STRATEGIES FOR LINKING MEDICAID ELIGIBILITY FILES TO VITAL STATISTICS BIRTH FILES TO ASSESS THE EFFICACY OF MEDICAID MANAGED CARE

Patrick J. Roohan, NYS Department of Health
John R. Piddock, NYS Department of Health
Peter B. Lannon, NYS Department of Health

I. Introduction

In 1996, the New York State Department of Health (NYSDOH) was awarded a grant from Health Resources and Services Administration (HRSA) to build the information system infrastructure to incorporate birth record data into the Medicaid Encounter Sata System (MEDS). The grant also provided resources to link mother/infant inpatient discharge data to birth record data. NYSDOH has experience in linking data sets for specific analytic purposes, but this grant was an opportunity to create linked files for multiple uses throughout the department. As managed care becomes the delivery of health care in Medicaid, linked file systems are essential to compare outcomes and quality of care to the previous fee-for-service system.

This paper will cover two major areas: the linking process and applications using the data. Multiple stages of matching are preformed using SAS® software to insure a high level of accuracy, while minimizing false positives. Use of this linked data set will be discussed including efforts to evaluate the efficacy of managed care in Medicaid.

II. The Process

To link the Medicaid eligibility data to the Vital Statistics birth files, there are three major stages of matching to be preformed:

1) The Vital Statistics (VS) birth file to all claims (with a delivery) from the Fee-for-Service (FFS) claims file.

2) The VS, with FFS matches, to Quality Assurance Reporting Requirements (QARR) data. QARR are annual performance indicator reports submitted by managed care plans in New York State. Information on mothers that delivered during the reporting year is required by managed care plans.

3) The VS, with FFS and QARR matches, to all women aged 10-50 that are enrolled for one or more months in that calendar year from the Medicaid Managed Care (MC) enrollment file.

Within these stages there are two principal steps; cleansing and matching of the data. Once the three stages are completed, the VS birth file with FFS, QARR, and MC matches is tested one last time to further ensure that the number of false positives are minimized. The valid records are then matched with the VS birth statistical file, giving the matched production data file.

II.a Cleansing of Data Files

The cleansing of data files is dependent on each stage and it's associated data files, due to their unique data elements. However, there are some common cleansing steps among the stages. They are:

- Eliminate of out-of-state records.
- Check coded variables to ensure they are valid and correct them if not; i.e. hospital codes correspond to the appropriate hospitals.
- Check the data files for duplicate records. These could be the result of either multiple births (i.e. twins) or an error with data entry. If duplicate records do exist, they are split off into another data file and addressed later.
- For QARR, add an indicator to each record to distinguish whether it is Medicaid or Commercial.
- Change the street address for each record to a standardized format.

The standardization of the street address is required because the characters used in a street address may take various forms between data files and even among records within a data file. Without having the addresses in a standardized format, this field would be, in essence, useless for matching. The SAS® code for the street address standardization macro is as follows.

```
%MACRO STREETS(STRVAR);

ST_NW_NM=' ' || &STRVAR || '     ';
ST_NW_NM=TRANSLATE(ST_NW_NM,'      ','#"*.');
ST_NW_NM=TRANWRD(ST_NW_NM,' ALLEY ',' AL ');
ST_NW_NM=TRANWRD(ST_NW_NM,' AND ',' & ');
ST_NW_NM=TRANWRD(ST_NW_NM,' AVENUE ',' AV ');
ST_NW_NM=TRANWRD(ST_NW_NM,' AVEN ',' AV ');
```

```
ST_NW_NM=TRANWRD(ST_NW_NM,' AVE ',' AV ');
                          .
                          .
                          .
ST_NW_NM=TRANWRD(ST_NW_NM,' PO BX ',' PO BOX ');
ST_NW_NM=TRANWRD(ST_NW_NM,' RIDGE ',' RI ');
ST_NW_NM=TRANWRD(ST_NW_NM,' ROAD ',' RD ');
ST_NW_NM=TRANWRD(ST_NW_NM,' ROUTE ',' RT ');
ST_NW_NM=TRANWRD(ST_NW_NM,' ROUT ',' RT ');
ST_NW_NM=TRANWRD(ST_NW_NM,' RTE ',' RT ');
ST_NW_NM=TRANWRD(ST_NW_NM,' SAINT ',' ST ');
ST_NW_NM=TRANWRD(ST_NW_NM,' SISTER ',' SR ');
ST_NW_NM=TRANWRD(ST_NW_NM,' SOUTH ',' S ');
ST_NW_NM=TRANWRD(ST_NW_NM,' SOUT ',' S ');
ST_NW_NM=TRANWRD(ST_NW_NM,' SO ',' S ');
ST_NW_NM=TRANWRD(ST_NW_NM,' SQUARE ',' SQ ');
ST_NW_NM=TRANWRD(ST_NW_NM,' STREET ',' ST ');
ST_NW_NM=TRANWRD(ST_NW_NM,' STRE ',' ST ');
ST_NW_NM=TRANWRD(ST_NW_NM,' STR ',' ST ');
ST_NW_NM=TRANWRD(ST_NW_NM,' TERRACE ',' TER ');
ST_NW_NM=TRANWRD(ST_NW_NM,' TERR ',' TER ');
                          .
                          .
                          .
ST_NW_NM=TRANWRD(ST_NW_NM,'1ST ','1 ');
ST_NW_NM=TRANWRD(ST_NW_NM,'1TH ','1 ');
ST_NW_NM=TRANWRD(ST_NW_NM,'2ND ','2 ');
ST_NW_NM=TRANWRD(ST_NW_NM,'2TH ','2 ');
                          .
                          .
                          .
ST_NW_NM=TRANWRD(ST_NW_NM,' FIRST ',' 1 ');
ST_NW_NM=TRANWRD(ST_NW_NM,' SECOND ',' 2 ');

ST_NW_NM = COMPRESS(ST_NW_NM,"$()',-@&:;{}[]_");
ST_NW_NM = COMPBL(ST_NW_NM);
&STRVAR  = TRIM(LEFT(ST_NW_NM));
DROP ST_NW_NM;
%MEND;

%STREETS(STREET);
```

The macro initially adds one space to the left and 5 spaces to the right of the field by means of concatenation. It then changes some non-essential symbols (#"*.) to blanks and certain key characters to a standardized version (i.e. " street ", " stre " and " str " are modified to " ST "). Finally, the street address has special characters and the blanks on the left of the field removed.

II.b Matching of Data Files

For each stage, the matching of the data files is achieved through an eight round process. By round, we mean a pass at the data using a combination of personal identifiers, or 'linker' variable, to try and find matching records. Each round has the same method;

- match the data files by means of the "matching macro",
- edit the matches to minimize the number of false matches at the expense of some potential matches, and
- process the unmatched records for the next round.

II.b.1 Matching Macro

For each round, matching starts by defining the 'linker' variable. The attached table shows what the 'linker' variable is for each round of the three stages. The 'linker' variable and the two cleansed data sets are then submitted to the "matching macro".

This macro uses pointer files for matching. First the macro sorts the two data sets by the 'linker' variable. It then sets up a pointer file for each of the data sets. The pointer files are then matched for every combination between the two data sets based on the linker variable, creating a merged pointer file. The two original pointer files are then deleted and the actual observations from the data sets are combined based upon the merged pointer file.

```
%MACRO MATCH(DATASET1,DATASET2,LINKER);

 PROC SORT DATA = &DATASET1;  BY &LINKER;
 PROC SORT DATA = &DATASET2;  BY &LINKER;

DATA POINTER1 (KEEP = START1 END1 &LINKER);
  RETAIN START1;
  SET &DATASET1; BY &LINKER;
   IF FIRST.&LINKER THEN START1 = _N_;
   IF LAST.&LINKER  THEN DO;
     END1 = _N_; OUTPUT;
   END;

DATA POINTER2 (KEEP = START2 END2 &LINKER);
  RETAIN START2;
  SET &DATASET2; BY &LINKER;
   IF FIRST.&LINKER THEN START2 = _N_;
   IF LAST.&LINKER  THEN DO;
      END2 = _N_; OUTPUT;
   END;

DATA MATCHEDP (KEEP = START1 END1 START2 END2);
  MERGE POINTER1 (IN = D1MATCH) POINTER2 (IN =
D2MATCH); BY &LINKER; IF D1MATCH AND D2MATCH;

PROC DELETE DATA = POINTER1 POINTER2;

DATA MATCHED (DROP = START1 END1 START2 END2);
  SET MATCHEDP;
  DO I = START1 TO END1;
    SET &DATASET1 POINT = I;
    DO J = START2 TO END2;
      SET &DATASET2 POINT = J; OUTPUT;
    END;
  END;
 PROC DELETE DATA = MATCHEDP;

%MEND MATCH;
```

The results of this macro are three data sets returned to the main program; the matched data set and two non-matched data sets. The matched data set contains the matched set

of records and all variables from both inputted data sets. Each of the two non-matched data sets contains ONLY the unmatched variables from that original data set (i.e. unmatched data set #2 contains ONLY records from the 2nd data set that did not match with a record from the 1st data set).

II.b.2 The Edits

After the records match, using the 'linker' variable, we then apply an edit to the matched records to ensure that the matches were valid. If the matched records meet the edit criteria, then the match is kept. If it did not, then both records are put back into the proper unmatched file to be used in subsequent rounds. For example, suppose we tried to match up records using only the mom's date of birth as the 'linker' variable. You would expect to have many matches, both valid and not. In order to try to electronically determine which of the matches are valid, we may apply the following edit to the matched records: the infant's last name or address must agree.

```
Data Matched;
  Set Matched;
  If xInfLst = InfLst or xSTAdd = STAdd;
Run;
```

Using this process eliminates the need to manually verify the matched records. This has several advantages:

1. Saves a large amount of time, especially considering there are over 100,000 matches.

2. Provides a consistent quality of matches over time. Using the same standards, the computer makes objective determination for each record. Further, it removes human elements, which leads to arbitrary selection.

3. Control over the quality of matches.

The only disadvantage is the trade off between letting a few bad matches through the edit process and maintaining a high percent of records matched. However, we can control this to some degree by use of the 'global' edit.

By 'global' edit, we mean edit criteria that were applied to every round of matching for that stage (note: the QARR stage does not use the global edit because of the limited number of data fields within the QARR data file). The global edit occurs after the matching of records and their passage through the regular edits.

```
Data Matched; Drop FieldCt;
  Set Matched; FieldCt = 0;
  xMomLst = left(compress(xMomLst,'-'));
  xInfLst = left(compress(xInfLst,'-'));
  MomLst  = left(compress(MomLst,'-'));
  InfLst  = left(compress(InfLst,'-'));
  DadLst  = left(compress(DadLst,'-'));
  If (xMomLst = DadLst or xMomLst = MomLst or
      xMomLst = InfLst) and
     (xMom1st gt '                ')
  then FieldCt = FieldCt + 1;
 If (xInfLst = DadLst or xInfLst = MomLst or
     xInfLst = InfLst) and
    (xInfLst gt '                 ')
    then FieldCt = FieldCt + 1;
 If xMom1st = Mom1st and xMom1st gt '          '
    then FieldCt = FieldCt + 1;
 If xInf1st = Inf1st and xInf1st gt '          '
    then FieldCt = FieldCt + 1;
 If xMomDOB = MomDOB and xMomDOB gt '       '
    then FieldCt = FieldCt + 1;
 If xInfDOB = InfDOB and xMomDOB gt '       '
    then FieldCt = FieldCt + 1;
 If substr(xMacRes,1,10) = substr(MacRes,1,10)
    and substr(xMacRes,1,10) gt '          '
    then FieldCt = FieldCt + 1;
 If xZip = Zip and xZip gt '     '
    then FieldCt = FieldCt + 1;
 If xCounty = County and xCounty gt '  '
    then FieldCt = FieldCt + 1;
 If xPlace = Place and xPlace gt '     '
    then FieldCt = FieldCt + 1;
 If xSSN = SSN and xSSN gt '          '
    then FieldCt = FieldCt + 1;
  If FieldCt gt 3;
Run;
```

It compares all of the printed fields and requires that at least 4 of the fields have to agree for the match to be considered valid, otherwise it is thrown back for matching in future rounds. The 'global' edit does have control over the quality of matches. If we wish for a higher quality of match, we increase the minimum number of fields in agreement to 5 or 6. For increased match rates, we could lower the cutoff to 3.

One final edit is performed at the end of the third, or MC, stage of matching. This edit occurs between when the eight matched files are combined and adding the unmatched VS file. The edit compares the matched records, excluding those matched in the eighth round, against the eight 'linker' variables to determine whether the records would have been matched in other rounds. If any record indicates that it would not have been matched in any other round, it is visually inspected to check the validity of its match. If the match is deemed invalid, it is returned to the unmatched files.

II.b.3 Process of Unmatched Records for The Next Round

After the matching and edits, the unmatched data sets are then checked to ensure that all of their data elements are still intact, that no elements from the other data set were accidentally added, and that they are ready for the next round. The first step of this process is to strip off all of the matched records from the original data sets containing the 'linker' variable, thus leaving all unmatched records after edits.

We achieve this by merging each of the two original unmatched data sets with the matched data set with the condition that all merged data elements containing a match are deleted. This ensures that the unmatched data sets contain no data elements from the matched data set. Further, this step also 'adds' back those records that did not pass the edits.

```
**  Unmatched birth records from original birth;

Proc sort data = MatR1; By CertNo; Run;
Proc Sort data = BR1; By CertNo; Run;

Data UnMatBR1;
 Merge MatR1 (in = DidMat) BR1 (in = DidBir);
 By CertNo; If Not DidMat;
Run;

Proc delete data = BR1; Run;

**  Unmatched FFS records from original FFS;

Proc sort data = MatR1; By MdNo; Run;
Proc Sort data = FFSR1; By MdNo; Run;

Data UnMatFR1;
 Merge MatR1 (in = DidMat) FFSR1 (in = DidFFS);
By MdNo;  If Not DidMat; Run;

Proc delete data = FFSR1;  Run;
```

The second step of this process uses a drop statement for the other unmatched data set's variables, thus ensuring that the unmatched data set does not contain any of the other unmatched data set's variables. In addition, those records that did not contain the 'linker' variable are added back.

```
**  Get the unmatched birth records ready for ;
**  next round.                                ;

Data UnMatBR1;
 Drop xAllRec1 xAllRec2 MdNo xCounty xStreet
 xZip XMom1st xMomLst xSSN xMomDOB xInf1st
 xInfLst xInfDOB xPlace xMacRes xMomDOBY
 xMomDOBM xMomDOBD xInfDOBM;
 Set UnMatBR1;

Proc Append base = UnMatBR1 data = NBR1;
Proc delete data = NBR1; Run;

Data UnMatBR1; Drop Linker;
 Set UnMatBR1;

**  Get the unmatched FFS records ready for ;
**  next round.                             ;

Data UnMatFR1;
 Drop AllRec1 AllRec2 CertNo County Street Zip
 MCD Mom1st MomLst SSN MomDOB Inf1st InfLst
 InfDOB DadLst Place MacRes MomDOBYY MomDOBDD
 MomDOBMM InfDOBMM InfLst2 MomLst2 Mom1st2;
 Set UnMatFR1;

Proc Append base = UnMatFR1 data = NFFSR1;
Proc delete data = NFFSR1; Run;

Data UnMatFR1; Drop Linker;
 Set UnMatFR1;
```

After the eight rounds are completed, the eight 'matched' files are combined into one file using the PROC APPEND statement.

II.c Process Files for the Next Stage

Between stages, there are a few steps to be performed. First we address the issue of duplicate/twin records. Here is where we separate the multiple births, and women who had two pregnancies within a year, from true duplicate records. Recall that before the matching process, a duplicate record data file for each data file was created. These files contain true duplicates, subsequent twins and subsequent records for moms who had more than one birth in a calendar year. The matched data file is checked for records containing duplicated mother's identification numbers. If one is found, we know that the same mother had matched to more than one birth record. Next is to compare the infant's first name and date of birth between the two data files to find out which of the multiple birth records truly matched. The incorrectly matched records of the two data files are then compared between data files based on mother's identification number, infants' date of birth and first name. If the correct records are found, they are then added to the valid matches.

Next, the matched file is then stripped of non-essential information and has a 'match' indicator added to the records. The unmatched VS file has a 'match' indicator added to its records and is reformatted mirroring that of the matched file. The two files are then combined. The file is then ready for either the next stage of matching (if the first or second stage is just completed) or applications requiring the data.

III. Applications

Previous to this project, the Vital Statistics birth file did not contain accurate information on the payer field. By linking to the Medicaid files and Quality Assurance Reporting Requirements (QARR) data, the opportunity to analyze birth outcomes by payer groups is now possible. The following are the current applications using this linked data set, with unlimited potential for further analysis.

III.a Prenatal Care and Birth Outcomes Performance Measurement

As part of QARR, managed care plans submit identifying information on mothers that have a delivery in the reporting year. The plans are not required to submit performance indicators, as the NYSDOH calculates them using the linked file. Quality measures calculated include trimester prenatal care began, risk-adjusted low

birthweight, prenatal care utilization, cesarean delivery rates, vaginal birth after cesarean (VBAC) rates and access to tertiary care centers for low birthweight infants.

The linked files allow NYSDOH to calculate risk-adjusted low birthweight rates for both Medicaid and commercial managed care. Using PROC LOGISTIC, logistic regression models are developed to calculate an expected low birth weight rate for each managed care plan. Variables included in the model include maternal age, race, ethnicity, parity, plurality, maternal education, smoking, drinking, substance abuse and medical risk factors. This methodology allows for plan-to-plan comparison of low birthweight rates of managed care plans, accounting for the various levels of risk.

III.b Comparison of Medicaid Fee-for-Service and Managed Care

Logistic regression models have been developed to compare the low birthweight rates of Medicaid recipients in fee-for-service (FFS) and managed care. Using the linked Medicaid birth certificate file, risk-adjusted odds ratios have been developed to compare the different health care delivery systems in Medicaid. Comparing FFS to managed care resulted in an odds ratio that was not significant adjusting for maternal age, race, ethnicity, parity, plurality, maternal education, smoking, drinking, substance abuse and medical risk factors. As New York State enrolls Medicaid recipients into managed care, the evaluation of birth outcomes will be an integral part of measuring quality of care.

III.c Regional Analysis of Birth Outcome Data

A coalition of representatives from Albany, Schenectady and Rensselaer Counties, along with managed care plans has formed to address birth outcome issues in the Capital District area of New York State. The linked Medicaid birth certificate file allows for regional analysis of prenatal care, low birthweight, cesarean delivery rates and teenage pregnancy rates. This >regional approach= is a partnership of both public (county governments) and private (managed care plans) efforts to improve prenatal care and birth outcomes.

III.d Risk-Adjusted Cesarean Section Rates

Crude rates of cesarean section rates for managed care plans do not account for the clinical factors that increase the risk of cesarean delivery. Using PROC LOGISTIC, logistic regression models are being developed to account for the variation in risk profiles of managed care plans. Plan-to-plan comparisons of cesarean section delivery rates will be computed, accounting for maternal age, parity, preexisting co-morbid conditions, gestation, infant body weight, obstetrical conditions and multiple gestation.

IV. Acknowledgements

Partial funding for this project was provided by HRSA, Maternal and Child Health Bureau.

The authors would like to thank Michael Sawyer for his contributions to this project.

The authors would like to thank Mike Zdeb for his background work in matching data sets with DOH.

TABLE: Listing of Linker Variables and Round Specific Edits

VS/FFS

Rd	Linker	Edit
1	Mom SSN	None
2 FFS	Mom Last + Mom First + Mom DOB + Infant DOB	None
VS	Mom Maiden + Mom First + Mom DOB + Infant DOB	None
3 FFS	Mom Last + Mom First + Mom DOB + Infant DOB	None
VS	Infant Last + Mom First + Mom DOB + Infant DOB	None
4	Mom DOB + Infant DOB	2 chars of Infant First Name or 3 chars of Address
5	5 chars of Address + Mom Year of Birth	Mom Month of Birth or Mom Year of Birth
6	Hospital of Birth + Mom DOB	Compare 3 chars of Mom, Infant and Dad Last Name.
7	4 of Infant Last Name + 3 of Infant First Name + Infant Month of Birth	First 4 of Address or Mom Year of Birth
8	4 of Mom Last + 3 of Mom First	4 of Address or Infant DOB or Infant First Name

VS/FFS/QARR

Rd	Linker	Edit
1	Mom First + Mom DOB + Infant DOB + PFI	Either Last Name
2	First 2 chars of Mom First Name + Mom DOB + PFI	First 2 chars & Last 2 chars of Either Last Name
3	7 chars of Mom First Name + PFI + Mom Year of Birth	7 chars of Mom Last Name + Month and Day of Mom and Infant DOB
4	5 chars of Mom First Name + Mom DOB + Infant DOB + PFI	None
5	5 chars of Mom First Name + YYMM of Mom Birth + YYMM of Infant Birth	5 chars of Either Last Name
6	5 chars of Mom First Name + PFI + Year of Mom Birth	Compare 3 chars of Mom, Infant & Dad Last Name.
7	Mom DOB + Infant DOB + PFI	
8	4 chars of Mom First Name + Mom DOB + MMDD of Infant Birth	4 chars of Either Last Name

VS/FFS/QARR/MC

Rd	Linker	Edit
1	Mom SSN	None
2	Mom CIN	None
3	Mom First Name + Mom DOB + 5 char of Address + Zip	Mom Maiden Name match either Mom, Infant, or Dad Last Name
4	Mom First Name + Mom DOB + 5 char of Address	5 chars of Mom Maiden Name must 5 chars of either Mom, Infant, or Dad Last Name
5	First 2 chars of Mom First Name + Mom DOB + 5 char of Address + Zip	First & Last 2 chars of Mom Maiden Name match First and Last 2 chars of either Mom Infant, or Dad Last Name
6	Mom First Name + Mom DOB + Zip	Mom Maiden Name match either Mom, Infant, or Dad Last Name
7	First 5 chars of Mom First name + 5 chars of Address + Year of Mom Birth	5 chars of Mom Maiden Name match 5 chars of either Mom, Infant, or Dad Last Name
8	First 3 chars of Mom First name + Year and Month of Mom Birth + Zip	First & Last 2 chars of Mom Maiden Name match First and Last 2 chars of either Mom, Infant, or Dad Last Name