Paper 4737-2020

Five Simple Ways to Know If Variables in a Table Are All Missing

Xia Ke Shan, iFRE Inc., Beijing, China; Kurt Bremser, Allianz Technology Austria

ABSTRACT

Before doing data analysis, we sometimes need to know if certain variables in a table are all missing, and need to drop these all-missing variables. This paper is introducing five simple ways to achieve this goal. Unlike data step code which needs more code to accomplish it, these five simple ways only cost several SAS® statements.

INTRODUCTION

Create some test data from a SASHELP dataset:

```
data have;
 set sashelp.class;
 call missing(name,weight,height);
run;
```

THE FIVE WAYS

PROC UNIVARIATE

This can only be applied to numeric variables, so it will not pick up NAME:

```
proc univariate data=have outtable=want noprint;
 var _numeric_;
run;

proc sql noprint;
 select _var_ into : missing_variables separated by ' '
  from want
    where _nobs_=0;
quit;

%put Missing variables are: &missing_variables ;
```

PROC MEANS

Like UNIVARIATE, this will only pick up numeric variables:

```
ods select none;
ods output summary=want;
proc means data=have n nmiss stackodsoutput;
 var _numeric_;
run;
ods select all;

proc sql noprint;
 select variable into : missing_variables separated by ' '
  from want
    where n=0;
```

```
quit;

%put Missing variables are: &missing_variables ;
```

PROC FREQ

This can be applied to both numeric and character variables:

```
ods select none;
ods output nlevels=want;
proc freq data=have nlevels;
 table _all_;
run;
ods select all;

proc sql noprint;
 select TableVar into : missing_variables separated by ' '
 from want
 where NNonMissLevels=0;
quit;

%put Missing variables are: &missing_variables ;
```

PROC SQL

This will be faster for big tables; it will pick up numeric and character variables:

```
/* firstl, get variable names */
proc transpose data=have(obs=0) out=vname;
 var _all_;
run;

/* then check if variable is all missing */
proc sql noprint;
select catx(' ','n(',_name_,') as',_name_) into :vnames separated by ','
 from vname;
create table temp as
select &vnames from have;
quit;

proc transpose data=temp out=want;
 var _all_;
run;

proc sql noprint;
 select _name_ into : missing_variables separated by ' '
 from want
 where col1=0;
quit;

%put Missing variables are : &missing_variables ;
```

PROC IML

```
proc iml;
  use have;
  read all var _char_ into char[c=vname_char];
  read all var _num_  into num[c=vname_num];
  close;
  missing_char=vname_char[loc(countn(char,'col')=0)];
  missing_num =vname_num [loc(countn(num ,'col')=0)];
  missing_var=missing_char//missing_num;
create want var{missing_var};
append;
close;
quit;

proc sql noprint;
 select missing_var into : missing_variables separated by ' '
 from want;
quit;

%put Missing variables are : &missing_variables ;
```

COMPARATIVE TESTS

Once again, let's make up some test data with more "meat":

```
data have;
 set sashelp.heart;
 call missing(sex,height,weight);
 do i=1 to 1000;
   output;
 end;
run;
```

Test all methods that can be applied to numeric and character variables:

Method	Time
PROC FREQ	7.42 seconds
PROC SQL	3.55 seconds
PROC IML	Ran out of memory

Table 1. Compare which one is fastest

Since IML needs to hold the complete table in memory, it will fail with sufficiently large datasets. SQL proves to be the fastest approach.

APPLYING THE METHODS ON ALL DATASETS IN A LIBRARY

Create several tables to search:

```
data have;
 set sashelp.class;
 call missing(age,name);
run;
data heart;
 set sashelp.heart;
 call missing(weight,height);
run;
data class;
 set sashelp.class;
run;
```

PROC FREQ

```
%let library=work; /*Here is the library I want to search*/

%macro find_missing_var(dsn);

ods select none;
ods output nlevels=temp;

proc freq data=&dsn nlevels;
table _all_;
run;

ods select all;

proc transpose data=temp(obs=0) out=vname;
 var _all_;
run;

proc sql noprint;
 select count(*) into :n
 from vname
 where upcase(_NAME_)='NNONMISSLEVELS';
quit;

%if &n=1 %then %do;

data temp1;
 length  table_name TableVar $ 100;
 table_name="&dsn";
 set temp(where=(NNonMissLevels=0));
run;

proc append base=want data=temp1(keep=table_name TableVar) force;
run;

%end;

%mend;

proc delete data=want;
run;

data _null_;
 set sashelp.vtable(
  keep=libname memname
  where=(libname="%upcase(&library)")
 );
 call execute(cats('%nrstr(%find_missing_var(',libname,'.',memname,'))'));
run;

title 'COLUMNS WITHOUT DATA';

proc print;
run;
```

PROC SQL

```
%let library=work; /*Here is the library I want to search*/

%macro find_missing_var(dsn);

proc transpose data=&dsn(obs=0) out=vname;
 var _all_;
run;
```

```
proc sql noprint;
select catx(' ','n(',_name_,') as',_name_) into :vnames separated by ','
 from vname;
create table temp as
 select &vnames from &dsn;
quit;

proc transpose data=temp out=temp1;
 var _all_;
run;

data temp2;
 length  table_name _name_ $ 100;
 table_name="&dsn";
 set temp1;
 if col1=0;
run;

proc append base=want data=temp2 force;
run;

%mend;

proc delete data=want;
run;

data _null_;
 set sashelp.vtable(
  keep=libname memname
  where=(libname="%upcase(&library)")
 );
 call execute(cats('%nrstr(%find_missing_var(',libname,'.',memname,'))'));
run;


title 'COLUMNS WITHOUT DATA';

proc print;
run;
```

PROC IML

```
%let library=work; /*Here is the library I want to search*/

proc iml;
dsn=datasets(&library);
do i=1 to nrow(dsn);
  use (dsn[i]);
  read all var _char_ into char[c=vname_char];
  read all var _num_  into num[c=vname_num];
  close;
  n_char=countn(char,'col');
  n_num=countn(num,'col');
  if ^ isempty(loc((t(n_char)//t(n_num))=0)) then do;
    idx=loc((t(n_char)//t(n_num))=0);
    name=(t(vname_char)//t(vname_num))[idx];
    table=repeat(dsn[i],nrow(name));

    want_table=want_table//table;
```

```
      want_name=want_name//name;
    end;
  end;
  create want var{want_table want_name};
  append;
  close;
  quit;

  title 'COLUMNS WITHOUT DATA';

  proc print noobs;
  run;
```

CONCLUSION

Unlike using a data step, these five ways use only several statements to accomplish the task, and are far more easy to use. PROC UNIVARIATE and PROC MEANS can only apply to numeric variables; PROC FREQ PROC SQL and PROC IML can apply to both numeric and character variables. PROC IML needs to load an entire table into memory, so IML cannot handle big tables and cost more time. PROC FREQ and PROC SQL both are multi-thread-capable which can handle a big table very quickly. And PROC SQL is more useful than PROC FREQ, for example, if I want to know the missing percent of lots of variables in a table, and save the result as a table. The following is the SQL code how to do it:

```
data have;
 set sashelp.heart;
run;

proc transpose data=have(obs=0) out=temp;
var _all_;
run;

data _null_;
 set temp end=last;
 if _n_=1 then call execute('proc sql;create table want as select ');
 call execute(cat('nmiss(',_name_,')/count(*) as ',_name_,'
format=percent8.2'));
 if last then call execute('from have;quit;');
 else call execute(',');
run;
```

REFERENCES

SAS Institute Inc. 2014. *SAS/IML® 13.2 User's Guide.* Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Xia Ke Shan
iFRE Inc.
Beijing, China
Phone: +8613521225927
E-mail: 12135835@qq.com