SESUG 2020 Paper 168

Controlling the Order of Data in SAS®

Imelda C. Go, Questar Assessment, Inc.

ABSTRACT

This paper will go over a number of SAS® features and techniques SAS programmers use to control the sort order of data in their data sets and reports. A custom sort order is very important because actual needs in the field are not limited to alphabetical, numeric, or alphanumeric ordering. There are many situations where a custom sort order is what is preferred in an organization and/or for reporting.

INTRODUCTION

SAS® programmers are not limited to defaults or PROC options that produce output sorted in a certain way. With a few simple techniques, you can sort your data however way you need to.

Sorting values in variables from an observation. This paper has examples of sorting values within a set of variables in an observation/record/row in the data set. That is, given a set of variables of the same type (either all numeric or all character), you can sort the values in the variables and reassign the values to the variables within the set. For example, A=2, B=3, C=1. You would like to sort the values such that, A=1, B=2, C=3.

Sorting observations (records/rows) in a data set. The other examples will focus on how to customize sorting observations in a data set. If you use BY-group processing in SAS, data will often be processed and provided in default sort order in SAS output data sets. Even if SAS provides the data in the order that is not desired, you will be able to change the sort order of its contents to suit your needs.

HOW SAS SORTS DATA VALUES

Before you produce your custom sort order, it helps to have a good understanding of how SAS sorts data.

- With a numeric variable, sorting is straightforward and the numeric values will be sorted per their position on the real number line as shown in Figure 1. Missing values will precede non missing values when in default ascending order. (SAS has three types of missing values: numeric, special numeric, and character.)

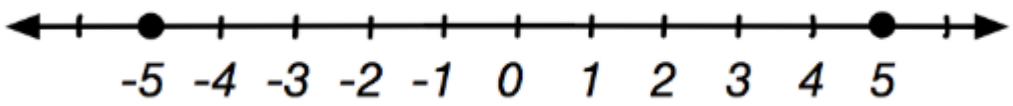

Figure 1. Example of a Real Number Line

With a character variable, the SAS programmer's concerns are greater:

- Consider all the types of characters that will appear in your data and determine how these will be sorted. Study how SAS sorts data and determine if the default sort order is sufficient for your needs.
  - Character values are case-sensitive.
  - What individual character values will appear your data? Will foreign letters, punctuation, symbols, and/or other special characters appear in the data?
  - How are numbers stored as character values sorted?
  - How are alphanumeric values sorted?

Table 1 provides examples of how SAS sorts data:

Variable Type	**Data Values**	**Values in Default Ascending Order**
Numeric	1, 2, 3, 10, 11	1, 2, 3, 10,11
Character	1, 2, 3, 10, 11	1, 10, 11, 2, 3
Character	aA, Aa, AA, aa	AA, Aa, aA, aa
Character	A, A-, B+, B, B-, C+, C, C-, D+, D, D-, F	A, A-, B, B+, B-, C, C+, C-, D, D+, D-, F
Character	Taylor, E. E. Taylor, E E Taylor	E. E. Taylor, E E Taylor, Taylor
Character	11, 10, 01, 02, 03	01, 02, 03, 10, 11

Table 1. Examples of How Data are Sorted by Default Ascending Order

- How do invisible non-printable characters affect the sort order and data processing?
  - The presence of non-printable characters (e.g., tab, hard returns, etc.) in the data can create problems. For example the string "ABC" will not be treated as the having the same value as "ABC" with a TAB mark ('09'x) is somewhere in the "ABC" string value. Since non-printable characters are just that (non-printable and "invisible"), the reader cannot view the non-printable characters without a special way of seeing them. You can remove them by using the following COMPRESS function with three arguments:
    - `compress(oldvalue,,'kw')`
    - The COMPRESS function removes all the non-printable characters in the character variable `oldvalue` (1st argument). The 2nd argument of the function is blank. The 3rd argument of KW means SAS will "keep the writable characters" and eliminates the non-printable characters from the string.
- Have you resolve inconsistencies in the data that can affect the sort order?

Table 2 lists a number of possible issues that you might encounter. It also lists a few character functions that you could use to resolve any data issues prior to sorting the data.

Possible Issues	**Examples of school variable values**	**Character Function**
Unwanted leading and trailing blanks	" School " vs. "School"	STRIP(school) removes leading and trailing blanks. LEFT, RIGHT, TRIM, TRIMN, STRIP are functions that remove blanks from strings.
Inconsistent case	"School" vs "SCHOOL"	UPCASE(school) converts all lower case letters to upper case. LOWCASE(school) converts all upper case letters to lower case.
Embedded multiple blanks	"High School" vs. "High School"	COMPBL(school) replaces two or more consecutive blanks with a single blank.
Change one character for another	"HIGH SCH00L" vs. "HIGH SCHOOL"	TRANSLATE(school,'O','0') changes all zeroes (0) to the capital letter O. Note that the O (what you want to change the character value to) is listed first and then followed by the character value you want to change.
Change one or more words in a string with other strings	"ABC ELEMENTARY SCHOOL" vs. "ABC ES"	TRANWRD(school,'Elementary School','ES') changes all occurrences of "Elementary School" to "ES". Note that what you want to change the value to is listed after you specify the value you want to change.

Table 2. Examples of Functions that Can Help with Data Issues

SORTING NUMERIC DATA PER OBSERVATION USING SORTN FUNCTION

Let us suppose you have the following `scores` data set with numeric student scores (`scores1-score5`):

student	score1	score2	score3	score4	score5
Jim	78	90	.	95	88
John	80	80	80	100	45
Mattthew	50	50	76	90	95

Let us suppose that we want the five scores for each student to be listed from lowest to highest in the `score1-score5` variables/columns. To sort the numeric values in ascending order, use the SORTN function:

```
data ordered; set scores;
   call sortn(of score1-score5);
```

The following data set will be produced:

student	score1	score2	score3	score4	score5
Jim	.	78	88	90	95
John	45	80	80	80	100
Mattthew	50	50	76	90	95

SORTING CHARACTER DATA PER OBSERVATION USING SORTC FUNCTION

Let us suppose you have the following `levels` data set with character student names (`student1-student4`):

medal	student1	student2	student3	student4
Gold	Mark	Ana	Gina	George
Silver	Lisa	Maria	Robert	Kim
Bronze	Sally	Emily	William	Gregory

Let us suppose that we want the four student names listed in alphabetical order in the `student1-student4` variables/columns. To sort the character values in `student1-student4` by name, use the SORTC function:

```
data ordered; set levels;
   call sortc(of student1-student4);
```

The following data set will be produced:

medal	student1	student2	student3	student4
Gold	Ana	George	Gina	Mark
Silver	Kim	Lisa	Maria	Robert
Bronze	Emily	Gregory	Sally	William

CREATING A CUSTOM SORT ORDER WITH ONE VARIABLE

Let us look at the `levels` data set below, which has records listed in Gold, Silver, and Bronze order:

medal	student1	student2	student3	student4
Gold	Mark	Ana	Gina	George
Silver	Lisa	Maria	Robert	Kim
Bronze	Sally	Emily	William	Gregory

Let us suppose we want to change the order of the records to Bronze, Silver, and Gold and produce the following:

medal	student1	student2	student3	student4
Bronze	Sally	Emily	William	Gregory
Silver	Lisa	Maria	Robert	Kim
Gold	Mark	Ana	Gina	George

This new order is not alphabetical by `medal` value order. We will need to apply a custom sort order to the data.

`proc format;`
`  invalue order`
`  'Bronze' = 1`
`  'Silver' = 2`
`  'Gold' = 3;`
	Create a user-defined informat that maps the values to a number that represents the order in which the records will appear.

`data ordered;`
`  set levels;`
`  neworder=input(level, order.);`
	Create a new numeric variable that will be used to sort on to produce the custom sort order.

`proc sort data=ordered;` `  by neworder;`	Sort on the new variable to produce the custom sort order.

The records will now appear in the custom sort order. Note that the variable `neworder` was added to the data set in order to use it as the BY variable in PROC SORT.

medal	student1	student2	student3	student4	neworder
Bronze	Sally	Emily	William	Gregory	1
Silver	Lisa	Maria	Robert	Kim	2
Gold	Mark	Ana	Gina	George	3

The two sets of PROC SQL code below will produce the same results:

medal	student1	student2	student3	student4
Bronze	Sally	Emily	William	Gregory
Silver	Lisa	Maria	Robert	Kim
Gold	Mark	Ana	Gina	George

Note that the PROC SQL does not need the addition of a new variable to sort the records. The ORDER BY portion of the code executes the sorting.

`proc sql;`
`create table ordered as`
`select * from levels`
`order by input(level, order.);`
`quit;`
	This also uses the user-defined format above. The sorting is done using the order defined in the informat.

`proc sql;`
`create table ordered as`
`select * from levels`
`order by case`
`when level = 'Bronze' then 1`
`when level = 'Silver' then 2`
`when level = 'Gold' then 3 end;`
`quit;`
	This does not use the user-defined informat above. The order of the records is articulated as shown.

CREATING A CUSTOM SORT ORDER WITH TWO OR MORE VARIABLES

We take the coding techniques used for sorting on the values of one variable and expand it to two variables. We can infer from this example how we can extend the code beyond two variables.

For this example, let us look at the list of gold, silver, and bronze medal recipients in the elementary, middle, and high schools. We want a data set that shows the medals in bronze, silver, and gold medal order and then by ES, MS, and HS level order. If we go by alphabetical order, the medal values will be listed as Bronze, Gold, and Silver; the level values will be listed as, ES, HS, and MS. These two sort orders are not alphabetical, and require a custom sort order for both variables.

medal	level	student1	student2	student3	student4
Gold	ES	Mark	Ana	Gina	George
Gold	HS	Chris	Susan	Leslie	Kevin
Gold	MS	Jim	Liz	James	Brenda
Silver	ES	Lisa	Maria	Robert	Kim
Silver	HS	Evelyn	Wendy	Stan	Eileen
Silver	MS	Helen	Steve	Ken	Amy
Bronze	ES	Sally	Emily	William	Gregory
Bronze	HS	Tina	Tim	Joel	Matthew
Bronze	MS	Pamela	Shirley	Justin	Richard

Our goal is to produce a data set sorted using two variables: by medal (Bronze, Silver, Gold order) and level (ES, MS, HS order).

`proc format;`
`  invalue order`
`  'Bronze' = 1`
`  'Silver' = 2`
`  'Gold' = 3;`
`  invalue school`
`  'ES' = 1`
`  'MS' = 2`
`  'HS' = 3;`
	Create two user-defined informats that map the values to a number that represents the order in which the records will appear.

`data ordered;`
`  set levels;`
`  neworder1=input(medal,order.);`
`  neworder2=input(level,school.);`
	Create a new numeric variable that will be used to sort on to produce the custom sort order.

`proc sort data=ordered;` `  by neworder1 neworder2;`	Sort on the new variable to produce the custom sort order.

The records will now appear in the custom sort order. Note that the variables `neworder1` and `neworder2` were added to the data set.

medal	level	student1	student2	student3	student4	neworder1	neworder2
Bronze	ES	Sally	Emily	William	Gregory	1	1
Bronze	MS	Pamela	Shirley	Justin	Richard	1	2
Bronze	HS	Tina	Tim	Joel	Matthew	1	3
Silver	ES	Lisa	Maria	Robert	Kim	2	1
Silver	MS	Helen	Steve	Ken	Amy	2	2
Silver	HS	Evelyn	Wendy	Stan	Eileen	2	3
Gold	ES	Mark	Ana	Gina	George	3	1
Gold	MS	Jim	Liz	James	Brenda	3	2
Gold	HS	Chris	Susan	Leslie	Kevin	3	3

The two sets of PROC SQL code below will produce the same results:

medal	level	student1	student2	student3	student4
Bronze	ES	Sally	Emily	William	Gregory
Bronze	MS	Pamela	Shirley	Justin	Richard
Bronze	HS	Tina	Tim	Joel	Matthew
Silver	ES	Lisa	Maria	Robert	Kim
Silver	MS	Helen	Steve	Ken	Amy
Silver	HS	Evelyn	Wendy	Stan	Eileen
Gold	ES	Mark	Ana	Gina	George
Gold	MS	Jim	Liz	James	Brenda
Gold	HS	Chris	Susan	Leslie	Kevin

Note that the PROC SQL code does not need the addition of new variables to sort the records. The ORDER BY portion of the code executes the sorting.

`proc sql;`
`create table studentlevels1 as`
`select * from levels`
`order by input(medal, order.),`
`input (level, school.);`
`quit;`
	This also uses the user-defined formats above. The sorting is done using the order defined in the informats.

`proc sql;`
`create table studentlevels2 as`
`select * from levels`
`order by`
`case when medal = 'Bronze' then 1`
`when medal = 'Silver' then 2`
`when medal = 'Gold' then 3 end,`
`case when level = 'ES' then 1`
`when level = 'MS' then 2`
`when level = 'HS' then 3 end;`
`quit;`
	This does not use the user-defined informat above. The order of the records is articulated as shown.

CONCLUSION

The SAS programming language offers different tools and techniques to produce a custom sort order.

REFERENCES

Cody, Ron. 2010. *SAS® Functions by Example*, Second Edition. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Imelda C. Go, Ph.D.
Questar Assessment, Inc.
igo@questarai.com