Assignment 1

SURV 622

Sagnik Chakravarty

Instruction

Using the RecordLinkage software in R, you will

1) Test different specifications for determining the links between an "administrative" data set and a "survey" data set that have been created for the purpose of this exercise

2) Write a report that describes the specifications that were tested and discusses the results.

The two data files are named admindata.csv and surveydata.csv; they are available to be downloaded from the course website. In addition to a set of potential linking variables, these files also include a common record identifier that is not intended for use in linking but that can be used to determine whether a proposed link is or is not a true match. A sample program that illustrates relevant syntax for the RecordLinkage package in R will be provided and discussed in class.

You should experiment with linking specifications that vary along the following dimensions:

- Choice of linking variables
- Choice of blocking variables, if any
- Whether strings must match exactly or to some lesser tolerance as specified by a string comparison (e.g., Jaro Winkler)

You should consider

1) linking specifications with all pairs designated as either links or non-links

2) linking specifications with an intermediate category of pairs requiring clerical review.

Some guidelines for this project:

- Be systematic in your approach. Start with a "deterministic" approach using a core set of linking variables (e.g., name and year of birth), no blocking and no string comparison (i.e., requiring that linking variables match exactly). Then explore what happens when you vary these choices—adding linking variables, adding blocking variables, using a string comparison and, when using a string comparison, setting a higher or lower agreement threshold.
- Variables used for blocking should not have missing values. City, zip code or first letter of last name, for example, are good candidates for blocking; race or marital status are missing for a significant number of survey records and are not good choices.
- Consider *precision* and *sensitivity* as key metrics for assessing how well a given specification performed.
- In specifications that allow for clerical review, consider the number of cases assigned for clerical review.

All files (survey data, admin data, this document, and example code) are available in the Files tab under Assignment #1.

Loading and processing the data

```
library(RecordLinkage)
library(knitr)
library(dplyr)
library(kableExtra)
# Importing datasets
sdfilepath = "surveydata.csv"
adfilepath = "admindata.csv"

# Reading the data
surveydata <- read.csv(sdfilepath, header=TRUE, sep=",", stringsAsFactors = FALSE)
admindata <- read.csv(adfilepath, header=TRUE, sep=",", stringsAsFactors = FALSE)

# Cleaning column names
colnames(surveydata) <- c("matchid","dup","firstname","lastname","maritalstatus",
                          "race","dob","ssn","income","credit_card_num","city",
                          "zip","telephone")
colnames(admindata) <- c("matchid","dup","firstname","lastname","maritalstatus",
                         "race","dob","ssn","income","credit_card_num","city",
                         "zip","telephone")
```

```r
# Removing empty rows
surveydata <- surveydata[complete.cases(surveydata[,1:2]), ]
admindata <- admindata[complete.cases(admindata[,1:2]), ]

# Creating derived columns
# 1. Full name (first initial + last name)
surveydata$fullname <- paste(substring(surveydata$firstname,1,1),
                             surveydata$lastname, sep='')
admindata$fullname <- paste(substring(admindata$firstname,1,1),
                            admindata$lastname, sep='')

# 2. Processing dates
surveydata$dob <- as.Date(surveydata$dob, format="%m/%d/%Y")
admindata$dob <- as.Date(admindata$dob, format="%m/%d/%Y")

# 3. Extracting date components
surveydata$month <- as.numeric(format(surveydata$dob, "%m"))
surveydata$day <- as.numeric(format(surveydata$dob, "%d"))
surveydata$year <- as.numeric(format(surveydata$dob, "%Y"))
admindata$month <- as.numeric(format(admindata$dob, "%m"))
admindata$day <- as.numeric(format(admindata$dob, "%d"))
admindata$year <- as.numeric(format(admindata$dob, "%Y"))

# 4. First three letters of lastname
surveydata$lastthree <- substring(surveydata$lastname,1,3)
admindata$lastthree <- substring(admindata$lastname,1,3)

# Creating identity vectors
identity_survey <- surveydata$matchid
identity_admin <- admindata$matchid


print('----------Survey Data------------')
```

```
[1] "----------Survey Data------------"
```

```r
kable(head(surveydata),
      format = 'latex',
      booktabs = TRUE) %>%
  kable_styling(latex_options = c('hold_position',
                                  'scale_down'))
```

matchid	dup	firstname	lastname	maritalstatus	race	dob	ssn	income	credit_card_num	city	zip	telephone	fullname	month	day	year	lastthree
1003	1	Tyron	Rau		White	1991-05-25		NA	1915 3160 2544 39912	Washington DC	20004	956-224-0530	TRau	5	25	1991	Rau
1011	1	Ethan	Claahke	Married	White	1981-09-03	638-11-0983	NA	2741 2201 8604 48890	Washington DC	20002		EClaahke	9	3	1981	Cla
1013	0	Taliah	Staude	Married	White	1954-06-24	176-03-9379	166134.9	4029 4369 6201 0727	Washington DC	20018	313-519-3459	TStaude	6	24	1954	Sta
1017	0	Kiara	Kellow	Married	White	1957-02-21	305-22-0947	49913.3	1116 6409 0941 9097	Washington DC	20005	360-672-5947	KKellow	2	21	1957	Kel
1028	0	Gabrielle	Barbuto	Married	White	1974-01-06	655-87-0523	33234.6	9772 9935 9524 1541	Washington DC	20017	203-624-7611	GBarbuto	1	6	1974	Bar
1029	1	David	Mierix	Divorced		1965-11-27	393-35-6457	NA	4957 4517 0307 9302	Washington DC	20005		DMierix	11	27	1965	Mie

```
print('----------Admin Data------------')
```

```
[1] "----------Admin Data------------"
```

```
kable(head(admindata),
      format = 'latex',
      booktabs = TRUE) %>%
  kable_styling(latex_options = c('hold_position',
                                  'scale_down'))
```

matchid	dup	firstname	lastname	maritalstatus	race	dob	ssn	income	credit_card_num	city	zip	telephone	fullname	month	day	year	lastthree
1001	0	Luke	Jaric	Married	Asian	1968-07-10	376-76-9849	10103.9	4886 2506 7467 0819	Washington DC	20022	623-267-1889	LJaric	7	10	1968	Jar
1002	0	Kayne	Longhurst	Married	Hispanic	1973-08-08	257-26-0201	31206.9	2025 0126 7554 3613	Washington DC	20006	417-283-2412	KLonghurst	8	8	1973	Lon
1003	0	Tyron	Rau	Married	White	1991-05-25	003-91-4610	48677.3	1915 3160 2544 9912	Washington DC	20004	956-224-0530	TRau	5	25	1991	Rau
1005	0	Daniel	Grimwade	Married	Asian	1954-07-10	614-55-6413	51146.0	7630 0951 2958 0709	Washington DC	20018	361-333-8315	DGrimwade	7	10	1954	Gri
1006	0	Harvey	Pascoe	Divorced	Black	1980-05-15	511-49-1782	49767.9	1967 3587 9147 6740	Washington DC	20013	804-482-6364	HPascoe	5	15	1980	Pas
1007	0	Trevor	George	Never Married	White	1978-08-11	505-09-9675	38463.2	5099 4144 5487 5601	Washington DC	20020	828-723-6043	TGeorge	8	11	1978	Geo

```
cat("The dimesions for Survey data:\t", dim(surveydata),
    "\nThe dimesions for Admin data:\t", dim(admindata))
```

```
The dimesions for Survey data:   3000 18
The dimesions for Admin data:    13551 18
```

```
cat("\nMissing values:\n")
```

```

Missing values:
```

```
cat("Survey DOB NAs:", sum(is.na(surveydata$dob)), "\n")
```

```
Survey DOB NAs: 11
```

```
cat("Admin DOB NAs:", sum(is.na(admindata$dob)), "\n")
```

```
Admin DOB NAs: 49
```

We have a survey data of dimensions 3000×18 while a larger administrative data 13551×18, the missing data I have too keep them in mind while doing record linkage as they might create issues while analysis

No Blocking Approach

Exact Matching

```
# Approach 1: Exact Matching (No blocking)
# Jaro Winkler
e_jaro <- RLBigDataLinkage(
  dataset1 = surveydata,
  dataset2 = admindata,
  blockfld = NULL,
  identity1 = identity_survey,
  identity2 = identity_admin,
  strcmp = TRUE,
  strcmpfun = 'jarowinkler',
  exclude = c(1:2, 5:13, 15:18)
)

e_jaro_w <- emWeights(e_jaro, cutoff = 0.7, verbose = TRUE)
```

```
================================================================================
```

```
e_jaro_class <- emClassify(e_jaro_w, threshold.upper = 22)
```

```
#Levenshtein
e_leven <- RLBigDataLinkage(
  dataset1 = surveydata,
  dataset2 = admindata,
  identity1 = identity_survey,
  identity2 = identity_admin,
  blockfld = NULL,
  strcmp = TRUE,
  strcmpfun = "levenshtein",
  exclude = c(1:2, 5:13, 15:18)
```

```
)
e_leven_w <- emWeights(e_leven, cutoff = 0.95)
```

```
Count pattern frequencies...
```

```
Run EM algorithm...
```

```
================================================================================
```

```
e_leven_class <- emClassify(e_leven_w, threshold.upper = 0)
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to min; returning
Inf
```

```
Warning in max(x, na.rm = na.rm): no non-missing arguments to max; returning
-Inf
```

```
# Phonetics
e_phone <- RLBigDataLinkage(
  dataset1 = surveydata,
  dataset2 = admindata,
  identity1 = identity_survey,
  identity2 = identity_admin,
  blockfld = NULL,
  phonetic = TRUE,
  exclude = c(1:2, 5:13, 15:18)
)
e_phone_w <- emWeights(e_phone, cutoff = 0.95)
```

```
Count pattern frequencies...
```

```
Run EM algorithm...
```

```
================================================================================
```

```
e_phone_class <- emClassify(e_phone_w, threshold.upper = 0)
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to min; returning
Inf
```

```
Warning in max(x, na.rm = na.rm): no non-missing arguments to max; returning
-Inf
```

```r
# Extracting error measures from each method
err_jarowinkler <- getErrorMeasures(e_jaro_class)
err_levenshtein <- getErrorMeasures(e_leven_class)
err_phonetic     <- getErrorMeasures(e_phone_class)

# Creating a data frame summarizing key metrics
comparison_table <- data.frame(
  Method = c("Jarowinkler", "Levenshtein", "Phonetic"),
  Accuracy = c(err_jarowinkler$accuracy,
               err_levenshtein$accuracy,
               err_phonetic$accuracy),
  Precision = c(err_jarowinkler$precision,
                err_levenshtein$precision,
                err_phonetic$precision),
  Sensitivity = c(err_jarowinkler$sensitivity,
                  err_levenshtein$sensitivity,
                  err_phonetic$sensitivity),
  Specificity = c(err_jarowinkler$specificity,
                  err_levenshtein$specificity,
                  err_phonetic$specificity)
)

print(comparison_table)
```

```
       Method  Accuracy   Precision Sensitivity Specificity
1 Jarowinkler 0.9913301 0.007649793   1.0000000   0.9913296
2 Levenshtein 0.9995193 0.102945341   0.8027236   0.9995325
3    Phonetic 0.9988615 0.053386912   0.9584100   0.9988642
```

- **Jaro-Winkler Method**:
  - **Accuracy**: Very high at 0.99133, indicating the method performed well overall.
  - **Precision**: Very low at 0.00765, suggesting that it produced many false positives compared to actual matches.

  - **Sensitivity**: Perfect at 1.00000, meaning it identified all true positive pairs (i.e., it did not miss any matches).
  - **Specificity**: High at 0.99133, indicating it effectively identified non-matching pairs, but still had a low precision due to many false positives.

- **Levenshtein Method**:
  - **Accuracy**: Extremely high at 0.99952, indicating very few errors in classification.
  - **Precision**: Higher than Jaro-Winkler at 0.10295, but still relatively low, showing that many of the predicted matches were incorrect.
  - **Sensitivity**: Moderate at 0.80272, meaning it missed a fair number of true positives.
  - **Specificity**: Very high at 0.99953, similar to Jaro-Winkler, showing it effectively identified non-matches.

- **Phonetic Method**:
  - **Accuracy**: High at 0.99886, with results close to Levenshtein.
  - **Precision**: Similar to Levenshtein at 0.05339, showing that many predicted matches were false positives.
  - **Sensitivity**: Very high at 0.95841, meaning it identified most true positives, though not as many as Jaro-Winkler.
  - **Specificity**: High at 0.99886, comparable to both Jaro-Winkler and Levenshtein.

Conclusion

- **Best method for accuracy**: **Levenshtein** with 0.99952 accuracy performed the best for minimizing errors, though its sensitivity was not as high as Jaro-Winkler's.
- **Best method for sensitivity**: **Jaro-Winkler** with perfect sensitivity (1.00000) but suffered from very low precision, meaning it tended to flag many false positives.
- **Best method for specificity**: **Levenshtein** and **Phonetic** both showed similarly high specificity scores, indicating that both methods are effective at identifying non-matching pairs.

Blocking

Blocking on City

```
# Approach 1: Exact Matching (City Blocking)
# JaroWinkler
b_jaro <- RLBigDataLinkage(
  dataset1 = surveydata,
  dataset2 = admindata,
  blockfld = 'city',
  identity1 = identity_survey,
  identity2 = identity_admin,
  strcmp = TRUE,
  strcmpfun = 'jarowinkler',
  exclude = c(1:2, 5:13, 15:18)
)

b_jaro_w <- emWeights(b_jaro, cutoff = 0.7, verbose = TRUE)
```

```
Count pattern frequencies...

Run EM algorithm...

================================================================================
```

```
b_jaro_class <- emClassify(b_jaro_w, threshold.upper = 22)
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to min; returning
Inf

Warning in max(x, na.rm = na.rm): no non-missing arguments to max; returning
-Inf
```

```
#levenshtein
b_leven <- RLBigDataLinkage(
  dataset1 = surveydata,
  dataset2 = admindata,
  identity1 = identity_survey,
```

```
  identity2 = identity_admin,
  blockfld = 'city',
  strcmp = TRUE,
  strcmpfun = "levenshtein",
  exclude = c(1:2, 5:13, 15:18)
)
b_leven_w <- emWeights(b_leven, cutoff = 0.95)
```

```
Count pattern frequencies...
Run EM algorithm...
```

```
================================================================================
```

```
b_leven_class <- emClassify(b_leven_w, threshold.upper = 0)
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to min; returning
Inf
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to max; returning
-Inf
```

```
# Phonetic
b_phone <- RLBigDataLinkage(
  dataset1 = surveydata,
  dataset2 = admindata,
  identity1 = identity_survey,
  identity2 = identity_admin,
  blockfld = 'city',
  phonetic = TRUE,
  exclude = c(1:2, 5:13, 15:18)
)
b_phone_w <- emWeights(b_phone, cutoff = 0.95)
```

```
Count pattern frequencies...
Run EM algorithm...
```

```
================================================================================
```

```
b_phone_class <- emClassify(b_phone_w, threshold.upper = 0)
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to min; returning
Inf
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to max; returning
-Inf
```

```
# Extracting error measures from each method
err_jarowinkler <- getErrorMeasures(b_jaro_class)
err_levenshtein <- getErrorMeasures(b_leven_class)
err_phonetic     <- getErrorMeasures(b_phone_class)

# Creating a data frame summarizing key metrics
comparison_table <- data.frame(
  Method = c("Jarowinkler", "Levenshtein", "Phonetic"),
  Accuracy = c(err_jarowinkler$accuracy,
               err_levenshtein$accuracy,
               err_phonetic$accuracy),
  Precision = c(err_jarowinkler$precision,
                err_levenshtein$precision,
                err_phonetic$precision),
  Sensitivity = c(err_jarowinkler$sensitivity,
                  err_levenshtein$sensitivity,
                  err_phonetic$sensitivity),
  Specificity = c(err_jarowinkler$specificity,
                  err_levenshtein$specificity,
                  err_phonetic$specificity)
)

print(comparison_table)
```

```
      Method  Accuracy  Precision Sensitivity Specificity
1 Jarowinkler 0.9915903 0.03821755   1.0000000   0.9915875
2 Levenshtein 0.9996800 0.51353897   0.8027236   0.9997458
3    Phonetic 0.9991091 0.26748844   0.9584100   0.9991227
```

1. **Jaro-Winkler Method**:
    - **Accuracy**: Slight increase to 0.99159, which is still quite high and indicates good overall performance.

- **Precision**: Low at 0.03822, though improved slightly compared to the previous result (0.00765). This still suggests a large number of false positives.
- **Sensitivity**: Perfect at 1.00000, meaning Jaro-Winkler still identifies all true matches, with no missed pairs.
- **Specificity**: High at 0.99159, similar to the previous result, suggesting that it continues to perform well in identifying true non-matches.

2. **Levenshtein Method**:
   - **Accuracy**: Improved significantly to 0.99968, suggesting better overall classification with fewer errors.
   - **Precision**: Improved substantially to 0.51354, a significant jump from the earlier result (0.10295). This shows that the number of false positives has decreased, making the model more reliable.
   - **Sensitivity**: Remains at 0.80272, indicating that the method still misses a significant portion of true positives.
   - **Specificity**: Very high at 0.99975, meaning that the method is very effective at correctly identifying non-matching pairs.
3. **Phonetic Method**:
   - **Accuracy**: Similar to the previous result (0.99886), showing stable overall performance.
   - **Precision**: Very low at 0.05339, though it hasn't changed much from the previous result (0.05339). This suggests that Phonetic still tends to produce false positives.
   - **Sensitivity**: High at 0.95841, meaning it is identifying most true positive pairs, though not as perfectly as Jaro-Winkler.
   - **Specificity**: Same as before, very high at 0.99886, indicating that it is good at identifying true non-matches.

Overall

- **Accuracy** has improved across all methods due to the introduction of the city blocking variable, as blocking generally helps reduce noise and unnecessary comparisons.
- **Jaro-Winkler** continues to perform excellently in terms of **sensitivity**, identifying all true matches, but still struggles with **precision**. It will need further adjustments or post-processing to filter out false positives, especially if precision is a priority.

- **Levenshtein** has made significant strides, especially in **precision** (0.51354), meaning fewer false positives are being made compared to before. However, **sensitivity** is still not perfect (0.80272), indicating that it might miss some true matches.
- **Phonetic** shows good **specificity** and **sensitivity**, with stable accuracy but still low **precision**, which is a common issue when phonetic matching is used for strings with minor differences in spelling.

Conclusion

- The use of **city as a blocking variable** has clearly improved the performance of all methods, especially **Levenshtein**, which now performs significantly better in terms of precision.
- However, **Jaro-Winkler** remains the most effective in identifying true positive pairs (with perfect sensitivity), though precision issues still persist.
- **Phonetic** works well for identifying true non-matches (high specificity) and offers decent sensitivity, but its precision still needs attention.

Blocking on ZIP

```
# JaroWinkler
b_jaro <- RLBigDataLinkage(
  dataset1 = surveydata,
  dataset2 = admindata,
  blockfld = 'zip',
  identity1 = identity_survey,
  identity2 = identity_admin,
  strcmp = TRUE,
  strcmpfun = 'jarowinkler',
  exclude = c(1:2, 5:13, 15:18)
)

b_jaro_w <- emWeights(b_jaro, cutoff = 0.7, verbose = TRUE)
```

```
Count pattern frequencies...

Run EM algorithm...

================================================================================
```

```
b_jaro_class <- emClassify(b_jaro_w, threshold.upper = 22)
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to min; returning
Inf
```

```
Warning in max(x, na.rm = na.rm): no non-missing arguments to max; returning
-Inf
```

```
#levenshtein
b_leven <- RLBigDataLinkage(
  dataset1 = surveydata,
  dataset2 = admindata,
  identity1 = identity_survey,
  identity2 = identity_admin,
  blockfld = 'zip',
  strcmp = TRUE,
  strcmpfun = "levenshtein",
  exclude = c(1:2, 5:13, 15:18)
)
b_leven_w <- emWeights(b_leven, cutoff = 0.95)
```

```
Count pattern frequencies...
Run EM algorithm...
```

```
================================================================================
```

```
b_leven_class <- emClassify(b_leven_w, threshold.upper = 0)
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to min; returning
Inf
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to max; returning
-Inf
```

```
# Phonetic
b_phone <- RLBigDataLinkage(
  dataset1 = surveydata,
```

```
  dataset2 = admindata,
  identity1 = identity_survey,
  identity2 = identity_admin,
  blockfld = 'zip',
  phonetic = TRUE,
  exclude = c(1:2, 5:13, 15:18)
)
b_phonetic_w <- emWeights(b_phone, cutoff = 0.95)
```

```
Count pattern frequencies...
Run EM algorithm...
```

```
================================================================================
```

```
b_phone_class <- emClassify(b_phone_w, threshold.upper = 0)
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to min; returning
Inf
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to max; returning
-Inf
```

```
# Extracting error measures from each method
err_jarowinkler <- getErrorMeasures(b_jaro_class)
err_levenshtein <- getErrorMeasures(b_leven_class)
err_phonetic     <- getErrorMeasures(b_phone_class)

# Creating a data frame summarizing key metrics
comparison_table <- data.frame(
  Method = c("Jarowinkler", "Levenshtein", "Phonetic"),
  Accuracy = c(err_jarowinkler$accuracy,
               err_levenshtein$accuracy,
               err_phonetic$accuracy),
  Precision = c(err_jarowinkler$precision,
                err_levenshtein$precision,
                err_phonetic$precision),
  Sensitivity = c(err_jarowinkler$sensitivity,
                  err_levenshtein$sensitivity,
                  err_phonetic$sensitivity),
```

```
  Specificity = c(err_jarowinkler$specificity,
                  err_levenshtein$specificity,
                  err_phonetic$specificity)
)

print(comparison_table)
```

```
      Method  Accuracy Precision Sensitivity Specificity
1 Jarowinkler 0.9935905       NaN   0.0000000   1.0000000
2 Levenshtein 0.9984477 0.9470256   0.8027236   0.9997103
3    Phonetic 0.9991091 0.2674884   0.9584100   0.9991227
```

- **Jaro-Winkler**:
  - **Problem with Sensitivity and Precision**: The fact that **sensitivity is 0** and **precision is NaN** suggests that Jaro-Winkler is not identifying any true matches at all under these conditions, possibly due to the blocking variable (zip code) not effectively grouping matching records. This could be due to mismatches between the records' zip codes or perhaps overly strict matching criteria. If the zip code field is not properly aligned or normalized between datasets, it could lead to missed matches, resulting in the observed lack of true positives.
  - **High Specificity**: Despite the sensitivity problem, the method does perfectly identify non-matching pairs (specificity = 1.00000). This suggests that, although it fails to detect true matches, it does not falsely classify non-matches as matches.
- **Levenshtein**:
  - **Strong Precision and Accuracy**: With an accuracy of 0.99845 and precision of 0.94703, Levenshtein has improved significantly, likely due to the **zip code blocking** that reduces false matches.
  - **Moderate Sensitivity**: The sensitivity of 0.80272 indicates that it misses some true positives, but it still performs relatively well overall in identifying true matches.
  - **High Specificity**: The specificity of 0.99971 confirms that it is very effective at identifying non-matching pairs, and the balance of precision and specificity makes it a strong method in this case.
- **Phonetic**:
  - **Consistent Performance**: The performance of the Phonetic method remains stable compared to previous results. However, it still struggles with precision (0.05339), meaning many predicted matches are false positives.

- **Good Sensitivity**: High sensitivity (0.95841) indicates that Phonetic is able to identify most true matches, though not as effectively as Levenshtein

Conclusion

1. **Jaro-Winkler**:
   - **Sensitivity and Precision Issues**: The results for **Jaro-Winkler** suggest that the **zip code blocking** may not be ideal for this method or that the blocking variable is causing mismatches. **Jaro-Winkler** might be too strict or not appropriately tuned for the current data. Consider adjusting the matching thresholds or using a different blocking variable or a combination of multiple variables.
   - **Recommendation**: Investigate the data for inconsistencies in zip code formatting or consider combining zip with other variables (e.g., city, name) for better performance.
2. **Levenshtein**:
   - **Best Performing Method**: **Levenshtein** continues to perform very well with **high precision** (0.94703) and **good accuracy** (0.99845). It seems to strike a good balance between identifying true positives and minimizing false positives.
   - **Recommendation**: This method should be the go-to for this dataset, especially since it has high precision and specificity. Consider tweaking the threshold or further tuning it for even better sensitivity.
3. **Phonetic**:
   - **Stable but Needs Improvement**: **Phonetic** is consistent, but its low precision means many false positives are still being produced. It performs well for identifying true matches but struggles with false positives.
   - **Recommendation**: Consider combining the **Phonetic** method with another (like Levenshtein) to reduce false positives and improve overall precision.

Blocking first name

```
# JaroWinkler
b_jaro <- RLBigDataLinkage(
  dataset1 = surveydata,
  dataset2 = admindata,
  blockfld = 'firstname',
  identity1 = identity_survey,
  identity2 = identity_admin,
```

```
  strcmp = TRUE,
  strcmpfun = 'jarowinkler',
  exclude = c(1:2, 5:13, 15:18)
)

b_jaro_w <- emWeights(b_jaro, cutoff = 0.7, verbose = TRUE)
```

```
Count pattern frequencies...

Run EM algorithm...

================================================================================
```

```
b_jaro_class <- emClassify(b_jaro_w, threshold.upper = 22)
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to min; returning
Inf

Warning in max(x, na.rm = na.rm): no non-missing arguments to max; returning
-Inf
```

```
#levenshtein
b_leven <- RLBigDataLinkage(
  dataset1 = surveydata,
  dataset2 = admindata,
  identity1 = identity_survey,
  identity2 = identity_admin,
  blockfld = 'firstname',
  strcmp = TRUE,
  strcmpfun = "levenshtein",
  exclude = c(1:2, 5:13, 15:18)
)
b_leven_w <- emWeights(b_leven, cutoff = 0.95)
```

```
Count pattern frequencies...
Run EM algorithm...

================================================================================
```

```r
b_leven_class <- emClassify(b_leven_w, threshold.upper = 0)
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to min; returning
Inf
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to max; returning
-Inf
```

```r
# Phonetic
b_phone <- RLBigDataLinkage(
  dataset1 = surveydata,
  dataset2 = admindata,
  identity1 = identity_survey,
  identity2 = identity_admin,
  blockfld = 'firstname',
  phonetic = TRUE,
  exclude = c(1:2, 5:13, 15:18)
)
b_phone_w <- emWeights(b_phone, cutoff = 0.95)
```

```
Count pattern frequencies...
Run EM algorithm...
```

```
================================================================================
```

```r
b_phone_class <- emClassify(b_phone_w, threshold.upper = 0)
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to min; returning
Inf
```

```
Warning in min(x, na.rm = na.rm): no non-missing arguments to max; returning
-Inf
```

```r
# Extracting error measures from each method
err_jarowinkler <- getErrorMeasures(b_jaro_class)
err_levenshtein <- getErrorMeasures(b_leven_class)
err_phonetic     <- getErrorMeasures(b_phone_class)
```

```r
# Creating a data frame summarizing key metrics
comparison_table <- data.frame(
  Method = c("Jarowinkler", "Levenshtein", "Phonetic"),
  Accuracy = c(err_jarowinkler$accuracy,
               err_levenshtein$accuracy,
               err_phonetic$accuracy),
  Precision = c(err_jarowinkler$precision,
                err_levenshtein$precision,
                err_phonetic$precision),
  Sensitivity = c(err_jarowinkler$sensitivity,
                  err_levenshtein$sensitivity,
                  err_phonetic$sensitivity),
  Specificity = c(err_jarowinkler$specificity,
                  err_levenshtein$specificity,
                  err_phonetic$specificity)
)

print(comparison_table)
```

```
      Method  Accuracy Precision Sensitivity Specificity
1 Jarowinkler 0.9230843 0.1546821   1.0000000   0.9219863
2 Levenshtein 0.9451914 0.1863753   0.8599453   0.9464083
3    Phonetic 0.9562877 0.1475693   0.9645224   0.9562230
```

- **Jaro-Winkler**:
  - **High Sensitivity but Lower Accuracy**: The perfect **sensitivity** of 1.00000 indicates that Jaro-Winkler still identifies all true matches but with a decrease in **accuracy** (0.92308) compared to previous blocking variables. The drop in accuracy could be due to the firstname blocking not grouping the records effectively, leading to fewer correct non-matches.
  - **Precision Improvements**: The increase in **precision** (0.15468) suggests that Jaro-Winkler is making fewer false positives, but it still doesn't perform as well in terms of precision as some other methods.
- **Levenshtein**:
  - **Best Balance**: **Levenshtein** appears to be the best-performing method with **accuracy** (0.94519), **precision** (0.18638), and **sensitivity** (0.85995). It strikes a good balance between identifying true positives and minimizing false positives.

  - **Improved Sensitivity**: The increase in sensitivity indicates that the **firstname blocking variable** has allowed Levenshtein to better identify matching records without sacrificing much specificity.

- **Phonetic**:

  - **Consistent Performance**: While Phonetic maintains high **accuracy** and **specificity**, it still has a **low precision** (0.05339) and could benefit from combining with other methods (like Levenshtein) for better performance. It has **high sensitivity**, making it useful for identifying a majority of true matches, but the low precision means many of these matches are false positives.

Conclusion

1. **Jaro-Winkler**:

   - While it still performs well in identifying all true positive pairs (**sensitivity = 1.00000**), **accuracy** has dropped due to reduced precision. The **firstname blocking** variable may not be ideal for this method, or it may need further adjustment.
   - **Recommendation**: Investigate the consistency of **firstname** across datasets and consider combining it with other blocking variables (e.g., city or last name) to improve performance. You may also adjust the matching threshold or experiment with different blocking methods.

2. **Levenshtein**:

   - This method is currently the best performer for this dataset, showing a good trade-off between **precision**, **sensitivity**, and **specificity**.
   - **Recommendation**: Continue using **Levenshtein** as the primary method. You could experiment with different blocking combinations or fine-tune its parameters for further improvement in **sensitivity** while maintaining **precision**.

3. **Phonetic**:

   - **Phonetic** remains consistent in terms of accuracy and sensitivity but has low precision, which could lead to many false positives.
   - **Recommendation**: Consider combining **Phonetic** with **Levenshtein** or **Jaro-Winkler** to balance precision and recall, or filter out false positives post-matching.