

PROC DATACHK Revisited: The DATACHK Macro

Jeffrey M. Abolafia and Carry W. Croghan
 Department of Biostatistics
 University of North Carolina

Introduction

PROC DATACHK, a SUGI Supplemental Library procedure in Version 5 of the SAS® System, provided a useful method for quickly checking numeric data in a selected SAS data set. For each numeric variable, PROC DATACHK listed the number of nonmissing observations, the number of missing observations, the five lowest values, the five highest values, and the number of observations having six prespecified missing values. While this information is available in PROC UNIVARIATE, a major advantage of PROC DATACHK was that output was presented concisely: one line per variable. With the release of Version 6 of the SAS System, the SUGI Supplemental Library procedures were discontinued. Since that time, a new procedure to replace PROC DATACHK has not been developed.

The DATACHK macro attempts to simulate PROC DATACHK. Like PROC DATACHK, the DATACHK macro was developed to quickly check numeric variables in a selected SAS data set. For each numeric variable, DATACHK concisely lists the number of nonmissing observations, the number of missing observations, the five lowest values, and the five highest values (the DATACHK macro does not list the six prespecified missing values).

Usage

The macro DATACHK contains four keyword parameters:

DATA	The name of an existing SAS data set. This can be a permanent or work data set. You can specify a one or two level name. This parameter is required.
VAR	A list of numeric variables separated by spaces. If the VAR parameters not specified then all numeric variables are processed. This parameter is optional.
BYS	A list of BY variables separated by spaces used to obtain separate analysis for groups defined by the BY variables.
ORDER	By default variables are printed in the order of their position in the data set. Specify ORDER=A to list variables alphabetically.

Below are several examples of using the macro DATACHK. Example 1 uses the macro without the optional VAR parameter. Example two invokes the macro using the VAR parameter. Example three demonstrates using the macro with by group processing.

```
%DATACHK (DATA=IN.FINAL)
```

```
%DATACHK (DATA=IN.FINAL, VAR=LDL HDL AGE)
```

```
%DATACHK (DATA=IN.FINAL, VAR=LDL HDL AGE, BYS=TRT)
```

Printed Output

The DATACHK macro prints the following information for each numeric variable in a selected SAS data set:

1. the number of nonmissing observations
2. the number of missing observations
3. the five largest values
4. the five smallest values
5. the variable label (if one exists).

Below is sample output from the DATACHK macro.

VARIABLE	N	MISSING	5 Smallest Values					5 Largest Values				
PACNO PACKET NUMBER	698	0	2010	2014	3015	3041	3049	95019	95022	95023	95039	96012
EVDAT	72	626	01JUL76	26JUL76	31JUL76	01AUG76	02AUG76	11JUN77	15JUN77	17JUN77	18JUN77	23JUN77
DRINKER	694	4	0	1							0	1
_CWL	698	0	438.4149	565.9099	650.3849	802.2349	867.2399	4125.64	4141.15	4176.87	4281.039	4916.199
QDL RANK FOR VARIABLE LDL	698	0	0	1	2	3		0	1	2	3	

Additional Features

1. DATACHK performs some error checking. If the input data set does not exist or if variables specified on the VAR parameter do not exist, DATACHK prints an appropriate message and will stop executing.
2. If a variable has a date format of DATEw., DDMYYw., MMDDYYw., or YYMMDDw., then the five largest and smallest values are printed with the DATE9. format.
3. If a variable has a time format of TIMEw. then the five largest and smallest values are printed with the TIME8. format.
4. Variables that do not have a date or time format are displayed with the BEST10. format.
5. Variable labels are printed if they exist.
6. By group processing is available. To use this feature, the input data set must be sorted by the variables specified in the BYS parameter.
7. By default variable are listed in positional order. By using the ORDER parameter, variables can be listed in alphabetical order.

Utilization of New SAS features to Improve Efficiency

Several new features of SAS 6.12 have been incorporated to improve the efficiency of the DATACHK macro. The macro function &SYSFUNC in conjunction with the new functions EXIST, OPEN, CLOSE, and VARNUM are utilized. The EXIST function is an efficient method of verifying the existence of a dataset. The OPEN, CLOSE, and VARNUM functions are used in the error checking algorithm to verify the existence of variables

specified in the VAR and BYS parameters. Additional SAS features used are temporary arrays, dictionary tables, and PROC SQL. Temporary arrays are used extensively throughout the macro. Two useful features of temporary arrays are: 1) variables created by temporary arrays are not contained in the output data set; and 2) elements of temporary arrays are automatically retained. The dictionary table SASHELP.VCOLUMN in combination with PROC SQL is used to obtain labels and formats. PROC SQL is also used to efficiently merge and order data.

REFERENCES

SAS Institute Inc. (1983), SUGI Supplemental Library User's Guide 1983 Edition. Cary, NC: SAS Institute Inc.

Acknowledgments

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. Other brand and product names are registered trademarks of their respective companies.

Contact Information

Jeffrey Abolafia
University of North Carolina
Department of Biostatistics
CB #8030, 203 Nations Bank Plaza
Chapel Hill, NC 27514
uccjma.csc@mhs.unc.edu

Carry W. Croghan
Clin Trials Research Inc.
P.O. Box 13991
Research Triangle Park, NC 27709
ccroghan@clintrialsresearch.com

APPENDIX

```

%MACRO BDATACHK(DATA=,VAR=,BYS=,ORDER=P);
%*****
%*** NAME: BDATACHK
%***
%*** VERSION: 1
%***
%*** FUNCTION:
%***
%*** DATACHK can be used to quickly check numeric variables
%*** in a selected SAS data set. It is similar to PROC DATACHK
%*** in SAS Version 5.18 . For each numeric variable, DATACHK
%*** lists the number of nonmissing observations, the number
%*** of missing observations, the five lowest values, and the
%*** five highest values. Variable labels are also included.
%*** DATACHK runs under SAS Version 6.11 or later. A copy of
%*** DATACHK that runs under earlier version of SAS can be
%*** obtained from the author.
%***
%*** PARAMETERS:
%***
%*** DATA The name of an existing SAS data set.
%*** This can be a permanent or work data
%*** set. You can specify a one or two level
%*** name. This parameter is required.
%***
%*** VAR a list of numeric variables, separated
%*** by spaces, to be checked. If VAR is not
%*** specified then all numeric variables are
%*** processed.
%***
%*** BYS a list of BY variables separated by spaces
%*** used to obtain separate analysis for groups
%*** defined by the BY variables
%***
%*** ORDER by default variables are printed in the
%*** order of their position in the data set.
%*** Specify ORDER=A to list variables
%*** alphabetically
%***
%*** EXAMPLES:
%***
%*** %DATACHK2(DATA=IN.FINAL)
%*** %DATACHK2(DATA=IN.FINAL,VAR=LDL HDL AGE)
%*** %DATACHK2(DATA=IN.FINAL,VAR=LDL,BYS=TRT)
%***
%*** AUTHOR: Jeffrey Abolafia / Carry Croghan
%***
%*** NOTE: Uses macros VARLIST and COUNT
%***
%*****
options nosource nonotes ;

%put ** MACRO DATACHK VERSION 1 IS EXECUTING ** ;

%let var=toupper(%var) ;
%let bys=toupper(%bys) ;
%let order=toupper(%order) ;

/* check that DATA exists */
%if %data= %then %let ex=NO ;
%else %do ;
%if %sysfunc(exist(%data)) %then %let ex=YES;
%else %let ex=NO ;
%end ;
%if %ex=NO %then %do ;
%put THE INPUT DATA SET DOES NOT EXIST DATACHK WILL STOP EXECUTING;
%goto nodata ; ** this will end execution ** ;
%end;

/* store all numeric variables in a global macro variable */
%varlist(data=%data,vars=numvars,type=n)

/* if no numeric variables are selected then end execution */
%if %numvars= %then %do ;
%put THERE ARE NO NUMERIC VARIABLES IN %DATA DATACHK WILL STOP
EXECUTION;
%goto nodata ; ** end execution ** ;
%end ;

/* count number of numeric variables and store in macro variable */
%count(list=%numvars,NUM=nvars)

/* count number of by variables and store in macro variable */
%if %bys ne %then %do;
%count(list=%bys,NUM=nbys)

/* store last by variable in a macro variable */
%let lastby=%scan(%bys,%nbys);
%end;

/* If VAR= parameter is specified check for the existence
of variables specified
*/

%if %var ne %then %do;
%let c=1;
%let prob=0 ;
%do %until(%scan(%var,%c)= ) ;
%let v = %scan(%var,%c) ;
%if %index(%numvars,%v)=0 %then %do;
%let prob=1 ;
%put THE VARIABLE %V NOT FOUND IN %DATA DATACHK WILL STOP
EXECUTION;
%end;
%let c=%eval(%c + 1) ;
%end ;
%if %prob=1 %then %goto nodata ; ** end execution ** ;
%end ;

/* If BYS= parameter is specified check for the existence
of by variables specified
*/

%if %bys ne %then %do;
%let prob=0 ;
%let op = %sysfunc(open(%data));
%do i=1 %to %nbys ;
%let byi = %scan(%bys,%i) ;
%let c = %sysfunc(varnum(%op,%byi));
%if %c=0 %then %do;
%let prob=1 ;
%put THE BY VARIABLE %BYI NOT FOUND IN %DATA DATACHK
WILL STOP EXECUTION;
%end;
%end;
%let c1 = %sysfunc(close(%op));
%if %prob=1 %then %goto nodata ; ** end execution ** ;
%end ;

/* end of variable check algorithm */

/* calculate N, # missing, 5 lowest and highest for each
numeric variable
*/

data outx(keep=name n_m hl-h5 ll-l5 %bys);
set %data(keep= %numvars %bys) end=eof ;
%if %bys ne %then %str(%bys);

/* store Ns, # missing, 5 highest and lowest in temporary arrays */
/* Note: elements in temporary arrays are automatically retained */

array vars(%nvars) %numvars ;
array ns(%nvars) TEMPORARY ;
array m(%nvars) TEMPORARY ;
array la(%nvars) TEMPORARY ;
array lb(%nvars) TEMPORARY ;
array lc(%nvars) TEMPORARY ;
array ld(%nvars) TEMPORARY ;
array le(%nvars) TEMPORARY ;
array ha(%nvars) TEMPORARY ;
array hb(%nvars) TEMPORARY ;
array hc(%nvars) TEMPORARY ;
array hd(%nvars) TEMPORARY ;
array he(%nvars) TEMPORARY ;

%if %bys ne %then %do;
%if first.%lastby %then %do %j=1 to %nvars;
ns[%j]= 0 ;
m[%j]= 0 ;
la[%j]= . ;
lb[%j]= . ;
lc[%j]= . ;
ld[%j]= . ;
le[%j]= . ;
ha[%j]= . ;
hb[%j]= . ;
hc[%j]= . ;
hd[%j]= . ;
he[%j]= . ;
%end;
%end;

do i = 1 to %nvars ; /* begin loop for each variable */

ns[i] + (vars[i] > .2); /* no. non-missing */
m[i] + (vars[i] <=.2); /* no. missing */

%if vars[i] > .2 %then %do; /* find 5 lowest/highest if not missing */

%if vars[i] > he[i] %then %do;
ha[i] = hb[i] ;
hb[i] = hc[i] ;
hc[i] = hd[i] ;
hd[i] = he[i] ;
he[i] = vars[i] ;
%end ;
%else %if hd[i] < vars[i] < he[i] %then %do;
ha[i] = hb[i] ;
hb[i] = hc[i] ;
hc[i] = hd[i] ;
hd[i] = vars[i] ;
%end ;
%else %if hc[i] < vars[i] < hd[i] %then %do;
ha[i] = hb[i] ;
hb[i] = hc[i] ;
hc[i] = vars[i] ;
%end ;
%else %if hb[i] < vars[i] < hc[i] %then %do;
ha[i] = hb[i] ;
hb[i] = vars[i] ;
%end ;

%end ;

```

```

else if ha[i] < vars[i] < hb[i] then do;
  ha[i] = vars[i];
end;

if .Z < vars[i] < la[i] then do;
  la[i] = ld[i];
  ld[i] = lc[i];
  lc[i] = lb[i];
  lb[i] = la[i];
  la[i] = vars[i];
end;
else if (vars[i] > .z) & (la[i] <= .z) then la[i] = vars[i];
else if (la[i] < vars[i] < lb[i]) & (lb[i] > .z) then do;
  la[i] = ld[i];
  ld[i] = lc[i];
  lc[i] = lb[i];
  lb[i] = la[i];
end;
else if (vars[i] > la[i]) & (la[i] > .z) & (lb[i] <= .z) then
  lb[i] = vars[i];
else if (lb[i] < vars[i] < lc[i]) & (lc[i] > .z) then do;
  la[i] = ld[i];
  ld[i] = lc[i];
  lc[i] = vars[i];
end;
else if (vars[i] > lb[i]) & (lb[i] > .z) & (lc[i] <= .z) then
  lc[i] = vars[i];
else if (lc[i] < vars[i] < ld[i]) & (ld[i] > .z) then do;
  la[i] = ld[i];
  ld[i] = vars[i];
end;
else if (vars[i] > lc[i]) & (lc[i] > .z) & (ld[i] <= .z) then
  ld[i] = vars[i];
else if (ld[i] < vars[i] < le[i]) & (le[i] > .z) then
  le[i] = vars[i];
else if (vars[i] > ld[i]) & (ld[i] > .z) & (le[i] <= .z) then
  le[i] = vars[i];
end;
end;

/* restructure data to 1 record per variable by group */
length name $8;
if &by= then istr(if eof=1 then do i=1 to &nvars;);
else if &by= ne then istr(if last.&lastby then do i=1 to
&nvars;);
call vname(vars[i],name);
n = ns[i];
m = m[i];
l1 = la[i];
l2 = lb[i];
l3 = lc[i];
l4 = ld[i];
l5 = le[i];
h1 = ha[i];
h2 = hb[i];
h3 = hc[i];
h4 = hd[i];
h5 = he[i];
output;
end;
run;

/* Merge in dictionary table containing variable labels and
formats(SASHELP.VCOLUMN). Then reformat variables for
display purposes.
*/
proc sql;
create table outz as
select a.*,label,format
from outx a,sashelp.vcolumn b
where a.name=b.name & libname="&libref" & memname="&dsn"
order by
  if &by= ne then &do;
    do i = 1 to &nvars;
      &&by=i;
    end;
  else if &order=P then &istr(varnum);
  else if &order=A then &istr(name);
quit;
run;

data _null_;
set outz end=eof;
if &by= ne then istr(by &by);

/* convert 5 highest and lowest to appropriate character format */
array ovar{10} l1-l5 h1-h5;
array nvar{10} $10 a1-a5 b1-b5;

do i= 1 to 10;
  if ovar{i}<=.z then nvar{i}=' ';

  /* if original variable had a date format of DATE7,
  DDMYYW, MMDDYY, or YYMMDDw. then format with the
  DATE7. format
  */
  else if (substr(format,1,4) = 'DATE' or
  (substr(format,1,6) in ('DDMMYY', 'MMDDYY', 'YYMMDD'))
  then nvar{i}=' ' || put(ovar{i},date9.);
end;

```

```

/* if original variable had a time format of TIMEw.
then format with the TIME8. format
*/
else if (substr(format,1,4) = 'TIME')
then nvar{i}=' ' || put(ovar{i},time8.);

/* otherwise format with the best10. format */
else nvar{i}=right(put(ovar{i},best10.));

end;

/* send output to standard print file */
file print notitles ls=173 ps=60 header=;
if &by= ne then &do;
  if (FIRST.&lastby) & (_N_>1) then put _page_;
  &end;
  put @1 name $8. @11 n 7. @22 m 7. @42 ' ' @44
  (a1-a5) ($10., +2) +1 ' ' +1 (b1-b5) ($10., +2) / @1 label
  @42 ' ' @103 ' ' / @42 ' ' @103 ' ' ;

return;
h;
put @58 "NUMERIC DATA CHECKS FOR DATA SET &uppercase(&DATA)" /
if &by= ne then &do;
  @59
  do i=1 to &nvars;
    &&by=i;
  end;
  &end;
  /*
  @1 'VARIABLE' @16 'N' @22 'MISSING'
  @66 '5 SMALLEST VALUES'
  @124 '5 LARGEST VALUES' / ;
return;
run;

** delete data set created during macro execution ** ;
proc printto log=work.temp.temp.log new;
run;
proc datasets library=work;
delete outx outz; ;
run;
quit;
proc printto ;
run;

&nodata;
options source notes;
&END BDATACHK; /* END OF MACRO DATACHK */

/* auxillary macros */
&macro varlist(data=&vars,&type=);

*****
***** VARLIST CREATE A GLOBAL MACRO VARIABLE THAT HOLDS A LIST *****
***** OF ALL VARIABLES IN A SELECTED DATA SET WHERE: *****
***** *****
***** DATA THE NAME OF AN EXISTING SAS DATA SET. *****
***** THIS CAN BE A PERMANENT OR WORK DATA *****
***** SET. YOU CAN SPECIFY A ONE OR TWO LEVEL *****
***** NAME *****
***** *****
***** VARS = THE NAME OF THE GLOBAL MACRO VARIABLE WHICH *****
***** WILL HOLD THE LIST OF VARIABLES IN THE *****
***** DATASET LIBREF.DSN *****
***** *****
***** TYPE = IF YOU WOULD LIKE ONLY A LIST OF NUMERIC *****
***** VARIABLES SET TYPE EQUAL TO N. IF YOU *****
***** WOULD LIKE A ONLY A LIST OF CHARACTER *****
***** VARIABLES SET TYPE EQUAL TO C. THE DEFAULT *****
***** IS A LIST OF ALL VARIABLES. *****
***** *****
***** EXAMPLE: *****
***** *****
***** LIBNAME SC V604 'C:\BIOS111\SASDATA'; *****
***** %VARLIST(DATA=SC.CLASS,VARS=N.VARS,TYPE=N) *****
***** *****
*****
***** %let data=&uppercase(&data); *****
*****
***** %global libref dsn; *****
***** if %index(&data,.)=0 then &do; /* one level */ *****
***** %let libref= WORK; *****
***** %let dsn = &data; *****
***** %end; *****
***** %else &do; *****
***** %let libref=%scan(&data,1,'. '); *****
***** %let dsn=%scan(&data,2,'. '); *****
***** %end; *****
*****
***** ** Create a global macro variable that will hold ** *****
***** ** the list of variables from the data set specified ** *****
***** ** by the libref and dsn parameters. The name of the ** *****
***** ** macro variable is value specified for the VARS ** *****
***** ** macro variable *****
*****
***** %global &vars; *****
***** proc sql noprint; *****
***** select name *****
***** into :&vars separated by ' ' *****
***** from sashelp.vcolumn *****
***** where libname="&libref" and memname="&dsn" *****

```

CODERS CORNER

```
      if tucase(stype)=N then & type='num' ;
      else if tucase(stype)=C then & type='char' ;
      if &var ne then & indexw("&var",name) > 0 ;
      if &bys ne then & indexw("&bys",name) = 0 ;
quit ;
}mend varlist ;

{MACRO COUNT(LIST=,NUM=);
{*****
{*** COUNTS NUMBER OF ELEMENTS IN A LIST AND CREATES A MACRO ***
{*** VARIABLE TO HOLD THE VALUE ***
{*** LIST = A MACRO VARIABLE CONTAINING A LIST ***
{*** OF ELEMENTS ***
{*** NUM = NEW GLOBAL MACRO VARIABLE WHICH ***
{*** HOLDS THE NUMBER OF ELEMENTS IN LIST ***
{*** EXAMPLE: {COUNT(LIST=&COVAR,NUM=NCOVAR) ***
{*****
{*** CREATE GLOBAL VARIABLE TO PASS BACK TO CALLING MACRO ***;
{GLOBAL &NUM;

{*** SET TO FIRST ELEMENT OF LIST AND ENTER LOOP IF NOT EMPTY ***;
{LET &NUM=0;
{LET WORD={SCAN(&LIST,&&NUM+1);

{DO {WHILE(&WORD NE);
    {LET &NUM={EVAL(&&NUM+1);
    {LET WORD={SCAN(&LIST,&&NUM+1);
    {END;
{MEND COUNT;

/* END OF AUXILIARY MACROS */
```