

Paper TU04

Testing Survey Skip Patterns Using Metadata-Generated SAS® Code

David Gilbertz, Northrop-Grumman, Inc., Atlanta, GA

ABSTRACT

Data analysts must routinely review incoming survey data to determine if the skip-pattern was followed precisely by the interviewer for each respondent. Interviews are reclassified as partial complete interviews if the skip pattern was not followed or if “refused” responses are numerous. Writing code to test for these problems is tedious and requires thorough code review to ensure the code follows the skip pattern accurately. The author’s goals were to reduce the time spent reviewing code, and to use metadata to simplify the code-writing process

The Behavioral Risk Factor Surveillance System (BRFSS), in use since 1984, is a telephone survey of about 200 questions asked of over 350,000 respondents from all 50 states, the District of Columbia and several U.S. territories. About 120 questions are selected by states as optional modules. Each question has a maximum of three conditional skips and metadata for each is written in Microsoft Excel. A reusable SAS program imports the metadata and writes SAS code using a series of PUT statements. The analyst need only edit the metadata for annual changes to the questionnaire. The time savings allows the analyst to test the skip patterns earlier than in previous years.

INTRODUCTION

Skip patterns, or conditional logic in questionnaires, are often used to eliminate interview questions based on demographic or other characteristics of a respondent. The variety of methods used to conduct surveys, such as paper, telephone, and web-based, can affect the correct application of skip patterns. Automated interviewing systems, such as Windows Computer Assisted Telephone Interview (Win-CATI), certainly help in following skip patterns correctly but are not fool proof. Checking skip patterns is particularly important with data in the public domain.

The Behavioral Risk Factor Surveillance System (BRFSS) is a public domain telephone survey available continuously since 1984. The public health departments of 53 states and territories individually collect the data from over 350,000 completed interviews, which is then compiled by the Centers for Disease Control and Prevention into a national SAS database available online at <http://www.cdc.gov/brfss/>. The 2006 survey contains 22 sections with a total of 84 questions, mostly asked using Win-CATI systems. States or territories can also choose among 17 optional modules containing 117 questions. At the end of the interview, each questionnaire is classified using disposition codes as a complete, partial complete or incomplete. Only complete and partial complete interviews are available in the final public-use database.

SAS code is written each year to verify that section and module skip patterns are followed correctly. BRFSS also checks that all responses in a module or section are not refusals. These and other data quality checks ensure the disposition code was assigned correctly. Writing SAS code to check so many sections and modules is tedious and time-consuming. The code must also be thoroughly reviewed several times. Otherwise, an error may cause a completed interview to be incorrectly classified as an incomplete.

THE PROBLEM

Developing SAS code for the purpose of dynamically constructing a program to test skip patterns, would reduce the annual chore of writing new code to accomplish the task. Since the logic of the program would only need to be developed once, it can be reused every year and would not require thorough retesting. Only the variables for each question would need to be rewritten. Metadata, in the form of an Excel or other tabular database, could serve this purpose and be read once whenever the metadata table changes. A secondary SAS program or product would be written as output and then used to actually test the data and skip pattern.

The BRFSS survey is not a continuous questionnaire but is broken up into sections or modules according to the subject matter of interest. Each section/module contains skip patterns which have up to three conditions that must be tested. Sometimes the entire section or module must meet a condition. Within each section, survey questions may or may not require “if then else” conditional logic. A percentage value must be created that measures the number of questions answered according to the skip pattern compared to those questions refused or not answered. A report must then be generated to show the sections and modules answered either correctly or incorrectly based on the percentage. If the skip pattern of a section or module is not followed then the disposition code must be assigned indicating sections or modules are only partially complete.

THE SOLUTION

To fully understand the solution, it is important to differentiate between the “primary” program and the “secondary” program or product. The primary program contains the PUT statements that write the secondary program named MODCHECK_STATEMENT.TXT using a FILE statement. The primary program only needs to be run once when the metadata is created or if the metadata table changes. The modcheck_statement.txt will do the real work throughout the year checking the data for problems in the skip pattern. The metadata is only read by the primary program.

A metadata spreadsheet was created in Microsoft Excel with the variables needed to dynamically write the SAS code: SECMOD, VAR, SKIP0, SKIP1 and SKIP2. All except SECMOD are snippets of Boolean logic code, which will return a 1 (true) or 0 (false) for each respondent’s data record. Boolean logic testing is an important tool for determining if a variable is within a certain range. Together with the SUM function, Boolean logic can count the number of true statements.

Let's look at the metadata used in the primary program more closely. Each row of the spreadsheet or metadata represents a question in the survey. SECMOD is a value assigned to each section or module and will keep the section/module together after sorting. Numbers assigned to modules have three digit assignments because modules are optional and are coded differently than sections, which are "core" questions asked of every respondent. VAR contains the variable assigned to a question and tests for all non-missing values except the refused value. SKIP0, SKIP1 and SKIP2 contain the variable(s) in the skip pattern and their conditional value. SKIP0 is the condition to be tested for an entire section or module. SKIP1 is the test for the first skip value and SKIP2 is the test for the 2nd skip value. After these records are read using the Import procedure, sorting is done by SECMOD, SKIP0, SKIP1 and SKIP2. To simplify, let's look at the spreadsheet for module 7 of the survey (Table 1).

SECMOD	VAR	SKIP0	SKIP1	SKIP2
107	11 <= asthmag <= 98	asthma2 = 1		
107	asattack in (1,2,7)	asthma2 = 1	asthnow = 1	
107	1 <= aservist <= 98	asthma2 = 1	asthnow = 1	
107	1 <= asdrvist <= 98	asthma2 = 1	asthnow = 1	
107	1 <= asrchkup <= 98	asthma2 = 1	asthnow = 1	
107	1 <= asactlim <= 888	asthma2 = 1	asthnow = 1	
107	1 <= asymptom <= 8	asthma2 = 1	asthnow = 1	
107	1 <= asnoslep <= 8	asthma2 = 1	asthnow = 1	asymptom ne 8
107	1 <= asthmed2 <= 8	asthma2 = 1	asthnow = 1	
107	1 <= asinhair <= 8	asthma2 = 1	asthnow = 1	

Table 1. Metadata for Module 7.

SECMOD is a 3-digit number, 107, which indicates it is module 7, the Asthma Module. The first question is ASTHMA2, which must be between 11 and 98, inclusively. SKIP0 is the condition for the entire module and shows that a previously asked question, ASTHMA2, is 1. For all the other questions in the module, SKIP1 is used, which is "ASTHNOW = 1." SKIP2 is not used except for question ASNOSLEP, in which case "ASYMPTOM ne 8." The code for Module 7 output from the program's PUT statements is shown below (Table 2).

SKIP PATTERN	CODE WRITTEN FROM 'PUT' STATEMENTS
	*MODULE 7;
	count + 1;
	if substr(mods,count,1) = '1' then do;
SKIP0 starts	if (asthma2 = 1) then do;
	numer = (11 <= asthmag <= 98);
	denom = 1;
SKIP1 starts	if (asthnow = 1) then do;
	numer + sum((asattack in (1,2,7))
	, (1 <= aservist <= 98)
	, (1 <= asdrvist <= 98)
	, (1 <= asrchkup <= 98)
	, (1 <= asactlim <= 888)
	, (1 <= asymptom <= 8)

	, (1 <= asthmed2 <= 8)
	, (1 <= asinhalr <= 8));
	denom + 8 ;
SKIP2 starts	if (asymptom ne 8) then do;
	numer + (1 <= asnoslep <= 8);
	denom + 1 ;
SKIP2 ends	end;
SKIP1 ends	end;
	if numer/denom => .5 then substr(mods ,count,1) = '0';
SKIP0 ends	end;
	else substr(mods ,count,1) = '0';
	end;

Table 2. SAS code from the MODCHECK_STATEMENT.TXT output file written using PUT statements, which will be used to test skip patterns in Module 7.

In the SAS code, a new variable, SKIP, is created to simplify processing using the sorted SKIP1 and SKIP2 variables and is used to determine where conditional groups begin and end. SKIP is a 4-character text string containing the Boolean logic values for first.skip1, last.skip1, first.skip2 and last.skip2. Table 3 shows the action taken for each SKIP value. NUMER and DENOM are

SKIP	ACTION
0000	ADD ANOTHER NUMERATOR IN THE MIDDLE OF A GROUP
0001	SKIP1 DOES NOT END BUT SKIP2 IS ENDING
0010	START SKIP2 (IF NOT MISSING)
0101	END SKIP1 AND SKIP2 GROUP
1010	START SKIP1 (IF NOT MISSING) AND START SKIP2 (IF NOT MISSING)
1111	SKIP1 AND/OR SKIP2 ARE STARTED AND ENDED FOR ONE STATEMENT

Table 3. Values of skip and conditional action to be taken in SAS code.

variables which keep track of questions answered appropriately (NUMER+1, DENOM+1) or which were not answered (NUMER+0, DENOM+1). SIGN_N and SIGN_D are variables that keep track whether a count must be started (SIGN_N = "=") or simply added (SIGN_N="+") to the numerator and denominator, respectively. NUMER and DENOM can be used to create a percentage of questions answered correctly. The respondent must answer at least 50% of the questions with non-refusal or non-missing responses for each section/module to be a completed interview. Table 4 shows the entire "primary" SAS program which uses PUT statements to write the end product or secondary SAS program called MODCHECK_STATEMENT.TXT. The primary program below is reusable each year unless the number of skips changes. Only the metadata must be rewritten annually for the new questionnaire.

```
*READ METADATA;
proc import datafile="modcheck.xls"
            out      =modcheck replace;
run;

proc sort data=modcheck;
```

```

by secmod skip0 skip1 skip2;
run;

*MINSEC: FIRST SECTION AFTER DEMOGRAPHICS;
%let minsec=12;
*MAXSEC: LAST CORE SECTION;
%let maxsec=22;

data mod1;
  set mod1;
  by secmod skip0 skip1 skip2;
  retain denom sign_d sign_n range;
  length name $40 sign_d sign_n $1 range $12 lastskip skip $4;
  file "c:\modcheck_statement.txt";

*SKIP: CONCATENATE FIRST AND LAST VARS FOR SKIP1 AND SKIP2;
skip = put(first.skip1,1.)||put(last.skip1,1.)||
       put(first.skip2,1.)||put(last.skip2,1.);

*LASTSKIP: RETAIN LAST VALUE FOR SKIP;
lastskip = lag(skip);
*THE VARIABLE (AND CODE) BEING TESTED;
var = trim(left(var));

*SKIP0: SKIP APPLIES TO ENTIRE SECTION OR MODULE;
skip0 = trim(left(skip0));
*SKIP1: SECOND SKIP IN PATTERN, ONLY APPLIES TO ONE OR MORE VARIABLES WITHIN
THE SECTION/MODULE;
skip1 = trim(left(skip1));
*SKIP2: THIRD SKIP IN PATTERN, APPLIES TO SUBGROUP OF VARIABLES IN SKIP1;
skip2 = trim(left(skip2));

if secmod < 100 then do;
  name = "*SECTION " || trim(left(put(secmod,2.))) || ";";
  range = "sec&minsec._&maxsec";
end;
else if secmod > 100 then do;
  name = "*MODULE " || trim(left(put(secmod-100,2.))) || ";";
  range = "mods";
end;

if first.secmod then do;
  *LABEL THE SECTION/MODULE;
  put name;
  *RESET COUNT FOR FIRST SECTION AFTER DEMOG AND FIRST MODULE;
  if secmod in (&minsec,101) then put "count = 1;";
  else put "count + 1;";
  *ADD STATEMENT SO MODULE IS ONLY TESTED IF USED;
  if secmod > 100 then put "if substr(mods,count,1) = '1' then do;";
  sign_d = "=";
  sign_n = "=";
  if skip0 ne " " then put "if (" skip0 ") then do;";
end;

*IF SKIP1 AND/OR SKIP2 ARE STARTED AND ENDED FOR ONE STATEMENT;
if skip = "1111" then do;
  if skip1 ne " " then do;

```

```

    put "if (" skip1 ") then do;";
end;
if skip2 ne " " then do;
    put "if (" skip2 ") then do;";
end;
denom = 1;
put "numer " sign_n " (" var ");";
put "denom " sign_d " 1;";
if sign_d = "=" then sign_d = "+";
if sign_n = "=" then sign_n = "+";
if skip2 ne " " then do;
    put "end;";
end;
if skip1 ne " " then put "end;";
end;

*1010,0010: START SKIP1 (IF NOT MISSING) AND/OR START SKIP2 (IF NOT MISSING);
else if skip in ("1010","0010") then do;
    if skip1 ne " " and lastskip ne "1011" then
        put "if (" skip1 ") then do;";
        if skip2 ne " " then put "if (" skip2 ") then do;";
        denom = 1;
        put "numer " sign_n " sum((" var "));
    *CHANGE ONLY NUMERATOR SIGN TO + ;
    if sign_n = "=" then sign_n = "+";
end;

*1011: START SKIP1 (IF NOT MISSING) AND START AND END SKIP2 (IF NOT MISSING);
else if skip in ("1011") then do;
    if skip1 ne " " then put "if (" skip1 ") then do;";
    if skip2 ne " " then put "if (" skip2 ") then do;";
    denom = 1;
    put "numer " sign_n " (" var ");";
    put "denom " sign_d " 1;";
    if sign_n = "=" then sign_n = "+";
    if sign_d = "=" then sign_d = "+";
    if skip2 ne " " then put "end;";
end;

*0001: SKIP1 DOES NOT END BUT NEW SKIP2 IS STARTING;
else if skip in ("0001") then do;
    if skip2 ne " " then put "if (" skip2 ") then do;";
    denom + 1;
    put "      , (" var "));";
    put "denom " sign_d " " denom ";";
end;

*0000: ADD ANOTHER NUMERATOR IN THE MIDDLE OF A GROUP;
else if skip in ("0000") then do;
    denom + 1;
    put "      , (" var "));";
end;

*0111: END SKIP1 GROUP AND START AND END SKIP2 GROUP;
else if skip in ("0111") then do;
    if skip2 ne " " then put "if (" skip2 ") then do;";
    denom = 1;

```

```

    put "numer " sign_n " (" var ");";
    put "denom " sign_d " " denom ";";
    if skip1 ne " " then put "end;";
    if skip2 ne " " then put "end;";
end;

*0101: END A SKIP1 AND SKIP2 GROUP;
else if skip in ("0101") then do;
    denom + 1;
    put "          , (" var ");";
    put "denom " sign_d " " denom ";";
    if sign_d = "=" then sign_d = "+";
    if skip1 ne " " then put "end;";
    if skip2 ne " " then put "end;";
end;

if last.secmid then do;
    if skip0 ne " " then do;
        put "if numer/denom => .5 then substr(" range ",count,1) = '0';";
        put "end;";
        put "else substr(" range ",count,1) = '0';";
    end;
    else put "if numer/denom => .5 then substr(" range ",count,1) = '0';";
    if secmod > 100 then put "end;";
end;
run;

```

Table 4. Primary SAS program used to read metadata and write program to test survey skip patterns.

CONCLUSION

Using metadata to write SAS code is an effective method for simplifying skip pattern testing. Creating the metadata is the most time consuming task in the process but can easily be modified if errors are found. The tedious task of code checking is no longer necessary and the primary program does not require any modification for a new survey. In this example, skip patterns with up to three skips were accurately tested. The final product, MODCHECK_STATEMENT.TXT, is created each time the primary program is submitted and is used throughout the year to do the actual work of testing the skip patterns. The primary program must only be resubmitted if the metadata changes.

Appendix 1. Selected section/module metadata for testing skip patterns.

SECMOD	VAR	SKIP0	SKIP1	SKIP2
13	drnkany4 in (1,2,7)			
13	101 <= alcdays4 <=230 or alcdays4 in (777,888)		drnkany4 = 1	
13	1 <= avedrnk2 <= 98		drnkany4 = 1	alcdays4 ne 888
13	1 <= drnk3ge5 <= 88		drnkany4 = 1	alcdays4 ne 888
13	1 <= maxdrnks <= 98		drnkany4 = 1	alcdays4 ne 888

102	drhpc in (1,2,7)	children notin (0,88,99)		
102	havhpc in (1,2,7)	children notin (0,88,99)	drhpc = 1	
102	ciflush2 in (1,2,7)	children notin (0,88,99)	childge6 = 1	
102	"011988" <= rcvfvch2 <= "122006"	children notin (0,88,99)	childge6 = 1	ciflush2 = 1
102	1 <= rnofvch2 <=7 or rnofvch2 = 77	children notin (0,88,99)	childge6 = 1	
116	ipvthrat in (1,2,7)	ipvsafe = 1		
116	ipvphyvl in (1,2,7)	ipvsafe = 1		
116	ipvpghrt in (1,2,7)	ipvsafe = 1		
116	ipvuwxex in (1,2,7)	ipvsafe = 1		
116	ipvpvl12 in (1,2,7)	ipvsafe = 1	ipvpghrt = 1 or ipvuwxex = 1	
116	ipvsxinx in (1,2,7)	ipvsafe = 1	ipvpghrt = 1 or ipvuwxex = 1	ipvpvl12 = 1
116	1 <= ipvreln <= 15 or ipvreln = 77	ipvsafe = 1	ipvpghrt = 1 or ipvuwxex = 1	

Contact Information

Your comments and questions are valued and encouraged. Please contact the author at:

David Gilbertz
Northrop Grumman Corporation
3375 Northeast Expressway
Koger Center/Harvard Building
Atlanta, Ga. 30341-3717
Work Phone: 770-488-6537
Fax: 770-234-6581
E-mail: DGilbertz@CDC.gov

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