SAS® Macros Are the Cure for Quality Control Pains

Implementing A Macro Driven Evaluation of Survey Data for Quality Control

Gary McQuown, Data and Analytic Solutions, www.DASconsultants.com

ABSTRACT

Performing Quality Control (QC) on survey data is an essential and challenging, but often under appreciated task. Differences such as survey length, skip patterns, variable ranges, and questionnaire length add to the difficulty of the task. Fortunately, prudent use of macros can reduce the chore while improving accuracy. This paper presents the macros and techniques used to perform quality control on the *1998 Survey of Small Business Finances* data for the Board of Governors of the Federal Reserve. The macros and routines described are used to perform various evaluations and cleaning procedures including the following:

1) Determine validity of each response.
2) Accommodate exceptions and missing values.
3) Differentiate between legitimate skips, conditional skips, and inappropriate skips.
4) Report errors in an appropriate format.

The macros and routines presented may be useful for anyone who wishes to improve upon their QC procedures or performs survey related work.

INTRODUCTION

The Board of Governors of the Federal Reserve System wished to determine and document the validity of responses received during their *1998 Survey of Small Business Finances*. The purpose of the task is to condition the survey data set to help control imputation of missing values and provide the economists and others with information about which values on the public data set are imputed. The 226 page survey consists of questions and criteria for over one thousand variables. Additional factors that contribute to the complexity of the task are the number and variety of skip patterns and the number of questions that can entail single or multiple responses. The successful completion of the task is accomplished through the use of macros imbedded within conditional statements that mirror the survey instrument design.

The end product is a data set that contains the original response values and a set of "shadow variables", or Xcodes, that describe the validity of the original response. Each response value is evaluated to determine if it conforms to a set of expected values, falls within an expected range, and whether a specific question is appropriately asked or legitimately skipped. For every variable, there must be a corresponding Xcode to store the result of the evaluation of that variable. If a value is evaluated as a legitimate skip, the missing response value is replaced with the special missing code of .S. This is the only situation in which the original value is overwritten.

The following is a list of the possible Xcodes values and the conditions that they denote. Conditions such as "Valid", "Invalid", "Don't Know", "Refuse", etc are general in nature and appear on most surveys. Conditions for Conditional Skips and Derived are more unusual.

Xcode = condition description

- 9 = Valid and within appropriate range.
- 1 = Failed a soft range check, but is otherwise valid.
- 2 = Don't Know (D or .D)
- 3 = Refuse (R or .R)
- 4 = Exception (X or .X)
- 5 = Legitimate skip. Based on preceding values, the question legitimately was not asked.
- 6 = Conditional skip, value may be, legitimate, missing or may require imputation due to a .D or .R in a previous response.
- 7 = Non-missing but should be missing
- 8 = Missing but should be non-missing
- 0 = External, merged from the external data set.
- v = Invalid categorical response, out of range or failed a "hard" check.
- w = Derived/calculated incorrectly, pertains to internal flags and counters
- m = Missing value after inappropriately missing conditional value

If the questionnaire design and the respondent's prior responses indicate that a response should be present, the PRESCHK macro (appendix 2) is used for the evaluation. The PRESCHK macro produces Xcode values of 9, 1, 2, 3, 4, 8, 0, v and w. If the response is expected to be missing, the MISSCHK (appendix 3) macro is used. The MISSCHK macro produces Xcode values of 5, 6, 7, and m. Variables should only be evaluated once. Multiple evaluation or evaluations by the incorrect macro will create duplicate values in the output data set and may falsely increase the error count.

ARRAYS

The first step in the use of the macros for survey review is to create two arrays, one for character variables (*section*_char) and a second for numeric variables (*section*_num). The XLISTBLD macro (appendix 1) is used to create corresponding arrays of Xcodes (_x*section*char and _x*section*num) from these arrays. For every variable to be evaluated, a matching Xcode variable is created to store the results of the evaluation. Xcodes are prefixed with the letter x, and followed by the name of the variable they represent (xf2_1 = f2_1). Thus, prior to Version 7, variable names should be limited to a length of seven. Variables are ordered the way they appear in the questionnaire so that the arrays can be used to process a series of variables within a line of code.

Example:

```
%let section = f1;
%let f1_char = f16 ;
%let f1_num  =
    f1     f2      f2_1   f3
    f3_1   f3_1_1  f4     f5
    f5_1   f6      f6_1   f6_1_1
    f7     f9_1    f9_2   f9_3
    f10_1  f10_1_1 f10_2  f10_1_2
    f10_3  f10_1_3 f11_1  f11_1_1
    f11_2  f11_1_2 f11_3  f11_1_3
    f13_1  f13_2   f13_3  f14_1
    f14_2  f14_3   f14v_1 f14v_2
    f14v_3 f15_1   f15_2  f15_3  ;

array _&section.char  (*) $ &f1_char;
array _x&section.char (*) $ 2
   %xlistbld(&f1_char);
array _&section.num  (*) &f1_num ;
array _x&section.num (*) $ 2
   %xlistbld(&f1_num) ;
```

The next step is to construct a series of if/then statements determined by the questionnaire's skip patterns and the respondent's answers. The if/then statements should direct SAS® to process every variable only once. Because the questionnaire is sequential, prior responses and skips dictate whether a specific question should be asked. Consequently, the value of the corresponding variable should be evaluated with the PRESCHK macro. If the question is to be skipped, the response should be missing and the value is checked with the MISSCHK macro. Both macros evaluate the value, assign the appropriate value to the associated Xcode and output that record to a permanent output file.

PRESCHK .. If a response should be present.

Syntax: %preschk(
var = *required* ,
range = *required* ,
repvar = *optional*,
high = *optional*,
low = *optional*)

Example : %preschk(
var = f3,
range =%str(1,2,.D,.R),
repvar = f1 f2 f2_1)

The purpose of PRESCHK is to evaluate a response when it is expected to be present. The parameters are:

VAR = the variable to be reviewed.
RRANGE = the acceptable range for numeric values.
REPVAR = a list of variables and their values to be printed on the error report.
LLOW = minimum value for numeric values.
HIGH = maximum value for numeric values.

As a variable is passed through the PRESCHK macro, the macro compares the name of the variable to all values in the character array.

If the name is located, TYPE is set to C and it is processed as a character variable. Character variable responses are evaluated to determine if they are present, missing, or equal to certain predetermined values in ("D","R"). Each instance results in the macro assigning a different value to the corresponding Xcode (see chart below).

If the variable is not located TYPE is set to N and it is processed as a numeric variable. Numeric responses are evaluated to determine if they are present, missing or equal to various special-missing values (.D,.R). Numeric values are also evaluated to determine if they are within a desired range (RNG).

The following examples of the RANGE parameters can be easily modified:

RANGE	DESCRIPTION OF RANGE
%STR(x,y,z)	ie. %str(1,2,.D,.R)
POSINT	var > 0 or in (.D,.R)
GEZERO	var >= 0 or in (.D,.R))
M1GZERO	var > 0 or in (-1,.D,.R))
M1GEZRO	var >= 0 or in (-1,.D,.R))
RNG	low <= var <= high or in (.D,.R)
ANYNUM	not missing or in (.D,.R)

MISSCHK .. If a response should be missing.

Syntax: %misschk(
var = *required*,
depvar = *optional*,
repvar = *optional*)

Example: %misschk(
var = S10_3,
depvar = S10,
repvar = S10 S10_1 S10_2)

The purpose of the MISSCHK macro is to identify errors when a value is expected to be missing. The parameters are

VAR = the variable to be reviewed.
REPVAR = a list of variables and values to be printed on the error report.
DEPVAR = a variable that if DK or RF will cause this variable to be computed.

After the macro determines if the variable is character or numeric, it checks to see if the value is missing. Any value present is considered an error. In some instances a response that is missing may need to be computed for analysis due to a "Don't Know" or "Refuse" in a previous response. If the REPVAR variable is DK or RF and this variable is missing, it is considered a conditional skip. If the dependent variable is missing but should have been present, a missing conditional skip value is assigned. All other missing values are considered legitimate skips.

Use Of Macros Within Conditional Statements

In the following example, all respondents are asked question F1 so a value is expected, and it is evaluated with the PRESCHK macro with range = %str(1, 2, .D, .R). Any value other than those in the %str will be evaluated as an error, and the corresponding Xcode will be assigned a value of v.

A response of 1 signifies that F2 should also be evaluated with the PRESCHK macro. The macro will determine that 1 is within the range given and assign a value of 9 to the corresponding Xcode.

If the response to F1 is 2, all questions between F1 and F4 are skipped and expected to be missing, therefore they are evaluated with the MISSCHK macro. If the response values are indeed missing, the MISSCHK macro assigns the corresponding Xcode a value of 5 and replaces the missing response value with a the special missing value of .S.

A response of either .D or .R signifies that all F2 and F3 variables should be missing, but the conditional criteria means that they may need to be imputed at a later date. Therefore they will be evaluated by the MISSCHK macro with the REPVAR parameter set to F1. Xcodes for conditional skips are assigned a value of 6 and the missing response value is not altered.

If F1 = 1, F2 is evaluated with PRESCHK with the RNG parameter set to GEZERO so that any value less than zero will be considered an error. If a dollar value is given, F2_1 is skipped and evaluated with the MISSCHK macro. If the response to F2 is .D or .R then F2_1 will be evaluated with the same macro and parameters as F2.

Example of Survey Question:

F1: Did [Firm] use owners' personal credit cards to pay business expenses ?
Yes…… 1
No …… 2 → Go to F4
DK or RF ……. D or R → Go to F4 (conditional)

F2: On average, how much per month in new business expenditures did the firm charge to the owners personal credit cards?
Amount $ ____→ Go to F3 (non negative number)
DK or RF ……. D or R

F2_1: Could you give an estimate?
Amount $ ____→ Go to F3 (non negative number)
DK or RF ……. D or R

Example of Code:

```
%preschk(var=f1,range=%str(1,2,.D,.R))
  if f1 = 1 then do;
    %preschk(var=f2,range=GEZERO)
    if f2 in (.D,.R) then
    do ;
      %preschk(var=f2_1,range=GEZERO)
    end ;  else
    do ;  /* f2 not .D,.R */
      %misschk(var=f2_1,repvar=f2)
    end;
  /* more code for F3, F3_1, F3_1_1 */
  end ; else  /*legitimate skip */
  if f1 = 2 then do;
     %misschk(var=f2,repvar=f1)
     %misschk(var=f2_1,repvar=f1)
     %misschk(var=f3,repvar=f1)
     %misschk(var=f3_1,repvar=f1)
     %misschk(var=f3_1_1,repvar=f1)
  end ; else
  if f1 in (.D,.R) then
/* conditional skip and do loop to
perform multiple checks sequentially */
```

```
  do i = 2 to 6;
  %misschk(var=_&section.num(i),
           depvar=f1)
end;
```

ERROR REPORT

Both PRESCHK and MISSCHK append observations to a permanent data set. This data set contains the SU_ID, variable name (varname) and value, section, and Xcode for every SU_ID Variable combination . It also contains the name and value of reference (repvar) and dependent (depvar) variables . The error report is generated by printing observations where the Xcode values are 1, 7, 8, v, or w.

The following example shows three errors found in section F. For the first observation, the response to question F1 is missing when it should be present. The second observation shows an instance when a value is present, but outside the acceptable range. The final example occurs when the value should have been skipped due to the value of a prior response. In each instance, the repvar variable names and their values are printed with the value of the variable in question.

Extract from Error Report:

Section	Variable	SU_ID	Xcode	Value(s)
F	F1	10004390	8	F1 =
	F2	10004520	v	F 2 = -1000
				F1 = 1
	F2_1	10006310	7	F2_1 = 2500
				F1 = 1
				F2 = 5000

CONCLUSION

The length and complexity of the Board of Governors of the Federal Reserve System's *1998 Survey of Small Business Finances* made quality control and the evaluation of the response variables more difficult. The procedures were accomplished by performing the processing in small sections with arrays and by standardizing the most common tasks with macros. PRESCHK processed values where the response was expected to be present and MISSCHK processed the remaining values that were expected to be missing. The macros and routines can be modified or directly incorporated into other projects. Changes would depend upon the specifics of the survey being reviewed. The most probable modifications will be in the allocation of alternative Xcode values, ranges, and the elimination of the conditional skip.

Appendix 1: MACRO XLISTBLD

```
%macro xlistbld (varlist) ;
  %local i ;
  %let i = 1 ;
  %do %while (%scan(&varlist,&i) ^= %str( )) ;
  x%trim(%scan(&varlist,&i))
  %let i = %eval(&i+1) ;
  %end ;
%mend xlistbld ;
```

Appendix 2: MACRO PRESCHK

```
%macro preschk (
          var=,range=,type=N,repvar=,low=,high= ) ;

 /* PRESCHK is used to verify that a variable is
present  when it is supposed to be.
    var = variable
    range = acceptable range of variable (from list)
    type = N for number, C for Character
    repvar = report variable for QC
    low = minimum value if range = RNG is used
    high = maximum value if range = RNG is used  */

   %local word i rep rvar ;
    %if %index( %STR( &&&section._char ),
         %str(       &var ) ) > 0 %then
    %do ;
    if &var = ' ' then
      %Let word = &var ;
      %let type = C ;
      %end ;   %else
    %do ;
      %let word = left(put(&var,12.)) ;
    if &var = . then
    %end ;
   /* variable has a missing value incorrectly */
     do ;
        x&var = '8' ;
        errtype = '8' ;
        variable = "&var" ;
        value = "&var=" || left( &word ) ;
        section = "&section" ;
        output bads ;
    %let i = 1 ;
    %do %while (%scan(&repvar,&i) ^= %str() ) ;
    %let rvar = %upcase(%scan(&repvar,&i)) ;
    %if %index( %STR( %upcase(&&&section._char)
                  ),%str( &rvar ) ) > 0 %then
      %let rep = &rvar ;
     %else
```

```
      %let rep = left(put(&rvar,12.)) ;
       value = "%scan(&repvar,&i) =" || left( &rep ) ;
       output bads ;
       %let i = %eval(&i + 1) ;
     %end ;
     end;
   %if &range ^= %str() %then
     %do ; else
/* When a range parameter is specified,
     check the value */
do;
%if %upcase(&range) = POSINT %then
   %str(if not( &var > 0 or &var in (.D,.R) ) ) ;
%else
%if %upcase(&range) = GEZERO %then
   %str(if not( &var >= 0 or &var in (.D,.R) ) ) ;
%else
%if %upcase(&range) = M1GZERO %then
   %str(if not( &var > 0 or &var in (-1,.D,.R) ) ) ;
%else
%if %upcase(&range) = M1GEZRO %then
   %str(if not( &var >= 0 or &var in (-1,.D,.R) ) ) ;
%else
%if %upcase(&range) = MON %then
   %str(if not ((1 <= &var <= 12) or &var in (.D,.R) ) )
;   %else
%if %upcase(&range) = MONX %then
   %str(if not ((1 <= &var <= 12) or &var in (.D,.R,.X)
)
            ); %else
%if %upcase(&range) = YEAR %then
   %str( if not ((1900 <= &var <= 1999) or &var in
            (.D,.R) ) ) ;
%else
%if %upcase(&range) = RNG %then
%str( if not ((&low <= &var <= &high) or &var in
(.D,.R) ) ) ;
%else
%if %upcase(&range) = ANYNUM %then
%str( if not ((.^= &var) or &var in (.D,.R) ) ) ;
%else
%if &type = N %then
%str( if &var not in (&range)) ;
%else
%str( if length(&var) = 1 and &var not in ('D','R') ) ;
        then do ;
        x&var = 'v' ;
        errtype = 'v' ;
        variable = "&var" ;
        value = "&var=" || left( &word ) ;
              section = "&section" ;
        output bads ;
        %let i = 1 ;
        %do %while (%scan(&repvar,&i) ^= %str() ) ;
          value = "%scan(&repvar,&i) =" || left(
%scan(&repvar,&i)) ;
          output bads ;
          %let i = %eval(&i + 1) ;
        %end ;
        end ; else
        do;    /* set to valid and not skipped */
         if &var = .D then x&var = '2'; else
         if &var = .R then x&var = '3'; else
         if &var = .X then x&var = '4'; else
         x&var = '9';
         errtype = '9' ;
         variable = "&var" ;
         value = "&var=" || left( &word ) ;
         section = "&section" ;
         output bads ;
        end;
      end ; /* value present */
    %end ; /* range supplied */
    %if &type = C and &var ^= ' ' %then
     %do ;
       if length(&var) > 1 then
       do;
         x&var = '9' ;
         errtype = '9' ;
         variable = "&var" ;
         value = "&var=" || left( &word ) ;
         section = "&section" ;
         output bads ;
       end;  else
       if length(&var) = 1 and &var in ('D')  then
       do;
         x&var = '2' ;
         errtype = '2' ;
         variable = "&var" ;
         value = "&var=" || left( &word ) ;
         section = "&section" ;
         output bads ;
       end;  else
       if length(&var) = 1 and &var in ('R')  then
       do;
         x&var = '3' ;
         errtype = '3' ;
         variable = "&var" ;
         value = "&var=" || left( &word ) ;
         section = "&section" ;
         output bads ;
       end;  else
       if length(&var) = 1 and &var not in ('R' 'D')
then
       do;
         x&var = 'v' ;
         errtype = 'v' ;
         variable = "&var" ;
         value = "&var=" || left( &word ) ;
         section = "&section" ;
         output bads ;
           %let i = 1 ;
           %do %while (%scan(&repvar,&i) ^= %str()
);
```

```
            %let rvar = %upcase(%scan(&repvar,&i)) ;
            %if %index( %STR(
%upcase(&&&section._char) ),%str( &rvar ) ) > 0
%then
             %let rep = &rvar ;
          %else
            %let rep = left(put(&rvar,12.)) ;
            value = "%scan(&repvar,&i) =" || left( &rep )
;
            output bads ;
            %let i = %eval(&i + 1) ;
          %end ;
        end;
      %end;
%mend preschk ;
```

Appendix 3: MACRO MISSCHK

```
%macro misschk (var=,depvar=,repvar=) ;

  /* This is a utility macro used to verify that a
variable is missing when it is supposed to be.
    var = variable to be evaluated
    depvar = dependent variable.. if special missing
             then conditional not legitimate skip
    repvar = report variable for QC, printed on error
             report           */

  %local word i type rvar rep ;
  %if %index( %STR( &&&section._char ),
             %str( &var  ) ) > 0 %then
    %do ;
    if &var = ' ' then
    %Let word = &var ;
    %let type = C ;
    %end ;
  %else
    %do ;
    %let word = left(put(&var,12.)) ;
    %let type = N ;
    if &var in(. ,.S) then
    %end ;
    do ;
    x&var = '5' ;
    %if &type = N %then
      %str(&var = .S ; ) ;
    %if &type = C %then
      %str(&var = 'S' ; ) ;
    do ;
    /* if there is a dependency var, then check it for
   DK, RF.  If so,  the tested variable's xvar is set to
   6 (conditionally legit skip), otherwise the xvar is
   set to 5 */
    %if &depvar ^= %str() %then
      %do ;
      if &depvar in (.D,.R) then
       do;
        x&var = '6' ;
        if &var = .S then &var = .;
        end;
        else
       if &depvar in (.) and x&depvar = '8' then
        x&var = 'M';
       %end ;
     end ;
     end ;
     else
     do ;
     x&var = '7' ;
     errtype = '7' ;
     variable = "&var" ;
     value = "&var=" || &word ;
     section = "&section" ;
     output bads ;
     %let i = 1 ;
     %do %while (%scan(&repvar,&i) ^= %str() ) ;
       %let rvar = %upcase(%scan(&repvar,&i)) ;

       %if %index( %STR(
%upcase(&&&section._char)
                    ),%str( &rvar ) ) > 0 %then
        %let rep = &rvar ;
       %else
        %let rep = left(put(&rvar,12.)) ;
       value = "&rvar=" || &rep ;
       output bads ;
       %let i = %eval(&i + 1) ;
     %end ;
     end ; %mend misschk ;
```


ACKNOWLEDGMENTS

The author would like to thank Michael C Hein, Dr. John Wolken, Mike Sadof, Bruce Gilsen, Emily Rosenberg and Dr. Dawn Li for their contributions to the macros and this paper.

CONTACT INFORMATION

Gary McQuown
Data and Analytic Solutions, Inc.

10502 Assembly Drive
Fairfax, VA 22030
Phone: (703) 628-5681
Email: gmcquown@DASconsultants.com
Web: www.DASconsultants.com