Recoding “ALL THAT APPLY” Variables from Handhelds and Portable Computers

Wafa Handley, Barbara Bibb, Lilia Filippenko, Jay Levinsohn, Donna Medeiros
RTI International, RTP, NC

ABSTRACT

How do you turn “ALL THAT APPLY” (ATA) questions into useful data structures? 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 and other tools to interface with the software. Developers use these tools to produce data structure documents which contain lists of variable names and data types for the collected data. The data structure is based on the final data format of interest such as SAS®, SPSS, Oracle or other databases. The tools enable developers to access the CAI software database to collect the necessary information to support their tasks. With the introduction of handheld and other mobile devices as a convenient mode of data collection, the range of methods and tools to collect data is quite broad. As yet there is no dominant mobile tool such as the Blaise questionnaire development software, so converting handheld database structures or handheld storage conventions for certain types of data for SAS® typically requires new programming.

Prior to the use of handhelds for data collection, application programmers at RTI International adopted three approaches to facilitate data analysis for the ATA survey items. The approaches utilize Blaise as the questionnaire development application used in the CAI software, and the SAS system software to produce the final data files.

The approach for the handheld data collection mode was developed for use on data retrieved from a multinational survey. ATA responses were stored as one variable per question compared to individual response storage options available with larger capacity portable devices. The single variables were then “deconstructed” using SAS® Decimal to Binary conversion functions to create analysis variables representing chosen responses.

Two of the earlier approaches use the SAS data structure document as a starting point to dynamically create variable names that correspond to the original list of variables. The third is a Microsoft® Visual Basic application that uses Blaise software to generate a SAS data set from the Blaise data structure.

Our presentation will: (1) demonstrate the need for recoding “All That Apply Variables”, (2) describe the different approaches taken in recoding, and (3) review the different approaches for *raison d’être*.

INTRODUCTION

ATA 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. In this paper we will discuss all approaches utilized by RTI programmers to prepare “Answer All That Apply” (ATA) variables for analysis, including the latest approach that was recently used to process the ATA variables collected by handheld computers.

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 Collected Data.

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. Note that the order of selection of the original responses will not be reflected in the new recoded variables.

	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.

An alternate recoding scheme of these variables is to use the convention of assigning a recode of 1 to indicate the responses that were chosen and 0 to indicate responses that were not chosen by respondents. The response data table is shown below in Figure 3.

	Ans1	Ans2	Ans3	Ans4	Ans5	Ans6
R1	0	0	1	1	0	0
R2	1	1	0	1	0	0
R3	1	1	1	1	1	1

Figure 3. Alternate Data Required For Analysis.

CAI APPROACHES

In order to facilitate data analysis for survey items, and to preserve the original input data, application programmers at RTI adopted four creative approaches to address these issues. Three of the approaches utilize Blaise as the questionnaire development application used in the CAI software, and the SAS system software to produce the final data files. The forth approach uses RTI's mobile data collection General Survey System to collect field data. Due to the limited storage capabilities of handheld computers, the ATA responses are stored as a single decimal value. This approach provides an efficient and convenient alternate to program the instrument and to store the data.

Two of the approaches use the SAS data structure document as a starting point to dynamically create variable names that correspond to the original list of variables. The third is a Microsoft Visual Basic application that uses Blaise software to generate a SAS data set from the Blaise database. It uses Visual Basic and the Blaise Component Pack, a tool provided by the CAI software, to create interactive display windows to illustrate to users how variable names in the Blaise database will be presented in the SAS data set. All four applications check for duplicate or non compliant variable names and create output files with variable names that need to be corrected.

SAS APPROACH

The first approach provides a simple and versatile SAS utility that uses the SAS software exclusively to manipulate the variables in the data structure document. Back up variables are created based on the instruments' variable names to store the original values entered by respondents. The newly created variables are verified for duplicate names, in order to avoid overwriting of values, and subsequently recoded as required.

The following is an outline describing the steps taken:

- Dynamically create the new variable names using the Blaise data structure document.

```
/*read data structure file*/
DATA getvars;
infile caitext truncover;
input @1 rawvar $10.;
```

```
numrec = _N_;
run;
```

- Generate text files containing blocks of statements and declarations of variable arrays/sets to manipulate the data.

```
/*rewrite in block format*/
DATA null;
set getvars;
file rvartext ls=80;
put rawvar $11. @;
run;
```

- Create looping mechanisms to cycle through the newly created variable names checking for duplicates and recoding variables to the required values needed for analysis.

```
/*check duplicate variable names*/
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; dupvar = zvals{i+k};
end; end; /* k loop */
arrvalue = arrvalue - 1;
end; /* i loop */       end; /* i < dim(zvals) */

/*recode ATA variables*/
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} = j;
end; end;
```

SAS AND THE BLAISE CAMELEON UTILITY APPROACH

The second approach uses the Blaise Cameleon Utility and data model to generate SAS Rename and Length statements to create the new toggle (0/1) variables. Additionally the Format and Label statements to add the labels and formats for these newly created variables are also generated by the Blaise API. Finally SAS code is generated to recode the variables to 1 for responses chosen and 0 for responses not chosen.

The following highlights the main steps used in this approach:

- Extract variable attributes and generate SAS statements to further manipulate the SAS data set. The new variables are named using the original array element names. The number of elements depends on the total number of response options available for each of the questions.

```
length rc1set11 3;
length rc2set11 3;
length rc3set11 3;
length rc4set11 3;
length rc5set11 3;
length rc6set11 3;
```

- Modify the Blaise data structure files to generate SAS Format and Labels statements to be applied to the data model.

```
VALUE TE_1F
1='blue' 2='green' 3='yellow' 4='purple' 5='white' 6='black' ;
```

```
VALUE rcTE_1F
-1='Dontknow' -2='Refusal' 1='Category selected';
```

- Convert ATA responses to toggle variables to generate a new set of values to determine whether values are chosen.

```
if set11 = 1 or set12 = 1 or set13 = 1 or set14 = 1 or set15 = 1 or set16 = 1
then rc1set11 = 1;
else if set11 = 9 then rc1set11 = -1; else if set11 = 8 then rc1set11 = -2;
if set11 = 2 or set12 = 2 or set13 = 2 or set14 = 2 or set15 = 2 or set16 = 2
then rc2set11 = 1;
if set11 = 3 or set12 = 3 or set13 = 3 or set14 = 3 or set15 = 3 or set16 = 3
then rc3set11 = 1;
if set11 = 4 or set12 = 4 or set13 = 4 or set14 = 4 or set15 = 4 or set16 = 4
then rc4set11 = 1;
if set11 = 5 or set12 = 5 or set13 = 5 or set14 = 5 or set15 = 5 or set16 = 5
then rc5set11 = 1;
if set11 = 6 or set12 = 6 or set13 = 6 or set14 = 6 or set15 = 6 or set16 = 6
then rc6set11 = 1;
```

SAS AND THE BLAISE COMPONENT PACKAGE APPROACH

The third approach uses the Blaise Component Package and Microsoft Visual Basic to write the SAS data files to produce the required data structure. The process uses an initialization file and intermediate data files to automatically create the final SAS data set. The final result is an application that allows users to dynamically update the initialization file in order to implement changes to variable names, labels, attributes and code assignments. Selected steps are described below.

- Use the Blaise Component Pack to develop software that integrates with the Blaise system in real time and accesses the data model.

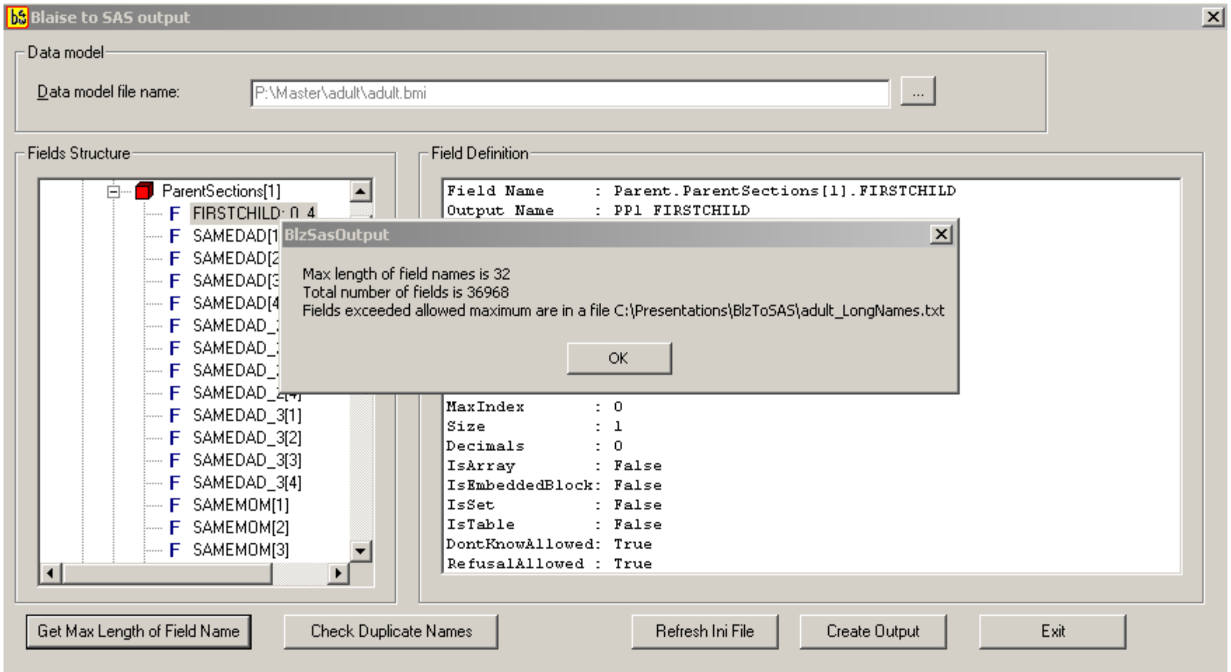
```
INI File                      Project
ProjectID=AAAA                Project ID
Folder=c:\Samples\include     Folder Name
LabelLength=40                Max Label Size
FieldNameLength=31            Max Name Size
UseTag=False                  Tags for Variables
DK=-4                         Value for DK
RF=-7                         Value for RF

INI File                Rename
R1=(Sect_1.=)           Rename 1: “Sect_1.” to “”
R2=(Sect_2.=)           Rename 2: “Sect_2.” to “”
R3=(Sect_3.=)           Rename 3: “Sect_3.” to “”
R4=(main_case.=)        Rename 4: “main case.” to “mc”
R5=(SectionTime.=Tmr_)
TotalRename=5           Total # of Rename =5
```

- Create an interactive application using Microsoft Visual Basic. Users are provided with a snapshot of variables, attributes and definitions and allowed to query the metadata.

```
 SAS Include Files
DATA out.timer;
INFILE '\\rtints29\caidata\blaise\Samples\timerDateDemo\include\timer.asc' LRECL =
392 TRUNCOVER;
%include '\\rtints29\caidata\blaise\Samples\timerDateDemo\include\timer_input.inc';
%include
'\\rtints29\caidata\blaise\Samples\timerDateDemo\include\timer_toggles.inc';
%include
```

```
'\\rtints29\caidata\blaise\Samples\timerDateDemo\include\timer_labels.inc';
%include
'\\rtints29\caidata\blaise\Samples\timerDateDemo\include\timer_formats.inc';
%include '\\rtints29\caidata\blaise\Samples\timerDateDemo\include\timer_dk_rf.inc';
RUN;
```



```
Field Name      : Parent.ParentSections[1].FIRSTCHILD
Output Name     : PP1 FIRSTCHILD
```

```
Max length of field names is 32
Total number of fields is 36968
Fields exceeded allowed maximum are in a file C:\Presentations\BlzToSAS\adult_LongNames.txt
```

```
MaxIndex        : 0
Size            : 1
Decimals        : 0
IsArray         : False
IsEmbeddedBlock: False
IsSet           : False
IsTable         : False
DontKnowAllowed: True
RefusalAllowed : True
```

SAS HANDHELDS’ APPROACH

Data from ATA variables collected by using handheld computers were stored as single decimal values. The values were saved as the sum of all responses marked, based on the mathematical expression, 2**n-1 where n is the question level 1…n. In handheld applications, the question is displayed as a single question contained on a single screen. The field interviewer reads each item response/level and selects that item if chosen by respondent. ATA values are stored in this manner for efficiency and convenience in data storage and in the application programming.

For example in the question below, respondents were asked about their knowledge or belief on diseases caused by smoke inhalation:

Question Text

E18VN. Based on what you know or believe what diseases could inhaling tobacco smoke from others cause?

[SELECT ALL THAT APPLY]

level	value
1 HEART DISEASE.........	1
2 LUNG DISEASE...........	2
3 LUNG CANCER............	4
4 OTHER CANCER..........	8
5 LOW BIRTH WEIGHT....	16
6 MISCARRIAGE	32
7 OTHER..................	64
8 DON'T KNOW............	128
9 REFUSED...............	256

Answers to this question are stored as a single number which is the sum of the values of each item chosen. If the respondent chose items 1, 4, and 6 then the answer is stored as 41.

In order to make the data collected suitable for analysis, the values stored for each question need to be “deconstructed” to represent the intended format. One way to deconstruct/reshape these values is to use a standard SAS decimal to binary conversion to get the data into one of the standard survey response 0/1 convention. To process the question variables into single variables per level, the number of answers/levels needs to be defined. The code below creates a binary string of 0/1 to indicate the ATA responses. The code simultaneously creates variables names that store the binary values read from right to left.

- Create a binary string of 0/1 which will indicate the ATA responses. In our E18VN question example, if a respondent chose options 1, 4 and 9, then the value stored in variable E18VN is 1+8+256, equal to 265. To illustrate:

```
/* if x =265, create a variable mybin in binary format */
mybin=put(x, binary16.);
/* convert back to num*/
numbin=input(mybin,16.);
/*output: mybin=0000000100001001; numbin=100001001*/
```

- Create a single ATA character string variable in binary format where responses chosen are coded as 1and responses not chosen are coded as 0. Then create a separate variable for each level in E18VN_1 through E18VN_9.

```
%macro deconstruct (varn, levels);
mybin&varn=put(&varn, binary&levels..);
array &varn.A (&levels) &varn. &levels - &varn._1;
do i=1 to &levels;
&varn.a(i)=substr(mybin&varn,i,1);
if &varn.a (i)=. then &varn. a (i)=0;
end;
%mend;
%deconstruct (E18VN, 9);
```

SELECTING THE SUITABLE METHOD

All four applications are robust and offer substantial savings in time and cost. The approaches eliminate manual procedures, reducing the risk of error, by automating the necessary steps to produce the required end result. Data is treated accurately and consistently in all three methods and is presented in the format required to facilitate analysis while preserving the survey input data.

The first approach provides a simple and versatile SAS utility that uses the SAS software exclusively to manipulate the variables in the data structure document. It requires SAS programming skills in reading external files, working with looping mechanisms and manipulating macro variables and data files. While knowledge of the Blaise data model or meta data is useful, it is not necessary for this application. Once the initial effort is completed, procedures that require minimal modifications are set in place for future cycles of the survey instrument.

The second approach requires knowledge of the Blaise Cameleon Utility and the data model in addition to SAS programming skills such as familiarity with the variable rename and variable attribute statements. The approach utilizes the Blaise tool to manipulate the data model and automatically identify variables that require renaming and recoding. The method provides an efficient approach to automate tedious manual procedures and produce an accurate end product.

The third approach presents an innovative interactive visual application that allows real-time access to data by analysts and other users. The approach requires technical expertise in the Blaise data model, the Blaise Component Pack and Microsoft Visual Basic. The result is a user friendly application that allows interactive participation and visualization of the data by authorized users. The application allows users to dynamically update the initialization file in order to implement changes to variable names, labels, attributes and code assignments.

The fourth approach is elegantly simple. It uses two straightforward and concise SAS steps. The SAS binary function transforms the numeric expression into a binary number which is then deconstructed into the desired 0/1responses to recode the newly created variables. A simple SAS Macro accomplishes the second step using the question variable name and the number of levels as parameters.

ACKNOWLEDGMENT

The authors acknowledge RTI International staff members Rita Thissen, for her input on the concept of this presentation, and Chuchun Chien and Barbara Felts for their assistance with image editing.

The authors also acknowledge Gary Franceschini for editing this paper.

REFERENCES

Chien, C., Handley, W., Felts, B.J., & Mai, Y. 2009. “A Simple SAS® Utility to Dynamically Create Variable Names and Recode the Associated Values.” *SESUG Users Group Conference*, Birmingham, AL.

Bibb, B.S., & Suresh, R. 2004. “Automatic Customization of Datasets Using Cameleon.” *9th International Blaise Users Conference*, Gatineau, Canada.

Filippenko, L., & Campbell, S. 2007. “Automatic Output from Blaise Databases to SAS Dataset Using Blaise API.” RTI International.

CONTACT INFORMATION

Wafa Handley, handley@rti.org, 919-541-6066
Barbara Bibb, bibb@rti.org, 919-541-7413
Lilia Filippenko, lfilippenko@rti.org, 919-541-6717
Jay Levinsohn, jrl@rti.org, 919-541-6399
Donna Medeiros, djm@rti.org, 919 -541-8788

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