

Paper SBC-116

An Innovative Approach to the “Other Specify” Recoding Procedure in Research Survey

Ruben Chiflikyan, Barbara Bibb, Mai Nguyen
RTI International, Research Triangle Park, NC

ABSTRACT

A common data quality control procedure in research survey is reviewing and recoding response data from the “Other Specify” questions. An “Other Specify” question is one which often follows a preceding question with a list of pre-coded responses. The “Other Specify” question provides a way to capture response data that is outside of the pre-coded list. Recoding the other specify fields consistently becomes difficult if there are multiple coders or if there is a massive amount of data to review. Using tools or computer applications to aid in the recoding process will reduce error and create more consistent results.

Our development team at RTI International has come up with a set of efficiently designed SAS® programs and a Microsoft Access application to semi-automate the extraction of the fields that need to be reviewed. This system of programs will programmatically manipulate the data and user inputs to facilitate the recoding process. The programs are designed to be highly re-useable for survey projects conducted by our company, especially in longitudinal surveys. Our paper will provide an overview of our process and then delve into the detailed technical design of the SAS programs. They were designed to extract data from the response dataset for recoding and to create an updated dataset with recoded values. The programs were written generically with macros using meta-data specified in an Excel spreadsheet to process the recoded data in the Microsoft Access database.

KEYWORDS: survey research, survey design, open-ended recoding, macros, macro functions, arrays.

GENERAL DESCRIPTION

Survey data is often used in social science fields and in research done by government agencies to determine the direction of government policy. Survey research data most closely resembles the answers to a multiple choice test. Most frequently, surveys present questions with a set of pre-selected answers. There are no correct or wrong answers. The respondent or subject answers each question based on their personal experiences or on their opinions or feelings. At times, the array of possible answers is large. The researcher might choose to present the most common answers as part of a list and then provide extra space for an “open ended” response. The researcher may be hoping for additional information or something that had not been considered in the survey design.

Open-ended responses to survey questions must be evaluated after data collection. To evaluate answers of this type, generally the answers are reviewed and coded manually by survey researchers. The reviewer determines if the open-ended answer is really one of the answers on the pre-coded list or, if the additional response should be captured and evaluated as if it were originally part of the possible set of choices. Such a review is done manually because much of this evaluation is subjective. Although it is subjective, it should be done as consistently as possible to get the most out of the data.

For example, suppose a relationship is being collected as the response to a question. The pre-coded list includes father, mother, son, daughter, sister, brother, wife, and husband. Suppose a respondent answered “half-brother” in the “Other Specify” field. The coder must determine if this response, for the purpose of this data collection, should be made into a “new” response or if it should be assigned the code for “brother.” Another respondent may have answered the question with the answer “friend.” Clearly, this response would be in a different category than the other responses presented. The researcher may decide to add a new code for a relationship of “friend.” Then, if a significant number of the responses to the question across the surveyed population were “friend,” the researcher could explore the data further.

In the rest of our presentation, we discuss the system we designed to assist researchers assign codes and the SAS coding required recoding open-ended responses once codes have been assigned.

RECODING PROCESS OVERVIEW

Our recoding process consists of 3 major steps:

- Extracting open-ended data from the respondent data for review and recoding. This is accomplished by a set of SAS programs. During extraction, the SAS programs use configurable meta-data input to extract open-ended response data, perform necessary transformations, and write the data into Microsoft Access tables that are more suitable for reviewers and coders.
- Reviewing and recoding data. This is a manual process performed by survey research staff using a small Access application. This step has two results: 1) a set of recoded values for open-ended responses, and 2) new response categories added by the coders.

- Creating final dataset. In this step, SAS programs transform and merge the original response dataset and the recoded data from the Access database into the final dataset.

DESCRIPTION OF THE PROGRAM

The following example will help describe our system. A hypothetical survey asked 5 respondents to answer the following 4 questions:

- (1) What is your favorite flower?
- (2) What is your favorite pet?
- (3) What sporting programs do you like to watch on TV?
- (4) How do you exercise?

Their responses were collected in the following variables: flower and flower_oth for (1), pet and pet_oth for (2), sp1, sp2, sp3, sp4, and sp_oth for (3), and ex1, ex2, ex3, ex4, ex5, ex_oth for (4). The variables “flower” and “pet” are type 1 variables with only a single response expected, while “sport” and “exercise” are type 2 variables where multiple responses are expected. The zrid is the identifier. The gender variable contains information that will remain unchanged during all the recoding procedure. Below, we present the sample of the SAS dataset:

Obs	zrid	gender	flower	flower_oth	pet	pet_oth	sp1	sp2	sp3
1	ID001	1	1		4	Text	4	.	.
2	ID002	1	3	Text	2		2	4	.
3	ID003	2	3	Orchid	4	Fish	3	.	.
4	ID004	2	3	Margarita	4	Parakeet	4	3	.
5	ID005	1	2		-1		1	3	.

Obs	sp4	sp_oth	ex1	ex2	ex3	ex4	ex5	ex_oth
1	.	Boxing, Wrestling	1	5	.	.	.	Biking
2	.	Figure skating	4	2	.	.	.	Canoeing,Skiing
3	.		1	3	5	.	.	Text
4	.	Text	1	3	5	4	.	Biking,Hiking,Skiing
5	.		-2	.	.	.	.	

Figure 1. The original SAS dataset to be recoded

The following initial formats were used for all 5 variables above:

```
Proc format;
  Value gender 1 = 'Male' 2 = 'Female';
  Value flower 1 = 'Rose' 2 = 'Tulip' 3 = 'Other flower';
  Value pet 1 = 'Dog' 2 = 'Cat' 3 = 'Hamster 4 = 'Other pet';
  Value sport 1 = 'Football' 2 = 'Basketball' 3 = 'Baseball' 4 = 'other sport';
  Value exercise 1 = 'Jogging' 2 = 'Walking' 3 = 'Swimming' 4 = 'Dancing'
    5 = 'Other exercise';
run;
```

Consider the data in the Figure 1. The respondent with zrid='ID001' has chosen the value 1 for the flower variable. This is the 'Rose' response. The flower_oth variable is empty because only one choice is allowed. The variable “pet” (type=1) has value of 4, which means that the pet_oth variable needs to have an answer. In this case, an answer was keyed as the response and it can not be re-coded. The sport variables (type=2) are presented with sp1=4. The open ended sp_oth variable contains “boxing” and “wrestling.” These responses will be transformed into new categories. The exercise variables (type=2) have ex1=1 and ex2=5. The response 5 is followed by the open ended answer in ex_oth field, “biking.” This is also not one of the pre-coded responses, so coders may add a new category.

After the recoding, the original formats may need to be changed to:

```
Proc format;
```

```

Value gender 1='Male' 2='Female';
Value flower 1='Rose' 2='Tulip' 3='Other flower' 4='Orchid' 5='Margarita';
Value pet 1='Dog' 2='Cat' 3='Hamster' 4='Other pet' 5='Parakeet' 6='Fish';
Value sport 1='Football' 2='Basketball' 3='Baseball' 4='Other sport'
5='Figure skating' 6='Boxing' 7='Wrestling';
Value exercise 1='Jogging' 2='Walking' 3='Swimming' 4='Dancing'
5='Other exercise' 6='Biking' 7='Canoeing' 8='Skiing' 9='Hiking';
run;

```

Variables that will not be modified are the zrid identifier and the gender variable.

GENERAL PROGRAM STRUCTURE

The program is intended to work within various environments. We are using the following folder structure.

```

%let bsc=0:\SESUG_2008 – main folder;
%let dbs=&bsc.\Access_Tables – all information related to Access tables and database;
%let exc= &bsc.\Excel_Tables – all information related to Excel tables;
%let errmsgs= &bsc.\Error_messages –all error messages repository;
%let err(1..11)=&bsc.\&errmsgs.\Test(1..11).txt – defining 11 text files where various error messages
are written to;
%let excin(1..5)=&bsc.\Excel_Tables\excel_(1..5).xls – for 5 input/output Excel spreadsheets used
during the data processing.

```

All variable attributes that are created and used throughout this program are defined globally.

EXTRACT AND TRANSFORM DATA FOR RECODING

The data extraction and transformation process requires reading configurable meta-data from an Excel spreadsheet, performing transformations using transient spreadsheets, and writing data into Access tables. The detailed description of these Excel spreadsheets and Access tables are in the Appendix.

The program meta-data inputs are provided in the "&excin1" Excel spreadsheet. They include all variable names of both types and array ranges associated with type 2 variables. The structure of this Excel table is shown in Figure 2:

1. The *varnam_old* variable contains variable names for both variables of type 1 and 2 and identifier(s).
2. The *type* variable takes value 1, 2, or 0 for variables of the type=1, type=2, and identifier(s) respectively.
3. The *array_num_old* variable shows the number of array elements for each variable of type 2.
4. The *varnam_old_root* variable is an internal variable.

For our example in Figure 1, the "&excin1" Excel spreadsheet has the following structure:

Obs	varnam_old	type	array_num_old	varnam_old_root
1	zrid	0		
2	flower	.		
3	flower_oth	1		flower
4	pet	.		
5	pet_oth	1		pet
6	sp1	.		
7	sp2	.		
8	sp3	.		
9	sp4	.		
10	sp_oth	2	4	sp
11	ex1	.		
12	ex2	.		

Obs	varnam_old	type	array_num_old	varnam_old_root
13	ex3	.		
14	ex4	.		
15	ex5	.		
16	ex_oth	2	5	ex

Figure 2. The content of the Excel 1 spreadsheet.

To minimize potential errors during data processing, we created 10 checking procedures throughout the program to catch all types of errors, including misspellings and structure inconsistencies. If an error is caught, the program is automatically aborted. The program then generates a detailed error message and saves it in the %errmsgs folder. Errors must be corrected before restarting the program. The validation procedures are as follows:

- Check whether the structure of data in the Excel 1 spreadsheet is as defined.
- Check whether all 4 variables in the Excel 1 table have valid SAS variable names. The *varnam_old*, *type*, *array_num_old*, and *varnam_old_root* variables are dynamically assigned to the corresponding *&varnamold*, *&types*, *&arraynumold*, and *&varnamoldroot* macro variables only if they have valid SAS variable names. The order of variables in the original Excel spreadsheet is important for this assignment.
- Check whether both *&types* and *&arraynumold* variables are numeric.
- Check for duplicates in the values of the *&varnamold* variable.
- Check whether values of the *&varnamold* variable are in the original response dataset.
- Check that the values of *&arraynumold* are consistent with the array size of type 2 variables.

To process data generically, original survey variables are renamed to variable names that can be indexed in our SAS macros. Variables of type=1 have new names starting with Q, while those for type=2 start with C. The subsequent 2-digit number shows variable number. Arrayed variables are identified with their array position in the variable suffix. The renamed variables are shown in the Excel 2 spreadsheet in Figure 3 below.

Obs	varnam_old	varnam_old_root	array_num_old	varnam_new	varnam_new_root
1	zrid			id	
2	flower			Q01	
3	flower_oth	flower		Q01_oth	Q01
4	pet			Q02	
5	Pet_oth	pet		Q02_oth	Q02
6	sp1			C01_1	
7	sp2			C01_2	
8	sp3			C01_3	
9	sp4			C01_4	
10	sp_oth	sp	4	C01_4_oth	C01
11	ex1			C02_1	
12	ex2			C02_2	
13	ex3			C02_3	
14	ex4			C02_4	
15	ex5			C02_5	
16	ex_oth	ex	5	C02_5_oth	C02

Figure 3. The original and renamed variables.

The original dataset is divided into the following 3 datasets:

1. the `opn.rest_old` dataset, which contains all variables that were not recoded;
2. the `t1_old` dataset, which contains type 1 variables only; and
3. the `t2_old` dataset, which contains type 2 variables only.

Macro variables `&vot1`, `&vnt1`, and `&ren_t1` contain the list of old variable names, new variable names, and the total number of variables for type 1 variables. Similarly, macro variables `&vot2`, `&vnt2`, and `&ren_t2` contain the list of old variable names, new variable names, and total number of variables for type 2 variables. Using the `%genrnm` macro below, we create datasets `opn.t1_new` and `opn.t2_new` from the `t1_old` and `t2_old` datasets respectively. The `opn.t1_new` contains the same data as `t1_old` but has the new variable names, as does the `opn.t2_new` dataset.

```
%macro genrnm;
%do L=1 %to 2;
data opn.t&L._new; set t&L._old;
rename %do I=1 %to &&ren_t&L; %scan(&&vot&L,&I,'*')= %scan(&&vnt&L,&I,'*') %end;
;
&idnew=trim(left(&id));
run; %end; %mend genrnm1;
%genrnm;
```

Further, we create `opn.zero`, which is the basic dataset that contains the names of all open-ended variables of both types.

Obs	zrid	flower_oth	pet_oth	sp_oth	ex_oth
1	ID001		Text	Boxing, wrestling	Biking
2	ID002	Text		Figure skating	Canoeing,Skiing
3	ID003	Orchid	Fish		Text
4	ID004	Margarita	Parakeet	Text	Biking,Hiking,Skiing
5	ID005				

Figure 4. Opn.zero dataset containing only open-ended variables.

We perform straightforward intermediate transformations using the `%create` macro below to create a dataset in a format coders find more convenient when recoding the open-ended responses.

```
%macro create;
* creating ds1--ds&nobs for each value of varname and y1--y&nobs datasets;
data %do i=1 %to &nobs;
ds&i (keep=&id %scan(&allvars,&i,' ') y&i
rename=(yield=&varname %scan(&allvars,&i,' ')=&text)) %end; ;
* defining character vars;
length %do i=1 %to &nobs; y&i %end; $&lngvar.;
length %do i=1 %to &nobs; %scan(&allvars,&i,' ') %end; $&lngval.;
set opn.zero;
&text=' ';
* creating variables with text values created with vname functions;
%do i=1 %to &nobs; y&i=vname(%scan(&allvars,&i,' '));%end;
run;
* sorting all datasets by id for subsequent setting;
%do i=1 %to &nobs; proc sort data=ds&i; by &id; run;%end;
* setting all datasets together and deleting obs with empty text values;
data temp;
set %do i=1 %to &nobs; ds&i %end; ;
by &id;
if &text^=' '; label &text="&text";
```

```

format text $&lngval.;
run;
data temp1; set temp; run;
proc sort data=temp; by &varname; run;
data labels;
length &varname $&lngvar;
set all_names;
if _N_>=2; *to exclude zrid;
&varname=allnames;
keep &varname &label;
run;

proc sort data=labels; by &varname; run;

data temp;
length &newcode $8 &text $&lngval.;
merge temp(in=in1) labels(in=in2);
by &varname;
if in1 and in2;
&newcode=' ';
run;

* adding root info. This is the request from coder;
data temp;
merge temp(in=in1) roots2(in=in2);
by &varname;
if in1 and in2;
run;

* sorting for visual purposes only;
proc sort data=temp;
by &varname &id;
run;

* leaving only variables requested by reviewer;
proc sql;
create table access.&outtbl
as select &id, &root, &varname, &text, &label, &newcode
from temp;
quit;
%mend create;
%create;

```

As a result of this operation, we generate the Access table as shown in Figure 5, which is sent to the coder. When first created, the *newcode* column will be empty. The coder may assign old and new categories to the *newcode* variable using information in the *text* variable where appropriate. The completed table is returned as input for creating the final dataset.

Obs	zrid	root	varname	text	label	newcode
1	ID001	ex	ex_oth	Biking	ex_oth	6
2	ID002	ex	ex_oth	Canoeing,Skiing	ex_oth	7,8
3	ID003	ex	ex_oth	Text	ex_oth	
4	ID004	ex	ex_oth	Biking,Hiking,Skiing	ex_oth	6,8,9
5	ID002	flower	flower_oth	Text	flower_oth	
6	ID003	flower	flower_oth	Orchid	flower_oth	4
7	ID004	flower	flower_oth	Margarita	flower_oth	5
8	ID001	pet	pet_oth	Text	pet_oth	
9	ID003	pet	pet_oth	Fish	pet_oth	6

Obs	zrid	root	varname	text	label	newcode
10	ID004	pet	pet_oth	Parakeet	pet_oth	5
11	ID001	sp	sp_oth	Boxing, Wrestling	sp_oth	6,7
12	ID002	sp	sp_oth	Figure skating	sp_oth	5
13	ID004	sp	sp_oth	Text	sp_oth	

Figure 5. New numerical categories are added.

CREATING FINAL DATASET

We designed the checks to determine if all variable names sent to the reviewer come back unchanged and if the *newcode* variable contains only numbers delimited by commas. After the data is validated, we are ready to transform and merge the recoded data with the original data and create the final dataset.

Because new categories may be added to the original list of response choices, the program must increase the array range accordingly. After a series of simple transformations, the Excel 3 table below is generated. Note that the *array_num* variable has been updated to the new array ranges.

Obs	varnam_old	type	array_num	varnam_old_root	varnam_new	varnam_new_root
1	sp1	2			C01_1	
2	sp2	2			C01_2	
3	sp3	2			C01_3	
4	sp4	2			C01_4	
5	sp5	2			C01_5	
6	sp6	2			C01_6	
7	sp7	2			C01_7	
8	sp8	2			C01_8	
9	sp_oth	2	8	sp	C01_8_oth	C01
10	ex1	2			C02_1	
11	ex10	2			C02_10	
12	ex_oth	2	10	ex	C02_10_oth	C02
13	ex2	2			C02_2	
14	ex3	2			C02_3	
15	ex4	2			C02_4	
16	ex5	2			C02_5	
17	ex6	2			C02_6	
18	ex7	2			C02_7	
19	ex8	2			C02_8	
20	ex9	2			C02_9	
21	flower	1			Q01	
22	flower_oth	1		flower	Q01_oth	Q01

Obs	varnam_old	type	array_num	varnam_old_root	varnam_new	varnam_new_root
23	pet	1			Q02	
24	pet_oth	1		pet	Q02_oth	Q02
25	zrid	0			id	

Figure 6. Excel 3 Spreadsheet with the increased arrays' ranges.

Next, we use the macro *type1*, shown below, and the updated variables in Figure 6 to update the *opn.t1_new* dataset with the recoded data.

```
%macro type1;
%global nn ids val varc;
%if &sel1=1 %then
%do;
data tbl_t1_m(keep=&idnew varnam text value var_cor);
length &idnew $8 varnam $14 text $70 value 4. var_cor $6;
set tbl_t1;
&idnew=trim(left(&id));
varnam=trim(left(&varnamnew));
value=trim(left(&newcode));
var_cor=trim(left(&varnamnewroot));
run;

proc sort data=tbl_t1_m;
by &idnew;
run;

proc sort data=opn.t1_new;
by &idnew;
run;

data unif1;
merge opn.t1_new(in=in1) tbl_t1_m(in=in2);
by &idnew;
if in2; * keeping only cases that come back from the coder;
run;

proc sql noprint;
{Below you can see the description of steps
{* the list of all ids in the transformed table.
This # can be larger than the # of ids in original table; }

{* creating macro variable with the roots of the variable names having other options;}
{* creating macro variable with the names of all new values added by coder; }
{* # of observations in your transformed table;}
quit;

%end;
%mend type1;
%type1;

* Note: this creates the recoded version of the original dataset with the
NEW variable names created by us earlier;
%macro trans7;
data t1_new_clean;
set opn.t1_new;
%if &sel1=1 %then
%do;
%do I=1 %to &nn;
%*if &I=1 %then %let els=;
%*else %let els=else;
if &idnew="%scan(&ids,&I, ' ')" then
do;
```

```

                %scan(&varc,&I,'*')=%scan(&val,&I,'*');
                %scan(&varc,&I,'*')_oth=' ';
            end;
        %end;
    %end;
run;
%mend trans7;
%trans7;

```

The updated opn.t1_new dataset is shown in Figure 7.

Obs	zrid	Q01	Q01_oth	Q02	Q02_oth
1	ID001	1		4	Text
2	ID002	3	ext	2	
3	ID003	4		6	
4	ID004	5		5	
5	ID005	2		-1	

Figure 7. The recoded variables of type 1.

Variables of type 2 require additional processing. Type 2 variable arrays may need to be increased and checked for duplicate category responses. The macro *%type2* and *%cleaning* are used to update the opn.t2_new dataset. The updated opn.t2_new dataset is shown in Figure 8.

```

%macro cleaning;
data t2_new_ext;
set t2_new_ext;
%do J=1 %to &nmn;
    %let old=%scan(&oldnums,&J,'*');
    %let new=%scan(&newnums,&J,'*');
    %let root=%scan(&roots,&J,'*');
    %do I=1 %to &new;
        if &root._&I=&old then &root._&I=&new;
    %end; %end; run;

data t2_new_clean;
set t2_new_ext;
%if &sel2=1 %then
%do;
    %do M=1 %to &nnn;
        %let ttr=%scan(&var_arr,&M,'*');
        %let arr&ttr=;
        %do L=1 %to %scan(&num_arr,&M,'*');
            %let arr&ttr=&&arr&ttr &ttr._&L;
        %end;
        array &ttr.arr(*) &&arr&ttr;
    %end;
* if conditions for cleaning original dataset;
%do K=1 %to &nn;
    if %scan(&val,&K,'*') ^=. and &idnew="%scan(&ids,&K,'*')" then
        do;
            %let qt=%scan(&varc,&K,'*');
            %let ar=%scan(&arn,&K,'*');
            *Note: this is to delete only the first time the array element with the maximum value. This correctly treats
            cases in which 2 observations have the same id and same variable name. This situation can encounter
            after creating several observations from the single one, when instead of one value you can have several
            values put by the coder;
            maxv=max(of &qt.arr(*)); *maximum value found for a given array;
            maxw=%scan(&arn,&K,'*'); * real maximum value for the same array;
            do I=1 to &ar;
                if &qt.arr(I)=maxv and &qt.arr(I)=maxw then &qt.arr(I)=.;
            end;
        end;
    %end;
%end;

```

```

end;
do I=1 to &ar;
  if &qt.arr(I)=%scan(&val,&K,'*') then do; leave; end;
  if &qt.arr(I)=. then do; &qt.arr(I)=%scan(&val,&K,'*'); leave; end;
end;
&qt._&ar._oth=' ' ; * deleting the other fields content;
end;
%end;
drop maxv maxw I;
%end;
run;
%mend cleaning;
%cleaning;

```

Obs	ID	C02_1	C02_2	C02_3	C02_4	C02_5	C02_6	C02_7	C02_8	C02_9	C02_10	C02_10_oth
1	ID001	1	6	.	.	.	.	.	.	.	.	.
2	ID002	4	2	7	8	.	.	.	.	.	.	.
3	ID003	1	3	5	.	.	.	.	.	.	.	Text
4	ID004	1	3	6	4	8	9	.	.	.	.	.
5	ID005	-2	.	.	.	.	.	.	.	.	.	.

Obs	C01_1	C01_2	C01_3	C01_4	C01_5	C01_6	C01_7	C01_8	C01_8_oth
1	6	7	.	.	.	.	.	.	.
2	2	5	.	.	.	.	.	.	.
3	3	.	.	.	.	.	.	.	.
4	4	3	.	.	.	.	.	.	Text
5	1	3	.	.	.	.	.	.	.

Figure 8. The recoded and renamed variables of type 2.

Finally, we rename the variables back to their original names by using the macro *%rcdback* below:

```

%macro rcdback;
%local I J;
data t2_new_clean;
set t2_new_clean;

%do J=1 %to &nn4;
  %do I=1 %to %scan(&arrnew4,&J,'*');
    if %scan(&vnewrt4,&J,'*')_&I=%scan(&arrnew4,&J,'*') then
      %scan(&vnewrt4,&J,'*')_&I=%scan(&ar4,&J,'*');
    %end;
  %end;
run;
%mend rcdback;
%rcdback;

```

The final recoded dataset with the original variable names is shown in Figure 9:

Obs	zrid	gender	flower	flower_oth	pet	pet_oth	sp1	sp2	sp3	sp4	sp5	sp6	sp7	sp_oth
1	ID001	1	1		4	Text	6	7	.	.	.	.	.	.
2	ID002	1	3	Text	2		2	5	.	.	.	.	.	.

Obs	zrid	gender	flower	flower_oth	pet	pet_oth	sp1	sp2	sp3	sp4	sp5	sp6	sp7	sp_oth
3	ID003	2	4		6		3	.	.	.	.	.	.	
4	ID004	2	5		5		4	3	.	.	.	.	.	Text
5	ID005	1	2		-1		1	3	.	.	.	.	.	

Obs	ex1	ex2	ex3	ex4	ex5	ex6	ex7	ex8	ex9	ex_oth
1	1	6	.	.	.	.	.	.	.	
2	4	2	7	8	.	.	.	.	.	
3	1	3	5	.	.	.	.	.	.	Text
4	1	3	6	4	8	9	.	.	.	
5	-2	.	.	.	.	.	.	.	.	

Figure 9. The final recoded dataset with type 1 and type 2 original variable names.

CONCLUSION

We developed a system of efficiently designed SAS programs and a Microsoft Access application to semi-automate extracting open-ended survey responses that need to be recoded. This system will programmatically manipulate the data and user inputs to facilitate the recoding process. The programs are designed to be highly re-useable for current and future survey projects conducted by our company. They are designed to extract data from the response dataset for recoding, and to create an updated dataset with recoded values. The programs were written generically with macros using metadata specified in Excel spreadsheets to process the recoded data in the Microsoft Access database.

ACKNOWLEDGEMENTS

The authors are grateful to Wafa Handley and Craig Hollingsworth at RTI International for many valuable remarks and suggestions.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Ruben Chiflikyan
RTI International
3040 Cornwallis Road, PO Box 12194
Research Triangle Park, NC, 27709-2194
Work Phone: (919) 541-6064
Fax: (919) 541-6178
E-mail: rchiflikyan@rti.org
Web: <http://www.rti.org>

Barbara S. Bibb
RTI International
3040 Cornwallis Road, PO Box 12194
Research Triangle Park, NC, 27709-2194
Work Phone: (919) 541-7413
Fax: (919) 541-6178
E-mail: bibb@rti.org
Web: <http://www.rti.org>

Mai Nguyen
RTI International
3040 Cornwallis Road, PO Box 12194
Research Triangle Park, NC, 27709-2194
Work Phone: (919) 541-8757
Fax: (919) 541-6178
E-mail: mnguyen@rti.org
Web: <http://www.rti.org>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

APPENDIX

Here is the description of all Excel spreadsheet and Access tables used by the SAS programs.

1. The Excel 1 spreadsheet is imported from Excel to SAS with the following 4 variables: *varnam_old*, *type*, *array_num_old*, and *varnam_old_root* variables.
 - a) The *varnam_old* variable contains the original type 1 and type 2 variables and identifiers.
 - b) The *type* variable has values 0, 1, and 2. The value of 0 is assigned to observations containing identifier(s) names. The value of 1 is assigned to observations containing type 1 variable names for textual input. The corresponding observations, which contain type 1 variable names for numeric input, are left empty. The value of 2 is assigned to observations containing type 2 variable names for textual input. The corresponding observations, which contain type 2 variable names for numeric input, are left empty.
 - c) The *array_num_old* variable contains array elements for variables of type 2 assigned to observations with variable names for textual input.
 - d) The *varnam_old_root* variable contains roots for variables of type 1 and type 2 assigned to observations with variables names for textual input.
2. The Excel 2 spreadsheet is exported from SAS to Excel with the 4 variables mentioned in the description above, along with *varnam_new*, *varnam_new_root*, and *array_num* variables.
 - a) The *varnam_new* variable contains the renamed variable names, including identifiers.
 - b) The *varnam_new_root* variable contains the roots of variables' new names extracted from the *varnam_new* variable values. These values are assigned to observations with variable names for textual input.
 - c)
3. The Access 1 table is exported from SAS to Access and contains the following variables: *zrid*, *root*, *varname*, *text*, *label*, and *newcode* variables. This table contains the empty *newcode* variable to be filled with numeric values by the coder based on information in the *text* variable.
 - a) The *zrid* variable contains the actual identifiers values.
 - b) The *root* variable contains the roots of the old variable names.
 - c) The *varname* variable contains old variable names of type 2 for textual input.
 - d) The *label* variable contains the actual labels.
 - e) The *text* variable contains the textual response from the open-ended question.
 - f) The *newcode* variable is a placeholder for the coder to assign recoded values.
4. The Access 2 table is imported back from Access to SAS and contains the same set of variables as in 3. The only distinction is that the *newcode* variable contains codes created by the coder based on information from other variables and related specifications. Some fields can be left empty, which means that the textual information contained in the *text* variable did not contain enough information.
5. The Excel 3 spreadsheet is exported from SAS to Excel with the same set of variables as in Excel 2 plus a new variable *array_num*. On the basis of analysis of the Access 2 table, the program creates, if necessary, new spots or array elements for each type 2 variable. Note that *array_num* variable now contains the modified value for the number of array elements.
6. The Excel 4 spreadsheet is imported from Excel to SAS with all the variables in Excel 3 along with the new *flag_del* variable. It equals 1 in those observations that is the last element of the type 2 variable arrays.