Package 'automatedRecLin'

May 8, 2026

Title Record Linkage Based on an Entropy-Maximizing Classifier

Version 1.1.0

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

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

Suggests knitr, rmarkdown, tinytest, xgboost

Depends R (>= 4.1.0)

LazyData true

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Adam Struzik [aut, cre] (ORCID: <https://orcid.org/0009-0002-2547-482X>), Maciej Beręsewicz [aut, ctb] (ORCID: <https://orcid.org/0000-0002-8281-4301>)

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

Repository CRAN

Date/Publication 2026-05-08 07:40:21 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
```

census *Fictional census data*

Description

This dataset was created by Paula McLeod, Dick Heasman and Ian Forbes, ONS, for the ESSnet DI on-the-job training course, Southampton, 25-28 January 2011. It contains fictional data representing some observations from a decennial Census.

Usage

```
census
```

Format

A `data.table` with 25343 records. Each row represents one record, with the following columns:

- `person_id` – a unique number for each person, consisting of postcode, house number and person number,
- `pername1` – forename,
- `pername2` – surname,
- `sex` – gender (M/F),
- `dob_day` – day of birth,
- `dob_mon` – month of birth,
- `dob_year` – year of birth,
- `hse_num` – house number, a numeric label for each house within a street,
- `enumcap` – an address consisting of house number and street name,
- `enumpc` – postcode,
- `str_nam` – street name of person's household's street,
- `cap_add` – full address, consisting of house number, street name and postcode,
- `census_id` – person ID with "CENS" added in front.

References

McLeod, P., Heasman, D., Forbes, I. (2011). Simulated data for the ESSnet DI on-the-job training course, Southampton, 25-28 January 2011. https://wayback.archive-it.org/12090/20231221144450/https://cros-legacy.ec.europa.eu/content/job-training_en

Examples

```
data("census")
head(census)
```

cis *Fictional customer data*

Description

This dataset was created by Paula McLeod, Dick Heasman and Ian Forbes, ONS, for the ESSnet DI on-the-job training course, Southampton, 25-28 January 2011. It contains fictional observations from Customer Information System, which is combined administrative data from the tax and benefit systems.

Usage

```
cis
```

Format

A `data.table` with 24613 records. Each row represents one record, with the following columns:

- `person_id` – a unique number for each person, consisting of postcode, house number and person number,
- `pername1` – forename,
- `pername2` – surname,
- `sex` – gender (M/F),
- `dob_day` – day of birth,
- `dob_mon` – month of birth,
- `dob_year` – year of birth,
- `enumcap` – an address consisting of house number and street name,
- `enumpc` – postcode,
- `cis_id` – person ID with "CIS" added in front.

References

McLeod, P., Heasman, D., Forbes, I. (2011). Simulated data for the ESSnet DI on-the-job training course, Southampton, 25-28 January 2011. https://wayback.archive-it.org/12090/20231221144450/https://cros-legacy.ec.europa.eu/content/job-training_en

Examples

```
data("cis")
head(cis)
```

`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,
  pairs = 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.
`pairs`	Optional. A `data.frame` or `data.table` with columns `a` and `b` indicating pairs for which comparison vectors should be created.
`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 $\boldsymbol{\gamma}_{ab} = (\gamma^1_{ab}, \gamma^2_{ab}, \ldots, \gamma^K_{ab})'$ based on specified K variables and comparison functions.

Value

Returns a list containing:

- `Omega` – a `data.table` with comparison vectors between 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 progress bar.
nfold	Number of cross-validation folds used in order to calculate the optimal sigma value (default is 2-fold cross-validation).
...	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 `comparison_vectors()`), 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 `comparison_vectors()` to compare pairs of records.
2. Train a machine learning classifier using the `Omega` element of the output of `comparison_vectors()`. The classifier should predict the probability of matching based on a given vector.
3. Use `custom_rec_lin_model()` 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)
 model_xgb <- xgboost::xgboost(x = as.matrix(vectors$Omega[, c("gamma_name", "gamma_surname")]),
                       y = factor(vectors$Omega$match),
                       objective = "binary:logistic", eval_metric = "logloss",
                       nrounds = 100, verbosity = 0, nthread = 1)
  custom_xgb_model <- custom_rec_lin_model(model_xgb, vectors)
  custom_xgb_model
}
```

`est_se_bootstrap` *Parametric Bootstrap for Standard Error Estimation in MEC*

Description

WORK IN PROGRESS

This function is currently under development.

Usage

```
est_se_bootstrap(mec_result, B = 100, alpha = 0.05)
```

Arguments

mec_result	An object of class mec_rec_lin.
B	A number of bootstrap iterations.
alpha	A significance level for calculating the confidence interval. Default is 0.05 (which yields a 95% confidence interval).

Author(s)

Adam Struzik

---

jarowinkler_complement

Jaro-Winkler Distance

Description

Creates a function that calculates the Jaro-Winkler distance between two strings, defined as $1 -$ Jaro-Winkler similarity.

Usage

```
jarowinkler_complement()
```

Value

Returns a function taking two string arguments, x and y, and returning 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,
  verbose = FALSE
)
```

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.

`verbose` Logical indicating whether to print progress messages.

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 the Jaro-Winkler distance for character variables (which is implemented in the automatedRecLin package as jarowinkler_complement()) 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
```

`mec_blocking` *Blocked Unsupervised Maximum Entropy Classifier for Record Linkage*

Description

Runs graph-based blocking using blocking(), fits one pooled unsupervised maximum entropy classifier (MEC) on selected within-block pairs, and applies the fitted density-ratio model blockwise.

Usage

```
mec_blocking(
  A,
  B,
  variables,
```

```
  comparators = NULL,
  methods = NULL,
  blocking_x = NULL,
  blocking_y = NULL,
  blocking_variables = variables,
  blocking_sep = " ",
  controls_blocking = list(),
  min_training_pairs = NULL,
  min_training_nonmatches = NULL,
  block_sampling_seed = NULL,
  nonmatch_sample_size = NULL,
  nonmatch_sampling_seed = NULL,
  prob_ratio = "2",
  start_params = NULL,
  nonpar_hurdle = TRUE,
  fixed_method = "Newton",
  delta = 0.5,
  eps = 0.05,
  max_iter_em = 10,
  tol_em = 1,
  controls_nleqslv = list(),
  controls_kliep = control_kliep(),
  true_matches = NULL,
  keep_blocking_result = FALSE,
  keep_training_data = FALSE,
  verbose = FALSE
)
```

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 MEC comparison vectors.
comparators	A named list of functions for comparing pairs of records.
methods	A named list of methods used for estimation ("binary" or "continuous_parametric"). Other unsupervised MEC methods are not supported by mec_blocking() at this stage.
blocking_x	Optional input passed as x to blocking().
blocking_y	Optional input passed as y to blocking().
blocking_variables	Variables used to create blocking strings when blocking_x and blocking_y are not supplied.
blocking_sep	Separator used when concatenating blocking_variables.
controls_blocking	A list of additional arguments passed to blocking(), except x and y.

`min_training_pairs`
: Minimum number of within-block training pairs. If `NULL`, `min_training_nonmatches` should also be `NULL`, and all blocks are used for training.

`min_training_nonmatches`
: Minimum lower bound on within-block nonmatches. If `NULL`, `min_training_pairs` should also be `NULL`, and all blocks are used for training.

`block_sampling_seed`
: Optional seed for random training-block sampling.

`nonmatch_sample_size`
: Number of pairs sampled from the full Cartesian product of `A` and `B` to estimate nonmatch distribution parameters. If `NULL`, all pairs are used.

`nonmatch_sampling_seed`
: Optional seed for nonmatch pair sampling.

`prob_ratio`
: Probability/density ratio type (`"1"` or `"2"`). The default `"2"` uses the blockwise fixed-point equation.

`start_params`
: Start parameters for the `"binary"` and `"continuous_parametric"` methods.

`nonpar_hurdle`
: Currently unused in `mec_blocking()`.

`fixed_method`
: A method for solving blockwise fixed-point equations using the FixedPoint() function.

`delta`
: A numeric value specifying the tolerance for the change in the estimated number of matches between MEC iterations.

`eps`
: A numeric value specifying the tolerance for the change in model parameters between MEC iterations.

`max_iter_em`
: Currently unused in `mec_blocking()`.

`tol_em`
: Currently unused in `mec_blocking()`.

`controls_nleqslv`
: Controls passed to the nleqslv() function (only if the `"continuous_parametric"` method has been chosen for at least one variable).

`controls_kliep`
: Currently unused in `mec_blocking()`.

`true_matches`
: A `data.frame` or `data.table` indicating known matches.

`keep_blocking_result`
: Logical indicating whether to store the raw object returned by blocking().

`keep_training_data`
: Logical indicating whether to store pooled training comparison vectors.

`verbose`
: Logical indicating whether to print progress messages.

Details

The function assumes one-to-one linkage. The blocking stage defines disjoint bipartite blocks. MEC is trained once on the pooled union of selected within-block Cartesian products and is then applied separately in each final block. The ANN distance returned by blocking() is not used.

If both `min_training_pairs` and `min_training_nonmatches` are `NULL`, all final blocks are used for pooled training. If both are supplied, blocks are sampled without replacement until both thresholds are reached. Supplying only one threshold is an error.

Nonmatch distribution parameters are estimated from a simple random sample from the full Cartesian product of `A` and `B`, not from the selected training blocks.

Value

Returns a list of class "mec_blocking" containing:

- M_est – a data.table with predicted matches and columns a, b, block, and ratio,
- n_M_est – estimated total number of matches across all blocks,
- flr_est – estimated false link rate (FLR),
- mmr_est – estimated missing match rate (MMR),
- training_rule – training-block selection rule used by the function,
- block_estimates – a data.table with block-level match-count and error-rate estimates,
- training_blocks – a data.table with blocks selected for pooled MEC training,
- block_summary – a data.table describing the final disjoint blocks,
- excluded_records – a list with records from A and B excluded by blocking,
- pooled_model – fitted pooled MEC density-ratio model used for blockwise scoring,
- b_vars – variables used for the "binary" method, with the prefix "gamma_",
- cpar_vars – variables used for the "continuous_parametric" method, with the prefix "gamma_",
- cnonpar_vars – variables used for the "continuous_nonparametric" method, currently NULL,
- hm_vars – variables used for the "hit_miss" method, currently NULL,
- b_params – parameters estimated using the "binary" method,
- cpar_params – parameters estimated using the "continuous_parametric" method,
- cnonpar_params – parameters estimated using the "continuous_nonparametric" method, currently NULL,
- hm_params – parameters estimated using the "hit_miss" method, currently NULL,
- ratio_kliep – result of kliep(), currently NULL,
- ratio_kliep_list – variable-specific KLIEP results, currently NULL,
- variables – key variables used for comparison,
- comparators – comparison functions used to create comparison vectors,
- methods – MEC estimation methods used for the key variables,
- nonmatch_sample_size – number of full Cartesian-product pairs used to estimate nonmatch parameters,
- nonmatch_sampling_seed – seed used for nonmatch-pair sampling,
- prob_ratio – probability/density ratio type used for blockwise match-count estimation,
- delta – tolerance for changes in the estimated number of matches,
- eps – tolerance for changes in model parameters,
- controls_nleqslv – controls passed to nleqslv(),
- controls_blocking – additional arguments passed to blocking(),
- blocking_result – raw object returned by blocking() if keep_blocking_result = TRUE; otherwise NULL,

- `training_Omega` – pooled training comparison vectors if `keep_training_data = TRUE`; otherwise `NULL`,
- `blocking_eval` – blocking diagnostics if `true_matches` is provided; otherwise `NULL`,
- `eval_metrics` – standard linkage quality metrics if `true_matches` is provided; otherwise `NULL`,
- `confusion` – confusion matrix if `true_matches` is provided; otherwise `NULL`.

Author(s)

Adam Struzik

Examples

```
df_1 <- data.frame(
  name = c("Emma", "Liam", "Olivia", "Noah", "Ava"),
  surname = c("Smith", "Jones", "Brown", "Davis", "Miller"),
  city = c("Boston", "Boston", "Austin", "Austin", "Denver")
)
df_2 <- data.frame(
  name = c("Emma", "Liam", "Olivia", "Noah", "Ava"),
  surname = c("Smith", "Jones", "Brown", "Davis", "Miller"),
  city = c("Boston", "Boston", "Austin", "Austin", "Denver")
)

blocking_x <- matrix(
  c(1, 0, 0, 1, 1, 1, 2, 0, 0, 2),
  ncol = 2,
  byrow = TRUE
)
blocking_y <- blocking_x

result <- mec_blocking(
  A = df_1,
  B = df_2,
  variables = c("name", "surname", "city"),
  blocking_x = blocking_x,
  blocking_y = blocking_y,
  controls_blocking = list(
    representation = "custom_matrix",
    ann = "kd",
    distance = "euclidean",
    seed = 1
  ),
  true_matches = data.frame(a = 1:5, b = 1:5)
)
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,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	A `rec_lin_model` object from `train_rec_lin()` or `custom_rec_lin_model()`.
`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 `custom_rec_lin_model()`).

true_matches	A data.frame or data.table indicating true matches.
verbose	Logical indicating whether to print progress messages.
...	Additional controls passed to predict.rec_lin_model() for custom ML model (used only if the object is from custom_rec_lin_model()).

Details

The predict.rec_lin_model() method 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(). 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.rec_lin_model() method 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(),
  verbose = FALSE
)
```

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).
verbose	Logical indicating whether to print progress messages.

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 $\boldsymbol{\gamma}_{ab} = (\gamma^1_{ab}, \gamma^2_{ab}, \ldots, \gamma^K_{ab})'$ 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 the Jaro-Winkler distance for character variables (which is implemented in the automatedRecLin package as jarowinkler_complement()) 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