SurvMeth 622 Assignment 1

Cheng, Chia Wen

2023-02-19

```
library(RecordLinkage)
library(dplyr)
library(mice)
```

Load data and pre-linkage data processing

The pre-linkage process is composed of the following seven steps:
1. Replace headers with desired names in the two data frames (for compliance with the big data command requirements).
2. Drop empty rows that don't have values in "ID" or "DUP."
3. Replace all empty strings with NA.
4. Create a column of "fullname" with the initial of first name and the last name.
5. Change the format of DOB and split DOB into Year, Month, and Day.
6. Extract the first three letters of the last name and create a new variable lastthree.
7. Create identity vectors with the matchid variable - a variable that indicates the correct matches of the survey and admin data.

```
## load data
admindata <- read.csv("C:/Users/Angela/    (cwcheng@umich.edu)/0. study abroad/academic/9. 2023 Winter/
admindata <- data.frame(admindata)
surveydata <- read.csv("C:/Users/Angela/    (cwcheng@umich.edu)/0. study abroad/academic/9. 2023 Winter
surveydata <- data.frame(surveydata)

## Remove periods from column names
colnames(surveydata) <- c("matchid","dup","firstname","lastname","maritalstatus","race","dob","ssn","in
colnames(admindata) <- c("matchid","dup","firstname","lastname","maritalstatus","race","dob","ssn","inc

## DROP EMPTY ROWS
surveydata <- surveydata[complete.cases(surveydata[ ,1:2]), ]
admindata <- admindata[complete.cases(admindata[ ,1:2]), ]

## Replace missing values with NA
surveydata <- surveydata %>%
  na_if(x = ., y = "")
admindata <- admindata %>%
  na_if(x = ., y = "")

## Create new column "fullname" with first initial last names concatenated
surveydata$fullname <- paste(substring(surveydata$firstname,1,1), surveydata$lastname, sep = '')
admindata$fullname <- paste(substring(admindata$firstname,1,1), admindata$lastname, sep = '')
```

```
## Separate day/month/year columns for DOB
surveydata$dob <- as.Date(surveydata$dob, "%m/%d/%y")
admindata$dob <- as.Date(admindata$dob, "%m/%d/%y")
surveydata$month <- as.numeric(format(surveydata$dob, format = "%m"))
surveydata$day <- as.numeric(format(surveydata$dob, format = "%d"))
surveydata$year <- as.numeric(format(surveydata$dob, format = "%Y"))
admindata$month <- as.numeric(format(admindata$dob, format = "%m"))
admindata$day <- as.numeric(format(admindata$dob, format = "%d"))
admindata$year <- as.numeric(format(admindata$dob, format = "%Y"))

## Create new column "lastthree" first three letters of last name (for block field)
surveydata$lastthree <- substring(surveydata$lastname,1,3)
admindata$lastthree <- substring(admindata$lastname,1,3)
str(surveydata)
```

```
## 'data.frame':    3000 obs. of  18 variables:
##  $ matchid        : int  1003 1011 1013 1017 1028 1029 1031 1036 1040 1041 ...
##  $ dup            : int  1 1 0 0 0 1 0 1 0 1 ...
##  $ firstname      : chr  "Tyron" "Ethan" "Taliah" "Kiara" ...
##  $ lastname       : chr  "Rau" "Claahke" "Staude" "Kellow" ...
##  $ maritalstatus  : chr  NA "Married" "Married" "Married" ...
##  $ race           : chr  "White" "White" "White" "White" ...
##  $ dob            : Date, format: "2019-05-25" "2019-09-03" ...
##  $ ssn            : chr  NA "638-11-0983" "176-03-9379" "305-22-0947" ...
##  $ income         : num  NA NA 166135 49913 33235 ...
##  $ credit_card_num: chr  "1915 3160 2544 39912" "2741 2201 8604 48890" "4029 4369 6201 0727" "1116 64
##  $ city           : chr  "Washington DC" "Washington DC" "Washington DC" "Washington DC" ...
##  $ zip            : int  20004 20002 20018 20005 20017 20005 20013 20007 20006 20003 ...
##  $ telephone      : chr  "956-224-0530" NA "313-519-3459" "360-672-5947" ...
##  $ fullname       : chr  "TRau" "EClaahke" "TStaude" "KKellow" ...
##  $ month          : num  5 9 6 2 1 11 11 5 5 8 ...
##  $ day            : num  25 3 24 21 6 27 6 23 27 24 ...
##  $ year           : num  2019 2019 2019 2019 2019 ...
##  $ lastthree      : chr  "Rau" "Cla" "Sta" "Kel" ...
```

```
## Create identity vectors with the matchid variable
identity_survey <- surveydata[ , "matchid"]
identity_admin <- admindata[ , "matchid"]
```

Example experiment: link by names with city as the blocking variable and Jaro-Winkler comparison method

The example experiment of data linkage uses string comparison comparing only the name fields. "City" is used as the blockfield, where only exact match on this field will be matched. The default string comparison, Jaro-Winkler, is used in this demonstration.

```
## Compare names and block with "city"
example <- RLBigDataLinkage(surveydata, admindata, identity1=identity_survey, identity2 = identity_admi

## Obtain the summary weights for the potential links
### "The appropriate value of cutoff depends on the choice of string comparator. The default is adjuste
examplew <- emWeights(example, cutoff = 0.95)
```

```
## ================================================================================
```

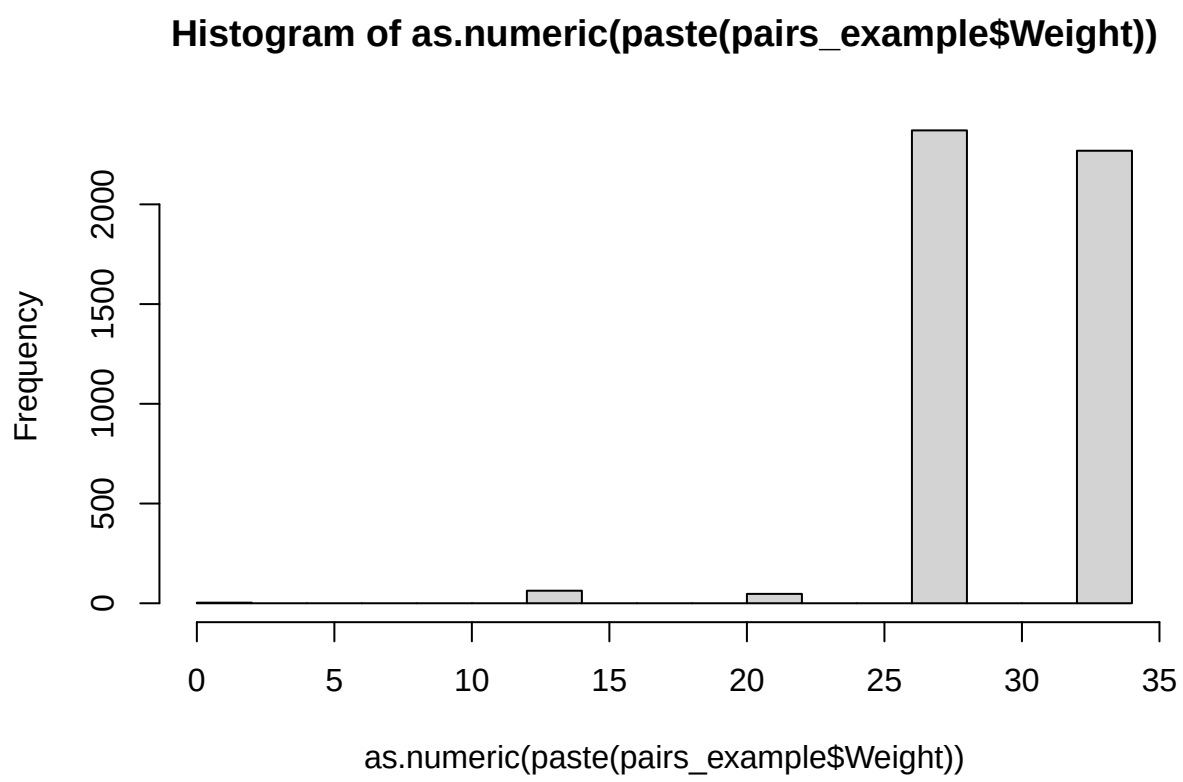
```
summary(examplew@Wdata[])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -19.79  -19.79  -19.79  -19.68  -19.79   33.93
```

```
pairs_example <- getPairs(examplew, min.weight = 0, single.rows = TRUE)
```

```
## ================================================================================
```

```
hist(as.numeric(paste(pairs_example$Weight)))
```




```
## Get thresholds manually from histogram and orint the results
example_class3 <- emClassify(examplew, threshold.lower = 25, threshold.upper = 30)
getTable(example_class3)
```

```
## < table of extent 0 x 0 >
```

```
getErrorMeasures(example_class3)
```

```
## $alpha
```

```
## [1] 0.1393914
##
## $beta
## [1] 2.165926e-05
##
## $accuracy
## [1] 0.9999366
##
## $precision
## [1] 0.9224328
##
## $sensitivity
## [1] 0.8606086
##
## $specificity
## [1] 0.9999783
##
## $ppv
## [1] 0.9224328
##
## $npv
## [1] 0.9999583
```

```
table_ex <- table(ff:::as.data.frame.ffdf(example_class3@prediction))
table_ex
```

```
##
##       N       P       L
## 8126016    2371    2269
```

For the example linkage experiment with names, using Jaro-Winkler comparison method and manually selected thresholds, we get a high sensitivity of 0.8606086 and a high precision of 0.9224328. There are 4247 links and 8126409 non-links.

Choices of linking variables and blocking variables

Names are fairly unique among a group of people but with opportunities of overlapping. While name is usually the easiest and the least concerned way in revealing maximum information among unique identifiers used in real life, such as SSN, I use name as (one of) the linking variable(s) in my experiments. In addition, this survey collects SSN and credit card number, which both obtain the feature of being genuinely unique. I would like to test the effects of adding them as part of the linking variables.
The blocking variables are selected based on three separately considered standards: 1) the variable being complete enough for comparing; 2) the variable does not have exhaustive categories; and 3) values within the variable share a degree of uniqueness. These three selection criteria will allow me to explore the patterns of diverse blocking variables. Thus, city, zip, DOB, and race are mixed-applied to each experiment.

Choices of thresholds and tolerance of differences

For the purpose of experimenting, I will apply optimized thresholds generated by the function in all experiments. I will also take turn conducting Jaro-Winkler's and Levenshtein's comparison methods with other factors holding constant to compare the results.

Initial experiment: link by names without blocking variable and string comparison

```
## Compare names
initial <- RLBigDataLinkage(surveydata, admindata, identity1=identity_survey, identity2 = identity_admi

## Obtain the summary weights for the potential links
initialw <- emWeights(initial, cutoff = 0.95)
```

```
## ================================================================================
```

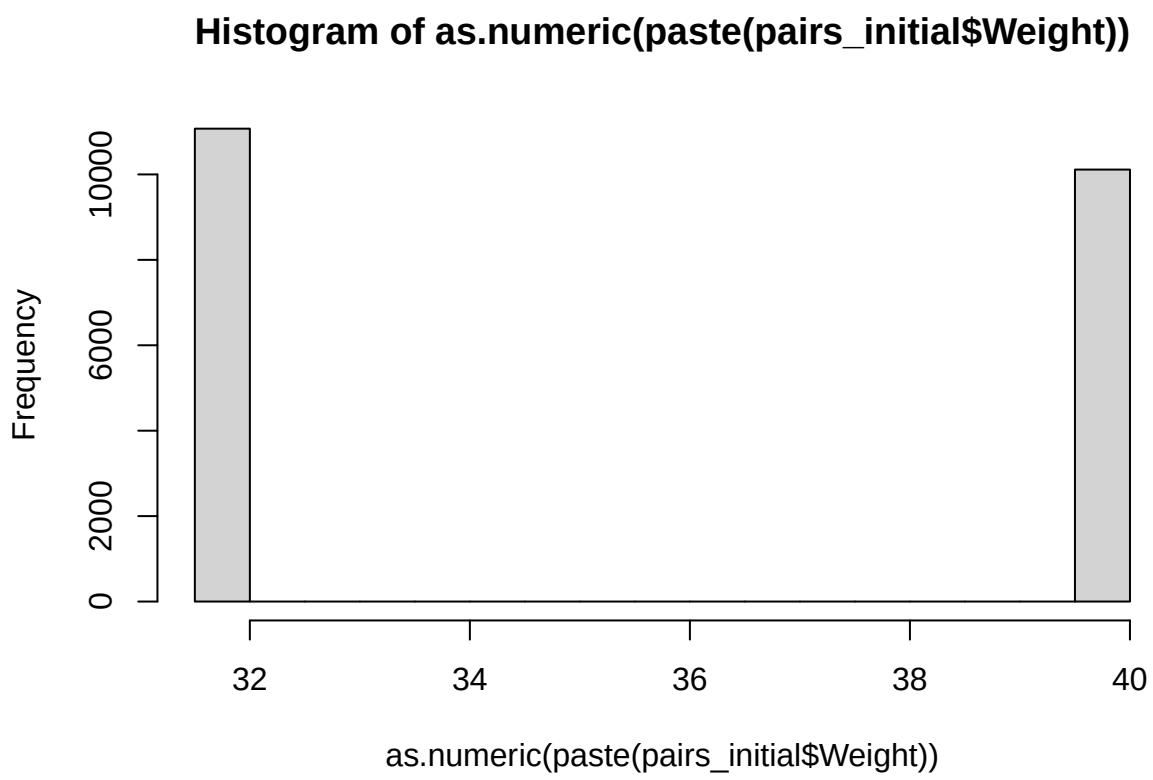
```
summary(initialw@Wdata[])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -28.31  -28.31  -28.31  -28.17  -28.31   39.60
```

```
pairs_initial <- getPairs(initialw, min.weight = 0, single.rows = TRUE)
```

```
## ================================================================================
```

```
hist(as.numeric(paste(pairs_initial$Weight)))
```



```
## Get "optimal" weight for setting threshold
threshold_initial <- optimalThreshold(initialw)
```

```
## ============================================================================
```

```
threshold_initial
```

```
## [1] 35.60984
```

```
## Print results
initial_class <- emClassify(initialw, threshold.upper = threshold_initial)
getTable(initial_class)
```

```
## < table of extent 0 x 0 >
```

```
getErrorMeasures(initial_class)
```

```
## $alpha
## [1] 0.3062201
##
## $beta
## [1] 0.000202434
##
## $accuracy
## [1] 0.9997771
##
## $precision
## [1] 0.1863753
##
## $sensitivity
## [1] 0.6937799
##
## $specificity
## [1] 0.9997976
##
## $ppv
## [1] 0.1863753
##
## $npv
## [1] 0.9999795
```

```
## Generate the classification table that shows the true status of the matches (from our identity vecto
table_i <- table(ff:::as.data.frame.ffdf(initial_class@prediction))
table_i
```

```
##
##        N        P        L
## 40642886        0    10114
```

For the initial linkage experiment with names, the optimal threshold we obtain is 35.60984. We get a fair sensitivity of 0.6937799 and a poor precision of 0.1863753. There are 10114 links and 40642886 non-links.

First experiment: link by names with city as the blocking variable and Levenshtein distance comparison method

The first experiment of data linkage uses string comparison comparing only the name fields. "City" is used as the blockfield, where only exact match on this field will be matched. The Levenshtein distance comparison method is used in this experiment.

```
colnames(surveydata)
```

```
##  [1] "matchid"         "dup"             "firstname"       "lastname"
##  [5] "maritalstatus"   "race"            "dob"             "ssn"
##  [9] "income"          "credit_card_num" "city"            "zip"
## [13] "telephone"       "fullname"        "month"           "day"
## [17] "year"            "lastthree"
```

```
## Compare names and block with "city"
first <- RLBigDataLinkage(surveydata, admindata, identity1=identity_survey, identity2 = identity_admin,

## Obtain the summary weights for the potential links
firstw <- emWeights(first, cutoff = 0.95)
```

```
## ============================================================================
```

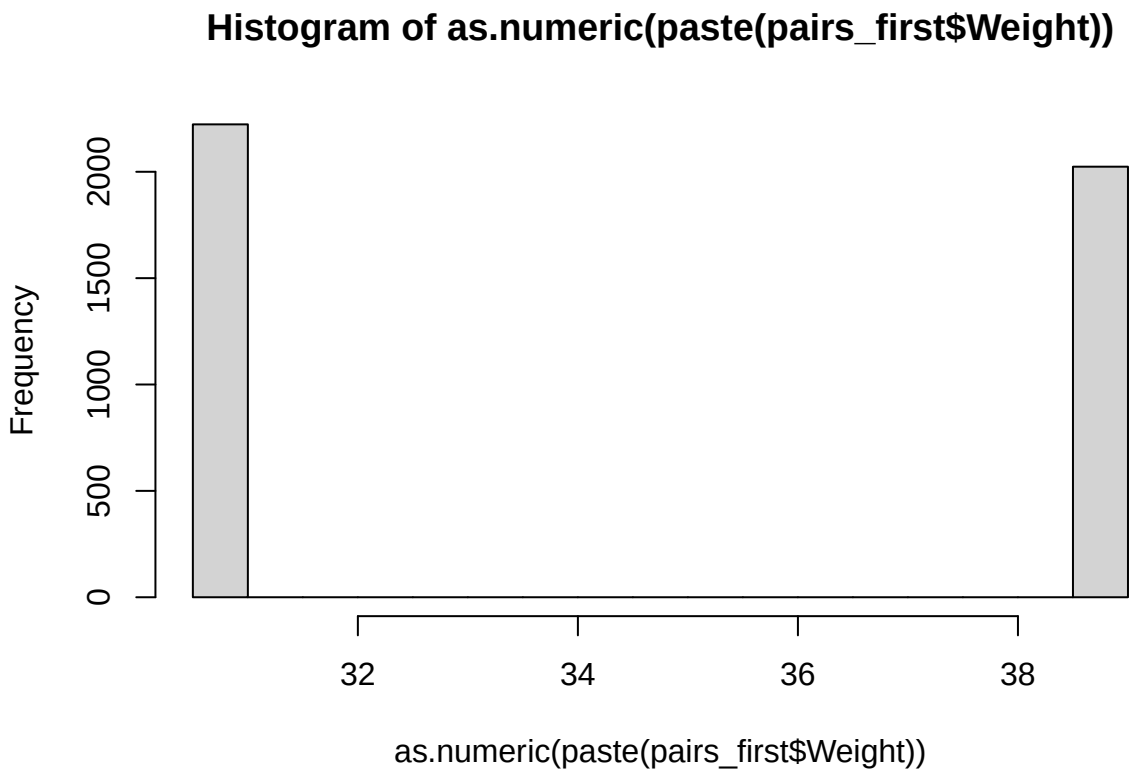
```
summary(firstw@Wdata[])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -25.86  -25.86  -25.86  -25.72  -25.86   38.50
```

```
pairs_first <- getPairs(firstw, min.weight = 0, single.rows = TRUE)
```

```
## ============================================================================
```

```
hist(as.numeric(paste(pairs_first$Weight)))
```




```
## Get "optimal" weight for setting threshold
threshold_first <- optimalThreshold(firstw)
```

```
## ============================================================================
```

```
threshold_first
```

```
## [1] 34.51171
```

```
## Print results
first_class <- emClassify(firstw, threshold.upper = threshold_first)
getTable(first_class)
```

```
## < table of extent 0 x 0 >
```

```
getErrorMeasures(first_class)
```

```
## $alpha
## [1] 0.3062201
##
## $beta
## [1] 1.710151e-05
##
```

```
## $accuracy
## [1] 0.9998806
##
## $precision
## [1] 0.9313241
##
## $sensitivity
## [1] 0.6937799
##
## $specificity
## [1] 0.9999829
##
## $ppv
## [1] 0.9313241
##
## $npv
## [1] 0.9998976
```

```
## Generate the classification table that shows the true status of the matches (from our identity vecto
table_1 <- table(ff:::as.data.frame.ffdf(first_class@prediction))
table_1
```

```
##
##       N       P       L
## 8128632       0    2024
```

For the first linkage experiment with names, using Levenshtein distance, the optimal threshold we obtain is 34.51171. We get a fair sensitivity of 0.6937799 and a high precision of 0.9313241. There are 2024 links and 8128632 non-links.

First additional experiment: link by name, SSN, credit card number with city as the blocking variable and Levenshtein distance comparison method

The first experiment of data linkage uses string comparison comparing only the name fields. "City" is used as the blockfield, where only exact match on this field will be matched. The Levenshtein distance comparison method is used in this experiment.

```
## Compare name, SSN, and credit card number and block with "city"
first_a <- RLBigDataLinkage(surveydata, admindata, identity1=identity_survey, identity2 = identity_admi

## Obtain the summary weights for the potential links
firstw_a <- emWeights(first_a, cutoff = 0.95)
```

```
## ================================================================================
```

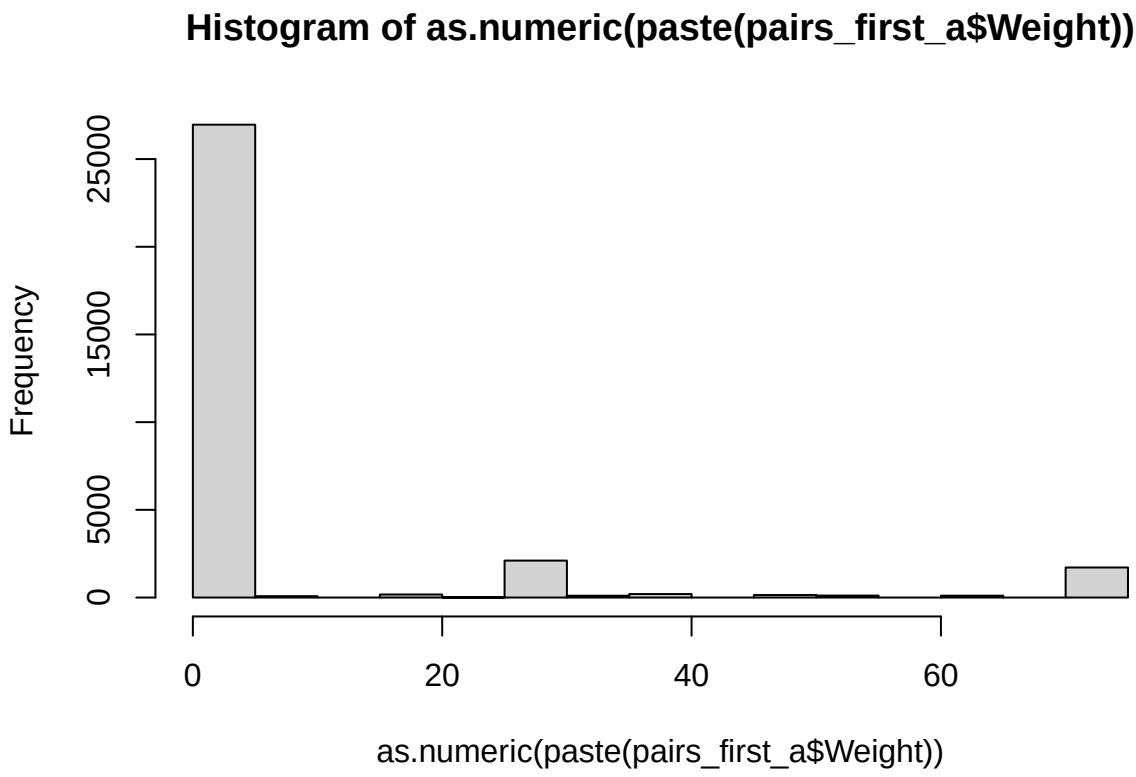
```
summary(firstw_a@Wdata[])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -9.535  -9.535  -9.535  -9.436  -9.535  71.918
```

```
pairs_first_a <- getPairs(firstw_a, min.weight = 0, single.rows = TRUE)
```

```
## ================================================================================
```

```
hist(as.numeric(paste(pairs_first_a$Weight)))
```




```
## Get "optimal" weight for setting threshold
threshold_first_a <- optimalThreshold(firstw_a)
```

```
## ================================================================================
```

```
threshold_first_a
```

```
## [1] 36.97144
```

```
## Print results
first_a_class <- emClassify(firstw_a, threshold.upper = threshold_first_a)
getTable(first_a_class)
```

```
## < table of extent 0 x 0 >
```

```
getErrorMeasures(first_a_class)
```

```
## $alpha
## [1] 0.2348178
##
## $beta
## [1] 0
##
## $accuracy
## [1] 0.9999215
##
## $precision
## [1] 1
##
## $sensitivity
## [1] 0.7651822
##
## $specificity
## [1] 1
##
## $ppv
## [1] 1
##
## $npv
## [1] 0.9999215
```

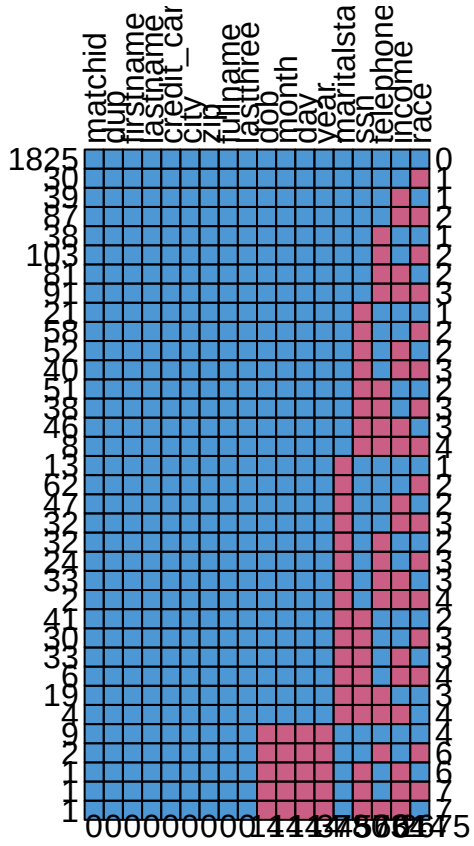
```
## Generate the classification table that shows the true status of the matches (from our identity vecto
table_1_a <- table(ff:::as.data.frame.ffdf(first_a_class@prediction))
table_1_a
```

```
##
##       N       P       L
## 8128577       0    2079
```

For the first additional linkage experiment with names, using Levenshtein distance, the optimal threshold we obtain is 36.97144. We get a fair to high sensitivity of 0.7651822 and a high precision of 1. The optimal threshold we get is 36.97144. There are 2079 links and 8128577 non-links.

Adding more variables for blocking and assisting linkage

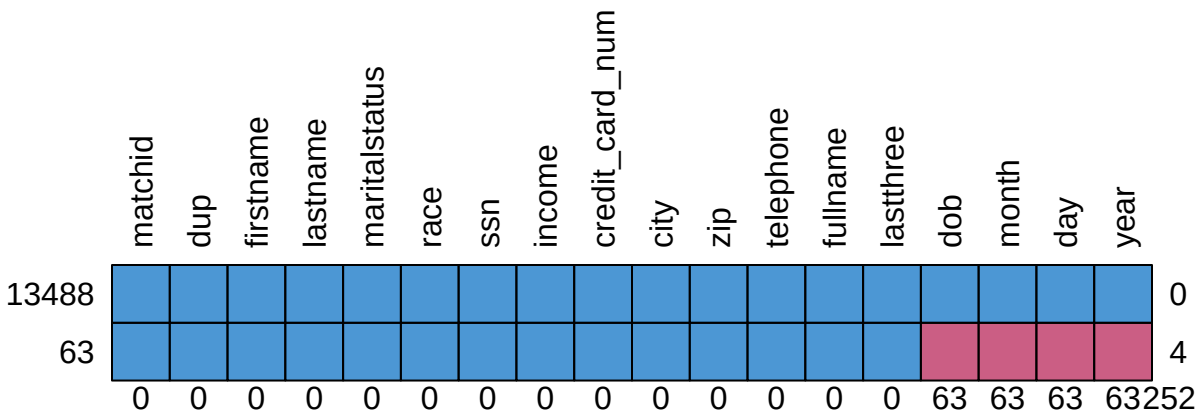
```
## Examine degree of data completion in the two data frames
mice::md.pattern(surveydata, rotate.names = TRUE)
```




```
##      matchid dup firstname lastname credit_card_num city zip fullname lastthree
## 1825       1   1         1        1               1    1   1        1         1
## 30         1   1         1        1               1    1   1        1         1
## 39         1   1         1        1               1    1   1        1         1
## 87         1   1         1        1               1    1   1        1         1
## 38         1   1         1        1               1    1   1        1         1
## 103        1   1         1        1               1    1   1        1         1
## 81         1   1         1        1               1    1   1        1         1
## 91         1   1         1        1               1    1   1        1         1
## 21         1   1         1        1               1    1   1        1         1
## 58         1   1         1        1               1    1   1        1         1
## 52         1   1         1        1               1    1   1        1         1
## 40         1   1         1        1               1    1   1        1         1
## 51         1   1         1        1               1    1   1        1         1
## 38         1   1         1        1               1    1   1        1         1
## 46         1   1         1        1               1    1   1        1         1
## 8          1   1         1        1               1    1   1        1         1
## 13         1   1         1        1               1    1   1        1         1
## 62         1   1         1        1               1    1   1        1         1
## 47         1   1         1        1               1    1   1        1         1
## 32         1   1         1        1               1    1   1        1         1
## 32         1   1         1        1               1    1   1        1         1
## 24         1   1         1        1               1    1   1        1         1
## 33         1   1         1        1               1    1   1        1         1
## 2          1   1         1        1               1    1   1        1         1
## 41         1   1         1        1               1    1   1        1         1
```

```
## 30          1   1          1         1                1    1   1         1          1
## 33          1   1          1         1                1    1   1         1          1
## 6           1   1          1         1                1    1   1         1          1
## 19          1   1          1         1                1    1   1         1          1
## 4           1   1          1         1                1    1   1         1          1
## 9           1   1          1         1                1    1   1         1          1
## 2           1   1          1         1                1    1   1         1          1
## 1           1   1          1         1                1    1   1         1          1
## 1           1   1          1         1                1    1   1         1          1
## 1           1   1          1         1                1    1   1         1          1
##             0   0          0         0                0    0   0         0          0
##       dob month day year maritalstatus ssn telephone income race
## 1825    1     1   1    1             1   1         1      1    1    0
## 30      1     1   1    1             1   1         1      1    0    1
## 39      1     1   1    1             1   1         1      0    1    1
## 87      1     1   1    1             1   1         1      0    0    2
## 38      1     1   1    1             1   1         0      1    1    1
## 103     1     1   1    1             1   1         0      1    0    2
## 81      1     1   1    1             1   1         0      0    1    2
## 91      1     1   1    1             1   1         0      0    0    3
## 21      1     1   1    1             1   0         1      1    1    1
## 58      1     1   1    1             1   0         1      1    0    2
## 52      1     1   1    1             1   0         1      0    1    2
## 40      1     1   1    1             1   0         1      0    0    3
## 51      1     1   1    1             1   0         0      1    1    2
## 38      1     1   1    1             1   0         0      1    0    3
## 46      1     1   1    1             1   0         0      0    1    3
## 8       1     1   1    1             1   0         0      0    0    4
## 13      1     1   1    1             0   1         1      1    1    1
## 62      1     1   1    1             0   1         1      1    0    2
## 47      1     1   1    1             0   1         1      0    1    2
## 32      1     1   1    1             0   1         1      0    0    3
## 32      1     1   1    1             0   1         0      1    1    2
## 24      1     1   1    1             0   1         0      1    0    3
## 33      1     1   1    1             0   1         0      0    1    3
## 2       1     1   1    1             0   1         0      0    0    4
## 41      1     1   1    1             0   0         1      1    1    2
## 30      1     1   1    1             0   0         1      1    0    3
## 33      1     1   1    1             0   0         1      0    1    3
## 6       1     1   1    1             0   0         1      0    0    4
## 19      1     1   1    1             0   0         0      1    1    3
## 4       1     1   1    1             0   0         0      0    1    4
## 9       0     0   0    0             1   1         1      1    1    4
## 2       0     0   0    0             1   1         0      1    0    6
## 1       0     0   0    0             1   0         1      0    1    6
## 1       0     0   0    0             1   0         1      0    0    7
## 1       0     0   0    0             1   0         0      0    1    7
##        14    14  14   14           378 450       573    604  614 2675
```

```
mice::md.pattern(admindata, rotate.names = TRUE)
```




```
##       matchid dup firstname lastname maritalstatus race ssn income
## 13488       1   1         1        1             1    1   1      1
## 63          1   1         1        1             1    1   1      1
##             0   0         0        0             0    0   0      0
##       credit_card_num city zip telephone fullname lastthree dob month day year
## 13488               1    1   1         1        1         1   1     1   1    1
## 63                  1    1   1         1        1         1   0     0   0    0
##                     0    0   0         0        0         0  63    63  63   63
##
## 13488   0
## 63      4
##       252
```

ZIP code seems to be complete within the two datasets, so I'd like to use it as the blocking variable. SSN and credit card number vary person by person if data is correctly written and is not modified or changed for privacy protection, so these two variables will be used for assisting the linking in the next few data linkage experiment.

Second experiment: link by name, SSN, and credit card number with more blocking variables (city and zip)

The second experiment link the two datasets by comparing similarity of name, SSN, and credit card number with more blocking variables to hopefully increase precision. Jaro-Winkler is applied to this experiment.

```
## Compare name, SSN, and credit card number and block with "city" and "zip"
second <- RLBigDataLinkage(surveydata, admindata, identity1=identity_survey, identity2 = identity_admin
### Jaro-Winkler distance is implemented initially to measure the similarity between strings. The cutof

## Obtain the summary weights for the potential links
secondw <- emWeights(second, cutoff = 0.95)
```

```
## ================================================================================
```

```
summary(secondw@Wdata[])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -14.41  -14.41  -14.41  -13.86  -14.41   66.20
```

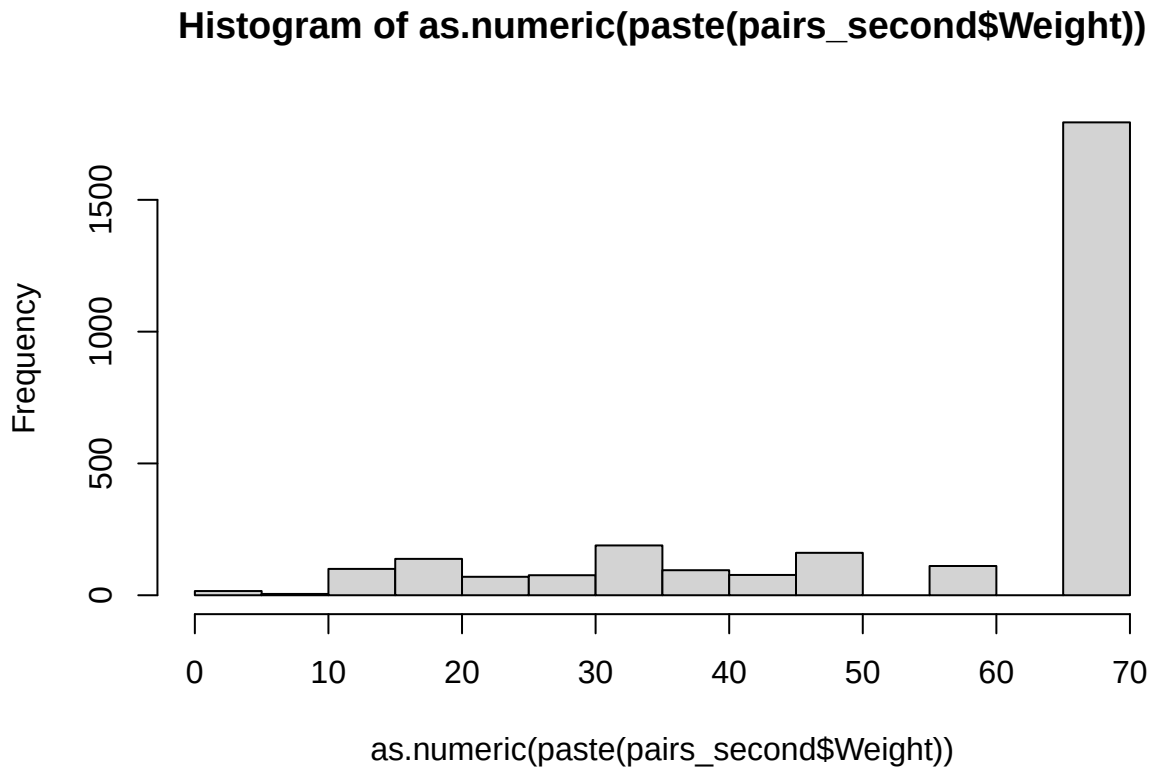
```
pairs_second <- getPairs(secondw, min.weight = 0, single.rows = TRUE)
```

```
## ================================================================================
```

```
summary(pairs_second$Weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.399  44.714  66.204  54.248  66.204  66.204
```

```
hist(as.numeric(paste(pairs_second$Weight)))
```



```
## Get "optimal" weight for setting threshold
threshold_second <- optimalThreshold(secondw)
```

```
## ============================================================================
```

```
threshold_second
```

```
## [1] 0.5385927
```

```
## Print results
second_class <- emClassify(secondw, threshold.upper = threshold_second)
getTable(second_class)
```

```
## < table of extent 0 x 0 >
```

```
getErrorMeasures(second_class)
```

```
## $alpha
## [1] 0.002576371
##
## $beta
## [1] 0.0002896603
##
## $accuracy
## [1] 0.9996957
##
## $precision
## [1] 0.9569209
##
## $sensitivity
## [1] 0.9974236
##
## $specificity
## [1] 0.9997103
##
## $ppv
## [1] 0.9569209
##
## $npv
## [1] 0.9999834
```

```
## Generate the classification table that shows the true status of the matches (from our identity vecto
table_2 <- table(ff::as.data.frame.ffdf(second_class@prediction))
table_2
```

```
##
##      N      P      L
## 421068      0   2832
```

The second experiment obtains a high sensitivity score of 0.9974236 and a high precision score of 0.9569209. The optimal threshold we get is 0.5385927. There are 2832 links and 421068 non-links.

Third experiment: link by name, SSN, and credit card number with more blocking variables (city and zip)

The third experiment link the two datasets by comparing similarity of name, SSN, and credit card number with more blocking variables to hopefully increase precision. Levenshtein is applied to this experiment.

```
## Compare name, SSN, and credit card number and block with "city" and "zip"
third <- RLBigDataLinkage(surveydata, admindata, identity1=identity_survey, identity2 = identity_admin,
### Jaro-Winkler distance is implemented initially to measure the similarity between strings. The cutof

## Obtain the summary weights for the potential links
thirdw <- emWeights(third, cutoff = 0.95)
```

```
## ================================================================================
```

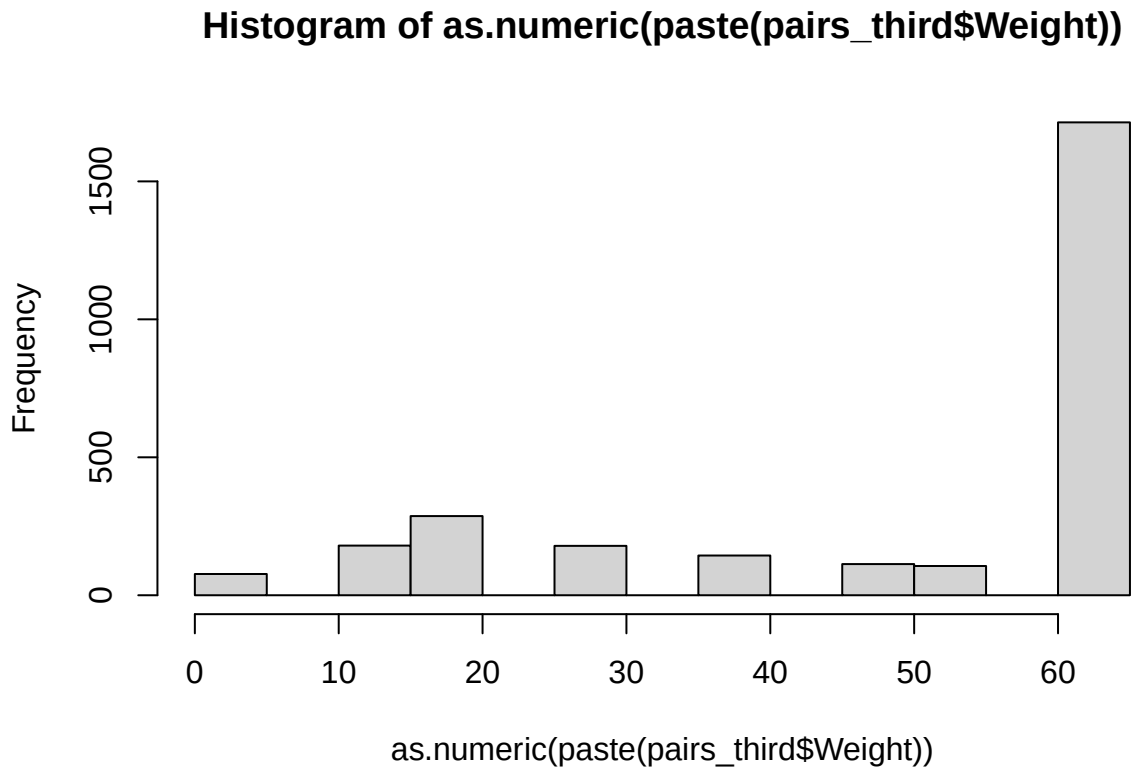
```
summary(thirdw@Wdata[])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -12.92  -12.92  -12.92  -12.44  -12.92   61.28
```

```
pairs_third <- getPairs(thirdw, min.weight = 0, single.rows = TRUE)
```

```
## ================================================================================
```

```
hist(as.numeric(paste(pairs_third$Weight)))
```



```
## Get optimal weight for setting threshold
threshold_third <- optimalThreshold(thirdw)
```

```
## ============================================================================
```

```
threshold_third
```

```
## [1] 0.1714824
```

```
## Print results
third_class <- emClassify(thirdw, threshold.upper = threshold_third)
getTable(third_class)
```

```
## < table of extent 0 x 0 >
```

```
getErrorMeasures(third_class)
```

```
## $alpha
## [1] 0.01435407
##
## $beta
## [1] 0.0002896603
##
## $accuracy
## [1] 0.9996202
##
## $precision
## [1] 0.9564286
##
## $sensitivity
## [1] 0.9856459
##
## $specificity
## [1] 0.9997103
##
## $ppv
## [1] 0.9564286
##
## $npv
## [1] 0.9999074
```

```
## Generate the classification table that shows the true status of the matches (from our identity vecto
table_3 <- table(ff:::as.data.frame.ffdf(third_class@prediction))
table_3
```

```
##
##      N      P      L
## 421100      0   2800
```

The third experiment obtains a high sensitivity score of 0.9856459 and a high precision score of 0.9564286. The optimal threshold we get is 0.1714824. There are 2800 links and 421100 non-links.

Fourth experiment: link by name, SSN, and credit card number with DOB and race as the blocking variable

DOB can rarely be exactly the same. While race is sometimes considered a sensitive information that respondents may not be willing to reveal, it is still an important characteristic of recognizing different people. The fourth experiment link the two datasets by comparing similarity of name, SSN, and credit card number with DOB and race as blocking variables. Jaro-Winkler is applied to this experiment.

```
## Compare name, SSN, and credit card number and block with "DOB" and "race"
fourth <- RLBigDataLinkage(surveydata, admindata, identity1=identity_survey, identity2 = identity_admin
### Jaro-Winkler distance is implemented initially to measure the similarity between strings. The cutof

## Obtain the summary weights for the potential links
fourthw <- emWeights(fourth, cutoff = 0.95)
```

```
## ================================================================================
```

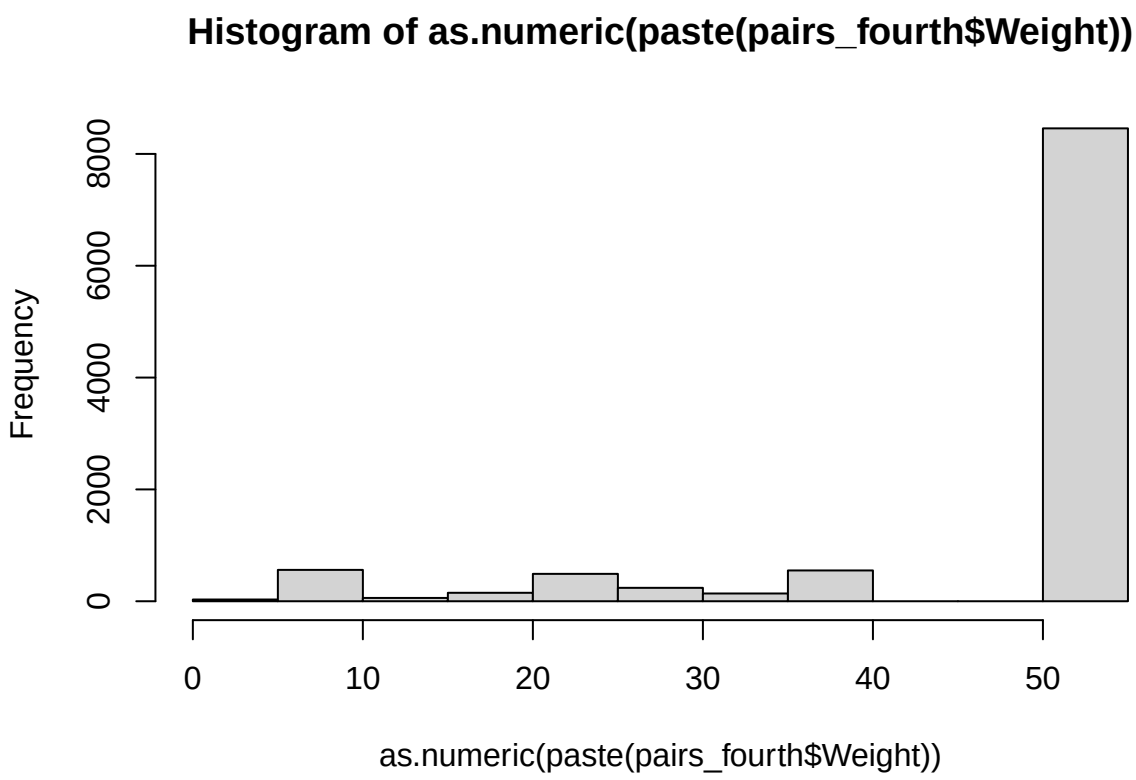
```
summary(fourthw@Wdata[])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -19.137 -19.137 -19.137  -5.885 -19.137  50.005
```

```
pairs_fourth <- getPairs(fourthw, min.weight = 0, single.rows = TRUE)
```

```
## ================================================================================
```

```
hist(as.numeric(paste(pairs_fourth$Weight)))
```




```
## Get optimal weight for setting threshold
threshold_fourth <- optimalThreshold(fourthw)
```

```
## ============================================================================
```

```
threshold_fourth
```

```
## [1] 37.5511
```

```
## Print results
fourth_class <- emClassify(fourthw, threshold.upper = threshold_fourth)
getTable(fourth_class)
```

```
## < table of extent 0 x 0 >
```

```
getErrorMeasures(fourth_class)
```

```
## $alpha
## [1] 0.1999072
##
## $beta
## [1] 0.1398968
##
```

```
## $accuracy
## [1] 0.857585
##
## $precision
## [1] 0.2003252
##
## $sensitivity
## [1] 0.8000928
##
## $specificity
## [1] 0.8601032
##
## $ppv
## [1] 0.2003252
##
## $npv
## [1] 0.9899221
```

```
## Generate the classification table that shows the true status of the matches (from our identity vecto
table_4 <- table(ff:::as.data.frame.ffdf(fourth_class@prediction))
table_4
```

```
##
##     N     P     L
## 42767     0  8611
```

The fourth experiment obtains a high sensitivity score of 0.8000928 and a poor precision score of 0.2003252. The optimal threshold we get is 37.5511. There are 8611 links and 42767 non-links.

Fifth experiment: link by name, SSN, and credit card number with DOB and race as the blocking variable

The fifth experiment link the two datasets by comparing similarity of name, SSN, and credit card number with DOB and race as blocking variables. Levenshtein is applied to this experiment.

```
## Compare name, SSN, and credit card number and block with "DOB" and "race"
fifth <- RLBigDataLinkage(surveydata, admindata, identity1=identity_survey, identity2 = identity_admin,
### Jaro-Winkler distance is implemented initially to measure the similarity between strings. The cutof

## Obtain the summary weights for the potential links
fifthw <- emWeights(fifth, cutoff = 0.95)
```

```
## ================================================================================
```

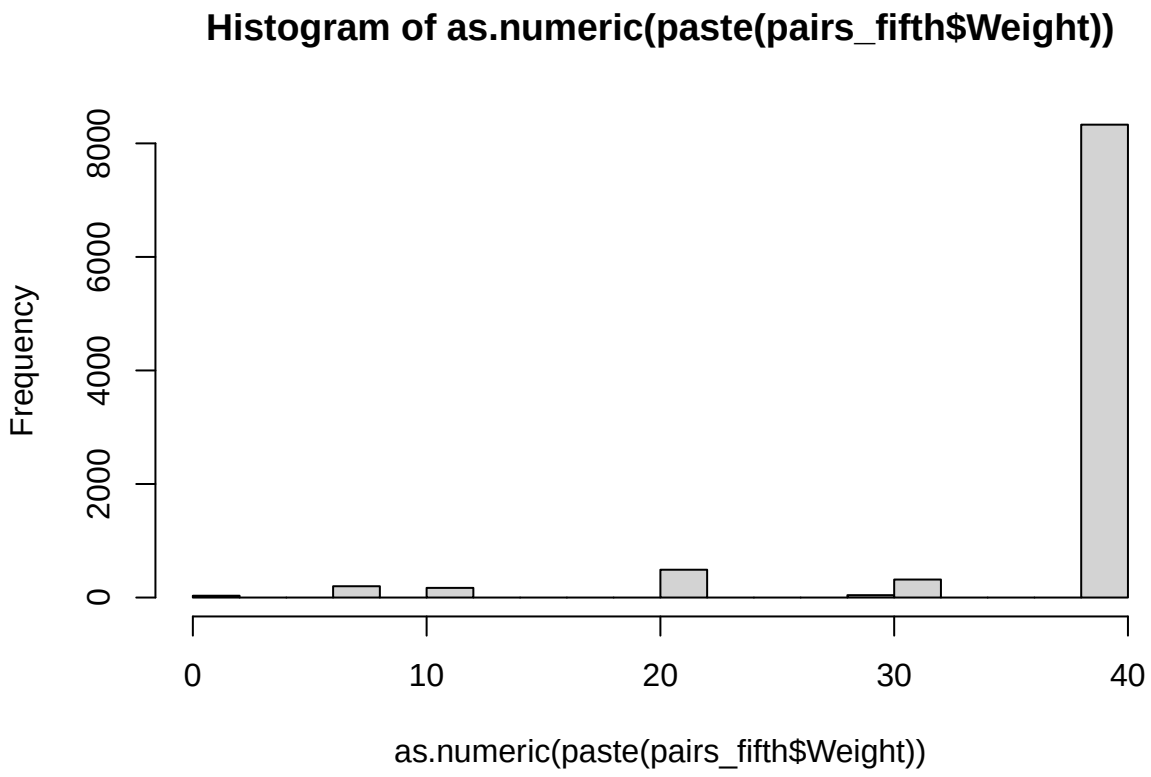
```
summary(fifthw@Wdata[])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -22.78  -22.78  -22.78  -11.19  -22.78   39.70
```

```
pairs_fifth <- getPairs(fifthw, min.weight = 0, single.rows = TRUE)
```

```
## ================================================================================
```

```
hist(as.numeric(paste(pairs_fifth$Weight)))
```




```
## Get optimal weight for setting threshold
threshold_fifth <- optimalThreshold(fifthw)
```

```
## ================================================================================
```

```
threshold_fifth
```

```
## [1] 30.11722
```

```
## Print results
fifth_class <- emClassify(fifthw, threshold.upper = threshold_fifth)
getTable(fifth_class)
```

```
## < table of extent 0 x 0 >
```

```
getErrorMeasures(fifth_class)
```

```
## $alpha
## [1] 0.2096475
##
## $beta
## [1] 0.1382512
##
## $accuracy
## [1] 0.8587528
##
## $precision
## [1] 0.2002585
##
## $sensitivity
## [1] 0.7903525
##
## $specificity
## [1] 0.8617488
##
## $ppv
## [1] 0.2002585
##
## $npv
## [1] 0.9894563
```

```
## Generate the classification table that shows the true status of the matches (from our identity vecto
table_5 <- table(ff:::as.data.frame.ffdf(fifth_class@prediction))
table_5
```

```
##
##     N     P     L
## 42869     0  8509
```

The fifth experiment obtains a fair to high sensitivity score of 0.7903525 and a poor precision score of 0.2002585. The optimal threshold we get is 30.11722. There are 8509 links and 42869 non-links.

Sixth experiment: link by name, SSN, and credit card number with DOB and city as the blocking variable

The sixth experiment link the two datasets by comparing similarity of name, SSN, and credit card number with DOB and city as blocking variables. Levenshtein is applied to this experiment.

```
## Compare name, SSN, and credit card number and block with "DOB" and "city"
sixth <- RLBigDataLinkage(surveydata, admindata, identity1=identity_survey, identity2 = identity_admin,
### Jaro-Winkler distance is implemented initially to measure the similarity between strings. The cutof

## Obtain the summary weights for the potential links
sixthw <- emWeights(sixth, cutoff = 0.95)
```

```
## ================================================================================
```

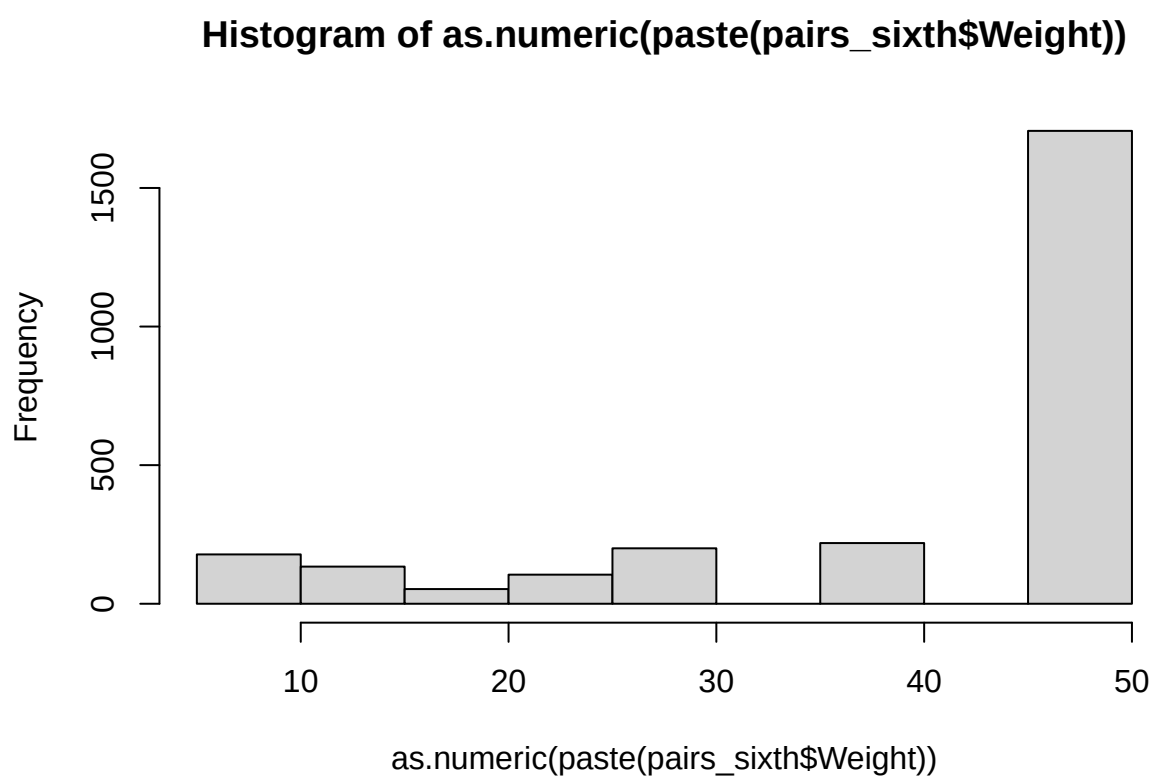
```
summary(sixthw@Wdata[])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -13.813 -13.813 -13.813  -8.235 -13.813  49.176
```

```
pairs_sixth <- getPairs(sixthw, min.weight = 0, single.rows = TRUE)
```

```
## ================================================================================
```

```
hist(as.numeric(paste(pairs_sixth$Weight)))
```




```
## Get optimal weight for setting threshold
threshold_sixth <- optimalThreshold(sixthw)
```

```
## ================================================================================
```

```
threshold_sixth
```

```
## [1] -2.576515
```

```
## Print results
sixth_class <- emClassify(sixthw, threshold.upper = threshold_sixth)
getTable(sixth_class)
```

```
## < table of extent 0 x 0 >
```

```
getErrorMeasures(sixth_class)
```

```
## $alpha
## [1] 0.01442308
##
## $beta
## [1] 0.0003082207
##
## $accuracy
## [1] 0.99819
##
## $precision
## [1] 0.9973802
##
## $sensitivity
## [1] 0.9855769
##
## $specificity
## [1] 0.9996918
##
## $ppv
## [1] 0.9973802
##
## $npv
## [1] 0.9982852
```

```
## Generate the classification table that shows the true status of the matches (from our identity vecto
table_6 <- table(ff:::as.data.frame.ffdf(sixth_class@prediction))
table_6
```

```
##
##     N     P     L
## 22743     0  2672
```

The sixth experiment obtains a high sensitivity score of 0.9855769 and a high precision score of 0.9973802. The optimal threshold we get is -2.576515. There are 2672 links and 22743 non-links.

Conclusion of the experiments

The six experiments differ in linking variables, blocking variables, and/or comparison method and tolerance of differences. They compose of multiple treatment groups and control groups for comparison. Each pair of experiments is concluded in the following:
1. The example and the first experiments both implement data linkage by only comparing name, and using "city" as the blocking variable. Jaro-Winkler comparison method with manually selected thresholds generate

a lower sensitivity score and a lower precision score than Levenshtein with optimal threshold.
2. The first additional experiment is a further examination of the effects of more linking variables to sensitivity and precision scores. It implements data linkage by comparing name, SSN, and credit card number, and uses "city" as the blocking variable. Both the first and the first additional experiment apply Levenshtein method for comparison, and utilize machine-selected optimized threshold. With the join of more linking variables, both the sensitivity score and the precision score are increased.
3. The second and the third experiments both implement data linkage by comparing name, SSN, and credit card number, and using "city" and "zip" as the blocking variables. Both apply optimal thresholds selected by the function as well. Jaro-Winkler comparison method generates a higher sensitivity score and a higher precision score than Levenshtein does. But the differences are slight.
4. The fourth and the fifth experiments both implement data linkage by comparing name, SSN, and credit card number, and using "DOB" and "race" as the blocking variables. Both apply optimal thresholds selected by the function. Jaro-Winkler comparison method generates a higher sensitivity score and a higher precision score than Levenshtein does. But the differences are very slight.
5. The second and the fourth experiments both implement data linkage by comparing name, SSN, and credit card number, while the former uses "city" and "zip" as the blocking variables, the latter uses "DOB" and "race." Both apply optimal thresholds selected by the function and Jaro-Winkler comparison method. "City" and "zip" as blocking variables generates a higher sensitivity score and a much higher precision score than "DOB" and "race" does.
6. The third and the fifth experiments both implement data linkage by comparing name, SSN, and credit card number, while the former uses "city" and "zip" as the blocking variables, the latter uses "DOB" and "race." Both apply optimal thresholds selected by the function and Levenshtein comparison method. "City" and "zip" as blocking variables generates a higher sensitivity score and a much higher precision score than "DOB" and "race" does.
7. The fifth and the sixth experiments both implement data linkage by comparing name, SSN, and credit card number, while the former uses "DOB" and "race" as the blocking variables, the latter uses "DOB" and "city." Both apply optimal thresholds selected by the function and Levenshtein comparison method. "DOB" and "race" as blocking variables generates a lower sensitivity score and a much lower precision score than "DOB" and "city" does.
8. The third and the sixth experiments both implement data linkage by comparing name, SSN, and credit card number, while the former uses "city" and "zip" as the blocking variables, the latter uses "DOB" and "city." Both apply optimal thresholds selected by the function and Levenshtein comparison method. "City" and "zip" as blocking variables generates a slightly higher sensitivity score and a lower precision score than "DOB" and "city" does.
9. Among all sets of experiments, the fourth experiment with name, SSN, and credit card number as the linking variables, "DOB" and "race" as the blocking variables, optimal thresholds selected by the function, and Jaro-Winkler method has the greatest links of 8611.
10. Without blocking variables and level of string comparison, the sensitivity remains similarly while the precision performs poorly, comparing the initial experiment and the first experiment. In addition, both links and non-links are tremendous.

There is not a single set of experiment that results in the highest sensitivity score and precision score simultaneously. However, the sixth experiment with name, SSN, credit card number as the linking variables, DOB and city as the blocking variables using Levenshtein and optimized threshold performs the best in precision with a score of 0.9973802. There are 2672 links and 22743 non-links. The second experiment with name, SSN, credit card number as the linking variables, city and zip as the blocking variables, using Jaro-Winkler and optimized threshold performs the best in sensitivity of 0.9974236. There are 2832 links and 421068 non-links.

To conclude, across our sample, the following patterns stand:
-Jaro-Winkler usually causes a higher score of sensitivity than Levenshtein does, while precision score varies; and
-adding "SSN" and "credit card number" to linking variables is effective in improving sensitivity and precision performances; and
-adding "zip", "DOB," and/or "city" to blocking variables is effective in improving sensitivity and precision

performances; and
-“race” is not a good blocking variable since both sensitivity score and precision score are lowered (because it has many missing values).