Package: FlexRL (via r-universe)

May 29, 2026

Title A Flexible Model for Record Linkage

Version 0.1.1

Description Implementation of the Stochastic Expectation Maximisation (StEM) approach to Record Linkage described in the paper by K. Robach, S. L. van der Pas, M. A. van de Wiel and M. H. Hof (2025, <doi:10.1093/jrsssc/qlaf016>); see citation(``FlexRL'') for details. This is a record linkage method, for finding the common set of records among 2 data sources based on Partially Identifying Variables (PIVs) available in both sources. It includes modelling of dynamic Partially Identifying Variables (e.g. postal code) that may evolve over time and registration errors (missing values and mistakes in the registration). Low memory footprint.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.4.0)

Imports Matrix (>= 1.7), progress(>= 1.2.3), testit(>= 0.13), Rcpp(>= 1.0.13)

LinkingTo Rcpp

Suggests knitr, rmarkdown

VignetteBuilder knitr

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

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

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

Date/Publication 2026-05-29 08:48:22 UTC

RemoteUrl https://github.com/robachowyk/flexrl

RemoteRef HEAD

RemoteSha 6776b28ef54f436c79c53a1277767d38e64c1f99

Contents

`cartesianProduct` *cartesianProduct*

Description

cartesianProduct

Usage

```
cartesianProduct(vec1, vec2)
```

Arguments

`vec1`	first IntegerVector of values to compute the cartesian product
`vec2`	second IntegerVector of values to compute the cartesian product

Value

IntegerMatrix: of 2 columns with the cartesian product of vec1 and vec2

createDataAlpha *createDataAlpha*

Description

createDataAlpha

Usage

```
createDataAlpha(nCoefUnstable, stable)
```

Arguments

nCoefUnstable	An integer value with the number of covariates (including the intercept): number of cov from A + number of cov from B + 1.
stable	A boolean value indicating whether the Partially Identifying Variable (PIV) concerned is stable.

Value

An empty data frame (if stable=FALSE) with nCoefUnstable + 2 columns for book keeping of the elements necessary to update the parameter for instability. There are nCoefUnstable + 2 of those elements: number of cov from A + number of cov from B + intercept + boolean vector indicating where the true values of the records (for the concerned PIV) are equal, vector of time gaps between records

Examples

```
PIVs_config = list( V1 = list(stable = TRUE),
                    V2 = list(stable = TRUE),
                    V3 = list(stable = TRUE),
                    V4 = list(stable = TRUE),
                    V5 = list( stable = FALSE,
                               conditionalHazard = FALSE,
                               pSameH.cov.A = c(),
                               pSameH.cov.B = c()) )

PIVs_stable = sapply(PIVs_config, function(x) x$stable)

nCoefUnstable = c(0,0,0,0,1)

Valpha = mapply( createDataAlpha,
                 nCoefUnstable = nCoefUnstable,
                 stable = PIVs_stable,
                 SIMPLIFY=FALSE)
```

DataCreation *DataCreation*

Description

This function is used to synthesise data for record linkage. It creates 2 data sources of specific sizes, with a common set of records of a specific size, with a certain amount of Partially Identifying Variables (PIVs). For each PIV, we specify the number of unique values, the desired proportion of mistakes and missing values. They can be stable or evolving over time (e.g. representing the postal code). For the unstable PIVs, we can specify the parameter(s) to be used in the survival exponential model that generates changes over time between the records referring to the same entities. When a PIV is unstable, it is later harder to estimate its parameters (probability of mistake vs. probability of change across time). Therefore we may want to enforce estimability in the synthetic data, which we do by enforcing half of the links to have a near-zero time gaps.

Usage

```
DataCreation(
  PIVs_config,
  Nval,
  NRecords,
  Nlinks,
  PmistakesA,
  PmistakesB,
  PmissingA,
  PmissingB,
  moving_params,
  enforceEstimability
)
```

Arguments

PIVs_config	A list (of size number of PIVs) where element names are the PIVs and element values are lists with elements: stable (boolean for whether the PIV is stable), conditionalHazard (boolean for whether there are external covariates available to model instability, only required if stable is FALSE), pSameH.cov.A and pSameH.covB (vectors with strings corresponding to the names of the covariates to use to model instability from file A and file B, only required if stable is FALSE, empty vectors may be provided if conditionalHazard is FALSE)
Nval	A vector (of size number of PIVs) with the number fo unique values per PIVs (in the order of the PIVs defined in PIVs_config)
NRecords	A vector (of size 2) with the number of records to be generated in file A and in file B
Nlinks	An integer with the number of links (record referring to the same entities) to be generated

`PmistakesA` A vector (of size number of PIVs) with the proportion of mistakes to be generated per PIVs in file A (in the order of the PIVs defined in PIVs_config)

`PmistakesB` A vector (of size number of PIVs) with the proportion of mistakes to be generated per PIVs in file B (in the order of the PIVs defined in PIVs_config)

`PmissingA` A vector (of size number of PIVs) with the proportion of missing to be generated per PIVs in file A (in the order of the PIVs defined in PIVs_config)

`PmissingB` A vector (of size number of PIVs) with the proportion of missing to be generated per PIVs in file A (in the order of the PIVs defined in PIVs_config)

`moving_params` A list (of size number of PIVs) where element names are the PIVs and element values are vectors (of size: 1 + number of covariates to use from A + number of covariates to use from B) with the log hazards coefficient (1st one: log baseline hazard, then: the coefficients for conditional hazard covariates from A, then: the coefficients for conditional hazard covariates from B)

`enforceEstimability` A boolean value for whether half of the links should have near-0 time gaps (useful for modeling instability and avoiding estimability issues as discussed in the paper)

Details

There are more details to understand the method in our paper, or on the experiments repository of our paper, or in the vignettes.

Value

A list with generated

- dataframe A (encoded: the categorical values of the PIVs are matched to sets of natural numbers),
- dataframe B (encoded),
- vector of Nvalues (Nval),
- vector of TimeDifference (for the links, when thewre is instability),
- matrix proba_same_H (number of links, number fo PIVs) with the proba that true values coincide (e.g. 1 - proba of moving)

Examples

```
PIVs_config = list( V1 = list(stable = TRUE),
                    V2 = list(stable = TRUE),
                    V3 = list(stable = TRUE),
                    V4 = list(stable = TRUE),
                    V5 = list( stable = FALSE,
                               conditionalHazard = FALSE,
                               pSameH.cov.A = c(),
                               pSameH.cov.B = c()) )
Nval = c(6, 7, 8, 9, 15)
NRecords = c(500, 800)
Nlinks = 300
```

```
PmistakesA = c(0.02, 0.02, 0.02, 0.02, 0.02)
PmistakesB = c(0.02, 0.02, 0.02, 0.02, 0.02)
PmissingA = c(0.007, 0.007, 0.007, 0.007, 0.007)
PmissingB = c(0.007, 0.007, 0.007, 0.007, 0.007)
moving_params = list(V1=c(),V2=c(),V3=c(),V4=c(),V5=c(0.28))
enforceEstimability = TRUE
DataCreation( PIVs_config,
              Nval,
              NRecords,
              Nlinks,
              PmistakesA,
              PmistakesB,
              PmissingA,
              PmissingB,
              moving_params,
              enforceEstimability)
```

`Deltafind` *Deltafind()*

Description

Deltafind()

Usage

```
Deltafind()
```

Value

IntegerMatrix: find the indices of the elements in the cpp map _DeltaMap representing the sparse linkage matrix Delta.

`ExpandGrid` *ExpandGrid*

Description

ExpandGrid

Usage

```
ExpandGrid(vector1, vector2)
```

Arguments

`vector1`	first IntegerVector of values to compute the cartesian product
`vector2`	second IntegerVector of values to compute the cartesian product

Value

IntegerMatrix: of 2 columns with the cartesian product of vec1 and vec2

`F2` *F2*

Description

F2

Usage

```
F2(U, nvals)
```

Arguments

`U`	IntegerVector with factor values corresponding to the patterns of Partially Identifying Variables (PIVs) observed among records in the concerned source
`nvals`	integer for the total number of possible patterns (among all sources)

Value

List: for each pattern in value, count of the records having the pattern in the concerned source

---

`F33` *F33*

Description

F33

Usage

```
F33(A, B, nvals)
```

Arguments

`A`	List with for each pattern in value, count of the records having the pattern in the concerned source
`B`	List with for each pattern in value, count of the records having the pattern in the concerned source
`nvals`	integer for the total number of possible patterns (among all sources)

Value

IntegerMatrix: nrow=nbr of potential links, ncol = 2, indicates the indices (of records from A, records from B) of records with matching patterns to be considered in the likelihood as potential links. It represents the cartesian product of both lists passed as parameters to represent all possible linked pairs of records.

`FlexRL` *FlexRL*

Description

A Flexible Model For Record Linkage

Arguments

`data`	is a list gathering information on the data to be linked • sources 'A' and 'B', • a vector 'Nvalues' gathering the number of unique values in each PIV, • 'PIVs_config' the list of PIVs to use for Record Linkage and details on how each should be handled by the algorithm, • potential bounds on the mistakes probabilities for each PIV: 'controlOnMistakes', • 'sameMistakes' whether there should be one parameter for the mistakes in A and B or whether each source should have its own (in case of small data sources it is recommended to set sameMistakes=TRUE) • whether the parameters for mistakes should be fixed in case of instability 'phiMistakesAFixed' and 'phiMistakesBFixed', • as well as the values they should be fixed to 'phiForMistakesA' and 'phiForMistakesB'
`StEMIter`	The total number of iterations of the Stochastic Expectation Maximisation (StEM) algorithm (including the period to discard as burn-in)
`StEMBurnin`	The number of iterations to discard as burn-in
`GibbsIter`	The total number of iterations of the Gibbs sampler (run in each iteration of the StEM) (including the period to discard as burn-in)
`GibbsBurnin`	The number of iterations to discard as burn-in

Details

The example below aims to link 2 synthetic data sources, with 5 PIVs (4 stable ones and 1 unstable, which evolve over time). PIVs may be considered stable if there is not enough information to model their dynamics. Since we synthesise the data we can model it. We may not give a bound on some PIVs in real settings, since there may be a lot of disagreements among the links for those variables (in situations where we would have liked to model their dynamics otherwise but we do not have enough information for this). Here we bound all the mistakes parameters since we know that the

mistakes probabilities are inferior to 10%. For a small example we prefer having one parameter for the probabilities of mistakes over the 2 data sources. We do need to fix the mistake parameter of the 5th PIV to avoid estimability problems here (since it is an 'unstable' variable). We know the true linkage structure in this example so we can compute performances of the method at the end.

There are more details to understand the method in our paper, or on the experiments repository of our paper, or in the vignettes, or in the documentation of the main algorithm ?FlexRL::StEM.

Value

The Stochastic Expectation Maximisation (StEM) function returns w list with:

- Delta, the (sparse) matrix with the pairs of records linked and their posterior probabilities to be linked (select the pairs where the proba>0.5 to get a valid set of linked records),
- as well as the model parameters chains:
  - gamma,
  - eta,
  - alpha,
  - phi.

Author(s)

Kayané Robach

Examples

```
PIVs_config = list( V1 = list(stable = TRUE),
                    V2 = list(stable = TRUE),
                    V3 = list(stable = TRUE),
                    V4 = list(stable = TRUE),
                    V5 = list( stable = FALSE,
                               conditionalHazard = FALSE,
                               pSameH.cov.A = c(),
                               pSameH.cov.B = c()) )
PIVs = names(PIVs_config)
PIVs_stable = sapply(PIVs_config, function(x) x$stable)
Nval = c(6, 7, 8, 9, 15)
NRecords = c(500, 800)
Nlinks = 300
PmistakesA = c(0.02, 0.02, 0.02, 0.02, 0.02)
PmistakesB = c(0.02, 0.02, 0.02, 0.02, 0.02)
PmissingA = c(0.007, 0.007, 0.007, 0.007, 0.007)
PmissingB = c(0.007, 0.007, 0.007, 0.007, 0.007)
moving_params = list(V1=c(),V2=c(),V3=c(),V4=c(),V5=c(0.28))
enforceEstimability = TRUE
DATA = DataCreation( PIVs_config,
                     Nval,
                     NRecords,
                     Nlinks,
                     PmistakesA,
                     PmistakesB,
```

```
                    PmissingA,
                    PmissingB,
                    moving_params,
                    enforceEstimability)
A                   = DATA$A
B                   = DATA$B
Nvalues             = DATA$Nvalues
TimeDifference      = DATA$TimeDifference
proba_same_H        = DATA$proba_same_H

# the first 1:Nlinks records of each files created are links
TrueDelta = base::data.frame( matrix(0, nrow=0, ncol=2) )
for (i in 1:Nlinks)
{
  TrueDelta = rbind(TrueDelta, cbind(rownames(A[i,]),rownames(B[i,])))
}
true_pairs = do.call(paste, c(TrueDelta, list(sep="_")))

encodedA = A
encodedB = B

encodedA[,PIVs][ is.na(encodedA[,PIVs]) ] = 0
encodedB[,PIVs][ is.na(encodedB[,PIVs]) ] = 0

data = list( A                  = encodedA,
             B                  = encodedB,
             Nvalues            = Nvalues,
             PIVs_config        = PIVs_config,
             controlOnMistakes  = c(TRUE, TRUE, TRUE, TRUE, TRUE),
             sameMistakes       = TRUE,
             phiMistakesAFixed  = TRUE,
             phiMistakesBFixed  = TRUE,
             phiForMistakesA    = c(NA, NA, NA, NA, 0),
             phiForMistakesB    = c(NA, NA, NA, NA, 0)
           )

fit = FlexRL::stEM(  data               = data,
                     StEMIter           = 50,
                     StEMBurnin         = 30,
                     GibbsIter          = 50,
                     GibbsBurnin        = 30,
                     musicOn            = TRUE,
                     newDirectory       = NULL,
                     saveInfoIter       = FALSE
                   )

DeltaResult = fit$Delta
colnames(DeltaResult) = c("idxA","idxB","probaLink")
DeltaResult = DeltaResult[DeltaResult$probaLink>0.5,]

results = base::data.frame( Results=matrix(NA, nrow=6, ncol=1) )
rownames(results) = c("tp","fp","fn","f1score","fdr","sens.")
if(nrow(DeltaResult)>1){
```

```
  linked_pairs    = do.call(paste, c(DeltaResult[,c("idxA","idxB")], list(sep="_")))
  truepositive    = length( intersect(linked_pairs, true_pairs) )
  falsepositive   = length( setdiff(linked_pairs, true_pairs) )
  falsenegative   = length( setdiff(true_pairs, linked_pairs) )
  precision       = truepositive / (truepositive + falsepositive)
  fdr             = 1 - precision
  sensitivity     = truepositive / (truepositive + falsenegative)
  f1score         = 2 * (precision * sensitivity) / (precision + sensitivity)
 results[,"FlexRL"] = c(truepositive,falsepositive,falsenegative,f1score,fdr,sensitivity)
}
```

`generateSequence` *generateSequence*

Description

generateSequence

Usage

```
generateSequence(n)
```

Arguments

n	integer superior to 1

Value

IntegerVector: with n values from 1 to n

`initDeltaMap` *initDeltaMap*

Description

initDeltaMap

Usage

```
initDeltaMap()
```

Value

void: Initialise the cpp map _DeltaMap representing the sparse linkage matrix Delta.

launchNaive *launchNaive*

Description

launchNaive

Usage

```
launchNaive(PIVs, encodedA, encodedB)
```

Arguments

PIVs	A vector of size the number of Partially Identifying Variables (PIVs) with their names (as columns names in the data sources).
encodedA	One data source (encoded: the categorical values of the PIVs have to be mapped to sets of natural numbers and missing values are encoded as 0).
encodedB	The other data source (encoded).

Value

The linkage set, a dataframe of 2 columns with indices from the first data source (A) and the second one (B) for which all the PIVs (when non-missing) matches in their values. When a PIV is missing, this method will match the record to all the records in the other file for which all other values of the PIVs match. Therefore, this 'naive' (or 'simplistic') method does not enforce the one-to-one assignment constraint of record linkage (one record in one file can at most be linked to one record in the other file). This method should only be used to judge the difficulty of the record linkage task: it gives information about the amount of duplicates between files and the discriminative power of all the PIVs together as a way to link the records.

Examples

```
PIVs_config = list( V1 = list(stable = TRUE),
                    V2 = list(stable = TRUE),
                    V3 = list(stable = TRUE),
                    V4 = list(stable = TRUE),
                    V5 = list( stable = FALSE,
                               conditionalHazard = FALSE,
                               pSameH.cov.A = c(),
                               pSameH.cov.B = c()) )
PIVs = names(PIVs_config)
PIVs_stable = sapply(PIVs_config, function(x) x$stable)
Nval = c(6, 7, 8, 9, 15)
NRecords = c(500, 800)
Nlinks = 300
PmistakesA = c(0.02, 0.02, 0.02, 0.02, 0.02)
PmistakesB = c(0.02, 0.02, 0.02, 0.02, 0.02)
PmissingA = c(0.007, 0.007, 0.007, 0.007, 0.007)
```

```
PmissingB = c(0.007, 0.007, 0.007, 0.007, 0.007)
moving_params = list(V1=c(),V2=c(),V3=c(),V4=c(),V5=c(0.28))
enforceEstimability = TRUE
DATA = DataCreation( PIVs_config,
                     Nval,
                     NRecords,
                     Nlinks,
                     PmistakesA,
                     PmistakesB,
                     PmissingA,
                     PmissingB,
                     moving_params,
                     enforceEstimability)
A                    = DATA$A
B                    = DATA$B
Nvalues              = DATA$Nvalues
TimeDifference       = DATA$TimeDifference
proba_same_H         = DATA$proba_same_H

TrueDelta = base::data.frame( matrix(0, nrow=0, ncol=2) )
for (i in 1:Nlinks)
{
  TrueDelta = rbind(TrueDelta, cbind(rownames(A[i,]),rownames(B[i,])))
}
true_pairs = do.call(paste, c(TrueDelta, list(sep="_")))

encodedA = A
encodedB = B

encodedA[,PIVs][ is.na(encodedA[,PIVs]) ] = 0
encodedB[,PIVs][ is.na(encodedB[,PIVs]) ] = 0

DeltaResult = launchNaive(PIVs, encodedA, encodedB)

results = base::data.frame( Results=matrix(NA, nrow=6, ncol=1) )
rownames(results) = c("tp","fp","fn","f1score","fdr","sens.")
if(nrow(DeltaResult)>1){
  linked_pairs    = do.call(paste, c(DeltaResult[,c("idxA","idxB")], list(sep="_")))
  truepositive    = length( intersect(linked_pairs, true_pairs) )
  falsepositive   = length( setdiff(linked_pairs, true_pairs) )
  falsenegative   = length( setdiff(true_pairs, linked_pairs) )
  precision       = truepositive / (truepositive + falsepositive)
  fdr             = 1 - precision
  sensitivity     = truepositive / (truepositive + falsenegative)
  f1score         = 2 * (precision * sensitivity) / (precision + sensitivity)
 results["Naive"] = c(truepositive,falsepositive,falsenegative,f1score,fdr,sensitivity)
}
results
```

---

loglik *loglik*

Description

Log(likelihood) of the linkage matrix.

Usage

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

Arguments

LLL	A (sparse) matrix with contributions to the complete likelihood of the linked records.
LLA	A vector with contributions to the complete likelihood of the non linked records from A.
LLB	A vector with contributions to the complete likelihood of the non linked records from B.
links	A matrix of 2 columns with indices of the linked records.
sumRowD	A boolean vector indicating, for each row of the linkage matrix, i.e. for each record in the smallest file A, whether the record has a link in B or not.
sumColD	A boolean vector indicating, for each column of the linkage matrix, i.e. for each record in the largest file B, whether the record has a link in A or not.
gamma	The proportion of linked records as a fraction of the smallest file.

Value

The Log(likelihood) of the linkage matrix.

Examples

```
LLL = Matrix::Matrix(0, nrow=13, ncol=15, sparse=TRUE)
LLA = c(0.001,0.001,0.001,0.001,1.43,0.02,0.007,0.001,2.1,0.0003,1.67,1.5,10)
LLB = c(0.001,0.001,0.001,0.001,1.22,0.008,0.01,0.04,3.9,0.0002,1.99,0,2.4,0.009,12)
linksR = as.matrix(base::data.frame(list(idxA=c(5,9,11,12,13), idxB=c(5,9,11,13,15))))
LLL[linksR] = 0.67
sumRowD = c(0,0,0,0,1,0,0,0,1,0,1,1,1)
sumColD = c(0,0,0,0,1,0,0,0,1,0,1,0,1,0,1)
gamma = 0.5
LL0 = loglik(LLL=LLL, LLA=LLA, LLB=LLB, links=linksR, sumRowD=sumRowD, sumColD=sumColD,
             gamma=gamma)
```

`loglikSurvival` *The log likelihood of the survival function with exponential model (-)*

Description

Log(likelihood) of the survival function with exponential model (as proposed in our paper), representing the probability that true values of a pair of records referring to the same entity coincide. See ?FlexRL::SurvivalUnstable. This function is only used if the PIV is unstable and evolve over time. If so the true values of a linked pair of records may not coincide. If you want to use a different survival function to model instability, you can change the function 'SurvivalUnstable' as well as this function 'loglikSurvival'.

Usage

```
loglikSurvival(alphas, X, times, Hequal)
```

Arguments

`alphas`	A vector of size 1+cov in A+cov in B with coefficients of the hazard (baseline hazard and conditional hazard)
`X`	A matrix with number of linked records rows and 1+cov in A+cov in B columns (first column: intercept, following columns: covariates from A and then from B to model instability) (used for optimisation: X concatenate the X obtained in each iteration of the Gibbs sampler)
`times`	A vector of size number of linked records with the time gaps between the record from each sources (used for optimisation: times concatenate the times vectors obtained in each iteration of the Gibbs sampler)
`Hequal`	A vector of size number of linked records with boolean values indicating wether the values in A and in B coincide (used for optimisation: times concatenate the times vectors obtained in each iteration of the Gibbs sampler)

Details

In our Stochastic Expectation Maximisation (StEM) algorithm (see ?FlexRL::stEM) we minimise - log(likelihood), which is equivalent to maximise log(likelihood). Therefore this function actually returns (and should return if you create your own) the opposite (-) of the log(likelihood) associated with the survival function defining the probabilities that true values coincide.

Value

The value of the opposite (-) of the log(likelihood) associated with the survival function defining the probabilities that true values coincide (as defined in the paper) (the algorithm minimises -log(likelihood) i.e. maximises the log(likelihood)).

Examples

```
nCoefUnstable = 1
alphaInit = rep(-0.05, nCoefUnstable)
Valpha = base::data.frame(list(cov=c(2,2.1,3.4,2.5,2.9),
                               times=c(0.001,0.2,1.3,1.5,2),
                               Hequal=c(TRUE, TRUE, TRUE, FALSE, FALSE)))
X = Valpha[,1:nCoefUnstable]
times = Valpha$times
Hequal = Valpha$Hequal
optim = stats::nlminb(alphaInit, loglikSurvival, control=list(trace=FALSE),
                      X=X, times=times, Hequal=Hequal)
alpha = optim$par
```

---

logPossibleConfig *logPossibleConfig*

Description

This function helps calculating the number of possible designs for Delta given by nB!/(nB-nLinks)! Needed to compute the log likelihood of the linkage matrix.

Usage

```
logPossibleConfig(Brecords, sumD)
```

Arguments

Brecords	Number of records in data source B (the largest).
sumD	Number of linked records (at a specific time point of the algorithm).

Value

The sum of logs of the vector going from nB - nLinks + 1 to nB.

Examples

```
sumColD = c(0,0,0,0,1,0,0,0,1,0,1,0,1,0,1)
links = base::data.frame(list(idxA=c(5,9,11,12,13), idxB=c(5,9,11,13,15)))
possconfig = logPossibleConfig(length(sumColD),nrow(links))
```

sampleD *sampleD*

Description

sampleD

Usage

```
sampleD(S, LLA, LLB, LLL, gamma, loglik, nlinkrec, sumRowD, sumColD)
```

Arguments

S	IntegerMatrix where each row correspond to the indices (from source A and source B) of records for which the true values matches (representing the potential links)
LLA	NumericVector gives the likelihood contribution of each non linked record from A
LLB	NumericVector gives the likelihood contribution of each non linked record from B
LLL	NumericVector gives the likelihood contribution of each potential linked records (from select)
gamma	NumericVector repeats the value of the parameter gamma (proportion of linked records) number of potential linked records (nrow of S) times
loglik	double for the value of the current complete log likelihood of the model
nlinkrec	integer for the current number of linked records
sumRowD	A LogicalVector vector indicating, for each row of the linkage matrix, i.e. for each record in the smallest file A, whether the record has a link in B or not.
sumColD	A LogicalVector vector indicating, for each column of the linkage matrix, i.e. for each record in the largest file B, whether the record has a link in A or not.

Value

List:

- new set of links
- new sumRowD
- new sumColD
- new value of the complete log likelihood
- new number fo linked records

sampleH *sampleH*

Description

sampleH

Usage

```
sampleH(
  nA,
  nB,
  links,
  survivalpSameH,
  pivs_stable,
  pivsA,
  pivsB,
  nvalues,
  nonlinkedA,
  nonlinkedB,
  eta,
  phi
)
```

Arguments

nA	IntegerVector of dimensions of registered values of the Partially Identifying Variables (PIVs) in A
nB	IntegerVector of dimensions of registered values of the PIVs in B
links	IntegerMatrix of 2 columns with the indices of the linked records
survivalpSameH	NumericMatrix with for each PIV the probability that true values coincide (if stable: filled with 1)
pivs_stable	LogicalVector indicating for each PIV whether it is stable of not (if not we expect survivalpSameH for that same element to not be filled with 1 but with lower values)
pivsA	List ith registered data from A
pivsB	List with registered data from B
nvalues	IntegerVector with number of unique values of each PIV
nonlinkedA	LogicalVector indicating for all records in A whether they are linked or not
nonlinkedB	LogicalVector indicating for all records in B whether they are linked or not
eta	List parameters of the PIVs distributions
phi	List parameters of the PIVs registration errors

Value

List:

- truePIVsA, true values underlying data in A
- truePIVsB, true values underlying data in B

`sampleL` *sampleNL*

Description

sampleNL

Usage

```
sampleL(
  GA,
  GB,
  survivalpSameH,
  choice_set,
  choice_equal,
  nval,
  phikA,
  phikB,
  eta
)
```

Arguments

`GA`	IntegerVector of registered values for a certain Partially Identifying Variable (PIV) for linked records from A
`GB`	IntegerVector of registered values for a certain PIV for linked records from B
`survivalpSameH`	NumericVector of probabilities that the concerned PIV values coincide between file A and file B
`choice_set`	IntegerMatrix of 2 columns (for A and for B) with possible joint true values underlying GA and GB
`choice_equal`	IntegerVector of booleans indicating whether the 2 true values (from A and B) in the choice set are equal
`nval`	integer for the number of unique values in the PIV concerned
`phikA`	NumericVector parameter for the registration errors in A for the PIV concerned
`phikB`	NumericVector parameter for the registration errors in B for the PIV concerned
`eta`	NumericVector parameter for the distribution of the PIV concerned

Value

IntegerVector: of indices from the joint latent true values choice set underlying GA and GB

`sampleNL` *sampleNL*

Description

sampleNL

Usage

```
sampleNL(G, eta, phi)
```

Arguments

`G`	IntegerVector of registered values for a certain Partially Identifying Variable (PIV) for non linked records
`eta`	NumericVector parameter for the distribution of the PIV concerned
`phi`	NumericVector parameter for the registration errors for the PIV concerned

Value

IntegerVector: of latent true values underlying G

`simulateD` *simulateD*

Description

simulateD

Usage

```
simulateD(
  data,
  linksR,
  sumRowD,
  sumColD,
  truepivsA,
  truepivsB,
  gamma,
  eta,
  alpha,
  phi
)
```

Arguments

`data` A list with elements:

- A: the smallest data source (encoded: the categorical values of the PIVs have to be mapped to sets of natural numbers and missing values are encoded as 0).
- B: the largest data source (encoded).
- Nvalues: A vector (of size number of PIVs) with the number fo unique values per PIVs (in the order of the PIVs defined in PIVs_config).
- PIVs_config: A list (of size number of PIVs) where element names are the PIVs and element values are lists with elements: stable (boolean for whether the PIV is stable), conditionalHazard (boolean for whether there are external covariates available to model instability, only required if stable is FALSE), pSameH.cov.A and pSameH.covB (vectors with strings corresponding to the names of the covariates to use to model instability from file A and file B, only required if stable is FALSE, empty vectors may be provided if conditionalHazard is FALSE).
- controlOnMistakes: A vector (of size number of PIVs) of booleans indicating potential bounds on the mistakes probabilities for each PIV. For each PIV, if TRUE there will be control on mistake and the mistake probability will not go above 10%. If FALSE there is no bound on the probability of mistake. WATCH OUT, if you suspect that a variable is unstable but you do not have data to model its dynamics the boolean value should be set to FALSE to allow the parameter for mistake to adapt for the instability. However if you model this instability, the boolean value should be set to TRUE to help the algorithm differenciate the mistakes from the changes over time.
- sameMistakes: A boolean value for whether there should be one parameter for the mistakes in A and B or whether each source should have its own parameter. Setting sameMistakes=TRUE is recommended in case of small data sources; the estimation with 2 parameters in that case will fail to capture the mistakes correctly while 1 parameter will be more adapted.
- phiMistakesAFixed A vector (of size number of PIVs) of booleans indicating whether the parameters for mistakes should be fixed in case of instability. It should be FALSE, except for unstable PIVs for which it may be set to TRUE in order to avoid estimability problems between the parameter for mistake and the parameter for changes across time.
- phiMistakesBFixed A vector (of size number of PIVs) of booleans indicating whether the parameters for mistakes should be fixed in case of instability. It should be FALSE, except for unstable PIVs for which it may be set to TRUE in order to avoid estimability problems between the parameter for mistake and the parameter for changes across time.
- phiForMistakesA A vector (of size number of PIVs) of NA or fixed values for the parameters for mistakes. It should be NA, except for unstable PIVs for which one wants to fix the parameter to avoid estimability problem (as indicated with the boolean values in phiMistakesAFixed). In that case it should be set the the expected value for the probability of mistake. If you have no idea: you can put it to 0, the algorithm is quite robust to wrongly fixed parameters.

- phiForMistakesB A vector (of size number of PIVs) of NA or fixed values for the parameters for mistakes. It should be NA, except for unstable PIVs for which one wants to fix the parameter to avoid estimability problem (as indicated with the boolean values in phiMistakesBFixed). In that case it should be set the the expected value for the probability of mistake. If you have no idea: you can put it to 0, the algorithm is quite robust to wrongly fixed parameters.

`linksR` A matrix of 2 columns with indices of the linked records.

`sumRowD` A boolean vector indicating, for each row of the linkage matrix, i.e. for each record in the smallest file A, whether the record has a link in B or not.

`sumColD` A boolean vector indicating, for each column of the linkage matrix, i.e. for each record in the largest file B, whether the record has a link in A or not.

`truepivsA` A matrix of the shape of data source A, representing the true values of the PIVs underlying the registered values present in A.

`truepivsB` A matrix of the shape of data source B, representing the true values of the PIVs underlying the registered values present in B.

`gamma` The proportion of linked records as a fraction of the smallest file.

`eta` The distribution weights for the PIVs.

`alpha` The parameter involved in the survival model for the probability of true values to coincide (parameter for instability).

`phi` The proportion of mistakes and missing for the PIVs.

Value

A list with:

- new set of links
- new sumRowD
- new sumColD
- new value of the complete log likelihood
- new number fo linked records

Examples

```
PIVs_config = list( V1 = list(stable = TRUE),
                    V2 = list(stable = TRUE),
                    V3 = list(stable = TRUE),
                    V4 = list(stable = TRUE),
                    V5 = list( stable = FALSE,
                               conditionalHazard = FALSE,
                               pSameH.cov.A = c(),
                               pSameH.cov.B = c()) )
PIVs = names(PIVs_config)
PIVs_stable = sapply(PIVs_config, function(x) x$stable)
Nval = c(6, 7, 8, 9, 15)
NRecords = c(13, 15)
```

```
Nlinks = 6
PmistakesA = c(0.02, 0.02, 0.02, 0.02, 0.02)
PmistakesB = c(0.02, 0.02, 0.02, 0.02, 0.02)
PmissingA = c(0.007, 0.007, 0.007, 0.007, 0.007)
PmissingB = c(0.007, 0.007, 0.007, 0.007, 0.007)
moving_params = list(V1=c(),V2=c(),V3=c(),V4=c(),V5=c(0.28))
enforceEstimability = TRUE
DATA = DataCreation( PIVs_config,
                     Nval,
                     NRecords,
                     Nlinks,
                     PmistakesA,
                     PmistakesB,
                     PmissingA,
                     PmissingB,
                     moving_params,
                     enforceEstimability)
A                    = DATA$A
B                    = DATA$B
Nvalues              = DATA$Nvalues

encodedA = A
encodedB = B

encodedA[,PIVs][ is.na(encodedA[,PIVs]) ] = 0
encodedB[,PIVs][ is.na(encodedB[,PIVs]) ] = 0

dataForStEM = list( A                    = encodedA,
                    B                    = encodedB,
                    Nvalues              = Nvalues,
                    PIVs_config          = PIVs_config,
                    controlOnMistakes    = c(TRUE, TRUE, TRUE, TRUE, TRUE),
                    sameMistakes         = TRUE,
                    phiMistakesAFixed    = TRUE,
                    phiMistakesBFixed    = TRUE,
                    phiForMistakesA      = c(NA, NA, NA, NA, 0),
                    phiForMistakesB      = c(NA, NA, NA, NA, 0)
)

initDeltaMap()
linksR = base::matrix(0,0,2)
linksCpp = linksR
sumRowD = rep(0, nrow(dataForStEM$A))
sumColD = rep(0, nrow(dataForStEM$B))
nlinkrec = 0
survivalpSameH = base::matrix(1, nrow(linksR), length(dataForStEM$Nvalues))
gamma = 0.5
eta = lapply(dataForStEM$Nvalues, function(x) rep(1/x,x))
phi = lapply(dataForStEM$Nvalues, function(x)  c(0.9,0.9,0.1,0.1))
nCoefUnstable = lapply( seq_along(PIVs_stable),
                        function(idx)
                        if(PIVs_stable[idx]){ 0 }
                        else{
```

```
                        ncol(dataForStEM$A[, dataForStEM$PIVs_config[[idx]]$pSameH.cov.A,
                              drop=FALSE]) +
                        ncol(dataForStEM$B[, dataForStEM$PIVs_config[[idx]]$pSameH.cov.B,
                              drop=FALSE]) +
                         1 } )
alpha = lapply( seq_along(PIVs_stable),
                function(idx) if(PIVs_stable[idx]){ c(-Inf) }else{
                  rep(log(0.05), nCoefUnstable[[idx]]) })
newTruePivs = simulateH(data=dataForStEM, links=linksCpp, survivalpSameH=survivalpSameH,
                        sumRowD=sumRowD, sumColD=sumColD, eta=eta, phi=phi)
truepivsA = newTruePivs$truepivsA
truepivsB = newTruePivs$truepivsB
Dsample = simulateD(data=dataForStEM, linksR=linksR, sumRowD=sumRowD, sumColD=sumColD,
                    truepivsA=truepivsA, truepivsB=truepivsB,
                    gamma=gamma, eta=eta, alpha=alpha, phi=phi)
linksCpp = Dsample$links
linksR = linksCpp + 1
```

`simulateH` *simulateH*

Description

simulateH

Usage

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

Arguments

`data` A list with elements:

- A: the smallest data source (encoded: the categorical values of the PIVs have to be mapped to sets of natural numbers and missing values are encoded as 0).
- B: the largest data source (encoded).
- Nvalues: A vector (of size number of PIVs) with the number fo unique values per PIVs (in the order of the PIVs defined in PIVs_config).
- PIVs_config: A list (of size number of PIVs) where element names are the PIVs and element values are lists with elements: stable (boolean for whether the PIV is stable), conditionalHazard (boolean for whether there are external covariates available to model instability, only required if stable is FALSE), pSameH.cov.A and pSameH.covB (vectors with strings corresponding to the names of the covariates to use to model instability from file A and file B, only required if stable is FALSE, empty vectors may be provided if conditionalHazard is FALSE).

- controlOnMistakes: A vector (of size number of PIVs) of booleans indicating potential bounds on the mistakes probabilities for each PIV. For each PIV, if TRUE there will be control on mistake and the mistake probability will not go above 10%. If FALSE there is no bound on the probability of mistake. WATCH OUT, if you suspect that a variable is unstable but you do not have data to model its dynamics the boolean value should be set to FALSE to allow the parameter for mistake to adapt for the instability. However if you model this instability, the boolean value should be set to TRUE to help the algorithm differenciate the mistakes from the changes over time.
- sameMistakes: A boolean value for whether there should be one parameter for the mistakes in A and B or whether each source should have its own parameter. Setting sameMistakes=TRUE is recommended in case of small data sources; the estimation with 2 parameters in that case will fail to capture the mistakes correctly while 1 parameter will be more adapted.
- phiMistakesAFixed A vector (of size number of PIVs) of booleans indicating whether the parameters for mistakes should be fixed in case of instability. It should be FALSE, except for unstable PIVs for which it may be set to TRUE in order to avoid estimability problems between the parameter for mistake and the parameter for changes across time.
- phiMistakesBFixed A vector (of size number of PIVs) of booleans indicating whether the parameters for mistakes should be fixed in case of instability. It should be FALSE, except for unstable PIVs for which it may be set to TRUE in order to avoid estimability problems between the parameter for mistake and the parameter for changes across time.
- phiForMistakesA A vector (of size number of PIVs) of NA or fixed values for the parameters for mistakes. It should be NA, except for unstable PIVs for which one wants to fix the parameter to avoid estimability problem (as indicated with the boolean values in phiMistakesAFixed). In that case it should be set the the expected value for the probability of mistake. If you have no idea: you can put it to 0, the algorithm is quite robust to wrongly fixed parameters.
- phiForMistakesB A vector (of size number of PIVs) of NA or fixed values for the parameters for mistakes. It should be NA, except for unstable PIVs for which one wants to fix the parameter to avoid estimability problem (as indicated with the boolean values in phiMistakesBFixed). In that case it should be set the the expected value for the probability of mistake. If you have no idea: you can put it to 0, the algorithm is quite robust to wrongly fixed parameters.

`links` A matrix of 2 columns with indices of the linked records.

`survivalpSameH` a matrix of size (nrow=number of linked records, ncol=number of PIVs), filled for each PIV in column, with 1 if the PIV is stable and with the probability for true values of the records to coincide as calculate by survival function if the PIV is unstable.

`sumRowD` A boolean vector indicating, for each row of the linkage matrix, i.e. for each record in the smallest file A, whether the record has a link in B or not.

`sumColD` A boolean vector indicating, for each column of the linkage matrix, i.e. for each record in the largest file B, whether the record has a link in A or not.

eta	The distribution weights for the PIVs.
phi	The proportion of mistakes and missing for the PIVs.

Value

A list with 2 matrices of the shapes of both data sources, representing the true values of the PIVs underlying the registered values present in the data sources.

Examples

```
PIVs_config = list( V1 = list(stable = TRUE),
                    V2 = list(stable = TRUE),
                    V3 = list(stable = TRUE),
                    V4 = list(stable = TRUE),
                    V5 = list( stable = FALSE,
                               conditionalHazard = FALSE,
                               pSameH.cov.A = c(),
                               pSameH.cov.B = c()) )
PIVs = names(PIVs_config)
PIVs_stable = sapply(PIVs_config, function(x) x$stable)
Nval = c(6, 7, 8, 9, 15)
NRecords = c(13, 15)
Nlinks = 6
PmistakesA = c(0.02, 0.02, 0.02, 0.02, 0.02)
PmistakesB = c(0.02, 0.02, 0.02, 0.02, 0.02)
PmissingA = c(0.007, 0.007, 0.007, 0.007, 0.007)
PmissingB = c(0.007, 0.007, 0.007, 0.007, 0.007)
moving_params = list(V1=c(),V2=c(),V3=c(),V4=c(),V5=c(0.28))
enforceEstimability = TRUE
DATA = DataCreation( PIVs_config,
                     Nval,
                     NRecords,
                     Nlinks,
                     PmistakesA,
                     PmistakesB,
                     PmissingA,
                     PmissingB,
                     moving_params,
                     enforceEstimability)
A                    = DATA$A
B                    = DATA$B
Nvalues              = DATA$Nvalues

encodedA = A
encodedB = B

encodedA[,PIVs][ is.na(encodedA[,PIVs]) ] = 0
encodedB[,PIVs][ is.na(encodedB[,PIVs]) ] = 0

dataForStEM = list( A                    = encodedA,
                    B                    = encodedB,
                    Nvalues              = Nvalues,
```

```
                    PIVs_config          = PIVs_config,
                    controlOnMistakes    = c(TRUE, TRUE, TRUE, TRUE, TRUE),
                    sameMistakes         = TRUE,
                    phiMistakesAFixed    = TRUE,
                    phiMistakesBFixed    = TRUE,
                    phiForMistakesA      = c(NA, NA, NA, NA, 0),
                    phiForMistakesB      = c(NA, NA, NA, NA, 0)
)

initDeltaMap()
linksR = base::matrix(0,0,2)
linksCpp = linksR
sumRowD = rep(0, nrow(dataForStEM$A))
sumColD = rep(0, nrow(dataForStEM$B))
nlinkrec = 0
survivalpSameH = base::matrix(1, nrow(linksR), length(dataForStEM$Nvalues))
gamma = 0.5
eta = lapply(dataForStEM$Nvalues, function(x) rep(1/x,x))
phi = lapply(dataForStEM$Nvalues, function(x)  c(0.9,0.9,0.1,0.1))
nCoefUnstable = lapply( seq_along(PIVs_stable),
                        function(idx)
                        if(PIVs_stable[idx]){ 0 }
                        else{
                       ncol(dataForStEM$A[, dataForStEM$PIVs_config[[idx]]$pSameH.cov.A,
                             drop=FALSE]) +
                       ncol(dataForStEM$B[, dataForStEM$PIVs_config[[idx]]$pSameH.cov.B,
                             drop=FALSE]) +
                        1 } )
alpha = lapply( seq_along(PIVs_stable),
                function(idx) if(PIVs_stable[idx]){ c(-Inf) }else{
                  rep(log(0.05), nCoefUnstable[[idx]]) })
newTruePivs = simulateH(data=dataForStEM, links=linksCpp, survivalpSameH=survivalpSameH,
                        sumRowD=sumRowD, sumColD=sumColD, eta=eta, phi=phi)
truepivsA = newTruePivs$truepivsA
truepivsB = newTruePivs$truepivsB
```

`sspaste2` *sspaste2*

Description

sspaste2

Usage

```
sspaste2(A)
```

Arguments

`A`	IntegerMatrix with values to form patterns

Value

CharacterVector: "sspaste2(A)" is a faster version in Rcpp of "do.call(paste, c(as.data.frame(A), list(sep="_")))"

`stEM` *Stochastic Expectation Maximisation (StEM) for Record Linkage*

Description

Stochastic Expectation Maximisation (StEM) for Record Linkage

Usage

```
stEM(
  data,
  StEMIter,
  StEMBurnin,
  GibbsIter,
  GibbsBurnin,
  musicOn = TRUE,
  newDirectory = NULL,
  saveInfoIter = FALSE
)
```

Arguments

`data` A list with elements:

- A: the smallest data source (encoded: the categorical values of the Partially Identifying Variables (PIVs) have to be mapped to sets of natural numbers and missing values are encoded as 0).
- B: the largest data source (encoded).
- Nvalues: A vector (of size number of PIVs) with the number fo unique values per PIVs (in the order of the PIVs defined in PIVs_config).
- PIVs_config: A list (of size number of PIVs) where element names are the PIVs and element values are lists with elements: stable (boolean for whether the PIV is stable), conditionalHazard (boolean for whether there are external covariates available to model instability, only required if stable is FALSE), pSameH.cov.A and pSameH.covB (vectors with strings corresponding to the names of the covariates to use to model instability from file A and file B, only required if stable is FALSE, empty vectors may be provided if conditionalHazard is FALSE).
- controlOnMistakes: A vector (of size number of PIVs) of booleans indicating potential bounds on the mistakes probabilities for each PIV. For each PIV, if TRUE there will be control on mistake and the mistake probability will not go above 10%. If FALSE there is no bound on the probability of mistake. WATCH OUT, if you suspect that a variable is unstable but you

do not have data to model its dynamics the boolean value should be set to FALSE to allow the parameter for mistake to adapt for the instability. However if you model this instability, the boolean value should be set to TRUE to help the algorithm differenciate the mistakes from the changes over time.

- sameMistakes: A boolean value for whether there should be one parameter for the mistakes in A and B or whether each source should have its own parameter. Setting sameMistakes=TRUE is recommended in case of small data sources; the estimation with 2 parameters in that case will fail to capture the mistakes correctly while 1 parameter will be more adapted.
- phiMistakesAFixed A vector (of size number of PIVs) of booleans indicating whether the parameters for mistakes should be fixed in case of instability. It should be FALSE, except for unstable PIVs for which it may be set to TRUE in order to avoid estimability problems between the parameter for mistake and the parameter for changes across time.
- phiMistakesBFixed A vector (of size number of PIVs) of booleans indicating whether the parameters for mistakes should be fixed in case of instability. It should be FALSE, except for unstable PIVs for which it may be set to TRUE in order to avoid estimability problems between the parameter for mistake and the parameter for changes across time.
- phiForMistakesA A vector (of size number of PIVs) of NA or fixed values for the parameters for mistakes. It should be NA, except for unstable PIVs for which one wants to fix the parameter to avoid estimability problem (as indicated with the boolean values in phiMistakesAFixed). In that case it should be set the the expected value for the probability of mistake. If you have no idea: you can put it to 0, the algorithm is quite robust to wrongly fixed parameters.
- phiForMistakesB A vector (of size number of PIVs) of NA or fixed values for the parameters for mistakes. It should be NA, except for unstable PIVs for which one wants to fix the parameter to avoid estimability problem (as indicated with the boolean values in phiMistakesBFixed). In that case it should be set the the expected value for the probability of mistake. If you have no idea: you can put it to 0, the algorithm is quite robust to wrongly fixed parameters.

`StEMIter` An integer with the total number of iterations of the Stochastic EM algorithm (including the period to discard as burn-in)

`StEMBurnin` An integer with the number of iterations to discard as burn-in

`GibbsIter` An integer with the total number of iterations of the Gibbs sampler (done in each iteration of the StEM) (including the period to discard as burn-in)

`GibbsBurnin` An integer with the number of iterations to discard as burn-in

`musicOn` A boolean value, if TRUE the algorithm will play music at the end of the algorithm, useful if you have to wait for the record linkage to run and to act as an alarm when record linkage is done

`newDirectory` A NULL value or: A string with the name of (or path to) the directory (which should already exist) where to save the environment variables at the end of each iteration (useful when record linkage is very long, to not loose everything and not restart from scratch in case your computer shut downs before record linkage is finished)

saveInfoIter A boolean value to indicate whether you want the environment variables to be saved at the end of each iteration (useful when record linkage is very long, to not loose everything and not restart from scratch in case your computer shut downs before record linkage is finished)

Value

A list with:

- Delta, the summary of a sparse matrix, i.e. a data frame with 3 columns: the indices from the first data source A, the indices from the second data source B, the non-zero probability that the records associated with this pair of indices are linked (i.e. the posterior probabilities to be linked). One has to select the pairs where this probability > 0.5 to get a valid set of linked records, (this threshold on the linkage probability is necessary to ensure the one-to-one assignment constraint of record linkage stating that one record in one file can at most be linked to one record in the other file).
- gamma, a vector with the chain of the parameter gamma representing the proportion of linked records as a fraction of the smallest file,
- eta, a vector with the chain of the parameter eta representing the distribution of the PIVs,
- alpha, a vector with the chain of the parameter alpha representing the hazard coefficient of the model for instability,
- phi, a vector with the chain of the parameter phi representing the registration errors parameters).

There are more details to understand the method in our paper, or on the experiments repository of our paper, or in the vignettes.

Examples

```
PIVs_config = list( V1 = list(stable = TRUE),
                    V2 = list(stable = TRUE),
                    V3 = list(stable = TRUE),
                    V4 = list(stable = TRUE),
                    V5 = list( stable = FALSE,
                               conditionalHazard = FALSE,
                               pSameH.cov.A = c(),
                               pSameH.cov.B = c()) )
PIVs = names(PIVs_config)
PIVs_stable = sapply(PIVs_config, function(x) x$stable)
Nval = c(6, 7, 8, 9, 15)
NRecords = c(500, 800)
Nlinks = 300
PmistakesA = c(0.02, 0.02, 0.02, 0.02, 0.02)
PmistakesB = c(0.02, 0.02, 0.02, 0.02, 0.02)
PmissingA = c(0.007, 0.007, 0.007, 0.007, 0.007)
PmissingB = c(0.007, 0.007, 0.007, 0.007, 0.007)
moving_params = list(V1=c(),V2=c(),V3=c(),V4=c(),V5=c(0.28))
enforceEstimability = TRUE
DATA = DataCreation( PIVs_config,
                     Nval,
```

```
                    NRecords,
                    Nlinks,
                    PmistakesA,
                    PmistakesB,
                    PmissingA,
                    PmissingB,
                    moving_params,
                    enforceEstimability)
A                   = DATA$A
B                   = DATA$B
Nvalues             = DATA$Nvalues

encodedA = A
encodedB = B

encodedA[,PIVs][ is.na(encodedA[,PIVs]) ] = 0
encodedB[,PIVs][ is.na(encodedB[,PIVs]) ] = 0

data = list( A = encodedA,
             B = encodedB,
             Nvalues = Nvalues,
             PIVs_config = PIVs_config,
             controlOnMistakes = c(TRUE,TRUE,FALSE,FALSE,FALSE),
             sameMistakes = TRUE,
             phiMistakesAFixed = FALSE,
             phiMistakesBFixed = FALSE,
             phiForMistakesA = c(NA,NA,NA,NA,NA),
             phiForMistakesB = c(NA,NA,NA,NA,NA))
 fit = stEM( data = data,
             StEMIter = 50,
             StEMBurnin = 30,
             GibbsIter = 50,
             GibbsBurnin = 30,
             musicOn = TRUE,
             newDirectory = NULL,
             saveInfoIter = FALSE )
```

`SurvivalUnstable` *The survival function with exponential model*

Description

The survival function with exponential model (as proposed in our paper) is representing the probability that true values of a pair of records referring to the same entity coincide. For a linked pair of record (i,j) from sources A, B respectively, P(HAik = HBjk | t_ij, ...) = exp(- exp(X.ALPHA) t_ij). This function is only used if the PIV is unstable and evolve over time. If so the true values of a linked pair of records may not coincide. If you want to use a different survival function to model instability, you can change this function 'SurvivalUnstable' as well as the associated log(likelihood) function 'loglikSurvival'. Also see ?FlexRL::loglikSurvival.

Usage

```
SurvivalUnstable(Xlinksk, alphask, times)
```

Arguments

Xlinksk	A matrix with number of linked records rows and 1+cov in A+cov in B columns, with a first column filled with 1 (intercept), and following columns filled with the values of the covariates useful for modelling instability for the linked records
alphask	A vector of size 1+cov in A+cov in B, with as first element the baseline hazard and following elements being the coefficient of the conditional hazard associated with the covariates given in X
times	A vector of size number of linked records with the time gaps between the record from each sources

Details

The simplest model (without covariates) just writes P(HAik = HBjk | t_ij, ...) = exp(- exp(alpha) . t_ij) The more complex model (with covariates) writes P(HAik = HBjk | t_ij, ...) = exp(- exp(X.ALPHA) . t_ij) and uses a matrix X (nrow=nbr of linked records, ncol=1 + nbr of cov from A + nbr of cov from B) where the first column is filled with 1 (intercept) and the subsequent columns are the covariates values from source A and/or from source B to be used. The ALPHA in this case is a vector of parameters, the first one being associated with the intercept is the same one than for the simplest model, the subsequent ones are associated with the covariates from A and/or from B.

Value

A vector (for an unstable PIV) of size number of linked records with the probabilities that true values coincide (e.g. 1 - proba to move if the PIV is postal code) defined according to the survival function with exponential model proposed in the paper

Examples

```
nCoefUnstable = 1
intercept = rep(1,5)
cov_k = cbind( intercept )
times = c(0.001,0.2,1.3,1.5,2)
survivalpSameH = SurvivalUnstable(cov_k, log(0.28), times)
```

Index