Some SAS techniques used in Health Care Data

Wei Yu

Health care data are usually large data sets and the distribution is skewed. From insurance side, we need to summarize data at policy and Claim levels. But for some statistics, we need to get the data from invoice and line payment levels. Under this circumstance, SQL and some data steps techniques are very useful to join the tables and manipulate the large data sets. For skewed data, median rather than mean is widely used to describe the population.

1) How to calculate the median

A. Data step method

Suppose we have a data set called aa.sas. The example records are as follows:

Table 1

Policy	ClaimNo	Medicalcost
99E00123	00123	100
99E00123	00256	150
99E00123	00369	230
99G00345	00251	1,000
99G00345	00345	2,050
99G00345	00362	100,000
99G00345	00397	250
99G00345	00412	5,600
99G00345	00452	7,700

```
proc means data=aa;
var medicalcost;
by policy;
output out=stat1(keep=policy mediancost ) Median=mediancost;
run;
```

Sometimes if we have large datasets and we use SQL to do a lot of work, then we may want to use SQL to get the median. Below are the steps that we want to calculate the median.

B. SQL method

Before we conduct the SQL method, we need to use data step to get a row number. We still use the data set aa.sas.

The first step is to sort the data. We can use SQL or data step to sort data.

```
proc sort data=paperdata out=aa;
by policy medicalcost;
run;
```

The second step is to get a row number for each record based on each policy.

```
data aa1;
set aa;
by policy;
```

```
if first.policy then seq=1;
else seq+1;
run;
```

After that, we got table 2:

Policy	ClaimNo	Medicalcost	seq
99E00123	123	100	1
99E00123	256	150	2
99E00123	369	230	3
99G00345	397	250	1
99G00345	251	1,000	2
99G00345	345	2,050	3
99G00345	412	5,600	4
99G00345	452	7,700	5
99G00345	362	100,000	6

Now we can write SQL code to get the median. For clear illustration, we will do it step by step.

/* get the records for calculating median first for each policy*/

```
proc sql;
create table aa2 as
select policy, max(seq+1)/2-0.5 as Lo , max(seq+1)/2+0.5 as Hi
from aa1
group by policy;quit;
/* join the original table aa1 to aa2*/
proc sql;
create table aa3 as
select a. * , b.lo, b.hi
from aa1 a left join aa2 b
on a.policy=b.policy; quit;
/* get the median*/
proc sql;
create table aa4 as
select policy, sum(medicalcost)/count(policy) as medianCost
from aa3
where seq between lo and hi
group by policy;  quit;
```

The result is as follows:

Policy	medianCost
99E00123	150
99G00345	3,825

2) <u>How to handle the medical payments from original invoices and appeal invoices at the same time</u>

In health care insurance, if providers submit medical bills, they may not get fully reimbursement at once. Later they will submit the appeal invoices. Some people from claim side in insurance company like to use

the saving ratio (paid/charged) to measure the performance. If so, we want to sum up the original and appeal payment and only use one original charged amount.

The following SQL codes explain how to join multiple tables at once. There are one original file and one appeal file.

/*esinvc: Invoice number; esappr: parent invoice number; esapch: child invoice number*/

```
proc sql;
create table bb as
select a.*,  b.espaid 'Espaid1' as espaid1, c.espaid 'espaid2' as
espaid2, d.espaid 'espaid3' as espaid3,
e.espaid 'espaid4' as espaid4, f.espaid 'espaid5' as espaid5, g.espaid
'espaid6' as espaid6,
h.espaid 'espaid7' as espaid7, i.espaid 'espaid8' as espaid8, j.espaid
'espaid9' as espaid9,
j.esapch 'esapch9' as esapch9
from original a left join appeal b on a.esinvc =b.esappr

                left join appeal c on b.esapch=c.esinvc
                left join appeal d on c.esapch=d.esinvc
                left join appeal e on d.esapch=e.esinvc
                left join appeal f on e.esapch=f.esinvc
                left join appeal g on f.esapch=g.esinvc
                left join appeal h on g.esapch=h.esinvc
                left join appeal i on h.esapch=i.esinvc
                left join appeal j on i.esapch=j.esinvc

; quit;
```

3) Sub query techniques

You can get one summary table at once using sub query techniques and do not generate many temporary tables.

I will use sub queries to get the median this time.

If we use table 2 and the sub query techniques below, we can get the table cc .

```
proc sql;
create table cc as
select aa1.* from
      (select policy, (max(seq+1)/2-0.5) as Lo , max(seq+1)/2+0.5 as Hi
       from aa1
       group by policy
      ) as temp
left join aa1
on  aa1.policy=temp.policy  and  aa1.seq  >=temp.lo  and  aa1.seq<=temp.hi;
quit;
```

Table 3: (data cc)

Policy	ClaimNo	Medicalcost	seq
99E00123	256	150	2
99G00345	345	2050	3
99G00345	412	5600	4

For policy 99E00123, the median is 150; for policy 99G00345, the median is (2050+5600)/2=3825;

If we add the summary function into the SQL codes above, we can get the median at once. In other words, table aa4 at page 2 can finish in one step if we use sub queries.

```
proc sql;
create table dd as
select policy, sum(medicalcost)/count(policy) as mediancost
from (
select aa1.* from
      (select policy, (max(seq+1)/2-0.5) as Lo , max(seq+1)/2+0.5 as Hi
       from aa1
       group by policy
      ) as temp
left join aa1
on aa1.policy=temp.policy and aa1.seq >=temp.lo and aa1.seq<=temp.hi )
group by policy; quit;
```

Table 4 (data dd)

Policy	median
99E00123	150
99G00345	3,825

Another example for Sub query techniques.

Suppose I have a written policy data set called policy.sas; I have another data set which contains claim payments called claim.sas. Now I want to get all the written policy after 1999 and get the sum of loss for each claim at policy level.

```
Proc sql;
Create table EE as
Select a.pm_policy, a.pm_efdte, totalloss
 from
 (select pm_policy, pm_efdte
 from policy
 where round(pm_efdte/10000,1) >=2000) as a
 left join

 (select pm_policy, pm_efdte, sum(cm_incrtot) as totalloss
 from claim
 where round(pm_efdte/10000,1) >=2000
 group by pm_policy, pm_efdte) as b
 on a.pm_policy=b.pm_policy and a.pm_efdte=b.pm_efdte; quit;
```