Package ‘bstrl’

May 8, 2026

Title Bayesian Streaming Record Linkage

Version 1.0.2

Description Perform record linkage on streaming files using recursive Bayesian updating.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

Imports BRL, doParallel, extraDistr, foreach, parallel

Depends R (>= 2.10)

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Brenda Betancourt [aut],
Andee Kaplan [aut],
Ian Taylor [aut, cre]

Maintainer Ian Taylor <ian.taylor@colostate.edu>

Repository CRAN

Date/Publication 2022-11-10 19:10:08 UTC

Contents

alllinks 2
bipartiteRL 3
extractlinks 4
fromentities 5
geco_30over_3err 6
geco_small 6
geco_small_result 7
islinked 8
multifileRL 8
PPRBupdate 10

precision 11
recall 12
SMCMCupdate 13
thinsamples 14

Index **16**

`alllinks` *Return a list of all linked pairs (directly or transitively)*

Description

Return a list of all linked pairs (directly or transitively)

Usage

```
alllinks(sl, idx = c("global", "local"))
```

Arguments

`sl`	A streaming link object
`idx`	Whether to use global (default) or local indices

Value

A list with two (global) or four (local) members, defining linked pairs of records. If 'global' indices, these members are called 'idx1' and 'idx2', where corresponding entries are the global indices of matched pairs of records. If 'local' indices, these members are called 'file1', 'record1', 'file2', and 'record2', where corresponding entries are the file number and record number within the file of matched pairs of records.

Examples

```
data(geco_small_result)
samples <- extractlinks(geco_small_result)
# List all linked pairs of records in the 42nd posterior sample
alllinks(samples[[42]], idx="global")
alllinks(samples[[42]], idx="local")
```

bipartiteRL *Perform baseline bipartite record linkage before streaming updates*

Description

This function establishes a baseline linkage between two files which can be built upon with streaming updates adding more files. It outsources the linkage work to the BRL package and appends information to the object which will allow streaming record linkage to continue

Usage

```
bipartiteRL(
  df1,
  df2,
  flds = NULL,
  flds1 = NULL,
  flds2 = NULL,
  types = NULL,
  breaks = c(0, 0.25, 0.5),
  nIter = 1000,
  burn = round(nIter * 0.1),
  a = 1,
  b = 1,
  aBM = 1,
  bBM = 1,
  seed = 0
)
```

Arguments

df1, df2	Files 1 and 2 as dataframes where each row is a record and each column is a field.
flds	Names of the fields on which to compare the records in each file
flds1, flds2	Allows specifying field names differently for each file.
types	Types of comparisons to use for each field
breaks	Breaks to use for Levenshtein distance on string fields
nIter, burn	MCMC run length parameters. The returned number of samples is nIter - burn.
a, b	Prior parameters for m and u, respectively.
aBM, bBM	Prior parameters for beta-linkage prior.
seed	Random seed to set at beginning of MCMC run

Value

A list with class "bstrlstate" which can be passed to future streaming updates.

Examples

```
data(geco_small)

# Names of the columns on which to perform linkage
fieldnames <- c("given.name", "surname", "age", "occup",
                "extra1", "extra2", "extra3", "extra4", "extra5", "extra6")

# How to compare each of the fields
# First name and last name use normalized edit distance
# All others binary equal/unequal
types <- c("lv", "lv",
           "bi", "bi", "bi", "bi", "bi", "bi", "bi", "bi")
# Break continuous difference measures into 4 levels using these split points
breaks <- c(0, 0.25, 0.5)

res.twofile <- bipartiteRL(geco_small[[1]], geco_small[[2]],
                           flds = fieldnames, types = types, breaks = breaks,
                           nIter = 10, burn = 5, # Very small number of samples
                           seed = 0)
```

`extractlinks` *Extract the links from a bstrlstate object into a list of streaminglinks objects.*

Description

Extract the links from a bstrlstate object into a list of streaminglinks objects.

Usage

```
extractlinks(state, unfinished = c("warn", "ignore", "fail"))
```

Arguments

`state`	A bstrlstate object returned by one of the main RL functions.
`unfinished`	What to do if passed an unfinished sampling state, e.g. from multifileRL. See details.

Details

The function does one of three things when passed an unfinished sampling state, e.g. from multifileRL after time limit expired. For 'ignore', the desired burn is performed and any remaining samples are discarded, returning only the number of completed post-burn samples. If the link state does not have any completed post-burn samples, an empty list is returned. If 'warn' (default), the same action is performed as 'ignore' but a warning is issued. If 'fail', any unfinished link state will cause the function to fail.

Value

A list of streaminglinks objects, one per posterior sample contained in 'state'.

Examples

```
data(geco_small_result)
posterior <- extractlinks(geco_small_result, unfinished="fail")
stopifnot(ncol(geco_small_result$Z) == length(posterior))
class(posterior)
class(posterior[[1]])
```

---

fromentities — *Create a streaming link object from known record entity id's*

Description

Create a streaming link object from known record entity id's

Usage

```
fromentities(...)
```

Arguments

`...`	Vectors containing entity IDs of each record in each file. Each parameter represents the records in a file. The values are an entity ID for the records in that file. Two records having the same entity ID are coreferent.

Value

A streaming link object with S3 class 'streaminglinks', defining the links between records implied by the supplied entity IDs

Examples

```
data(geco_30over_3err)

file1 <- geco_30over_3err[[1]]
file2 <- geco_30over_3err[[2]]
file3 <- geco_30over_3err[[3]]
file4 <- geco_30over_3err[[4]]

fromentities(file1$entity, file2$entity, file3$entity, file4$entity)
```

geco_30over_3err *Simulated Noisy Records*

Description

A dataset containing several files of noisy simulated records. Records are simulated using GeCo (Tran, Vatsalan, and Cristen (2013)) and organized into files of 200 records each. The columns in each file consist of two ID columns for validating links:

Usage

```
data(geco_30over_3err)
```

Format

A list of 7 data.frames. Each data.frame has 200 rows and 16 columns.

Details

- rec.id. Contains the entity number and duplicate number of each record. This is unique to a record.
- entity. Contains the entity number of which this record is a copy. Is identical for all records which are noisy duplicates of the same original.

The columns also consist of fields used to perform linkage, into which 3 errors have been randomly inserted:

- given.name, surname. Text fields with potential typographical errors.
- age, occup, extra1, ..., extra10. Categorical fields with potential swapped category errors.

Linkage may be performed on either the full dataset or on only a subset of the fields.

See Also

geco_small

geco_small *Simulated Noisy Records (smaller set)*

Description

A dataset containing several files of noisy simulated records. Records are simulated using GeCo (Tran, Vatsalan, and Cristen (2013)) and organized into files of 10 records each. These files are subsets of the larger dataset. The columns in each file consist of two ID columns for validating links:

Usage

```
data(geco_small)
```

Format

A list of 7 data.frames. Each data.frame has 10 rows and 16 columns.

Details

- rec.id. Contains the entity number and duplicate number of each record. This is unique to a record.
- entity. Contains the entity number of which this record is a copy. Is identical for all records which are noisy duplicates of the same original.

The columns also consist of fields used to perform linkage, into which 3 errors have been randomly inserted:

- given.name, surname. Text fields with potential typographical errors.
- age, occup, extra1, ..., extra10. Categorical fields with potential swapped category errors.

Linkage may be performed on either the full dataset or on only a subset of the fields.

See Also

geco_30over_3err

---

geco_small_result *Example linkage result object for small dataset*

Description

This object is useful for examples and for testing code designed to process record linkage results. It is identical to the PPRB result obtained from 4 files in the 'streaming-rl-howto' vignette included with this package.

Usage

```
data(geco_small_result)
```

Format

An object of class 'bstrlstate', as is returned by the streaming update functions PPRBupdate, SMCMCupdate, multifileRL, or bipartiteRL.

See Also

geco_small, PPRBupdate

islinked — *Return True/False whether the two record are coreferent*

Description

Return True/False whether the two record are coreferent

Usage

```
islinked(sl, file1, record1, file2, record2)
```

Arguments

sl	A streaming link object
file1, record1	The file number and record number of the first record
file2, record2	The file number and record number of the second record. Note that file2 must be greater than file1.

Value

A boolean value. True if these two records are linked within sl, False otherwise.

Examples

```
data(geco_small_result)
samples <- extractlinks(geco_small_result)

# Are record 9 in file 1 and record 7 in file 4 linked in the first posterior sample?
islinked(samples[[1]], file1=1, record1=9, file2=4, record2=7)

# In what proportion of posterior samples are record 9 in file 1 and record 7 in file 4 linked?
mean(sapply(samples, islinked, file1=1, record1=9, file2=4, record2=7))

# In what proportion of posterior samples are record 8 in file 1 and record 1 in file 2 linked?
mean(sapply(samples, islinked, file1=1, record1=8, file2=2, record2=1))
```

multifileRL — *Perform multifile record linkage via Gibbs sampling "from scratch"*

Description

Perform multifile record linkage via Gibbs sampling "from scratch"

Usage

```
multifileRL(
  files,
  flds = NULL,
  types = NULL,
  breaks = c(0, 0.25, 0.5),
  nIter = 1000,
  burn = round(nIter * 0.1),
  a = 1,
  b = 1,
  aBM = 1,
  bBM = 1,
  proposals = c("component", "LB"),
  blocksize = NULL,
  seed = 0,
  refresh = 0.1,
  maxtime = Inf
)
```

Arguments

`files`	A list of files
`flds`	Vector of names of the fields on which to compare the records in each file
`types`	Types of comparisons to use for each field
`breaks`	Breaks to use for Levenshtein distance on string fields
`nIter`, `burn`	MCMC run length parameters. The returned number of samples is nIter - burn.
`a`, `b`	Prior parameters for m and u, respectively.
`aBM`, `bBM`	Prior parameters for beta-linkage prior.
`proposals`	Which kind of full conditional proposals to use for the link vectors.
`blocksize`	What blocksize to use for locally balanced proposals. By default, LB proposals are not blocked
`seed`	Random seed to set at beginning of MCMC run
`refresh`	How often to output an update including the iteration number and percent complete. If refresh >= 1, taken as a number of iterations between messages (rounded). If 0 < refresh < 1, taken as the proportion of nIter. If refresh == 0, no messages are displayed.
`maxtime`	Amount of time, in seconds, after which the sampler will terminate with however many samples it has produced up to that point. The sample matrix columns for any unproduced samples will be filled with NAs

Value

An object of class "bstrlstate"

Examples

```
data(geco_small)

# Names of the columns on which to perform linkage
fieldnames <- c("given.name", "surname", "age", "occup",
                "extra1", "extra2", "extra3", "extra4", "extra5", "extra6")

# How to compare each of the fields
# First name and last name use normalized edit distance
# All others binary equal/unequal
types <- c("lv", "lv",
           "bi", "bi", "bi", "bi", "bi", "bi", "bi", "bi")
# Break continuous difference measures into 4 levels using these split points
breaks <- c(0, 0.25, 0.5)

# Three file linkage using first three files in example dataset
multifile.result <- multifileRL(geco_small[1:3],
                                flds = fieldnames, types = types, breaks = breaks,
                                nIter = 2, burn = 1, # Very small run for example
                                proposals = "comp",
                                seed = 0)
```

PPRBupdate *Perform a PPRB update of record linkage with a new file*

Description

Perform a PPRB update of record linkage with a new file

Usage

```
PPRBupdate(
  state,
  newfile,
  flds = NULL,
  nIter = NULL,
  burn = 0,
  blocksize = NULL,
  threestep = TRUE,
  seed = 0,
  refresh = 0.1
)
```

Arguments

state	Existing record linkage state. Returned by either bipartiteRL, PPRBupdate, SMCMCupdate, or multifileRL.

newfile	A data.frame representing the new file: one row per record
flds	Names of fields in the new file to use for comparison. Only used if no global field names were specified in bipartiteRL initially.
nIter	Number of iterations for which to run the PPRB sampler. By default, this is the same as the number of samples present in 'state'.
burn	Number of initial iterations to discard. The total number of samples returned is nIter - burn.
blocksize	Size of blocks to use for locally balanced proposals. Default performs unblocked locally balanced proposals.
threestep	Whether to perform three Gibbs sampling steps per iteration, with past Z's updated with PPRB, m and u updated with full conditionals, and the current Z updated with locally balanced proposals. If false, a two step Gibbs sampler is used where past Z's, m and u are updated together using PPRB and the current Z is updated with locally balanced proposals
seed	Random seed to set at the beginning of the MCMC run
refresh	How often to output an update including the iteration number and percent complete. If refresh >= 1, taken as a number of iterations between messages (rounded). If 0 < refresh < 1, taken as the proportion of nIter. If refresh == 0, no messages are displayed.

Value

An object of class 'bstrlstate' containing posterior samples and necessary metadata for passing to future streaming updates.

Examples

```
data(geco_small)
data(geco_small_result)

# Add fifth file to previous four-file link result
file5.result <- PPRBupdate(geco_small_result, geco_small[[5]],
                           nIter=2, burn=1) # Very small run for example
```

---

precision — *Calculate the precision of estimated links relative to true links*

Description

Calculate the precision of estimated links relative to true links

Usage

```
precision(sl.est, sl.true)
```

Arguments

`sl.est`	streaminglinks object representing link estimates
`sl.true`	streaminglinks object representing true links

Value

The precision of the estimated links.

Examples

```
data(geco_small)
data(geco_small_result)

sl.true <- fromentities(geco_small[[1]]$entity, geco_small[[2]]$entity,
                        geco_small[[3]]$entity, geco_small[[4]]$entity)

posterior <- extractlinks(geco_small_result)
# Compare one posterior sample to previously computed known truth
class(sl.true)
precision(posterior[[42]], sl.true)
```

`recall` *Calculate the recall of estimated links relative to true links*

Description

Calculate the recall of estimated links relative to true links

Usage

```
recall(sl.est, sl.true)
```

Arguments

`sl.est`	streaminglinks object representing link estimates
`sl.true`	streaminglinks object representing true links

Value

The recall of the estimated links.

Examples

```
data(geco_small)
data(geco_small_result)

sl.true <- fromentities(geco_small[[1]]$entity, geco_small[[2]]$entity,
                        geco_small[[3]]$entity, geco_small[[4]]$entity)

posterior <- extractlinks(geco_small_result)
# Compare one posterior sample to previously computed known truth
class(sl.true)
recall(posterior[[42]], sl.true)
```

SMCMCupdate *Perform an SMCMC update of record linkage with a new file*

Description

Perform an SMCMC update of record linkage with a new file

Usage

```
SMCMCupdate(
  state,
  newfile,
  flds = NULL,
  nIter.jumping = 5,
  nIter.transition = 10,
  cores = 1,
  proposals.jumping = c("component", "LB"),
  proposals.transition = c("LB", "component"),
  blocksize = NULL,
  seed = 0
)
```

Arguments

`state` Existing record linkage state. Returned by either bipartiteRL, PPRBupdate, SMCMCupdate, or multifileRL. This is used as the ensemble of samples in SMCMC update

`newfile` A data.frame representing the new file: one row per record

`flds` Names of fields in the new file to use for comparison. Only used if no global field names were specified in bipartiteRL initially.

`nIter.jumping, nIter.transition` Number of iterations to use in the jumping kernel and transition kernel, respectively, for each ensemble sample.

`cores` Number of cores to use for parallel execution. If cores == 1, update is run sequentially. A cluster is created using parallel::makeCluster().

`proposals.jumping`, `proposals.transition` Which kernel to use for Z updates in the jumping and transition kernels, respectively.

`blocksize` Size of blocks to use for locally balanced proposals. Default performs unblocked locally balanced proposals.

`seed` Random seed to set at the beginning of the MCMC run. This is ignored if cores > 1.

Value

An object of class 'bstrlstate' containing posterior samples and necessary metadata for passing to future streaming updates.

Examples

```
data(geco_small)
data(geco_small_result)

# Add fifth file to previous four-file link result
filtered <- thinsamples(geco_small_result, 2) # Filter ensemble to 2 - very small for example
file5.result <- SMCMCupdate(filtered, geco_small[[5]],
                        nIter.jumping=1, nIter.transition=1, # Very small run for example
                            proposals.jumping="LB", proposals.transition="LB",
                            blocksize=5)
```

`thinsamples` *Thin a bstrlstate object*

Description

Thin a bstrlstate object by keeping only one sample every n, until the desired count remains.

Usage

```
thinsamples(state, count)
```

Arguments

`state` Object of class bstrlstate, output by bipartiteRL, SMCMCupdate, PPRBupdate, or multifileRL

`count` The number of desired samples after filtering

Details

This is useful to do before an SMCMC update. SMCMC produces independent samples, so fewer are required to get the same quality estimates.

Value

An object of class bstrlstate, containing count samples.

Examples

```
data(geco_small_result)
filtered <- thinsamples(geco_small_result, 50)
stopifnot(ncol(filtered$Z) == 50)
```

Index

∗ **datasets**
    geco_30over_3err, 6
    geco_small, 6
    geco_small_result, 7

alllinks, 2

bipartiteRL, 3

extractlinks, 4

fromentities, 5

geco_30over_3err, 6
geco_small, 6
geco_small_result, 7

islinked, 8

multifileRL, 8

PPRBupdate, 10
precision, 11

recall, 12

SMCMCupdate, 13

thinsamples, 14