The Problem with NODUPS

Jack Hamilton

First Health
West Sacramento, California

Abstract

The NODUPLICATES option in the SORT procedure supposedly deletes duplicate records from a SAS® data set. In reality, its actions are somewhat unpredictable, due to an oddity in PROC SORT. This oddity can also lead to vastly inflated processing time in any PROC SORT that uses a DROP or KEEP statement. This paper presents examples and suggests some alternatives.

Keywords

Efficiency, NODUPKEYS, NODUPLICATES, PROC SORT, SORT Procedure, SQL

Overview of the NODUPLICATES Option

According to the *SAS Procedures Guide*, the NODUPLICATES option

> checks for and eliminates 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. If an exact match is found, the observation is not written to the output data set.

Why You Might Delete Duplicates

The most common usage of the NODUPLICATES option is to prepare a data set for use in a subsequent MERGE statement (or equivalent).

Here's an example. Suppose you have a TRAINS data set containing a record for each train running between Sacramento and San Francisco, and a PASSNGRS data set containing a record for each passenger on one of those trains. The datasets look like this:

TRAINS

TRAIN	DATE	NAME
5	01Jun1997	California Zephyr
5	02Jun1997	California Zephyr
11	01Jun1997	Coast Starlight
11	02Jun1997	Coast Starlight
721	01Jun1997	Capitol
721	02Jun1997	Capitol

PASSNGRS

TRAIN	DATE	FARE
5	02Jun1997	$13.00
11	01Jun1997	$12.00
721	01Jun1997	$9.00
721	01Jun1997	$10.00

Suppose what you want is a data set containing the information in PASSENGRS, with the addition of the train name from TRAINS:

PASS2

TRAIN	NAME	DATE	FARE
5	California Zephyr	02Jun1997	$13.00
11	Coast Starlight	01Jun1997	$12.00
721	Capitol	01Jun1997	$9.00
721	Capitol	01Jun1997	$10.00

One obvious way to obtain this result is to sort the datasets and do a merge:

```
data pass2;
  merge trains   (in=intrains)
        passngrs (in=inpass);
   by train;

  if (intrains and inpass);

run;
```

Unfortunately, this produces a message in the log:

```
NOTE: MERGE statement has more than
one
      data set with repeats of BY
      values.
NOTE: The data set WORK.PASS2 has 6
      observations and 4 variables.
```

and incorrect results:

TRAIN	NAME	DATE	FARE
5	California Zephyr	02JUN19	13
5	California Zephyr	02JUN19	13
11	Coast Starlight	01JUN19	12
11	Coast Starlight	02JUN19	12
721	Capitol	01JUN19	9
721	Capitol	01JUN19	10

Why? Well, that's just the way SAS software works. To make this merge work successfully, you must first delete the duplicate records in TRAINS, so that there is only one record per train number.

Here Comes Trouble

To eliminate those duplicates, you could use PROC SORT with the NODUPLICATES option:

```
 proc sort data=trains
           out=trains3
           noduplicates;
  by train;
 run;
```

This produces a message saying that no duplicates were deleted:

```
 NOTE: 0 duplicate observations were
       deleted.
 NOTE: The data set WORK.TRAINS3 has 6
       observations and 3 variables.
```

When you look at the output dataset you see why:

TRAIN	DATE	NAME
5	01JUN19	California Zephyr
5	02JUN19	California Zephyr
1	01JUN19	Coast Starlight
1	02JUN19	Coast Starlight
721	02JUN19	Capitol
721	01JUN19	Capitol

The different dates make the records non-unique. So let's DROP the DATE variable in the sort:

```
 proc sort data=trains (drop=date)
           out=trains4
           noduplicates;
  by train;
 run;
```

But look at the resulting message and dataset:

```
 NOTE: 0 duplicate observations were
       deleted.
 NOTE: The data set WORK.TRAINS4 has 6
       observations and 2 variables.
```

The log says that no observations were deleted, even though there are clearly duplicates in the output dataset:

TRAIN	NAME
5	California Zephyr
5	California Zephyr
11	Coast Starlight
11	Coast Starlight
721	Capitol
721	Capitol

What happened?

NODUPLICATES Doesn't Always Work

The problem, it turns out, is that there is a bug in PROC SORT. If you use a DROP or KEEP option, dropped variables aren't dropped until the new data set is written, even when DROP or KEEP is used as a data set option on the input dataset, so duplicates appear in the output.

Although it seems to produce wrong results only when NODUPLICATES or NODUPKEYS is used, this problem also results in more processor usage than is strictly necessary, as the example later in this paper will show. The KEEP/DROP problem is described in SAS Usage Note V6-SORT-9102.

I have not had an opportunity to test the behavior of PROC SORT under version 7 or 8 of SAS software.

Never Use NODUPLICATES

After discovering problems similar to the one above, and finding a whole host of bugs related to PROC SORT in the SAS Usage Notes, I decided that I simply couldn't trust PROC SORT with the NODUPLICATES option. There are many cases in which they appear to work correctly, but I don't want remember which cases do and which cases don't.

I think I'm better off with the blanket rule **Never Use NODUPLICATES**. I will sometimes do unnecessary work, but I won't have to worry that I'm getting bad results without knowing it (we first discovered this problem when a complicated SQL query wasn't returning the expected results. SQL allows merges with duplicate keys in multiple input data sets, so we didn't get the warning shown in the first merge statement above. It took hours of digging to find the problem).

Why I Don't Use NODUPKEYS

The NODUPKEYS option is similar to NODUPLICATES, but eliminates duplicates based only on the BY variables, not the entire record. I don't use NODUPKEYS because of the randomness it introduces into the output. If you have several keys with the same BY values, PROC SORT will pick one, and in general you don't know which (it depends on the sort algorithm used, which in turn depends on the number of observations in the dataset, the options you set for SORT system options, and various other things). My opinion: if you're carrying other variables, you ought to care what their values are, so you don't use NODUPKEYS. If you don't care what the values are, you should just drop them, so you don't need NODUPKEYS.

What To Do Instead

There are at least three easy workarounds:

Use A SORT Followed by a Data Step

This is the easiest solution. Do a regular sort, followed by a data step to delete the duplicates:

```
proc sort data=trains (drop=date)
          out=trains5;
 by train;
run;

NOTE: The data set WORK.TRAINS5 has 6
      observations and 2 variables.

data trains5;
 set trains5;
  by train;
 if first.train;
run;

NOTE: The data set WORK.TRAINS5 has 3
      observations and 2 variables.
```

Use A Data Step View, PROC SORT, and A Data Step

I said above that PROC SORT doesn't drop variables until output. This means that all variables in the input dataset will be lugged around during the sort, even when they won't be used. This can result in much longer execution times than necessary. One way around this is to use a VIEW on the input data set.

```
data trains6 / view=trains6;
 set trains (keep=train name);
run;

NOTE: DATA STEP view saved on file
      WORK.TRAINS6.


proc sort data=trains6
          out=trains7;
 by train;
run;

NOTE: The data set WORK.TRAINS7 has 6
      observations and 2 variables.

data trains7;
 set trains7;
  by train;
 if first.train;
run;

NOTE: The data set WORK.TRAINS7 has 3
      observations and 2 variables.
```

Use PROC SQL

A SELECT statement in PROC SQL will also delete duplicates:

```
proc sql;
 create table trains8 as
  select distinct train, name
  from   trains
  order  by train;
NOTE: Table WORK.TRAINS8 created,
      with 3 rows and 2 columns.
```

The SAS usage notes say that there may be problems with this solution under some circumstances, so you might want to stick to the SORT/data step solution if you want to be absolutely sure. I have never encountered one of these problems myself. See SAS Usage Notes V6-SQL-E428, V6-SQL-D929, V6-SQL-6925, and V6-SQL-4953.

Some Sample Timings

I created three sample datasets with varying numbers of observations and variables and ran them through PROC sort with NODUPLICATES, plus the three alternatives, to get timings. This example doesn't tickle the duplicates bug, but it does show that PROC SORT is doing something funny.

Elapsed in mm:ss.hh	**Small**	**Medium**	**Large**
NODUPS	.33	.55	18:55.57
SET	1.05	1.63	19:18.54
SET/VIEW	2.34	3.88	3:52.70
SQL	.44	1.19	1:55.95

Results are for SAS for Windows®.
Similar results were obtained under OpenVMS™

SAS Versions

The problem was originally discovered in SAS 6.09 running on a DEC Alpha under OpenVMS . The examples in this paper were creating using SAS 6.10 under Microsoft Windows 3.1 running on a 60 MHz Pentium Processor® with 16MB of RAM.

References

SAS Institute Inc (1990), SAS Procedures Guide, Version 6, Third Edition, Cary, NC, SAS Institute. Inc.

Contact Information

Jack Hamilton
METRICS Department
First Health
750 Riverpoint Drive
West Sacramento, California 95605 USA

1 (916) 374-3833
JackHamilton@FirstHealth.com

Trademarks

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Microsoft and Windows are registered trademarks of Microsoft Corporation. DEC and OpenVMS are trademarks of Digital Equipment Corporation. Pentium is a registered trademark of Intel Corporation.