Paper CT-11

An Introduction to Criteria-based Deduplication of Records

Elizabeth Heath – RTI International, RTP, NC
Priya Suresh – RTI International, RTP, NC

ABSTRACT

When survey respondents are allowed to select whether they will complete a paper or electronic version of a survey, a few respondents will inadvertently submit two versions of the survey. Because the survey needs only one data submission from each respondent, multiple submissions per respondent are first identified, reviewed for completeness and other criteria provided by the survey, and then the criteria are applied for keeping only one record per respondent. We will show how SAS® can be used to apply selection criteria to identify and remove duplicate records for a respondent.

INTRODUCTION

In preparing data delivery files, it is standard practice to check for and remove duplicate records. The SAS sort procedure is the foundation of deduplicating records. For a good reference on the SAS sort procedure and using SAS to deduplicate records, see Heidi Markovitz's paper entitled "Dup, Dedup, DUPOUT - New in PROC SORT" in the Proceedings of the 14th Annual SESUG Conference, October 2006. We prepared this paper for beginning SAS users to show how duplicate data records can be removed from survey data and how data must be reviewed to determine criteria for deduplicating survey data.

SAMPLE DATA

For discussion purposes, we will use the fictitious survey data shown in Table 1. The survey contains an identification number for each respondent and seven questions. In Table 1, observations 1 and 5 have no missing values for questions Q4 and Q6, while the other observations have missing values for Q4 or Q6. The missing question Q4 and Q6 values are due to legitimate skips based on gate questions Q3 and Q5. Observations 4 and 7 have incomplete records. The data also include submission date and whether the data collection mode was electronic (e) or paper (p). The identification number and questions are numeric and the submission date is a numeric SAS date.

Obs	id	Q1	Q2	Q3	Q4	Q5	Q6	Q7	mode	subdt
1	10	1	5	1	3	1	7	4	p	1/3/2011
2	11	3	6	2	.	2	.	5	p	2/7/2011
3	12	4	3	2	.	1	6	3	p	1/15/2011
4	12	4	3	2	.	.	.	.	p	4/2/2011
5	21	2	3	1	4	1	3	4	e	3/1/2011
6	22	1	5	2	.	1	4	5	p	2/2/2011
7	10	1	5	1	3	.	.	.	e	12/1/2010
8	15	2	4	1	2	2	.	6	p	3/21/2011
9	11	3	6	2	.	2	.	5	e	1/7/2011

Table 1. Fictitious sample data

CHECK FOR DUPLICATE RECORDS

Visual inspection of Table 1 shows that there are two records for respondents with IDs of 10, 11, and 12. While it is easy to find duplicate data submissions in the fictitious data shown in Table 1, it is impractical to visually check for duplicate records when there are hundreds if not thousands of respondents for a survey . A quick way to check for duplicate records is to run the sort procedure with the NODUPKEY option for the variable ID. In the sort procedure statement that follows, unique IDs are written to the dataset called unqIDs, in order to not remove duplicate records from the source dataset called sesug. As shown in Output 1, the sort procedure finds three duplicate IDs. The next step is to examine the duplicate records and evaluate how to remove the duplicate records.

```
proc sort data = sesug (keep=id) nodupkey out= unqIDs;
  by id;
run;
```

```
NOTE: There were 9 observations read from the data set WORK.SESUG.
NOTE: 3 observations with duplicate key values were deleted.
NOTE: The data set WORK.UNQIDS has 6 observations and 1 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.04 seconds
      cpu time            0.04 seconds
```

Output 1. Program log for PROC SORT Statement

EXAMINE DATA IN DUPLICATE RECORDS

Before data collection started, the project anticipated that they might receive more than one data submission from a few respondents and initially thought that they would keep the first data submission. Now that we know that there are duplicate records, we need to examine the data in the duplicate records to see if the decision to keep the first submitted record is the right decision.

To check this, the data are sorted by ID and ascending record submission date with the following code, in order to display the duplicate record with earliest submission date first.

```
proc sort data =  sesug;
  by ID subDT;
run;
```

As the data are sorted by ID and ascending record submission date, the sort procedure determines whether the ID occurs one time, which is the first and last occurrence of the ID, or if the ID occurs more than one time. As shown in Table 2, for duplicate ID number 10, the December 2010 record is the first occurrence of ID 10 and the January 2011 record is the last occurrence of ID 10, while unique ID number 21 is the first and last occurrence of ID 21.

ID	Record submission date	Unique or duplicate ID	First or last ID
10	12/01/2010	Duplicate	First.ID
10	01/03/2011	Duplicate	Last.ID
21	03/01/2011	Unique	First.ID and Last.ID

Table 2. Example of first and last ID numbers for data sorted by ID and ascending submission date

After sorting the survey data by ID and ascending date, records with duplicate IDs are written to dataset dups with the following code.

```
data dups;
  set sesug;
  by ID subDT;
  if not (first.ID and last.ID);
run;
```

Output 2 shows that if the first submitted record were used as the deduplication criteria, for ID number 10, the December 2010 record would be kept but it has less data than the January 2011 record. For ID numbers 11 and 12, the criteria of keeping the first submitted record is fine. After reviewing Output 2, project staff decided that they prefer keeping records with more non-missing data over simply keeping the first submitted record.

Obs	id	Q1	Q2	Q3	Q4	Q5	Q6	Q7	mode	subdt
1	10	1	5	1	3	.	.	.	e	12/01/2010
2	10	1	5	1	3	1	7	4	p	01/03/2011
3	11	3	6	2	.	2	.	5	e	01/07/2011
4	11	3	6	2	.	2	.	5	p	02/07/2011
5	12	4	3	2	.	1	6	3	p	01/15/2011
6	12	4	3	2	.	.	.	.	p	04/02/2011

Output 2. Duplicate record output from sort and data step

RECORD COMPLETENESS

To check the effect of sorting data by record completeness, the completeness of each record is determined by counting the number of non-missing responses to questions that are asked of all respondents. This means that questions Q4 and Q6, which can be skipped, are excluded from the completeness count in the code that is shown below.

```
arraY numQ (*) Q1 Q2 Q3 Q5 Q7;
compNum = 0;
do i = 1 to dim(numQ);
  if numQ(i) ^= . then compNum = compNum + 1;
end;
drop i;
```

As shown in Output 3, deduplicating data by record completeness is good for ID numbers 10 and 12 but the duplicate records for ID number 11 have identical completeness counts. As a result of the data in Output 3, the project decided to deduplicate records based on the highest record completeness and then by the earliest submission date.

Obs	id	CompNum	subdt
1	10	3	12/01/2010
2	10	5	01/03/2011
3	11	5	01/07/2011
4	11	5	02/07/2011
5	12	5	01/15/2011
6	12	3	04/02/2011

Output 3. Completeness and submission date for duplicate records

DEDUPLICATE THE DATA

As shown in the following code, deduplication criteria are implemented by first sorting the data by ID number, descending record completeness, and ascending submission date, and then writing the first occurrence of each ID number to dataset survey_unq and all other records, which are duplicates, to dataset survey_dups. A final check is done for duplicate ID numbers in dataset suvery_unq. As shown in Output 4, the final check for duplicate records shows there are no duplicates in dataset survey_unq, and Output 5 shows the deduplicated data in survey_unq.

```
proc sort data = sesug;
  by id descending compnum subdt;
run;

data survey_unq survey_dups;
  set sesug;
  by id descending compnum subdt;
  if first.ID then output survey_unq;
  else output survey_dups;
run;

proc sort data = survey_unq nodupkey out=ck4dups;
  by id;
run;
```

```
NOTE: There were 6 observations read from the data set WORK.SURVEY_UNQ.
NOTE: 0 observations with duplicate key values were deleted.
NOTE: The data set WORK.CK4DUPS has 6 observations and 10 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds
```

Output 4. Final check for duplicate records

Obs	id	Q1	Q2	Q3	Q4	Q5	Q6	Q7	mode	subdt	CompNum
1	10	1	5	1	3	1	7	4	p	01/03/2011	5
2	11	3	6	2	.	2	.	5	e	01/07/2011	5
3	12	4	3	2	.	1	6	3	p	01/15/2011	5
4	15	2	4	1	2	2	.	6	p	03/21/2011	5
5	21	2	3	1	4	1	3	4	e	03/01/2011	5
6	22	1	5	2	.	1	4	5	p	02/02/2011	5

Output 5. Deduplicated data

CONCLUSION

The SAS sort procedure can be used to identify duplicate records, deduplicate the data using criteria such as non-missingness, and verify the absence of duplicate records after deduplication criteria have been applied. As shown in this paper with the sample data, SAS can also be used to evaluate the effectiveness of proposed deduplication criteria.

REFERENCES

Markovitz, Heidi Markovitz, 2006, CC14 Dup, Dedup, DUPOUT - New in PROC SORT, *Proceedings of the 14th Annual SouthEast SAS Users Group (SESUG) Conference*, Atlanta, GA, October 8–10, 2006.

RECOMMENDED READING

- Base SAS Procedures Guide

CONTACT INFORMATION

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

Elizabeth Heath
Phone: (919) 485-2786
E-mail: eah@rti.org

Priya Suresh
Phone: (919) 541-7428
E-mail: psoh@rti.org

Our mailing address:
RTI International
PO Box 12194
RTP NC 27709-2194