Paper 221-25

# The Problem With NODUPLICATES

Jack Hamilton, First Health, West Sacramento, California USA


## ABSTRACT

The NODUPLICATES option of PROC SORT does not delete duplicate records from a SAS dataset under certain conditions. This paper explains why, and suggests workarounds.

This paper applies to SAS® System software version 6.12.


## UPDATES

I did not have access to Version 8 when I wrote this paper. After V8 becomes available, I will revise this paper as needed. I will also incorporate comments received at SUGI. The new version will be available on my web page at the address shown below. You will be able to register for notification of future updates.

## DELETING DUPLICATES

It is often useful in SAS programming to delete duplicate records from a data set. PROC SORT has an option which seems designed to handle this problem, NODUPLICATES.

### THE NODUPLICATES OPTION

According to the SAS Procedures Guide, Version 6, PROC SORT with the NODUPLICATES option "checks for and eliminates duplicate observations".

## THE PROBLEM WITH NODUPLICATES

So NODUPLICATES sounds like exactly what you need. Unfortunately, there's some fine print. A bit later, the SAS Procedures Guide says:

> This option causes PROC SORT to compare all variable values for each observation to the previous one written to the output data set. If an exact match is found, the observation is not written to the output data set."

In other words, the documentation doesn't say that duplicates will be deleted – it says that *adjacent* duplicates will be deleted. Might you ever have nonadjacent duplicates? Yes, as an example will show.

### NON-ADJACENT DUPLICATES – EXAMPLE 1

Consider the following example:

```
data test;
   input A B $ @@;
cards;
1 A 2 B 2 A 3 C 3 X
4 D 1 A 3 C 3 C 5 E
run;

proc sort data=test out=nodups1 nodups;
   by b;
run;
```

The resulting data set will look like this:

| OBS | A | B |
|---|---|---|
| 1 | 1 | A |
| 2 | 2 | A |
| 3 | 1 | A |
| 4 | 2 | B |
| 5 | 3 | C |
| 6 | 4 | D |
| 7 | 5 | E |
| 8 | 3 | X |

Observation 3 is a duplicate of observation 1. Obs 3 was written because it wasn't the same as the previous record written, obs 2. When it starts to write obs 3, PROC SORT doesn't know or care about obs 1, and so doesn't know that obs 3 is a duplicate and shouldn't be written to the output data set.

### NON-ADJACENT DUPLICATES – EXAMPLE 2

Non-adjacent duplicates can also occur even when you do sort on all the variables in the data set – or rather, when it appears that you are sorting on all the variables. This can happen when you use a KEEP= or DROP= dataset option in PROC SORT.

Consider the following code, which uses the input data set created in the previous example:

```
proc sort noduplicates
     data=test (keep=b)
     out=nodups2;
  by b;
run;
```

The resulting data set:

| OBS | B |
|---|---|
| 1 | A |
| 2 | A |
| 3 | A |
| 4 | B |
| 5 | C |
| 6 | D |
| 7 | E |
| 8 | X |

Again, duplicates have not been deleted. Why? Because of a bug in PROC SORT in version 6 – variables are not dropped until output, even if the KEEP or DROP option is coded on the input data set. The duplicate code logic looks at observations before the variables are dropped, so it doesn't detect the duplicates. This problem is reported to have been fixed in version 8.

## EASY WORKAROUNDS

There are two easy workarounds: sort by all the variables, or use PROC SQL. A third method, a double sort, can be used to delete duplicates caused by using a KEEP or DROP data set option.

### SORT BY ALL THE VARIABLES

If you put every variable in the BY list, the problem probably won't occur, because duplicate observations will be adjacent when PROC SORT checks for duplicates.

An easy way to sort by every variable is to put the special keyword _ALL_ at the end of the BY statement:

```
proc sort data=test out=nodups3 nodups;
   by b _all_;
run;
```

Note that this technique doesn't work if you have coded a KEEP or DROP option. It appears that the _ALL_ list refers to the variables in the output data set rather than those in the input data set.

**USE PROC SQL**

Using the DISTINCT option in PROC SQL will also eliminate the duplicates, probably:

```
proc sql;
   create table nodups4 as
      select  distinct *
      from    test
      order   by b;
quit;
```

**SORT TWICE**

You can eliminate duplicates caused by using KEEP= or DROP= if you sort the data twice; on the second sort, use the FORCE option:

```
proc sort noduplicates
     data=test (keep=b)
     out=nodups2;
  by b;
run;

proc sort noduplicates force
     data=nodups2
     out=nodups2;
  by b;
run;
```

**PROBABLY?**

You might have noticed those "probably"s in my description of the workarounds. In a perfect world, they would work. But they don't, or at least they're not guaranteed to. There are still some bugs in SAS Software.

Perhaps I shouldn't call them bugs, because they are (or were) documented in the Usage Notes, but there are some counter-intuitive features in the way sorting and duplicate deletion are handled in both PROC SORT and PROC SQL. I won't describe them here, and in fact I haven't been able to reproduce them myself (most are version or platform specific), but if the Usage Notes say that the two solutions above won't always work, I'll believe that they won't always work.

## A MORE DIFFICULT WORKAROUND

You can use FIRST./LAST. logic in a data step to delete duplicate records.

**FIRST./LAST.LOGIC**

Using FIRST. or LAST. process is more work to program, but if coded properly always works, or at least gives you a nice clean abend if it doesn't:

```
proc sort data=test out=nodups5
     noduplicates;
   by b a;

data nodups6;
   set nodups6;
      by b a;
   if first.a;
run;
```

Unfortunately, you have to list all the variables in the second BY statement; you can't use _ALL_ as you can with PROC SORT. (See my paper *Some Utility Applications Of The Dictionary Tables in PROC SQL* for one method of producing a variable list programatically.)

I specified NODUPLICATES in the PROC SORT because it might delete some duplicates even if it doesn't delete them all, thus reducing the amount of work done by the data step.

## CONCLUSION

The NODUPLICATES option of PROC SORT must be used carefully. I won't say that you should never use it, but you should be aware of the possible problems and ensure that they don't apply to your usage.

## REFERENCES


Hamilton, Jack; *Some Utility Applications Of The Dictionary Tables in PROC SQL*; http://www.qsl.net/kd6ttl/sas/sqlutilov.pdf

SAS Institute Inc.(1990) *SAS Language: Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

-, *Using the KEEP= data set option with PROC SORT*, SAS Usage Note V6-SORT-9102, http://www.sas.com/service/techsup/unotes/V6/9/9102.html

-, *Using the SAS sort with NODUPLICATES/NODUPKEY and EQUALS*, SAS Usage Note V6-SORT-B272, http://www.sas.com/service/techsup/unotes/V6/B/B272.html


## TRADEMARK CITATION



## CONTACT INFORMATION


Jack Hamilton
First Health
750 Riverpoint Drive
West Sacramento, California 95605
JackHamilton@FirstHealth.com

http://www.qsl.net/kd6ttl/sas/sas.htm


Selected papers are also available at:

http://www.sashelp.com/Articles/ViewPapers.asp