EDITING QUESTIONNAIRE-DATA SKIP PATTERNS WITH SAS® MACROS
OR
HOW SAS MACROS SAVED MY NECK

Susan Bolotin, Research Triangle Institute

Introduction: Questionnaire Data and Skip Patterns

At the Research Triangle Institute we frequently deal with questionnaire data. Often, trained interviewers fill out the forms and editors review their contents before they get to data entry. Nevertheless, it is common for inconsistencies to escape detection and turn up in the data files. They are particularly well represented in questionnaire skip patterns.

A skip pattern has two parts: 1) the routing question, whose answer determines whether a skip should be performed, and 2) the variable or variables to be skipped if the routing question's value indicates a skip. In a questionnaire, the skip instruction typically follows a coded response. For example, if Question 1 is a "Yes-No" question, "Skip to Question 4" might be written after the "No" response (Figure 1).

Skip-pattern Problems

All sorts of things can go wrong with a skip pattern. If the routing variable does not have one of the expected codes (for instance, if it has been left blank), there is no way of knowing whether the associated variables should be skipped. If there is not an error here, and the routing variable does its job, there is still the possibility of errors of omission or commission within the skip pattern. In other words, the responses skip when they shouldn't, or don't skip when they should.

Editing a skip pattern requires knowing what to do in these situations. In contract research, this may depend on what the contract says will be delivered. Or, someone who is going to use or analyze the data may want to see what he has gotten himself into. The programmer often has nothing to do with deciding what the editing is supposed to accomplish, so that question will not be addressed here.

The assumption that the programmer will be using explicit and complete specifications is not always warranted. In the example I will be using, however, I was fortunate enough to have specifications that indicated what action the edit should take in all possible situations.

The example I will be using is a fairly complicated one, but the method is appropriate for any edit that applies the same rules to all skip patterns.

I decided to use SAS macros to do my skip-pattern editing because I found myself in the following situation: I had a gargantuan editing task before me, consisting of similar sets of comprehensive and tedious edits on data files from seven different questionnaires. These edits included data conversions and some logical imputations as well as skip-pattern checks. I had a lot of editing to do, but I didn't have very much time.

SAS and Editing

As usual, I planned to use SAS for all the editing. SAS has two particularly convenient editing applications.

One is the creation of permanent intermediate files at successive stages of the editing process. Storing the data in intermediate files allows the user to break up an editing task into pieces, so that rerunning later stages does not necessitate rerunning the entire edit. And if these files are SAS data sets, the SET statement will read or write the entire file, eliminating the need for variable-by-variable input or output.

SAS also brings a distinct advantage to the testing phase, in which before-and-after PROC PRINTs show exactly what the program has really done to the data.

Why I Needed Macros

The magnitude of the editing task facing me became clearer after I had converted eight pages of specifications for the first questionnaire into roughly 4000 lines of SAS statements. At that point I reached the skip-pattern section and found the following:

a) two pages of flow chart describing in detail the generalized edit steps for any skip pattern (Figures 2 and 3), and
b) a flow chart of the skip-pattern logic, showing all the skip patterns for this questionnaire, taking up both halves of a legal-sized sheet divided lengthwise (like a much longer and fancier version of the chart in Figure 1).

The implication was horrendous: These were pictures worth thousands of words that I couldn't conceive of writing, even if I had the time, which I didn't.

This is where SAS macros came in, since they would uncomplainingly generate SAS statements I wouldn't even consider writing. It's important to remember that this is how macros work: Every time I invoke a macro, it generates statements and inserts them into my program, just the way I would if I weren't trying to be clever. In contrast, a subroutine or function executes the same set of statements each time it is called.

In this case, my notion of cleverness was a macro that would apply the instructions given in Figures 2 and 3 to any skip-pattern situation. It had to check the values of the variables within the skip pattern to see if they were missing (-1 or -8 in the files concerned) or if the values were legitimately missing (correctly skipped) it would convert them to -3. If it found errors in the skip pattern, it would convert some values to -6 and flag others with a '6', depending on the situation.

All of the flagging in these edits involved attaching the flag to the front of the variable. This entailed increasing the field width of the variables by one. That is, adding a flag of '6' to a value of '1' in a two-digit field would result in a value of '601' and require a three-digit field.

To further complicate the flagging, some variables could be flagged with a '2' at an earlier stage of the editing. If the error processing needed to flag a value that was originally '1' with a '6', it had to produce a value of '601', whether the value was unflagged ('1') or already flagged ('201').

There was also the matter of nested skip patterns (Figure 4). I needed to check inner skip patterns if the outer skip pattern was a "no skip" situation without errors.

To accomplish all this, I needed a macro that would know when to skip, incorporate the variable field widths for use in flagging, and keep track of situations requiring the examination of nested skip patterns. I wanted to be able to give it all the necessary information in the form of arguments, and let it do the rest.

How To Use a Macro for Skip-pattern Editing: An Example

The macro BSKIP (the "B" stands for "big") does all of this. It uses positional and keyword parameters, as shown in the %MACRO statement that begins its definition:

%MACRO BSKIP (RVAR,RWID,SARR,WARR,ERRA, CONT,SKVS,NEST=NST,SKP=SK);

The parameters are defined as follows:

RVAR - the routing variable
RWID - the field width of the (unflagged) routing variable
SARR - an array of the variables within the skip pattern
WARR - an array of the corresponding field widths
ERRA - a statement label
CONT - another label
SKVS - yet another label
NEST - a keyword parameter for the nesting flag, assigned the default value of "NST"
SKP - a keyword parameter for the skip indicator, assigned the default value of "SK"

I have already explained that the macro needs to know the routing variable and the variables within the skip pattern and their field widths. But what am I doing with three statement labels?

Because the three different error situations ultimately require parts of the same error-processing routine, and the two different error-free situations make an early exit, I used GO TO statements in my macro text. This is not a great example of structured programming, but structuring would have meant invoking another macro within this macro. That would have generated duplicate sets of SAS statements every time I invoked BSKIP, which was generating quite a few statements already.

By using GO TO statements in the macro text and supplying statement labels as parameters, I have not significantly increased the demand on the user--all he needs to do is avoid repeating labels within a program and he'll stay out of trouble. Program efficiency is not always a good argument, but macros are strong and silent, allowing the programmer to be powerfully inefficient.
The last two parameters are keyword parameters with default values assigned in the %MACRO statement. Using the default values means not having to include these parameters when invoking the macro. The default value of the nesting flag is the variable NST. If the edit involves checking nested skip patterns, and the default is not overridden, the macro will set NST=1.

Similarly, the skip indicator defaults to the variable SK. If the edit program sets SK=1 to indicate a skip and SK=2 for no skip (actually, I have delegated this job to macros too, and will provide more details later), the macro will work fine with the default value.

The following somewhat pared-down example presents the macro BSKIP in a program context. I have assumed that the program has already checked the routing variable TQ30A (field width 2) to see if it indicates a skip and has set the variable SK accordingly:

```
ARRAY S30V TQ30B1 TQ30B2;
ARRAY S30W W1 W2;
W1=2;
W2=2;
%BSKIP(TQ30A,2,S30V,S30W,ERF15,CONT15,SKIN15)
```

In this example, I have supplied BSKIP with the routing variable, its width, the array of variables within the skip pattern, the array of their widths, and three statement labels, letting the keyword parameters quietly default.

**How I Really Used Macros: A Better Example**

As I have already mentioned, BSKIP was not the only macro I used to edit skip patterns, although it was the fanciest. I used three other macros as well: one, SSKIP, much like BSKIP, to handle skip patterns in which a single variable is skipped; and two variants of a macro to set the skip indicator. The skip-indicator macros, SKIN1 and SKIN2, reflect the most common coding patterns for skips in this particular questionnaire.

In a real edit situation (Figure 5), each skip-pattern edit will require a minimum of two statements: one invoking a skip-indicator macro and one a skip macro. Additional statements may be needed. Since BSKIP uses arrays, these arrays (of the variables to be skipped and their field widths) must be set up before BSKIP is invoked. And editing a nested skip pattern requires a statement that tests whether the nesting flag is set.

On the whole, however, a program to edit skip patterns will consist largely of statements invoking macros. The result looks concise, but only to the user. Even SAS has its limits.

One of the questionnaires I edited with these macros involved quite a few skips, although the resulting edit program did not look particularly imposing. But with all the statements generated by real macros, the program was so big that it caused an overflow in an internal compiler table. Dividing the edits between two programs eliminated the problem.

**Conclusion: How You Too Can Use Macros**

Skip patterns are not confined to questionnaires, but occur in other types of data-collection forms as well. Even so, skip-pattern editing is a fairly narrow application, and the specific illustration I have presented may not relate directly to the concerns of many SAS users. But any editing situation that applies the same editing rules to different groups of variables becomes a candidate for a program using macros.

If you are confronting a situation that makes you yearn for a subroutine, you can probably come up with a macro to do the job. I would urge you to consider macros in editing applications if you would like to make things easy on yourself, and especially if your neck needs saving.
**FIRST MACRO: BSKIP**

```plaintext
%MACRO BSKIP(EVVAR,RWID,ESARR,WARR,ERRA,
 CTNT,SKYVS,NEST=NST,SKP=SK);  
%*** BIG SKIP MACRO. TAKES ROUTING
% VARIABLE, ITS FIELD WIDTH, 
% ARRAY OF VARIABLES WITHIN SKIP
% PATTERN, ARRAY OF CORRESPONDING
% FIELD WIDTHS, SKIP INDICATOR, 
% 3 STATEMENT LABELS, AND A
% NESTED-SKIP FLAG;
% ENEST=0;
%*** CHECK IF ROUTING VARIABLE
% INDICATES SKIP;
% IF &SKP=1 THEN DO;
% DO OVER ESARR;
% IF ESARR NE -1 & ESARR NE -8
% THEN GO TO EERRA;
% END;
% DO OVER ESARR;
% &ESARR=-3;
% X*** SKIP OKAY—PASSES CHECK;
% GO TO CONT;
% END;
% ELSE IF &SKP=2 THEN DO;
% X*** ROUTING VARIABLE INDICATES NO SKIP;
% IF &ESARR NE -1 & &ESVAR NE -8
% THEN GO TO CONT;
% X*** ALL SKIPPED WHEN IT SHOULDN'T BE;
% GO TO EERRA;
% END;
% X*** INVALID ROUTING VARIABLE
% X*** SKIP INDICATOR—ERROR B;
% ELSE IF &RVAR=ESK1 OR &RVAR=ESK2
% THEN &IND=1;
% ELSE IF &ESARR=ENS1 OR &ESARR=ENS2
% THEN &IND=2;
% END
% END SKIN2;
% Definition of Macros SKIN1 and SKIN2
```

**SECOND MACRO: SSKIP**

```plaintext
%MACRO SSKIP(EVVAR,RWID,ESVAR,SWID,
 ERRA,CTNT,SKYVS,SKP=SK);  
%*** MACRO FOR SINGLE-VARIABLE SKIP, 
% MUCH LIKE BSKIP MACRO;
%*** DON'T NEED NESTED-SKIP FLAG IN 
% THIS CASE;
%*** CHECK IF ROUTING VARIABLE
% INDICATES SKIP;
% IF &SKP=1 THEN DO;
% IF &ESVAR NE -1 & &ESVAR NE -8
% THEN GO TO EERRA;
% &ESVAR=-3;
% X*** SKIP OKAY—PASSES CHECK;
% GO TO CONT;
% END;
% ELSE IF &SKP=2 THEN DO;
% X*** ROUTING VARIABLE INDICATES NO SKIP;
% IF &ESVAR NE -1 & &ESVAR NE -8
% THEN GO TO &CONT;
% X*** SKIPPED WHEN IT SHOULDN'T BE;
% GO TO EERRA;
% END;
% X*** INVALID ROUTING VARIABLE
% X*** SKIP INDICATOR—ERROR B;
% ELSE IF &RVAR=ESK1 OR &RVAR=ESK2
% THEN &IND=1;
% ELSE &IND=6;
% &ESVAR=&ESVAR+&IND*10•••ERWID;
% X*** EXAMINE VARIABLES WITHIN 
% SKIP PATTERN;
% ESKYS: DO OVER ESARR;
% IF ESARR=ESKRD THEN DO;
% IF &ESARR=ENS1 OR &ESARR=ENS2
% THEN &IND=2;
% END;
% ELSE &IND=1;
% IF INT(ESARR/10**ESWID)=2
% THEN FLAG=4;
% ELSE FLAG=6;
% &ESARR=ESARR+FLAG*10**ESWID;
% END;
% CONT:
% %MEND SSKIP;
% Definition of Macro SSKIP
```

**Definition of Macro BSKIP**

```plaintext
XMACRO BSKIP(EVVAR,RWID,ESARR,WARR,ERRA,
CTNT,SKYVS,NEST=NST,SKP=SK);
%*** BIG SKIP MACRO. TAKES ROUTING
% VARIABLE, ITS FIELD WIDTH, 
% ARRAY OF VARIABLES WITHIN SKIP
% PATTERN, ARRAY OF CORRESPONDING
% FIELD WIDTHS, SKIP INDICATOR, 
% 3 STATEMENT LABELS, AND A
% NESTED-SKIP FLAG;
% ENEST=0;
%*** CHECK IF ROUTING VARIABLE
% INDICATES SKIP;
% IF &SKP=1 THEN DO;
% DO OVER ESARR;
% IF ESARR NE -1 & ESARR NE -8
% THEN GO TO EERRA;
% END;
% DO OVER ESARR;
% &ESARR=-3;
% X*** SKIP OKAY—PASSES CHECK;
% GO TO CONT;
% END;
% ELSE IF &SKP=2 THEN DO;
% X*** ROUTING VARIABLE INDICATES NO SKIP;
% IF &ESARR NE -1 & &ESARR NE -8
% THEN GO TO CONT;
% X*** ALL SKIPPED WHEN IT SHOULDN'T BE;
% GO TO EERRA;
% END;
% X*** INVALID ROUTING VARIABLE
% X*** SKIP INDICATOR—ERROR B;
% ELSE IF &RVAR=ESK1 OR &RVAR=ESK2
% THEN &IND=1;
% ELSE &IND=6;
% &ESVAR=&ESVAR+&IND*10•••ERWID;
% X*** EXAMINE VARIABLES WITHIN 
% SKIP PATTERN;
% ESKYS: DO OVER ESARR;
% IF ESARR=ESKRD THEN DO;
% IF &ESARR=ENS1 OR &ESARR=ENS2
% THEN &IND=2;
% END;
% ELSE &IND=1;
% IF INT(ESARR/10**ESWID)=2
% THEN FLAG=4;
% ELSE FLAG=6;
% &ESARR=ESARR+FLAG*10**ESWID;
% END;
% CONT:
% %MEND BSKIP;
% Definition of Macro BSKIP
```

**Definition of Macro SSKIP**

```plaintext
XMACRO SSKIP(EVVAR,RWID,ESVAR,SWID,
ERRA,CTNT,SKYVS,SKP=SK);
%*** MACRO FOR SINGLE-VARIABLE SKIP, 
% MUCH LIKE BSKIP MACRO;
%*** DON'T NEED NESTED-SKIP FLAG IN 
% THIS CASE;
%*** CHECK IF ROUTING VARIABLE
% INDICATES SKIP;
% IF &SKP=1 THEN DO;
% IF &ESVAR NE -1 & &ESVAR NE -8
% THEN GO TO EERRA;
% &ESVAR=-3;
% X*** SKIP OKAY—PASSES CHECK;
% GO TO CONT;
% END;
% ELSE IF &SKP=2 THEN DO;
% X*** ROUTING VARIABLE INDICATES NO SKIP;
% IF &ESVAR NE -1 & &ESVAR NE -8
% THEN GO TO &CONT;
% X*** SKIPPED WHEN IT SHOULDN'T BE;
% GO TO EERRA;
% END;
% X*** INVALID ROUTING VARIABLE
% X*** SKIP INDICATOR—ERROR B;
% ELSE IF &RVAR=ESK1 OR &RVAR=ESK2
% THEN &IND=1;
% ELSE &IND=6;
% &ESVAR=&ESVAR+&IND*10•••ERWID;
% X*** EXAMINE VARIABLE WITHIN 
% SKIP PATTERN;
% ESKYS: DO OVER ESARR;
% IF ESARR=ESKRD THEN DO;
% IF &ESARR=ENS1 OR &ESARR=ENS2
% THEN &IND=2;
% END;
% ELSE &IND=1;
% IF INT(ESARR/10**ESWID)=2
% THEN FLAG=4;
% ELSE FLAG=6;
% &ESARR=ESARR+FLAG*10**ESWID;
% END;
% CONT:
% %MEND SSKIP;
% Definition of Macro SSKIP
```
Q.1. Do you have any teeth?
1. Yes
2. No (Skip to Q.4)

Q.2. How many times a day do you brush your teeth?
1. (Don't brush at all) 
2. Number of times

Q.3. Do you use toothpaste?
1. Yes
2. No

Q.4. Have you ever had surgery on your gums?
1. Yes
2. No

Figure 1. Example of skip pattern in questionnaire and corresponding flow chart

Figure 2. General edit steps for skip patterns

Figure 3. Error-processing routines A and B, expanded

Figure 4. Nested skip patterns
Acknowledgments

I would like to thank Dr. Graham Burkheimer, Jr. and Jean Richardson for the detailed and exhaustive edit specifications that made the development of the above-mentioned macros possible, not to mention necessary. Figures 2, 3, and 4 are very similar to charts provided by Dr. Burkheimer.

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

For additional information, contact:

Susan Bolotin
Research Triangle Institute
P.O. Box 12194
Research Triangle Park, NC 27709
(919) 541-6095