SESUG Paper 109-2023

How to Merge SAS® Datasets with PII

Julie Plano, Keli Sorrentino, Yale School of Public Health

ABSTRACT

As a programmer being asked to combine datasets is a standard task. However, most don't anticipate the danger if not executed correctly. This can be particularly problematic when the merge involves personally identifiable information (PII). Our paper examines cases in which PII is used to complete a match-merge. Examples include problems that can arise before and during the matching process as well as necessary data cleaning techniques to ensure safe merging.

INTRODUCTION

We've all been there; you receive a couple datasets with the promise of a link worthy variable (or two) and are asked to merge them. The goal is to combine the multiple datasets into one usable set of information. To be successful we need to explore the data, forecast any problems, implement data cleaning where necessary, and avoid any trouble prior to and during the match-merge.

Our work includes using personal information to link various types of data e.g. vaccination records, hospital admissions and vital statistics (birth and death). These data present unique challenges. First and foremost they are produced by different entities. The naming conventions and variable attributes are not consistent, the number of observations for an individual can vary, and unique identifiers usually require some data wrangling before being ready to merge. The following discussion will include ways to prepare variables prior to the match. Describe techniques on the creation and use of a concatenated string variable. In addition, we will demonstrate how to complete a one-to-one merge and match-merge in a SAS data step as well as a join in PROC SQL.

EXAMPLE DATA

Obs	LastName	FirstName	Date
1	GarvusX	JEAN	01JAN2020
2	DEMEIDA	LAILA	01FEB2022
3	STEVEN	BILL	01APR2021
4	Smith	CHRISTOPHER	01AUG2022
5	Smith	CHRISTOPHER	01AUG2022
6	Fry	Joe	18SEP1998

Dataset A

Obs	LastName	FirstName	Date
1	GARVUS	JEAN	01JAN2020
2	DEMEIDA	Laila Mae	01FEB2022
3	DEMEIDA	Laila Mae	01FEB2022
4	STEVEN	BILL	01APR2021
5	SMITH	CHRISTOPHER	01AUG2022

Dataset B

Figure 1. Example Data

VARIABLE PREPARATION

Different data sources will store the same variables in different ways. Careful date importing and the use of SAS functions to standardize key variables across datasets will ensure a successful match.

When importing data from multiple sources be sure to stay aware of the consistency of not only formats and data types but also variable length. Modifications using SAS functions will only be beneficial if the key variables are read in with the same length. Ways to do this include;

- If importing a .csv use `guessingrows=max`. Guessing rows tells SAS to scan 'x' number of rows to determine variable length. This method will assign the length of the longest value.
- Syntax in an INFILE statement can be modified to correct character length. An easy way to do this, instead of writing your own code, is to use the import wizard. Once run, in the log you can find the code. By pressing the ALT key and selecting the necessary lines (to the right of the line number) you can highlight, copy, and paste to your SAS program. Make changes as necessary and re-run the edited syntax.
- Inspect the PROC CONTENTS of each file to determine the greatest length of key variables, subsequently including syntax in the data step (before the set statement) to correct variable length. Use a format, informat and length statement.

For our project, first name, last name and date of birth are integral. However, because they are string variables (values which can contain characters and numbers) we need to make sure the variables appear identical in each dataset. The UPCASE function converts all lowercase letters to uppercase. Useful for multiple word names, the SCAN function counts words from left to right, e.g. suffixes or two-word names. In this example, Laila Mae is now only Laila. In the event your date is attached to a time, the DATEPART function identifies the date portion of the variable and returns the SAS date (data not shown).

```
data Functions;
set datasetB;
 Last = upcase(lastname);
 First=upcase(firstname);
 FirstN = scan(First, 1, '');
 format Date yymmdds10.;
run;
```

Obs	Last	FirstN	Date
1	GARVUS	JEAN	2020/01/01
2	DEMEIDA	LAILA	2022/02/01
3	DEMEIDA	LAILA	2022/02/01
4	STEVEN	BILL	2021/04/01
5	SMITH	CHRISTOPHER	2022/08/01

Figure 2. Functions Result Dataset B

Next, the SUBSTR function is used to extract a segment of the first and last name variables. This can be useful when comparing string identifiers that are prone to errors due to spelling variations. The number of characters to subset is completely up to the programmer. Validate the number through trial and error, attempting fewer or greater characters to see what best fits the data.

The below example uses the SUBSTR function to separate the character name into a portion of the string. The syntax labels the new variable name (L) and invokes the function, in parentheses the original variable name, starting position and the total number of characters, then returns a substring. Using the CATS function, the string variables are linked together.

```
data StringMatch;
set datasetB;
   L = substr(Last_name,1,3);
   F = substr(First_name,1,3);
   DobChar = put(Bdate, yymmdds10.);
   StringMatch = cats(L,F,DobChar);

run;
```

Concatenating pieces of string variables for comparison can be very useful in the match process and improve results, especially when linking names and dates across datasets. Below, the first 3 letters of the last name, first 3 letters of the first name and date of birth, as a character variable, make up the new variable 'StringMatch'.

Obs	L	F	DobChar	StringMatch
1	GAR	JEA	2020/01/01	GARJEA2020/01/01
2	DEM	LAI	2022/02/01	DEMLAI2022/02/01
3	DEM	LAI	2022/02/01	DEMLAI2022/02/01
4	STE	BIL	2021/04/01	STEBIL2021/04/01
5	SMI	CHR	2022/08/01	SMICHR2022/08/01

Figure 3. String Match Result Dataset B

This new variable creates an additional option, most of the time matching successfully. Be cautious though, names can be similar if not exact and occasionally there is an errant match. The example below is in fact not the same person, the 5th letter of the last name is the only difference in their identifying variables.

Name	Date of Birth	
FRIEL, JOE	07/30/1950	STRING = FRIJOE1950/07/30
FRIER, JOE	07/30/1950	STRING = FRIJOE1950/07/30

Figure 4. String Match Error

Since a level of error or misclassification can occur. Implement manual review or account for this later in edit checks.

DATA STEP MERGE

The one-to-one merge in SAS is simply done using a merge statement and no key variables. Successful merges of this type would have datasets with an equal number of observations and exclusive variable names. The data must be precisely ordered as the merge will match observation one in dataset A with observation one in dataset B, observation two in dataset A with observation two in dataset B and so on. This method, in most circumstances, does not work well.

```
data OneMerge;
merge datasetA datasetB;
run;
```

When key variables are present in both datasets, a BY statement should be included in the merge. The SORT procedure must be used to ensure key variables in both files have been indexed. Datasets can be sorted by multiple variables. The default order of a numeric variable is smallest to largest. A character value would be sorted by default in alphabetical order. For match-merge syntax to execute correctly data must be sorted by variables specified in the BY statement.

```
proc sort data = datasetA; by last firstN;
proc sort data = datasetB; by last firstN;

data MatchMerge;
merge datasetA datasetB;
by last firstN;
run;
```

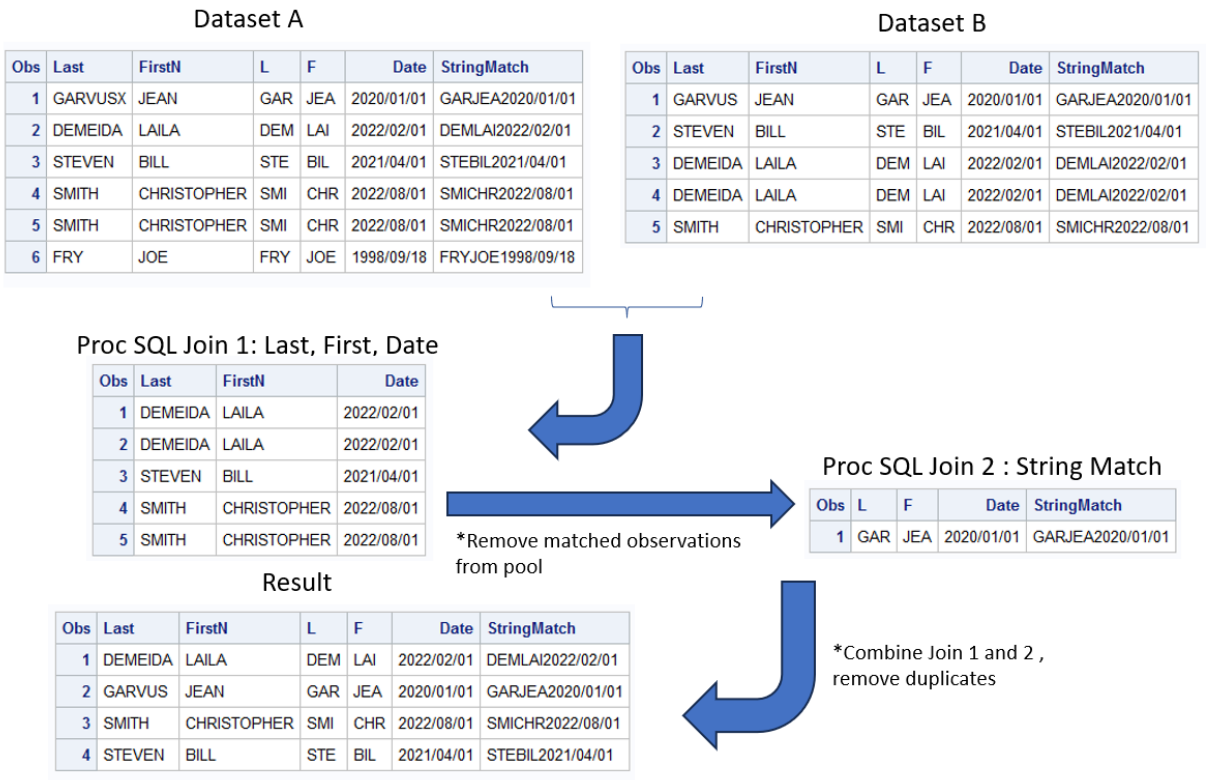
MANY TO MANY JOIN; PROC SQL

PROC SQL allows the programmer to combine data tables through queries, the act of combining columns from each table horizontally, is called a join. The default result of a join is a Cartesian Product; all rows from table A are joined with all the rows in table B. In the example syntax below, CREATE TABLE saves the name of the new table, SELECT chooses the key variables, using an asterisk will select all columns from the dataset (be aware this produces errors in the log), FROM indicates the name of data tables to be combined in the join, WHERE expression requires the observations in both data tables.

```
proc sql;
   create table NewTable as
   select a.*, b.*
   from a1 as a, b1 as b
   where
   a1.last = b1.last
   and a1.firstN = b1.firstN
   and a1.Date = b1.Date;
quit;
```

```
proc sql;
   create table NewTableSTR as
   select a.*, b.*
   from c1 as a, d1 as b
   where
   c1.StringMatch = d1.StringMatch;
quit;
```

Figure 5 shows dataset A and B with standardized variables. PROC SQL join 1, on Last, FirstN and Date, returns observations with exact matches on all three variables. Notice in the first join observation 1 is not matched; the last names are not exact. Matched observations are removed from the pool to be matched and then SQL code for join 2 is run. In the second join, now including 'StringMatch', observation 1 does create a match result. Observation 6 from dataset A is not included in either as there is no match in dataset B. If there are duplicates, they will be output in the new dataset. Using PROC SORT with a NODUPKEY option will remove duplicates based on key variables and return the result table below.



Dataset A

Obs	Last	FirstN	L	F	Date	StringMatch
1	GARVUSX	JEAN	GAR	JEA	2020/01/01	GARJEA2020/01/01
2	DEMEIDA	LAILA	DEM	LAI	2022/02/01	DEMLAI2022/02/01
3	STEVEN	BILL	STE	BIL	2021/04/01	STEBIL2021/04/01
4	SMITH	CHRISTOPHER	SMI	CHR	2022/08/01	SMICHR2022/08/01
5	SMITH	CHRISTOPHER	SMI	CHR	2022/08/01	SMICHR2022/08/01
6	FRY	JOE	FRY	JOE	1998/09/18	FRYJOE1998/09/18

Dataset B

Obs	Last	FirstN	L	F	Date	StringMatch
1	GARVUS	JEAN	GAR	JEA	2020/01/01	GARJEA2020/01/01
2	STEVEN	BILL	STE	BIL	2021/04/01	STEBIL2021/04/01
3	DEMEIDA	LAILA	DEM	LAI	2022/02/01	DEMLAI2022/02/01
4	DEMEIDA	LAILA	DEM	LAI	2022/02/01	DEMLAI2022/02/01
5	SMITH	CHRISTOPHER	SMI	CHR	2022/08/01	SMICHR2022/08/01

Proc SQL Join 1: Last, First, Date

Obs	Last	FirstN	Date
1	DEMEIDA	LAILA	2022/02/01
2	DEMEIDA	LAILA	2022/02/01
3	STEVEN	BILL	2021/04/01
4	SMITH	CHRISTOPHER	2022/08/01
5	SMITH	CHRISTOPHER	2022/08/01

Proc SQL Join 2 : String Match

Obs	L	F	Date	StringMatch
1	GAR	JEA	2020/01/01	GARJEA2020/01/01

Result

Obs	Last	FirstN	L	F	Date	StringMatch
1	DEMEIDA	LAILA	DEM	LAI	2022/02/01	DEMLAI2022/02/01
2	GARVUS	JEAN	GAR	JEA	2020/01/01	GARJEA2020/01/01
3	SMITH	CHRISTOPHER	SMI	CHR	2022/08/01	SMICHR2022/08/01
4	STEVEN	BILL	STE	BIL	2021/04/01	STEBIL2021/04/01

Figure 5. PROC SQL Process and Result

CONCLUSION

Which is better a data step merge or SQL? As we know, in SAS, there are a multitude of options that can be used to complete the same task. The size of your data may be a consideration, your time and ability may also come into play. The data step may be your comfort zone, however SQL conserves computer resources. Use the option that best fits your programming style and data question.

In this paper, we describe a deterministic match -- we look for exact matches between records. Fuzzy matching, another reliable method, matches records based on some decided upon level of similarity. In

other projects (not described here) we use these techniques, which include the functions SOUNDEX and COMPGED. If resources and time allow, the process could be improved by using both deterministic and probabilistic methods.

Merging safely should be of utmost importance. First and foremost know your data! Yanni Loukissas author of*, All Data Are Local: Thinking Critically in a Data-Driven Society*, speaks on working with data sources, an important point to keep in mind as technology advances and data grows, *"We should approach data sets with an awareness that data are created by humans and their dutiful machines, at a time, in a place, with the instruments at hand, for audiences that are conditioned to receive them."*

When different sources, people and software create and share data unique issues which arise. Cleaning and standardizing key variables before a merge or join can greatly improve match success.

CONTACT INFORMATION

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

Julie Plano
Yale School of Public Health
203.764.9292
Julie.colburn@yale.edu