Paper 11863-2016

Uncommon Techniques for Common Variables

Christopher J. Bost, MDRC, New York, NY

ABSTRACT

If a variable occurs in more than one data set being merged, the last value (from the variable in the right-most data set listed on the MERGE statement) is kept. It is critical, therefore, to know the name and sources of any common variables. This paper reviews simple techniques to note overwriting in the log, to summarize the names and sources of all variables, and to identify common variables before merging files.

INTRODUCTION

It is simple to merge SAS® data sets. A DATA step with a MERGE statement and a BY statement is used to perform a match-merge. The only requirement is that observations in all input data sets are sorted with PROC SORT, indexed, or already in sort order by values of the BY variable(s).

If the data sets have common variables (other than the BY variable[s]), values are overwritten. It is better to know if there are common variables and pre-process data as needed. The following sections describe how to use options, PROC FREQ, and PROC SQL to identify common variables before files are merged.

CHECKING FOR COMMON VARIABLES WITH OPTIONS

The option MSGLEVEL=I can be used to include information in the log when variables will be overwritten. Consider the following example that merges three SAS data sets: DS1, DS2, and DS3.

DS1 contains variables id, a, b, and c. DS2 contains variables ID, B, C, and D. DS3 contains variables id, d, e, and f. The three data sets can be sorted by ID and merged. Variables b and c in data set DS1 will be overwritten by variables B and C in data set DS2. Variable D in data set DS2 will be overwritten by variable d in data set DS3:

```
proc sort data=ds1;
by id;
run;

proc sort data=ds2;
by id;
run;

proc sort data=ds3;
by id;
run;

options msglevel=i;
data ds123;
merge ds1 ds2 ds3;
by id;
run;
```

The following is included in the log:

```
INFO: The variable b on data set WORK.DS1 will be overwritten by data set WORK.DS2.
INFO: The variable c on data set WORK.DS1 will be overwritten by data set WORK.DS2.
INFO: The variable D on data set WORK.DS2 will be overwritten by data set WORK.DS3.
```

Output 1. Information included in the log by MSGLEVEL=I.

This details the variables that will be overwritten. If it was intended, this information is documentation. If it was not intended, this information documents a problem. If the resulting data set is not what was needed, these data sets were also sorted and merged unnecessarily. This could be "costly" with larger data sets.

It is possible to check if any variables will be overwritten without sorting and without merging observations. For example:

```
options msglevel=i obs=0;
data _null_;
merge ds1 ds2 ds3;
by id;
run;
options msglevel=n obs=max;
```

The OPTIONS statement sets MSGLEVEL=I for information about any variables that will be overwritten. It also sets OBS=0 to read and write 0 observations.

The DATA _NULL_; statement tells SAS to run a DATA step but not create a data set.

The MERGE statement specifies data sets DS1, DS2, and DS3.

The BY statement specifies to merge observations by ID.

The RUN; statement ends the step.

The second OPTIONS statement sets MSGLEVEL=N to print only the usual errors, warnings, and notes. It also sets OBS=MAX to restore reading and writing all observations.

The information included in the log is the same as Output 1:

```
INFO: The variable b on data set WORK.DS1 will be overwritten by data set WORK.DS2.
INFO: The variable c on data set WORK.DS1 will be overwritten by data set WORK.DS2.
INFO: The variable D on data set WORK.DS2 will be overwritten by data set WORK.DS3.
```

Output 2. Information included in the log by MSGLEVEL=I.

"Pros" of this technique:

- It is fast and efficient. Only descriptor information is combined.
- Any variable that will be overwritten is documented in the log.

There are no real "cons" to this technique. It is advisable to use MSGLEVEL=I when merging data sets.

CHECKING FOR COMMON VARIABLES WITH PROC FREQ

The view SASHELP.VCOLUMN can be used to identify common variables. It retrieves information about variables in all SAS data sets in currently defined libraries. Run PROC CONTENTS on the view to display its structure:

```
proc contents data=sashelp.vcolumn;
run;
```

The results are:

Data Set Name	SASHELP.VCOLUMN	Observations	.
Member Type	VIEW	Variables	18
Engine	SQLVIEW	Indexes	0
Created	06/25/2015 01:24:08	Observation Length	523
Last Modified	06/25/2015 01:24:08	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	Default		
Encoding	Default		

Output 3a. PROC CONTENTS of SASHELP.VCOLUMN.

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Flags	Label
10	format	Char	49	P--	Column Format
12	idxusage	Char	9	P--	Column Index Type
11	informat	Char	49	P--	Column Informat
9	label	Char	256	P--	Column Label
6	length	Num	8	P--	Column Length
1	libname	Char	8	P--	Library Name
2	memname	Char	32	P--	Member Name
3	memtype	Char	8	P--	Member Type
4	name	Char	32	P--	Column Name
15	notnull	Char	3	P--	Not NULL?
7	npos	Num	8	P--	Column Position
16	precision	Num	8	P--	Precision
17	scale	Num	8	P--	Scale
13	sortedby	Num	8	P--	Order in Key Sequence
18	transcode	Char	3	P--	Transcoded?
5	type	Char	4	P--	Column Type
8	varnum	Num	8	P--	Column Number in Table
14	xtype	Char	12	P--	Extended Type

Output 3b. PROC CONTENTS of SASHELP.VCOLUMN (continued).

Run PROC PRINT on SASHELP.VCOLUMN and include select variables and observations for inspection:

```
proc print data=sashelp.vcolumn;
var libname memname name;
where libname='WORK' and memname in ('DS1','DS2','DS3');
run;
```

The PROC PRINT statement starts the procedure.

The VAR statement specifies to print LIBNAME (libref), MEMNAME (data set name), and NAME (variable).

The WHERE statement limits observations to those for variables in data sets DS1, DS2, and DS3 in the WORK library. Note that values of LIBNAME and MEMNAME are stored in uppercase. The results are:

Obs	libname	memname	name
1	WORK	DS1	id
2	WORK	DS1	a
3	WORK	DS1	b
4	WORK	DS1	c
5	WORK	DS2	ID
6	WORK	DS2	B
7	WORK	DS2	C
8	WORK	DS2	D
9	WORK	DS3	id
10	WORK	DS3	d
11	WORK	DS3	e
12	WORK	DS3	f

Output 4. PROC PRINT of select variables and observations in SASHELP.VCOLUMN.

Note that there is one observation per variable in each data set and that variable names are in mixed case.

Document the names and sources of all variables with a crosstab of NAME and MEMNAME:

```
proc freq data=sashelp.vcolumn order=formatted;
tables name*memname/nopercent norow nocol;
format name $upcase.;
where libname='WORK' and memname in ('DS1','DS2','DS3');
run;
```

The PROC FREQ statement starts the procedure. ORDER=FORMATTED orders results by the formatted values of variables.

The TABLES statement specifies a crosstab of NAME (variable name) and MEMNAME (data set name). The NOPERCENT, NOROW, and NOCOL options after the slash suppress printing any percentages.

The FORMAT statement associates the $UPCASE. format with NAME to treat values as uppercase.

The WHERE statement limits observations to those for variables in data sets DS1, DS2, and DS3 in the WORK library. The results are:

```
The FREQ Procedure

Table of name by memname

name(Column Name)     memname(Member Name)

Frequency|DS1     |DS2     |DS3     |  Total
---------+--------+--------+--------+
A        |       1 |       0 |       0 |       1
---------+--------+--------+--------+
B        |       1 |       1 |       0 |       2
---------+--------+--------+--------+
C        |       1 |       1 |       0 |       2
---------+--------+--------+--------+
D        |       0 |       1 |       1 |       2
---------+--------+--------+--------+
E        |       0 |       0 |       1 |       1
---------+--------+--------+--------+
F        |       0 |       0 |       1 |       1
---------+--------+--------+--------+
ID       |       1 |       1 |       1 |       3
---------+--------+--------+--------+
Total            4         4         4        12
```

Output 5. PROC FREQ crosstab of NAME and MEMNAME in SASHELP.VCOLUMN.

In each table cell, 1 indicates a variable is in a data set and 0 indicates a variable is not in a data set.

If the value of Total for any row is greater than 1, it means that variable is in more than one data set.

“Pros” of this technique:

- It is “metadata driven.”
- It uses a single PROC FREQ step.
- It documents variables in any number of data sets.

There are no real “cons” to this technique, although it can produce a lot of output when there is a large number of variables.

CHECKING FOR COMMON VARIABLES WITH PROC SQL

The PROC SQL DICTIONARY table DICTIONARY.COLUMNS can be used to identify common variables. It has information about variables in all SAS data sets in currently defined libraries. Use DESCRIBE TABLE to display its structure:

```
proc sql;
describe table dictionary.columns;
quit;
```

Results are printed in the log:

```
40         describe table dictionary.columns;
NOTE: SQL table DICTIONARY.COLUMNS was created like:

create table DICTIONARY.COLUMNS
  (
   libname char(8) label='Library Name',
   memname char(32) label='Member Name',
   memtype char(8) label='Member Type',
   name char(32) label='Column Name',
   type char(4) label='Column Type',
   length num label='Column Length',
   npos num label='Column Position',
   varnum num label='Column Number in Table',
   label char(256) label='Column Label',
   format char(49) label='Column Format',
   informat char(49) label='Column Informat',
   idxusage char(9) label='Column Index Type',
   sortedby num label='Order in Key Sequence',
   xtype char(12) label='Extended Type',
   notnull char(3) label='Not NULL?',
   precision num label='Precision',
   scale num label='Scale',
   transcode char(3) label='Transcoded?'
  );

41         quit;
```

Output 6. DESCRIBE TABLE output for DICTIONARY.COLUMNS.

Run a query on DICTIONARY.COLUMNS and select columns and rows for inspection:

```
proc sql;
select libname,memname,name
from dictionary.columns
where libname='WORK' and memname in ('DS1','DS2','DS3');
quit;
```

The PROC SQL statement starts the procedure.

The SELECT clause selects columns LIBNAME (libref), MEMNAME (data set name), and NAME (variable).

The FROM clause reads rows from DICTIONARY.COLUMNS.

The WHERE clause limits rows to those for columns in tables DS1, DS2, and DS3 in the WORK library. Note that values of LIBNAME and MEMNAME are stored in uppercase. The results are:

```
Library    Member      Column
Name       Name        Name
--------------------------------
WORK       DS1         id
WORK       DS1         a
WORK       DS1         b
WORK       DS1         c
WORK       DS2         ID
WORK       DS2         B
WORK       DS2         C
WORK       DS2         D
WORK       DS3         id
WORK       DS3         d
WORK       DS3         e
WORK       DS3         f
```

Output 7. PROC SQL query of select columns and rows in DICTIONARY.COLUMNS.

Note that there is one row per column in each table and that column names are in mixed case.

Run a query that selects LIBNAME (libref), MEMNAME (data set name), and NAME (variable) when NAME occurs more than once:

```
proc sql;
select libname,memname,upcase(name) as name
from dictionary.columns
where libname='WORK' and memname in ('DS1','DS2','DS3')
group by upcase(name)
having count(*) > 1
order by name,memname;
quit;
```

The SELECT clause converts values of NAME to uppercase.

The GROUP BY clause groups rows by uppercase values of NAME.

The HAVING clause post-processes each group and includes it (i.e., all rows in that group) in results when there is more than one row (i.e., the variable occurs in more than one table).

The ORDER BY clause sorts results by NAME and MEMNAME.

The results are:

```
Library   Member
Name      Name      name
--------------------------------------------------
WORK      DS1       B
WORK      DS2       B
WORK      DS1       C
WORK      DS2       C
WORK      DS2       D
WORK      DS3       D
WORK      DS1       ID
WORK      DS2       ID
WORK      DS3       ID
```

Output 8. PROC SQL query of NAME values that occur more than once.

Variables B, C, D, and ID occur in more than one data set. Variables B and C occur in DS1 and DS2. Variable D occurs in DS2 and DS3. Variable ID occurs in DS1, DS2, and DS3.

Query results can be saved in a SAS data set. Document the names and sources of all variables that occur in more than one data set with a crosstab of NAME and MEMNAME:

```
proc sql;
create table common as
select libname,memname,upcase(name) as name
from dictionary.columns
where libname='WORK' and memname in ('DS1','DS2','DS3')
group by upcase(name)
having count(*) > 1;
quit;

proc freq data=common;
tables name*memname/nopercent norow nocol;
run;
```

The CREATE TABLE clause saves query results in data set COMMON.

The PROC FREQ step processes data set COMMON.

The results are:

```
The FREQ Procedure

Table of name by memname

name                    memname(Member Name)

Frequency|DS1      |DS2      |DS3      |  Total
---------+--------+--------+--------+
B        |      1 |      1 |      0 |      2
---------+--------+--------+--------+
C        |      1 |      1 |      0 |      2
---------+--------+--------+--------+
D        |      0 |      1 |      1 |      2
---------+--------+--------+--------+
ID       |      1 |      1 |      1 |      3
---------+--------+--------+--------+
Total            3         4         2         9
```

Output 9. PROC FREQ crosstab of NAME and MEMNAME (common variables only).

This crosstab documents the names and sources of variables that occur in more than one data set.

“Pros” of this technique:

- It documents only “problematic” (common) variables and sources.
- Data can be pre-processed as desired (e.g., drop the ID variable).

The are no real “cons” to this technique, but it requires PROC SQL syntax that might be new to some.

CONCLUSION

Inspect data sets for common variables before merging. Use MSGLEVEL=I to note variables that will be overwritten in the log. Use PROC FREQ and SASHELP.VCOLUMN to summarize variables and sources. Use PROC SQL, DICTIONARY.COLUMNS, and PROC FREQ to summarize variables that occur more than once.

Metadata about variables and sources can be summarized with PROC TABULATE and PROC REPORT. See the appendix for additional examples.

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

Metadata about variables and sources can be summarized with more than one procedure. Results can be formatted different ways. Consider the following examples.

PROC FREQ WITH RENAMED VARIABLES

NAME and MEMNAME can be renamed to something more “intuitive” (e.g., Variable and Dataset) in the crosstab. Variable labels can be suppressed. Results can be output to Rich Text Format:

```
proc freq data=sashelp.vcolumn(rename=(name=Variable memname=Dataset))
  order=formatted;
tables variable*dataset/nopercent norow nocol;
format variable $upcase.;
attrib _all_ label=' ';
where libname='WORK' and dataset in ('DS1','DS2','DS3');
run;
```

The results are:

Table of Variable by Dataset				
Variable	**Dataset**			
Frequency	**DS1**	**DS2**	**DS3**	**Total**
A	1	0	0	1
B	1	1	0	2
C	1	1	0	2
D	0	1	1	2
E	0	0	1	1
F	0	0	1	1
ID	1	1	1	3
Total	4	4	4	12

Output A1. PROC FREQ crosstab with renamed variables.

PROC TABULATE WITH A SYMBOL

The 1s and 0s under DS1, DS2, and DS3 in Output A1 are frequencies. PROC TABULATE can count frequencies and format the results with a special character:

```
ods escapechar='^';
proc format;
value bullet .=' '
              1='^{unicode 25AA}'; *25AA is Unicode for Black Small Square;
run;

proc tabulate data=sashelp.vcolumn order=formatted style=[just=center];
class name memname;
format name $upcase.;
table name=' ',
      memname=' '*n=' '*format=bullet.
      /box='Variable';
where libname='WORK' and memname in ('DS1','DS2','DS3');
run;
```

ODS ESCAPECHAR= specifies the inline formatting symbol used to enable special formatting.

The PROC FORMAT statement starts the procedure.

The VALUE statement:

- Creates a temporary numeric format named BULLET.
- Labels missing values with a blank space (i.e., nothing).
- Labels values of 1 with the Unicode character 25AA (▪).

Specify a Unicode character starting with the previously assigned ODS ESCAPECHAR, a left brace, the word UNICODE, a space, the value for the respective character, and a right brace.

Under Windows, click the **Start** button and search for "character map" to see all of the symbols that can be displayed. Click on any symbol to display its Unicode value at the bottom. Alternately, use a search engine to find lists of Unicode characters and values.

The RUN; statement ends the FORMAT procedure.

The PROC TABULATE statement starts the procedure. ORDER=FORMATTED orders results by the formatted values of variables. STYLE=[JUST=CENTER] centers the results in table cells.

The CLASS statement specifies using values of NAME and MEMNAME to group data.

The FORMAT statement associates the $UPCASE. format with NAME to treat values as uppercase.

The TABLE statement:

- Specifies NAME in the row dimension.
- Specifies MEMNAME*N in the column dimension.
- Formats values of N with the BULLET. format.
- Uses =' ' after NAME, MEMNAME, and N to suppress column headers.
- Uses /BOX= to print 'Variable' in the upper left box of the table. (It is blank by default.)

The WHERE statement limits observations to those for variables in data sets DS1, DS2, and DS3 in the WORK library. Note that values of LIBNAME and MEMNAME are stored in uppercase.

The RUN; statement ends the TABULATE procedure.

The results are:

Variable	DS1	DS2	DS3
A	▪		
B	▪	▪	
C	▪	▪	
D		▪	▪
E			▪
F			▪
ID	▪	▪	▪

Output A2. PROC TABULATE table with symbol.

PROC REPORT WITH TRAFFIC LIGHTING

The symbols in Output A2 help make an effective visual. It is necessary to scan each row, however, to see which variable occurs in more than one table. PROC REPORT can count frequencies, format the results with a special character, and highlight rows that meet specified criteria:

```
ods escapechar='^';
proc format;
value bullet .=' '
              1='^{unicode 25AA}'; *25AA is Unicode for Black Small Square;
run;

proc report data=sashelp.vcolumn;
where libname='WORK' and memname in ('DS1','DS2','DS3');
column name memname,n total;
define name/group ' ' format=$upcase. order=formatted style=[font_weight=bold];
define memname/across ' ';
define n/' ' format=bullet. center;
define total/computed noprint;
compute total;
  if sum(_c2_,_c3_,_c4_)>1 then do;
    call define(_row_,'style','style=[background=lightyellow]');
  end;
endcomp;
run;
```

ODS ESCAPECHAR= and the PROC FORMAT step are the same as in the PROC TABULATE example.

The PROC REPORT statement starts the procedure.

The WHERE statement limits observations to those for variables in data sets DS1, DS2, and DS3 in the WORK library. Note that values of LIBNAME and MEMNAME are stored in uppercase.

The COLUMN statement specifies the arrangement of columns in the report.

The DEFINE statements specify how to use and display report items:

- NAME is a group variable. Uppercase values will be in a bold font (like column headings).
- MEMNAME is an across variable. Each of its values will get its own column in the report.
- N is a statistic to calculate. Values will be formatted with the BULLET. format and centered.
- TOTAL is a computed variable. Its values will be calculated, but not printed, in the report.
- Note that all column headers are suppressed by labeling the variables with a blank (‘ ’).

The COMPUTE statement begins a compute block:

- It contains programming statements that PROC REPORT will execute as it builds the report.
- IF the sum of columns 2, 3, and 4 in a row (i.e., values for DS1, DS2, and DS3) is greater than 1 (i.e., a variable occurs in more than one data set), THEN style the row with a light yellow background.
- The ENDCOMP; statement ends the compute block.

The results are:

	DS1	DS2	DS3
A	▪		
B	▪	▪	
C	▪	▪	
D		▪	▪
E			▪
F			▪
ID	▪	▪	▪

Output A3. PROC REPORT table with traffic lighting.