The Problem With NODUPLICATES, Continued

Jack Hamilton

First Health

West Sacramento, California

JackHamilton@FirstHealth.com

What Should NODUPLICATES Do?

PROC SORT has two options to control whether "duplicate" output records will be written:

- **NODUPLICATES** "checks for and eliminates duplicate observations".
- **NODUPKEY** "checks for and eliminates observations with duplicate BY values. This option differs from NODUPLICATES because the NODUPKEY option compares only the BY values, not the entire observation."
-
    - Quoted from the SAS Procedures Guide, Version 6

What's The Problem?

The problem is that NODUP doesn't always eliminate all duplicates. There are two reasons:

- It does what it’s documented to do, not what you might expect it to do. The documentation is perhaps more subtle than necessary.
- There are bugs. They are not all fixed in the current version (6.12). At least some of the bugs are gone in Version 8.

What We Expect It To Do

From the name, and the short description, you might expect that a dataset run through PROC SORT NODUP would come out with no identical records.

What It Actually Does

The problem is in the fine print, which 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.”

So, Why Aren’t Dups Found?

The problem is that duplicate records might not end up next to each other during output, so the deletion won’t happen.

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 nodup;
 by b;
run;
proc print data=test;
run;
```

The Results

```
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
```

Observations 1 and 3 are duplicates.   Obs 3 was written because it wasn’t the same as the previous record written (obs 2). PROC SORT doesn’t know or care about observation 1 at this point. Work through this example with pencil and paper, and you’ll see why it happens.

Two Workarounds

There are two easy workarounds: sorting by all the variables, or using PROC SQL.

Sorting by all the variables

If you put every variable in the BY list, the problem probably won't occur, because duplicate observations will necessarily be together on output.   An easy way to sort by every variable is to put **`_ALL_`** at the end:

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

Using PROC SQL

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

```
proc sql;
   create table nodups2 as
      select  distinct *
      from    test
      order   by b;
quit;
```

But There Are Bugs

You might have noticed those “probably”s in my description of the first two workarounds. In a perfect world, they would work. But they don’t, or at least not every time.

Perhaps I shouldn’t call them bugs, because they’re documented in the Usage Notes, but there are some counter-intuitive “features” in the way sorting is handled in both PROC SORT and PROC SQL. I won’t describe them all here, and I haven’t been able to reproduce them all myself, but if the Usage Notes say the two solutions above won’t always work, I’ll believe that they won’t.

Using FIRST./LAST. Logic

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

```
proc sort data=test out=test2;
 by b a;
run;
data nodups2;
   set test2;
     by b a;
   if first.a;
run;
```

Unfortunately, you have to list all the variables; you can’t use _ALL_ here (but see my "SQL Utilities" paper for ideas on how to create that list).

So, What To Do?

I have decided for myself that I will not use the NODUP option. If you study all the Usage Notes, and never change to a new release, and always run on the same hardware, you can figure out when it's safe. Those are not conditions I'm willing to live with. It's easy to use alternative code, so that's what I do.

What About NODUPKEYS?

The Usage Notes indicate that problems might occur with NODUPKEYS as well. It apparently doesn't happen as often, and I've never heard of a case of it happening, but why take the risk? Again, the logic is easy to code in a data step.