1;

    IF LAST.id THEN
        maxvis = MAX(visit,maxvis);

    IF lastobs THEN
        CALL SYMPUT('maxvis',LEFT(maxvis)) ;

    DROP maxvis ;
    RUN;

    %PUT max # of visits is: &maxvis ;
```

Thanks to the %PUT statement, in the log I get the following:

```
max # of visits is: 3
```

The next step is a simple modification of the DATA step used in EXAMPLE 3, making use of the visit counter DATA step variable I've added and the macro variable indicating the maximum number of visits for any ID, which tells me how big the arrays need to be:

```
DATA testh4 ;
    SET testv4b ;

    BY id ;
    ARRAY vis{&maxvis} visdt1-visdt&maxvis ;
    ARRAY hr{&maxvis} hr1-hr&maxvis ;
    ARRAY sbp{&maxvis} sbp1-sbp&maxvis ;
    ARRAY dbp{&maxvis} dbp1-dbp&maxvis ;
    RETAIN hr1-hr&maxvis sbp1-sbp&maxvis dbp1-dbp&maxvis visdt1-visdt&maxvis;

    IF FIRST.id THEN
        DO i = 1 TO &maxvis;
            vis{i} = . ;
            hr{i} = . ;
            sbp{i} = . ;
            dbp{i} = . ;
        END;
```

```

vis{visit} = visitdt ;
hr{visit} = pulse ;
sbp{visit} = systolic;
dbp{visit} = diastolic;

IF LAST.id THEN OUTPUT ;

KEEP id hr1-hr&maxvis sbp1-sbp&maxvis dbp1-dbp&maxvis visdt1-visdt&maxvis;
FORMAT visdt1-visdt&maxvis date9. ;
RUN;

```

And the result is the desired data set, shown below.

DATA TESTH4												
ID	VISDT1	VISDT2	VISDT3	HR1	HR2	HR3	SBP1	SBP2	SBP3	DBP1	DBP2	DBP3
01	07FEB2004	25APR2004	.	72	88	.	80	84	.	184	177	.
02	09NOV2003	31DEC2003	17MAR2004	64	63	65	65	71	79	144	160	172
03	06JUN2003	14JAN2004	.	78	70	.	72	79	.	191	183	.

CONCLUSIONS

I have found this simple use of SYMPUT to be very handy in data manipulations I sometimes need to make for reporting or further processing of longitudinal data. As long as the input data has the same structure with respect to the variables that need to be transposed, it allows great flexibility in terms of the imbalance of the data. If you have another use for this technique I'd love to hear about it.

ACKNOWLEDGMENTS

SAS is a Registered Trademark of the SAS Institute, Inc. of Cary, North Carolina. ® indicates US registration.

¹Ralph Leighton very generously shared with me a method to use a PROC TRANSPOSE step for each variable that needs to be transposed and then MERGE (BY id) the resulting files back to a single file. The code involved in this strategy can also be greatly reduced by folding it into a macro. So, if you are facile with PROC TRANSPOSE, consider that route!

CONTACT INFORMATION

I welcome comments, suggestions and questions at:

Christianna S. Williams, PhD
 Cecil G. Sheps Center for Health Services Research
 University of North Carolina at Chapel Hill
 725 Martin Luther King Boulevard
 Campus Box # 7590
 Chapel Hill, North Carolina 27599
 Email: Christianna_Williams@unc.edu