Identify and Remove Any Variables, Character or Numeric, That Have Only Missing Values
Wei Xu, Boston Scientific, Marlborough, Massachusetts

ABSTRACT
Have you ever identify and remove any variables, character or numeric that have only missing values? If there are some variables that have only missing value, you will need to delete the variables per your analysis requests.

This paper will describe a simple, straightforward way to use SAS® SQL dictionary tables to create a list of character variables and a list of numeric variables from the data set and store the result in macro variables CLIST (character) and NLIST (numeric). The count of each type is also captured using PROC SQL of SAS®, and in a DATA _NULL_ which creates an array for the character variables and an array for the numeric variables. The final step will create a macro that generates the list of variables to be dropped for the DROP statement.

THE SETUP
You will need a few things to get this set up.
(1) Create macro variables of the library and data set to be modified using PROC SQL.
(2) In a DATA _NULL_ which creates an array for the character variables and an array for the numeric variables.
(3) The macro generates the list of variables to be dropped for the DROP statement.

THE PROC SQL
Create macro variables of the library and data set to be modified.

/**************************
%let lib=work;
%let mem=testdata;
/**************************

Use SQL dictionary tables to create a list of character variables and a list of numeric variables from the data set and store the result in macro variables CLIST (character) and NLIST (numeric). The count of each type is also captured.

/******************************************************************************************************/
proc sql noprint;
  select name, put(count(name),5.-L) into :clist separated by ' ', :charct
  from dictionary.columns
  where libname=upcase("&lib") and memname=upcase("&mem") and type='char';

  select name, put(count(name),5.-L) into :nlist separated by ' ', :numct
  from dictionary.columns
  where libname=upcase("&lib") and memname=upcase("&mem") and type='num';
quit;
/******************************************************************************************************/

THE DATA _NULL_
In a DATA _NULL_ that creates an array for the character variables and an array for the numeric variables. Create two more arrays, one for character and one for numeric, where the variables will serve as flags. The values are initially set to ‘false’ to indicate that they have only missing values. Any time a non-missing value is found for a variable, the corresponding flag variable is set to ‘true’.
data _null_;  
    array char(*) &clist;  
    array num(*) &nlist;  
    array c_allmiss (&charct) $ (&charct*'true');  
    array n_allmiss (&numct) $ (&numct*'true');  
    set testdata end=done;  
    do i=1 to dim(c_allmiss);  
       if char(i) ne ' ' then c_allmiss(i)='false';  
    end;  
    do i=1 to dim(n_allmiss);  
       if num(i) ne . then n_allmiss(i)='false';  
    end;  
/* Once the entire data set has been processed, loop through the flag arrays and create a macro variable for any variable that still has a flag set to 'false'. Keep count of how many there are and put that number into a macro variable as well. */  
    if done then do;  
       cnt=0;  
       do i= 1 to dim(c_allmiss);  
        if c_allmiss(i) ='true' then do;  
            cnt+1;  
        call symput('var'||put(cnt,3.-l),vname(char(i)));  
        end;  
       end;  
       do i=1 to dim(n_allmiss);  
        if n_allmiss(i)='true' then do;  
            cnt+1;  
        call symput('var'||put(cnt,3.-l),vname(num(i)));  
        end;  
       end;  
       call symput('cnt',put(cnt,3.-l));  
    run;  
    /*****************************************************************************/  
/* THE MACRO */  
THE MACRO  
This macro generates the list of variables to be dropped for the DROP statement.  
/*****************************************************************************/  
%macro dropem;  
   %do i = 1 %to &cnt;  
        &&var&i.  
   %end;  
%mend;  
/*****************************************************************************/  
/* THE MACRO CALL */  
Finally, create a new data set similarly named to the original and issue the DROP statement.  
/*****************************************************************************/  
data &lib..&mem;
set &lib.&mem.;
drop %dropem;
run;

/*****************************/

THE RESULT
This program would result in removing any variables, character or numeric that have only missing values:

1. You need to identify the variables that have only missing values from the dataset.
2. Remove any variables, character or numeric that have only missing values from the dataset.

QUESTIONS
How many of you have been used PROC SQL to create SAS data sets, data views, and macro variables?

PROC SQL is a powerful tool in the SAS system. The SAS SQL procedure enables you to
• Retrieve and manipulate data that is stored in tables or views.
• Create tables, views, and indexes on columns in tables.
• Create SAS macro variables that contain values from rows in a query's result.
• Add or modify the data values in a table's columns or insert and delete rows. You can also modify the table itself by adding, modifying, or dropping columns.
• Send DBMS-specific SQL statements to a database management system (DBMS) and retrieve DBMS data.

CONCLUSION
This is a simple way to identify and remove any variables, character or numeric that have only missing values.

The advantages are:
(1) The new macro call can be created to remove any variables, character or numeric that have only missing values from the dataset.

/*****************************/

%check(lib=work, mem=testdata);

/*****************************/

(2) This macro %CHECK should be located in a central place such as project macro area and each project can call the macro from there.

ACKNOWLEDGEMENTS
SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

CONTACT INFORMATION
Wei Xu
Boston Scientific
100 Boston Scientific Way
Marlborough, MA 01752-1234
(508) 683-4264
wei.xu@bsci.com