Paper 248-31

Programming with the KEEP, RENAME, and DROP Data Set Options Stephen Philp, Pelican Programming, Los Angeles, CA

ABSTRACT

One of the more frustrating things for a new user learning SAS can be the multitude of ways of accomplishing the same thing, each with its own subtleties. The topic of dropping, keeping and renaming variables in data sets is no exception. Using a DATA step, there are two ways of manipulating variables for keeping, dropping, or renaming: DATA step statements and data set options. First we will review the basic workings of the DATA step, which will help you understand how each approach differs, which will then help you to create more efficient code and avoid some potentially costly mistakes.

INTRODUCTION

As you likely know, a SAS data set is a table. It has rows (observations) and columns (variables). Besides adding and deleting rows, the modifications you can make to the structure of a table are adding columns, deleting columns and renaming columns. To modify columns in a SAS data set you use DROP/KEEP and RENAME. There are two ways to apply a column modification to a data set: using DATA step statements and data set options. In order to understand the differences in the behavior between the statements and the options we will first review the basic workings of the DATA step, then apply that knowledge to create clearer, more efficient code using the data set options.

DATA STEP PROCESSING

SAS stores its data in tables called data sets. These tables have observations (rows) and variables (columns). Data sets can be thought of as having two logical parts: a "data" part and a "descriptor" part. The "descriptor" part holds a description of the data set including variable attributes. The "data" part holds the actual data. When working with data sets in DATA steps, it is helpful to peek under the hood and understand how DATA step processing works.

First of all, a DATA step has two phases: the compile phase and the execution phase. When you submit a DATA step for execution SAS first checks the syntax, compiles the statements and then sets up the Program Data Vector (PDV). The PDV is an area of memory where SAS stores variable values and attributes. It is the PDV that stores an observation as it is being processed in the DATA step.

Name	_N_	_ERROR_	Var1	Var2	Var3	Var4	Var5
Туре	Z	Ν	Ν	\$	\$	\$	\$
Length	8	8	8	20	5	12	1
Retain	yes	yes	no	no	no	no	No
Format	best12.	best12.	best12.	\$w.	\$w.	\$w.	\$w.
Value	1	0	64	Trader	ggyy	moment	Χ

Figure 1. Example Program Data Vector

When a DATA step is preparing to read from a data set, it uses the information in the data set's "descriptor" section to find out what variables to include in the PDV.

After the compile phase the DATA step executes. During execution the DATA step loops by first reading values into the PDV, executing statements that may change the values in the PDV and then eventually writing the values in the PDV out as an observation into the new data set.

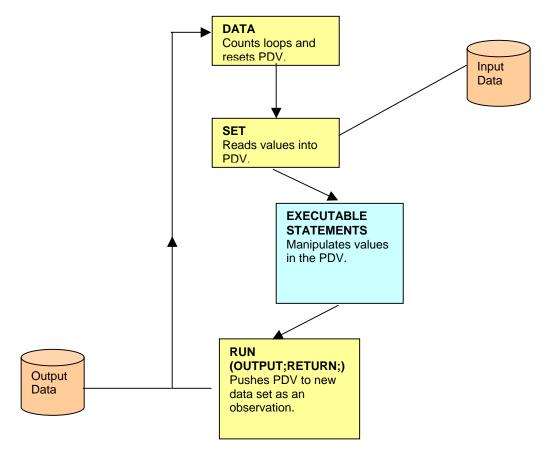


Figure 2. Simplified DATA Step Loop.

Corresponding to the two phases of the DATA step, there are two types of DATA step statements: compile time statements and execution time statements. As their names suggest compile time statements do their work at compile time and execution statements do their work during the execution of the DATA step. Some examples of execution time statements are:

- Assignment statements (variable = value;).
- If-then/else statements.
- Do loops.
- Generally any statement which relies upon the values of variables stored in the PDV.

Some examples of compile time statements are:

- Retain statement.
- Array declarations.
- Drop statement.
- Keep statement.
- Rename statement.

SYNTAX

Remember, a SAS statement is syntactically different from an option. SAS statements are defined as beginning with a keyword and ending with a semicolon. Data set options appear in parentheses next to a data set. Beyond dropping, keeping and renaming variables there are a number of data set options available to the programmer. Multiple options are separated with spaces.

(option-1=value1 <...option-n=value-n)

Some Examples:

- Data test(keep= answer1 answer2 key);
- Proc summary data = students(keep=grade passFail) nway;
- Set travelers(rename= (luggage = bags));

Notice the RENAME= option has an additional set of parentheses where you specify old variable name = new variable name.

WHAT THEY DO

Now that we have examined the syntax, let's review what these do.

- DROP: specifies variables to be dropped (filtered out) from a table.
- KEEP: specifies which variables will be kept in a table.
- RENAME: renames a variable. OldVariable = NewVariable.

All three of these actions affect the structure of a table when it is being input or output. The KEEP/DROP/RENAME compile time statements always affect the output table. This can cause some confusion in that the order of these statements generally does not affect the DATA step's behavior. Consider the following two DATA steps:

```
data one;
  drop words;
  set grammars;
  if substr(var1, 1, 4) = 'THIS' then do;
  ...
run;

data two;
  set grammars;
  if substr(var1, 1, 4) = 'THIS' then do;
  ...
  drop words;
run;
```

Both DATA steps have exactly the same behavior even though the DROP statement appears in different places.

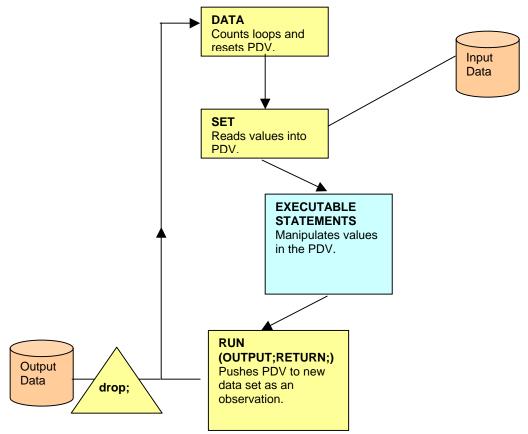


Figure 3. Drop Statement Affects Output Data.

Remember, the KEEP, DROP and RENAME statements always affects the data being output.

Unlike their equivalent data step statements, data set options apply only to the data set with which they appear. That way they can affect the data set being read or written. This is one of the more compelling reasons for using the equivalent data set options.

Whether you use an option or the equivalent statement, there is one additional timing rule that always stays the same:

KEEP/DROP always happens before the RENAME.

USING THE OPTIONS

The KEEP=, DROP= and RENAME= options are more flexible than their respective statements in that they can be applied to the data set being written, the data set being read or both. This gives the programmer much more control over how the DATA step processes data and can lead to more efficient code. Given the DATA step:

```
data halfYear;
  set QTRone
     QTRtwo;
run;
```

The data set being written is *halfYear* and there are two data sets being read: *QTRone QTRtwo*. This is illustrated in figure 4.

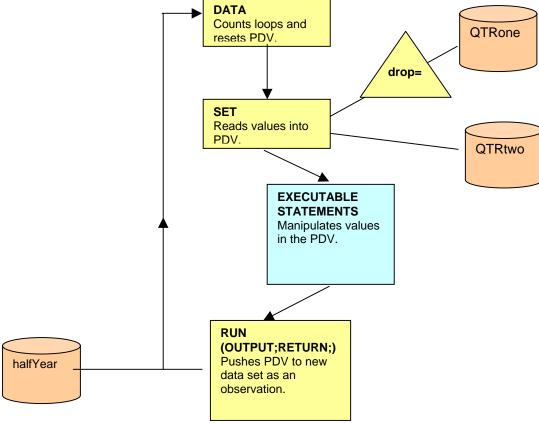


Figure 4. Reading two data sets.

When a data set option is applied to a data set being read from, it applies its action before the data is read into the DATA step. Such as:

```
data halfYear;
  set QTRone(drop = startDate )
      QTRtwo;
run;
```

In this example the variable *startDate* is being dropped (filtered out) from *QTRone* before being read into the data step. In fact, it keeps the DATA step from knowing the variable even exists in the data set. This keeps it from being created in the PDV.

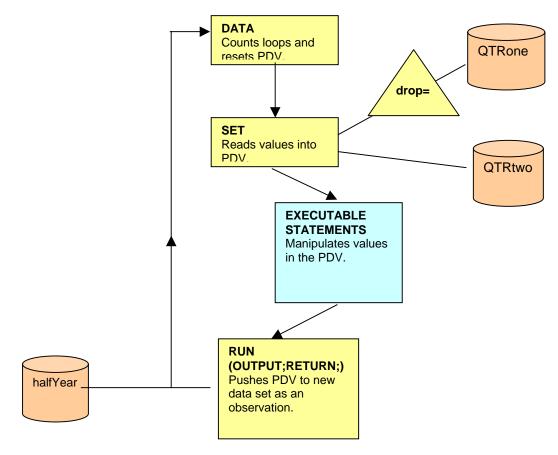


Figure 5. Drop= option affecting one input data set.

Using data set options you can also specify that a variable should be dropped when the observation is being written to the new data set. In this case you would put the option next to the data set on the data statement.

```
data halfYear(drop = startDate );
   set QTRone
        QTRtwo;
run;
```

This makes the variable available to the DATA step but prohibits it from being written to the data set halfYear.

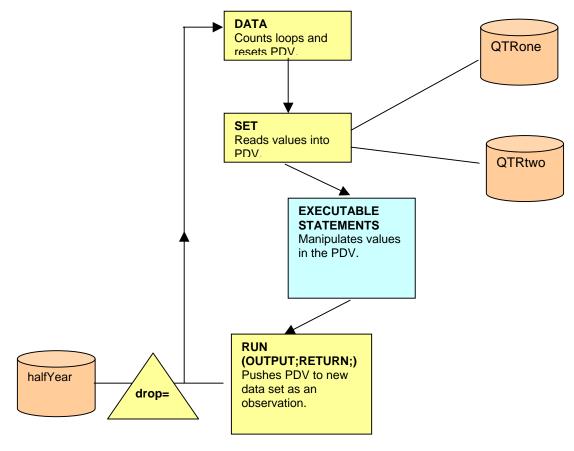


Figure 6. Drop= option affecting output data set.

The same can be applied to KEEP= and RENAME= as well. Consider the following DATA step:

```
data halfYear( rename = ( startDate = beginDate ) );
  set QTRone
     QTRtwo;
run;
```

In this case, the variable *startDate* is being renamed when the observation is being written to *halfYear*. There will no longer be a variable named *startDate* in *halfYear*. But *startDate* is still available during the processing of the DATA step.

Again, notice the different syntax of the RENAME= option. The variables being renamed are placed within "(" and ")". This allows you to rename multiple variables. Such as:

```
( rename= ( var1=newVar1 var2=newVar2 var3=newVar3 ) )
```

You can combine options on the same data set. Consider:

A big drawback to using the statements instead of the options is that the statements apply equally to all data sets being created. Consider a data step that is creating two data sets. One data set has only unique observations and the other data set has all the duplicates. For the unique data set we only want to keep the key variables, but for the duplicates data set we want to keep all the variables. For this case we would use a keep= option on the unique data set. Since the option only applies to the data set it is next to, the other data set is not affected.

```
data unique(keep=key)
      duplicates;
set myData;
by key;
if first.key and last.key then output unique;
else output duplicates;
run;
```

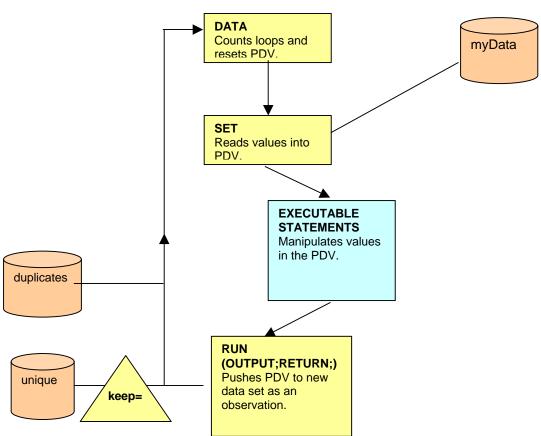


Figure 7. Keep= option affecting one output data set.

USING DROP/KEEP/RENAME IN A MERGE

These data set options can be especially useful in a data step merge. Consider the following two data sets.

FirstNames

ID	NAME
1	Robert
2	Tom
3	Reese
4	Nicole

LastNames

D	NAME			
1	De Niro			
2	Hanks			
3	Witherspoon			
4	Kidman			

Here we have two data sets *firstNames* and *lastNames* that each has a variable called *name*. What happens if we try to merge them to get both names onto one data set? The DATA step will overlay any variables that exist on multiple data sets. If we tried to merge them by ID, which name would be in the resulting data set?

```
data names;
  merge FirstNames LastNames;
  by id;
run;
```

We would end up with one *name* variable in the data set which would have the last names in it. It is important to note that by default DATA steps will overwrite values on overlapping variables in a merge without even a warning. You can change this by specifying msgLevel=i in your SAS system options to force a warning in your log whenever a merge overwrites variable values.

Names

ID		NAME
	1	De Niro
	2	Hanks
	3	Witherspoon
	4	Kidman

Whenever variables overlap in a merge like this, the value from the last data set to contribute an observation to the merge is the one you get; but in this case, not what we want. We want both the first name and the last name. So we need to rename the variables before they get overlaid. To do this, we need to use the RENAME= option on the data sets in the merge statement.

Now we get a resulting data set with both names.

Names

ID		FIRSTNAME	LASTNAME
	1	Robert	De Niro
	2	Tom	Hanks
	3	Reese	Witherspoon
	4	Nicole	Kidman

The same applies to performing a join in the SQL procedure. However, in SQL you will get a warning in your log if you have overlapping variables. Also, the logic of which value gets kept is opposite of the merge: the value comes from the first named data set, not the last.

Names

ID	NAME
1	Robert
2	Tom
3	Reese
4	Nicole

USING OPTIONS IN PROCEDURES

Another reason to use the data set options is that they can be used with procedures as well as DATA steps. This adds a lot of flexibility and can often result in reduced steps and much more efficient code. Let's say you have a big data set with 100 variables and many thousands of observations. You want to create a temporary data set out of it with only 5 variables. You also want your temporary data set sorted. One (inefficient) way to do this would be:

```
data work.myTempData;
  set saved.bigDataSet;
  keep variable1
      variable2
      variable3
      variable4
      variable5;
;
run;

proc sort data = myTempData;
  by variable1 variable2;
run;
```

The first improvement you could make would be to keep the variables you want before the data is read into the data step using your KEEP= option in the set statement. Remember, the KEEP statement allows all the variables to be processed in the DATA step and then restricts which variables are written to the new data set. The KEEP= option on the set statement allows only the variables you specify into the DATA step and is more efficient.

But we still have an extra step that we don't need. We could apply the KEEP= option on the data set in PROC SORT and then use the procedure's OUT= option to create our temporary data set.

Now we are only reading the data set once and we are restricting the procedure to process only the variables we want.

MULTIPLE OPTIONS IN ONE PROCEDURE

Many times we use the SUMMARY procedure to do a count of observations. We can accomplish this by combining the DROP= and RENAME= data set options.

```
proc summary data= students NWAY;
  class gender;
  output out= counts( drop=_type_ rename=( _freq_=count) );
run;
```

In this case, we are using the RENAME= option to take advantage of the automatic _freq_ variable that PROC SUMMARY creates. We are also dropping the _type_ variable since it is not needed.

WARNINGS

If you use the KEEP, DROP and RENAME statements you should know that SAS will issue a WARNING and not an ERROR in the log if the variable does not exist. This can lead to some potentially costly errors if that variable is used somewhere else in the code and the warning is missed. The data set options on the other hand, will issue an ERROR if a variable is specified that does not exist. In fact, there are two SAS system options that control how SAS responds in this situation.

- Input Data Sets-- DKRICOND = Error | Warning | NoWarning
- Output Data Sets-- DKROCOND = Error | Warning | NoWarning

CONCLUSION

Understanding how the DROP=, KEEP= and RENAME= data set options work helps the programmer to understand better how the DATA step is working. They can also lead to clearer and more efficient code. Data set options offer the programmer much more flexibility in specifying how SAS processes data than their statement counterparts.

POINTS TO REMEMBER

- The statements always affect how the data is written to the output data step. The options can be applied to either the input or the output data set.
- The statements affect all output data sets equally if more that one is being created. The data set options can be applied individually to each data set.
- Specify a data set option in parentheses after a data set name.
- When renaming a variable always remember old = new.
- Keep/Drop always happens before the rename.
- Restricting the number of variables going into a DATA step can be more efficient.

REFERENCES

Heaton, E., (2003) "SAS System Options Are Your Friends", in SUGI28 Conference Proceedings, Cary, NC: SAS Institute.

ACKNOWLEDGMENTS

Thank you Orla Hayden and Michael Shreve for your generous time proofreading.

CONTACT INFORMATION

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

Stephen Philp

Pelican Programming

Los Angeles, CA

Stephen@pelicanprogramming.com

www.pelicanprogramming.com

http://datasteps.blogspot.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.