Automatically Renaming Common Variables Before Merging  
Christopher J. Bost, MDRC, New York, NY

ABSTRACT
SAS® merges observations based on values of a common BY variable. If there are additional common variables, SAS saves only the last values encountered. This paper shows how to automatically rename common variables with a user-specified prefix to preserve the values of all variables. PROC SQL and DICTIONARY.COLUMNS are used to create a macro variable that stores renaming information.

INTRODUCTION
When SAS merges two data sets that contain a variable with the same name, the output data set contains one variable with that name and the last value encountered for that variable. To keep all values in the input data sets, any common variables must be renamed.

SAMPLE DATA SETS
Two SAS data sets are used in this paper, ONE and TWO:

<table>
<thead>
<tr>
<th>Data set ONE</th>
<th>Data set TWO</th>
</tr>
</thead>
<tbody>
<tr>
<td>id a b c</td>
<td>id b C d</td>
</tr>
<tr>
<td>1 1 2 3</td>
<td>1 22 33 1</td>
</tr>
<tr>
<td>2 4 5 6</td>
<td>2 55 66 2</td>
</tr>
<tr>
<td>3 7 8 9</td>
<td>3 88 99 3</td>
</tr>
</tbody>
</table>

Data sets ONE and TWO each contain four variables and three observations. There is one observation for each ID (i.e., no duplicates). Observations are sorted in ascending order by ID. Each observation in data set ONE matches one observation in data TWO by ID. Note that data sets ONE and TWO both contain variables named ID as well as B and C.

MATCH-MERGING WITH COMMON VARIABLES
Use a match-merge to combine observations in data sets ONE and TWO by ID:

```sas
data onetwo;
  merge one two;
  by id;
run;
```

The resulting data set looks as follows:

<table>
<thead>
<tr>
<th>Data set ONETWO</th>
</tr>
</thead>
<tbody>
<tr>
<td>id a b c d</td>
</tr>
<tr>
<td>1 1 22 33 1</td>
</tr>
<tr>
<td>2 4 55 66 2</td>
</tr>
<tr>
<td>3 7 88 99 3</td>
</tr>
</tbody>
</table>

Data set ONETWO contains five variables and three observations. It contains all variables in data sets ONE and TWO. Variables occur once in the order in which they were first encountered in data set ONE and data set TWO.
Values are saved from the data set in which each variable was last encountered. In other words, only the values of variables B and C from data set TWO were preserved. This might not be the desired output data set.

No error, warning, or note is written to the SAS log by default. The system option MSGLEVEL can be set to print additional information. For example:

```sas
options msglevel=i;
```

When this OPTIONS statement is executed before the preceding DATA step, SAS writes the following to the log:

```
INFO: The variable b on data set WORK.ONE will be overwritten by data set WORK.TWO.
INFO: The variable c on data set WORK.ONE will be overwritten by data set WORK.TWO.
```

Note, however, that these messages are only informational. The resulting data set is identical.

**RENAME= DATA SET OPTION**

To preserve all values in both input data sets, use the RENAME= data set option to rename common variables in either data set before merging observations:

```sas
data onetwo;
  merge one two(rename=(b=TWOb C=TWOC));
  by id;
run;
```

The old=new pairs of old and new variable names are specified in parentheses after the RENAME= data set option. RENAME= is executed before the merge. The resulting data set looks as follows:

<table>
<thead>
<tr>
<th>Data set ONETWO</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Data set ONETWO contains seven variables and three observations. It contains all variables in data sets ONE and TWO. The variables B and C in data set TWO were renamed TWOb and TWOC, respectively. All variables have unique names. All values were preserved. This is the desired output data set.

Note that this approach requires renaming common variables “manually” (i.e., all renaming pairs must be hard-coded). Merging data sets with large numbers of variables can make renaming common variables time-consuming and error-prone. A DICTIONARY table can be used to identify common variables and generate the syntax to rename variables.

**DICTIONARY TABLES**

PROC SQL retrieves information about all SAS datasets (among other things) and stores it in special read-only “DICTIONARY tables.” SAS automatically assigns the libref DICTIONARY for these tables.

DICTIONARY.COLUMNS stores information about the variables in all data sets. To display the structure of this dictionary table, use the DESCRIBE TABLE statement:

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

The DESCRIBE TABLE statement displays the CREATE TABLE statement that would produce the specified table’s structure. SAS writes the following to the log:
NOTE: SQL table DICTIONARY.COLUMNS was created like:

```sql
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'
);
```

Each line in the CREATE TABLE statement contains a variable (column) name, the variable type (char[acter] or num[eric]), the number of bytes used in parentheses (for character variables), and the variable label.

To list the names of variables in a data set, query DICTIONARY.COLUMNS using a WHERE clause that specifies the LIBNAME and MEMNAME of the data set. For example:

```sql
proc sql;
select libname, memname, name
from dictionary.columns
where libname='WORK' and memname='ONE';
quit;
```

The query results are:

<table>
<thead>
<tr>
<th>Library Name</th>
<th>Member Name</th>
<th>Column Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK</td>
<td>ONE</td>
<td>id</td>
</tr>
<tr>
<td>WORK</td>
<td>ONE</td>
<td>a</td>
</tr>
<tr>
<td>WORK</td>
<td>ONE</td>
<td>b</td>
</tr>
<tr>
<td>WORK</td>
<td>ONE</td>
<td>c</td>
</tr>
</tbody>
</table>

Note that values of LIBNAME (“Library Name”) and MEMNAME (“Member Name”) are stored in uppercase. These values must be specified in uppercase on the WHERE clause.

NAME (“Column Name”) values have the case used when SAS originally created the variable.
USING DICTIONARY.COLUMNS TO RENAME VARIABLES

DICTIONARY.COLUMNS contains all the information needed to rename the common variables in two data sets. One PROC SQL step can query DICTIONARY.COLUMNS, select common variables in the two data sets, and store renaming information in a macro variable.

First, select the uppercase values of variable names from data set ONE with the exception of ID, the variable on which observations are merged:

```
select upcase(name)
from dictionary.columns
where libname='WORK' and memname='ONE' and upcase(name) ne 'ID';
```

The query results are:

<table>
<thead>
<tr>
<th>Column Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
</tbody>
</table>

Second, select variable names from data set TWO when their uppercase values are in the list of values generated by the previous query:

```
select name
from dictionary.columns
where libname='WORK' and memname='TWO' and upcase(name) in ( previous query )
```

The query results are:

<table>
<thead>
<tr>
<th>Column Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
</tr>
<tr>
<td>C</td>
</tr>
</tbody>
</table>

Note that the UPCASE function is used to compare uppercase values of variable names. Variables named c and C are identical to SAS but will not be selected by the above query without UPCASE. Ultimately, values of NAME (in their original case) are selected when the uppercase values of NAME in data sets ONE and TWO match.

Third, select each common variable name generated by the previous query, followed by an equals sign (=), the prefix 'TWO', and the variable name. Use the TRIM function to remove trailing blanks. Use the concatenation operator (||) to connect character strings:

```
select trim(name) || '=' || 'TWO' || name
from ( previous query )
```

The query results are:

<table>
<thead>
<tr>
<th>_______________</th>
</tr>
</thead>
<tbody>
<tr>
<td>b=TWOb</td>
</tr>
<tr>
<td>C=TWOc</td>
</tr>
</tbody>
</table>

Finally, assign the query results to a macro variable with the INTO clause:

```
into :renamelist separated by ' '
```

The query results are stored in the macro variable RENAMELIST, separated by blanks.
The above three queries can be combined into one statement. The first query becomes the inner subquery. The second query becomes the outer subquery. The third query becomes the outer query:

```sql
proc sql noprint;
select trim(name) || ' = ' || 'TWO' || name
into :renamelist separated by ' ' from
(select name
  from dictionary.columns
  where libname='WORK' and memname='TWO' and upcase(name) in
    (select upcase(name)
      from dictionary.columns
      where libname='WORK' and memname='ONE' and upcase(name) ne 'ID'));
quit;
```

The SQL subqueries are nested in parentheses. Processing starts with the innermost subquery. The inner subquery returns values used by the outer subquery. The outer subquery returns values used by the outer query.

The final query results are not printed because of the NOPRINT option on the PROC SQL statement. Results are stored in macro variable RENAMELIST, however, and can be displayed with the %PUT statement:

```plaintext
%put &renamelist;
```

SAS writes the following to the log:

```
b=TWOb  C=TWOC
```

**RENAME= DATA SET OPTION REVISITED**

Use the RENAME= data set option to rename common variables in data set TWO before merging observations. Reference the macro variable RENAMELIST instead of hard-coding values:

```sas
data onetwo;
  merge one two(rename=(&renamelist));
  by id;
run;
```

When this code is executed, &RENAMELIST is replaced with “b=TWOb C=TWOC” (i.e., without the quotes). The resulting data set looks as follows:

<table>
<thead>
<tr>
<th>Data set ONETWO</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Variables in data set TWO that also exist in data set ONE were renamed with the prefix TWO. The values of all variables were preserved. This is the desired output data set.

A macro to rename common variables before merging is included in the Appendix.

**CONCLUSION**

PROC SQL can be used to query DICTIONARY.COLUMNS, select common variables in two data sets, and store renaming information in a macro variable. This macro variable can be referenced in a RENAME= data set option before merging the two data sets to preserve all values.
REFERENCES

CONTACT INFORMATION
Christopher J. Bost
Director, Research Technology Unit
MDRC
16 East 34th Street, 19th Floor
New York, NY 10016
(212) 340-8613 telephone
(212) 684-0832 fax
christopher.bost@mdrc.org
www.mdrc.org

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.
APPENDIX
A macro can be used to rename common variables before merging:

```sas
%macro MergeRename(inlib1=work, /* libref for input data set 1 */
    inds1=,          /* name of input data set 1    */
    inlib2=work,     /* libref for input data set 2 */
    inds2=,          /* name of input data set 2    */
    outlib=work,     /* libref for output data set */
    outds=,          /* name of output data set     */
    byvar=,          /* name(s) of BY variable(s)   */
    prefix=);        /* prefix to use when renaming */

proc sql noprint;
select trim(name)||'='||"&prefix"||name
into :renamelist separated by ' ' from
(select name
    from dictionary.columns
    where libname=upcase("&inlib2") and memname=upcase("&inds2")
    and upcase(name) in
        (select upcase(name)
            from dictionary.columns
            where libname=upcase("&inlib1") and memname=upcase("&inds1")
            and not indexw(upcase("&byvar"),upcase(name))
        )
));
quit;

data &outlib..&outds;
merge &inlib1..&inds1
    &inlib2..&inds2(rename=(&renamelist));
by &byvar;
%mend MergeRename;
```

Sample macro call:

```sas
%MergeRename(inlib1 = work,
    inds1 = one,
    inlib2 = work,
    inds2 = two,
    outlib = work,
    outds = onetwo,
    byvar = id,
    prefix = TWO)
```

INLIB1, INLIB2, and OUTLIB default to WORK if not specified. All other macro parameters are required.

BYVAR may include more than one variable. Variable lists (e.g., ID1-ID3), however, may not be used.

Observations in both input data sets must be in sort order, sorted with PROC SORT, or indexed by the variable(s) on the BYVAR parameter. Values must be in ascending order.

The combination of the PREFIX value and a common variable name must not duplicate another variable name or exceed 32 characters.

Additional DATA step statements may follow the macro call and will be included in the “merge” DATA step.

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