Package: chomper (via r-universe)

June 14, 2026

Title A Comprehensive Hit or Miss Probabilistic Entity Resolution Model

Version 0.1.3

Description Provides Bayesian probabilistic methods for record linkage and entity resolution across multiple datasets using the Comprehensive Hit Or Miss Probabilistic Entity Resolution (CHOMPER) model. The package implements three main inference approaches: (1) Evolutionary Variational Inference for record Linkage (EVIL), (2) Coordinate Ascent Variational Inference (CAVI), and (3) Markov Chain Monte Carlo (MCMC) with split and merge process. The model supports both discrete and continuous fields, and it performs locally-varying hit mechanism for the attributes with multiple truths. It also provides tools for performance evaluation based on either approximated variational factors or posterior samples. The package is designed to support parallel computing with multi-threading support for EVIL to estimate the linkage structure faster.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

LinkingTo Rcpp, RcppArmadillo, RcppThread

Imports Rcpp

Depends R (>= 3.5)

LazyData true

Suggests blink, ggplot2, knitr, patchwork, rmarkdown, salso, spelling

VignetteBuilder knitr

URL https://github.com/hjkim8987/chomper

BugReports https://github.com/hjkim8987/chomper/issues

Language en-US

NeedsCompilation yes

Author Hyungjoon Kim [aut, cre], Andee Kaplan [aut], Matthew Koslovsky [aut]

Maintainer Hyungjoon Kim <hjkim8987@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-03-16 16:40:26 UTC

RemoteUrl https://github.com/cran/chomper

RemoteRef HEAD

RemoteSha 77097e0f34713ac6fd95ac8388a586a87b3e05e8

Contents

`chomper-package` *chomper: A Comprehensive Hit or Miss Probabilistic Entity Resolution Model*

Description

Provides Bayesian probabilistic methods for record linkage and entity resolution across multiple datasets using the Comprehensive Hit Or Miss Probabilistic Entity Resolution (CHOMPER) model. The package implements three main inference approaches: (1) Evolutionary Variational Inference for record Linkage (EVIL), (2) Coordinate Ascent Variational Inference (CAVI), and (3) Markov Chain Monte Carlo (MCMC) with split and merge process. The model supports both discrete and continuous fields, and it performs locally-varying hit mechanism for the attributes with multiple truths. It also provides tools for performance evaluation based on either approximated variational factors or posterior samples. The package is designed to support parallel computing with multi-threading support for EVIL to estimate the linkage structure faster.

Author(s)

Maintainer: Hyungjoon Kim <hjkim8987@gmail.com>

Authors:

- Andee Kaplan <Andee.Kaplan@colostate.edu>
- Matthew Koslovsky <Matt.Koslovsky@colostate.edu>

See Also

Useful links:

- https://github.com/hjkim8987/chomper
- Report bugs at https://github.com/hjkim8987/chomper/issues

`chomperCAVI` *CHOMPER with a single Coordinate Ascent Variational Inference*

Description

Fit the CHOMPER model with a single Coordinate Ascent Variational Inference (CAVI) to estimate the linkage structure across multiple datasets. It returns the approximate variational factors of the linkage structure that maximize the evidence lower bound (ELBO) and other parameters of the CHOMPER model.

Usage

```
chomperCAVI(
  x,
  k,
  n,
  N,
  p,
  M,
  discrete_fields,
  continuous_fields,
  hyper_beta,
  hyper_phi = c(),
  hyper_tau = c(),
  hyper_epsilon_discrete = c(),
  hyper_epsilon_continuous = c(),
  hyper_sigma = matrix(nrow = 0, ncol = 2),
  overlap_prob = 0.5,
  tol_cavi = 1e-05,
  max_iter_cavi = 100,
  max_time = 86400,
  custom_initializer = FALSE,
```

```
  use_checkpoint = FALSE,
  initial_values = NULL,
  checkpoint_values = NULL,
  verbose_internal = TRUE
)
```

Arguments

x	A list of data frames, each representing a dataset.
k	The number of datasets to be linked.
n	The number of rows in each dataset (vector of length k).
N	The number of columns in each dataset.
p	The number of fields in each dataset.
M	The number of categories for each discrete field (vector of length of discrete fields).
discrete_fields	The indexes of the discrete fields (1-based index).
continuous_fields	The indexes of the continuous fields (1-based index).
hyper_beta	The hyperparameters for the beta distribution (matrix of size p x 2).
hyper_phi	The hyperparameters for softmax representation (vector of length of discrete fields).
hyper_tau	The temperature parameter (vector of length of discrete fields).
hyper_epsilon_discrete	The range parameter for the comprehensive hit of discrete fields (vector of length of discrete fields).
hyper_epsilon_continuous	The range parameter for the comprehensive hit of continuous fields (vector of length of continuous fields).
hyper_sigma	The hyperparameters for the Inverse Gamma distribution (matrix of size length of continuous fields x 2).
overlap_prob	The presumed probability of overlap across the datasets.
tol_cavi	The tolerance for the coordinate ascent variational inference for the convergence.
max_iter_cavi	The maximum number of iterations for the coordinate ascent variational inference.
max_time	The maximum time limit for the execution in seconds.
custom_initializer	Whether to use a custom initializer for the initial values.
use_checkpoint	Whether to use a checkpoint.
initial_values	The initial values for the parameters (optional).
checkpoint_values	The checkpoint values for the parameters (optional).
verbose_internal	Whether to print the internal C++ messages (TRUE: print, FALSE: not print).

Value

A list of the approximated parameters of the variational factors and other information containing:

- `nu`: A list of parameter matrices for the approximate multinomial posterior of the linkage structure.
- `omega`: A matrix of parameter vectors for the approximate beta posterior of the distortion ratio.
- `rho`: A list of parameter matrices for the approximate Bernoulli posterior of the distortion indicators.
- `gamma`: A list of parameter matrices for the approximate multinomial posterior of discrete true latent values.
- `alpha`: A list of parameter vectors for the approximate Dirichlet posterior (theta) of the discrete true latent values.
- `eta_tilde`: A matrix of parameter vectors for the mean of the approximate normal posterior of continuous true latent values.
- `eta_mean`: A vector of mean parameters for the approximate normal posterior (eta_tilde) of the continuous true latent values.
- `eta_var`: A vector of variance parameters for the approximate normal posterior (eta_tilde) of the continuous true latent values.
- `sigma_tilde`: A matrix of parameter vectors for the variance of the approximate normal posterior of continuous true latent values.
- `sigma_shape`: A vector of shape parameters for the approximate inverse gamma posterior (sigma_tilde) of the continuous true latent values.
- `sigma_scale`: A vector of scale parameters for the approximate inverse gamma posterior (sigma_tilde) of the continuous true latent values.
- `ELBO`: The final maximum ELBO from a single CAVI.
- `niter`: The number of iterations for a single CAVI.
- `interruption`: Whether the CHOMPER-CAVI is interrupted. The fitting is interrupted if the elapsed time reaches the maximum time limit.
- `cavi_elapsed_time`: The maximum elapsed time of a single CAVI iteration.
- `elapsed_time`: The elapsed time of the entire CAVI process in seconds.

Examples

```
# 1. Generate sample data for testing
sample_data <- generate_sample_data(
  n_entities = 10,
  n_files = 3,
  overlap_ratio = 0.7,
  discrete_columns = c(1, 2),
  discrete_levels = c(3, 3),
  continuous_columns = c(3, 4),
  continuous_params = matrix(c(0, 0, 1, 1), ncol = 2),
  distortion_ratio = c(0.1, 0.1, 0.1, 0.1)
)
```

```
# 2. Get file information and remove `id` from the original data
n <- numeric(3)
x <- list()
for (i in 1:3) {
  n[i] <- nrow(sample_data[[i]])
  x[[i]] <- sample_data[[i]][, -1]
}
N <- sum(n)

# 3. Set Hyperparameters
hyper_beta <- matrix(
  rep(c(N * 0.1 * 0.01, N * 0.1), 4),
  ncol = 2, byrow = TRUE
)

hyper_sigma <- matrix(
  rep(c(0.01, 0.01), 2),
  ncol = 2, byrow = TRUE
)

# 4. Fit CHOMPER-CAVI
result <- chomperCAVI(
  x = x,
  k = 3, # number of datasets
  n = n, # rows per dataset
  N = N, # columns per dataset
  p = 4, # fields per dataset
  M = c(3, 3), # categories for discrete fields
  discrete_fields = c(1, 2),
  continuous_fields = c(3, 4),
  hyper_beta = hyper_beta, # hyperparameter for distortion rate
  hyper_sigma = hyper_sigma, # hyperparameter for continuous fields
  hyper_phi = c(2.0, 2.0),
  hyper_tau = c(0.01, 0.01),
  hyper_epsilon_discrete = c(0, 0),
  hyper_epsilon_continuous = c(0.001, 0.001),
)
```

`chomperEVIL` *CHOMPER with Evolutionary Variational Inference for Record Linkage*

Description

Fit the CHOMPER model with Evolutionary Variational Inference for record linkage (EVIL) to estimate the linkage structure across multiple datasets. It returns the approximate variational factors of the linkage structure that maximize the evidence lower bound (ELBO) and other parameters of the CHOMPER model.

Usage

```
chomperEVIL(
  x,
  k,
  n,
  N,
  p,
  M,
  discrete_fields,
  continuous_fields,
  hyper_beta,
  hyper_phi = c(),
  hyper_tau = c(),
  hyper_epsilon_discrete = c(),
  hyper_epsilon_continuous = c(),
  hyper_sigma = matrix(nrow = 0, ncol = 2),
  overlap_prob = 0.5,
  n_parents = 5,
  n_children = 10,
  tol_cavi = 1e-05,
  max_iter_cavi = 100,
  tol_evi = 1e-05,
  max_iter_evi = 50,
  n_threads = 1,
  max_time = 86400,
  custom_initializer = FALSE,
  use_checkpoint = FALSE,
  initial_values = NULL,
  checkpoint_values = NULL,
  verbose_internal = TRUE
)
```

Arguments

x	A list of data frames, each representing a dataset.
k	The number of datasets to be linked.
n	The number of rows in each dataset (vector of length k).
N	The number of columns in each dataset.
p	The number of fields in each dataset.
M	The number of categories for each discrete field (vector of length of discrete fields).
discrete_fields	The indexes of the discrete fields (1-based index).
continuous_fields	The indexes of the continuous fields (1-based index).
hyper_beta	The hyperparameters for the beta distribution (matrix of size p x 2).

`hyper_phi` The hyperparameters for softmax representation (vector of length of discrete fields).

`hyper_tau` The temperature parameter (vector of length of discrete fields).

`hyper_epsilon_discrete`
The range parameter for the comprehensive hit of discrete fields (vector of length of discrete fields).

`hyper_epsilon_continuous`
The range parameter for the comprehensive hit of continuous fields (vector of length of continuous fields).

`hyper_sigma` The hyperparameters for the Inverse Gamma distribution (matrix of size length of continuous fields x 2).

`overlap_prob` The presumed probability of overlap across the datasets.

`n_parents` The number of parents for a generation.

`n_children` The number of children for the next generation.

`tol_cavi` The tolerance for the coordinate ascent variational inference for the convergence.

`max_iter_cavi` The maximum number of iterations for the coordinate ascent variational inference.

`tol_evi` The tolerance for the evolutionary variational inference for the convergence.

`max_iter_evi` The maximum number of iterations for the evolutionary variational inference.

`n_threads` The number of threads for parallel computation.

`max_time` The maximum time limit for the execution in seconds.

`custom_initializer`
Whether to use a custom initializer for the initial values.

`use_checkpoint` Whether to use a checkpoint.

`initial_values` The initial values for the parameters (optional).

`checkpoint_values`
The checkpoint values for the parameters (optional).

`verbose_internal`
Whether to print the internal C++ messages (TRUE: print, FALSE: not print).

Value

A list of the approximated parameters of the variational factors and other information containing:

- `nu`: A list of parameter matrices for the approximate multinomial posterior of the linkage structure.
- `omega`: A matrix of parameter vectors for the approximate beta posterior of the distortion ratio.
- `rho`: A list of parameter matrices for the approximate Bernoulli posterior of the distortion indicators.
- `gamma`: A list of parameter matrices for the approximate multinomial posterior of discrete true latent values.

- alpha: A list of parameter vectors for the approximate Dirichlet posterior (theta) of the discrete true latent values.
- eta_tilde: A matrix of parameter vectors for the mean of the approximate normal posterior of continuous true latent values.
- eta_mean: A vector of mean parameters for the approximate normal posterior (eta_tilde) of the continuous true latent values.
- eta_var: A vector of variance parameters for the approximate normal posterior (eta_tilde) of the continuous true latent values.
- sigma_tilde: A matrix of parameter vectors for the variance of the approximate normal posterior of continuous true latent values.
- sigma_shape: A vector of shape parameters for the approximate inverse gamma posterior (sigma_tilde) of the continuous true latent values.
- sigma_scale: A vector of scale parameters for the approximate inverse gamma posterior (sigma_tilde) of the continuous true latent values.
- ELBO: A vector of maximum ELBO at each generation.
- niter: The number of generations EVIL created.
- interruption: Whether the CHOMPER-EVIL is interrupted. The fitting is interrupted if the elapsed time reaches the maximum time limit.
- maximum_elapsed_time: The maximum elapsed time of a single CAVI iteration throughout the entire EVIL process.
- elapsed_time: The elapsed time of the entire EVIL process in seconds.

Examples

```
# 1. Generate sample data for testing
sample_data <- generate_sample_data(
  n_entities = 10,
  n_files = 3,
  overlap_ratio = 0.7,
  discrete_columns = c(1, 2),
  discrete_levels = c(3, 3),
  continuous_columns = c(3, 4),
  continuous_params = matrix(c(0, 0, 1, 1), ncol = 2),
  distortion_ratio = c(0.1, 0.1, 0.1, 0.1)
)

# 2. Get file information and remove `id` from the original data
n <- numeric(3)
x <- list()
for (i in 1:3) {
  n[i] <- nrow(sample_data[[i]])
  x[[i]] <- sample_data[[i]][, -1]
}
N <- sum(n)

# 3. Set Hyperparameters
hyper_beta <- matrix(
```

```
  rep(c(N * 0.1 * 0.01, N * 0.1), 4),
  ncol = 2, byrow = TRUE
)

hyper_sigma <- matrix(
  rep(c(0.01, 0.01), 2),
  ncol = 2, byrow = TRUE
)

# 4. Fit CHOMPER-EVIL
result <- chomperEVIL(
  x = x,
  k = 3, # number of datasets
  n = n, # rows per dataset
  N = N, # columns per dataset
  p = 4, # fields per dataset
  M = c(3, 3), # categories for discrete fields
  discrete_fields = c(1, 2),
  continuous_fields = c(3, 4),
  hyper_beta = hyper_beta, # hyperparameter for distortion rate
  hyper_sigma = hyper_sigma, # hyperparameter for continuous fields
  hyper_phi = c(2.0, 2.0),
  hyper_tau = c(0.01, 0.01),
  hyper_epsilon_discrete = c(0, 0),
  hyper_epsilon_continuous = c(0.001, 0.001),
  n_threads = 1
)
```

`chomperMCMC` *CHOMPER with Markov chain Monte Carlo with Split and Merge Process*

Description

Fit the CHOMPER model with Markov chain Monte Carlo (MCMC) with split and merge process to estimate the linkage structure across multiple datasets. It returns the posterior samples of the linkage structure and other parameters of the CHOMPER model.

Usage

```
chomperMCMC(
  x,
  k,
  n,
  N,
  p,
  M,
  discrete_fields,
```

```
  continuous_fields,
  hyper_beta,
  hyper_phi = c(),
  hyper_tau = c(),
  hyper_epsilon_discrete = c(),
  hyper_epsilon_continuous = c(),
  hyper_sigma = matrix(nrow = 0, ncol = 2),
  n_burnin = 1000,
  n_gibbs = 1000,
  n_split_merge = 10,
  max_time = 86400,
  custom_initializer = FALSE,
  use_checkpoint = FALSE,
  initial_values = NULL,
  checkpoint_values = NULL,
  verbose_internal = TRUE
)
```

Arguments

x	A list of data frames, each representing a dataset.
k	The number of datasets to be linked.
n	The number of rows in each dataset (vector of length k).
N	The number of columns in each dataset.
p	The number of fields in each dataset.
M	The number of categories for each discrete field (vector of length of discrete fields).
discrete_fields	The indexes of the discrete fields (1-based index).
continuous_fields	The indexes of the continuous fields (1-based index).
hyper_beta	The hyperparameters for the beta distribution (matrix of size p x 2).
hyper_phi	The hyperparameters for softmax representation (vector of length of discrete fields).
hyper_tau	The temperature parameter (vector of length of discrete fields).
hyper_epsilon_discrete	The range parameter for the comprehensive hit of discrete fields (vector of length of discrete fields).
hyper_epsilon_continuous	The range parameter for the comprehensive hit of continuous fields (vector of length of continuous fields).
hyper_sigma	The hyperparameters for the Inverse Gamma distribution (matrix of size length of continuous fields x 2).
n_burnin	The number of burn-in iterations for the MCMC.
n_gibbs	The number of Gibbs sampling iterations for the MCMC.

`n_split_merge` The number of split and merge iterations for the MCMC.

`max_time` The maximum time limit for the execution in seconds.

`custom_initializer`
Whether to use a custom initializer for the initial values.

`use_checkpoint` Whether to use a checkpoint.

`initial_values` The initial values for the parameters (optional).

`checkpoint_values`
The checkpoint values for the parameters (optional).

`verbose_internal`
Whether to print the internal C++ messages (TRUE: print, FALSE: not print).

Value

A list containing the posterior samples.

A list of the posterior samples and other information containing:

- `lambda`: A list of posterior samples (integer vectors) of the linkage structure.
- `z`: A list of posterior samples (binary matrices) of the distortion indicators.
- `y`: A list of posterior samples (matrices) of the true latent records.
- `beta`: A list of posterior samples (numeric vectors) of the distortion ratio.
- `theta`: A list of posterior samples (numeric vectors) of the probabilities of discrete true latent values.
- `eta`: A list of posterior samples (numeric vectors) of the mean of continuous true latent values.
- `sigma`: A list of posterior samples (numeric vectors) of the variance of continuous true latent values.
- `n_sample`: Total number of posterior samples after burn-in.
- `n_shift`: Total number of accepted split and merge results after burn-in.
- `elapsed_time`: The elapsed time of the entire MCMC process in seconds.
- `interruption`: Whether the CHOMPER-MCMC is interrupted. The fitting is interrupted if the elapsed time reaches the maximum time limit.

Examples

```
# 1. Generate sample data for testing
sample_data <- generate_sample_data(
  n_entities = 10,
  n_files = 3,
  overlap_ratio = 0.7,
  discrete_columns = c(1, 2),
  discrete_levels = c(3, 3),
  continuous_columns = c(3, 4),
  continuous_params = matrix(c(0, 0, 1, 1), ncol = 2),
  distortion_ratio = c(0.1, 0.1, 0.1, 0.1)
)
```

```
# 2. Get file information and remove `id` from the original data
n <- numeric(3)
x <- list()
for (i in 1:3) {
  n[i] <- nrow(sample_data[[i]])
  x[[i]] <- sample_data[[i]][, -1]
}
N <- sum(n)

# 3. Set Hyperparameters
hyper_beta <- matrix(
  rep(c(N * 0.1 * 0.01, N * 0.1), 4),
  ncol = 2, byrow = TRUE
)

hyper_sigma <- matrix(
  rep(c(0.01, 0.01), 2),
  ncol = 2, byrow = TRUE
)

# 4. Fit CHOMPER-MCMC
result <- chomperMCMC(
  x = x,
  k = 3, # number of datasets
  n = n, # rows per dataset
  N = N, # columns per dataset
  p = 4, # fields per dataset
  M = c(3, 3), # categories for discrete fields
  discrete_fields = c(1, 2),
  continuous_fields = c(3, 4),
  hyper_beta = hyper_beta, # hyperparameter for distortion rate
  hyper_sigma = hyper_sigma, # hyperparameter for continuous fields
  hyper_phi = c(2.0, 2.0),
  hyper_tau = c(0.01, 0.01),
  hyper_epsilon_discrete = c(0, 0),
  hyper_epsilon_continuous = c(0.001, 0.001),
  n_burnin = 0,
  n_gibbs = 100,
  n_split_merge = 10
)
```

`flatten_posterior_samples` *Flatten the posterior samples, lambda, into a matrix*

Description

This function converts a list of posterior samples of lambda into a matrix. Before calculating the posterior similarity matrix using `psm_mcmc`, it is necessary to flatten the posterior samples into a matrix.

Usage

```
flatten_posterior_samples(samples, k, N)
```

Arguments

samples	a list of MCMC samples
k	number of files to be linked
N	total number of records

Value

an N by number of MCMC samples matrix

Examples

```
# 1. Create a list of posterior samples of linkage structure
number_of_files <- 2

n_file1 <- 2
n_file2 <- 3

number_of_records <- n_file1 + n_file2

number_of_samples <- 10
lambda <- list()
for (i in 1:number_of_samples) {
  lambda[[i]] <- list(
    sample(1:number_of_records, n_file1, TRUE),
    sample(1:number_of_records, n_file2, TRUE)
  )
}

# 2. Converts a list of posterior samples of lambda into a matrix
flatten_posterior_samples(lambda, number_of_files, number_of_records)
```

---

generate_sample_data *Generate synthetic data for record linkage*

Description

Generate synthetic data for record linkage with given number of entities, files, and overlap ratio. Each variable can follow either a multinomial or a Gaussian distribution. User can specify the existence of multiple truths for each variable.

Usage

```
generate_sample_data(
  n_entities,
  n_files,
  overlap_ratio,
  discrete_columns,
  discrete_levels,
  continuous_columns,
  continuous_params,
  distortion_ratio,
  discrete_fuzziness = NULL,
  continuous_fuzziness = NULL
)
```

Arguments

`n_entities`	The number of entities.
`n_files`	The number of files.
`overlap_ratio`	The ratio of overlapping entities across the files.
`discrete_columns`	The indices of the discrete columns (1-based index).
`discrete_levels`	The levels of the discrete columns (vector of length of discrete columns).
`continuous_columns`	The indices of the continuous columns (1-based index).
`continuous_params`	The parameters of the continuous columns (matrix of size length of continuous columns x 2).
`distortion_ratio`	The distortion ratio of the columns (vector of length of total columns).
`discrete_fuzziness`	The configuration of the multiple truths of the discrete columns (optional).
`continuous_fuzziness`	The configuration of the multiple truths of the continuous columns (optional).

Value

A list of matrices containing the noisy synthetic data. Each matrix represents a file.

Examples

```
# 1. Set number of entities, files, and overlap ratio
n_entities <- 25
n_files <- 2
overlap_ratio <- 0.9

# 2. Set attributes information
```

```
discrete_columns <- 1:4
discrete_levels <- rep(5, 4)
continuous_columns <- 5:6
continuous_params <-
  matrix(c(0, 10, 10, 10),
    ncol = 2, byrow = TRUE # means and variances
  )

# 3. Set distortion ratio and fuzziness information
distortion_ratio <- rep(0.01, 6)
discrete_fuzziness <- matrix(c(4, 1),
  ncol = 2, byrow = TRUE
)
continuous_fuzziness <- matrix(c(5, 0.5^2, 6, 0.5^2),
  ncol = 2, byrow = TRUE
)

# 4. Generate synthetic data
simulation_data <- generate_sample_data(
  n_entities, n_files, overlap_ratio,
  discrete_columns, discrete_levels,
  continuous_columns, continuous_params,
  distortion_ratio,
  discrete_fuzziness, continuous_fuzziness
)
```

`italy` *Italian Survey on Household Income and Wealth (ISHIW) data from 2020 and 2022*

Description

Two data files collected in 2020 and 2022 surveys. Totally 38,255 records from 28,000 participants exist across two survey. Categorical variables are recorded with integers, and continuous variables are standardized to have a mean of 0 and a variance of 1.

Usage

```
italy
```

Format

A list of matrices with 9 variables. 15,198 rows and 23,057 rows are contained in surveys in 2020 and 2022, respectively.

ID Unique ID of each entity

SEX sex

ANASC year of birth

CIT nationality (whether or not Italian)

NASCREG administrative region of birth

STUDIO education level

IREG administrative region of current residence

NETINCOME net income

VALABIT estimated price of residence

Source

https://www.bancaditalia.it/statistiche/tematiche/indagini-famiglie-imprese/bilanci-famiglie/distribuzione-microdati/index.html

---

`performance` *Evaluate the performance of the linkage structure estimation*

Description

Based on the true linkage structure and the estimate, it will calculate several metrics including true positive, true negative, false positive, false negative, false positive rate, and false negative rate. This package recommends using the output of `links` function of the `blink` package as an argument to `performance` function.

Usage

```
performance(estimation, truth, N, return_matrix = FALSE)
```

Arguments

`estimation`	estimated linkage structure
`truth`	true linkage structure
`N`	total number of records
`return_matrix`	if true, it also returns the matrix of linkage structure

Value

a list with performance metrics. If `return_matrix` is true, it also returns the matrix of linkage structure used for the evaluation.

Examples

```
# 1. True linkage structure
total_number_of_records <- 6

truth <- matrix(
  list(c(1), c(2, 4), c(3), c(2, 4), c(5, 6), c(5, 6))
)

# 2. Estimated linkage structure
estimation <- matrix(
  list(c(1), c(2, 4), c(3), c(2, 4), c(5), c(6))
)

# 3. Calculate performance metrics
performance(estimation, truth, total_number_of_records, FALSE)
```

---

psm_mcmc *Calculate the posterior similarity matrix*

Description

This function returns a posterior similarity matrix based on the MCMC samples of lambda, obtained from chomperMCMC.

Usage

```
psm_mcmc(samples)
```

Arguments

samples	a total number of records by number of MCMC samples matrix with MCMC samples

Value

a posterior similarity matrix of all possible pairs

Examples

```
# 1. Create a matrix with posterior samples of linkage structure
n_file1 <- 2
n_file2 <- 3

number_of_records <- n_file1 + n_file2

number_of_samples <- 10
lambda_matrix <- matrix(nrow = number_of_samples, ncol = number_of_records)
for (i in 1:number_of_samples) {
```

```
  lambda_matrix[i, ] <- sample(1:number_of_records, number_of_records, TRUE)
}

# 2. Calculate a posterior similarity matrix
psm_mcmc(lambda_matrix)
```

---

psm_vi *Calculate the posterior similarity matrix*

Description

This function returns a posterior similarity matrix based on the parameters of the approximated variational factor, nu, obtained from either chomperEVIL or chomperCAVI.

Usage

```
psm_vi(probs_field)
```

Arguments

probs_field	a list of matrices with posterior probabilities

Value

a posterior similarity matrix of all possible pairs

Examples

```
# 1. Create an approximate posterior distribution of linkage structure
n_file1 <- 2
n_file2 <- 3

nu1 <- matrix(runif(n_file1^2 * n_file2), nrow = n_file1)
for (i in 1:n_file1) {
  nu1[i, ] <- nu1[i, ] / sum(nu1[i, ])
}

nu2 <- matrix(runif(n_file1 * n_file2^2), nrow = n_file2)
for (i in 1:n_file2) {
  nu2[i, ] <- nu2[i, ] / sum(nu2[i, ])
}

# 2. Convert into the appropriate type to run a function
approximate_posterior <- list(nu1, nu2)

# 3. Calculate a posterior similarity matrix
psm_vi(approximate_posterior)
```

simulation.high *Synthetic data with high overlap ratio (70%)*

Description

A synthetic data to illustrate the functionality of `chomper` model. A list contains 2 matrices to perform entity resolution, and records are generated with 750 unique entities. Each file contains 8 variables including unique id and 646 and 629 records, respectively. Categorical variables are recorded with integers, and continuous variables are standardized to have a mean of 0 and a variance of 1.

Usage

```
simulation.high
```

Format

A list of matrices with 8 variables. 646 rows and 629 rows are contained in data, respectively.

id Unique ID of each entity

X01 categorical variable with a single truth of 8 levels

X02 categorical variable with a single truth of 8 levels

X03 categorical variable with a single truth of 8 levels

X04 categorical variable with 1-adjacent multiple truths of 8 levels

X05 categorical variable with 1-adjacent multiple truths of 8 levels

X06 continuous variable with multiple truths

X07 continuous variable with multiple truths

Source

Generated by the package authors. Authors used `generate_sample_data` function in `R/generate_sample_data.R`.

simulation.low *Synthetic data with low overlap ratio (30%)*

Description

A synthetic data to illustrate the functionality of `chomper` model. A list contains 2 matrices to perform entity resolution, and records are generated with 750 unique entities. Each file contains 8 variables including unique id and 478 and 497 records, respectively. Categorical variables are recorded with integers, and continuous variables are standardized to have a mean of 0 and a variance of 1.

Usage

```
simulation.low
```

Format

A list of matrices with 8 variables. 478 rows and 497 rows are contained in data, respectively.

id Unique ID of each entity

X01 categorical variable with a single truth of 8 levels

X02 categorical variable with a single truth of 8 levels

X03 categorical variable with a single truth of 8 levels

X04 categorical variable with 1-adjacent multiple truths of 8 levels

X05 categorical variable with 1-adjacent multiple truths of 8 levels

X06 continuous variable with multiple truths

X07 continuous variable with multiple truths

Source

Generated by the package authors. Authors used `generate_sample_data` function in `R/generate_sample_data.R`.

`simulation.medium` *Synthetic data with medium overlap ratio (50%)*

Description

A synthetic data to illustrate the functionality of `chomper` model. A list contains 2 matrices to perform entity resolution, and records are generated with 750 unique entities. Each file contains 8 variables including unique id and 569 and 556 records, respectively. Categorical variables are recorded with integers, and continuous variables are standardized to have a mean of 0 and a variance of 1.

Usage

```
simulation.medium
```

Format

A list of matrices with 8 variables. 569 rows and 556 rows are contained in data, respectively.

id Unique ID of each entity

X01 categorical variable with a single truth of 8 levels

X02 categorical variable with a single truth of 8 levels

X03 categorical variable with a single truth of 8 levels

X04 categorical variable with 1-adjacent multiple truths of 8 levels

X05 categorical variable with 1-adjacent multiple truths of 8 levels

X06 continuous variable with multiple truths

X07 continuous variable with multiple truths

Source

Generated by the package authors. Authors used `generate_sample_data` function in `R/generate_sample_data.R`.

Index