Package: FlexRL (via r-universe)

November 8, 2026

Title Flexible Record Linkage and Linked Data Quality Assessment

Version 2.0.0

Description Probabilistically link records that refer to the same entities across two data sources without a unique identifier, using partially identifying variables such as product code, brand, category, birth year, sex or postal code. 'FlexRL' implements a Stochastic Expectation Maximisation (StEM) approach to Record Linkage (Robach et al., 2025, <doi:10.1093/jrsssc/qlaf016>). The model accounts for registration errors (missing values and mistakes) and for variables that change over time, enforces one-to-one assignment, and has a low memory footprint. The package also provides tools for inference on linked data: two estimators of the false discovery proportion of a linkage (Robach et al., 2025, <doi:10.1002/sim.70292>), based on linkage scores and on synthetic data, and diagnostics comparing the linked sample with the source data. These tools also apply on the linkage output of other record linkage packages.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.1.0)

Imports Matrix (>= 1.7), Rcpp (>= 1.0.13), cli, graphics, grDevices, stats, utils

LinkingTo Rcpp

VignetteBuilder knitr

Suggests foreach, knitr, rmarkdown, testthat (>= 3.0.0), arf, mice, synthpop, BRL, diyar, fastLink, fedmatch, multilink, reclin2

Config/testthat/edition 3

URL https://github.com/robachowyk/FlexRL, https://robachowyk.github.io/FlexRL/

BugReports https://github.com/robachowyk/FlexRL/issues

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Kayané ROBACH [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-5725-623X>), Michel H. HOF [aut, cph]

Maintainer Kayané ROBACH <k.c.robach@amsterdamumc.nl>

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

Date/Publication 2026-11-08 15:23:08 UTC

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

RemoteRef HEAD

RemoteSha 063e6a890380f15d36322db59bd4e92a518bce65

Contents

compute_augmRL_FDP_synth . . . 3
compute_RL_FDP_score . . . 5
create_data_alpha . . . 6
FDP_score . . . 7
FDP_synth . . . 8
FlexRL . . . 10
link_with_BRL . . . 11
link_with_diyar . . . 12
link_with_fastLink . . . 13
link_with_fedmatch . . . 14
link_with_FlexRL . . . 15
link_with_multilink . . . 15
link_with_reclin2 . . . 16
log_lik . . . 17
log_possible_config . . . 18
mmd . . . 18
naive_linkage . . . 19
plot.discrepancy_curves . . . 20
plot.FDP_curves . . . 20
plot.RL_diagnostics . . . 21
plot_distributions . . . 22
plot_linkage_scores . . . 23
plot_StEM_convergence . . . 24
prepare_data . . . 25
print.RL_diagnostics . . . 26
prop_level . . . 27
RL_agreement . . . 28
RL_diagnostics . . . 29
simulate_data . . . 31
simulateD . . . 33
simulateH . . . 35

smd 37
StEM 38
support_iou 41
survival_model 41
survival_S 43
synthesise 44

Index **45**

`compute_augmRL_FDP_synth` *Estimate the false discovery proportion of a record-linkage method via synthetic augmentation*

Description

Repeatedly augments file B with synthetic records (`synthesise()`), runs the chosen record-linkage method link_with_*, and compares the synthetic-vs-real proportion among linked pairs to estimate the false discovery proportion, for a range of score thresholds (0.50 to 0.99). See https://doi.org/10.1002/sim.70292 for the method.

Usage

```
compute_augmRL_FDP_synth(
  synth_method,
  encodedA,
  encodedB,
  PIVs,
  n_synth = NULL,
  restrict_support_intersection = TRUE,
  maxIter4CV = 10,
  n_repeats = 10,
  RL_method,
  ...
)
```

Arguments

`synth_method`	Passed to `synthesise()`: `"arf"`, `"synthpop"`, or `"mice"`.
`encodedA, encodedB`	The two encoded data sources as returned by `prepare_data()` (`encodedA` is the smaller one).
`PIVs`	Character vector, names of the PIVs.
`n_synth`	Integer, number of synthetic records to generate per iteration (default: 10% of `nrow(encodedB)`).
`restrict_support_intersection`	Passed to `synthesise()`.

maxIter4CV	Integer, max number of retries per iteration if no valid FDP estimate is obtained.
n_repeats	Integer, number of augmentation iterations to average over.
RL_method	One of "multilink", "fastLink", "BRL", "reclin2", "diyar", "fedmatch", "FlexRL".
...	Extra arguments forwarded to the chosen link_with_* wrapper (i.e. to the underlying record-linkage package).

Value

List with FDP_score_estimator, FDP_synth_estimator, Linked_pairs_augm, Linked_pairs: data frames (n_repeats rows x 50 thresholds, 0.50 to 0.99), and FDP_score_sd, FDP_synth_sd Numeric, standard deviations of the corresponding estimators over the n_repeats iterations per threshold (NA when fewer than two valid repeats are available).

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                      cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
                   V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                   V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list( V1 = c(), V2 = c(),
                            V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params, TRUE )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
PIVs <- names(PIVs_config)
compute_augmRL_FDP_synth( "arf", prep_data$encodedA, prep_data$encodedB, PIVs,
                          NULL, TRUE, 1, 2, "BRL", flds = PIVs,
                          types = rep("bi",length(PIVs)) )
compute_augmRL_FDP_synth( "arf", prep_data$encodedA, prep_data$encodedB, PIVs,
                          NULL, TRUE, 1, 2, "FlexRL", data = prep_data,
                          StEM_iter = 5, StEM_burnin = 2,
                          gibbs_iter = 5, gibbs_burnin = 2, n_post_sample = 5 )
```

`compute_RL_FDP_score` *Estimate the false discovery proportion of a record-linkage method via model specific linkage scores*

Description

Estimate the false discovery proportion of a record-linkage method via model specific linkage scores

Usage

```
compute_RL_FDP_score(
  encodedA,
  encodedB,
  PIVs,
  maxIter4CV = 10,
  n_repeats = 10,
  RL_method,
  ...
)
```

Arguments

`encodedA`	Encoded data source as returned by `prepare_data()` (`encodedA` is the smaller one).
`encodedB`	Encoded data source as returned by `prepare_data()` (`encodedB` is the larger one).
`PIVs`	Character vector, names of the PIVs.
`maxIter4CV`	Integer, max number of retries per iteration if no valid FDP estimate is obtained.
`n_repeats`	Integer, number of augmentation iterations to average over.
`RL_method`	One of `"multilink"`, `"fastLink"`, `"BRL"`, `"reclin2"`, `"diyar"`, `"fedmatch"`, `"FlexRL"`.
`...`	Extra arguments forwarded to the chosen link_with_* wrapper (i.e. to the underlying record-linkage package).

Value

List with `FDP_score_estimator`, `Linked_pairs`: data frames (`n_repeats` rows x 50 thresholds, 0.50 to 0.99), and `FDP_score_sd` Numeric, standard deviation of `FDP_score_estimator` over the `n_repeats` iterations per threshold (`NA` when fewer than two valid repeats are available).

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
```

```
                              bound_mistakes = c(0.10,0.10),
                              fix_mistakes = c(NA,NA)),
                        V3 = list(dynamics = "flexible",
                              bound_mistakes = c(NA,NA),
                              fix_mistakes = c(NA,NA)),
                        V4 = list(dynamics = "structured",
                              bound_mistakes = c(NA,NA),
                              fix_mistakes = c(0.03,0.03),
                              cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                     cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
                   V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                   V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list( V1 = c(), V2 = c(),
                            V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params, TRUE )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
PIVs <- names(PIVs_config)
compute_RL_FDP_score( prep_data$encodedA, prep_data$encodedB, PIVs, 1, 2,
                      "BRL", flds = PIVs, types = rep("bi",length(PIVs)) )
compute_RL_FDP_score( prep_data$encodedA, prep_data$encodedB, PIVs, 1, 2,
                      "FlexRL", data = prep_data, StEM_iter = 5,
                      StEM_burnin = 2, gibbs_iter = 5, gibbs_burnin = 2,
                      n_post_sample = 5 )
```

`create_data_alpha` *Book-keeping data frame for parameters of PIVs dynamics*

Description

Internal helper used by `StEM()` to accumulate, across Gibbs iterations, the covariates, true-value agreement indicator, and time gaps needed to re-estimate the survival (hazard) parameters of a dynamic structured PIV.

Usage

```
create_data_alpha(n_coef_unstable, stable)
```

Arguments

`n_coef_unstable`	Integer, number of hazard coefficients for this PIV (1 for the baseline hazard, plus one per covariate from file A and file B).
`stable`	Logical, whether this PIV is stable (`dynamics != "structured"`).

Value

An empty data frame with `n_coef_unstable + 2` columns (the covariates/intercept, plus `Hequal` and `times`) if `stable` is `FALSE`; `NULL` if the PIV is stable (nothing to accumulate).

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                      cov2=c())) )
PIVs_stable <- sapply(PIVs_config, function(x) x$dynamics != "structured")
n_coef_unstable = c(0,0,0,3)
Valpha <- mapply(create_data_alpha, n_coef_unstable = n_coef_unstable,
                 stable = PIVs_stable, SIMPLIFY = FALSE)
```

`FDP_score` *Model specific, linkage score based, false discovery proportion at a given threshold*

Description

Model specific, linkage score based, false discovery proportion at a given threshold

Usage

```
FDP_score(LinkScore, threshold)
```

Arguments

`LinkScore`	Numeric vector, linkage scores of the candidate pairs.
`threshold`	Numeric, score threshold above which a pair is declared linked.

Value

A list with `FDP_score` Numeric, `1 - mean(score | score > threshold)`, the estimated FDP at `threshold` and `n_linked` Integer, number of linked records.

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                      cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
                   V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                   V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list( V1 = c(), V2 = c(),
                            V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params, TRUE )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
fit <- StEM( data = prep_data, StEM_iter = 10, StEM_burnin = 5,
             gibbs_iter = 10, gibbs_burnin = 5, n_post_sample = 10 )
FDP_score(fit$Delta$x, 0.5)
FDP_score(fit$Delta$x, 0.75)
```

`FDP_synth` *Model agnostic, based on synthetic data, false discovery proportion at a given threshold*

Description

Model agnostic, based on synthetic data, false discovery proportion at a given threshold

Usage

```
FDP_synth(idxA, idxB, LinkScore, threshold, n_records_A, n_records_B, n_synth)
```

Arguments

`idxA`	Integer vector, indices in A of linked records (for a previously set threshold).
`idxB`	Integer vector, indices in B of linked records (for a previously set threshold).
`LinkScore`	Numeric vector, linkage scores of the candidate pairs. If `NULL`, consider all given pairs as linked.

threshold	Numeric, score threshold above which a pair is declared linked.
n_records_A	Integer, number of records in A.
n_records_B	Integer, number of records in B.
n_synth	Integer, number of synthetic records to generate per iteration.

Value

A list with FDP_synth Numeric, proportion of synthetic falsely linked records, n_linked_real Integer, number of real linked records and n_linked_all Integer, total number of linked records (real and synthetic).

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                      cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
                   V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                   V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list( V1 = c(), V2 = c(),
                            V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params, TRUE )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
PIVs <- names(PIVs_config)
n_synth <-as.integer(0.10 * nrow(prep_data$encodedB))
new_data <- synthesise("arf", prep_data$encodedA[, c(PIVs,"local_id","source",
                      "date",prep_data$PIVs_config$V4$cond_hazard_cov$covA)],
                      prep_data$encodedB[, c(PIVs,"local_id","source","date",
                      prep_data$PIVs_config$V4$cond_hazard_cov$covB)], PIVs,
                      n_synth, TRUE)
new_data$dataA[PIVs][is.na(new_data$dataA[PIVs])] <- 0
new_data$dataB[PIVs][is.na(new_data$dataB[PIVs])] <- 0
# cannot model dynamics for synthetic data, set dates to 0
new_data$dataA$date[is.na(new_data$dataA$date)] <- 0
new_data$dataB$date[is.na(new_data$dataB$date)] <- 0
arguments <- list(data = prep_data, StEM_iter = 5, StEM_burnin = 2,
```

```
                  gibbs_iter = 5, gibbs_burnin = 2, n_post_sample = 5)
fit_flexrl <- link_with_FlexRL(new_data$dataA, new_data$dataB, arguments)
FDP_synth(fit_flexrl$idxA, fit_flexrl$idxB, fit_flexrl$LinkScore, 0.5,
          nrow(prep_data$encodedA), nrow(prep_data$encodedB),
          n_synth)
```

FlexRL *FlexRL: A Flexible Model for Record Linkage*

Description

Links records that refer to the same entities across two data sources without a unique identifier, using partially identifying variables. These are stable (not changing over time, probability of mistakes could be bounded), flexible (dynamic but no information to model changes over time) or structured (dynamic and information to model changes over time, probability of mistakes could be fixed). The main function `StEM()` fits a latent-variable model by stochastic expectation-maximisation; it models registration errors (missing values and mistakes) and changes over time. `prepare_data()` prepares the data sources for record linkage, `RL_diagnostics()` gathers diagnostics (FDP estimation and discrepancy metrics for inference on the linked data.

Details

Methodological paper: doi:10.1093/jrsssc/qlaf016. False discovery proportion estimation: doi:10.1002/sim.70292. Experiments repository: `https://github.com/robachowyk/FlexRL-experiments`.

Author(s)

Kayané Robach

See Also

Useful links:

- `https://github.com/robachowyk/FlexRL`
- `https://robachowyk.github.io/FlexRL/`
- Report bugs at `https://github.com/robachowyk/FlexRL/issues`

Examples

```
# Link two simulated sources with 4 PIVs: two stable, one flexible, one
# structured. The true links are known, so performance can be computed.
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
```

```
                              bound_mistakes = c(NA,NA),
                              fix_mistakes = c(NA,NA)),
                        V4 = list(dynamics = "structured",
                              bound_mistakes = c(NA,NA),
                              fix_mistakes = c(0.03,0.03),
                              cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                     cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
                   V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                   V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list(V1 = c(), V2 = c(),
                           V3 = c(), V4 = log(c(0.7, 0.6, 0.5)))
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params,
                           TRUE, survival_model("exponential") )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
fit <- StEM( data = prep_data, StEM_iter = 5, StEM_burnin = 2,
             gibbs_iter = 5, gibbs_burnin = 2, n_post_sample = 5 )

# linked pairs and performance against the true pairs
linked <- fit$Delta[fit$Delta$x > 0.5, ]
linked_pairs <- paste(linked$i, linked$j, sep = "_")
true_pairs   <- paste(prep_data$true_pairs[[1]], prep_data$true_pairs[[2]], sep = "_")
tp <- length(intersect(linked_pairs, true_pairs))
fp <- length(setdiff(linked_pairs, true_pairs))
fn <- length(setdiff(true_pairs, linked_pairs))
c(LinkageDecisionRule = 0.5, FDP = fp / (tp + fp), Sensitivity = tp / (tp + fn))

# diagnostics for inference on the linked data
diag <- RL_diagnostics(fit, prep_data$encodedA, prep_data$encodedB,
                       names(PIVs_config),
                       list(V1 = FALSE, V2 = FALSE, V3 = FALSE, V4 = TRUE),
                       names(PIVs_config),
                       true_pairs = prep_data$true_pairs, FDP_estimation = TRUE,
                       RL_method = "FlexRL", data = prep_data,
                       StEM_iter = 5, StEM_burnin = 2,
                       gibbs_iter = 5, gibbs_burnin = 2, n_post_sample = 5,
                       maxIter4CV = 1, n_repeats = 1)
diag # print(diag)
print(diag, threshold = 0.75)
plot(diag, "scores")
plot(diag, "distributions", threshold = 0.75)
plot(diag, "convergence")
plot(diag, "FDP")
plot(diag, "discrepancy")
```

`link_with_BRL` *Run BRL through the uniform wrapper interface*

Description

Links dataA and dataB with the package and returns the linked pairs in the common format used by compute_RL_FDP_score() and compute_augmRL_FDP_synth(). dataA and dataB can be the outputs of synthesise() or the original encoded data sources.

Usage

```
link_with_BRL(dataA, dataB, arguments, ...)
```

Arguments

dataA	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
dataB	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
arguments	List of extra arguments forwarded to BRL::compareRecords() and BRL::bipartiteGibbs() (e.g. flds, types, nIter).
...	Ignored; kept for compatibility with RL_diagnostics().

Details

More details about BRL on https://cran.r-project.org/web/packages/BRL/index.html

Value

Named list with idxA, idxB and LinkScore (row indices in dataA, dataB and linkage scores of potential linked records).

---

link_with_diyar *Run diyar through the uniform wrapper interface*

Description

Links dataA and dataB with the package and returns the linked pairs in the common format used by compute_RL_FDP_score() and compute_augmRL_FDP_synth(). dataA and dataB can be the outputs of synthesise() or the original encoded data sources.

Usage

```
link_with_diyar(dataA, dataB, arguments, ...)
```

Arguments

dataA	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
dataB	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
arguments	List of extra arguments forwarded to diyar::prob_score_range() and diyar::links_wf_probabilis (e.g. attribute, probabilistic, return_weights).
...	Ignored; kept for compatibility with RL_diagnostics().

Details

More details about diyar on https://cran.r-project.org/web/packages/diyar/index.html

Value

Named list with idxA, idxB and LinkScore = NULL (row indices of default potential linked records in dataA, dataB, this package does not return scores).

---

link_with_fastLink *Run fastLink through the uniform wrapper interface*

Description

Links dataA and dataB with the package and returns the linked pairs in the common format used by compute_RL_FDP_score() and compute_augmRL_FDP_synth(). dataA and dataB can be the outputs of synthesise() or the original encoded data sources.

Usage

```
link_with_fastLink(dataA, dataB, arguments, ...)
```

Arguments

dataA	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
dataB	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
arguments	List of extra arguments forwarded to fastLink::fastLink() (e.g. varnames, threshold.match, tol.em, return.all).
...	Ignored; kept for compatibility with RL_diagnostics().

Details

More details about fastLink on https://cran.r-project.org/web/packages/fastLink/index.html

Value

Named list with idxA, idxB and LinkScore (row indices in dataA, dataB and linkage scores of potential linked records).

`link_with_fedmatch` *Run fedmatch through the uniform wrapper interface*

Description

Links dataA and dataB with the package and returns the linked pairs in the common format used by compute_RL_FDP_score() and compute_augmRL_FDP_synth(). dataA and dataB can be the outputs of synthesise() or the original encoded data sources.

Usage

```
link_with_fedmatch(dataA, dataB, arguments, ...)
```

Arguments

dataA	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
dataB	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
arguments	List of extra arguments forwarded to fedmatch::merge_plus() (e.g. by, match_type, unique_key_1, unique_key_2, multivar_settings).
...	Ignored; kept for compatibility with RL_diagnostics().

Details

More details about fedmatch on https://cran.r-project.org/web/packages/fedmatch/index.html

Value

Named list with idxA, idxB and LinkScore (row indices in dataA, dataB and linkage scores of potential linked records).

link_with_FlexRL *Run FlexRL through the uniform wrapper interface*

Description

Links dataA and dataB with the package and returns the linked pairs in the common format used by compute_RL_FDP_score() and compute_augmRL_FDP_synth(). dataA and dataB can be the outputs of synthesise() or the original encoded data sources.

Usage

```
link_with_FlexRL(dataA, dataB, arguments, ...)
```

Arguments

dataA	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
dataB	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
arguments	List of extra arguments forwarded to StEM() (e.g. data, StEM_iter, StEM_burnin, gibbs_iter, gibbs_burnin).
...	Ignored; kept for compatibility with RL_diagnostics().

Details

More details about FlexRL on https://cran.r-project.org/web/packages/FlexRL/index.html

Value

Named list with idxA, idxB and LinkScore (row indices in dataA, dataB and linkage scores of potential linked records).

link_with_multilink *Run multilink through the uniform wrapper interface*

Description

Links dataA and dataB with the package and returns the linked pairs in the common format used by compute_RL_FDP_score() and compute_augmRL_FDP_synth(). dataA and dataB can be the outputs of synthesise() or the original encoded data sources.

Usage

```
link_with_multilink(dataA, dataB, arguments, ...)
```

Arguments

dataA	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
dataB	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
arguments	List of extra arguments forwarded across the multilink pipeline (create_comparison_data(), reduce_comparison_data(), specify_prior(), gibbs_sampler(), find_bayes_estimate(), relabel_bayes_estimate()): e.g. records, types, breaks, duplicates, n_iter.
...	Ignored; kept for compatibility with RL_diagnostics().

Details

More details about multilink on https://cran.r-project.org/web/packages/multilink/index.html

Value

Named list with idxA, idxB and LinkScore = NULL (row indices of default potential linked records in dataA, dataB, this package does not return scores).

---

link_with_reclin2 *Run reclin2 through the uniform wrapper interface*

Description

Links dataA and dataB with the package and returns the linked pairs in the common format used by compute_RL_FDP_score() and compute_augmRL_FDP_synth(). dataA and dataB can be the outputs of synthesise() or the original encoded data sources.

Usage

```
link_with_reclin2(dataA, dataB, arguments, ...)
```

Arguments

dataA	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
dataB	Data source to link: the encoded source from prepare_data(), or the augmented sources from synthesise().
arguments	List of extra arguments forwarded across the reclin2 pipeline (pair(), compare_pairs(), problink_em(), predict(), select_threshold()): e.g. on, formula, type, add, variable, score, threshold.
...	Ignored; kept for compatibility with RL_diagnostics().

Details

More details about reclin2 on https://cran.r-project.org/web/packages/reclin2/index.html

Value

Named list with `idxA`, `idxB` and `LinkScore` (row indices in `dataA`, `dataB` and linkage scores of potential linked records for the `select_threshold()` argument).

---

log_lik *Log-likelihood of the linkage matrix*

Description

Log-likelihood of the linkage matrix

Usage

```
log_lik(LLL, LLA, LLB, links, sumRowD, sumColD, gamma)
```

Arguments

`LLL`	(Sparse) matrix of log-likelihood contributions for linked records.
`LLA`	Numeric vector of log-likelihood contributions for non-linked records from A.
`LLB`	Numeric vector of log-likelihood contributions for non-linked records from B.
`links`	2-column matrix of indices (A, B) for the currently linked records.
`sumRowD`	Logical vector, one entry per record in A: does it form a link?
`sumColD`	Logical vector, one entry per record in B: does it form a link?
`gamma`	Numeric, proportion of linked records as a fraction of the smaller file.

Value

Numeric, the log-likelihood of the linkage matrix.

Examples

```
LLL <- Matrix::Matrix(0, nrow = 13, ncol = 15, sparse = TRUE)
LLA <- stats::runif(13, 0, 2)
LLB <- stats::runif(15, 0, 2)
links <- as.matrix(data.frame(idxA = c(5, 9, 11, 12, 13),
                              idxB = c(5, 9, 11, 13, 15)))
LLL[links] <- 0.67
sumRowD <- (seq_len(13) %in% links[, 1])
sumColD <- (seq_len(15) %in% links[, 2])
gamma <- 0.5
log_lik(LLL, LLA, LLB, links, sumRowD, sumColD, gamma)
```

log_possible_config *Log number of possible linkage configurations*

Description

Computes `log(nB! / (nB - sumD)!)`, i.e. the log number of ways to choose `sumD` ordered links among `n_records_B` records of the larger file; used in `log_lik()` to normalise the likelihood of the linkage matrix.

Usage

```
log_possible_config(n_records_B, sumD)
```

Arguments

`n_records_B`	Integer, number of records in the larger data source (B).
`sumD`	Integer, number of currently linked records.

Value

Numeric, `sum(log((n_records_B - sumD + 1):n_records_B))`, or `0` if `sumD == 0`.

Examples

```
log_possible_config(n_records_B = 15, sumD = 5)
```

mmd *Maximum Mean Discrepancy between two sets of variables*

Description

Joint (multivariate) measure of distributional discrepancy between `x` and `y`, using a Gaussian (RBF) kernel with bandwidth set by the median pairwise distance (biased estimator, Gretton et al. 2012).

Usage

```
mmd(x, y)
```

Arguments

`x, y`	Numeric matrices with the same number of columns.

Value

Numeric, the (biased) MMD estimate.

Examples

```
x <- matrix(rnorm(50), ncol = 5)
y <- matrix(rnorm(30) + 0.5, ncol = 5)
mmd(x, y)
```

`naive_linkage` *Naive (deterministic exact-match) record linkage*

Description

Links records that agree exactly on every non-missing PIV. Does not enforce the one-to-one assignment constraint, so should be used only to gauge the difficulty of the linkage task (amount of duplication, and discriminative power of the PIVs together).

Usage

```
naive_linkage(PIVs, encodedA, encodedB, na_match = TRUE, na_is_zero = TRUE)
```

Arguments

`PIVs`	Character vector, names of the PIVs (columns present in both files).
`encodedA`	The data source (PIVs encoded to natural numbers).
`encodedB`	The data source (PIVs encoded to natural numbers).
`na_match`	Logical; if `TRUE`, a missing PIV value is treated as matching any value in the other file (default `TRUE`).
`na_is_zero`	Logical; if `TRUE`, missing values are already coded as `0` (FlexRL's convention); if `FALSE`, `NA` will be recoded to `0` (default `TRUE`).

Value

Data frame with columns `idxA`, `idxB`: pairs of record indices that match.

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
```

```
                                                        cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
                   V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                   V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list( V1 = c(), V2 = c(),
                            V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params, TRUE )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
naive_linkage( names(prep_data$PIVs_config),
               prep_data$encodedA, prep_data$encodedB )
```

`plot.discrepancy_curves` *Plot post-linkage diagnostics over thresholds*

Description

Plot post-linkage diagnostics over thresholds

Usage

```
## S3 method for class 'discrepancy_curves'
plot(x, ...)
```

Arguments

`x`	A `discrepancy_curves` object.
`...`	Ignored.

Value

`x`, invisibly; called for its plotting side effect.

`plot.FDP_curves` *Plot FDP estimates against the linkage decision threshold*

Description

Plot FDP estimates against the linkage decision threshold

Usage

```
## S3 method for class 'FDP_curves'
plot(x, ...)
```

Arguments

x	An object of class "FDP_curves".
...	Ignored.

Value

x, invisibly; called for its plotting side effect.

---

plot.RL_diagnostics *Plot post-linkage diagnostics*

Description

Plot post-linkage diagnostics

Usage

```
## S3 method for class 'RL_diagnostics'
plot(x, type, threshold = NULL, ...)
```

Arguments

x	An RL_diagnostics object, see RL_diagnostics().
type	One of "scores" (linkage score histogram), "distributions" (linked subset vs. data sources, per variable, at threshold), "convergence" (StEM trace plots, see plot_StEM_convergence()), "FDP" (FDP estimates by threshold, only if FDP_estimation = TRUE was used) or "discrepancy" (MMD, SMD, IoU and agreement by threshold, see plot.discrepancy_curves()).
threshold	Numeric, the linkage decision rule defining the linked subset for type = "distributions" (ignored for methods without scores).
...	Passed on to the underlying plotting helper.

Value

x, invisibly; called for its plotting side effect.

plot_distributions *Compare distributions of shared variables across data sets (histograms/barplots)*

Description

Overlays, for each variable in `common_vars`, the empirical distribution in every data set of `data_list` (e.g. a baseline file vs. the linked set).

Usage

```
plot_distributions(data_list, common_vars, threshold, colours = NULL)
```

Arguments

`data_list`	Named list of data frames, each containing `common_vars`.
`common_vars`	Character vector, variables to compare.
`threshold`	Numeric, the linkage decision rule that defined the linked data; only used to label the linked data in the legend.
`colours`	Optional vector of colours, one per element of `data_list`.

Value

`NULL`, invisibly; called for its plotting side effect.

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                      cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
                   V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                   V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list( V1 = c(), V2 = c(),
                            V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
```

```
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params, TRUE )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
fit <- StEM( data = prep_data, StEM_iter = 10, StEM_burnin = 5,
             gibbs_iter = 10, gibbs_burnin = 5, n_post_sample = 10 )
threshold_strict <- stats::quantile(fit$Delta$x, 0.75)
data_list = list(data_baseline = prep_data$encodedA,
data_select = prep_data$encodedA[fit$Delta[fit$Delta$x > threshold_strict, "i"],])
common_vars = names(PIVs_config)
plot_distributions(data_list, common_vars)
common_vars = c("Xe", "Xf")
plot_distributions(data_list, common_vars)
```

`plot_linkage_scores` *Plot the distribution of linkage scores*

Description

Plot the distribution of linkage scores

Usage

```
plot_linkage_scores(n_pairs, LinkScore)
```

Arguments

`n_pairs`	Integer, total number of candidate pairs considered (`nrow(A) * nrow(B)`).
`LinkScore`	Numeric vector, linkage scores of the pairs above 0 (e.g. `Delta$x`).

Value

`NULL`, invisibly; called for its plotting side effect.

Examples

```
fit_Delta_x <- c(0,0,0,0,0,0,0,0,0,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,
0.2,0.2,0.2,0.2,0.4,0.4,0.4,0.4,0.4,0.5,0.6,0.7,0.7,0.7,0.7,0.7,0.7,0.8,0.8)
plot_linkage_scores(1000, fit_Delta_x)
```

plot_StEM_convergence *Monte Carlo convergence (trace) plots for a fitted StEM model*

Description

Trace-plots the raw StEM chains of gamma, eta, alpha and phi across iterations, to visually judge whether the chains have stabilised and pick an adequate StEM_burnin for StEM(). One plot per parameter: gamma (proportion linked), eta (PIVs distribution), alpha (hazard coefficients, for dynamic PIVs only), and phi (agreement/missing rates).

Usage

```
plot_StEM_convergence(fit)
```

Arguments

fit	List as returned by StEM(), containing the raw chains gamma, eta, alpha, phi (StEM_iter rows each).

Value

NULL, invisibly; called for its plotting side effect.

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                      cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
                   V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                   V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list( V1 = c(), V2 = c(),
                            V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params, TRUE )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
```

```
                            PIVs_config, TRUE, "entity_id", TRUE )
fit <- StEM( data = prep_data, StEM_iter = 10, StEM_burnin = 3,
             gibbs_iter = 10, gibbs_burnin = 3, n_post_sample = 10 )
plot_StEM_convergence(fit)
```

`prepare_data` *Prepare two data sources for* `StEM()`

Description

Wraps the data-preparation steps needed before calling `StEM()`: tags each source with a `source` column, labels the larger source as `B`, adjusting `cond_hazard_cov`/`bound_mistakes`/`fix_mistakes` accordingly if A and B are swapped), drops records whose PIV values fall outside the support shared by both files, warns if two PIVs are strongly associated (Cramer's V > 0.3), which may degrade record linkage performance, encodes every PIV to natural numbers using levels pooled across both sources, and encodes missing values to `0`.

Usage

```
prepare_data(
  data1,
  data2,
  label1,
  label2,
  PIVs_config,
  same_mistakes = TRUE,
  uniq_id = NULL,
  restrict_support_intersection = TRUE
)
```

Arguments

`data1`	Data frame, the raw data source (whichever has more rows becomes `B`).
`data2`	Data frame, the raw data source (whichever has more rows becomes `B`).
`label1`	Character, label recorded in the `source` column for `data1`.
`label2`	Character, label recorded in the `source` column for `data2`.
`PIVs_config`	Named list describing each PIV — see `simulate_data()`.
`same_mistakes`	Logical, will A and B share one mistake-probability parameter per PIV.
`uniq_id`	Optional column name (present in both files) with the true entity identifier, used to build `true_pairs` for evaluation; `NULL` if unavailable.
`restrict_support_intersection`	Logical; if `TRUE` (default), records with an out-of-common-support PIV value are dropped (and a warning issued); if `FALSE`, only the warning is issued.

Value

A list ready to use as the data argument of StEM(): encodedA, encodedB, n_values, PIVs_config, same_mistakes, and true_pairs (NULL if uniq_id was not supplied).

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                      cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
                   V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                   V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list( V1 = c(), V2 = c(),
                            V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params, TRUE )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
str(prep_data, max.level = 1)
```

---

print.RL_diagnostics *Print post-linkage diagnostics*

Description

Print post-linkage diagnostics

Usage

```
## S3 method for class 'RL_diagnostics'
print(x, threshold = 0.5, ...)
```

Arguments

x	An RL_diagnostics object, see RL_diagnostics().
threshold	Numeric, the linkage decision rule at which the measures are displayed (default 0.5).
...	Ignored.

Value

x, invisibly; called for its printing side effect.

---

prop_level *Proportion of a data set with a given variable at a given level*

Description

Proportion of a data set with a given variable at a given level

Usage

```
prop_level(df, var, level)
```

Arguments

df	Data frame.
var	Character, column name in df.
level	Value to match against df[[var]].

Value

Numeric, proportion of rows where df[[var]] == level (na.rm=TRUE).

Examples

```
df <- data.frame(colour = sample(c("orange", "purple"), 100, replace = TRUE))
prop_level(df, "colour", "purple")
```

RL_agreement *Agreement rate of pairs on shared variables*

Description

For a set of pairs, computes how often the two records agree on each variable in common_vars.

Usage

```
RL_agreement(data1, data2, common_vars, pairs, na.rm = TRUE, na_match = NULL)
```

Arguments

data1	Data frame containing common_vars.
data2	Data frame containing common_vars.
common_vars	Character vector, names of the variables to compare (must exist in both data1 and data2).
pairs	Data frame/matrix/list with 2 columns of indices (into data1, data2) for the pairs to evaluate.
na.rm	Logical; if TRUE (default), pairs with a missing value on a variable are excluded from that variable's agreement rate.
na_match	Logical, required if na.rm = FALSE: should a missing value be treated as agreeing (TRUE) or disagreeing (FALSE) with any value?

Value

List with agreements (named numeric vector, one entry per variable in common_vars) and, if available, true_agreements.

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                      cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
```

```
                    V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                    V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list( V1 = c(), V2 = c(),
                            V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params, TRUE )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
fit <- StEM( data = prep_data, StEM_iter = 10, StEM_burnin = 5,
             gibbs_iter = 10, gibbs_burnin = 5, n_post_sample = 10 )
linked_pairs <- fit$Delta[fit$Delta$x > 0.5, ]
RL_agreement( prep_data$encodedA, prep_data$encodedB,
              names(PIVs_config), linked_pairs )
RL_agreement( prep_data$encodedA, prep_data$encodedB,
              names(PIVs_config), prep_data$true_pairs )
```

`RL_diagnostics` *Post-linkage diagnostics*

Description

Gathers, from a fitted linkage, the diagnostics needed to judge the linked data before using it for inference: false discovery proportion estimate, agreement among PIVs of linked pairs, comparison of the linked subset with each source file (standardised mean differences, support overlap, maximum mean discrepancy).

Usage

```
RL_diagnostics(
  fit,
  encodedA,
  encodedB,
  compare_vars,
  vars_type_cont,
  PIVs,
  true_pairs = NULL,
  FDP_estimation = TRUE,
  ...
)
```

Arguments

`fit`	List with either `Delta` (a data frame with columns `i`, `j`, `x`, as returned by `StEM()`) or the three elements `idxA`, `idxB`, `LinkScore` (e.g. the output of a link_with_* wrapper).
`encodedA`	Data source used for linkage (same encoding and row order as passed to the linkage method).

encodedB	Data source used for linkage (same encoding and row order as passed to the linkage method).
compare_vars	Character vector, variables (PIVs or others) to compare between the linked subset and each source file.
vars_type_cont	Named list or logical vector, one entry per compare_vars: TRUE if the variable is treated as continuous, FALSE if categorical (one SMD per level).
PIVs	Character vector, partially identifying variables to compare between the linked records from A and from B.
true_pairs	Optional data frame with 2 columns of true (A, B) indices, when known, to report the realised FDP and sensitivity.
FDP_estimation	Logical; if TRUE, run compute_RL_FDP_score() and compute_augmRL_FDP_synth() to estimate the FDP over thresholds 0.50 to 0.95. RL_method and other arguments must then be given in ....
...	Arguments passed to compute_RL_FDP_score() and compute_augmRL_FDP_synth() when FDP_estimation = TRUE: RL_method, n_repeats, maxIter4CV, optionally synth_method (default "arf"), n_synth, PIVs (default compare_vars), and the arguments of the chosen link_with_* wrapper.

Details

All measures are computed for every threshold from 0.50 to 0.99 (by 0.01) on the linkage scores; for a method without scores (or with a single score value) they are computed once, for all returned pairs. print(x, threshold =) summarises diagnostics at one threshold, plot() shows the curves.

Value

An object of class "RL_diagnostics", a list with:

discrepancy_measures	data frame (class "discrepancy_curves"), one row per threshold xi: number of linked pairs, agreement rate per variable (see RL_agreement()), standardised mean differences (one per continuous variable or per level, see smd()), support overlap per variable (see support_iou()) and multivariate discrepancy (see mmd()), each for the linked subset of A vs. A and of B vs. B
FDP_measures	FDP estimates by threshold (class "FDP_curves"; the curve data frame holds the average over the n_repeats iterations and, in the *_sd columns, their standard deviation, NA when fewer than two valid repeats) if FDP_estimation = TRUE, else NULL
true_agreement	only if true_pairs is supplied: agreement rate of the true pairs on each of compare_vars, to compare with the agreement of the linked pairs
true_performance	only if true_pairs is supplied: data frame with the realised FDP and sensitivity by threshold
idxA, idxB, LinkScore, RL_method, n_pairs, compare_vars, A, B	the linkage and data needed by the print() and plot() methods
gamma, eta, alpha, phi	the StEM chains from fit, if present

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                      cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
                   V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                   V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list( V1 = c(), V2 = c(),
                            V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params, TRUE )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
PIVs <- names(PIVs_config)
PIVs_type <- list(V1=FALSE, V2=FALSE, V3=FALSE, V4=TRUE)
fit_brl <- link_with_BRL( prep_data$encodedA, prep_data$encodedB,
                          list( flds = PIVs,
                                types = rep("bi",length(PIVs)),
                                nIter = 500 ) )
diag_brl <- RL_diagnostics(fit_brl, prep_data$encodedA, prep_data$encodedB,
                           PIVs, PIVs_type, PIVs, true_pairs = prep_data$true_pairs,
                           FDP_estimation = TRUE, RL_method = "BRL",
                           flds = PIVs, types = rep("bi",length(PIVs)), nIter = 500,
                           maxIter4CV = 1, n_repeats = 1)
diag_brl # print(diag_brl)
print(diag_brl, threshold = 0.75)
plot(diag_brl, "scores")
plot(diag_brl, "distributions", threshold = 0.75)
plot(diag_brl, "FDP")
plot(diag_brl, "discrepancy")
```

`simulate_data` *Simulate two linked data sources for record linkage benchmarking*

Description

Creates two synthetic data sources of given sizes sharing a given number of common entities ("links"), each described by a set of Partially Identifying Variables (PIVs). For every PIV, choose the

number of possible values, the proportion of mistakes and of missing values, and whether it is stable over time, flexible (may change but change is not modelled), or structured (expected to change over time, with a survival model for the hazard of change). For structured PIVs, `enforce_estimability` forces half of the linked pairs to have a near-zero time gap, which helps separate "mistake" from "change over time" when fitting the model.

Usage

```
simulate_data(
  PIVs_config,
  n_values,
  n_records,
  n_links,
  p_mistake,
  p_missing,
  cond_hazard_params,
  enforce_estimability,
  model_dynamics = survival_model("exponential")
)
```

Arguments

`PIVs_config`	Named list, one entry per PIV, each a list with: `dynamics` (`"stable"`, `"flexible"`, or `"structured"`), `bound_mistakes` (length-2 numeric/NA, upper bound on the mistake probability in file 1 / file 2), `fix_mistakes` (length-2 numeric/NA, mistake probability fixed to this value in file 1 / file 2), and, only for `dynamics = "structured"`, `cond_hazard_cov` (a list with `cov1` and `cov2`, the names of covariates in file 1 / file 2 used to model the hazard of change).
`n_values`	Integer vector, number of unique values per PIV (same order as `PIVs_config`).
`n_records`	Integer vector of length 2, number of records to generate in file 1 and file 2 (file 2 must be the larger of the two).
`n_links`	Integer, number of records shared between the two files.
`p_mistake`	Named list (one entry per PIV) of length-2 numeric vectors, proportion of mistakes to introduce in file 1 / file 2.
`p_missing`	Named list (one entry per PIV) of length-2 numeric vectors, proportion of missing values to introduce in file 1 / file 2.
`cond_hazard_params`	Named list (one entry per PIV) of numeric vectors for the survival model generating the changes (see `survival_model()`); only used for `dynamics = "structured"` PIVs.
`enforce_estimability`	Logical; if `TRUE`, half of the linked pairs are given a near-zero time gap to help estimate the instability parameters.
`model_dynamics`	Object from `survival_model()` used to generate the changes of the structured PIVs (default exponential).

Value

A list with: `data1`, `data2` the two simulated (encoded) data frames, `n_values` number of unique values per PIV, `time_difference` time gap between linked records (`NA` if no PIV is structured), `proba_same_H` matrix (n links x n PIVs) of probabilities that the true values coincide, `true_pairs` data frame with the true 1/2 indices of the linked records

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                      cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
                   V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                   V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list( V1 = c(), V2 = c(),
                            V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params, TRUE )
str(gen_data, max.level = 1)
```

`simulateD` *Simulate the linkage matrix D*

Description

Given the current draw of true PIV values, samples the linkage matrix D from its conditional distribution, and returns the updated log-likelihood.

Usage

```
simulateD(
  data,
  linksR,
  truepivsA,
  truepivsB,
  gamma,
```

```
  eta,
  alpha,
  phi,
  model_dynamics = survival_model("exponential"),
  threads = 4
)
```

Arguments

data	List with encodedA, encodedB (the two encoded data sources, missing values as 0), n_values, PIVs_config and same_mistakes; see StEM().
linksR	2-column matrix (1-indexed) of the currently linked (A, B) indices.
truepivsA	Matrix of true PIV values, as returned by simulateH().
truepivsB	Matrix of true PIV values, as returned by simulateH().
gamma	Numeric, proportion of linked records as a fraction of the smaller file.
eta	List (one per PIV) of the distribution of true values.
alpha	List (one per PIV) of survival-model parameters (see survival_model()).
phi	List (one per PIV) of registration-error parameters.
model_dynamics	Object from survival_model() giving the survival function of the structured PIVs.
threads	Integer, number of OpenMP threads for the C++ per-pair likelihood loops (default 4; identical results for any value; ignored if the package was built without OpenMP).

Value

A list (Dsample, Rcpp sampleD() output) with: links updated set of links, sumRowD updated sumRowD, sumColD updated sumColD, loglik updated value of the complete log likelihood, nlinkrec updated number of linked records

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                      cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
```

```
                  V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                  V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list( V1 = c(), V2 = c(),
                            V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params, TRUE )
data_StEM <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
PIVs_stable <- sapply(data_StEM$PIVs_config, function(x)
                        x$dynamics != "structured")
linksR = base::matrix(0,0,2)
linksCpp = linksR
sumRowD = rep(0, nrow(data_StEM$encodedA))
sumColD = rep(0, nrow(data_StEM$encodedB))
nlinkrec = 0
survivalpSameH = base::matrix(1, nrow(linksR), length(data_StEM$n_values))
gamma = 0.5
eta = lapply(data_StEM$n_values, function(x) rep(1/x,x))
phi = lapply(data_StEM$n_values, function(x)  c(0.9,0.9,0.1,0.1))
n_coef_unstable = lapply( seq_along(PIVs_stable), function(idx)
 if(PIVs_stable[idx]){ 0 }else{
   ncol(data_StEM$encodedA[, data_StEM$PIVs_config[[idx]]$cond_hazard_cov$covA,
                           drop=FALSE]) +
   ncol(data_StEM$encodedB[, data_StEM$PIVs_config[[idx]]$cond_hazard_cov$covB,
                           drop=FALSE]) + 1 } )
alpha = lapply( seq_along(PIVs_stable),
                function(idx) if(PIVs_stable[idx]){ c(-Inf) }
                else{ rep(log(0.05), n_coef_unstable[[idx]]) }
              )
newTruePivs = simulateH(data=data_StEM, links=linksCpp,
                        survivalpSameH=survivalpSameH,
                        sumRowD=sumRowD, sumColD=sumColD, eta=eta, phi=phi)
truepivsA = newTruePivs$truepivsA
truepivsB = newTruePivs$truepivsB
Dsample = simulateD(data=data_StEM, linksR=linksR,
                    truepivsA=truepivsA, truepivsB=truepivsB,
                    gamma=gamma, eta=eta, alpha=alpha, phi=phi)
linksCpp = Dsample$links
linksR = linksCpp + 1
```

`simulateH` *Simulate the true PIV values underlying the registered records*

Description

Draw the latent true values of each PIV given the currently registered (possibly mistaken or missing) values, the current linkage status, and the current parameters.

Usage

```
simulateH(data, links, survivalpSameH, sumRowD, sumColD, eta, phi)
```

Arguments

data	List with encodedA, encodedB (the two encoded data sources, missing values as 0), n_values, PIVs_config and same_mistakes; see StEM().
links	2-column matrix of (A, B) indices for the currently linked records.
survivalpSameH	Matrix (n links x n PIVs); 1 for stable PIVs, and the survival probability that the true value is unchanged for unstable PIVs.
sumRowD	Logical vector, one entry per record in A: does it form a link?
sumColD	Logical vector, one entry per record in B: does it form a link?
eta	List (one per PIV) of the distribution of true values.
phi	List (one per PIV) of length-4 vectors: agreement prob. in A, agreement prob. in B, missing prob. in A, missing prob. in B.

Value

List with truepivsA and truepivsB, matrices (same shape as the input data) of simulated true PIV values.

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                      cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
                   V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                   V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )
cond_hazard_params <- list( V1 = c(), V2 = c(),
                            V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params, TRUE )
data_StEM <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
PIVs_stable <- sapply(data_StEM$PIVs_config, function(x)
```

```
                        x$dynamics != "structured")
linksR = base::matrix(0,0,2)
linksCpp = linksR
sumRowD = rep(0, nrow(data_StEM$encodedA))
sumColD = rep(0, nrow(data_StEM$encodedB))
nlinkrec = 0
survivalpSameH = base::matrix(1, nrow(linksR), length(data_StEM$n_values))
gamma = 0.5
eta = lapply(data_StEM$n_values, function(x) rep(1/x,x))
phi = lapply(data_StEM$n_values, function(x)  c(0.9,0.9,0.1,0.1))
n_coef_unstable = lapply( seq_along(PIVs_stable), function(idx)
 if(PIVs_stable[idx]){ 0 }else{
   ncol(data_StEM$encodedA[, data_StEM$PIVs_config[[idx]]$cond_hazard_cov$covA,
                           drop=FALSE]) +
   ncol(data_StEM$encodedB[, data_StEM$PIVs_config[[idx]]$cond_hazard_cov$covB,
                           drop=FALSE]) + 1 } )
alpha = lapply( seq_along(PIVs_stable),
                function(idx) if(PIVs_stable[idx]){ c(-Inf) }
                else{ rep(log(0.05), n_coef_unstable[[idx]]) }
              )
newTruePivs = simulateH(data=data_StEM, links=linksCpp,
                        survivalpSameH=survivalpSameH,
                        sumRowD=sumRowD, sumColD=sumColD, eta=eta, phi=phi)
truepivsA = newTruePivs$truepivsA
truepivsB = newTruePivs$truepivsB
```

---

smd *Standardised mean difference between a selected set and a baseline*

Description

When missing values are encoded as 0m the smd will report information on the missingness in the linked sample vs. the source.

Usage

```
smd(data_select, data_baseline, var, continuous = TRUE)
```

Arguments

`data_select`	Data frame to compare (e.g. the linked set vs. the original file).
`data_baseline`	Data frame to compare (e.g. the linked set vs. the original file).
`var`	Character, column name to compare.
`continuous`	Logical; if `TRUE`, compares means of `var` directly; if `FALSE`, compares the proportion at each observed level of `var` (default `TRUE`).

Value

Named list, one SMD per variable (`continuous = TRUE`) or per level (`continuous = FALSE`).

Examples

```
base <- data.frame(age = rnorm(200, 40, 10), sex = sample(c("M", "F"),
                   200, replace = TRUE))
select <- data.frame(age = rnorm(80, 43, 10), sex = sample(c("M", "F"), 80,
                   replace = TRUE, prob = c(0.6, 0.4)))
smd(select, base, "age")
smd(select, base, "sex", continuous = FALSE)
```

StEM *Stochastic Expectation-Maximisation for record linkage*

Description

Fits the FlexRL model with a Stochastic EM algorithm: each iteration runs a Gibbs sampler (alternating between simulating the true PIV values and the linkage matrix D) and then updates the model parameters (`gamma`, `eta`, `alpha`, `phi`) from the post-burn-in Gibbs draws. See the methodology paper https://10.1093/jrsssc/qlaf016 for details.

Usage

```
StEM(
  data,
  StEM_iter = 30,
  StEM_burnin = 15,
  gibbs_iter = 20,
  gibbs_burnin = 10,
  music_on = FALSE,
  new_directory = NULL,
  save_info_iter = FALSE,
  gamma0 = NULL,
  phiA0 = NULL,
  phiB0 = NULL,
  n_post_sample = 1000,
  model_dynamics = survival_model("exponential"),
  threads = 1
)
```

Arguments

`data`	List, typically the output of `prepare_data()`, with: `encodedA` (smaller source, PIVs encoded to natural numbers, `0` = missing), `encodedB` (larger source, encoded), `n_values`, `same_mistakes` (logical: same mistake parameter shared by A and B?), `PIVs_config` (named list, one entry per PIV, each a list with: `dynamics` (`"stable"`, `"flexible"`, or `"structured"`), `bound_mistakes` (length-2 numeric/NA, upper bound on the mistake probability in file 1 / file 2), `fix_mistakes` (length-2 numeric/NA, mistake probability fixed to this value in file 1 / file 2), and, only for `dynamics` = `"structured"`, `cond_hazard_cov` (a list with `cov1`

	and cov2, the names of covariates in file 1 / file 2 used to model the hazard of change).
StEM_iter	Integer, total number of StEM iterations (including burn-in).
StEM_burnin	Integer, number of StEM iterations discarded as burn-in.
gibbs_iter	Integer, total number of Gibbs iterations per StEM step (including burn-in).
gibbs_burnin	Integer, number of Gibbs iterations discarded as burn-in (0 lets the algorithm auto-detect burn-in from the stabilisation of the linked count).
music_on	Logical; if TRUE, opens a short tune in the browser when the algorithm finishes.
new_directory	Path to an existing directory to save progress after each iteration, or NULL to disable.
save_info_iter	Logical; save the environment at the end of each iteration (only used if new_directory is not NULL).
gamma0	Optional starting values for gamma; at random if NULL.
phiA0	Optional starting values for phi; at random if NULL.
phiB0	Optional starting values for phi; at random if NULL.
n_post_sample	Integer, number of posterior draws used to estimate the final linkage probabilities Delta. Default is set to 1000, we recommend not lowering it.
model_dynamics	Object from survival_model(): the survival model for the change over time of the structured PIVs (default exponential).
threads	Integer, number of OpenMP threads used by the C++ Gibbs kernels for their per-pair loops (default 4). Results are identical for any value. Ignored — the code runs sequentially — if the package was built without OpenMP support.

Value

A list with: Delta sparse-matrix summary (i, j, x) of posterior linkage probabilities; a pair is a valid link candidate once x > 0.5 (one-to-one constraint), gamma, eta, alpha, phi the StEM chains for each parameter.

Examples

```
PIVs_config <- list( V1 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V2 = list(dynamics = "stable",
                               bound_mistakes = c(0.10,0.10),
                               fix_mistakes = c(NA,NA)),
                     V3 = list(dynamics = "flexible",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(NA,NA)),
                     V4 = list(dynamics = "structured",
                               bound_mistakes = c(NA,NA),
                               fix_mistakes = c(0.03,0.03),
                               cond_hazard_cov = list(cov1=c("Xe", "Xf"),
                                                      cov2=c())) )
n_values  <- c( 5, 6, 7, 12 )
```

```
p_mistake <- list( V1 = c(0.02, 0.02), V2 = c(0.02, 0.02),
                   V3 = c(0.05, 0.05), V4 = c(0.02, 0.02) )
p_missing <- list( V1 = c(0.005, 0.005), V2 = c(0.005, 0.005),
                   V3 = c(0.005, 0.005), V4 = c(0.005, 0.005) )

cond_hazard_params1 <- list( V1 = c(), V2 = c(),
                             V3 = c(), V4 = log(c(0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params1, TRUE )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
fit1 <- StEM( data = prep_data, StEM_iter = 5, StEM_burnin = 2,
              gibbs_iter = 5, gibbs_burnin = 2, n_post_sample = 5,
              model_dynamics = survival_model("exponential") )
apply(fit1$alpha$V4[2:5,], 2, mean)
cond_hazard_params1$V4
head(fit1$Delta[fit1$Delta$x > 0.5, ])

cond_hazard_params2 <- list( V1 = c(), V2 = c(),
                             V3 = c(), V4 = log(c(0.3, 0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params2, TRUE,
                           model_dynamics = survival_model("weibull") )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
fit2 <- StEM( data = prep_data, StEM_iter = 5, StEM_burnin = 2,
              gibbs_iter = 5, gibbs_burnin = 2, n_post_sample = 5,
              model_dynamics = survival_model("weibull"))
apply(fit2$alpha$V4[2:5,], 2, mean)
cond_hazard_params2$V4
head(fit2$Delta[fit2$Delta$x > 0.5, ])

cond_hazard_params3 <- list( V1 = c(), V2 = c(),
                             V3 = c(), V4 = log(c(0.3, 0.7, 0.6, 0.5)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params3, TRUE,
                           model_dynamics = survival_model("gompertz") )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
fit3 <- StEM( data = prep_data, StEM_iter = 5, StEM_burnin = 2,
              gibbs_iter = 5, gibbs_burnin = 2, n_post_sample = 5,
              model_dynamics = survival_model("gompertz"))
apply(fit3$alpha$V4[2:5,], 2, mean)
cond_hazard_params3$V4
head(fit3$Delta[fit3$Delta$x > 0.5, ])

cond_hazard_params4 <- list( V1 = c(), V2 = c(),
                             V3 = c(), V4 = log(c(0.6, 0.5, 0.4, 0.3, 0.2, 0.1)) )
gen_data <- simulate_data( PIVs_config, n_values, c(150, 200), 100,
                           p_mistake, p_missing, cond_hazard_params4, TRUE,
                          model_dynamics = survival_model("piecewise", cuts = c(1,2,3)) )
prep_data <- prepare_data( gen_data$data1, gen_data$data2, "1", "2",
                           PIVs_config, TRUE, "entity_id", TRUE )
```

```
fit4 <- StEM( data = prep_data, StEM_iter = 2, StEM_burnin = 1,
              gibbs_iter = 2, gibbs_burnin = 1, n_post_sample = 2,
              model_dynamics = survival_model("piecewise", cuts = c(1,2,3)))
apply(fit4$alpha$V4, 2, mean)
cond_hazard_params4$V4
head(fit4$Delta[fit4$Delta$x > 0.5, ])
```

`support_iou` *Intersection-over-union of two histogram supports*

Description

Intersection-over-union of two histogram supports

Usage

```
support_iou(h1, h2)
```

Arguments

`h1`	`histogram` object (as returned by `graphics::hist()`).
`h2`	`histogram` object (as returned by `graphics::hist()`).

Value

Numeric, IoU of the ranges over which `h1` and `h2` have non-zero counts.

Examples

```
h1 <- graphics::hist(rnorm(200), plot = FALSE)
h2 <- graphics::hist(rnorm(200) + 1, plot = FALSE)
support_iou(h1, h2)
```

`survival_model` *Survival models for the dynamics of a PIV*

Description

A structured PIV may change between the two registrations. The probability that the true value of a linked pair is unchanged after a time gap `t` is modelled by a survival function `S(t | X, alpha)`, where `X` holds an intercept and the covariates given in `cond_hazard_cov` and `alpha` the parameters estimated by `StEM()`. This function builds the model object used by `StEM()`, `simulateD()` and `simulate_data()`.

Usage

```
survival_model(
  type = c("exponential", "weibull", "gompertz", "piecewise", "custom"),
  cuts = NULL,
  S = NULL,
  n_par = NULL,
  init = NULL
)
```

Arguments

type	One of "exponential", "weibull", "gompertz", "piecewise", "custom".
cuts	Numeric vector of cut points for type = "piecewise" (e.g. c(1, 3) gives three intervals [0,1), [1,3), [3, Inf)).
S	For type = "custom": function (X, alpha, times) returning the survival probability of each linked pair; X is a matrix with an intercept column first, then the covariates.
n_par	For type = "custom": function of ncol(X) returning the length of alpha.
init	For type = "custom": function of ncol(X) returning the starting values of alpha.

Details

Available models (h is the hazard, lambda = exp(X alpha_cov) the proportional-hazards term):

"exponential" S(t) = exp(-lambda t); alpha = coefficients of X (default, the model of the methodology paper).

"weibull" S(t) = exp(-(lambda t)^k) with shape k = exp(alpha[1]); alpha = log-shape, then coefficients of X.

"gompertz" h(t) = lambda exp(g t), so S(t) = exp(-lambda (exp(g t) - 1) / g) with g = alpha[1]; alpha = g, then coefficients of X.

"piecewise" piecewise-constant baseline hazard on the intervals defined by cuts, times lambda; alpha = one log-hazard per interval, then coefficients of the covariates (no intercept).

"custom" S, n_par and init supplied by the user.

For every model the negative log-likelihood used in the M-step is -sum(Hequal log S + (1 - Hequal) log(1 - S)), where Hequal indicates that the true values of the linked pair agree. Parameters are estimated with stats::nlminb() (numerical gradient).

Value

An object of class "survival_model": a list with type, S, negloglik, n_par, init and par_names.

Examples

```
X <- cbind(intercept = rep(1, 5))
times <- c(0.001, 0.2, 1.3, 1.5, 2)
Hequal <- c(TRUE, TRUE, TRUE, FALSE, FALSE)
```

```
expo <- survival_model("exponential")
expo$S(X, alpha = log(0.3), times)
stats::nlminb(expo$init(ncol(X)), expo$negloglik, X = X, times = times,
              Hequal = Hequal)$par

weib <- survival_model("weibull")
weib$S(X, alpha = c(0.5, -1), times)
stats::nlminb(weib$init(ncol(X)), weib$negloglik, X = X, times = times,
              Hequal = Hequal)$par

gomp <- survival_model("gompertz")
gomp$S(X, alpha = c(0.5, -1), times)
stats::nlminb(gomp$init(ncol(X)), gomp$negloglik, X = X, times = times,
              Hequal = Hequal)$par

pw <- survival_model("piecewise", cuts = 1)
pw$S(X, alpha = c(-2, -1), times)
stats::nlminb(pw$init(ncol(X)), pw$negloglik, X = X, times = times,
              Hequal = Hequal)$par

# a custom model: log-logistic with scale exp(X alpha[-1])
# and shape exp(alpha[1])
loglogistic <- survival_model("custom",
  S = function(X, alpha, times) {
    lambda <- exp(as.matrix(X) %*% alpha[-1])
    1 / ( 1 + (lambda * times)^exp(alpha[1]) )
  },
  n_par = function(n_cov) n_cov + 1,
 init = function(n_cov) c(stats::rnorm(1, 0, 0.1), stats::runif(n_cov, log(0.01), log(1))))
loglogistic$S(X, alpha = c(0, -2), times)
stats::nlminb(loglogistic$init(ncol(X)), loglogistic$negloglik, X = X, times = times,
              Hequal = Hequal)$par
```

`survival_S` *Evaluate a survival model's S, C++ fast path for the built-in families*

Description

Internal dispatcher used by the Gibbs hot path (`StEM()` through `simulateD()` and the per-sweep `survivalpSameH` refresh): built-in models evaluate through the fused, OpenMP-parallel `survivalS_cpp()` kernel; `type = "custom"` models keep calling the user's R function, unchanged.

Usage

```
survival_S(model, X, alpha, times, threads = 4)
```

Arguments

`model`	Object built by `survival_model()`.
`X`	Covariate matrix, intercept column first (same contract as `model$S`).

alpha	Parameter vector (layout per family, see survival_model()).
times	Numeric vector of time gaps.
threads	Integer, number of OpenMP threads for the C++ kernel (default 4; ignored without OpenMP; identical results for any value).

Value

Numeric vector of survival probabilities, one per row of X.

---

synthesise *Augment file B with synthetic records*

Description

Fits a generative model on encodedB[, PIVs] and draws n_synth new synthetic records from it, appended to encodedB with source = "synthetic"; used by compute_augmRL_FDP_synth() to estimate the false discovery proportion of a record-linkage method without ground truth.

Usage

```
synthesise(
  method,
  encodedA,
  encodedB,
  PIVs,
  n_synth,
  restrict_support_intersection = TRUE
)
```

Arguments

method	One of "arf" (arf::adversarial_rf()), "synthpop" (synthpop::syn()), or "mice" (mice::mice()).
encodedA	The (already prepared / encoded) data source.
encodedB	The (already prepared / encoded) data source.
PIVs	Character vector, names of the PIVs to synthesise.
n_synth	Integer, number of synthetic records to generate.
restrict_support_intersection	Logical; drop synthetic records whose PIV values fall outside the support shared with encodedA (default TRUE).

Value

List with dataA (unchanged encodedA, support-restricted) and dataB (encodedB plus the synthetic records).

Index

arf::adversarial_rf(), *44*

compute_augmRL_FDP_synth, 3
compute_augmRL_FDP_synth(), *12–16*, *30*, *44*
compute_RL_FDP_score, 5
compute_RL_FDP_score(), *12–16*, *30*
create_data_alpha, 6

FDP_score, 7
FDP_synth, 8
FlexRL, 10
FlexRL-package *(*FlexRL*)*, 10

graphics::hist(), *41*

link_with_BRL, 11
link_with_diyar, 12
link_with_fastLink, 13
link_with_fedmatch, 14
link_with_FlexRL, 15
link_with_multilink, 15
link_with_reclin2, 16
log_lik, 17
log_lik(), *18*
log_possible_config, 18

mice::mice(), *44*
mmd, 18
mmd(), *30*

naive_linkage, 19

plot.discrepancy_curves, 20
plot.discrepancy_curves(), *21*
plot.FDP_curves, 20
plot.RL_diagnostics, 21
plot_distributions, 22
plot_linkage_scores, 23
plot_StEM_convergence, 24
plot_StEM_convergence(), *21*
prepare_data, 25
prepare_data(), *3*, *5*, *10*, *12–16*, *38*
print.RL_diagnostics, 26
prop_level, 27

RL_agreement, 28
RL_agreement(), *30*
RL_diagnostics, 29
RL_diagnostics(), *10*, *12–16*, *21*, *27*

simulate_data, 31
simulate_data(), *25*, *41*
simulateD, 33
simulateD(), *41*, *43*
simulateH, 35
simulateH(), *34*
smd, 37
smd(), *30*
stats::nlminb(), *42*
StEM, 38
StEM(), *6*, *10*, *15*, *24–26*, *29*, *34*, *36*, *41*, *43*
support_iou, 41
support_iou(), *30*
survival_model, 41
survival_model(), *32*, *34*, *39*, *43*, *44*
survival_S, 43
synthesise, 44
synthesise(), *3*, *12–16*
synthpop::syn(), *44*