

Reordering Variables in a SAS® Data Set

Imelda C. Go, Lexington County School District One, Lexington, SC

ABSTRACT

A programmer may have a number of reasons to want to reorder variables in a data set. One reason is to take advantage of PROC PRINT output that prints variables according to their order on the data set when the VAR statement is not used. Another reason is to logically order the variables prior to exporting the data into a spreadsheet. This would eliminate the need to reorder the columns in the spreadsheet since reordering is done prior to exporting. Whatever the reasons are, there are a number of ways to reorder the variables in a data set. The DATA step may be used to reorder the variables by using one of the ATTRIB, ARRAY, FORMAT, INFORMAT, LENGTH, or RETAIN statements. PROC SQL can also be used to reorder the variables.

INTRODUCTION

In general, the order of variables in a data set is based on the order in which the variables were created. As programmers manipulate their data sets, they hardly ever keep track of the order of variables on the data set. After all, the order of the variables can be changed later through a number of different ways.

Why is there a need to reorder the variables in a data set? The answer may not be obvious to those who have not had the need to reorder variables in a data set before. Here are a few reasons:

When procedures are used without a VAR statement, the procedure is applied to all usable variables according to the order of variables on the data set. Not all variables may be usable because certain procedures only work with certain types of data. For example, PROC MEANS cannot be applied to character variables. The effect of not using a VAR statement would be similar to using special SAS name lists with the VAR statement:

```
*specify all variables
var _all_;
*specify all numeric variables
var _numeric_;
*specify all character variables
var _character_;
```

Name range lists are processed depending on the position of the variables in a data set. Such lists may be used with several types of statements as a convenience. In the examples below, variables A and Z are the first and last variables in the name range list. A and Z are considered for processing. Any other variable created *after* variable A and *before* variable Z are also considered for processing. If a specific variable type is specified, then only variables of the same type are considered.

```
*specify all variables
keep A--Z;
*specify all numeric variables
keep A-numeric-Z;
*specify all character variables
keep A-character-Z;
```

Data sets may need to be exported (e.g., into a spreadsheet). Instead of rearranging the columns of data in the spreadsheet, the order of the variables is changed prior to exporting the data into the spreadsheet. Having the columns of data in their preferred or logical order also make it easier to view data sets within SAS.

The following examples were contrived for the purpose of illustrating the different methods for reordering variables. Consider the following sample data set.

```
data sample;
input d c b a;
cards;
4 3 2 1
;
```

The order of the variables based on the INPUT statement is d, c, b, and a. An excerpt from the PROC CONTENTS output confirms the order indicated by the value in the # column.

```
-----Alphabetic List of Variables and Attributes-----
#   Variable   Type   Len   Pos
-----
4   a          Num    8    24
3   b          Num    8    16
2   c          Num    8     8
1   d          Num    8     0
```

The same order appears when PROC PRINT is used without the VAR statement.

```
proc print;

Obs    d    c    b    a
1      4    3    2    1
```

Using a name range list of c--b in the VAR statement produces the following results.

```
proc print;
var c--b;

Obs    c    b
1      3    2
```

The name range list of b--c in the VAR statement is invalid. The following error will appear.

```
87  proc print;
88  var b--c;
ERROR: Starting variable after ending
variable in data set.
```

One obvious way of controlling the order of the variables is to rewrite the INPUT statement.

```
data sample;
input a 7 b 5 c 3 d 1;
cards;
4 3 2 1
;

proc print;

Obs    a    b    c    d
1      1    2    3    4
```

This is not worth the trouble and is not possible for data sets that result when no INPUT statement is involved.

SAS NOTES FOR REORDERING VARIABLES

The following are SAS Notes that were obtained from www.sas.com.

V6-SYS.DATA-8946

How to reorder variables in a SAS data set

Any of the following statements may be used to change the order of variables in the program data vector:

ATTRIB, ARRAY, FORMAT, INFORMAT, LENGTH, and RETAIN.

Note that only the variables whose positions are relevant need to be listed in the above statements. Variables not listed in these statements will retain their same positional order following the listed variables.

For any of these statements to have the desired effect, they must be placed prior to the SET statement in the DATA step.

Products: BASE
Component: SYS.DATA
Priority: N/A
Status: Usage Issue
Date: Mon, 27 Jun 1994

System	Release Reported	Release Fixed
Solaris	6.09 TS027	
IBM OS/2	6.08 TS404	
VSE/ESA (VSE)	6.08 TS404	
Windows 3.11	6.08 TS404	
VM/ESA (CMS)	6.08 TS404	
OS/390 (MVS)	6.08 TS404	
OpenVMS VAX	6.08 TS404	
AIX/6000	6.09 TS027	
ConvexOS	6.09 TS027	
HP-UX Operating Systems	6.09 TS027	
OpenVMS Alpha	6.09 TS027	
Windows NT	6.09 TS027	
DEC Ultrix	6.09 TS027	

No Fixes Available

DATA STEP EXAMPLES

The following shows SAS code for changing the order in six different ways using the DATA step.

ATTRIB Statement

The ATTRIB statement associates a format, informat, label, and/or length with one or more variables.

```
**example for associating a format;  
data attrib_method;  
  attrib a b c d format=1.;  
  set sample;  
  
**example for associating an informat;  
data attrib_method;  
  attrib a b c d informat=1.;  
  set sample;  
  
**example for associating a label;  
data attrib_method;  
  attrib a b c d label='example';  
  set sample;  
  
**example for associating a length;  
data attrib_method;  
  attrib a b c d length=3.;  
  set sample;
```

ARRAY Statement

The ARRAY statement defines elements of an array.

```
data array_method;  
  array fixorder a b c d;  
  set sample;
```

FORMAT Statement

The FORMAT statement associates formats with variables.

```
data format_method;  
  format a b c d 1.;  
  set sample;
```

INFORMAT Statement

The INFORMAT statement associates informats with variables.

```
data informat_method;  
  informat a b c d 1.;  
  set sample;
```

LENGTH Statement

The LENGTH statement specifies the number of bytes for storing variables.

```
data length_method;  
  length a b c d 3.;  
  set sample;  
  
/* Be careful to specify the correct length.  
851  length a b c d 1.;  
  --  
  352  
ERROR 352-185: The length of numeric variables is 3-8.*/
```

RETAIN Statement

In contrast to the other DATA step statements, the RETAIN statement only requires the variables to be listed in the desired order.

```
data retain_method;
  retain a b c d;
  set sample;
```

USING PROC SQL

PROC SQL also provides a way to reorder variables in a SAS data set that already exists.

```
data sample;
input d c b a;
cards;
4 3 2 1
;
```

The PROC SQL example uses the CREATE TABLE statement to store the results of the query into a *table*. The resulting table is named *reordered* and will have the variables in the order in which the variables are listed in the SELECT statement. Only source data set variables that are listed will be included in the resulting table.

```
proc sql;
  create table reordered as
  select a, b, c, d
  from sample;

proc print data=reordered;
```

Obs	a	b	c	d
1	1	2	3	4

COMMENTS

The DATA step and PROC SQL statements have their advantages and disadvantages. Using PROC SQL is straightforward in that all required variables are listed in the SELECT statement. However, there is more to list when a large number of variables is involved. Variable attributes are not required as they are for most of the DATA step statements.

Except for the RETAIN statement, the DATA step statements require information about the variables, such as an attribute (format, informat, length, label). Compatible attributes must be carefully specified so that no data loss occurs. For example, when the variable has a length of 20 characters, using a length of 15 characters may result in the loss of data.

One approach with the DATA step statements is to use whatever attribute the variable already has as shown by PROC CONTENTS output.

Each ARRAY statement may only involve all numeric or all character variables. When two types of variables are involved, a series of ARRAY statements can be used to reorder the variables. The series is written until all the variables are enumerated. An example is shown below.

```
data sample;
input d c b a;
f='6';
e='5';
cards;
4 3 2 1
;

data array_method;
  array fixorder a b c d;
  array fix $ e f;
  set sample;

proc print data=array_method;
```

Obs	a	b	c	d	e	f
1	1	2	3	4	5	6

The SAS *Notes* about the DATA step statements say, "Variables not listed in these statements will retain their same positional order following the listed variables." Hence, the RETAIN statement does not require that all of the variables be listed. That is an advantage the RETAIN statement has over PROC SQL. However, there may be situations where it might be better to list all the variables to make it very clear what the final result will be.

CONCLUSION

PROC SQL and the DATA step's RETAIN statement provide a way to reorder data set variables by listing variable names according to the new order. In contrast, the other DATA step statements are not as straightforward and have a potential for human error in specifying attributes.

REFERENCES

SAS Institute Inc., *SAS® Language Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999. 1256 pp.

SAS Institute Inc., *SAS OnLineDoc®*, Version 8, Cary, NC: SAS Institute Inc., 1999.

TRADEMARK NOTICE

SAS is a registered trademark or trademark of the SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Imelda C. Go (icgo@juno.com)
Lexington County School District One
Lexington, SC