

Mining Medicaid Encounter Data: What, Why and How?

Wei Jiang, Ph.D., Mercer Human Resource Consulting, Atlanta, GA

ABSTRACT

State Medicaid agencies are increasingly using the encounter model to control Medicaid budgets. This introduces such complexities as numerous payment and system errors in exact payment duplication, partial payments and denials by exception codes and dispositions. This paper explores the utilization of Base SAS® and PROC SQL in mining raw, multi-million-observation, error-ridden encounter data into precise data for capitated rate setting in actuarial science. Mining procedures include creating a switch and a unique group number for each encounter with both header and line payment and assigning flags to delete duplicate and denied claims.

INTRODUCTION

During the past decade or so, more and more State Medicaid agencies switched from fee for service (FFS) to managed care, where Medicaid beneficiaries receive care through health maintenance organizations (HMOs) or managed care organizations (MCOs) to improve care access and slow expenditure growth. Services by HMOs or MCOs are recorded in what is called encounter data, which is usually in large size and includes various non-service related data. To be in compliance with the Health Insurance Portability and Accountability Act of 1996 (HIPAA), the MCO internal control number or the variable MCO_ICN displayed in the paper has been encrypted. Please also note that all the examples in this paper are *only* a trivial portion of the data set used. Hundreds of other variables including exception codes and dispositions 2 -19, place of service, header first date of payment, and MCO payment amount are not included in the examples throughout this paper due to space limit.

ISSUES IN ENCOUNTER DATA CITED

Each encounter consists of a header and detailed lines. Let us look at Example 1 that shows an encounter with a header marked as "00" under the variable line_number and a detailed line or "01":

Example 1

MCO_ICN	CLAIM_STATUS	LINE_NUMBER	EXC_CD_1	EXC_DIS_1	...	EXC_CD_20	EXC_DIS_20
DF58734	N	00	101	5	...	325	5
DF58734	N	01	423	5	...	265	5

Despite the header and detailed lines available in each encounter, many encounters vary in the number of their detailed lines and, therefore, result in the irregularity of the structure of this encounter data. Unlike Example 1 above, Example 2 illustrates five detailed lines that all belong to a single encounter:

Example 2

MCO_ICN	CLAIM_STATUS	LINE_NUMBER	EXC_CD_1	EXC_DIS_1	...	EXC_CD_20	EXC_DIS_20
DF58734	N	00	427	5	...	203	2
DF58734	N	01	101	5	...	364	5
DF58734	N	02	290	5	...	423	5
DF58734	N	03	290	5	...	423	5
DF58734	N	04	101	5	...	423	5
DF58734	N	05	290	5	...	423	5

Consequently, this difference or inconsistency in line numbers presents a challenge as to how each encounter can be grouped and summarized for MCO and other payments that need to be calculated for Medicaid rate setting, which is the purpose of mining this data set.

APPROACHES TO PROBLEM SOLVING

Along with the different number of detailed lines in each encounter, in general there are two other issues in the encounter data set discussed in the paper. They are the existence of duplicate claims and that of denied claims. Those duplicate and denied claims need to be deleted – at the header level or the line level or even both. Compared

with FFS data, it is more difficult to deal with encounter data where health care costs may not be specifically or directly related to the services provided to Medicaid beneficiaries.

Switch and Group Number

In order to solve all the issues listed above in this type of encounter data and perform queries such as PROC SUMMARY for each encounter at a later time, it is critical that each encounter – both the header and its associated lines - must be assigned to a unique group number, no matter how many lines an encounter has. Each encounter is considered a group. A switch that will distinguish the header from detailed lines for each encounter needs to be created first. This switch will make the assignment of a unique group number to each encounter possible.

```

OPTIONS symbolgen mprint mlogic;

%GLOBAL lib num lnum tnum cnt swt gnum x cde y z;
%LET lib=encount;
%LET num=NUMBER;
%LET lnum=line_&n;
%LET tnum=TEMP_&n;
%LET cnt=counter;
%LET swt=switch;
%LET gnum=GROUP_&num;
%LET x=exception_code_;
%LET cde="101";
%LET y=(&x.1 = &cde or &x.2 = &cde or &x.3 = &cde or &x.4 = &cde or
&x.5 = &cde or &x.6 = &cde or &x.7 = &cde or &x.8 = &cde or &x.9
= &cde or &x.10 = &cde or &x.11 = &cde or &x.12 = &cde or &x.13 =
&cde or &x.14 = &cde or &x.15 = &cde or &x.16 = &cde or &x.17 =
&cde or &x.18 = &cde or &x.19 = &cde or &x.20 = &cde);
%LET z=exception_disposition_;

%*set up a switch where "0" for lines and "1" for the header;
%MACRO header;
    %LOCAL n;
    %LET n=1;
    DATA &lib..file_&n;
        SET &lib..file;
        if &lnum = "00" then &tnum = 1;
            else &tnum = 0;
    RUN;
%MEND header;

%header

```

This way, the variable temp_number that assigns the number “1” to the header and “0” to all detailed lines of each encounter will be created:

Example 3

MCO_ICN	CLAIM_STATUS	LINE_NUM BER	EXC_CD _1	EXC_DIS _1	...	EXC_CD_ 20	EXC_DIS_ 20	TEMP_NUM BER
DF58734	N	00	101	5	...	325	5	1
DF58734	N	01	423	5	...	265	5	0

The number “1” will serve as the switch for the creation of an exclusive encounter/group number as follows:

```

%*set up a group number for each encounter;
%MACRO setgnum (n=2);
    DATA &lib..file_&n;
        SET &lib..file_1;
        retain &swt;

```

```

        if &tnum then do;
            &swt = 1;
            &cnt + 1;
        end;
        if &swt then &gnum = &cnt;
        drop &swt;

    RUN;
%MEND setgnum;

%setgnum

```

Two new variables TEMP_NUMBER and GROUP_NUMBER are thus created:

Example 4

MCO_ICN	CLAIM_	LINE_NU	EXC_	EXC_	...	EXC_	EXC_DIS	TEMP_NU	GROUP_
	STATU	MBER	CD_1	DIS_1		CD_20	_20	MBER	NUMBE
	S								R
DF58697	N	00	427	5	...	203	2	1	6
DF58697	N	01	101	5	...	364	5	0	6
DF58697	N	02	290	5	...	423	5	0	6
DF58697	N	03	290	5	...	423	5	0	6
DF58697	N	04	101	5	...	423	5	0	6
DF58697	N	05	290	5	...	423	5	0	6

Duplicate Claims Deletion

Many encounters have an exact duplicate claim known as the exception code "101" in the variables EXC_CD_1 through EXC_CD_20. This exception code should have been excluded in the data set:

Example 5

MCO_ICN	CLAIM_	LINE_NUM	EXC_	EXC_	...	EXC_C	EXC_DIS_	TEMP_NUM	GROUP_
	STATUS	BER	CD_1	DIS_1		D_20	20	BER	NUMBER
DF58652	N	00	101	5	...			1	2
DF58652	N	01	290	5	...	101	5	0	2
DF58652	N	02	423	5	...	101	5	0	2

In Example 5 above, the duplicate claim appears in both of the lines numbers "01" and "02" of this encounter. The deletion logic is that if all lines are duplicates ("101"), they must be deleted, along with their header. The codes below will serve this purpose:

```

%*keep line number "00" (header) not associated with exception code
"101" to bring back later;
%MACRO h_keeper;
    %local n2 n3;
    %LET n2=2;
    %LET n3=keep;
    PROC SQL;
        CREATE table &lib..file_&n3 as
        SELECT *
        FROM &lib..file_&n2
        GROUP by &gnum
        HAVING count(&gnum)=1;
    QUIT;
%MEND h_keeper;

```

%h_keeper

In addition to the above circumstance, there are two more situations in which the duplicate claim "101" must be removed. Here is the logic: The claim is an exact duplicate if one of the 20 exception codes in the header is equal to "101". In this case, the whole claim has to be deleted:

Example 6

MCO_ICN	CLAIM_S TATUS	LINE_NUM BER	EXC _CD_ 1	EXC_ DIS_1	...	EXC_C D_20	EXC_DIS_ 20	TEMP_NUM BER	GROUP_ NUMBER
DF58933	N	00	427	5	...	101	2	1	9
DF58933	N	01	423	5	...	364	5	0	9
DF58933	N	02	290	5	...	423	5	0	9
DF58933	N	03	290	5	...	423	5	0	9

The removal can be performed using the following:

```
%*delete detailed lines with exception code "101";
%MACRO l_deletion (n3, n4);
    DATA &lib..file_&n4 (drop=i);
        SET &lib..file_&n3;
        array exc(20)&x.1 - &x.20;
        do i = 1 to 20;
            if exc(i) = &cde then delete;
        end;
    RUN;
%MEND l_deletion;

%l_deletion (keep,4)

%*keep the original file as record;
%MACRO fileforhd (n4, n5, n6);
    DATA &lib..file_&n5 &lib..file_&n6;
        SET &lib..file_&n4;
    RUN;
%MEND fileforhd;

%fileforhd (4,5,6)

%*delete rows where count(group_number)=1;
%MACRO fileforhd (n5, n6);
    PROC SQL;
        DELETE *
        FROM &lib..file_&n5
        WHERE &gnum in
            (SELECT &gnum
             FROM &lib..file_&n6
             GROUP by &gnum
             HAVING COUNT(&gnum) = 1);
    QUIT;
%MEND fileforhd;

%fileforhd (5,6)
```

Elimination of duplicate claims in the other situation is relatively easy. The line is an exact duplicate if one of the 20 exception codes is equal to "101". In this case, the line such as "02" in Example 7 below has to be deleted while the header and the other line "01" will remain unchanged:

Example 7

MCO_ICN	CLAIM_STATUS	LINE_NUMBER	EXC_CD_1	EXC_DIS_1	...	EXC_CD_20	EXC_DIS_20	TEMP_NUMBER	GROUP_NUMBER
DF58673	N	00	427	5	...			1	5
DF58673	N	01	423	5	...	364	5	0	5
DF58673	N	02	101	2	...	423	5	0	5

Deletion can be carried out using an array function similar to the one in Example 6.

Flag Assignment

A large number of encounters possess the procedure and/or revenue codes that should have been excluded, for example, "431" (Procedure/Revenue code not covered) and "439" (Procedure code not allowed for service date):

Example 8

MCO_ICN	CLAIM_STATUS	LINE_NUMBER	EXC_CD_1	EXC_DIS_1	...	EXC_CD_20	EXC_DIS_20	TEMP_NUMBER	GROUP_NUMBER
5S45639	N	00	427	5	...			1	227
5S45639	N	01	439	2	...	265	5	0	227
5S45639	N	02	290	5	...	431	2	0	227
5S45639	N	03	290	5	...	364	5	0	227
5S45639	N	04	290	5	...	101	2	0	227

Flag "0" needs to be allocated to the claims denied only by where a) any of the 20 exception codes is equal to "431" and/or "439"; and b) exception disposition (1-20) = "2" (Claims are denied). Flag "2" should be assigned to the claims denied by other exceptions codes, for instance, "445" (1st diagnosis code not covered):

Example 9

MCO_ICN	CLAIM_STATUS	LINE_NUMBER	EXC_CD_1	EXC_DIS_1	...	EXC_CD_20	EXC_DIS_20	TEMP_NUMBER	GROUP_NUMBER
7A38773	N	00	431	2	...	364	5	1	11119
7A38773	N	01	423	5	...	423	5	0	11119
7A38773	N	02	424	5	...	445	2	0	11119

The remaining encounters will be specified with Flag "P" for Paid status, if their exception disposition is not equal to "2" or denial. Here are the codes for wiping out those procedure and revenue codes while keeping paid encounters:

```
%*sort by tcn and line_number for array purpose;
%MACRO sort1 (n5,tcn);
    PROC SORT DATA = &lib..file_&n5;
        by &tcn &lnum;
    RUN;
%MEND sort1;

%sort1 (5,TCN_Transaction_Control_Numb)

%*assign flags "0", "2" and "P" appropriately;
%MACRO prexc (n5,n7,tcn,allo);
    DATA &lib..file_&n7 (drop = i);
```

```

SET &lib..file_&n5;
array exc (20) $ &x.1 - &x.20;
array disp (20) $ &z.1 - &z.20;
by &tcn;
FLAG = "P";
do i = 1 to 20;
    if exc(i) in ("431", "439") and disp(i) = "2" and
        &allo = "K" and FLAG ^= "2" then FLAG = "0";
    if exc(i) not in ("445", "434") and disp(i) = "2" and
        &allo = "K" then FLAG = "2";
end;
RUN;
%MEND prexc;



```

The data step above will generate the following:

Example 10

MCO_ICN	CLAIM_STATUS	LINE_NUMBER	EXC_CD_1	EXC_DIS_1	...	EXC_CD_20	EXC_DIS_20	TEMP_NUMBER	GROUP_NUMBER	FLAG
7A38773	N	00	431	2	...	364	5	1	11119	0
7A38773	N	01	423	5	...	423	5	0	11119	P
7A38773	N	02	424	5	...	445	2	0	11119	2

The next step is to take out flags "0" and "2" to complete the deletion of non-covered procedure, revenue and diagnosis codes:

```

%*remove flags "0" and "2";
%MACRO removepe (n7=7,n8=8);
    DATA file_&n8;
        set file_&n7;
        if flag = "2" or flag = "0" then delete;
    RUN;
%MEND removepe;

%removpe

```

CONCLUSION

There is no question that Medicaid encounter data can be extremely complicated, but don't let this giant monster scare you off. Techniques and strategies used in this paper include the creation of a switch and a distinctive group number for each encounter and the deletion of duplicate claims and irrelevant procedure, revenue and diagnosis codes and, because of the massive number of observations and variables and many inconsistencies in healthcare encounter data, can be applied to data mining in any fields facing with huge and dirty datasets.

REFERENCES

Carpenter, Arthur L. (2004). *Carpenter's Complete Guide to the SAS Macro Language*. Cary, NC: SAS Institute Inc.

Making Medicaid Work. (2007). SHPS and Center for Health Transformation. Louisville, KY

Scerbo, Marge, *et al.* (2005). *Health Care Data and SAS*. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Wei Jiang
Mercer Human Resource Consulting
3475 Piedmont Road NE Suite 800
Atlanta, GA 30305
Work Phone: (404)442-3100
Fax: (404)442-3300
E-mail: jim.jiang@mercer.com
Web: www.mercerGHSC.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.