PharmaSUG 2012 - Paper CC24

Need a hand to locate your variables in an entire library?

Ying Guo, PAREXEL International, Indianapolis, IN

ABSTRACT

The ability to quickly identify the location of a variable in an entire library is critical during table, figure and listing (TFL) development, especially, when there is only limited information available for some of the variables. For example, sometimes, there may only be a partial variable name available, or perhaps you may only know one of the values for a variable. It is really time consuming to manually search for the expected variable in an entire library. Instead of opening each dataset and manually searching for the expected variables, the macro presented in this paper is a SAS® macro program designed to identify the location of a variable in a time-efficient and user-friendly way. It can automatically search all variables in a library regardless of the dataset structure and variable names. The macro is designed to obtain the location of a variable from one of following conditions: 1) A known partial or full variable name; 2) A known partial or full variable label; 3) A known partial or full variable value. The output of this macro will provide the dataset name and the full variable name of the variables searched. Use of this macro will significantly shorten search time, and help to increase TFL production efficiency.

INTRODUCTION

When we encounter a large number of datasets, it is not easy to find one or several variables quickly from the entire library. Programmers have many styles finding their variables. The most popular way is to use proc contents and proc datasets to list all variables and labels. However, if you only know the value of one variable, then you need to search all datasets values to find it. Here are some examples of how a programmer can search for a variable.

```
/*create content information using proc contents*/;
Proc contents data=<dataset name>;
Run;

/*create datasets information using proc datasets*/;
Proc datasets lib=<library name>;
Run;

/*search variables contained certain value*/;
Data temp;
  Set <dataset name>;
  Where index(<variable name>, "%%%%");
Run;
```

In order to search variables which only values are known, each variable need to be check. The dataset names need to be changed manually for each dataset checked. The variable names also needed for manual input. If you have a library of 10 datasets and an average of 50 variables in each dataset, then you'll have 10 X 50 = 500 data steps to do so. This will be time consuming. Locating a variable in an entire library is critical during table, figure and listing. Without an efficient way to do so will significantly delay our programming process.

SOLUTION

In order to expedite reviewing of the datasets and locate the variables in a more time-efficient way, a SAS macro is developed. This macro will automatically process and check each variable name and values in each datasets for giving values. When giving values are found, the macro will automatically generate an exception dataset which will contain the dataset name and variable name for desired variables. The exception dataset will be output to an rtf file. There is no need to type in variable names and dataset names; all that is required is to specify the name and the location of the input library and the output rtf file.

In this macro, sas resource table is used to generate variable names and variable labels. After libname is assigned, SAS will generate a datasets including all variable details. The following procedure will get data from SAS resource.

```
/*get datasets information*/;
proc sql noprint;
  create table ds as
```

```
  select * from dictionary.tables
  where libname=upcase("&lib");
quit;

/*get variable information*/;
proc sql noprint;
  create table cont as
  select * from dictionary.columns
  where libname=upcase("&lib");
quit;
```

After getting datasets and variable information, auto check will be performed for each character variable. Following functions are used to serve different purposes.

a. Index function is used to search for the values which are partially known. For example, if you'd like to find any variables that have headache information; this function will output all variables which include headache as full/partial value. That means variable contained "severe headache", "headache recovered" are all included.
b. Like function is used for the values which known value is not exactly matched. For example, if you want to find a variable which has a name of Chris Smith, but you are not sure is Chris Smith or Christopher Smith. If you just put Chris, you may include all people with Chris as the first name. Using like function as "like 'Chris%Smith'" will solve the problem.
c. Sound like function are used in this procedure in case you may only know how to pronounce it, but not aware of how it spells. For example, a person named Chris may spell like Kris. In this case, sound like will pick up them all.

These functions make the search powerful.

THE OUTPUT OF THE MACRO

The macro produces a rtf output. The report will list all the name of the qualified variables and their datasets name.

dataset name	variable name
GYN2Q	PROJCODE
IFO	PROJCODE
IFO7Q	PROJCODE
IPT1	PROJCODE
IPT2	PROJCODE
ME_1Q	PROJCODE
GYN2Q	STUDY
GYN2Q	PROTNO
GYN2Q	PID
IFO3	STUDY
IFO3	PROTNO
IFO3	PID
IFO7Q	STUDY
IFO7Q	PROTNO
IFO7Q	PID
ME_1Q	STUDY

dataset name	variable name
ME_1Q	PROTNO
ME_1Q	PID

Figure 1. This showed a sample figure.

SUMMARY

In summary, this macro is a very useful tool to find the variable efficiently. It can not only check all variable names and labels, but also check the data values. This will enhance the ability of the macro to find ideal variables. In addition, the macro will perform automatically, regardless of different structures and names for datasets; therefore, checks can be performed in a time-efficient manner. Finding variables quickly is critical in SAS programming, especially for TFL production. This macro will increase the efficiency of daily work.

CONCLUSION

This macro is simple and practical. It will be a very useful tool for people who want to review data quickly and correctly.

CONTACT INFORMATION

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

Name: Ying Guo
Enterprise: PAREXEL International
Address: 2520 Meridian Parkway, Suite 200
City, State ZIP: Durham NC 27713
Work Phone: 1-978-311-1811
E-mail: evelyn.guo@parexel.com
Web: www.pxl.com

APPENDIX

```
%macro find(lib=test, var=code, label=age, phase=1158, likep=1%8, soundlp=new,
outfile="&_spec.test.xls");

proc sql noprint;
  create table ds as
  select * from dictionary.tables
  where libname=upcase("&lib");
quit;

proc sql noprint;
  create table cont as
  select * from dictionary.columns
  where libname=upcase("&lib");
quit;

/*find var when partial/full var name is known*/;
data final(keep=memname name label);
  set cont;
  if index(upcase(name), upcase("&var")) then output;
  if index(upcase(label), upcase("&label")) then output;
run;


/*find var when var name is unknown*/;

data _null_;
  set ds;
  call symput('dsname'||left(put(_n_,8.)),memname);
run;



/* determine number of variables in data set to establish a counter*/;

proc sql noprint;
  select count(*) into :dscnt from ds
quit;


%do i=1 %to &dscnt;


/*count variables in datasets*/;
proc sql noprint;
  select count(*) into :varcnt&i from cont
  where memname="&&dsname&i";
quit;

/*read variable names in datasets*/;
data _null_;
  set cont;
  where memname="&&dsname&i";
  call symput('vname'||left(put(_n_,8.)),name);
  call symput('vtype'||left(put(_n_,8.)),type);
run;

/*find the Character variables*/;

%do j=1 %to &&varcnt&i;
  %if &&vtype&j=char %then %do;
```

```
    data temp1;
      set &lib..&&dsname&i;
      if index(upcase(&&vname&j), upcase("&phase")) then output;
      if &&vname&j like "&likep" then output;
      if &&vname&j=*"&soundlp" then output;
    run;

    proc sql noprint;
      select count(&&vname&j) into:obs from temp1;
    quit;

    %if &obs ne 0 %then %do;
      data temp;
        length memname $32. name $32.;
        memname="&&dsname&i";
        name="&&vname&j";
      run;

      data final;
        set final temp;
      run;

    %end;
  %end;
%end;


%end;
%let outfile=&_spec.test.rtf;
ods rtf file="&outfile";

proc report data=final nowindows headline headskip;
  column memname name;
  define memname / width=35 left "dataset name" flow id;
  define name / width=30 left "variable name";
  compute after ;
  ;
  endcomp;
run;

ods rtf close;

%mend;
```