NYC Health

Epi Research Report

October 2016

An Illustrated Guide to Probabilistic Matching: Linking New York City Correctional Health and Vital Statistics Data

David Lee, MPH, MBA, Sungwoo Lim, DrPH, MS, Fatos Kaba, MA

Data matching provides an opportunity for in depth investigation of research questions. The New York City (NYC) Department of Health and Mental Hygiene (DOHMH) has completed a number of data matching projects to better understand health outcomes and target programs appropriately. This report describes a transparent, customizable and reproducible method for performing probabilistic data matching.

Key Points:

- Matching independent health-related datasets allows leveraging of routinely collected data for research and evaluation, and to inform program and policy formation.
- The New York City Health Department explored options for a systematic, reproducible, and low-burden method for one such data matching project and found that the RecordLinkage package for R offered reproducibility and transparency.
- A probabilistic match of electronic health records and vital statistics mortality data using the software was assessed with human review; sensitivity was calculated as 97.00% and specificity as 98.96%.
- The method can be used to build workforce capacity for matching "messy" datasets.

Background

Linkage of data from multiple health-related datasets expands the value of routinely collected data for population health research, evaluation of public health initiatives, and program planning. The NYC DOHMH has had limited capacity to match records across multiple systems, despite having access to many administrative and surveillance datasets. Limited technical expertise and lack of access to matching technology have prevented workforce growth in the area of data linkage.

Despite these barriers, the Department has successfully completed multiple large-scale data-matching projects. [1,2,3] However, these projects employed domain-specific matching algorithms[3] or proprietary platforms,[1,2] limiting reproducibility and scalability.

The purpose of this document is to present an alternative matching approach that not only performs competitively with proprietary software, but is also highly transparent and customizable. We describe matching approaches and present a case study using the RecordLinkage package for R to match datasets from NYC Correctional Health Services and the Office of Vital Statistics. Sample code is also provided for illustrative purposes. The use of RecordLinkage or other open source solutions allows analysts to develop expertise in record linkage techniques, expanding staff capacity for working between different datasets.

1 Pfeiffer M, Slopen M, Curry A, McVeigh K. Creation of a linked inter-agency data warehouse: the Longitudinal Study of Early Development. A Research Report from the New York City Department of Health and Mental Hygiene, 2012. (http://www1.nyc.gov/assets/doh/downloads/pdf/episrv/lsed-white-paper.pdf). (Accessed June 26, 2015.)

2 Levanon Seligson A, Lim S, Singh T, et al. New York/New York III Supportive Housing Evaluation: Interim Utilization and Cost Analysis. A report from the New York City Department of Health and Mental Hygiene in collaboration with the New York City Human Resources Administration and the New York State Office of Mental Health, 2013. (http://www1.nyc.gov/assets/doh/downloads/pdf/mental/housing-interim-report.pdf). (Accessed June 26, 2015.)

Description of Matching Process

Matching approaches can be broadly described as **deterministic** or **probabilistic**. The former involves finding exact matches between variables in two or more datasets. Thus, it is especially useful when unique identifiers such as social security number (SSN) are available. In deterministic matching, the matching algorithm is also generally more straightforward to understand and readily reproducible.

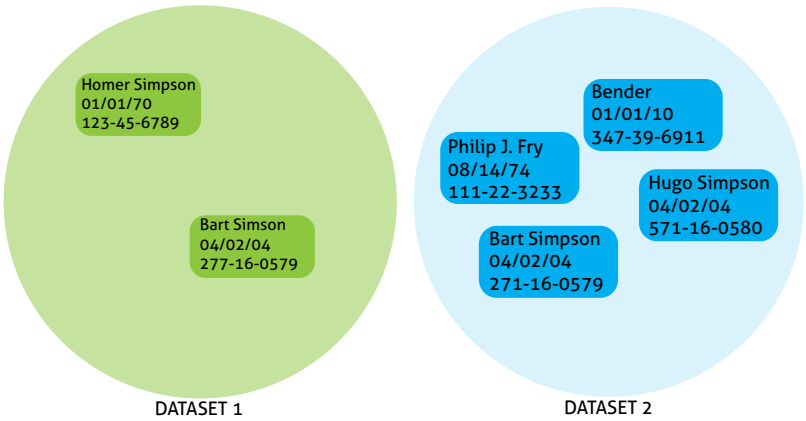
However, administrative data often include spelling errors and incomplete data, causing deterministic methods to "miss" true matches. For example, deterministic matching by exact name would not identify "John Doe" and "John Do" as the same person. To address this issue, Department analysts have employed "matching keys,"[4] a sequence of specific deterministic matching rules.

The list of rules becomes increasingly complicated both as the number of fields to be matched increases and the quality (consistency) of data decreases. Rules are also determined by expert review, which might introduce bias.

Probabilistic matching is often used when such data entry problems are frequent in a dataset. In this method, matching is conducted in iterative processes, called **passes**.[5] The first part of a pass is called blocking. In blocking, analysts specify variables by which records from both datasets are compared. All records that have an exact match on these variables are considered potential matches and are set aside. These potential matches are all in the same "block." Within this block, potential matches are already similar across the blocking variables which defined the block. However, there are many other variables which hold information that can be used to tease apart true matches from false matches. To further identify potential matches, analysts initiate **linking** by specifying another list of variables, which are used to assign numeric values, or **weights**, to indicate the likelihood of the potential matches being a real match.[6] To do this, analysts must also provide comparator functions for each variable, which determine the way by which the variables forming the matched pair are compared. For example, in order to compare first names, Census Bureau analysts often employ a Soundex function, which assigns a higher weight for potential matches with names that are pronounced similarly in the English language despite differences in spelling.[7]

Figure 1) Illustrated example of a probabilistic matching pass

1) Two separate datasets are cleaned and standardized in preparation for the first pass.



3 Drobnik A, Pinchoff J, Bushnell G,Terranova E, Fuld J. Matching New York City Viral Hepatitis,Tuberculosis, Sexually Transmitted Diseases and HIV Surveillance Data, 2000-2010. New York City Department of Health and Mental Hygiene: Epi Research Report, October 2013; 1-12. (http://www1.nyc.gov/assets/doh/downloads/pdf/epi/epiresearch-PCSI.pdf). (Accessed June 26, 2015.)

4 Drobnik A, Pinchoff J, Bushnell G, et al. Matching HIV, Tuberculosis, Viral Hepatitis, and Sexually Transmitted Diseases Surveillance Data, 2000-2001: Identification of Infectious Disease Syndemics in New York City. *J Public Health Manag Pract.* 2014;20(5):506-512.

5 See Appendix 2 for an illustrated example of pass construction.

6 Weights are composites of agreement and disagreement points, which are both calculated from match and non-match probabilities. See here for a good overview (Australian Bureau of Statistics. Information Paper: Death registrations to Census linkage project – Methodology and Quality Assessment: Australia, 2011-2012. Published 2013.)

7 http://www.archives.gov/research/census/soundex.html

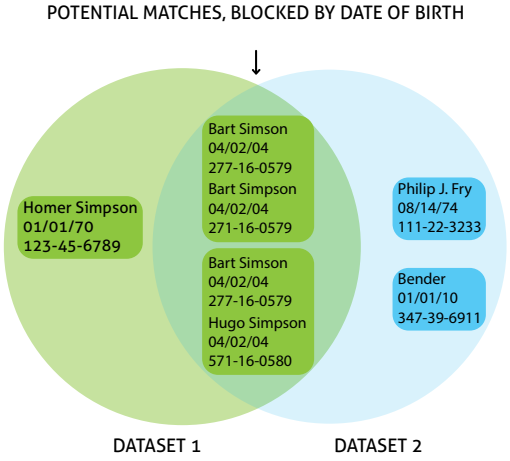
After linking, analysts review the potential match pairs and the computed weights to set a **threshold (cut-off) weight**. The threshold is commonly set by first visually checking the distribution of weights and identifying a weight at which every pair above it appears to be a true match and a lower weight at which pairs are true non-matches.[8] Domain experts then review the pairs that fall between these two weights and decide on a threshold weight that should differentiate pairs that are true matches and true non-matches.

Records with weights greater than or equal to the threshold are considered matches and are excluded from future passes. Those with weights smaller than the threshold are included in the next pass following the steps outlined above, with the expectation that they might return potential matches. This process continues until the analysts are satisfied with the number of matched records or additional passes no longer yield an appreciable number of matched records.

Figure 1) Cont'd. Illustrated example of a probabilistic matching pass

2) The pass is "blocked" by date of birth. The overlap region represents the block, which contains all pairs (potential matches) between the two datasets that exactly match on date of birth. The potential pairs are assigned weights based on linking variables (in this case, full name and social security number).

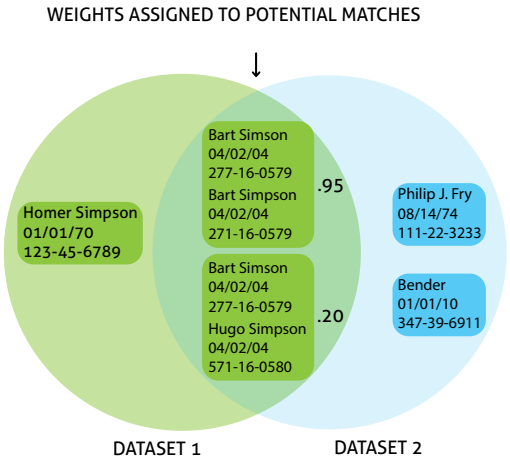
In our example, our block includes a potential match between Bart Simson from Dataset 1 and Bart Simpson from Dataset 2, as well as a match pair between Bart Simson from Dataset 1 and Hugo Simpson from Dataset 2.



3) Analysts review computed weights and set a threshold weight. All records in potential pairs with weights above the threshold are considered true matches and are set aside. All other records (those unblocked, or belonging to pairs with weights below the threshold in this pass) move onto the next pass, where analysts must determine new blocking and linking criteria.

In our example, let us consider a threshold weight of .80, which was decided on by the analyst. The Bart Simpson—Bart Simpson pair is assigned a weight of .95 by the program because only one character of the full name did not match and only one digit of the Social Security Number did not match. Because this weight is greater than .80, the pair is recognized as a true match, while the Bart Simson—Hugo Simpson pair is not. Thus, the Hugo Simpson record is moved to the next pass.

Note: Names used in this illustration are fictitious.



8 See Appendix 2 for an example.

Case Study: Matching Correctional Health and Vital Statistics Data

NYC Correctional Health Services provides medical and mental health assessment, treatment, and mental health discharge planning to all people incarcerated in New York City jails. Correctional Health was interested in assessing mortality risk and causes of death after release from jail among formerly incarcerated persons in NYC to explore potential interventions during incarceration. To assess this question all incarceration records for people discharged from NYC jails between June 1, 2011, and December 31, 2012 were matched to death certificate data of the same period. Correctional Health planned to update these data for future analyses, so sustainability of the matching solution was a high priority.

The project required routine merging of two large datasets (NYC Correctional Health Service electronic health records and Vital Statistics death registry).[9] Correctional Health explored options for a systematic, reproducible, and low-burden method for data matching. Several public domain/open-source platforms were considered, including Link Plus,[10] Link King[11] for SAS and FEBRL[12] for Python. RecordLinkage[13] for R was ultimately selected due to its customizability and transparency.

The Correctional Health dataset of all persons discharged from NYC jails between June 1, 2011, and December 31, 2012 included 65,535 records, in which inmates are identified by unique New York State ID (NYSID) number and each episode of incarceration is identified by a unique book and case number. Each record represented a distinct episode and thus, multiple records could refer to the same person. The Vital Statistics file of death certificate data contained unique records (82,366 rows), each uniquely identified by death certificate number.

Multiple Correctional Health records could match to the same Vital Statistics record, as a person could have multiple episodes of incarceration but only one death, raising the question of whether or not to create rules based on expert knowledge to remove duplicate records in the Correctional Health data before matching. It was decided to perform record matching without first de-duplicating, reasoning that the additional human review before matching would potentially introduce bias. Additionally, de-duplication would also be accomplished through matching.

Table 1) Defining passes: blocking and linking criteria

	Pass 1		Pass 2		Pass 3	
Variable	**Block**	**Link**	**Block**	**Link**	**Block**	**Link**
Social Security Number (SSN)	X			Edit		X
Date of birth (DOB) – Full		X				
DOB - Year				X	X	
DOB – Month				X		
DOB – Day				X		
First Name - Part 1		X	Phon		First 2 cha	
First Name - Part 2		X		Phon		
Last Name - Part 1		X	Phon		First 2 cha	
Last Name - Part 2		X		Phon		

KEY: X = exact match on the criterion was employed within the pass

Edit = SSN compared based on "edit distance," the number of single-character edits that differ between two strings

Phon = matches made on phonetic transformation of the name string

First 2 cha = matches made comparing the first two characters of the name string

9 This project, like other projects that have matched data housed in separate offices or agencies, required data sharing agreements between the separate offices that collect and maintain the data. Data sharing agreements assure that confidentiality and security of data are maintained and define the parameters of use of the data.

10 http://www.cdc.gov/cancer/npcr/tools/registryplus/lp.htm

11 www.the-link-king.com/

12 http://sourceforge.net/projects/febrl/

13 http://crantastic.org/packages/RecordLinkage

Blocking criteria included: social security number (SSN), date of birth (DOB), and name. Names were first standardized to account for differences in formatting within and between datasets. Spaces and hyphens were used as delimiters to split first and last name fields each into two separate variables. For example, a full name of "John-Smith Doe" would yield value of "John" for the First Name – Part 1 variable, a value of "Smith" for the First Name – Part 2 variable, a value of "Doe" for Last Name – Part 1 variable, and a null value for the Last Name – Part 2 variable. Similarly, DOB was parsed into the year, month, and day of birth.[14]

Pass 1: Records were blocked on exact SSN and linked using five variables: full DOB, and the four name variables. We tested using both the epiWeights function and the emWeights function to calculate weights, choosing the latter because it returned a larger number of record pair clusters. EpiWeights returns weights between 0 and 1.0, while the latter returns weights between negative infinity and positive infinity. This created pairs of possible linkages whose weights were manually reviewed. A threshold weight of -4.90 was determined. This pass matched 97 Vital Statistics records to 140 Correctional Health records.[15]

Pass 2: First and last names were passed through a Soundex algorithm, which converts a string into a four-character code based on how the string is pronounced in spoken English, thereby making a "phonetic transformation." Records were then blocked on these transformations of the first part of first name and first part of last name. Linkage criteria included: year, month, and date of birth; phonetic transformations each of all four parts of name; and SSN. In this pass, SSN was compared based on "edit distance," which is the number of single-character edits that differ between two strings. For example, "111111111" is more similar to "111111112" than it is to "111111122" because one less single-character edit is needed to make the compared strings match. After another manual review, the threshold weight of 7.98 was selected. This pass matched an additional 108 Vital Statistics records to 169 Correctional Health records.

Pass 3: Records were blocked on the first two characters of the first part of the first name and first two characters of the first part of the last name. For example, "John Doe" was assessed on "JO" and "DO". Linkage criteria included full DOB and SSN. In this pass, we compared exact SSN instead of using edit distance as we did in the previous pass. Thus, even slight differences between SSNs would bare the same influence on weight as two completely different SSN. A threshold weight of 6.06 resulted in a match of 11 additional Vital Statistics records to 16 Correctional Health records.

In total, 216 Vital Statistics records were matched to 325 Correctional Health records.

Evaluating Performance of Matching

Probabilistic linkage is convenient and highly reproducible because after setting the threshold weight, the algorithm essentially does the "heavy lifting" of computing weights and determining matches based on the threshold without further human involvement. However, because these steps are automated, it is very important to evaluate probabilistic matching algorithms to determine whether the blocking and linking variables and threshold weights are appropriate.

After the final pass, we assessed the performance of this method against the gold standard of human review. A team of two analysts completed blinded, manual review of record pairs. Below, we describe specific evaluation steps.

Sampling and human review of matches

Thirty percent of pairs were randomly sampled for human review. Half of these pairs were previously determined to be matches (above threshold weights) and half were non-matches (below threshold weights). Matched pairs were sampled according to strata by frequency of weight, to mimic the distribution of the weights in the census of matched pairs. Unmatched pairs were a simple random sample.

Both reviewers separately determined whether or not a pair was a match using agreed-upon rules.[16] An Access database and form were created so that the reviewers could view and log their determination of record pairs, while remaining blinded to the software's actual matching determination.

After human reviewers independently scored the record pairs, each reviewer's Access databases along with their scores were merged. Only one discordant

14 See Appendix 1 for details on preparing data, including guidelines and example SAS code.

15 See Appendix 2 for example code used for constructing passes and for determining weights.

16 See Appendix 3 for rules used for human review.

pair was found; consensus was reached among both reviewers regarding the appropriate classification of the discordant pair.

Calculating Sensitivity and Specificity

With human review as the gold standard, sensitivity was calculated as 97.00% and specificity as 98.96%. This means that 97% of human-determined matches were correctly identified by the matching algorithm, and 99% of human-determined non-matches were correctly identified by the matching algorithm. These results suggest the validity of the data match, especially important considering recent research that emphasizes the importance of maximizing specificity for estimation.

Conclusion

Matching records across datasets presents an opportunity to understand health outcomes in a way that may not be possible through analysis of the single sources of data. In this project, matched data on formerly-incarcerated persons and death certificates allowed the NYC Health Department to characterize mortality risk and cause of death post-discharge.

Data matching requires technical skills and appropriate software to optimize the match. Probabilistic record matching using R and the RecordLinkage package provides a low-cost and highly transparent platform for identifying and merging elements from "messy" datasets. While considerable time investment might be required to become proficient in matching using probabilistic methods, these skills will continue to be important as data sharing and integration become more common to answer emergent public health questions.

Table 2) Matches and non-matches according to RecordLinkage and human review

	Human: Match	**Human: Non-Match**	**Total**
RecordLinkage: Match	97	1	98
RecordLinkage: Non-match	3	95	98
Total	100	96	196

Suggested citation: Lee D, Lim S, Kaba F. An Illustrated Guide to Probabilistic Matching: Linking New York City Correctional Health and Vital Statistics Data. New York City Department of Health and Mental Hygiene: *Epi Research Report*, October, 2016; 1-13.

Acknowledgements: Cynthia Driver, Jennifer Fuld, Kinjia Hinterland, Ross MacDonald, Neil Vora, Regina Zimmerman, Jamie Neckles

Appendix 1) Guidelines for pre-processing data before attempting matching

Standardizing *within* dataset

Analysts should first standardize data so that data elements are consistent between records in the same dataset. The steps applied to the Correctional Health dataset, along with relevant SAS code, are outlined below as an example:

1) **Assign unique identifiers** to each row of each dataset, so that the output file of matched pairs can be merged back to the original datasets.

```
id=_N_;
```

2) **Standardize values** in each dataset. Analysts should be especially mindful of: hyphenated names, reversal of first and last names, abbreviated middle names, abbreviated address or streets, and business keywords (ex: Corp. vs Corp).

```
*We replace hyphens in names with spaces, as well as transform to upper case;
first_name=upcase(tranwrd(first_name,"-",""));
last_name=upcase(tranwrd(last_name,"-",""));

*Split first name and last name, each into 2 components;
length first1 $ 20;
length first2 $ 20;
length last2 $ 20;
length last2 $ 20;

first1 = scan(first_name,1," ");
first2 = scan(first_name,2," ");
last1 = scan(last_name,1," ");
last2 = scan(last_name,2," ");

*Pull out birth years, month, and date and standardize digits;
dobyyyy=substr(put(dob,mmddyys10.),7,4);
dobmm=substr(put(dob,mmddyys10.),1,2);
dobdd=substr(put(dob,mmddyys10.),4,2);
```

3) **Identify and delete missing values**, so that any values coded as missing (ex: 999999999 for a missing SSN) do not a dversely impact weighing.

```
if SSN <= 99999999 and SSN > 1 then social=cat(0,ssn);
else if SSN > 99999999 and SSN < 999999999  then social=SSN;
```

4) In other words, any variable that will be compared phonetically (ex: Soundex) or by a string comparator (ex: Levenshtein edit distance) should be a string variable. Analysts should be mindful of variables typically entered as numeric data, such as SSN; these variables should be redefined as strings (and reformatted if necessary) to ensure compatibility with such comparison functions.

Ensuring comparability between datasets

After each dataset is standardized, analysts must ensure comparability so that the RecordLinkage package can properly read and compare the input files. Thus, matching variables should be consistent across the two datasets. For example, consider these elements of date of birth:

1) Variable name. If one dataset contains a variable named DOB, the other one should have a DOB variable (as opposed to names such as DateofBirth, Dob, Birth, etc.).

2) Data structure and type. If one file separated date of birth into three variables for month, day, and year components the other dataset should do the same. The data type, such as string versus numeric, should also be consistent.

3) Coding. Consider, for example, a variable that indicates age group (1-18, 19-30, 31+). These levels should be consistent between datasets. This step is crucial as different coding would not produce any logical errors during matching, but would greatly affect weighing.

Appendix 2) Constructing Pass 1

We first imported the Correctional Health data, which were already cleaned in SAS. Note that columns are read as factors, thus retaining leading zeros. After checking variables, we retain only those that we need in a new data frame called **chsDataWorking**. A data frame can roughly be thought of as the R analog to a SAS dataset.

```
> library(RecordLinkage)

> chsData <- read.csv("chs_data.csv",colClasses="factor")
> names(chsData)
[1] "NYSID"                    "BnC"                  "SSN"
[4] "AKA_Name_1"               "AKA_Name_2"           "AKA_Name_3"
[7] "Race"                     "Hisp"                 "Zip"
[10] "Borough_of_Residence"    "Marital"              "Country_Of_Origin"
[13] "Last_Name"               "First_Name"           "Dob"
[16] "Sex"                     "adm_Date"             "dsch_Date"
[19] "Education"               "id"                   "name"
[22] "first1"                  "first2"               "last2"
[25] "last1"                   "dobyyyy"              "dobmm"
[28] "dobdd"                   "social"               "nysid_new"
> chsDataWorking <- chsData [c(20,22,23,25,24,15,26,27,28,29)]
```

We renamed some variables to be consistent with the Vital Statistics data that will be read in later and checked both datasets to ensure consistency.

```
> names(chsDataWorking) <- gsub("Dob","dobFull",names(chsDataWorking))
> names(chsDataWorking)
[1] "id" "first1" "first2" "last1" "last2" "dobFull" "dobyyyy" "dobmm" "dobdd" "social"
> names(vitalDataWorking)
[1] "id" "first1" "first2" "last1" "last2" "dobFull" "dobyyyy" "dobmm" "dobdd" "social"
```

We blocked on SSN, linking on the name components and the full date of birth. Note that RecordLinkage links on all variables, unless explicitly told not to. In our case, we chose to exclude in our first pass: id, dobyyyy, dobmm, dobdd, social.

```
> rpairsFuzzy <- compare.linkage(chsDataWorking,vitalDataWorking,
+ blockfld=c(10),
+ exclude=c(1,7,8,9,10)
+ )
```

This new **rpairsFuzzy** data frame held the pairs constructed (among other data). We next assigned weights. Below, we used the **emWeights()** function to employ an Expectation-Maximization algorithm, although the package also supports the **epiWeights()** function.[17]

```
> myWeights <- emWeights(rpairsFuzzy)
```

17 See Sariyar, M. and Borg, A. The RecordLinkage package: detecting errors in data. The R Journal 2010; 2(2):61-67. (http://journal.r-project.org/archive/2010-2/RJournal_2010-2_Sariyar+Borg.pdf) (Accessed July 13, 2015.)

The **getpairs()** function organizes potential matches. The min and max weights were set to -100 (indicating a complete non-match) and +100 (indicating a complete match) respectively so that we could see all potential matches, since we had not yet decided on a threshold weight. We specified "single.rows=TRUE" so that pairs would be easier to scan by eye.

```
> myResults <- getPairs(myWeights,max.weight=100,min.weight=-
+ 100,single.rows=TRUE)
```

We examined the variables returned by the **myResults** data frame below. Elements 1 through 11 were derived from the first dataset (chsDataWorking) and elements 12 through 22 from the second (vitalDataWorking). The weight variable was computed by the **emWeights()** function.

```
> names(myResults)
[1] "id1" "id.1" "first1.1" "first2.1" "last1.1" "last2.1" "dobFull.1" "dobyyyy.1" "dobmm.1"
[10] "dobdd.1" "social.1" "id2" "id.2" "first1.2" "first2.2" "last1.2" "last2.2" "dobFull.2"
[19] "dobyyyy.2" "dobmm.2"   "dobdd.2"   "social.2"  "weight"
```

In order to determine a threshold (cutoff) weight for true matches, we constructed a histogram using the **hist()** function. Looking at the distribution, we saw that weights fell into five distinct values with a skewed distribution. We observed a dramatic change in the distribution around a weight of 15. We next viewed the rows representing the potential pairs with the **fix()** function to review the pairs with weights above and below 15. Manual review of those potential pairs according to pre-established human review rules revealed those pairs above 15 to be true matches.[18] Similarly, we proceeded to check each weight for which we observed marked shifts in distribution according to the histogram in order to identify the point at which all pairs could be classified as a true non-match. After looking over the potential pairs with **fix()**, we set the threshold weight for true non-matches to -4.904825.

```
> hist(myResults$Weight)
```

18 See Appendix 3.


```
> fix(myResults)
```

We extracted and retained the id and Weight variables to re-merge with our data later on.

```
> myResults.chs <- myResults[c(2,23)]
> names(myResults.chs)[1] <- "id"
> myResults.vital <- myResults[c(13,23)]
> names(myResults.vital)[1] <- "id"
> names(myResults.chs)
[1] "id"     "Weight"
```

We "set aside" those records which were successfully linked (i.e., blocked and belonging to a pair with a weight equal to or greater than the threshold) and stored them in the chsDataLinked and vitalDataLinked series of data frames. The pass variable keeps track of the number of the pass in which the linkage is made.

```
> myResults.chs$pass <- 1
> chsDataLinked <- merge(chsDataWorking,myResults.chs,by="id")
> chsDataLinked <- chsDataLinked[!chsDataLinked$Weight < -4.904825,]
> chsDataLinked.1 <- unique(chsDataLinked)
>
> myResults.vital$pass <- 1
> vitalDataLinked <- merge(vitalDataWorking,myResults.vital,by="id")
> vitalDataLinked <- vitalDataLinked[!vitalDataLinked$Weight < -4.904825,]
> vitalDataLinked.1 <- unique(vitalDataLinked)
```

All other records (those unblocked, or belonging to a pair with weight under the threshold) were placed into new data frames to be fed into the next pass using the next **compare.linkage()** call.

```
> chsData2 <- merge(chsDataWorking,myResults.chs,by="id",all.x=TRUE)
> chsData2 <- subset(chsData2,Weight < -4.904825 | is.na(Weight)==TRUE)
> chsData2 <- chsData2[-c(11,12)]
>
> vitalData2 <- merge(vitalDataWorking,myResults.vital,by="id",all.x=TRUE)
> vitalData2 <- subset(vitalData2,Weight < -4.904825 | is.na(Weight)==TRUE)
> vitalData2 <- vitalData2[-c(11,12)]
> vitalData2 <- unique(vitalData2)
> vitalData2 <- vitalData2[order(vitalData2$id),]
```

Appendix 3) Rules for human review

The following rules were used for evaluating the performance of the probabilistic matching algorithm via human review of sampled pairs:

1) If social security number (SSN) matches or nearly matches and first name (FN) and last name (LN) match or nearly match or are reversed but there is no birth date or birth date does not match, it is a match.

2) If birth date matches or nearly matches and FN and LN match or nearly match or are reversed but SSN does not match or is not there, it is a match.

3) If no SSN and birth date are present in the data, or they both do not match, even if first and last name match or nearly match or are reversed, it is not a match.

4) If SSN matches or nearly matches but not birth date, if the first or last name matches or nearly matches or are reversed it is a match; if neither first nor last name match or are not reversed, it is not a match.

5) If FN and LN match or nearly matches or are reversed and the names are very uncommon, and DOB is an exact match, if SSN is missing it is a match. If SSN is present for both individuals and is not matching, it is not a match.


NYC
Health