

A Not So Rapidly Developed FSEDIT® Application or What a Non SAS® Programmer Can Do with SAS®

Charlotte R. Wenger

Department of Medicine/Division of Medical Oncology
The University of Texas Health Science Center at San Antonio

This FSEDIT application was developed by a SAS end user in order to save time and to increase the productivity of the Oncology Research Network (ORN). In the ORN we follow over 45,000 breast cancer patients and travel to 36 health care institutions around the country each year to obtain data. Previously the data manager would print data forms, fill them out on location, return to her office and enter the data in the database, verify her data entry, and file the data forms. With 300-800 data forms per trip and multiple data items per form, you can imagine that this was time consuming. We now carry an electronic copy of the data on our Macintosh PowerBook and use an FSEDIT application to directly verify and update cases.

The ORN database is stored and managed using Informix on a Sun SPARC server. In preparation for a trip, we run an SQL query in Informix which unloads data for the cases from a particular institution to a flat file and then "Fetch" the data to the PowerBook.

Data are then loaded into a SAS data set.

```
libname scsug 'Macintosh HD:sas';
data scsug.demo;
  infile 'Macintosh HD:sas:demo.dmi' dlm='|' dsd missover;
  input accno client name :$20. er pgr dna1 dna2 dna3 sphase bsite :$25. comment :$25.
        sdate deadend ptname :$25. lfmon lfyr recmon recyr dmon dyr cause status
        dob byr age menst race tsize exnode posnode surgmon surgyr histo surg trmt
        primet hosp_id :$20. doctor :$20. de dc cdate;
  format sdate de dc dob cdate mmddyy10.;
  informat sdate de dc dob cdate mmddyy10.;
run;
```

The **format** and **informat** are used by **FSEDIT** when the dates are displayed or new dates are entered.

The original data set is copied to another library to serve as a backup of the original and to compare to the updated data set after the trip to produce a record of what changes were made.

```
libname scsug1 'Macintosh HD:sas:crw';  
proc copy  
    in = scsug  
    out = scsug1  
    memtype = data;  
select demo;  
run;
```

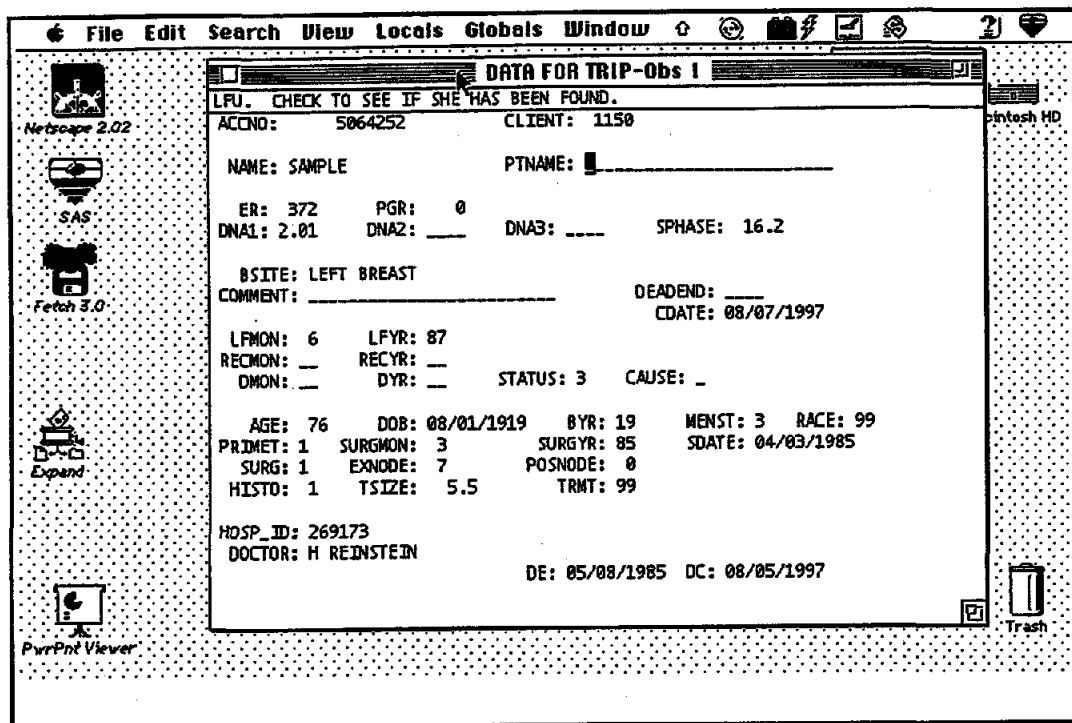
Finally, the data set is sorted to match the order that will be encountered at the hospital, for example, alphabetical by name.

```
proc sort  
    data=scsug.demo;  
by name;  
run;
```

The data manager is now ready to run the **FSEDIT** application.

```
proc fsedit  
    data=scsug.demo  
    screen=scsug.display.trip9.screen;  
run;
```

PROC FSEDIT is a way to view and update a SAS data set one record at a time. The default display of the data set can be customized and each data entry field can have a set of special attributes. A Screen Control Language (SCL) program can also be included that will perform certain tasks based on the data entered. The screen entry specified in the **screen=** option of the **FSEDIT** statement is where the customized information is stored. An illustration of our PowerBook screen and customized **FSEDIT** window is shown.



Our application uses some of FSEDIT's special field attributes to minimize data entry errors. The protect attribute is used with some fields to prevent them from being updated. The cursor skips over the protected fields so that they can not be changed. Other fields have designated minimum and maximum values. If a value is entered that is out of the specified range, an error message appears and it is not possible to move to another record until a valid value is entered. We also specify which fields should be in upper case. The data can be entered in lower case and it will change to upper case when the return key is pressed.

SCL is a programming language that can be used in FSEDIT to manipulate data and control windows. Statements are grouped into "sections" and control how the program is executed. Each section begins with a reserved SCL label and ends with a return statement. The 5 sections and their functions are as follows.

FSEINIT: Statements in this section are executed when FSEDIT is invoked, before the FSEDIT window opens.

INIT: Statements in this section are performed for each record, before it is displayed.

MAIN: Statements in the main section are executed each time the user modifies a field and presses the ENTER key.

TERM: Statements in this section are executed just before the next record is displayed.

FSETERM: Statements in this section are executed when the user ends the FSEDIT session.

We use SCL to make the application user friendly. The code in the FSEINIT section puts a name in the FSEDIT window header. The name, along with the observation number, are displayed throughout the FSEDIT session.

```
FSEINIT:
    call wname ('DATA FOR TRIP');
return;
```

The code in the INIT section causes a message to appear in the message line when each new record is displayed. The message reminds the data manager how to handle the case.

```
INIT:
    if deadend >= 0 and deadend < 20
        then _msg_ = 'DO NOT UPDATE. DEADEND.';
    if status = 2
        then _msg_ = 'DO NOT UPDATE. DEAD.';
    if status = 3
        then _msg_ = 'LFU. CHECK TO SEE IF SHE HAS BEEN FOUND.';
    if (deadend = . or deadend = 500) and (status = 1 or status = 4)
        then _msg_ = 'PLEASE UPDATE.';
    if (deadend = . or deadend = 500) and (lfmon = .)
        then _msg_ = 'PLEASE ADD DATA.';
return;
```

The MAIN section of the program causes date fields (dc, cdate, de) that show when a record was entered or changed to be updated when certain fields are updated. This prevents the data manager from having to update these fields manually.

MAIN:

```
    if field('modified','lfmon lfyr recmon recyr dmon dyr status cause age byr menst primet
            surgmon surgyr surg exnode posnode histo tsize trmt hosp_id doctor')
        then dc = today();
    if field('modified','name bsite comment deadend')
        then cdate = today();
    if modified(lfmon) and de = . and (deadend = . or deadend = 500)
        then de = today();
return;
```

We do not have any code in the TERM and FSTERM sections.

After returning from a trip, the data manager runs another PROC SORT to put the data set in patient accession number order. The data are then put in the format necessary to load them in to the Informix database.

```
libname scsug 'Macintosh HD:sas';
data _null_;
set scsug.demo;
file 'Macintosh HD:sas:demo.unl';
put accno +(-1) '|' client +(-1) '|' name $20. +(-1) '|' er +(-1) '|' pgr +(-1) '|' dna1
    +(-1) '|' dna2 +(-1) '|' dna3 +(-1) '|' sphase +(-1) '|' bsite $25. +(-1) '|'
comment $25. +(-1) '|' sdate +(-1) '|' deadend +(-1) '|' pname $25. +(-1)
'|' lfmon +(-1) '|' lfyr +(-1) '|' recmon +(-1) '|' recyr +(-1) '|' dmon +(-1) '|'
dyr +(-1) '|' cause +(-1) '|' status +(-1) '|' dob +(-1) '|' byr +(-1) '|' age
+(-1) '|' menst +(-1) '|' race +(-1) '|' tsize +(-1) '|' exnode +(-1) '|' posnode
+(-1) '|' surgmon +(-1) '|' surgyr +(-1) '|' histo +(-1) '|' surg +(-1) '|' trmt
+(-1) '|' primet +(-1) '|' hosp_id $20. +(-1) '|' doctor $20. +(-1) '|' de
+(-1) '|' dc :+(-1) '|' cdate +(-1) '|';
format sdate de dc dob cdate mmddyy10.;
run;
```

After the data are loaded in to the database, a series of Informix SQL quality control queries are run. If there are any problems, the SAS data set is corrected using FSEDIT, and the data are reloaded from the corrected data set.

Finally, the original data set is compared to the final corrected data set to produce another data set (demodiff) consisting of the original and changed data. Brief and transpose are options that control the detail in printed reports. The outnoequal, outbase and outcomp options specify the content of the output data set.

```
libname scsug 'Macintosh HD:sas';
libname scsug1 'Macintosh HD:sas:crw';
  proc compare
    base = scsug1.demo
    compare = scsug.demo
    out = demodiff
    brief transpose outnoequal outbase outcomp;
      by accno;
      id accno;
run;
```

The data in the resulting data set (demodiff) are written to a flat file with a **put** statement in a simple format we want for printing. For each record that has at least one changed field, the original record is printed on three lines followed by the changed record.

This paper has detailed an FSEDIT application developed by an end user with a limited knowledge of SAS. Although this first working version is relatively simple, it has met our goals of saving time and increasing the productivity of the Oncology Research Network. As my knowledge increases, I hope to write more SCL code to include window regions with legends for our coded data. I also want to develop an EIS for showing our clients their data graphically and for showing survival curves.

Documentation used to Develop the Application

SAS Institute Inc., *SAS[®] Screen Control Language: Usage, Version 6, First Edition*, Cary, NC: SAS Institute Inc., 1991. 690 pp.

SAS Institute Inc., *SAS® Screen Control Language: Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc., 1990. 464 pp.

SAS Institute Inc., *SAS/FSP® Software: Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc., 1989. 451 pp.

SAS Institute Inc., *SAS® Language: Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc., 1990. 1042 pp.

SAS Institute Inc., *SAS® Procedure Guide: Version 6, Third Edition*, Cary, NC: SAS Institute Inc., 1990. 705 pp.

Acknowledgments

I would like to thank Susan G. Hilsenbeck, Ph.D. for her constant willingness to teach me more SAS and for her comments and suggestions regarding this paper.

The Oncology Research Network is supported in part by NIH Grant CA30195.