

A Simple SAS® Utility to Dynamically Create Variable Names and Recode the Associated Values

Chuchun Chien, Wafa Handley, Barbara Felts, Yun Mai
RTI International, RTP, NC

ABSTRACT

With the advent of Computer Assisted Interviewing (CAI) as a primary venue for survey data collection, most commercially available CAI software provides developers with utilities or Application Programming Interface (API) tools. Developers use these tools to produce a SAS® data structure document which contains a list of SAS compliant variable names and data types for the collected data. In order to facilitate data analysis for some of the data items collected in a CAI survey, such as “Answer All That Apply” responses, and to store and therefore preserve the original data input during data collection, we have developed a simple but versatile utility. This utility uses the list of SAS variable names and their data types to dynamically create variable names that are analogous to the original list of variables. We have used SAS arrays, macro variables, and some creative looping techniques to recode the values of these variables based on the response item chosen, rather than the order in which they were entered during data collection. In our paper we will: (1) demonstrate the need for such utility, (2) provide program code for creating the variable names, (3) verify the uniqueness of the variable names in order to avoid overriding of values that might result from duplicate variable names, and (4) reveal our creative use of looping through arrays to recode these variables.

INTRODUCTION

“Answer All That Apply” questions are a staple of survey instruments and Computer Assisted Interviewing (CAI). Questions where multiple categories of responses are applicable are often used to collect data on a wide range of subject matter ranging from demographic information on race and ethnic origin, to the types of music respondents might listen to. If researchers are interested in the types of music a segment of the population prefers, then the possible options for responses could be: (1) Classical, (2) Country, (3) Hip-Hop, (4) Jazz, (5) Pop/Rock, 6) Soul R&B.

PROBLEM

During CAI data collection, the values for the types of music preferred are stored in the order entered by respondents. A viable snapshot of the data collected could look like the table below, representing three respondents (R1-R3) and the six response variables (Ans1-Ans6) shown with their respective values in Figure 1.

	Ans1	Ans2	Ans3	Ans4	Ans5	Ans6
R1	3	4	-	-	-	-
R2	1	2	4	-	-	-
R3	2	6	3	5	4	1

Figure 1. Snapshot of Data Collection.

In this example, R1's first choice is Hip-Hop, followed by Jazz. R2's preferences are Classical, followed by Country and Jazz in that order. R3 listens to all types of music with Country being the most preferred and Classical the least preferred category. Note that the values stored in the variable Ans1 would always be the first answer that respondents have keyed in with values ranging from one to six. Similarly, the values stored in Ans2- Ans6 variables will always be the second through sixth answers the respondents have keyed in. Analysts, however, are interested in the distribution of respondents listening to the different types of music in the response categories based on the target population.

SOLUTION

The goal of the programmers is to create data that corresponds to the keyed responses as illustrated below in Figure 2. In order to preserve the original values entered by respondents and facilitate data analysis, we dynamically create backup variables from the SAS data structure document and recode the instrument variables with the appropriate values.

	Ans1	Ans2	Ans3	Ans4	Ans5	Ans6
R1	.	.	3	4	.	.
R2	1	2	.	4		
R3	1	2	3	4	5	6

Figure 2. Data Required For Analysis.

APPROACH

Our approach was to:

- Create variable names based on the names provided by the CAI software during data collection.
- Loop through the new variable names to check for duplicate values.
- Recode the keyed values of "All That Apply" variables with the appropriate value, corresponding to the variable name/number.

The flow chart in Figure 3 illustrates the various processes and steps taken to accomplish our goal.

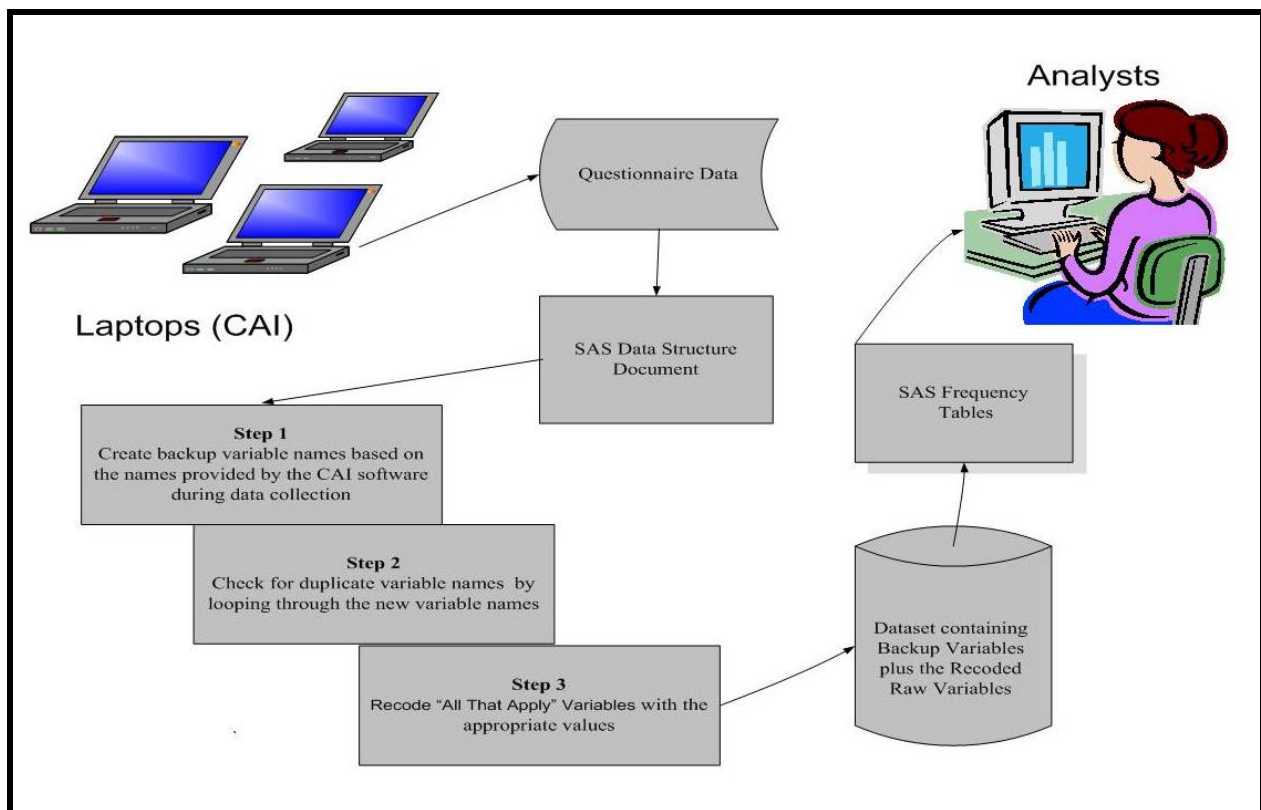


Figure 3. Flow Chart of Approach.

CREATE BACKUP VARIABLE NAMES

To dynamically create the new variable names based on the questionnaire names, we start with the SAS data structure document generated by the CAI software. The instrument's variable names are read from the SAS document to a temporary SAS data set. As we work with thousands of variables in each survey cycle we manage the processing by creating text files containing SAS statements or declarations (such as Arrays) in addition to text files containing Length statements with the applicable block of variables. These text files are used as SAS Include Files in subsequent programs.

The following code snippet demonstrates our first step:

```
Filename caitext "\\filedirectory\caistruct.sas";
Filename zvarrr "z\\filedirectory\zcai.txt";
Filename rvarrr "r\\filedirectory\rcaiarr.txt";

DATA getvars;
  infile caitext trunccover;
  input @1 rawvar $10.;
  numrec = _N_;
  run;
/* rewrite the variables in a block format to use as include files */
DATA null;
  set getvars;
  file rvarrr ls=80;
  put rawvar $11. @;
run;

DATA zvars;
length zrawvar $10 /*fstchar $1 remchar $ 6 */;
  set getvars;
  zrawvar = "z"||rawvar;
  file zvarrr ls=80;
  put zrawvar $11. @;
/* rewrite the variables in an array format*/
data null;
  set getvars
  end = lastrec;
  if _n_ = 1 then
    do;
      file rvarrr ls=80;
      put "array rawvars {*} ";
      put rawvar @;
    end;
  else if lastrec= 1 then
    do;
      file rvarrr ls=80;
      put rawvar $11. @;
      put; put ";";
    end;
  else do;
    file rvarrr ls=80;
    put rawvar $11. @;
  end;
end;
```

CHECK FOR DUPLICATE VARIABLE NAMES

Since we get a multitude of variables, with existing questions in addition to newly added questions, and hence new variables in each survey cycle, the potential for dynamically creating a duplicate name is significant. In order to eliminate duplicate variable names, we developed a looping mechanism that goes through an array and compares each element with the preceding elements in order to identify duplicate values as shown in the code below.

```
Filename zvar text "\\filedirectory\zcaiarr.txt";
Filename rvar arr "\\filedirectory\rcaiarr.txt";

DATA dupvars;

%include zvar text;
%include rvar arr;

%getvarcnt;

length val1-val&count zname $10;

array zvals{*} val1-val&count;

do i=1 to dim(zvars);
    call vname (zvars{i},zname);
    zvals{i} = zname;
end;
arrvalue = dim(zvals)-1;
do i= 1 to dim(zvals);
    if i < dim(zvals) then
    do;
        do k=1 to arrvalue;
            if zvals{i} = zvals{i+k} then
            do;
                put zvals{i}= zvals{i+k}= i= k=;
                dupvar = zvals{i+k};
                output dupvars;
            end;
        end; /* k loop */
        arrvalue = arrvalue - 1;
    end; /* i < dim(zvals) */
end; /* i loop */
```

Note the following SAS statements that are used in the above code:

- The Macro call %getvarcnt
- The Array Declaration (array zvals{*} val1-val&count;)
- The Length statement (length val1-val&count zname \$10)

In order to dynamically create dimensions for the array size and the length statement, we created a macro that generates the number of variables in the text file which was created in the preceding step. We used the generated number as the macro variable (**&count**) to define the size of the array, and the number of variables in the length statement, thereby creating dynamic dimensions for each cycle of the survey.

```
%macro getvarcnt;
data getcnt;
    infile "zcaiarr.txt" truncover end=last;
    input @1 inline $char50.;
    count = _n_;
    if last then do;
        put count=;
        call symput('count',trim(left(count)));
    end;
run;
%mend getvarcnt;
```

RECODE "ALL THAT APPLY" VARIABLES

At this step we store the initial values in the backup variables and reinitialize the original variables to missing values, using a DO (index=) TO (expression) loop. Once the values are stored and the original variables are reinitialized, a second DO (index=) TO (expression) loop is used to assign the appropriate response using the values from the backup variables as the indices for the array variables. Since the value from the backup variable is a number from one to six in our example, this value is used to assign the (j)th number to the (j)th element. If j = 4 then allap {4} is assigned a value of 4. Missing values such as "refusal" and "don't know" response options are also assigned in this step, but removed from Figure 3 for brevity. The source code is shown below:

```
array allap{*} ALLAPPL1 - ALLAPPL6;
array zallap{*} zALLAPPL1 - zALLAPPL6;

/* recode the arrays so that each element will have its corresponding value*/
/* instrument assigns values in keying order*/

i=0;
j=0;
do i=1 to dim(allap);
  zallap{i} = allap{i};
  allap{i} = .;
end;
do i=1 to dim(allap);
  if zallap{i} ne . then
    do;
      j = zallap{i};
      allap{j} = i;
    end;
end;
end;
```

RESULTS

The Proc Summary is used to produce the distribution table, the results of which are shown in Figure 4. The table shows the backup variables with the initial responses and the instrument variables with the recoded values. Dots indicate that response category was not chosen. For example, in line eight the variable ALLAPPL1 is replaced with a missing value, and variable ALLAPPL2 has a value of two, because the first keyed response had a value of two; variable ALLAPPL3 is replaced with the value of three because the second keyed response had a value of three in ALLAPPL2. In lines fifteen and sixteen variables ALLAPPL1-ALLAPPL5 are recoded to missing and variable ALLAPPL6 is recoded to 6, since that was the only response which was chosen.

Obs	ALLAPPL1	ALLAPPL2	ALLAPPL3	ALLAPPL4	ALLAPPL5	ALLAPPL6	zALLAPP1	zALLAPP2	zALLAPP3	zALLAPP4	zALLAPP5	zALLAPP6
1	1	2	3	4	5	.	1	2	3	4	5	.
2	1	2	.	.	.	.	1	2	.	.	.	.
3	1	.	3	.	.	.	1	3	.	.	.	.
4	1	.	.	.	5	.	1	5	.	.	.	.
5	1	.	.	.	.	6	1	6	.	.	.	.
6	1	.	.	.	.	6	1	6	.	.	.	.
7	1	.	.	.	.	.	1	.	.	.	.	.
8	.	2	3	.	.	.	2	3	.	.	.	.
9	.	2	.	.	.	6	2	6	.	.	.	.
10	.	2	.	.	.	.	2	.	.	.	.	.
11	.	.	3	.	.	6	6	3	.	.	.	.
12	.	.	3	.	.	.	3	.	.	.	.	.
13	.	.	.	4	5	.	4	5	.	.	.	.
14	.	.	.	.	5	.	5	.	.	.	.	.
15	.	.	.	.	.	6	6	.	.	.	.	.
16	.	.	.	.	.	6	6	.	.	.	.	.
17	.	.	.	.	.	.	.	.	.	.	.	.

Figure 4. Recoding Results.

SUMMARY

The steps described provide the desired outcome using a few lines of code and the creative use of SAS loops, macro variables, and include files. For each survey cycle, we start with thousands of SAS compliant variables, create secondary variable names as needed for storing initial data, and verify the new variable names for duplicate names to avoid overriding of values. Finally, we recode the questionnaire raw variables, as required, with minimal or no change in code for variations in the number or content of the questions in the instrument that might occur in each cycle of the survey.

ACKNOWLEDGMENT

The authors acknowledge the work of Barbara Bibb, at RTI International, on a variation of this approach to address a similar issue. The alternate process utilized the SAS data structure document to generate SAS Rename statements, and to convert the "Answer All That Apply" variables to a new set of "Toggle" variables used to determine whether a response option was chosen.

The authors also acknowledge Gary Franceschini for editing this paper.

CONTACT INFORMATION

Chuchun Chien, cchien@rti.org, 919-485-5552

Wafa Handley, handley@rti.org, 919-541-6066

Barbara Felts, bfelts@rti.org, (919) 541-6938

Yun Mai, yunmai@rti.org, 404-592-9420

Each author can be reached at the mailing address:

RTI International
P.O. Box 12194
Research Triangle Park, NC 27709-2194
www.rti.org

TRADEMARK NOTICE

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 registered trademarks or trademarks of their respective companies.