Paper 175-2011

HIPAA Compliance: Meeting Privacy Requirements in Combining Multiple Datasets using SAS Functions

Siew C. Ang, PhD, Paul M. Darden, MD
Department of Pediatrics, University of Oklahoma Health Sciences Center, Oklahoma City, OK

ABSTRACT

Research in healthcare often involves identified data. These datasets contain protected health information (PHI) that has to be removed to meet use and disclosure requirements for research under HIPAA. There are varying databases in which patient information is stored. These can be multiple datasets on the same patients or different patients that need to be combined for analysis. When combining multiple datasets for further analysis, it becomes a challenge in aligning these datasets, and at the same time, protecting PHI. This paper highlights the importance of examining the format of each dataset, and discusses steps to prepare the datasets before removing the PHI. Subsequently, we introduce, in progressively complex methods, combinations of SAS® functions, such as DO-LOOPs, SEED, RANUNI, NODUPKEY, EOF (end of file) functions, to replace the PPI with research appropriate identifiers, specifically variables embedded in medical records and administrative datasets. At the end of this paper, a MACRO LOOP routine with ARRAY, CALL VNAME, CALL SYMPUT, PROC APPEND that encompasses all these functions is presented to combine all six datasets for further analysis.

INTRODUCTION

Data collections are filled with protected health information (PHI), such as names, addresses, medical records, date of birth, social security numbers, account numbers, vehicle identifiers and telephone/fax numbers. Epidemiologists and researchers are required to meet HIPAA compliance by replacing the PHI in the datasets prior to analysis to protect patient privacy – the de-identification process.

Unique identifiers are easy to generate using SAS. However, when the need to combine multiple datasets arises, several challenges may arise. For example, the formats in which hospitals store their patient information may vary tremendously. The coding systems, the formats of the variables, and the degrees of inconsistencies in the datasets may be very different. In addition, care is required to avoid any duplicated identifiers between these datasets when trying to combine them.

This paper addresses the complexity when combining datasets from multiple sources assumed to be six hospitals in this case. This paper is divided into two sections. Section A addresses the issues encountered when preparing the datasets for de-identification. Section B addresses the de-identification process and provides solutions to some of the possible issues encountered. We will first demonstrate a simple way to remove PHI, follow by more complicated alternatives. Subsequently, we offer a more robust solution to generate the unique identifiers for each hospital dataset in separate program executions. The final method uses a MACRO procedure that improves on the earlier methods. It combines all the work to generate the random identifiers, ensure no duplicates, merge and append all six hospital datasets in a single execution. With each of these methods we discuss the assumptions, strengths and potential pitfalls.

DISCLAIMER: all the medical records throughout this paper have been modified for HIPAA compliance.

A) PREPARING THE DATASETS FOR DE-IDENTIFICATION

i) TYPES OF DATASETS

This step is crucial to successful de-identification. Notice the format in which each hospital stores their information in the datasets, whether it is a single row per patient, or multiple entries per patient. If the dataset has multiple entries per patient, it is essential to explore why these entries are generated, whether it is a single visit with multiple diagnoses stored in different rows, multiple procedures in the same visit or multiple visits on the same day. For this exercise it is assumed that we are working with datasets that contains one patient per row. This can be easily achieved using PROC SORT NODUPKEY OUT=. If needed, at the end of the de-identification process, the reduced dataset may be merged with the original dataset which has multiple entries per patient along with their new IDs. This last process deserves an entirely new paper by itself and will not be included in this paper.

ii) DATA CLEANING

Examine the variables in all six datasets to explore their format, levels, ranges and total counts using PROC FREQ. Pay particular attention to the variable that constitutes the unique identifier. Exclude any medical records which are not applicable, e.g. temporary IDs. Keep only the variables of interests in new datasets. Take note of the total number of observations for each dataset after excluding observations that are not applicable.

iii) VARIABLES ALIGNMENT ACROSS DATASETS

In order to concatenate the datasets at some point in time for further analysis, it is necessary to ensure the consistency of the information in the variables by aligning all variables by the variable names and formats.

B) DE-IDENTIFICATION PROCESS OF THE DATASETS

In this section, we demonstrate four methods for de-identifying the datasets in progressive complexity. In general, the first two methods use a version of random numbers as their unique identifiers. The last two methods treat the random numbers as intermediate step from which to create a new unique identifier. Specifically, the first method generates the random numbers 6 times, one for each dataset. The second method generates the random numbers for all six datasets in one step. The third method uses the random numbers as an intermediate step to randomize the order of the observations and then to create a new variable of equal intervals as identifiers for each dataset. In the final method, we introduce a condensed, neatly packaged MACRO LOOP that is inclusive of the third method whereby all six hospital datasets are processed in one execution step and concatenated as a finished product at the end of the execution.

One key requirement for success to all these de-identification methods is that the original datasets are not assigned a new ID based on any pre-sorted order of the original IDs. We want to assign the new IDs at random. An assumption for all these methods is that each dataset contains one patient per row without duplicates.

i) FIRST METHOD TO DE-IDENTIFICATION

Generating unique identifiers for each dataset

This is the simplest method to generate unique identifiers. We first generate the random numbers, round and sort them from the smallest to largest values, remove any possible duplicate numbers, merge them with Hospital A (HospA) to create new IDs (see Figure 1). Repeat this process for all other hospitals, i.e. HospB, HospC, HospD, HospE, and HospF.

```
data work.rand1;
   seed = 3689; *Step1;
      do _n_ = 1 to 236; *Step2;
            R2 = ranuni(seed); *Step3;
            R1 = round(R2*10000); *Step4;
            output ; *Step5;
         drop seed R2 ;
      end;
   run;
   proc sort data = work.rand1 nodupkey out = work.rand2; *Step6;
      by R1; run;

data work.merged_file;
      merge work.rand2 (in = n) work.HospA (in = d); * Step7;
      if d;
   run;
   proc print data = work.merged_file (obs = 5); *Step8;
      var mrec r1 sex race ins; run;

data New_HospA; set work.merged_file; *Step9;
      drop mrec;
      rename R1 = mrec1; run;
```

Step 1. Set a number for the seed, so that it generates the same random numbers each time one runs the program. Otherwise, if the seed is set to ‘0,’ it will generate a different set of values for each execution based on the computer time.

Step 2. This DO loop generates the total number of random numbers. Generate more than the actual observations to account for duplications. A suggestion is to create at least twice as many as one’s actual total. In the above example, HospA has n = 118, so we create 118*2=236 random numbers on the safe side.

Step 3. The function RANUNI(seed) uses the seed to return a random number, R2, generated from a uniform distribution on the interval (0,1).

Step 4. Variable R1 is the result of multiplying R2 times 10,000. The goal is to create a numerical variable, that is large enough to result in enough whole number IDs for all the observations. The next step is to remove the decimals. Either the ROUND or INT functions would work. ROUND function rounds up the decimals, INT function truncates the decimals.

Step 5. The Output statement, within the DO loop generates an observation in the dataset Rand1 for each iteration of the loop. “Drop seed R2” drops the variables “seed” and “R2.” Now we only have one column, R1, in dataset Rand1.

Step 6. PROC SORT NODUPKEY OUT = will ensure that the R1 in dataset Rand2 (the output dataset) has no duplicate values of R1. This is a critical step. If there are duplicate values you will inadvertently merge the data of two different patients or delete the data of one of these patients. Thus, a list of unique identifiers is created in dataset named Rand2.

Step 7. At the end of the MERGE procedure: HospA takes the first 118 values of R1. In this case “mrec” is the unique identifier only found in the file “HospA”. “if d” includes all the observations from dataset HospA. Note that the number of observations in the new dataset work.merged_file should be identical to work.HospA.

Step 8. Cross check to examine whether the new IDs are attached to each observation and that the merge goes correctly.

Step 9. Drop the variable “mrec” in the new dataset named New_HospA. Rename R1 to mrec1, which is the new unique identifier.

Repeat Step 1 to Step 10 for the other datasets: HospB to HospF.

Strengths:

a) This process is simple and easy to execute.
b) The SAS program is readily accessible, i.e. one may find this program in reference books or via the internet.

Caveats:

There are many caveats to this method:

a) The start and the end of random digits are between 0 and 1. While the set of random numbers generated may be controlled by fixing the seed to a specific value, the specific values of these random numbers can’t be controlled by the researchers.
b) The intervals between the unique identifiers are not the same, e.g. between observations #2 and #1, the difference for R1 is 44 – 10 = 34; but for observations #6 and #5, the difference is 128 - 79 = 49.
c) One could not tell the total observations in the dataset just by looking at the new IDs because the beginning value of the new IDs is not 1, and the incremental value is not 1 either.
d) Multiple executions are required to complete the de-identification process for all six datasets, thus increasing the chance for human errors.
e) One has to remember to make changes in Step1 and Step2 for each execution to account for total observations and to avoid duplicate IDs. It is also important to change the file names starting from Step 7 for each hospital dataset.
f) With this method, there is no protection against generating duplicate IDs between the 6 datasets. One suggestion to help debug this problem is to create an additional variable named Hospital that assigns HospA = 1, HospB = 2, etc… HospF = 6. When IDs are sorted, we could identify possible duplicated IDs and the hospitals they came from. In essence your unique ID would be a combination of mrec1 and hospital ID.
g) This method has some security weaknesses. The new ID has the same order as the old MREC. This means that if someone knows the sequence of the observations in the initial patient file they can re-identify the de-identified file.

Figure 1. Datasets in the Process of De-identification (Method 1)

a) HospA

mrec	sex	race	Ins	. . .
5358	M	W	Private	. . .
5022	F	W	Medicaid	. . .
5408	F	B	Medicaid	. . .
5103	M	B	Medicaid	. . .
5001	F	B	Medicaid	. . .

b) Merged_file

R1	mrec	sex	race	Ins	. . .
10	5358	M	W	Private	. . .
44	5022	F	W	Medicaid	. . .
172	5408	F	B	Medicaid	. . .
327	5103	M	B	Medicaid	. . .
447	5001	F	B	Medicaid	. . .

c) New_HospA

mrec1	sex	race	Ins	. . .
10	M	W	Private	. . .
44	F	W	Medicaid	. . .
172	F	B	Medicaid	. . .
327	M	B	Medicaid	. . .
447	F	B	Medicaid	. . .

Notation:
a) Original dataset with medical records (mrec, modified); b) Sorted R1 (random digits) added to the dataset;
c) R1 renamed as mrec1, mrec was dropped from the dataset, New_HospA.

ii) SECOND METHOD FOR DE-IDENTIFICATION

Generating unique identifiers for all six dataset in one step

This method is a slight variation from Method 1. Instead of executing "do _n_ = 1 to 236" in Step 2, we are generating random numbers twice the total of all six datasets, i.e. 118+123+…147 = 813*2 = 1626. Between Step 6 and Step 7, we need an additional step 6a, but two choices: first choice, we concatenate all six datasets into a master dataset, and then merge with the unique identifiers from Step 6. Second choice, we divide the unique identifiers into six parts and merge each part with each hospital dataset before removing the old IDs and concatenating all the datasets.

Both choices have pros and cons. The first choice is simpler but with several possible complications: a) these duplicated IDs would pose a potential complication for the merging process; b) we may not know which hospitals these records come from because they are combined. The second choice, though more tedious and cumbersome, avoids this issue and ensures the success of the merge process. To elaborate, dataset Rand2 in Step 6 is divided into six subsets named H1, H2, …, H6, see SAS code below. The de-identification process continues from Step 7 to Step 10 by merging H1 with HospA, H2 with HospB, etc... see Figure 2. You would need to check the number of random numbers after the de-duplication in step 6 above to assure that you have enough new ID numbers.

```
data H1 H2 H3 H4 H5 H6; set work.Rand2;
   if _n_ le 118 then output H1;
   else if 119 le _n_ le 241 then output H2;
   else if 242 le _n_ le 377 then output H3;
   …
   else if 667 le _n_ le 813 then output H6;
   run;
```

Step 6a. Output the random numbers to datasets H1 - H6 in order to merge with hospital datasets.

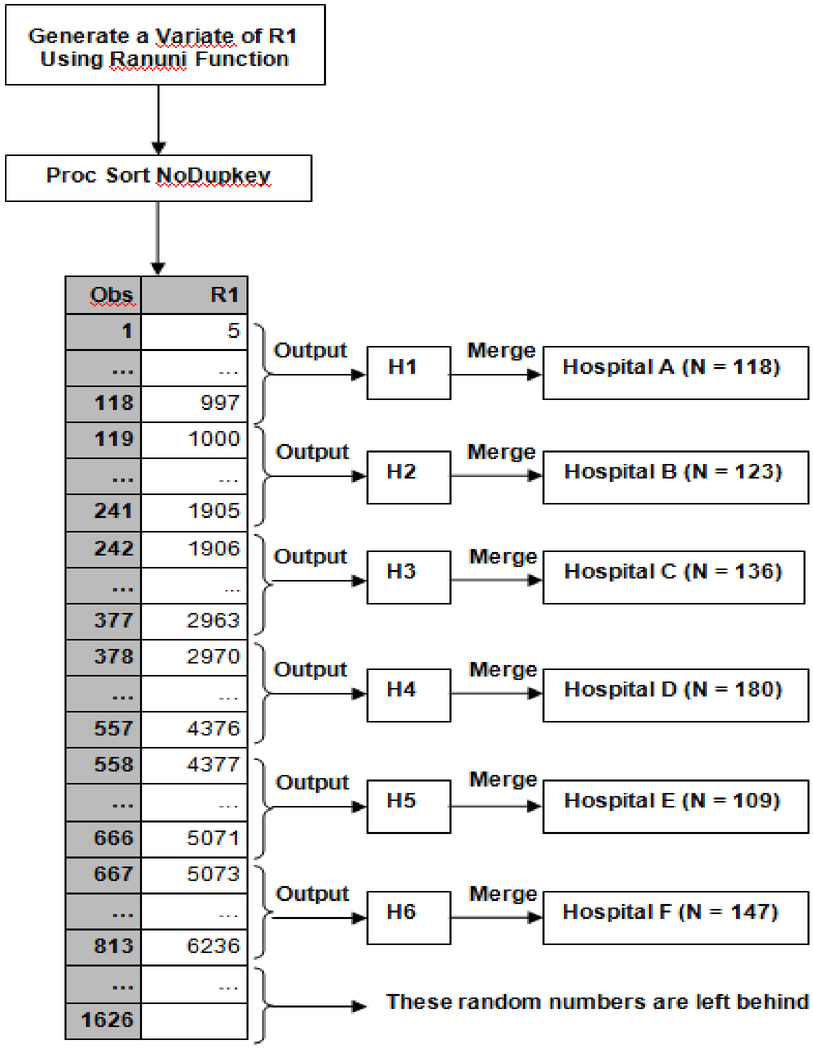
Strengths:

a) In addition to all the strengths from the Method 1, this method ensures no duplicate IDs cross all six datasets when they are concatenated for final analysis.
b) It does not require an extra variable named Hospital as suggested in Caveats (e), section above for a unique identification purposes. You may well wish to have a hospital ID for analytic reasons.

Caveats:

a) This method has weaknesses from Method 1 (a) to (d).
b) If all hospital datasets are appended prior to merging the unique identifiers, we may have conflict of IDs from the different hospitals, as discussed in an earlier paragraph.
c) On the other hand, if unique identifiers are created prior to merging with each hospital dataset, extra care is needed to create enough unique identifiers for each subset so that all the observations in each hospital dataset will have a unique identifier. There are higher chances for errors to occur.

Figure 2. Generating Unique Identifiers for Multiple Datasets in One Step (Method 2)



iii) THIRD METHOD OF DE-IDENTIFICATION

Generating a same-interval unique ID for each dataset

This method is a more systematic way to generate unique IDs across multiple datasets. Instead of directly using the random numbers as the unique identifiers, it is treated as a reference to be sorted, while a counter adding to a constant is used to create a new list of unique identifiers. Figure 3 summarizes the steps to de-identification with Method 3. Figure 4a-e illustrates the beginning, intermittent and final datasets for HospA for de-identification. Figure 4f and 4g indicate the final datasets for HospB and HospF before all six datasets are appended for further analysis.

```
data work.rand1;
     seed1 = 3689;
       do _n_ = 1 to 236;
          R1 = ranuni(seed1); *Step1;
              output ;
          drop seed1 ;
       end;
run;

data work.merged_file;
       merge work.rand1 (in = n) work.HospA (in = d); * Step2;
           if d;
       run;

proc sort data = work.merged_file ; * Step3;
       by R1; run;

data work.new_file; set work.merged_file; * Step4;
       count + 1;
           MREC2 = count + 1000 /*y1*/; * Step5;
       run;

proc print data = work.new_file (obs = 10); *Step6;
       var mrec R1 mrec2 sex dob; run;

data New_HospA; set work.new_file; *Step7;
       drop R1 count mrec;
   run;
```

Step 1. Generate a column of random numbers, R1, for each hospital, derived from a uniform distribution, with the range of 0 and 1 (all of them in decimals). In this case, we don't need to multiply R1 by 10,000 because R1 is only the ranking reference, not the actual unique identifiers. There may be duplicate values of R1. As R1 will not be the unique IDs this is not a problem.

Step 2. Merge Rand1 with HospA dataset in the new dataset named "merged_file." "if d" includes all the observations from dataset HospA.

Step 3. Sort by R1. It randomizes the observations in the dataset so that the sequence of the new ID does not reflect the order of the original dataset.

Step 4. In the dataset named "new_file," the records identifier is a sequential number starting with 1001 and incrementing by 1 up to the number of observations or patients in the dataset.

Step 5. A new variable named, "mrec2," is created by adding a constant *y1* = 1000 (for HospA), to the "count" variable. This constant may be changed to 10,000 or even 100,000 if needed, depending on the N size of the largest dataset. If it has 1000 or more observations, then y1=10,000. If N=10,000+, then y1 = 100,000. Adjusting this constant according to one's larges N size will ensure no duplicated identifiers when appending the datasets in future steps.

Step 6. Cross validating the variables "mrec2" and "mrec" with "sex" and "dob" to make sure the merge takes place properly.

Step 7. Drop the variables "R1", "count", "mrec" in the new dataset named "New_HospA."

Repeat Step 1 to Step 7 for all six hospital datasets. Change the constant, *y1*, in Step 5 whereby *y1* = 1000, 2000, 3000, etc… until 6000 for each iteration of HospA to HospF.

Strengths:

a) It is relatively simple to generate, just two extra steps compare to Method 1. The benefits outweigh the trouble it takes for these extra steps.
b) The new IDs are controlled by researchers: all datasets start with an investigator assigned number and ends with nth based on their total observations in the file. New IDs have intervals = 1. The total number of observations in the dataset is discernable by new ID.
c) Each hospital can be identifiable by the constant built into the system. It does not require a new variable for hospital identification as part of the unique identifier.
d) This is a stronger method because the order of the observations does not mirror the original patient dataset.

Caveats:

a) One has to perform a manual edit to the SAS code for each execution for each hospital dataset (see Figure 3).
b) It is cumbersome and time consuming. It opens up opportunity for simple human errors such as wrong placement, wrong number entries, or a corrupted dataset if a step is omitted.

Figure 3. Process of De-identification with Method 3

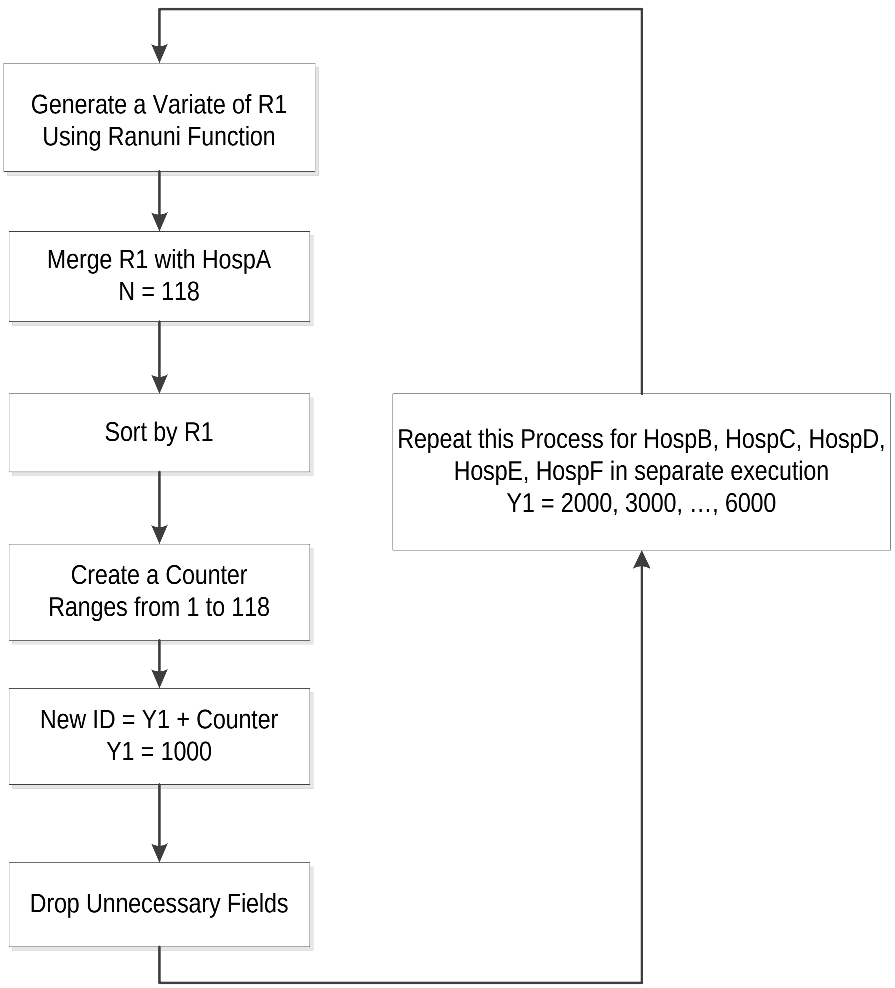


Figure 4. Datasets in the Process of De-identification (Method 3)

a) HospA

mrec	sex	race	Ins	
5358	M	W	Private	. . .
5022	F	W	Medicaid	. . .
5408	F	B	Medicaid	. . .
5103	M	B	Medicaid	. . .
5001	F	B	Medicaid	. . .

b) Merged_file

R1	mrec	sex	race	Ins
0.3269200736	5358	M	W	Private
0.657272037	5022	F	W	Medicaid
0.9518651636	5408	F	B	Medicaid
0.7919097472	5103	M	B	Medicaid
0.0501697483	5001	F	B	Medicaid

c) Sorted Merged_file

R1	mrec	sex	race	Ins
0.0501697483	5001	F	B	Medicaid
0.3269200736	5358	M	W	Private
0.657272037	5022	F	W	Medicaid
0.7919097472	5103	M	B	Medicaid
0.9518651636	5408	F	B	Medicaid

d) New_file

R1	mrec	sex	race	Ins	count	mrec2
0.0501697483	5001	F	B	Medicaid	1	1001
0.3269200736	5358	M	W	Private	2	1002
0.657272037	5022	F	W	Medicaid	3	1003
0.7919097472	5103	M	B	Medicaid	4	1004
0.9518651636	5408	F	B	Medicaid	5	1005

e) New_HospA

sex	race	Ins	mrec2	
F	B	Medicaid	1001	. . .
M	W	Private	1002	. . .
F	W	Medicaid	1003	. . .
M	B	Medicaid	1004	. . .
F	B	Medicaid	1005	. . .

f) New_HospB

sex	race	Ins	mrec2	
F	W	Medicaid	2001	. . .
M	W	Medicaid	2002	. . .
M	B	Medicaid	2003	. . .
M	W	Medicaid	2004	. . .
F	B	Medicaid	2005	. . .

g) New_HospF

sex	race	Ins	mrec2	
F	W	Private	6001	. . .
.	W	Private	6002	. . .
M	B	Medicaid	6003	. . .
F	W	Private	6004	. . .
M	B	Medicaid	6005	. . .

Notation:
a) Original dataset with mrec (modified); b) R1 (random digits) merged to the dataset; d) Sorted by R1;
c) Two extra columns added: a counter and mrec2, mrec2 = count + 1000, this 1000 could be 10,000 or 100,000 or more depending on the number of observations in the largest dataset; e) R1, mrec and count dropped from the dataset, New_HospA; f) Final dataset for HospB after de-identification; g) Final dataset for HospF after de-identification.

iv) FOURTH METHOD OF DE-IDENTIFICATION

A MACRO for generating unique random numbers, merge the random numbers with hospital datasets, create new random unique identifiers and append all hospital datasets in one package

In the following section, we introduce a MACRO that encompasses all steps in Method 3 and links these steps in a loop that executes for all six hospital de-identification process sequentially. This routine then outputs a final combined dataset that concatenates all six de-identified datasets. Information we may derive from the new ID variable in this final combined dataset is shown in the table below: the sequence number for each hospital, their first and last ID numbers, and their total observations based on the last ID. Table 1 indicates the summary information for the final concatenated dataset.

Table 1. New Patient Identification Information by Hospitals

Practices	Sequence Num	First ID	Last ID	N
Hospital A	1000	1001	1118	118
Hospital B	2000	2001	2123	123
Hospital C	3000	3001	3136	136
Hospital D	4000	4001	4180	180
Hospital E	5000	5001	5109	109
Hospital F	6000	6001	6147	147

Figure 5. Dataset XY1 with HospA to HospF Datasets Appended (Method 4)

R1	mrec	sex	race	Ins	var1	count	mrec2	. . .
0.0501697483	5001	F	B	Medicaid	1000	1	1001	. . .
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
0.9789089129	5169	F	B	Medicaid	1000	118	1118	. . .
0.0005107801	847	F	W	Medicaid	2000	1	2001	. . .
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
0.8125569895	807	F	B	Medicaid	2000	123	2123	. . .
0.0005107801	1426	M	W	.	3000	1	3001	. . .
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
0.9642754555	1923	F	B	.	3000	136	3136	. . .
0.0005107801	630	F	.	Private	4000	1	4001	. . .
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
0.1158247702	679	F	B	Private	4000	180	4180	. . .
0.0005107801	944	M	W	Medicaid	5000	1	5001	. . .
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
0.9746636694	909	F	W	Medicaid	5000	109	5109	. . .
0.0005107801	8871	F	W	Private	6000	1	6001	. . .
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
0.9001393293	8425	M	B	Medicaid	6000	147	6147	. . .

```
%let hosp = HospA HospB HospC HospD HospE HospF; /*Change the names if necessary*/

%macro LOOP;                        /*Call for the macro named 'LOOP'*/

data work.x;
   seed1 = 3689;
      do _n_ = 1 to 1626;
            R1 = ranuni(seed1); /*Refer to earlier explanation*/
            output ;
            drop seed1 ;
      end; run;

data work._null_;
   array hp {*} &hosp; /*Creates an array hp with 6 elements, HospA – HospF*/
   length hpname $10.; /*Set the length of names, character format*/
   do j = 1 to dim(hp);

/*Vname gets the name of the character variable from the array hp*/
      call vname (hp(j), hpname);

/*Call symput then assigns the name to the macro variable HP*/
      call symput ('HP'|| trim(left(j)), hpname);
   end;

/*The array hp has 6 dimensions, one for each hospitals assigned to the array.
Call Symput assigns this value of 6 to the macro variable HPdim */
      call symput ('HPdim', dim(hp)); run;

%do i = 1 %to &HPdim;

data xx&i.;
   merge work.x1 (in = n) work.&&HP&i. (in = d);
      if d;                /*Merge all the observations in hospital datasets*/
   run;

proc sort data = work.xx&i.;
   by R1; run;             /*Sort data x by R1 */

data work.xy&i.; set work.xx&i.;
   format var1 5.0;        /*Set var1 as numeric*/
   count + 1;              /*Count with the increment of 1, equal interval*/

/*Create a constant numeric variable with 4 digits. It could be 5 digits or more,
depends on the N size of the largest dataset*/
      var1 = &i.*1000;

/*Create a new ID by adding a constant to the counter*/
      mrec2 = var1 + count;
   run;

%end;

%do k = 2 %to &HPdim; /*Append data xy2 – xy6 to xy1 to create final dataset*/
   proc append base = xy1 data = xy&k.;
   run;
%end;

%mend LOOP;

%LOOP;

quit;
```

Strengths:

a) It has all the strengths in Method 3 (a) to (c).
b) No manual edits to the SAS codes are required, except for the first line of the SAS codes, i.e. the hospital dataset names.
c) It is less time consuming to execute compared to all other three methods above.
d) Less human errors are possible in the process of de-identification.

Caveats:

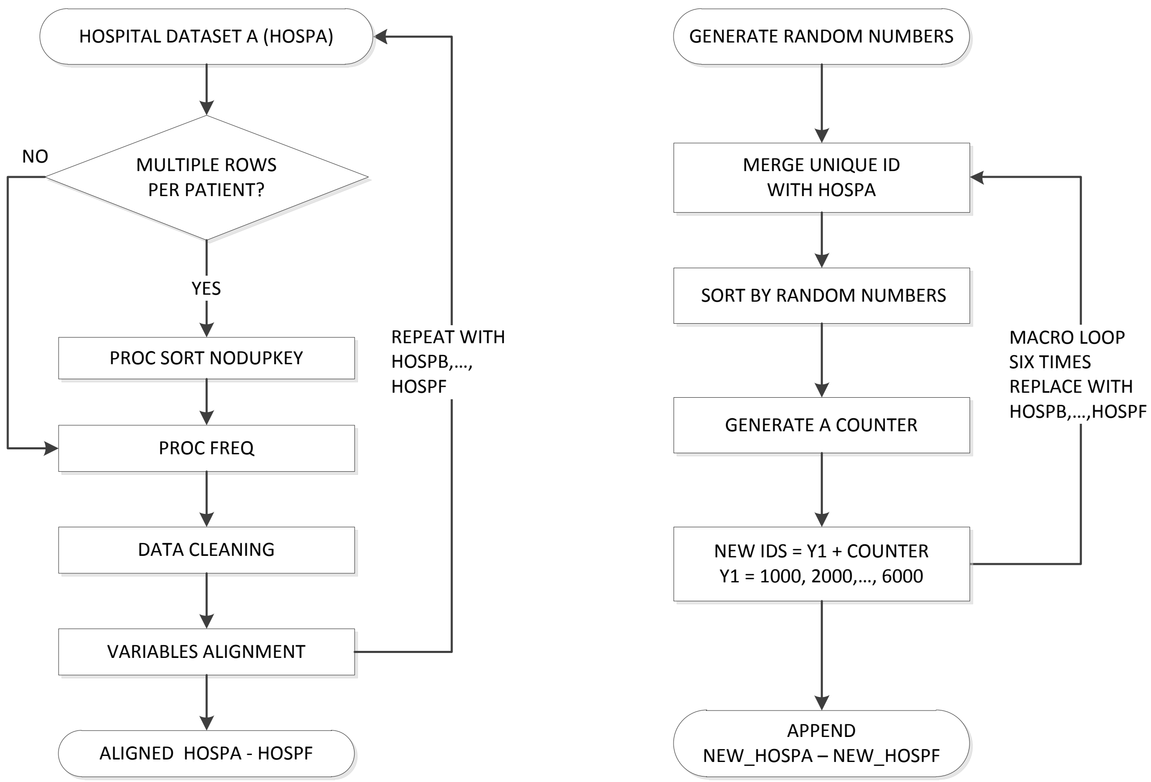
a) All variable formats and variable names in the hospital datasets must be aligned in similar order for this routine to work successfully.
b) Much more difficult to debug the SAS routine when errors actually occur, because the log will not show the error messages in MACRO loops. A solution is to execute a hospital dataset outside the loop to examine the log messages.

CONCLUSION

De-identified datasets are safer to work with and most boards dealing with human subjects (IRB) may require these procedures and certainly will appreciate this attention to patient confidentiality. In addition, HIPAA compliance is required by Federal law. At the same time, researchers need to have a general sense of where the data came from after de-identification and that these identifiers are unique and ideally random. If re-identifying may be necessary and is allowed by the IRB, a dataset that contains the new ID and old ID for each patient should be saved from dataset XY1 before deleting the old ID from XY1. This identification dataset should be saved and stored separately for security purposes. Figure 6a depicts the data preparation process prior to de-identification described in section A. Figure 6b portrays the actual de-identification process with method 4 from section B.

Figure 6. Overview of the Data Manipulation and De-identification Process

A) **B)**



REFERENCES

- Jiang, Yongwen & Hesser, Jana (March 2007). *Creating Standard Behavioral Risk Factor Surveillance System (BRFSS) Reports using SAS Macro*. Paper Presentation. BRFSS 2007 Annual Conference. Decatur, GA: CDC.

ACKNOWLEDGMENTS

We would like to thank Roxanne Brantes from Oklahoma and Helen Carey from Hawaii for editing the initial drafts.

CONTACT INFORMATION

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

Name:	Siew C. Ang, PhD
Enterprise:	Department of Pediatrics, University of Oklahoma Health Sciences Center
Address:	1200 N. Phillips Avenue, Suite 12400
City, State ZIP:	Oklahoma City, OK 73104
Work Phone:	(405) 271-4407
Fax:	(405) 271-8709
E-mail:	siewching-ang@ouhsc.edu