PharmaSUG2010 - Paper CC17

Looking for columns with all missing values

Binoy Varghese, Cybrid, Inc., Harrisburg, PA

ABSTRACT

The objective of this paper is to present a generic macro that will scan a data library to identify columns with all missing values, drop them from associated datasets, output lean datasets and generate a report for review. The metadata required to automate this processing is obtained using PROC DATASETS and Output Delivery System.

INTRODUCTION

Visually examining raw datasets for data issues is a basic step that is performed prior to programming of analysis datasets. Sometimes there are columns within datasets that have all missing values. The possibility of such an occurrence in a locked database is minimal but it is not uncommon to find such columns in unclean data that have been delivered to the programming team to get a head start on the creation of analysis datasets.

Identifying such cases in small datasets with few columns and observations may not be labor intensive. But, when it comes to thick (# columns) and tall (# observations) datasets (e.g. Phase III Lab data), this effort can consume programming hours that can otherwise be used more effectively. Regardless, it is beneficial to have a push button macro which will automatically perform this task.

This macro may also be used to locate uninitialized variables in analysis datasets. Although, such cases can be eliminated by careful programming and review of the log file, it may be worthwhile to check, just in case.

METHODOLOGY

The complete SAS Code for the macro: FIND_NULL_COL is presented in the subsequent section. This macro can be used without any modification in most cases.

Macro call parameters:

- INLIB = Required parameter that specifies LIBNAME of the source data library.
- DSN = Optional parameter to restrict processing to a specific dataset.
- OUTLIB = Optional parameter that specifies LIBNAME of the destination data library. If this parameter is specified, the macro will output datasets after dropping columns with all missing values.

The steps below briefly describe the logic used in developing the macro.

OBTAIN METADATA INFORMATION

Step 1 is to obtain metadata information about all datasets in the library using PROC DATASETS and Output Delivery System. The code below creates a dataset "VARLIST" with the required metadata information.

```
ods listing close;
ods output variables=varlist(keep=member variable);

proc datasets library=<library name> memtype=data;
     contents data=_all_;
     quit;
run;

ods output close;
ods listing;
```

Note: Column "MEMBER" in dataset "VARLIST" contains values with the <library name> prepended to the <dataset name>. The <library name> is removed for ease of use in later stages of the macro. This is done within a data step by using:

```
member=compress(tranwrd(member, upcase("<library name>."),''));
```

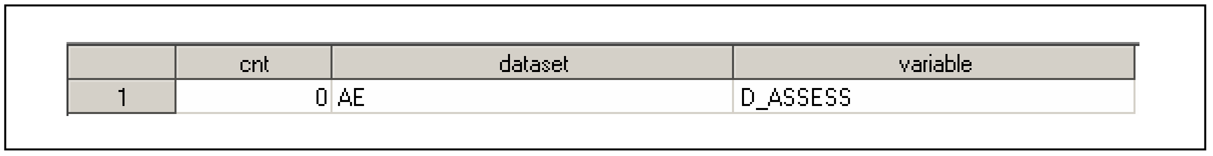
COUNT NON-MISSING VALUES FOR EACH COLUMN

Step 2 is to produce intermediate datasets which gives the count of non-missing values within a column. A PROC SQL query is used in conjunction with metadata information “VARLIST” and CALL EXECUTE data step function to achieve this on the fly.

The simplified version of the PROC SQL query used in the macro is shown below.

```
proc sql;
     select count(<column name>)
     from <library name>.<dataset name>
     where <column name> is not missing;
quit;
```

Each PROC SQL query produces a dataset for every column as shown below:



	cnt	dataset	variable
1	0	AE	D_ASSESS

Figure 1. Sample of intermediate dataset produced by PROC SQL query.

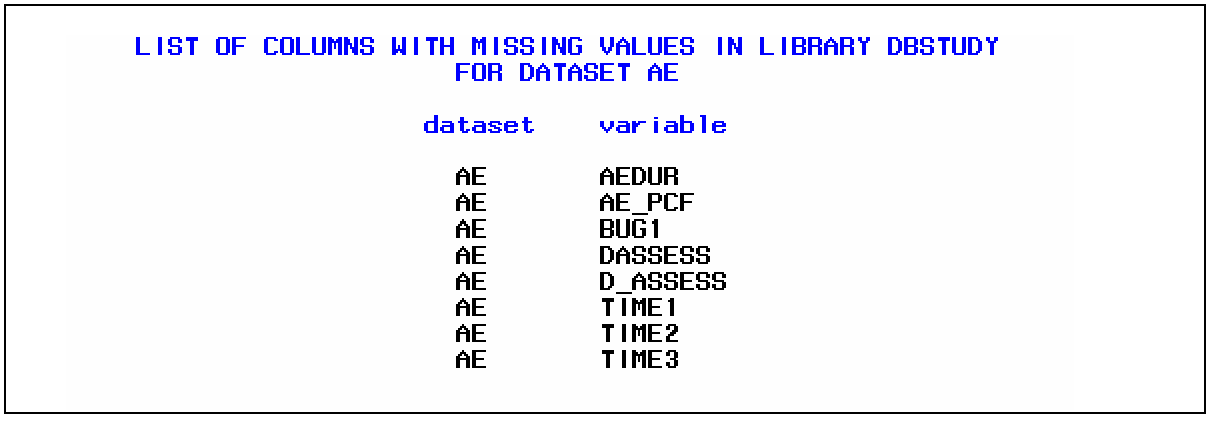
IDENTIFY COLUMNS WITH ALL MISSING VALUES

Step 3 is to append all intermediate datasets generated by the PROC SQL query. Columns with all missing values are identified using a where filter:

```
where cnt=0;
```

PRINT LIST FILE

Step 4 produces a list file of columns with missing values and the corresponding datasets.



```
LIST OF COLUMNS WITH MISSING VALUES IN LIBRARY DBSTUDY
                   FOR DATASET AE

              dataset     variable

                AE        AEDUR
                AE        AE_PCF
                AE        BUG1
                AE        DASSESS
                AE        D_ASSESS
                AE        TIME1
                AE        TIME2
                AE        TIME3
```

Figure 2. Sample of list file.

OUTPUT DATASETS

Step 5 is applicable only if the user has specified the destination data library. In this case, all datasets are copied from the source to destination data library using PROC DATASETS.

A data step with DROP statement is created on the fly for datasets containing columns with missing values. These datasets are read in from the output data library and written back to the output data library after removing these columns.

COMPLETE SAS CODE

```
*************************************************************************************;
* Purpose:                                                                          *;
*       - Find columns with missing values in a single dataset or entire data library  *;
*       - Output .lst file identifying these columns and associated datasets         *;
*       - Drop these columns and output the datasets (Optional)                      *;
*                                                                                   *;
* Macro Parameters:                                                                 *;
*       - INLIB:  LIBNAME of source data library. (Required)                        *;
*       - DSN:    Dataset name. (Optional)                                          *;
*                   If DSN is specified, this macro will look for columns with       *;
*                   missing values in the dataset specified, otherwise it will look  *;
*                   for such columns in all datasets                                *;
*       - OUTLIB: LIBNAME of destination data library. (Optional)                   *;
*                   If OUTLIB is not specified, this macro will output .lst file and, *;
*                   stop, otherwise, it will generate the .lst file, drop columns with *;
*                   missing values from the associated datasets and output them to   *;
*                   OUTLIB.                                                         *;
*************************************************************************************;

%macro find_null_col(inlib=, dsn=, outlib=);

options nofmterr;

/*
    obtain metadata information
*/

ods listing close;
ods output variables=varlist(keep=member variable);

proc datasets library=%upcase(&inlib) memtype=data;
    contents data=_all_;
    quit;
run;

ods output close;
ods listing;

data varlist(keep=member variable rename=(member=name));
    set varlist;
    member=compress(tranwrd(member,%upcase("&inlib.."),''));
run;

/*
    count non-missing values for each column
*/

data _null_;
    set varlist end=t;
    %if &dsn ne %then %do; where compress(name)=%upcase("&dsn"); %end;
     call execute(
                  "proc sql; create table ds" || compress(put(_n_,best.))
                || " as select count(" || compress(variable)
                || ") as cnt, '" || compress(name) || "' as dataset length=100,'"
                || compress(variable) || "' as variable length=100 from &inlib.."
                || compress(name) || " where "
                || compress(variable) || " is not missing; quit;"
                 );
    if t then call symput("cnt",compress(put(_n_,best.)));
run;

/*
    identify columns with all missing values
*/

data rpt;
    set
     %do index=1 %to &cnt; ds&index %end; ;
    where cnt=0;
    ;
run;
```

```
proc sql noprint;
select count(*) into: nvar from rpt;
quit;

/*
    print list file
*/

title1 "LIST OF COLUMNS WITH MISSING VALUES IN LIBRARY %UPCASE(&INLIB)";

%if &dsn ne %then %do; title2 "FOR DATASET %UPCASE(&DSN)"; %end;

%if &nvar = 0 %then %do;

data rpt;
    msg=" There are no columns with missing values.";
    output;
run;

proc print data=rpt noobs;
run;

%end;
%else %do;

proc print data=rpt noobs;
    var dataset variable;
run;

%end;

/*
    output datasets
*/

%if &outlib ne %then %do;

proc datasets nolist memtype=data;
    copy in=&inlib out=&outlib;
    %if &dsn ne %then %do; select &dsn; %end;
    quit;
run;

data _null_;
    length varlist $1000;
    retain varlist;
    set rpt;
    by dataset;
    if first.dataset then varlist='';
     varlist=left(trim(varlist)) || ' ' || trim(variable);
    if last.dataset then
        call execute(
                    "data &outlib.." || compress(dataset) || "(drop=" || trim(varlist)
                  || ");" || "set &outlib.." || compress(dataset)
                  || ";" || "run;"
                    );
run;

%end;

title1; title2;

proc datasets lib=work nolist kill;
quit;
run;

%mend  find_null_col;
```

CONCLUSION

FIND_NULL_COL macro can be used out-of-box to identify columns with all missing values. The programming uses PROC DATASETS over DICTIONARY tables to reduce processing time. PROC SQL enables to scan columns for missing values without knowing its data type. CALL EXECUTE data step function enables to write this macro with relatively fewer lines of code. The execution time of this macro depends on the size of datasets and the number of datasets being scanned. This macro has been successfully tested on raw data extracts of several studies using SAS version 9.2 on Windows Operating System.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Binoy Varghese
Enterprise: Cybrid, Inc
Address: 4807 Jonestown Road, Suite 253
City, State ZIP: Harrisburg, PA 17109
E-mail: mailme@binoyvarghese.com
Web: www.clinicalsasprogramming.com