SAS Programming and HEDIS: An Introductory Overview

Christopher Plummer, Keystone Health Plan Central, Camp Hill, PA

ABSTRACT

The Health Plan Employer Data and Information Set, or HEDIS®, a product of the National Committee for Quality Assurance (NCQA), "is a set of standardized performance measures designed to ensure that purchasers and consumers have the information they need to reliably compare the performance of managed health care plans. The performance measures in HEDIS are related to many significant public health issues such as cancer, heart disease, smoking, asthma and diabetes." (NCQA, *WWW.NCQA.ORG*)

For Health Plans, HEDIS presents interesting Information Systems and programming challenges that the SAS System® has been critical in helping our team over-come. This paper will explain the pitfalls we encountered and, more importantly, the solutions that kept us rolling. Procedures discussed will include Connect®, Upload®, Download® and SQL®, as well as By-Group processing in the Data Step. Someone new to the SAS® System will get a comprehensive example of what can be done using these components.

INTRODUCTION

If you have limited experience with, or exposure to, SAS programming and want to know about some of the neat things you can do, this paper should serve as an excellent starting point. First, I will briefly explain HEDIS and the challenges it presented us, and, second, I will explain how the many useful components of SAS enabled us to meet those challenges.

HEDIS

Definition

Arguably, HEDIS is the most important quantitative criteria for managed care organizations as it provides a useful and meaningful basis of comparison of historical performance. This basis is achieved by adherence to particular technical specifications for measuring 56 key indicators of medical delivery. Comprised in a 6 volume set, these specifications cover 8 domains: Effectiveness of Care, Access/Availability to Care, Satisfaction with the Experience of Care, Use of Services, Cost of Care, Informed Health Care Choices and Health Plan Descriptive Information.

These specifications define which indicators to study, the period, member inclusion criteria and the data collection and rate calculation procedures. For example, with the "Childhood Immunization Status" measure, the denominator criteria includes members of the plan that turn 2 during the reporting year, that had continuous coverage between the first and second birthday (with one break of up to 45 days). Inclusion in each of the measure's six numerators requires members in the denominator to have had a different immunization shot administered by the second birthday.

The "Effectiveness" domain includes the most complicated and resource intensive measures because after selecting the denominator, a sample is drawn and medical chart reviews are usually necessary to find numerator information that is missing from the plan's claims and encounter database.

Measures

NCQA has attempted to shift attention from the process of care to the outcomes from care, but basically, HEDIS answers the "how many" question of managed care. How many kids were immunized? How many heart attack survivors received Beta-Blockers? At this time, no measure reports the result of a medical intervention (e.g., how effective was the immunization in preventing the illness?)

Notable measures include:

- Childhood Immunizations
- Adolescent Immunizations
- Breast Cancer Screening
- Cervical Cancer Screening
- Chlamydia Screening in Women
- Check-Ups After Delivery
- Controlling High Blood Pressure
- Comprehensive Diabetic Care

Uses & Examples

HEDIS rates can be found in various newspaper and magazine articles under titles like “Best Health Plan,” such as those found in Newsweek, U.S. News & World Report and Managed Healthcare. For these articles, HEDIS numbers are voluntarily collected from Plans across the nation and re-worked according to the methodology that the publisher thinks presents the most useful information. One of the areas that HEDIS has impacted the most involves the selection of health plans by employers. Routinely, HEDIS rates are requested when premium rates and contract information is gathered, and many employers hire consultants to design and implement comparative studies including the HEDIS rates and other related information. The requests for, and use of, HEDIS rates in the contracting phase has steadily grown since NCQA first introduced the measures.

KHPC

History

Keystone Health Plan Central, a 200,000+ member Blue Cross Blue Shield HMO Affiliate, has been operating since 1984 from its headquarters in Central Pennsylvania. In addition to its commercial product, KHPC has a Medicare+Choice product called SeniorBlue that started operating in 1996.

NCQA completed a review of both product lines earlier this year and will announce their findings and the accreditation status of each later this year. In 1999, NCQA started to include HEDIS rates in assessing accreditation status, and currently, KHPC enjoys the distinction of the *Excellent* status, which is an achievement that less than 10% of all plans accomplish. KHPC has been reporting HEDIS results for its commercial product since 1994 and Medicare results since 1996. These results, and the method KHPC employs in producing them, are audited by an independent NCQA certified organization much like financial information is audited.

Information Systems

At KHPC, claims for services and encounters (under capitation, some procedures are performed under a per capita arrangement (not fee-for-service) but the encounter is still recorded) are processed and the data stored in a mainframe DB2 database. The KHPC Research and Analysis department downloads these claims monthly and stores them in SAS datasets on local servers. The R&A Repository, as it is called, also includes third party data (i.e., lab vendor data, behavioral health data) that has been imported into SAS datasets, and it is against the Repository that all HEDIS code is run.

By its design, HEDIS encourages the use of standardized codes, collection forms and

complete data in expansive information systems. The better the IS; the easier it is to report HEDIS numbers. A complete audit of a plan's IS, data collection, storage and access policies is conducted each year. A plan has to explain any changes, deficiencies or complicated procedures that might effect the HEDIS results, and if the explanation is unsatisfactory or a bias in the results is found, the plan may receive an unreportable score for the measures effected.

Staffing Issues

As explained, R&A writes the code and calculates the rates for HEDIS, but another important ingredient is the nurses that must collect information from the medical record because the claims and encounter database are not complete. These nurses have to be trained in HEDIS criteria and instructed on how the information must be collected, and therefore, their part is most important in a successful HEDIS production. Using the guidelines that R&A prepares, the nurses visit the physicians' offices and review medical charts looking for information that meets numerator criteria.

HEDIS CHALLENGES

The rest of this paper will focus on code that creates the denominator for Childhood Immunizations, which measures what percentage of the Plan's population, meeting the Continuous Enrollment criteria, received certain vaccinations by age 2.

Simply stated, Continuous Enrollment (CE) is having healthcare insurance coverage for a stated period of time. Continuous Enrollment (CE) for the Measure specifies that a child must be enrolled for "twelve months immediately preceding the child's second birthday," with "no more than one gap … of up to 45 days" (NCQA 1998, p.35) during this period.

One challenge you face is that a child could be covered under multiple contracts, as occurs when both parents have coverage with the same Plan. When this occurs, the child could have multiple identification numbers (IDs) and multiple coverage periods. Periods that overlap or abut must be coalesced into one record with an effective and termination date.

THE SAS SOLUTION

The enrollment data is housed as DB2® files on our mainframe, but with Proc Connect and Proc SQL, this data is easily retrievable as coded below. First, working from the inside out, you select the fields you need from the DB2 tables and convert them to SAS V6 8 character length names. (You won't need to do this with Version 8, but you might want to for simplicity's sake.)

```
(select a.mem_med_record_id as medrecn,
        a.sub_id||a.mem_suffix_cd as id,
        a.mem_first_name as mem_fn,
        a.mem_last_name as mem_ln,
        a.mem_middle_initial as mem_mi,
        a.mem_sex_cd as sex,
        a.mem_birth_dt as dob,
        b.mem_status_effdt as eff_date,
        b.mem_status_cnldt as cnl_date
```

Second, you select the tables from which you are pulling the fields.

```
from  khp2apr.vk00023s a,
      khp2apr.vk00025s b
```

Third, you link the tables together on the common fields.

```
where a.sub_id=b.sub_id and
      a.mem_suffix_cd=b.mem_suffix_cd and
```

Fourth, you add code to limit the records meeting your time criteria.

```
 ((b.mem_status_cnldt is null or
   b.mem_status_cnldt > '1996-12-31') and
   b.mem_status_effdt < '1999-01-01'))
```

Fifth, you add code to create the new fields *bday1* and *bday2* (1st and 2nd birthdays) at the top. You put the month of the *dob*, the day of the *dob* minus one if the *dob* is Feb. 29 and the

year 1997 or 1998 into the mdy function to calculate the birthdays.

```
mdy(month(dob),day(dob)-(month(dob)=2 and
day(dob)=29),1997) as bday1
format=mmddyy8.,
mdy(month(dob),day(dob)-(month(dob)=2 and
day(dob)=29),1998) as bday2
```

... and then you use the new fields when you add some additional criteria to the bottom,

```
where year(dob)=1996 and
     (cnl_date>=(calculated bday1) or
cnl_date=.) and
      eff_date<=(calculated bday2));
```

Sixth, you need to add the code that starts Proc Connect at the beginning.

```
options remote=sas;
options comamid=tcp;
signon "c:\sas\tcptso.scr";

rsubmit;
proc sql;
connect to db2 (ssid=db2p);
```

Seventh, you add the code that sends the data to the PC and ends the connection.

```
proc download data=dwn_ld out=rawmembs;
run;

endrsubmit;

signoff "c:\sas\tcptso.scr";
```

Finally, your code is ready to pull all enrollment data for members born in 1996 (and, therefore, turning two in 1998) and whose coverage became active at least prior to their second birthday or terminated at least after their first. The criteria are loose, but they will remove some unwanted records and make your download quicker. You could apply more stringent criteria, but some systems have time and space limits (because of cost) that will only allow so much processing “on-line”. You will apply tighter criteria in the following sets of code to fully remove all unneeded records..

Removing Unneeded Records

Now that your data is on the PC, you can use SQL code to further remove unneeded records. Members that clearly do not have at least a minimum effective date prior to 45 days after the first birthday and a termination date that is null or after the second birthday need to be removed. The resulting dataset will have members whose coverage spans the 12 months between birthdays 1 and 2, as specified in the measure.

```
proc sql;
create table mults as
select distinct * from rawmembs
where medrecn in
(select distinct medrecn
 from rawmembs
 group by medrecn
 having (min(eff_date)<=(bday1+45) and
(max(cnl_date)>=bday2 or cnl_date=.)))
order by medrecn, eff_date;
```

Coalescing The Coverage

Next, you can coalesce the coverage of those members who have multiple periods (records) and check for Continuous Enrollment using By-Group processing in a Data Step. This code is a little more tricky and requires that you understand the automatic variables that SAS creates when a by statement is used.

Because of the "by medrecn;" statement, SAS will automatically create the variables *first.medrecn* and *last.medrecn* in the code below. When a member has multiple records in the dataset and SAS encounters the first record it will assign a value of 1 to *first.medrecn* and 0 to *last.medrecn*. You can think of it as 1=True and 0=False. As you will see in the code below, these auto-variables are very handy for controlling the processing.

Before you can apply the dataset code, first, you need to sort the dataset by the variable that you want the records grouped by, the *medrecn*. You also want to further sort by the effective date so

that the coverage dates will be in chronological order.

You will create some temporary variables with a retain statement that you will use to store information for comparisons with other records. For instance, for members with multiple records, you will want to compare the first record's termination date with the second record's effective date to see if a gap in coverage occurred and for how long. Remember that the specification allows one gap of no more than 45 days, so you will need to track the number and length of any gaps.

The temporary variables (temporary because you remove them with a drop statement) *edate*, *ledate*, *cdate*, *brk*, *done* and *days* will be assigned values throughout the process. Some permanent variables will take these values when needed.

Essentially, *done* controls the flow of the processing: when done=0 processing continues, when done=1 criteria have been met and when done=3 the gap criterion has been violated and the member fails with no further testing. The temporary variables *edate*, *ledate* and *cdate* retain date information, and the variables *brk* and *days* retain the count of gaps and the length in days, respectively.

There are three sections of actual processing which correspond to the two types of members: those with only one record of coverage (one section applies) and those with multiple records (two sections apply). The process is simple for the first type of member; either they meet the criteria or they are excluded.

The second type of member requires checking using the retained variables if the first record does not meet the criteria alone. The additional records must be tested and if they meet the criteria, the final record will include the *medrecn*, one effective date (the original one retained in *ledate*), one termination date (that may be null if the member is still active) and name, age and sex information.

The code starts like a normal data-step …

```
data denom(drop=brk done edate cdate ledate
                days);
set mults;
format edate cdate ledate mmddyy8.;
```

but, then you add the by and retain statements.

```
by medrecn;
retain edate cdate ledate .
       brk done days 0;
```

Then you add the first section of criteria …

```
/*single record per medrecn handled here*/
if (first.medrecn and last.medrecn and
(cnl_date=. or cnl_date=>bday2)) then do;
       done=1; brk=0; end;
```

and the second …

```
/*first of multiple records per medrecn
handled here*/
else if first.medrecn then do;

   /*is there an initial break?*/
     if eff_date>bday1 then do;
           brk=1; days=eff_date-bday1; end;
     else do;
           brk=0; days=0; end;

/*does this record pass on its own?*/
     if (cnl_date=. or cnl_date=>bday2)
then done=1;
     else do; edate=eff_date;
cdate=cnl_date; ledate=eff_date; done=0;
end; end;
```

and the third.

```
/*additional of multiple records per
medrecn handled here*/
else if done=0 then do;

   /*is there a break?*/
    if eff_date>cdate then do;
       brk+1; days=eff_date-cdate;

   /*is there multiple breaks or is the
     break longer than 45 days?*/
    if (days>45 or brk>1) then done=3; end;

    if (done=0 and (cnl_date=. or
        cnl_date=>bday2)) then do;
          done=1; eff_date=edate; end;
    else if (done=0 and last.medrecn=0)
```

```
            then cdate=cnl_date; end;
```

The final record is output if all criteria have been met…

```
if done=1 then output;
run;
```

CONCLUSION

The Childhood Immunization denominator has been created with members who have met the criteria and you have used Proc Connect, Proc SQL, Proc Download, by-group processing and the retain statement. You can use Proc Upload in just the same fashion as Proc Download if you wanted to send some data up to the mainframe. For example, if you wanted to collect the claims for the members in the denominator you just created, you could use SQL to create a table of only the *medrecn*s and upload it with the following code.

```
proc sql;
create table uplink as select distinct
medrecn from denom;

options remote=sas;
options comamid=tcp;
signon "c:\sas\tcptso.scr";


rsubmit;
proc upload data=uplink;
run;
```

Once you upload this file, you can match on the *medrecn* field in the claims files.

TRADEMARKS

REFERENCES

Reprinted with permission from the National Committee for Quality Assurance; HEDIS 1999, Volume 2: Technical Specifications; ©1998 by NCQA.

CONTACT INFORMATION

Christopher Plummer
Community Health Care Analyst
300 Corporate Center Drive
Camp Hill, PA 17011
Phone: (717) 760-9544
Fax: (717) 975-6895
Email: Christopher.Plummer@KHPC.com

SAS CODE APPENDIX

```
options remote=sas;
options comamid=tcp;
signon "c:\sas\tcptso.scr";

rsubmit;
proc sql;
connect to db2 (ssid=db2p);
create view dwn_ld as select distinct * from
(select medrecn, id, mem_fn, mem_ln, mem_mi, sex, dob format=mmddyy8.,
mdy(month(dob),day(dob)-(month(dob)=2 and day(dob)=29),1997) as bday1 format=mmddyy8.,
mdy(month(dob),day(dob)-(month(dob)=2 and day(dob)=29),1998) as bday2 format=mmddyy8.,
eff_date format=mmddyy8., cnl_date format=mmddyy8.
from connection to db2
(select a.mem_med_record_id as medrecn,
        a.sub_id||a.mem_suffix_cd as id,
        a.mem_first_name as mem_fn,
        a.mem_last_name as mem_ln,
        a.mem_middle_initial as mem_mi,
        a.mem_sex_cd as sex,
        a.mem_birth_dt as dob,
        b.mem_status_effdt as eff_date,
        b.mem_status_cnldt as cnl_date
from    khp2apr.vk00023s a,
        khp2apr.vk00025s b
where   a.sub_id = b.sub_id and a.mem_suffix_cd = b.mem_suffix_cd and
      ((b.mem_status_cnldt is null  or b.mem_status_cnldt > '1996-12-31') and
        b.mem_status_effdt < '1999-01-01'))
where   year(dob)=1996 and
       (cnl_date>=(calculated bday1) or cnl_date=.) and
        eff_date<=(calculated bday2));

proc download data=dwn_ld out=rawmembs;
run;

%put sqlxrc=**&sqlxrc** sqlxmsg=**&sqlxmsg**;
endrsubmit;

signoff "c:\sas\tcptso.scr";


/*************************************************************/


proc sql;
create table mults as select distinct * from rawmembs
where medrecn in(select distinct medrecn from rawmembs group by medrecn
having (min(eff_date)<=(bday1+45) and (max(cnl_date)>=bday2 or cnl_date=.)))
order by medrecn, eff_date;


/*************************************************************/

data denom(drop=brk done edate cdate ledate days);
set mults;
by medrecn;
retain edate cdate ledate . brk done days 0;
format edate cdate ledate mmddyy8.;

/*single record per medrecn handled here*/
if (first.medrecn and last.medrecn and (cnl_date=. or cnl_date=>bday2)) then do;
       done=1; brk=0; end;

/*first of multiple records per medrecn handled here*/
else if first.medrecn then do;
```

SAS CODE APPENDIX

```
   /*is there an initial break?*/
     if eff_date>bday1 then do;
           brk=1; days=eff_date-bday1; end;
     else do;
           brk=0; days=0; end;

   /*does this record pass on its own?*/
     if (cnl_date=. or cnl_date=>bday2) then done=1;
     else do; edate=eff_date; cdate=cnl_date; ledate=eff_date; done=0; end; end;

/*additional of multiple records per medrecn handled here*/
else if done=0 then do;

         /*is there a break?*/
           if eff_date>cdate then do;
              brk+1; days=eff_date-cdate;

             /*is there multiple breaks or is break longer than 45 days?*/
               if (days>45 or brk>1) then done=3; end;

      if (done=0 and (cnl_date=. or cnl_date=>bday2)) then do;
          done=1; eff_date=edate; end;
      else if (done=0 and last.medrecn=0) then cdate=cnl_date; end;

/*final test*/
if done=1 then output;
run;
```