Package ‘automatedRecLin’

November 18, 2025

Title Record Linkage Based on an Entropy-Maximizing Classifier

Version 1.0.0

Description The goal of 'automatedRecLin' is to perform record linkage (also known as entity resolution) in unsupervised or supervised settings. It compares pairs of records from two datasets using selected comparison functions to estimate the probability or density ratio between matched and non-matched records. Based on these estimates, it predicts a set of matches that maximizes entropy. For details see: Lee et al. (2022) <https://www150.statcan.gc.ca/n1/pub/12-001-x/2022001/article/00007-eng.htm>, Vo et al. (2023) <https://ideas.repec.org/a/eee/csdana/v179y2023ics0167947322002365.html>, Sugiyama et al. (2008) <doi:10.1007/s10463-008-0197-x>.

License GPL-3

Encoding UTF-8

URL https://github.com/ncn-foreigners/automatedRecLin, http://ncn-foreigners.ue.poznan.pl/automatedRecLin/

BugReports https://github.com/ncn-foreigners/automatedRecLin/issues

RoxygenNote 7.3.2

Imports data.table, densityratio, FixedPoint, methods, nleqslv, purrr, reclin2, stats, utils

Suggests tinytest, xgboost

Depends R (>= 4.1.0)

LazyData true

NeedsCompilation no

Author Adam Struzik [aut, cre],
Maciej Beręsewicz [aut, ctb] (ORCID: <https://orcid.org/0000-0002-8281-4301>)

Maintainer Adam Struzik <adastr5@st.amu.edu.pl>

Repository CRAN

Date/Publication 2025-11-18 09:20:02 UTC

Contents

`abs_distance` *Absolute Distance Comparison Function*

Description

Creates a function that calculates the absolute distance between two values.

Usage

```
abs_distance()
```

Value

Returns a function taking two arguments, `x` and `y`, and returning their absolute difference.

Author(s)

Adam Struzik

Examples

```
cmp <- abs_distance()
cmp(1, 5) # returns 4
```

`A_example` *A_example dataset*

Description

An example dataset containing artificial personal data.

Usage

```
A_example
```

Format

A `data.frame` with 10 records. Each row represents one record, with the following columns: `name`, `surname`, and `city`. Some records can be matched with records in the `B_example` dataset.

Examples

```
data("A_example")
A_example
```

`B_example` *B_example dataset*

Description

An example dataset containing artificial personal data.

Usage

```
B_example
```

Format

A `data.frame` with 12 records. Each row represents one record, with the following columns: `name`, `surname`, and `city`. Some records can be matched with records in the `A_example` dataset.

Examples

```
data("B_example")
B_example
```

`comparison_vectors` *Create Comparison Vectors for Record Linkage*

Description

Creates comparison vectors between records in two datasets based on specified variables and comparison functions.

Usage

```
comparison_vectors(A, B, variables, comparators = NULL, matches = NULL)
```

Arguments

`A`	A duplicate-free `data.frame` or `data.table`.
`B`	A duplicate-free `data.frame` or `data.table`.
`variables`	A character vector of key variables used to create comparison vectors.
`comparators`	A named list of functions for comparing pairs of records.
`matches`	Optional. A `data.frame` or `data.table` indicating known matches.

Details

Consider two datasets: A and B. For each pair of records $(a, b) \in \Omega$, the function creates a comparison vector $\boldsymbol{\gamma}_{ab} = (\gamma_{ab}^1, \gamma_{ab}^2, \ldots, \gamma_{ab}^K)'$ based on specified K variables and comparison functions.

Value

Returns a list containing:

- `Omega` – a `data.table` with comparison vectors between all records from both datasets, including optional match information,
- `variables` – a character vector of key variables used for comparison,
- `comparators` – a list of functions used to compare pairs of records,
- `match_prop` – proportion of matches in the smaller dataset.

Note

Each comparison function must return another function, which serves as the actual comparator.

Author(s)

Adam Struzik

Examples

```
df_1 <- data.frame(
"name" = c("John", "Emily", "Mark", "Anna", "David"),
"surname" = c("Smith", "Johnson", "Taylor", "Williams", "Brown")
)
df_2 <- data.frame(
  "name" = c("Jon", "Emely", "Marc", "Michael"),
  "surname" = c("Smitth", "Jonson", "Tailor", "Henderson")
)
comparators <- list("name" = jarowinkler_complement(),
                    "surname" = jarowinkler_complement())
matches <- data.frame("a" = 1:3, "b" = 1:3)
result <- comparison_vectors(A = df_1, B = df_2, variables = c("name", "surname"),
                             comparators = comparators, matches = matches)
result
```

`control_kliep` *Controls for the* `kliep` *Function*

Description

Controls for the kliep function used in the package.

Usage

```
control_kliep(scale = NULL, progressbar = FALSE, nfold = 2, ...)
```

Arguments

`scale`	`"numerator"`, `"denominator"` or `NULL`, indicating whether to standardize each numeric variable according to the numerator means and standard deviations, the denominator means and standard deviations, or apply no standardization at all.
`progressbar`	Logical indicating whether or not to display a progressbar.
`nfold`	Number of cross-validation folds used in order to calculate the optimal sigma value (default is 2-fold cv).
`...`	Additional arguments.

Value

Returns a list with parameters.

Author(s)

Adam Struzik

`custom_rec_lin_model` *Create a Custom Record Linkage Model*

Description

Creates a supervised record linkage model using a custom machine learning (ML) classifier.

Usage

```
custom_rec_lin_model(ml_model, vectors)
```

Arguments

`ml_model`	A trained ML model that predicts the probability of a match based on comparison vectors.
`vectors`	An object of class `comparison_vectors` (a result of the `comparison_vectors` function), used for training the `ml_model`.

Details

The `custom_rec_lin_model` function creates a custom record linkage model, based on known matches and non-matches (which might later serve as a classifier for pairs outside training data). The procedure of creating a custom model based on training data is as follows.

1. Use the `comparison_vectors` function to compare pairs of records.
2. Train a machine learning classifier using the `Omega` element of the output of the `comparison_vectors` function. The classifier should predict the probability of matching based on a given vector.
3. Use the `custom_rec_lin_model` function with appropriate arguments.

Value

Returns a list containing:

- `b_vars` – here `NULL`,
- `cpar_vars` – here `NULL`,
- `cnonpar_vars` – here `NULL`,
- `b_params` – here `NULL`,
- `cpar_params` – here `NULL`,
- `cnonpar_params` – here `NULL`,
- `ratio_kliep` – here `NULL`,
- `ratio_kliep_list` – here `NULL`,
- `ml_model` – ML model used for creating the record linkage model,
- `pi_est` – a prior probability of matching,

- `match_prop` – proportion of matches in the smaller dataset,
- `variables` – a character vector of key variables used for comparison,
- `comparators` – a list of functions used to compare pairs of records,
- `methods` – here `NULL`,
- `prob_ratio` – here `"2"`.

Author(s)

Adam Struzik

Examples

```
if (requireNamespace("xgboost", quietly = TRUE)) {
  df_1 <- data.frame(
    "name" = c("James", "Emma", "William", "Olivia", "Thomas",
    "Sophie", "Harry", "Amelia", "George", "Isabella"),
    "surname" = c("Smith", "Johnson", "Brown", "Taylor", "Wilson",
    "Davis", "Clark", "Harris", "Lewis", "Walker")
  )
  df_2 <- data.frame(
    "name" = c("James", "Ema", "Wimliam", "Olivia", "Charlotte",
    "Henry", "Lucy", "Edward", "Alice", "Jack"),
    "surname" = c("Smith", "Johnson", "Bron", "Tailor", "Moore",
    "Evans", "Hall", "Wright", "Green", "King")
  )
  comparators <- list("name" = jarowinkler_complement(),
                      "surname" = jarowinkler_complement())
  matches <- data.frame("a" = 1:4, "b" = 1:4)
  vectors <- comparison_vectors(A = df_1, B = df_2, variables = c("name", "surname"),
                               comparators = comparators, matches = matches)
  train_data <- xgboost::xgb.DMatrix(
    data = as.matrix(vectors$Omega[, c("gamma_name", "gamma_surname")]),
    label = vectors$Omega$match
  )
  params <- list(objective = "binary:logistic",
                 eval_metric = "logloss")
  model_xgb <- xgboost::xgboost(data = train_data, params = params,
                                nrounds = 100, verbose = 0)
  custom_xgb_model <- custom_rec_lin_model(model_xgb, vectors)
  custom_xgb_model
}
```

`jarowinkler_complement` *Jaro-Winkler Distance Complement*

Description

Creates a function that calculates the complement of the Jaro-Winkler distance between two strings (i.e., $1 -$ Jaro-Winkler distance).

Usage

```
jarowinkler_complement()
```

Value

Returns a function taking two string arguments, `x` and `y`, and returning the complement of the Jaro-Winkler distance.

Author(s)

Adam Struzik

`mec` *Unsupervised Maximum Entropy Classifier for Record Linkage*

Description

Implements several extensions to the maximum entropy classification (MEC) algorithm for record linkage (see Lee et al. (2022)), iteratively estimating probability/density ratios to classify record pairs into matches and non-matches based on comparison vectors.

Usage

```
mec(
  A,
  B,
  variables,
  comparators = NULL,
  methods = NULL,
  duplicates_in_A = FALSE,
  start_params = NULL,
  nonpar_hurdle = TRUE,
  set_construction = NULL,
  target_rate = 0.03,
  max_iter_bisection = 100,
  tol = 0.005,
  delta = 0.5,
  eps = 0.05,
  max_iter_em = 10,
  tol_em = 1,
  controls_nleqslv = list(),
  controls_kliep = control_kliep(),
  true_matches = NULL
)
```

Arguments

`A`	A duplicate-free `data.frame` or `data.table`.
`B`	A duplicate-free `data.frame` or `data.table`.
`variables`	A character vector of key variables used to create comparison vectors.
`comparators`	A named list of functions for comparing pairs of records.
`methods`	A named list of methods used for estimation (`"binary"`, `"continuous_parametric"`, `"continuous_nonparametric"` or `"hit_miss"`).
`duplicates_in_A`	Logical indicating whether to allow `A` to have duplicate records.
`start_params`	Start parameters for the `"binary"`, `"continuous_parametric"` and `"hit_miss"` methods.
`nonpar_hurdle`	Logical indicating whether to use a hurdle model or not (used only if the `"continuous_nonparametric"` method has been chosen for at least one variable).
`set_construction`	A method for constructing the predicted set of matches (`"size"`, `"flr"` or `"mmr"`).
`target_rate`	A target false link rate (FLR) or missing match rate (MMR) (used only if `set_construction == "flr"` or `set_construction == "mmr"`).
`max_iter_bisection`	A maximum number of iterations for the bisection procedure (used only if `set_construction == "flr"` or `set_construction == "mmr"`).
`tol`	Error tolerance in the bisection procedure (used only if `set_construction == "flr"` or `set_construction == "mmr"`).
`delta`	A numeric value specifying the tolerance for the change in the estimated number of matches between iterations.
`eps`	A numeric value specifying the tolerance for the change in model parameters between iterations.
`max_iter_em`	A maximum number of iterations for the EM algorithm (used only if the `"hit_miss"` method has been chosen for at least one variable).
`tol_em`	Error tolerance in the EM algorithm (used only if the `"hit_miss"` method has been chosen for at least one variable).
`controls_nleqslv`	Controls passed to the nleqslv function (only if the `"continuous_parametric"` method has been chosen for at least one variable).
`controls_kliep`	Controls passed to the kliep function (only if the `"continuous_nonparametric"` method has been chosen for at least one variable).
`true_matches`	A `data.frame` or `data.table` indicating known matches.

Details

Consider two datasets without duplicates: A and B. Let the bipartite comparison space $\Omega = A \times B$ consist of matches M and non-matches U between the records in files A and B. For any pair of records $(a, b) \in \Omega$, let $\boldsymbol{\gamma}_{ab} = (\gamma_{ab}^1, \gamma_{ab}^2, \ldots, \gamma_{ab}^K)'$ be the comparison vector between a set of key variables. The original MEC algorithm uses the binary comparison function to evaluate record pairs across two datasets. However, this approach may be insufficient when handling datasets with frequent errors across multiple variables.

We propose the use of continuous comparison functions to address the limitations of binary comparison methods. We consider every semi-metric, i.e., a function $d : A \times B \to \mathbb{R}$, satisfying the following conditions:

1. $d(x, y) \geq 0$,
2. $d(x, y) = 0$ if and only if $x = y$,
3. $d(x, y) = d(y, x)$.

For example, we can use $1 -$ Jaro-Winkler distance for character variables (which is implemented in the `automatedRecLin` package as the `jarowinkler_complement` function) or the Euclidean distance for numerical variables. The `automatedRecLin` package allows the use of a different comparison function for each key variable (which should be specified as a list in the `comparators` argument). The default function for each key variable is cmp_identical (the binary comparison function).

The `mec` function offers different approaches to estimate the probability/density ratio between matches and non-matches, which should be specified as a list in the `methods` argument. The available methods suitable for the binary comparison function are `"binary"` and `"hit_miss"`. Both assume that $\gamma_{ab}^k|M$ and $\gamma_{ab}^k|U$ follow Bernoulli distributions. `"binary"` and `"hit_miss"` both estimate the parameters for the matches iteratively, but `"binary"` estimates the parameters for the non-matches only at the start, while `"hit_miss"` does so iteratively using a hit-miss model (for details see Lee et al. (2022)). `"binary"` is the default method for each variable.

For the continuous semi-metrics we suggest the usage of `"continuous_parametric"` or `"continuous_nonparametric"` method. The `"continuous_parametric"` method assumes that $\gamma_{ab}^k|M$ and $\gamma_{ab}^k|U$ follow hurdle Gamma distributions. The density function of a hurdle Gamma distribution is characterized by three parameters $p_0 \in (0, 1)$ and $\alpha, \beta > 0$ as follows:

$$f(x; p_0, \alpha, \beta) = p_0^{\mathbb{I}(x=0)}[(1 - p_0)v(x; \alpha, \beta)]^{\mathbb{I}(x>0)},$$

where

$$v(x; \alpha, \beta) = \frac{\beta^\alpha x^{\alpha-1} \exp(-\beta x)}{\Gamma(\alpha)}$$

is the density function of a Gamma distribution (for details see Vo et al. (2023)). At the beginning, the algorithm estimates the parameters for the non-matches and then does it iteratively for the matches. The `"continuous_nonparametric"` method does not assume anything about the distributions of the comparison vectors. It iteratively directly estimates the density ratio between the matches and the non-matches, using the Kullback-Leibler Importance Estimation Procedure (KLIEP). For details see Sugiyama et al. (2008).

The `mec` function allows the construction of the predicted set of matches using its estimated size or the bisection procedure, described in Lee et al. (2022), based on a target False Link Rate (FLR) or missing match rate (MMR). To use the second option, set `set_construction = "flr"` or `set_construction = "mmr"` and specify a target error rate using the `target_rate` argument.

The assumption that A and B contain no duplicate records might be relaxed by allowing A to have duplicates. To do so, set `duplicates_in_A = TRUE`.

Value

Returns a list containing:

- `M_est` – a `data.table` with predicted matches,
- `n_M_est` – estimated classification set size,
- `flr_est` – estimated false link rate (FLR),
- `mmr_est` – estimated missing match rate (MMR),
- `iter_bisection` – the number of iterations in the bisection procedure,
- `b_vars` – a character vector of variables used for the `"binary"` method (with the prefix `"gamma_"`),
- `cpar_vars` – a character vector of variables used for the `"continuous_parametric"` method (with the prefix `"gamma_"`),
- `cnonpar_vars` – a character vector of variables used for the `"continuous_nonparametric"` method (with the prefix `"gamma_"`),
- `hm_vars` – a character vector of variables used for the `"hit_miss"` method (with the prefix `"gamma_"`),
- `b_params` – parameters estimated using the `"binary"` method,
- `cpar_params` – parameters estimated using the `"continuous_parametric"` method,
- `hm_params` – parameters estimated using the `"hit_miss"` method,
- `ratio_kliep` – a result of the kliep function,
- `variables` – a character vector of key variables used for comparison,
- `set_construction` – a method for constructing the predicted set of matches,
- `eval_metrics` – standard metrics for quality assessment (if `true_matches` is provided),
- `confusion` – confusion matrix (if `true_matches` is provided).

Author(s)

Adam Struzik

References

Lee, D., Zhang, L.-C. and Kim, J. K. (2022). Maximum entropy classification for record linkage. Survey Methodology, Statistics Canada, Catalogue No. 12-001-X, Vol. 48, No. 1.

Vo, T. H., Chauvet, G., Happe, A., Oger, E., Paquelet, S., and Garès, V. (2023). Extending the Fellegi-Sunter record linkage model for mixed-type data with application to the French national health data system. Computational Statistics & Data Analysis, 179, 107656.

Sugiyama, M., Suzuki, T., Nakajima, S. et al. Direct importance estimation for covariate shift adaptation. Ann Inst Stat Math 60, 699–746 (2008). doi:10.1007/s104630080197x

Examples

```
df_1 <- data.frame(
  name = c("Emma", "Liam", "Olivia", "Noah", "Ava",
           "Ethan", "Sophia", "Mason", "Isabella", "James"),
  surname = c("Smith", "Johnson", "Williams", "Brown", "Jones",
              "Garcia", "Miller", "Davis", "Rodriguez", "Wilson"),
  city = c("New York", "Los Angeles", "Chicago", "Houston", "Phoenix",
           "Philadelphia", "San Antonio", "San Diego", "Dallas", "San Jose")
)

df_2 <- data.frame(
  name = c(
    "Emma", "Liam", "Olivia", "Noah",
    "Ava", "Ehtan", "Sopia", "Mson",
    "Charlotte", "Benjamin", "Amelia", "Lucas"
  ),
  surname = c(
     "Smith", "Johnson", "Williams", "Brown",
    "Jnes", "Garca", "Miler", "Dvis",
    "Martinez", "Lee", "Hernandez", "Clark"
  ),
  city = c(
    "New York", "Los Angeles", "Chicago", "Houston",
    "Phonix", "Philadelpia", "San Antnio", "San Dieg",
    "Seattle", "Miami", "Boston", "Denver"
  )
)
true_matches <- data.frame(
  "a" = 1:8,
  "b" = 1:8
)

variables <- c("name", "surname", "city")
comparators <- list(
  "name" = jarowinkler_complement(),
  "surname" = jarowinkler_complement(),
  "city" = jarowinkler_complement()
)
methods <- list(
  "name" = "continuous_parametric",
  "surname" = "continuous_parametric",
  "city" = "continuous_parametric"
)

set.seed(1)
result <- mec(A = df_1, B = df_2,
              variables = variables,
              comparators = comparators,
              methods = methods,
              true_matches = true_matches)
result
```

`predict.rec_lin_model` *Predict Matches Based on a Given Record Linkage Model*

Description

Predicts matches between records in two datasets based on a given record linkage model, using the maximum entropy classification (MEC) algorithm (see Lee et al. (2022)).

Usage

```
## S3 method for class 'rec_lin_model'
predict(
  object,
  newdata_A,
  newdata_B,
  duplicates_in_A = FALSE,
  set_construction = c("size", "flr", "mmr"),
  fixed_method = "Newton",
  target_rate = 0.03,
  tol = 0.005,
  max_iter = 50,
  data_type = c("data.frame", "data.table", "matrix"),
  true_matches = NULL,
  ...
)
```

Arguments

`object`	A `rec_lin_model` object from the `train_rec_lin` or `custom_rec_lin_model` functions.
`newdata_A`	A duplicate-free `data.frame` or `data.table`.
`newdata_B`	A duplicate-free `data.frame` or `data.table`.
`duplicates_in_A`	Logical indicating whether to allow `A` to have duplicate records.
`set_construction`	A method for constructing the predicted set of matches (`"size"`, `"flr"` or `"mmr"`).
`fixed_method`	A method for solving fixed-point equations using the FixedPoint function.
`target_rate`	A target false link rate (FLR) or missing match rate (MMR) (used only if `set_construction == "flr"` or `set_construction == "mmr"`).
`tol`	Error tolerance in the bisection procedure (used only if `set_construction == "flr"` or `set_construction == "mmr"`).
`max_iter`	A maximum number of iterations for the bisection procedure (used only if `set_construction == "flr"` or `set_construction == "mmr"`).

`data_type`	Data type for predictions with a custom ML model (`"data.frame"`, `"data.table"` or `"matrix"`; used only if `object` is from the `custom_rec_lin_model` function).
`true_matches`	A `data.frame` or `data.table` indicating true matches.
`...`	Additional controls passed to the `predict` function for custom ML model (used only if the `object` is from the `custom_rec_lin_model` function).

Details

The `predict` function estimates the probability/density ratio between matches and non-matches for pairs in given datasets, based on a model obtained using the `train_rec_lin` or `custom_rec_lin_model` functions. Then, it estimates the number of matches and returns the predicted matches, using the maximum entropy classification (MEC) algorithm (see Lee et al. (2022)).

The `predict` function allows the construction of the predicted set of matches using its estimated size or the bisection procedure, described in Lee et al. (2022), based on a target False Link Rate (FLR) or missing match rate (MMR). To use the second option, set `set_construction = "flr"` or `set_construction = "mmr"` and specify a target error rate using the `target_rate` argument.

By default, the function assumes that the datasets `newdata_A` and `newdata_B` contain no duplicate records. This assumption might be relaxed by allowing `newdata_A` to have duplicates. To do so, set `duplicates_in_A = TRUE`.

Value

Returns a list containing:

- `M_est` – a `data.table` with predicted matches,
- `set_construction` – a method for constructing the predicted set of matches,
- `n_M_est` – estimated classification set size,
- `flr_est` – estimated false link rate (FLR),
- `mmr_est` – estimated missing match rate (MMR),
- `iter` – the number of iterations in the bisection procedure,
- `eval_metrics` – standard metrics for quality assessment, if `true_matches` is provided,
- `confusion` – confusion matrix, if `true_matches` is provided.

Author(s)

Adam Struzik

References

Lee, D., Zhang, L.-C. and Kim, J. K. (2022). Maximum entropy classification for record linkage. Survey Methodology, Statistics Canada, Catalogue No. 12-001-X, Vol. 48, No. 1.

Vo, T. H., Chauvet, G., Happe, A., Oger, E., Paquelet, S., and Garès, V. (2023). Extending the Fellegi-Sunter record linkage model for mixed-type data with application to the French national health data system. Computational Statistics & Data Analysis, 179, 107656.

Sugiyama, M., Suzuki, T., Nakajima, S. et al. Direct importance estimation for covariate shift adaptation. Ann Inst Stat Math 60, 699–746 (2008). doi:10.1007/s104630080197x

Examples

```
df_1 <- data.frame(
  "name" = c("James", "Emma", "William", "Olivia", "Thomas",
  "Sophie", "Harry", "Amelia", "George", "Isabella"),
  "surname" = c("Smith", "Johnson", "Brown", "Taylor", "Wilson",
  "Davis", "Clark", "Harris", "Lewis", "Walker")
)
 df_2 <- data.frame(
  "name" = c("James", "Ema", "Wimliam", "Olivia", "Charlotte",
  "Henry", "Lucy", "Edward", "Alice", "Jack"),
  "surname" = c("Smith", "Johnson", "Bron", "Tailor", "Moore",
  "Evans", "Hall", "Wright", "Green", "King")
)
comparators <- list("name" = jarowinkler_complement(),
                    "surname" = jarowinkler_complement())
matches <- data.frame("a" = 1:4, "b" = 1:4)
methods <- list("name" = "continuous_nonparametric",
                "surname" = "continuous_nonparametric")
model <- train_rec_lin(A = df_1, B = df_2, matches = matches,
                       variables = c("name", "surname"),
                       comparators = comparators,
                       methods = methods)

df_new_1 <- data.frame(
  "name" = c("John", "Emily", "Mark", "Anna", "David"),
  "surname" = c("Smith", "Johnson", "Taylor", "Williams", "Brown")
)
df_new_2 <- data.frame(
  "name" = c("John", "Emely", "Mark", "Michael"),
  "surname" = c("Smitth", "Johnson", "Tailor", "Henders")
)
predict(model, df_new_1, df_new_2)
```

`train_rec_lin` *Train a Record Linkage Model*

Description

Trains a supervised record linkage model using probability or density ratio estimation, based on Lee et al. (2022), with several extensions.

Usage

```
train_rec_lin(
  A,
  B,
  matches,
  variables,
  comparators = NULL,
```

```
  methods = NULL,
  prob_ratio = NULL,
  nonpar_hurdle = TRUE,
  controls_nleqslv = list(),
  controls_kliep = control_kliep()
)
```

Arguments

`A`	A duplicate-free `data.frame` or `data.table`.
`B`	A duplicate-free `data.frame` or `data.table`.
`matches`	A `data.frame` or `data.table` indicating known matches.
`variables`	A character vector of key variables used to create comparison vectors.
`comparators`	A named list of functions for comparing pairs of records.
`methods`	A named list of methods used for estimation (`"binary"`, `"continuous_parametric"` or `"continuous_nonparametric"`).
`prob_ratio`	Probability/density ratio type (`"1"` or `"2"`).
`nonpar_hurdle`	Logical indicating whether to use a hurdle model or not (used only if the `"continuous_nonparametric"` method has been chosen for at least one variable).
`controls_nleqslv`	Controls passed to the nleqslv function (only if the `"continuous_parametric"` method has been chosen for at least one variable).
`controls_kliep`	Controls passed to the kliep function (only if the `"continuous_nonparametric"` method has been chosen for at least one variable).

Details

Consider two datasets: A and B. Let the bipartite comparison space $\Omega = A \times B$ consist of matches M and non-matches U between the records in files A and B. For any pair of records $(a, b) \in \Omega$, let $\boldsymbol{\gamma}_{ab} = (\gamma_{ab}^1, \gamma_{ab}^2, \ldots, \gamma_{ab}^K)'$ be the comparison vector between a set of key variables. The original MEC algorithm uses the binary comparison function to evaluate record pairs across two datasets. However, this approach may be insufficient when handling datasets with frequent errors across multiple variables.

We propose the use of continuous comparison functions to address the limitations of binary comparison methods. We consider every semi-metric, i.e., a function $d : A \times B \to \mathbb{R}$, satisfying the following conditions:

1. $d(x, y) \geq 0$,
2. $d(x, y) = 0$ if and only if $x = y$,
3. $d(x, y) = d(y, x)$.

For example, we can use $1 -$ Jaro-Winkler distance for character variables (which is implemented in the `automatedRecLin` package as the `jarowinkler_complement` function) or the Euclidean distance for numerical variables. The `automatedRecLin` package allows the use of a different comparison function for each key variable (which should be specified as a list in the `comparators` argument). The default function for each key variable is cmp_identical (the binary comparison function).

The `train_rec_lin` function is used to train a record linkage model, when M and U are known (which might later serve as a classifier for pairs outside Ω). It offers different approaches to estimate the probability/density ratio between matches and non-matches, which should be specified as a list in the methods argument. The method suitable for the binary comparison function is `"binary"`, which is also the default method for each variable.

For the continuous semi-metrics we suggest the usage of `"continuous_parametric"` or `"continuous_nonparametric"` method. The `"continuous_parametric"` method assumes that $\gamma_{ab}^k|M$ and $\gamma_{ab}^k|U$ follow hurdle Gamma distributions. The density function of a hurdle Gamma distribution is characterized by three parameters $p_0 \in (0,1)$ and $\alpha, \beta > 0$ as follows:

$$f(x; p_0, \alpha, \beta) = p_0^{\mathbb{I}(x=0)}[(1-p_0)v(x;\alpha,\beta)]^{\mathbb{I}(x>0)},$$

where

$$v(x;\alpha,\beta) = \frac{\beta^\alpha x^{\alpha-1}\exp(-\beta x)}{\Gamma(\alpha)}$$

is the density function of a Gamma distribution (for details see Vo et al. (2023)). The `"continuous_nonparametric"` method does not assume anything about the distributions of the comparison vectors. It directly estimates the density ratio between the matches and the non-matches, using the Kullback-Leibler Importance Estimation Procedure (KLIEP). For details see Sugiyama et al. (2008).

Value

Returns a list containing:

- `b_vars` – a character vector of variables used for the `"binary"` method (with the prefix `"gamma_"`),
- `cpar_vars` – a character vector of variables used for the `"continuous_parametric"` method (with the prefix `"gamma_"`),
- `cnonpar_vars` – a character vector of variables used for the `"continuous_nonparametric"` method (with the prefix `"gamma_"`),
- `b_params` – parameters estimated using the `"binary"` method,
- `cpar_params` – parameters estimated using the `"continuous_parametric"` method,
- `cnonpar_params` – probability of exact matching estimated using the `"continuous_nonparametric"` method,
- `ratio_kliep` – a result of the kliep function,
- `ratio_kliep_list` – an object containing the results of the kliep function,
- `ml_model` – here `NULL`,
- `pi_est` – a prior probability of matching,
- `match_prop` – proportion of matches in the smaller dataset,
- `variables` – a character vector of key variables used for comparison,
- `comparators` – a list of functions used to compare pairs of records,
- `methods` – a list of methods used for estimation,
- `"prob_ratio"` – probability/density ratio type.

Author(s)

Adam Struzik

References

Lee, D., Zhang, L.-C. and Kim, J. K. (2022). Maximum entropy classification for record linkage. Survey Methodology, Statistics Canada, Catalogue No. 12-001-X, Vol. 48, No. 1.

Vo, T. H., Chauvet, G., Happe, A., Oger, E., Paquelet, S., and Garès, V. (2023). Extending the Fellegi-Sunter record linkage model for mixed-type data with application to the French national health data system. Computational Statistics & Data Analysis, 179, 107656.

Sugiyama, M., Suzuki, T., Nakajima, S. et al. Direct importance estimation for covariate shift adaptation. Ann Inst Stat Math 60, 699–746 (2008). doi:10.1007/s104630080197x

Examples

```
df_1 <- data.frame(
  "name" = c("James", "Emma", "William", "Olivia", "Thomas",
  "Sophie", "Harry", "Amelia", "George", "Isabella"),
  "surname" = c("Smith", "Johnson", "Brown", "Taylor", "Wilson",
  "Davis", "Clark", "Harris", "Lewis", "Walker")
)
 df_2 <- data.frame(
  "name" = c("James", "Ema", "Wimliam", "Olivia", "Charlotte",
  "Henry", "Lucy", "Edward", "Alice", "Jack"),
  "surname" = c("Smith", "Johnson", "Bron", "Tailor", "Moore",
  "Evans", "Hall", "Wright", "Green", "King")
)
comparators <- list("name" = jarowinkler_complement(),
                    "surname" = jarowinkler_complement())
matches <- data.frame("a" = 1:4, "b" = 1:4)
methods <- list("name" = "continuous_nonparametric",
                "surname" = "continuous_nonparametric")
model <- train_rec_lin(A = df_1, B = df_2, matches = matches,
                       variables = c("name", "surname"),
                       comparators = comparators,
                       methods = methods)
model
```

Index