

AUTO-EDITING: A DATA MANAGEMENT TOOL FOR RESEARCH

by William S. Calvert, Service Research, Inc.*
and L. Swartz, NNMCM, Bethesda, Md.

ABSTRACT

We have developed a tool for efficient editing of research data. With our approach, code is generated and executed in real-time fashion using %INCLUDE and the SAS® MACRO environment. The system is driven by a user defined, parameter text file which may be created by non-programming personnel. This "auto-editing" system can scan multiple SAS files, edit for incorrect data, and collect the errors from multiple files to produce specialized output reports for research analysts or data collection personnel. With these reports it is much easier to identify and resolve data errors. The system we present for automatically generating edit programs was originally developed for a major multi-site national study. The application of "auto-editing" to the problems of similar studies are discussed. Suggestions to make life easier for the researcher through good data management practice, are also presented.

INTRODUCTION

The idea for "auto-editing" evolved by necessity during a longitudinal, multi-site study. A data coordinating center had been newly organized to coordinate the data collection, management, and study protocol for one of the first national studies of child care in the United States. The data collection instruments as well as the data management procedures were being established from the ground up with little chance to pause. Our infant study subjects were constantly getting older and a new battery of data collection instruments had to be developed and put in place almost every month. While some data editing was done at batch conversion time when the forms for each day were processed, an overall picture of kinds of errors that were occurring was not easily formed from the small, daily, groups of data for almost a hundred files. The sooner discrepancies were found and communicated to data collection sites and researchers, the better; although the data was not to be statistically analyzed for the main hypotheses until the children were three years old. Consequently, except for critical errors, several months of errors could be collected for presentation in a single reporting cycle.

HOW IT WORKS

Standardized naming conventions were established for the research files according to the collection instrument and age of the children. A four character name was used in conjunction with prefix and suffix notations to indicate file type and stage of preparation. For example a four character name would start with 'F', have two digits for age and a letter (or alpha-numeric) for the form type (F01A, F01B, ... F01X). Each observation point designated by a two digit infant age in months, could have a group or battery of forms, or just a single one. The one-month observation point in our study, included a two booklets of interviewer recorded data, and several one-page questionnaires (subject self-administered instruments). Infants and their families were recruited over a ten-month period. Each new observation point overlapped subsequent data collection instruments. In other words, the single recruitment form overlapped the 20 different forms being used for the one-month observation, which overlapped the 30 different types of forms for the 6-month observation battery (not to mention other observation points). The inventory of data files continued to grow.

The overlap of forms and their data files began to impact the center more as the study progressed. There was a need to establish an overview and assure a thorough accounting of the kinds of data and the extent of missing data in the completed segments of the project. This information was also to be communicated to the data collection sites at the interviewer level to help maintain or improve data quality where necessary. Messages of missing data from the almost one hundred different files (as designated by the four character name) needed to be gathered by subject ID and interviewer so that a succinct output form could be generated.

If the data collection form was a questionnaire, all questions were generally expected to be answered; responses for interview booklets and observation forms usually became more complex, with the appropriateness of an answer relying on the response to previous questions. These are usually called "skip patterns". For example, the tasks required in looking at multiple

files for missing data might be:

F01A - all questions should be answered
 F01B - all questions should be answered
 F01C - all questions should be answered
 F02A - answer only questions (2,6,7,8)
 F02B - All questions
 F02C - only questions 1-10 should be answered
 . by all participants.
 .
 .

F03X - etc.

For forms where all questions should be answered, (any missing data is a problem), the system was simple. Please see Display 1. "Sample Program Listing". The program works as follows:

1. Begin the macro loop.
2. Create SAS code using the file specific parameter text file. Parameter text files are ASCII data files containing the variable list or PROC CONTENTS output. The macro code uses this information to produce an "if" statement for each variable that tests for missing and when true, creates an error message.
3. Run the new SAS program just created.
4. Accumulate file error messages.
5. Indicate next file to be edited and return to step 1.
6. Once all files are edited: sort by subject ID and produce report or mailer (report for data collection site).

After this first stage of implementation, the task of picking out variables for missing error messages from files with skip patterns was undertaken. The solution was to add an indicator or flag to the parameter text file. We used an "N" in column 15 to indicate that the data was necessary. The auto-edit program required only minor change. The code generated by the AUTOEDIT macro, in pseudo code, would be:

If variable is missing and code equals "N"
 (data required), then generate an error
 message.

The parameter text file for each data file, created as an ASCII text file, was structured as indicated below:

```
0001 ID          N F07BQ00SITES/LOC/SUBID
0002 REL         F07BQ01RELATNSHP OF SUBJ
0003 INTM07B     N F07BQ02DATEINTERVIEW/MO
0004 VARIABLE6   N F07BTIMES...      VARIABLE3 eq 5
```

(Again, most of the data used in this file can come from a PROC CONTENTS).

The first 8 columns (after the line numbers) contain the variable name. The indicator code is at column 15. Next is the label associated with the variable. The label contains the form number, question number, and a description of the question. Space remains for additional information. Note line 4 of the example above. This extra information could be used to generate more complex edits (nested 'if' conditionals) of the data files. The pseudo code logic could be:

If variable 3 equal 5, and Variable 6 is
 missing, create error message.

"Auto-editing" progressed from a simple

"IF Variable = ."

statement to a nested IF condition which could create not only a "data is missing" error message but also an "answer should have been skipped," or "answer out of range". To do this the indicator code took on a wide range of values which triggered specific types of code logic. Variable names, operators, and values were added at the end of the record to provide the necessary information for the code generation logic. (See Display 2. "Parameter Text File Examples" for more details.) The range of edit information available expanded and became more useful.

The information collected was rearranged and put into an appropriate report and/or mailing form. The AUTOEDIT macro proved to be very efficient. When data edits were sorted by form and question, a kind of PROC FREQ on "problem questions" could be readily produced for research analysts. This same edit information file could be sorted again on subject ID to produce ready-to-mail edit reports for the sixty-some interviewers to check out.

The ready-to-mail report form was designed to summarize problems and to function as an interactive communication form so that data collection sites could respond to edits and data problems. It also made it

easier for data center personnel to identify and propose solutions for the problem(s). (Please see Display 3. "Ready-To-Mail Edit Report")

From the initial requirement of editing for missing data, "auto-editing" grew to a system with messages for missing forms, messages for the presence of sections of forms, and more complex custom code. The procedural steps were expanded to accommodate the increased complexity. The approximate sequence of steps is shown below:

1. Merge different forms sequences by ID to discern missing forms, and create standard error message.
2. Check data across files to ascertain whether answers in other files are correct and create standard error message.
3. Begin Macro for auto-editing.
4. Create code by using parameter text file with edit logic.
5. Run the generated program and create standard error message(s).
6. Check if subject ID corresponds to master subject ID file and create error message.
7. Accumulate all error messages.
8. Give parameters for next file to edit; return to step 3, if not finished.
9. Include complex, custom code for booklets, if needed.
10. Merge file errors, ID errors; auto-edit messages and custom-code messages into one file.
11. Create question/edit verification report by file, question number. (See Display 4. "Question/Edit Verification Report").
12. Sort by interviewer and subject ID; and produce mailers (site specific error reports).

DISCUSSION:

The program presented is a simple "auto-editing" system. We have indicated with examples how it might be modified to accommodate more complex requirements. The power of this system could be realized in any environment where there are a large number of related files which have different and changing edit requirements. This is the usual condition in the research environment where different projects and their data files are always evolving. If there are many data collection source documents and data files, as there often are in longitudinal studies, the benefits of this tool are increased. With "auto-editing", several types of benefits are obtained: flexibility, ease of maintenance, production of timely management information, enforcement of good data management practice, and higher quality research data.

Researchers are often expected to be expert computer users and data managers, as well as experts in their particular academic field. Often this is not the case. This situation usually means that stated information processing requirements are not as they should be and are often subject to change and turn-about. Even the best researchers can make mistakes requiring rework. Flexibility is valuable. A simple change in the parameter text file can fix a mistake or oversight and the system is ready to run again. Logic changes to how the "auto-edit" process works can be made in a single program instead of many.

Ease of maintenance is related to flexibility. System changes (on the mainframe where you have little control) are easier to adapt to if you have fewer lines of actual code to modify. The macro structure is actually more simple in concept than the large number of programs that might otherwise be needed to accomplish the same tasks.

From the researcher's point of view the flexibility and ease of maintenance benefits are just under-pinnings for timely management information. To keep multiple sites collecting data that is comparable across sites requires constant communication among and between the sites and the data center. This communication is necessary to identify and resolve data collection problems. "Auto-editing" provides information which makes it easier to identify study wide problems as well as site specific problems.

A good research data management system must be in place early in the study. It consists of a process and a mechanism for monitoring the process. Process monitoring information is essential to keep the data management system working and to optimize it as required. The overview of the research data collection process gained from the monitoring of data quality using "auto-editing" provides some of this information. Another necessary part of the data management system comes from the requirements of automation: Names of things must be consistent. Data collection instruments, such as questionnaires and interview booklets, have to be analyzed to determine the "correctness" of answers, and the range of appropriate responses. Data inventory control is essential. Standard naming conventions for "auto-editing" ease data inventory problems and simplify documentation requirements. The parameter text file used by the system we present provides a single source document (for each file) containing data item specific information resulting from analysis of data collection instruments. And the parameter text file can be maintained by non-programming personnel, conserving valuable resources without risk to data quality.

"Auto-editing" can easily be expanded and improved to accommodate new or different requirements. In concept it is limited mainly by the creativity and perseverance of the programming staff and available resources. On the technical side, it is important to recognize the differences between numeric and character data. Character data presents some special problems. But in general terms, the strongest limitation is the requirement for advanced planning and order. This limitation is common to most human endeavor.

In summary, the "Auto-editing" system makes life easier for the researcher by improving data quality through focused and timely information about the data collection process, conserving hard won resources, reducing the risk of data acquisition errors and encouraging good data management practice.

•••

SAS is a registered trademark of the SAS Institute, Inc. Cary, North Carolina, USA.

* Service Research, Inc. is located at 401 Hull Place, Rockville, MD 20852 FAX (301) 738-1041

Display 1.

Sample Program Listing

```

*--- Creates SAS program using Parameter Text File(s);
      libname save 'c:\auto'; * PC version ;

%MACRO AUTOEDIT(FORM);

  FILENAME VAR&FORM "&FORM" ;
  DATA VARS&FORM (KEEP=ITEMNUM VARNAME FRM LABEL);
    INFILE VAR&FORM;
    INPUT @1 VARNAME $8.
           @15 FRM $4.
           @23 LABEL $40.
           ;
    ITEMNUM + 1;
  RUN;

  %let nof="\auto\CODE&FORM" ;
  FILENAME CODE&FORM &nof ;
  DATA _NULL_ ;
    SET VARS&FORM END=EOF;
    FILE CODE&FORM;
    IF _N_ = 1 THEN DO;
      PUT @1'DATA CORR&FORM(KEEP=ID FRM MSG VARNAME );'
         / @3 'SET SAVE.&FORM;' /
         @3 'LENGTH MSG $ 65;';
    END ;
    * execute for all records/variables ;
    * simplified code only for numeric missing variables;
    PUT @3 'IF ' @7 VARNAME @20 ' = . THEN DO;';
    @1 'MSG = "QUES. '
    @14 LABEL @49 '**DATA IS MISSING*';'
    @5 'VARNAME = ' VARNAME '";'
    @5 'FRM = ' FRM '";'
    @5 'OUTPUT CORR&FORM; END;';
  RUN ;

  %INCLUDE CODE&FORM;
  RUN ;

  PROC APPEND BASE=AUTOERRS DATA=CORR&FORM FORCE;
  RUN;

  %MEND AUTOEDIT;

*--- Invoke the macro to get the work done
      (as many times as necessary);

%AUTOEDIT(F01A);
%AUTOEDIT(F01B);
%AUTOEDIT(F02A);
%AUTOEDIT(F03X);

*--- Add code here to generate desired reports
      from accumulated errors;

PROC SORT DATA=AUTOERRS;
  BY ID FRM MSG;
PROC PRINT DATA=AUTOERRS;
RUN;

*>>end of code<<*>

```

Display 2.

Parameter Text File Examples

1. One IF level, missing data:

Variable	Code	Form Label...	LogcVar Op Value.....
MOMQ02B	N	F07A Q02BMOTHER HAPPY WORKING AT HOME	MOMQ02A EQ 1

This modified variable text file, which could originally be created by a PROC CONTENTS, could mean "If MOMQ02A equals 1, then MOMQ02B should be answered".

2. One IF level, missing or incorrectly answered data:

Variable	Code	Form Label	LogcVar Op Value.....
MOMQ02B	R	F07A Q02BMOTHER HAPPY WORKING AT HOME	MOMQ02A EQ 1

An "R" in the code field could indicate that the "reverse logic" is also true. That is, not only if MOMQ02A equals 1 should MOMQ02B be answered, but also if MOMQ02A does not equal 1, MOMQ02B should not be answered. This can be coded so either message will be produced.

3. Multiple answers to one question:

Variable	Code	Form Label	LogcVar Op Value.....
CGNONE12	B	F05C Q12BTIMES UNREL CARGIVR AT HOME	MOMQ02A EQ 1
CGFEWAM	M	F05C Q12BTIMES UNREL CARGIVR AT HOME	MOMQ02A EQ 1
CGFEWAF	M	F05C Q12BTIMES UNREL CARGIVR AT HOME	MOMQ02A EQ 1
CGFEWEV	M	F05C Q12BTIMES UNREL CARGIVR AT HOME	MOMQ02A EQ 1
CGMEDAM	M	F05C Q12BTIMES UNREL CARGIVR AT HOME	MOMQ02A EQ 1
CGMEDAF	M	F05C Q12BTIMES UNREL CARGIVR AT HOME	MOMQ02A EQ 1
CGMEDEV	M	F05C Q12BTIMES UNREL CARGIVR AT HOME	MOMQ02A EQ 1
.	.	.	.
CGMNYEV	E	F05C Q12BTIMES UNREL CARGIVR AT HOME	MOMQ02A EQ 1

Another question type that was coded in one of the tests was that of multiple selection. An example would be:

- Q12. What time periods does an unrelated care giver stay in the home?
(Mark at least one response)
- a. Never
 - b. 1-3 days/week, morning
 - c. 1-3 days/week, afternoon
 - d. 1-3 days/week, evening
 - .
 - .
 - j. 6-7 days/week, evening

In this case there would be multiple variables related to one question, and your parameter file might look like the above example. The programming code would then be created to scan whether "B through E" were all blank and then produce an error message. That is:

```

If code = "B", create program logic: If "Variabl1 = . AND"
If code = "M", create program logic: "Variabl# = . AND"
and lastly,
If code = "E", program logic: "Varilast = . THEN DO;".

```

Display 3.

Ready-To-Mail Edit Report

National Child Care Study
Bethesda, Md.

11/20/92 p. 1

EDITING MESSAGES FOR RESEARCH FORMS
(Please reply within 30 days to DC)

Subject ID= 100104A

Interviewer ID = 02

Form	Question	Edit Message	Variable	Edit Response	Batch
F02A	Q11A DOES BABY SLEEP THRU SITTER?	DATA IS MISSING	SLEEP11A		11/12/90
F02A	Q23C CARE GIVER RELATED TO INFANT?	DATA SHD BE SKIPPD	RELA23C		10/02/90
F03D	Q06J FATHER FEEDS INFANT TIMES	DATA IS MISSING	FATH06J		09/14/90
F04B	Q56B MOTHER HOME ALONE WITH INFANT	ANSWER INAPPROPRIATE	ALON56B		12/30/90

Interviewer 02: _____
Date: _____

Display 4.

Question/Edit Verification Report

11/02/1992

p.1

Edit Totals by Form & Question

FORM: F02A

Question 12: Times CareGiver home with Infant

Edit: Data is Missing

SubjId	Res	Intrvwr	BatchDate
100323A	02	012	12/31/92
873214A	01	009	11/12/92

TOTAL: 23

FORM: F02A

Question 12: Times CareGiver Home with Infant

Edit: Answer should be skipped

SubjId	Res	Intrvwr	BatchDate
100323A	02	012	12/31/92
873214A	01	009	11/12/92

TOTAL: 16

FORM: F02A

Question 15: Nap Times for other Siblings

Edit: Data Out of Range

SubjId	Res	Intrvwr	BatchDate
100323A	02	012	12/31/92
873214A	01	009	11/12/92

TOTAL: 02