Paper 100

THE NUANCES OF COMBINING MULTIPLE HOSPITAL DATA

Jontae Sanders, MPH, Charlotte Baker, DrPH, MPH, CPH, and C. Perry Brown, DrPH, MSPH,
Florida Agricultural and Mechanical University

ABSTRACT

Hospital discharge data can be used for the surveillance of various health conditions in a population. To maximize our ability to tell the story of a population's health, it is often necessary to combine multiple years of data. This step can be tedious as there are many factors to take into account such as changes in variable names or data formats between years. Once you have resolved these issues, the data can be successfully combined for analysis. This paper will demonstrate many factors to look for and how to handle them when combining hospital data across years.

INTRODUCTION

Surveillance of health conditions has been used to describe characteristics of health outcomes among populations. Hospital data has been a valuable resource for assessing and managing health outcomes as they arise among populations. When surveying health outcomes over time, it is often a necessity to combine multiple hospital data sets to help assess the situation. There are many nuances to combining hospital data sets and it requires a lot of time to properly combine them. There are two approaches to combining hospital data: concatenating and merging. In this paper, we will explain the approaches to combining hospital data sets through concatenation, and the approaches to fixing the nuances that arise.

COMBINING FILES THROUGH CONCATENATION

Concatenating hospital data is done by combining two or more SAS® data sets in a series. When combined, the total number of observations in the new data set is identical to the sum of the number of observations in the original data sets. The SET statement in the DATA step is used to concatenate multiple hospital data sets into a single data set.

Although, hospital data can be combined by both concatenating and merging, the process of concatenation in this discussion used. The nuances to be discussed apply to both procedures. Hospital discharge data sets may have the same variable names and formats across years, in that case concatenation is easy. This may not always be the case.. The nuances discussed in this paper can occur with both concatenation and merging.

THE NUANCES

Several issues can arise while attempting to combine data that could hinder your process. These issues include variables among the various hospital data sets not having corresponding characteristics (numeric value vs character value), variable names not being exactly the same, different variable length, and user error resulting in attempts to combine dissimilar data sets.

If a character variable is read into SAS® as a numeric variable in one hospital data set and as a character variable in another hospital data set, SAS® recognizes the conflict and will not concatenate the data sets. Further, if variable names are not the same among all hospital data sets to be concatenated, SAS® will read the variable as an error in the new data set and not combine the variable.

NUMERIC VS CHARACTER VARIABLES

When multiple hospital data sets are read, SAS® determines the new characteristics of the variables according to the first data set SAS® reads. For example, the first data set read shows variable a as being numeric but all the other data sets show it as being character. SAS® will assign the new data set a numeric format for that variable and prevent other data from being written to the new data set. This becomes clear when, for example, you can see the variable names are the same but get errors in the log showing that the data sets will not combine. The following is a typical error message.

```
ERROR: Variable race has been defined as both character and numeric.
```

Figure 1: Error Message

SAS® will refuse to combine the data sets until the problem of different variable formats is solved. To avoid the issue, the programmer should run PROC CONTENTS on each hospital data set to identify the type of variable. PROC CONTENTS shows the contents of each data set and prints the index of the data library. This information includes what type of data SAS® thinks a variable is (character or numeric). An example of PROC CONTENTS code and output is below:

```
LIBNAME example "c:/hospital";
DATA new;
SET hospital.data92;
RUN;

PROC CONTENTS DATA = new;
RUN;
```

Variables in Creation Order

#	Variable	Type	Len	Format	Informat	Label
1	year	Num	8	F4.		year
2	src_admt	Num	8	F2.		src_admt
3	race	Char	1	$1.	$1.	race
4	prin_dc	Char	5	$5.	$5.	prin_dc
5	sec_dc1	Char	5	$5.	$5.	sec_dc1

Figure 2: Example of PROC CONTENTS output

INPUT VERSUS PUT STATEMENTS

In order to fix the problem of a variable being read as both numeric and character in different data sets, the SAS® INPUT and PUT statements can be used to appropriately assign the variable characteristics. The structure of the PUT statement is:

```
variable = PUT (source,format);
```

The PUT statement in SAS® can be used to convert a numeric value to a character value for a given variable. The PUT statement does not allow you to directly change the type of variable in SAS® from numeric to character. To do this, you must create a new character variable using a PUT function in a DATA step. Once you have changed the value of the variable from numeric to character, you can then use the DROP statement to drop the original numeric variable and RENAME statement to rename the new variable back to its original variable name. Examples of how to do this are below.

The first example shows how to use the PUT function to change variable types.

```
DATA new;
SET hospital.data92;
dischstat = PUT(discharg,2.);
gender = PUT(sex,1.);
admsrc = PUT(src_admt,2.);
RUN;
```

The second example shows how to keep the original variable name.

```
DATA new;
SET hospital.data92;
dischstat = PUT(discharg,2.);
DROP discharg;
RENAME dischstat = discharg;
RUN;
```

If you plan to keep the original variable name of discharg for the character variable, use the DROP and RENAME statements following the PUT function.

This syntax creates a variable dischstat as character from the numeric discharg variable. The DROP statement keeps the numeric variable discharg from being written to the new data set and the RENAME statement renames the new character variable from dischstat to discharg.

Unlike the PUT function, the INPUT function converts character values into numeric values. The structure of the INPUT function is as follows:

```
variable = INPUT(source,informat);
```

In the example below, you can see the character variable sex being converted to the numeric variable gender and the character variable src_admt being converted to the numeric variable admsrc.

```
DATA new;
SET hospital.data92;
dischstat = INPUT(discharge,2.);
gender = INPUT(sex,1.);
admsrc = INPUT(src_admt,2.);
RUN;
```

VARIABLE NAMES

Concatenating hospital data sets can become a problem when variable names vary among the various available data sets. Over time, patient identifiers could have changed among populations as could have variables indicating ”days of procedure”, “age”, and “payer”. The changes can become headaches. SAS® will not identify the erroneous variable names though it may display errors in your log. Variables must be renamed so that they are able to concatenate successfully. In order to avoid this issue, the programmer should run PROC CONTENTS on each hospital data set to be sure variable names are identical between years. To fix the identified problems, it is necessary to rename variables in a DATA step so that they have the same name. In the example below, three variables were renamed and the old variables were dropped. One could also use the RENAME statement to rename the three variables.

```
DATA new;
SET hospital.data92;
DROP day_proc admt_age p_payer;
daysproc = day_proc;
age = admt_age;
payer = p_payer;
RUN;
```

The above syntax renames all three variables and drops the original variable names.

IMPROPER DATA SETS

When concatenating multiple hospital data sets, it is imperative that the correct data sets are combined. Combining multiple data sets could cause user confusion and subsequent SAS® errors due to multiple DATA steps and SET statements. Data sets could be improperly renamed or applied in the improper order. In order to control for such errors, every programmer should utilize comments so that proper notes to self (or other team members) and instructions about what needs to be done can minimize confusion. It is very frustrating to return to a program and not be able to figure out what you did and why you did it. If someone else needs to look over your program, the task is made much easier if the code is commented.

There are two methods for writing comments in SAS®. In the first, the comment begins with an asterisk and continues until SAS® encounters a semicolon.

```
*THIS DATA STEP IS FOR CONCATENATED HOSPITAL DATA FOR YEARS 1992-2000;
```

SAS® will ignore the line commencing with * during execution.

In the second, the comment begins with a forward slash and an asterisk and continues until SAS® encounters an asterisk and a forward slash.

```
/*THIS DATA STEP IS FOR CONCATENATED HOSPITAL DATA FOR YEARS 1992-2000*/
```

SAS® will ignore the information between /* and */ during execution.

COMMON VARIABLES

Concatenating and merging can both be done to combine multiple data sets but there are some nuances that should be paid attention to when attempting to do either. Variables must be sorted for MERGE statements but not for SET statements. Improperly sorting common variables using the BY statement while using the MERGE statement can cause errors in the output. PROC SORT is commonly used to sort data sets by a common variable. If there is no need to add additional variables or the programmer is only interested in stacking data sets on top of each other, concatenating the data can help the user avoid additional steps such as using a BY statement in the DATA step and PROC SORT.

CONCLUSION

This paper was written to highlight potential nuances of combining hospital data sets. When combining data sets, be sure to pay attention and know the data at hand so that errors can be avoided or dealt with as they arise.

REFERENCES

Available at

1. http://support.sas.com/documentation/cdl/en/basess/58133/HTML/default/viewer.htm#a002645432.htm
2. http://support.SAS.com/documentation/cdl/en/basess/64003/HTML/default/viewer.htm#p0y8rrzaeklkkin1tlyxeq5qskkp.htm
3. http://support.SAS.com/onlinedoc/913/getDoc/en/lrdict.hlp/a000180357.htm
4. http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a001081414.htm

ACKNOWLEDGMENTS

The author would like to thank Florida A & M University faculty and staff for all their help, comments and support in producing this paper. The author would also like to thank the Florida Agency for Healthcare Administration, State Center for Health Information and Policy Analysis.

CONTACT INFORMATION

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

Name: Jontae Sanders
Enterprise: Institute of Public Health, Florida A&M University
Address: 1515 S. Martin Luther King, Jr. Blvd, SRC207-F
City, State ZIP: Tallahassee, Florida 32307
Fax: (850) 599-8830
E-mail: sandersjontae@yahoo.com