Create CHARS revisit file

Eric Ossiander
November 25, 2014

This describes how I created a CHARS revisit file for 2009–2013 CHARS records, including the observation records.

I used the following fields in the linking process:

birth date
name
last 4 digits of SSN
sex
zipcode of residence
county of residence
Hispanic ethnicity
race
state of residence

I used the RecordLinkage package in R for most of the linking. In all of the record linking that I did in R, I used birth date as a blocking field (i.e. I required that the birth date on one hospitalization record match the birth date on another in order to evaluate them as a link). First, I computed a probabilistic linkage weight for each record pair. Second, I used a machine learning algorithm to predict which record pairs were links. (This required me to manually code a training set, to create a statistical model for predicting links.) Then I manually reviewed all of the record pairs which were predicted not to be a link by the machine learning algorithm, but which had a high probabilistic weight, and all record pairs which were predicted to be a link, but had a low weight. I also used a SAS program to compute a probabilistic linkage weight for all record pairs (i.e. not blocking on birth date), and manually reviewed all of the record pairs that had a high probabilistic weight in which birth dates did not match. I combined the three linked sets (the machine-linked pairs, the manual review of the machine linking, and the manual coding of the non-birth date matching pairs). Then I checked for hospitalization records that were in more than one link set, and manually adjudicated those links.

Process

Data items

The items that help identify people in the CHARS file, and therefore can be used for linking are:

name
dob
social security number (last 4 digits)
age
sex
race, ethnicity
hospital code
place of residence (zipcode and county)

All of these items are in the confidential files (names `chr_r2012.sas7bdat`, etc).

It looks like names are present on a few records in 2008, and on almost all records in 2009 and following years. In 2007 and earlier (and in the 2008 records that don't have names), first two

letters of names are on the files. Birthdates are apparently on all files. SSN is two-thirds missing in 2008, better in 2009, and about 20% missing in 2012. It is almost entirely missing in 2007. Race is reported on about 40% of 2008 records, very few before that year, and almost all records after that year.

Therefore, I can link only the 2009–2013 files.

Create CHARS file

Listing 1: Create CHARS file for linking

```
libname chars 'c:\data\chars';
data clink(keep=seq_no_enc staytype adm_date age country countyres dis_date dob firstname
                ssnL4 hispanic hospital lastname miname race_ami race_asi race_blk
                race_haw race_wht sex stateres status zipcode zipplus4
                lastname_sdx firstname_sdx suffix);
   length firstname lastname $ 20 suffix $ 4 lastname_sdx firstname_sdx $ 4;
   set chars.chr_r2009(rename=(SSN=ssnL4 firstname=firsttemp lastname=lasttemp))
       chars.chr_r2010(rename=(SSN=ssnL4 firstname=firsttemp lastname=lasttemp))
       chars.chr_r2011(rename=(SSN=ssnL4 firstname=firsttemp lastname=lasttemp))
       chars.chr_r2012(rename=(SSN=ssnL4 firstname=firsttemp lastname=lasttemp))
       chars.chr_r2013(rename=(SSN=ssnL4 firstname=firsttemp lastname=lasttemp))
       chars.chro_r2009(rename=(SSN=ssnL4 firstname=firsttemp lastname=lasttemp))
       chars.chro_r2010(rename=(SSN=ssnL4 firstname=firsttemp lastname=lasttemp))
       chars.chro_r2011(rename=(SSN=ssnL4 firstname=firsttemp lastname=lasttemp))
       chars.chro_r2012(rename=(SSN=ssnL4 firstname=firsttemp lastname=lasttemp))
       chars.chro_r2013(rename=(SSN=ssnL4 firstname=firsttemp lastname=lasttemp));
   firsttemp = compress(firsttemp," ''-_,.&");
   if (substr(firsttemp,1,4) = 'BABY') or (firsttemp in ('A','B','BOY','GIRL','BA','BB',
        'BG','BBABY','G','GA','GB','NB','BA-','BB-','BG-')) then firsttemp = '';
   firstname = firsttemp;
   if race_ami in ('U','R') then race_ami = '';
   if race_asi in ('U','R') then race_asi = '';
   if race_blk in ('U','R') then race_blk = '';
   if race_haw in ('U','R') then race_haw = '';
   if race_wht in ('U','R') then race_wht = '';
   if hispanic in ('U','R') then hispanic = '';
/*
remove the suffixes II, III, IV, V, VI, VII, VIII, ESQ, JR, and SR
from lastnames and place them in a separate suffix field.
Used with UB04 data.
*/
   if _N_ = 1 then do;
        retain __re __reIII;
        pattern = "/( II| III| IV| V| VI| VII| VIII| ESQ|.JR|.SR)$/i";
        __re = prxparse(pattern);
        __reIII = prxparse('/III$/');
   end;
   lasttemp = translate(lasttemp,' ','.,');
   call prxsubstr(__re, TRIM(lasttemp), position, length);
   if position ^= 0 then do;
        suffix    = substr(lasttemp, position + 1, length - 1);
        lasttemp2 = substr(lasttemp, 1, position - 1);
   end;
   else lasttemp2 = lasttemp;

   lastname  = compress(lasttemp2," ''-_,.&");
   lastname_sdx  = soundex(lastname);
   firstname_sdx = soundex(firstname);
run;
```

Compute an *ad hoc* linking weight

I will use the RecordLinkage package in R for most of the linking. The use of that package will require blocking on birthdate, but there are sure to be some true matches in which birthdates don't match. Therefore, I will write a SAS program to compute an *ad hoc* probabilistic weight for each pair of records in the 2009-2012 CHARS files, and manually review all the record pairs that have a high weight on which the birthdates don't match.

Fields I will use, and the points I will give for a matching value are:

item	match	different
birthdate	20	-20
firstname	10 (2 for soundex match)	-10
lastname	15 (4 for soundex)	-15
middleinit	2	-3
sex	2	-20
zipcode	3	-2
county	3	-5
ssnL4	15	-10
race_ami	5	-5
race_asi	5	-5
race_blk	5	-5
race_haw	5	-5
race_wht	5	-5
hispanic	5	-5
statecode	1	-5
hospital	5	-5

Listing 2: compute test link scores

```
/*
Sort the file first so that I can be sure that when it is divided
into parts at different times and on different computers, the parts
are divided correctly.

For each record, I will evaluate its similarity with each of the other records
by computing a score using the points described above. In the output dataset,
I will keep records that have a score of at least 1.
Maximum score is ?.
*/
proc sort data=clink;
    by seq_no_enc staytype;
run;
data clinkcopy;
   set clink;
   rename
     adm_date       = c_adm_date
     age            = c_age
     country        = c_country
     countyres      = c_countyres
     dis_date       = c_dis_date
     dob            = c_dob
     firstname      = c_firstname
     firstname_sdx  = c_firstname_sdx
     hispanic       = c_hispanic
     hospital       = c_hospital
     lastname       = c_lastname
     lastname_sdx   = c_lastname_sdx
     miname         = c_miname
     race_ami       = c_race_ami
     race_asi       = c_race_asi
     race_blk       = c_race_blk
```

```
      race_haw       = c_race_haw
      race_wht       = c_race_wht
      seq_no_enc     = c_seq_no_enc
      sex            = c_sex
      ssnl4          = c_ssnl4
      stateres       = c_stateres
      status         = c_status
      staytype       = c_staytype
      suffix         = c_suffix
      zipcode        = c_zipcode
      zipplus4       = c_zipplus4
      ;
run;

%macro linker(part,first,last);
/*
require match on fname or lname first, and discard birthdate matches.
*/
data chars.adhocweights_part&part.;
   set clink(firstobs=&first obs=&last);
   if _n_/1000 = round(_n_/1000) then put _n_=;
   do i = 1 to 3713485;
      set clinkcopy point=i;
      if (firstname ne c_firstname and lastname ne c_lastname) or
          dob = c_dob then score = 0;
      else score =
         (countyres     = c_countyres     and countyres   ne '')*3  +
         (countyres     ne c_countyres    )*(-5) +
         (month(dob)    = month(c_dob)    and dob          ne .)*5   +
         (month(dob)    ne month(c_dob)   )*(-4)  +
         (day(dob)      = day(c_dob)      and dob          ne .)*5   +
         (day(dob)      ne day(c_dob)     )*(-4)   +
         (year(dob)     = year(c_dob)     and dob          ne .)*3   +
         (year(dob)     ne year(c_dob)    )*(-4)   +
         (firstname     = c_firstname     and firstname    ne '')*10 +
         (firstname     ne c_firstname    )*(-10) +
         (hispanic      = c_hispanic      and hispanic     ne '')*2  +
         (hispanic      ne c_hispanic     )*(-2) +
         (lastname      = c_lastname      and lastname     ne '')*15  +
         (lastname      ne c_lastname     )*(-15) +
         (miname        = c_miname        and miname       ne '')*2  +
         (miname        ne c_miname       )*(-3) +
         (race_ami      = c_race_ami      and race_ami     ne '')*1  +
         (race_ami      ne c_race_ami     )*(-2) +
         (race_asi      = c_race_asi      and race_asi     ne '')*1  +
         (race_asi      ne c_race_asi     )*(-2) +
         (race_blk      = c_race_blk      and race_blk     ne '')*1  +
         (race_blk      ne c_race_blk     )*(-2) +
         (race_haw      = c_race_haw      and race_haw     ne '')*1  +
         (race_haw      ne c_race_haw     )*(-2) +
         (race_wht      = c_race_wht      and race_wht     ne '')*1  +
         (race_wht      ne c_race_wht     )*(-2) +
         (sex           = c_sex           and sex          ne '')*2  +
         (sex           ne c_sex          )*(-20) +
         (zipcode       = c_zipcode       and zipcode      not in ('','99999'))*3  +
         (zipcode       ne c_zipcode      )*(-2) +
         (zipcode = c_zipcode and zipcode not in ('','99999') and zipplus4 = c_zipplus4 and
                    zipplus4 not in ('','9999'))*15  +
         (firstname_sdx = c_firstname_sdx and firstname_sdx ne '')*4  +
         (firstname_sdx ne c_firstname_sdx)*(-8) +
         (lastname_sdx  = c_lastname_sdx  and lastname_sdx ne '')*6 +
         (lastname_sdx  ne c_lastname_sdx )*(-10) +
         (ssnl4         = c_ssnl4         and ssnl4 not in ('','9999'))*15 +
         (ssnl4         ne c_ssnl4        )*(-10) +
         (stateres      = c_stateres      and stateres     ne '')*1 +
         (stateres      ne c_stateres     )*(-5) +
         (hospital      = c_hospital      and hospital     ne '')*7 +
         (hospital      ne c_hospital     )*(-5)
```

```
        ;
      if score ge 1 then output;
      *output;
      end;
run;
%mend;

%linker(1,1,560000);
%linker(2,560001,1120000);
%linker(3,1120001,1680000);
%linker(4,1680001,2240000);

home:
%linker(5,2240001,2990000);
%linker(6,2990001,3713485);


data chars.revisit_adhocweights;
    set chars.adhocweights_part1 chars.adhocweights_part2 chars.adhocweights_part3
        chars.adhocweights_part4 chars.adhocweights_part5 chars.adhocweights_part6;
run;

proc freq data=chars.revisit_adhocweights;
    where dob ne c_dob;
    tables score;
run;
```

Prepare files for linking

Prepare files to write to R. I convert strings that indicate missing values (such as 'U' for the race codes, and '9999' for SSN) to blanks so that the linking routines won't think these represent good information.

Listing 3: CHARS files to csv for R

```
/*
the length statements are to ensure fields are in a consistent order
when I read them into R, and the fields are ordered for easiest use
during the classification of the training set.
*/
proc sort data=clink;
    by seq_no_enc staytype;
run;
data clink2;
   keep seq_no_enc staytype countyres dob firstname hispanic lastname miname race_ami
        race_asi race_blk race_haw race_wht sex ssnL4 stateres
        zipcode zipplus4 hospital suffix;
   length seq_no_enc $ 10 staytype $ 1 dob 8 firstname $ 20 miname $ 1 lastname $ 20
          suffix $ 4 ssnL4 $ 4 sex $ 1 zipcode $ 5 zipplus4 $ 4 countyres $ 2 hospital $ 4
          hispanic race_wht race_blk race_ami race_asi
          race_haw $ 1 stateres $ 2 ;
   set clink;
   if sex = 'U' then sex = '';
   if statecode = '99' then statecode = '';
   if zipcode = '99999' then zipcode = '';
   if zipplus4 in ('0000','9999') then zipplus4 = '';
   if ssnL4 = '9999' then ssnL4 = '';
run;
proc export data=clink2
   outfile = "c:\data\chars\charslink2009_2013.txt"
   dbms = csv
   replace
   ;
run;
```

Perform linking

Now read the files into R.

I created a new string comparator function to use with the RecordLinkage package, to detect if the value of last name on one record is contained within the value of last name on another record. This will help with people who get married and combine last names, or with people who have many names, and get their last names recored inconsistently—sometimes as one name, and sometimes as two names.

When I perform the manual matching for the training set, and when I do the manual review of selected machine-linked pairs, the standard that I use to declare a pair a match is that I think it is almost certain to be a match. If I think there is a plausible scenario by which a pair is not a match, then I do not score it as a match. Some examples:

- A pair in which each member was born during the data collection period (2009-2013), in which the first names and SSNs are blank, but all other information indicates that the pair is related (same lastname, zipcode, etc). These could be twins, so I do not score them a match. (Even though there are many more singletons than twins born, it is not common for singletons to have more than one hospital admission, so two records for related babies are not unlikely to be for twins.)
- A pair in which first name, middle initial, and SSN are the same, but last names are different. These are likely one person who changed his or her last name, and I generally score this as a match. There are about 200 people for each birthdate, and 9,999 different last 4 digits of SSN, so it is unlikely (although not impossible) that two different people would have the same birthdate, first name, middle initial, and SSN. The fact that I see this pattern almost exclusively among adult women supports my conclusion that this pattern results from a name change.
- A pair in which first name, middle initial, and last name are all the same, but SSNs are different. If the name is common (say William Smith or Jose Garcia or Hong Nguyen) I would probably score this as not a pair; otherwise I would score it as a pair.

Here is the new string comparator function:

```
> NameContains <- function(str1, str2){
+ # Function to compare two strings.
+ # Returns:
+ #   1 if the shorter string is contained in the other
+ #   0 otherwise
+ # if the shorter string is longer than 6 characters, then only
+ # the first 6 characters are used.
+   score     <- rep(NA,length(str1))
+   longname  <- rep(NA,length(str1))
+   shortname <- rep(NA,length(str1))
+   for(i in 1:length(str1)){
+     if(str1[i]=='' | str2[i]=='' | is.na(str1[i]) | is.na(str2[i])) score[i] <- NA else{
+       if(str1[i] == str2[i]) score[i] <- 1 else {
+         if(nchar(str1[i]) >= nchar(str2[i])){longname[i] <- str1[i];
+           shortname[i] <- str2[i]} else {longname[i] <- str2[i]; shortname[i] <- str1[i]}
+         if(nchar(shortname[i]) < 3) score[i] <- 0 else {
+           if(nchar(shortname[i]) > 6)shortname[i] <- substr(shortname[i],1,6)
+           score[i] <- if(grepl(shortname[i],longname[i]))1 else 0
```

```
+     }}}}
+   return(score)
+ }
```

```
%<<>>=
library(RecordLinkage)
clink.big <- read.csv("../../../data/chars/charslink2009_2013.txt",
                      colClasses=c(rep("character",20)),
                      col.names=c("seq_no_enc","staytype","dob","firstname","miname","lastname",
                      "suffix","ssnL4","sex","zipcode","zipplus4","county","facility","hispanic",
                      "race.wht","race.blk","race.ami","race.asi","race.haw","statecode"))
clink <- clink.big[,c(1:13)]
clink$firstname.sdx <- soundex(clink$firstname)
clink$lastname.sdx  <- soundex(clink$lastname)
clink$subname       <- clink$lastname
# if ssnL4 is '1111' for babies, convert it to blank
clink$ssnL4 <- ifelse(substr(clink$dob,7,10)>2008&clink$ssnL4=='1111','',clink$ssnL4)

# I will try feb, mar and apr together to build a predictive model
clink.febtoapr <- clink[substr(clink$dob,1,2)=="02" | substr(clink$dob,1,2)=="03" | substr(clink$dob,1,2)=="04",]
clink.febtoapr.pairs <- compare.dedup(clink.febtoapr,blockfld=c(3),exclude=c(1,2),
                                      strcmp=c(16),strcmpfun=NameContains)
clink.febtoapr.pairs.fsWt <- fsWeights(clink.febtoapr.pairs)
save(clink.febtoapr.pairs.fsWt,file='febtoaprpairs.RData')
rm(clink.febtoapr.pairs)

clink.febtoapr.train10 <- getMinimalTrain(clink.febtoapr.pairs.fsWt,nEx=10)
save(clink.febtoapr.train10,file="clink.febtoapr.train10.RData")
clink.febtoapr.train10 <- editMatch(clink.febtoapr.train10)

model.revisit.bag <- trainSupv(clink.febtoapr.train10,method='bagging')
save(model.revisit.bag,file="model.revisit.RData")

load(file='model.revisit.RData')

# January
clink.jan <- clink[substr(clink$dob,1,2)=="01",]
clink.jan.pairs <- compare.dedup(clink.jan,blockfld=c(3),exclude=c(1,2),
                                 strcmp=c(16),strcmpfun=NameContains)
clink.jan.pairs.fsWt <- fsWeights(clink.jan.pairs)
rm(clink.jan.pairs)
result.jan.bag <- classifySupv(model.revisit.bag,newdata=clink.jan.pairs.fsWt)
save(result.jan.bag,clink.jan,clink.jan.pairs.fsWt,file="janpairs.RData")
save(result.jan.bag,file="result.jan.RData")
rm(list=c('clink.jan','clink.jan.pairs.fsWt','result.jan.bag'))

# February
clink.feb <- clink[substr(clink$dob,1,2)=="02",]
clink.feb.pairs <- compare.dedup(clink.feb,blockfld=c(3),exclude=c(1,2),
                                 strcmp=c(16),strcmpfun=NameContains)
clink.feb.pairs.fsWt <- fsWeights(clink.feb.pairs)
rm(clink.feb.pairs)
result.feb.bag <- classifySupv(model.revisit.bag,newdata=clink.feb.pairs.fsWt)
save(result.feb.bag,clink.feb,clink.feb.pairs.fsWt,file="febpairs.RData")
save(result.feb.bag,file="result.feb.RData")
rm(list=c('clink.feb','clink.feb.pairs.fsWt','result.feb.bag'))

# March
clink.mar <- clink[substr(clink$dob,1,2)=="03",]
clink.mar.pairs <- compare.dedup(clink.mar,blockfld=c(3),exclude=c(1,2),
                                 strcmp=c(16),strcmpfun=NameContains)
clink.mar.pairs.fsWt <- fsWeights(clink.mar.pairs)
rm(clink.mar.pairs)
result.mar.bag <- classifySupv(model.revisit.bag,newdata=clink.mar.pairs.fsWt)
save(result.mar.bag,clink.mar,clink.mar.pairs.fsWt,file="marpairs.RData")
save(result.mar.bag,file="result.mar.RData")
rm(list=c('clink.mar','clink.mar.pairs.fsWt','result.mar.bag'))
```

```
# April
clink.apr <- clink[substr(clink$dob,1,2)=="04",]
clink.apr.pairs <- compare.dedup(clink.apr,blockfld=c(3),exclude=c(1,2),
                                 strcmp=c(16),strcmpfun=NameContains)
clink.apr.pairs.fsWt <- fsWeights(clink.apr.pairs)
rm(clink.apr.pairs)
result.apr.bag <- classifySupv(model.revisit.bag,newdata=clink.apr.pairs.fsWt)
save(result.apr.bag,clink.apr,clink.apr.pairs.fsWt,file="aprpairs.RData")
save(result.apr.bag,file="result.apr.RData")
rm(list=c('clink.apr','clink.apr.pairs.fsWt','result.apr.bag'))

# May
clink.may <- clink[substr(clink$dob,1,2)=="05",]
clink.may.pairs <- compare.dedup(clink.may,blockfld=c(3),exclude=c(1,2),
                                 strcmp=c(16),strcmpfun=NameContains)
clink.may.pairs.fsWt <- fsWeights(clink.may.pairs)
rm(clink.may.pairs)
result.may.bag <- classifySupv(model.revisit.bag,newdata=clink.may.pairs.fsWt)
save(result.may.bag,clink.may,clink.may.pairs.fsWt,file="maypairs.RData")
save(result.may.bag,file="result.may.RData")
rm(list=c('clink.may','clink.may.pairs.fsWt','result.may.bag'))

# June
clink.jun <- clink[substr(clink$dob,1,2)=="06",]
clink.jun.pairs <- compare.dedup(clink.jun,blockfld=c(3),exclude=c(1,2),
                                 strcmp=c(16),strcmpfun=NameContains)
clink.jun.pairs.fsWt <- fsWeights(clink.jun.pairs)
rm(clink.jun.pairs)
result.jun.bag <- classifySupv(model.revisit.bag,newdata=clink.jun.pairs.fsWt)
save(result.jun.bag,clink.jun,clink.jun.pairs.fsWt,file="junpairs.RData")
save(result.jun.bag,file="result.jun.RData")
rm(list=c('clink.jun','clink.jun.pairs.fsWt','result.jun.bag'))

# July
clink.jul <- clink[substr(clink$dob,1,2)=="07",]
clink.jul.pairs <- compare.dedup(clink.jul,blockfld=c(3),exclude=c(1,2),
                                 strcmp=c(16),strcmpfun=NameContains)
clink.jul.pairs.fsWt <- fsWeights(clink.jul.pairs)
rm(clink.jul.pairs)
result.jul.bag <- classifySupv(model.revisit.bag,newdata=clink.jul.pairs.fsWt)
save(result.jul.bag,clink.jul,clink.jul.pairs.fsWt,file="julpairs.RData")
save(result.jul.bag,file="result.jul.RData")
rm(list=c('clink.jul','clink.jul.pairs.fsWt','result.jul.bag'))

# August
clink.aug <- clink[substr(clink$dob,1,2)=="08",]
clink.aug.pairs <- compare.dedup(clink.aug,blockfld=c(3),exclude=c(1,2),
                                 strcmp=c(16),strcmpfun=NameContains)
clink.aug.pairs.fsWt <- fsWeights(clink.aug.pairs)
rm(clink.aug.pairs)
result.aug.bag <- classifySupv(model.revisit.bag,newdata=clink.aug.pairs.fsWt)
save(result.aug.bag,clink.aug,clink.aug.pairs.fsWt,file="augpairs.RData")
save(result.aug.bag,file="result.aug.RData")
rm(list=c('clink.aug','clink.aug.pairs.fsWt','result.aug.bag'))

# September
clink.sep <- clink[substr(clink$dob,1,2)=="09",]
clink.sep.pairs <- compare.dedup(clink.sep,blockfld=c(3),exclude=c(1,2),
                                 strcmp=c(16),strcmpfun=NameContains)
clink.sep.pairs.fsWt <- fsWeights(clink.sep.pairs)
rm(clink.sep.pairs)
result.sep.bag <- classifySupv(model.revisit.bag,newdata=clink.sep.pairs.fsWt)
save(result.sep.bag,clink.sep,clink.sep.pairs.fsWt,file="seppairs.RData")
save(result.sep.bag,file="result.sep.RData")
rm(list=c('clink.sep','clink.sep.pairs.fsWt','result.sep.bag'))

# October
```

```
clink.oct <- clink[substr(clink$dob,1,2)=="10",]
clink.oct.pairs <- compare.dedup(clink.oct,blockfld=c(3),exclude=c(1,2),
                                 strcmp=c(16),strcmpfun=NameContains)
clink.oct.pairs.fsWt <- fsWeights(clink.oct.pairs)
rm(clink.oct.pairs)
result.oct.bag <- classifySupv(model.revisit.bag,newdata=clink.oct.pairs.fsWt)
save(result.oct.bag,clink.oct,clink.oct.pairs.fsWt,file="octpairs.RData")
save(result.oct.bag,file="result.oct.RData")
rm(list=c('clink.oct','clink.oct.pairs.fsWt','result.oct.bag'))

# November
clink.nov <- clink[substr(clink$dob,1,2)=="11",]
clink.nov.pairs <- compare.dedup(clink.nov,blockfld=c(3),exclude=c(1,2),
                                 strcmp=c(16),strcmpfun=NameContains)
clink.nov.pairs.fsWt <- fsWeights(clink.nov.pairs)
rm(clink.nov.pairs)
result.nov.bag <- classifySupv(model.revisit.bag,newdata=clink.nov.pairs.fsWt)
save(result.nov.bag,clink.nov,clink.nov.pairs.fsWt,file="novpairs.RData")
save(result.nov.bag,file="result.nov.RData")
rm(list=c('clink.nov','clink.nov.pairs.fsWt','result.nov.bag'))

# December
clink.dec <- clink[substr(clink$dob,1,2)=="12",]
clink.dec.pairs <- compare.dedup(clink.dec,blockfld=c(3),exclude=c(1,2),
                                 strcmp=c(16),strcmpfun=NameContains)
clink.dec.pairs.fsWt <- fsWeights(clink.dec.pairs)
rm(clink.dec.pairs)
result.dec.bag <- classifySupv(model.revisit.bag,newdata=clink.dec.pairs.fsWt)
save(result.dec.bag,clink.dec,clink.dec.pairs.fsWt,file="decpairs.RData")
save(result.dec.bag,file="result.dec.RData")
rm(list=c('clink.dec','clink.dec.pairs.fsWt','result.dec.bag'))

%@
```

Find the range of scores for predicted links and predicted non-links:

```
%<<>>=
load(file='result.jan.RData')
load(file='result.feb.RData')
load(file='result.mar.RData')
load(file='result.apr.RData')
load(file='result.may.RData')
load(file='result.jun.RData')
load(file='result.jul.RData')
load(file='result.aug.RData')
load(file='result.sep.RData')
load(file='result.oct.RData')
load(file='result.nov.RData')
load(file='result.dec.RData')

ranges <- data.frame(Nmin=rep(NA,12),Nmax=rep(NA,12),Lmin=rep(NA,12),Lmax=rep(NA,12))
ranges[1,c(1,2)] <- with(result.jan.bag[result.jan.bag$prediction=='N'],range(Wdata))
ranges[1,c(3,4)] <- with(result.jan.bag[result.jan.bag$prediction=='L'],range(Wdata))
ranges[2,c(1,2)] <- with(result.feb.bag[result.feb.bag$prediction=='N'],range(Wdata))
ranges[2,c(3,4)] <- with(result.feb.bag[result.feb.bag$prediction=='L'],range(Wdata))
ranges[3,c(1,2)] <- with(result.mar.bag[result.mar.bag$prediction=='N'],range(Wdata))
ranges[3,c(3,4)] <- with(result.mar.bag[result.mar.bag$prediction=='L'],range(Wdata))
ranges[4,c(1,2)] <- with(result.apr.bag[result.apr.bag$prediction=='N'],range(Wdata))
ranges[4,c(3,4)] <- with(result.apr.bag[result.apr.bag$prediction=='L'],range(Wdata))
ranges[5,c(1,2)] <- with(result.may.bag[result.may.bag$prediction=='N'],range(Wdata))
ranges[5,c(3,4)] <- with(result.may.bag[result.may.bag$prediction=='L'],range(Wdata))
ranges[6,c(1,2)] <- with(result.jun.bag[result.jun.bag$prediction=='N'],range(Wdata))
ranges[6,c(3,4)] <- with(result.jun.bag[result.jun.bag$prediction=='L'],range(Wdata))

ranges[7,c(1,2)]  <- range(result.jul.bag$Wdata[result.jul.bag$prediction=='N'])
ranges[7,c(3,4)]  <- range(result.jul.bag$Wdata[result.jul.bag$prediction=='L'])
```

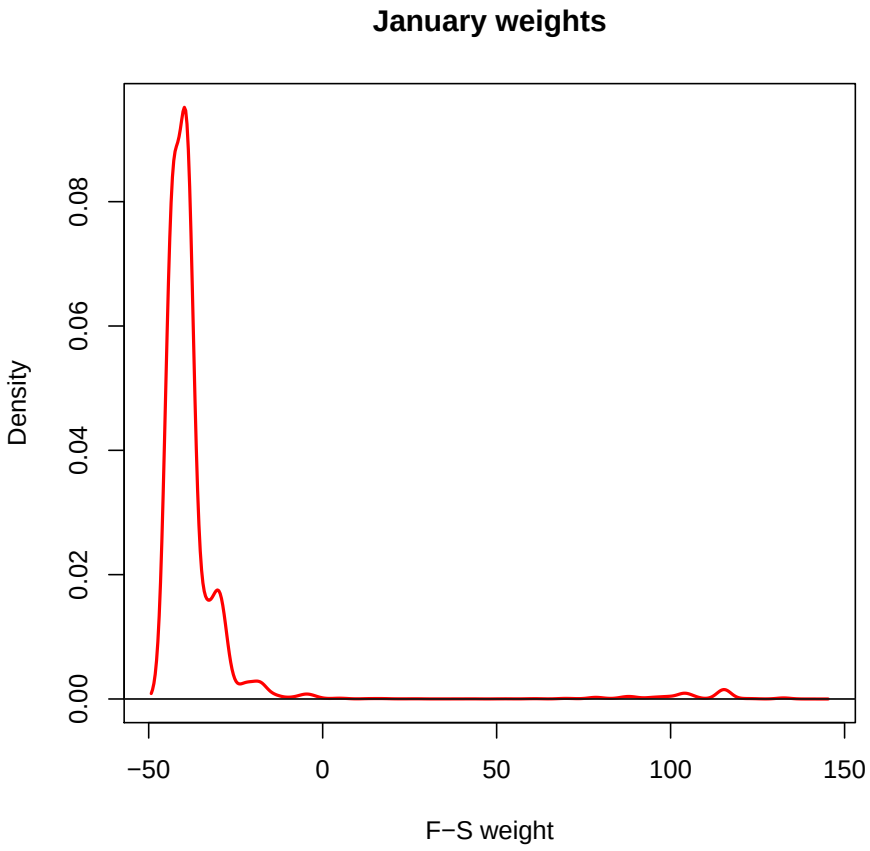

Figure 1: Density of the Fellegi-Sunter weights for record pairs of patients born in January.

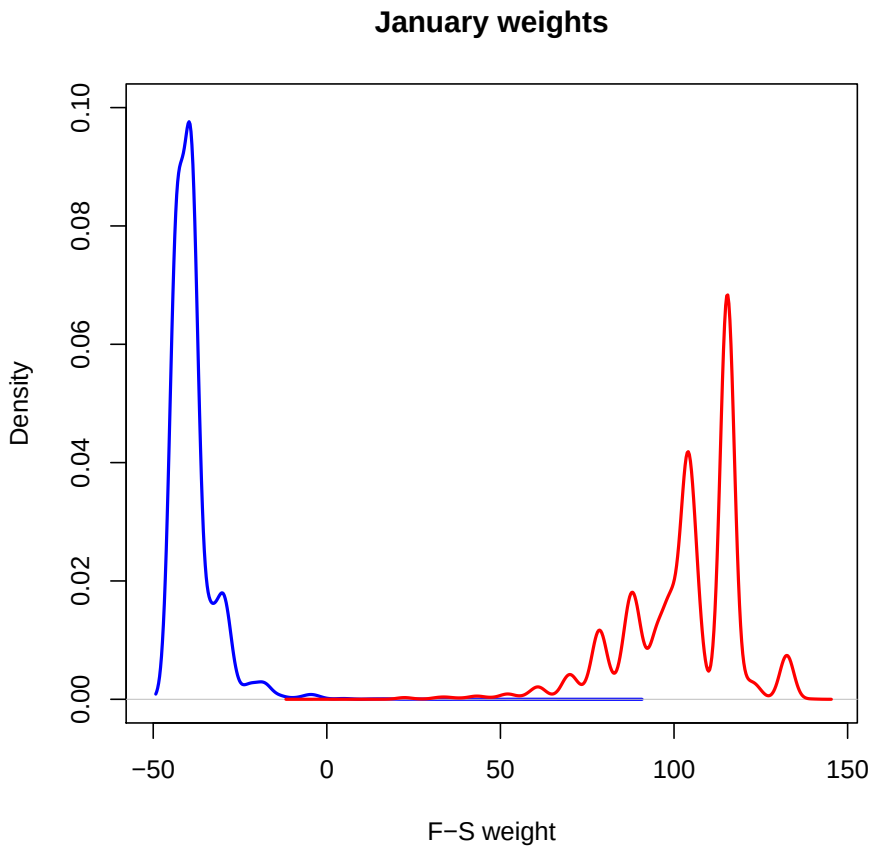

Figure 2: Fellegi-Sunter weights for record pairs of patients born in January, by predicted link status. Each line is a separate density plot (meaning they are not to scale).

```
ranges[8,c(1,2)]  <- range(result.aug.bag$Wdata[result.aug.bag$prediction=='N'])
ranges[8,c(3,4)]  <- range(result.aug.bag$Wdata[result.aug.bag$prediction=='L'])
ranges[9,c(1,2)]  <- range(result.sep.bag$Wdata[result.sep.bag$prediction=='N'])
ranges[9,c(3,4)]  <- range(result.sep.bag$Wdata[result.sep.bag$prediction=='L'])
ranges[10,c(1,2)] <- range(result.oct.bag$Wdata[result.oct.bag$prediction=='N'])
ranges[10,c(3,4)] <- range(result.oct.bag$Wdata[result.oct.bag$prediction=='L'])
ranges[11,c(1,2)] <- range(result.nov.bag$Wdata[result.nov.bag$prediction=='N'])
ranges[11,c(3,4)] <- range(result.nov.bag$Wdata[result.nov.bag$prediction=='L'])
ranges[12,c(1,2)] <- range(result.dec.bag$Wdata[result.dec.bag$prediction=='N'])
ranges[12,c(3,4)] <- range(result.dec.bag$Wdata[result.dec.bag$prediction=='L'])

max(ranges[,2])
min(ranges[,3])

countsbymonth <- rep(NA,12)
countsbymonth[1] <- length(result.jan.bag$Wdata)
countsbymonth[2] <- length(result.feb.bag$Wdata)
countsbymonth[3] <- length(result.mar.bag$Wdata)
countsbymonth[4] <- length(result.apr.bag$Wdata)
countsbymonth[5] <- length(result.may.bag$Wdata)
countsbymonth[6] <- length(result.jun.bag$Wdata)

countsbymonth[7] <- length(result.jul.bag$Wdata)
countsbymonth[8] <- length(result.aug.bag$Wdata)
countsbymonth[9] <- length(result.sep.bag$Wdata)
countsbymonth[10] <- length(result.oct.bag$Wdata)
countsbymonth[11] <- length(result.nov.bag$Wdata)
countsbymonth[12] <- length(result.dec.bag$Wdata)

%@
```

The maximum weight among predicted non-links is 67.9, and the minimum weights among predicted links is -6.5. I will assess the accuracy of links in 10-point intervals between -10 and 70.

```
%<<>>=
id1.all.part1   <- c(result.jan.bag$pairs$id1,result.feb.bag$pairs$id1,result.mar.bag$pairs$id1,
               result.apr.bag$pairs$id1,result.may.bag$pairs$id1,result.jun.bag$pairs$id1)
id2.all.part1   <- c(result.jan.bag$pairs$id2,result.feb.bag$pairs$id2,result.mar.bag$pairs$id2,
               result.apr.bag$pairs$id2,result.may.bag$pairs$id2,
               result.jun.bag$pairs$id2)
Wdata.all.part1 <- c(result.jan.bag$Wdata,result.feb.bag$Wdata,result.mar.bag$Wdata,
                     result.apr.bag$Wdata,result.may.bag$Wdata,result.jun.bag$Wdata)
prediction.all.part1 <- unlist(list(result.jan.bag$prediction,result.feb.bag$prediction,
                           result.mar.bag$prediction,result.apr.bag$prediction,
                           result.may.bag$prediction,result.jun.bag$prediction))
month.part1 <- c(rep('01',length(result.jan.bag$Wdata)),rep('02',length(result.feb.bag$Wdata)),
          rep('03',length(result.mar.bag$Wdata)),rep('04',length(result.apr.bag$Wdata)),
          rep('05',length(result.may.bag$Wdata)),rep('06',length(result.jun.bag$Wdata)))

id1.all.part2   <- c(result.jul.bag$pairs$id1,result.aug.bag$pairs$id1,result.sep.bag$pairs$id1,
                     result.oct.bag$pairs$id1,result.nov.bag$pairs$id1,result.dec.bag$pairs$id1)
id2.all.part2   <- c(result.jul.bag$pairs$id2,result.aug.bag$pairs$id2,
                     result.sep.bag$pairs$id2,result.oct.bag$pairs$id2,
                     result.nov.bag$pairs$id2,result.dec.bag$pairs$id2)
Wdata.all.part2 <- c(result.jul.bag$Wdata,result.aug.bag$Wdata,result.sep.bag$Wdata,
                     result.oct.bag$Wdata,result.nov.bag$Wdata,result.dec.bag$Wdata)
prediction.all.part2 <- unlist(list(result.jul.bag$prediction,result.aug.bag$prediction,
                           result.sep.bag$prediction,result.oct.bag$prediction,
                           result.nov.bag$prediction,result.dec.bag$prediction))
month.part2 <- c(rep('07',length(result.jul.bag$Wdata)),rep('08',length(result.aug.bag$Wdata)),
                 rep('09',length(result.sep.bag$Wdata)),rep('10',length(result.oct.bag$Wdata)),
                 rep('11',length(result.nov.bag$Wdata)),rep('12',length(result.dec.bag$Wdata)))

wp <- data.frame(id1.all=c(id1.all.part1,id1.all.part2),id2.all=c(id2.all.part1,id2.all.part2),
                 Wdata.all=c(Wdata.all.part1,Wdata.all.part2),
                 prediction.all=unlist(list(prediction.all.part1,prediction.all.part2)),
```

```
                month=c(month.part1,month.part2))

wp$prediction.all <- unlist(list(prediction.all.part1,prediction.all.part2))

tab.nonlink<-table(cut(wp$Wdata.all[wp$prediction.all=='N'],
                  breaks=c(-50,-10,0,10,20,30,40,50,60,70,95,155)))
tab.link<-table(cut(wp$Wdata.all[wp$prediction.all=='L'],
                  breaks=c(-50,-10,0,10,20,30,40,50,60,70,95,155)))
xtable(cbind(tab.nonlink,tab.link))
%@
```

Table 1: Frequency table of Fellegi-Sunter weights by machine-predicted links.

FS weight	predicted non-link	predicted link	total pairs
(-50,-10]	240715759	0	240715759
(-10,0]	1326114	259	1326373
(0,10]	169970	220	170190
(10,20]	132641	476	133117
(20,30]	29981	6397	36378
(30,40]	9254	13068	22322
(40,50]	9465	23554	33019
(50,60]	9119	41449	50568
(60,70]	1684	133590	135274
(70,95]	347	1262175	1262522
(95,155]	0	3955116	3955116
total	242404334	5436304	247840638

I will review all the links with a weight of 20 or less, and all the non-links with a weight of 60 or more. From the other 10-point intervals, I will select a large enough sample to estimate the machine-linking accuracy to within 1 percentage point (by this I mean a 95% CI that is no more than 2 percentage points wide). For the purpose of sample size estimation, I assume the prediction accuracy to be 95% for both predicted links and non-links with a weight between 20 and 60, 99% for predicted links with a weight higher than 40, and 99% for predicted non-links with a weight lower than 20.

The formula for calculating sample size is:

$$n = \frac{p(1-p)}{0.005^2}$$

where p is the estimated prediction accuracy. The estimated required sample size is 400 for intervals in which the assumed linking accuracy is 99%, and 1,900 for intervals in which the assumed accuracy is 95%. For the 4 intervals where the FS weight is between 20 and 60, I will draw a random sample of 1,900 of all the records, without stratifying on predicted link status. For the other intervals, I am already planning to review all the links with weights below 20, and all the non-links with weights above 60, so for those intervals, I will select a random sample of 400 from each 10-point interval among non-links with weights below 20, and links with weights above 60.

This will require me to manually code 12,586 pairs.

I will label each record pair with a stratum identifier as follows:

stratum	definition	sample size
1	not sampled (weight $\leq$ -10 or weight $>$ 95)	0
2	sampled with certainty (links with weight $\leq$ 20; non-links with weight $>$ 60)	
3	non-links with -10 $<$ weight $\leq$ 0	400
4	non-links with 0 $<$ weight $\leq$ 10	400
5	non-links with 10 $<$ weight $\leq$ 20	400
6	links and non-links with 20 $<$ weight $\leq$ 30	1,900
7	links and non-links with 30 $<$ weight $\leq$ 40	1,900
8	links and non-links with 40 $<$ weight $\leq$ 50	1,900
9	links and non-links with 50 $<$ weight $\leq$ 60	1,900
10	links with 60 $<$ weight $\leq$ 70	400
11	links with 70 $<$ weight $\leq$ 95	400

```
%<<>>=
stratum <- rep(NA,length(wp[,1]))
for(i in 1:length(stratum)){
   if(wp$Wdata[i] <= -10 | wp$Wdata[i] > 95)
                            {stratum[i] <- 1}
   else if ((wp$prediction[i] == 'L' & wp$Wdata[i] <= 20) | (wp$prediction[i] == 'N' &
       wp$Wdata[i] > 60))   {stratum[i] <- 2}
   else if (wp$prediction[i] == 'N'  & wp$Wdata[i] > -10                          &
       wp$Wdata[i] <= 0)    {stratum[i] <- 3}
   else if (wp$prediction[i] == 'N'  & wp$Wdata[i] > 0                            & wp$Wdata[i] <= 10){stratum[i] <- 4}
   else if (wp$prediction[i] == 'N'  & wp$Wdata[i] > 10                           & wp$Wdata[i] <= 20){stratum[i] <- 5}
   else if (wp$Wdata[i] > 20         & wp$Wdata[i] <= 30)                                              {stratum[i] <- 6}
   else if (wp$Wdata[i] > 30         & wp$Wdata[i] <= 40)                                              {stratum[i] <- 7}
   else if (wp$Wdata[i] > 40         & wp$Wdata[i] <= 50)                                              {stratum[i] <- 8}
   else if (wp$Wdata[i] > 50         & wp$Wdata[i] <= 60)                                              {stratum[i] <- 9}
   else if (wp$prediction[i] == 'L'  & wp$Wdata[i] > 60                           & wp$Wdata[i] <= 70){stratum[i] <- 10}
   else if (wp$prediction[i] == 'L'  & wp$Wdata[i] > 70                           & wp$Wdata[i] <= 95){stratum[i] <- 11}
   else stratum[i] <- 12
}
wp <- data.frame(wp,stratum)

strata.count <- data.frame(table(wp$stratum),c(0,1,400,400,400,1900,1900,1900,1900,400,400))
colnames(strata.count) <- c('stratum','N','n')

library(TeachingDemos)
char2seed('revisit') # this seed is 2121383

sample2 <- wp[which(wp$stratum==2),]
sample3 <- wp[which(wp$stratum==3),][sample(strata.count[3,2],strata.count[3,3]),]
sample4 <- wp[which(wp$stratum==4),][sample(strata.count[4,2],strata.count[4,3]),]
sample5 <- wp[which(wp$stratum==5),][sample(strata.count[5,2],strata.count[5,3]),]
sample6 <- wp[which(wp$stratum==6),][sample(strata.count[6,2],strata.count[6,3]),]
sample7 <- wp[which(wp$stratum==7),][sample(strata.count[7,2],strata.count[7,3]),]
sample8 <- wp[which(wp$stratum==8),][sample(strata.count[8,2],strata.count[8,3]),]
sample9 <- wp[which(wp$stratum==9),][sample(strata.count[9,2],strata.count[9,3]),]
sample10 <- wp[which(wp$stratum==10),][sample(strata.count[10,2],strata.count[10,3]),]
sample11 <- wp[which(wp$stratum==11),][sample(strata.count[11,2],strata.count[11,3]),]
allsamples <- rbind(sample2,sample3,sample4,sample5,sample6,sample7,sample8,sample9,sample10,sample11)
set.seed(seed=NULL)

load("result.jan.RData")
samplerows            <- allsamples[which(allsamples$month=='01'),]
sample.jan            <- result.jan.bag[rownames(samplerows)]
```

```
sample.jan$Wdata      <- samplerows$Wdata
sample.jan$prediction <- samplerows$prediction
sample.jan$stratum    <- samplerows$stratum
rm(result.jan.bag)

load("result.feb.RData")
samplerows            <- allsamples[which(allsamples$month=='02'),]
sample.feb            <- result.feb.bag[as.numeric(rownames(samplerows))-countsbymonth[1]]
sample.feb$Wdata      <- samplerows$Wdata
sample.feb$prediction <- samplerows$prediction
sample.feb$stratum    <- samplerows$stratum
rm(result.feb.bag)

load("result.mar.RData")
samplerows            <- allsamples[which(allsamples$month=='03'),]
sample.mar            <- result.mar.bag[as.numeric(rownames(samplerows))-sum(countsbymonth[c(1:2)])]
sample.mar$Wdata      <- samplerows$Wdata
sample.mar$prediction <- samplerows$prediction
sample.mar$stratum    <- samplerows$stratum
rm(result.mar.bag)

load("result.apr.RData")
samplerows            <- allsamples[which(allsamples$month=='04'),]
sample.apr            <- result.apr.bag[as.numeric(rownames(samplerows))-sum(countsbymonth[c(1:3)])]
sample.apr$Wdata      <- samplerows$Wdata
sample.apr$prediction <- samplerows$prediction
sample.apr$stratum    <- samplerows$stratum
rm(result.apr.bag)

load("result.may.RData")
samplerows            <- allsamples[which(allsamples$month=='05'),]
sample.may            <- result.may.bag[as.numeric(rownames(samplerows))-sum(countsbymonth[c(1:4)])]
sample.may$Wdata      <- samplerows$Wdata
sample.may$prediction <- samplerows$prediction
sample.may$stratum    <- samplerows$stratum
rm(result.may.bag)

load("result.jun.RData")
samplerows            <- allsamples[which(allsamples$month=='06'),]
sample.jun            <- result.jun.bag[as.numeric(rownames(samplerows))-sum(countsbymonth[c(1:5)])]
sample.jun$Wdata      <- samplerows$Wdata
sample.jun$prediction <- samplerows$prediction
sample.jun$stratum    <- samplerows$stratum
rm(result.jun.bag)

load("result.jul.RData")
samplerows            <- allsamples[which(allsamples$month=='07'),]
sample.jul            <- result.jul.bag[as.numeric(rownames(samplerows))-sum(countsbymonth[c(1:6)])]
sample.jul$Wdata      <- samplerows$Wdata
sample.jul$prediction <- samplerows$prediction
sample.jul$stratum    <- samplerows$stratum
rm(result.jul.bag)

load("result.aug.RData")
samplerows            <- allsamples[which(allsamples$month=='08'),]
sample.aug            <- result.aug.bag[as.numeric(rownames(samplerows))-sum(countsbymonth[c(1:7)])]
sample.aug$Wdata      <- samplerows$Wdata
sample.aug$prediction <- samplerows$prediction
sample.aug$stratum    <- samplerows$stratum
rm(result.aug.bag)

load("result.sep.RData")
samplerows            <- allsamples[which(allsamples$month=='09'),]
sample.sep            <- result.sep.bag[as.numeric(rownames(samplerows))-sum(countsbymonth[c(1:8)])]
sample.sep$Wdata      <- samplerows$Wdata
sample.sep$prediction <- samplerows$prediction
sample.sep$stratum    <- samplerows$stratum
rm(result.sep.bag)
```

```
load("result.oct.RData")
samplerows            <- allsamples[which(allsamples$month=='10'),]
sample.oct            <- result.oct.bag[as.numeric(rownames(samplerows))-sum(countsbymonth[c(1:9)])]
sample.oct$Wdata      <- samplerows$Wdata
sample.oct$prediction <- samplerows$prediction
sample.oct$stratum    <- samplerows$stratum
rm(result.oct.bag)

load("result.nov.RData")
samplerows            <- allsamples[which(allsamples$month=='11'),]
sample.nov            <- result.nov.bag[as.numeric(rownames(samplerows))-sum(countsbymonth[c(1:10)])]
sample.nov$Wdata      <- samplerows$Wdata
sample.nov$prediction <- samplerows$prediction
sample.nov$stratum    <- samplerows$stratum
rm(result.nov.bag)

load("result.dec.RData")
samplerows            <- allsamples[which(allsamples$month=='12'),]
sample.dec            <- result.dec.bag[as.numeric(rownames(samplerows))-sum(countsbymonth[c(1:11)])]
sample.dec$Wdata      <- samplerows$Wdata
sample.dec$prediction <- samplerows$prediction
sample.dec$stratum    <- samplerows$stratum
rm(result.dec.bag)

rm(samplerows)
save(list=ls(pat='sample'),file='RevisitTestSamples.RData')

sample.jan <- editMatch(sample.jan)
sample.feb <- editMatch(sample.feb)
sample.mar <- editMatch(sample.mar)

wrongprediction.jan <- sample.jan[2*sample.jan$pairs$is_match+1 != as.numeric(sample.jan$prediction)]
wrongprediction.feb <- sample.feb[2*sample.feb$pairs$is_match+1 != as.numeric(sample.feb$prediction)]
wrongprediction.mar <- sample.mar[2*sample.mar$pairs$is_match+1 != as.numeric(sample.mar$prediction)]
wrongprediction.apr <- sample.apr[2*sample.apr$pairs$is_match+1 != as.numeric(sample.apr$prediction)]
wrongprediction.may <- sample.may[2*sample.may$pairs$is_match+1 != as.numeric(sample.may$prediction)]
wrongprediction.jun <- sample.jun[2*sample.jun$pairs$is_match+1 != as.numeric(sample.jun$prediction)]
wrongprediction.jul <- sample.jul[2*sample.jul$pairs$is_match+1 != as.numeric(sample.jul$prediction)]
wrongprediction.aug <- sample.aug[2*sample.aug$pairs$is_match+1 != as.numeric(sample.aug$prediction)]
wrongprediction.sep <- sample.sep[2*sample.sep$pairs$is_match+1 != as.numeric(sample.sep$prediction)]
wrongprediction.oct <- sample.oct[2*sample.oct$pairs$is_match+1 != as.numeric(sample.oct$prediction)]
wrongprediction.nov <- sample.nov[2*sample.nov$pairs$is_match+1 != as.numeric(sample.nov$prediction)]
wrongprediction.dec <- sample.dec[2*sample.dec$pairs$is_match+1 != as.numeric(sample.dec$prediction)]

wrongprediction.jan <- editMatch(wrongprediction.jan)
wrongprediction.feb <- editMatch(wrongprediction.feb)
wrongprediction.mar <- editMatch(wrongprediction.mar)
wrongprediction.apr <- editMatch(wrongprediction.apr)
wrongprediction.may <- editMatch(wrongprediction.may)
wrongprediction.jun <- editMatch(wrongprediction.jun)
wrongprediction.jul <- editMatch(wrongprediction.jul)
wrongprediction.aug <- editMatch(wrongprediction.aug)
wrongprediction.sep <- editMatch(wrongprediction.sep)
wrongprediction.oct <- editMatch(wrongprediction.oct)
wrongprediction.nov <- editMatch(wrongprediction.nov)
wrongprediction.dec <- editMatch(wrongprediction.dec)

correctmatches.func <- function(dframe1,dframe2){
   match1 <- with(dframe1,data.frame(id1=pairs$id1,id2=pairs$id2,stratum,Wdata,prediction,match=pairs$is_match))
   match2 <- with(dframe2,data.frame(id1=pairs$id1,id2=pairs$id2,match.corrected=pairs$is_match))
   match3 <- merge(match1,match2,by=c('id1','id2'),all=T)
   match.final <- rep(NA,length(match3[,1]))
   for(i in 1:length(match3[,1])) {
      match.final[i] <- if(is.na(match3$match.corrected[i])) match3$match[i] else match3$match.corrected[i]
      }
   data.frame(match3,match.final)
}
```

```
corrected.jan <- correctmatches.func(sample.jan,wrongprediction.jan)
corrected.feb <- correctmatches.func(sample.feb,wrongprediction.feb)
corrected.mar <- correctmatches.func(sample.mar,wrongprediction.mar)
corrected.apr <- correctmatches.func(sample.apr,wrongprediction.apr)
corrected.may <- correctmatches.func(sample.may,wrongprediction.may)
corrected.jun <- correctmatches.func(sample.jun,wrongprediction.jun)
corrected.jul <- correctmatches.func(sample.jul,wrongprediction.jul)
corrected.aug <- correctmatches.func(sample.aug,wrongprediction.aug)
corrected.sep <- correctmatches.func(sample.sep,wrongprediction.sep)
corrected.oct <- correctmatches.func(sample.oct,wrongprediction.oct)
corrected.nov <- correctmatches.func(sample.nov,wrongprediction.nov)
corrected.dec <- correctmatches.func(sample.dec,wrongprediction.dec)


revisit.eval <- rbind(with(corrected.jan,data.frame(stratum,Wdata,prediction,match=match.final)),
                      with(corrected.feb,data.frame(stratum,Wdata,prediction,match=match.final)),
                      with(corrected.mar,data.frame(stratum,Wdata,prediction,match=match.final)),
                      with(corrected.apr,data.frame(stratum,Wdata,prediction,match=match.final)),
                      with(corrected.may,data.frame(stratum,Wdata,prediction,match=match.final)),
                      with(corrected.jun,data.frame(stratum,Wdata,prediction,match=match.final)),
                      with(corrected.jul,data.frame(stratum,Wdata,prediction,match=match.final)),
                      with(corrected.aug,data.frame(stratum,Wdata,prediction,match=match.final)),
                      with(corrected.sep,data.frame(stratum,Wdata,prediction,match=match.final)),
                      with(corrected.oct,data.frame(stratum,Wdata,prediction,match=match.final)),
                      with(corrected.nov,data.frame(stratum,Wdata,prediction,match=match.final)),
                      with(corrected.dec,data.frame(stratum,Wdata,prediction,match=match.final)))

save(list=ls(pat='sample|wrongprediction'),revisit.eval,file='RevisitTestSamples.RData')

%@
```

to do: 1. use revisit.eval to estimate the proportion of pairs correctly classified by ml

2. Correct the predicted links (in result.jan through result.dec) that were changed after manual review (correct sample links are in corrected.jan through corrected.dec)

3. produce a file of the linked pairs, for further processing in SAS.

I assessed the macine-linking accuracy among pairs that were not selected for manual review. I assume that the machine-predicted links in stratum 1 (all pairs with weight $\leq$ -10 and all predicted links with weight $>$ 95) are all correct.

Table 2: Frequency table of Fellegi-Sunter weights by machine-predicted links.

FS weight	predicted non-link	predicted link	total pairs
(-50,-10]	240715759	0	240715759
(-10,0]	1326114	259	1326373
(0,10]	169970	220	170190
(10,20]	132641	476	133117
(20,30]	29981	6397	36378
(30,40]	9254	13068	22322
(40,50]	9465	23554	33019
(50,60]	9119	41449	50568
(60,70]	1684	133590	135274
(70,95]	347	1262175	1262522
(95,155]	0	3955116	3955116
total	242404334	5436304	247840638

stratum	definition	sample size
1	not sampled (weight $\leq$ -10 or weight $>$ 95)	0
2	sampled with certainty (links with weight $\leq$ 20; non-links with weight $>$ 60)	
3	non-links with -10 $<$ weight $\leq$ 0	400
4	non-links with 0 $<$ weight $\leq$ 10	400
5	non-links with 10 $<$ weight $\leq$ 20	400
6	links and non-links with 20 $<$ weight $\leq$ 30	1,900
7	links and non-links with 30 $<$ weight $\leq$ 40	1,900
8	links and non-links with 40 $<$ weight $\leq$ 50	1,900
9	links and non-links with 50 $<$ weight $\leq$ 60	1,900
10	links with 60 $<$ weight $\leq$ 70	400
11	links with 70 $<$ weight $\leq$ 95	400

	total pairs		sample		% non-links	% links
stratum	nonlinks	links	nonlinks	links	correct	correct
1	240,715,759	3,955,116	0	0	100	100
2	2,031	955	2,031	955	70.0	87.1
3	1,326,114	0	400	0	100	
4	169,970	0	400	0	99.8	
5	132,641	0	400	0	98.8	
6	29,981	6,397	1,547	353	95.1	95.2
7	9,254	13,068	811	1,089	93.3	95.7
8	9,465	23,554	535	1,365	80.2	98.8
9	9,119	41,449	347	1,553	89.9	96.3
10	0	133,590	0	400		99.8
11	0	1,262,175	0	400		100

An estimated 99.95% of the machine-predicted links, and an estimated 99.997% of the machine-predicited non-links agree with a manual classification.

```
%<<>>=
p.nonlink <- with(revisit.eval[revisit.eval$prediction=='N',],table(match,stratum,useNA='ifany'))

strata.count$N.nonlink <- c(
240715759,
      2031,
  1326114,
    169970,
    132641,
     29981,
      9254,
      9465,
      9119,
          0,
          0)
strata.count$N.link <- c(3955116,
      955,
        0,
        0,
        0,
 6397,
 13068,
 23554 ,
 41449  ,
  133590,
1262175)

%@
```