SESUG Paper 69-2021

Utilizing SAS® MACROS to Deduplicate Your Data

William Zachary Smith, RTI International

ABSTRACT

Data deduplication is an imperative step for researchers when producing high-quality data products. If survey respondents do not have a unique identifier, data are particularly prone to data duplication since participants could have completed a survey multiple times. To produce quality data, researchers must identify and remove duplicate records. This paper explains the use of a SAS® macro that creates matching scores between every possible combinational pair of respondents in a dataset. This macro allows users to input specific datasets, variables, and score criteria that will be used to identify potential duplicates. The output of this macro will provide you with a list of potential matches based on their specifications, sorted by match probability to provide researchers with a quick and efficient way to deduplicate their datasets.

INTRODUCTION

RTI International is responsible for many large-scale surveys across the country, and identifying duplicate respondents in our data is always an important step in ensuring the delivery of high-quality data products and client satisfaction. One such survey that utilizes this paper's deduplication macro is titled the Survey of Earned Doctorates (SED). Contracted by the National Science Foundation (NSF), the SED survey is an annual census of all individuals receiving a research doctorate from an accredited U.S. institution in a given academic year, and collects information such as educational history, demographic characteristics and postgraduation plans.

When processing the survey data, duplicate data must be identified to ensure that respondents in a given cycle have one record per degree, and respondents who completed the survey for a prior year degree can be identified. Often respondents will take the survey a few months before they graduate, and due to unforeseen circumstances may have to pause their education and finish their degree later. When resuming their degree and finishing their doctorate, institutions will require the respondent to take the survey again, and this creates duplicate records in the survey data.

Since all survey responses are assigned a unique identification number, this macro – titled **"%duplicate_identif**y" – is used to identify these duplicates by comparing respondents' provided information, and calculating match scores between every possible pair of respondents. This macro is relatively easy to use, and only requires a few assumptions before being able to plug in this macro's parameters and begin identifying deduplicates in your dataset(s).

DATA ASSUMPTIONS

Before using this macro, you must be sure that specific variables exist in your data that give you the ability to make adequate comparisons. At the minimum, this macro requires a dataset that contains a respondent's birthdate, sex, and social security number (SSN) in character formats. It is also recommended that the data contains a unique identification number for each respondent, a respondent's first name and last name, and other miscellaneous variables that can be used for identification (this paper will also use a respondent's middle name, high school graduation year and their birth country). An example of a simulated data structure that follows these conditions can be seen below in Figure 1:

	id	first_name	last_name	middle_Name	ssn	birthdate_char	sex	hsgradyear	birthcountry
1	1	Francklin	Chritchley	Theo	246-61-8035	09/12/1958	Male	2004	United States
2	2	Cointon	Renne	Bidget	478-83-3409	11/06/1958	Male	1992	Indonesia
3	3	Maddi	Bolzmann	Jeth	783-35-4669	10/30/2010	Male	1990	Poland
4	4	Andee	Crolly	Desiree	708-62-0095	04/09/1987	Female	2000	Philippines
5	5	Garret	Trubshawe	Jodi	403-53-4323	06/11/1982	Female	2004	Portugal
6	6	Dasha	Seaborn	Dulci	859-99-6420	07/21/1949	Female	1998	China
7	7	Ottilie	Gaishson	Zolly	441-29-9445	05/13/1996	Female	2005	Indonesia
8	8	Florida	Kaes	Lethia	704-47-1357	12/09/1977	Female	1985	Sweden
9	9	Miller	Craker	Donelle	220-41-4064	08/31/1966	Female	2008	Czech Republic
10	10	Olivie	Hayer	Leigha	213-83-9114	02/09/1975	Female	2011	China

Figure 1: Example of a Proper Data Structure for the %duplicate_identify Macro[1]

MACRO PARAMETERS

Before being able to call upon **%duplicate_identify**, an understanding of this macro's parameters are recommended. The parameters are described below, with the full code used to build this macro available in Appendix A.

```
%duplicate_identify(dataset1,dataset2,blockingvar,compvar1,compvar2,
compvar3,compvar4,compvar5,compscore,sex_var,ssn4_var,ssn4score,
birthdate_var,birthdatescore,blockssn4ind,blockbirthind);
```

dataset1: This parameter correspondents to the dataset that you want deduplicated.

dataset2: This parameter allows users to specify a 2nd dataset that they want deduplicated. This allows users to determine if there are duplicate records between datasets. If you are only interested in deduplicating a single dataset (only looking for duplicate records within a single dataset), this parameter must be the same as dataset1.

blockingvar: This parameter corresponds to the variable that you want every potential duplicate pair identified by this macro to have an exact match on. For example, if you require all social security numbers to be an exact match when determining duplicates, you will specify social security number as your blocking variable. If you have multiple blocking variables you want to use it's recommended to run this macro multiple times, specifying different blocking variables each time and then synthesizing the resulting outputs across the multiple blocking variable configurations for the best results. Defining and using a blocking variable is required for the use of this macro, otherwise SAS will produce an error.

compvar1-5: These parameters correspond to the comparison variables you would like to be used in the matching score calculation, in addition to birthdate, sex and SSN, and it's recommended that *compvar1-compvar5* variables be in character format. This macro needs a minimum of 3 comparison variables and can have up to 5. If you use less than 5, leave the unused *compvar4*/*compvar5* parameters blank. The **compged()** function is used within this macro to create matching scores on each comparison variable specified for each potential match pair by computing the generalized edit distance for each string. The generalized edit distance represents the minimum-cost sequence of operations required to turn one string into another string. The smaller the number, the closer the match, as each operation required to turn one string into another adds on to the total "cost" (SAS Institute Inc. 2016). Please refer to the SAS

[1] To protect all SED respondents and their personally identifiable information, a dataset with these conditions was randomly generated for the use of this paper from the data simulation website www.mockaroo.com.

documentation on the **compged()** function, provided in the references, for a full list of what certain differences between two strings "costs". The final match score is created by summing across these comparison variables, a social security number matching score, and a birthdate matching score. The code used in this macro to compute these scores is shown below, using the code for the macro variable *compar4* as an example:

```
%if &compvar4^='' %then %do;
&compvar4._comp=compged(&compvar4._data1,&compvar4._data2);
%end;

%if &compvar4= %then %do;
&compvar4._comp=0;
%end;
```

compscore: This parameter corresponds to highest match score you're comfortable with after summing the **compged()** scores calculated for each comparison variable. The comparison score calculation can be seen below.

```
comp_score=&compvar1._comp+&compvar2._comp+&compvar3._comp+&compvar4._c
omp+&compvar5._comp;
```

Sex_var: This parameter corresponds to the variable name that represents sex in your dataset. It is assumed that all datasets that are run through this macro have a sex variable. Sex must be an exact match for this macro to consider a pair a potential match, and therefore no cutoff score needs to be specified.

Ssn4_var: This parameter corresponds to the variable name that represents SSN in your dataset. It is assumed that all datasets that are run through this macro have a social security number variable.

ssn4score: This parameter corresponds to the highest match score you're comfortable with after using the **compged()** function on SSNs. Since SSN is typically required to have a near perfect match when identifying duplicates, a low cutoff score of 25 is recommended. This will allow for potential typos and human errors that can occur when inputting their social security number.

birthdate_var: This parameter corresponds to the variable name that represents birthdate in your dataset. It is assumed that all datasets that are run through this macro have a birthdate variable.

birthdatescore: This parameter corresponds to the highest match score you're comfortable with after using the **compged()** function on birthdates. As with SSNs, birthdate matches are typically required to have a near perfect match when identifying duplicates and a lower cutoff score of 25 is recommended.

blockssn4ind: This is an indicator variable used to indicate if your blocking variable is SSN. If so, match scores will not be calculated for social security number since they are required to be a direct and perfect match as a blocking variable. A value of 1 means that SSN is used as a blocking variable, and a value of 0 means that SSN is not being used as a blocking variable.

blockbirthind: This is an indicator variable used to indicate if your blocking variable is the respondents' birthdates. If so, match scores will not be calculated for birthdates since they are required to be a direct

and perfect match as a blocking variable. A value of 1 means that birthdate is used as a blocking variable, and a value of 0 means that birthdate is not being used as a blocking variable.

EXAMPLE

Below is an example of calling **%duplicate_identify** using the dataset shown in Figure 1.

```
%duplicate_identify(database_match,database_match,ssn4,first_name,
last_name,middle_name,hsgradyear,birthcountry,
250,ssn,25,birthdate_char,25,1,0);
```

In this example, only one dataset is used, titled "database_match". SSN is defined as the blocking variable, meaning any potential duplicate pairs created by this macro will have exact matches on SSN.

The 5 comparison variables used are *first_name*, *last name*, *middle_name*, *hsgradyear* and *birthcountry* with a cutoff score of 250. This means some of the potential matches identified can have slightly different first names, last names, middle names, high school graduation years and birth countries so that typos and misspellings can be accounted for.

The social security number variable is titled SSN, with a cutoff score of 25, and the birthdate variable is titled *birthdate_char* with a cutoff score of 25. However, since this example uses SSN as the blocking variable, it will not be used in the match score calculations. Therefore, *blockssn4ind* is set to 1, and *blockbirthind* is set to 0.

At the end of the score calculations, the potential matches are sorted by their final match scores and outputted in descending order for manual review. Since a lower match score corresponds with a higher probability of a match, the pairs with the highest chances of being a match/duplicate are at the top. With the specifications used above, all potential matches identified by this macro will have matching social security numbers (on the last 4 digits), with very similar first names, last names, birthdays, high school graduation year and birth countries.

OUTPUT AND RESULTS

Below are some match pairs that were identified using the **%duplicate_identify** macro with the parameters specified above (for sake of space, high school graduation date and country born cannot be seen in these figures but are included in this macro's output).

At the top of the dataset are cases that are a perfect match, shown below in Figure 2:

	linkage_id	ssn	birthdate_char_data1	birthdate_char_data2	sex_data1	sex_data2	first_name_data1	first_name_data2	last_name_data1	last_name_data2	middle_name_data1	middle_name_data2
1	1	246-61-8035	09/12/1958	09/12/1958	Male	Male	Francklin	Francklin	Chritchley	Chritchley	Theo	Theo
2	2	478-83-3409	11/06/1958	11/06/1958	Male	Male	Cointon	Cointon	Renne	Renne	Bidget	Bidget
3	3	783-35-4669	10/30/2010	10/30/2010	Male	Male	Maddi	Maddi	Bolzmann	Bolzmann	Jeth	Jeth
4	4	708-62-0095	04/09/1987	04/09/1987	Female	Female	Andee	Andee	Crolly	Crolly	Desiree	Desiree
5	5	403-53-4323	06/11/1982	06/11/1982	Female	Female	Garret	Garret	Trubshawe	Trubshawe	Jodi	Jodi
6	6	859-99-6420	07/21/1949	07/21/1949	Female	Female	Dasha	Dasha	Seaborn	Seaborn	Dulci	Dulci

Figure 2: **Examples of highly matching pairs**

In the middle of the dataset are cases that are a perfect match, shown below in Figure 3:

	linkage_id	ssn	birthdate_char_data1	birthdate_char_data2	sex_data1	sex_data2	first_name_data1	first_name_data2	last_name_data1	last_name_data2	middle_name_data1	middle_name_data2
73	73	355-26-5964	10/21/1962	10/21/1963	Male	Male	Sioux	Sioux	Baseley	Baseley	Diena	Diena
74	74	627-72-3352	08/01/1969	07/01/1969	Female	Female	Vally	Villy	Burdas	Burdas	Gillian	Gillian

Figure 3: Examples of moderately matching pairs

Lastly, at the bottom of the dataset are cases that are likely not matches but do have some matching data, shown below in Figure 4:

	linkage_id	ssn	birthdate_char_data1	birthdate_char_data2	sex_data1	sex_data2	first_name_data1	first_name_data2	last_name_data1	last_name_data2	middle_name_data1	middle_name_data2
215	215	710-47-4353	12/27/1988	12/27/1988	Male	Male	Randy	Raoul	Manchin	Mosdill	Caitlyn	Catharina
216	216	710-47-4353	10/29/1987	12/27/1988	Male	Male	Bernie	Bernie	Gurry	Smith	Lucas	Lary

Figure 4: **Examples of low matching pairs**

CONCLUSION

Ultimately, this macro provides users with an easy-to-use process for identifying duplicates within a single dataset or between two datasets. However, there are some disadvantages that should be considered when using this macro, along with areas for future improvements.

1. Running this macro on very large datasets can take a considerable amount of time and processing resources. Once computed, the final output requires manual review and large datasets with a high volume of potential pairs can still be burdensome to analyze.
2. Less specific blocking variables can generate many false-positive matching pairs. For example, using respondent's last name as a blocking variable could identify multiple rows as duplicates for respondents with the same last name.
3. Only one blocking variable can be used with this macro. If multiple blocking variables are required, it is recommended that you run the **%duplicate_identify** macro multiple times with varying configurations, and then synthesizing your potential matches across configurations, for the best results.

Future improvements in progress include adding a process of identifying all match pairs that have an exact match across all specified comparison variables, social security number and birthdate, and then outputting these separately as "guaranteed matches" to reduce the burden of manual review.

REFERENCES

SAS Institute Inc. 2016. *SAS® 9.4 Functions and CALL Routines*: Reference, Fifth Edition, pp 488-494. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

William Zachary Smith
RTI International
919-541-6987
wzsmith@rti.org

Other brand and product names are trademarks of their respective companies.

APPENDIX A: CODE USED TO BUILD %DUPLICATE_IDENTIFY

```
/*Importing Datafile*/
proc import
datafile='/rtpnfil02/rtpnfil02_vol7/sed17/Users/wzsmith/Fake_Data_SESU
G.xlsx' out=database
DBMS=xlsx replace;
run;

/*Create Variables for Matching*/
data database_match;
retain id first_name last_name middle_Name ssn birthdate_char sex
hsgradyear birthcountry;
set database;
/*In Case Dataset does not have IDs for each case, create them*/
ID=_N_;
/*Make sure Birthdate is in character format*/
birthdate_char = put(birthdate,mmddyy10.);
keep id first_name last_name middle_Name ssn birthdate_char sex
hsgradyear birthcountry;
run;

/*Macro Definition*/
%macro
duplicate_identify(dataset1,dataset2,blockingvar,compvar1,compvar2,com
pvar3,compvar4,compvar5,compscore,ssn_var,ssnscore,birthdate_var,birth
datescore,blockssnind,blockbirthind);

/*Create duplicate dummy dataset for merging and comparisons*/
data matchdata1;
set &dataset1;
run;

data matchdata2;
set &dataset2;
run;

/*Rename every variable to eliminate any merging issues*/
proc sql;
select name||"="||cats(name,'_data1') into:rename_list separated by "
"
from dictionary.columns
where libname="WORK" and memname="MATCHDATA1";
quit;

proc datasets library=work nolist;
```

```
modify matchdata1;
rename &rename_list;
quit;

proc sql;
select name||"="||cats(name,'_data2') into:rename_list separated by "
"
from dictionary.columns
where libname="WORK" and memname="MATCHDATA2";
quit;

proc datasets library=work nolist;
modify matchdata2;
rename &rename_list;
quit;

/*Make it so both datasets are able to be merged on blocking
variable*/
data matchdata1_&blockingvar;
set matchdata1;
rename &blockingvar._data1=&blockingvar;
if &blockingvar._data1^='';
run;

data matchdata2_&blockingvar;
set matchdata2;
rename &blockingvar._data2=&blockingvar;
if &blockingvar._data2^='';
run;

/*Create dataset with all possible case pairings that have matching
blocking variable values*/
proc sort data=matchdata1_&blockingvar; by &blockingvar; run;
proc sort data=matchdata2_&blockingvar; by &blockingvar; run;

proc sql;
create table &blockingvar._matches as
select * from matchdata1_&blockingvar, matchdata2_&blockingvar
where
matchdata1_&blockingvar..&blockingvar=matchdata2_&blockingvar..&blocki
ngvar and
matchdata1_&blockingvar..id_data1^=matchdata2_&blockingvar..id_data2;
quit;

proc sort data=&blockingvar._matches nodupkey; by id_data1 id_data2;
run;

data &blockingvar._match_scores;
set &blockingvar._matches;

/*Create Scores for the comparison variables specified in the macro*/
%if &compvar1^='' %then %do;
```

```
&compvar1._comp=compged(&compvar1._data1,&compvar1._data2);
%end;

%if &compvar1= %then %do;
&compvar1._comp=0;
%end;

%if &compvar2^= %then %do;
&compvar2._comp=compged(&compvar2._data1,&compvar2._data2);
%end;

%if &compvar2= %then %do;
&compvar2._comp=0;
%end;

%if &compvar3^= %then %do;
&compvar3._comp=compged(&compvar3._data1,&compvar3._data2);
%end;

%if &compvar3= %then %do;
&compvar3._comp=0;
%end;

%if &compvar4^= %then %do;
&compvar4._comp=compged(&compvar4._data1,&compvar4._data2);
%end;

%if &compvar4= %then %do;
&compvar4._comp=0;
%end;

%if &compvar5^= %then %do;
&compvar5._comp=compged(&compvar5._data1,&compvar5._data2);
%end;

%if &compvar5= %then %do;
&compvar5._comp=0;
%end;

comp_score=&compvar1._comp+&compvar2._comp+&compvar3._comp+&compvar4._
comp+&compvar5._comp;

/*Create Birthdate Score (Assumed to have this variable)*/
%if &blockbirthind^=1 %then %do;
birthdate_comp=compged(&birthdate_var._data1,&birthdate_var._data2);
%end;

/*Create ssn Score (Assumed to have this variable)*/
%if &blockssnind^=1 %then %do;
ssn_comp=compged(&ssn_var._data1, &ssn_var._data2);
%end;
```

```
/*Output the matches that meet the criteria specified in the macro*/
%if &blockssnind=1 %then %do;
if (comp_score<=250 and sex_data1=sex_data2) or birthdate_comp<=25
then keep=1;
%end;

%if &blockbirthind=1 %then %do;
if (comp_score<=250 and sex_data1=sex_data2) or ssn_comp<=25 then
keep=1;
%end;

%if &blockbirthind^=1 and &blockssnind^=1 %then %do;
if (comp_score<=250 and sex_data1=sex_data2) or ssn_comp<=25 or
birthdate_comp<=25 then keep=1;
%end;

if keep=1;
run;

%if &blockssnind=1 %then %do;
proc sort data=&blockingvar._match_scores
out=&blockingvar._match_scores_sorted; by comp_score birthdate_comp;
run;
%end;

%if &blockbirthind=1 %then %do;
proc sort data=&blockingvar._match_scores
out=&blockingvar._match_scores_sorted; by comp_score ssn_comp; run;
%end;

%if &blockbirthind^=1 and &blockssnind^=1 %then %do;
proc sort data=&blockingvar._match_scores
out=&blockingvar._match_scores_sorted; by comp_score ssn_comp
birthdate_comp; run;
%end;

data &blockingvar._match_scores_sorted;
retain linkage_id id_data1 id_data2 &blockingvar &ssn_var._data1
&ssn_var.data2 &birthdate_var._data1 &birthdate_var._data2
sex_data1 sex_data2 &compvar1._data1 &compvar1._data2 &compvar2._data1
&compvar2._data2 &compvar3._data1 &compvar3._data2 &compvar4._data1
&compvar4._data2 &compvar5._data1 &compvar5._data2;
set &blockingvar._match_scores_sorted;
linkage_id=_N_;
run;
%mend;

%duplicate_identify(database_match,database_match,ssn,first_name,last_name,mi
ddle_name,hsgradyear,birthcountry,250,ssn,25,birthdate_char,25,1,0);
```