**Paper DM11**

# Preventing Data Loss when Combining SAS Datasets

John Cantrell, University of North Carolina, Chapel Hill, NC


## ABSTRACT:

When using the SAS ® data step to combine SAS datasets, unanticipated data loss can occur when the different input datasets contain character variables with identical names and different lengths. When combining datasets by matched merge, concatenation, or interleaving, identically named character variables not included in a by statement are not checked for length consistency. The length of the resulting character variable in the output dataset may be the smaller of the two lengths from the input datasets, resulting in data loss through truncation. This data loss occurs without errors, warnings, or notes being written to the SAS log. The discussion of the data loss problem includes:

1. a SAS program with associated log and output demonstrating how data loss occurs
2. a SAS macro and techniques using data step code that can be used to prevent data loss.


## INTRODUCTION:

SAS offers versatile and powerful tools for combining multiple datasets, including PROC SQL and the SAS datastep. These tools for dataset combination are both very useful and widely used, probably accounting for a significant part of the success of SAS.

The datastep can be used to combine datasets by match-merging, one-to-one merging, interleaving, and concatenation (SAS Institute, Inc., 1995, SAS Institute, Inc., 2005). These four types of dataset combination, when accomplished through the datastep, are vulnerable to data loss in character variables. This paper illustrates how this data loss occurs and shows how to avoid it using a simple SAS macro.

## THE PROBLEM: DATA LOSS

The SAS data step can combine datasets by match-merging, one-to-one merging, interleaving, and concatenation. Match-merging uses a MERGE and a BY statement. One-to-one merging uses a MERGE statement. Concatenation uses a SET statement. Interleaving uses a SET and a BY statement.

When combining datasets using these techniques, SAS provides notes, warnings, and error messages in the log, alerting programmers to problems associated with dataset combination. Additional potential problems associated with dataset combination are discussed in SAS User Group Proceedings (Foley, 1997, 1998). Even with all the attention focused on issues surrounding data set combination, one problem has apparently gone unnoticed.

The problem could involve combination of any number of datasets, but for the sake of simplicity, I will discuss the combination of two datasets.

The examples below can be reproduced by running the code in Appendix A.

### MATCH-MERGING AND ONE-TO-ONE MERGING:

The problem occurs analogously in both match-merging and one-to-one merging. This example will discuss match-merging, but, the concepts also apply to one-to-one merging.

Imagine that two data sets are combined by one of the four methods mentioned above, that both datasets contain a character variable named “var,” and that var is not used in a BY statement. The input datasets, the datasets that we are combining, are named in either a MERGE or a SET statement.

If the two instances of the variable var are of different lengths, what will be the length of the variable var in the output dataset? The answer is: the first dataset mentioned in the MERGE or SET statement determines the length of var in the output dataset. In the example below, a MERGE statement names two datasets, “two” and “three,” that will be merged by matching values of a variable named “id”:

```
data twothree;
  merge two three;
  by id;
run;
```

| The Input Datasets | | | |
|---|---|---|---|
| dataset two | | dataset three | |
| Id | var | Id | Var |
| 1 | 21 | 0 | 3ks |
| 2 | 23 | 1 | 31a |
| 3 | 24 | 2 | 35a |
| 4 | 25 | 3 | 34a |
| 5 | 27 | 4 | 350 |
| 6 | 2g | 8 | 370 |
| 7 | 2k | 9 | 3gv |

The preceding code and input datasets produce output dataset twothree, as follows:

| dataset twothree | |
|---|---|
| Id | var |
| 0 | 3k |
| 1 | 31 |
| 2 | 35 |
| 3 | 34 |
| 4 | 35 |
| 5 | 27 |
| 6 | 2g |
| 7 | 2k |
| 8 | 37 |
| 9 | 3g |

Note that in the output dataset twothree, var has a value of “31” where id=1. This result was obtained because dataset “two” and dataset “three” both contain a character variable named “var”. The values from the dataset named last (in this case, dataset “three”) overwrote values from the dataset named first (in this case, dataset “two”). When SAS finds a value of “id” that is the same in datasets “two” and “three”, for example id=1, the merge statement produces a row of data in the output dataset with the matched value of “id” and a value for “var” from dataset “three”, which was “31a”.

It happens that variable “var” in dataset “two” has a length of two and that variable “var” in dataset “three” has a length of three. Since the dataset named first in the MERGE statement controls the length of the character variable in the output dataset, the output dataset will contain a variable “var” of length two. While values from “three” that are three characters long are overwriting values from “two” that are two characters long, the results are being stored in a variable “var” in the output dataset that is only two characters long. The result of putting three characters of information, such as “31a” into a two-character variable is truncation or data loss, which is why in dataset twothree, for id=1, the value of var is “31”, not “31a”.

**INTERLEAVING AND CONCATENATION:**

The data loss problem described above also occurs when interleaving and concatenating datasets. This example will discuss interleaving, but, the concepts also apply to concatenation.

Similar to the match-merging of datasets two and three above, if we interleave datasets two and three with the code below:

```
data twothree;
  set two three;
  by id;
run;
```

| The Input Datasets | | | |
|---|---|---|---|
| dataset two | | dataset three | |
| Id | var | Id | Var |
| 1 | 21 | 0 | 3ks |
| 2 | 23 | 1 | 31a |
| 3 | 24 | 2 | 35a |
| 4 | 25 | 3 | 34a |
| 5 | 27 | 4 | 350 |
| 6 | 2g | 8 | 370 |
| 7 | 2k | 9 | 3gv |

The preceding code and input datasets produce output dataset twothree, as follows:

| dataset twothree | |
|---|---|
| Id | var |
| 0 | 3k |
| 1 | 21 |
| 1 | 31 |
| 2 | 23 |
| 2 | 35 |
| 3 | 24 |
| 3 | 34 |
| 4 | 25 |
| 4 | 35 |
| 5 | 27 |
| 6 | 2g |
| 7 | 2k |
| 8 | 37 |
| 9 | 3g |

As in the previous example of match-merging, values from input dataset three have lost a character in output dataset twothree because they were stored in a variable that is only two characters long.

**THE LOG:**

In both cases, examining the log reveals no indication that data were lost (using SAS Version 9.1). What can be done to correct the problem?

## THE SOLUTION: GOOD CODING PRACTICES AND MACROS

Many professionals who use SAS adhere to coding standards that will help programmers to avoid some of these problems (Foley, 1997, 1998). While such coding standards are a good start, the macro verifyVariables will find potential data truncation problems in datasets. VerifyVariables, shown in Appendix B below, will write error message to the SAS log, alerting programmers to possible data loss due to inconsistent character variable lengths.

**MATCH-MERGING AND ONE-TO-ONE MERGING:**

In the cases of match-merging and one-to-one merging many SAS programmers adhere to coding standards that require the re-naming of variables (except for those in a BY statement) to avoid identically named variables in input datasets (Foley, 1997).

The datastep below renames the variable “var” to “varTwo”. Since renaming this variable eliminates identically named variables in the two datasets, values from dataset “three”, variable “var” are not affected by values or size attributes from dataset “two”, variable “varTwo,” eliminating data loss through truncation.

```
data two;
  set two (rename (var=varTwo));
run;
```

If, in a special case, it is advantageous to not change the names of same-named character variables in two datasets to be combined, the lengths of these character variables should be the same. If sameness of length requires a change in the length of one of the variables, the length of the smaller variable should be increased.

The datastep below re-sizes the variable “var” from two characters to three characters long. Renaming the old variable and dropping it is necessary because SAS will not change the size of an existing variable. To “re-size” the variable the programmer must re-name the original variable, create a new variable with the same name as the original variable and a different (correct) length, copy the data from the old variable into the new variable, and then drop the old variable.

```
data two (drop=varOld);
  set two (rename (var=varOld));
  length var $3.;
  var=varOld;
run;
```

To use the macro verifyVariables to prevent data loss, run the macro before the datastep that merges datasets two and three, as follows:

```
%verifyVariables(two,three,merge);
data twothree;
  merge two three;
  by id;
run;
```

This invocation of verifyVariables checks datasets two and three for variables with the same name in both datasets, and for character variables with the same name but differing lengths, issuing an error message to the log if either condition is found.

**CONCATENATION:**

In the case of concatenation or interleaving re-size character variable lengths so that identically named variables have identical lengths, thereby avoiding data loss. An example of how to re-size variables is shown in the preceding section entitled "Match-Merging and One-to-One Merging."

To use the macro verifyVariables to prevent data loss, run the macro before the datastep that interleaves datasets two and three, as follows:

```
%verifyVariables(two,three,set);
data twothree;
  set two three;
  by id;
run;
```

This invocation of verifyVariables checks datasets two and three for character variables with the same name but differing lengths, issuing an error message to the log if this condition is found.

## CONCLUSION:

SAS offers versatile and powerful tools for combining multiple datasets, including MERGE and SET statements within the SAS datastep. However, these methods may yield unintended and unwanted results, including data loss through truncation. Data loss through truncation can be avoided by using appropriate coding standards and by running the macro verifyVariables to check datasets before attempting to combine them. Perhaps SAS will address this type of data checking in a future release, rendering code such as verifyVariables obsolete.

## APPENDIX A:

The following program will be available for download at http://www.unc.edu/~jcantrel/ on or before the beginning of the PharmaSUG05 meeting.

```
Name:    mergeTest.sas
Author:  John Cantrell
Date:    1/25/05
data two;
  length id $1. var $2.;
  input id $ var $;
  cards;
1 21
2 23
3 24
4 25
5 27
6 2g
7 2k
run;
data three;
  length id $1. var $3.;
  input id $  var $;
  cards;
1 31a
2 35a
3 34a
4 350
8 370
9 3gv
0 3ks
run;
proc sort data=two; by id; run;
proc sort data=three; by id; run;

/*matched merges*/
data twothree;
  merge two three;
  by id;
run;
data threetwo;
  merge three two;
  by id;
run;
```

```
ods listing close;
ods chtml file="C:\outputdirectory\two.xls";
proc print data=two noobs;
title "dataset two";
run;
ods chtml file="C:\outputdirectory\three.xls";
proc print data=three noobs;
title "dataset three";
run;
ods chtml file="C:\outputdirectory\mergethreetwo.xls";
proc print data=threetwo noobs;
title "dataset threetwo:  produced by merge three two; by id;";
run;
ods chtml file="C:\outputdirectory\mergetwothree.xls";
proc print data=twothree noobs;
title "dataset twothree:  produced by merge two three; by id;";
run;
ods chtml close;
ods listing;

/*concatenation*/
%verifyVariablesSet(two,three)
data twothree;
  set two three;
run;
data threetwo;
  set three two;
run;
ods listing close;
ods chtml file="C:\outputdirectory\concatthreetwo.xls";
proc print data=threetwo noobs;
title "dataset threetwo:  produced by set three two;";
run;
ods chtml file="C:\outputdirectory\concattwothree.xls";
proc print data=twothree noobs;
title "dataset twothree:  produced by set two three;";
run;
ods chtml close;
ods listing;

/*interleaving*/
data twothree;
  set two three;
  by id;
run;
data threetwo;
  set three two;
  by id;
run;
ods listing close;
ods chtml file="C:\outputdirectory\interleavthreetwo.xls";
proc print data=threetwo noobs;
title "dataset threetwo:  produced by set three two; by id;";
run;
ods chtml file="C:\outputdirectory\interleavtwothree.xls";
proc print data=twothree noobs;
title "dataset twothree:  produced by set two three; by id;";
run;
ods chtml close;
ods listing;
```

## APPENDIX B:

The following program will be available for download at http://www.unc.edu/~jcantrel/ on or before the beginning of the PharmaSUG05 meeting.

```
Name:    verifyVariables.sas
Author:  John Cantrell
Date:    1/25/05

Description:
VerifyVariables checks variables in pairs of datasets prior to dataset combination.  For
combination by concatenation or interleaving (using SET statement), each dataset in the pair
is checked to see if character variables of the same name in the two datasets are of
consistent length.  If inconsistencies in length are found, an ERROR message is written in
the log.  For combination by merging (using MERGE statement), each dataset in the pair is
first checked to see if there are any variables with the same name.  If variables with the
same name occur in both datasets, an ERROR message is written to the log.  Then each dataset
in the pair is checked to see if character variables of the same name in the two datasets
are of consistent length.  If inconsistencies in length are found, an ERROR message is
written in the log.

Inputs are macro parameters, including:
&dsn1 is input dataset name, either temporary or permanent (qualified with a libname).
&dsn2 is input dataset name, either temporary or permanent (qualified with a libname).
&statementType specifies the type  of dataset combination that the macro is checking for.
     &statementType=SET for concatenation and interleaving combination.
     statementType=MERGE for match-merging or one-to-one merging combination.

Outputs are error statements written to log concerning variable lengths.

%macro verifyVariables(dsn1,dsn2,statementType);

/*  Error checking for presence of required parameters.*/
%if &dsn1= or &dsn2= or &statementType= %then %do;
  %put ERROR:  You must supply parameter dsn1, dsn2, and statementType to macro
verifyVariables.;
  %put ERROR:  dsn1 and dsn2 are datasets (with libnames if appropriate).  statementType is;
  %put ERROR:  either:;
  %put ERROR:  "merge" or "m" if a MERGE statement follows the macro;
  %put ERROR:   or "set" or "s" if a SET statement follows the macro.;
  %goto exit;
%end;

/*Don't need to print anything*/
ods listing close;

/*Get variable list for first dataset.*/
proc contents
data=&dsn1 out=contentXVVS1 nodetails;
run;

/* Organize variable list from first dataset.  Keep only important variables.*/
data contentXVVS1 (drop=oldname);
  set contentXVVS1 (keep=name type length
rename=(name=oldname type=type1 length=length1));
  name=upcase(oldname);
run;

/* Sort by Name to prepare for merge.  */
proc sort data=contentXVVS1;
by name;
run;
```

```
/* Get variable list for second dataset.*/
proc contents
data=&dsn2 out=contentXVVS2 nodetails;
run;

/* Organize variable list from second dataset.  Keep only important variables.*/
data contentXVVS2 (drop=oldname);
  set contentXVVS2 (keep=name type length
rename=(name=oldname type=type2 length=length2));
  name=upcase(oldname);
run;

/*Sort by Name to prepare for merge.  */
proc sort data=contentXVVS2;
by name;
run;

/*Merge first and second datasets.  Keep only records where variable name is in both
datasets.  */
data bothXVVS;
  merge contentXVVS1 (in=a) contentXVVS2 (in=b);
  by name;
  if a and b;
run;
```

```
/*For dataset combination of type “SET”, check for variables with the same name.*/
/*For dataset combination of type “MERGE”, check for character variables with the */
/*same name and different lengths.  If either condition is found, write an */
/*ERROR statement to the log. */
data bothXVVS;
  set bothXVVS;
  if upcase("&statementType")="SET" or upcase("&statementType")="S" then do;
    if type1=2 and length1^=length2 then do;
      put;
  put
"ERROR:  The character variable " name " is in both datasets &dsn1 and &dsn2 but is of"
" inconsistent length.  If datasets &dsn1 and &dsn2 are combined data loss by truncation "
" may occur.  To prevent data loss re-size the smaller of the two character variables "
“named “ name " before combining datasets &dsn1 and &dsn2..";
      put ;
  _ERROR_+1;
    end;
  end;
  if upcase("&statementType")="MERGE" or upcase("&statementType")="M" then do;
    put;
put
"ERROR:  The variable " name " is in both datasets &dsn1 and &dsn2.  If variable"
name " is to be used in a BY statement following a MERGE statement, then there"
" is no problem.  However, if variable " name " is not to be used in a BY"
" statement, and if datasets &dsn1 and &dsn2 are merged, the data in one"
" of the variables named " name " will overwrite the data in the other"
" variable named " name ", causing data loss.  To avoid this data loss,"
" re-name one of the variables named " name ".";
put;
_ERROR_+1;
    if type1=2 and length1^=length2 then do;
      put;
  put
"ERROR:  The character variable " name " is in both datasets &dsn1 and &dsn2 but is of"
" inconsistent length.  If datasets &dsn1 and &dsn2 are combined and neither of the "
" variables named " name " are re-named then data loss by truncation may occur.  To "
" prevent data loss increase the size of the smaller of the two character variables “
" named " name " so that both variables named " name " are of the same length “
“ before combining datasets &dsn1 and &dsn2..";
      put ;
  _ERROR_+1;
    end;
  end;
run;

/*  Clean up work library by deleting datasets used in macro.*/
proc datasets;
  delete bothXVVS contentXVVS1 contentXVVS2;
quit;

/*  Turn output to listing back on.*/
ods listing;

/*  Label mend statement with “%exit” for error checking at beginning of macro.*/
%exit:  %mend verifyVariables;
```

## REFERENCES:

Foley, Malachy J. (1997) "Advanced Match-Merging: Techniques, Tricks and Traps", Proceedings of the Twenty-Second Annual SAS Users Group International, Conference, 199-206.

Foley, Malachy J. (1998) “MATCH-MERGING: 20 Some Traps and How to Avoid Them” Proceedings of the Twenty-Third Annual SAS Users Group InternationalConference.

SAS Institute, Inc. (1995), Combining and Modifying SAS Data Sets, Version 6, First Edition, Cary, NC: SAS Institute Inc.

SAS Institute, Inc. (1995), SAS Online Documentation, Version 9.1, SAS OnlineDoc>Base SAS>
SAS Language Reference: Concepts>DATA Step Concepts>
Reading, Combining, and Modifying SAS Data Sets>Combining SAS Data Sets: Methods,
Cary, NC: SAS Institute Inc.

## ACKNOWLEDGMENTS:

I would like to thank Elizabeth Hooten, for her encouragement and support, John Crouch, for his patient mentoring, and Richard Burgess, for his example of the highest level of professionalism.

## CONTACT INFORMATION:

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

John Cantrell
University of North Carolina
Health Policy & Administration
1107E McGavran-Greenberg
CB# 7411
Chapel Hill, NC 27599-7411
(919)843-6495
john_cantrell@unc.edu