

Combine Datasets using *Inexact* Character Variables in SAS

By

Kulwant Rai¹

University of Virginia

July 6, 2011

Abstract

This project proposes an algorithm to combine datasets using inexact character variables, and implements this algorithm using SAS software. When this algorithm is used together with PROC SQL it allows combining datasets based on several inexact variables, hence producing better matches. This procedure offers realistic solutions for accurate and more complete matching of inexact data fields. Apart from being simple and easy to understand, the strength of this procedure is to condense information from several inexact character variables into one numerical measure. Thus, many valid matches can be captured that previously could only be made by an individual's exhaustive visual inspection of the datasets. The procedure will save time and effort for anyone trying to combine datasets with inexact character variables.

Introduction

The algorithm developed in this paper allows for a more efficient merging of datasets with inexact character variables. The need for such a tool arises when names within different datasets are not exact and unmatchable by computer but could be matches by a "human eye." For example, the two datasets can have different variations of name for the same company like "Ladenburg Thalmann Financial Services Inc" and "Fin Svcs, Ladenburg Thalmann Inc." These data fields are usually imprecise due to errors or variations in reporting that make typical linking by exact agreement insufficient, but the algorithm developed in this project will accommodate such matches.

¹ Contact information: kr9c@virginia.edu

The need to combine datasets using inexact text variable arises when a researcher is trying to use two different data sources that don't share unique identifier. This situation is typical for the researchers trying to combine data from different sources; for example, SDC Plantium and WRDS, or Placement Tracker and WRDS. Such a list could be endless because there is no way to resolve name variations when data is combined from different sources. The typical computer linking procedures requiring exactness have been previously shown to miss a substantial fraction of valid matches.

The attractive feature of the proposed algorithm is that it provides numerical measure for the match. In particular, it calculates the number of letters that match in the two character string from "right to left" and "left to right," the sum of these divided by the total number of letters in the two strings provide the "proportion of common letters" in the strings. Performance test reveal that when the proportion of common letters is more than 0.80 the two names, almost certainly, belong to the same company. A successful and reliable linkage procedure, such as this, offers great opportunities for researchers.

Algorithm description

To obtain proportion of common letters in the two text strings (call them string1, and string2) search for the shorter length string in the longer string from left-to-right and record the maximum number of letters that match, call it "match_left2right". Then do the same right-to-left and record it as "match_right2left." Then calculate proportion of common letters as

$$PCL = \text{sum}(\text{match_left2right}, \text{match_right2left}) / \text{sum}(\text{length}(\text{string1}), \text{length}(\text{string2}))$$

SAS Code

The above algorithm is coded as a user defined function in SAS software using “PROC FCMP;” SAS.9.2 is required to use this function in DATA step or “PROC SQL.” Below is the complete code with description:

Invoke PROC FCMP, and name the function “PCL”, short for proportion of common letters, which will take input of two string variables:

```
proc fcmp outlib=sasuser.funcs.trial;  
function pcl(string1 $, string2 $);
```

Create a few “dummy” variables and initialize them to zero:

```
match_left2right=0;  
match_left2right1=0;  
match_right2left=0;  
match_right2left1=0;
```

Check if string1 has greater length than string2, and if so, obtain the maximum number of character in string2 that are also in the same order in string1 from left-to-right and record them in variable “match_left2right.”

```
if length(string1) >= length(string2) then do;  
  do i=1 to min(length(string1),length(string2)) by 1;  
    do j=min(length(string1),length(string2))+1-i to 1 by -1;  
      if index(string1,substr(string2,i,j)) >0 then match_left2right1=j;  
      if match_left2right1 > match_left2right then match_left2right=match_left2right1;  
    end;  
  end;
```

Also, obtain the maximum number of character in string2 that are also in the same order in string1 from right-to-left and record them in variable “match_right2left.”

```
  do i=1 to min(length(string1),length(string2)) by 1;  
    do j=i to 1 by -1;  
      if index(string1,substr(string2,i-j+1,j)) > 0 then match_right2left1=j;  
      if match_right2left1 > match_right2left then match_right2left=match_right2left1;  
    end;
```

```
end;  
end;
```

However, if string2 has greater length than string1, obtain the maximum number of character in string1 that are also in the same order in string2 from left-to-right and record them in variable "match_left2right."

```
else if length(string2) >= length(string1) then do;
```

```
do i=1 to min(length(string2),length(string1)) by 1;  
do j=min(length(string2),length(string1))+1-i to 1 by -1;  
if index(string2,substr(string1,i,j)) >0 then match_left2right1=j;  
if match_left2right1 > match_left2right then match_left2right=match_left2right1;  
end;  
end;
```

Also, obtain the maximum number of character in string1 that are also in the same order in string2 from right-to-left and record them in variable "match_right2left."

```
do i=1 to min(length(string2),length(string1)) by 1;  
do j=i to 1 by -1;  
if index(string2,substr(string1,i-j+1,j)) > 0 then match_right2left1=j;  
if match_right2left1 > match_right2left then match_right2left=match_right2left1;  
end;  
end;  
end;
```

Now we can calculate the proportion of common letters, return its value and end the function:

```
pcl = sum(match_left2right,match_right2left)/sum(length(string1),length(string2));  
if string1="" or string2="" then pcl=.;  
return(pcl);  
endsub;
```

Below is an example of calculation undertaken by the code using DATA step:

String1	String2	Match_ left2right	Match_ right2left	PCL
INTERNEURON PHARMACEUTICALS INC	INTERNEURON PHARMACEUTICALS INC	31	31	1.00
COMMONWEALTH BIOTECHNOLOGIES INC	COMMONWEALTH BIOTECHNOLOGIES IN	31	31	0.98
ACCESS ANYTIME BANCORP INC	ACCESS ANYTIME BANCORP	22	22	0.92
NETWORK CONNECTION INC (THE)	NETWORK CONNECTION INC	22	22	0.88
GENEREX BIOTECHNOLOGY CORPORATION	GENEREX BIOTECHNOLOGY CORP DEL	26	26	0.83
ALLOU HEALTH & BEAUTY INC	ALLOU HEALTH & BEAUTY CARE INC	22	22	0.80
SOUTH TEXAS DRILLING & EXPLORATION INC	SOUTH TEXAS DRILLING & EXPL INC	27	27	0.78
AMERICAN ORIENTAL BIOENGINEERING INC	AMERICAN ORIENTAL BIOENGINRG IN	26	26	0.78
CREATIVE COMPUTER APPLICATIONS INC	CREATIVE COMPUTER APPLICS INC	24	24	0.76
NETWORK EVENT THEATRE INC	NETWORK EVENT THEATER INC	19	19	0.76
TORTOISE ENERGY INFRASTRUCTURE CORPORATION	TORTOISE ENERGY INFRASTRUCT COR	27	27	0.74
SANTA CRUZ OPERATION (THE) INC	SANTA CRUZ OPERATION INC THE	21	21	0.72
GENERAL DATACOMM INDUSTRIES INC	GENERAL DATACOMM INDS INC	20	20	0.71
GENERAL DATACOMM INDUSTRIES INC	GENERAL DATACOMM INDS INC	20	20	0.71
EUROTECH LIMITED	EUROTECH LTD	10	10	0.71
CONSTELLATION ENERGY PARTNERS LLC	CONSTELLATION ENERGY PTNRS L L	22	22	0.70
FLEXIBLE SOLUTIONS INTERNATIONAL INC	FLEXIBLE SOLUTIONS INTL INC	22	22	0.70
SI DIAMOND TECHNOLOGY INC	SI DIAMOND TECH INC	15	15	0.68
PEOPLES BANCTRUST COMPANY INC (THE)	PEOPLES BANCTRUST CO INC	20	20	0.68
PERMA-FIX ENVIRONMENTAL SERVICES INC	PERMA FIX ENVIRONMENTAL SVCS IN	19	19	0.57
Fin Svcs, Ladenburg Thalmann Inc	Ladenburg Thalmann Financial Services Inc	19	19	0.52
MED-DESIGN CORPORATION (THE)	MED DESIGN CORP	11	11	0.51

Merging two datasets using SQL and PCL function

Using the above function we can calculate the proportion of common letters IF the two text strings are provided in the same row. However, we usually need to combine two datasets using inexact strings, which can be accomplished by combining the above PCL function with PROC SQL. This can be best illustrated by an example; let's say the "master" dataset has 2 observations for which we need to find the match from the "base" dataset that has 21 observations. In this example it is known at the outset that common name variations and inaccuracies may be present. Each file has a unique ID number for each observation. This ID number is quite useful in the process of combining datasets, initially important variables unnecessary to the matching process do not need to be carried along, but only a smaller file of each origin containing only the variables that are useful in matching can be retained for the task, and later the ID variables can be used to capture the remaining information.

In viewing this data to attempt to match-merge the records, examples are of common inexact fields such as name, which do not agree precisely, and are not usually matched by the computer, but that the names are really of the same company. The proposed method can accommodate such match-merges. This is illustrated below.

Master dataset:

Masterid	String1
101	ACCESS ANYTIME BANCORP INC
102	NETWORK CONNECTION INC (THE)
103	Fin Svcs, Ladenburg Thalmann Inc

Base dataset:

Baseid	String2
1	INTERNEURON PHARMACEUTICALS INC
2	COMMONWEALTH BIOTECHNOLOGIES IN
3	ACCESS ANYTIME BANCORP
4	NETWORK CONNECTION INC
5	GENEREX BIOTECHNOLOGY CORP DEL
6	AMERICAN ORIENTAL BIOENGINRG IN
7	ALLOU HEALTH & BEAUTY CARE INC
8	NETWORK EVENT THEATER INC
9	CREATIVE COMPUTER APPLICS INC
10	SOUTH TEXAS DRILLING & EXPL INC
11	TORTOISE ENERGY INFRASTRUCT COR
12	PERMA FIX ENVIRONMENTAL SVCS IN
13	GENERAL DATACOMM INDS INC
14	SANTA CRUZ OPERATION INC THE
15	GENERAL DATACOMM INDS INC
16	CONSTELLATION ENERGY PTNRS L L
17	FLEXIBLE SOLUTIONS INTL INC
18	PEOPLES BANCTRUST CO INC
19	EUROTECH LTD
20	MED DESIGN CORP
21	SI DIAMOND TECH INC
22	Ladenburg Thalmann Financial Services Inc

The following SQL statements will provide us with the best match for the string variable in master dataset:

```
proc sql;

create table matched_output as

select *, pcl(masterdataset.string1, basebasedataset.string2) as pcl,

from masterdataset , basebasedataset

group by masterdataset.masterid

having max(pcl)=pcl;

quit;
```

Following is the output produced by the above sql statement:

masterid	string1	string2	baseid	pcl
101	ACCESS ANYTIME BANCORP INC	ACCESS ANYTIME BANCORP	3	0.92
102	NETWORK CONNECTION INC (THE)	NETWORK CONNECTION INC	4	0.88
103	Fin Svcs, Ladenburg Thalmann Inc	Ladenburg Thalmann Financial Services Inc	22	0.52

Making Use of Several Inexact Character Variables to Combine Datasets

In several instances there is more than one inexact character variables available in the two datasets. This additional information can easily be incorporated in a single measure to combine the two datasets. Lets assume we have three inexact character variables (NAME, TICKER, and CUSIP²). The following SQL statement can incorporate all the information from the three strings into PCL:

```
proc sql;

create table matched_output as
```

² Usually CUSIP is a exact identifier, however, several time it is not, for example, different digits for CUSIP in the two datasets.

```

select *, mean(pcl(master_name, base_name), pcl(master_ticker, base_ticker),
pcl(master_CUSIP , base_CUSIP)) as PCL

from master,base

group by masterid

having max(pcl)=pcl;

quit;

```

Performance Analysis with Large Sample

Performance analysis of the above procedure is conducted on the data obtained from Placement Tracker with 12000 observations and WRDS with 50000 observations. Former data is treated as “master” dataset, while the latter is the “base” dataset. These two datasets had three inexact common variables, namely, Firm Name, Ticker, and Cusip³. Before, implementing the above procedure, firm names (in both datasets) were made uppercase and few symbols were removed using the following data step:

```

data dataset;set dataset;
name=upcase(name);
name=compress(name, '.');
name=compress(name, '@');
name=compress(name, ':');
name=compress(name, '/');
name=compress(name, '&');
name=compress(name, '*');
name=compress(name, ',');
name=compress(name, '-');
name=compress(name, '!');
name=compress(name);
run;

```

³ CUSIP in the master dataset is missing or altered while downloading using excel.

Due to the large amount of data being processed PROC SQL could not conduct the calculations in one full sweep. Therefore, “master” data was divided in ten observations at a time and then matched with the “base” data; these matched observations were then combined into one file. The following macro implements this process.

```
Data master1;set master;  
serialNumber=_n_;  
run;  
%let j = 0;  
%MACRO DivideAndConquer;  
%DO i = 10 %TO 12000 %by 10;  
data master2;set master1;  
serialNumber > &j and serialNumber <=&i;  
run;  
%let j=&i;  
  
proc sql;  
create table matched_output10 as  
select *, mean(pcl(master_name, base_name), pcl(master_ticker, base_ticker),  
pcl(master_CUSIP , base_CUSIP)) as PCL  
from master2,base  
group by masterid  
having max(pcl)=pcl;  
quit;
```

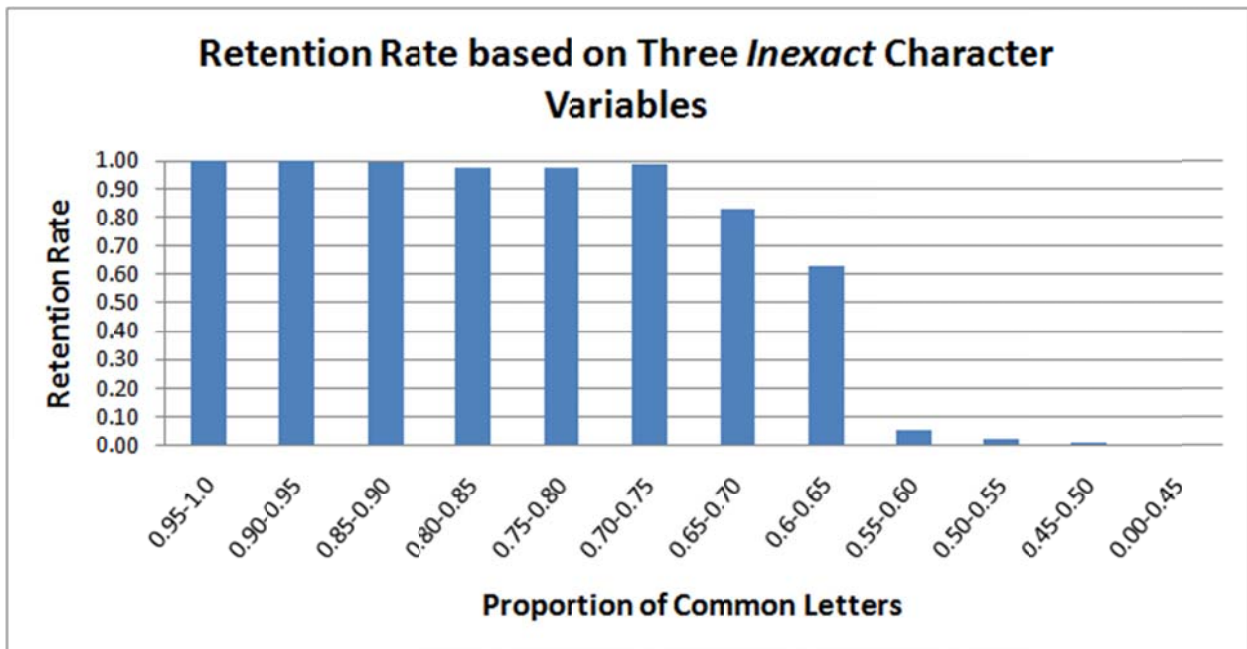
```
PROC APPEND BASE= matched_output DATA= matched_output10;run;
```

```
DM 'clear log';
```

```
%END;
```

```
%MEND DivideAndConquer;
```

The matches produced by the above were manually checked to make sure the variation of the names refer to the same company. The following graph describes the “Retention Rate,” which is defined as the number of correct matches divided by the total matches in a given range of PCL. These results illustrate that the retention rate is very high for PCL above 0.85, but starts to fall sharply below PCL of 0.65 and reaches zero for PCL below 0.45.



Conclusion

Often errors or variations in reported names can occur because two datasets have different sources of data collection which could result in different levels in precision and accuracy. The

process described here can replicate the 'human eye' to successfully create such matches that are inexact, and keep necessary manual inspections to a minimum. This technique will save considerable amount of time and effort for researchers. The strength of the procedure lies in providing a numerical measure for the match between the two strings – proportion of common letters. The procedure goes a step further and is able to include the information regarding a match from several inexact character variables. The ideas in this paper provide a concrete and workable solution that will considerably improve researchers' efficiency by decreasing the visual workload.