SAS® GLOBAL FORUM 2018

USERS PROGRAM

ZIPpy Safe Harbor De-Identification Macros

April 8 – 11 | Denver, CO

#SASGF

ZIPpy Safe Harbor De-Identification Macros

Abigail Chatfield[1,2], Jessica Parker[1], Paul Egeler[1]

[1]Spectrum Health Office of Research Administration, [2]Grand Valley State University

INTRODUCTION

- The U.S. Health Insurance Portability and Accountability Act (HIPAA) of 1996 Privacy Rule regulates the use of individually identifiable protected health information (PHI) to researchers (1)
- Honest Broker is an individual that separates the researcher from identifying PHI
- The Safe Harbor method is a common way to de-identify datasets by removing all 18 HIPAA Identifiers and editing the following variables (1,2):
  - Dates: Only Years are included
  - Zip Codes: Convert to 3 digit zip codes and set those that have a population less than 20,000 to 000
  - Age ≥ 90: Change to 90 or 90+
  - DOB Year ≥ 90th Year: Set year to 90th year
- To accomplish this task for our Honest Broker services we created MACROs to accomplish these tasks. For the zip code population decision 2010 census data is utilized.

BACKGROUND

- Download the 2010 Census Data to determine exclusionary three digit zip codes (3)
- Majority of states have either ≤10 or 11-20 3 digit Zip Codes (30% and 44%, respectively) (Figure 1).
- Of the 50 states in the US 12 of them include the excluded three digit zip codes listed in Table 1.

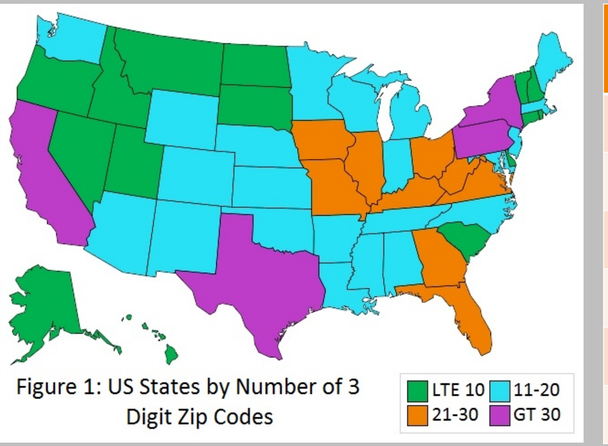

Figure 1: US States by Number of 3 Digit Zip Codes

Table 1: 3 Digit Zip Codes Converted to 000		
036	059	102
202	203	204
205	369	556
692	753	772
821	823	878
879	884	893

ZIP CODE MACRO

Transforms a 5 digit zip code to a 3 digit zip code according to HIPAA guidelines then removes the 5 digit variable.

```
%MACRO did_zip5(data=, zip5_var=);
   %if &vartype eq n %then %do;
   data &data;
      set &data;
      if &zip5_var<=00999 and &zip5_var>=00000 then
        zip3=substrn(&zip5_var,-1,3)*1;
      else if &zip5_var<=09999 and &zip5_var>=01000
        then zip3=substrn(&zip5_var,0,3)*1;
      else if &zip5_var>=10000 then
        zip3=substrn(&zip5_var,1,3)*1;
      if zip3 in (&zip_exc) then zip3=000;
         else zip3=zip3;
      format zip3 z3.;
      drop &zip5_var;
   run;
   %end;

   %else %if &vartype eq c %then %do;
   data &data;
      set &data;
      zip3=input(&zip5_var,3.);
      if zip3 in (&zip_exc) then zip3=000;
         else zip3=zip3;
      format zip3 z3.;
      drop &zip5_var;
   run;
   %end;
%MEND did_zip5;
```

Table 2: Zip Code MACRO Variable Changes		
Original 5 Digit Zip	Original 3 Digit Zip	HIPAA 3 Digit Zip
10292	102	000
20205	202	000
20210	202	000
20370	203	000
20380	203	000
20420	204	000
20425	204	000
20436	204	000
20442	204	000
20520	205	000
20525	205	000
20535	205	000
20540	205	000
75310	753	000
75323	753	000
75342	753	000
77234	772	000
77240	772	000
77255	772	000

ZIPpy Safe Harbor De-Identification Macros

Abigail Chatfield[1,2], Jessica Parker[1], Paul Egeler[1]

[1]Spectrum Health Office of Research Administration, [2]Grand Valley State University

DOB MACRO

Transforms date of birth year to the 90th year if the patient is greater than or equal to 90 years old.

```
%MACRO did_dob(data=,dob_var=,last_dt=);
   data &data;
      set &data;
      dob_year=year(&dob_var);
      if dob_year < (year(&last_dt)-90) then do;
         dob_year=year(&last_dt)-90;
      end;
      drop &dob_var;
   run;
%mend did_dob;
```

Table 3: Date of Birth MACRO Variable Changes

Original Date of Birth	New Date of Birth Year
1924	1926
1924	1927
1925	1926
1925	1927
1926	1926
1926	1927

AGE MACRO

Transforms age where patient is equal to or older than 90 years old to a numeric and character variable.

```
%macro did_age(data=,age_var=);
   data &data;
      set &data;
      if &age_var >= 90 then do;
         age_num=90; age_char='90+';
      end;
      else if age ne . then do;
         age_num=&age_var;
         age_char=put(&age_var,3.);
      end;
      drop &age_var;
   run;
%mend did_age;
```

Table 4: Age MACRO Variable Changes

Original Age	New Numeric Age	New Character Age
88	88	88
89	89	89
90	90	90+
91	90	90+
92	90	90+

DATE MACRO

Transforms dates passed through the macro into year instead of the actual date.

```
%macro did_dates(data=,dates=);
   %let dt_n=%sysfunc(countw(&dates));
   %do r=1 %to &dt_n;
   data &data;
      set &data;
      %scan(&dates,&r)_yrs=year(%scan(&dates,&r));
      drop %scan(&dates,&r);
   run;
   %end;
%mend did_dates;
```

Table 5: Date MACRO Variable Changes

Original Admit Date	New Admit Year
07/31/2016	2016
08/01/2016	2016
08/02/2016	2016
Original Death Date	**New Death Year**
07/21/2017	2017
07/23/2017	2017
07/25/2017	2017

CONCLUSION

- These pre built MACROs can reduce, if not eliminate, the confusion that comes with de-identifying PHI.
- The MACROs alone can be utilized, but they can also be utilized within the overall MACRO to de-identify a dataset with only one MACRO command line.
- Because the code is dynamic and utilized on different datasets with various data names it reduces the amount of code writing and increases efficiency.

REFERENCES

1. Electronic Code of Federal Regulations. 2018. “Other requirements relating to uses and disclosures of protected health information.” Accessed February 15, 2018. Available at https://www.ecfr.gov/cgi-bin/text-idx?SID=0ed1e92de1da217b01f07097dd9e62c6&mc=true&node=se45.1.164_1514&rgn=div8%3E%20Wednesday,%20September%2013,%202017

2. U.S. Department of Health & Human Services. 2018. “Guidance Regarding Methods for De-identification of Protected Health Information in Accordance with the Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule.” Accessed January 29, 2018. Available at https://www.hhs.gov/hipaa/for-professionals/privacy/special-topics/de-identification/index.html#zip.

3. United States Census Bureau. 2017. “American Fact Finder.” Accessed October 2, 2017. https://factfinder.census.gov/faces/nav/jsf/pages/download_center.xhtml.

SAS® GLOBAL FORUM 2018

April 8 – 11 | Denver, CO
Colorado Convention Center

#SASGF

2669-2018

ZIPpy Safe Harbor De-Identification Macros

Abigail A. Chatfield, Grand Valley State University and Spectrum Health Office of Research Administration; Jessica L. Parker and Paul W. Egeler, Spectrum Health Office of Research Administration

ABSTRACT

The U.S. Health Insurance Portability and Accountability Act of 1996 (HIPAA) Privacy Rule regulates the use of individually identifiable protected health information (PHI) to researchers. Researchers need to abstract the data that is within the patient charts for statistical purposes, but often times do not need PHI. Unfortunately, accessing this data exposes researchers to PHI. One way to protect patient data, while ensuring researchers are compliant with the HIPAA privacy rules, is to use de-identified data provided by an Honest Broker.

Stated within the law, de-identification may be achieved either by a) an expert using statistical and scientific principles and methods, or more commonly, through b) the Safe Harbor method. The Safe Harbor method requires the removal of 18 types of identifiable information, such as names, dates, addresses, phone numbers, account numbers, etc.

This paper will focus on the creation and utilization of Safe Harbor Macros to create a de-identified dataset by properly removing and changing necessary geographic information, identifiable dates, and medical information.

INTRODUCTION

The U.S. Health Insurance Portability and Accountability Act (HIPAA) of 1996 Privacy Rule regulates the use of individually identifiable protected health information (PHI) to researchers (Electronic Code of Federal Regulations, 2018). Researchers need to extract the data that is within the patient charts for statistical purposes, but often do not need PHI. Unfortunately, accessing this data from medical records exposes researchers to PHI, and in turn heightens the risk of potential PHI data breaches, and increases the Internal Review Board (IRB) administrative burden to the researcher. One way healthcare facilities are actively trying to protect patient data, while ensuring researchers are compliant with the HIPAA privacy rules, is to use de-identified data provided by an Honest Broker.

As defined by Merriam-Webster dictionary (2017), an Honest Broker is “a neutral mediator”. In the medical field, an Honest Broker acts as a buffer between the researcher and PHI, making sure the researcher only receives the information that was specified in their protocol. Often, the Honest Broker will simply remove all identifiable information and send the finalized dataset onward to the researcher. This is defined in the law as Safe Harbor de-identification. More specifically, the Safe Harbor method involves the removal of all 18 types of identifiable information, such as names, dates, addresses, ZIP codes, and phone numbers to be compliant with the HIPAA privacy rules.

The HIPAA privacy rules state that you may add more granularity to your analysis by using the initial 3-digits of a ZIP Code, provided that the geographic subdivision has a sufficient population size. In this paper, and with the use of SAS MACROs, we will guide you through the process of determining the population size of each 3-digit ZIP Code Tabulation Area (ZCTA) using publicly available U.S. Census data and then transform an existing 5-digit ZIP Code variable into a 3-digit ZCTA variable so it conforms with the HIPAA privacy rules. In addition, to creating the appropriate 3-digit ZCTA, the SAS MACROs will also transform and edit dates, date of birth, and age to meet HIPAA guidelines.

HIPAA REGULATIONS REGARDING DE-IDENTIFICATION

Table 1 lists the 18 identifiers defined by the HIPAA privacy rule (Electronic Code of Federal Regulations, 2018).

1. Names	7. Social security numbers	13. Device identifiers and serial numbers
2. All geographic subdivisions smaller than a state	8. Medical record numbers	14. Web universal resource locators (URLs)
3. All elements (except years) of dates directly related to an individual	9. Health plan beneficiary numbers	15. Internet protocol (IP) address numbers
4. Telephone numbers	10. Account numbers	16. Biometric identifiers, including fingerprints and voiceprints
5. Fax numbers	11. Certificate/license numbers	17. Full-face photographic images and any comparable images
6. Electronic mail addresses	12. Vehicle identifiers and serial numbers, including license plate numbers	18. Any other unique identifying number, characteristic, or code

Table 1. 18 HIPAA Identifiers

One of the above identifiers is "All geographic subdivisions smaller than a state" (U.S. Department of Health & Human Services, 2018). Therefore, a patient's street address, city, county, and ZIP code are all considered HIPAA identifiers and they must be removed from a dataset to avoid a confidentiality breach. However, there are instances where you can report the 3-digit ZCTA and still be compliant with the law. According to the U.S. Department of Health & Human Services (2018),

> *Covered entities may include the first three digits of the ZIP code if, according to the current publicly available data from the Bureau of the Census: (1) The geographic unit formed by combining all ZIP codes with the same three initial digits contains more than 20,000 people; or (2) the initial three digits of a ZIP code for all such geographic units containing 20,000 or fewer people is changed to 000.*

Provided your protocol with this additional information is accepted by the Institutional Review Board (IRB), you can gather more data regarding your population and potentially obtain more granular results.

In the United States, the majority of states have less than 10 or 11 to 20 3-digit ZCTAs (30% and 44%, respectively). In Figure 1 the number of 3-digit ZCTAs by state is shown.

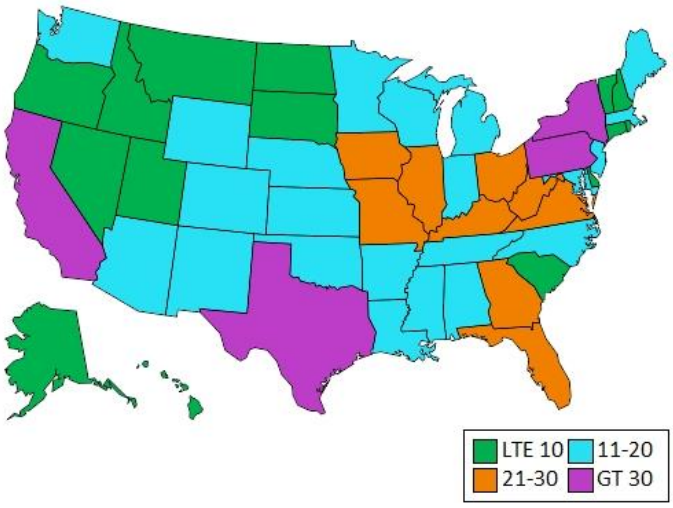

Figure 1. 3-Digit ZCTAs by State

Another identifier mentioned above is "All elements of dates" (U.S. Department of Health & Human Services, 2018). Aside from year, all aspects of dates must be removed from a dataset as well. Dates

such as death date, admission date, and discharge date must be stripped down to only the year of the event because these are considered identifiable dates.

Additionally, there is a stipulation for including a patient's age and date of birth year in research. "Ages that are explicitly stated, or implied, as over 89 years old must be recoded as 90 or above" (U.S. Department of Health & Human Services, 2018). This not only applies to the actual age, but also the date of birth year. Since there is a lower population of people in the older age groups, people that are over 89 years of age are considered at risk of being identified.

In the following sections of this paper, we will discuss how to address these situations to make sure the provided dataset by the honest broker is in accordance with the laws of patient privacy.

OVERVIEW OF GETTING CENSUS DATA

As mentioned previously, the U.S. Census data is publicly available online. The Census Bureau website has a convenient step-by-step guide on how to access and download the data a researcher needs. In this section, we will guide you through the process on how to download the population and ZIP Code data needed to create the 3-digit ZTCA (United States Census Bureau, 2017). The 2010 Census Data is the most recent file that is used for the below macros and output.

To begin, head to <https://factfinder.census.gov>. On the main toolbar, select Download Center – there are four short steps here to obtain the final data ZIP file:

1. In Step One, Start, select "I know the dataset or table(s) that I want to download" and hit the *Next* button.
2. In Step Two, Dataset, choose the program titled "Decennial Census" from the dropdown menu. A second dialog box will appear. In this box, click on the first entry titled "2010 SF1 100% Data" and hit *Add To Your Selections*. Click the *Next* button.
3. In Step Three, Geographies, choose the "….. 5-Digit ZIP Code Tabulation Area – 860" geographic type from the first dropdown menu. Once this is selected, a second dropdown menu and a dialog box will appear. We are not specifying a state; therefore, the second dropdown menu is to remain on the default selection. In the dialog box, click on the entry labeled "All 5-Digit ZIP Code Tabulation Areas within United States and Puerto Rico" and hit *Add To Your Selections*. Click the *Next* button.
4. In Step Four, Search Results, it prompts the user to select a table to download. Check the box for the table titled "TOTAL POPULATION". Click the *Download* button located at the bottom of the table. A pop-up box will appear – make sure the option box for "Include descriptive data element names" is checked and hit *Ok*.

At this point, the ZIP file will begin to download, and you will be prompted to choose a location on the computer to save it to.

EXPLANATION OF MACROS

Once the file containing the 5-digit ZIP codes and their populations is saved, you may start the process of creating the 3-digit ZCTAs, modifying the other HIPAA variables that can be kept in the dataset and removing variables that cannot be modified. The following macros are created to help with the de-identification process. They may be used individually, or in conjunction with each other, depending on the needs of your dataset. Here, we will explain the parameters needed for each macro to run, and the purpose of each. These entire macros may be found in the Appendix.

CENSUS MACRO

First, you assign the file directory DDIR to the path where the census data is stored:

```
%LET DDIR = %BQUOTE(<CENSUS DATA STORAGE PATH>);
```

You do this by using the %LET macro variable and the %BQUOTE macro function. The %BQUOTE function Mask(s) special characters and mnemonic operators in a resolved value at

macro execution (SAS, n.d.). For our purposes, it masks the apostrophe in our file path and runs without errors.

The first macro will read in the data set with the 5-digit ZIP codes and the populations. This is defined as the parameter census_data:

```
%MACRO zip3_exc (census_data=);
   data all_zips;
         length id $ 14 zip5 5 zip3 3 geography $ 11;
         infile "&ddir\&census_data..csv" firstobs = 3 dsd dlm = ",";
         input id zip5 geography ttl_pop;
```

From the import, you can see this macro assumes the ZIP code variable in the dataset is numeric. We will discuss character ZIP code variables in a following section.

The IF statements in conjunction with the SUBSTRN function below allows us to account for leading zeros in the 5-digit ZIP codes while creating the 3-digit ZIP code variable:

```
if zip5 <= 00999 and zip5 >= 00000 then zip3=substrn(zip5,-1,3);
   else if zip5 <= 09999 and zip5 >=01000 then zip3=substrn(zip5,0,3);
   else if zip5 >= 10000 then zip3=substrn(zip5,1,3);
   format zip5 z5. zip3 z3.;
run;
```

In the first statement, we are addressing the zip codes that consist of two leading zeros by starting at the position -1. In the second IF statement, we are addressing the zip codes with one leading zero by starting at the position 0. In the final IF statement, we are addressing the zip codes that do not have leading zeros by starting at position 1.

In the SQL procedure below, you will create an exclusion list of 3-digit ZCTAs, which consists of codes that have a population less than or equal to 20,000 people and store them in permanent SAS file called cens.zip3_exc:

```
libname cens "&ddir";
   proc sql;
         create table cens.zip3_exc as
         select zip3
         from all_zips
         group by zip3
         having sum(ttl_pop) le 20000;
   quit;
%MEND zip3_exc;
```

Using publicly available 2010 census data that you download, there are 18 ZCTA codes that are excluded. Table 2 lists these ZCTA codes

036	059	102
202	203	204
205	369	556
692	753	772
821	823	878
879	884	893

Table 2. Excluded 3-Digit ZIP Codes

DE-IDENTIFY ZIP MACRO

This macro will convert 5-digit ZIP Codes to the appropriate 3-digit ZCTAs and it accounts for the possibility of both numeric and character ZIP code variables. Here, you are also de-identifying the ZIP codes based upon the exclusion list created in the PROC SQL step above. The parameters that you will need to pass through the macro in Table 3.

Parameter	Description
DATA	The dataset that includes the ZIP codes and populations
ZIP5_VAR	The variable that contain the 5-digit ZIP code

Table 3. De-Identify ZIP Macro Parameters

```
%MACRO did_zip5(data=, zip5_var=);
   %local dsid varnum vartype rc;
   %let dsid    = %sysfunc(open(&data.,i));
   %let varnum  = %sysfunc(varnum(&dsid,&zip5_var.));
   %let vartype = %sysfunc(vartype(&dsid,&varnum));
   %let rc      = %sysfunc(close(&dsid));
```

The above code determines if the ZIP Code variable is numeric or character. The LOCAL macro statement indicates that you are creating four local macro variables that will exist only inside this DID_ZIP5 macro.

```
proc sql noprint;
   select zip3 into: zip_exc separated by ","
   from cens.zip3_exc;
quit;
```

The SQL procedure creates a macro variable list, identified as zip_exc, of the 3 digit zip codes that were found to be excluded from the census macro.

If the local macro variable VARTYPE returns a value of N, indicating a numeric variable, SAS will proceed with the following IF statement:

```
%if &vartype eq n %then %do;
   data &data;
     set &data;
         if &zip5_var <= 00999 and &zip5_var >= 00000 then
               zip3=substrn(&zip5_var,-1,3)*1;
         else if &zip5_var <= 09999 and &zip5_var >=01000 then
               zip3=substrn(&zip5_var,0,3)*1;
         else if &zip5_var >= 10000 then zip3=substrn(&zip5_var,1,3)*1;
         if zip3 in (&zip_exc) then zip3=000;
         else zip3=zip3;
         format zip3 z3.;
         drop &zip5_var;
   run;
%end;
```

Here, the new 3-digit ZIP code is created with the IF statements we already saw above. You then utilize the exclusionary macro list zip_exc created above. If the 3-digit code is within that exclusion list, it is changed to 000 for HIPAA privacy rules. If it is not included in that list, it is not changed.

If the local macro variable VARTYPE returns a value of C, indicating a character variable, the macro will proceed with the ELSE IF macro statement below:

```
%else %if &vartype eq c %then %do;
   data &data;
     set &data;
         zip3=input(&zip5_var,3.);
         if zip3 in (&zip_exc) then zip3=000;
               else zip3=zip3;
         format zip3 z3.;
         drop &zip5_var;
   run;
%end;
%MEND did_zip5;
```

When the 5-digit ZIP code variable is a character, SAS will create a numeric 3-digit ZIP code using the INPUT statement. If the numeric 3-digit ZIP code is in the exclusion list created in the PROC SQL code, then it is changed to 000. If not, it is not changed.

AGE MACRO

If the dataset contains the patient's age, this macro will check to see if the age is greater than or equal to 90 and then change it to comply with the HIPAA privacy rule. The parameters needed to pass through the age macro are in Table 4.

Parameter	Description
DATA	The name of the dataset being de-identified
AGE_VAR	The variable that contains patient age

Table 4. Age Macro Parameters

```
%MACRO did_age(data=, age_var=);
   data &data;
     set &data;
         if &age var >= 90 then do;
               age_num=90;
               age_char='90+';
         end;
         else if age ne . then do;
               age num=&age var;
               age_char=put(&age_var,3.);
         end;
         drop &age_var;
   run;
%MEND did_age;
```

Inside this macro, the IF statement creates two new variables: age_num and age_char, which are numeric and character age variables respectively. The age_char variable is created to identify those patients that are recorded as an age of 90 but are really 90 years or above, as instructed by the HIPAA privacy rule. If the patient is 90 years or older, age_num will be recorded as 90, and age_char will be recorded as '90+'. The two variables are created to note that the numeric age value was altered. If the patient is under 90 their actual age will be recorded in both the age_num and age_char variables.

DATE OF BIRTH MACRO

Additionally, if the dataset contains the patient's date of birth as a variable, the following macro will de-identify it to the patient's year of birth. The parameters needed for the date of birth macro are found in Table 5.

Parameter	Description
DATA	The name of the dataset being de-identified
DOB_VAR	The variable that contains patient date of birth
LAST_DT	The variable or date that the data was last pulled on

Table 5. Date of Birth Macro Parameters

```
%MACRO did_dob(data=,dob_var=,last_dt=);
   data &data;
     set &data;
         dob year=year(&dob var);
         if dob year < (year(&last dt)-90) then do;
               dob_year=year(&last_dt)-90;
         end;
         drop &dob_var;
   run;
%MEND did_dob;
```

The last_dt parameter is important because if a study is being conducted any amount of time after the data is collected, it allows you to know if the patient was over the age threshold when the data was collected and change the patient date of birth accordingly. This parameter can be a variable, such as admit date or surgery date, or it can be a specific date by using the MDY() function.

Based on this date, if the patient is under 90 years old, the year of the date of birth variable will be outputted to the new dob_year variable. If the patient is 90 years or older, the year of birth will be changed to the 90th year from the date that is indicated in last_dt.

DATE MACRO

By using the date macro, you can convert all date variables in your dataset to the year of the event. These events could be things such as admission date, discharge date, death date, or the surgery date. The parameters needed for the date macro are found in Table 6.

Parameter	Description
DATA	The name of the dataset being de-identified
DATES	The variables that contain dates needing to be de-identified

Table 6. Date Macro Parameters

```
%MACRO did_dates(data=, dates=);
   %let dt_n=%sysfunc(countw(&dates));

   %do r=1 %to &dt_n;
         data &data;
           set &data;
               %scan(&dates,&r)_yrs=year(%scan(&dates,&r));
               drop %scan(&dates,&r);
         run;
   %end;
%MEND did_dates;
```

In this macro, you can specify more than one variable for the DATES parameter to be changed. This parameter will be read in as a string. In order to read each variable separately a couple of things need to be included in the macro. First, you are creating a macro variable DT_N, which counts the number of dates listed in the DATES macro parameter. Then, the DO loop runs through the DATA step DT_N number of times and changes all variables listed in DATES to the respective year. Within the DO loop the SCAN function is used to search through the specified string to find the indicated argument. R is used as a counter to denote what variable in the string you want the SCAN function to select, where R is a number from 1 to DT_N.

DE-IDENTIFICATION MACRO

This final de-identification macro encompasses the components of the above macros and removes additional HIPAA components that cannot be modified. Table 7 describes the parameters needed to run through the de-identification macro.

Parameter	Description
RAND_SEED	An optional random seed
INPUT_DATA	The name of the dataset being de-identified
SUBJECT_ID	A unique subject identification variable
DROP_VARS	Name of variables to be dropped from the de-identification dataset
ZIP5_VAR	The 5-digit ZIP code variable to be converted to a 3-digit ZCTA
DATE_VAR	The variables that contain dates needing to be de-identified
AGE_VAR	The variable that contains patient age
DOB_VAR	The variable that contains patient date of birth
LAST_DT	The variable or date that the data was last pulled on

Table 7. De-Identification Macro Parameters

The RANDOM_SEED parameter is useful for consistency and reproducible research purposes. For the DROP_VARS parameter, you can list a set of variables to drop It is not limited to only one variable which makes it more functional. You need to drop any variables that would be classified as HIPAA's 18 identifiers that can't otherwise be modified like social security number, patient name, and medical record number.

In the PROC SQL procedure below, you will create a list of unique subject IDs and randomly sort them. Next, in the DATA step you will assign a unique subject ID to each participant based on the random sorting:

```
%MACRO did(rand_seed=, input_data=, subject_id=, drop_vars=,
            zip5_var=, date_var=, age_var=, dob_var=);

proc sql noprint;
   create table &input_data._keys as
         select &subject_id
         from (
               select distinct &subject_id
               from &input_data
         ) as a
         order by ranuni(&rand_seed);
   quit;

data &input_data._keys;
   set &input_data._keys;
   key_id = _n_;
run;
```

Then, in the PROC SQL procedure, you create a new master dataset that contains the de-identified key. In the DATA step, you are dropping the original subject ID and all variables specified in the DROP_VARS macro parameter:

```
proc sql noprint;
   create table &input_data._master as
         select a.key_id, b.*
         from &input_data._keys as a, &input_data. as b
         where a.&subject_id eq b.&subject_id;
quit;

data &input_data._did;
   set &input_data._master;
   drop &subject_id &drop_vars;
run;
```

In the IF statements below, you are calling the DID_ZIP5, DID_AGE, DID_DOB, and DID_DATES macros described in detail above:

```
%if %sysevalf(%superq(zip5_var)=,boolean) eq 0 %then
   %did_zip5(data = &input_data._did, zip5_var = &zip5_var.);

%if %sysevalf(%superq(age_var)=,boolean) eq 0 %then
   %did_age(data = &input_data._did, age_var = &age_var.);

%if %sysevalf(%superq(dob var)=,boolean) eq 0 %then
   %did_dob(data = &input_data._did, dob_var = &dob_var.,
   last_dt=&last_dt.);

%if %sysevalf(%superq(date_var)=,boolean) eq 0 %then
   %did_dates(data = &input_data._did, dates = &date_var.);
```

```
%MEND did;
```

If the required input parameter for the specified macro is not blank, then you are telling SAS to perform the de-identification procedures of the individual macros. If you do not have one of those variables in your dataset, you can pass the macro variable as a blank space and the individual macro will be skipped over.

UTILIZING THE MACROS

With the SAS macro program saved on your computer, you can now put this code to use. To start, you will need to assign a directory, which we called DIDMACRO to where the program with the macros is saved:

```
filename didmacro "<MACRO FILE PATH>";
```

We recommend you save the macros in a separate program from the program in which you use these macros on a dataset. This will help keep your projects separate from the behind the scenes work and keep your code clean and succinct.

Next, call the macros into your current program by using the %INCLUDE macro function:

```
%include didmacro ('Deidentify Final MACRO.sas');
```

With the file directory set and the macros called, you may now read in your dataset and de-identify according to the HIPAA standards. Below we walk you through an example utilizing a dummy dataset created from mockaroo.com (Mockaroo, LLC, 2018).

First, we will read in our dataset using the following code:

```
data de_id_test;
   infile "<DATASET FILE PATH>\SAS SUGI Fake Dataset.csv" delimiter = ','
   missover dsd lrecl=32767 firstobs=2;
   informat fname lname email gender $8. st_address city state $30. zip_cd
   best5. dob mmddyy10. phone race ssn $25. cpi mrn fin best20. admit_dt
   date_death mmddyy10. age best3.;
   format fname lname email gender $8. st_address city $100. state $30.
   zip_cd z5. dob mmddyy10. phone race ssn $25. cpi mrn fin best20.
   admit_dt date_death mmddyy10. age best3.;
   input fname lname email gender st_address city state zip_cd dob phone
   race ssn cpi mrn fin admit_dt date_death age;
run;
```

Once the dataset is read in you can run the macro statement for the overall de-identification macro listed below:

```
%did(rand_seed=8189, input data=de_id_test, SUBJECT_ID=fin,
drop_vars=fname lname email st_address city phone ssn cpi mrn,
zip5_var=ZIP_CD, DATE_VAR=admit_dt date_death, age_var=AGE,
dob_var=DOB);
```

Based on the dataset ‘de_id_test’, the subject id is identified as fin. The variables that need to be dropped include patient first name (fname), last name (lname), email address (email), street address (st_address), city (city), phone number (phone), social security number (ssn), and medical record numbers (cpi and mrn). The dataset also includes a 5-digit ZIP code (zip_cd), two dates that are associated with the patient (admit_dt and date_death), an age variable (age), and the date of birth of the patients (dob).

With all the appropriate macro parameters passed, the macro statement may be run. Table 8 shows a comparison of the variables that were in the dummy dataset we created above before versus after the de-identification macro.

Original Dataset Variables	De-identified Dataset Variables
FNAME	<Removed>

LNAME	<Removed>
EMAIL	<Removed>
GENDER	GENDER
ST_ADDRESS	<Removed>
CITY	<Removed>
STATE	STATE
ZIP_CD	ZIP3
DOB	DOB_YEAR
PHONE	<Removed>
RACE	RACE
SSN	<Removed>
CPI	<Removed>
MRN	<Removed>
FIN	KEY_ID
ADMIT_DT	ADMIT_DT_YRS
DATE_DEATH	DATE_DEATH_YRS
AGE	AGE_NUM, AGE_CHAR

Table 8. Variables Before and After De-Identification

Below in Table 9, Table 10, Table 11, and Table 12 are examples of how the data is transformed from the four individual macros.

Original 5-digit ZIP	Original 3-digit ZIP	HIPAA 3-digit ZIP
10292	102	000
20205	202	000
20370	203	000
20420	204	000
20520	205	000
75310	753	000
77234	772	000

Table 9. ZIP Macro Variable Changes

The first column shows the original 5-digit ZIP codes in the dataset. Using the ZIP macro, it is transformed into a 3-digit ZCTA. Based upon Table 2, we see that all of the ZCTAs listed are in the exclusion 3-digit ZIP codes. Therefore, they are changed to 000 to meet the HIPAA privacy rule guidelines.

Original Admit Date	New Admit Year
07/31/2016	2016
08/01/2016	2016
08/02/2016	2016

Table 10. Date Macro Variable Changes

The first column shows the original variable entries for the admission date of the patients. Since this is considered an identifiable date, it will be changed to the corresponding year. The second column shows how the macro transforms these dates into a new variable that contains only the year of admission.

Original Age	New Numeric Age	New Character Age
88	88	88
89	89	89
90	90	90+
91	90	90+
92	90	90+

Table 11. Age Macro Variable Changes

The first column shows the original variable entries for the age of the patients. Some patients are over the age threshold of 90, so they will be changed by the macro. In this process, two new variables are created

– numeric age and character age. If the patient is 90 or above, the numeric age is changed to 90 and the character age is changed to 90+ to identify that these patients may actually be older.

Original Date of Birth	New Date of Birth
1924	1926
1924	1927
1925	1926
1925	1927
1926	1926
1926	1927
1927	1927
1928	1928
1929	1929

Table 12. Date of Birth Macro Variable Changes

The first column shows the original variable entries for the birth year of patients. Based on the variable or date that the data was last pulled on, the new date of birth will be changed if it is more than 90 years prior.

CONCLUSION

By using and understanding the purposes of these SAS macros, the confusion that comes with de-identifying datasets that contain PHI may be minimized, if not eliminated. You may easily transform ages of patients, identifiable dates to years, and 5-digit ZIP codes to 3-digit ZCTAs. To be able to do this on a variety of different datasets with one SAS program drastically reduces the amount of code you will be writing, which in turn increases the efficiency of the de-identification process. It should be noted that this code is a useful tool for you to use in your de-identification process, but you should always confirm that the final dataset is stripped of all PHI and conforms to HIPAA privacy rules and all other applicable laws.

REFERENCES

Electronic Code of Federal Regulations. 2018. “Other requirements relating to uses and disclosures of protected health information.” Accessed February 15, 2018. https://www.ecfr.gov/cgi-bin/text-idx?SID=0ed1e92de1da217b01f07097dd9e62c6&mc=true&node=se45.1.164_1514&rgn=div8%3E%20Wednesday,%20September%2013,%202017.

Merriam-Webster. (2017). “Honest Broker.” Accessed September 22, 2017. https://www.merriam-webster.com/dictionary/honest%20broker.

U.S. Department of Health & Human Services. 2018. “Guidance Regarding Methods for De-identification of Protected Health Information in Accordance with the Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule.” Accessed January 29, 2018. https://www.hhs.gov/hipaa/for-professionals/privacy/special-topics/de-identification/index.html#zip.

United States Census Bureau. 2017. “American Fact Finder.” Accessed October 2, 2017. https://factfinder.census.gov/faces/nav/jsf/pages/download_center.xhtml.

SAS. “%BQUOTE and %NRBQUOTE Functions.” Accessed January 12, 2018. http://support.sas.com/documentation/cdl/en/mcrolref/62978/HTML/default/viewer.htm#p06cx7fegzmzpen1m9991yljxiav.htm.

Mockaroo, LLC. 2018. “Mockaroo.” Accessed January 10, 2018. https://mockaroo.com/.

CONTACT INFORMATION

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

Abigail Chatfield
Grand Valley State University, Spectrum Health Office of Administrative Research
chatfiea@mail.gvsu.edu

Jessica Parker
Spectrum Health Office of Administrative Research
jessica.parker2@spectrumhealth.org

Paul Egeler
Spectrum Health Office of Administrative Research
Paul.egeler@spectrumhealth.org

Appendix

```
%let ddir = %bquote(<CENSUS DATA FILE PATH>);

%macro zip3_exc (census_data= );
   *read in all zip code tabulation areas from file;
   data all zips;
      length id $ 14 zip5 5 zip3 3 geography $ 11;
      infile "&ddir\&census_data..csv" firstobs = 3 dsd dlm = ",";
      input id zip5 geography ttl_pop;
        if zip5 <= 00999 and zip5 >= 00000 then zip3=substrn(zip5,-1,3);
            else if zip5 <= 09999 and zip5 >=01000 then
                  zip3=substrn(zip5,0,3);
            else if zip5 >= 10000 then zip3=substrn(zip5,1,3);
      format zip5 z5. zip3 z3.;
   run;
   *find 3-digit zctas with population less than or equal to 20k;
   libname cens "&ddir";
   proc sql;
      create table cens.zip3_exc as
      select zip3
      from all_zips
      group by zip3
      having sum(ttl_pop) le 20000;
   quit;
%mend zip3_exc;

%macro did_zip5(data=, zip5_var=);
   *declare local variables;
   %local dsid varnum vartype rc;

   *check to see if zip5 variable is numeric;
   %let dsid =    %sysfunc(open(&data.,i));
   %let varnum =  %sysfunc(varnum(&dsid,&zip5_var.));
   %let vartype = %sysfunc(vartype(&dsid,&varnum));
   %let rc =      %sysfunc(close(&dsid));

   %put &dsid;
   %put &varnum;
   %put &vartype;
   %put &rc;

   proc sql noprint;
      select zip3 into: zip_exc separated by ","
      from cens.zip3_exc;
   quit;
   *create 3-digit zip and drop 5-digit zip;
   *transform zip3s with le 20000 to 000;
   %if &vartype eq n %then %do;
   data &data;
      set &data;
        if &zip5_var <= 00999 and &zip5_var >= 00000 then
                  zip3=substrn(&zip5_var,-1,3)*1;
            else if &zip5 var <= 09999 and &zip5_var >=01000 then
                  zip3=substrn(&zip5_var,0,3)*1;
            else if &zip5_var >= 10000 then
                  zip3=substrn(&zip5_var,1,3)*1;
```

```
      if zip3 in (&zip_exc) then zip3=000;
          else zip3=zip3;
    format zip3 z3.;
    drop &zip5_var;
  run;
  %end;

  %else %if &vartype eq c %then %do;
  data &data;
    set &data;
      zip3=input(&zip5_var,3.);
      if zip3 in (&zip_exc) then zip3=000;
      else zip3=zip3;
    format zip3 z3.;
    drop &zip5_var;
  run;
  %end;
%mend did_zip5;

*converting age greater than or equal to 90 to a new numeric and character
var & keeping the age for those less than 90;
%macro did_age(data=,age_var=);
  data &data;
    set &data;
      if &age_var >= 90 then do;
          age_num=90;
          age_char='90+';
      end;
      else if age ne . then do;
          age num=&age var;
          age_char=put(&age_var,3.);
      end;
    drop &age_var;
  run;
%mend did_age;

*also converting dob to a year field and setting the year for those over 90
to the 90th year;
%macro did_dob(data=,dob_var=,last_dt=);
  data &data;
    set &data;
      dob year=year(&dob var);
      if dob_year < (year(&last_dt)-90) then do;
      dob_year=year(&last_dt)-90;
    end;
    drop &dob_var;
  run;
%mend did_dob;

*converting the date values to years;
%macro did_dates(data=,dates=);
  %let dt_n=%sysfunc(countw(&dates));
  %put &dt n;
  %do r=1 %to &dt_n;
    data &data;
      set &data;
          %scan(&dates,&r)_yrs=year(%scan(&dates,&r));
```

```
      drop %scan(&dates,&r);
   run;
   %end;
%mend did_dates;

%macro did(rand_seed, input_data=, subject_id=, drop_vars=, zip5_var=,
               date_var=, age_var=, dob_var=, last_dt=);
*first create a list of unique subject ids and randomly sort them;
   proc sql noprint;
      create table &input_data._keys as
      select &subject_id
      from (
            select distinct &subject_id
            from &input_data
            ) as a
      order by ranuni(&rand_seed);
   quit;

*assign an id number based on random sorting in previous step;
   data &input_data._keys;
      set &input_data._keys;
      key_id = _n_;
   run;

*create a new master dataset that contains de-identified key;
   proc sql noprint;
      create table &input_data._master as
      select a.key_id, b.*
      from &input_data._keys as a, &input_data. as b
      where a.&subject_id eq b.&subject_id;
   quit;

*drop original subject id and all 'drop variables';
      data &input_data._did;
            set &input_data._master;
            drop &subject_id &drop_vars;
      run;

*if the zip5 var field is not blank, remove zip5 and replace with zip3;
   %if %sysevalf(%superq(zip5_var)=,boolean) eq 0 %then
      %did_zip5(data = &input_data._did, zip5_var = &zip5_var.);

*if the age_var field is not blank, remove zip5 and replace with 90+ or 90 if
needed;
   %if %sysevalf(%superq(age var)=,boolean) eq 0 %then
      %did_age(data = &input_data._did, age_var = &age_var.);

*if the dob_var field is not blank, remove dob and replace with dob year if
90+ then set to 90th year;
   %if %sysevalf(%superq(dob var)=,boolean) eq 0 %then
      %did_dob(data = &input_data._did, dob_var =
      &dob_var.,last_dt=&last_dt.);

*if the date var field is not blank, remove dates and replace with years;
   %if %sysevalf(%superq(date var)=,boolean) eq 0 %then
      %did_dates(data = &input_data._did, dates = &date_var.);
```

```
   data _null_;
   file print;

      hbar1 = repeat("=",80);
      hbar2 = repeat("-",80);
      put hbar1;
      put ;
      put "de-identification complete";
      put ;
      put hbar2;
      put "the following variables have been dropped:";
      put "  &subject_id &drop_vars";
      put hbar2;
      %if %sysevalf(%superq(date_var)=,boolean) eq 0 %then %do;
        put hbar2;
        put "the following date variables have been switched to years:";
        put "  &date_var";
      %end;
      %if %sysevalf(%superq(age_var)=,boolean) eq 0 %then %do;
        put hbar2;
        put "two age variables were created one numeric and one character:";
        put "if age gte 90 then numeric age=90 & character age=90+";
        put "otherwise numeric age and character age both equal age";
      %end;
      %if %sysevalf(%superq(dob_var)=,boolean) eq 0 %then %do;
        put hbar2;
        put "if age < 90 then birthdate was converted to exact birth year";
        put "else birth year was adjusted to the 90th year";
      %end;
      %if %sysevalf(%superq(zip5_var)=,boolean) eq 0 %then %do;
        put hbar2;
        put "the 5-digit zip code variable '&zip5_var' has been removed";
        put "and the variable 'zip3' has been inserted.";
        put "the 'zip3' variable has replaced 3-digit zip code tabulation
            areas";
        put "with populations less than or equal to 20,000 with '000'";
      %end;
        put hbar2;
        put "the following datasets have been created:";
        put "  - master dataset with new key:      &input_data._master";
        put "  - de-identified dataset:            &input_data._did";
        put "  - key to link subject_id to key_id: &input_data._keys";
        put hbar2;
        put "run macro %nrstr(%rematch()) to reassociate any dataset to
            master";
        put hbar1;

   run;
%mend did;

/*macro for re-matching de-identified data*/
%macro rematch(did_data=, key_data =);
   proc sql noprint;
      create table &did_data._rematch as
      select  a.*, b.*
      from &key_data as a, &did_data as b
      where a.key_id eq b.key_id;
```

```
   quit;

   data _null_;
      file print;
        hbar1 = repeat("=",80);
        hbar2 = repeat("-",80);
        put hbar1;
        put ;
        put "re-matching complete";
        put ;
        put hbar2;
        put "the dataset &did_data._rematch has been created";
        put "this dataset contains all did_data variables in addition to";
        put "the subject_id variable that was used during de-identification";
        put hbar1;
   run;
%mend rematch;
```