Paper 032-31

Using the DATASETS Procedure

David Fickbohm Homegain inc. Emeryville CA

ABSTRACT

The purpose of this paper is to explain why, and how to use the datasets procedure. Datasets is a very powerful procedure. It is not used as much as it could / should be used. This paper will cover the basic concepts and syntax of the datasets procedure. The instructions that manipulate data files; age, append, change, copy, delete, exchange, and, save will be presented. This paper will cover the informational instructions contents. Finally the audit instructions will be discussed.

INTRODUCTION

Why is the datasets procedure not used as much as other procedures? One reason is that three of the older SAS® procedures - Copy, Contents, and Append - were around before proc datasets. People learned those three and stuck with them. Why use the datasets procedure instead of several procedures? Datasets is more efficient, more versatile, and has features far beyond copy, contents, and append. Using the datasets procedure lets you remember, become proficient, and rely on one procedure rather than waste brain storage and time working with three. This paper will try to explain why, when, and how to use the datasets procedure.

BASIC IDEA

A SAS data set contains descriptor information and data values. SAS data files can be native files created by SAS or interface SAS files, those created by other software (oracle, DB2, etc.) but referenced by a SAS engine. SAS views hold only descriptors. All of these can be modified by Proc datasets.

A SAS library is a collection of one or more SAS files. These files can be called members. Members or files, they are recognized by the SAS System and are referenced and stored as a unit. Each library is associated with a SAS library engine; this is the software interface between SAS and a library.

Proc datasets reads and changes information about SAS datasets without actually changing the data. This makes Proc datasets much more efficient than the data step which must read each record in, whether or not it will be changed, in order to make changes to the descriptor information.

Proc datasets allows run group processing, meaning you can start a procedure and have the commands execute without ending it. Commands can be stacked up and executed together. An error in one command stops processing for all remaining statements. You can use the FORCE option to force the proc to continue, but my advice is *not* to use FORCE. I would rather find my error, correct it, and rerun the entire package. Then I know what to expect in subsequent steps and procedures. The FORCE command does have other, very powerful uses as we will see.

GETTING TO KNOW PROC DATASETS

SYNTAX

The syntax is basically the same for all functions, although defaults do vary, and not all options apply to all commands.

```
PROC DATASETS <options>;
<COMMANDS> <options>;
QUIT; RUN;
```

Options that apply to most commands follow:
Detail / Nodetails – controls the output of observations, variables, and labels
List / Nolist – controls listing of directory members. The default is LIST.
Library – Specifies the SAS Library. If a library is not specified, SAS looks at the last (_last_) library referenced. If no library has been specified, the default is the work library.
FORCE – used to force the proc to continue even though an error has been found.
MEMTYPE – specifies the member type. **NOWARN -** Suppresses warning messages.

OPERATION

Being a procedure used to manage data, Proc datasets is not intended to create output. If run with no options or commands,

```
Proc Datasets; Quit; Run;
```

Only a warning is produced

It is a very good idea to always include a lib statement specifying the library with which you want to work.

```
LIBNAME SS06 'C:\SALESJUN05';
PROC DATASETS LIB = SS06; QUIT; RUN;
```

The output of the statements above is displayed in the LOG window, not in the OUTPUT window. The code above will produce a list of all SAS data sets and SAS catalogs that reside in the specified library.

DETERMINING THE CONTENTS OF A LIBRARY

Using proc datasets here is more efficient than proc contents. It requires less keying. You can produce the same report using proc contents with the all nods option:

```
PROC CONTENTS DATA = XX06._ALL_ NODS;
```

Proc Datasets produces the same report by default.

DETERMINING THE CONTENTS OF A LIBRARY AND THE ATTRIBUTES OF THE VARIABLES IN THE DATASETS.

Rather than use PROC CONTENTS and PROC DATASETS, you can use the ALL option in Proc datasets and get all of the information that both procedures would produce while running one. The information about the library members is displayed in the log. The information about the contents of library members is displayed in output.

```
PROC DATASETS LIB = SS06;
CONTENTS DATA =_all_; QUIT; RUN;
```

MANIPULATING FILES

COPYING AND MOVING SAS DATASETS BETWEEN CATALOGS

Proc Datasets makes copying and moving datasets from one catalog to another very easy. The SELECT and EXCLUDE options can be used to specify or limit which datasets within a library are copied from one library to another. Changing the keyword COPY to MOVE moves a member instead of copying it. The member is in both libraries after copying. The member is only in the out= library after moving.

```
LIBNAME SS06 'C:\TEMP1';
LIBNAME KEPR 'T:\DAVEKEEP';
PROC DATASETS;
COPY IN = SS06 OUT = KEPR; QUIT; RUN;
```

This moves all SAS members in SS06 to KEPR. Adding a SELECT or EXCLUDE allows you to be more specific. The code below copies only members AAA and BBB to KEPR.

```
LIBNAME SS06 'C:\TEMP1';
LIBNAME KEPR 'T:\DAVEKEEP';
PROC DATASETS;
COPY IN = SS06 OUT = KEPR;
SELECT AAA BBB;
QUIT; RUN;
```

The code below copies all members EXCEPT CCC.

```
LIBNAME SS06 'C:\TEMP1';
LIBNAME KEPR 'T:\DAVEKEEP';
PROC DATASETS;
COPY IN = SS06 OUT = KEPR;
EXCLUDE CCC;
QUIT; RUN;
```

SAVE OR DELETE A LIBRARY MEMBER, KILL ALL LIBRARY MEMBERS

The SAVE command specifies which members will be kept in a SAS library. All others, not specified, will be deleted. The DELETE command specifies which members will be deleted from a SAS library. The KILL command instructs SAS to delete all members in a library.

The code below will cause all members except AAA and BBB to be deleted.

```
LIBNAME SS06 'C:\TEMP1';
PROC DATASETS LIBRARY = SS06;
SAVE AAA BBB; QUIT; RUN;
```

The code below will only delete library members AAA and BBB.

```
LIBNAME SS06 'C:\TEMP1';
PROC DATASETS LIBRARY = SS06;
DELETE AAA BBB; QUIT; RUN;
```

The code below deletes all members in a library. BE VERY CAREFUL with KILL.

```
LIBNAME SS06 'C:\TEMP1';
PROC DATASETS LIBRARY = SS06 KILL;
QUIT; RUN;
```

THE USE OF APPEND VS SET

The APPEND command adds observations from one dataset to another. Unlike the SET command which reads in all observations from the datasets being concatenated, the APPEND command only reads in observations from the dataset being appended. This saves you processing time.

The code below appends June05_claims to MAY05_CLAIMS.

If the two datasets being concatenated contain different variables, data types, and/or lengths, the FORCE option can be used. This is a case where the FORCE option is very appropriate and very powerful.

```
PROC DATASETS LIBRARY = WORK FORCE;
APPEND OUT = MAY05_CLAIMS   DATA = JUNE05_CLAIMS;
QUIT; RUN;
```

THE USE OF CHANGE

The CHANGE command renames one or more members within a SAS library.
You specify the old member name on the left and the new member name on the right.

```
PROC DATASETS LIBRARY = YY;
CHANGE MAY05_CLAIMS = MAY_JUNE05_CLAIMS;
QUIT; RUN;
```

THE USE OF AGE

The AGE command renames a group of related SAS members in a library.
The code below will add day1 to the library and delete day7.

```
proc datasets library=daily nolist;
age today day1-day7;
quit; run;
```

```
LOG LISTING
NOTE: Deleting DAILY.DAY7 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY6 to DAILY.DAY7 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY5 to DAILY.DAY6 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY4 to DAILY.DAY5 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY3 to DAILY.DAY4 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY2 to DAILY.DAY3 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY1 to DAILY.DAY2 (memtype=DATA).
NOTE: Ageing the name DAILY.TODAY to DAILY.DAY1 (memtype=DATA).
```

I used the member name day1-day7 for clarity. The member names do not have to describe the data. .

THE MODIFY COMMAND

The MODIFY command gives you the ability to change a specific library member or attributes of the library member's variables.

The code below works with the member jun05_payments in the work library. It adds a label to the member, it renames a variable, assigns an appropriate label to the newly renamed variable, and finally a format is assigned to an existing variable. The RENAME command can ONLY be used after a modify command.

```
PROC DATASETS LIBRARY = WORK;
MODIFY JUN05_PAYMENTS (LABEL = 'NEW_MEMBER_LABEL');
  RENAME OLD_VARIABLE_NAME = NEW_VARIABLE_NAME;
  LABEL  NEW_VARIABLE_NAME = LABEL_FOR_RENAMED_VARIABLE;
  FORMAT EXISTING_VARIABLE_NAME COMMA11.2;
QUIT; RUN;
```

As no observations are read in or written out during processing, for an existing library member, the MODIFY command is the best way to add a label, rename a variable, or change a format or informat.

AUDIT STATEMENT

The AUDIT statement takes on one of two forms, depending on whether you are initiating the audit trail or suspending, resuming, or terminating event logging in an audit file.

INITIATE

Creates an audit file that has the same name as the SAS data file and a data set type of AUDIT. The audit file logs additions, deletions, and updates to the SAS data file.

```
PROC DATASETS LIB = MYLIB;
   AUDIT MAY_SALES;
   INITIATE;
QUIT; RUN;
```

SUSPEND

This suspends event logging to the audit file, but does NOT delete the audit file.

```
PROC DATASETS LIB = MYLIB;
  AUDIT MAY_SALES;
  SUSPEND;
QUIT; RUN;
```

RESUME

This resumes event logging to the audit file, if it was suspended.

```
PROC DATASETS LIB = MYLIB;
  AUDIT MAY_SALES;
  RESUME;
QUIT; RUN;
```

TERMINATE

This terminates event logging and deletes the audit file.

```
PROC DATASETS LIB = MYLIB;
  AUDIT MAY_SALES;
  TERMINATE;
QUIT; RUN;
```

EXCHANGE

This instruction exchanges the names of two SAS files in a SAS library.

```
PROC DATASETS LIB = MYLIB;
  EXCHANGE SALES = JUN_SALES;
QUIT; RUN;
```

CONCLUSION

We have seen that PROC DATASETS is a many featured, multi faceted, very useful tool. Since PROC DATASETS does not read in nor write out observations, PROC DATASETS is very efficient. It is simple to use and easy to learn.

Caution should be exercised when working with PROC DATASETS. Realize that PROC DATASETS is an interactive procedure, so all commands execute immediately and in the order they appear.

For more information on PROC DATASETS, see the SAS Procedures Guide or look up PROC DATASETS in the on-line SAS documentation. If all else fails, contact me using the contact information below.

CONTACT INFORMATION

Please feel free to contact me with questions and comments about this paper:

David Fickbohm
HomeGain+
1250 45th St. Suite 200
Emeryville, CA 94608
510 594 4151 day phone
510 655 0848 fax
davidf@homegain.com