Paper CT03

Fairy-tale of Procedure: Proc Sort

Soujanya Konda, IQVIA, Düsseldorf, Germany

ABSTRACT

The entire clinical world revolves around data. Its representation, interpretation and analysis bring a fruitful product to existence. There are various significant programming techniques used to shape them. Though the ordering of data does not look like an imperative element, it plays a vital role throughout the life cycle of datasets and reports. Proc Sort is a crucial tool used for this task. It is very docile and an influential procedure that helps us to orchestrate data. This presentation will drive discussing tips and tricks for best utilizing Proc Sort.

INTRODUCTION

Ordering data is crucial and a vital step to orchestrate the clinical data. Proc Sort is the procedure that is used to sort the data per your requirements. This paper takes a deep dive into Proc Sort to beyond basics options to order the data.

PROC SORT

Proc Sort orders data by the values of one or more numeric or character variables. Proc Sort creates a new dataset or replaces the original dataset.

```
data class;
 set sashelp.class;
run;

proc sort data=class;
 by name age;
run;
```

SORTSIZE

Internal Sort is performed when Proc Sort has enough memory and it is usually performed when the data is read in. When there is not enough space, an external sort is performed, where Proc Sort sets up a temporary utility files on hard disk which can also slower the overall process.

Below example illustrates an example with SORTSIZE option in Proc Sort.

```
44
45
46    proc sort data=dm out=dm1 sortsize=5gb;
47     by usubjid;
48    run;

NOTE: There were 257431 observations read from the data set WORK.DM.
NOTE: The data set WORK.DM1 has 257431 observations and 325 variables.
NOTE: Compressing data set WORK.DM1 decreased size by 93.80 percent.
      Compressed is 1996 pages; un-compressed would require 32180 pages.
NOTE: PROCEDURE SORT used (Total process time):
      real time           11.62 seconds
      cpu time            17.23 seconds
```

```
49
50
51    proc sort data=dm out=dm2 sortsize=250mb;
52     by usubjid;
53    run;

NOTE: There were 257431 observations read from the data set WORK.DM.
NOTE: The data set WORK.DM2 has 257431 observations and 325 variables.
NOTE: Compressing data set WORK.DM2 decreased size by 93.80 percent.
      Compressed is 1996 pages; un-compressed would require 32180 pages.
NOTE: PROCEDURE SORT used (Total process time):
      real time           9.72 seconds
      cpu time            14.83 seconds
```

TAGSORT

When Proc Sort is invoked with the TAGSORT option the entire dataset is not read in. Instead, with the TAGSORT option the dataset is read as tags which are specified in BY statement.

The example below illustrates without and with TAGSORT option

```
298   proc sort data=class out=class_1_ ;
299      by usubjid age;
300   run;

NOTE: There were 320926 observations read from the data set WORK.CLASS.
NOTE: The data set WORK.CLASS_1_ has 320926 observations and 107 variables.
NOTE: Compressing data set WORK.CLASS_1_ decreased size by 91.47 percent.
      Compressed is 1441 pages; un-compressed would require 16891 pages.
NOTE: PROCEDURE SORT used (Total process time):
      real time           7.02 seconds
      cpu time            10.46 seconds
```

```
295   proc sort data=class out=class_1 tagsort;
296      by usubjid age;
297   run;

NOTE: Tagsort reads each observation of the input data set twice.
NOTE: The data set WORK.CLASS_1 has 320926 observations and 107 variables.
NOTE: Compressing data set WORK.CLASS_1 decreased size by 91.47 percent.
      Compressed is 1441 pages; un-compressed would require 16891 pages.
NOTE: PROCEDURE SORT used (Total process time):
      real time           3.53 seconds
      cpu time            3.55 seconds
```

PRESORTED

The PRESORTED option lets the procedure perform a check of whether the input data is sorted properly. If the dataset is already sorted, the dataset is copied, and a note written to the log: “Sort order of input data set has been verified”.

The example below illustrates without and with PRESORTED option:

```
304  proc sort data=class out=classw  ;
305     by usubjid age;
306  run;

NOTE: There were 320926 observations read from the data set WORK.CLASS.
NOTE: The data set WORK.CLASSW has 320926 observations and 107 variables.
NOTE: Compressing data set WORK.CLASSW decreased size by 91.47 percent.
      Compressed is 1441 pages; un-compressed would require 16891 pages.
NOTE: PROCEDURE SORT used (Total process time):
      real time           6.82 seconds
      cpu time            9.95 seconds


301  proc sort data=class out=classy presorted ;
302     by usubjid age;
303  run;

NOTE: Sort order of input data set has been verified.
NOTE: Input data set is already sorted; it has been copied to the output data set.
NOTE: There were 320926 observations read from the data set WORK.CLASS.
NOTE: The data set WORK.CLASSY has 320926 observations and 107 variables.
NOTE: Compressing data set WORK.CLASSY decreased size by 91.47 percent.
      Compressed is 1441 pages; un-compressed would require 16891 pages.
NOTE: PROCEDURE SORT used (Total process time):
      real time           3.40 seconds
      cpu time            3.41 seconds
```

REVERSE ORDER

Here the reverse collating sequence is used instead of a normal collating sequence while the Proc Sort procedure is used.

```
proc sort data=sashelp.class out=class;
 by name sex decending age;
run;
```

	NAME	SEX	AGE	HEIGHT	WEIGHT
1	Alfred	M	14	69	112.5
2	Alice	F	13	56.5	84
3	Barbara	F	13	65.3	98
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84
10	John	M	12	59	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90
13	Louise	F	12	56.3	77
14	Mary	F	15	66.5	112
15	Philip	M	16	72	150
16	Robert	M	12	64.8	128
17	Ronald	M	15	67	133
18	Thomas	M	11	57.5	85
19	William	M	15	66.5	112

SORT WITH DIFFERENT OPTIONS

The example below illustrates the usage of different options like **"format", "label", "keep", "drop" or "rename"** etc. along with Proc Sort procedure.

```
proc format;
   value $SEX 'F'='Female'
              'M'='Male';

quit;

proc sort data=class
          out=class2 (keep=name age sex
                height weight rename=(name=subject));
   label WEIGHT='Weight at Baseline';
   format sex $sex.;
   by age;
   where age > 12;
run;
```

OUTPUT DATA

Obs	SUBJECT	SEX	AGE	HEIGHT	Weight at Baseline
1	Alice	Female	13	56.5	84.0
2	Barbara	Female	13	65.3	98.0
3	Jeffrey	Male	13	62.5	84.0
4	Alfred	Male	14	69.0	112.5
5	Carol	Female	14	62.8	102.5
6	Henry	Male	14	63.5	102.5
7	Judy	Female	14	64.3	90.0
8	Janet	Female	15	62.5	112.5
9	Mary	Female	15	66.5	112.0
10	Ronald	Male	15	67.0	133.0
11	William	Male	15	66.5	112.0
12	Philip	Male	16	72.0	150.0

NODUP AND NODUPKEY

NODUP deletes the duplicate observations in dataset. We have two output datasets. If the exact match is found, then that observation is written in "*Out*" dataset and the duplicate observations are written in "*Dupout*" dataset

```
data class;
 set sashelp.class sashelp.class;
run;
```

```
proc sort data=class out=class2 dupout=dup nodup;
 by age;
run;
```

OUTPUT DATA

Obs	NAME	SEX	AGE	HEIGHT	WEIGHT
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0

NODUPKEY deletes the duplicate observation specified in the BY statement and the removed observations will not be written in output dataset.

```
proc sort data=class out=class2 nodupkey;
 by age;
run;
```

OUTPUT DATA

Obs	NAME	SEX	AGE	HEIGHT	WEIGHT
1	Joyce	F	11	51.3	50.5
2	James	M	12	57.3	83.0
3	Alice	F	13	56.5	84.0
4	Alfred	M	14	69.0	112.5
5	Janet	F	15	62.5	112.5
6	Philip	M	16	72.0	150.0

NOUNIQUEKEY

NOUNIQKEY helps to identify the duplicate observations. This option is available from version SAS 9.3.This option works as opposite to NODUPKEY. A sort key is unique when the observation containing a key within a BY group.

```
proc sort data=class out=class2 uniqueout=obs_delted nouniquekey;
 by name;
run;
```

SOURCE DATA

Obs	NAME	SEX	AGE	HEIGHT	WEIGHT
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0
20	Alfred	M	14	69.0	112.5
21	Alice	F	13	56.5	84.0
22	Barbara	F	13	65.3	98.0
23	Carol	F	14	62.8	102.5
24	Henry	M	14	63.5	102.5
25	James	M	12	57.3	83.0

NONUNIQUE OBSERVATIONS OUTPUT DATA

Obs	NAME	SEX	AGE	HEIGHT	WEIGHT
1	Alfred	M	14	69.0	112.5
2	Alfred	M	14	69.0	112.5
3	Alice	F	13	56.5	84.0
4	Alice	F	13	56.5	84.0
5	Barbara	F	13	65.3	98.0
6	Barbara	F	13	65.3	98.0
7	Carol	F	14	62.8	102.5
8	Carol	F	14	62.8	102.5
9	Henry	M	14	63.5	102.5
10	Henry	M	14	63.5	102.5
11	James	M	12	57.3	83.0
12	James	M	12	57.3	83.0

UNIQUE OBSERVATIONS OUTPUT DATA

Obs	NAME	SEX	AGE	HEIGHT	WEIGHT
1	Jane	F	12	59.8	84.5
2	Janet	F	15	62.5	112.5
3	Jeffrey	M	13	62.5	84.0
4	John	M	12	59.0	99.5
5	Joyce	F	11	51.3	50.5
6	Judy	F	14	64.3	90.0
7	Louise	F	12	56.3	77.0
8	Mary	F	15	66.5	112.0
9	Philip	M	16	72.0	150.0
10	Robert	M	12	64.8	128.0
11	Ronald	M	15	67.0	133.0
12	Thomas	M	11	57.5	85.0
13	William	M	15	66.5	112.0

EQUALS AND NONEQUALS

Observations with same BY variable values maintains the order with these two options. Sort order of resultant dataset differs in NOEQUALS, whereas in EQUALS it resembles the input dataset

```
proc sort data=class out=byyears noequals;
 by name;
run;
```

EQUALS OUTPUT DATA

Obs	NAME	SEX	AGE	HEIGHT	WEIGHT
1	Alfred	M	14	69.0	70.0
2	Alfred	M	14	69.0	112.5
3	Alice	F	13	56.5	70.0
4	Alice	F	13	56.5	84.0
5	Barbara	F	13	65.3	98.0
6	Barbara	F	13	65.3	50.0
7	Carol	F	14	62.8	70.0
8	Carol	F	14	62.8	102.5
9	Henry	M	14	63.5	170.0
10	Henry	M	14	63.5	102.5
11	James	M	12	57.3	90.0
12	James	M	12	57.3	83.0

NOEQUALS OUTPUT DATA

Obs	NAME	SEX	AGE	HEIGHT	WEIGHT
1	Alfred	M	14	69.0	112.5
2	Alfred	M	14	69.0	70.0
3	Alice	F	13	56.5	84.0
4	Alice	F	13	56.5	70.0
5	Barbara	F	13	65.3	98.0
6	Barbara	F	13	65.3	50.0
7	Carol	F	14	62.8	102.5
8	Carol	F	14	62.8	70.0
9	Henry	M	14	63.5	102.5
10	Henry	M	14	63.5	170.0
11	James	M	12	57.3	83.0
12	James	M	12	57.3	90.0

COLLATING SEQUENCE

SAS users no longer work exclusively with English-language data. The available translational tables are ASCII, DANISH, FINNISH, ITALIAN, NORWEGIAN, POLISH, REVERSE, SPANISH, and SWEDISH. These option works when ordering of the data to be achieved in above specified translational tables.

The below screen snap illustrates the alphanumeric characters in each language sorts

Alphanumeric Characters Sorted for Each Language

```
Danish:     0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÆØÅabcdefghijklmnopqrstuvwxyzæøå
Finnish:    0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖabcdefghijklmnopqrstuvwxyzåäö
Italian:    0123456789AÀBCÇDEÉÈFGHIÌJKLMNOÒPQRSTUÙVWXYZaàbcçdeéèfghiìjklmnoòpqrstuùvwxyz
Norwegian:  0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÆØÅabcdefghijklmnopqrstuvwxyzæøå
Spanish:    0123456789AÁaáBbCcDdEÉeéFfGgHhIÍiiJjKkLlMmNnÑñOÓoóPpQqRrSsTtUÚuúÜüVvXwXxYyZz
Swedish:    0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖabcdefghijklmnopqrstuvwxyzåäö
```

.

LINGUISTIC

This option considered powerful and vital option to character or alpha numeric data. Below specified few examples illustrates with different scenarios.

INPUT DATA

Obs	AETERM
1	Fever
2	Pain
3	pain
4	fever
5	Swelling
6	swell
7	Rash
8	Skin RASH

```
proc sort data=ae out = ae2 SORTSEQ =LINGUISTIC ;
      by aeterm;
run;
```

OUTPUT DATA

Obs	AETERM
1	fever
2	Fever
3	pain
4	Pain
5	Rash
6	Skin RASH
7	swell
8	Swelling

```
proc sort data=ae out = ae2 SORTSEQ =LINGUISTIC (case_first=upper);
 by aeterm;
run;
```

OUTPUT DATA

Obs	AETERM
1	Fever
2	fever
3	Pain
4	pain
5	Rash
6	Skin RASH
7	swell
8	Swelling

NUMERIC_COLLATION

This option helps to sort the data which is combination of characters and numbers .

INPUT DATA

Obs	VISIT
1	Day 22
2	Day 3
3	Day 2
4	Day 1
5	Day 11

```
proc sort data=visits out = vis1 SORTSEQ =LINGUISTIC (NUMERIC_COLLATION=ON);
 by visit;
run;
```

OUTPUT DATA

Obs	VISIT
1	Day 1
2	Day 2
3	Day 3
4	Day 11
5	Day 22

ALTERNATE_HANDLING

This option helps sort data with spaces and special characters.

INPUT DATA

Obs	INV
1	INVESTIGATOR1
2	INVESTIGATOR3
3	INVESTIGATOR11
4	INVESTIGATOR2
5	INVESTIGATOR33
6	INVESTIGATOR22

```
proc sort data=invest out=inv1
 SORTSEQ =LINGUISTIC (ALTERNATE_HANDLING=SHIFTED);
 by inv;
run;
```

OUTPUT DATA

Obs	INV
1	INVESTIGATOR1
2	INVESTIGATOR11
3	INVESTIGATOR2
4	INVESTIGATOR22
5	INVESTIGATOR3
6	INVESTIGATOR33

ALTERNATE_HANDLING WITH STRENGTH

Strength denotes the collation level and there are five collation levels specified in the below table.

INPUT DATA

Obs	INV
1	StANN
2	STALLEN
3	STAL
4	STELLER
5	Ac STELLER
6	AcSTELLER

```
proc sort data=invest out=inv1 SORTSEQ =LINGUISTIC (STRENGTH=3
CASE_FIRST=UPPER);
 by inv;
run;
```

OUTPUT DATA

Obs	INV
1	Ac STELLER
2	AcSTELLER
3	STAL
4	STALLEN
5	StANN
6	STELLER

The below table illustrates the strength related to the collation levels. There are five collation-levels of strength.

Value	Type of Collation	Description
PRIMARY or 1	PRIMARY specifies differences between base characters (for example, "a" < "b").	It is the strongest difference. For example, dictionaries are divided into different sections by base character.
SECONDARY or 2	Accents in the characters are considered secondary differences (for example, "as" < "às" < "at").	A secondary difference is ignored when there is a primary difference anywhere in the strings. Other differences between letters can also be considered secondary

		differences, depending on the language.
TERTIARY or 3	Upper and lowercase differences in characters are distinguished at the tertiary level (for example, "ao" < "Ao" < "aò"). For an example, see Linguistic Sorting Using ALTERNATE_HANDLING=.	A tertiary difference is ignored when there is a primary or secondary difference anywhere in the strings. Another example is the difference between large and small Ka
QUATERNARY or 4	When punctuation is ignored at level 1-3, an additional level can be used to distinguish words with and without punctuation (for example, "a-b" < "ab" < "aB"). For an example, see Linguistic Sorting Using ALTERNATE_HANDLING= and STRENGTH=.	The quaternary level should be used if ignoring punctuation is required or when processing Japanese text. This difference is ignored when there is a primary, secondary, or tertiary difference.
IDENTICAL or 5	When all other levels are equal, the identical level is used as a tiebreaker. The Unicode code point values of the Normalization Form D (NFD) form of each string are compared at this level, just in case there is no difference at levels 1-4.	This level should be used sparingly, because code-point value differences between two strings rarely occur. For example, only Hebrew cantillation marks are distinguished at this level.

CONCLUSION

Proc Sort is easiest procedure to order data. With the addition of these advanced options we have a powerful procedure to get the desired sorted data. It saves a remarkable amount of time for programmers when the data is alphanumeric, with special characters, and other complex challenges. Proc Sort will play a pivotal role in ordering and analyzing the data.

REFERENCES

http://documentation.sas.com/?cdcId=vdmmlcdc&cdcVersion=8.11&docsetId=proc&docsetTarget=p02bhn81rn4u64n1b6l00ftdnxge.htm&locale=en#n0h3wa9p7apg0kn154t3w6wutgib
https://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a000146878.htm#a003070987
https://sasnrd.com/sas-proc-sort-options/
https://support.sas.com/resources/papers/proceedings/proceedings/sugi31/030-31.pdf
https://www.pharmasug.org/proceedings/2015/QT/PharmaSUG-2015-QT14.pdf

ACKNOWLEDGMENTS

I take this opportunity to thank my managers at IQVIA, whose support represent in this conference. A special thanks to Srivalli Konda who gave me helping hand in meticulously organizing these words.

RECOMMENDED READING

- Base SAS® Procedures Guide
- SAS® For Dummies®

CONTACT INFORMATION

Author Name : Soujanya Konda
Company : IQVIA
Address : Germany
City / Postcode : Dusseldorf
Work Phone: : +49 17676818002
Email : soujanya.konda@quintiles.com,soujitheuniq@gmail.com