An Easy Way to Split a SAS® Data Set into Unique and Non-Unique Row Subsets

Thomas E. Billings, MUFG Union Bank, N.A., San Francisco, California

ABSTRACT

The Base SAS® SORT procedure options NOUNIQUEKEY and UNIQUEOUT= (new as-of SAS 9.3+) provide an easy way to sort and split a SAS data set into 2 derivative subsets: one that contains only the rows that are unique per the BY variables (unique in the context of the entire input file), and another that contains all rows from the input file that have multiple instances of the BY variables. We show how these options are easier than equivalent legacy logic (a plain PROC SORT paired with a DATA step) and how this method produces results that are different from the split provided by the older NODUPKEY and DUPOUT= options. We also show how PROC SQL can be used to accomplish the same processing, and conclude that the new options for PROC SORT are an efficient way to split a data set based on unique/non-unique sets of BY variables. User level: beginner+. SAS products: Base SAS. Keywords: SORT, SQL, unduplication.

INTRODUCTION

A common requirement in programming is to split a SAS® data set into 2 component subsets, based on the cardinality of rows that is defined in reference to a set of variables – in this context, the BY variables specified in an invocation of the Base SAS SORT procedure. This procedure provides multiple, different options to accomplish this task.

The NODUPKEY and DUPOUT= options of PROC SORT have been supported in SAS for a very long time, whereas the NOUNIQUEKEY and UNIQUEOUT= options are newer, and have been production status starting in SAS 9.3+. These latter two (newer) options are not as well-known as NODUPKEY, and in this paper we highlight the newer options and discuss how they differ from PROC SORT NODUPKEY and PROC SQL SELECT DISTINCT. We also show how the same result can be accomplished using PROC SORT with an associated DATA step and first. and last.byvariable logic.

CREATING A TEST DATA SET

Let's begin by creating a small, sample/artificial data set to use to explore the PROC SORT options and related alternatives. Here is the relevant code:

```
options nocenter;

data test;
      infile datalines dsd dlm="09"x;
      * note: delimited input used as data were pasted into the code/datalines from
        a spreadsheet, resulting in tab-delimited text;
      * you might need to change the delimiter if running on another (non-ascii)
system;
      input ID_1 ID_2 $ Code_1 Code_2 $;
      datalines;
4     X     9     X9
```

```
4       X       11      X9
2       B       3       B3
2       B       5       B5
1       A       1       A1
3       C       7       C7
5       Z       17      Y17
4       Y       13      Y13
;
run;
proc print data=test;
       title "Raw test data for sorts";
run;
```

The resultant raw SAS data set is shown below. Note the duplicate rows for the pairs defined by variables (ID_1, ID_2).

Raw test data for sorts

Obs	ID_1	ID_2	Code_1	Code_2
1	4	X	9	X9
2	4	X	11	X9
3	2	B	3	B3
4	2	B	5	B5
5	1	A	1	A1
6	3	C	7	C7
7	5	Z	17	Y17
8	4	Y	13	Y13

SORTING AND SPLITTING THE DATA SET

Using NOUNIQUEKEY and UNIQUEOUT= options (SAS 9.3+). Now let's sort and split the file using the NOUNIQUEKEY and UNIQUEOUT= options, using the following code.

```
proc sort data=test out=nonunique_rows  uniqueout=unique_rows nouniquekey;
       by ID_1 ID_2;
run;

proc print data=unique_rows;
       title "Unique rows, UNIQUEOUT= SORT option";
       title2 "BY ID_1 ID_2";
run;

proc print data=nonunique_rows;
       title "Non_unique rows, NOUNIQUEKEY SORT option";
    title2 "BY ID_1 ID_2";
run;
```

The code above splits the raw data set into 2 files, based on whether the row set identified in the BY variables is unique in the input data set or has multiple instances. The results are:

Unique rows, UNIQUEOUT= SORT option				
BY ID_1 ID_2				
Obs	ID_1	ID_2	Code_1	Code_2
1	1	A	1	A1
2	3	C	7	C7
3	4	Y	13	Y13
4	5	Z	17	Y17

Non_unique rows, NOUNIQUEKEY SORT option				
BY ID_1 ID_2				
Obs	ID_1	ID_2	Code_1	Code_2
1	2	B	3	B3
2	2	B	5	B5
3	4	X	9	X9
4	4	X	11	X9

Legacy method to provide the same sort/split. The same result can be achieved in Base SAS without using PROC SORT options, by doing a simple sort that is followed by a DATA step that uses first. and last.byvariable logic customized for the sorting variables, with the OUTPUT statement. This code yields the exact same split as shown above:

```
proc sort data=test out=test2;
      by ID_1 ID_2;
run;

proc print data=test2;
      title "Sorted test data";
run;

data unique non_unique;
      set test2;
      by ID_1 ID_2;

      if (first.ID1 and last.ID1) then
            output unique;
      else if (first.ID_2 and last.ID_2) then
            output unique;
      else output non_unique;
run;

proc print data=unique;
      title "Unique rows, legacy method";
run;

proc print data=non_unique;
      title "Non_unique rows, legacy method";
run;
```

The new PROC SORT options provide a more efficient approach, compared to the legacy method.

PROC SQL can provide similar functionality. The same split can be done in PROC SQL but it is considerably more work than using the PROC SORT options. The basic steps in SQL are:

1. Sort the file and use GROUP BY to get a row count for each value of the BY variable row set
2. Do a join, input data set with row count data set (#1) to insert row counts and sort the result
3. Create a sorted extract file, all rows where the BY variable row count is 1 (unique)
4. Create a sorted extract file, all rows where the BY variable row count is >1 (non-unique)

Sample code follows; it produces the same 2 derivative files as the methods above.

```
* can do same thing in proc sql but more work (4 steps);

* get row counts for BY groups;
PROC SQL;
      CREATE TABLE WORK.ID_vars_count AS
            SELECT t1.ID_1,
                  t1.ID_2,
                  (COUNT(t1.ID_1)) LABEL="row_count" AS row_count
            FROM WORK.TEST t1
                  GROUP BY t1.ID_1,
                        t1.ID_2
                  ORDER BY t1.ID_1,
                        t1.ID_2;
QUIT;

* insert row counts for BY groups into raw data file;
PROC SQL;
   CREATE TABLE WORK.test_plus_count AS
   SELECT t1.ID_1,
          t1.ID_2,
          t1.Code_1,
          t1.Code_2,
          t2.row_count
      FROM WORK.TEST t1
           LEFT JOIN WORK.ID_VARS_COUNT t2 ON
                  (t1.ID_1 = t2.ID_1) AND (t1.ID_2 = t2.ID_2)
      ORDER BY t1.ID_1,
               t1.ID_2;
QUIT;

* split file - create extract with unique rows;
PROC SQL;
   CREATE TABLE WORK.sql_unique AS
   SELECT t1.ID_1,
          t1.ID_2,
          t1.Code_1,
          t1.Code_2
      FROM WORK.TEST_PLUS_COUNT t1
      WHERE t1.row_count = 1
      ORDER BY t1.ID_1,
               t1.ID_2;
QUIT;

proc print data=WORK.sql_unique;
      title "SQL: extract with unique rows";
run;

* split file - create extract with non-unique rows;
PROC SQL;
   CREATE TABLE WORK.sql_non_unique AS
   SELECT t1.ID_1,
          t1.ID_2,
```

```
          t1.Code_1,
          t1.Code_2
      FROM WORK.TEST_PLUS_COUNT t1
      WHERE t1.row_count > 1
      ORDER BY t1.ID_1,
               t1.ID_2;
QUIT;

proc print data=WORK.sql_non_unique;
       title "SQL: extract with non-unique rows";
run;
```

ALTERNATIVES: PROC SORT NODUPKEY AND DUPOUT=; PROC SQL SELECT DISTINCT

The files created by PROC SORT options NODUPKEY and DUPOUT= are different from the files created by the NOUNIQUEKEY and UNIQUEOUT= options. Let's compare using our sample data set; run this code:

```
proc sort data=test out=ndk_test nodupkey dupout=ndk_dups;
       by ID_1 ID_2;
run;

proc print data=ndk_test;
title "Nodupkey primary output file";
run;

proc print data=ndk_dups;
title "Nodupkey dupout= output file";
run;
```

And we get the following 2 derivative files:

Nodupkey primary output file

Obs	ID_1	ID_2	Code_1	Code_2
1	1	A	1	A1
2	2	B	3	B3
3	3	C	7	C7
4	4	X	9	X9
5	4	Y	13	Y13
6	5	Z	17	Y17

Nodupkey dupout= output file

Obs	ID_1	ID_2	Code_1	Code_2
1	2	B	5	B5
2	4	X	11	X9

Comparing the 2 files above with those produced by NOUNIQUEKEY and UNIQUEOUT= options confirms that the output files are indeed different.

PROC SQL SELECT DISTINCT. The SELECT DISTINCT statement is widely used in SQL, but the processing performed/end result is (usually) different from that done by PROC SORT. (Note: in this context, SELECT DISTINCT is used in conjunction with an associated ORDER BY clause.) SELECT

DISTINCT creates a data set containing the unique rows from the input data structure (which may be a single data set or the results of multiple, complex joins) that is defined by and contains **only** the variables specified in the SELECT statement. In contrast to this, PROC SORT outputs **all** of the variables in the data set unless the data set options KEEP= and/or DROP= (or similar KEEP, DROP statements) are used in the PROC invocation to constrain the variables in the output file(s). Another difference: SELECT DISTINCT does not create an extract with the non-unique rows.

CONCLUSION AND SUMMARY

The SAS PROC SORT options NOUNIQUEKEY and UNIQUEOUT= options provide an easy and efficient way to sort and split an input file into 2 subsets: one with truly unique rows per a set of BY variables, and one with rows that are multiples. The older PROC SORT options NODUPKEY and DUPOUT=; also SQL SELECT DISTINCT (used with an ORDER BY clause) are very useful tools, and the processing they perform is generally different from that done via NOUNIQUEKEY and UNIQUEOUT=..

As programmers we should use the optimal tool for the application, and as these newer PROC SORT options are very useful, they should be in your programmer toolkit. The following table summarizes these features and documents their differences.

COMPARISON OF PROC SORT OPTIONS, SQL SELECT DISTINCT

PROC SORT Option/SQL	Description of output data set
NOUNIQUEKEY	Contains ALL (and ONLY) rows from input data set that are multiples per BY variables
UNIQUEOUT=	Contains ONLY rows that are unique per BY variables; these rows do NOT have ANY multiple instances in input data set
NODUPKEY	Output data set contains rows that are unique per BY variables, plus 1 row from each row set that has multiple BY variables
DUPOUT=	The remaining rows from the input data set, leftover after NODUPKEY extraction, i.e., part of the multiple rows
SQL SELECT DISTINCT	Unique rows based on (and containing ONLY) the specified SELECT variables

APPENDIX 1:
BSD 2-CLAUSE COPYRIGHT LICENSE (OPEN SOURCE)

```
* All program code in this paper is released under a Berkeley Systems
  Distribution BSD-2-Clause license, an open-source license that permits
  free reuse and republication under conditions;

/*
Copyright (c) 2017, MUFG Union Bank, N.A.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
```

```
1. Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
*/
```

REFERENCES

Note: all URLs quoted or cited herein were accessed in May 2017.

SAS Institute, Inc. (2017) online documentation:

- SORT Procedure; PROC SORT Statement. *Base SAS® 9.4 Procedures Guide, Seventh Edition.* URL: http://support.sas.com/documentation/cdl/en/proc/70377/HTML/default/viewer.htm#p02bhn81rn4u64n1b6l00ftdnxge.htm

CONTACT INFORMATION

A list of the author's SAS-related papers, including URLs for free access, is available at:
http://www.sascommunity.org/wiki/Presentations:Tebillings_Papers_and_Presentations
or use this alternate short URL: http://goo.gl/uocYNc

Thomas E. Billings
MUFG Union Bank, N.A.
Basel II - Retail Credit BTMU
350 California St.; 6th floor
San Francisco, CA 94104

Phone: 415-273-2522
Email: tebillings@gmail.com