Paper 324-2013

Dealing with Duplicates

Christopher J. Bost, MDRC, New York, NY

ABSTRACT

Variable values may be repeated across observations. If a variable is an identifier, it is important to determine if values are duplicated. This paper reviews techniques for detecting duplicates with PROC SQL, summarizing duplicates with PROC FREQ, and outputting duplicates with PROC SORT.

INTRODUCTION

Data quality checks include looking for duplicate identifiers. For example, if a data set is supposed to have only one observation per person, it is important to confirm that there are no duplicate identifiers. If there are duplicates, they should be output to another data set for inspection.

On the other hand, if a data set can have more than one observation per person, duplicate identifiers are expected. In that situation, it is important to summarize patterns of duplicates (i.e., how many values occur once, how many values occur twice, and so on).

This paper details simple strategies to detect, summarize, and output observations with duplicate identifiers.

SAMPLE DATA

SAS® data set TEST is used in this paper. It contains fourteen (unsorted) observations and two variables:

Obs	id	x
1	104	11
2	102	12
3	102	22
4	103	11
5	101	11
6	105	13
7	105	23
8	106	12
9	106	22
10	105	33
11	107	12
12	107	22
13	108	11
14	109	11

Output 1. Data set TEST

There are 9 distinct values of ID (101 through 109).

There are 5 values of ID that occur **once** (101, 103, 104, 108, and 109).

There are 3 values of ID that occur **twice** (102, 106, and 107).

There is 1 value of ID that occurs **three** times (105).

DETECTING DUPLICATES WITH PROC SQL

Use PROC SQL to count the number of unique values and the number of observations. The syntax is:

```
proc sql;
select count(distinct id) as Ndistinct,
       count(*) as Nobs
from test;
quit;
```

The PROC SQL statement starts the procedure.

The SELECT clause counts the number of distinct values of ID and saves the result as NDISTINCT.

The SELECT clause also counts the number of rows (observations) and saves the result as NOBS.

The FROM clause processes table (data set) TEST. The QUIT; statement ends the SQL procedure.

PROC SQL produces the following output:

Ndistinct	Nobs
9	14

Output 2. Detecting duplicates with PROC SQL

There are 9 distinct values of ID among the 14 rows (observations) in table (data set) TEST. This means that there are duplicate values of ID.

SUMMARIZING DUPLICATES WITH PROC FREQ

Use PROC FREQ to count the number of times each ID occurs and save the results to a SAS data set. Then use PROC FREQ again to count the number of times each frequency occurs. The syntax is:

```
proc freq data=test;
tables id/out=freqout;
run;
```

The PROC FREQ statement starts the procedure.

The TABLES statement specifies a one-way frequency of ID. The OUT= option after the slash stores the results in data set FREQOUT.

(Note that results of this step could be voluminous depending on the data source. It would typically be suppressed by using the NOPRINT option after the slash. All results are printed here for illustrative purposes.)

PROC FREQ produces the following output:

id	Frequency	Percent	Cumulative Frequency	Cumulative Percent
101	1	7.14	1	7.14
102	2	14.29	3	21.43
103	1	7.14	4	28.57
104	1	7.14	5	35.71
105	3	21.43	8	57.14
106	2	14.29	10	71.43
107	2	14.29	12	85.71
108	1	7.14	13	92.86
109	1	7.14	14	100.00

Output 3. Frequency table of ID

The ID value 101 occurs once; the ID value 102 occurs twice; the ID value 103 occurs once; and so on. This output is stored in data set FREQOUT:

Obs	id	COUNT	PERCENT
1	101	1	7.1429
2	102	2	14.2857
3	103	1	7.1429
4	104	1	7.1429
5	105	3	21.4286
6	106	2	14.2857
7	107	2	14.2857
8	108	1	7.1429
9	109	1	7.1429

Output 4. Data set FREQOUT

Note that values of id, Frequency, and Percent from Output 3. are stored in the output data set as ID, COUNT, and PERCENT. The number of times each value of ID occurs is stored as COUNT.

Use PROC FREQ once more to count the number of times each value of COUNT occurs. The syntax is:

```
proc freq data=freqout;
tables count;
run;
```

The PROC FREQ statement starts the procedure.

The TABLES statement specifies a one-way frequency of COUNT.

PROC FREQ produces the following output:

Frequency Count				
COUNT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	5	55.56	5	55.56
2	3	33.33	8	88.89
3	1	11.11	9	100.00

Output 5. Summarizing duplicates with PROC FREQ

There are 5 unique values (COUNT=1); there are 3 duplicates (COUNT=2); and there is 1 triplicate (COUNT=3).

OUTPUTTING DUPLICATES WITH PROC SORT

Use PROC SORT to output all observations with unique values of ID to one data set and all observations with non-unique values of ID to another data set. The syntax is:

```
proc sort data=test nouniquekeys uniqueout=singles out=dups;
by id;
run;
```

The PROC SORT statement starts the procedure. Observations in data set TEST are sorted in ascending order by ID.

The NOUNIQUEKEYS keyword deletes any observation where the value of ID is unique (i.e., occurs only once).

The UNIQUEOUT= keyword stores deleted observations with unique values in the specified data set (SINGLES).

The OUT= keyword stores all other observations (i.e., with non-unique values of ID) in the specified data set (DUPS).

Data set SINGLES and data set DUPS can be printed for inspection:

Obs	id	x
1	101	11
2	103	11
3	104	11
4	108	11
5	109	11

Output 6. Data set SINGLES

Obs	id	x
1	102	12
2	102	22
3	105	13
4	105	23
5	105	33
6	106	12
7	106	22
8	107	12
9	107	22

Output 7. Data set DUPS

Five observations with unique values of ID (101, 103, 104, 108, and 109) are stored in data set SINGLES.

Nine observations with non-unique values of ID (102, 105, 106, and 107) are stored in data set DUPS.

NOUNIQUEKEYS and UNIQUEOUT= are new keywords in SAS 9.3. The NODUPKEY and DUPOUT= keywords still work, but they split observations with non-unique values between two data sets: the first occurrence is stored in the OUT= data set and any subsequent occurrences are stored in the DUPOUT= data set. Storing all observations with unique values in one data set and all observations with non-unique values in another data set is more useful.

CONCLUSION

PROC SQL, PROC FREQ, and PROC SORT can be used to detect, summarize, and output duplicates. These techniques are useful to have in your SAS “toolkit” for checking identifiers. A macro to automate these techniques is included in the appendix.

CONTACT INFORMATION

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

Christopher J. Bost
MDRC
16 East 34th Street
New York, NY 10016
(212) 340-8613
christopher.bost@mdrc.org
chrisbost@gmail.com

APPENDIX

The following macro determines if values of a specified variable are repeated across observations. If duplicates are detected, it summarizes the number of occurrences and outputs unique observations and non-unique observations to respective data sets.

```
%macro dupcheck(inputds=,
                var=,
                uniqueds=singles,
                dupds=dups);

*Detecting duplicates with PROC SQL;
proc sql;
title "Checking for duplicates of &var in &inputds";
select count(distinct &var) as Ndistinct,
       count(*) as Nobs
       into :Ndistinct,:Nobs
from &inputds;
quit;

%if %sysevalf(&Ndistinct/&Nobs) ne 1 %then %do;

    *Summarizing duplicates with PROC FREQ;
    proc freq data=&inputds;
    tables &var/out=freqout noprint;
    run;

    proc freq data=freqout;
    tables count;
    title "FREQ of singles, doubles, etc. of &var in data set &inputds";
    run;

    *Outputting duplicates with PROC SORT;
    proc sort data=&inputds nouniquekeys uniqueout=&uniqueds out=&dupds;
    by &var;
    run;

%end;

%mend dupcheck;
```

The following sample macro call checks for duplicates of variable ID in data set TEST:

```
%dupcheck(inputds=test,var=id)
```

Note: This macro is not guaranteed to work in every situation.