Paper 205-28

Linkage of Patient Registries and Clinical Data Sets without Patient Identifiers

Paul D. Frederick, Ovation Research Group, Seattle, WA

ABSTRACT

Patient registries seek to capture data from broader-based patient populations having a single diagnosis and may lack clinical detail or precision whereas clinical data are drawn from intensive discovery and are limited to select patients related to the study. Clinical data may be used to validate information collected in the patient registry, and patient registry data may complement ongoing clinical studies, such as to provide essential baseline or historical data. A simple and versatile record-linkage is performed between mock registry and clinical datasets that represent data collected in hospitals nationwide. The linkage presumes the lack of common unique patient identifiers and a multi-key approach is considered for defining matches. Composite keys are developed based on shared attributes within the data schemata. Various composite keys are developed consisting of hospital, discharge and arrival dates, age, and gender. Blocking is used to reduce number of non-matches resulting from cartesian product match merges. Finally, exact- and fuzzy-matching techniques are used to link records. Simple methods to deduplicate matches are also proposed. Risks to privacy and confidentiality of subjects are of notable concern. SAS® BASE and SAS DATA STEP on Windows NT SAS version 6.12 are used for the match-merge process.

INTRODUCTION

There are often tradeoffs when designing data collection tools - a process that is driven primarily by scarce resources. Where the Registry dataset seeks to obtain many cases it does so by relaxing the protocols that govern the process of data management and quality and by limiting the quantity of information collected. Where the Clinical dataset seeks to collect a great deal of quality information under managed processes it does so by limiting the total number of records it can populate. Each repository of information is designed and developed independently, each may share a common data schema, and each has its own strengths and weaknesses. A mutually beneficial cross sharing of information between the two repositories can add value to the knowledge of each as well as provide economic and scientific synergies. This paper will outline a simple record-linkage between two mock datasets that do not contain a common primary-key identifier.

PREPARING DATASETS

Before the match-merge process can begin, both datasets must adhere to conventions. Integrating the representation of data is essential for the match-merge process and naming conventions facilitate code development.

INTEGRATION

The datasets are presumed to represent similar patient populations characterized by a single diagnosis, and hospitals from the Clinical and Registry datasets must have a common source identification number. If the Clinical dataset is longitudinal in design, the event record for which the diagnosis is reported should be selected and subsequent visit information should be drawn up into the event record. In this demonstration, both 60-day and 1-year mortality fields are brought into the event record in the Clinical dataset. In the Registry data, baseline data also include a medication. In this example, two heterogeneous Clinical and Registry datasets representing data collected nationwide are presented in Appendix 1 and 2.

All field names should be unique across the Clinical and Registry datasets. The variable types and lengths of the fields on which the match is performed should compare identically. Formats and informats should be removed. Field nomenclature should be the similar. The convention is to prefix the field name with one character that identifies the data source of all fields. Identically suffix the field when fields are shared between the two data sources. Fields can be labeled. In fields that are either binary or logical, e.g., in gender or cpage (Age Check), values should be coded as 1 or 0. Field naming conventions are shown in Figure 1.

FIELDS IN COMMON

CLINICAL Dataset	REGISTRY Dataset	Field Labels
CPID	RPID	Patient ID
CSTATE	RSTATE	State
CHOSPID	RHOSPID	Hospital
CGENDER	RGENDER	Gender
CADMIT_D	RADMIT_D	Admission
CDISCH_D	RDISCH_D	Discharge
CFIRST	RFIRST	First Initial
CMIDDLE	RMIDDLE	Middle Initial
CLAST	RLAST	Last Initial
CDEATH	RDEATH	Death [In-Hospital]
CDOB	RDOB	DOB
CTHERAPY	RTHERAPY	Therapy
CRANDSCO	RRANDSCO	Random Score
CNOTBLNK	RNOTBLNK	Completeness Score
CAGE	RAGE	Patient Age [Yrs]
CHXDIAB	RHXDIAB	History of Diabetes

Figure 1.

FORCING TEMPORAL CONCURRENCE

Reduce the number of records that are not related in regards to time period. Using the Clinical Dataset, the minimum admission date and maximum discharge date are determined for each hospital and then applied to the larger the Registry dataset. From the Registry dataset, records are removed in which non-overlapping discharge or admission dates occur with respect to the Clinical Dataset (see Appendix 3 and 5).

BLOCKING AND REDUCTION

Limiting the match-merge process to geographically identical locations can greatly reduce processing time. Blocking is used to reduce number of non-matches resulting from cartesian product match merges. The prepared Registry and Clinical datasets are split into separate datasets based on the state in which the patient was admitted. This keeps the number of cartesian product match merges to a minimum (see Appendix 6). Any hospitals not shared between the datasets are removed along with their patients (see Appendix 5).

MATCH-MERGE

To match records between the Registry and Clinical datasets, an iterative program module is used to identify up to 5-Key exact or fuzzy matches. In order to make exact or fuzzy comparisons with respect to common fields, one record from the first SAS dataset is held open while it reads all the records of the second database for possible matches to the one record in the first. This process is accomplished by using the POINT option under the SET statement in a datastep (see Appendix 7). When all attributes within the key were the same between Registry and Clinical datasets, the match was considered exact. Variance allowances for select attributes are permitted. Variables, such as Age, can be

permitted an allowance of up to ±1 or ±2 years. Matches on dates can be fuzzed as well. Records with missing values in any field represented in the 5 key-ID are removed (see Appendix 1 and 2).

MATCHED FIELDS

Hospital ID, Patient Age, Gender, Admission Date, and Discharge Date comprise the basic composite key. The composite key can consist of supplemental fields such as patient initials or date of birth (DOB), if known. DOB should not be used in the match but rather can be used to calculate an age in years of the patient if a self-reported age is not already provided. The same standard for age calculation should be used on each side. The FLOOR function used in calculating age of patient best describes a patient's self-reported age (see Appendix 1 and 2). The composite keys can be classified into a 5-Key ID (Hospital ID, Age, Gender, Admit Date, & Discharge) and different variations within the 5-key such as a 4-key, 3-key, or 2-key ID (see Appendix 7). From each composite-key match, a separate dataset is appended to the next with each set coded for the type of match (see Appendix 8).

DEDUPLICATION

The purpose of deduplication is to identify and remove multiple records containing the same or similar information as defined by the chosen composite key. It is possible that duplicate records may exist in either the Registry or Clinical dataset before the match-merge process begins. Although the deduplication process must occur after the match-merge process, as it is done in this paper (see Appendix 9), deduplication can optionally occur at the level of each dataset and prior to the match-merge process. Using the combined method of deduplication is particularly helpful with large datasets or datasets that lack high standards for data collection.

SPREADING

During the match-merge process, spreading (replication of same record) occurs whenever more then one hit occurs in either dataset (see Appendix 7). Hits are defined as a record in which the composite-key ID in the first dataset is found in one or more records base on the same composite-key ID in the second. To identify replication, records are arbitrarily assigned a unique record called CSYSID and RSYSID identifier based on _N_ processing at the level of the initial datastep found in Appendix 1 and 2. The unique system identifier is needed for tiebreaking. The use of unique patient identification numbers may be problematic, especially when the dataset contains patient IDs that are not unique. Therefore, the patient identification number is not used even though they are present in each dataset.

METHODS FOR TIEBREAKING

A simple method based on weighting or scoring algorithms is used to assign preferential treatment to the match whenever a replicate group is identified based on repeating values of CSYSID and RSYSID. Whenever a replicate CSYSID is produced, it means that a duplicate record exists in the Registry dataset, and a replicate RSYSID means that a duplicate record exists in the Clinical dataset. Logical fields that test equalities are produced for exact and fuzzy matches. The field prefix of CP designates an exact match, and FZ designates a fuzzy match between two fields in opposite datasets (see Appendix 7).

When an exact match occurs at the level of the 5-Key ID or less, tiebreakers can be used to determine the preferred record among the replicates. In Appendix 1 and 2, a completeness score is generated so that a record having the least amount of information receives less preference. Replicates that record a mortality event may be given preference over the other as well. Patient initials may be known and preferential treatment can be given to either partial or complete matches in patient initials. A random score generated in each datastep (see Appendix 1 and 2) ultimately breaks the tie if no other criteria are available.

PROC SORT is used in combination with FIRST. processing to position and select records with higher preference in the case where replicates (spreaders) exist (see Appendix 9).

TESTING AND ANALYSIS

The Clinical dataset can serve to validate the Registry dataset using a field that was not part of the match but is common to both datasets. The binary fields of HXDIAB or DEATH can be used in PROC FREQ using the AGREE option to produce kappa statistics (See Appendix 9). In addition, baseline covariates collected in the Registry dataset, e.g. medications and possible procedures, can be analyzed in SAS/STAT® PROC PHREG (not shown here) for possible effects on 60-day and 1-year survival data, which are provided in the matched record from the Clinical dataset.

DISCUSSION

In Appendix 9, the processing speed is greatly enhanced by using only the fields that are directly related to the match. The RSYSID and the CSYSID fields can be used to merge the matched records back into the Registry or Clinical datasets for corroboration of the match and accessing other data that are not directly related to the merge. An error-free "match" is not guaranteed. Although the match-merge process is designed to control matching error, the conclusions to be drawn from any match should ultimately rely on the quality of each data repository. Mutually distrusting parties will want to preserve the integrity, privacy, and confidentiality of each data repository. A blinded (encrypted) patient ID should be used in lieu of the actual patient ID provided that a crosswalk file to the originally patient ID is securely held by each party. Policies regarding the release of the content of cross-information that occurs between parties should be governed by policy guidelines that are put in place prior to data sharing. The data repositories should be isolated and protected with operating system authentication and other security protocols and should not be shared with external parties.

CONCLUSION

Two different datasets sharing common fields and patients are matched. A simple match-merge between two similar datasets is demonstrated. The better the data source reliability, the more likely the success and quality of each hit in the match-merge process. As concern over patient confidentiality grows, date fields such as admission date, discharge date, birth date and hospital ID or hospital location are less likely to be available to researchers in developing unique composite keys for data integration.

REFERENCES

Patridge, C., "The Fuzzy Feeling SAS Provides: Electronic Matching of Records without Common Keys", *Proceedings of the Twenty-Second Annual SAS® Users Group International Conference*, 1997; San Diego, California.

Contact Information

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

Paul D. Frederick
Ovation Research Group
5910 Evanston Avenue N.
Seattle, WA 98103
Work Phone: (206) 789-4143
Fax: (206) 789-4173
Email: pfrederick@ovation.org

APPENDIX 1 (CLINICAL DATA):

```
400101101M05/16/199605/29/1996U A101010/21/19460
400201101M05/16/199605/21/1996B P100107/08/19651
400301102F06/01/199606/02/1996J R100003/04/19651
400401102F08/15/199608/16/1996VSA100110/10/19561
400502201M07/02/199607/04/1996NIF010006/28/19460
400602202F06/20/199607/02/1996L I000009/06/19630
400702202M08/09/199608/11/1996QQN001007/13/19510
400803301F04/01/200204/02/2002I L110010/09/19080
400903301M07/12/199607/26/1996Q F101007/28/19600
401003302F02/06/199602/07/1996CKH100004/19/19640
401103302M06/03/199606/10/1996F N000108/09/19651
401203302M06/03/199606/10/1996F N000008/09/19651
401304401F05/15/199607/19/1996ZZA100012/23/19660
401404401F02/01/199502/10/1995IOV000101/01/19650
401504402F09/18/199609/19/1996TZP010006/18/19801
401604402F09/18/199609/19/1996TZP0.0106/18/19801
401704402M07/15/199607/24/1996MRT100003/04/19610
401805501F08/15/199608/22/1996CTL100004/05/19630
401905502M07/22/199608/05/1996M F010005/06/19680
402005502F09/11/199609/12/1996SIJ000006/07/19601
402105502F09/11/199609/12/1996SIJ000106/07/19600
402206601M11/18/199611/20/1996BGX100108/09/19820
402306601F03/15/199603/16/1996YJQ110009/09/19680
402406610M10/18/199610/19/1996MEZ100110/10/19311
402501102F06/01/199606/02/1996J R100003/04/19651
402602202F06/20/199607/02/1996L I000009/06/19630
402705502M07/22/199608/05/1996M F010005/06/19680
402806601M11/18/199611/20/1996RGX100108/09/19820
```

APPENDIX 2 (REGISTRY DATA):

```
500101101M05/15/199605/29/1996UDA010/21/1946100
500201101F05/15/199605/21/1996DTB011/01/1955100
500301101F05/16/199605/21/1996B P007/08/1965110
500401102M06/11/199606/21/1996XTP012/25/1962100
500501102F06/01/199606/02/1996J R003/04/1965110
500601102F08/15/199608/16/1996VSA010/10/1956110
500702201M02/15/199702/16/1997A Q010/15/1965101
500802201M07/02/199607/04/1996NIF106/28/1964001
500902201M05/18/199605/21/1996LEA012/04/1906101
501002202F06/20/199607/02/1996L I009/06/1963001
501102202M08/09/199608/11/1996QQN007/13/1950000
501202202F03/11/200203/12/2002HLT001/01/1910000
501303301F04/01/200204/02/2002I L110/09/1908100
501403301M10/10/199610/11/1996FSL010/11/1966010
501503301M07/12/199607/26/1996QWF007/28/1960100
501603302F11/11/199611/15/1996W F112/15/1965111
501703302F02/06/199602/07/1996CKH004/19/1964100
501803302M06/03/199606/10/1996F N008/09/1965011
501904401F05/15/199605/19/1996ZZA012/23/1966100
502004401M07/07/199607/08/1996E T002/28/1966001
502104401F02/01/199502/10/1995IOV001/01/1965001
502204402F09/18/199609/19/1996TZP106/18/1980010
502304402M10/15/199610/22/1996D R009/10/1985000
502404402M07/16/199607/23/1996MRT003/04/1961101
502505501F01/06/199701/07/1997U E101/01/1960001
502605501F08/16/199608/22/1996CTL004/05/1961100
502705501M05/18/199605/19/1996ZPG006/07/1910100
502805502M07/22/199608/05/1996M F105/06/1968000
503005502M09/11/199609/22/1996A M007/12/1916000
503106601M11/18/199611/20/1996RGX008/09/1982100
503206601F03/04/199603/15/1996YJQ009/09/1968100
503306601M10/18/199610/19/1996MEZ010/10/1931110
503406602F10/19/199610/23/1996K N010/10/1960111
503501102F06/01/199606/02/1996J R003/04/1965110
503602202F06/20/199607/02/1996L I009/06/1963001
503705501F08/16/199608/22/1996CTL004/05/1961100
```

APPENDIX 1 (CONTINUED):

```
Data CLINICAL (Drop=NB1-NB4);
Input
@1 CPID 4. @5 CSTATE $2. @7 CHOSPID 3.
@10 CGENDER $1.
@11 CADMIT_D MMDDYY10.
@21 CDISCH_D MMDDYY10.
@31 CFIRST $1. @32 CMIDDLE $1. @33 CLAST $1.
@34 CHXDIAB 1.
@35 CDEATH 1. @36 C60DEATH 1. @37 C1YRDEATH 1.
@38 CDOB MMDDYY10.
@48 CTHERAPY 1.
;
LABEL
CPID='Patient ID [C]'
CSTATE='State [C]'
CHOSPID='Hospital [C]'
CGENDER='Gender [C]'
CADMIT_D='Admission [C]'
CDISCH_D='Discharge [C]'
CFIRST='First [C]'
CMIDDLE='Middle [C]'
CLAST='Last [C]'
CHXDIAB='History of Diabetes [C]'
CDEATH='Death [In-Hospital] [C]'
C60DEATH='Death [60-Day] [C]'
C1YRDEATH='Death [1-Year] [C]'
CDOB='DOB [C]'
CTHERAPY='Therapy [C]'
CAGE='Patient [C]'
CRANDSCO='Random Score [C]'
CNOTBLNK='Completeness Score [C]'
;

FORMAT CADMIT_D CDISCH_D CDOB MMDDYY10.;
CAGE=FLOOR((CADMIT_D - CDOB)/365.25);
CRANDSCO=ranuni(5234527);

/* ------------------------------------------
Scoring Step to Determine Completeness of Record
------------------------------------------ */
If CDEATH ne . then NB1=1;
If C60DEATH ne . then NB2=1;
If C1YRDEATH ne . then NB3=1;
If CTHERAPY ne . then NB4=1;
CNOTBLNK=SUM(NB1, NB2, NB3, NB4);

/* ------------------------------------------
Delete Records When 5-Key ID Elements missing
------------------------------------------ */
IF CSTATE='' then delete;
IF CDOB=. or CAGE=. then delete;
IF CHOSPID=. then delete;
IF CGENDER='' then delete;
IF CADMIT_D=. then delete;
IF CDISCH_D=. then delete;

DATALINES;
Add CLINICAL Data Here
;
Run;
Proc Contents; Run;
```

APPENDIX 2 (CONTINUED):

```
Data REGISTRY (Drop=NB1-NB4);
Input
@1 RPID 4. @5 RSTATE $2. @7 RHOSPID 3.
@10 RGENDER $1.
@11 RADMIT_D MMDDYY10.
@21 RDISCH_D MMDDYY10.
```

```
@31 RFIRST $1. @32 RMIDDLE $1. @33 RLAST $1.
@34 RDEATH 1.
@35 RDOB MMDDYY10.
@45 RHXDIAB 1.
@46 RTHERAPY 1.
@47 RMEDICATION 1.

;

LABEL
RPID='Patient ID [R]'
RSTATE='State [R]'
RHOSPID='Hospital [R]'
RGENDER='Gender [R]'
RADMIT_D='Admission [R]'
RDISCH_D='Discharge [R]'
RFIRST='First [R]'
RMIDDLE='Middle [R]'
RLAST='Last [R]'
RDEATH='Death [R]'
RDOB='DOB [R]'
RHXDIAB='History of Diabetes [R]'
RTHERAPY='Therapy [R]'
RMEDICATION='Medication [R]'
RAGE='Patient [R]'
RRANDSCO='Random Score [R]'
RNOTBLNK='Completeness Score [R]'
;

FORMAT RADMIT_D RDISCH_D RDOB MMDDYY10.;
RAGE=FLOOR((RADMIT_D - RDOB)/365.25);
RRANDSCO=ranuni(5234527);

/* ------------------------------------------
Scoring Step to Determine Completeness of Record
------------------------------------------ */
If RDEATH ne . then NB1=1;
If RTHERAPY ne . then NB2=1;
If RMEDICATION ne . then NB3=1;
If RHXDIAB ne . then NB4=1;
RNOTBLNK=SUM(NB1, NB2, NB3, NB4);

/* ------------------------------------------
Delete Records When 5-Key ID Elements missing
------------------------------------------ */
IF RSTATE='' then delete;
IF RDOB=. or RAGE=. then delete;
IF RHOSPID=. then delete;
IF RGENDER='' then delete;
IF RADMIT_D=. then delete;
IF RDISCH_D=. then delete;

DATALINES;
Add REGISTRY Data Here;
Run;
Proc Contents; Run;
```

APPENDIX 3:

```
%Macro MM(DFile,Pfx,ByVar,Var1,VN1,Var2,VN2);

/* ------------------------------------------ */
/*  Create Dataset                            */
/*  - For Each Hospital                       */
/*  - Produce Max Min Admission Date Ranges   */
/*  - Produce Max Min Discharge Date Ranges   */
/*  - Create Dataset for later merge          */
/* ------------------------------------------ */

Proc Sort Data=&DFile.; By &ByVar.; Run;
Proc Univariate Data=&DFile. NoPrint;
var &Var1. &Var2.;
Output
Out=&Pfx.MAXMIN
Max=&Pfx.MAX&VN1. &Pfx.MAX&VN2.
Min=&Pfx.MIN&VN1. &Pfx.MIN&VN2.
;
By &ByVar.;
Run;

%Mend MM;

%MM(CLINICAL,C,CHOSPID,CADMIT_D,AD,CDISCH_D,DD)
%MM(REGISTRY,R,RHOSPID,RADMIT_D,AD,RDISCH_D,DD)
```

APPENDIX 4:

```
%Macro INITIALS(DFile, Pfx, DName);

/* ------------------------------------------ */
/*  - Write to Log Any 5-Key replicates FYI   */
/*  - Determine Length of Patient Initials    */
/*  - Create System Unique Record ID          */
/* ------------------------------------------ */

Proc Sort Data=&DFile.;

By
&Pfx.STATE &Pfx.HOSPID &Pfx.GENDER
&Pfx.ADMIT_D &Pfx.DISCH_D &Pfx.AGE;
Run;

Data &DFile.1;

Length &Pfx.NINITS2 $3.;
Length &Pfx.NINITS3 $3.;
Length &Pfx.LINITS 4;

Set  &DFile.;

By
&Pfx.STATE &Pfx.HOSPID &Pfx.GENDER
&Pfx.ADMIT_D &Pfx.DISCH_D &Pfx.AGE;

IF Not (Last.&Pfx.AGE and First.&Pfx.AGE)
then Put &Pfx.PID=
"Possible Replicate Based on
&Pfx.STATE &Pfx.HOSPID &Pfx.GENDER
&Pfx.ADMIT_D &Pfx.DISCH_D &Pfx.AGE
";

&Pfx.SYSID=_N_;
label &Pfx.SYSID = "System &DName. ID";

If
&Pfx.FIRST Ne ' ' AND
&Pfx.MIDDLE Ne ' ' AND
&Pfx.LAST Ne ' '  Then Do;
  &Pfx.NINITS3 = Trim(&Pfx.FIRST) ||
                 Trim(&Pfx.MIDDLE) ||
                 Trim(&Pfx.LAST);
  &Pfx.NINITS2 = Trim(&Pfx.FIRST) ||
                 Trim(&Pfx.LAST);
  &Pfx.LINITS=Length(&Pfx.NINITS3);
End;
Else
If
&Pfx.FIRST Ne ' '
AND &Pfx.LAST Ne ' '  Then Do;
  &Pfx.NINITS2 = Trim(&Pfx.FIRST) ||
                 Trim(&Pfx.LAST);
  &Pfx.LINITS=Length(&Pfx.NINITS2);
End;
Label &Pfx.LINITS="Length of Initials
```

```
[&DName.]";
Label &Pfx.NINITS3="3 Initials [&DName.]";
Label &Pfx.NINITS2="2 Initials [&DName.]";
/* ---------------------------------------- */
/* strip formats                            */
/* ---------------------------------------- */
Informat _All_; Format  _All_;
Run;

%Mend INITIALS;
%INITIALS(CLINICAL,C, CLINICAL);
%INITIALS(REGISTRY,R, REGISTRY);
```

APPENDIX 5:

```
/* ---------------------------------------- */
/*  Datastep                                */
/*  - Apply Date Ranges                     */
/*  - From CLINICAL Dataset                 */
/*  - To REGISTRY Dataset                   */
/*  - Eliminate Hospitals in CLINICAL That  */
/*  - are Not in REGISTRY                   */
/*  - Remove REGISTRY Patients that Fall    */
/*  - Outside Min/Max Arrival/Discharge Dates */
/*  - 2 day allowance for Min/Min           */
/* ---------------------------------------- */

Proc Sort Data=WORK.REGISTRY1; By RHOSPID; Run;
Proc Sort Data=WORK.CMAXMIN; By CHOSPID; Run;
Data WORK.REGISTRY2
(Label='SELECTED PATIENTS & HOSPITALS');
Merge
WORK.REGISTRY1(In=A)
WORK.CMAXMIN(In=B Rename=(CHOSPID=RHOSPID));
By RHOSPID;
If A and B;   /* <--REDUCTION */
IF    /* <--FORCING TEMPORAL CONCURRENCE */
CMAXDD Ne . And CMINAD Ne . And
RDISCH_D Ne . And RADMIT_D Ne . Then DO;
   IF (RDISCH_D > (CMAXDD+2)) Then DELETE;
   IF (RADMIT_D < (CMINAD-2)) Then DELETE;
End;
Run;
```

APPENDIX 6:

```
%Macro State(Pfx); /* <--BLOCKING */
/* Continue states [01-50] As Needed */
     IF &Pfx.state = '01' Then OUTPUT &Pfx.01;
ELSE IF &Pfx.state = '02' Then OUTPUT &Pfx.02;
ELSE IF &Pfx.state = '03' Then OUTPUT &Pfx.03;
ELSE IF &Pfx.state = '04' Then OUTPUT &Pfx.04;
ELSE IF &Pfx.state = '05' Then OUTPUT &Pfx.05;
ELSE IF &Pfx.state = '06' Then OUTPUT &Pfx.06;
ELSE IF &Pfx.state = '07' Then OUTPUT &Pfx.07;
ELSE IF &Pfx.state = '08' Then OUTPUT &Pfx.08;
%Mend State;

/* Continue states [C01-C50] As Needed */
Data C01 C02 C03 C04 C05 C06 C07 C08;
Set WORK.CLINICAL1; %State(C);
Run;

/* Continue states [R01-R50] As Needed */
Data R01 R02 R03 R04 R05 R06 R07 R08;
Set WORK.REGISTRY2; %State(R);
Run;
```

APPENDIX 7:

```
%macro outstate(File);
Data
aMatch&File. bMatch&File. cMatch&File.
dMatch&File. eMatch&File. fMatch&File.
UnMat&File.;

Set Work.R&file (
Keep=RPID RSYSID RSTATE RHOSPID RAGE RGENDER
RADMIT_D RDISCH_D RDOB RNINITS2 RNINITS3
RLINITS RNOTBLNK RDEATH RRANDSCO RHXDIAB
);

Do i=1 To n;

Set Work.C&file. (
Keep=CPID CSYSID CHOSPID CSTATE CAGE CGENDER
CADMIT_D CDISCH_D CDOB CNINITS2 CNINITS3
CLINITS CNOTBLNK CDEATH CRANDSCO CHXDIAB
) Point=I Nobs=N;

If i gt n Then Goto Startovr;
If _Error_ = 1 Then Abort;

/* ---------------------------------------- */
/* - Create logic/flag fields               */
/* - for cross-field comparison checks [Ck] */
/* ---------------------------------------- */
%Let Lg=%Str(Length=4 label=);
attrib CPHOSPID &Lg.'Hospital Ck';
attrib CPSTATE &Lg.'State Ck';
attrib CPAGE &Lg.'Age Ck';
attrib FZCPAGE1 &Lg.'Age Ck[FZ 1 Yrs]';
attrib FZCPAGE2 &Lg.'Age Ck[FZ 2 Yrs]';
attrib CPGENDER &Lg.'Gender Ck';
attrib CPADM &Lg.'Arr Date Ck';
attrib FZCPADM1 &Lg.'Arr Date Ck[FZ 1 Yrs]';
attrib FZCPADM2 &Lg.'Arr Date Ck[FZ 2 Yrs]';
attrib CPDIS &Lg.'Disch Date Ck';
attrib FZCPDIS1 &Lg.'Disch Date Ck[FZ 1 Yrs]';
attrib FZCPDIS2 &Lg.'Disch Date Ck[FZ 2 Yrs]';
attrib CPDOB &Lg.'Date of Birth Ck';
attrib CPINITIALS &Lg.'Patient Initials Ck';

/* ---------------------------------------- */
/* - Produce logic/flag fields              */
/* ---------------------------------------- */

CPHOSPID=(RHOSPID=CHOSPID); /* <-EXACT */
CPSTATE=(RSTATE=CSTATE);
CPAGE=(RAGE=CAGE);
FZCPAGE1=(Abs(RAGE-CAGE) in (0 1)); /* <-FUZZ */
FZCPAGE2=(Abs(RAGE-CAGE) in (0 1 2));
CPGENDER=(CGENDER=RGENDER);
CPADM=(CADMIT_D=RADMIT_D);
FZCPADM1=(Abs(RADMIT_D-CADMIT_D) in (0 1));
FZCPADM2=(Abs(RADMIT_D-CADMIT_D) in (0 1 2));
CPDIS=(CDISCH_D=RDISCH_D);
FZCPDIS1=(Abs(RDISCH_D-CDISCH_D) in (0 1));
FZCPDIS2=(Abs(RDISCH_D-CDISCH_D) in (0 1 2));
CPDOB=((CDOB ne .) and (RDOB ne .)
and CDOB=RDOB);

CPINITIALS=0;
If RLINITS=3 and CLINITS=3
and trim(RNINITS3)=trim(CNINITS3)
then CPINITIALS=1; Else
If RLINITS=2 and CLINITS=2
and trim(RNINITS2)=trim(CNINITS2)
then CPINITIALS=1;
```

```
/* ---------------------------------------- */
/* 5 Match Files in decrease order preference */
/* ---------------------------------------- */
if CPHOSPID=1 and CPAGE=1 and CPGENDER=1 and
CPADM=1 and CPDIS=1 then do;
output amatch&file.; end;
else if CPHOSPID=1 and CPAGE=1 and CPGENDER=1
and CPADM=1 and FZCPDIS1=1 then do;
output bmatch&file.; end;
else if CPHOSPID=1 and CPAGE=1 and CPGENDER=1
and FZCPADM1=1 and CPDIS=1 then do;
output cmatch&file.; end;
else if CPHOSPID=1 and CPAGE=1 and CPGENDER=1
and CPADM=1 then do;
output dmatch&file.; end;
else if CPHOSPID=1 and (FZCPADM1=1 or
FZCPDIS1=1)
and CPINITIALS=1 then do;
output ematch&file.; end;
else if CPHOSPID=1 and CPDOB=1 then do;
output fmatch&file.;
end; else output unmat&File.;

Startovr: If i Gt n Then Goto Getnext; end;
   Getnext: If i Gt n Then i=1;
   If _Error_ = 1 Then _Error_ = 0;
Run;

%mend outstate;
/* Continue states [01-50] As Needed */
%OutState(01); %OutState(02); %OutState(03);
%OutState(04); %OutState(05); %OutState(06);
```

APPENDIX 8:

```
/* ---------------------------------------- */
/* Append State-Based in decreased preference */
/* ---------------------------------------- */
%Macro Set(Mtype, MPrefNo);
Data &Mtype. (label="Match Priority:
&MPrefNo.");
Length MType 4;
set /* Continue states [01-50] As Needed */
&MType.01 &MType.02 &MType.03
&MType.04 &MType.05 &MType.06
;
MTYPE=&MPrefNo.;
Run;
%Mend Set;
%Set(aMatch, 1) %Set(bMatch, 2) %Set(cMatch, 3)
%Set(dMatch, 4) %Set(eMatch, 5) %Set(fMatch, 6)
/* Can Set if Wanted: %Set(UnMat, 99)  */

/* set Prioritized DataSets*/
Data AllMtch1 (label='Matched Data w/
Replicates');
set
Work.aMatch Work.bMatch Work.cMatch
Work.dMatch Work.eMatch Work.fMatch
/* Can Set if Wanted: Work.UnMat */;
Label MTYPE="Priority or Preference of Match";
run;
```

APPENDIX 9:

```
/* ---------------------------------------- */
/* rectify replicates on REGISTRY side      */
/* replicate CLINICAL IDS [CSYSID]          */
/* means spreaders on REGISTRY side         */
/* ---------------------------------------- */
Proc sort data=WORK.AllMtch1;
by CSYSID
descending CPHOSPID descending CPAGE
descending CPGENDER descending CPADM
descending CPDIS descending CPDOB
descending CPINITIALS descending RDEATH
descending RNOTBLNK descending RRANDSCO;
run;
Title 'Replicates in the CLINICAL Dataset';
proc print data=WORK.AllMtch1;
Var CSYSID RSYSID CPID RPID CPHOSPID CPAGE
CPGENDER CPADM CPDIS CPDOB CPINITIALS RDEATH
CDEATH RNOTBLNK CNOTBLNK RRANDSCO CRANDSCO
MTYPE; Run;

Data WORK.AllMtch2
(Label='No Replicates - REGISTRY side');
Set WORK.AllMtch1;
By CSYSID;
if first.CSYSID;
Run;

proc print data=WORK.AllMtch2;
Var CSYSID RSYSID CPID RPID CPHOSPID CPAGE
CPGENDER CPADM CPDIS CPDOB CPINITIALS RDEATH
CDEATH RNOTBLNK CNOTBLNK RRANDSCO CRANDSCO
MTYPE; Run;

/* ---------------------------------------- */
/* rectify replicates on CLINICAL side      */
/* replicate REGISTRY IDS [RSYSID]          */
/* means spreaders on CLINICAL side         */
/* ---------------------------------------- */
Proc sort data=AllMtch2;
by RSYSID
descending CPHOSPID descending CPAGE
descending CPGENDER descending CPADM
descending CPDIS descending CPDOB
descending CPINITIALS descending CDEATH
descending CNOTBLNK descending CRANDSCO ;
run;
Title 'Replicates in the REGISTRY Dataset';
proc print data=AllMtch2;
Var RSYSID CSYSID RPID CPID CPHOSPID CPAGE
CPGENDER CPADM CPDIS CPDOB CPINITIALS CDEATH
CNOTBLNK RRANDSCO CRANDSCO MTYPE;
Run;

Data WORK.AllMtch3
(Label='No Replicates on REGISTRY/CLINICAL
sides');
Set AllMtch2;
By RSYSID;
if first.RSYSID;
Run;

Proc Format; Value MType
1='5Key Match' 2='5Key Fuzz Disch[1 Day] Match'
3='5Key Fuzz Arr[1 Day] Match' 4='4Key Match'
5='3Key Match' 6='2Key Match'; Run;

Title 'Matched CLINICAL/REGISTRY File';
proc print data=WORK.AllMtch3;
Var RSYSID CSYSID RPID CPID CPHOSPID CPAGE
CPGENDER CPADM CPDIS CPDOB CPINITIALS CDEATH
CNOTBLNK RRANDSCO CRANDSCO MTYPE;
Format MType Mtype.; Run;

Title 'Agreement Between Two Sources of Data';
Proc Freq Data=AllMtch3; tables CHXDIAB*RHXDIAB
CDEATH*RDEATH / Agree;
Run;
Proc Contents Data=WORK.AllMtch3; Run;
```