Paper CC12

Handling Data with Multiple Records per Subject: 4 Quick Methods to Pull Only the Records You Want

Elizabeth Leslie, Kaiser Permanente, Atlanta, GA

ABSTRACT

Storing data with Multiple Records per Subject (MRPS) is a common practice in a variety of fields. There are often times when an analyst needs to select only one record that meets certain criteria, such as the latest test result for a subject, or the highest value of a particular field. This paper covers four techniques to select only records that meet a specific criteria and how to avoid duplicates when more than one record qualifies. The techniques covered use a DATA step with by-group processing, PROC SORT, PROC SQL with GROUP BY, and PROC SQL with subqueries.

INTRODUCTION

When records are stored with multiple rows for one subject, selecting only one record for each subject that meets your criteria seems straight forward, but can actually be tricky. If there are multiple records that meet the selection criteria in the original data set, the output data set could have multiple records for a single subject. These circumstances can be avoided by refining the selection criteria. The simulated medical data used in this paper does have several instances of test results from the same day to illustrate this point.

The example data, TEST_RESULT, consists of test results for four different patients. Each patient has five result values and dates stored on five rows, for a total of 20 observations. The code to generate this data set and the code used for each example can be found at the end of this paper.

ID	RESULT_VALUE	RESULT_DATE
0001	7.1	4-Jan
0001	5.3	6-Jan
0001	5.3	7-Jan
0001	5.0	14-Jan
0001	6.8	14-Jan
0002	6.0	2-Jan
0002	5.8	2-Jan
0002	5.7	5-Jan
0002	6.4	6-Jan
0002	8.5	6-Jan
0003	6.1	7-Jan
0003	7.5	9-Jan
0003	4.7	12-Jan
0003	5.1	13-Jan
0003	7.5	13-Jan
0004	7.2	4-Jan
0004	7.2	4-Jan
0004	5.5	6-Jan

0004	4.9	12-Jan
0004	6.0	14-Jan

Table 1. Contents of the TEST_RESULT table with Multiple Records per Subject

Our goal is to pull the only latest test result for each subject and, for those patients that have multiple results from the same day, to pull the smallest result value for that day. This is what the desired output table should look like, with four IDs and four result values:

ID	RESULT_VALUE	RESULT_DATE
0001	6.8	14-Jan
0002	6.4	6-Jan
0003	5.1	13-Jan
0004	6.0	14-Jan

Table 2. Desired Output with Latest Test Results

TECHNIQUE 1: DATA STEP WITH BY GROUP PROCESSING

The DATA step lends itself very well to this type of data selection because it processes each row of data sequentially. Like many other SAS procedures, a BY statement can be added to the DATA step for group processing, but the data must first be sorted by the BY variables. Since our objective is to find the latest test result for each subject, the keyword DESCENDING needs to be added before the DATE variable.

```
proc sort data=test_result;
by ID descending result_date result_value;
run;
```

Notice that the data was also sorted by RESULT_VALUE. Adding RESULT_VALUE to the BY statement ensures that if there is more than one result on the same day, the lowest value for RESULT_VALUE will be listed first in the data set. Here is what the first few rows of the sorted data looks like:

ID	RESULT_VALUE	RESULT_DATE
0001	5.0	14-Jan
0001	6.8	14-Jan
0001	5.3	7-Jan
0001	5.3	6-Jan
0001	7.1	4-Jan
0002	6.4	6-Jan
0002	8.5	6-Jan
	…	

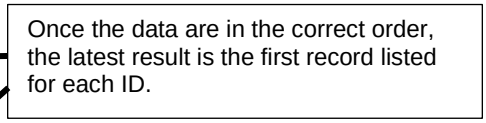

Table 3. First 7 Rows of the Sorted TEST_RESULT Table

Now that the first record for each ID is the latest test result, by group processing with a subsetting IF statement can be used to select only that record.

```
data result_set1;
set test_result;
by ID;
if first.id;
run;
```

When the BY statement is added to the DATA step, two temporary binary variables are created during the processing, *first.variable-name* and *last.variable-name*. The temporary variable first.ID will be equal to 1 for the first record of each ID and 0 for the rest. Since there are four individual ID's, first.ID will be equal to 1 only four times.

The subsetting IF restricts the output data set to only those rows where first.ID does not equal 0, i.e. the first record for each ID which is the lowest, latest result value for each subject. A quick inspection of the output data set, RESULT_SET1, shows the desired results:

ID	RESULT_VALUE	RESULT_DATE
0001	5.0	14-Jan
0002	6.4	6-Jan
0003	5.1	13-Jan
0004	6.0	14-Jan

Table 4. RESULT_SET1

TECHNIQUE 2: PROC SORT

Another quick technique to find only the latest results is using PROC SORT with the NODUPKEY option. When the NODUPKEY option is used, PROC SORT will compare the value of each BY variable with the last BY variable printed to the output data set. PROC SORT will then only write a record to the output data set if the BY variable values are different.

Caution: the input data set MUST be sorted in the desired order before using PROC SORT with the NODUPKEY option. Unlike the DATA step, if the data is not presorted, PROC SORT will still run without any errors, but it will produced unexpected results.

To use this method, PROC SORT will need to be called twice. First use PROC SORT **without** the NODUPKEY option to sort the data so the desired row is the first row for each ID (this is the same code used to sort in the previous example):

```
proc sort data=test_result;
by ID descending result_date result_value;
run;
```

Then use PROC SORT **with** the NODUPKEY option. The NODUPKEY option can be used to only print the first row for each ID to the output data set, RESULT_SET2. Be sure to specify an output data set or else the original data set will be overwritten and data will be lost.

```
proc sort data=test_result out=result_set2 nodupkey;
by ID;
run;
```

A quick look at the output data set shows that the desired results were achieved.

ID	RESULT_VALUE	RESULT_DATE
0001	5.0	14-Jan
0002	6.4	6-Jan
0003	5.1	13-Jan
0004	6.0	14-Jan

Table 4. RESULT_SET2

TECHNIQUE 3: PROC SQL GROUP BY

PROC SQL is a very useful tool that can handle many data management issues. We can take advantage of the GROUP BY functionality of PROC SQL to select the patient's latest test result.

So far, multiple records on the same day have been discussed, but not exact duplicate records. It is important to know if the data has exact duplicates, such as the same record entered twice. Having exact duplicates will not affect the results from the first two techniques; the by group processing used by the DATA step and PRAC SORT will only

select one of the duplicate values. For the last two techniques using PROC SQL, the DISTINCT keyword must be added to the queries to select only one of the duplicate values.

In order to account for the results from the same day, there will need to be several parts to the query.

First, find the latest test results for the patients (note: PROC SQL does not require pre-sorting):

```
proc sql;
      create table result_set3 as
      select distinct *
      from test_result
      group by ID
      having result_date=max(result_date);
quit;
```

By adding the GROUP BY clause, the summary function, MAX(), pulls the maximum (latest) date for each ID, so the results will return any records where the date matches the maximum date for that ID.

ID	RESULT_VALUE	RESULT_DATE
0001	5.0	14-Jan
0001	6.8	14-Jan
0002	6.4	6-Jan
0002	8.5	6-Jan
0003	5.1	13-Jan
0003	7.5	13-Jan
0004	6.0	14-Jan

As expected, there are multiple records returned when there were results from the same day. Another query is necessary to find the lowest values from that day.

Table 5. RESULT_SET3_STEP1

The next step is to select the minimum result value when there are multiple results from the same day. A second PROC SQL query using the same GROUP BY methodology will accomplish the desired results. There are two ways to accomplish this.

You can query the result table from the first query:

```
proc sql;
create table result_set3 as
select distinct * from result_set3a
group by ID
having result=min(result);
quit;
```

Or you can create an in-line view by inserting the first query into the FROM clause of a second query:

```
proc sql;
create table result_set3 as
select distinct * from
  (
    select distinct *
    from test_result
    group by ID
    having result_date=max(result_date)
  )
group by ID
having result_value=min(result_value);
quit;
```

The in-line view selects the records with latest date for each ID

The outer query selects the records with the smallest result value for each ID from the results returned by the in-line view

The second set of code is more efficient because it avoids creating an intermediate data set and avoids a second call

to PROC SQL.

A quick look at RESULT_SET3 shows the query was successful.

ID	RESULT_VALUE	RESULT_DATE
0001	5.0	14-Jan
0002	6.4	6-Jan
0003	5.1	13-Jan
0004	6.0	14-Jan

Table 6. RESULT_SET3

TECHNIQUE 4: PROC SQL SUBQUERY

For those programmers who are more comfortable using subqueries rather than GROUP BY in PROC SQL, the same results can be achieved with subqueries. A subquery is a nested query in a WHERE clause, similar to an in-line view which is a nested query in a FROM clause.

Again, steps need to be taken to avoid duplicates in the output data set when there are multiple results from the same day. The first subquery in the WHERE clause selects the latest date for each ID and the second subquery after the AND in the WHERE clause selects the lowest result value for each ID.

```
proc sql;
create table result_set4 as
select a.id, a.result_value, a.result_date
from test_result as a
where a.result_date = (
            select max(result_date)
            from test_result as b
            where a.id=b.id
                  )
      and a.result_value = (
                  select min(result_value)
                  from test_result as c
                  where a.id=c.id
                  and a.result_date=c.result_date
                  );
quit;
```

The 1st subquery selects the records with the latest date for each ID

The 2nd subquery selects the lowest result when there are multiple results from the same day

Here is a look at the result data set:

ID	RESULT_VALUE	RESULT_DATE
0001	5.0	14-Jan
0002	6.4	6-Jan
0003	5.1	13-Jan
0004	6.0	14-Jan

Table 6. RESULT_SET4

CONCLUSION

The data set used in these examples is a very small data set with only 20 observations. However, these same techniques can be applied to much larger data sets. The important thing is to know the data. Know how many distinct subjects there are and whether or not there are circumstances where more than one record will be returned for each subject and steps can be taken to avoid duplicates.

SAS CODE

```
/*The example data set, TEST_RESULT*/

      data test_result;
        input ID $ result_value result_date mmddyy10.;
        format result_value 5.1 result_date date5.;
        datalines;
        0001        7.1        1/4/2013
        0001        5.3        1/6/2013
        0001        5.3        1/7/2013
        0001        5.0        1/14/2013
        0001        6.8        1/14/2013
        0002        6.0        1/2/2013
        0002        5.8        1/2/2013
        0002        5.7        1/5/2013
        0002        6.4        1/6/2013
        0002        8.5        1/6/2013
        0003        6.1        1/7/2013
        0003        7.5        1/9/2013
        0003        4.7        1/12/2013
        0003        5.1        1/13/2013
        0003        7.5        1/13/2013
        0004        7.2        1/4/2013
        0004        7.2        1/4/2013
        0004        5.5        1/6/2013
        0004        4.9        1/12/2013
        0004        6.0        1/14/2013
      ;
      run;

/*Technique 1: DATA step with by group processing*/

      proc sort data=test_result;
      by ID descending result_date result_value;
      run;

      data result_set1;
      set test_result;
      by ID;
      if first.id;
      run;

/*Technique 2: PROC SORT with NODUPKEY option*/

      proc sort data=test_result;
      by ID descending result_date result_value;
      run;

      proc sort data=test_result out=result_set2 nodupkey;
      by ID;
      run;
```

```
/*Technique 3: PROC SQL with GROUP BY*/

      proc sql;
      create table result_set3 as
      select distinct * from
        (
          select distinct *
          from test_result
          group by ID
          having result_date=max(result_date)
        )
      group by ID
      having result_value=min(result_value);
      quit;


/*Technique 4: PROC SQL with subqueries*/

      proc sql;
      create table result_set4 as
      select a.id, a.result_value, a.result_date
      from test_result as a
      where a.result_date = (
                  select max(result_date)
                  from test_result as b
                  where a.id=b.id
                        )
            and a.result_value = (
                        select min(result_value)
                        from test_result as c
                        where a.id=c.id
                        and a.result_date=c.result_date
                        );
      quit;
```

RECOMMENDED READING

- *SAS® Certification Prep Guide: Base Programming for SAS®9, Third Edition*. Cary, NC: SAS Institute, Inc.
- *SAS® Certification Prep Guide: Advanced Programming for SAS®9, Third Edition*. Cary, NC: SAS Institute, Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Elizabeth Leslie
Enterprise: Kaiser Permanente of Georgia
Address: Nine Piedmont Center, 3495 Piedmont Road, N.E.
City, State ZIP: Atlanta, GA 30305
Work Phone: (404) 364-7346
E-mail: Elizabeth.leslie@kp.org
Web: www.linkedin.com/pub/beth-leslie/50/957/a26