

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 these data if listings are made so that the data can be scanned quickly.

This paper describes a macro, named PRNT, which can automatically generate informative and compact listings by supplying, in the simpler cases, little more than the name of the data set.

Listings are done by PROC PRINT and are made compact by splitting up long character strings into smaller strings so that they do not cause page-wrap. One method is to use a multi-page format splitting up long character variables over two or more pages each so that the first one fills the page to the right margin and the second begins at the left margin of the next page. Another way is to use column-wrap to break up the long variables so that they wrap around not to the left margin of the page, but to the left boundary of the same column, albeit creating additional lines within the file. Also using small print and printing only as many variables as can be displayed on a page without creating a *split* page helps eliminate wasted blank space.

Documentation such as the number of records, the number of values of the major sort key, and the date and time the file was last modified can be included in the title lines. Larger print is useful for the titles.

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

Introduction

Why Create Tailored Listings?

When beginning work with a new database it is necessary to both get the 'feel' of the data, (i.e. its range, number of digits and how much is missing), and have a source for looking up data values, e.g., for checking on outliers or missing data. Creating listings helps in getting familiar with the data and provides a valuable reference for future use. The listing is also an excellent place to make notes about data peculiarities as they are discovered, and keeping all of the notes in one document may help to identify trends. Thus, a carefully designed listing should be even more useful.

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 the ability to find a listing quickly.

Why Write A Macro to Make Listings?

Because making listings is somewhat of a repetitive process, a macro, once written, saves duplication of code and repeated modifications. Also when a new feature is designed, changing the macro to incorporate it is easier than tracing back through many separate codes to make the change.

The Listing

Prevent Wrap-around Data Listings (page-wrap).

The main concern in making the printout easy to read is preventing the procedure from wrapping long records around in order to get the whole record on the page. Usually this is accomplished by using a multi-page format created by two or more PROC PRINT calls giving each one only as many variables as can fit on a page without wrapping. This approach still may be inefficient. Consider the case of several short variables which may partially fill the page followed by a long character string which won't fit in the remaining free columns. With a multi-page format the long variable would have to be moved to the next page, potentially leaving much 'dead' space on the first page. Additionally, a character variable may be too long even to fit on a page. You may use as many as 20 characters to the inch when printing listings without losing legibility, and with paper 8.5 inches wide there are about 160 columns to use; occasionally this is inadequate. For these situations PRNT provides variable splitting and column-wrap.

The long variable could either be split so as to fill out the first page with the first segment, putting the second part in the left side of the second page by a macro, called SPLIT, or column-wrap may be employed to shorten the space required (although introducing additional lines beneath it) by another macro, called COL_WRAP.

How to Split a Long Variable with SPLIT

The macro SPLIT automatically partitions a long variable. This macro reads the output SAS data set of PROC CONTENTS and fetches 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, but for our listing we only care about the maximum length attained. The macro 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, 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 a variable, e.g., named *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 length(reason)>150 then
  reasonB = substr(reason,151);
```

This variable, *reason* will be dropped from the listing and replaced by *reasonA* and *reasonB*.

Prevent long variables from occupying many columns across the page by with COL WRAP.

By a call to a macro called COL_WRAP, PRNT can limit the length of all of the variables to, say, 30. Whenever COL_WRAP encounters an observation with any variables with values of more than 30 characters, it will chop off the tail-end of the variable at the end of the last word on or before the 30th character. (If the last word is more than 15 characters long then it will chop after the 30th character regardless.) The tail-end(s) are kept for the next line; all other variables are blanked out. If the tail-end is still too long the process is reiterated. The continued (or wrapped) variable is indented by one character for appearance's sake.

If there are relatively few long character strings in the data, this method will conserve a large amount of 'dead' space in the listing. On the other hand, if a variable consistently has long character string values throughout the data set, the conservation of blank space may be better served by using SPLIT to create new variables with the tail-ends rather than additional lines.

Comparisons

In either case page wrap will still occur if PROC PRINT cannot fit a whole observation (even after being modified by COL_WRAP) on one line. But the advantages of using one of these two methods over PROC PRINT alone are control over how the long character strings are broken up and considerable reduction white space. For one-page snapshots of the data sets you may have to permit page wrap, as well as using COL_WRAP and/or SPLIT. However, for long listings a multi-page format employing SPLIT and/or COL_WRAP will require fewer pages.

Use the VAR statement to split up a data set's variables over two or more PRNT calls.

Alternatively, you may wish to manually execute two or more calls to macro PRNT to split the listing into a multi-page format. To do this use the VAR statement which specifies which variables will be included in the printout. With no VAR statement the data set would be printed wrapping around if necessary. With PRNT, 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 will show how many can be printed on one page, although a good guess can be made by making a 'first look' run by PRNT. To do this use:

```
%prnt (labs,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. Writing a one page call to PRNT for each data set enables you to quickly see what kind of data are in the database.

Title Lines

The Title Line Tells about the SAS Data Set.

The most pertinent information about the file itself can be collected on the title line - a project identifier, file name, number of records, number of patients, number of records actually listed if it is not a complete listing, 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 a printer to read as may occur 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* data. The date at which 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. If it is suppressed then another date, e.g., the date last modified, could be included in a title line.

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.

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=
        obs=
        npag=0
        xx=30);
```

The first variable 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 mainly to eliminate variables or records that are not wanted.

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

If *cnts* is left blank, 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 a single data set.

Opt supplies whatever options you desire that PROC PRINT use. *Ps* specifies page size and appears on an options statement.

Npat would normally be left blank and, thus, causes 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 absent.

Obs is the number of observations to be printed. If no number is assigned to it, all will be printed.

Npag is the number of pages to be printed. If *npag*=0 then it is not used, but if greater than 0 then it overrides the obs request. For example, *Npag*=2 will cause two pages to be printed.

Acknowledgments

SAS is a registered trademark of SAS Institute, Inc., in the USA and other countries.

Hugh Geary
Neoprobe Corporation
425 Metro Place North
Dublin, Ohio 43017

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

APPENDIX

Complete Macro Listing

```

*LISTER.sas  SUGI Produces a Proc Contents & Print output 3July97;
*****
*\ Program  f:\users\mis\hugh\LISTER.sas
*\ Purpose  To print a proc contents output followed by a complete and
*\          tailored listing of selected data sets;
*\ Study    NEOPROBE ISS
*\ progrmr  Hugh Geary    May, Jan 97, May, March 96
*]*****
options nfmterr ls=132 ls=160 ps=90 ps=126  date number missing=' ';
%let      csi=; *b; * Use Courier New 6 pt - ls=160 ps=126;
%global nn2;
data _null_ put ' #1 #1 #1';
data _null_ put ' #2 #2 #2';

*****
%macro Mac_Vars;

*libname aa 'F:\trials\&db.&db2\&nam';
*data &nam; *set aa.&nam;

filename xxx 'F:\trials\&db.&db2\&nam..dbf';
proc dbf db5=xxx out=&nam;
*data &nam;* set &nam;* delete;
*proc contents data=&nam;* title 'Testing empty dataset';

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

%if &obs= %then %let obs=obs=&nnn;
*else %if &nnn<%scan(&obs,2,'=') %then %let obs=obs=&nnn;
%put '&obs=&obs';

%if &nnn=0 %then %do;
%let npat=0; %let npag=0; %let Lines=; %goto Endit;
%end;

proc contents data=&nam %if &var> %then (keep=&id &var); noprint out=cnts;
proc sort data=cnts; by varnum;

data _null_ set cnts end=last; * Set Up the AUX. macro variables;
vnum=left(put(varnum,3));
call symput('vv'||vnum,name); * put var names in VVii;
if format='DATE' then call symput('tt'||vnum,'D');
else call symput('tt'||vnum,put(type,1.));
* Create an id&i to identify the Proc Print ID vars;
do jj=1 to 99 until (scan("&id",jj)=' ' | name=scan(upcase("&id"),jj));
end; * upcase is necessary;
if name=scan(upcase("&id"),jj) then call symput('an_id'||vnum,name);
else call symput('an_id'||vnum,' ');
if last then call symput('nvv',vnum);
run;

```

```

%let chrvars=;                                /* Make a list of the character vars;
%do ii=1 %to &nvv;
  %if &&tt&ii=2 %then; %let chrvars=&chrvars &&vv&ii; %end;
  %put &chrvars;

proc contents data=&nam noprint out=dates
  (where=(format='DATE'));
data _null_; if 0 then set dates nobs=dd;          * get the date vars;
call symput ('dd',trim(left(put(dd,2.))));        * so that they can;
if dd>0 then do; i=0;                             * be assigned
date7.;
  do until (last);
    set dates end=last; i+1;
    call symput ('dat'||trim(left(put(i,2.))),name); * dat1,dat2,...;
  end;
end;
stop; run;
proc sort data=&nam; by &by;                        * SORT the data set;
                                                    * COUNT THE PATIENTS;
%if &npat^=0 %then %do;                            * If patient not on the data set, set npat=0;
data patients; set &nam; by &by;                  * such as for a Lab Range data set;
if first.&by;                                       * Make sure &by just distinguishes patients;
data _null_; if 0 then set patients nobs=npat;
call symput ('npat',trim(left(put(npat,4.)))); stop; run;
%end;

data Tempx; set &nam (&dso &obs);                  * Apply the DSO & OBS ;
format &chrvars;
%if &dd>0 %then %do;
  format %do ii=1 %to &dd; &&dat&ii %end; date7.; * REassign to date7.;
%end;
%Endit;
%mend Mac_Vars;

*****;
%macro Col_Wrap;
                                                    /* W R A P   T H E   C O L U M N S ;
  * Two Outputs: Tempx & the macro vars. WWii;
  * Only the WWii are needed from the first call;
%do ii=1 %to &nvv;
  %if &&uu&ii>&xx %then %let zz&ii=zyxwv&ii;
  %else %let zz&ii= ; ;                          * temporary holding vars;
  %let ww&ii=ww&ii;                               * to count the # wrapped lines for each var;
%end;

data Tempx (keep=first L_cnt %do ii=1 %to &nvv; &&vv&ii %end;)
  ww (keep=vnum www rename=(www=ww));
set Tempx end=last;
array ww %do ii=1 %to &nvv; ww&ii %end; (%do ii=1 %to &nvv; 0 %end;);

%do ii=1 %to &nvv;
  %if &&uu&ii>&xx %then %do; &&zz&ii=&&vv&ii; %end;
  %end;

first=1; flag=1;
do while (flag=1);
  flag=0;
  %do ii=1 %to &nvv;
    %if &&uu&ii>&xx %then %do;
      if length(&&zz&ii)>&xx then do;
        do jj=&xx to %eval(&xx/2) by -1 until(substr(&&zz&ii,jj+1,1)=' ');
        end;
        if jj<%eval(&xx/2) then jj=&xx;
        &&vv&ii=substr(&&zz&ii,1,jj);
      end;
    end;
  end;
  %do ii=1 %to &nvv;
    %if &&uu&ii>&xx %then %do;
      if length(&&zz&ii)>&xx then do;
        do jj=&xx to %eval(&xx/2) by -1 until(substr(&&zz&ii,jj+1,1)=' ');
        end;
        if jj<%eval(&xx/2) then jj=&xx;
        &&vv&ii=substr(&&zz&ii,1,jj);
      end;
    end;
  end;
  flag=1;
end;

```

```

        &&zz&i=substr(&&zz&i,1+jj);
        _i_ = &i;
        ww + 1;
        flag=1;
        end;
    else do; &&vv&i=&&zz&i; &&zz&i=' '; end;
    %end;
%else %do;
    if ^first then do;
        if "&&an_id&i"=' ' then do;
            %if &&tt&i=1 %then &&vv&i=.;
            %else %str(&&vv&i=' ');; end;
        * Assuming a 'Long' would never be an ID var;
    * Without the %if then else in the above line it would have created both of ;
    * the statements var=. & var=' ' one of which would always be unexecuted but ;
    * would still cause the note numeric. converted to char. or vv;
    end;
    %end;
    %*put &&uu&i &&ww&i '&&zz&i=' &&zz&i '&&vv&i=' &&vv&i '&id=' &id;
    %end;
    L_cnt + 1;
    output Tempx;
    first=0;
    end;
* if the "by - id" construction only involved one variable then the following
    %*if %quote(&&vv&i) ^= %quote(&id);
* could be used but the quote function was necessary to prevent the note:
    char converted to numeric;

    if last then do;
        do over ww;
            %* Put wrapping counts in a data set to sort them;
            www=ww; vnum= _i_ ; sww + ww; nww+(ww>0); output ww; end;
            call symput ('nww',left(put(nww,best.)));
            call symput ('sww',left(put(sww,best.)));
            * # of VARs wrapped;
            * Total count of wrappings;
        end;
    run;

proc sort data=ww; by descending ww;
proc print; title 'Number of Wrapped Lines by Variable - Sorted by freq';

data ww; set ww;
    k+1;
    call symput('QQ' || left(put(k,2.)),left(put(vnum,2.)));
    * Create a mapping back to ;
    * the original ordering;
run;
%do kk=1 %to &nww; %put &kk &QQ&kk; * &&&ww&QQ&kk; %end;

data _null_; if 0 then set tempx nobs=nn2;
    call symput ('nn2',trim(left(put(nn2,4.)))); stop; run;
    * Before Col-Wrap nnn records;
    * After Col-Wrap nn2 records;
%put '&nww= &nww ; run; %put '&nn2=&nn2;
%mend Col_wrap;

*****;
%macro Max_Len;
    data var_len (keep=varnum UU);
        set Tempx (&LLL) end=last;
        retain uu1-uu&nv (&do ii=1 %to &nv; 0 %end;);
        %do ii=1 %to &nv;
            %if &&tt&i=2 %then len=length(&&vv&i);
            %else %if &&tt&i=D %then len=7;
            %else len=length(trim(left(put(&&vv&i,best.))));
            uu&i=max(uu&i,len);
        %end;
    * Compute the UU&i;
    * maximum lengths ;
    * actually attained;

```

```

if last then do;
  %do ii=1 %to &nv;
    call symput('UU' || left(put(&ii,2.)),left(put(uu&ii,3.)));
    varnum=&ii;    uu=uu&ii;    output;
  %end;
end;
%mend Max_Len;

*****
%macro Paging;
  * Next (if NPAG is given) compute the # of ;
  * lines that will fit on &NPAG pages;
  * first - count number of lines per record ;
proc print data=cnts; title 'Cnts merged with Var_len';
  var libname memname name type length varnum label format formatl formatd
  nobs modate sortedby;
proc print data=var_len; Title 'Var_len';
  * Having wrapped the columns there may be ;
  * more than one record from some
observations.;
  data _null_; merge cnts var_len end=last; by varnum;
  if _n_=1 then do; nlpobsH=1; * Number of lines per record;
    nlpobsV=1; end;
  uu=min(uu, &xx);
  col_xh =1+ 4*( _N_=1) + max(uu,length(name)); * for horizontal var names;
  col_xv =1+ 4*( _N_=1) + uu; * for vertical var names;
  * 4 columns for the obs #;
  obslen_h + col_xh;
  obslen_v + col_xv;

  if col_sumH + col_xH>&ls. then do;
    extraH+(&ls-col_sumH); col_sumH=4+col_xH; nlpobsH + 1; end;
    else col_sumH + col_xH;
  if col_sumV + col_xV>&ls. then do;
    extraV+(&ls-col_sumV); col_sumV=4+col_xV; nlpobsV + 1; end;
    else col_sumV + col_xV;
  put varnum name col_sumH= col_sumV= nlpobsH= nlpobsV=;

  if last then do;
    obsppagH=int((&ps.-2-&ss)/nlpobsH) - 3; * Maximize the Number;
    obsppagV=int((&ps.-2-&ss)/nlpobsV) - 10; * of records per page;
    * Horiz. vs Vert.labels;

    obsppag=max(obsppagH, obsppagV);
  put nlpobsH= nlpobsV= obsppag=;
  * next print ALL obs if they would fit on only 2 more pages;
  if &nn2>(&npag + 2)*obsppag then numlines = &npag*obsppag;
    else numlines = &nn2;
  put numlines= obsppag=;
  call symput ('Lines','obs=' || trim(left(put(numlines,4.))));
end;
run;
* Calculate how many columns are not being used (add them to the long vars);

  * Now if the Lines # would break it off a multi-record
  observation before the end, then recalculate the Lines #;
data _null_; if %scan(&Lines,2,'=')<&nn2 then do;
  obsn1=%scan(&lines,2,'=') + 1;
  do obsn=obsn1 to 1 by -1 until(first=1); put obsn=;
    set temp point=obsn; * Keep only complete multi-;
  end; * record obs. for printing ;
  obsn=obsn-1;
  call symput('Lines','obs=' || trim(left(put(obsn,4.))));

```

```

put obsn= obsn= obsn= first= id= ;
end; stop;
run;

%put '&ps'=&ps '&npag'=&npag '&nnn'=&nnn '&nn2'=&nn2 '&obs'=&obs
'&lines'=&lines;
%mend Paging;

*****;

%macro prnt (nam,subt=%str( ),dt=,by=id,id=,dso=,var=,npat=,ps=126,ls=160,
npag=0,obs=,tds=,opt=,xx=30);
* 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;

%Mac_Vars;
%if &nnn>0 %then %do;
  %let LLL=;
  %Max_Len;
  data _null_ ; put ' #3 #3 #3';
  %Col_Wrap;
  %if &npag>0 %then %let LLL=obs=%eval(&npag*&ps);
      %else %let LLL=;;      * ie for when NPAG is not used;

  %Max_Len;
  %if &npag>0 %then %Paging;
      %else %let Lines=&obs;

  data temp2 ; set tempX (&lines);      * After Col-Wrap there are;
  if first=1;      * fewer obs left per page.;
  data _null_ ; if 0 then set temp2 nobs=fff ;      * REcount the observations;
  call symput ('numobs',trim(left(put(fff,4.)))); * that will be printed ;
  if fff=&nnn then call symput('numobs','ALL'); stop; run;
%end;

*options symbolgen mprint mlogic;

%if &npag>0 %then %do;
  title "&db %upcase(&nam) (&npag pages &numobs of the &nnn Obs & &npat
Patients)";
  %end;
  %else %do; title "&db %upcase(&nam) (&nnn Observations & &npat Patients)";
%end;

title2 "&subt";
%if &cont= %then proc contents data=&nam position;; * position;

%if &npag>0 %then %do;
  proc print data=TempX (&Lines drop=first) &opt;
    %if &id^= %then %do; by &id; id &id; %end; &var;
  %end;

run;
%mend prnt;
*****;

%prnt(adverse, npag=2,id=id,subt=);

```