Abstract

Although data entry and quality assurance systems have been well described, practical applications synthesizing such methods are not always obvious to the user. This paper describes an inexpensive data entry application using quality assurance tools for entering multiple data forms collected at different time points. Data entry was conducted in interactive mode on a mainframe (OS/MVS**) using SAS/FSP*, Version 5.18; data management was conducted in batch mode using Base SAS, Version 5.18 under ISPF**. Simple macros were developed and maintained in batch files for automatic sorting of databases, comparing databases for 'cleanup,' and updating master files. Double entry was used to assure data quality. This application is targeted for small academic research groups, requiring computer-intensive resources for entering, managing and analyzing data, especially for larger studies. This data entry application utilizes basic quality assurance tools, does not require programming sophistication, and can be managed by one part-time person using part-time data entry help.

A. Introduction

Data entry and data quality can be accomplished using basic quality assurance techniques without the need for sophisticated applications software. This basic application is based on previous SUGI papers (1-5) that dealt with general ideas and processes for developing data quality. This basic application has been developed resulting from those papers. A flow-of-control was developed to manage the paper forms received for entry; data entry was done in interactive mode using SAS/FSP; programs were developed to manage the data in batch mode.

B. Setting

1. Clinical Research Site

University Hospitals of Cleveland, an 874-bed teaching hospital, is the clinical site for the Hospital Outcomes Project for the Elderly (HOPE). The HOPE Study, funded by The John A. Hartford Foundation, is designed to study functional decline in elderly patients, hospitalized for an acute illness. The study is being conducted in two phases, the first of which has just been completed. The natural history of functional decline was defined in Phase I of this study by using health status instruments to collect data from 300 hospital patients, their surrogates, and nurses caring for the patients.

2. Resources and Objectives

University Hospitals has a mainframe computer operating under OS/MVS**, with Base SAS* and SAS/FSP* software. Information Services establishes mainframe accounts for users, with restrictions to using ISPF. Because CLISTs and JCL programming were restricted, programming was limited to the available software and batch file members under ISPF. The clinical research budget allowed for one half-time data manager and half-time data entry help.

The objectives for developing an application were simple: build screens for data entry to match the flow of data collection; develop a control system for managing the large volume of forms to be entered; develop a 'toolbox' of programs for assuring data quality. Previous SUGI papers (1-5) assisted in developing the application presented here.

C. Methods

1. The Data Forms and Collection Times

Data were collected on survey forms by trained data collectors (see Table 1 for summary). Surrogate data were collected as a substitute for patient data if a given patient was too ill or unavailable for an interview. Data were collected at hospital admission, day 2, and discharge. Follow-up data were collected 30 and 90 days after discharge. Once a subject was discharged and the forms were complete, the inpatient data were transferred for entry. When 30- and 90-day data
were complete, those forms were transferred for entry. If the patient was interviewed, there was a maximum of 11 forms to be entered; if the surrogate was interviewed instead of the patient, there was a maximum of 6 forms.

Each form has a tracking area on the upper left-hand side (Figure 1). Should any given form be removed for review at any point after the data have been collected, its entry status can quickly be determined. Entry fields are located in a column down the left-hand side of the form. Briefly, the data collectors sign off after collecting and reviewing each form. The entry fields are signed off by the data entry staff at each step in the entry process. "Audit 1" is signed off once the matched files have been compared and edited. The subject ID at the top of the form is the only indexed field coded for entry.

2. Data Entry System

Two accounts were set up for data entry and master files. The first account was used for data entry only; the second account was set up for management and analysis of the files. Having two separate accounts allowed for concurrent data entry and analysis. The overall data entry flow is summarized in Figure 2. Identical files were set up in the entry account for data comparisons and master files using Proc Compare; this was done by modifying Rosen's (1) methods for our files. When the data forms were received, they were divided up according to type (patient, surrogate type, nurse) and time period (admission, day 2, etc.) into individual folders marked 'Entry 1.' Each folder contained a plastic-covered data instruction sheet with the name of the target database and its corresponding screen. Operator 1 entered the forms into the working files, signed both entry logs and the 'entry 1' tracking field on each item, and transferred the forms into the 'Entry 2' folders. Operator 2 then entered the forms into the null data files, signed the entry logs and the 'entry 2' tracking field, and transferred the forms for review. Both data collection forms and data entry folders were color-coded by time of collection to prevent mistaking and erroneous data entry.

Data were entered in interactive mode and managed in batch mode. Screens were developed to match each data collection form. The data fields for entry were left-justified to facilitate keying in data column-wise. Special fields, such as time of collection and data completion flags, were automatically set in PROC FSEDIT and protected from user entry. Batch programs were developed and run in the management account to automatically sort data, scan data to assure unique subject IDs, and do logic checks on data gathered within each form.

3. Data Quality Assurance

Data quality was assured in three ways. First, field attributes were set up in Proc FSEDIT to classify minimum and maximum entry codes for each field as data were entered. Certain fields were also initialized and protected from entry. Secondly, because of the limited data checking capabilities of Proc FSEDIT, logic checks of data were done in batch files using Proc MEANS and Proc UNIVARIATE (4-5). Error trap programming in the data step was incorporated into entry status reports to flag errors early in the data entry process. Thirdly, double entry was used as described previously (1) and discussed further below.

Double entry was used as the main data quality assurance tool. Data were entered in "batches" of at least 20 subjects with complete inpatient and follow-up sections. File comparisons and updates were done in groups of 50.

Batch files were written as program members managed by ISPF for automatic sorting of all files. A macro %COMPARE (Appendix 1) was written to compare the working and null files for the data "cleanup" process. This 'user-friendly' macro could be invoked by a user by specifying the parameters at the top of the program, such as inclusive subject IDs, variables to be compared, other indexed fields. Default parameters were set for the macro but could be changed by simple editing at the top of the program. When the batch file was submitted, the subject ID groups of interest were compared at the record level. Any fields that did not match between comparison files were output in a report by record for each variable. The report was reviewed against the data forms, and the work files were edited for the update process.

A simple batch file for updating the master files (Appendix 2) was developed with macro variable strings to specify which data sets were to be updated and which subjects or records were to be included as transaction files. The program
invoking the macro was submitted and a report was generated as a record of the update transaction. The data forms were then signed off under ‘audit 1,’ and all paperwork was filed in an audit log notebook.

Data completion reports were generated by using subject IDs and data entry flags at regular intervals. This was accomplished by match-merging relevant databases by subject ID and unique data entry flags (Appendix 3) to determine the amount of data entered. Because the flags were defined as numeric fields, they could be manipulated arithmetically -- e.g., doing sums to ascertain the completeness of each database, group, or time period. Reports based on the 'entry 1' databases reflect the status of subjects entered; reports based on the differences between master and 'entry 2' databases reflect subject data awaiting database cleanups; reports based on the master databases reflect the amount of data entered for analysis.

D. Discussion

Using SAS software in the mainframe allows for the rapid turnaround of data that is critical to the success of this project. Because Information Services automatically backs up files on disk, we did not need resources to back up files. Entering and managing data on the mainframe has the advantage of rapid and direct data access for quality reviews and analysis. By using SAS software, our site was able to rapidly build simple data entry applications and a quality assurance toolbox. Separating entry files and master files facilitated data quality by restricting data analysis to master files only. Having two separate management accounts provided data security; Data could also be rapidly analyzed for abstracts and presentations without disrupting data entry; this was particularly important given the fact that there was always a large volume of data to be entered. The final databases can be easily accessed for future long-term analyses for research papers and further clinical studies. Because of the statistical procedures available in SAS, the multivariate procedures used by this site, and the fact that databases are readily available for analysis, the final databases will be maintained in the mainframe.

Double entry has been demonstrated to be one of the best tools for quality assurance (1). Although double entry requires twice the data entry time, it does not have to be expensive. Temporary help can be hired for doing data entry; our research group hires students for this job because they learn quickly and work very enthusiastically. A well organized paper flow system can more than compensate for applications development sophistication. The system has error traps for managing potential problems, such as a repeated indexed field, or data entered into the wrong files. Proc Compare will issue error messages if there are problems with the indexed fields. Our methods for updating master files employ cumulative data comparisons to make sure master files have not been changed. The entry staff do not have to learn a whole application or use complex codes to access databases. Our overall rate of entry error has been less than 0.1%. Finally, all databases are evaluated using logic checks, range checks, and outlier checks in order to flag data forms requiring audits.

The simplicity and flexibility of this application allow it to be easily integrated into future studies, or moved to a PC system for data entry. With the Version 6 enhancements available in SAS/FSP, internal data checks can be performed at the entry level instead of through external programs. Overall, Version 6 will provide more opportunities to improve this application for supporting future studies requiring data entry control and rigorous quality assurance.

Bibliography

I would like to thank Barbara Juknialis and Seth Landefeld for their critical reviews.

The author may be reached at:

Table 1. Data groups and times of collection

<table>
<thead>
<tr>
<th>TIME OF COLLECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Admission Day 2</td>
</tr>
<tr>
<td>PATIENT</td>
</tr>
<tr>
<td>NURSE</td>
</tr>
<tr>
<td>SURROGATE</td>
</tr>
<tr>
<td>SURROGATE SUBSTITUTE *</td>
</tr>
<tr>
<td>Total forms</td>
</tr>
</tbody>
</table>

* Surrogate data were gathered if the patient was too ill or unavailable for interview.

Figure 1. Data Form Tracking Area

FOR OFFICE USE ONLY

<table>
<thead>
<tr>
<th>ID</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ACTIVITIES OF DAILY LIVING (CURRENT)

Introduction: We would like to ask some questions about how (the participant) was taking care of himself/herself on the day he/she was admitted to the hospital (or transferred to this room).

NOTE: IF PARTICIPANT HAS BEEN TRANSFERRED TO THIS UNIT FROM A CRITICAL CARE UNIT, ASK THE QUESTIONS AS: "ON THE DAY THE PARTICIPANT WAS TRANSFERRED TO THIS UNIT..."

1. On the day (the participant) was admitted to the hospital, did s/he need help bathing, either a sponge bath, tub bath, or shower? Would you say s/he needed...

   1 = no help
   2 = help
   3 = unable to do
   7 = don't know
   0 = missing data

1343
Figure 2. Data Entry Flow-of-Control and Double Entry

**Data Entry Account**

[first folder set]  [second folder set]

ENTRY 1  ENTRY 2

ENTRY LIBRARY

<----> data monitoring sorts, inspection, status reports

----> data entry logs to audit files

GROUP OF SUBJECTS WITH COMPLETE ENTRY

COMPARE FILES

(See Appendix 1)

REPORTS ----> audit logs

Errors  No Errors

EDITS <-----

**Master File Account**

UPDATE FILES

(See Appendix 2)

MASTER LIBRARY

----> STATUS REPORTS

(See Appendix 3)
Appendix C. Programming to update master files with "down" files.

OPTIONS AUTOSOURCE DUOQUE IMPLICIT SYMBOLOUSH MACROGEN;
* ---------------------------------------------;
* UPDATE;
* GENERIC UPDATE MEMBER FOR UPDATING ALL MASTER DATABASES;
* A MACRO TOOLBOX EXISTS TO ALLOW FOR EDITING AT THE TOP;
* ---------------------------------------------;
* MACRO TOOLBOX: EDIT HERE;
* %LET DISH = LIB1.master;
* %LET DISH = LIB2.workfiles;
* %LET DISH = LIB3.backup;
* %INDEXED FIELDS;
* %DEFAULT IDS = SUBJECT;
* %LET IDS = SUBJECT CODE;
* 1. FIRST and last subject id for transaction file;
* %LET FIRSTID = 1;
* %LET LASTID = 300;
* DO NOT CHANGE DISH;
* %LET DISH = TEMP;
* %DEFAULT PARAMETERS: DO NOT CHANGE HERE;
* 1. IMAGE COPY BACKUP OF MASTER DATA;
* DATA &DSN3;
* SET &DSN3;
* PROC CONTENTS DATA=&DSN3;
* TITLE "BACKUP OF MASTER DATA &DSN1 PRIOR TO UPDATE";
* 2. CREATE TRANSACTION FILE;
* DATA &DSN4;
* SET &DSN4;
* IF &FIRSTID < SUBJECT < &LASTID;
* 3. UPDATE THE MASTER FILES;
* DATA &DSN5;
* UPDATE &DSN3 &DSN4;
* BY &IDS;
* PROC CONTENTS DATA=&DSN4;
* TITLE "SUMMARY REPORT OF UPDATED MASTER FILE &DSN4";
* 4. REPORTS OF UPDATE PROCEDURE FOR THE AUDIT LOG;
* %COMPARE THE ORIGINAL MASTER DATABASE AGAINST THE WORKING FILE;
* PROC COMPARE DATA=&DSN3 COMPARE=&DSN4 HOEQUALEQUAL CRITERION=0
* METHOD=PERCENT; ID &IDS;
* TITLE "COMPARISON REPORT OF MASTER DATA FILES WITH WORKING FILES";
* FOOTNOTE "MASTER FILE: PRE-UPDATE";
* COMPARING THE MASTER DATABASE WITH THE WORKING DATABASE;
* PROC COMPARE DATA=&DSN1 COMPARE=&DSN4 HOEQUALEQUAL CRITERION=0
* METHOD=PERCENT; ID &IDS;
* TITLE "COMPARISON REPORT OF WORKING DATABASE WITH WORKING FILES";
* FOOTNOTE "MASTER FILE POST-UPDATE";
* PROC PRINT DATA=&DSN1;
* VAR SUBJECT DAY;
* TITLE "&DSN1: REPORT OF CURRENT MASTER FILE POST-UPDATE";
* PROC PRINT DATA=&DSN1;
* VAR SUBJECT DAY;
* TITLE "&DSN1: TRANSACTION DATABASE";

Appendix 1. Macro %COMPARE for comparisons of data files.

OPTIONS AUTOSOURCE DUOQUE IMPLICIT SYMBOLOUSH MACROGEN;
* ---------------------------------------------;
* COMPARE;
* DATASET COMPARISONS FOR DOUBLE ENTRY METHODS;
* %ASSUME 1-11-96;
* ---------------------------------------------;
* MACRO TOOLBOX: EDIT HERE;
* %LET SOURCE1=LIB WORKFILE;
* %LET SOURCE1=LIB NULLFILE;
* %FIRST observation default;
* %LET FIRST=1;
* ---------------------------------------------;
* SECTION 2: OPTIONS;
* 1. KEEP STATEMENTS: DEFAULT TO ALL VARIABLES (ALL);
* 2. INDEXED FIELDS: DEFAULT SUBJECT ID (SUBJECT);
* 3. FIRST AND LAST SUBJECTS: DEFAULT FIRSTID=1 LASTID=300;
* 1 option to change keep variables;
* %LET KEEPS=ALL;
* 2 indexed fields;
* %LET IDS = SUBJECT CODE;
* 3 first and last subject id to be compared;
* %LET FIRSTID=300;
* %LET LASTID=500;
* %DEFAULT PARAMETERS: DO NOT CHANGE HERE;
* ---------------------------------------------;
* DEFINITION OF THE MACRO;
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* PROC COMPARE DATA=DSN1 METHOD=PERCENT ID &IDS;
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* PROC COMPARE DATA=DSN1 METHOD=PERCENT ID &IDS;
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEPS);
* MACRO %COMPARE(DSN1=DSN2 &KEEP=KEEP

Appendix 2. Program code for entry status reports using system flags.

DATA PATIENT;
  WORK(WORK) KEEP=ID FLAG() P72(KEEP=ID FLAG2) P73(KEEP=ID FLAG3);
  BY ID;
  LABEL ID=SUBJECT FLAG1= ADMISSION FLAG2= DAY 2 FLAG3= DISCHARGE;
  PROC PRINT DATA=PATIENT;
  TITLE "ENTRY SUMMARY REPORT OF PATIENT DATA FROM ADMISSION TO DISCHARGE";
  PROC MEANS SUM NOPRINT MAJOR=1;
  VAR FLAG FLAS;
  OUTPUT OUT=TOTAL SUM=FLAGSUM TOTALSUM FLAGSUM2 FLAGSUM3;
  TITLE "REPORT OF COMPLETE SUBJECT DATA";
  TITLE "PERCENT PATIENT DATA";
  TITLE "PERCENT PATIENT DATA";
  TITLE "PERCENT PATIENT DATA";
  TITLE "PERCENT PATIENT DATA"

Note: Flags are automatically set to a numeric 1 when data is entered and compared for each form. Otherwise, the flag is set to missing. Having numeric flags allow for sorting each group of data to determine data discrepancies.