Paper 190-28

PROC DATASETS: Managing Data Efficiently

Daphne Ewing, Synteract, Inc., Ambler, PA

ABSTRACT

PROC DATASETS is a data set management tool. There are things that the DATASETS procedure can do that can also be done using other techniques, although the other methods may be inefficient.

This paper will show the user the basics of PROC DATASETS. This will include some of the powerful tasks it can perform along with how these similar tasks can be completed using other methods within the BASE SAS product. The result will be a method for determining the best approach to gaining the information needed and performing these tasks using the appropriate tool.

INTRODUCTION

Being a procedure that is used as a tool to manage data, PROC DATASETS is not intended to "generate" output. This is noted by running the procedure with no options or commands.

```
proc datasets;
quit;
run;
```

If a library is not specified, the default is for SAS to look at the WORK library. Given this code, at the beginning of a program, the result will actually be a warning message ("*WARNING: No matching members in directory.*") in that PROC DATASETS doesn't find any elements in the WORK library. If PROC DATASETS is used without a LIB statement within the program, the library used would be the last (_LAST_) one referenced. But generally speaking, it is not a good idea to let SAS determine this, the programmer should provide the library in the DATATSETS procedure as shown below:

```
libname db 'c:\temp';
proc datasets lib=db;
quit;
run;
```

The results of the above statements are displayed in the LOG window, not in the OUTPUT window. A list containing all of the SAS data sets and SAS catalogs that reside in the specified library is displayed in the LOG window. This is similar to the PROC CONTENTS procedure.

DETERMING CONTENTS OF LIBRARY

Most people use the PROC CONTENTS procedure to display a list of data sets found in a specified library (using the NODS option). PROC DATASETS procedure generates this by default. The syntax of these two are:

```
proc datasets lib=db;
quit;
run;

proc contents data=db._all_ nods;
run;
```

The above two statements generate exactly the same results, although the DATASETS output goes to the LOG window and the CONTENTS output goes to the output window. Alternatively, Windows Explorer is an option to navigate to the directory noted in the LIBNAME statement, the window will display the same list (and more) of SAS files contained in that directory.

If more information is needed about each data set within a library (e.g. individual variable attributes), SAS needs to be instructed to do this with the DATASETS procedure, while in the above CONTENTS procedure SAS statements were added to remove the detail automatically provided. Below is an example of how to get the usual detail from PROC CONTENTS using the DATASETS procedure:

```
proc datasets lib=db;
  contents data=_all_;
quit;
run;
```

The CONTENTS procedure with NODS removed is shown below:

```
proc contents data=db._all_;
run;
```

The results of the above two procedures now is exactly the same. Both steps send the results to the OUTPUT window, and neither is really more difficult to program than the other.

COPYING DATA SETS/CATALOGS

When there are data sets contained in a library that need to be copied or moved to another location, there are several methods available. The simplest of which would be to use the Windows Explorer tool to copy files from one physical location to another. If, however, the program is relying on this happening, it will be important to have the program ensure that it occurs appropriately from the program. Within SAS, a similar method to the use of Windows Explorer would be to enter the host-system mode (X command) to execute a Windows or Dos command from within the program:

```
options noxwait;
x 'copy c:\temp\*.sas7*
       c:\newtemp';
```

The NOXWAIT system option is helpful because it closes the DOS window that was executed with the X command when the statements sent to it are completed. When this command executes, the system date/time stamp on the files is maintained as experienced when using this command within the operating system itself.

The next logical option would be the COPY procedure. This is relatively straight forward, but unlike the X command above, the date/time stamp on the files copied are changed to the date/time the procedure is executed:

```
libname newdb 'c:\newtemp';
proc copy in=db out=newdb;
run;
```

Lastly, the DATASETS procedure can be used to perform the copy in a similar way as noted below.

```
proc datasets nolist;
  copy in=db out=newdb;
quit;
run;
```

The log begins to explain the efficiency that may be gained in using DATASETS instead of other procedures. Looking at the REALTIME and CPU TIME information provided from each of these methods, the PROC DATASETS is more efficient.

Both of the above procedures have options that will limit the members of the IN library that are copied to the OUT library destination:

```
proc copy in=db out=newdb;
  select adverse formats;
run;

proc datasets nolist;
  copy in=db out=newdb;
  exclude adverse formats;
quit;
run;
```

These above two statements perform the same thing as either of the previous commands in this last COPY procedure was instructed to SELECT only two members (ADVERSE and FORMATS) from the IN library, while the EXCLUDE statement used in the DATASETS procedure to copy all but these two files thus copying all members from the IN library to the OUT library, like the last example.

There are situations when moving the files from one location to the other is more appropriate. Using the MOVE option in either procedure or using the host-system mode executing a MOVE statement achieves this.

```
x 'move c:\temp\adverse.sas7bdat
       c:\newtemp';

proc copy in=db out=newdb move;
  select summary formats;
run;

proc datasets nolist;
  copy in=db out=newdb move;
quit;
run;
```

Again, the above statements combined move all files from the IN library to the OUT library. From a robust program perspective, using the X command does not allow for a dynamic program in that the path and filename have to be presented. This could be overcome using SAS macro variables (which is beyond the scope of this paper).

RENAMING DATA SETS/CATALOGS

In the event that files need to be renamed, the X command shown below can do the job:

```
x 'rename
    c:\temp\adverse.sas7bdat
    c:\newtemp\newae.sas7bdat';
```

Or using the CHANGE command within PROC DATASETS:

```
proc datasets lib=db nolist;
  change adverse = newae;
quit;
run;
```

Please note that the CHANGE command requires the library statement exist on the DATASETS line, not in the actual CHANGE command. The SAS Data Step can also create a new data set with the new name:

```
data db.newae;
  set db.adverse;
run;
```

However, with the Data Step, the old data set still exists and additional code would have to be added to the program in order to remove the old file thus mimicking the PROC DATASETS function.

DELETING DATA SETS/CATALOGS

It is very easy to use windows explorer or the X command in a similar fashion to the examples shown above to delete files. The DATASETS procedure is more efficient and gives the processing control to the program instead of assuming that an external step had been performed, e.g. deleting the files outside of SAS.

```
proc datasets nolist;
  copy in=db out=work;
quit;
run;

proc datasets lib=work nolist;
  delete adverse summary;
quit;
run;
```

This only deletes the specified files on the DELETE command. In order to delete all members within a library, use the KILL option on the DATASETS statement. Although other SAS procedures have the option of using an _ALL_ option, the DELETE statement within PROC DATASETS does not allow this option. The KILL option is used as follows:

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

This is VERY helpful in situations where the “working” files created tend to use up a large amount of memory, once the logic of the program has been checked, KILLing the working files will result in a more efficient program. Another important reason to issue the above statement at the end of a program is when programs are run in batch, this will clean up the working library to be sure any “old” files are not left around to be erroneously used.

If there are files that must remain within a SAS library, the KILL option will not suffice. Instead, the DATASETS procedure provides the SAVE statement which will delete all members in a library except for the ones noted on the SAVE statement:

```
proc datasets lib=db nolist;
  save formats;
quit;
run;
```

This will delete all files in the DB library except the FORMATS catalog.

MODIFYING DATA SETS/VARIABLES

Making changes to the attributes of a member within a library is very typically done using the Data Step. The types of change to be discussed below become FAR more efficient when using the DATASETS procedure. The difference between these two methods is that when the Data Step is used, every record contained in the data set has to be processed. When using the DATASETS procedure, SAS is manipulating the header information of the data set and therefore doesn’t process each observation making it far more efficient. As the size of the data sets being modified increases, the efficiency increases dramatically when using the PROC DATASETS.

The following two examples show how to add a label to a data set. First, using the Data Step:

```
data db.adverse(label=
                'Adverse Events');
  set db.adverse;
run;
```

Using the DATASETS procedure:

```
proc datasets lib=db nolist;
  modify adverse (label=
                'Adverse Events');
quit;
run;
```

Both of the above statements add the “Adverse Event” label to the DB.ADVERSE data set. The REAL TIME is substantially higher when using the Data Step method.

Quite often, as a project moves forward, there may be times when the variable labels may have been left off of a variable. These are very helpful in describing what is being captured and therefore adding them is important. Most programmers choose the Data Step approach first:

```
data db.adverse;
  set db.adverse;

  label pat = 'Subject Number'
        inv = 'Site Number';
run;
```

However, the DATASETS method is by far more efficient and not more difficult to program:

```
proc datasets lib=db nolist;
  modify adverse;
    label pat = 'Subject Number'
          inv = 'Site Number';
quit;
run;
```

A couple of other common tasks in daily programming are the need to rename variables and apply formats/informats to variables. Using the Data Step, the following code might be used:

```
data db.adverse;
  set db.adverse(rename=(inv=site
                         pat=subj));
    format todate visdate
           fromdate mmddyy10.;
    informat todate visdate
             fromdate mmddyy10.;
run;
```

With the DATASETS procedure, the code to perform the same task is:

```
proc datasets lib=db nolist;
  modify adverse;
    rename inv=site pat=subj;
    format todate visdate
           fromdate mmddyy10.;
    informat todate visdate
             fromdate mmddyy10.;
quit;
run;
```

Comparing the CPU TIME for the above two tasks that are doing exactly the same thing, the DATASETS procedure completes more quickly. The REAL TIME however is even more substantial

CONCLUSION

As seen in the different examples provided, there are many methods of accomplishing the same result. This is the case with so many aspects of SAS. However, the DATASETS procedure can substantially improve efficiency. Since the DATASETS procedure does not have to process each record in the data set(s) it is working with, the efficiency is gained. Where the greatest efficiency is gained is with larger files.

ACKNOWLEDGEMENTS

Thanks to Ken Friedman who wrote the Coders' Corner Paper titled "Using PROC DATASETS for Efficient SAS Processing" who got me started on expanding on the basics.

AUTHOR INFORMATION

Please feel free to contact the author with questions/comments about the paper:

Daphne Ewing
Synteract, Inc.
714 N. Bethlehem Pike, Suite 300
Ambler, PA 19002
(215) 283-9470 x604
(215) 283-9366 (fax)
dewing@synteract.com
www.synteract.com