

Duplicates or “Can you repeat that?”

Lawrence Heaton-Wright, Quintiles, Bracknell, Berkshire, UK

WHAT IS A DUPLICATE?

A duplicate is defined in the Oxford English Dictionary as:

“Either of two or more specimens of a thing exactly or virtually alike”

What does this mean for SAS® users? In our specialised world this means data with two or more observations which have all variables with exactly the same values or several identification variables (e.g. identification variables in a PROC COMPARE) with the same values.

WHY DUPLICATES ARE A PROBLEM

Duplicate observations are a fact of life in the pharmaceutical industry. These often occur when we are creating data sets which are used in summary tables. We generally subset the input data by selecting only those observations (e.g. ITT subjects) and variables we need to create the summary statistics.

Duplicate observations are also encountered when using PROC COMPARE to compare 2 data sets. To use COMPARE (generally) a set of identification variables are required which (should) determine a unique record. Sometimes we find that our data gives us duplicate identification variables (unscheduled visits anyone?).

HOW CAN DUPLICATE OBSERVATIONS BE REMOVED?

USING PROC SORT

There are two methods of removing duplicate observations in a PROC SORT. We can use the options NODUPKEY or NODUPRECS.

THE NODUPKEY OPTION

From SAS Help:

“Checks for and eliminates observations with duplicate BY values. If you specify this option, then PROC SORT compares all BY values for each observation to those for the previous observation that is written to the output data set. If an exact match is found, then the observation is not written to the output data set.”

THE NODUPRECS (OR NODUP) OPTION

From SAS Help:

“Checks for and eliminates duplicate observations. If you specify this option, then PROC SORT compares all variable values for each observation to those for the previous observation that was written to the output data set. If an exact match is found, then the observation is not written to the output data set.”

WHAT'S THE DIFFERENCE BETWEEN NODUPKEY AND NODUPRECS?

NODUPKEY checks observations based on the “key” variables: the list of variables in the BY statement. NODUPRECS checks all variables, not just “key” variables.

PhUSE 2006

COMPARING NODUPKEY AND NODUPRECS

Using the example data shown below we will investigate the differences between the NODUPKEY and NODUPRECS options.

```
DATA duplicate_example;
  INPUT subjid visit value @@;
  CARDS;
1 1 87 1 2 79 1 3 86 1 3 86 1 3 85 1 4 91 1 5 87
;
RUN;

PROC SORT DATA=duplicate_example OUT=nodupkey_example NODUPKEY;
  BY subjid visit;
RUN;
```

Running this procedure gives the following results:

Obs	subjid	visit	value
1	1	1	87
2	1	2	79
3	1	3	86
4	1	4	91
5	1	5	87

```
PROC SORT DATA=duplicate_example OUT=noduprecs_example NODUPRECS;
  BY subjid visit;
RUN;
```

Running this procedure gives the following results:

Obs	subjid	visit	value
1	1	1	87
2	1	2	79
3	1	3	85
4	1	3	86
5	1	4	91
6	1	5	87

As can be seen from the results the NODUPKEY only produces 5 observations (1 for each combination of SUBJID and VISIT) and the NODUPRECS produces 6 observations thanks to the VISIT=3 record which has 3 observations with VALUE equalling 86, 86 and 85. Therefore NODUPRECS only considered 2 of these observations to be duplicates as, obviously, 86 is the same as 86.

A NOTE OF CAUTION WITH NODUPRECS

The action of NODUPRECS means that the initial sort order of the data set is important as the processor compares the new observation to the previous observation sent to the output data set. However, the NODUPKEY option appears to compare all identification variable combinations prior to writing the observation to the output data set. The initial order of the input data set is important only for determining which observation is written to the output data set. The first record with the unique variable combination (from the BY statement) is written to the output data set and this is used to compare against other BY variable combinations.

If we change the initial sort order of our data set we can see that the action of the NODUPRECS processing is subtly different to that which we would expect.

```
DATA duplicate_order_example;
  INPUT subjid visit value @@;
  CARDS;
1 1 87 1 2 79
1 3 86
1 3 85
1 3 86
1 4 91 1 5 87
;
RUN;

PROC SORT DATA=duplicate_order_example OUT=noduprecs_example NODUPRECS;
  BY subjid visit;
RUN;
```

Obs	subjid	visit	value
1	1	1	87
2	1	2	79
3	1	3	86
4	1	3	85
5	1	3	86
6	1	4	91
7	1	5	87

HOW DO I KNOW WHICH OBSERVATIONS HAVE BEEN REMOVED AS DUPLICATES?

In version 8 you had no way of knowing. In version 9 there is the DUPOUT option. This enables users to send the duplicate observations to a data set specified in the DUPOUT= option.

```
PROC SORT DATA=duplicate_example OUT=nodupkey_example DUPOUT=duplicate NODUPKEY;
  BY subjid visit;
RUN;
```

USING DATA STEP BY-PROCESSING

To be absolutely certain which observation we are selecting as our unique record we can always use BY-processing within the DATA step which allows the usage of the FIRST and LAST processing available within the DATA step.

```
PROC SORT DATA=duplicate_example;
  BY subjid visit value;
RUN;

DATA firstdot_example;
  SET duplicate_example;
  BY subjid visit value;
  IF FIRST.visit;
RUN;
```

Obs	subjid	visit	value
1	1	1	87
2	1	2	79
3	1	3	85
4	1	4	91
5	1	5	87

PhUSE 2006

The only thing to note here is the action of FIRST and LAST processing. When BY-processing is requested in a DATA step then the FIRST and LAST variables are automatically initialised by the PDV (Program Data Vector) with values of 1 and 0. One indicates that the observation fulfils the criteria (i.e. FIRST.SUBJID=1 indicates that this observation is the first SUBJID in the data set). The FIRST and LAST variables are defined in the hierarchical order of the BY statement (i.e. FIRST.VISIT and LAST.VISIT are defined within SUBJID, etc.).

SUBJID	FIRST. SUBJID	LAST. SUBJID	VISIT	FIRST. VISIT	LAST. VISIT	VALUE	FIRST. VALUE	LAST. VALUE
1	1	0	1	1	1	87	1	1
1	0	0	2	1	1	79	1	1
1	0	0	3	1	0	85	1	1
1	0	0	3	0	0	86	1	0
1	0	0	3	0	1	86	0	1
1	0	0	4	1	1	91	1	1
1	0	1	5	1	1	87	1	1

Duplicate observations can be identified where the FIRST and LAST variables are both not equal to 1 (i.e. IF NOT (FIRST.visit AND LAST.visit)).

USING PROC SQL

As always with SAS® there are many different ways to achieve the same result. Using the SQL procedure we can use the SELECT statement in conjunction with the DISTINCT keyword. When using the SELECT DISTINCT statement all of the variables in the SELECT statement are included in the calculation of duplicate observations.

```
PROC SQL;
  CREATE TABLE sql_distinct AS
    SELECT DISTINCT * FROM DATA=duplicate_example
    ORDER BY subjid, visit, value;
QUIT;
```

DID YOU REALLY WANT TO REMOVE THOSE DUPLICATES?

In certain circumstances we don't want to remove duplicate observations from a data set. The most common example of this is when summarising adverse event data. We are interested in the number of subjects with a certain adverse event term but we are also interested in the number of events within each term. This is when you wouldn't want to remove duplicate observations. However, being able to identify these records is important as this could help in ensuring the summary is correct or perhaps in the creation of a footnote to explain such data anomalies.

IN CONCLUSION

There are several different methods of identifying and/or removing duplicate observations from SAS data sets from the simple options of the SORT procedure to the (slightly) more long-winded DATA step processing utilising FIRST and LAST BY statement processing. SQL can also be utilised to eliminate duplicates. In fact, this is one of the simplest and (probably) most-widely used parts of PROC SQL.

If using version 9 there is the DUPOUT option which allows users to capture duplicate observations which are removed from the output data set. The DATA step with By-processing is an extremely useful methodology which allows users to be extremely precise about the duplicates captured.

Do not blindly use the NODUPKEY option as this may result in observations being removed from data sets which are not the duplicate records you sought. This is why other methods of identifying duplicate observations are presented here; as with everything in SAS programming; it's all about investigating, interrogating and knowing the make-up of your data.

REFERENCES

Ed. L. Brown, The New Shorter Oxford English Dictionary, Clarendon Press, Oxford.
SAS® Procedures Guide, Version 8

CONTACT INFORMATION

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

Lawrence Heaton-Wright
Quintiles Limited
Station House, Market Street
Bracknell, Berkshire, RG12 1HX
United Kingdom
Work Phone: +44 1344 708320
Fax: +44 1344 708106
Email: lawrence.heaton-wright@quintiles.com
Web: www.quintiles.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.