CC28

Dup, Dedup, DUPOUT

Heidi Markovitz, Board of Governors of the Federal Reserve System

ABSTRACT

The easy to use DUPOUT option of PROC SORT saves duplicate records removed from a dataset during the sort process. This paper discusses the art of identifying and removing duplicates from data. It also demonstrates related SORT options and alternate methods of de-duping.

INTRODUCTION

There are many reasons for eliminating duplicate records from data repositories.

- Only one copy of each type is needed, as in compiling a list of customers from a list of sales.
- There should be only one of each type, as in a deck of playing cards.
- The data is unfamiliar. Its exploration requires searching for duplicates.

This paper shows how to identify and eliminate duplicate data with the DUPOUT feature, introduced in version 9, and other SAS® tools using simulated examples from the Federal Reserve Board.

THE SCENE

At the Federal Reserve Board of Governors ("the FED") we collect data on deposits, reserves, and other financial measures from all depository institutions (banks, credit unions, savings and loans, etc.) operating in the USA. Data arrive in a variety of forms and frequencies from the 12 Federal Reserve district banks, other agencies, and the depository institutions ("DIs") themselves.

We manage these flows to assure that we capture all the required pieces from each DI, that we don't miss any DI reports, and that figures from different sources regarding the same DI are reconciled with each other. This traffic is further complicated when banks merge and reorganize to create new entities and eliminate old ones.

In my division, Monetary Affairs, we pride ourselves on having maintained accurate weekly records of monetary aggregates for over 50 years. In part, the high quality of our product is due to our vigilance in avoiding data duplication which would lead to miscounting.

dedup_demo Data Set

Obs	bank_id	report_ date	deposits	di_city	comment
1	1	20061002	5670	Kansas City	
2	1	20061009	7500	Kansas City	Merged, new name
3	2	20061009	305	York Hills	
4	1	20060925	5672	Kansas City	
5	3	20060930	28	Arapaho	
6	2	20060925	310	Alameda	
7	3	20061009	29	Arapaho	
8	3	20061002	39	Arapaho	Change frequency
9	3	20061002	30	Arapaho	Same date, diff value
10	2	20061002	305	York Hills	Moved location

Figure 1-Data Used in Examples

THE DATA

The data used in the examples is a mock list of deposits held by various institutions on each of a range of dates. There should be one observation for each DI doing business on each reporting date. DIs are permanently identified by their bank_id.

DUPS OK, KEEP JUST ONE – EXAMPLE 1

Even when a data set is expected to have more than one of each type of record, de-duping may be useful to produce a list of existing types. In the sample data set, each DI has a record for every date it reported. One way to obtain a list of all DIs that have ever reported their deposits is to run PROC SORT with the NODUPKEY option. In the example below, the original 10 deposits records are reduced to a list of 3 institutions.

Code:

```
PROC SORT DATA=dedupe_demo(KEEP=bank_id) NODUPKEY see note a
   OUT=nodup_by_id;    see note b
   BY bank_id;     see note c
RUN;
```

Log:

```
NOTE: There were 10 observations read from the data set WORK.DEDUPE_DEMO.
NOTE: 7 observations with duplicate key values were deleted.
NOTE: The data set WORK.NODUP_BY_ID has 3 observations and 1 variables.
```

PROC PRINT Output:

List of Unduplicated IDs, Common NODUPKEY Use

Obs	bank_id
1	1
2	2
3	3

Notes:

a. Since the only significant variable for this purpose is the depository institution ID, the KEEP= data set option is used on the input data set. It assures that the specified variable is the only one processed. For very large data sets, this option speeds the SORT and saves storage by reducing the size of the work files and the OUT data set.

b. The OUT= option of PROC SORT stores the sorted de-duped data in a new data set called nodup_by_id. Because the purpose of this step was to extract a subset of the data without disturbing the original file, the procedure creates a new data set. It does NOT replace the input data set.

c. Producing a simple sorted list of participants from a long list of transactions is very easy. This approach may also be used with multiple BY variables. For instance, a list of DIs and cities would be created by replacing the previous BY statement and adding di_city to the KEEP= option like this:

```
PROC SORT DATA=dedupe_demo(KEEP=bank_id di_city) NODUPKEY
   OUT=nodup_by_id;
   BY bank_id di_city;
```

That would yield a list of 4 depository institutions, since bank_id 2 was in two locations.

KILL THE DUPS – EXAMPLE 2

In other situations, the fact that there are duplicate records is considered a problem. While the sample data set should have multiple records per DI (identified by its bank_id), it should contain only one record for each bank_id/date combination. To assure that there is one and only one record for each DI on each date, sort the data set with the NODUPKEY option.

Code:

```
PROC SORT DATA=dedupe_demo NODUPKEY see note a
   OUT=nodup_by_id_date;
   BY bank_id report_date;
RUN;
```

Log:

```
NOTE: There were 10 observations read from the data set WORK.DEDUPE_DEMO.
NOTE: 1 observations with duplicate key values were deleted.
NOTE: The data set WORK.NODUP_BY_ID_DATE has 9 observations and 5 variables.
```

PROC PRINT Output:

nodup_by_id_date

Obs	bank_id	report_date	deposits	di_city	comment
1	1	20060925	5672	Kansas City	
2	1	20061002	5670	Kansas City	
3	1	20061009	7500	Kansas City	Merged, new name
4	2	20060925	310	Alameda	
5	2	20061002	305	York Hills	Moved location
6	2	20061009	305	York Hills	
7	3	20060930	28	Arapaho	
8	3	20061002	39	Arapaho	Change frequency
9	3	20061009	29	Arapaho	

Notes:

a. This time all the data in each record is needed, so there is no KEEP= option on either the input or output data set.

The log shows that there was in fact an extra record in the file, but the NODUPKEY got rid of it, so that's OK. Or is it? If the file should have only one record for each DI and date, how would we know that the correct one had been retained? Comparison of this tiny list with the input list in Figure 1 shows that observation 9 (bank_id=3, report_date=20061002) from Figure 1 is the deleted one. But visual comparison is not feasible in a production-size

file. We could not tell which record was eliminated or whether the whole record was duplicated or if two different reports were made for the same date.

A list of deleted records would be helpful. The new DUPOUT option on the SORT statement provides this without adding program steps.

SHOW THE DUPLICATES – EXAMPLE 3

DUPOUT can be used with the NODUPKEY or NODUPREC option, to name a SAS data set that will contain the duplicate records eliminated from the main output data set.

Code:

```
PROC SORT DATA=dedupe_demo DUPOUT=dup_id_date     see Note a
   NODUPKEY   see Note b
   OUT=nodup_by_id_date;
   BY bank_id report_date;
RUN;
```

Log:

```
NOTE: There were 10 observations read from the data set WORK.DEDUPE_DEMO.
NOTE: 1 observations with duplicate key values were deleted.
NOTE: The data set WORK.DUP_ID_DATE has 1 observations and 5 variables.
NOTE: The data set WORK.NODUP_BY_ID_DATE has 9 observations and 5 variables.
```

PROC PRINT DUPOUT= Data Set Output:

Duplicate Date, Id Records Eliminated With DUPOUT

Obs	bank_id	report_ date	deposits	di_city	comment
1	3	20061002	30	Arapaho	Same date, diff value

PROC PRINT OUT= Data Set Output:

nodup_by_id_date

Obs	bank_id	report_ date	deposits	di_city	comment
1	1	20060925	5672	Kansas City	
2	1	20061002	5670	Kansas City	
3	1	20061009	7500	Kansas City	Merged, new name
4	2	20060925	310	Alameda	
5	2	20061002	305	York Hills	Moved location
6	2	20061009	305	York Hills	
7	3	20060930	28	Arapaho	
8	3	20061002	39	Arapaho	Change frequency
9	3	20061009	29	Arapaho	

Notes:

a. Syntax for the DUPOUT option in a PROC SORT statement is:
PROC SORT *other options* DUPOUT=SAS-data-set *other options*;

b. The DUPOUT option is effective only when used with the NODUPKEY or NODUPREC options. Without one of these options, the log will show a WARNING message and the DUPOUT data set will be created with 0 records.

The DUPOUT data set listing shows there were two transactions submitted for bank_id number 3 on October 2, 2006. Comparing it with the list for the primary OUT= data set shows that the “duplicate” records actually reported different amounts. That suggests further research into the data flows from that DI to determine why two conflicting reports were submitted.

A SQL WAY TO SHOW THE DUPS – EXAMPLE 4

When duplicates are not expected in a production data set, the NODUPKEY and DUPOUT options may be added to PROC SORT as a defensive mechanism. If the DUPOUT data set is not empty, as in the previous example, a reasonable next step is to examine all the duplicate records, including those not deleted. This would be followed by procedural analysis, such as examining data flow from the problem DI. Here is a SQL way to list all the duplicates.

Code:

```
TITLE1 "List All Transactions with Duplicate ID/Date";
PROC SQL;
   SELECT dedupe_demo.*    see note a
     FROM work.dedupe_demo
    GROUP BY bank_id, report_date   see note b
   HAVING COUNt(*) > 1
    ORDER BY bank_id, report_date;
QUIT;
```

PROC SQL html output, list of Duplicates see note c

List All Transactions with Duplicate ID/Date

bank_id	report_date	deposits	di_city	comment
3	20061002	39	Arapaho	Change frequency
3	20061002	30	Arapaho	Same date, diff value

Notes:

a. "dedupe_demo.*" selects all the variables in the data set called dedupe_demo.

b. GROUP…HAVING clauses evaluate groups of records to determine which to select. In this case, PROC SQL looks at all records in dedupe_demo with matching bank_id and date, and counts them. If there is more than one record for a particular bank_id/date, it outputs all of them.

c. This output was produced in html format because of the SAS DISPLAY MANAGER settings. In recent SAS releases, html is the default mode for all report output under MS/Windows and Unix (not sure about other platforms). To change the default, under MS/Windows, select Tools from the top menu. Click on Options -> Preferences ->Results, and choose your preferred output mode. Alternately, output styles can be controlled by Output Delivery System (ODS) program statements.

PERFORMANCE NOTE

This entire exercise - producing a sorted de-duped file and listing the duplicates - could be performed with PROC SQL. However PROC SORT is generally much faster than PROC SQL when processing a whole data set. Also, if the input data set is not expected to contain duplicates, the PROC SORT with NODUPKEY and DUPOUT provides needed protection against bad duplicate data without adding much overhead when there are no duplicates.

DATA STEP DE-DUPING – EXAMPLE 5

If the data is unfamiliar, identifying duplicates may be only one of the exploratory activities to be performed. A DATA step in which several data tests can be done may be the best way to segregate duplicate records. In this example, the data is sorted without de-duping, then de-duped in a DATA step.

Code in Log:

```
     PROC SORT DATA=dedupe_demo /*no de-duping options*/
        OUT= by_id_date_wo_nodup;
        BY bank_id report_date;
     RUN;

     NOTE: There were 10 observations read from the data set WORK.DEDUPE_DEMO.
     NOTE: The data set WORK.BY_ID_DATE_WO_NODUP has 10 observations and 5 variables.

     DATA dedup_id_date
        dup_id_date;   /*2 output data sets, duplicates and de-duped records*/
        SET by_id_date_wo_nodup END=endata;
        BY bank_id report_date;
        IF endata  THEN DO;
           IF dup_count > 0
              THEN CALL SYMPUT('anyerr', 'Y');    see note a
           PUT dup_count ' bank_id/report date combinations appeared on multiple deposit
     records.';
        END;
     ---other evaluation code---
        IF NOT FIRST.report_date OR NOT LAST.report_date  THEN DO;  see note b
           OUTPUT dup_id_date;
           IF FIRST.report_date
              THEN dup_count + 1;
        END;
        IF FIRST.report_date
           THEN OUTPUT dedup_id_date;    see note c
        DROP dup_count;
     RUN;

     NOTE: There were 10 observations read from the data set WORK.BY_ID_DATE_WO_NODUP.
     NOTE: The data set WORK.DEDUP_ID_DATE has 9 observations and 5 variables.
     NOTE: The data set WORK.DUP_ID_DATE has 2 observations and 5 variables.
```

Notes:

a. This data should not contain duplicate records. If it does the DATA step sets MACRO variable anyerr=Y. By putting an error flag in a MACRO variable, later steps in the program can check the value to decide whether to proceed normally or respond to an error.

b. By sorting and reading the file with the same BY statement, the program can determine whether the current record is the first or the last of its type in the file. In this DATA step, that means the first or last record with this id/date combination. If a record has a unique id/date combo, it will be both the first and last of its type. If not, the program will increment the duplicate counter and put the record into the duplicates data set. This duplicates file will be the same as the one produced by the SQL step in Example 4.
c. The first record with every id/date combination is output to the de-duped file, which will be identical with the file produced by the PROC SORT with NODUPKEY option in Example 2.

This approach requires the input data to be read twice, once to sort and once to de-dupe and evaluate. It creates three data sets: the sorted original, the de-duped data set, and the file containing all the duplicates. A DATA step may be the most flexible way to do data testing when data is expected to be unexpected or illegal duplicates are likely to appear or other validation tests must be done, because one data step can contain many tests.

CONCLUSION

Data may contain unwanted duplicate elements. The DUPOUT option of PROC SORT provides a simple way to trap duplicate records deleted by the NODUPKEY option. The best method to use for de-duping will always be a matter of judgment based on the actual situation.

BIBLIOGRAPHY

"SAS Help and Documentation" delivered with SAS for Windows
See the results of the Federal Reserve Board's use of SAS software at http://www.federalreserve.gov/econresdata/statisticsdata.htm. Production of the H.6 Money Stock and H.3 Reserve Balances releases inspired this paper.

ACKNOWLEDGMENTS

Thanks to Bruce Gilsen at the FED and Carla Mast who helped with the original paper. Thanks too, to Don Henderson and Howard Schreier who improved some of my examples with post-talk feedback.

CONTACT INFORMATION

Heidi Markovitz
Board of Governors of the Federal Reserve System
20th & C Street NW
Washington, DC 20551
(C) 305-803-8407
Email: Heidi.Markovitz@frb.gov