Paper CC-11

Using DICTIONARY Views to Eliminate Tedious Visual Review

Christine Davies, RTI International, Durham, NC

ABSTRACT

The SASHELP DICTIONARY views are a powerful tool that may be overlooked at times. As an example of their utility, they were used to overcome the challenge of inconsistent naming of datasets across years for a project. This is illustrated in the example code where the DICTIONARY views in the SASHELP library were used to populate macro variables with dataset names, which were then used in conjunction with the FILEEXIST function to perform conditional processing on the dataset based on date last modified.

INTRODUCTION

Ever try to set up code that would access datasets from multiple years of a project? Ever run into inconsistent dataset naming across that project's years? When posed with the challenge of developing code that would summarize sample sizes across each year of a project, many challenges were encountered, such as inconsistent naming of the datasets and datasets not existing for a particular timeframe due to sampling issues or fielding issues. With the help of the automatically session-generated SASHELP Views and the FILEEXIST function, a solution was in sight!

The code described in this paper was developed for a project in which many samples were selected per year for several years. Each sample was optionally followed by up to two supplemental samples. The goal was to print a report that listed the sample size of every main sample and supplemental sample for all years of the project. For this particular project, there were between from 50-100 libraries that needed to be checked for any given year, but for simplicity only one library will be used to illustrate the solution.

DATASETS USED IN THIS PAPER

d01_sample_1 is the name always used for the main sample file.
d0*_sample_2 is always used for the first supplemental sample.
    The third character of this dataset name is inconsistent.
d0*_sample_3 is always used for the second supplemental sample.
    The third character of this dataset name is inconsistent.

USING SASHELP DICTIONARY VIEWS

SAS® DICTIONARY tables and views can be used to obtain information about SAS files. The information is generated at runtime and available once a session is started. SAS users can quickly and conveniently obtain useful information about their SAS session with a number of read-only SAS DICTIONARY tables or SASHELP views. At any time during a SAS session, information about currently defined system options, libnames, tables, columns and their attributes, formats, indexes, and more can be accessed and captured. In this example, the SASHELP.VTABLE view is used.

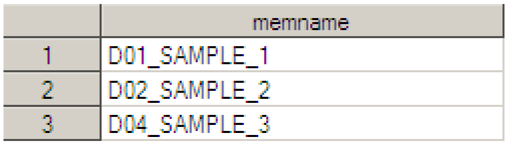
VTABLE displays information about all of the datasets in all defined libraries for the current SAS session. This is a convenient way to identify the names of the datasets in a particular library. Since for this project's purpose, the dataset names needed to be identified and then used later in the code for conditional processing, the VTABLE was used to create a temporary dataset that contained the names of the datasets and then CALL SYMPUTX was used to create macro variables out of the dataset names.

CODE TO CREATE TEMPORARY DATASET OF DATASET NAMES

```
proc sql;
      create table all_sample_files as
      select memname from
      sashelp.vtable
```

```
            where libname="%upcase(out)" and
                  substr(memname,5,6)="%upcase(sample)";
            quit;
```

RESULTING DATASET



	memname
1	D01_SAMPLE_1
2	D02_SAMPLE_2
3	D04_SAMPLE_3

CODE TO CREATE MACRO VARIABLES

```
data _null_;
      set all_sample_files;
      if substr(memname,12,1)='1' then call symputx("sample",memname);
      if substr(memname,12,1)='2' then call symputx("supp1",memname);
      if substr(memname,12,1)='3' then call symputx("supp2",memname);
run;
```

Note these macro variables are created inside a macro, and will be set to blank if the dataset name does not exist in the variable MEMNAME in the all_sample_files dataset.

Once the dataset names have been defined as macro variables, it is time to check if the sample file exists in the directory and to see when it was last modified. If we were interested in summarizing frame sizes, sample sizes, and supplemental sample sizes for calendar year 2009, then we would be interested in all sample files that were last modified between 12/2008 and at anytime during 2009. Typically, sample files created in December in the preceding year aren't fielded until January of the following year, so we want to be sure to include these in our summary findings.

CODE TO CHECK FOR FILE'S EXISTENCE

```
 %if %sysfunc(fileexist("...\&sample..sas7bdat")) %then %do;
```

FILEEXIST returns 1 if the external file exists and 0 if the external file does not exist. Although your operating environment utilities might recognize partial physical filenames, you must always use fully qualified physical filenames with FILEEXIST. If the file does exist then additional processing will be done based on the date last modified. This date can be obtained from the same useful VTABLE which was originally used to identify the dataset names.

```
            proc sql;
                  create table mod_date_file as
                  select modate from
                  sashelp.vtable
                  where libname="%upcase(out)" and
                  memname="%upcase(&sample.)";
            quit;
```

Since we are interested in the month and year of the date last modified but the MODATE variable is in date time format, we need to break apart MODATE from the VTABLE into these two pieces.

CODE TO CREATE MONTH AND YEAR MACRO VARIABLES FROM MODDATE

```
            data _null_;
                  set mod_date_file;

                  mod_date=datepart(modate);
                  year_taken=year(mod_date);
                  month_taken=month(mod_date);

                  call symput("year_taken",put(year_taken,8.0));
                  call symput("month_taken",put(month_taken,8.0));
            run;
```

Using the following code, the conditional processing can begin. This code is embedded within the FILEEXIST check and if the date last modified meets the time period under analysis then the sample size is taken from the sample file.

CODE TO DETERMINE SAMPLE SIZE

```
%if (&month_taken=12 and &year_taken=2008) or &year_taken=2009
%then %do;

            proc sql;
                  select nobs into :n
                  from
                  sashelp.vtable
                  where libname="%upcase(&LIB.)" and
                  memname="%upcase(&sample.)";
            quit;

%end;
```

The preceding code was repeated for both of the supplemental samples, if they exist. See Appendix for full code listing.

CONCLUSION

By using SASHELP.VTABLE we are able to produce the desired output for sample sizes and supplemental sample sizes without needing to know if the files first exist and without needing to know their exact dataset name. This method saved our project considerable amount of time and labor by eliminating the need for visual review.

REFERENCES

Lafler, Kirk Paul. "Exploring DICTIONARY Tables and Views", *Proceedings of the 30th Annual SASUsers Group International Conference*. April 2005. Available at http://www2.sas.com/proceedings/sugi30/070-30.pdf

SAS site http://support.sas.com/documentation/cdl/en/lrdict/62618/HTML/default/a000210912.htm

ACKNOWLEDGEMENTS

Thanks to Joey Morris, Ryan Whitworth, Susan McRitchie, and Laurie Cluff for reading the draft of this paper and providing comments.

CONTACT INFORMATION

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

Name: Christine Davies
Enterprise: RTI, International
Address: 3040 East Cornwallis Road
City, State ZIP: Durham, NC 27709
Work Phone: 919-267-3747
E-mail: cdavies@rti.org

APPENDIX

```
*-----------------------------------------------------------------
ENTIRE CODE
------------------------------------------------------------------;


%let lib=out;
%let loc=C:\Temp\;


%macro c;
/*GRAB NAMES OF SAMPLE FILE AND THE SUPPLEMENTAL SAMPLE FILES*/
proc sql;
      create table all_sample_files as
      select memname from
      sashelp.vtable
      where libname="%upcase(&LIB.)" and
                  substr(memname,5,6)="%upcase(sample)";
quit;

data _null_;
      set all_sample_files;
      if substr(memname,12,1)='1' then call symputx("sample",memname);
      if substr(memname,12,1)='2' then call symputx("supp1",memname);
      if substr(memname,12,1)='3' then call symputx("supp2",memname);
run;

/*Get Sample Count for First Sample if it exists*/
%if %sysfunc(fileexist("&loc.\&sample..sas7bdat")) %then %do;
            proc sql;
                  create table mod_date_file as
                  select modate from
                  sashelp.vtable
                  where libname="%upcase(&LIB.)" and
                  memname="%upcase(&sample.)";
            quit;

            data _null_;
                  set mod_date_file;
                  if modate eq . then stop;
                  mod_date=datepart(modate);
                  year_taken=year(mod_date);
                  month_taken=month(mod_date);
                  call symput("year_taken",put(year_taken,8.0));
                  call symput("month_taken",put(month_taken,8.0));
            run;

            %if (&month_taken=12 and &year_taken=2008) or &year_taken=2009
            %then %do;

                  proc sql;
                        select nobs into :n
                        from
                        sashelp.vtable
                        where libname="%upcase(&LIB.)" and
                        memname="%upcase(&sample.)";
                  quit;
            %end;
%end;

/*Get Sample Count for First Supplemental Sample if it exists*/
%if %sysfunc(fileexist("&loc.\&supp1..sas7bdat")) %then %do;
            proc sql;
                  create table mod_date_file2 as
                  select modate from
                  sashelp.vtable
                  where libname="%upcase(&LIB.)" and
                  memname="%upcase(&supp1.)";
            quit;

            data _null_;
                  set mod_date_file2;
                  if modate eq . then stop;
                  mod_date=datepart(modate);
                  year_taken2=year(mod_date);
                  month_taken2=month(mod_date);
                  call symput("year_taken2",put(year_taken2,8.0));
                  call symput("month_taken2",put(month_taken2,8.0));
            run;

            %if (&month_taken2=12 and &year_taken2=2008) or &year_taken2=2009
            %then %do;

                  proc sql;
                        select nobs into :n_s1
                        from
                        sashelp.vtable
                        where libname="%upcase(&LIB.)" and
                        memname="%upcase(&supp1.)";
                  quit;
            %end;

%end;

/*Get Sample Count for Second Supplemental Sample if it exists*/
%if %sysfunc(fileexist("&loc.\&supp1..sas7bdat")) %then %do;
            proc sql;
                  create table mod_date_file3 as
                  select modate from
                  sashelp.vtable
                  where libname="%upcase(&LIB.)" and
                  memname="%upcase(&supp2.)";
            quit;

            data _null_;
                  set mod_date_file2;
                  if modate eq . then stop;
                  mod_date=datepart(modate);
                  year_taken3=year(mod_date);
                  month_taken3=month(mod_date);
                  call symput("year_taken3",put(year_taken3,8.0));
                  call symput("month_taken3",put(month_taken3,8.0));
            run;
            %if (&month_taken3=12 and &year_taken3=2008) or &year_taken3=2009
            %then %do;

                  proc sql;
                        select nobs into :n_s2
                        from
                        sashelp.vtable
                        where libname="%upcase(&LIB.)" and
                        memname="%upcase(&supp2.)";
                  quit;
            %end;

%end;

%mend;
```