SAS® Macros for Data De-identification

Elliot Inman & David Maydew, SAS Institute Inc.

2015

ABSTRACT

By sharing data, researchers maximize the potential to understand the data and uncover meaningful insights. Sharing data can also help to ensure the reproducibility of scientific findings and reduce the chances of serious technical errors in an analysis. While it was once rare for researchers to share source data, even during peer review of academic research, data sharing is increasingly becoming the norm and, in some cases, mandated by law.

However, not all data can or should be shared. For example, some data include personally-identifiable information about unique individuals who provided information with an expectation of privacy. This is true for health care data that include confidential information about unique individuals such as a diagnoses, treatments, and other sensitive data. The original data may include names, addresses, and other details that would link study data to real, specific individuals. Those data are and should be protected and private. On the other hand, enabling researchers with a sincere scientific interest in those data access to the core elements of the data – without revealing any personally-identifiable information about unique patients – has the potential to advance our understanding of diseases and treatments.

To facilitate the sharing of meaningful, detailed data while protecting the privacy of study participants, researchers must implement de-identification strategies that respect participant privacy and data integrity. This paper presents a set of SAS® macros that facilitate the de-identification of data. The code demonstrates data anonymization, unique ID masking, targeted deletion of records and variables, and techniques to ensure cell size minimums for reporting. These macros operate across an entire library of data sets simultaneously, searching all data for issues of concern and systematically applying changes across all data sets to retain the relational integrity of the data. The code also implements a variety of control measures that allow the user to verify that the code ran as expected. Macro code is presented in full with comments that describe all of the functionality to allow users to deploy and modify the code to meet their specific needs.

INTRODUCTION

The purpose of this code is to facilitate the de-identification of data sets by simplifying several standard processes for doing so, including:

- Identifying and deleting specific variables;
- Identifying and deleting specific cases;
- Implementing an anonymous identifier; and
- Identifying cases that, together, do not meet minimum cell sizes for reporting.

The macros here perform these operations on one or more SAS data sets in a folder. For example, to delete a variable from a SAS data set, the user must specify a particular data set and the variable. That operation is performed on a single data set. Other operations are performed on all data sets in the folder. For example, the implementation of a unique identifier is executed across multiple data sets to maintain the referential links between the files such that "John Smith" is "1" in every file, "Mary Jones" is "2" in every file, and so on.

For some macros, separate data sets are created to save records and other data associated with the macro. For most macros, the results of the process are also reflected in a brief listing output showing the effects of the macro. If the macro ran, but actually took no action on any data set, no listing output or data set will be produced. For example, if the variable named to be removed is not in the data set, no listing is output. As always, users should check the log files of all programs to ensure code executed without warning or error.

The macros here are designed to simplify the process of de-identification without over-simplifying the process. Execution of some macros could be further automated, although the code to do so would be more complex and running such operations in a purely automated fashion might create other risks. As an example, users could modify this code so that all cases that violate minimal cell size conditions are automatically eliminated from all data sets. Would such deletions distort the meaning of the released data, impose a sample bias, or eliminate rare cases that are important for scientific study? Users are encouraged to give careful consideration to the application of these macros in fulfilling the goal of sharing meaningful data that protects the true identities of human subjects.

REQUIRED SKILLSET

This code assumes the user is familiar with SAS® BASE Language and SAS® MACRO Language and has access to this software to run the code. No statistical expertise is required. This code assumes the user has access to the original data sets and can create a temporary directory and copies of the target data sets so that destructive edits can be made.

LEGAL DISCLAIMER

The sample macro code included in this white paper may be used to support basic de-identification processes through the use of SAS® MACRO Language code. These macros are provided as-is and SAS has no obligation to provide continued support for their use. Use of any part or all of this code is not guaranteed to fulfill any legal or other requirements of any specific data governance standard or law governing de-identification of data. Users assume all responsibility for ensuring that the final publication of any de-identified data meets all legal requirements for release of such data. Users assume all legal and ethical responsibility for determining the appropriateness of releasing data analyzed and/or modified using this code.

OVERVIEW OF THE MACROS

Table 1 provides a summary of the macros, including a brief description of the functionality, scope, output, and the effects on the data sets processed. After the table, each macro is presented in its entirety, beginning with some additional startup code that should be executed at the start of the process. The startup code contains an initial *proc datasets* call to read all of the files in the folder into temporary memory and a later *proc datasets* call to write all of the files in memory permanently to the folder. Note that macros that are run against all files in the folder will also execute against any temporary files generated previously by other macros. In some cases, results reported will include analysis of those files as well; however, those instances will be obvious and will not affect the functioning of the individual macros.

Note that users ***should always*** create a second copy of all data to be de-identified to implement these macros. While not all macros are destructive, some are and edits to the files cannot be undone. To execute the code, the user must specify a library name for the location of the copies of the files to be de-identified.

MACRO	Brief Description	Scope	Destructive Deletion	Check / Output
Find_Variable	Identifies all files with a critical variable.	All files in directory, one variable.	No.	Printed list of files in which the variable was found.
Drop_Variable	Deletes one or more variables from one specific file.	One named file, multiple variables.	Yes. Deletes that specific variable(s).	Printed list of deleted variables.
Anonymous_ID_to_all	Replaces a known unique ID with a random ID across multiple files, maintaining referential integrity.	All files in directory, one variable (the unique ID).	Yes. Replaces values of the unique ID with new, randomly generated IDs.	Output SAS data set with previous and new ID values.
Record_Removal	Removes all cases based on a specific variable with a particular value.	All files in directory, record-level edits only.	Yes. Deletes the record(s) that fulfill the variable / value criteria.	Output SAS data set with removed records and printed list.
Find_Min_Cell_Size	Identifies records that, based on multiple variables, constitute less than X count of records in a file.	All files in directory, no edits.	No.	Printed list of cases that match search criteria and unique IDs for those cases.

Table 1. MACRO Definitions and Functionality

STARTUP CODE

```
/*  Declare the Library where the files to be de-identified are located.
DO NOT RUN these MACROs against the only version of the target data sets.  Make a copy of those
data sets and store them in a folder with only those data sets (e.g., C:\TO_BE_DEIDED) and
declare that library here.  These SAS data sets are read into work and destructive edits applied
to the data.*/

libname folder "C:\DEID";

* Reads all data sets in the library into work for editing;
proc datasets library=folder noprint;
       copy out=work;
       run;
quit;


/* This final proc datasets has been commented out here.  It should be uncommented and
run at completion of running one or more of the macros presented here.

Writes all data sets in the library into the permanent folder
proc datasets library=work noprint;
       copy out=folder;
       run;
quit;
*/
```

FIND_VARIABLE

```
/*  FIND_VARIABLE:  Identify All Files with a Critical Variable */
/*  FIND_VARIABLE finds that variable name or that keyword in a variable label and identifies the
data sets in the selected directory that include the variable or derivatives.  This MACRO will
run only one variable at a time, for example %FIND_VARIABLE(SSN) where "SSN" is the variable you
are seeking.
*/

%macro FIND_VARIABLE(var);
       proc contents data=_all_ out=Variable_Census noprint;
       run;

       data Variable_Census_Sample;
              set Variable_Census;
              SAS_Data_Set=MEMNAME;
              Variable=NAME;
              where find(upcase(NAME), upcase("&var")) or find(upcase(Label),
                     upcase("&var"));
              keep SAS_Data_Set Variable Label;
       run;

       proc print data=Variable_Census_Sample;
              id SAS_Data_Set Variable;
              var Label;
              title "Files with the variable or title including &var";
       run;

       proc datasets noprint;
              delete Variable_Census Variable_Census_Sample;
              title " ";
              *Reset title to nothing;
              run;
       quit;

%mend FIND_VARIABLE;

/* Example Call:
%FIND_VARIABLE(USUBJID)
*/
```

DROP_VARIABLE

```
/*  DROP_VARIABLE:  Drop one or more variables from a data set

This macro will drop the variables specified from the data set.
Thus, DROP_VARIABLE(mydata, v1) would drop the variable v1 from the data set mydata.
DROP_VARIABLE(mydata, v1 v2 v3) would drop the variables v1, v2, and v3 from the data set mydata
Output includes a listing of the variables dropped in the analysis.
*/


%macro DROP_VARIABLE(dsn, varlist);
        *Data set name and list of variables to be dropped (separated by spaces);

        proc contents data=&dsn out=before_&dsn noprint;
        run;

        data &dsn;
                set &dsn;
                drop &varlist;
        run;

        proc contents data=&dsn out=after_&dsn noprint;
        run;

        data before_after;
                /* Append the two data sets */
                set before_&dsn after_&dsn;
        run;

        proc freq data=before_after noprint;
                tables NAME / out=Before_and_After_Counts;
        run;

        proc print data=before_and_after_counts;
                id Name;
                var Count;
                title "Variables Dropped during This Process";
                where COUNT=1;
        run;

        proc datasets noprint;
                delete before_&dsn after_&dsn before_after before_and_after_Counts;
                title " ";
                run;
        quit;

%mend DROP_VARIABLE;

/* Example Call:
%DROP_VARIABLE(labs, sex age)
```

ANONYMOUS_ID_TO_ALL

```
/*  ANONYMOUS_ID_TO_ALL:  Finds the selected Identification Variable, generates a new
random ID, and applies that ID to all files maintaining the referential integrity of
those links.

WARNING:  Each time this is executed, a new masterlookup table is generated,
overwriting any previous assignment.


Requires a known unique ID used across records in multiple files.  Requires a single
master table that includes all of the possible IDs used across the data sets.

Saves a SAS data set called "masterlookup" which includes the original ID and the new ID.
Masterlookup is the only retained record of the original and generated ID.  It should be
saved by the user in a separate folder from this workspace to record this match should
it become necessary later to verify a match.

Save the master ID table "masterlookup" as a separate file for future reference.


In the MACRO call below:
masterdataset is the master file that includes all instances of the ID such as an
enrollment file or subject demographics file
mastervariable is the master ID variable that is used to identify unique cases
across all files in the directory.
*/


%macro ANONYMOUS_ID_TO_ALL(masterdataset, mastervariable);
        /*
        WARNING:  A new lookup is created each time this is executed, invalidating
        any previous lookup generated.

        The proc datasets immediately below this deletes ANY previous masterlookup data set.
        */
        proc datasets noprint;
                delete masterlookup;
                run;
        quit;

        data &masterdataset;
                set &masterdataset;
                random=ranuni(1);
                ANON_ID=.;
        run;

        proc sort data=&masterdataset;
                by random;
        run;

        data &masterdataset;
                retain ANON_ID;
                set &masterdataset;
                ANON_ID=_N_;
                drop random;
        run;
```

```
proc contents data=_ALL_ out=all_datasets(keep=memname) noprint;
run;

proc sort data=all_datasets nodupkey;
       by memname;
run;

data masterlookup;
       set &masterdataset;
       keep ANON_ID &mastervariable;
run;

proc sort data=masterlookup;
       by &mastervariable;
run;

data all_datasets;
       set all_datasets;
       row=_N_;
       call symput('numrows', _N_);
run;

%do i=1 %to &numrows;

       data _NULL_;
              set all_datasets;
              where &i=row;
              call symput('dsn', memname);
       run;

       proc sort data=&dsn;
              by &mastervariable;
       run;

       data &dsn;
              retain ANON_ID;
              merge &dsn (in=a) masterlookup;
              by &mastervariable;
              if a;
       run;

       data &dsn;
              set &dsn;
              drop &mastervariable;
       run;

       proc sort data=&dsn;
              by ANON_ID;
       run;

%end;

proc sort data=masterlookup;
       by ANON_ID;
run;
```

```
        proc datasets noprint;
                delete all_datasets;
                run;
        quit;

%mend ANONYMOUS_ID_TO_ALL;

/* Example Call:
%ANONYMOUS_ID_TO_ALL(dm,ID)
*/
```

RECORD_REMOVAL

```
/*  RECORD_REMOVAL:  Removes one or more records from ALL data sets in the directory
using a specific identifier.

If the variable chosen is a unique identifier, a unique case can be removed from
all files in the directory.  If the variable chosen includes a parameter for
which there are multiple cases, multiple records are deleted.   For example,
RECORD_REMOVAL(ID, 3) removes any record for which the variable "ID" is equal to
three (3).  In a master patient index or enrollment file, there may be only one
instance of that record.  In a file including all lab work performed, there may
be multiple records associated with patient #3.  For that file, all records for
patient #3 will be deleted.

*/


%macro RECORD_REMOVAL(variable, param);
        proc contents data=_ALL_ out=all_datasets(keep=memname) noprint;
        run;

        proc sort data=all_datasets nodupkey;
                by memname;
        run;

        data all_datasets;
                set all_datasets;
                row=_N_;
                call symput('numrows', _N_);
        run;

        %do i=1 %to &numrows;

                data _NULL_;
                        set all_datasets;
                        where &i=row;
                        call symput('dsn', memname);
                run;

                data transport;
                        retain dataset;
                        set &dsn;
                        length SAS_Data_Set $32;
                        length Reason $32;
                        SAS_Data_Set="&dsn";
                        Reason="'&variable' = &param";
                        where &variable=&param;
                        keep SAS_Data_Set reason;
                run;

                proc append base=deleted_records data=transport force;
                run;

                proc datasets noprint;
                        delete transport;
                        run;
                quit;
```

```
			data &dsn;
				set &dsn;
				if &variable=&param then delete;
			run;

		%end;

		proc datasets noprint;
			delete all_datasets;
			run;
		quit;

		proc sort data=deleted_records;
			by reason;
		run;

		proc freq data=deleted_records;
			tables SAS_Data_Set / nocum nopercent;
			by reason;
			title "Number of Records Removed from Each SAS Data Set by Reason Removed";
		run;

	%mend RECORD_REMOVAL;

	/* Example Call:
	%RECORD_REMOVAL(anon_id, 12)
	*/
```

FIND_MIN_CELL_SIZE

```
/*  FIND_MIN_CELL_SIZE:  Finds cases that, based on up to 10 variables, represent less than
a certain minimum cell size.

Multiple parameters can be set by the user:
minimumsize:  the minimum cell size
unique_ID:  variable name already in the data set that can uniquely identify specific records
numofvars  :  the number of variables to be used to identify a grouping (up to 10)
Variables  :  v1 to v10

For example, if a user wants to find all combinations of two variables (Location, Diagnosis) that
appear fewer than 5 times in the data set and the data include SSNs, the code to execute the
macro might look like:

%FIND_MIN_CELL_SIZE(5, SSN, 2, Location, Diagnosis)



*/
%macro FIND_MIN_CELL_SIZE(minimumsize, unique_id, numofvars, v1, v2, v3, v4,
                          v5, v6, v7, v8, v9, v10);
       %let allvars = &v1;

       %do i=2 %to &numofvars;
              %let allvars = &allvars., &&v&i;
       %end;

       proc contents data=_ALL_ out=all_datasets(keep=memname) noprint;
       run;

       proc sort data=all_datasets nodupkey;
              by memname;
       run;

       data all_datasets;
              set all_datasets;
              row=_N_;
              call symput('numrows', _N_);
       run;

       %do i=1 %to &numrows;

              data _NULL_;
                     set all_datasets;
                     where &i=row;
                     call symput('dsn', trim(memname));
              run;

              data &dsn._modified;
                     set &dsn;
                     Variable_Combination=catx(',', &allvars);
              run;

              /*  Identify combinations of variable values less than the minimum count */
              proc freq data=&dsn._modified noprint;
                     table Variable_Combination / out=Frequencies_of_Combination missing;
              run;
```

```
        data Frequencies_of_Combination;
               set Frequencies_of_Combination;
               length SAS_Data_Set $32;
               Count_in_File=Count;
               Percent_in_File=Percent;
               Below_Count="Y";
               SAS_Data_Set="&dsn";
               where count < &minimumsize;
        run;

        /*  Get unique_ids for cases matching the problematic variable combination */
        proc sort data=&dsn._modified;
               by Variable_Combination;
        run;

        proc sort data=Frequencies_of_Combination;
               by Variable_Combination;
        run;

        data IDs;
               merge &dsn._modified Frequencies_of_Combination;
               by Variable_Combination;
               keep SAS_Data_Set Variable_Combination &unique_id Below_Count;
        run;

        data IDs;
               set IDs;
               where Below_Count="Y";
        run;

        proc append base=Below_Minimum_Count data=Frequencies_of_Combination force;
        run;

        proc append base=Below_Count_IDs data=IDs force;
        run;

        proc datasets noprint;
               delete &dsn._modified Frequencies_of_Combination IDs;
               run;
        quit;

%end;

proc print data=Below_Minimum_Count;
        id Variable_Combination;
        var SAS_Data_Set Count_in_File Percent_in_File;
        title "Combinations of &allvars with Fewer than &minimumsize Observations";
run;

proc print data=Below_Count_IDs;
        id Variable_Combination;
        var SAS_Data_Set &unique_id;
        title "&unique_id for Cases Below Minimum Cell Count Criteria";
run;

proc datasets noprint;
```

```
                delete Below_Minimum_Count all_datasets;
                title " ";
                run;
        quit;

%mend FIND_MIN_CELL_SIZE;

/* Example Call:
%FIND_MIN_CELL_SIZE(5, ANON_ID, 2, Race, Sex)
*/
```

NEXT STEPS

These macros are intended to provide a quick start for programmers interested in de-identifying data. Implementation of these macros requires basic SAS programming skills, but this code has been written to minimize the challenge. In the interest of simplicity, we have repeated some code in multiple individual macros to make it easier to run each as standalone code. We have used a simple proc print for summarizing the execution of the macros, leaving that output pointed to the default location instead of, for example, creating an external running log of all macros executed.

Additional code could be written to add functionality such as the ability to shift dates or to automate the replacement of data that, due to combinations of critical variables, fail minimum cell size conditions. Other code could be written to provide end-users a greater range of options such as default deletion of any reference linking the original identifiers to anonymized identifiers. As with all SAS code, these macros could be executed via a user-friendly html-style interface that requires no programming skill or even a more automated process that executes macros in a batch script.

As these macros demonstrate, certain aspects of the de-identification process are mechanical. But de-identification is not a purely mechanical process. Given the legal and indeed moral aspects of commitments to maintain the privacy of individual human participants, there will always be a need for a careful human review of de-identification algorithms, processes, and methods and resulting research data to be shared.

While the sharing of research data was once rare, the process is becoming far more common and, in some cases, mandated by changes in the law and policy. From government agencies to academic journals, sharing of data is quickly becoming the norm. Sharing of data undoubtedly helps to ensure that research findings are sound. At the same time, sharing data can help to extend the value of data collected to a broader community of researchers, rapidly increasing the intellectual energy available to drive that research. It is our hope that these macros will help to facilitate the secure sharing of such data.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Elliot Inman, Ph.D.
Manager, Software Development
SAS Solutions OnDemand / Health Life Sciences R&D
801 SAS Campus Drive
Cary, NC 27513
Elliot.Inman@sas.com
www.sas.com