Counting Observations in a PROC FSEDIT Subset
John Paul Elrod, Johns Hopkins University

INTRODUCTION

When using the WHERE command to display a subset of data in PROC FSEDIT or PROC FSBROWSE, there is normally no way to know how many observations are in the subset you are viewing without scrolling through all the records. The techniques shown in this Code Corner paper provide a way to simply issue a COUNT command to have SAS/FSP return a note on the message line saying, for example,

Command ===> count
NOTE: 524 Observations are in this Subset.

This SAS/FSP Screen Control Language program implements two custom commands that facilitate the display of an observation count when using a WHERE-like command to subset the data set.

SCL Code

The first step in the code is to issue the CONTROL ALWAYS command, which is needed to create any custom command in SAS/FSP.

The next step is to open a second data set identical to the one that is being displayed by the SAS/FSP procedure. The FSEINIT and FSETERM sections contain the code to open and close the second data set. To do this, the SCL OPEN function must know the name of the data set that is currently being displayed. Because there is no SCL function or method for determining the name of the data set being used by the FSP procedure, the data set name must be either passed to the SCL program via a macro variable (shown in the example) or hard-coded in the SCL program.

%let dsname = sample.data;
proc fseedit data = sample.data
   screen = sample.screen.fseedit.screen;
run;

The third step is to create a custom command called SUBSET. The WORD function and a DO loop are used to detect and process a SUBSET command issued by the user. The syntax of the SUBSET command is identical to the SAS/FSP WHERE command with one important exception: everything after the word SUBSET (the WHERE condition itself) must be enclosed in double quotes, for example,

Command ===> subset "total = 500"

instead of

Command ===> where total = 500

This makes it possible to easily parse the entire WHERE condition as one "word" using the WORD(2) function. The WHERE condition is then appended to the word "where" and stored in a temporary variable, wclause. The CALL EXECCMD function is used to pass the contents of wclause back to SAS/FSP for processing against the data set being displayed. Next, a WHERE function is used to apply the same WHERE condition to the secondary data set opened in the previous step, and another CALL EXECCMD passes the COUNT command to SAS/FSP so that the count will be automatically displayed after the user issues a SUBSET command, if the command was successful.

The code for the COUNT command simply determines if a WHERE clause is in effect for the secondary data set. If it is, the code uses a DO WHILE loop to actually read and count all observations that are in the subset and displays the note to the user via the _MSG_ variable. Note that it is actually the records in the secondary data set that are counted, not the data set being displayed.

Usage

The following examples show how the syntax of the SUBSET command mimics the WHERE command. Note that character constants in the WHERE condition must be enclosed in single quotes to avoid problems with the WORD(2) function.

Command ===> subset "quantity = 2000"
Command ===> subset "also status = 'Open'"
Command ===> subset "undo"
Command ===> subset "updated > '01JAN92'd"
Command ===> subset "clear"

Performance

Depending on the size of the data set and the size of the subset, the process of counting the records may take from 1 to 60 seconds. Performance can be greatly increased, however, if the data set and the variables used in the WHERE condition are indexed.

Caution

One word of caution about this procedure is necessary. In order for this to work properly, the SUBSET command must always be used to process subsets instead of the SAS/FSP WHERE command. This includes clearing the clause with a SUBSET "CLEAR" command. If any combination of WHERE and SUBSET commands are used at the same time, the data set being displayed and the secondary data set used by the SCL program will become "out of sync" and the user will be confused by misleading messages from the COUNT function.
command. The way we have handled this problem is simply to not tell our users about the SAS/FSP where command and to create documentation for the subset command similar to the SAS/FSP where command documentation. If, however, the program gets out of sync, the user can usually clear the problem by issuing the following command on the command line:

```
where clear; subset "clear"
```

CONCLUSION

The ability to know how many observations are in a subset has been a feature much appreciated by our end users. This algorithm has been successfully used at Johns Hopkins for the past two years. The code can easily be added to any existing SCL program.

AUTHOR

John Paul Elrod
Johns Hopkins University
300 Whitehead Hall
Baltimore, Maryland 21218
(410) 516-8732
jpe@jhuspo.ca.jhu.edu

SASe and SAS/FSP® are registered trademarks of
SAS Institute Inc., Cary, North Carolina

```
length wclause  $ 70;
length count   5;

PSEINIT:
  control always;
  dname = symget("dname");
  dsid  = open(dname,"r");
  return;

INIT:
  return;

MAIN:
  if word(1,"u") = "SUBSET" then do;
    wclause = "where " || word(2) || ";
    execmd(wclause);
    sysrc = where(dsid,word(2));
    if sysrc = 0 then call execmd("COUNT;");
  end;

  if word(1,"u") = "COUNT" then do;   /** COUNT command **/
    sysrc = rewind(dsid);
    insubset = attrn(dsid,"insubset");
    if ~ insubset then do;
      numobs = attrn(dsid,"nobs");
      count = trim(left(put(numobs,5.,1.)));
      msg_ = "NOTE: " || count || " Observations are in this File."
    end;
    else do;
      counter = 0;
      do while (fetch(dsid) ^= -1);
        counter + 1;
      end;
      count = trim(left(put(counter,5.,1.)));
      msg_ = "NOTE: " || count || " Observations are in this Subset."
    end;
  end;

  return;

TERM:
  return;

PSETERM:
  call close(dsid);
  return;
```