A Macro Tool for Quickly Producing a Handy Documented Listing of SAS® Data Sets for Use as a Reference While Writing Programs to Analyze the Same

Hugh Geary, Neoprobe Corporation, Dublin, Ohio

Abstract

When beginning a new project I often have a number of SAS data sets with a total of, perhaps, 200 or more variables to analyze. Appreciable time can be saved when developing programs for processing this data if listings can be made which enable me to scan through the data quickly.

This paper describes a macro I wrote to make listings both informative and compact. The title line has the name of the file in large print followed by the number of records, the number of patients (or values of the major sort key), the date and time the file was last modified and page number.

The listings are done by PROC PRINT and are made compact by using small print and printing only as many variables as can be displayed on a page without creating a *split* page. When a character variable is so long that it will not fit in one page width, my macro automatically breaks it into two variables.

Control parameters for PROC PRINT features and other features are passed through macro variables. A PROC CONTENTS listing may also be produced.

Introduction

Why Make Listings?
When I move to a new database I need to both get the 'feel' of the data, i.e. its range, number of digits in the numbers and how much is missing, and have a source for looking up data values, such as for checking on outliers or missing data. Creating listings helps in getting familiar with the data and provides a valuable reference for my own use in the weeks ahead. The listing is also a good place to make notes about data peculiarities as they are discovered because from them sometimes a trend may be identified over time. If the data set name can be printed very close to the top edge of the paper, several listings can be bundled together by a large clip in stair step fashion so that the titles of each of the listings can all be seen at once increasing your ability to find a listing quickly.

Listings Are Best Made by Macros.
Because making listings is somewhat of a repetitive process, a macro, once it is written, saves duplicating the code and modifying it repeatedly. Also when a new feature is thought of, changing the macro is easier than going back through many separate codes.; in fact using a macro encourages new features because they can be more easily incorporated into the previously written macro calls than in separate programs.

Title Lines

The Title Line Tells about the SAS Data Set.
On the title line I try to put all of the information needed about the file itself - a project identifier, file name, number of records, number of patients, number of records actually listed if it is not a complete listing, the date the SAS data set was last modified and a page number. Printing the file name in large bold print may require embodying special control characters in the title for the printer to read such as I have done while using a VAX computer and certain printers. Counting the records, *nobs*, can be done without reading the data set. While counting the patients requires sorting and reading the data set, it is very useful to be able to see at a glance how many patients are actually in the data set, or, in effect, how many *don't have* any data. The date the data set was last modified is found in the output SAS data set of PROC CONTENTS. This date can be put into a macro variable with a symget statement and then put in the title statement. If an options statement:

```
options number nodate;
```

is used the page number will be printed automatically and the date and time of printing suppressed (suppressed when the date last modified is desired here.)

A second title line can be made by assigning whatever text is desired in it to a macro parameter called 'subt'; For example,

```
  subt=%str(Part 1 - the Lung Related Variables);
```

%str ( ) is needed only if the string contains certain characters.

The Listing

Prevent Wrap-around Data Listings.
PROC PRINT is used by my macro, PRNT, to do the printing. The main concern in making the printout easy to read is in preventing the procedure from having to wrap long records around in order to get the whole record on the page. Usually this is accomplished by using two or more PROC PRINT calls giving each one only as many (but preferably the maximum number of) variables as it can fit on a page without wrapping. Sometimes a character variable may be so long that it will not fit across the page. I use 20 characters to the inch when printing listings and with paper 8.5 inches wide it gives me about 160 columns to use and occasionally this is not enough. In this case I break the long variable into two (or more) substrings with another macro, called SHORTEN, which is used for only this purpose.

How to Split a Long Variable.
Actually the macro, SHORTEN, does this for you automatically. This macro reads the output SAS data set of PROC CONTENTS and fetches the names of all of the variables declared to have a length of greater than 8. Some may never attain or not even come close to their declared length. and for our listing we only care about the length actually attained. Then it reads the data set about to be listed, noting the actual length of each of these variables in each record and retains the maximum length of each variable. Then for each variable attaining a maximum greater than, say, in my case, 150, it creates two or more new variables, an A version, a B version and so on as needed and drops the original. Thus if variable, called say, reason, had actually attained a length of 190 then the following would be done:

```
   mnx=min(length(reason),150);
   reasonA = substr(reason,1,mnx);
   If mnx>150 then
     reasonB = substr(reason,151);
```

Reason will be dropped from the listing and replaced by reasonA and reasonB;

How to split up a data set's variables over more than one print call.
The VAR statement specifies which variables will be included in the printout. With no VAR statement the data set would be printed wrapping around if necessary. One macro parameter, called var, is used in the following way to create a var statement:

```
  var=var V1 V2 V3 ... Vn;   as in  var=var patient
date age sex race height weight;
```

where V1, V2 etc would be the actual variables names. The first var is the name of the macro variable and what follows the equal sign is the actual var statement.
Use two or more calls to the macro, PRNT, specifying different variables on the var statements in each call but only as many as will fit across one page of output. Trial and error tells you how many can be crammed across one page although a good guess can be made by making a 'first look' run by PRNT. To do this use:

```
  %prnt (labs,obs=zz,npag=1);
```

which produces a one page listing with as many observations as PROC PRINT can put on a page. The observations will wrap as necessary. If a number is assigned to obs instead of "zz" then PRNT will print that many observations regardless of how many pages it takes. If obs is assigned a null value then all of the observations are printed but if "zz" is assigned then PRNT needs to use the value of npag to see how many pages to fill and then compute how many observations would be required. Writing several one page calls to PRNT enables me to quickly see what kind of data is in the database.

Other Features

Explaining the Other Macro Parameters.
The macro parameters explained so far have been: var and subt. An example of a macro call using all of the possible macro parameters is:

```
  %Lister (medhist,
            subt=Part One - Respiratory Data,
              by=patient  labdate,
              id=patient,
             dso=keep=patient labdate pulm lung1
lung2 lung3 resp aero1 aero2 reason,
             var=var labdate pulm lung1 aero1 resp,
            cnts=no,
             opt=uniform,     Or   opt=noobs split,
and other PROC PRINT options
              ps=90,
            npat=);
```

The first parameter is the only positional parameter and it names the SAS data set to be printed.All of the other parameters are keyword parameters and so may be listed in any order or omitted.

Subt is for a subtitle. *By* names the sort variables for use by PROC SORT. *Id* names the variable(s) to be used in the special by - id construction:

```
   by sort-key;  id sort-key;
```

which separates the records belonging to one sort-key value from records belonging to other sort-key values by a blank line, and printing the sort-key only once.

Dso is for a data set option, such as, keep, drop, where or whatever else might be useful. It is used in the proc sort statement and used mainly to eliminate variables or records that are not wanted for some reason.

Var, as explained earlier, specifies the variables to be printed by PROC PRINT.

If *cnts* is left blank then PROC CONTENTS will print its usual output, otherwise none will be produced. This is useful when several executions of the macro are necessary to print all of the variables of one data set.

Opt supplies whatever options it is desired that PROC PRINT use. *Ps* specifies the page size and appears on an options statement;

Npat would normally be left blank and thus cause the macro to count the number of patients in the SAS data set. However there is often at least one data set in the database that does not contain the variable used as the major sort key in all of the others, in this case, set npat= to any non-blank character, as npat=x, so it will not try to count what is not there.

Hugh Geary
Neoprobe Corporation
630 Morning Street
Worthington, Oh 43085

(614) 793-7500 ext 168 (work)
(614) 793-7520 (fax)
(614) 885-3164 (home)
e-mail hgeary@neoprobe.com

APPENDIX
Complete Macro Listing

```
*PRNT.sas  SUGI22                                              2jan97,23may96++;
*************************************************************************;
*\ Program   PRNT macro
*\ Purpose   To print a proc contents output followed by a complete and
*\           tailored listing of selected data  sets;
*\ progmr    Hugh Geary  May, March 96
*************************************************************************;
options nofmterr ls=132 ls=160 ps=90 ps=126  date number missing=' ';
* Use Courier New 6 pt ls=160 ps=126;

%macro Shorten (dsn);  * or ChopAB;   * Find how many LONG var there are;
                                    * 150 is OK if it FITS on one line;
* Formats will be recomputed for lengths over 8;
* Vars >150 will be chopped in two. The A version  of length=150,
  the B version with the remainder;

proc contents data=&dsn  noprint  out=contnts (where=(length>8));

data _null_; if 0 then set contnts nobs=nn;
  call symput ('nn',trim(left(put(nn,2.))));       * assuming fewer than 100;
  if nn>0 then do;   i=0;
    do until (last);
      set contnts  end=last;  i+1;                 * _n_ not avaliable here;
      call symput ('long'||trim(left(put(i,2.))),name);  * get the var names;
      end;
    end;
  stop;  run;

 %put  The number of variables of length > 8 is:  &nn &nn;

 %if &nn>0 %then %do;              * i.e. if some LONG var were actually found;
  data _null_; set &dsn  end=last;     %* how many actually ever exceed 150 ?;
  retain mm1-mm&nn;
  if _n_=1 then do;  %do ii=1 %to &nn;  mm&ii=0;  %end;   end;
  %do  ii=1 %to &nn;  mm&ii=max(mm&ii,length(&&long&ii));  %end;

  if last then do;
    jj=0;  kk=0;                    * Find the actual greatest length found;
    %do ii=1 %to &nn;               * in the whole file of each variable;
      if mm&ii>150 then do;
        jj=jj+1;
        jjchar=trim(left(put(jj,2.)));
        call symput('j'||trim(left(put(jj,2.))),"&ii"); *saving the old index;
                                                 * Next get length of var name;
        if length("&&long&ii")>7
          then call symput('long7'||jjchar,substr("&&long&ii",1,7));
          else call symput('long7'||jjchar,trim("&&long&ii"));
        call symput("mmA&ii",'150');
        call symput("mmB&ii",trim(left(put(max(mm&ii-150,2),3.))));
        end;
      else do;
        kk=kk+1;               * save the index of those not actually so long;
        call symput('k'||trim(left(put(kk,2.))),"&ii");
        call symput("mm&ii",trim(left(put(mm&ii.,3.)))); * storing actual lengths;
        end;
      %end;
    call symput('jj',trim(left(put(jj,2.))));  * store last value of jj & kk;
    call symput('kk',trim(left(put(kk,2.))));  * jj + kk = nn;
    end;
  run;
  * Create the A & B variables now (A second pass of &dsn is required.);
  * Append an A to the name & give it length 150;
  * Append a  B to create name for the remainder and drop the original;

  %if &jj>0 | &kk>0 %then %do;
    data &dsn (drop=%do ii=1 %to &jj; &&&&long&&j&ii %end;);
      length %do ii=1 %to &jj; &&long7&ii..A  $&&&&mmA&&j&ii
                               &&long7&ii..B  $&&&&mmB&&j&ii %end;
             %do ii=1 %to &kk; &&&&long&&k&ii $&&&&mm&&k&ii  %end; ;
      set &dsn  end=last;
      format %do ii=1 %to &kk; &&&&long&&k&ii $&&&&mm&&k&ii.... %end;
             %do ii=1 %to &jj; &&long7&ii..A  $&&&&mmA&&j&ii....
                               &&long7&ii..B  $&&&&mmB&&j&ii.... %end;;
      %do ii=1 %to &jj;    * Processing the "long" vars, every record in file;
        mnX = min(150,length(&&&&long&&j&ii));
        &&long7&ii..A=substr(&&&&long&&j&ii,1,mnX);
        if mnX>150 then
        &&long7&ii..B=substr(&&&&long&&j&ii,151);
       %end;
    %end;
  %end;
%mend Shorten;


%macro prnt (nam,subt=%str( ),dt=,by=id,id=,dso=,var=,npat=,ps=126,ls=160,
 obs=,tds=,opt=);                             * dso is data set option;
*  setting ps=92 frequently causes pages that are split into two parts to
   run over 2 lines onto a second page;

%if &subt^= %then %let ss=0;      %* but if more titles than 2 are used then;
            %else %let ss=0;      %* ss must = Number of title lines - 2;
                                * Space has already been reserved for 2 titles;
options ls=&ls ps=&ps;          * So it was not necessary to compute &ss;

filename xxx "F:\trials\&db.&db2\&nam..dbf";
proc dbf db4=xxx  out=&nam;

data _null_; if 0 then set  &nam nobs=nnn;           * Count the records;
 call symput ('nnn',trim(left(put(nnn,4.))));  stop;  run;
 %if &nnn=0 %then %let obs=obs=0;

 %if &nnn>0 %then %do;
  proc contents data=&nam  noprint  out=dates (where=(format='DATE'));

  data _null_; if 0 then set dates nobs=dd;
    call symput ('dd',trim(left(put(dd,2.))));
  if dd>0 then do;   i=0;
      do until (last);
        set dates  end=last;  i+1;
        call symput ('dat'||trim(left(put(i,2.))),name); * get the date vars;
        end;
      end;
    stop;  run;
  proc sort data=&nam  out=&nam (&dso);  by &by;
  %if &dd>0 %then %do;
    format %do ii=1 %to &dd; &&dat&ii %end; date7.;
    %end;

  %if &npat^=0 %then %do;         * If patient not on the data set, set npat=0;
    data patients; set &nam; by &by;
      if first.&by;                 * Make sure &by just distinguishes patients;
    data _null_; if 0 then set patients nobs=npat;        * Count the patients;
      call symput ('npat',trim(left(put(npat,4.))));  stop; run;
    %end;

  %Shorten (&nam);
  %if &obs=zz %then %do;          %* Have OBS computed here (if so requested);
    proc contents data=&nam noprint out=cntnts;
    proc sort data=cntnts; by varnum;     * to count number of lines per obs;
    data _null_; set cntnts  end=last;

      if format='DATE' then col_xh = 1 + max(7,length(name)); *hor var names;
                           else col_xh = 1 + max(formatL,length(name));
      if format='DATE' then col_xv = 1 + 7;          * for vertical var names;
                           else col_xv = 1 + formatL;
      obslen_h + col_xh;
      obslen_v + col_xv;

      if _n_=1 then do; nlpobsH=1;                  * Number of lines per obs;
                        nlpobsV=1;  end;
      if col_sumH + col_xH>&ls. then do; col_sumH=4+col_xH; nlpobsH + 1; end;
                                else     col_sumH + col_xH;
      if col_sumV + col_xV>&ls. then do; col_sumV=4+col_xV; nlpobsV + 1; end;
                                else     col_sumV + col_xV;

      if last then do;
        if      obslen_H<=&ls. then obsppagH=&ps - 5 -&ss; * # of obs per page;
        else if obslen_V<=&ls. then obsppagV=&ps - 12-&ss;
        else do;                    obsppagH=int((&ps.-2-&ss)/nlpobsH) - 3;
                                    obsppagV=int((&ps.-2-&ss)/nlpobsV) - 10;
                                    end;
        numobs=&npag*max(obsppagH,obsppagV);
        if &nnn<=numobs + 2*max(obsppagH,obsppagV) then numobs=&nnn;
           * ie print all if they would fit on 2 more pages;
        call symput ('obs','obs=' || trim(left(put(numobs,4.))));
        end;
     run;
    %end;

*options nosymbolgen nomprint nomlogic;
  %end;
 %if &obs=zz %then %do;
 title "&db %upcase(&nam) ((&npag pages &obs out of) &nnn records &npat patients)";
   %end;
 %else %do; title "&db %upcase(&nam) (&nnn records &npat patients)";   %end;

 title2 "&subt";
%if &cont=  %then %do;
  proc contents data=&nam;   %end;

proc print data=&nam (&obs) &opt;
 %if &id^= %then %do; by &id; id &id; %end; &var;

run;
%mend prnt;
  /* &csi.!p&csi.1;90r&csi.19m&csi1m                   &db
  %upcase(&nam) &csi.15m(&nnn records &npat patients) &csi.4w&csi.22m;
     (used for the VAX) */
      * Define db & db2 here instead of in each macro call;
%let DB=Study2;   %let DB2=\SAS-dat;

 * if obs is left null, PRNT will process all of the observations and assign
   the actual number to OBS.  If you assign obs a number, PRNT will process
   that many observations. If you write obs=zz then you need to supply
   npag= a number and it will compute how many observations are needed to fill
   that many pages and assign that number to obs for you;

%prnt(medhist,obs=zz,npag=2,cnts=no);
%prnt(medhist,obs=obs=200,dso=where=(Tdate>'1jan97'd),
 var=var=patient tdate Lung1-Lung3 aero1 aero2,opt=noobs uniform);
```