

# **The Complexities of an Address**

Barry Mullins, MPH

Health Integrity LLC, Dallas, TX

## **ABSTRACT**

Addresses can come in many forms that are messy, unorganized and not standardized. Trying to match people by their addresses could take days to manually match, but with SAS can be done quickly. However, without standardizing the addresses, SAS can miss matches or even give you false positives.

This paper will discuss the steps to standardizing and fuzzy matching addresses. I will show examples of standardizing addresses and parsing out the pieces using PRXCHANGE and TRANWRD. Also, I will discuss the importance of removing common words, like street, to help increase the accuracy of matching addresses to one another using COMPGED. The examples are depicted in SAS EG 4.2.

## **INTRODUCTION**

There are many reasons a person may want to match people, businesses, and addresses from multiple databases. Businesses may have old mailing lists and want to merge them to a new client list without having duplicates. Investigators may want to know if services are being provided to multiple people at the same address or if multiple people live at the same address for finding insurance fraud. A hospital may want to confirm the most recent billing address for a clinic visit. Matching the addresses could be manually performed, however, that could be a full-time job considering there can easily be over 100,000 entries in one of these databases. SAS can do most of the work for you with a short program and once the program is written it can be used for future queries or is even transferrable to other areas of study.

Significant problems with matching is that there may not be a common identifier, such as Social Security number, there is chance for human error, such as typographical errors, and there may be no standardization in the way an address is entered. Without a common identifier, fuzzy matching is going to have to be used. Fuzzy matching is the use of multiple variables to match on, such as name, street address, zip code and/or date of birth. In order to fuzzy match effectively, the values of a variable need to be standardized and then scored. In the following examples, TRANWRD and PRX (Perl Regular Expressions) are going to be used to standardize the data, and COMPGED will score the results for address matching. However, please note that these functions are not the only ways to fuzzy match.

## **STANDARDIZATION OF DATA**

No matter the field of study, data is dirty. It is messy, incomplete or even missing, and addresses are no different. To effectively and efficiently match the address data, it first needs to be standardized. To standardize the addresses, common values will be deleted and then different parts of the address will be parsed out into new variables.

The following examples will use the two spreadsheets below. There are 12 matches and 1 nonmatching address in each dataset.

A	B	C	D	E	A	B	C	D	E		
1	Street Address	City	State	Zip_Code	Name	1	Street Address	City	State	Zip_Code	Name
2	101 Park Street Suite 100	HOUSTON	TX	77036	Mia Modano	2	101 Park Street Suite 100	HOUSTON	TX	77036	Mia Modano
3	453 Grey St #4	DALLAS	TX	75254	Jamie Benn	3	453 Grey St	DALLAS	TX	75254	Jamie Benn
4	900 Main Rd, PO Box 54	HOUSTON	TX	77074	Lou Erickson	4	900 Main Road	HOUSTON	TX	77074	Lou Erickson
5	300 Northwest Bank Street	HOUSTON	TX	77099	Alex Chasson	5	200 NW Bank St	HOUSTON	TX	77099	Alex Chasson
6	9876 Jupiter Avenue, Room 3	HOUSTON	TX	77072	Erk Cole	6	9876 Jupiter Ave, Rm 3	HOUSTON	TX	77072	Erk Cole
7	7555 Manne Court	HOUSTON	TX	77042	Kari Lehtonen	7	7555 Manne Court	HOUSTON	TX	77042	Kari Lehtonen
8	894 Franklin Blvd	HOUSTON	TX	77042	Brenden Dillon	8	894 Franklin Boulevard	HOUSTON	TX	77042	Brenden Dillon
9	11 Green Drive	SAN ANTONIO	TX	78238	Venson Fisher	9	103 Green DR, Ste 12, box 100	SAN ANTONIO	TX	78238	Venson Fisher
10	65 Dallas Lane	SAN ANTONIO	TX	78238	Cody Eakin	10	65 Dallas Lane	SAN ANTONIO	TX	78238	Cody Eakin
11	7200 San Lorenzo Av	FORT WORTH	TX	76110	Trevor Daley	11	7200 San Lorenzo Av	FORT WORTH	TX	76110	Trevor Daley
12	123 First Boulevard	MCGALLEN	TX	78504	Stephane Ribidas	12	123 First Blvd	MCGALLEN	TX	78504	Stephane Ribidas
13	99 Transport Highway	SAN ANTONIO	TX	78219	Sergei Gonchar	13	99 Transport Highway	SAN ANTONIO	TX	78219	Sergei Gonchar
14	12 SAS Circle	SACHSE	TX	75048	Ray Whitney	14	54 Youth Drive	SACHSE	TX	75048	Lane MacDemid
15						15					
16						16					
17						17					

There are two lists of addresses that need to be matched. As can be seen, there are matches, but they are not exact. These lists could easily be manually matched since they are small, but with records of 100,000 or more, it would not be efficient.

The first step in standardizing is to create SAS Macros that clean the data and parse the values into new variables.

### Macro 1 – Address

```
%MACRO ADDRESS (VAR=,VAR2=);
&VAR = TRANWRD(&VAR,"NORTH ","N ");
&VAR = TRANWRD(&VAR,"NORTHWEST ","NW ");
&VAR = TRANWRD(&VAR,"NORTHEAST ","NE ");
&VAR = TRANWRD(&VAR,"SOUTH ","S ");
&VAR = TRANWRD(&VAR,"SOUTHWEST ","SW ");
&VAR = TRANWRD(&VAR,"SOUTHEAST ","SE ");
&VAR = TRANWRD(&VAR,"EAST ","E ");
&VAR = TRANWRD(&VAR,"WEST ","W ");
```

The SAS character function, TRANWRD, has several uses and one of these is replacing words with new words or strings. This function was used to shorten directions.

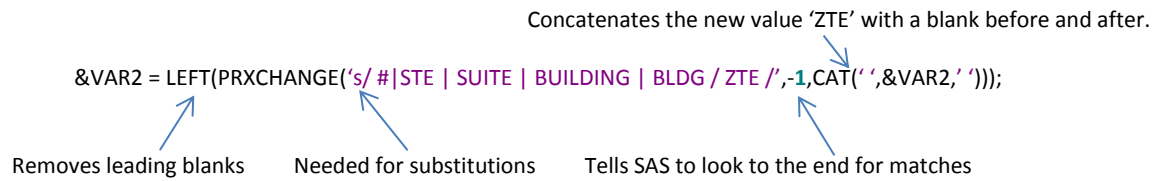
```
If _n_=1 then do;
  keys=prxparse("/s\sST |\sSTREET |\sSAVE |\sAV |\sAVENUE |\sDR |\sDRIVE |\sLN |\sLANE |\sRD |\sROAD |\sPKWY
|\sPARKWAY |\sBLVD |\sBOULEVARD |\sPL |\sPLACE |\sPLAZA |\sCT |\sCRT |\sCOURT |\sCIR
|\sCIRCLE /I");
end;
retain keys;

&VAR2 = prxchange(keys,-1,&VAR);
```

The Perl Regular Expression functions, PRXPARSE and PRXCHANGE, are then used. PRX functions look at patterns in the data, but for this purpose, exact character strings were used. PRXPARSE “holds” the patterns for other PRX functions. In the next step it is seen that the PRXCHANGE function finds the patterns that the PRXPARSE (keys) is holding and converts them to the new pattern, which is blanks. The loop scans each address and replaces the common words, such as Street and DR, as blanks. This helps to increase the accuracy of scoring the matches.

```
&VAR2 = LEFT(PRXCHANGE('s/ #|STE | SUITE | BUILDING | BLDG /ZTE /,-1,CAT(' ',&VAR2,' ')));
&VAR2 = LEFT(PRXCHANGE('s/ FRWY | FREEWAY / FWY /,-1,CAT(' ',&VAR2,' ')));
&VAR2 = LEFT(PRXCHANGE('s/ EXPWY | EXPRESSWAY | EXPWY / EXPY /,-1,CAT(' ',&VAR2,' ')));
&VAR2 = LEFT(PRXCHANGE('s/ HIWAY | HIGHWAY / HWY /,-1,CAT(' ',&VAR2,' ')));
&VAR2 = LEFT(PRXCHANGE('s/ RM | ROOM / ZRM /,-1,CAT(' ',&VAR2,' ')));
&VAR2 = LEFT(PRXCHANGE('s/ P O BOX | PO BOX | POBOX | PO BOX | P OBOX | PO BOS | BOX | POB | PMB | P O DRAWER | PO
DRAWER | POST OFFICE DRAWER | DRAWER | PBS BOX / ZPB /,-1,CAT(' ',&VAR2,' ')));
&VAR2 = TRANWRD(&VAR2,"ROUTE ","RTE ");
%MEND ADDRESS;
```

The last several statements of this macro convert the many iterations of each character string into one common string.



The PRXCHANGE function has several advantages over TRANWRD. First, you can have multiple strings to look for in one statement. Secondly, TRANWRD can change character strings that you do not want changed. For example, if you use TRANWRD to replace 'ST' with 'Street', then it will also change ST (Saint) Lucia, to Street Lucia.

The second macro (Breakup\_ADD) breaks the pieces of an address, e.g. suite, into new variables.

### Macro 2 – Breakup\_ADD

```

%MACRO BREAKUP_ADD (PATTERN=,VAR=,NUM=,NEWVAR=);

IF _N_ =1 THEN DO;
    RETAIN ExpID&NUM;
    PATTERN="/&PATTERN/";
    ExpID&NUM=PRXPARSE(PATTERN);
END;

CALL PRXSUBSTR(ExpID&NUM, &VAR, POSITION&NUM);

```

The first half of the macro creates a DO Loop that looks for the patterns in the addresses that were created in the first macro, e.g. ZRM. When the pattern is found, it outputs the starting position of the pattern.

```

IF POSITION&NUM = 1 THEN DO;
    MATCH = SUBSTR(&VAR,POSITION&NUM);
    &NEWVAR=MATCH;
END;

IF INDEX(&VAR,"&PATTERN") THEN &NEWVAR=SUBSTR(&VAR,POSITION&NUM);
IF INDEX(&VAR,"&PATTERN") THEN SUBSTR(&VAR,POSITION&NUM)="";

%MEND BREAKUP_ADD;

```

The last half searches for the pattern using the INDEX function. When the pattern is found, SAS parses out the pattern with extra information to the newly created variable using the starting position found from the first part of the macro. The second INDEX statement replaces the piece of information that was moved into a blank space.

Let's look at the macros in action to see exactly what is occurring.

```

DATA LIST1;
SET LIST1;

LIST1_NAME=COMPRESS(COMPBL(Name),".*<>");
STREET_LIST1=UPCASE(COMPRESS(COMPBL(Street_Address),".*<>"));
ZIP5=Zip_Code;

```

The first few statements delete characters and spaces from the Name and Street Address for each line.

```
%ADDRESS (VAR=STREET_LIST1,VAR2=STREET1);
```

The macro ADDRESS goes through the following steps.

1. Scans the variable, STREET\_LIST1, and shortens directions.
  - 200 Northwest Bark Street → 200 NW Bark Street
2. Creates a new variable, STREET1, from STREET\_LIST1 and deleted common character strings
  - 200 NW Bark Street → 200 NW Bark
3. Lastly, it changes other pieces of an address to a new string (pattern).
  - 1001 Park Street Suite 100 → 1001 Park ZTE 100

```
%BREAKUP_ADD (PATTERN=ZPB,VAR=STREET1,NUM=1,NEWVAR=PO_BOX_STREET);  
%BREAKUP_ADD (PATTERN=ZTE,VAR=STREET1,NUM=2,NEWVAR=SUITE_STREET);  
%BREAKUP_ADD (PATTERN=ZTE,VAR=PO_BOX_STREET,NUM=3,NEWVAR=SUITE_STREET);  
%BREAKUP_ADD (PATTERN=ZRM,VAR=STREET1,NUM=4,NEWVAR=RM_STREET);  
%BREAKUP_ADD (PATTERN=ZRM,VAR=SUITE_STREET,NUM=5,NEWVAR=RM_STREET);
```

The macro BREAKUP\_ADD goes through the following steps.

1. Searches variable for specified pattern and outputs the starting position.
  - The first %BREAKUP\_ADD macro statement searches for the pattern, 'ZPB', in variable, STREET1.
  - When pattern is found, it outputs the starting location to the Position variable.
2. If a pattern is found, SAS uses the starting position that was output and extracts the address from that point to the end of the string.
  - 900 Main ZPB 54
    - i. Variable: STREET1 → 900 Main
    - ii. Variable: PO\_BOX\_STREET → ZPB 54
3. This is completed for each pattern: ZPB, ZTE and ZRM.
4. In the case that there are two patterns in one string, two extra statements are ran for Suite, ZTE, and Room, ZRM.
  - 100 Green ZPB 100 ZTE 12
    - i. The first %BREAKUP\_ADD macro statement moves first pattern 'ZPB' and all information after it to the variable PO\_BOX\_STREET.
      - Variable: STREET1 → 100 Green
      - Variable: PO\_BOX\_STREET → ZPB 100 ZTE 12
    - ii. Then another statement scans the variable, PO\_BOX\_STREET, for the pattern 'ZTE', which then moves the suite information to the variable SUITE\_STREET.
      - Variable: PO\_BOX\_STREET → ZPB 100
      - Variable: SUITE\_STREET → ZTE 12

```
PO_BOX_STREET = STRIP(TRANWRD(PO_BOX_STREET,'ZPB',''));  
SUITE_STREET = STRIP(TRANWRD(SUITE_STREET,'ZTE',''));  
RM_STREET = STRIP(TRANWRD(RM_STREET,'ZRM',''));
```

```
PO_BOX_STREET=COMPRESS(PO_BOX_STREET,'#');  
SUITE_STREET=COMPRESS(SUITE_STREET,'#');
```

```
/*KEEP LIST1_NAME Street_Address STREET1 ZIP5 PO_BOX_STREET SUITE_STREET RM_STREET; */
RUN;
```

The last statements clean up the variables by removing the patterns.

```
PROC SORT DATA=LIST11 OUT=LIST1_NODUP NODUP;
BY STREET1;
RUN;
```

To remove duplicate streets, a PROC SORT with the NODUP option is run. This removes the duplicates that are in the same spreadsheet. For this project, duplicates were not wanted even if two entries had a different name in the LIST1\_NAME variable. If in your analysis, you want all duplicates, then this SAS code should not be run.

	Street_Address	LIST1_NAME	ZIP5	STREET1	ExpID1	POSITION1	PO_BOX_STREET	ExpID2	POSITION2	SUITE_STREET
1	1 Green Drive	Vernon Fidler	78238	1 GREEN	2	0		3	0	
2	1001 Park Street S...	Mike Modano	77036	1001 PARK	2	0		3	11 100	
3	12 SAS Circle	Ray Whitney	75048	12 SAS	2	0		3	0	
4	123 Flint Boulevard	Stephane Robidas	78504	123 FLINT	2	0		3	0	
5	200 Northwest Bar...	Alex Chiasson	77099	200 NW BARK	2	0		3	0	
6	453 Grey St #4	Jaime Benn	75254	453 GREY #4	2	0		3	0	
7	65 Dallas Lane	Cody Eakin	78238	65 DALLAS	2	0		3	0	
8	7200 San Lorenzo...	Trevor Daley	76110	7200 SAN LORE...	2	0		3	0	
9	7555 Mamie Court	Kari Lehtonen	77042	7555 MAMIE	2	0		3	0	
10	894 Franklin Blvd	Brenden Dillon	77042	894 FRANKLIN	2	0		3	0	
11	900 Main Rd, PD...	Loui Erickson	77074	900 MAIN	2	10 54		3	0	
12	9876 Jupiter Aven...	Erik Cole	77072	9876 JUPITER	2	0		3	0	
13	99 Transport High...	Sergei Gonchar	78219	99 TRANSPORT...	2	0		3	0	

As can be seen in the output above, the SAS Macros that were created scan the Street\_Address variable, shorten directions and remove common character strings that are output into the STREET1 variable. They also scan the data for the specified patterns, output the starting position into the Position variable and then extract that information from the address and put it in a new variable. The same program is run for the second list to get standardized addresses.

### SCORING THE MATCHES WITH COMPGED

Fuzzy matching/merging is not 100% accurate, so a measure is needed to help differentiate how accurate a match is by scoring. To score the “accuracy” of an address match, the SAS function COMPGED is used. COMPGED measures the “generalized edit difference” between character strings. A COMPGED score of 0 means that the character strings are exactly alike. The higher the score, the bigger the difference between the two strings. For example, the difference between “dog” and “do g” is an extra blank, and COMPGED would give it a score of 10 points.

More on Computing Generalized Edit Distances can be found at:

<http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a002206133.htm>

The example below is scoring the differences for the addresses before they are standardized.

```
Proc sql noprint ;
create table ACTUAL_STREET_MATCH as
```

```

select A.NAME as LIST1_NAME ,
       A.Street_Address as LIST1_STREET, /*Note: names should be changed if same name is in both datasets*/
       A.ZIP_CODE as ZIP5 ,
       B.NAME as LIST2_NAME ,
       B.Street_Address as LIST2_STREET ,
       B.ZIP_CODE as ZIP5S ,
compged(A.Street_Address,B.Street_Address,400,'I') AS SCORE

```

For COMPGED, the two variables being compared are put in parenthesis (A.variable1,B.variable2,# - the highest score, difference accepted, 'I' – states that case is not important ). The score of 400 is quite large and will give many false positives; however, since these addresses are not standardized, then they will have higher scores.

```

From LIST1 AS A,
LIST2 AS B
WHERE A.ZIP_CODE=B.ZIP_CODE

```

The WHERE statement tells SAS that only addresses with exactly the same Zip Code are to be scored with COMPGED. This creates a more accurate score because multiple zip codes could have the same street names. A 3 digit variable could also be created from the original Zip Code with the first 3 digits to make a broader search. This could be helpful if there are data entry errors in the Zip Codes.

```

AND A.Street_Address NE ""
AND CALCULATED SCORE<390

```

It also states that only scores less than 390 are to be included in the output.

```

Order by SCORE;
quit ;

```

	LIST1_NAME	LIST1_STREET	ZIP5	LIST2_NAME	LIST2_STREET	ZIP5S	SCORE
1	Trevor Daley	7200 San Lorenzo...	76110	Trevor Daley	7200 San Lorenzo...	76110	0
2	Cody Eakin	65 Dallas Lane	78238	Cody Eakin	65 Dallas Lane	78238	0
3	Sergei Gonchar	99 Transport Highw...	78219	Sergei Gonchar	99 Transport Highw...	78219	0
4	Mike Modano	1001 Park Street S...	77036	Mike Modano	101 Park Street Sui...	77036	20
5	Jaine Brenn	453 Grey St H#	75254	Jaine Brenn	453 Grey St	75254	30
6	Kari Lehtonen	7355 Marie Court	77042	Kari Lehtonen	7355 Marie Court	77042	100
7	Brenden Dillon	894 Franklin Blvd	77042	Brenden Dillon	894 Franklin Boule...	77042	350

Looking at the output above, there are 7 matches found out of the 12 matching addresses.

The last example shows scoring the differences for the addresses after standardization.

```

Proc sql noprint ;
create table STANDARD_STREET_MATCH as
select A.LIST1_NAME ,
       A.ZIP5 ,
       A.STREET1 ,
       A.PO_BOX_STREET ,
       A.SUITE_STREET ,
       A.RM_STREET ,
       B.LIST2_NAME ,
       B.ZIP5S ,
       B.STREET2 ,
       B.PO_BOX_STREETS ,
       B.SUITE_STREETS ,
       B.RM_STREETS ,
compged(A.STREET1,B.STREET2,250,'I') AS SCORE

```

The high score limit in this example for the standardized list is 250, which is lower than what was used for the actual addresses.

```

From LIST1_NODUP AS A,
LIST2_NODUP AS B
WHERE A.ZIP5=B.ZIP55
AND A.STREET1 NE ""
AND CALCULATED SCORE<240
order by SCORE;
quit ;

```

The output for the standardized scoring, below, has given much better results. All 12 matches out of 13 were found. The last two received scores of 100 and 200 because there were typographical errors that standardization cannot correct, however, the name and Zip Code are the same, so it would be very likely that these are matches.

	LIST1_NAME	ZIP5	STREET1	PO_BOX_STREET	SUITE_STREET	RM_STREET	LIST2_NAME	ZIP55	STREET2	PO_BOX_STREETS	SUITE_STREETS	RM_STREETS	SCORE
1	Serge Gonchar	78219	99 TRANSPORT HWY				Serge Gonchar	78219	99 TRANSPORT HWY				0
2	Lou Erickson	77074	900 MAIN	54			Lou Erickson	77074	900 MAIN				0
3	Cody Eakin	78238	65 DALLAS				Cody Eakin	78238	65 DALLAS				0
4	Trevor Daley	76110	7200 SAN LORENZO				Trevor Daley	76110	7200 SAN LORENZO				0
5	Brenden Dillon	77042	894 FRANKLIN				Brenden Dillon	77042	894 FRANKLIN				0
6	Stephane Flodidas	78504	123 FLINT				Stephane Flodidas	78504	123 FLINT				0
7	Erik Cole	77072	9876 JUPITER			3	Erik Cole	77072	9876 JUPITER			3	0
8	Alex Chasson	77099	200 NW BARK				Alex Chasson	77099	200 NW BARK				0
9	Mike Modano	77036	1001 PARK		100		Mike Modano	77036	101 PARK		100		20
10	Jame Benn	75254	453 GREY #4				Jame Benn	75254	453 GREY				90
11	Kari Lehtonen	77042	7555 MAMIE				Kari Lehtonen	77042	7555 MAMIE				100
12	Vernon Fider	78238	1 GREEN				Vernon Fider	78238	100 GREEN	100	12		200

## CONCLUSION

This paper has presented several key functions in SAS that can help with fuzzy matching, TRANWRD and PRX functions. Fuzzy matching is not 100%, but with this skillset, you can drastically cut the time down in matching addresses. Just remember: Clean data is good data.

## REFERENCES

- 1.) SAS(R) 9.2 Language Reference: Dictionary, Fourth Edition
- 2.) An Introduction to Perl Regular Expressions in SAS 9. Ron Cody, Robert Wood Johnson Medical School, Piscataway, NJ. SAS Institute Inc. Proceedings of the Thirty-first Annual SAS Users Group International Conference 2006.
- 3.) The Basics of the PRX Functions. David Cassell, Design Pathways, Corvallis, OR. Proceedings of the SAS Global Forum 2007.
- 4.) Fuzzy Matching using the COMPGED Function. Paulette Staum, Paul Waldron Consulting, West Nyack, NY. Proceedings of Northeast SAS User Group 2007.

## CONTACT

For comments and questions, please contact the author at:

Barry Mullins, MPH

[mullinsb@healthintegrity.org](mailto:mullinsb@healthintegrity.org)