Generating Reliable Population Rates Using SAS® Software

Jack Shoemaker, Greensboro, NC

ABSTRACT

The business of health insurance has always been to manage medical costs so that they don't exceed premium revenue. Monitoring and knowing about these patient populations will mean the difference between success and financial ruin. At the core of this monitoring are population rates like per member per month costs and utilization per thousand. This paper describes techniques using SAS® software that will generate these population rates for an arbitrary set of population dimensions. Keeping the denominators in sync with the numerators is key for implementing trustworthy drill-down applications involving population rates.

1. DATA PREPARATION

The basic metric for population risk management is something expressed as a population rate. That is, a sum or count of some activity or financial quantity divided by the corresponding membership associated with that activity. A very broad metric is cost per member per month (PMPM) which is the total costs for a set of members divided by the number of member-months associated with those costs. For example, if I incur $240 in medical costs in calendar year 2016, then I represent $20 PMPM for the year. That is, $240 divided by 12 months.

1.1 MEMBERSHIP DATA

Note that the denominator in the example above is 12. This is so because I was a member for 12 months in calendar year 2016 so therefore represent 12 member-months. In population risk management, the denominators are almost always the product of members *over* a period of time rather than just a unary count of members. If you need to compute member-months as the fundamental denominator for population risk metrics, it follows that your membership system must allow you to compute member-months. The general form for the membership system is shown below.

```
MEMBER identifier
VECTOR of population attributes
START date for the attribute vector
END date for the attribute vector
```

Generalized membership data table

The population attributes for a particular member will change over time. That is, a particular member will have one or more rows in a membership table as shown above. The start and end dates describe a mutually exclusive and non-overlapping set of time spans describing the population attributes for a given member.

To look up or retrieve a particular set of attributes for a particular member on a particular activity date, you execute a where statement similar to this pseudo code:

```
where m.member = a.member and
  a.activity_date between ( m.start and m.end );
```

The conjugate above assumes that both start and end will have non-missing values. However, in an effective-dated record system as described in the generalized membership table, there will always be currently active records for some members. That is, records that represent the current set of population attributes and therefore do not have a true end date. In order to make the where clause easier to code and understand, it is best to set otherwise NULL end dates to some arbitrarily future date. One common convention is to set NULL end dates to ‘31DEC2999’D. This will likely ensure that your SAS programs function for the remainder of your lifetime and will give future Y3K consultants some work to do. Another solution is to set NULL end dates to 6589336. This magic value represents January 1, 20001 which is one day greater than what the SAS date formats can display. So, NULL end dates will appear as a set of asterisks when displayed. Consider the following code.

```
data _null_;
  x = 6589336;
  put x= date10.;
  run;

x=**********
```

The point is to set NULL end dates to some future date so that you can simply anywhere conjugates that you use to look up the appropriate membership record based on activity date.

The vector of population attributes is any set of columns that you use to describe the population. A good way to look at this vector is as a set of independent hierarchical dimensions. The table below proposes a set of five such dimensions that will likely cover most analytic needs.

```
PRODUCT: Line of business -> Product -> Benefit Plan
FUNDING: Sponsor -> Group -> Rate Tier
MARKETING: State -> Region -> County
MANAGEMENT: Network -> IPA -> PCP
AGE/SEX COHORT
```

Independent population dimensions

Although there may be similarities between certain levels across dimensions, it is good practice to think about the attributes as collections of hierarchical levels rather than try to get your entire organization to use similar definitions and lexicons. And, although the example above presents a balanced table with three levels within each hierarchy, there is no reason that it need be so. Having a ragged arrangement is fine. And, by establishing a method of bundling attributes into higher-level concepts you make adding and modifying these attributes relatively easy because you are touching and therefore affecting only a subset of the entire attribute vector.

The AGE/SEX cohort deserves some special notice. Strictly speaking, age will vary over the span of time described by the start and end dates. To the extent that age is often a determinant of premium level (funding), it is good practice to coordinate the definition of age to match that of the premium rules. Premium often arrives on a monthly basis. Furthermore, as the earlier discussion foreshadowed, a fundamental quantity for the denominators is member-months. For these reasons, it is often a good idea to transform the generalized membership table from above to a table where the unit of observation is a member month. And, in doing so compute the age at some fixed point in the month like the first, fifteenth, or last day of the month. Experience has shown that using the last day of the month for age calculations produces the most stable results; however, the choice is somewhat arbitrary. What is import is that you pick a day for some good reason and stick with it. Here is how a typical transformation may look.

Generalized (original) table

MEMBER	DOB	START	END	LOB	PRODUCT	ETC…
1234	09/10/60	01/01/12	12/31/12	COM	PPO	

Transformed (rotated) table

MEMBER	AGE	START	END	LOB	PRODUCT	ETC…
1234	51	01/01/12	01/31/12	COM	PPO	
1234	51	02/01/12	02/29/12	COM	PPO	
1234	51	03/01/12	03/31/12	COM	PPO	
1234	51	04/01/12	04/30/12	COM	PPO	
1234	51	05/01/12	05/31/12	COM	PPO	
1234	51	06/01/12	06/30/12	COM	PPO	
1234	51	07/01/12	07/31/12	COM	PPO	
1234	51	08/01/12	07/31/12	COM	PPO	
1234	52	09/01/12	09/30/12	COM	PPO	
1234	52	10/01/12	10/31/12	COM	PPO	
1234	52	11/01/12	11/30/12	COM	PPO	
1234	52	12/01/12	12/31/12	COM	PPO	

Comparison of generalized and transformed membership data table

To be sure, the rotated table will require more disk storage; however, that additional storage requirement is a small price to pay for greater computational ease. By using the rotated membership table, computing member-months is a simple matter of counting the rows. For example, the following pseudo code will provide a count of member-months by month, LOB, PRODUCT, and age grouping as defined by the user-defined format called ‘age’.

```
proc format;
  value age
    low - 50 = 'young'
    51 - high = 'old';
  run;

proc summary data = rotated missing;
  format age age5.;
  class end lob product age;
  output out = reduced( rename = ( _freq_ = MemberMonths ) );
  run;
```

1.2 ACTIVITY DATA

The numerators for the population rates come from the activity data which consists of institutional and professional medical claims, encounters with allied professionals, pharmacy prescriptions, durable-medical equipment (DME), lab-test results, and another other available administrative data that describes and records the delivery of a healthcare service or supply. The generalized form of the activity data is shown below.

```
WHO
    Member
    Provider
WHAT
    Type of service
    Procedure code
    Revenue code
    DRG
    NDC code
    LOINC code
WHY
    Diagnosis code
    Admission reason
WHERE
    Place of service
WHEN
    Activity date (date of service, admission date, fill date)
    Paid date
HOW MUCH
    Quantity of service
    Charge for service
    Paid for service
    Test result
```

Generalized activity data table

The activity data are used to construct the numerators for the population rates of interest. Some combination of the 'what', 'why', and 'where' attributes will define a service category of interest. For example, professional office visits may be defined as any professional claim with a certain set of CPT-4 codes in the procedure code field. For these service categories, the numeric quantities in the how much attributes are summarized. And, finally, based on the member and activity date, the associated population attributes are brought to bear so that these quantities may be summarized on the population dimensions as well.

2. COMPUTING POPULATION RATES

Coordinating the relevant dimensions during these summarizations is essential for computing accurate and credible rates. The use of macro symbols to hold the list of classification attributes makes this task much easier. For example, the population attributes from the membership data section above may be captured to a macro variable called POPVARS as follows:

```
%let POPVARS = age end lob product;
```

Now let's assume that the categories of interest for the activity data are type of service (TOS) and procedure code (PROCCODE). We could then create a new macro variable called ACTVARS which will store all of the classification variables for the activity data.

```
%let ACTVARS = &POPVARS. tos proccode;
```

Now the calls to PROC SUMMARY can reference these macro symbols which will guarantee that the classification variables and therefore the levels of summation are in a known and consistent manner. For example:

```
proc summary data = rotated missing;
  format age age5.;
  class &POPVARS.;
  output out = denom(
    rename = ( _type_ = poptype _freq_ = MemberMonths ) );
  run;

proc summary data = activity missing;
  format age age5.;
  class &ACTVARS.;
  var paid;
  output out = numer( rename = ( _freq_ = ActivityLines ) );
  run;
```

Note that the automatic _type_ variable in the DENOM data set has been renamed to poptype. This allows us to create a poptype variable from the NUMER data set so that we can use poptype as a MERGE variable to coordinate and align the numerators and denominators. Since we know the exact order of the classification dimensions because we employed macro symbols to generate the list of column names, we can use the BRSHIFT() function to create the associated poptype value based on the _type_ value in the NUMER data set. The rule is to right shift the _type_ value the number of classification variables in &ACTVARS not in &POPVARS. In this case, &ACTVARS has two additional classification variables, so we need to right shift by two places. For example,

```
data numer;
  set numer;
  poptype = brshift( _type_, 2 );
  run;
```

Now we can merge the NUMER and DENOM data sets to match the proper denominators and numerators as follows:

```
data rates;
  merge numer denom;
  by &POPVARS. poptype;
  PMPM = paid / MemberMonths;
  run;
```

This works because _type_ is a bit mask indicating the contribution to any summarization level. Consider the level of highest interaction in the NUMER data set. Because there are 6 classification variables, the value of _type_ will be 63, or ‘111111’B. Right shifting this value by 2 bits, the value of poptype becomes ‘001111’B or 15 which is the value associated with the highest level of interaction in the DENOM data set. Note that ‘111100’B (60), ‘111110’B (62), and ‘111101’B (61) will also result in poptype values of ‘001111’B (15). And, this is exactly what is desired.

You can further generalize the pseudo code above by computing the value of ‘2’ from the variable lists &POPVARS and &ACTVARS; however that is a topic best left for papers on macro coding techniques. For our purposes, the data set called RATES is what we want. It is the data set that we can use for analysis or input into a PROC OLAP application. Presumably the population and activity dimensions are exactly what you would want for OLAP analysis. Or put another way, you should construct your lists of the population and activity variables with this in mind.

If you find that the preceding discussion about shifting bits on a _type_ variable and merging together a NUMER and DENOM data set, then a third-party product like Futrix Health may be what you want instead. Futrix takes the membership and activity data sets as described as input and does much of this work and more under the covers. The results are fully and accurately computed population rates with drill-anywhere functionality.

2.1 COUNTING VISITS AND ADMISSIONS

Although the pseudo code above works well for a quantity like paid that sums nicely across rows in the activity data, counting other quantities like office visits and admissions requires more thought and consideration. The normal unit of observation in the activity data for a professional claim is a line item which is a record of a specific procedure code or *adjustment* to that procedure code. So, simple rule that just counts line items with certain E&M CPT-4 codes will certainly over count visits because there may be multiple line items associated with a particular visit. For analysis purposes it almost always a good idea to “net out” the adjustment lines so that you are left with the final adjudicated

version of the claim line. The degree t which you can successfully do this depends a great deal on how the native transaction system presents adjustments. Even if you net out adjustments, there still may be multiple line items per claim which can result in over counting visits. If visits to a doctor is a meaningful and useful measure for your analysis, you are well advised to build a separate table for the purposes of counting visits. This visit table should be constructed such that you have one row per visit defining visit in whatever way makes sense for your organization. One common, though by no means foolproof, way to count visits is to define a visit as a unique combination of provide, member, and date of service. The advantage of this approach is that it is fairly easy to code and understand.

```
proc sort data = prof;
  by member provider DOS;
  run;

data visits( keep = member provider DOS);
  set prof;
  by member provider DOS;
  if first.DOS;
  run;
```

Counting hospital admissions or discharges is an even greater challenge because in addition to multiple line items per admission and adjustments, it is common for hospitals to generate interim claims for long lengths of stay. Notwithstanding, as with the discussion on counting visits above, it is good practice to create a table for which the unit of observation is a single confinement. A useful confinement detail table will contain the following information.

```
MEMBER identifier
HOSPITAL identifier
ADMISSION date
DISCHARGE date
LENGTH OF STAY – actual, paid, and authorized
DISCHARGE status
PAID amount
SOURCE claim identifiers
```

Confinement detail table

There are likely more attributes that you will find useful. The point is to bake all the rules and exceptions into a single process that creates a simplified table for analysis. For example, using the table above admissions and discharges are just row counts and patient-days is just a sum of the length-of-stay.

3. VARIANCE ANALYSIS

The objective of population risk management is to monitor how you are doing against some financial target and then manage healthcare delivery to meet this target. A risk-bearing entity has two tools at their disposal – lowering unit cost through price negotiation and lowering utilization through medical management and adoption of best clinical practices. It helps to know which lever to press and why. This is where variance analysis can help.

Consider an entity with a budget of $182.50 PMPM broken down as follows:

TYPE OF SERVICE	UTILIZATION	UNIT COST	PMPM
Professional	2,500 Visits/1000	200 per Visit	41.67
Outpatient	500 Procs/1000	2,000 per Proc	83.33
Inpatient	60 Admits/1000	11,500 per Admit	57.50
			182.50

Hypothetical budget

Now consider the experience of a member of this risk-bearing entity, say a primary care physician or clinic:

TYPE OF SERVICE	UTILIZATION	UNIT COST	PMPM
Professional	2,700 Visits/1000	180 per Visit	40.50
Outpatient	400 Procs/1000	2,500 per Proc	83.33
Inpatient	50 Admits/1000	13,000 per Admit	54.17
			178.00

Actual results

So, everything looks swell because these actual results are 4.50 under budget overall. But this aggregate analysis misses a more telling story. The principle of variance analysis is to decompose overall variance into that caused by

differences in utilization and into that caused by differences in unit price. This is accomplished through a two-step process. First, the difference in utilization is applied to the budgeted unit cost. Then the difference in unit cost is applied to the actual utilization. For our example above, here is what the variance analysis reveals. Figures in parenthesis are negative (bad) variances against budget.

TYPE OF SERVICE	UTILIZATION	UNIT COST	TOTAL
Professional	(3.33)	4.50	1.17
Outpatient	16.67	(16.67)	0.00
Inpatient	9.58	(6.25)	3.33
	22.92	(18.42)	4.50

Variance decomposition

From this we see that utilization of professional services is costing $3.33 more PMPM than budgeted, but this is offset by a favorable $4.50 on unit cost. On the other hand, utilization is under budget for both outpatient and inpatient, but is mostly offset by negative variances on unit cost. In other words, if the unit costs for outpatient and inpatient could be brought more in line with budget, this group would be going very well. Or put another way, if the favorable variance on utilization is only a blip or the result of lower-than-expected patient acuity, this entity could turn into negative territory very quickly.

CONTACT INFORMATION

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

Jack Shoemaker
Greensboro, NC 27455
(336) 202-2165

jack.shoemaker@gmail.com