

Data set options - what comes first

Jim Snider, AmSouth Bank, Birmingham AL

INTRODUCTION

I make frequent use of data set options, yet find it difficult to remember which ones take precedence when multiple options reference the same variable. Of course, it's critical to know what comes first to avoid errors resulting from referencing variable names not recognized by SAS. The information in this presentation is handy reference material that I use to keep things straight when using multiple inter-related options.

USING DATA SET OPTIONS

- Data set options override system options; examples would be FIRSTOBS=, OBS=, COMPRESS=
- You do not have any flexibility in the use of parenthesis on data set options.

It may help to visualize the data step process as three components:

1. Input processing
 2. Program processing
 3. Output processing
1. Input processing -
 - Data set options used on the set statement will be applied in the Input process. The order of processing for the most common options is:
 - 1) Keep= / Drop=
 - 2) Rename=
 - 3) Where=
2. Program processing -
 - Data set options are not performed in program processing. However, there are corresponding statements for a data step or procedure that perform similar functions. The primary differences are: 1) the actions are performed when the program processes, rather than in I/O, 2) the actions will affect any and all files named on the DATA statement.
 - The Keep / Drop and Rename statements are executed, in that order, at the end of the program processing. Other statements, such as Where and If, that require evaluation of each observation, are executed prior to Drop, Keep and Rename.
3. Output processing -
 - Data set options used on the data statement will be applied in the Output process. The order of processing for the most common options:
 - 1) Keep= / Drop=
 - 2) Rename=
 - 3) Where=

*** Create test file;

```
data testit;
  input oldname $10. @15 city $15. @31 state $2. ;
cards;
John Carver    Greenville    NC
Mary Jones    Chicago      IL
Don Smith     Phoenix      AZ
run;
```

* Example 1 ;

- ◆ Variable is renamed before SAS applies filter, actions occur in the same order as listed.

```
data testit1 ;
  set testit(keep=oldname state
              rename=(oldname=newname)
              where=(newname like 'John%')) ;
run;
```

/* Result -

NOTE: There were 1 observations read from the data set WORKLIB.TESTIT.

WHERE newname like 'John%';

NOTE: The data set WORKLIB.TESTIT1 has 1 observations and 2 variables. */

** Example 2 ;*

- ◆ Filter is applied in I/O, variable is renamed at end of program processing. In program process, actions occur in this order: where, keep/drop, rename.

```
data testit2 ;
  set testit(where=(oldname like 'John%')) ;
  keep oldname state ;
  rename oldname = newname ;
run;
```

```
/* Result -
  NOTE: There were 1 observations read from the data set WORKLIB.TESTIT.
  WHERE oldname like 'John%';
  NOTE: The data set WORKLIB.TESTIT2 has 1 observations and 2 variables.
*/
```

** Example 3 ;*

- ◆ When the filter is applied to the output data set, it is performed in the output process, following the program processes.

```
data testit3(where=(newname like 'John%')) ;
  set testit ;
  rename oldname = newname ;
  keep oldname state ;
run;
```

```
/* Result -
  NOTE: There were 3 observations read from the data set WORKLIB.TESTIT.
  NOTE: The data set WORKLIB.TESTIT3 has 1 observations and 2 variables.
*/
```

** Example 4 ;*

- ◆ When the filter is put in the program, it is "appended" to the input process for efficiency purposes. Therefore, it occurs before other program process operations.

```
data testit4 ;
  set testit ;
  keep oldname state ;
  rename oldname = newname ;
  where oldname like 'John%' ;
run;
```

```
/* Result -
  NOTE: There were 1 observations read from the data set WORKLIB.TESTIT.
  WHERE oldname like 'John%';
  NOTE: The data set WORKLIB.TESTIT4 has 1 observations and 2 variables.
*/
```

** Example 5 ;*

- ◆ This example illustrates two things. The If statement filters data in the program process, just like the Where statement, only it doesn't affect the number of records that are read. This is why the Where statement is touted as being more efficient than the If statement. Secondly, as the program demonstrates, you can use the same data set option on input and output as well as using the corresponding statement in the program.

```
data testit5a testit5b(keep=newname) ;
  set testit(keep=oldname city state) ;
  keep oldname city;
  rename oldname = newname ;
  if substr(oldname,1,4) = 'John' ;
run;
```

```
/* Result -
  NOTE: There were 3 observations read from the data set WORKLIB.TESTIT.
  NOTE: The data set WORKLIB.TESTIT5A has 1 observations and 2 variables.
  NOTE: The data set WORKLIB.TESTIT5B has 1 observations and 1 variables.
*/
```

** Example 6 ;*

- ◆ When the keep, rename and where options are applied to the output data set, they occur in that order: keep, rename, where.

```
data testit6(keep=oldname state
              rename=(oldname=newname)
              where=(newname like 'John%')) ;
  set testit ;
run;
```

```
/* Result -
   NOTE: There were 3 observations read from the data set WORKLIB.TESTIT.
   NOTE: The data set WORKLIB.TESTIT6 has 1 observations and 2 variables.
*/
```

CONCLUSION

Data set options are a great feature of SAS and contribute to efficient coding and processing. I hope this information will assist you in avoiding errors when you make use of data set options in your SAS programs.

CONTACT INFORMATION

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

Jim Snider
AmSouth Bank

james.snider@amsouth.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are trademarks of their respective companies.