SESUG 2022 Paper 157

Leveraging the CONTENTS PROCEDURE for easier SET statements

Tamar Roomian, MS MPH, Stryker Neurovascular

ABSTRACT

When setting together multiple data sets in the data step, two common problems can arise. First, the data sets can share variable names in common that are of different datatypes, resulting in error messages in the log. The error messages list which variables are both numeric and character, but fail to mention for which data sets, making it difficult for the user to resolve the issue using the log messages alone. Second, the data sets can share character variables of different lengths, which results in truncation of data when set together. The SAS Institute has published a macro program, %union, to combine data sets that have variables in common of different lengths, however it can only be used for two data sets at a time, and it does not account for differences in case or format length. When joining more than two data sets that contain hundreds of variables in common, resolving these discrepancies becomes time consuming and tedious. This paper will demonstrate two macro programs that both take advantage of the OUT statement of the CONTENTS procedure. The first generates an easy-to-read table of all variables in common across the data sets that have mismatched datatypes and lists which type is in which data set. The second takes the maximum length and maximum format length, and generates code saved in a macro variable to be used in the data step. Together, these two programs make setting multiple data sets in the data step faster, easier, and with less programming.

INTRODUCTION

When setting together multiple data sets in the data step, two common problems can arise:

First, the data sets can share variables in common that are of different datatypes resulting in error messages in the log.

Second, the data sets can share variables of different lengths, which results in truncation of data.

The SAS Institute has a macro program %union to combine data sets that have variables in common of different lengths (The SAS Institute, 2020), however it can only be used for two data sets at a time, does not account for differences in variable name case or format length.

To illustrate, three small, simple data sets will be used (dataset1-dataset3). Each dataset has three variables:

- name
- dob
- height

The following program can be used to create the three datasets needed:

```
data work.dataset1;
      length      name $3;
      input       name $      dob mmddyy10. height;
      format      name $3.    dob mmddyy10.;
      datalines;
Bob 12/05/1988 69
Sam 05/18/1978 72
Jan 04/23/1992 62
;
run;

data work.dataset2;
      length      name $9     dob $10;
      input       name $      dob $  height;
      format      name $10.   dob $10. ;
      datalines;
Alex 01/13/1961 70
David 10/06/1998 67
Christine 09/19/1981 65
;
run;

data work.dataset3;
      length name $10                  height $2;
      input  name $     dob mmddyy10. height $;
      format name $10.  dob mmddyy10. height $2.;
      datalines;
Jared 02/05/2000 75
Alessandra 06/29/1975 66
Isabella 04/07/1993 60
;
run;
```

Name is a character variable in the three data sets but of different lengths. Dob represents date of birth. It is a numeric variable in dataset1-dataset2 but character in dataset3. Height is a number value. It is a character variable in dataset3 but numeric in dataset1 and dataset2.

When the data is set together in the DATA step, the following messages appear in the log:

```
data combine;
set dataset1 dataset2 dataset3;
ERROR: Variable dob has been defined as both character and numeric.
ERROR: Variable height has been defined as both character and numeric.
run;
```

The error does not tell us which type is in which data set, leaving the programmer to manually check.

After resolving the errors, the following warning appears:

```
WARNING: Multiple lengths were specified for the variable name by input data set(s). This
can cause truncation of data.
```

When joining more than two data sets that contain hundreds of variables in common, resolving these discrepancies becomes time consuming and tedious. This paper will demonstrate two macro programs that both take advantage of the OUT statement of the CONTENTS Procedure.

A MACRO PROGRAM TO RECONCILE DIFFERENT DATATYPES ACROSS MULTIPLE DATA SETS

The first macro program, 'type,' requires one argument. List the data sets with the same variables in the %let dsn= statement, separated by spaces. The data sets must be in the work library.

First, the number of data sets is counted and assigned to the dsn_count macro variable.

```
/*list data sets in work library in %let dsn= statement*/

%let dsn= ;

%let dsn_count= %sysfunc(countw(&dsn));
```

Next, a do loop and OUT statement in the CONTENTS procedure creates data sets of meta data with the suffix _c:

```
%macro type;
%do i=1 %to &dsn_count;
      proc contents data=%scan(&dsn, &i) noprint out= %scan(&dsn,
      &i)_c;
      run;
%end;
quit;
```

The meta data is set together. Only the data set name, the variable names, and the data type are kept. The variable names are made lowercase to resolve any case differences that may exist.

```
data varlist;
set
      %do j=1 %to &dsn_count;
            %scan(&dsn, &j)_c (keep=memname name type)
      %end;
      ;
name=lowcase(name);
run;
```

The SQL Procedure is used to count the number of distinct data types for each variable name. A value of 2 means both character and numeric exist.

```
proc sql;
create table match as
select
      name
      , count (distinct type) as match
from varlist
group by name
;
quit;
```

A format is created to make the values of type clear to the user.

```
proc format;
value type  1="Num"
            2="Char"
;
run;
```

PROC SQL is used to create a table that has a row for each variable name and column for each data set. Each cell designates if the variable for that data set is character or numeric.

```
/*table with all variables and data set types*/
proc sql;
create table compare as
select
A.name as variable
      %do k=1 %to &dsn_count;
            , %scan(&dsn, &k)_c.type as %scan(&dsn, &k) format=type.
      %end;
, match


from (select distinct name from varlist) as A
      %do m=1 %to &dsn_count;
            left join %scan(&dsn, &m)_c on lowcase(%scan(&dsn,
            &m)_c.name)=A.name
      %end;
left join match on A.name=match.name
;
quit;
```

Finally, the table is limited only to variables that exist as both numeric and character:

```
title "Variables with mismatching data types";
proc print data=compare;
where match>1;
run;
title;
%mend;
```

Using the example data sets, the macro can be called:

```
%let dsn=dataset1 dataset2 dataset3;

%let dsn_count= %sysfunc(countw(&dsn));

%type;
```

The result from our example datasets is as follows:

Obs	variable	dataset1	dataset2	dataset3	match
1	dob	Num	Char	Num	2
2	height	Num	Num	Char	2

This table is much clearer than manually checking each data set. The programmer can see that dob is a character variable in dataset2 and height is a character variable in dataset3. Now, the programmer can decide which variables in which data sets should be converted from numeric to character and vice versa.

Note that the program does not specify which variables to be converted or which format to use. That is up to the discretion of the programmer.

To continue the example, the following program is submitted to resolve the data type discrepancies:

```
data dataset2a;
set dataset2;
dob2=input(dob, mmddyy10.);
format dob2 mmddyy10.;
drop dob;
rename dob2=dob;
run;

data dataset3a;
set dataset3;
height2=input(height, 8.);
drop height;
rename height2=height;
run;

data combine;
set dataset1 dataset2a dataset3a;
run;
```

After setting the data together, now a warning about data truncation appears in the log:

```
WARNING: Multiple lengths were specified for the variable name by input data set(s). This can
         cause truncation of data.
```

The PRINT Procedure reveals that the name variable was truncated:

Obs	name	dob	height
1	Bob	12/05/1988	69
2	Sam	05/18/1978	72
3	Jan	04/23/1992	62
4	Ale	01/13/1961	70
5	Dav	10/06/1998	67
6	Chr	09/19/1981	65
7	Jar	02/05/2000	75
8	Ale	06/29/1975	66
9	Isa	04/07/1993	60

This can be resolved by a macro program in the next section.

A MACRO PROGRAM TO RESOLVE MULTIPLE LENGTHS WHEN SETTING MULTIPLE DATA SETS

As before, the data sets must be in the work library. In addition, variables of the same name must be the same type prior to setting. The macro program takes two arguments

1. listing the data sets in the %let dsn= statement
2. a name for the output data set in the %let out= statement

The number of data sets is counted and saved in the macro variable dsn_count.

The macro program takes advantage of the OUT statement of PROC CONTENTS. However, we only need meta data on character variables, so the output is restricted to type=2.

```
%let dsn= ;

%let dsn_count= %sysfunc(countw(&dsn));

%let out= ;


%macro union ;
%do i=1 %to &dsn_count;
 proc contents data=%scan(&dsn, &i) noprint out=out&i(keep=name type
format length where=(type=2));
 run;
%end;
```

The outputted data sets (called "out") are set together to create one dataset with multiple rows per variable name per length. The lowcase function ensures that any existing case discrepancies are resolved.

```
data variable_list;
set out1-out&dsn_count;
name=lowcase(name);
run;
```

PROC SQL is used to select the maximum length for each variable name, use this information to write the SAS program language for assigning length and format in the select statement, and save each into macro variables &length and &format.

```
proc sql noprint;
select
cat("length ", name, "$ ", max(length), ";") as length
, cat("format ", name, "$", max(length), ".", ";") as format

into :length separated by " ", :format separated by " "

from variable_list
group by name
;
quit;
```

The macro variables are called in the final set statement to assign the maximum length and maximum format length when combining the data sets.

```
data &out;
&length
&format
set &dsn;
run;
%mend;

%union;
```

After the program is called, the warning of multiple lengths and truncation of data will no longer appear in the log.

```
%let dsn=dataset1 dataset2a dataset3a;
%let dsn_count= %sysfunc(countw(&dsn));
%let out=combine
%union;
```

PROC PRINT confirms that the name variable appears as expected:

Obs	name	dob	height
1	Bob	12/05/1988	69
2	Sam	05/18/1978	72
3	Jan	04/23/1992	62
4	Alex	01/13/1961	70
5	David	10/06/1998	67
6	Christine	09/19/1981	65
7	Jared	02/05/2000	75
8	Alessandra	06/29/1975	66
9	Isabella	04/07/1993	60

CONCLUSION

The meta data generated by the OUT= statement of PROC CONTENTS can be leveraged to resolve two common problems when setting multiple data sets sharing the same variables: variables of different types and variables of different lengths.

REFERENCES

The SAS Institute. 2020. "Sample 33407: Combining Data Sets Containing Character Variables of Different Lengths." Accessed March 11, 2022. https://support.sas.com/kb/33/407.html.

CONTACT INFORMATION

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

Tamar Roomian
tamar.roomian@stryker.org