ET05

Many to One Using a SAS® DATA Step and PROC MEANS

Jennifer L. Waller, Medical College of Georgia, Augusta, GA

ABSTRACT

Medical claims data from Medicaid, Medicare, or private insurance companies contain information regarding medical procedures, diagnoses, prescriptions, and more. When using these data, often a single individual will have multiple claims within a given time period of data extraction. To add to this complexity, each claim can have up to 7 diagnosis fields and 3 procedure fields. Obtaining a single observation regarding whether an individual had a diagnosis or a procedure requires examining many variables within a single claim observation and then examining the many claims that pertain to the individual. One might think that determining an individual's diagnoses and procedures a daunting task. However, there is an easy solution. Using ARRAY's and DO loops in the DATA step and then PROC MEANS, data across many variables and many observations for each individual can be combined into a single observation to determine diagnosis and comorbidity status.

INTRODUCTION

In the recent past, many clinical researchers have turned to administrative data bases to study medical conditions and the health care, costs and utilization for these conditions. Some administrative data bases come in the form of medical claims data from sources such as Medicaid, Medicare, or private insurance companies. These data contain information regarding medical procedures, diagnoses, prescriptions, costs, and type of service such as inpatient, outpatient ER, office, dental, or lab visits.

As the researcher defines the problem to be examined, it is imperative that the programmer and statistician understand the nature of the data contained in these claims data bases. Often a single individual will have multiple claims within a given time period of data extraction that correspond to facility, provider and prescription claims. Facility claims contain information about inpatient, outpatient, lab, and ER visits with diagnosis and procedure information. Professional claims contain information about office visits with diagnosis and procedure information. Prescription claims contain information about the drug filled. Thus, when extracting information from claims data the programmer must look in several places in several types of claims. To add to this complexity, each facility or provider claim can have up to 7 diagnosis fields and 3 procedure fields.

Often, the researcher is not interested in when a diagnosis was made or when a procedure was performed. Rather, the researcher is interested in whether a diagnosis was made during the study period or whether a procedure was performed. They may also be interested in how many visits of a particular type an individual has or how many times the individual refilled a particular prescription. Thus the claims data that has the information in many fields within the claim and across many claims for an individual must be reduced to a single observation per individual.

Obtaining this single observation regarding whether an individual has a diagnosis or a procedure, and determining the number of times a diagnosis was made or a procedure performed, requires examining many variables within a single claim observation and then examining the many claims that pertain to the individual. One might think that determining an individual's diagnoses and procedures a daunting task. However, there is any easy solution using ARRAY's and DO loops in the DATA step and then PROC MEANS. Creating a single observation for an individual from many variables and many claims will be performed by creation of a comorbidity index using the diagnosis fields.

THE CHARLSON COMORBIDITY INDEX

The Charlson Comorbidity Index (CCI) (Charlson et al., 1987) is an index used to determine not only the number of co-existing diagnoses a person has, but also to weight the number of co-existing diagnoses by

the severity of the diagnosis. Deyo et al. (1992) adapted the CCI for use with the International Classification of Diseases (ICD-9-CM) in administrative data bases. Sixteen different diagnoses with multiple ICD-9-CM codes for each diagnosis are used in the CCI including myocardial infarction, congestive heart failure, peripheral vascular disease, cerebrovascular disease, dementia, chronic pulmonary disease, rheumatologic disease, peptic ulcer disease, mild liver disease, diabetes, diabetes with chronic complications, hemiplegia or paraplegia, renal disease, any malignancy including leukemia and lymphoma, moderate or severe liver disease, metastatic solid tumors, and AIDS. Each of these conditions is assigned a weight based on the severity of the disease. The weighted sum of the conditions is then created and this sum becomes the CCI score and can range from a low of 0 to a high of 27.

CREATING THE DIAGNOSES USING THE SAS® DATA STEP

The first step in creating a single observation is to read in your data and flag the claims that contain the diagnoses needed to create the CCI. In this example, the data set being read into SAS® contains 5 diagnoses character fields of length 5 that contain the ICD-9-CM code for a particular diagnosis. The idea is to flag the claim with an indicator variable and there are a couple of ways to do this. The first is a lengthy IF-THEN statement, and the second uses an ARRAY and DO loop.

WHAT THE DATA SET LOOKS LIKE BEFORE FLAGGING DIAGNOSES

The data set arrives to the programmer as a text file with a record length of 279. Below is a partial printout of those fields of interest to be used in flagging the diagnoses after having created a permanent SAS data set from the raw claims data.

Obs	studyid	dx1	dx2	dx3	dx4	dx5	proc1	proc2	proc3
1432	M0006	585	25040	2859	2753	2520			
1433	M0006	585	25000	2859	2520				
1436	M0006	585	25042	2859	V048	5888			
1437	M0006	4439							
1438	M0006	5184	78609						
1439	M0006	585	25042	2859	2520				
2141	M0010	3310	2941	4019	53081	2859			
2185	M0010	4439					93926		
2186	M0010	56030					74000		
2203	M0010	3310					99311		
11754	M0048	5640							
11755	M0048	5691	340	56489	6180	5640	458	4869	4620
11756	M0048	340					J0475		
11774	M0048	44020					75716		
11775	M0048	44020					75774		
11776	M0048	7854	4019				71020		
11777	M0048	340					62368		

EXAMPLE 1 – LENGTHY IF-THEN STATEMENT

Using just two of the diagnosis categories, myocardial infarction and peripheral vascular disease, used in creating the CCI, it is illustrated how to flag a claim observation using a series of lengthy IF-THEN statements. The five diagnosis variables, named dx1-dx5, are character and each has length 5. Because ICD-9-CM codes can be of minimum length 3 to a maximum length of 5, depending on the code,

use of the substr function can be used. The code below examines each diagnosis field within each IF-THEN statement and then sets a diagnosis flag to a value of 1 if the condition is met.

```
data claims;
  set in.claims;
  ************************************************
  ** Identification of claims with a MI diagnosis. **
  ************************************************;
  if substr(dx1,1,3) in ('410','412') or
     substr(dx2,1,3) in ('410','412') or
     substr(dx3,1,3) in ('410','412') or
     substr(dx4,1,3) in ('410','412') or
     substr(dx5,1,3) in ('410','412') then mi=1;
  ************************************************
  ** Identification of claims with a PVD diagnosis. **
  ************************************************;
  if substr(dx1,1,3)='441' or substr(dx1,1,4) in ('4439','7854','V434') or
     substr(dx2,1,3)='441' or substr(dx2,1,4) in ('4439','7854','V434') or
     substr(dx2,1,3)='441' or substr(dx3,1,4) in ('4439','7854','V434') or
     substr(dx4,1,3)='441' or substr(dx4,1,4) in ('4439','7854','V434') or
     substr(dx5,1,3)='441' or substr(dx5,1,4) in ('4439','7854','V434')
     or proc1='3848' or proc2='3848' or proc3='3848' then pvd=1;
run;
```

Notice how the code must be duplicated for each variable in order to identify a diagnosis in any one.

EXAMPLE 2 – USING AN ARRAY AND DO LOOP

Using an ARRAY and DO loop one can minimize the amount of code needed to create the same flag variable for each diagnosis and allow for a little more efficiency. An array in SAS® is used to create a set of elements (variables) for which a similar function will be performed. Thus an array is useful in helping to reduce the amount of code that would have been reproduced for each variable. All variables in an array must be of the same type, either numeric or character. To specify that the array contains character variables a '$' in the ARRAY statement is used after the array name. For more information on the syntax of the ARRAY statement see the SAS Online Documentation (2004). Following the definition of the array, the DO loop is specified with a set of instructions/functions to perform on each element of the array. There are several types of DO loops available in SAS®, including a DO loop with an indexing argument, a DO WHILE a condition is satisfied, a DO UNTIL a condition is satisfied, and a DO OVER the elements of the array. As an additional comment, each DO loop must have a resulting "end;" statement to finish the array processing.

In choosing which DO loop to use, the DO OVER loop was selected because there was no need to examine the variables in the array until or while a condition was met, nor was there the need to utilize the index to perform an operation of a subset of the array elements (i.e. using the DO with an index). Rather performing the functions inside the DO loop over all variables in the array was of interest. The SAS code follows for the five diagnosis variables, dx1-dx5, and for the two diagnoses used in the lengthy IF-THEN statements above.

```
  array dx $ dx1-dx5;
  do over dx;
    ************************************************
    ** Identification of claims with a MI diagnosis. **
    ************************************************;
    if substr(dx,1,3) in ('410','412') then mi=1;
    ************************************************
    ** Identification of claims with a PVD diagnosis. **
    ************************************************;
```

```
  if substr(dx,1,3)='441' or substr(dx,1,4) in ('4439','7854','V434')
     or proc1='3848' or proc2='3848' or proc3='3848' then pvd=1;

  then pvd=1;
end;
run;
```

WHAT THE DATA SET LOOKS LIKE NOW

At this point in the program, we have gone from diagnosis codes that can have occurred in many fields within a particular observation to one flagged variable for each diagnosis of interest.

Obs	studyid	dx1	dx2	dx3	dx4	dx5	proc1	proc2	proc3	mi	pvd
1432	M0006	585	25040	2859	2753	2520				.	.
1433	M0006	585	25000	2859	2520					.	.
1436	M0006	585	25042	2859	V048	5888				.	.
1437	M0006	4439								.	1
1438	M0006	5184	78609							.	.
1439	M0006	585	25042	2859	2520					.	.
2141	M0010	3310	2941	4019	53081	2859				.	.
2185	M0010	4439					93926			.	1
2186	M0010	56030					74000			.	.
2203	M0010	3310					99311			.	.
11754	M0048	5640								.	.
11755	M0048	5691	340	56489	6180	5640	458	4869	4620	.	.
11756	M0048	340					J0475			.	.
11774	M0048	44020					75716			.	.
11775	M0048	44020					75774			.	.
11776	M0048	7854	4019				71020			.	1
11777	M0048	340					62368			.	.

USING PROC MEANS TO COMBINE DIANOSIS DATA ACROSS CLAIMS

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=clms; by studyid;
run;

proc means data=clms sum noprint;
  var mi chf pvd cvd ccidem cpd rhemz pud mliverd
      diabnc diabc hpplegia renald cancer msliver mcancer aids;
  by studyid;
```

```
  output out=sumcci sum=smi schf spvd scvd sccidem scpd srhemz spud smliverd
                    sdiabnc sdiabc shpplega srenald scancer smsliver
                    smcancer saids;
run;
```

The SAS® data set “sumcci” has 18 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 number of each type of diagnosis.

Obs	studyid	smi	schf	spvd	scvd	sccidem	scpd	srhemz	spud	smliverd	sdiabnc	sdiabc	shpplega
1	M0001	.	1	.	21	1	1	.	.	.	.	.	.
2	M0002	.	7	.	.	1	.	.	.	.	.	.	.
3	M0003	3	3	.	6	.	.	.	.	.	.	.	.
4	M0004	.	.	.	.	1	.	.	.	.	.	.	.
5	M0006	.	2	1	14	.	.	.	.	.	23	33	8
6	M0007	.	.	.	.	.	.	.	.	.	16	.	.
7	M0008	.	.	.	.	.	.	.	.	.	.	.	.
8	M0009	.	.	.	.	.	.	.	.	.	.	.	.
9	M0010	.	.	1	.	24	.	.	.	.	.	.	.
10	M0011	.	.	.	.	1	.	.	.	.	.	.	.

Obs	srenald	scancer	Smsliver	smcancer	saids
1	.	.	.	.	.
2	13	2	.	.	.
3	.	.	.	.	.
4	.	.	.	.	.
5	203	.	.	.	.
6	.	4	.	.	.
7	.	.	.	.	.
8	.	.	.	.	.
9	.	.	.	.	.
10	.	.	.	.	.

CREATING THE CCI SCORE FOR EACH INDIVIDUAL

The first step in calculating the CCI score for each individual 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. “smi” is first in the “s” array, and “imi” is first in the “i” array).

```
data cci;
  set sumcci; by studyid;
  array s smi schf spvd scvd sccidem scpd srhemz spud smliverd
          sdiabnc sdiabc shpplega srenald scancer smsliver smcancer
          saids;
  array i imi ichf ipvd icvd iccidem icpd irhemz ipud imliverd
          idiabnc idiabc ihpplega irenald icancer imsliver imcancer
          iaids;
  do over s;
    if s>=1 then i=1;
      else i=0;
  end;
```

After creating the indicator variables, the number of diagnoses that have the same CCI weight is determined and the corresponding weighted sum. The end result is the sum of the weight sums to obtain the overall CCI score.

```
  cci1=0; cci2=0; cci3=0; cci6=0;
  array one imi ichf ipvd icvd iccidem icpd irhemz ipud imliverd idiabnc;
  array two idiabc ihpplega irenald icancer;
  array six imcancer iaids;
  do over one;
    if one=1 then cci1=cci1+1;
  end;
  do over two;
    if two=1 then cci2=cci2+2;
  end;
  do over six;
    if six=1 then cci6=cci6+6;
  end;
  if imsliver=1 then cci3=cci3+3;
  cci=cci1+cci2+cci3+cci6;
run;
```

The final data set with one observation per individual is below.

Obs	studyid	imi	ichf	ipvd	lcvd	iccidem	icpd	irhemz	ipud	imliverd	idiabnc	idiabc
1	M0001	0	1	0	1	1	1	0	0	0	0	0
2	M0002	0	1	0	0	1	0	0	0	0	0	0
3	M0003	1	1	0	1	0	0	0	0	0	0	0
4	M0004	0	0	0	0	1	0	0	0	0	0	0
5	M0006	0	1	1	1	0	0	0	0	0	1	1
6	M0007	0	0	0	0	0	0	0	0	0	1	0
7	M0008	0	0	0	0	0	0	0	0	0	0	0
8	M0009	0	0	0	0	0	0	0	0	0	0	0
9	M0010	0	0	1	0	1	0	0	0	0	0	0
10	M0011	0	0	0	0	1	0	0	0	0	0	0

Obs	ihpplega	irenald	icancer	imcancer	iaids	cci
1	0	0	0	0	0	4
2	0	1	1	0	0	6
3	0	0	0	0	0	3
4	0	0	0	0	0	1
5	1	1	0	0	0	10
6	0	0	1	0	0	3
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	2
10	0	0	0	0	0	1

CONCLUSIONS

Creating a single observation for an individual with data that come from many observations with many variables can seem, initially, like a difficult task. However, using ARRAYs and DO loops in the DATA step to flag individual observations and then creating the sum of these indicator variables for each individual using PROC MEANS, this task can become relatively simple and straight forward.

The complete code for creating the CCI is given in the Appendix.

AUTHOR BIOGRAPHY AND CONTACT INFORMATION

Jennifer Waller obtained a PhD in Biostatistics from the School of Public Health at the University of South Carolina in 1994. She is an associate professor in the Department of Biostatistics at the Medical College of Georgia where she has worked for the past 8 years teaching, consulting and doing collaborative research. She has used SAS for 16 years. Jennifer can be contacted by mail, phone or E-mail

Jennifer L. Waller
Department of Biostatistics (AE-3031)
Medical College of Georgia
Augusta, GA 30912-4900
(706) 721-0814
jwaller@mcg.edu

DISCLAIMER

REFERENCES

1. Charlson ME, Pompei P, Ales KL, Mackenzie CR. A new method of classifying prognostic comorbidity in longitudinal studies: development and validation. Journal of Chronic Disease 40: 272-383, 1987.
2. Deyo RA, Cherkin DC, Ciol MA. Adapting a clinical comorbidity index for use with ICD-9-CM administrative databases. Journal of Clinical Epidemiology 45(6):613-619, 1992.

Appendix

```
libname in 'path to sasdatasetname';

data claims;
  set in.sasdatasetname;
  array dx dx1-dx5;
  do over dx;
    if substr(dx,1,3) in ('410','412') then mi=1;
    if substr(dx,1,3) in ('428') then chf=1;
    if substr(dx,1,3)='441' or substr(dx,1,4) in ('4439','7854','V434')
       or proc1='3848' or proc2='3848' or proc3='3848' then pvd=1;
    if substr(dx,1,3) in ('430','431','432','433','434','435',
       '436','437','438') then cvd=1;
    if substr(dx,1,3)='290' then ccidem=1;
    if substr(dx,1,3) in ('490','491','492','493','494','495','496',
       '500','501','502','503','504','505') or
       substr(dx,1,4)='5064' then cpd=1;
    if substr(dx,1,4) in ('7100','7101','7104','7140','7141','7142') or
       substr(dx,1,3)='725' or substr(dx,1,5)='71481' then rhemz=1;
    if substr(dx,1,3) in ('531','532','533','534') then pud=1;
    if substr(dx,1,4) in ('5712','5715','5716','5714') then mliverd=1;
    if substr(dx,1,4) in ('2500','2501','2502','2503','2507') then diabnc=1;
    if substr(dx,1,4) in ('2504','2505','2506') then diabc=1;
    if substr(dx,1,3)='342' or substr(dx,1,4)='3441' then hpplegia=1;
    if substr(dx,1,3) in ('582','583','585','586','588') then renald=1;
    if substr(dx,1,3) in ('140','141','142','143','144','145','146','147',
       '148','149','150','151','152','153','154','155','156','157','158','159',
       '160','161','162','163','164','165','166','167','168','169',
       '170','171','172','174','175','176','177','178','179'
       '180','181','182','183','184','185','186','187','188','189',
       '190','191','192','193','194','195',
       '200','201','202','203','204','205','206','207','208') then cancer=1;
    if substr(dx,1,4) in ('5722','5723','5724','5725','5726','5727','5728',
       '4560','4561') or
       substr(dx,1,5) in ('45620','45621') then msliver=1;
    if substr(dx,1,3) in ('196','197','198') or substr(dx,1,4) in ('1990','1991')
       then mcancer=1;
    if substr(dx,1,3) in ('042','043','044') then aids=1;
  end;
  keep studyid dx1-dx5 proc1-proc3 mi chf pvd cvd ccidem cpd rhemz pud mliverd
       diabnc diabc hpplegia renald cancer msliver mcancer aids;
run;

proc sort data=claims; by studyid;
run;

proc means data=claims sum noprint;
  var mi chf pvd cvd ccidem cpd rhemz pud mliverd
      diabnc diabc hpplegia renald cancer msliver mcancer aids;
  by studyid;
  output out=sumcci sum=smi schf spvd scvd sccidem scpd srhemz spud smliverd
```

```
                    sdiabnc sdiabc shpplega srenald scancer smsliver smcancer
                    saids;
run;

data cci;
  set sumcci; by studyid;
  array s smi schf spvd scvd sccidem scpd srhemz spud smliverd
          sdiabnc sdiabc shpplega srenald scancer smsliver smcancer
          saids;
  array i imi ichf ipvd icvd iccidem icpd irhemz ipud imliverd
          idiabnc idiabc ihpplega irenald icancer imsliver imcancer
          iaids;
  do over i;
    if s>=1 then i=1;
      else i=0;
  end;
  cci1=0; cci2=0; cci3=0; cci6=0;
  array one imi ichf ipvd icvd iccidem icpd irhemz ipud imliverd idiabnc;
  array two idiabc ihpplega irenald icancer;
  array six imcancer iaids;
  do over one;
    if one=1 then cci1=cci1+1;
  end;
  do over two;
    if two=1 then cci2=cci2+2;
  end;
  do over six;
    if six=1 then cci6=cci6+6;
  end;
  if imsliver=1 then cci3=cci3+3;
  cci=cci1+cci2+cci3+cci6;

run;
quit;
run;
```