Paper RX-06-2013

Creation and Implementation of an Analytic Dataset for a Multi-Site Surveillance Study using Electronic Health Records (EHR) and Medical Claims Data

Renuka Adibhatla, HealthPartners Institute for Education & Research, Minneapolis, MN
Gabriela VazquezBenitez, HealthPartners Institute for Education & Research, Minneapolis, MN
Mary Becker, HealthPartners Institute for Education & Research, Minneapolis, MN
Amy Butani, HealthPartners Institute for Education & Research, Minneapolis, MN

ABSTRACT

SAS® code is utilized for research projects of the Health Maintenance Organization Research Network (HMORN). The HMORN is a consortium of 18 health care delivery organizations with integrated research divisions with over 15 million patients. Multi-site research is conducted by utilizing the HMORN Virtual Data Warehouse (HMORN VDW), a distributed data network where each site locally stores their data in standardized data structures, in this case as SAS datasets. HMORN VDW is an excellent source of data for surveillance and observational studies of chronic conditions such as diabetes and cardiovascular disease. SAS programs are shared among HMORN project sites in order to extract site-specific data from each site which is then combined to represent the overall HMORN patient population and results.

This paper focuses on the various programming techniques, including some borrowed from previous SAS conference papers, and lessons learned from an ongoing HMORN multi-site project involving 11 HMORN research centers. These methods were applied to produce a clean analytic dataset with extensive use of PROC SQL, arrays, and macros. The paper clearly demonstrates the advantages of utilizing distributed SAS code for extracting data, the benefits from an HMORN VDW for multi-site research, and the required organization of the SAS datasets and programs. The paper also discusses the challenges presented with data volumes and storage.

SAS 9.2 was used on both Windows and UNIX environments. The paper is intended for beginner to intermediate level SAS programmers, preferably with minimal knowledge in EHR and claims data.

INTRODUCTION

The HMORN is the nation's preeminent source of population-based research that measurably improves health and health care. The HMORN's Virtual Data Warehouse (HMORN VDW) facilitates multi-site research while protecting patient privacy and proprietary health practice information. The HMORN VDW is a federated database where each site locally stores their data in identical data structures, in this case as SAS datasets. The VDW build involves each site extracting from vastly different source systems, manipulating and storing data to attempt cohesion of variables and value possibilities across sites.

HMORN VDW CONCEPT

The VDW facilitates collaboration with other research organizations by allowing the pooling and manipulation of common data elements. The VDW is "virtual" in the sense that the research data remain at each HMORN site; the VDW is a local database designed in collaboration with other HMORN sites and built to a common set of standardized file definitions. Content areas and data elements represent commonly required elements for research studies. The primary content areas include insurance enrollment, demographics, pharmaceutical dispensing, utilization, and cancer tumor registry information. Other content areas are census, vital signs, laboratory values, and death records. For each content area, a data dictionary specifies the common format for each element such as variable name, variable label, extended definition, code values, and value labels. Local site programmers have mapped and transformed data elements from their local data systems into the standardized set of variable definitions, names, and codes. The VDW facilitates multi-site research within the HMORN because much of the preparatory work for pooling existing data across multiple sites has been done.

Multi-Site Research in Prevention and Treatment of a Chronic Disease

The overall goal of the multi-site project is to develop a comprehensive, longitudinal clinical registry of a population of 1.3 million insured patients with the chronic disease (cases), and a similar database of all insured members without the chronic disease (controls) from 11 integrated healthcare delivery systems. The registry will cover the period from 2005-2012, and capture demographic and clinical information from EHRs and other patient-reported system databases. The registry will be used to conduct surveillance, identify and monitor trends in the chronic illness incidence and prevalence, in the chronic illness treatment patterns, and in the incidence and prevalence of chronic illness complications.

METHODS

SAS programs were developed by a lead site and distributed to participating sites in order to build datasets from each site's VDW. Data included electronic health record (EHR) data such as heights, weights, blood pressure readings; lab results; insurance claims data included diagnoses and procedures; prescription medication data; demographics; and mortality data. A random study identification number was assigned to each study subject, and each site uploaded data to a secured website. The lead site accessed the data, combining all sites' data for the purposes of performing quality assurance and analyzing the diverse population that is representative of national health trends.
Although the structure and content of data received from each site is uniform, dataset size is dependent upon the size of a site's patient/member population. This particular project demanded server space equal to 350 GB. Each site had up to 42 SAS datasets, 21 each for cases and controls.

CREATION OF AN ANALYTIC DATASET

The objective of the SUPREME-DM study was to compare trends in occurrence of macrovascular complications in adults with and without the chronic illness from 2005-2011, by demographic subgroups and clinical characteristics. In order to arrive at the final analytic dataset to perform surveillance analysis, three SAS programs were created. Each program produced a permanent dataset that would be used in the subsequent SAS program. For large multi-site studies, fragmenting a large SAS program into smaller independent SAS programs helps to better manage the SAS programs, in addition to providing more flexibility and efficiency in managing edits. In the following sections, some examples of code used in each of the programs will be illustrated.
Program1 – Creation of a site specific inclusion cohort with all baseline characteristics for both cases and controls.
Program2 – Creation of a site specific surveillance population dataset based on the inclusion cohort for cases and controls.
Program3 – Creation of a site specific surveillance events dataset based on the surveillance population for cases and controls

PROGRAM 1 – CREATING A SITE SPECIFIC COHORT

This cohort was created based on data present in standard SAS datasets such as demographics, enrollment, vitals, lab results, medications, diagnoses and mortality.
Every member/patient can have multiple insurance enrollment periods; therefore, the records had to be collapsed so that each has at a minimum of 90 days continuous insurance enrollment. Furthermore, only enrollment periods that included the index date (the date the person was diagnosed with the chronic disease) had to be kept.
Each individual can contribute several rows in each of the above datasets depending on the number of claims, diagnoses, lab results, and vitals in a given period of time. The goal was to create a cohort reduced to a single observation per individual that contained all the requested variables from the above datasets.

USING REGULAR EXPRESSIONS FOR EXTRACTING DIAGNOSIS CODES FROM THE DX TABLE

A person with a chronic illness often has one or more co-morbidities. The diagnosis table contains information about inpatient, outpatient, and ER visits with diagnosis and procedure information. The protocol required the below dx codes to be extracted from the dx table. A common way of doing this would be using the SUBSTR () function as shown below.

Figure 1. List of Dx codes to extract from the Diagnosis table.

Dx Codes to pull from diagnosis table	410-414.9, 429.2, 410.0-410.91, 411.1, 411.8, 433-434.91, 430-432.9,435-435, 436-438, 428-428.9, 404.x1, 410-414, 420-421, 423-424, 426-427, 429, 430-438, 440-444, 447, 451-453, 557, 785.0-785.3, V42.2, V45.0, V45.81, V45.82, V53.3, 362.11, 401.x-405.x, 437.2, 427.31, 410-414, 429.2, 430-438

```
proc sql;
create table case_dx as select a.*, b.*
from enrollbl2 as a left join macrovascular_dx as b
where substr(dx,1,3) in ('410','411','412','413','414',
'420','421','423','424','426','427','429','430','431','432','433','434','435'
'436','437','438','440','441','442','443','444','447','451','452','453','557','785'
,…..);
```

Using SUBSTR () for extracting the above codes may not be so tedious; however, there are times when the requested diagnosis codes could be a long list. In that case, use of SUBSTR () could get rather lengthy.
An alternative way to do the above would be by using Perl regular expressions in SAS. Although SAS already has a powerful set of string functions, regular expressions can sometimes provide a more compact solution to a complicated string manipulation task. To read more about SAS regular expressions, please refer to the paper cited in the references section.
Use of a regular expression prxmatch to extract diagnosis codes is shown below:

```
proc sql;
create table case_dx as select a.*, b.*
from enrollbl2 as a left join macrovascular_dx as b
on a.studyid = b.studyid
and prxmatch("/^4(1|4)[0-4]|^43[0-8]|^404|^447|^45[1-3]|^557|785[\.0-
3]|V42\.2|V45\.0|V45\.81|V45\.82|V53\.3|362\.11|^40[1-]|^42[01346789]|427\.31/",dx)
order by studyid ;
quit;
```

start	end	studyid	adate	dx
1/1/2007	10/31/2009	000006	6/27/2005	785.2
1/1/2007	10/31/2009	000006	11/01/2004	401.9
1/1/2007	10/31/2009	000006	06/27/2005	401.9
1/1/2007	10/31/2009	000006	06/10/2005	401.9
1/1/2007	10/31/2009	000006	12/06/2005	401.9
1/1/2007	10/31/2009	000006	06/13/2005	401.9
1/1/2007	10/31/2009	000006	06/10/2005	785.2
1/1/2007	10/31/2009	000006	08/17/2005	401.1

Figure 2. List of dx codes extracted using prxmatch.

After extracting all the diagnosis codes for every individual during a study period, the next step is to create flags to determine if an individual had a particular set of comorbidities before the index date.

```
data case_dx_2;
    set case_dx;
    by studyid;
if substr(dx,1,3) in ('410','411','412','413','414') or dx = '429.2'
then bchd = 1; else bchd = 0;
if substr(dx,1,3) = '410' then bmi_cs = 1; else bmi_cs = 0;
if substr(dx,1,3) = '433' or substr(dx,1,3) = '434' then bistk = 1; else bistk = 0;
if substr(dx,1,3) = '428' then bchf = 1; else bchf = 0;
if substr(dx,1,3) in '426','427','429','430','431','432','433','434','435',
                      'V42','V45','V53') then bcvd = 1; else bcvd = 0;
if dx = '362.11' or dx = '437.2' or substr(dx,1,3) in
('401','402','403','404','405')then bhtn = 1; else bhtn = 0;
run;
```

start	end	studyid	adate	dx	bchd	bmi_cs	bistk	bchd	bcvd	bhtn
1/1/2007	10/31/2009	000006	6/27/2005	785.2	0	0	0	0	1	0
1/1/2007	10/31/2009	000006	11/01/2004	401.9	0	0	0	0	0	1
1/1/2007	10/31/2009	000006	06/27/2005	401.9	0	0	0	0	0	1
1/1/2007	10/31/2009	000006	06/10/2005	401.9	0	0	0	0	0	1
1/1/2007	10/31/2009	000006	12/06/2005	401.9	0	0	0	0	0	1
1/1/2007	10/31/2009	000006	06/13/2005	401.9	0	0	0	0	0	1
1/1/2007	10/31/2009	000006	06/10/2005	785.2	0	0	0	0	1	0
1/1/2007	10/31/2009	000006	08/17/2005	401.1	0	0	0	0	0	1

Figure 3. Creating flags for each comorbidity.

USING PROC MEANS TO COMBINE DIAGNOSIS DATA

In this instance, the researcher is not interested in the exact diagnosis date, but, rather, whether a diagnosis was made during the study period. An individual's claims data contains many claims that must be reduced to a single observation per individual stating a '1' if a diagnosis was made or a '0' if there was no diagnosis of a particular disease.

An easy solution for this is, using ARRAY's and DO loops in the DATA step and then PROC MEANS.

After creating all flagged diagnosis variables, the next step is to collapse the data across the many claims for each individual. To do this, sort the data with the individual's unique identifier and then use PROC MEANS to create a sum of the number of times each diagnosis occurred for each individual and output the sums to a SAS data set.

```
proc sort data = case_dx_2;
by studyid;
run;

proc means data = case_dx_2 sum noprint;
var bchd bmi_cs bistk bCHF bCVD bHTN;
by studyid;
output out = case_dx_3 sum = schd smi_cs sistk schf scvd shtn;
run;
```

studyid	_TYPE_	_FREQ_	schd	smi_cs	sistk	schf	scvd	shtn
000006	0	8	0	0	0	0	2	6

Figure 4. PROC MEANS to combine diagnosis data.

The SAS data set "case_dx_3" has 6 variables that include the unique identifier for each individual as well as the number of times each diagnosis occurred for each individual. We have now gone from diagnoses that could occur in many fields across many claims to one observation per subject that contains the count of each type of diagnosis.

The final step is to create indicator variables from the diagnosis sum variables using two ARRAY statements and a single DO loop. ARRAY "s" contains the summed diagnosis variables and ARRAY "i" contains the new indicator variables to be created. It is important to make sure that the order of the variables in each array is exactly the same. This ensures that the indicator that will be created corresponds to the correct summed diagnosis variable (e.g. "schd" is first in the "s" array, and "bchd" is first in the "i" array). Note the use of the DO Over loop which is used to perform the operations in the DO loop over all the elements in the array.

The SAS dataset case_dx4 has been reduced to one row per subject that indicates the presence or absence of a condition by 1 or 0, irrespective of the number of occurrences over a period of time.

```
data case_dx4(drop=_TYPE_ _FREQ_ schd smi_cs sistk schf scvd shtn);
    set case_dx_3;
    by studyid;
array s schd smi_cs sistk sCHF sCVD sHTN;
array i bchd bmi_cs bistk bchf bcvd bhtn;
do over s;
if s>=1 then i=1;
else i=0;
end;
run;
```

studyid	bchd	bmi_cs	bistk	bchf	bcvd	bhtn
000006	0	0	0	0	1	1

Figure 5. Using DO Over loop to create indicators.

The final dataset generated in program 1 was an inclusion cohort which was reduced to a single row per individual. The common attributes in the inclusion cohort are start (insurance enrollment start date), end (insurance enrollment end date), studyid (unique person identifier), birthdate, gender, height, weight, indexdt, lab results, diagnosis (indicate by 1 or 0), onset date of diagnosis, race/ethnicity, and medications. An example of the final reduced dataset generated by program 1 is shown below.

start	end	studyid	birth_date	gender	indexdt	age_indexdt	sbp_bl	dbp_bl
4/1/2007	7/31/2007	000006	10/15/1963	f	1/5/2006	42	123.5	83

tobacco	year	hdl	hdl_dt	dmmed	bpmed	bchd	bmi_cs	bistk	bchf	bcvd	bhtn	bmi
1	2006	32	10/26/2005	1	1	0	0	0	0	1	1	45

Figure 6. Final baseline cohort.

PROGRAM 2 - CREATING A POPULATION DATASET BY YEAR.

For a surveillance study, it is necessary to compute the person time and population characteristics. The researcher likes to know the number of days a person had enrollment in a year starting from 2005-2011.

The goal of the second program was to create a population dataset by year such that the investigator would be able to evaluate a person's smoking status, body mass index (bmi), Medicare/Medicaid status and age for every year of enrollment from 2005-2011. Arrays and nested DO loops were used for this purpose.

In this example, calculation of age at each calendar year from 2005-2011 will be shown.
Denom_age_case_07 is a reduced dataset based on the inclusion cohort that was generated by program1. Only the studyid, indexdt, age_indexdt and gender are kept. Additional variables a_y05-a_y11 will be added to this dataset.

The below data step creates an array of 7 elements a_y05-a_y11. The first DO loop initializes the elements of the array to 0 and proceeds to the next step which happens to be another iterative DO loop. The second DO loop initializes the counter (j) to the start year, in this case 2005. A third iterative DO loop is used to and applies the conditions specified to populate the elements y1-y8.

```
data denom_age_case_07;
set anal.bl_case_07_cohort (keep=studyid gender age_indexdt indexdt);
array a[7] a_y05-a_y11;
YR_INDEX=YEAR(INDEXDT);
do i=1 to 7;
a[i]=.;
end;
do i=1 to 7;
j=2004+i;
do k=0 to 6;
if yr_index+k= j then a[i]=age_INDEXDT+k;
end;
end;
DROP I K J;
run;
```

The logical steps followed by the above code are outlined below:

1. Create an array of 7 elements a_y05-a_y11.
2. Create an iterative DO loop and create an index i that initializes all the elements a_y05-a_y11 to missing.
3. Create another iterative DO loop with an index j that increments from 2004 until it reaches the ending index value, in this case 2004+7 i.e 2011.

4. As the second DO loop iterates through each element, it checks for the given condition in the third iterative DO loop and populates the elements.
5. In the case shown below, SAS sets the value of j to 2005 and as it iterates from ay05_ay11, it checks for the condition between the third DO loop and the End statement.
6. When j=2005, it checks for the given condition within the 3rd DO loop. The condition does not evaluate to true, so it exits out of the DO loop and the value in a_y05 remains a missing value.
7. When j=2006, the condition still does not apply.
8. When j = 2007 and k=0 the condition is satisfied. The values in the elements a_y07-a_y11 are populated as shown below.

studyid	gender	indexdt	age_indexdt	a_y05	a_y06	a_y07	a_y08	a_y09	a_y10	a_y11	yr_index
000006	m	1/1/2007	49	.	.	49	50	51	52	53	2007

Figure 7. Creation of an array to determine age at each year.

We now need to reorganize the file from one record per subject to a hierarchical file of person-year records as shown in Figure 8.

```
data denom_age2_case_07;
set denom_age_case_07;
array a[7] a_y05-a_y11;
do i=1 to 7;
age=a[i]; year=2004+i; output;
end;
keep studyid age year;
run;
```

Create an array for age that contains the variables corresponding to the age at specific time points.
Create an iterative DO loop with an index that uses the first measurement time as the starting index value and the last measurement time as the ending index value.
Assign the array name to a new variable (age) that is not an array name, e.g. age=a[i]
Use an output statement before ending the DO loop to output the observation to the data set. End the DO loop. Using a KEEP statement keep only those variables you want to keep.

studyid	Age	year
000006	.	2005
000006	.	2006
000006	49	2007
000006	50	2008
000006	51	2009
000006	52	2010
000006	53	2011

Figure 8: Transform dataset from wide to long.

An example of a reduced dataset generated by the program 2 is shown below.

studyid	gender	Indexdt	tobacco	wt	ht	race	pyear	year	age	bmi
600001	f	1/5/2006	1	266.2	66.38	WH	0.986	2006	42	30
600001	f	1/5/2006	1	165.9	66.38	WH	0.578	2007	43	29
600006	m	1/1/2007	99	260.75	63.00	MI	1	2007	49	33
600006	m	1/1/2007	99	253.93	63.00	MI	1	2008	50	34
600006	m	1/1/2007	99	258.93	63.00	MI	0.830	2009	51	34

Figure 9: An example of the final reduced SAS dataset generated by program 2.

PROGRAM3 - CREATION OF A MACROVACSULAR EVENTS DATASET

After creation of a surveillance population table, the final step was to create a macrovascular surveillance events table to count the number of events taking place in a study period.

A separate dataset for each of the events had to be created in order to count the number of events occurring by year in a population. There about 9 events such as MI (myocardial infarction), hsk, isk, chf, mi_cs. Each of these events has to be extracted from the SAS dataset "macrovascular_dx_case_95" and joined with the baseline cohort created in program 1 based on different criteria to generate an event-specific SAS dataset.

Instead of writing the same PROC SQL code over and over 9 times, the same outcome can be accomplished by using macros and writing the PROC SQL code just once. While this alone can be helpful, macros are even more powerful when adding parameters which are macro variables whose values are set when the macro is invoked. Parameters give macros flexibility.

Create a macro called events. The macro has been assigned 3 parameters

Where - Since the where condition in the proc sql varies for each event
Datevout – To create a separate dataset for each event
Filedat - Specify if referring to a case file or a control file because, every site has two sets of SAS datasets – case and control. We need to be able to run the same macro for both cases and controls.

```
%macro events (where, datevout, filedat);
proc sql;
  create table &datevout as
  select distinct a.studyid, a.&select, b.studyid,  b.adate, b.ddate, b.dx,
b.enctype, b.principal_dx, year(b.adate) as y_event
  from bl&filedat._cohort A LEFT JOIN macrovascular_dx_&filedat. b ON a.studyid=
b.studyid
  &where
 order by a.studyid, b.adate;
quit;
%mend event;
```

In the above code, all the macro variables have been highlighted. Also notice that the macro variable &select does not seem to be defined as a parameter in the events macro. This will be talked about in the next section.

The events macro is invoked as shown below where all the macro variables have values assigned. The macro has been invoked twice because the values of the macro variables vary each time. Notice that the macro variable &filedat were assigned another macro variable instead of a value. This will be talked about in the next section.

```
%events(where=where (substr(b.dx,1,3)='410'  ) and b.adate>a.&select and b.enctype
eq 'IP' and b.principal_dx='P',
      datevout=mi_&filedat, filedat=&filedat);

%events(where=where  substr(dx,1,3) in ('430','431','432')  and b.adate>a.&select
and b.enctype eq 'IP'  ,
      datevout=hsk_&filedat, filedat=&filedat);
```

ORGANIZATION OF THE SAS DATASETS FOR A MULTI-SITE STUDY.

An important requirement of a multi-site study, in addition to being consistent with the following several standards, is to remain consistent with the file names. If the SAS datasets across sites are not consistent, the ability to use macros for greater efficiency will be diminished. An example of how the SAS datasets were named in this multi-site study is shown below:
DEMOG_CONTROL_90, DEMOG_CASE_90, LAB_CONTROL_90, LAB_CASE_90
In the above files, 'DEMOG' and 'LAB' refer to the kind of data the SAS dataset holds, for e.g. demographics information, lab information etc.

'CONTROL' and 'CASE' refers to if the SAS datasets contain information for the control or case population.
'90' represents the site number. Since there were 11 sites participating in this study, each site was designated a site number.
Each site had 42 SAS datasets--21 each for cases and controls. All the files were placed in a site-specific folder site89, site90…….site99.
Each of the above mentioned programs were tested against one site. Once the programs were thought to be error free and working as desired, each program was run against all the remaining sites in only one instance.
In order to run the same program across all sites, the following modifications were made to the SAS programs

1. In the LIBNAME statement, the path names to all the site specific folders were concatenated as shown below. By logically concatenating two or more SAS libraries, the code can reference them all with one libref. A library can be specified with its physical filename or its previously assigned libref.

```
libname supreme
('\\Project\site89','\\Project\site90','\\Project\site91','\\Project\site92','\\Pro
ject\site93','\\Project\site94','\\Project\site95','\\Project\site96','\\Project\si
te97','\\Project\site98','\\Project\site99');
```

2. A macro 'allsites' was created at the start of each program. The parameters "site" and "select" are assigned to it. Since the macro variable site has more than one value, a simple IN operator was used. The IN operator can be used in a macro program in versions SAS 9.2 and beyond, the caveat being that the system option minoperator has to be used either inside the macro call or at the start of the program.
3. Next, another macro 'survfile' was defined and a parameter "filedat" was assigned. This parameter is used to indicate if the file is a control file or a case file.

The remaining part of the program can be included within these two macro definitions in order to run the program for both case and control files for all the sites. A complete structure of the SAS program is shown below.

```
options minoperator ;
%macro allsites(site,select);
%if &site in (89 90 91 92 93 94 95 96 97 98 99) %then %do;
%end;
%macro survfile(filedat);
                  %macro events (where, datevout, filedat);
proc sql;
          create table &datevout as
select distinct a.studyid, a.&select, b.studyid,  b.adate, b.ddate, b.dx,
b.enctype, b.principal_dx, year(b.adate) as y_event
from bl&filedat._cohort A LEFT JOIN macrovascular_dx_&filedat. b ON a.studyid=
b.studyid
          &where
          order by a.studyid, b.adate;
quit;
%mend event;
%events(where=where (substr(b.dx,1,3)='410'  ) and b.adate>a.&select and b.enctype
eq 'IP' and b.principal_dx='P',
          datevout=mi_&filedat, filedat=&filedat);
%events(where=where  substr(dx,1,3) in ('430','431','432')  and
b.adate>a.&select and b.enctype eq 'IP'  ,
          datevout=hsk_&filedat, filedat=&filedat);

%mend survfile;
%survfile(filedat=case_&site);
%survfile(filedat=control_&site);

%mend allsites;
%inop(89,select=indexdt);
%inop(90,select=indexdt);
%inop(91,select=indexdt_c);
%inop(92,select=indexdt_c);
%inop(93,select=indexdt);
%inop(94,select=indexdt);
```

```
%inop(95,select=indexdt);
%inop(96,select=indexdt);
%inop(97,select=indexdt);
%inop(98,select=indexdt);
%inop(99,select=indexdt);
```

The outer most macro %allsites is invoked first and assigns the value ‘89’ to the site parameter and “indexdt” to the select parameter.

The macro %survfile is invoked next and assigns the value ‘89’ to the parameter “case_&site”. It was able to assign ‘89’ obtained from the preceeding macro that was invoked first.

The macro %events was invoked last and by this time we have a clear set of values assigned to each of the parameters. The WHERE parameter allows for a different set of conditions specified in the WHERE clause of the PROC SQL. In the SELECT statement, one of the columns selected can differ for each site. If the site is 89, the SELECT column would be a.indexdt. If the site was 91, the select column would be a.indexdt_c
The &datevout macro variable allows creation of an event-specific SAS dataset.
When the above macro is run, the following SAS files are generated:

mi_case_89, mi_control_89, hsk_case_89, hsk_control_89, mi_case_90, mi_control_90……mi_case_99, mi_control_99, hsk_case_99, hsk_control_99

USE OF FORMATS IN PROC SQL

Finally, we would like to group the population in each site based on similar characteristics such as age, bmi, gender, race etc. Having pre-defined user-formats would be really useful and can be applied in proc sql with minimum coding effort. A nice feature of using FORMATS for grouping variables is that the method can replace IF/THEN/ELSE code or the CASE statement in PROC SQL.

It is recommended to permanently save your formats to use in other code or to be used by other users. This saves having to regenerate them at each point of usage. For this analytic dataset, several formats were created and stored in a separate SAS program. Every time there is a need to apply formats, the permanent SAS program can be executed using the %INCLUDE statement at the beginning of a SAS program. An example of the formats defined in the permanent SAS program for variables such as age, bmi and tobacco are shown below.

```
PROC FORMAT;
VALUE AGE
18-19 = '18-19'
20-44 = '20-44'
45-54 = '45-54'
55-64 = '55-64'
65-74 = '65-74'
75-84 = '75-84'
85-199 = '85+';

VALUE BMI
1-19.999='<20'
20-24.999='z 20-24'
25-29.999='25-29'
30-34.999='30-34'
35-39.999='35-39'
40-94.999='40-94'
95-199.999='95+';

value smk
1='Current smoker'
other='No'
99='Not Patient';
```

We can apply the PUT function with a user-defined format by PROC FORMAT in the SELECT statement to create groups based on the formats.

```
PROC SQL;
CREATE TABLE anal.SURV_POP_COUNTS_case_95
AS SELECT year(indexdt)as yr_entry, put(age,age5f.) as age1,gender,race,
put(bmi,bmi.)as bmi1,put(tobacco,smk.) as tobacco, COUNT(STUDYID) AS n_POP,
SUM(PYEAR) AS PYEARS, mi_p_e, sum(nmi) as mi_e, sum(nisk) as isk_e, sum(nhsk) as
hsk_e
FROM surv_pop_events_case_95
GROUP BY yr_entry, YEAR, AGE1, gender, racecat2, patient, bmi1, tobacco;
```

yr_entry	age1	gender	race	bmi1	tobacco	n_pop	pyears	mi_e	isk_e	hsk_e
2005	20-24	F	WH	25-29	No	2	1	0	1	1
2005	20-24	F	MI	<20	No	4	1	0	0	2
2007	80-84	M	WH	30-34	Current smoker	3	0.66	2	2	0

Figure 10: Final analytic SAS dataset

In the final analytic dataset (figure 10), the variables age1, bmi1 and tobacco contain the formatted values. Also notice that the variable studyid is not present in the dataset anymore. The final analytic dataset is about population subgroups based on similar characteristics and not on individual characteristics. So, the variable n_pop replaces the variable studyid and gives a count of the number of individuals present in each subgroup.

DATA QUALITY CHECKS AND REPORTS

It is a good practice to produce data quality checks, especially when dealing with multi-site data. Creating three independent SAS programs helped with checking for data quality and data completeness within sites at every step. Each of the above three SAS programs had a report generated to check for any inconsistencies in the data for each site.
Some of the common data checks would be the use of PROC MEANS to get the maximum, minimum, mean and standard deviation of continuous variables such as height, weight, bmi, blood pressure readings and lab values. PROC FREQ is another SAS procedure that could help give an estimate of the expected counts in each group.
It is always recommended to have data checks earlier on in the programs as this will minimize the use of both time and resources.

CONCLUSION

One of the hallmarks of the HMORN is the existence of a Virtual Data Warehouse (VDW). The VDW is a rich collection of information from which analytic datasets can be constructed. The advantages of having standardized SAS datasets across multiple sites is that it allows for code sharing. A SAS program written at one site can be used by other sites to run locally against their database.
Take advantage of past and current SAS proceedings which are freely available to expand your SAS skills. Knowledge sharing among colleagues and peers is important, because often times you will be pleasantly surprised by how efficient and simple somebody else's approach might be. Don't be afraid to explore and experiment with new SAS methods and functions, because there are a number of ways to arrive at the same result .

REFERENCES

Jennifer L. Waller, Many to One Using a SAS® DATA Step and PROC MEANS

Ron Cody, An Introduction to Perl Regular Expressions in SAS 9

Jennifer L Waller, How to Use Arrays and DO Loops: Do I DO Over or Do I Do i?

Hornbrook MC, Hart G, Ellis JL, Bachman DJ, Ansell G, Greene SM, Wagner EH, Pardee RE, Schmidt MM, Geiger A, Butani AL, Field T, Fouayzi H, Miroshnik I, Liu L, Diseker R, Wells K, Krajenta R, Lamerato L, Neslund-Dudas C. Building a virtual cancer research organization. J Natl Cancer Inst Monogr 2005(35):12-25.

Nichols GA, Desai J, Elston Lafata J, Lawrence JM, O'Connor PJ, Pathak RD, et al.

Construction of a Multisite DataLink Using Electronic Health Records for the Identification, Surveillance, Prevention, and Management of Diabetes Mellitus: The SUPREME-DM Project. Prev Chronic Dis 2012;9:110311.

ACKNOWLEDGMENTS

The authors would like to thank Patrick O'Connor for his encouragement and support and Jay Desai for his valuable comments. We would also like to thank Teri Defor and Beth Molitor for their support and understanding.

The authors also wish to thank Michael Wilson for willing to proofread the paper.

RECOMMENDED READING

http://en.wikipedia.org/wiki/HMORN_Virtual_Data_Warehouse

http://hmoresearchnetwork.org/

CONTACT INFORMATION

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

Renuka Adibhatla
Health Partners Institute for Education & Research
8170 33rd Ave. S.
PO Box 1524
Bloomington, MN, 55440-1524

renuka.x.adibhatla@healthpartners.com