The SORT Procedure: Beyond the Basics
David Fickbohm, Homegain. Com, Emeryville, CA

ABSTRACT
The sort procedure is a very useful procedure with many options that are often overlooked. There are many
options that you can use in both a data step as well as in the SORT procedure, eliminating the need for
extra data steps. I will explain these options in detail, showing examples of how to use them. All examples
shown were done in SAS systems for PCs version 9.1.3.

INTRODUCTION
I think the first thing I learned to do in SAS was to sort data using PROC SORT. At the time I did not know
you can create a new dataset, subset your data, rename, drop, keep, format, or label your variables in the
same procedure? In this paper I explain how to do all of these things and more using PROC SORT.

OUT= OPTION
I do not know how many times I created a data set by using a DATA step and then sorted this new data set
using PROC SORT. But I assure you I have not done that since I learned that I could create a new dataset
in PROC SORT. To create a new dataset you add the OUT= option. Without the OUT= option, PROC
SORT overwrites the original data set

Here is an example:

data WUSS;
input student 1-2 sex $ 3-4 age 5-7;
datalines;
01 F 25
04 M 23
03 M 27
05 F 22
02 F 29
03 M 27
04 M 23
; run;
proc sort data=WUSS out = WUSS0;
by student;
run;

Here, I first created a data set, called WUSS, which will be used throughout this paper. This data set
contains student identification numbers, sex, age at entry, and performance status. I created a new data set
called WUSS0 that is sorted by student number using PROC SORT. The output data set WUSS1 follows:

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>SEX</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>F</td>
<td>25</td>
</tr>
<tr>
<td>02</td>
<td>F</td>
<td>29</td>
</tr>
<tr>
<td>03</td>
<td>M</td>
<td>27</td>
</tr>
<tr>
<td>04</td>
<td>M</td>
<td>23</td>
</tr>
<tr>
<td>05</td>
<td>F</td>
<td>22</td>
</tr>
</tbody>
</table>

This is just a basic sort leaving all the same variables and number of observations as in the WUSS dataset
but the data set WUSS1 is sorted by student number.

SORTING BY THE VALUES OF MULTIPLE VARIABLES
To sort a dataset by the values of multiple variables list the variables in the order in which you want the data
set sorted. PROC SORT first arranges the observations in the order of the first variable in the by statement;
then it sorts observations with a given value by the second variable, and so on. Here is an example:

proc sort data=WUSS out = WUSS1;
by student sex;
run;

WUSS1
<table>
<thead>
<tr>
<th>STUDENT</th>
<th>SEX</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>F</td>
<td>22</td>
</tr>
</tbody>
</table>
DESCENDING OPTION
It may be useful to reverse the order of the BY variable at times. You can do this using the DESCENDING option. This option will sort the data set in descending order by the variable that immediately follows the word DESCENDING in the BY statement. Here is an example followed by the output data set:

```
proc sort data=wuss out=wuss2;
   by descending student;
run;
```

WUSS2 data set:
STUDENT SEX AGE
05 F 22
04 M 23
04 M 23
03 M 27
03 M 27
02 F 29
01 F 25

You can see that the data set WUSS2 is now sorted with student number 5 first and student number 1 last.

DROP=, KEEP=, AND RENAME= OPTIONS
The DROP=, KEEP=, and RENAME= options are available to you within the SORT procedure just as they are within a DATA step. Here are some examples using these options along with the output data sets:

```
proc sort data=wuss out=wuss3(keep=student age);
   by student;
run;
```

WUSS3 data set:
STUDENT AGE
01 25
02 29
03 27
03 27
04 23
04 23
05 22

```
proc sort data=wuss out=wuss4(rename=(student=st));
   by student;
run;
```

WUSS4 data set:
ST SEX AGE
01 F 25
02 F 29
03 M 27
03 M 27
04 M 23
04 M 23
05 F 22

There are a few of things to keep in mind when using these options. First, if you use the RENAME= option, SAS changes the name of the variable in that procedure. Second, if you use RENAME= with either the

- 2 -
DROP= or the KEEP= options, the DROP= and the KEEP= options are applied before RENAME=. Thus, use the “old name” in the DROP= and KEEP= options. The order in which you place the RENAME=, DROP=, and KEEP= options does not matter and does not change process order. Using parentheses, a list of multiple variables to rename must be enclosed in parentheses, renaming just one variable does not, and the KEEP= variables should not be enclosed in parentheses. Another thing to remember is that you cannot drop and rename the same variable in the same statement. Here is an example of using both the KEEP= and RENAME= options:

```plaintext
proc sort data=wuss out=wuss5(rename=(student=st age=entryage) keep=student age);
by student;
run;
```

WUSS5 data set:
ST ENTRYAGE
01 25
02 29
03 27
03 27
04 23
04 23
05 22

Notice that in this example you have to use the variable names STUDENT and AGE in the KEEP= statement instead of ST and ENTRYAGE. Also notice that you have to use STUDENT instead of ST in the BY statement. You can actually rearrange this example in a couple of different ways and end up with the same result.

Below are two alternate ways:

```plaintext
proc sort data=wuss out=wuss5(keep=student age rename=(student=st age=entryage));
by student;
run;
```

```plaintext
proc sort data=wuss(keep=student age rename=(student=st age=entryage)) out=wuss5;
by st;
run;
```

Note that in the second rearrangement above that the variable name ST must be used instead of STUDENT in the BY statement because the KEEP= and RENAME= options are used before the OUT= option versus after it like in the original configuration.

**FORMAT AND LABEL STATEMENTS**

Other statements that are the same in the SORT procedure as in a DATA step are the FORMAT and LABEL statements. You can apply a variable format or create variable labels within PROC SORT. Let’s look at format first. An example follows:

```plaintext
proc format;
value $SEX 'F'='Female' 'M'='Male';
run;
proc sort data=wuss out=wuss6;
format sex $SEX.;
by student;
run;
```

WUSS6 data set:
STUDENT SEX AGE
01 Female 25
02 Female 29
03 Male 27
The format $SEX was created in the FORMAT procedure. The $SEX format was applied to the variable SEX in PROC SORT statement. In the output data set, instead of 'F' and 'M', you see the formatted values 'Female' and 'Male'. Note that the FORMAT statement does not permanently alter the variables in the input data set.

Next we will look at an example using labels:

```
proc sort data=wuss out=wuss7;
label age='Age at Entry';
by student;
run;
proc print data=wuss7 label;
run;
```

The output looks like:

```
Age at Entry
Obs student sex Age at Entry
1 01 F 25
2 02 F 29
3 03 M 27
4 03 M 27
5 04 M 23
6 04 M 23
7 05 F 22
```

By using the PRINT procedure with the label option following the PROC SORT statement, you can see the labels created for the variables PS and AGE. Like the FORMAT statement, the LABEL statement does not permanently alter the variables in the input data set.

**WHERE= OPTION OR WHERE STATEMENT**

When it is necessary to subset your data the SORT procedure allows you to do this using the WHERE= option or WHERE STATEMENT. Both of these work similarly by selecting observations that meet the condition specified in the WHERE expression before SAS brings them into the PROC SORT for processing. This can improve the efficiency of your SAS programs because SAS is not required to read all observations from the input data set. Here is an example:

```
proc sort data=wuss(where=(age>50)) out=wuss8;
by student;
run;
```

Here is an example using the WHERE statement:

```
proc sort data=wuss out=wuss8;
where age>25;
by student;
run;
```

Both of these produce the same output data set:

```
STUDENT SEX AGE
02 P 29
03 M 27
03 M 27
```

A nice feature of the WHERE= option and WHERE statement is being able to use some exclusive WHERE expressions such as: BETWEEN-AND, ? or CONTAINS, IS NULL or IS MISSING, LIKE (matches patterns), =* (sounds like), and SAME-AND.

**FIRSTOBS= AND OBS= OPTIONS**
There can be circumstances when you have a very large data set and you would like to split it up into smaller, more manageable data sets. This may be a time when it would be helpful to use the FIRSTOBS= and OBS= options within the SORT procedure. The FIRSTOBS= option causes SAS to begin reading at a specified observation or record. The OBS= option specifies at which observation SAS processing ends. The two options are often used together to define a range of observations to be processed. However, you do not have to use both options together. If you do not include the OBS= option with the FIRSTOBS= option, by default PROC SORT will stop at the last observation. If you do not include the FIRSTOBS= option with the OBS= option, by default PROC SORT will start at the first observation.

Here is an example of using both the FIRSTOBS= and OBS= options:

```
proc sort data=wuss(firstobs=3 obs=5) out=wuss9;
by student;
run;
```

WUSS9 data set:

```
STUDENT SEX AGE
02 F 29
03 M 27
05 F 22
```

In this example, the procedure first took observations 3, 4, and 5 from the original, unsorted data set WUSS and then sorted it by STUDENT.

**NODUPRECS AND NODUPKEY OPTIONS**

The NODUPRECS (or NODUP) and NODUPKEY options work similarly in that they both can eliminate unwanted observations, but NODUP compares all the variables in your data set while NODUPKEY compares just the BY variables. More specifically, if you use the NODUP option, PROC SORT compares all variable values for each observation to those for the previous observation that was written to the output data set. If an exact match is found, the observation is not written to the output data set. If you specify the NODUPKEY option, PROC SORT compares all BY variable values for each observation to those for the previous observation written to the output data set. If an exact match using the BY variable values is found, the observation is not written to the output data set. First I will show an example of the NODUP option:

```
proc sort data=wuss nodup out=wuss10;
by student;
run;
```

WUSS10 data set:

```
STUDENT SEX AGE
01 F 25
02 F 29
03 M 27
04 M 23
05 F 22
```

Notice in the output data set WUSS10, the second observation for student 03 is eliminated because all the variable values are the same. However, the second observation for student 04 is not eliminated because performance score is different for these two observations. An example of the NODUPKEY option follows:

```
proc sort data=wuss nodupkey out=wuss11;
by student;
run;
```

WUSS11 data set:

```
STUDENT SEX AGE
01 F 25
02 F 29
03 M 27
04 M 23
05 F 22
```
In this output data set you can see that the second observations for both student 03 and 04 are eliminated because only the BY variable (in this case STUDENT) had to be the same. In this example, using NODUPKEY instead of NODUP may not be the best option to use since by using NODUPKEY, you eliminate the second observation for student 04 even though there was a different performance score for this observation. This may or may not be what you want to do.

DUPOUT

DUPOUT= SAS-data-set
Dupout specifies the output data set to which duplicate observations are written. Here is an example:

```sas
proc sort data=wuss nodupkey out=wuss11 dupout = wuss11dup;
by student;
run;
```

WUSS11 data set:
STUDENT SEX AGE
01 F 25
02 F 29
03 M 27
04 M 23
05 F 22

WUSS11dup data set:
STUDENT SEX AGE
03 M 23
04 M 27

INTEGRITY CONSTRAINTS:

Sorting the input data set and replacing it with the sorted data set preserves both referential and general integrity constraints, as well as any indexes that they may require. A sort that creates a new data set will not preserve any integrity constraints or indexes.

OPTIONS

Options can include one collating-sequence-option and multiple other options. The order of the two types of options does not matter and both types are not necessary in the same PROC SORT step.

Collating-Sequence-Options
There are collating sequences other than ASCII and EBCDIC. The collating sequence is set each time a SAS session opens. The collating sequence is determined by the session encoding ("SAS System Options" -> "Environmental Control" -> "Language Control" -> "Encoding". The encoding that is set in relation to the operating system. The collating sequence can be set to another collating-sequence using the SORTSEQ = option and the name of the collating sequence, DANISH, NORWEGIAN, etc.

Other Options
These include but are not limited to NODUPKEY, DUPOUT. The REVERSE option reverses the collation order for character variables. EQUALS maintains relative order within BY groups. NOEQUALS does not maintain relative order within BY groups. EQUALS is the default. Using NOEQUALS can save CPU time and memory. Check Base SAS procedures guide for a complete list and their uses.

STORED SORT INFORMATION
PROC SORT records the BY variables, collating sequence, and character set that it uses to sort the data set. This information is stored with the data set to help avoid unnecessary sorts.

Before PROC SORT sorts a data set, it checks the stored sort information. If you try to sort a dataset the way that it is currently sorted, PROC SORT will not perform the sort. It will write a message to the log to that effect. To override this behavior, use the FORCE option. If you try to sort a data set the way that it is currently sorted and you specify an OUT= dataset, then PROC SORT simply make a copy of the input or DATA = data set.

CONCLUSION
I hope you have learned that there are many different options available in the SORT procedure. Hopefully you have learned more about these options and will use them to enhance your programs and improve efficiency.

REFERENCES
Proceedings of the Thirty-First Annual SAS Users Group International Conference

ACKNOWLEDGEMENTS
I would like to thank my wife Nancy for her support and her editing skills. This paper would not have been completed with her.
I would like to thank Bob Maggio, SAS Technical Support, for his review and helpful comments on this paper.

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Please feel free to contact the author at:
davefickbohm@yahoo.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 registered trademarks or trademarks of their respective companies.