Dropping variables from a large SAS® data set when all their values are missing
Gwen Babcock, New York State Department of Health, Albany, NY

ABSTRACT
Several methods have been proposed to drop variables whose values are all missing from a SAS data set. These methods typically use PROC SQL, macro variables, and/or arrays, all of which may be slow or fail when applied to large data sets. One method uses hash tables and CALL EXECUTE, but it works only on character variables. Therefore, I have developed an alternative method which can be applied to data sets of any size. The strategy is to use PROC MEANS and PROC FREQ to identify variables whose values are all missing. These variables are then dropped using DATA steps generated by CALL EXECUTE. This method, which uses intermediate level coding techniques, will complement previously proposed methods because it works well on large data sets, and on both character and numeric variables. An example is given using US Census American Community Survey data with SAS 9.3 under Windows XP.

INTRODUCTION
With relatively small data sets, you can often run PROC FREQ or some other procedure which provides descriptive statistics, visually inspect the results, and use the DROP data set option to remove variables whose values are all missing (Zdeb, 2011). However, for large data sets, this method is very tedious and time consuming. Therefore, two macros have been proposed to remove variables from a SAS data set when all its values are missing. The first, the %DROPMISS macro (Sridharma, 2006 and 2010) fails with large data sets. The second, the %drop_blank_vars macro proposed by Tanimoto (2010), works only on character variables. Both use advanced coding techniques: Sridharma uses macros, and Tanimoto uses macros and hash tables. The method proposed here uses intermediate coding techniques, is fast, and works on the largest data sets. In addition, with minor modifications, this method can be used to remove coded missing values, such as 99999999 or -1, etc.

PREPARATION
To prepare to drop all variables with missing values, I first specified the libraries and data sets to use. I wished to preserve the label and compression of the original data set, so I obtained that information using PROC CONTENTS and placed it in macro variables using CALL SYMPUTx in a DATA step. (For a discussion of alternative ways to access table metadata, see Carpenter (2013)). I also make a copy of the original data set to use for manipulation, so that the original is preserved in case of need.

/*specify any libraries used*/
libname sas 'E:\census\ACS\2007_2011\SAS\sasblock';
/*specify the input and output data sets*/
%let dsin=sas.acs2010_5yr_blkgrp;
%let dsout=sas.acs2010_5yr_blkgrp2;
/*specify an alternative missing value for numeric variables. this value should be a number*/
%let altnumericmissvalue=-1;
proc contents data=&dsin noprint out=mycontents (keep=memlabel compress);
run;
data _null_;set mycontents(obs=1);
call symputx("mylabel",memlabel);
call symputx("mycompress",compress);
run;
data &dsout (compress &mycompress.);
set &dsin;
run;

FINDING NUMERIC VARIABLES WITH ONLY MISSING VALUES
Once you have finished the preparation, You can find the number of non-missing values for all numeric variables by using the "N" statistic in PROC MEANS.
proc means data=&dsout noprint;
output out=checknumeric(drop=_:) n=/*get the number of NON-MISSING for all numeric variables*/
run; /*one observation, many variables*/

The data set output from PROC MEANS has only one observation. It has one variable for each numeric variable in the input data set. I transpose the data and select only those variables which have no values.

```
proc transpose data=checknumeric out=checknumeric2 (drop=_label_ rename=(col1=numhere _name_=vname));
run;
```

```
data dropnumeric;
set checknumeric2 (where=(numhere=0));
run;
```

PROC MEANS is very fast. However, for datasets containing hundreds of variables and millions of observations, finding variables with all missing values can be done more quickly by first testing a subset of the data. For example, the OBS=10000 data set option can be used to test the first 10,000 observations. Only the variables whose values are all missing in the subset would be tested in the entire dataset. Appendix A shows the code for this approach.

### FINDING NUMERIC VARIABLES WITH ONLY MISSING CODED VALUES

If the missing values are coded, a slight variation on the theme is needed. The assumption behind this code is that if the maximum value and minimum value are both equal to the coded missing value, then all of the values for the variable are equal to the coded missing value. First, I use PROC MEANS to get the minimum and maximum values of all the numeric variables, transpose them as before, and then merge them. Then I select those whose minimum and maximum values are both equal to the coded missing value.

```
proc means data=&dsout noprint;
output out=biggest(drop=_:) max=;
run;
proc means data=&dsout noprint;
output out=smallest(drop=_:) min=;
run;
```
FINDING CHARACTER VARIABLES WITH ONLY MISSING VALUES

Since PROC MEANS can only be used with numeric variables, I use PROC FREQ to find character variables whose values are all missing. The NLEVELS option produces a list of variables with the number of distinct missing and non-missing values. If the number of distinct non-missing values is zero, then the variable should be dropped. I used the ODS OUTPUT statement to get the list of variables in a data set. I used _CHAR_ as a shortcut to list all of the character variables.

ods output NLevels=checkchar;
ods html close; /* suppress default output, because I only want a data set*/
proc freq data=&dsout nlevels;
tables _char_ /noprint;
run;
ods html;
ods output close;
The resulting data set looks like this:

<table>
<thead>
<tr>
<th>Obs</th>
<th>TableVar</th>
<th>TableVarLabel</th>
<th>NLLevels</th>
<th>NMissLevels</th>
<th>NNonMissLevels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FILEID</td>
<td>File Identification</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>STUSAB</td>
<td>State Postal Abbreviation</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>SUMLEVEL</td>
<td>Summary Level</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>COMPONENT</td>
<td>geographic Component</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>LOGRECNO</td>
<td>Logical Record Number</td>
<td>15463</td>
<td>0</td>
<td>15463</td>
</tr>
<tr>
<td>6</td>
<td>US</td>
<td>US</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>REGION</td>
<td>Region</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>DIVISION</td>
<td>Division</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>STATECE</td>
<td>State (Census Code)</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>STATE</td>
<td>State (FIPS Code)</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

From this data set, I select only the variables to be dropped:

```sas
data dropchar (rename=(TableVar=vname));
set checkchar (where=(NNonMissLevels=0));
run;
```

I could have used PROC FREQ with TABLES _ALL_ to get the missing values for all variables, including numeric ones, but it is slower than PROC MEANS. If your data set is small and you are not concerned about coded missing values, or you have few numeric variables, then using PROC FREQ alone may be more convenient. However, if your dataset has millions of observations, it would be faster to first use both PROC MEANS (for numeric variables) and PROC FREQ (for character variables) on a subset of the observations. Only those variables whose values are all missing in that subset would be used in a PROC MEANS/PROC FREQ of the entire dataset. That would require more complicated code (see Appendix A). With any of these approaches, I could also choose to drop character variables with only one distinct value; the information they contain may be more appropriately placed in the metadata rather than the data set itself.

**COMBINING AND PARSING THE LIST OF VARIABLES TO DROP**

A simple SET statement in a DATA step is used to combine the three lists of variables to drop. The new list contains one observation per variable to drop. However, for maximum speed, I want to drop as many variables as possible in one DATA step. Therefore, the list of variables must be converted to the minimum number observations that can hold the list of variables. Since the maximum length of a character variable is 32,767, and the maximum length of a variable name is 32, a character variable should be able to hold a list of at least 992 SAS variable names. It may be able to hold more if the variable names are shorter.

First, I combine the three lists of variables. The result is one data set containing one observation for each variable whose values are all missing or all the specified coded value.

```sas
data allvarstodrop;
format vname $32.; /*maximum length of SAS variable name. Use this statement to make sure the names are not truncated*/
set dropchar(keep=vname) dropnumeric (drop=numhere) dropnumericalt (drop=greatest least) vars=varsize=length(vname)+1;
run;
```

Next, I joined the variable names into lists using the CATX function. When the list reaches the maximum size of a SAS character variable, or the last variable name is read, an observation is output.
DROPPING THE VARIABLES

CALL EXECUTE (Whitlock, 1997) is used to generate the code that drops the variables with missing values. For each observation in the varstodroplist data set, a DATA step is generated which drops a set of variables. The label and compression of the original data set are used for the new data set.

```sas
data varstodroplist (keep=varlist varsizesum);
format varlist $32767.; /*this is the maximum size of a character variable*/
retain varsizesum varlist;
set allvarstodrop end=eof;
varsizesum=sum(of varsize varsizesum);
if varsizesum>32767 then do;
  output;
  putlog vname varsizesum;
  varsizesum=varsize; varlist=vname; /*reset*/
end;
else varlist=catx(' ',varlist,vname);
if eof=1 then output;
run;
```

Here is an example of the code generated by one iteration of this DATA step, with the list of variables to drop shortened for clarity. The full code as written into the SAS log is shown in Appendix B.

```sas
data sas.acs2011_5yr_blkgrp2(label='2007-2011 US Census American Survey estimates and margins of error for NYS block groups, processed by gdb02' compress=BINARY);
set sas.acs2011_5yr_blkgrp2(drop=B25129m31 B25129m32 B25129m33 B25129m34 B25129m35 B25129m36...);
run;
```

CONCLUSIONS

This method for dropping variables whose values are all missing is fast and works on data sets with large numbers of variables. It complements other methods. Since it does not use macros or hash tables, the code is less complex, and easier to use and debug. It can also be used to remove numeric variables whose values have all codes for missing, such as -1 or 999999.

REFERENCES

- Carpenter, Arthur L. (2013) "How Do I...?" There is more than one way to solve that problem; Why continuing to learn is so important. Proceedings of the SAS Global Forum 2013. 029-2013 p.10-12.
APPENDIX A: FASTER VERSION OF CODE USING PRE-TESTING

This code is more complex, but faster than that discussed previously, especially for datasets containing hundreds of variables and millions of observations. The general approach is to pre-test a subset of observations using PROC MEANS or PROC FREQ to determine which variables contain only missing values in the subset. These variables are then tested in the entire dataset, and dropped if they contain only missing values.

/******************************************************
*the purpose of this program is to drop variables with
*missing values from a SAS dataset.
*This code is designed to have a similar function to
*The DROPMISS macro of Selvaratnam Sridharma
*see "Dropping Automatically Variables with Only Missing Values"
* SAS Global Forum 2010
*But the DROPMISS macro fails sometimes, so this is an alternative
*Note: this program is more complex than the 'drop missing variables' program
* but will be faster, especially on datasets with large numbers of observations
* it was tested on NYS outpatient data with about 653 variables and over 19 million observations
*programmed by Gwen LaSelva
*inputs:
  \library name
  name of input SAS dataset
  name of output SAS dataset

*/
alternative missing value for numeric variables, for example -1 or 999999
sample size for preliminary determination of which variables to drop
*outputs: SAS dataset*
******************************************************************************
/*specify any libraries used*/
libname sas 'E:\census\ACS\2007_2011\SASblockout';
libname op "P:\Sections\EPHTdata\SPARCSdeidentified\Outpatient";
/*specify the input and output datasets*/
%let dsin=op.outpatient_s_2011_prime;
%let dsout=sas.want;
/*specify an alternative missing value for numeric variables.
this value should be a number*/
%let altnumericmissvalue=0;
/*specify the number of observations to sample. The size should be a fraction of the total number of observations. It should be your best guess of the minimum number of observations needed to find non-missing values in all variables whose values are not all missing. Changing this will affect the speed of the program*/
%let samplesize=10000;
/*put the time in the log*/
data null;
temptime=put(time(),hhmm5.);
putlog "start time=" temptime;
run;
/*shouldn't need to modify below this line*/
/*STEP 1: PREPARATION*/
/*get the label and compression of the input dataset and put them into a macro variables,
use them at the end so the new dataset has the same label and compression as the original*/
proc contents data=&dsin noprint out=mycontents (keep=memlabel compress);
run;
data null;
set mycontents(obs=1);
call symputx("mylabel",memlabel);
call symputx("mycompress",compress);
run;
/*first, make output dataset to work with, so we don't accidentally mess
up the original dataset if something goes wrong*/
data &dsout (compress= &mycompress.);
set &dsin;
run;
/*STEP 2: FIND ALL NUMERIC VARIABLES WITH ALL MISSING VALUES*/
/*proc means with the N statistic should count number of non-missing values*/
/*start with a subset of observations*/
proc means data=&dsout (obs=&samplesize.) noprint;
output out=sampledropnumeric(drop=_:) n=;/*get the number of NON-MISSING for all numeric variables*/
run;
proc transpose data=sampledropnumeric out=sampledropnumeric2 (rename=(col1=numhere _name_=vname));
run;
data maybedropnumeric (keep=vname varsize);
set sampledropnumeric2 (where=(numhere=0));
varsize=length(vname)+1;
run;

/*test the variables who are all missing in the first subset of observations in the entire dataset*/
data numvarstotestlist (keep=varlist varsizesum);
format varlist $32767.; /*this is the maximum size of a character variable*/
retain varsizesum varlist;
set maybedropnumeric end=eof;
varsizesum=sum(of varsize varsizesum);
if varsizesum>32767 then do;
   output;
   putlog vname varsizesum;
   varsizesum=varsize; varlist=vname; /*reset*/
end;
else varlist=catx(' ',varlist,vname);
if eof=1 then output;
run;
data _null_; set numvarstotestlist ;
call execute("proc means data=&dsout noprint;");
call execute("output out=checknumeric1(drop=_:) n=;"между);
call execute("var");
call execute(varlist);
call execute(";run;");
if _n_=1 then call execute("data checknumeric; set checknumeric1; run;"));
else call execute("data checknumeric; merge checknumeric checknumeric1; run;"));
run;
proc transpose data=checknumeric out=checknumeric2 (drop=_label_ rename=(col1=numhere _name_=vname)); run;
data dropnumeric;
set checknumeric2 (where=(numhere=0)); run;
/*Use PROC FREQ to get character variables whose values are all missing it is slower than PROC MEANS*/
/*start with a subset of observations*/
ods output NLevels=maybedropchar;
ods html close; /*suppress default output, because we only want a dataset*/
proc freq data=&dsout (obs=&samplesize.) nlevels;
tables _char_ /noprınt;
run;
ods html;
ods output close;
data maybedropchar2 (rename=(TableVar=vname));
set maybedropchar (where=(NNonMissLevels=0));
varsize=length(TableVar)+1;
run;
/*test the variables who are all missing in the first subset of observations in the entire dataset*/
data charvarstotestlist (keep=varlist varsizesum);
format varlist $32767.; /*this is the maximum size of a character variable*/
retain varsizesum varlist;
set maybedropchar2 end=eof;
varsizesum=sum(of varsize varsizesum);
if varsizesum>32767 then do;
  output;
  putlog vname varsizesum;
  varsizesum=varsize; varlist=vname; /*reset*/
end;
else varlist=catx(' ',varlist,vname);
if eof=1 then output;
run;

data _null_; 
set charvarstotestlist ;
call execute("ods output NLevels=checkchar1;ods html close;");
call execute("proc freq data=&dsout nlevels; tables");
call execute(varlist);
call execute("/noprint; run; ods html; ods output close;");
if _n_=1 then call execute("data dropchar; set checkchar1 (rename=(TableVar=vname) where=(NNonMissLevels=0)); run;");
else call execute("data cropchar; set dropchar checkchar1 (rename=(TableVar=vname) where=(NNonMissLevels=0)); run;");
run;
/*You could use proc freq for all of the variables, but it would be slow*/

/*now, find out if any of the variables contain only the alternative missing value*/
/*let us assume that if the minimum and maximum values are both equal to the alternative missing value, the variable should be dropped*/
/*start with a subset of observations*/
proc means data=&dsout (obs=&samplesize.) noprint;
output out=maybebiggest(drop=_:) max=;
run;
proc means data=&dsout (obs=&samplesize.) noprint;
output out=maybesmallest(drop=_:) min=;
run;
proc transpose data=maybebiggest out=maybebiggest2 (drop= rename=(col1=greatest _name_=vname));
run;
proc transpose data=maybesmallest out=maybesmallest2 (drop= rename=(col1=least _name_=vname));
run;
proc sort data=maybebiggest2; by vname; run;
proc sort data=maybesmallest2; by vname; run;
data testnumericalt;
merge maybebiggest2 maybesmallest2;
by vname;
varsize=length(vname)+1;
if least=&altnumericmissvalue. and greatest=&altnumericmissvalue. then output;
run;
/*test the variables who are all the alternative missing value in the first subset of observations in the entire dataset*/
data numvarstotestlist (keep=varlist varsizesum);
format varlist $32767.; /*this is the maximum size of a character variable*/
retain varsizesum varlist;
set testnumericalt end=eof;
varsizesum=sum(of varsize varsizesum);
if varsizesum>32767 then do;
  output;
  putlog vname varsizesum;
  varsizesum=varsize; varlist=vname; /*reset*/
end;
else varlist=catx( ' ', varlist, vname);
if eof=1 then output;
run;

data _null_; set numvarstotestlist ;
call execute("proc means data=&dsout noprint;";
call execute("output out=biggest1(drop=_:) max=; var");
call execute(varlist);
call execute(";run;";
call execute("proc means data=&dsout noprint;";
call execute("output out=smallest1(drop=_:) min=; var");
call execute(varlist);
call execute(";run;";
if _n_=1 then do;
  call execute("data biggest; set biggest1; run;";
  call execute("data smallest; set smallest1; run;";
end;
else do;
  call execute("data smallest; merge smallest smallest1; run;";
  call execute("data biggest; merge biggest biggest1; run;";
end;
run;

proc transpose data=biggest out=biggest2 (drop= rename=(coll=greatest _name_=vname));
run;
proc transpose data=smallest out=smallest2 (drop= rename=(coll=least _name_=vname));
run;

proc sort data=biggest2; by vname; run;
proc sort data=smallest2; by vname; run;

data dropnumericalt;
merge biggest2 smallest2;
by vname;
if least=&altnumericmissvalue. and greatest=&altnumericmissvalue. then output;
run;
/*STEP 3: COMBINE LISTS OF CHARACTER AND NUMERIC VARIABLES TO DROP*/
/*for convenience, combine the lists of character and numeric variables to drop*/
data allvarstodrop;
format vname $32.; /*maximum length of SAS variable name. Use this statement to make sure the names are not truncated*/
set dropchar(keep=vname) dropnumeric (drop=numhere) dropnumericalt (drop=greatest least);
varsize=length(vname)+1;
run;
*proc print data=allvarstodrop (obs=10);run;

/*STEP 4: DROP THE VARIABLES*/
/* create a character variable containing the list of variables to be dropped. A character variable can be up to 32767 characters, and can therefore hold a list of at least 992 SAS variables, maybe more*/
/*if the sum of varsize is more than 32767 need to create additional observations in the dataset to hold the additional variables*/
data varstodroplist (keep=varlist varsizesum);
format varlist $32767.; /*this is the maximum size of a character variable*/
retain varsizesum varlist;
set allvarstodrop end=eof;
varsizesum=sum(of varsize varsizesum);
if varsizesum>32767 then do;
  output;
  putlog vname varsizesum;
  varsizesum=varsize; varlist=vname; /*reset*/
end;
else varlist=catx(' ',varlist,vname);
if eof=1 then output;
run;

/*use call execute to repeat the dropping for each set of variable names*/
data _null_;
set varstodroplist;
call execute("data &dsout(label='&mylabel' compress=&mycompress.);");
call execute("set &dsout(drop=");
call execute(varlist);
call execute("));");
call execute("run;");
run;

/*put the end time in the log*/
data _null_;
temptime=put(time(),hhmm5.);
putlog "end time=" temptime;
run;

APPENDIX B: CODE GENERATED BY THE FINAL DATA STEP CALL EXECUTE STATEMENTS
This code was created by one iteration of the final DATA step, using CALL EXECUTE statements. It is this code which drops the variables that were previously determined to have only missing values. The code is shown as it appears in the SAS log.

1219 + data sas.acs2011_5yr_blkgrp2(label='2007-2011 US Census American Survey estimates and margins of error for NYS block groups, processed by gdb02' compress=BINARY);
1220 + set sas.acs2011_5yr_blkgrp2(drop=
1221 + B25129m31 B25129m32 B25129m33 B25129m34 B25129m35 B25129m36 B25129m37 B25129m38 B25129m39
NOTE: There were 15463 observations read from the data set SAS.ACS2011_5YR_BLKGRP2.

NOTE: The data set SAS.ACS2011_5YR_BLKGRP2 has 15463 observations and 6462 variables.

NOTE: Compressing data set SAS.ACS2011_5YR_BLKGRP2 decreased size by 73.58 percent.
    Compressed is 4094 pages; un-compressed would require 15495 pages.

NOTE: DATA statement used (Total process time):
    real time           7.93 seconds
    cpu time            7.28 seconds