Paper 97-28

An Efficient Approach to Combine SAS® Data Sets with Voluminous Variables That Need Name and Other Changes

Grace Chiu and Edward Heaton, Westat (Durham, NC and Rockville, MD)

ABSTRACT

In some longitudinal studies involving the collection of participant data by means of questionnaires, researchers modify their data collection instrument, i.e. questionnaire, after baseline data collection has started. A questionnaire could go through several versions before the entire data-collection process is finished. Consequently, SAS programmers are often required to consolidate multiple wide, dissimilar data sets for analysis.

This paper demonstrates an approach to such a task that builds a Microsoft Excel file as a reference guide to finalize all changes and processes the required changes in SAS. Users will learn to use the **SQL** procedure's **SELECT INTO** clause to build two parallel lists of macro variables for the renaming of numerous variables, macros to change variables, and **PROC APPEND** to enforce gold standard attributes.

Basic programming knowledge in SAS is required. This paper should benefit data management and data processing professionals.

INTRODUCTION

Researchers may need to modify the questionnaires after baseline data collection has started. Questions may be dropped, added, rearranged, or changed in subsequent waves of data collection. These are SAS data sets with the question number built into the variable name; such a naming system is customarily preferred by principal investigators because of handy reference to questions in a questionnaire. Because of this preference, common data elements often have different variable names from one version to another, and same variables sometimes may mean different things in different versions. Furthermore, data standards may vary across the waves of the data collection resulting in differing data types, response patterns, and formats. In other words, since there is no linkage key available from one version of data to another, programmers must match each variable from a version to variables in another version to be sure that they are of the same question, data type, etc. before they can combine them into a single analysis file.

There are three ways to perform the rename:

1. Rename variables one by one;
2. Use arrays to rename strings of variables, section by section; and
3. Create macro variable lists to rename hundreds of variables in one round.

We would not recommend the renaming of variables one at a time for hundreds of variables since it is highly labor intensive and error prone. The use of arrays to rename strings of variables is an improvement and would be fine if only a subset of variables are to be combined for targeted analysis. However, one still has to define separate numeric arrays and character arrays. Going through the array definition process for hundreds of variables is still labor intensive and error prone. Instead, we suggest an approach as follows:

1. Prepare a list of the variables from each data set, including the variable number,
2. Arrange the variables from each data set by their relative position as if they are pair-wise lists, and
3. Use these lists to perform the rename process.

METHODS

Regardless of which approach one takes, some manual preparation is necessary to map variables across all versions as a cross-reference guide. This crucial step would help all parties involved (i.e. the researchers, persons in charge of questionnaire design and of data collection, as well as persons to perform data analysis) come together to see all necessary changes that need to be made and decide the structure of the final combined dataset.

STEP A:

Create a cross-reference table of variable names plus other crucial information where the common variables from each data set, regardless of their names, are on common rows of the table.

Our illustration uses data sets named YEAR1, YEAR2, and YEAR3. Each has over a thousand variables. Among the YEAR1 variables, at least 55% need to be renamed because a question added in the second year of the data collection caused the question numbers to change. The magnitude of the name shifts in YEAR3 is even more massive. Over 92% of the variables in YEAR2 need name conversion because of the expansion of coding a couple of questions in the third year of the survey. Furthermore, revisions in a question require specific manipulation of about 20 variables in YEAR2. (See changes in Vnum across version for same data items in Appendix C Sections of Excel Cross-reference Table.)

For the cross-reference table, we create a list of the variable names, variable numbers, data types, and variable labels from **PROC CONTENTS** for each year data and export them to Excel by calling the following macro:

```
*-----------------------------------*;
* Macro SasToExcel (see Appendix A-1)*;
*-----------------------------------*;

%SasToExcel( in=year1 , out=out1 )
%SasToExcel( in=year2 , out=out2 )
%SasToExcel( in=year3 , out=out3 )
```

Once in Excel, manually match the common items side-by-side so that those common items are aligned into single rows. This is a crucial step in data management, as it should reveal whether the changes in a questionnaire yield intended results; particularly, variables with same name but different meanings should be identified at this stage. The cut-and-paste process is quite simple if the order of questions remains unchanged. However, special attention must be paid to the mapping of out-of-order questions. Such tedious but essential work will enable us to know the datasets very well, paving a smoother way for subsequent data analysis.

After these manipulations, add a column to the Excel table and classify the variables into one of the following categories:

1. No-transformation-necessary,
2. To-be-<u>A</u>dded,
3. To-be-<u>D</u>ropped,
4. To-be-<u>R</u>enamed,
5. To-be-<u>C</u>hanged (i.e., changes in data values or formats such as character to numeric or vice versa).

We will treat the first category as a special case of rename (i.e. category 4). Variables with same names but in fact different are classified as category 3; they need to be added back to the final data set as variables with different names. As dropping and adding does not require transformations, we only need to address conversion of the last 2 categories. (Theoretically the last two categories may not be mutually exclusive as some to-be-renamed variables may still need change in type. If that is the case, another category will be added to cover all needed operations. For this paper, we will treat them as mutually exclusive for simplicity.)

STEP B:
Perform SAS operations using the Excel cross-reference table as a guide:

1. VARIABLES THAT NEED RENAMED*:*

Since there are no common variable lists available, the first step is to prepare the lists using the Excel table as a reference guide. The preparation of **LIST1** and **LIST2** is hard-coded and requires double-checking. Afterward, the process becomes more automatic.

```
*------------------------------------*;
* Output NAME and VARNUM for list    *;
* processing                         *;
*------------------------------------*;

Proc contents
   data=data.year1
   out=list1( keep= name varNum )
;
Run ;

Proc contents
   data=data.year2
   out=list2( keep= name varNum )
;
Run ;

/* Hard code list from YEAR1: */

Data list1 ;
   Set list1( where=(
    (    ( 2 <= varNum <= 1057 )
    and ( varNum not in ( 977, 981 ))
   ) ) ;
Run ;

/* Corresponding list from YEAR2: */
```

```
Data list2 ;
   Set list2( where=(
    (   (    2 <= varNum <=  468)
      | ( 490 <= varNum <= 1083) )
    and ( varNum not in (
         167 , 169,
         1000, 1001, 1005, 1006,
         1034, ))
   ) ) ;
Run ;
```

We now have two data sets with same number of records and each has two variables with values of **NAME** and **VARNUM**. How to turn them into two lists of matching names? The trick is to use the **VARNUM** as it contains the **NAME**'s sequential position in the original data. In other words, we only need to sort the lists by **VARNUM**:

```
Proc sort data=list1 ;   By varNum ;
Run ;
Proc sort data=list2 ;   By varNum ;
Run ;
```

(Note: Such list mapping is version-specific. We need to map another two lists when dealing with YEAR2 and YEAR3 data.)

Then we are ready to call the **%_Rename** macro to create the dataset:

```
*------------------------------------*;
* Macro  _Rename (see Appendix B-1)  *;
*------------------------------------*;

%_Rename(   data=data.year1
          , out =data.v1tov2_ren)
```

With the **LIST1** and **LIST2** prepared above, the **SQL** procedure in the **%_RENAME** macro uses the **SELECT INTO** clause to produce macro variables for each variable name list (i. e., **&V1** and **&V2**) as well as a system macro variable (**&SQLOBS**) which is equivalent to the number of variable-name pairs.

In the **DATA** step inside the **%_RENAME** macro, the macro variables **&V1** and **&V2** are scanned in pairwise fashion for **&SQLOBS** times into the **RENAME** statement, producing a data set with variable names compatible to YEAR2.

Depending on when versions of the data need processed, one might rename YEAR1 to the YEAR2 data format and then combine them for intermediate analysis. The intermediate could then be converted into the YEAR3 data style when YEAR3 data is available, until all data collections have the same structure. If the gold standard is represented by YEAR2, then reverse the standard when applying the macros.

The **%_RENAME** macro takes care of over 98% of the work in YEAR1 dataset. Only a remaining 2% need recode in data value or type and format changes, which makes the task much more manageable.

2. TO-BE-CHANGED VARIABLES:

In YEAR1 data set, we performed the recoding individually since the situations are unique. At any rate, it is relatively simple to use a macro for repeated recoding patterns. (See **%_RECODE** in Appendix B-2.)

If variable names are the same but the data types are different, we usually create a temporary variable with the desired format first, then drop the original variable and change the name of the temporary variable to its original name. Macro **%_TypeCHANGE** can be used for multiple variables that need conversions in a **DATA** step (See Appendix B-3). There are 3 keyword parameters required to call this macro: **invar** lists the variables; **type** lists corresponding type changes, i.e. CtoN or NtoC; and corresponding new **format** in the case of CtoN type of change, old **format** for NtoC. The latter is because the **PUT** function is used inside the macro to convert numeric to character type, and SAS requires the format for the SOURCE variable in a **PUT** function. Otherwise one will get a warning message from the SAS log.

```
*-------------------------------------*;
*Macro _TypeCHANGE (see Appendix B-3)*;
*-------------------------------------*;

%_TypeCHANGE (
   invar = N11 N13 N14
 , type  = NtoC CtoN CtoN
 , format= %str(1. 4. 4.)
  )
```

After such data handling, we then use **PROC APPEND** to update all conversion results. First, all variables need to be merged together (i.e., those already renamed, already recoded, and those that had type changes) by record ID to turn them into one intermediate file. Then use YEAR2 data as the standard for update. The use of work files in our example is intended to allow for data checks before final output.

```
Proc append
   base=work.year2
   data=intermediate
;
Run ;
```

If there are variables in YEAR1 but not in YEAR2 that need to be kept in the final analysis dataset, the **FORCE** option of the **PROC APPEND** will be required.

A check on variable distribution (by version) can be done by calling **%_REPORT** at any stage of conversion:

```
*-------------------------------------*;
* Macro _Report (see Appendix B-4)    *;
*-------------------------------------*;

%_Report(
   varList= n50 n122 n20101
 , inData=data.v1tov2_ren
)
%_Report(
   varList= n53 n67_mo n203d_01
 , inData=work.year2
)
```

This concludes the data combining process for YEAR1 and YEAR2. For additional versions of data combination, one will repeat the same process starting from variable mapping in Excel.

CONCLUSION

In this paper we have demonstrated how to rename voluminous variables across versions of SAS datasets, use macros to change variables, and produce a final file according to a standard data vector. Another approach is to use the Excel cross-reference table as control data for all conversions. However, to fully illustrate the Excel data driven processing will be the subject matter of another paper.

DISCLAIMER: The contents of this paper are the work of the authors and do not necessarily represent the opinions, recommendations, or practices of Westat.

ACKNOWLEDGMENTS:

We want to thank Ian Whitlock, Duke Owen and Marsha Shepherd of Westat for their help with the logistics for this paper. We also want to thank all of the insightful SAS-Lers for their selfless contributions. They have proven invaluable for our career growth.

CONTACT INFORMTION:

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

Grace Chiu, PhD
Senior Systems Analyst
Westat
1009 Slater Road, Suite 110
Durham, NC 27703
Tel: (919) 941-8310
Fax: (919) 941-9355
Email: Chiu@niehs.nih.gov

APPENDIX

```
*-------------------------------------*;
* A-1: Export from contents to Excel *;
*-------------------------------------*;

%macro SasToExcel ( data= , out= ) ;

   Proc contents
      data=data.&data
      out =&data (
           keep=name varNum type label
           )
      noPrint
   ;
   Run ;

   Proc sort data=&data ;
      By varNum ;
   Run ;
   Proc export
      dbms   =excel
      data   =work.&data
      outFile=&out
      replace
   ;
   Run ;

%mEnd SasToExcel ;


*-------------------------------------*;
* B-1: _Rename macro                  *;
*-------------------------------------*;

%macro _Rename (
   data    =
 , out     =
 , fromList=list1
 , toList  =list2
) ;

   *---------------------------------*;
   * Obtain macro variables for name *;
   * pairs.                          *;
   *---------------------------------*;

   Proc sql noPrint ;
      Select
         name into :v1 separated by " "
         from &fromList
      ;
      Select
         name into :v2 separated by " "
         from &toList
      ;
   Quit ;

   *---------------------------------*;
   * Use do loop to perform rename   *;
   *---------------------------------*;

   Data &out ;
      Set &data ;
      Rename
         %do i=1 %to &sqlObs;
            %scan(&v1,&i)=%scan(&v2,&i)
         %end ;
      ;
   Run ;

%mEnd _Rename;


*-------------------------------------*;
* B-2: Recode value of 2 to 0         *;
*-------------------------------------*;

%macro _Recode ( from= , to= ) ;

   If ( &from eq 2 ) then &to = 0 ;

%mEnd _Recode ;


*-------------------------------------*;
* B-3: Change type of variables       *;
*-------------------------------------*;
```

```
%macro _TypeChange (
   inVar =
 , type  =
 , format=%str()
);

%let _i = 1 ;
%let var = %scan( &inVar , &_i ) ;

%do %while ( &var ne %str() ) ;
   %let _i  = %eval( &_i + 1 ) ;
   %let var = %scan( &inVar , &_i ) ;
%end ;

%let numVar = %eval( &_i - 1 ) ;

%do i=1 %to &numVar ;

   %let fmt&i = %scan( &format , &i ) ;
   %let var&i = %scan( &inVar , &i ) ;
   %let typ&i = %scan( &type , &i ) ;
   %let len&i =
      %sysFunc( compress(&&fmt&i,'.$') );

   %if %upCase(&&typ&i)=CTON %then %do ;
      Format &&var&i ;
      Length tempA&i %unquote(&&len&i) ;
      tempA&i =input(&&var&i,&&fmt&i...);
      Drop &&var&i ;
      Rename tempA&i = &&var&i ;
   %end ;
   %else %if %upCase(&&typ&i)=NTOC
      %then %do ;
      Format &&var&i ;
      Length tempB&i $%unQuote(&&len&i) ;
      tempB&i = put(&&var&i,&&fmt&i...) ;
      Drop &&var&i ;
      Rename tempB&i=&&var&i ;
   %end ;

%end ;

%mEnd _TypeChange ;

*-------------------------------------*;
* B-4: _Report Macro                  *;
*-------------------------------------*;

%macro _Report (
   data=
 , var =%str( )
) ;

Title3 "Crosstab of selected variables x
Version, data= &data" ;

   Proc Freq data=&data ;
      Tables (&var) * nVers
        / nopct nocol norow missPrint
      ;
   Run ;

%mEnd _Report ;
```

C. Sections of Excel Cross-reference Table:

Year 1					Year 2				Year 3			
NAME	type	Vnum	LABEL		NAME	type	Vnum		NAME	Type	Vnum	LABEL
ID	1	2	Subject ID		ID	1	2		ID	1	2	Subject ID
NVERS	1	4	version number		NVERS	1	4		NVERS	1	4	version number
N1_DY	1	5	mother's date of birth (day)		N1_DY	1	5		N1_DY	1	5	mother's date of birth (day)
N1_MO	1	6	mother's date of birth (month)		N1_MO	1	6		N1_MO	1	6	mother's date of birth (month)
N1_YR	1	7	mother's date of birth (year)		N1_YR	1	7		N1_YR	1	7	mother's date of birth (year)
…					…				…			
…					…				…			
N56	1	166	ever had induced abortion		N56	1	166	R	N58	1	**247**	ever had induced abortion
					N56_NO	1	167	R	N58_NO	1	248	# of abortions
N57	1	167	# of abortions		N57	1	**168**	R	N59	1	249	ever had tubal pregnancy
					N57_NO	1	169	R	N59_NO	1	250	# of tubal pregnancies
									N60MO01	1	251	month pregnancy ended 01
									N60MO02	1	252	month pregnancy ended 02
									N60MO03	1	253	month pregnancy ended 03
…					…				…			
…					…				…			
N61D_D01	2	189	describe 4th birth defect (ab) 01		N61D_D01	2	**191**	R	N63D_D01	2	275	describe 4th birth defect (ab) 01
N61D_D02	2	190	describe 4th birth defect (ab) 02		N61D_D02	2	192	R	N63D_D02	2	276	describe 4th birth defect (ab) 02
N61D_D03	2	191	describe 4th birth defect (ab) 03		N61D_D03	2	193	R	N63D_D03	2	277	describe 4th birth defect (ab) 03
…					…				…			
…					…				…			
N65J	1	205	using sterilization		N65J	1	207	**D**				(expanded from n65j to n64h and n64i)
N65A	1	196	using condoms		N65A	1	198	C	N64A	1	**278**	ever used condoms
									…			
									…			
N65C	1	198	using shots		N65C	1	200	C	N64G	1	**284**	ever used shots
									N64H	1	**285**	ever used female sterilization
									N64I	1	**286**	ever used male sterilization
N65B	1	197	using diaphragm		N65B	1	199	C	N64J	1	**287**	ever used diaphragm
N62	1	192	ever used contraceptives		N62	1	194	C	N64_NO	1	**292**	resp never used birth control
									N65A	1	**293**	bc method #1 last used
									N65B	1	**294**	bc method #2 last used
									N65C	1	**295**	bc method #3 last used
N63	1	193	bc use when index baby conceived		N63	1	195	C	N66	1	296	stop using before pregnancy
N64_MO	1	194	mo last used bc before index baby		N64_MO	1	196	R	N67_MO	1	297	mo last used bc before index baby
N64_YR	1	195	yr last used bc before index baby		N64_YR	1	197	R	N67_YR	1	298	yr last used bc before index baby
…					…				…			
…					…				…			
N119MO3C	2	**466**	reason #3 took other med (mo 3)		N119MO3C	2	**468**	R	N121MO3C	2	**556**	reason #3 took other med (mo 3)
					N120_MO1	1	469	R	N122_MO1	1	557	any other tx (mo 1)
					N120TX1A	1	470	R	N122TX1A	1	558	any other tx #1 (mo 1)
…					…				…			
…					…				…			
					N121MO3B	2	488	R	N123MO3B	2	576	reason #2 had other tx (mo 3)
					N121MO3C	2	489	R	N123MO3C	2	577	reason #3 had other tx (mo 3)
N120	1	**467**	did mom work for pay (mos 1-3)	R	N122	1	**490**	R	N124	1	578	did mom work for pay (mos 1-3)
N121	1	468	mom's type of employment	R	N123	1	491	R	N125	1	579	mom's type of employment
…					…				…			
…					…				…			
N181_EXA	1	975	explain food #1 avoided (mos 1-3)	R	N183_EXA	1	998	R	N185_EXA	1	1086	explain food #1 avoided (mos 1-3)
N181_EXB	1	976	explain food #2 avoided (mos 1-3)	R	N183_EXB	1	999	R	N185_EXB	1	1087	explain food #2 avoided (mos 1-3)
N181_EXX	1	**977**	food avoided blankfilled	**C**	N183_EXC	1	**1000**	R	N185_EXC	1	1088	explain food #3 avoided (mos 1-3)
					N183_EXD	1	1001	R	N185_EXD	1	1089	explain food #4 avoided (mos 1-3)
…					…				…			
…					…				…			
N199D_04	2	1056	4th deformity 04	R	N201D_04	2	1082	R	N203D_04	2	1169	4th deformity 04
NINDEX	1	1057	index baby		NINDEX	1	1083		NINDEX	1	1170	index baby