Some Unexpected Results with PROC SORT and NODUPRECS

presented by David Franklin, Wednesday, May 2, 2007

Introduction

At first sight the NODUPSRECS option (alias NODUP) on the SORT procedure takes out duplicate records for a specified incoming SAS dataset. The SAS Reference uses the following text to describe the option:

> *"checks for and eliminates duplicate observations. If you specify this option, then 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, then the observation is not written to the output data set."*

However this may not always the case to the user and one has to be careful in its usage. The following mini-paper outlines the problem and goes about some ways to get around it.

The Dataset

A printout of the dataset that will be used for this paper is given below:

```
SITE    ITEM                  ABNFLAG
001     General Appearance       N
001     Skin                     Y
001     Neck                     N
001     Breasts                  Y
001     Lungs                    N
002     Skin                     N
002     General Appearance       N
002     Breasts                  N
002     Neck                     N
002     Lungs                    N
```

From this dataset a printout of the unique ITEM/ABNFLG values is requested.

The Solution

The first solution would probably be

```
proc sort data=mh0 (keep=ITEM ABNFLAG)
          out=mh1 NODUPRECS;
    by ITEM;
run;
```

which yields the following result:

Quick Tip

To get the list of SAS modules licensed at Millennium, the following SAS code is useful:

```
proc setinit noalias;
run;
```

Note that the output goes to the SAS Log.

```
ITEM                  ABNFLAG
ITEM                  ABNFLAG
Breasts                  Y
Breasts                  N
General Appearance       N
General Appearance       N
Lungs                    N
Lungs                    N
Neck                     N
Neck                     N
Skin                     Y
Skin                     N
```

This clearly is not a unique list of ITEM/ABNFLG values. Lets see if moving the KEEP option changes anything.

```
proc sort data=mh0 out=mh2 (keep=ITEM ABNFLAG)
          NODUPRECS ;
    by ITEM;
run;
```

results in the following output:

```
ITEM                  ABNFLAG
Breasts                  Y
Breasts                  N
General Appearance       N
General Appearance       N
Lungs                    N
Lungs                    N
Neck                     N
Neck                     N
Skin                     Y
Skin                     N
```

Clearly this does not work. Now lets try one other solution.

```
data mh2;
    set mh0;
    keep ITEM ABNFLAG;
run;
proc sort data=mh2 out=mh3 NODUPRECS ;
    by ITEM;
run;
```

which yields

```
ITEM                  ABNFLAG
Breasts                  Y
Breasts                  N
General Appearance       N
Lungs                    N
Neck                     N
Skin                     Y
Skin                     N
```

This is what is wanted. But is this another way without having to do a separate data step before the SORT procedure call? Not surprisingly SAS does supply one. There is an option called SORTDUP that controls the SORT procedure's application of the NODUP option to physical or logical records and has the following syntax:

SORTDUP=PHYSICAL | LOGICAL

where

PHYSICAL removes duplicates based on all the variables that are present in the data set (default).
LOGICAL removes duplicates based on only the variables remaining after any DROP= and/or KEEP= data set options are processed.

Now changing the SORTDUP option to LOGICAL and using the following code:

```
options SORTDUP=LOGICAL;
run;
proc sort data=mh0 (keep=ITEM ABNFLAG)
          out=mh2 NODUPRECS;
    by ITEM;
run;
```

yields the following output

```
ITEM                  ABNFLAG
Breasts                  Y
Breasts                  N
General Appearance       N
Lungs                    N
Neck                     N
Skin                     Y
Skin                     N
```

SAS does give another solution that is useful, the NODUPKEY option in the SORT procedure:

```
proc sort data=mh0 (keep=ITEM ABNFLAG)
          out=mh2 NODUPKEY;
    by _ALL_;
run;
```

which yields

```
ITEM                  ABNFLAG
Breasts                  N
Breasts                  Y
General Appearance       N
Lungs                    N
Neck                     N
Skin                     N
Skin                     Y
```

Note that this solution grabs the variables the variables that we are interested in using the KEEP statement and uses the special _ALL_ variable to select the unique observations. It has the advantage of no requiring the SORTDUP option to be see a special way.

Conclusion

SAS can produce strange results on your data that are unexpected. Always look at your input and output data and see if the comparison is what is expected.

Using the Input with CARDS Statement

SAS provides several ways for a user to input data using a CARDS or DATALINES statement. In the examples the CARDS statement is used but the two statements for the most part can be used interchangeably. (The CARDS statement originates from the time when punch cards were used to store data.) Note that for each example the following statement was used to define the CITY and DISTANCE variables:

```
length city $15 distance 8;
```

List Input, known for its simplicity

```
input city $ distance;
cards;
Amsterdam 370
Montreal 5200
Auckland 22720
;
```

Delimited Input, using the DLM= option (example uses '~' but it is possible to use other characters as the delimiter)

```
infile cards dlm='~';
input city $ distance;
cards;
Paris~330
New York~5530
Sydney~215660
;
```

"Double Space" as a delimiter

```
input city $ & distance;
cards;
Frankfurt  640
Rio de Janeiro  11060
Singapore  10810
;
```

Named Input

```
input city= $ distance=;
cards;
city=Copenhagen distance=953
distance=6800 city=Nairobi
city=Tokyo distance=15260
;
```

Column Input

```
input city $ 1-11 distance 13-20;
cards;
Rome        1430
Mexico City 10640
Hong Kong   13200
;
```

Formatted Input

```
input city $11. @13 distance 5.;
cards;
Stockholm   1450
Los Angeles 8780
Bangkok     12860
;
```

It is possible to mix the methods inside an INPUT statement but use caution as unexpected results can easily occur.

Quick Tip

Two interesting functions arrived in version 9. Instead of writing

```
if .z<value<30 then flag='N';
else if 30<=value then flag='H';
else if missing(value) then flag='M';
```

it is possible to use the IFC function if the returned value is character and IFN function if the returned value is numeric. Using the IFC function, that has the following syntax

IFC(*logical, value-if-true, value-if-false <,value-if-missing>*);

the code could be written as

```
flag=IFC(value<30,'N','H','M');
```