Paper 093-2012

Three Simple Steps to Recovering and Inserting Data Using Do Loops, Indexed Macro Variables, and PROC SQL UPDATE Statements

Stacey D. Collins, University of Michigan – Institute for Social Research, Ann Arbor, MI, USA

ABSTRACT

When recovering data records from paradata or log files, it can be tedious to parse and restore data to the correct variable within the data set, particularly when the number or order of variables which contain data can vary between records. This paper presents an efficient way to use do loops and indexed macro variables to parse variable names and data from a paradata file, and load them into the main data set using PROC SQL UPDATE statements. This code is useful for restoring records one at a time, and does not require the programmer to hard code any variable names or variable orders for the purpose of inserting the data. The code is presented in SAS® 9.2 in a windows environment.

INTRODUCTION

In the Survey Research Operations (SRO) department at the University of Michigan's Institute for Social Research, one of the things the group deals with regularly is making sure data collected from an individual survey is correctly captured and stored in the larger database system. For example, when capturing data from a web-based survey, two things should happen, the survey responses should be written to the main data table, and the paradata (data about the process, such as browser information, keystrokes, timings, etc.) should be written to the paradata file. Sometimes, however, errors occur and one or both of the expected processes fail. In cases when the survey responses (i.e. the answers) are not written to the main data table, it becomes necessary to restore the data from the associated paradata file.

In the context of a complex survey instrument, such as the web-based survey in this example, the next question of the survey often depends on responses provided in previous questions. While the main data table includes variables for all possible survey questions, an individual survey will only include answers for the questions that were presented to the respondent. Therefore, restoring data is more complicated than simply recovering the answers and inserting them in order. Here, it is necessary to recover the question/answer pair so that the answers are restored to the proper variables in the main data table, and variables representing questions that were not asked remain blank. If one is not careful, this can turn into a programming nightmare.

This paper presents code which will 1) parse the variable names and data (i.e. questions and answers) from the paradata file, 2) store the variables and data as indexed macro variables (thus maintaining the relationship between each question and answer), and 3) insert the data into its proper variable within the main data table. This can be done without hard coding any variable names or variable orders, and can be used with any number of variables.

1. PARSING A DATA RECORD FROM THE PARADATA FILE

In this web-based survey example, the raw paradata file captures a lot of information, including browser size, keystroke data, and timings, as mentioned above. But, it also captures the questions and answers that appear on each page because, as you recall, that will not be the same for each individual survey. Specifically, one row in the raw paradata file represents one page of the survey instrument. The question names are concatenated and stored in one field in the raw paradata file, while the answers are concatenated and stored in another. An example of the raw paradata fields of interest appear in Table 1.

SurveyID	Page_Num	Page_Questions	Page_Answers
aaa123	3	Space.A1e^Space.A1c^Space.A1a^Space.A1b^Space.A1d^	1^3^1^1^1^
aaa123	4	Conversations.A2e^Conversations.A2g^Conversations.A2c^Conversations.A2a^Conversations.A2b^Conversations.A2d^Conversations.A2f^	1^1^6^1^1^1^1^
aaa123	5	Distractions.A3c^Distractions.A3a^Distractions.A3b^Distractions.A3d^	1^1^1^1^

Table 1. Example of Data in Raw Paradata File

To begin the process of restoring the data, the data is first parsed so that each question and answer is stored in one row of data. The &pk macro variable subsets the raw paradata file so that only the individual survey of interest is included for parsing. Here, the unique questions and answers are separated by carets, so a do loop with the scan function is used to perform the parsing. Since the number of questions and answers vary by page (Page_Num), the do loop runs from 1 to 1,000 (knowing there are always less than 1,000 questions/answers on a page) and a simple IF-THEN OUTPUT with an ELSE LEAVE statement is included to end the loop when all of the questions and answers have been parsed.

SYNTAX 1.1

```
%LET pk = aaa123;

DATA Paradata_Parsed (KEEP = PrimaryKeyValue PrevPageNo Question Answer);
SET rawparadata;
WHERE primarykeyvalue = "&pk";
DO i = 1 TO 1000;
       question = SCAN(prevpagequestions, i, '^');
       answer = SCAN(prevpageanswers, i, '^');
       IF NOT MISSING(question) THEN OUTPUT;
       ELSE LEAVE;
END;
RUN;
```

Using Page_Num = 3 as an example, there are 5 questions/answers that need to be parsed out. The do loop will scan for the ith question each iteration. Every time a question/answer is parsed, it will output to the new data set, Paradata_Parsed. On the 6th iteration, the SCAN function will return missing values for question and answer, so the ELSE clause will take effect, ending the do loop, and thus not wasting additional processing time. Similarly, where Page_Num = 4, there are 7 questions/answers, and again, the do loop will continue scanning and outputting until the 8th iteration, when it will end. The process will continue for every row in the raw paradata file until all question/answer pairs have been parsed and output.

The parsed data set now looks like this:

SurveyID	Page_Num	Question	Answer
aaa123	3	Space.A1e	1
aaa123	3	Space.A1c	3
aaa123	3	Space.A1a	1
aaa123	3	Space.A1b	1
aaa123	3	Space.A1d	1
aaa123	4	Conversations.A2e	1
aaa123	4	Conversations.A2g	1
aaa123	4	Conversations.A2c	6
aaa123	4	Conversations.A2a	1
aaa123	4	Conversations.A2b	1
aaa123	4	Conversations.A2d	1
aaa123	4	Conversations.A2f	1
aaa123	5	Distractions.A3c	1
aaa123	5	Distractions.A3a	1
aaa123	5	Distractions.A3b	1
aaa123	5	Distractions.A3d	1

Table 2. Example of Data in Parsed Paradata Data Set

2. EXPORT DATA TO INDEXED MACRO VARIABLES

Next, the questions and answers are exported to indexed macro variables that will be used to insert the data in the following step. The macro variables will be indexed by row number, so that the question/answer combination can be referenced together.

SYNTAX 2.1

```
DATA _null_;
SET work.Paradata_Parsed END = end_of_dataset;
CALL SYMPUTX('Q_'||STRIP(PUT(_N_,8.)), STRIP(TRANSLATE(question,'_','.')));
CALL SYMPUTX('A_'||STRIP(PUT(_N_,8.)), STRIP(answer));
IF (end_of_dataset) THEN CALL SYMPUTX('N_Rows', STRIP(PUT(_N_,8.)));
RUN;
```

Since row numbers are numeric, the put statement with the strip function will convert the row number to a character value, without spaces, that is then concatenated as part of the macro variable name. Questions are stored as the macro variables “Q_1”, “Q_2”, etc, and similarly, answers are stored as “A_1”, “A_2”, etc. The values in the question variable of the parsed data set are actually the variables in the main data table into which the answers will be inserted. Because the questions, as read from the raw paradata file, contain periods, and SAS has converted those periods to underscores in the main data table, a TRANSLATE function has been added. Therefore, “Space.A1e” will be stored as “Space_A1e” in macro variable “Q_1”, and will match the variable name in the main data table. Finally, an IF statement is included to find the number of rows in the parsed data set and store it as another macro variable, “N_Rows”. This will be used later to assure we have inserted all of the necessary question/answer combinations.

3. INSERT THE DATA

Finally, the data stored in the answer set of macro variables is ready to be inserted into the main data table. First, a blank row is inserted into the main data table with the primary key value of the individual survey being restored.

SYNTAX 3.1

```
PROC SQL;
INSERT INTO paradata.maindata
SET PrimaryKey = "&pk"
;
QUIT;
```

Next, a macro is created to update the variables (questions) with the data (answers). This do loop will run through each set of question/answer macro variables (i.e. “Q_1” / ”A_1”, etc.) and update the variable in the blank record with the correct corresponding data, using a PROC SQL UPDATE statement. The do loop is run from 1 to &N_Rows, which, as you recall, was created as the last set of question/answers in the parsed data set. Therefore the loop will stop when all question/answer pairs have been restored to the main data table.

SYNTAX 3.2

```
%MACRO insertloop();

%DO j = 1 %TO &N_Rows;

PROC SQL;
UPDATE paradata.maindata
SET &&Q_&j = "&&A_&j"
WHERE primarykey = "&pk"
;
QUIT;

%END;

%MEND insertloop;

%insertloop()
```

CONCLUSION

By using the indexed macro variables to store both the variable names (questions) and data (answers), the restored data can easily be inserted into the main data table without having to hard code variable names, or figure out the order in which the variables appear. This code is useful for parsing and restoring one record at a time, and is dynamic so that it can be easily applied to any record or data set with paradata present in the raw paradata file. In addition, while the paradata file in this example is specific to the web-based surveys conducted in SRO, the need to parse and restore data may be present in other situations as well. If similar data is available in other types of log or keystroke files, the parsing code (1) can be adjusted to recover the necessary information and data can be stored in a data set similar to Table 2. Then, the rest of the code (2, 3) is still applicable with minimal modification.

ACKNOWLEDGMENTS

I would like to thank Philip Wright, from the University of Michigan, who first taught me about the uses of indexed macro variables. I have found that information invaluable in my day to day work. Thank you, Phil!

CONTACT INFORMATION

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

Stacey D. Collins
University of Michigan – Institute for Social Research
330 Packard St.
Ann Arbor, MI 48109
734-647-6352
sbarbosa@umich.edu