Using Proc SQL to and SAS® Macro to Blind Formatted Dates on a Group of Datasets within a Single Directory
Diana Ventura, Harvard University, Center for Biostatistics in AIDS Research, Boston, MA

ABSTRACT
Sometimes as part of making datasets anonymous, it is necessary to “blind” dates and convert them into the number of days form a specified date. This is common practice when data need to be made anonymous to preserve confidentiality. When this needs to be done to a large number of datasets it makes sense to create a macro to isolate the dates and compile the metadata into macro variables in order to apply the necessary changes to the group of datasets. The Proc SQL DICTIONARY COLUMNS dataset contains a variable named memname. The memname variable can be read as a macro variable. The macro variable allows SAS to dynamically generate the dataset names so that additional SAS macro code can be to used anonymize date variables within each individual table of a group of datasets. This type of utility can reduce the time needed to “blind” dates for confidentiality. This helpful utility can also be modified to perform a wide range of other tasks such as modifying the labels of each blinded date to indicate that it has been converted.

INTRODUCTION
This paper presents a macro that is a helpful utility for blinding dates on a group of datasets. Why create a macro to blind dates in a group of datasets? To blind a date, one can use a method of calculating the number of days from a specified date variable. For example, blindate = (datevar – birth date). This allows people without access to exact birth date information to use the data for analysis involving timelines. To wade through a large number of datasets to perform this type of calculation is cumbersome and time consuming, yet it is not unusual to have many datasets on which we need to perform the same tasks. Creating macro variables for the names of datasets and variables streamlines these tasks by eliminating multiple lines of repetitive code. Using memname from the DICTIONARY COLUMNS dataset as a macro variable allows code to be generated dynamically and eliminates the time and effort necessary to enter all the dataset names by hand. Creating additional macro variables to calculate the number of days between a specified point or blind dates creates reusable code, which means that a different group of datasets can be blinded quickly and easily simply by changing LIBNAME and a few other parameters for the group of datasets. Capturing the names of the datasets into a macro variable and creating a macro to each formatted date variable in a group of datasets may seem complicated but it can be accomplished in several easy steps: creating a dataset with subject identifiers and date used to calculate the blinded date (e.g. birth date mentioned above), reading the dataset names into macro variables, reading the formatted date variables into SAS, and using the macro variables to blind the dates. We will individually consider each step.

FIRST A NOTE ABOUT MACRO PAREMETERS
The following macro variables are user–specified parameters.
%macro blind(lib, blindate, blindlabel, blindsn);
Lib= the libname, mydir.
blindate= the specified date to perform the calculation to blind the data into the number of days, birthdt.
blindlabel= the label specified to indicate the blinded date, birth date.
blindns= the dataset with the subject identifier and the blindate to be used in the macro, blindbirth.

When the macro is invoked the parameters are specified, for example:
%blind (mydir, birthdt, birth date, dateblind);
CREATING A DATASET WITH SUBJECT IDENTIFIERS AND THE DATE TO CALCULATE THE BLINDED DATE

In order to blind the dates on a group of datasets, make the following calculation on each dataset in the directory:

datevarb = (datevar – birth date).

The datevar could be a visit date, for example. In this calculation we want the number of days to be contained in the datevarb variable. To perform this calculation the birth date variable must exist for each subject in the dataset for which the calculation is being performed. If the variable birth date does not exist in a dataset, to calculate the blinded date it must temporarily be inserted to perform the calculation and then subsequently dropped. The dataset should contain only the subject ID and date for blinding, otherwise all the other variables will be added to each dataset in the directory.

<table>
<thead>
<tr>
<th>Subjectid</th>
<th>Birthdt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>01/01/74</td>
</tr>
<tr>
<td>0002</td>
<td>08/05/68</td>
</tr>
<tr>
<td>0003</td>
<td>03/12/83</td>
</tr>
<tr>
<td>0004</td>
<td>12/08/52</td>
</tr>
</tbody>
</table>

GETTING DATASET NAMES & DATASET COUNT VARIABLES

Using Proc SQL to get dataset names involves using SQL procedures to acquire two macro variables. The first macro variable holds the names of each dataset in the directory. The second macro is used to obtain the total number of datasets.

Step 1: Obtain dataset names and the count variable for the total number of datasets in the directory into macro variables.

```
proc sql noprint;
  select distinct memname, count(distinct memname) into :tbl separated by ' ', :ntbl
  from dictionary.columns
  where libname=%upcase("&lib") and format in (DATE9, MMDDYY);
quit;
```

<table>
<thead>
<tr>
<th>&amp;lib Macro</th>
<th>Count Macro</th>
<th>Datasets in the directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>MYDIR</td>
<td>2</td>
<td>clindata</td>
</tr>
<tr>
<td></td>
<td></td>
<td>demodata</td>
</tr>
</tbody>
</table>

Note: A WHERE statement must be used to specify the directory location of the specific datasets to be included in the analysis. The DICTIONARY COLUMNS dataset captures runtime information on all tables, indexes, and macros used during a SAS session. To avoid confusion, a WHERE statement is used to capture information concerning a group of datasets in a single directory. Because dates are being formatted, the where statement must also include code to capture only the datasets with dates that are to be blinded. DATE and MMDDYY are examples. More date conventions may be specified as datasets require.
GETTING DATE USED TO BLIND THE DATASETS INTO THE DATASETS

Step 3: Getting the date used to blind the dates into every dataset in the directory

The dataset names and the dataset count variable are now put to use. The %do i= 1 %to &ntbl; starts a DO LOOP in motion. The %let table = %scan (&tbl., &i.) code puts each respective dataset name into the macro variable &table, so the code automatically cycles through every dataset in the directory completing all of the work for you.

%do i= 1 %to &ntbl;
%let table = %scan (&tbl., &i.);
proc sort data=mydir.&table; by patid; run;
data &table;
merge mydir.&table (in=a)
&blindsn (in=b);
by patid;
if a;
run;

In this sort and subsequent merge statement, the dataset blindsn is merged with every table in the directory. The inclusion of data to blind the dates is temporary; it will be used to blind the dates, and subsequently dropped from each of the datasets.

GETTING VARIABLE NAMES, VARIABLE LABELS, & VARIABLE NAME COUNT VARIABLES

Step 4: Getting the variable names, variable labels and count variables necessary to blind the dates in every dataset in the directory

The %do i= 1 %to &ntbl DO LOOP is still running, and the next step is to obtain the variable names and the variable count for all of the formatted data variables. Upon obtaining these macro variables, we will then employ them in a second DO LOOP that will blind the dates.

proc contents noprint data=&table out=c&table;
proc sql noprint;
select distinct name, label,
count(distinct name)
into :vname separated by ' ',
:vlabel separated by ' ',
:nvname
from c&table
where format in ('DATE' 'MMDDYY');
quit;

Proc CONTENTS easily captures the information needed to process the dates. The out= statement is used to capture the metadata (data about the data) into a dataset. The “c” is added to the front of the &table macro to prevent a collision between working datasets with the same names. The “c” for contents captures metadata about the datasets, not data itself. Proc SQL is used with the outputted contents datasets to obtain the variable names, labels, and the total number of variable names with formatted dates for the current dataset initiated by %do i= 1 %to &ntbl; statement.

NOTE: The difference between the where libname=%upcase("&lib") and (format="DATE9." or
format="MMDDYY."); used in the initial Proc SQL statement and where format in ('DATE' 'MMDDYY'); has to do with the differences in how the outputted c&table files indicates a date variable and how the DICTIONARY COLUMNS dataset holds date variables.

**USING THE MACRO VARIABLES TO BLIND THE DATES**

Step 5: Macro variables are now used to make changes in each of the datasets in the directory.

```sas
data blind.&table; set orig.&table;
%do X=1 %to &nvname;
%let name = %scan (&vname., &x.);
&name.b = (&name - &blindate);
label &name.b = "&name in days from &blindlabel";
drop &name &blindate &blindate.b;
%end;
run;
```

Essentially this step involves applying the metadata obtained from the Proc CONTENTS and then placed into the variable name macro (or vname above) to the actual datasets in the directory. An iterative DO LOOP is used to move the macro to perform analysis on each of the variables in the current dataset initiated by the %do i= 1 %to &ntbl code. The statement %do x=1 %to &nvname; sets up the second loop to move from 1 to the total count of variable names determined in the previous Proc SQL step.

---

### How the Code Resolves

<table>
<thead>
<tr>
<th>Code</th>
<th>Do loop</th>
<th>macro variable</th>
<th>Resolves to</th>
<th>Converts the variables and labels to days between the blind date and formatted date variables in the dataset.</th>
</tr>
</thead>
<tbody>
<tr>
<td>%do i= 1 %to &amp;ntbl</td>
<td>Do 1 to 2</td>
<td>table1</td>
<td>Clindata</td>
<td>Clindata</td>
</tr>
<tr>
<td>%do x=1 %to &amp;nvname;</td>
<td>Do 1 to 3</td>
<td>3 date variables below</td>
<td>Visitdt, Testdt, Drawdt</td>
<td>Visitdtb=(Visitdt-birthdt); Testdtb=(Testdt-birthdt); Drawdtb=(Drawdt-birthdt);</td>
</tr>
<tr>
<td>&amp;name.b = (&amp;name - &amp;blindate);</td>
<td>1</td>
<td>name1</td>
<td>Visidt</td>
<td>Visidtb=(Visitdt-birthdt);</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>name2</td>
<td>Testdt</td>
<td>Testdtb=(Testdt-birthdt);</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>name3</td>
<td>Drawdt</td>
<td>Drawdtb=(Drawdt-birthdt);</td>
</tr>
<tr>
<td>label &amp;name.b = &quot;&amp;name in days from &amp;blindlabel&quot;;</td>
<td>1</td>
<td>name1</td>
<td>Visidt</td>
<td>Visidtb=&quot;Visitdt in days from birth date&quot;;</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>name2</td>
<td>Testdt</td>
<td>Testdtb=&quot;Testdt in days from birth date&quot;;</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>name3</td>
<td>Drawdt</td>
<td>Drawdtb=&quot;Drawdt in days from birth date&quot;;</td>
</tr>
</tbody>
</table>

---

So the macro simulates the following code:

```sas
data mydir.clindata;
Visitdtb=(Visitdt-birthdt);
Testdtb=(Testdt-birthdt);
Drawdtb=(Drawdt-birthdt);
label Visitdtb="Visitdt in days from birth date";
Testdtb="Testdt in days from birth date";
Drawdtb="Drawdt in days from birth date";
Run;
```
The above hard code might not seem like much work. But when one has hundreds of datasets to process in a given week or month, the use of these kinds of macros becomes imperative and saves substantial time. The macro variable created from memname and variable information within datasets can be used to perform more code manipulation; the blinding of dates is just one example of how metadata information such as memname can be used to streamline repetitive tasks.

CONCLUSION
Using Proc SQL’s DICTIONARY COLUMNS to create macro variables can shorten the time it takes to generate code. A few lines of reusable code can save hours of typing. Anywhere a dataset name is used the macro variables can be used in its place. In addition, using macro variables to blind dates or perform other data manipulation tasks can save many hours of programming time. This allows many datasets to be manipulated and analyzed at one time. The macro code presented here eliminates the need for pages of repetitive code and allows groups of datasets and variables to be manipulated with ease.

CONTACT INFORMATION
I welcome your comments and questions. Contact the author at:
Diana Ventura, M.Ed., M.Div.
Chief Data Administrator
Center for Biostatistics and AIDS Research
Harvard University School of Public Health
dventura@sdac.harvard.edu

TRADEMARK INFORMATION
SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.