Abstract:
Kaiser Permanente Colorado (KPCO) participates in the Cancer Research Network’s (CRN’s) Strategic Data Resources Core (SDRC). This committee is responsible for overseeing the creation of Virtual Data Warehouses (VDWs) across 10 HMOs throughout the United States. The VDWs are comprised of different SAS® tables which are to be uniform in structure across sites.

This paper will show, in three steps, how KPCO used SAS to reformat local enrollment data into the structure prescribed by the SDRC:
1) Code each record for different types of insurance coverage.
2) Combine overlapping membership periods for records with the same insurance coverage.
3) Reformat KPCO’s data into one record per member per month and year of membership.

Advantages and disadvantages of different methods of pulling and storing data will be discussed. Pulling data in an ad hoc manner is one method, while using indexes to store data in one or three year files is another.

Background:
Although each of the 10 sites within the national CRN can access their local data, the structure and format of that data varies greatly by site. Varying structure makes it nearly impossible to run efficient SAS code at multiple sites in a timely manner. The SDRC was established in part to define national SAS VDW table structures. Each site was responsible for converting their data to the VDW prescribed structure and storing that data.

Data Structures:
The SDRC-defined VDW enrollment table required one record per person per month and year combination in which that person was enrolled, with a medical contract, at a site. Five YES/NO fields were defined for insurance carrier, so that it would be evident if a member were covered by two insurance types in any given month. Other fields to be included were MRN (the identifier unique to each individual), enrollment month, and enrollment year. The prescribed structure of the enrollment table was as follows:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Variable Definition</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRN</td>
<td>Identifier unique to an individual</td>
<td>Character. Unique to each HMO</td>
<td>Used to link across files</td>
</tr>
<tr>
<td>Enr_Month</td>
<td>Numeric 4</td>
<td>1-12</td>
<td></td>
</tr>
<tr>
<td>Enr_Year</td>
<td>Numeric 4</td>
<td>Values of the form 1980</td>
<td>Whatever time period works at your HMO</td>
</tr>
<tr>
<td>Ins_Medicare</td>
<td>Insurance Medicare</td>
<td>&quot;Y&quot;=Yes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Char 1</td>
<td>&quot; &quot; = No or missing</td>
<td></td>
</tr>
<tr>
<td>Ins_Medicaid</td>
<td>Insurance Medicaid</td>
<td>&quot;Y&quot;=Yes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Char 1</td>
<td>&quot; &quot; = No or missing</td>
<td></td>
</tr>
<tr>
<td>Ins_Commercial</td>
<td>Insurance Commercial</td>
<td>&quot;Y&quot;=Yes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot; &quot; = No or missing</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A year's worth of data for one person then might look like this:

<table>
<thead>
<tr>
<th>Enr_Month</th>
<th>Enr_Year</th>
<th>Ins_Medicare</th>
<th>Ins_Medicaid</th>
<th>Ins_Commercial</th>
<th>Ins_PrivatePay</th>
<th>Ins_Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2002</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>2002</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>3</td>
<td>2002</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>2002</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>5</td>
<td>2002</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>6</td>
<td>2002</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>7</td>
<td>2002</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>8</td>
<td>2002</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>9</td>
<td>2002</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Rather than one record per person per month per year, KPCO’s DB2 membership table contains fields for the enrollment start and cancel dates. Thus, a year of enrollment data can be kept in just one record rather than 12. KPCO has multiple benefit fields which, either alone or when combined, comprise a contracting entity. One benefit field specifies the group while another specifies the subgroup of the member. Dual coverage for a member (i.e., simultaneous coverage by Medicare and Private Pay), can either be identified by a combination of two fields or two separate records (one for each carrier). There can be overlapping membership periods, some of which can be attributed to dual insurance coverage.

A portion of KPCO’s membership data structure is shown below, followed by a sample of their membership data:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Variable Definition</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRN</td>
<td>Identifier unique to an individual; Numeric 8</td>
<td>123456789 (example)</td>
<td>Used to link across files</td>
</tr>
<tr>
<td>BEGDATE</td>
<td>Numeric 8</td>
<td>20000101</td>
<td>Date coverage for a member began</td>
</tr>
<tr>
<td>ENDDATE</td>
<td>Numeric 8</td>
<td>20001231</td>
<td>Date coverage for a member ended</td>
</tr>
<tr>
<td>GRP_ID</td>
<td>Numeric 8</td>
<td>6616 (example)</td>
<td>Number used to identify the member's group number</td>
</tr>
<tr>
<td>SGRP_ID</td>
<td>Char 3</td>
<td>001 (example)</td>
<td>Number used to identify a member's subgroup number</td>
</tr>
</tbody>
</table>
Sample data for two MRNs (unique identifier):

<table>
<thead>
<tr>
<th>MRN</th>
<th>begdate</th>
<th>enddate</th>
<th>xxmed_pln_id</th>
<th>grp_id</th>
<th>sgrp_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>20010101</td>
<td>20010630</td>
<td>RG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12345</td>
<td>20020101</td>
<td>20020630</td>
<td>0</td>
<td></td>
<td>8085</td>
</tr>
<tr>
<td>12345</td>
<td>20020101</td>
<td>20021231</td>
<td>RS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12345</td>
<td>20030101</td>
<td>20031231</td>
<td>MF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56789</td>
<td>20020101</td>
<td>20020430</td>
<td>RG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56789</td>
<td>20020201</td>
<td>20020630</td>
<td>RG</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From Simple to Complex: Transforming Data to the Correct Format

Step 1: IF/THEN/ELSE
This was the simplest step which utilized IF/THEN/ELSE logic. Each individual was coded as having one or more of the five types of insurance coverage defined by the SDRC. A section of the code, identifying Medicare and Medicaid individuals, is shown below:

```
DATA enroll_tmp1;
  SET benefitb;
  medicare=0;
  medicaid=0;

  /* Medicare / Medicaid */
  IF xxmed_pln_id IN('C','MF','RG','RM','RO','RP','RS','MF') THEN medicare = 1;
  IF grp_id IN(9990,9991,9992,9993,9994,9995,9996,9997,9998) THEN medicaid = 1;
  ELSE IF grp_id = 1300 AND sgrp_id IN('080','081') THEN DO;
    medicaid = 1;
    medicare = 1;
  END;
RUN;
```

Similar coding identified individuals as fitting into other, comm., and pvtpay fields. The sample KPCO data shown in the introduction then looked as follows:

<table>
<thead>
<tr>
<th>MRN</th>
<th>begdate</th>
<th>enddate</th>
<th>medicare</th>
<th>medicaid</th>
<th>other</th>
<th>comm</th>
<th>pvtpay</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>20010101</td>
<td>20010630</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12345</td>
<td>20020101</td>
<td>20020630</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>12345</td>
<td>20020101</td>
<td>20021231</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12345</td>
<td>20030101</td>
<td>20031231</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>56789</td>
<td>20020101</td>
<td>20020430</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>56789</td>
<td>20020201</td>
<td>20020630</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Note that although the format of the data is to be Y/N for the different payer fields, it is programmed to be 1/0 at this point.

Step 2: Overlapping Membership Periods
KPCO’s membership system allows for overlapping membership records which needed to be combined. For an example, look at the data above. MRN 56789 has the same values for insurance coverage (both have medicare equal 1 and all other payer fields equal 0). The membership periods for the two observations overlap and could be combined to create one record. The one observation then would have begdate equal to 20020101 and enddate equal to 20020630.

The last two observations for MRN 12345 can also be combined because there is no gap in coverage dates and the insurance coverage is the same for both observations. The second and third observations for MRN 12345 have overlapping membership periods, but different values for the payer fields. Therefore those observations cannot be combined.

The following code shows how membership periods are combined¹. First, the data are sorted by relevant variables and then a combination of the lag function and IF/THEN/ELSE statements are used to format the data. The resulting data is shown following the code.

```sas
PROC SORT DATA = bttemp.membership1b NODUPKEY;
   BY indv_hrn begdate enddate medicare medicaid pvtpay other comm;
RUN;

PROC SORT DATA=bttemp.membership1b;
   BY indv_hrn medicare medicaid pvtpay other comm begdate enddate;
DATA data3(drop=indv_hrn2 medicare2 medicaid2 other2 commercial2 pvtpay2 enddate
      RENAME=(begdate=begdate begin_dos end2=end_dos));
SET bttemp.membership1b;
RETAIN end2;
indv_hrn2=LAG1(indv_hrn);
medicare2=LAG1(medicare);
medicaid2=LAG1(medicaid);
other2=LAG(other);
comm2=LAG1(comm);
pvtpay2=LAG1(pvtpay);
IF indv_hrn2=indv_hrn AND medicare2=medicare AND medicaid2=medicaid AND
   other2=other AND comm2=comm AND pvtpay2=pvtpay AND begdate le (end2+1)
THEN DO;
   begdate=end2;
   end2=max(enddate,end2);
END;
ELSE DO;
   seq+1;
   end2=enddate;
END;
FORMAT end2 mmddyy10.;
RUN;
```

Step 3: Macros, Arrays, and DO Loops
Finally the data to be reformatted so there was one record per member per month and year of membership. This was done using a DO loop. The DO loop required knowing the greatest number of observations; first per member, accounting for all the payer fields, and then for all the resulting observations. A first SQL query was used to get a count of the number of observations per member and payer. A second query used that count to determine the maximum number of all observations. The resulting data set, maxnum2, was then utilized by the NULL data step, where a macro variable, maxn, was created. &Maxn contains the maximum number of observations.

```
/* Determine the max # of observations, but indv_hrn and payer, per person */
PROC SQL;
    CREATE TABLE maxnum AS
    SELECT indv_hrn, medicare, medicaid, pvtpay, other, comm, COUNT(begdate) AS maxnum
    FROM test3
    GROUP BY indv_hrn, medicare, medicaid, pvtpay, other, comm;
    CREATE TABLE maxnum2 AS
    SELECT MAX(maxnum) AS maxnum
    FROM maxnum;
QUIT;

DATA _null_; SET maxnum2;
    CALL SYMPUT('maxn',TRIM(LEFT(PUT(maxnum,5.))));
RUN;
```

In the example case, maxn is equal to two, since two observations of MRN 12345 have the same payer.
The data were then transformed via PROC TRANSPOSE, and combined with the maximum number of all observations in the previous step via merging data sets. This was all done as follows, in preparation for array processing.

```
PROC SORT DATA=test3;
  BY indv_hrn medicare medicaid other commercial pvtpay;
/* Transpose data to be able to assign months of membership */
PROC TRANSPOSE DATA=test3 OUT=begdate PREFIX=begdate;
  VAR begdate;
  BY indv_hrn medicare medicaid other commercial pvtpay;
PROC TRANSPOSE DATA=test3 OUT=enddate PREFIX=enddate;
  VAR enddate;
  BY indv_hrn medicare medicaid other commercial pvtpay;
DATA test4;
  MERGE begdate enddate;
  BY indv_hrn medicare medicaid other commercial pvtpay;
  DROP _name_;
DATA test5;
  MERGE test4 maxnum;
  BY indv_hrn;
RUN;
```

Arrays and a DO loop were used to determine if a person was a member during a specific month. In the following code, \&startyear and \&endyear can be defined as any years. For testing purposes, KPCO used \&startyear=2001 and \&endyear=2003. Note that at this point the insurance variables were recoded as N/Y.

```
%LET startyear=2001;
%LET endyear=2003;

%MACRO dates(lastday,month,monthnum);
  %DO j=&startyear %TO &endyear;
    /* Determines if individuals were members during a particular month */
    DATA &month.&j;
    SET test5;
    DO i=1 TO &maxn;
      count=0;
      ARRAY begdate(&maxn) begdate1-begdate&maxn;
      ARRAY enddate(&maxn) enddate1-enddate&maxn;
      IF begdate(i) NE . AND indv_hrn NE . THEN DO;
        IF begdate(i) LE "&lastday.&month.&j."d AND enddate(i) GE "1.&month.&j."d THEN DO;
          enr_month=&monthnum; enr_year=&j.;
          OUTPUT &month.&j.;
        END;
      END;
    END;
  END;
RUN;
```
In the example, MRN 12345 had 30 records. One record was present in each month from January, 2001 through June, 2001 and then again from January, 2002 through December, 2003.

From the example, the January, 2002 record looked as follows:

<table>
<thead>
<tr>
<th>Enr_Month</th>
<th>Enr_Year</th>
<th>Ins_Medicare</th>
<th>Ins_Medicaid</th>
<th>Ins_Commercial</th>
<th>Ins_PrivatePay</th>
<th>Ins_Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2002</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

MRN 56789 had six records. One record was present in each month from January, 2002 through June, 2002. The April, 2002 record looked as follows:

<table>
<thead>
<tr>
<th>Enr_Month</th>
<th>Enr_Year</th>
<th>Ins_Medicare</th>
<th>Ins_Medicaid</th>
<th>Ins_Commercial</th>
<th>Ins_PrivatePay</th>
<th>Ins_Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2002</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
The code was now written so the data could be in the nationally prescribed format. The next step was to determine how best to access and store the data, and whether the data should be stored at all.

**Accessing and Storing the Data:**
Special considerations needed to be given to future storing and accessing of the data. The data could be accessed in any of a multi-year data set, yearly data sets, or monthly data sets. Data did not necessarily need to be stored if a program could be written to access and format data easily.

To test the best way to access the data, a sample request was made to pull enrollment data for a subset of women aged 45-64. If the women were present in each of the previous 12 months of enrollment data, from January 1, 2002 to December 31, 2002, they were kept in the data set.

Three ways of storing the enrollment data were tried:
1) All enrollment data was pulled for 2001-2003; one record per member per month of enrollment, stored in one data set.
2) All enrollment data was pulled for 2001-2003, one record per member per month of enrollment, stored in separate data sets by year (three data sets).
3) Enrollment data was pulled "on the fly" as needed for this request (members as of 1/1/2002 who were also members through 12/31/2002)

The time spent pulling and formatting the three-year data set was 10:19.58 real (6.03 CPU) while the time spent pulling and formatting the one-year data sets (three-separate data sets) was 9:40.55 real (8.69 CPU).

An index was set up on the three-year data set to help increase efficiency when pulling the data. Setting up an index (enrollment month and year) for the three-year file took 21:43.31 real time (34.52 CPU).

The subset of women aged 45-60 was then merged with the enrollment files (three-year data set, one-year data set, and "on the fly" data set). Here are the results of those merges:

- Three-year data set :  32.54 sec real (3.16 sec CPU)
- One-year data set:  31.73 sec real (3.09 sec CPU)
- "On the fly" data set:  2:05.99 real (19.94 sec CPU)

There was not much difference between pulling data from the three-year versus the one-year data sets. This is mainly due to the fact that an index was created for enrollment month and year. Pulling data "on the fly" took a lot more time, but it did not require the file creation times that the one and three-year files did. If a person is to be given just one data request, the "on the fly" method would be the recommended way to get the data. However, due to the fact many data requests will be made to access this data, using the one and three-year files was deemed better in the long run.

**Conclusion**
Much was learned from this entire process. There are many ways of coding which makes pulling and formatting data easier. In addition, there are many ways to store data and each must be fully evaluated before a decision is made on which method to use.
Trademarks
SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries.
© indicates USA registration.

References

Contact Information:
Beth Newsom
Kaiser Permanente Colorado
Elizabeth.E.Newsom@kp.org

Jennifer Ellis
Kaiser Permanente Colorado
Jenn.L.Ellis@kp.org