Paper 124-2012

A Simple yet Effective Way to Perform a Variable Cross Walk Between Multiple Datasets

Binh C. Le, Centers for Disease Control and Prevention, Atlanta, Georgia.
Huong T. Pham, Macro International Inc., Atlanta, Georgia

ABSTRACT

Analyzing data from multiple complex datasets requires that analysts have knowledge of each dataset's structure. Comparing variable names, labels, and formats between more than two complex datasets can be challenging. A data dictionary and SAS procedures such as PROC COMPARE are commonly used to complete this task. However, PROC COMPARE only allows comparison of two datasets at a time. We present relatively simple procedures that data analysts can use to identify the changes in the structures of two or more than two datasets which can pinpoint areas in which existing SAS code need to be modified to account for changes in the variable names, labels, or formats.

INTRODUCTION

In an ideal situation, the data structure (i.e., variable names, values, labels and formats) for an ongoing survey conducted over many different years would remain the same. This will allow data analysts develop and use the same SAS code to analyze and manage datasets. In reality however, this situation rarely occurs. In the field of behavioral studies, for example, surveys are not static, questions may be added, the text may be altered, or the response options may be changed. These changes may require new variable names and labels and, sometimes result in changes to values or formats of existing variables. Using a SAS procedure like PROC COMPARE is not sufficient to identify these changes when working with more than two datasets. Therefore, another strategy is needed to identify differences in the data structure.

PROC CONTENTS is often used to describe the structure of the specified SAS dataset. The information includes the names and types (numeric or character) of the variables in the data set. The output of PROC CONTENTS will display valuable information at both the data set level such as name, engine, creation date, and number of observations, as well as at the variable level such as name, type, length, format, label, and position.

The PROC CONTENTS outputs for different datasets can also be saved as temporary or permanent SAS datasets. Then these outputted datasets can be merged to compare names, labels, formats, or lengths among variables from different datasets. This function of the PROC CONTENTS in combination with a simple MACRO program will be presented and discussed in this paper.

THE DATA

The U.S Centers for Disease Control and Prevention (CDC)'s National HIV Behavioral Surveillance System (NHBS) is an ongoing data collection system that conducts annual behavioral surveys (cycles) of three populations at high risk for HIV infection: Men who has sex with men (MSM), heterosexuals at increased risk for HIV (HET) and injection drug users (IDU).

For this paper, we will compare the structures of six datasets collected between 2003 and 2010: two cycles for each population. The NHBS interview data are collected on computers using Questionnaire Development System (QDS) but eventually imported into a SAS dataset where the data are merged with laboratory test results and recruitment data. For comparison across time and cycles, NHBS staff attempt to maintain consistent instruments, but due to the nature of the epidemic and changes to prevention approaches, revisions are needed periodically. These changes result in different variable names, values, and labels.

The SAS program presented in this paper was developed to support NHBS data collection monitoring, data analyses and quality control by providing a variable crosswalk between the six NHBS datasets, which each contain ~1000 variables.

PART I: WORKING WITH TWO DATASETS

A MACRO PROGRAM TO OUTPUT PROC CONTENT DATA

The included SAS program is created to compare the variable names and labels of two temporary NHBS datasets from the first cycle among MSM (SET1) and IDU (SET2).

The MACRO program named CRWLK below will run the PROC CONTENTS of each dataset. It will create two temporary datasets; dataset A from SET1 and dataset B from SET2.

```
%MACRO CRWLK (V, V1);
PROC CONTENTS DATA=&V1 NOPRINT OUT= &v
(KEEP= name type format label length engine nobs);
RUN;

DATA &V;
SET &V;
&v="&v1";
label_&v1=label;
DROP label;
RUN;

%MEND;
%CRWLK(A,SET1);
%CRWLK(B,SET2);
```

The `%MACRO CRWLK (V, V1)` is used to create a macro program named CRWLK and two macro variables named *v* and *v1;*

The `OUT= &v` option is necessary to output data and save it to a SAS temporary folder. When running `%CRWLK(A,SET1),` the value A is assigned to &v and as a result, a temporary dataset named A is created. Likewise, a temporary dataset named B is created when running `%CRWLK(B,SET2).`

The `KEEP` option is used to keep only variables of interest. In this macro program, only *name, type, format, label, length, engine,* and *nobs* variables are kept.

The DATA STEP is necessary to modify the structure of the datasets A and B to allow the two datasets to be merged without overwriting data.

Within the data step, the `&v="&v1"` code, the value A is assigned to &v and the value SET1 is assigned to &v1 when running `%CRWLK(A,SET1);` similarly, value B is assigned to &v and value SET2 is assigned to &v1 when running `%CRWLK(B,SET2).` As a result the tracking variables named *a* and *b* are created and respectively assigned the values: SET1 and SET2.

The `label_&v1=label` code is included to rename the *label* variable in each dataset to prevent overwriting when the two datasets are merged (in Tables 1 and 2, the *label_SET1* and *label_SET2* variables are created to replace the original variable *label*).

The `DROP label` code is used to drop the original *label* variable from each dataset.

Comparison of variable names and labels is only possible if the `&v="&v1"` and the `label_&v1=label` code are included in the MACRO program.

In the newly created datasets, A and B, dataset A has following variables: *name, type, format, label_SET1, length, engine, nobs*, and *a* (Table 1); and the dataset B has following variables: *name, type, format, label_SET2, length, engine, nobs*, and *b* (Table 2)

Table 1. Data structure of dataset A

	Variable Name	Variable Type	Variable Length	Variable Format	Observations in Data Set	Engine Name	label_SET1	a
1	AFRAID	1	8		0	V9	Afraid of finding out had HIV	SET1
2	AGE	1	8		0	V9	Age	SET1
3	AGEINJ	1	8		0	V9	Age first injected	SET1
4	AGO_12M	1	8		0	V9	12 months ago	SET1
5	ANTIRET	1	8		0	V9	Ever taken antiretrovirals	SET1

Table 2. Data structure of dataset B

	Variable Name	Variable Type	Variable Length	Variable Format	Observatio in Data	Engine Name	label_SET2	b
1	ABLE	1	8		0	V9	Able to complete health survey	SET2
2	AFRAID	1	8		0	V9	Afraid of finding out had HIV	SET2
3	AGE	1	8		0	V9	age today	SET2
4	AGEINJ	1	8		0	V9	Age first injected drugs	SET2
5	AGOZYRS	1	8		0	V9		SET2

A SIMPLE DATA STEP PROCEDURE TO MERGE DATA

To compare variable names and labels of datasets A and B, the two datasets will be merged using the SAS procedures below.

The *name* variable is the unique key to merge the two datasets (in Figure 1 and Figure 2, the variable *name* is located in the second column labeled “Variable Name”). Before merging, PROC SORT is used to sort each dataset by the *name* variable.

A DATA STEP is used to merge the two datasets. The variable *track* was created by concatenating variable *a* and variable *b*. If a variable name is found in both datasets, the value of variable *track* will be “SET1SET2”, if a variable name is found in only one dataset; the value of variable *track* will either be “SET1” or “SET2”.
In this example, the *variable_track* and *label_track* variables are created. The variable *variable_track* is used to compare the variable names; *label_track* is used to compare the variable labels.

```
PROC SORT DATA=A; BY name; RUN;
PROC SORT DATA=B; BY name; RUN;

DATA SET1_SET2;
MERGE A B;
BY name;

track=compress(trim(a)||trim(b));
/*tracking to see if a variable is in both DATASETS*/
if track="SET1SET2" then variable_track="variables found in both SET1 and set2"; else
if track="SET1" then variable_track="variables found in SET1 only"; else
if track="SET2" then variable_track="variables found in SET2 only";

/*tracking to see if a variable in both DATASETS has the same labels*/
if variable_track="variables in both SET1 and SET2" then do;
if label_SET1 ^= label_SET2 then label_track="label name changed    ";
else label_track="label name not changed"; end;

/*tracking to see if a variable in dataset SET1 has missing label*/
if variable_track="variables in SET1 only" then do;
if label_SET1=" " then label_track="SET1 only-no label";
```

```
else label_track="SET1 only-yes label"; end;

/*tracking to see if a variable in dataset SET2 has missing label*/
if variable_track="variables in SET2 only" then do;
if label_SET2=" " then label_track="SET2 only-no label";
else label_track="SET2 only-yes label"; end;

DROP a b track;
RUN;
```

When the above program is executed, the merged dataset named SET1_SET2 is created and saved in the SAS library, temporary folder. Table 3 displays a sample of the data structure of the dataset SET1_SET2.

Table 3. Data structure of dataset SET1_SET2

	Variable Name	label_SET1	label_SET2	variable_track	label_track
1	ABLE		Able to complete health survey	Variables found in SET2 only	SET2 only-yes label
2	AFRAID	Afraid of finding out had HIV	Afraid of finding out had HIV	Variables found in both SET1 and SET2	Label name not changed
3	AGE	Age	age today	Variables found in both SET1 and SET2	Label name changed
4	AGEINJ	Age first injected	Age first injected drugs	Variables found in both SET1 and SET2	Label name changed
5	AGOZYRS			Variables found in SET2 only	SET2 only-no label

THE RESULTS

Dataset SET1_SET2 could be analyzed in many different ways to output information of interest. Figures 4a and 4b show screen shots of the resulting frequencies of variables *variable_track* and *label_track,* respectively, obtained from the following PROC FREQ procedure:

```
PROC FREQ DATA= SET1_SET2;
TABLES variable_track label_track;
RUN;
```

Table 4a. Frequency of variable *variable_track*

variable_track	Frequency	Percent
Variables found in SET1 only	175	13.75
Variables found in SET2 only	612	48.08
Variables found in both SET1 and SET2	486	38.18

Table 4b. Frequency of variable *label_track*

label_track	Frequency	Percent
Label name changed	138	10.84
Label name not changed	348	27.34
SET1 only-no label	12	0.94
SET1 only-yes label	163	12.80
SET2 only-no label	61	4.79
SET2 only-yes label	551	43.28

Table 4a indicates that 486 variables were found in both datasets. Table 4b shows that among these 486 variables, 138 had different variable labels. The following PROC PRINT is used to get the output as shown in Table 5, where the data is limited only to those 138 variables with different variable labels.

```
PROC PRINT DATA=SET1_SET2;
WHERE label_track="Label name changed";
VAR name label_SET1 label_SET2;
RUN;
```

Table 5. Frequency of variable *label_track*

Obs	NAME	label_SET1	label_SET2
3	AGE	Age	age today
4	AGEINJ	Age first injected	Age first injected drugs
28	ATTRMENB	Other friends: Attracted to or has sex w	Friends who are not gay, lesbian, or bis
32	ATTRMENF	No, haven't told anyone: Attracted to or	Someone else: Attracted to or has sex wi
36	ATTRMWB	Other friends: Attracted to or has sex w	Friends who are not gay, lesbian, or bis
40	ATTRMWF	No, haven't told anyone: Attracted to or	Someone else: Attracted to or has sex wi
44	ATTRWOMB	Other friends: Attracted to or has sex w	Friends who are not gay, lesbian, or bis
48	ATTRWOMF	No, haven't told anyone: Attracted to or	Someone else: Attracted to or has sex wi
56	BIRTHSEX	Birth sex	sex at birth

PART II: WORKING WITH MULTIPLE DATASETS

A MACRO PROGRAM FOR MULTIPLE DATASETS

A similar approach using a MACRO program and DATA STEPS can be used when working with multiple datasets. The example below demonstrates how to apply the strategy to examine the data structure of six datasets (dataset SET1-SET6).

```
%MACRO CRWLK (V,V1);
PROC CONTENTS DATA=&V1 NOPRINT OUT= &v
(KEEP= name type format label length engine nobs);
RUN;

DATA &V;
SET &V;
label_&v1=label;
&v="&v1";
DROP label;
RUN;

%MEND;
%CRWLK(A,SET1);
%CRWLK(B,SET2);
%CRWLK(C,SET3);
%CRWLK(D,SET4);
%CRWLK(E,SET5);
%CRWLK(F,SET6);

/*note: If working with more than 6 datasets similar macro run commands need to be
added. For example
%CRWLK(G, SET7);
%CRWLK (H, SET8);*/
```

The `%CRWLK(A,SET1);%CRWLK(B,SET2); %CRWLK(C,SET3); %CRWLK(D,SET4);%CRWLK(E,SET5); %CRWLK(F,SET6);` MACRO commands correspond respectively to data SET1-SET6. If working with more than 6 datasets similar MACRO commands can be added to the list.

When the macro program is executed, 6 datasets (A to F) will be created. To compare differences in variable names and labels or create a cross walk between the 6 datasets, the 6 datasets should be merged together.

A SIMPLE DATA STEP PROCEDURE TO MERGE ALL DATASETS

```
PROC SORT DATA=A; BY name; RUN;
PROC SORT DATA=B; BY name; RUN;
```

```
PROC SORT DATA=C; BY name; RUN;
PROC SORT DATA=D; BY name; RUN;
PROC SORT DATA=E; BY name; RUN;
PROC SORT DATA=F; BY name; RUN;

/*note: If working with more than 6 datasets similar PROC SORT need to be added. For
example
PROC SORT DATA=G NODUPKEY; BY name; RUN;
PROC SORT DATA=H NODUPKEY; BY name; RUN; */

DATA SET1_SET6;
MERGE A B C D E F; /*note: If working with more than 6 datasets, add G H*/
BY name;
track=compress(trim(a)|| trim(b)|| trim(c)|| trim(d)|| trim(e)|| trim(f));

/*note: If working with more than 6 datasets, trim(g)|| trim(h)… need to be added*/
/*tracking to see if a variable is in all dataset*/
/*note: If working with more than 6 datasets, add SET7, SET8… values to track
variable; and result variable */;

if track="SET1SET2SET3SET4SET5SET6" then variable_track="variables found in all 6
datasets  "; else
if track="SET1" then variable_track="variables found in set1 only"; else
if track="SET2" then variable_track="variables found in set2 only"; else
if track="SET3" then variable_track="variables found in set3 only"; else
if track="SET4" then variable_track="variables found in set4 only"; else
if track="SET5" then variable_track="variables found in set5 only"; else
if track="SET6" then variable_track="variables found in set6 only"; else
variable_track="variables found in sets 2,3,4, or 5";

RUN;
```

Dataset SET1-SET6 created from the DATA STEP above includes all information necessary to perform a variable crosswalk among the 6 datasets. Table 6 below shows a part of the data structure of the dataset SET1_SET6.

Table 6. A part of data structure of SET1_SET6 dataset

	Variable Name	Variable Type	Variable Length	Variable Format	Observations in Data Set	Engine Name	track	variable_track
1	ABLE	1	8		0	V9	SET2	variables found in set2 only
2	ABLE_SP	2	20		0	V9	SET6	variables found in set6 only
3	AFRAID	1	8		0	V9	SET1SET2SET3SET4SET5SET6	variables found in all 6 datasets
4	AGE	1	8		0	V9	SET1SET2SET3SET4SET5SET6	variables found in all 6 datasets
5	AGEINJ	1	8		0	V9	SET1SET2SET3SET4SET5SET6	variables found in all 6 datasets
6	AGO2YRS	1	8		0	V9	SET2SET3SET4SET5SET6	variables found in sets 2,3,4, or 5

THE RESULTS

In the dataset SET1_SET6, the variable *variable_track* could be calculated in many different ways to best fit the specific needs of variable comparisons. In this SAS example, the variable *variable_track* has been calculated to show only 8 specific values (see Table 7). The 6 datasets have 168 common variables. Dataset SET1 has 174 unique variables; dataset SET2 has 386 unique variables; dataset SET3 has 172 unique variables; dataset SET4 has 10 unique variables; dataset SET5 has only 1 unique variable; and dataset SET6 has 17 unique variables.

```
PROC FREQ DATA=SET1_SET6;
TABLES variable_track;
RUN;
```

Table 7. Frequency of variable *variable_track*

result	Frequency	Percent
variables found in all 6 datasets	168	8.40
variables found in set1 only	174	8.70
variables found in set2 only	386	19.31
variables found in set3 only	172	8.60
variables found in set4 only	10	0.50
variables found in set5 only	1	0.05
variables found in set6 only	17	0.85
variables found in sets 2,3,4, or 5	1071	53.58

The variable *variable_track* is calculated to perform a crosswalk of the variable names only. In many cases, the variable names are the same but the labels are different. To perform a crosswalk of the variable labels, another variable (e.g. *label_track* - see PART I) should be calculated. This variable can be calculated in many different ways to track changes in the variable labels when the variable names are the same in multiple datasets.

CONCLUSION

Comparing variable names and labels between two or more datasets is critical in data management and analyses. The MACRO program and the DATA STEPS we present in this paper are relatively simple but effective methods for comparing variable names and labels between two or more complex datasets.

REFERENCES

Gallagher KM, Sullivan PS, Lansky A, Onorato IM. (2007) “Behavioral surveillance among people at risk for HIV infection in the U.S.: the National HIV Behavioral Surveillance System” ***Public Health Rep. 2007;122 Suppl 1:32-8***.

ACKNOWLEDGMENTS AND DISCLAIMATIONS

We would like to thank Teresa J. Finlayson and Nevin Krishna for their comments on this paper.

The findings and conclusions in this paper are those of the authors and do not necessarily represent the views of the Centers for Disease Control and Prevention.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Name: Binh C. Le
Agency: Center for Disease Control and Prevention
Address: 1600 Clifton Road NE.
City, State ZIP: Atlanta, GA 30333
Work Phone: (404) 639- 2057
Fax: (404)639-8640
E-mail: bil7@cdc.gov