Using the SAS® System for Error Control in Clinical Trials Databases

William C. Murphy, Eastman Dental Center, Rochester, NY
Howard M. Proskin, Eastman Dental Center, Rochester, NY
Gary B. Freeman, The Hardardt Group, Bernardsville, NJ

ABSTRACT

In clinical trials, information is often collected via hard copy on case report forms, and must be entered into computerized databases prior to processing. The need to minimize errors in this transfer process is of the utmost importance if valid conclusions are to be obtained from the study. Consequently, a protocol for data entry and error detection must be established and strictly adhered to. The SAS DATA step can readily scan for illegal values as raw data is input. Duplicate data entry followed by the use of PROC COMPARE provides a way to locate discrepancies resulting from data-entry errors. Furthermore, PROC COMPARE can assist in providing a quantitative assessment of the number of questionable entries in the final data set. By using the SAS Macro language, it is possible to automate this process to accommodate almost any study protocol.

INTRODUCTION

In clinical trials, the information on each subject is often gathered on paper case report forms. Furthermore, several examinations over a period of time are gathered on each subject. In order to analyze these data, the information must be converted into computer form. As the study grows in size and scope, the possibility for errors expands greatly. However, the need to minimize error is of critical importance in order to obtain valid conclusions from the study. Consequently, a strict protocol for data input and error detection must be followed. Using the SAS system, the error management is a relatively straightforward process and it as the added advantage of readily providing the capability for later statistical analysis.

In order to assure data integrity, we must have a standard of comparison. Unfortunately on new data this standard does not in general exist. Consequently this necessitates the duplicate input of the data by at least two different people with no contact with each other. Different people are necessary, because the same person has the tendency to commit the same mistakes. It may be noted that an optical character recognition system could be used, but two different systems would be highly recommended since a given device also has a tendency to commit certain types of errors.

SETUP

From our two data input sources, raw data is loaded into different directories and assigned library names:

```
libname group_1 'c:\entries1';
libname group_2 'c:\entries2';
```

To establish the names of the files we wish to cross-check, we use a simple null DATA step with SYMPUT to create macro variables containing the desired data file name. Furthermore, to aid in identification of the data source we also include macro variables containing title information:

```
data _null_;
  length xfile $ 8 xtitle $ 20;
  retain index;
  infile cards missover eof=LAST;
  input xfile $ @10 xtitle & $;
  index=trim(left(_n_));
  call symput('file'||index,trim(xfile));
  call symput('head'||index,trim(xtitle));
  return;
  LAST: call symput('tot',index);
        return;
  cards;
exam1  First Examination
exam2  Second Examination
exam3  Third Examination
        .
        .
examN  Nth Examination
;
run;
```

where the macro variables &file1, &file2, etc. contain the names of the data sets we will be examining and &head1, &head2, etc. contain the associated title for out printed error listings. The macro variable &tot will contain the total number of data sets that are to be examined. It should be noted that the above DATA step is very general and could readily be used for any number and names of

files by changing the entries after the cards statement and the LENGTH statement for the title size variable, xtitle.

PRELIMINARIES

The most egregious errors such as values outside of a range and incorrect dates could potentially be avoided by using an input overlay and error checking scheme such as that provided by SAS/FSP® software. However this would necessitate the data entry people to have SAS on their computers. Our operation is confined to IBM®-compatible personal computers, many with limited memory and disk space. This necessitates the use of a more generic input program, usually a simple ASCII text editor such as QEdit® with very limited formatting and no error correcting capabilities. But it has the added benefit of letting the data entry people use software packages that they are comfortable with and work at their own rate, perhaps at off-hours on their home computers.

Consequently each set of the raw data must be separately screened for typographic and range errors. The most common typographic mistakes seem to be extra or too few spaces and carriage returns. These errors can readily be detected with the SAS LENGTH function. Lines that fall outside of a given length can then be printed for identification and correction. The macro program lines needed to check for length can be written:

```
%macro ruler;
  %do i=1 %to %tot;
      data long short;
        length linlen $ Big;
        infile &&file&i lrecl=Big;
        input line & $;
        linlen=length(line);
        if linlen>MaxLine then output long;
          else if linlen<MinLine then output short;
      proc print data=long;
        title3 "&&head&i";
        title4 "Lines Too Long;
      proc print data=short;
        title4 "Lines Too Short";
        run;
  %end;
%mend ruler;
```

where the 'Minline' and 'MaxLine' values are dependent on the data set being examined. The 'Big' value in the LRECL option of the INFILE statement and in the LENGTH statement must be made large enough to prevent the INPUT from truncating the lines of raw data. This macro routine must be run before any other tests are performed, otherwise any line length problems will lead to more errors. The %do loop allows us to scan all the different data sets.

Experience has shown that variable line size is a large source of potential error. Consequently a constant line size is recommended in the design of future studies. This can be assured by assigning a SAS missing value (other than blank) to any empty spaces to make all lines equal in length. This would also assure that the data was intentional left blank rather than be ambiguous like a space. A simple visual scan of the data for a ragged line edge could also identify illegal lines.

After assuring the lines are of a proper length, the raw data are converted into SAS data sets. The data step that creates the SAS file will readily identify any values that do not conform to the appropriate INFORMAT. These range errors will be reported in the SAS log. It may be advisable to change the ERRORS= option in the system OPTION statement to assure detection of all range errors.

Another common source of error is duplicate entries or completely missing entries for a given set of identification variables. These can also be identified by a simple SAS process.

ID ERRORS

To check for duplicate entries within a data set we write another routine also inside a macro loop:

```
%let idvar=id;
%macro idcomp;
  %do i=1 %to &tot;
    data only1(keep=&idvar) dup1(keep=&idvar);
      set group_1.&&file&i;
      retain oldid . ;
      by &idvar;
      if &idvar=oldid then output dup1;
        else output only1;
      oldid=&idvar;
    proc print data=dup1;
      title3 "&&head&i";
      title4 "Duplicate ID's (Group 1)";
      run;
```

where we have assumed that the data set is sorted by the identification variables defined on the first line. Again this program is very general with only the contents of &idvar changing for different applications. We would also run a similar check for duplicate information in the other entry group. The data set 'only1' is generated to help determined if there are subjects missing from both data sets.

Our next step would be to look for data missing from one data set. We can locate these by merging the different groups of data sets. For this cross checking, we add the following code to our macro loop:

```
    data in1not2 in2not1;
      merge lst1(in=test1) lst2(in=test2);
      by &idvar;
      if test1 and not test2 then output in1not2;
      if test2 and not test1 then output in2not1;
    proc print data=in1not2;
```

```
    title4 'ID in Group 1 but not in Group 2';
    run;
  proc print data=in2not1;
    title4 'ID in Group 2 but not in Group 1';
    run;
  %end;
%mend idcomp;
```

where our conditional statements have created two listing of the data that were found to be missing in one data set or the other. If possible, the identification variables should follow a simple sequence so that a check of data missing from both data sets could readily be performed.

COMPARISONS

Once the initial data sets are free from duplicates and missing data, we can move onto the next step: comparing the two data sets for inconsistencies. To do this the SAS COMPARE procedure offers an easy way to compare two data sets and write a report of the errors found. To utilize this procedure, we write another macro program around it:

```
%macro docomp;
 %do i=1 %to &tot;
   proc compare transpose nolistequal nosummary
     data=group_1.&&file&i compare=group_2.&&file&i;
     title3 "&&head&i";
     id &idvar;
     run;
   %end;
 %mend docomp;
```

Since we are only interested in spotting the individual discrepancies and to prevent too confusing an output, we use the NOLISTEQUAL and the NOSUMMARY options. The TRANSPOSE option causes differences between the two data sets to be printed out by the identification variables, thus readily pointing out the location of discrepancies.

The points of disagreement are then referred to the original data entry people. The exact discrepancy is not revealed to prevent bias. When the present set of problems are resolved, the data sets are than run through the same comparisons as before. This cyclic protocol is followed until the source of error is down to a group of errors that are judgement calls: the original forms were so difficulty to read that the best estimate or a blank was entered for this information.

When the errors in the database are reduced to this level, PROC COMPARE can be run without the TRANSPOSE and NOSUMMARY options. The output can then be used to estimate the possible errors remaining in the data sets for the different variables. A high error rate for any variable may not only force the removal of that data from the study but it can provide insights on why that variable caused trouble. For example, values from multiple choice questions are much less prone to error than from fill-in-the-blank questions .

LOOSE ENDS

The main program should have a title1 and title2 statement. It is recommended that they should refer to the company and the study to avoid mixing with other studies. Furthermore, make sure that the pages of output are dated to avoid confusing different runs of the same study. Finally, error checking more than any other programming demands keeping good documentation at all levels of the study.

CONCLUSIONS

Using multiple entries can help to minimize errors in inputting data from handwritten forms. The data values can readily be scanned for illegal values using the SAS DATA step. Merging duplicate data sets can be used to fine missing data. Comparison made with PROC COMPARE can zero in on further discrepancies in the data sets. After the data sets are reconciled as much as possible, PROC COMPARE will also provide an estimate of the error rate in the final database. Using the SAS macro language, the above error management protocol can readily be automated.

ACKNOWLEDGEMENTS

SAS and SAS/FSP are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

IBM is a registered trademark or trademark of International Business Corporation.

Other brand and product names are registered trademarks or trademarks of their respective companies.