

A Simple Solution to the PROC SORT NODUP Option Problem

Pete Lund, Washington State Caseload Forecast Council, Olympia, Washington

ABSTRACT

PROC SORT offers a "nodup" option, which is purported to "check for and eliminate duplicate observations. This option causes PROC SORT to compare all variable values for each observation to the previous one written to the output data set." The documentation further states that "if an exact match is found, the observation is not written to the output dataset." It has long been known that these statements are not quite true. The NODUP option only works if the duplicate observations are adjacent to each other in the data vector after the sort is completed. This has caused many a duplicate record to be written out and many a programmer to ponder the unexpected results.

This paper presents a simple, one step approach to the problem. The approach presented does not require sorting the dataset by every variable, which is a solution but can be cumbersome if the dataset contains a large number of variables.

THE PROBLEM

The NODUP option on PROC SORT is purported to eliminate duplicate observations in a dataset as it is sorted. In reality however, those duplicate observations are only deleted if they are adjacent after the sort is completed and SAS is writing the output dataset. Take the following simple example:

```
data temp;
  input a b c;
  cards;
1 2 3
1 1 3
1 2 3
1 2 3
;
run;

proc sort nodup;
  by a;
run;
```

NOTE: 1 duplicate observations were deleted.
NOTE: The data set WORK.TEMP has 3 observations and 3 variables.
As you can see, there are three duplicate observations, two of which should have been deleted. Only one was.

That is because as the records leave the sort procedure (in the same order as written in the data step) the duplicate observations are not contiguous. Each observation is compared with the previous observation as the output file is being written. If it is a duplicate, it is not written out. As can be seen in the above dataset, the third observation is the same as the first. However, it is compared to the second when determining duplicates and is thus kept.

THE SOLUTIONS

SQL – Select Distinct

There are a number of simple alternatives to this problem. The simplest is to use PROC SQL and the DISTINCT keyword.

```
proc sql;
  create table temp2 as
  select distinct *
  from temp
  order by a;
quit;
```

The * tells SQL to select all the variables in the dataset TEMP. The DISTINCT keyword will prevent the writing of any duplicate records. The ORDER BY will sort by the variable(s) listed. Run against the dataset created earlier, the above SQL creates a table (dataset), TEMP2, with the expected 2 rows (observations).

For sorting large datasets PROC SQL...ORDER BY is often slower than PROC SORT. There is another solution which uses PROC SORT and the NODUPKEY option.

SORT – NODUPKEY

The NODUPKEY option is similar to the NODUP option in that duplicate observations are eliminated. However, with the NODUPKEY option the entire record does not have to be duplicated, only the key fields (those listed in the BY statement).

If we were to sort our earlier dataset by ALL the fields in the dataset the duplicate observations would be forced to be adjacent and the NODUPKEY would eliminate the duplicate records. For example,

```
proc sort data=temp out=temp2 nodupkey;
```

```
by a b c;
run;
```

NOTE: 2 observations with duplicate key values were deleted.

NOTE: The data set WORK.TEMP2 has 2 observations and 3 variables.

Now, we've eliminated all the duplicates and the data is sorted in the correct order.

This option is fine for datasets with a small number of observations. What happens if your dataset has 50, 100, 500 variables? Do you want to type in all those variable names? Another solution combines the accuracy of a NODUPKEY with all variables and the simplicity of referencing just the variables you want to sort by.

A Simple Macro

In Appendix A of this paper is a macro that will sort a dataset and remove all duplicate observations.

This little macro has three parameters:

- DSN – the incoming dataset name
- OUTDSN – the outgoing dataset name
- BYVARS – the variables to sort by

The first thing the macro does is to use the SASHELP.VCOLUMN view to fetch the names of all the variables in the incoming dataset. The variable names are just values in this view, so we need to get them in a quoted, comma-separated list. The following macro statement will create a new by-variable list (BYVARS2) simply replacing all the spaces with “,” (quote-comma-quote).

```
%let byvars2 =
%str(%)%upcase(%sysfunc(tranwrd(&byvars,%str(
),%str(","))))%str(%);
```

If BYVARS = idnum status, the above would create BYVARS2 = “IDNUM,”STATUS”.

We can then query SASHELP.VCOLUMN to get all the other variable names from the incoming dataset. This will allow us to build a PROC SORT BY statement with out desired BY variables listed first, followed by all the other variables in the dataset. The following query will create a macro variable, VARS, containing those other variable names.

```
proc sql noprint;
  create table _temp
  like &dsn;
```

```
select name into :vars
  separated by ' '
from sashelp.vcolumn
where libname eq 'WORK' and
  memname eq '_TEMP' and
  name not in ("%byvars2");
```

```
drop table _temp;
quit;
```

This looks more complicated than it really is. The important things to note:

- ✓ Create an empty table _temp with all the variables in the incoming dataset. This way we don't have to worry about whether a one or two-level name was passed when VCOLUMN is queried. It stores the libname and the member name separately.
- ✓ Create a macro variable VARS with the : operator
- ✓ Do not include any variables in the BYVARS2 list
- ✓ BYVARS2 needs to be %unquoted

So, to sort and unduplicate the dataset we had at the beginning the call to the macro would be:

```
%nd_sort(temp,temp,a)
```

Appendix B of this paper shows a more sophisticated version of the same macro containing some error checking and option handling.

Author Contact Information

Both versions are available electronically, via e-mail, from the author:

Pete Lund
 Washington State Forecast Council
 515 E 15th Ave SE
 Olympia, WA 98504-0962
 (360) 902-0086 voice
 (360) 902-0084 FAX
 peter.lund@cfc.wa.gov

Appendix A

```

*-----*
* Macro:      ND_SORT
* Programmer: Pete Lund
* Date:       May 1998
*
* Purpose:    Sorts and removes duplicates from a dataset
*-----*
* Parameters: DSN      - incoming dataset name
*             OUTDSN   - outgoing dataset name
*             BYVARS   - by variables, separated by spaces
*-----*
* Sample Call: to sort the dataset MYDATA by IDNUM, DATE and STATUS and create dataset A.TODAY
*
*             %nd_sort(mydata,a.today,idnum date status)
*-----*

%macro nd_sort(dsn,outdsn,byvars);

  options nonotes; * keep SQL note from displaying in the log;

  *-----*
  * the SQL query is going to treat the by variable names as a
  * list of values - replace the spaces between the names with
  * quotes and commas
  *-----*

  %let byvars2 = %str(%)%upcase(%sysfunc(tranwrd(&byvars,%str( ),%str(","))))%str(%);

  *-----*
  * query the VCOLUMN view for the names of the variables in the
  * incoming dataset - do not include those in the by variable list
  *-----*

  proc sql noprint;
    create table _temp
    like &dsn;

    select name into :vars
      separated by ' '
    from sashelp.vcolumn
    where libname eq 'WORK' and memname eq '_TEMP' and
      name not in ("%&byvars2");

  options notes; * turn notes back on so SORT notes display;

  *-----*
  * sort, with NODUPKEY, by all the variables in the dataset
  * the BY variables, followed by all the other variables
  *-----*

  proc sort data=&dsn out=&outdsn nodupkey;
    by &byvars &vars;
  run;
%mend;

```

Appendix B

```

*-----*
* Macro:      ND_SORT                               *
* Programmer: Pete Lund                             *
* Date:       May 1998                              *
*-----*
* Purpose:    Sorts and removes duplicates from a dataset *
*-----*
* Parameters: DSN      - incoming dataset name       *
*             OUTDSN   - outgoing dataset name      *
*             BYVARS    - by variables, separated by spaces *
*-----*
* Sample Call: to sort the dataset MYDATA by IDNUM, DATE *
*               and STATUS and create dataset A.TODAY *
*               %nd_sort(mydata,a.today,idnum date status) *
*-----*
* Note:       requires macro HEADER - available from: *
*             Pete Lund                               *
*             WA State Caseload Forecast Council     *
*             peter.lund@cfc.wa.gov                  *
*-----*

%macro nd_sort(dsn,outdsn,byvars);

  %local libref dataset options;

  options nonotes;

  %macro TrimLeft(mvar);
    %qsysfunc(trim(%qsysfunc(left(&mvar))))
  %mend;

  %macro CheckDSN(chkdsn,which);

  *-----*
  * split the dsn parameter into libref, dataset name and options *
  *-----*

  %if %index(&chkdsn,.) gt 0 %then
    %do;
      %let libref = %upcase(%scan(&chkdsn,1,.));
      %let dataset = %upcase(%scan(&chkdsn,2));
      %if %index(&chkdsn,%str(%)) gt 0 %then
        %let options = %substr(&chkdsn,%index(&chkdsn,%str(%)));
      %end;
    %else
      %do;
        %let libref = WORK;
        %let dataset = %upcase(%scan(&chkdsn,1));
        %if %index(&chkdsn,%str(%)) gt 0 %then
          %let options = %substr(&chkdsn,%index(&chkdsn,%str(%)));
        %end;
      %end;

  *-----*
  * before starting, check the following: *
  * - was a valid libname passed *
  * - was a valid dataset passed *
  *-----*

  proc sql noprint;
    select libname
    from sashelp.vslib
    where libname eq "&libref";
    %if %sqllobs eq 0 %then

```

Coder's Corner

```

%do;
  %Header(/C,ND_SORT ERROR!,,/D,/L,Libname &libref is not assigned);
  %let error = 1;
%end;

select memname
from sashelp.vtable
where libname eq "&libref" and memname eq "&dataset";
%if &sqlobs eq 0 and &which eq IN %then
  %do;
    %Header(/C,ND_SORT ERROR!,,/D,/L,Dataset &dataset does not exist);
    %let error = 2;
  %end;
quit;
%mend;

%let error = 0;
%CheckDSN(&dsn,IN);
%if &error ne 0 %then %goto quit;

*-----*;
* if no outgoing dataset name was passed, set it to the *
* incoming dataset - if there was an outgoing dataset, *
* check to make sure everything is valid *
*-----*;

%if %quote(&outdsn) eq %then %let outdsn = &dsn;
%else
  %do;
    %CheckDSN(&outdsn,OUT);
    %if &error = 1 %then %goto quit;
  %end;

*-----*;
* the SQL query is going to treat the by variable names as a *
* list of values - replace the spaces between the names with *
* quotes and commas *
*-----*;

%let byvars2 = %upcase(%sysfunc(tranwrd(&byvars,%str( ),%str(","))));

*-----*;
* query the VCOLUMN view for the names of the variables in the *
* incoming dataset - do not include those in the by variable list *
*-----*;

proc sql noprint;
  create table _temp
  like &dsn;

  select name into :vars
  separated by ' '
  from sashelp.vcolumn
  where libname eq 'WORK' and memname eq '_TEMP' and
  name not in ("%byvars2");

  drop table _temp;
quit;

options notes;

*-----*;
* sort the dataset *
*-----*;

proc sort data=&dsn out=&outdsn nodupkey;
  by &byvars &vars;
run;

%quit;
options notes;
%mend;

```