Paper GH-13

Assign Overpayment to Insurance Data with Adjustments

Qiling Shi, NCI Information Systems, Inc., Nashville, Tennessee

ABSTRACT

The purpose of this study is to assign overpayment to insurance data with adjustments using SAS®. Suppose we have an algorithm to calculate the overpayment for each insurance ID, we need to join back with the original data to get the adjustments and assign the overpayments to the insurance claims with adjustments. Since the overpayments are calculated by rolling up the original dataset, we need to join back to get the adjustments by the same insurance ID. There are some rules we need to consider to assign the values of overpayments after this join back. A SAS macro with procedures such as PROC SQL and DATA STEPS is employed to do the data analysis.

INTRODUCTION

An insurance Claim is filed by a policyholder stating that an insured event has occurred and that the insurance company should provide coverage. An insurance adjuster looks at the parts of the claim to compute the right payment amount.

Insurance adjustment may be very complicated. For example, claims that were rejected as duplicates may be adjusted, cancelled or resubmitted. Claims rejected for eligibility or other billing errors may be adjusted when the eligibility or billing issue is resolved. At different times, billing transactions that have already been processed may be paid or repaid with a duplicate payment.

The Division of Medicaid and the fiscal agent allow adjusting and voiding of claims. If a paid claim is being adjusted, the Provider Identification Number and the Recipient cannot be changed. The voided amount originally paid will appear as a negative amount and that amount will be deducted from payments until the overpayment is recovered.

Usually we identify the overpayments for each insurance ID by different models or various definitions of rule violations. In most cases, we need to roll up the original data by the insurance ID and then calculate the overpayments. Hence we get a tuple with the information for overpayments according to each insurance ID. But we need some other information besides overpayments such as adjustments, dates of services and service codes etc. In order to get this kind of information, for example the adjustments, we will join the overpayment tuple back with the original data. To avoid the inflation of the overpayments, we will assign the overpayments to the original claims which have adjustment indicators equaling zero or to the first claims with positive adjustments if there are no original claims found or the original claims have been voided. For the other claims, we will assign the overpayments as missing.

PROGRAM CODE

```
*************************************************************************;
*         1. Join back with the original data to get the adjustments
;
*         2. When the adjustments in (0,2,4) and not the first id
;
*            by sorting, assign the overpay amount. Otherwise assign
;
*            the missing value as overpay.
;
*************************************************************************;

%macro joinadj(old_data, new_data, keepvar, dataout);

*Join back with the original data to get the adjustments;
proc sql;
    create table join_data as
    select a.*, &keepvar.
    from &old_data a, &new_data. b
    where a.id = b.id;
quit;
```

```
*Create a new sorting ID: temp_indicator;
data new_sort;
    set join_data;
    if adjustment_indicator in (0,2,4) then temp_indicator =0;
        else if adjustment_indicator =3 then temp_indicator =1;
run;

*Sort the data by id, temp_indicator adjustment_indicator;
proc sort data=new_sort;
    by id temp_indicator adjustment_indicator;
run;

*Assign the overpay as missing to non-first IDs by the sorting order;
data &dataout.;
    set new_sort;
    by id temp_indicator adjustment_indicator;
    if not first.id then overpay=.;
run;

%mend joinadj;

*Read in the insurance data;
data insurance;
  infile "C:\Documents and Settings\shiq\Desktop\New Folder\NESUG\joinadj.txt";
  input id $6. amount_charged  adjustment_indicator;
run;

* Calculate the overpayments;
proc sql;
    create table flag as
    select  id, sum(amount_charged) as overpay
    from insurance
    group by id
    having overpay >  0;
quit;

*use the macro to get back the adjustments and assign overpay;
options symbolgen mprint mlogic;
%joinadj(insurance, flag, overpay, new);
```

RESULTS

The demonstrated data set “INSURANCE” has 18 observations and 3 variables. One variable is called “ID” which represents the insurance claim identifications. “AMOUNT_CHARGED” means the amount a provider charged on the health care insurance agents or companies. “ADJUSTMENT_INDICATOR” provides codes for different adjustment activities. Here the code “0” represents the original claims. The codes “2” and “4” mean claims with positive partial adjustments. The code “3” indicates the claims with negative partial adjustments.

Table 1: The Dataset “INSURANCE” used as an example.

	id	amount_charged	adjustment_indicator
1	100001	60	0
2	100001	-60	3
3	100001	60	2
4	100002	20	2
5	100002	30	4
6	100002	-20	3
7	100002	20	4
8	100003	10	0
9	100004	30	0
10	100004	-30	3
11	100005	40	0
12	100006	80	0
13	100006	-50	3
14	100006	35	4
15	100006	-35	3
16	100007	67	2
17	100008	70	0
18	100008	-70	3

For the same claim ID, there are different records with different adjustments. We want to find out the overpayments for each insurance ID.

The following is the table derived from “INSURANCE” containing the overpayment for the corresponding insurance ID. Here we calculate the overpayments as the summation of “AMOUNT_CHARGED” grouped by “ID”.

Table 2: Dataset “FLAG” used to check voided claims for the same claim IDs.

	id	overpay
1	100001	60
2	100002	50
3	100003	10
4	100005	40
5	100006	30
6	100007	67

We only keep the insurance ID with positive overpayments. That means removing those insurance IDs which do not have overpayments or the overpayments are negative. There are 6 distinct insurance IDs having positive overpayments identified.

Table 3: Dataset “JOIN_DATA”.

	id	amount_charged	adjustment_indicator	overpay
1	100001	60	0	60
2	100001	-60	3	60
3	100001	60	2	60
4	100002	20	2	50
5	100002	30	4	50
6	100002	-20	3	50
7	100002	20	4	50
8	100003	10	0	10
9	100005	40	0	40
10	100006	80	0	30
11	100006	-50	3	30
12	100006	35	4	30
13	100006	-35	3	30
14	100007	67	2	67

From table 3, we can get all the claims with positive overpayments after join the table “FLAG” back with the table “INSURANCE” by “ID”. But in table 3, the amount of overpayments has been inflated. Originally we only have $257 overpayments, but now we have $617 overpayments. The problem is that we assign the overpayments to all the claims. For example, for insurance ID “100001”, the overpayment $60 has been assigned three times to different adjustments. To avoid this problem, we should only assign the overpayments to the original claims with adjustment indicator equaling zero. If there is no original claim for this insurance claim ID, we will assign the overpayments to the claims with positive adjustments.

Table 4: Dataset “NEW_SORT”.

	id	amount_charged	adjustment_indicator	overpay	temp_indicator
1	100001	60	0	60	0
2	100001	60	2	60	0
3	100001	-60	3	60	1
4	100002	20	2	50	0
5	100002	30	4	50	0
6	100002	20	4	50	0
7	100002	-20	3	50	1
8	100003	10	0	10	0
9	100005	40	0	40	0
10	100006	80	0	30	0
11	100006	35	4	30	0
12	100006	-50	3	30	1
13	100006	-35	3	30	1
14	100007	67	2	67	0

We create a new sorting ID called “TEMP_INDICATOR”. If ADJUSTMENT_INDICATOR equals 0, 2 or 4, then TEMP_INDICATOR =0. Otherwise if ADJUSTMENT_INDICATOR equals 3 then TEMP_INDICATOR =1. Then first we sort the dataset by “ID”. Within each ID group, sort the dataset by “TEMP_INDICATOR” in an ascending order. So it will put claims with ADJUSTMENT_INDICATOR equals 0, 2 or 4 before the claims with ADJUSTMENT_INDICATOR equals 3. Within the group with ADJUSTMENT_INDICATOR equals 0, 2 or 4, sort the dataset by ADJUSTMENT_INDICATOR in an ascending order. So the first claim should have ADJUSTMENT_INDICATOR equals 0. If there is no original claim for a single insurance ID, the first claim should have ADJUSTMENT_INDICATOR equals 2. For example, for insurance ID “100002” and “100007”, the first claim has ADJUSTMENT_INDICATOR equals 2.

Table 5: Dataset “NEW”.

	id	amount_charged	adjustment_indicator	overpay
1	100001	60	0	60
2	100001	60	2	.
3	100001	-60	3	.
4	100002	20	2	50
5	100002	30	4	.
6	100002	20	4	.
7	100002	-20	3	.
8	100003	10	0	10
9	100005	40	0	40
10	100006	80	0	30
11	100006	35	4	.
12	100006	-50	3	.
13	100006	-35	3	.
14	100007	67	2	67

From table 5, we know that we assign the overpayments only to the original claims. If there is no original claim for the insurance ID, for example, for insurance ID “100002” and “100007”, we assign the overpayments to the claims with ADJUSTMENT_INDICATOR equaling 2. If there is no claim with ADJUSTMENT_INDICATOR equals 2, the first claim should have ADJUSTMENT_INDICATOR equals 4. The total overpayment is $257 which is exactly as we calculated for the dataset “FLAG”.

DISCLAIMER

All opinions and suggestions stated in this paper do not necessarily reflect the opinions and suggestions of NCI Information Systems, Inc.

REFERENCES

1. “Detecting Medicaid Data Anomalies Using Data Mining Techniques”, Southeast SAS Users Group Conference, 2010.
2. “Find Potential Fraud Leads Using Data Mining Techniques”, Southeast SAS Users Group Conference, 2011.
3. “Remove Voided Claims for Insurance Data”, Western Users of SAS Software Conference, 2011.

CONTACT INFORMATION

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

Qiling Shi, Mathematics PhD, Certified Fraud Examiner
NCI Information Systems, Inc.
2636 Elm Hill Pike, Suite 115
Nashville, TN, 37214
Email: shiq@nciinc.com Email: shiqiling@gmail.com
Website: http://www.linkedin.com/in/qilingshi