PharmaSUG 2011 - Paper TT02

Create a Format from a SAS[®] Data Set

Ruth Marisol Rivera, i3 Statprobe, Mexico City, Mexico

ABSTRACT

Many times we have to apply formats and it could be hard to create them specially if there are a lot of values to consider. But if you have all the values on a SAS data set you can pull them from there instead of creating the format manually (typing the data). This paper shows how to do that just by saying which variable you are going to use for the start of the range values (or a single value) and which value you want to receive back (a variable value or a pre-established value).

INTRODUCTION

The formats can be used for doing a lot of things; one useful application is having one format with a couple of values that can be matched with a data set (so you can avoid a merge statement).

For example suppose that you have to show a listing which has 3 SAS data sets as source

Let's see what the listing specification may look like:

<u>Listing of reactions during next five days after taken Medicine "A".</u>

Patient **Reactions.patient**	**Type of reaction Reactions.reac_type**	**Day number Reactions.day**	**Had it? Reactions.response**	**Maximum duration of reaction maxdur.max_dur**	**Did the patient decide to discontinue? discont.discontinued**
1	Pain	1	Yes	2	Yes
		2	Yes		
		3	No		
		4	Yes		
		5	No		
	Fever	1	No	1	
		2	No		
		3	No		
		4	Yes		
		5	No		
2	Pain	1	No	1	No
		2	No		
		3	Yes		
		4	No		
		5	No		
	Fever	1	Yes	3	
		2	Yes		
		3	Yes		
		4	No		
		5	No		

Note: The response for **Did the patient decide to discontinue?** should only appear in the first row for each patient.

SAS data sets:

Data set REACTIONS:

Patient	Reac_type	response	Day
1	PAIN	YES	1
1	PAIN	YES	2
1	PAIN	NO	3
1	PAIN	YES	4
1	PAIN	NO	5
1	FEVER	NO	1
1	FEVER	NO	2
1	FEVER	NO	3
1	FEVER	YES	4
1	FEVER	NO	5
2	PAIN	NO	1
2	PAIN	NO	2
2	PAIN	YES	3
2	PAIN	NO	4
2	PAIN	NO	5
2	FEVER	NO	1
2	FEVER	NO	2
2	FEVER	NO	3
2	FEVER	NO	4
2	FEVER	NO	5

Data set DISCONT:

Patient	Discontinued
1	YES
1	YES
2	NO
2	NO

POSSIBLE WAYS TO DO IT

1. With the “traditional” way (doing a merge) we would have to do this:

   1 Do a PROC SORT to take care of duplicated records on REACTION and DISCONT datasets.
   2 Merge these two data sets.

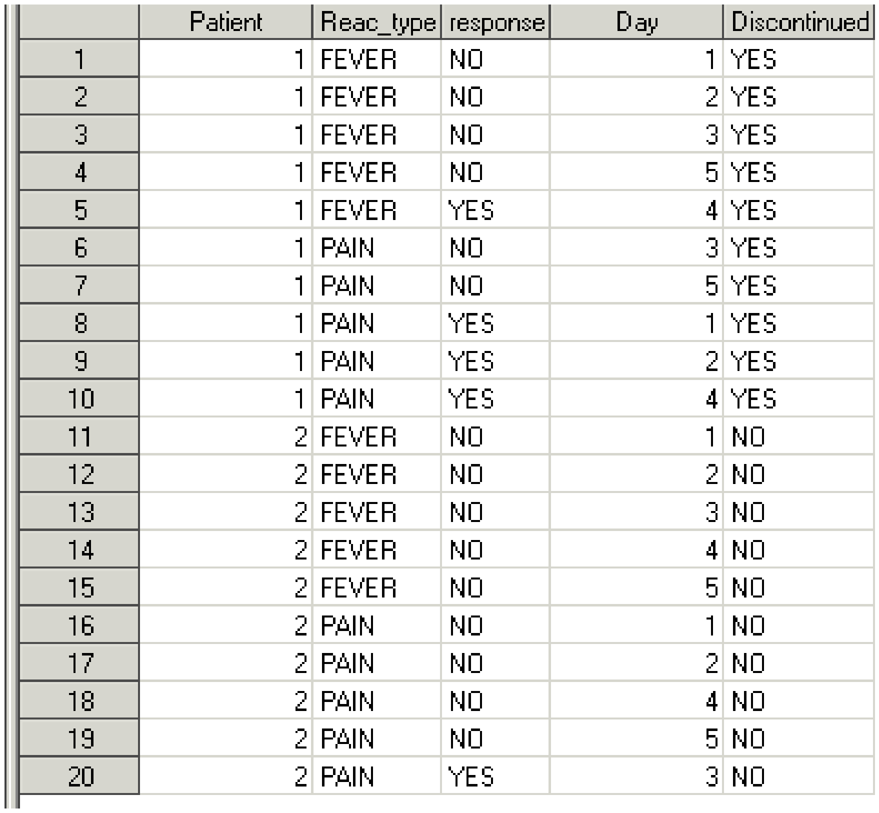
```
proc sort data=reactions ; by patient reac_type response day; run;
proc sort data=discont   ; by patient discontinued; run;

data all;
    merge reactions discont;
    by patient;
run;
```

Note that if we are not aware about duplicate records we can get this in the log:

NOTE: MERGE statement has more than one data set with repeats of BY values.

And we also get the new value on ALL records (instead of just one as we need for the listing).



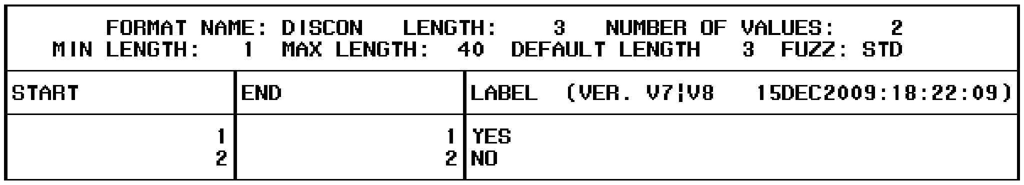
	Patient	Reac_type	response	Day	Discontinued
1	1	FEVER	NO	1	YES
2	1	FEVER	NO	2	YES
3	1	FEVER	NO	3	YES
4	1	FEVER	NO	5	YES
5	1	FEVER	YES	4	YES
6	1	PAIN	NO	3	YES
7	1	PAIN	NO	5	YES
8	1	PAIN	YES	1	YES
9	1	PAIN	YES	2	YES
10	1	PAIN	YES	4	YES
11	2	FEVER	NO	1	NO
12	2	FEVER	NO	2	NO
13	2	FEVER	NO	3	NO
14	2	FEVER	NO	4	NO
15	2	FEVER	NO	5	NO
16	2	PAIN	NO	1	NO
17	2	PAIN	NO	2	NO
18	2	PAIN	NO	4	NO
19	2	PAIN	NO	5	NO
20	2	PAIN	YES	3	NO

2. Using a format you only need to do this (please see the complete macro code at the end of the paper on appendix A):

   1 Create a format for DISCONT.
   2 Use a simple data set statement to put the formatted values on the corresponding row.

```
%doafmt  ( fmtname=discon,  dsn=discont,     start=patient,  label=discon, type=N,
perm=N,  end=,    other=,  print=N);
```

It will create a format like this:



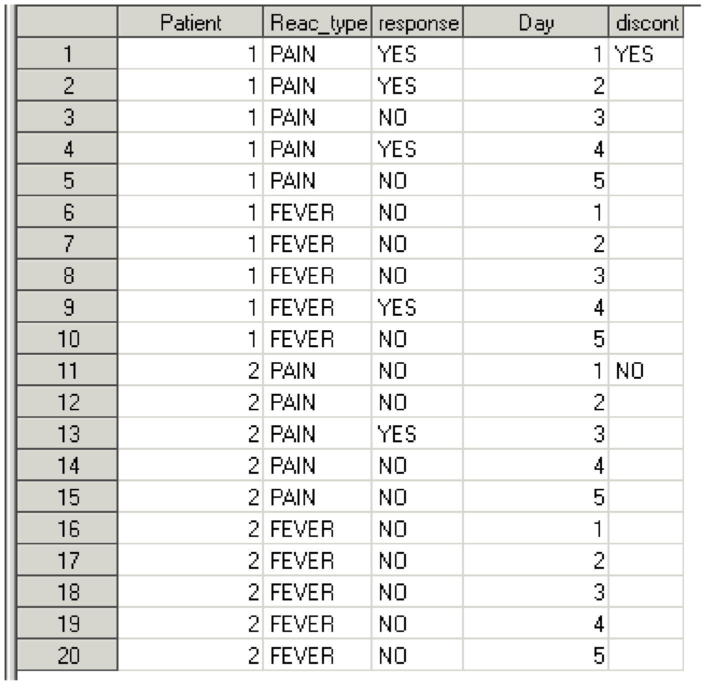
```
     FORMAT NAME: DISCON   LENGTH:    3   NUMBER OF VALUES:    2
   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH   3  FUZZ: STD
```

START	END	LABEL (VER. V7|V8 15DEC2009:18:22:09)
1	1	YES
2	2	NO

And then just create the new variable with the value we want (in the place we want):

```
data all;
    set reactions;
          if first.patient then discont=put(patient,discon.);
run;
```

And then we'll have our data set ready to be printed:



	Patient	Reac_type	response	Day	discont
1	1	PAIN	YES	1	YES
2	1	PAIN	YES	2	
3	1	PAIN	NO	3	
4	1	PAIN	YES	4	
5	1	PAIN	NO	5	
6	1	FEVER	NO	1	
7	1	FEVER	NO	2	
8	1	FEVER	NO	3	
9	1	FEVER	YES	4	
10	1	FEVER	NO	5	
11	2	PAIN	NO	1	NO
12	2	PAIN	NO	2	
13	2	PAIN	YES	3	
14	2	PAIN	NO	4	
15	2	PAIN	NO	5	
16	2	FEVER	NO	1	
17	2	FEVER	NO	2	
18	2	FEVER	NO	3	
19	2	FEVER	NO	4	
20	2	FEVER	NO	5	

CONCLUSION

This is the text for the paper’s conclusion.

When you are pulling a small amount of data, or even just one value, from multiple datasets, creating formats and using them in a simple DATA step statement can be easier than sorting them (taking care of duplicates), sometimes renaming the variables (when they have identical names on the different data sets) and merging them (which involves checking that all the BY variables are correct). And talking about time-consuming- it’s much more efficient to use a single DATA step with a PUT function than a data set MERGE , especially if we’re dealing with thousands of records.

ACKNOWLEDGMENTS

Thank you to Nancy Brucken for her feedback when building this paper. I really appreciate it!

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Ruth Marisol Rivera
Enterprise: i3 Statprobe
Address: Insurgentes Sur #416 4th floor, Col Del Valle
City, State ZIP: 03100, Mexico, DF
Work Phone: ´52(55) 5005 5525
E-mail: Marisol.rivera@i3statprobe.com

APPENDIX A

Code for the macro %doafmt.

```
/* ---------------------------------------------------------------------------
MACRO NAME:   %doafmt
DESCRIPTION:   This macro creates a format from a SAS data set or view.
USAGE:  %doafmt( NAME, DSN, START=, LABEL=, TYPE=,  PERM=, END=, OTHER=, PRINT=   );
PARMS:  - Required -
NAME -  Name of format.  ( "$" not needed see "TYPE" below ).
( i.e.:  MYFMT  equates to $MYFMT for character formats or MYFMT for numeric formats. )
DSN  -  Name of SAS data set or view used to create format.
Single level name assumes "work."
( i.e.:    "sasuser.myfmt"  pulls from libref "SASUSER" while  "myfmt"  pulls from "WORK."
)
START - Name of the variable used for the start of the range values or single value. (
Left-of-equal-sign value )
( i.e.:  START=MODEL where MODEL is a variable on DSN )
LABEL - Name of the variable used for the formatted-value (label)
or a quoted value.  ( right-of-equal-sign value )
( i.e.: LABEL=DESC where DESC is a variable on DSN or LABEL="YES"    see selection example
below )
             - Optional _
TYPE -  Type of format.  N = Numeric Format
             P = Picture Format
                         I = Numeric Informat
                   C = Character Format   ( Default )
                         J = Character Informat
When type is "C" or "J" the resulting format is prefixed with a dollar sign ( $ ).
PERM -  Flag to create a permanent or temporary format.
             N = temporary format created. ( Default )
 Y = permanent format created must have  libref "LIBRARY" allocated via libname statement.
END   - Name of the variable used for the end of the range values. (left-of-equal-sign
value )
OTHER - A quoted value for the OTHER option of formats.
PRINT - Flag to print formats as they are created.
              N = Do not print format. ( Default )
                          Y = Print the format.
 EXAMPLES:  %doafmt( myfmt, sasuser.myfmt, start=model, label=desc ); Generates a
character format called "$myfmt" from dsn "sasuser.myfmt" using the variable "model" for
range values and the variable "desc" for label values.
             %doafmt( myfmt, sasuser.myfmt, start=model, label="YES" );
Generates a character format called "$myfmt" from dsn "sasuser.myfmt" using the variable
"model" for range values and assigns "YES" as label values constants.

  ------------------------------------------------------------------------- */
```

```
%macro doafmt
   ( fmtname,  dsn,     start=,  label=,
     type=C,   perm=N,  end=,    other=,  print=N );

   %let type  = %upcase( &type  );
   %let perm  = %upcase( &perm  );
   %let print = %upcase( &print );
   %let other = %quote( &other );
   %if %substr( &fmtname, 1,1 ) eq %str($) %then %do;
       %let type = C;
       %let fmtname = %substr( &fmtname, 2 );
   %end;

   DATA WORK._FMT_;        /* create a dsn used on cntlin= of PROC FORMAT */
        LENGTH  START   LABEL $200;
        RETAIN FMTNAME "&fmtname"  TYPE "&type";
        KEEP FMTNAME  TYPE  START  LABEL
             %if &other ne %str( ) %then %str( HLO);
             %if &end   ne %str( ) %then %str( END);
             %str(;);
        SET &dsn      END=EOF;
        START = &start;
       %if &end   ne %str( ) %then %str( END=&end;);
        LABEL = &label;

        OUTPUT;
        %if &other ne %str( ) %then %do;
            IF EOF THEN DO;
               START = 'OTHER';
               LABEL = "&other";
               HLO='O';
               OUTPUT;
            END;
        %end;
    RUN;

    PROC SORT DATA=WORK._FMT_ NODUPKEY;    /* remove dups on the start value */
         by START;
    RUN;

    PROC FORMAT CNTLIN=WORK._FMT_
         %if &perm  eq %str(Y) %then %str( LIBRARY=LIBRARY );
         %if &print eq %str(Y) %then %str( FMTLIB );
         %str(;);
    RUN;
 %mend doafmt;
```