ABSTRACT

What if the usual sort and usual group processing would eliminate the existing logical order and grouping of the data that must be maintained to ensure the needed analysis is correct. This paper presents, to intermediate SAS programmers, some uses of the NOTSORTED option of the BY statement when working with group processing in data steps and procedure steps.

INTRODUCTION

Beginning SAS® programmers soon realize that the proc sort procedure is one of the most frequently used procedures. Often, proc sort with a BY statement is used at the start of some type of group processing. But what if the usual sort and the usual group processing eliminates the desired logical order and logical grouping of the data and consequently produces an inappropriate analysis. This paper presents some uses of BY’s NOTSORTED option when working with group processing in data steps and procedure steps.

First, this paper describes a data problem and addresses the problem by presenting three solutions. All three solutions are based on the power of BY’s NOTSORTED option. Then we examine how BY’s NOTSORTED option impacts data step processing. Next we examine how BY’s NOTSORTED option impacts procedure processing and the processing of various commonly used procedures. To simplify the discussion and display, we use small illustrative data sets, such as DS1 (below), and remove the SAS headers and date information.

A DATA ANALYSIS PROBLEM SOLVED USING BY’S NOTSORTED OPTION

In this section, a data analysis problem is described and analyzed. Then three solutions are presented: a data step solution, a procedure step solution, and a pre-processing solution. Each is based on the power of BY’s NOTSORTED option.

A Data Analysis Problem
Throughout this section and much of this paper, we use the data set DS1. The following typical code creates a typical summary data set SUMDS1.

```
DATA SET: DS1
STATE   AMT
PA      200
PA      100
CA      300
CA      200
MA      200
MA      400
PA      100
PA      300
```

```
CODE
proc sort data=DS1 out=DS1s;
by STATE; run;
data SUMDS1 (drop=AMT);
set DS1s;
by STATE;
if first.STATE then TOTAMT = 0;
TOTAMT+AMT;
if last.STATE then output; run;
```

```
DATA SET: SUMDS1
STATE   TOTAMT
CA      500
MA      600
PA      700
```
Notice that the original data set DS1 is grouped but the groups are not ordered in either ascending or descending order and there are two groupings for the STATE variable with value 'PA'. The problem appears because the summary data set must look like SUMDS2.

<table>
<thead>
<tr>
<th>DATA SET: SUMDS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE</td>
</tr>
<tr>
<td>PA</td>
</tr>
<tr>
<td>CA</td>
</tr>
<tr>
<td>MA</td>
</tr>
<tr>
<td>PA</td>
</tr>
</tbody>
</table>

With the sort we get both an inappropriate order and grouping. If the sort is not executed and the above data step is executed for DS1 the SAS Log contains the error message:

OUTPUT ERROR: BY variables are not properly sorted on data set WORK.DS1.

By the relational nature of SQL, proc sql cannot do BY-NOTSORTED group processing. Consequently, for beginners there is no self-evident way to manipulate DS1 and output SUMDS2.

**A Data Step Solution**

As usual, we rely on base SAS data step coding for one or more solutions. The above code can be modified to provide a solution by removing the proc sort call and making only a few changes to the above data step code. The solution uses the power of BY's NOTSORTED option.

```sas
data SUMDS2 (drop=AMT);
set DS1;
by STATE NOTSORTED;
if first.STATE then TOTAMT = 0;
TOTAMT+AMT;
if last.STATE then output; run;
```

**A Procedure Solution**

This solution also uses the power of BY's NOTSORTED option. Using only a CLASS statement will not give the desired data set SUMDS2. Also notice, this code does not use a proc sort.

```sas
proc means data=DS1 sum maxdec=0 noprint;
by STATE NOTSORTED;
var AMT; output out=sumds2(drop=_type_ _freq_ ) sum(amt)=TOTAMT ; run;
```

**A Pre-processing Solution**

This code adds a column, GrpNum, such that any follow-up analysis can use the column for sorting and grouping. In this way, we can avoid the further use of BY's NOTSORTED option.

<table>
<thead>
<tr>
<th>DATA SET: DS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE</td>
</tr>
<tr>
<td>PA</td>
</tr>
<tr>
<td>PA</td>
</tr>
<tr>
<td>CA</td>
</tr>
<tr>
<td>CA</td>
</tr>
<tr>
<td>MA</td>
</tr>
<tr>
<td>MA</td>
</tr>
<tr>
<td>PA</td>
</tr>
<tr>
<td>PA</td>
</tr>
</tbody>
</table>
THE DATA STEP'S BY STATEMENT WITH THE NOTSORTED OPTION

Variables in a BY statement are called BY variables. A BY group is a set of contiguous rows that have the same values for all BY variables. The BY variables are used to form BY groups.

The BY statement applies only to the SET, MERGE, MODIFY, or UPDATE statement that immediately precedes it in the DATA step, and only one BY statement can accompany each of these statements in a data step.

Now we examine how BY’s NOTSORTED option impacts data step processing. If you do not use BY’s NOTSORTED option, the rows in the data set must either be sorted by the BY variables specified or the data set must be indexed appropriately. That is, NOTSORTED specifies that the rows are not necessarily in sorted order, but are logically grouped in some other way (perhaps chronological order or into categories). In other words, the NOTSORTED option does not mean the data is unsorted, rather that the data is arranged in groups (according to the values of the BY variables) and that the groups are not necessarily in ascending or descending order. The NOTSORTED option informs SAS that the rows are grouped by the BY variables, but are not presented in a sorted order. Briefly, NOTSORTED indicates that BY group processing takes place on grouped, rather than sorted data. Anytime any one of the BY variables changes value, SAS interprets this as a new BY group. Once again, if rows with the same values for the BY variables are not contiguous, a data step with BY-NOTSORTED processing treats each contiguous set as a separate BY group.

The data sets that are listed in the SET, MERGE, or UPDATE statements must be sorted by the values of the variables that are listed in the BY statement or have an appropriate index. As a default, SAS expects the data sets to be in ascending order. Consequently, NOTSORTED cannot be used with the MERGE statement, UPDATE statement, or when the SET statement lists more than one data set. For completeness, MODIFY does not require sorted data, but sorting can improve performance. When using the SET statement with multiple data sets (interleaving) and the NOTSORTED option is specified in the BY statement, unpredictable row groupings could result and the following error message is produced:

```
BY NOTSORTED MAY NOT BE USED WITH SET STATEMENT WHEN MORE THAN ONE DATA SET IS SPECIFIED.
```

The NOTSORTED option can appear once or more times anywhere in the BY statement.

The BY statement also has the GROUPFORMAT option. This option specifies the data step uses the formatted values, instead of the internal values, of the BY variables to determine where BY groups begin and end, and consequently how FIRST.var and LAST.var get assigned. Although the GROUPFORMAT option can appear anywhere in the BY statement, the option applies to all BY variables. If, in addition, the NOTSORTED option is used the rows can be grouped by the formatted value of the BY variables without needing the data set to be sorted or indexed.

```
proc format;
value $stfmt 'MA'='Small'
'PA','CA'='Large'; run;
```
data state;
set ds1;
by state GROUPFORMAT NOTSORTED;
format state $stfmt.; run;

proc print data=state;
by state NOTSORTED;
sum amt; run;

<table>
<thead>
<tr>
<th>STATE</th>
<th>AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>800</td>
</tr>
<tr>
<td>Large</td>
<td>600</td>
</tr>
</tbody>
</table>

THE PROCEDURE STEP’S BY STATEMENT WITH THE NOTSORTED OPTION

Recall, the variables in a BY statement are called BY variables. A BY group is a set of contiguous rows that have the same values for all BY variables. The BY variables values are used to form BY groups.

First we make comments that apply to procedures that use the BY statement and allow the NOTSORTED option on the BY statement. Next, in the subsections, we examine how BY’s NOTSORTED option impacts several frequently used types of procedure steps. As the usage of the BY statement differs in each procedure, for details refer to the SAS documentation.

A procedure does not use an index if you specify NOTSORTED. More accurately, when you use the NOTSORTED option the requirement for ordering or indexing rows according to the values of BY variables is suspended for BY-group processing.

Only one BY statement can be used in each PROC step. A procedure creates output for each BY group. If rows with the same values for the BY variables are not contiguous, a procedure with BY-NOTSORTED processing treats each contiguous set as a separate BY group. The statistics procedures perform separate analyses for each BY group and the reporting procedures produce a report for each BY group.

A procedure with a BY statement expects an input data set that is sorted by the order of the BY variables or one that has an appropriate index. An error occurs if the input data set does not meet these criteria. Either sort it with the PROC SORT or create an appropriate index on the BY variables or use the NOTSORTED or DESCENDING option in the procedure step BY statement.

Here are a few representative procedures supporting the BY statement with the NOTSORTED option: btl, calendar, chart, compare, corr, forms, freq, glm, means, mi, nested, plot, print, rank, report, score, standard, summary, tabulate, timeplot, transpose, and univariate. Most of the general comments about the BY statement NOTSORTED option for a procedure step apply to the various SAS statistical procedure steps, such as btl, corr, glm, nested, score, and standard. The following subsections make additional comments and provide examples for several of the more common procedure steps and how BY’s NOTSORTED option impacts procedure processing.
The Sort Procedure Step's By Statement With The NOTSORTED Option
The BY statement in the SORT procedure specifies how to sort the data. Consequently, the NOTSORTED option cannot be used in a PROC SORT step. In other procedures, the BY statement specifies how the data are currently sorted.

The Print Procedure Step's BY Statement With The NOTSORTED Option
All base procedures except PROC PRINT process BY groups completely independently. PROC PRINT is able to report the number of rows in each BY group and the number of rows in all BY groups. Also, PROC PRINT can sum numeric variables in a BY group and across all BY groups.

We will see the results of a) not using the NOTSORTED option, b) positioning NOTSORTED at various locations in the BY statement, c) using NOTSORTED multiple times in the BY statement, d) and not specifying any by variables in the BY-NOTSORTED statement.

<table>
<thead>
<tr>
<th>DATA SET: STCNTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST COUNTY AMT</td>
</tr>
<tr>
<td>PA 03 300</td>
</tr>
<tr>
<td>CA 01 500</td>
</tr>
<tr>
<td>CA 01 100</td>
</tr>
<tr>
<td>MA 01 600</td>
</tr>
<tr>
<td>PA 03 600</td>
</tr>
<tr>
<td>PA 02 200</td>
</tr>
</tbody>
</table>

**CODE**
proc print data=stcnty;
by st county; run;

**OUTPUT**
ERROR: Data set WORK.STCNTY is not sorted in ascending sequence. The current by-group has st = PA and the next by-group has st = CA.
NOTE: The SAS System stopped processing this step because of errors.

**CODE**
proc print data=stcnty;
by NOTSORTED st county;
sum amt; run;

**OUTPUT**
   ---- ST=PA COUNTY=03
   Obs   AMT
   1     300
   ---- ST=CA COUNTY=01
   Obs   AMT
   2     500
   3     100
   ------ ----
   COUNTY 600
   ST 600
   ---- ST=MA COUNTY=01
   Obs   AMT
   4     600
   ---- ST=PA COUNTY=03
   Obs   AMT
   5     600
   ---- ST=PA COUNTY=02
   Obs   AMT
   6     200
   ------ ----
   ST 800
   ----
   2300
The following code produces similar results with a different layout:

```latex
\begin{verbatim}
proc print data=stcnty;
  by NOTSORTED; sum amt; run;
\end{verbatim}
```

```
<table>
<thead>
<tr>
<th>Obs</th>
<th>st</th>
<th>county</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PA</td>
<td>03</td>
<td>300</td>
</tr>
<tr>
<td>2</td>
<td>CA</td>
<td>01</td>
<td>500</td>
</tr>
<tr>
<td>3</td>
<td>CA</td>
<td>01</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>MA</td>
<td>01</td>
<td>600</td>
</tr>
<tr>
<td>5</td>
<td>PA</td>
<td>03</td>
<td>600</td>
</tr>
<tr>
<td>6</td>
<td>PA</td>
<td>02</td>
<td>200</td>
</tr>
</tbody>
</table>

===

2300
```

For proc print, when you use an ID statement, the row numbers are not printed. The variables in the ID variable list are listed on the left-hand side of the report. If all BY variables appear in the same order at the beginning of the ID statement, proc print uses a special layout.

```latex
\begin{verbatim}
proc print data=DS1;
  by state NOTSORTED; run;
\end{verbatim}
```

```
<table>
<thead>
<tr>
<th>STATE=PA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs  AMT</td>
</tr>
<tr>
<td>1     200</td>
</tr>
<tr>
<td>2     100</td>
</tr>
</tbody>
</table>

----STATE=CA

| Obs  AMT |
| 3     300 |
| 4     200 |

----STATE=MA

| Obs  AMT |
| 5     200 |
| 6     400 |

----STATE=PA

| Obs  AMT |
| 7     100 |
| 8     300 |
```

## Posters

NESUG 2006

---

**OUTPUT**

The same as the last output.
The Means Procedure Step's BY Statement With The NOTSORTED Option
Proc means runs significantly faster for large numbers of groups if the data is first sorted with proc sort and then a BY statement with the classification variables instead of a CLASS statement. The CLASS statement informs the means procedure to build a table of all possible combinations of class variables. The BY statement allows proc means to process a single group, then write it to the output data set, and access the same storage to process the next group. The BY statement needs either a sorted data set or a grouped data set with BY's NOTSORTED option. For example code, see a previous section, 'A Data Analysis Problem Solved Using BY's NOTSORTED Option'.

**DATA SET: DSmean1**
<table>
<thead>
<tr>
<th>STATE</th>
<th>AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA</td>
<td>200</td>
</tr>
<tr>
<td>PA</td>
<td>100</td>
</tr>
<tr>
<td>CA</td>
<td>300</td>
</tr>
<tr>
<td>MA</td>
<td>400</td>
</tr>
<tr>
<td>PA</td>
<td>300</td>
</tr>
</tbody>
</table>

**CODE**
proc means data=DSmean1 mean maxdec=0;
by state NOTSORTED;
class state / order=data;
var amt; run;

**OUTPUT**

<table>
<thead>
<tr>
<th>STATE</th>
<th>CountY</th>
<th>CountY</th>
<th>State</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA</td>
<td>03</td>
<td>03</td>
<td>PA</td>
<td>300</td>
</tr>
<tr>
<td>CA</td>
<td>01</td>
<td>01</td>
<td>CA</td>
<td>500</td>
</tr>
<tr>
<td>MA</td>
<td>01</td>
<td>01</td>
<td>MA</td>
<td>600</td>
</tr>
<tr>
<td>PA</td>
<td>02</td>
<td>02</td>
<td>PA</td>
<td>200</td>
</tr>
</tbody>
</table>

The following does not use a CLASS statement.

**CODE**
proc means data=DSmean1 mean maxdec=0;
by state NOTSORTED;
var amt; run;

**OUTPUT**

<table>
<thead>
<tr>
<th>STATE</th>
<th>Analysis Variable: AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE</td>
<td>N Obs</td>
</tr>
<tr>
<td>PA</td>
<td>2</td>
</tr>
<tr>
<td>CA</td>
<td>1</td>
</tr>
<tr>
<td>MA</td>
<td>1</td>
</tr>
<tr>
<td>PA</td>
<td>1</td>
</tr>
</tbody>
</table>
Analysis Variable: AMT
Mean
300
STATE=MA
Analysis Variable: AMT
Mean
400
STATE=PA
Analysis Variable: AMT
Mean
300

The Univariate Procedure Step’s BY Statement With The NOTSORTED Option
When categorical variables are created from a continuous variable, proc univariate can check the
accuracy, with code like:

```
proc univariate data=ds1 noprint;
by state NOTSORTED;
output out=sumry n=NonMissing
median=Median
qrange=QRange; run;
```

| DATA SET: SUMRY |
|-----------------|-----------------|-----------------|-----------------|
| STATE    | NonMissing | Median | QRange |
| PA      | 2          | 300    | 100   |
| CA      | 2          | 500    | 100   |
| MA      | 2          | 100    | 200   |
| PA      | 2          | 600    | 200   |

The Freq Procedure Step’s BY Statement With The NOTSORTED Option
Proc freq shows the distribution of categorical data values and can reveal some data
regularities. The first sample code shows the usual proc freq analysis for data set DS1. Notice
the output has grouped all the PA data into a single group. However, the second sample code
that uses BY’s NOTSORTED option produces a separate analysis for each by grouping.
Specifically, there are two by groupings for State=’PA’.

```
proc freq data=ds1;
run;
```

<table>
<thead>
<tr>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE</td>
</tr>
<tr>
<td>CA</td>
</tr>
<tr>
<td>MA</td>
</tr>
<tr>
<td>PA</td>
</tr>
</tbody>
</table>

The following code uses BY-NOTSORTED processing:

```
proc freq data=ds1;
by state NOTSORTED;
table state / nopercent nocum; run;
```

<table>
<thead>
<tr>
<th>OUTPUT</th>
</tr>
</thead>
</table>
| STATE=PA
STATE  | Frequency |
| PA      | 2          |
| STATE=CA
STATE  | Frequency |
| CA      | 2          |
| STATE=MA
STATE  | Frequency |
| MA      | 2          |
| PA      | 2          |
The Report Procedure Step's BY Statement With The NOTSORTED Option

Proc report can produce much of what is done with proc print, proc means, proc sort, proc tabulate, and the data step.

```
PROC REPORT DATA=DS1 NOWD
   HEADLINE HEADSKIP LS=90
   PS=50 SPLIT='*' CENTER;
   COLUMN STATE AMT;
   BY STATE
   BREAK AFTER STATE / SKIP;
   DEFINE STATE / 'STATE'
      GROUP ORDER=INTERNAL
      WIDTH=5;
   DEFINE AMT / 'AMOUNT'
      WIDTH=10;
RUN;
```

**OUTPUT**

```
---- STATE=PA
  STATE AMOUNT
  ----
  PA 300

---- STATE=CA
  STATE AMOUNT
  ----
  CA 500

---- STATE=MA
  STATE AMOUNT
  ----
  MA 600
```

The Transpose Procedure Step's BY Statement With The NOTSORTED Option

Proc transpose does not transpose BY groups. Instead, for each BY group, proc transpose creates a row for each variable that it transposes. If the BY statement's NOTSORTED option is not specified, the rows must be sorted by all the BY variables or all the BY variables must be indexed appropriately. The following sample code shows how a 'wide' data set can be made 'long'. In this example, so as not to loose the link between 200 and 240 (the association between each AMT with a specific WT), a variable (UI) is first added to uniquely identify each row in the original data set.

**DATA SET:** DS6

<table>
<thead>
<tr>
<th>ST</th>
<th>AMT</th>
<th>WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA</td>
<td>200</td>
<td>240</td>
</tr>
<tr>
<td>PA</td>
<td>100</td>
<td>220</td>
</tr>
<tr>
<td>CA</td>
<td>300</td>
<td>130</td>
</tr>
<tr>
<td>CA</td>
<td>200</td>
<td>175</td>
</tr>
<tr>
<td>MA</td>
<td>200</td>
<td>180</td>
</tr>
<tr>
<td>MA</td>
<td>400</td>
<td>195</td>
</tr>
<tr>
<td>PA</td>
<td>100</td>
<td>105</td>
</tr>
<tr>
<td>PA</td>
<td>300</td>
<td>101</td>
</tr>
</tbody>
</table>

```
DATA DS6; SET DS6;
   UI=_N_; RUN;
PROC TRANSPOSE DATA=DS6 OUT=DS6T
   (RENAME=(_NAME_=ATTRIB));
   BY UI NOTSORTED;
   VAR AMT WT; RUN;
```

**OUTPUT**

```
UI ST ATTRIB COL1
1  PA AMT 200
1  PA WT  240
2  PA AMT 100
2  PA WT  220
3  CA AMT 300
3  CA WT  130
4  CA AMT 200
```
The COMPARE Procedure Step With The ID Statement, BY Statement, and The NOTSORTED Option

Proc compare compares two data sets: the base data set and the comparison data set. This procedure determines matching variables and matching rows. Proc compare can produce lengthy output. Use the various options to specify the comparisons to make and the degree of detail in the report. The compare procedure has two (2) NOTSORTED options - on the BY statement and on the ID statement. To use the BY statement with proc compare, both the base and comparison data sets must be sorted on the BY variables. The nature of the comparison depends on whether all BY variables are in the comparison data set and, if so, whether their attributes match those of the BY variables in the base data set.

The ID statement specifies the variables in the base data set that the procedure uses to match rows. These variables are the ID variables. The data set must be sorted by the ID variables or have an appropriate index. ID variables also identify rows on the printed report and in the output data set. The NOTSORTED option on the ID statement specifies that the rows have been grouped in some way, but are not necessarily in sorted order. Use the NOTSORTED option on the ID statement, if you do not need to sort or index the data set by the ID variables. If the NOTSORTED option is specified, proc compare matches the first row in the base data set with the first row in the comparison data set, the second with the second, and so on. Also, if the NOTSORTED option is used and the ID values of corresponding rows are not the same, proc compare prints an error message and stops processing. If the data sets are not sorted or indexed by the common ID variables and you do not specify the NOTSORTED option, proc compare prints a warning message and continues to process the data sets as if the NOTSORTED option had been specified.
In the lengthy COMPARE Procedure output (not displayed) the whole fourth BY group (second ‘PA’ BY group) has not been analyzed, because the ‘Pa’ error stops execution. After fixing the discrepancies run the proc compare again.

**CONCLUSION**

Intermediate level SAS programmers can benefit by knowing the use of BY’s NOTSORTED option in both data steps and in various procedures.

**ACKNOWLEDGMENTS AND CONTACT INFORMATION**

The authors want to thank their families for their support, encouragement and generosity. Karuna thanks Jay Samudral, her husband, and children, Rohini and Rahul. Gregory thanks Elaine Zanutto, his wife, and children, Kevin and Lauren.

The author’s have a more extensive paper on a similar topic posted at http://www.philasug.org/arc.htm#app (PhilaSUG website > Archive of Paper Presentations > Winter 2006).

The content of this paper is the work of the authors and does not necessarily represent the opinions, recommendations, or practices of Octagon Research Solutions or Centocor R&D. This information is provided as a service to readers. This text and code are provided ‘as is’. There are no warrantees, expressed or implied, as to merchantability or fitness for a particular purpose regarding the materials or code contained herein.

SAS and all other SAS Institute Inc. product or service names are registered trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Your comments and questions are appreciated and will be acknowledged. Contact the authors:

Karuna Samudral                        Gregory M. Giddings  
2890 Denbeigh Dr., Hatfield, PA 19440    519 S. Melville St., Philadelphia, PA 19143  
karunasamudral@gmail.com  gmgiddings2@yahoo.com