PhUSE 2008

Paper PO09

Let SASHELP Help You

Catriona O'Hare, ICON Clinical Research, Dublin, Ireland

ABSTRACT

SASHELP tables are data views in the SASHELP Library. They automatically collect and store information about the current SAS session and provide a wealth of information about your SAS session that can be accessed to improve and automate your code. The main SASHELP views I will explore are VCOLUMN, VTABLE, and VSLIB. Using these, I will present some simple techniques to make programming easier and more reusable across studies and explore some techniques to help in the programming of CDISC datasets. I will also explore a way to use SASHELP to present a summary of the datasets created e.g. size, number of observations, number of variables and date created.

INTRODUCTION

In SAS, there is a library called SASHELP. It is made available at the start of a SAS session and contains views which are read-only SAS Datasets. The view contains information about the current SAS session. The information ranges from information on active datasets, active paths, to information of active macros.

In the pharmaceutical industry, programmers find themselves doing the same tasks over and over again. So we strive to make code more reusable, dynamic and cut down the possibility of mistakes. Also we would like to make code dynamic so little alteration is needed when reusing the code and therefore cutting down time spent on tasks. In this paper I explore some examples of how SASHELP views can be utilized to make processes more automated and dynamically written.

SASHELP VIEWS

SASHELP views are read-only objects that store dynamic information about a SAS session. These views are part of the SAS System and are stored in the SASHELP data library. They where introduced in Version 6 and much of the information found in these views would be very difficult to obtain otherwise. They behave in the same way as SAS datasets and can be manipulated similarly using the data and procedure steps.

The types of information in the views are:

- libraries, catalogs, and data sets
- external files allocated to the session
- macro & data set variables attributes
- indexes, titles, footnotes, and views
- system option settings

SASHELP views available.

SASHELP VIEWS	CONTAINS
VCATALG	Lists information for each SAS catalog: name, location, and type of catalog
VSCATLG	Lists all SAS Catalogs and the libraries they are located in.
VCOLUMN	Lists data set variable details such as type, length, position, format, label and indexing
VEXTFL	Lists the file references assigned in the SAS session and name of the external file they are assigned to.
VINDEX	Lists indexes, datasets where it is located and variables they index

VMACRO	Lists details about macro variables such as scope and current value
VMEMBER	Lists all data sets, views, and catalogs
VOPTION	Lists information of status of system options: name, description, current setting and scope
VSACCES	Lists all SAS/ACCESS views available in the session
VSLIB	Lists all library names and associated paths
VTABLE	Lists all Library Reference, data set name and other information for every dataset/view available in the SAS session.
VSTABLE	Lists data set names and libraries they are located in.
VSTABVW	Lists data set and view names and libraries they are located in.
VVIEW	Lists all views and the engine in SASHELP
VSVIEW	Lists all views in SASHELP
VTITLE	Lists location, number and value of each title and footnote available in the SAS session
VSTYLE	Lists information related to select ODS styles.

USING SASHELP

There are many ways of using SASHELP to help you. You can compare datasets in directories, look up the path of source data and use to validate datasets. Below are 5 examples of using SASHELP views VCOLUMN, VTABLE and VSLIB.

EXAMPLE 1

A requirement of a submission to the FDA is to provide CDISC datasets as XPT files. Below is an example converting a library where the CDISC datasets are located to XPT files in another area using the SASHELP.VTABLE. The Advantage of the code is that it does not matter what is in the library, it will always work. Therefore if a new dataset had to be created at a later date it will still work. It can be reused for different projects by just changing the RAW and PATH macro variables.

```
*******************************************************;
%let raw=raw;              /* library where the datasets are located*/
%let path=h:\99999\xpt;    /* path where XPT files output to*/
*******************************************************;

%macro export(dset);

    libname trans XPORT  "&path\&dset..XPT";

    proc copy in=&raw out=trans memtype=data;
    select &dset.;
    run;

%mend export;

data _null_;
set sashelp.vtable (where=(libname=upcase("&raw")));
  memname=upcase(memname);
  /*calling the export macro for each dataset*/
  call execute('%export('||memname||')');
run;
```

EXAMPLE 2

Below is an example of using VCOLUMN. When converting data to CDISC, SDTM or any other type of Specification, a programmer sometimes needs to know all the visits that are in the raw data as sometimes they need to change the format of the visit. Below is an example of how to pull all datasets from a library and find a list of Visits.

```
*******************************************************;
```

```
%let raw=rawa;            /* library where the datasets are located*/
*******************************************************;

data _null_;
set sashelp.vcolumn;
/* find the names of the datasets that has visits in it*/
 where libname=upcase("&raw") and upcase(name) = ('VISIT');
    n+1;
/* and create macros variables using VCOLUMN so can set all*/
/* datasets together*/
    call symput('dataset'||compress(put(n,best.)),compress(memname));
    call symput('n',compress(put(n,best.)));
run;
*******************************************************;
/* Set all datasets with visits together*/
%macro visit;

   data all;
   set
   %do i=1 %to &n;
      &raw..&&dataset&i
    %end;
    ;
    keep visit visitnum;
    run;

%mend visit;
%visit;

*******************************************************;
/* to create final dataset with one observation for each visit*/
proc sort data=all out=visits nodupkey;
by visitnum visit;
run;
```

EXAMPLE OF VISITS OUTPUT

VISIT	VISITNUM
SCREENING	0
DAY 1	1
DAY 2	2
DAY 3	3
DAY 4	4
DAY 5	5
DAY 6	6
STUDY TERMINATION	90
UNSCHEDULED	99

EXAMPLE 3

Below is an example of using SASHELP views to help with validating datasets. To check if a variable label is missing in any dataset in a library the following can be used.

```
Data _null_;
Set sashelp.vcolumn(where =(upcase(libname='RAW'));
     If missing(LABEL) then put "WARNING: The variable "  NAME  "from  " LIBNAME
"." MEMNAME "does not have a label";
Run;
```

EXAMPLE 4

Below is an example of using SASHELP views to help in CDISC SDTM model in the creation of SV (subject visits) dataset. SV dataset lists all the visits for each subject with start date of visit and end date of visit. Below is an example of using SASHELP view VCOLUMN to work out the start and end date of the each visit by using the CDISC SDTM datasets already created.

```
*******************************************************;
%let raw=sdtm;            /* library where the raw datasets are located*/
*******************************************************;

/**  Finds all the name of the SDTM datasets */
proc sort data=sashelp.vcolumn out=vcolumn;
where libname=upcase("&raw") ;
by memname name;
run;

/**  Finds all the SDTM datasets that have variable VISITNUM in it*/
proc sort data=vcolumn out=visdata(keep=memname) nodupkey;
where name='VISITNUM';
by memname ;
run;

data sv1 ;
merge  vcolumn visdata(in=a);
 by memname ;
/* keep only datasets that have visitnum variable and have **DTC variable and*/
/* are not SV or SE. (if rerunning creating of SV makes sure not to reuse SV)*/
  if a and memname not in ('SV','SE')  and name=substr(name,1,2)||'DTC';
   n+1;
/* create macros variables so can set all datasets selected together*/
  call symput('name'||compress(put(n,best.)),compress(name));
  call symput('dset'||compress(put(n,best.)),compress(memname));
  call symput('n',compress(put(n,best.)));
run;

*******************************************************;
/* Set all datasets together removing any visitnum/Date that are missing*/
%macro sets;

     data sv2;
     set
        %do i=1 %to &n;
         &raw..&&dset&i(rename=(&&name&i=datetime))
        %end;
          ;

         if missing(datetime) or missing(visitnum) then delete;

         keep visit visitnum datetime usubjid  ;
     run;

%mend sets;
%sets;

*******************************************************;
/* sort data to have all date of each visit in order*/
proc sort data=sv2 out=sv3 nodupkey;
 by usubjid visitnum  datetime;
run;

/* sort data to find first and last date per subject and visit*/
data svstart(rename=(datetime=start)) svend(rename=(datetime=end));
```

```
set sv3;
 by usubjid visitnum  datetime;
     if first.visitnum then output svstart;/* first date*/
     if last.visitnum then output svend; /* last date*/
run;


*******************************************************;
/* find the svstdtc and svendtc for CDISC SDTM dataset SV*/
data sva;
merge svstart svend;
 by usubjid visitnum ;
   svstdtc=scan(start,1,'T');
   svendtc=scan(end,1,'T');
run;
```

EXAMPLE: SVA Output

USUBJID	SVSTDTC	SVENDTC	VISIT	VISITNUM
9999-01	2006-10-12	2006-10-12	SCREENING	1
9999-01	2006-10-13	2006-10-23	PERIOD 1	2
9999-02	2006-10-11	2006-10-18	SCREENING	1
9999-02	2006-10-19	2006-10-23	PERIOD 1	2

EXAMPLE 5

A lot of different information can be pulled from SASHELP views. Below is a macro that creates an rtf Microsoft word file with information for different libraries with examples of output. This can be useful when datasets and information about datasets have to be handed over to another department or to a sponsor. The output contains the location of the libraries, name, label, size, number of observations, number of variables and last modified date time of the datasets in the each library. The SASHELP views used are VTABLE and VSLIB.

This is also a good help for validation for when you are creating ADAM CDISC datasets from SDTM CDISC datasets. You can check if the number of observations is the same if applicable, and the date of the SDTM datasets are before the ADAM datasets and therefore seeing if the correct datasets have been used.

Information for Datasets
Created by: oharec Date:01SEP08

Location: H:\\XXXX\999-999\\DATASETS\RAW					
Dataset	**Dataset Label**	**Size**	**Number of Observations**	**Number of variables**	**Last Modified Date Time**
AE	Adverse Events	65 KB	40	29	29AUG2008:16:09:55
CM	Concomitant Medications	113 KB	111	21	29AUG2008:16:09:55
Location: H:\\XXXX\999-999\\DATASETS\ANALYSIS					
Dataset	**Dataset Label**	**Size**	**Number of Observations**	**Number of variables**	**Last Modified Date Time**
ADAE	Adverse Events	129 KB	40	59	01SEP2008:13:01:16
ADCM	Concomitant Medications	305 KB	111	57	01SEP2008:13:01:18

```
%macro datasetinfo(libs=,path=);
```

```
*********************************************************************;
** bring in need SASHELP views*;
proc sort data=sashelp.vtable out=vtable;
by libname;
run;

proc sort data=sashelp.vslib out=vslib ;
by libname;
run;

/* merge view together by libraries*/
data info;
merge vtable vslib;
 by libname;

** works out how many and what libraries are needed per &libs**;
%let i = 1 ;
   %let lib&i = %scan ( &libs, &i) ;
   %do %while ( &&lib&i ^= ) ;
    %put libname # &i = &&lib&i;
      %let i = %eval ( &i + 1 ) ;
     %let lib&i = %scan ( &libs , &i) ;
   %end;
  %let nlib=%eval(&i-1);
  %put # of libnames = &nlib;
   %do i=1 %to &nlib;
    if libname=upcase("&&lib&i") then lib=&i;
   %end;
   if missing(lib) then delete;

run;

*********************************************************************;
/* create dataset for final output, find path, size, creating macros*/
/* for paths and libraries*/
data info2;
 length memlabel $60.;
  set info;

       ** path of libraries**;
      if index(path,"'") then pathname=scan(compress(path,'()'),2,"'");
       else pathname=path;


     **To find size ;
      size = left(trim(put(((npage * bufsize)/1024)+1,best.)||' KB'));

     **format date Modified;
      format modate datetime19.;

    ** create macros of paths and libraries*;
      call symput('path'||compress(put(lib,best.)),upcase(pathname));
      call symput('lib'||compress(put(lib,best.)),libname);

    keep memname modate nobs nvar size memlabel libname lib pathname;
 run;

*********************************************************************;
 ** output data into a RTF file using proc report*;
*********************************************************************;
  options number nodate orientation=portrait papersize=A4;
```

```
  ods rtf file="&path\Datasets_info.rtf" bodytitle startpage=yes;

  title1 j=c  f=Times bold 'Information of Datasets';
  title2 j=c  f=Times bold "Created by: &sysuserid              Date:&sysdate";

proc report data=info2 ls=119 ps=45 split='#' spacing=1 nowindows center missing  ;
column  lib memname memlabel size nobs nvar modate ;

define lib      /  order   noprint;
define memname  /  display left    flow    'Dataset';
define memlabel /  display left    flow    'Dataset Label';
define size     /  display center  flow    'Size';
define nobs     /  display center  flow    'Number of#Observations';
define nvar     /  display center  flow    'Number of#variables';
define modate   /  display left    flow    'Last Modified Date Time';

/* Break after each library so each library start in new page*/
break after lib / page;

/* to output path of library at the top of each page*/
compute before _page_;
 line ' ';
 length txt $80.;
 %do j=1 %to &nlib;
       if lib=&j then txt="Location: &&path&j";
 %end;
  line @1 txt $80.;
endcomp;

run;

  ods rtf close;
  quit;



%mend datasetinfo;

/* MACRO CALL*/
%datasetinfo(libs=RAW ANA         /* libraries wanted in documents*/,
             path=H:/9999         /* path want document to go*/ );
```

CONCLUSION

In this paper I have given examples of using SASHELP which demonstrates the advantages that can be gained by using SASHELP. From the examples you can see that a SASHELP view creates endless possibilities in the realm of automating and improving programming and validation techniques. SASHELP views are full of information that can be exploited to make you a more dynamic programmer, whatever your level.

CONTACT INFORMATION

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

Catriona O'Hare
ICON Clinical Research
South County Business Park
Leopardstown
Dublin 18
Ireland

Work Phone: +353 1 291 2312

Fax: +353 1 291 2723
Email: catriona.ohare@iconplc.com
Web: www.iconclinical.com