

# Package ‘blocking’

June 13, 2025

**Type** Package

**Title** Various Blocking Methods for Entity Resolution

**Version** 1.0.0

**Description** The goal of 'blocking' is to provide blocking methods for record linkage and deduplication using approximate nearest neighbour (ANN) algorithms and graph techniques. It supports multiple ANN implementations via 'rnndescent', 'RcppHNSW', 'RcppAnnoy', and 'mlpack' packages, and provides integration with the 'reclin2' package. The package generates shingles from character strings and similarity vectors for record comparison, and includes evaluation metrics for assessing blocking performance including false positive rate (FPR) and false negative rate (FNR) estimates. For details see: Papadakis et al. (2020) <[doi:10.1145/3377455](https://doi.org/10.1145/3377455)>, Sterts et al. (2014) <[doi:10.1007/978-3-319-11257-2\\_20](https://doi.org/10.1007/978-3-319-11257-2_20)>, Dasylva and Goussanou (2021) <<https://www150.statcan.gc.ca/n1/en/catalogue/12-001-X202100200002>>, Dasylva and Goussanou (2022) <[doi:10.1007/s42081-022-00153-3](https://doi.org/10.1007/s42081-022-00153-3)>.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**URL** <https://github.com/ncn-foreigners/blocking>,  
<https://ncn-foreigners.ue.poznan.pl/blocking/>

**BugReports** <https://github.com/ncn-foreigners/blocking/issues>

**RoxygenNote** 7.3.2

**Imports** text2vec, tokenizers, RcppHNSW, RcppAnnoy, mlpack, rnndescent,  
igraph, data.table, RcppAlgos, methods, readr, utils, Matrix

**Suggests** tinytest, knitr, rmarkdown, reclin2

**VignetteBuilder** knitr

**Depends** R (>= 4.1.0)

**NeedsCompilation** no

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

**Maintainer** Maciej Beręsewicz <maciej.beresewicz@ue.poznan.pl>

Repository CRAN  
Date/Publication 2025-06-13 05:10:02 UTC

Contents

|                           |           |
|---------------------------|-----------|
| blocking . . . . .        | 2         |
| census . . . . .          | 5         |
| cis . . . . .             | 6         |
| controls_ann . . . . .    | 7         |
| controls_txt . . . . .    | 8         |
| control_annoy . . . . .   | 8         |
| control_hnsw . . . . .    | 9         |
| control_kd . . . . .      | 10        |
| control_lsh . . . . .     | 11        |
| control_nnd . . . . .     | 11        |
| est_block_error . . . . . | 13        |
| foreigners . . . . .      | 16        |
| pair_ann . . . . .        | 17        |
| RLdata500 . . . . .       | 18        |
| <b>Index</b>              | <b>20</b> |

---

|          |   |
|----------|---|
| blocking | <i>Block records based on character vectors</i> |
|----------|---|

---

Description

Function creates shingles (strings with 2 characters, default) or vectors using a given model (e.g., GloVe), applies approximate nearest neighbour (ANN) algorithms via the [rnn descent](#), [RcppHNSW](#), [RcppAnnoy](#) and [mlpack](#) packages, and creates blocks using graphs via [igraph](#).

Usage

```
blocking(  
  x,  
  y = NULL,  
  representation = c("shingles", "vectors"),  
  model,  
  deduplication = TRUE,  
  on = NULL,  
  on_blocking = NULL,  
  ann = c("nnd", "hnsw", "annoy", "lsh", "kd"),  
  distance = c("cosine", "euclidean", "l2", "ip", "manhatan", "hamming", "angular"),  
  ann_write = NULL,  
  ann_colnames = NULL,  
  true_blocks = NULL,  
  verbose = c(0, 1, 2),
```

```

graph = FALSE,
seed = 2023,
n_threads = 1,
control_txt = controls_txt(),
control_ann = controls_ann()
)

```

## Arguments

|                |  |
|----------------|--|
| x              | reference data (a character vector or a matrix),   |
| y              | query data (a character vector or a matrix), if not provided NULL by default and thus deduplication is performed,  |
| representation | method of representing input data (possible <code>c("shingles", "vectors")</code> ; default "shingles"),   |
| model          | a matrix containing word embeddings (e.g., GloVe), required only when representation = "vectors",  |
| deduplication  | whether deduplication should be applied (default TRUE as y is set to NULL),  |
| on             | variables for ANN search (currently not supported),  |
| on_blocking    | variables for blocking records before ANN search (currently not supported),  |
| ann            | algorithm to be used for searching for ann (possible, <code>c("nnd", "hnsw", "annoy", "lsh", "kd")</code> ), default "nnd" which corresponds to nearest neighbour descent method), |
| distance       | distance metric (default cosine, more options are possible see details),   |
| ann_write      | writing an index to file. Two files will be created: 1) an index, 2) and text file with column names,  |
| ann_colnames   | file with column names if x or y are indices saved on the disk (currently not supported),  |
| true_blocks    | matrix with true blocks to calculate evaluation metrics (standard metrics based on confusion matrix as well as all metrics from <a href="#">compare</a> are returned).             |
| verbose        | whether log should be provided (0 = none, 1 = main, 2 = ANN algorithm verbose used),   |
| graph          | whether a graph should be returned (default FALSE),  |
| seed           | seed for the algorithms (for reproducibility),   |
| n_threads      | number of threads used for the ANN algorithms and adding data for index and query,   |
| control_txt    | list of controls for text data (passed only to <a href="#">itoken_parallel</a> or <a href="#">itoken</a> ), used only when representation = "shingles",                            |
| control_ann    | list of controls for the ANN algorithms.   |

## Value

Returns a list containing:

- `result` – `data.table` with indices (rows) of `x`, `y`, `block` and distance between points
- `method` – name of the ANN algorithm used,
- `deduplication` – information whether deduplication was applied,
- `representation` – information whether shingles or vectors were used,
- `metrics` – metrics for quality assessment, if `true_blocks` is provided,
- `confusion` – confusion matrix, if `true_blocks` is provided,
- `colnames` – variable names (`colnames`) used for search,
- `graph` – `igraph` class object.

### Author(s)

Maciej Beręsewicz, Adam Struzik

### Examples

```
## an example using RcppHNSW

df_example <- data.frame(txt = c("jankowalski", "kowalskijan", "kowalskimjan",
"kowaljan", "montypython", "pythonmonty", "cyrkmontypython", "monty"))

result <- blocking(x = df_example$txt,
                  ann = "hns",
                  control_ann = controls_ann(hns = control_hns(M = 5, ef_c = 10, ef_s = 10)))

result

## an example using GloVe and RcppAnnoy
## Not run:
old <- getOption("timeout")
options(timeout = 500)
utils::download.file("https://nlp.stanford.edu/data/glove.6B.zip", destfile = "glove.6B.zip")
utils::unzip("glove.6B.zip")

glove_6B_50d <- readr::read_table("glove.6B.50d.txt",
                                col_names = FALSE,
                                show_col_types = FALSE)

data.table::setDT(glove_6B_50d)

glove_vectors <- glove_6B_50d[, -1]
glove_vectors <- as.matrix(glove_vectors)
rownames(glove_vectors) <- glove_6B_50d$X1

## spaces between words are required
df_example_spaces <- data.frame(txt = c("jan kowalski", "kowalski jan", "kowalskim jan",
"kowal jan", "monty python", "python monty", "cyrk monty python", "monty"))

result_annoy <- blocking(x = df_example_spaces$txt,
                        ann = "annoy",
                        representation = "vectors",
                        model = glove_vectors)
```

```

result_annoy

options(timeout = old)

## End(Not run)

```

---

census

---

*Fictional census data*


---

## Description

This data set 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](https://cros-legacy.ec.europa.eu/content/job-training_en)

### Examples

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

---

cis

*Fictional customer data*

---

### Description

This data set 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](https://wayback.archive-it.org/12090/20231221144450/https://cros-legacy.ec.europa.eu/content/job-training_en)

### Examples

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

## Description

Controls for ANN algorithms used in the package.

## Usage

```
controls_ann(  
  sparse = FALSE,  
  k_search = 30,  
  nnd = control_nnd(),  
  hnsw = control_hnsw(),  
  lsh = control_lsh(),  
  kd = control_kd(),  
  annoy = control_annoy()  
)
```

## Arguments

|          |   |
|----------|---|
| sparse   | whether sparse data should be used as an input for algorithms,  |
| k_search | number of neighbours to search,   |
| nnd      | parameters for <a href="#">rnnd_build</a> and <a href="#">rnnd_query</a> (should be inside <a href="#">control_nnd</a> function),   |
| hnsw     | parameters for <a href="#">hnsw_build</a> and <a href="#">hnsw_search</a> (should be inside <a href="#">control_hnsw</a> function), |
| lsh      | parameters for <a href="#">lsh</a> function (should be inside <a href="#">control_lsh</a> function),                                |
| kd       | kd parameters for <a href="#">knn</a> function (should be inside <a href="#">control_kd</a> function),                              |
| annoy    | parameters for <a href="#">RcppAnnoy</a> package (should be inside <a href="#">control_annoy</a> function).                         |

## Value

Returns a list with parameters.

## Author(s)

Maciej Beręsewicz

---

|              |   |
|--------------|---|
| controls_txt | <i>Controls for processing character data</i> |
|--------------|---|

---

**Description**

Controls for text data used in the blocking function (if representation = shingles), passed to [tokenize\\_character\\_shingles](#).

**Usage**

```
controls_txt(  
  n_shingles = 2L,  
  n_chunks = 10L,  
  lowercase = TRUE,  
  strip_non_alphanum = TRUE  
)
```

**Arguments**

|                    |   |
|--------------------|---|
| n_shingles         | length of shingles (default 2L),                                |
| n_chunks           | passed to (default 10L),  |
| lowercase          | should the characters be made lower-case? (default TRUE),       |
| strip_non_alphanum | should punctuation and white space be stripped? (default TRUE). |

**Value**

Returns a list with parameters.

**Author(s)**

Maciej Beręsewicz

---

|               |   |
|---------------|---|
| control_annoy | <i>Controls for the Annoy algorithm</i> |
|---------------|---|

---

**Description**

Controls for Annoy algorithm used in the package (see [RcppAnnoy](#) for details).

**Usage**

```
control_annoy(n_trees = 250, build_on_disk = FALSE, ...)
```



**Arguments**

|               |   |
|---------------|---|
| n_trees       | An integer specifying the number of trees to build in the Annoy index.                    |
| build_on_disk | A logical value indicating whether to build the Annoy index on disk instead of in memory. |
| ...           | Additional arguments.   |

**Value**

Returns a list with parameters.

---

|              |  |
|--------------|--|
| control_hnsw | <i>Controls for the HNSW algorithm</i> |
|--------------|--|

---

**Description**

Controls for HNSW algorithm used in the package (see [RcppHNSW::hnsw\\_build\(\)](#) and [RcppHNSW::hnsw\\_search\(\)](#) for details).

**Usage**

```
control_hnsw(M = 25, ef_c = 200, ef_s = 200, grain_size = 1, byrow = TRUE, ...)
```

**Arguments**

|            |  |
|------------|--|
| M          | Controls the number of bi-directional links created for each element during index construction.  |
| ef_c       | Size of the dynamic list used during construction.   |
| ef_s       | Size of the dynamic list used during search.   |
| grain_size | Minimum amount of work to do (rows in the dataset to add) per thread.  |
| byrow      | If TRUE (the default), this indicates that the items in the dataset to be indexed are stored in each row. Otherwise, the items are stored in the columns of the dataset. |
| ...        | Additional arguments.  |

**Value**

Returns a list with parameters.

control\_kd

*Controls for the k-d tree algorithm***Description**

Controls for KD algorithm used in the package (see [knn](#) for details).

**Usage**

```
control_kd(
  algorithm = "dual_tree",
  epsilon = 0,
  leaf_size = 20,
  random_basis = FALSE,
  rho = 0.7,
  tau = 0,
  tree_type = "kd",
  ...
)
```

**Arguments**

|              |   |
|--------------|---|
| algorithm    | Type of neighbor search: 'naive', 'single_tree', 'dual_tree', 'greedy'.   |
| epsilon      | If specified, will do approximate nearest neighbor search with given relative error.  |
| leaf_size    | Leaf size for tree building (used for kd-trees, vp trees, random projection trees, UB trees, R trees, R* trees, X trees, Hilbert R trees, R+ trees, R++ trees, spill trees, and octrees). |
| random_basis | Before tree-building, project the data onto a random orthogonal basis.  |
| rho          | Balance threshold (only valid for spill trees).   |
| tau          | Overlapping size (only valid for spill trees).  |
| tree_type    | Type of tree to use: 'kd', 'vp', 'rp', 'max-rp', 'ub', 'cover', 'r', 'r-star', 'x', 'ball', 'hilbert-r', 'r-plus', 'r-plus-plus', 'spill', 'oct'.   |
| ...          | Additional arguments.   |

**Value**

Returns a list with parameters.

---

|             |                                       |
|-------------|---------------------------------------|
| control_lsh | <i>Controls for the LSH algorithm</i> |
|-------------|---------------------------------------|

---

**Description**

Controls for LSH algorithm used in the package (see [lsh](#) for details).

**Usage**

```
control_lsh(  
  bucket_size = 10,  
  hash_width = 6,  
  num_probes = 5,  
  projections = 10,  
  tables = 30,  
  ...  
)
```

**Arguments**

|             |  |
|-------------|--|
| bucket_size | The size of a bucket in the second level hash.                       |
| hash_width  | The hash width for the first-level hashing in the LSH preprocessing. |
| num_probes  | Number of additional probes for multiprobe LSH.                      |
| projections | The number of hash functions for each table.                         |
| tables      | The number of hash tables to be used.                                |
| ...         | Additional arguments.  |

**Value**

Returns a list with parameters.

---

|             |                                       |
|-------------|---------------------------------------|
| control_nnd | <i>Controls for the NND algorithm</i> |
|-------------|---------------------------------------|

---

**Description**

Controls for NND algorithm used in the package (see [rnnd\\_build](#) and [rnnd\\_query](#) for details).

**Usage**

```
control_nnd(
  k_build = 30,
  use_alt_metric = FALSE,
  init = "tree",
  n_trees = NULL,
  leaf_size = NULL,
  max_tree_depth = 200,
  margin = "auto",
  n_iters = NULL,
  delta = 0.001,
  max_candidates = NULL,
  low_memory = TRUE,
  n_search_trees = 1,
  pruning_degree_multiplier = 1.5,
  diversify_prob = 1,
  weight_by_degree = FALSE,
  prune_reverse = FALSE,
  progress = "bar",
  obs = "R",
  max_search_fraction = 1,
  epsilon = 0.1,
  ...
)
```

**Arguments**

|                             |  |
|-----------------------------|--|
| <code>k_build</code>        | Number of nearest neighbors to build the index for.  |
| <code>use_alt_metric</code> | If TRUE, use faster metrics that maintain the ordering of distances internally (e.g. squared Euclidean distances if using <code>metric = "euclidean"</code> ), then apply a correction at the end.           |
| <code>init</code>           | Name of the initialization strategy or initial data neighbor graph to optimize.  |
| <code>n_trees</code>        | The number of trees to use in the RP forest. Only used if <code>init = "tree"</code> .   |
| <code>leaf_size</code>      | The maximum number of items that can appear in a leaf. Only used if <code>init = "tree"</code> .   |
| <code>max_tree_depth</code> | The maximum depth of the tree to build (default = 200). Only used if <code>init = "tree"</code> .  |
| <code>margin</code>         | A character string specifying the method used to assign points to one side of the hyperplane or the other.   |
| <code>n_iters</code>        | Number of iterations of nearest neighbor descent to carry out.   |
| <code>delta</code>          | The minimum relative change in the neighbor graph allowed before early stopping. Should be a value between 0 and 1. The smaller the value, the smaller the amount of progress between iterations is allowed. |
| <code>max_candidates</code> | Maximum number of candidate neighbors to try for each item in each iteration.  |

|                           |   |
|---------------------------|---|
| low_memory                | If TRUE, use a lower memory, but more computationally expensive approach to index construction. If set to FALSE, you should see a noticeable speed improvement, especially when using a smaller number of threads, so this is worth trying if you have the memory to spare. |
| n_search_trees            | The number of trees to keep in the search forest as part of index preparation. The default is 1.  |
| pruning_degree_multiplier | How strongly to truncate the final neighbor list for each item.   |
| diversify_prob            | The degree of diversification of the search graph by removing unnecessary edges through occlusion pruning.  |
| weight_by_degree          | If TRUE, then candidates for the local join are weighted according to their in-degree, so that if there are more than max_candidates in a candidate list, candidates with a smaller degree are favored for retention.   |
| prune_reverse             | If TRUE, prune the reverse neighbors of each item before the reverse graph diversification step using pruning_degree_multiplier.  |
| progress                  | Determines the type of progress information logged during the nearest neighbor descent stage.   |
| obs                       | set to C to indicate that the input data orientation stores each observation as a column. The default R means that observations are stored in each row.   |
| max_search_fraction       | Maximum fraction of the reference data to search.   |
| epsilon                   | Controls trade-off between accuracy and search cost.  |
| ...                       | Additional arguments.   |

## Value

Returns a list with parameters.

---

|                 |  |
|-----------------|--|
| est_block_error | <i>Estimate errors due to blocking in record linkage</i> |
|-----------------|--|

---

## Description

Function computes estimators for false positive rate (FPR) and false negative rate (FNR) due to blocking in record linkage, as proposed by Dasylva and Goussanou (2021). Assumes duplicate-free data sources, complete coverage of the reference data set and blocking decisions based solely on record pairs.

**Usage**

```
est_block_error(
  x = NULL,
  y = NULL,
  blocking_result = NULL,
  n = NULL,
  N = NULL,
  G,
  alpha = NULL,
  p = NULL,
  lambda = NULL,
  tol = 10^(-4),
  maxiter = 100,
  sample_size = NULL
)
```

**Arguments**

|                 |   |
|-----------------|---|
| x               | Reference data (required if n and N are not provided).  |
| y               | Query data (required if n is not provided).   |
| blocking_result | data.frame or data.table containing blocking results (required if n is not provided).   |
| n               | Integer vector of numbers of accepted pairs formed by each record in the query data set with records in the reference data set, based on blocking criteria (if NULL, derived from blocking_result). |
| N               | Total number of records in the reference data set (if NULL, derived as length(x)).  |
| G               | Number of classes in the finite mixture model.  |
| alpha           | Numeric vector of initial class proportions (length G; if NULL, initialized as rep(1/G, G)).  |
| p               | Numeric vector of initial matching probabilities in each class of the mixture model (length G; if NULL, randomly initialized from runif(G, 0.5, 1)).  |
| lambda          | Numeric vector of initial Poisson distribution parameters for non-matching records in each class of the mixture model (length G; if NULL, randomly initialized from runif(G, 0.1, 2)).              |
| tol             | Convergence tolerance for the EM algorithm (default $10^{-6}$ ).  |
| maxiter         | Maximum number of iterations for the EM algorithm (default 1000).   |
| sample_size     | Bootstrap sample (from n) size used for calculations (if NULL, uses all data).  |

**Details**

Consider a large finite population that comprises of  $N$  individuals, and two duplicate-free data sources: a register and a file. Assume that the register has no undercoverage, i.e. each record from the file corresponds to exactly one record from the same individual in the register. Let  $n_i$  denote the number of register records which form an accepted (by the blocking criteria) pair with record  $i$  on

the file. Assume that:

- two matched records are neighbours with a probability that is bounded away from 0 regardless of  $N$ ,
- two unmatched records are accidental neighbours with a probability of  $O(\frac{1}{N})$ .

The finite mixture model  $n_i \sim \sum_{g=1}^G \alpha_g (\text{Bernoulli}(p_g) * \text{Poisson}(\lambda_g))$  is assumed. When  $G$  is fixed, the unknown model parameters are given by the vector  $\psi = [(\alpha_g, p_g, \lambda_g)]_{1 \leq g \leq G}$  that may be estimated with the Expectation-Maximization (EM) procedure.

Let  $n_i = n_{i|M} + n_{i|U}$ , where  $n_{i|M}$  is the number of matched neighbours and  $n_{i|U}$  is the number of unmatched neighbours, and let  $c_{ig}$  denote the indicator that record  $i$  is from class  $g$ . For the E-step of the EM procedure, the equations are as follows

$$\begin{aligned}
 P(n_i | c_{ig} = 1) &= I(n_i = 0)(1 - p_g)e^{-\lambda_g} + I(n_i > 0) \left( p_g + (1 - p_g) \frac{\lambda_g}{n_i} \right) \frac{e^{-\lambda_g} \lambda_g^{n_i-1}}{(n_i - 1)!}, \\
 P(c_{ig} = 1 | n_i) &= \frac{\alpha_g P(n_i | c_{ig} = 1)}{\sum_{g'=1}^G \alpha_{g'} P(n_i | c_{ig'} = 1)}, \\
 P(n_{i|M} = 1 | n_i, c_{ig} = 1) &= \frac{p_g n_i}{p_g n_i + (1 - p_g) \lambda_g}, \\
 P(n_{i|U} = n_i | n_i, c_{ig} = 1) &= I(n_i = 0) + I(n_i > 0) \frac{(1 - p_g) \lambda_g}{p_g n_i + (1 - p_g) \lambda_g}, \\
 P(n_{i|U} = n_i - 1 | n_i, c_{ig} = 1) &= \frac{p_g n_i}{p_g n_i + (1 - p_g) \lambda_g}, \\
 E[c_{ig} n_{i|M} | n_i] &= P(c_{ig} = 1 | n_i) P(n_{i|M} = 1 | n_i, c_{ig} = 1), \\
 E[n_{i|U} | n_i, c_{ig} = 1] &= \left( \frac{p_g (n_i - 1) + (1 - p_g) \lambda_g}{p_g n_i + (1 - p_g) \lambda_g} \right) n_i, \\
 E[c_{ig} n_{i|U} | n_i] &= P(c_{ig} = 1 | n_i) E[n_{i|U} | n_i, c_{ig} = 1].
 \end{aligned}$$

The M-step is given by following equations

$$\begin{aligned}
 \hat{p}_g &= \frac{\sum_{i=1}^m E[c_{ig} n_{i|M} | n_i; \psi]}{\sum_{i=1}^m E[c_{ig} | n_i; \psi]}, \\
 \hat{\lambda}_g &= \frac{\sum_{i=1}^m E[c_{ig} n_{i|U} | n_i; \psi]}{\sum_{i=1}^m E[c_{ig} | n_i; \psi]}, \\
 \hat{\alpha}_g &= \frac{1}{m} \sum_{i=1}^m E[c_{ig} | n_i; \psi].
 \end{aligned}$$

As  $N \rightarrow \infty$ , the error rates and the model parameters are related as follows

$$\begin{aligned}
 \text{FNR} &\xrightarrow{p} 1 - E[p(v_i)], \\
 (N - 1) \text{FPR} &\xrightarrow{p} E[\lambda(v_i)],
 \end{aligned}$$

where  $E[p(v_i)] = \sum_{g=1}^G \alpha_g p_g$  and  $E[\lambda(v_i)] = \sum_{g=1}^G \alpha_g \lambda_g$ .

**Value**

Returns a list containing:

- FPR – estimated false positive rate,
- FNR – estimated false negative rate,
- iter – number of the EM algorithm iterations performed,
- convergence – logical, indicating whether the EM algorithm converged within `maxiter` iterations.

**References**

Dasylda, A., Goussanou, A. (2021). Estimating the false negatives due to blocking in record linkage. *Survey Methodology, Statistics Canada, Catalogue No. 12-001-X, Vol. 47, No. 2.*

Dasylda, A., Goussanou, A. (2022). On the consistent estimation of linkage errors without training data. *Jpn J Stat Data Sci* 5, 181–216. [doi:10.1007/s42081022001533](https://doi.org/10.1007/s42081022001533)

**Examples**

```
## an example proposed by Dasylda and Goussanou (2021)
## we obtain results very close to those reported in the paper

set.seed(111)

neighbors <- rep(0:5, c(1659, 53951, 6875, 603, 62, 5))

errors <- est_block_error(n = neighbors,
                          N = 63155,
                          G = 2,
                          tol = 10^(-3),
                          maxiter = 50)

errors
```

---

foreigners

*Fictional 2024 population of foreigners in Poland*

---

**Description**

A fictional data set of the foreign population in Poland, generated based on publicly available information while maintaining the distributions from administrative registers.

**Usage**

```
foreigners
```



**Format**

A data.table with 110000 records. Each row represents one record, with the following columns:

- fname – first name,
- sname – second name,
- surname – surname,
- date – date of birth,
- region – region (county),
- country – country,
- true\_id – person ID.

**Examples**

```
data("foreigners")
head(foreigners)
```

---

|          |   |
|----------|---|
| pair_ann | <i>Integration with the reclin2 package</i> |
|----------|---|

---

**Description**

Function for the integration with the **reclin2** package. The function is based on [pair\\_minsim](#) and reuses some of its source code.

**Usage**

```
pair_ann(
  x,
  y = NULL,
  on,
  deduplication = TRUE,
  keep_block = TRUE,
  add_xy = TRUE,
  ...
)
```

**Arguments**

|               |  |
|---------------|--|
| x             | reference data (a data.frame or a data.table),   |
| y             | query data (a data.frame or a data.table, default NULL),                                 |
| on            | a character with column name or a character vector with column names for the ANN search, |
| deduplication | whether deduplication should be performed (default TRUE),                                |
| keep_block    | whether to keep the block variable in the set,   |
| add_xy        | whether to add x and y,  |
| ...           | arguments passed to <a href="#">blocking</a> function.                                   |

**Value**

Returns a [data.table](#) with two columns `.x` and `.y`. Columns `.x` and `.y` are row numbers from data.frames `x` and `y` respectively. Returned `data.table` is also of a class `pairs` which allows for integration with the [compare\\_pairs](#) function.

**Author(s)**

Maciej Beręsewicz

**Examples**

```
# example using two datasets from reclin2

if (requireNamespace("reclin2", quietly = TRUE)) {

  library(reclin2)
  data("linkexample1", "linkexample2", package = "reclin2")

  linkexample1$txt <- with(linkexample1, tolower(paste0(firstname, lastname, address, sex, postcode)))
  linkexample1$txt <- gsub("\\s+", "", linkexample1$txt)
  linkexample2$txt <- with(linkexample2, tolower(paste0(firstname, lastname, address, sex, postcode)))
  linkexample2$txt <- gsub("\\s+", "", linkexample2$txt)

  # pairing records from linkexample2 to linkexample1 based on txt column

  pair_ann(x = linkexample1, y = linkexample2, on = "txt", deduplication = FALSE) |>
  compare_pairs(on = "txt", comparators = list(cmp_jarowinkler())) |>
  score_simple("score", on = "txt") |>
  select_threshold("threshold", score = "score", threshold = 0.75) |>
  link(selection = "threshold")
}
```

---

RLdata500

*RLdata500 dataset from the RecordLinkage package*


---

**Description**

This data is taken from **RecordLinkage** R package developed by Murat Sariyar and Andreas Borg. The package is licensed under GPL-3 license.

The RLdata500 table contains artificial personal data. Some records have been duplicated with randomly generated errors. RLdata500 contains fifty duplicates.

**Usage**

```
RLdata500
```

**Format**

A `data.table` with 500 records. Each row represents one record, with the following columns:

- `fname_c1` – first name, first component,
- `fname_c2` – first name, second component,
- `lname_c1` – last name, first component,
- `lname_c2` – last name, second component,
- `by` – year of birth,
- `bm` – month of birth,
- `bd` – day of birth,
- `rec_id` – record id,
- `ent_id` – entity id.

**References**

Sariyar M., Borg A. (2022). RecordLinkage: Record Linkage Functions for Linking and Deduplicating Data Sets. R package version 0.4-12.4, <https://CRAN.R-project.org/package=RecordLinkage>

**Examples**

```
data("RLdata500")  
head(RLdata500)
```

# Index

## \* datasets

- census, [5](#)
- cis, [6](#)
- foreigners, [16](#)
- RLdata500, [18](#)

blocking, [2](#), [17](#)

census, [5](#)  
cis, [6](#)  
compare, [3](#)  
compare\_pairs, [18](#)  
control\_annoy, [7](#), [8](#)  
control\_hnsw, [7](#), [9](#)  
control\_kd, [7](#), [10](#)  
control\_lsh, [7](#), [11](#)  
control\_nnd, [7](#), [11](#)  
controls\_ann, [7](#)  
controls\_txt, [8](#)

data.table, [18](#)

est\_block\_error, [13](#)

foreigners, [16](#)

hnsw\_build, [7](#)  
hnsw\_search, [7](#)

igraph, [2](#)  
itoken, [3](#)  
itoken\_parallel, [3](#)

knn, [7](#), [10](#)

lsh, [7](#), [11](#)

mlpack, [2](#)

pair\_ann, [17](#)  
pair\_minsim, [17](#)

RcppAnnoy, [2](#), [7](#), [8](#)

RcppHNSW, [2](#)

RcppHNSW::hnsw\_build(), [9](#)

RcppHNSW::hnsw\_search(), [9](#)

RLdata500, [18](#)

rnnd\_build, [7](#), [11](#)

rnnd\_query, [7](#), [11](#)

rnndescent, [2](#)

tokenize\_character\_shingles, [8](#)