Package: BRL (via r-universe)

September 7, 2024

Title Beta Record Linkage

Version 0.1.0

Description Implementation of the record linkage methodology proposed by Sadinle (2017) <doi:10.1080/01621459.2016.1148612>. It handles the bipartite record linkage problem, where two duplicate-free datafiles are to be merged.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports utils

RoxygenNote 7.0.2

URL https://github.com/msadinle/BRL

BugReports https://github.com/msadinle/BRL/issues

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

RemoteUrl https://github.com/msadinle/brl

RemoteRef HEAD

RemoteSha 7b8b37d5b76e2cc8d5efcc313ef21986b99c2d54

Contents

`bipartiteGibbs` *Gibbs Sampler Used for Beta Record Linkage*

Description

Run a Gibbs sampler to explore the posterior distribution of bipartite matchings that represent the linkage of the datafiles in beta record linkage.

Usage

```
bipartiteGibbs(cd, nIter = 1000, a = 1, b = 1, aBM = 1, bBM = 1, seed = 0)
```

Arguments

`cd`	a list with the same structure as the output of the function `compareRecords`, containing: `comparisons` matrix with `n1*n2` rows, where the comparison pattern for record pair (i, j) appears in row `(j-1)*n1+i`, for i in $1, \ldots, n1$, and j in $1, \ldots, n2$. A comparison field with $L + 1$ levels of disagreement, is represented by $L+1$ columns of TRUE/FALSE indicators. Missing comparisons are coded as FALSE, which is justified under an assumption of ignorability of the missing comparisons, see Sadinle (2017). `n1,n2` the datafile sizes, `n1 = nrow(df1)` and `n2 = nrow(df2)`. `nDisagLevs` a vector containing the number of levels of disagreement per comparison field. `compFields` a data frame containing the names of the fields in the datafiles used in the comparisons and the types of comparison.
`nIter`	number of iterations of Gibbs sampler.
`a, b`	hyper-parameters of the Dirichlet priors for the m and u parameters in the model for the comparison data among matches and non-matches, respectively. These can be vectors with as many entries as disagreement levels among all comparison fields. If specified as positive constants, they get recycled to the required length. If not specified, flat priors are taken.
`aBM, bBM`	hyper-parameters of beta prior on bipartite matchings. Default is `aBM=bBM=1`.
`seed`	seed to be used for pseudo-random number generation. By default it sets `seed=0`.

Value

a list containing:

`Z` matrix with `n2` rows and `nIter` columns containing the chain of bipartite matchings. A number smaller or equal to `n1` in row `j` indicates the record in datafile 1 to which record `j` in datafile 2 is linked at that iteration, otherwise `n1+j`.

`m,u` chain of m and u parameters in the model for the comparison data among matches and non-matches, respectively.

References

Mauricio Sadinle (2017). Bayesian Estimation of Bipartite Matchings for Record Linkage. *Journal of the American Statistical Association* 112(518), 600-612. [Published] [arXiv]

Examples

```
data(twoFiles)

myCompData <- compareRecords(df1, df2, flds=c("gname", "fname", "age", "occup"),
                             types=c("lv","lv","bi","bi"))

chain <- bipartiteGibbs(myCompData)
```

BRL *Beta Record Linkage*

Description

Beta record linkage methodology for probabilistic bipartite record linkage: the task of merging two duplicate-free datafiles that lack unique identifiers. This function runs all the steps of beta record linkage: creates comparisons of the records, runs Gibbs sampler on bipartite matchings, and derives point estimate of bipartite matching (this determines the final linkage). The parameters of BRL consist of all the parameters needed to run compareRecords, bipartiteGibbs and linkRecords, except for intermediate input/output, and in addition to a parameter burn for the burn-in period of the Gibbs sampler.

Usage

```
BRL(
  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,
  lFNM = 1,
  lFM1 = 1,
  lFM2 = 2,
  lR = Inf
)
```

Arguments

df1, df2	two datasets to be linked, of class data.frame, with rows representing records and columns representing fields. Without loss of generality, df1 is assumed to have no less records than df2.
flds	a vector indicating the fields to be used in the linkage. Either a character vector, in which case all entries need to be names of columns of df1 and df2, or a numeric vector indicating the columns in df1 and df2 to be used in the linkage. If provided as a numeric vector it is assumed that the columns of df1 and df2 are organized such that it makes sense to compare the columns df1[,flds] and df2[,flds] in that order.
flds1, flds2	vectors indicating the fields of df1 and df2 to be used in the linkage. Either character vectors, in which case all entries need to be names of columns of df1 and df2, respectively, or numeric vectors indicating the columns in df1 and df2 to be used in the linkage. It is assumed that it makes sense to compare the columns df1[,flds1] and df2[,flds2] in that order. These arguments are ignored if flds is specified. If none of flds,flds1,flds2 are specified, the columns with the same names in df1 and df2 are compared, if any.
types	a vector of characters indicating the comparison type per comparison field. The options are: "lv" for comparisons based on the Levenshtein edit distance normalized to $[0, 1]$, with 0 indicating no disagreement and 1 indicating maximum disagreement; "bi" for binary comparisons (agreement/disagreement); "nu" for numeric comparisons computed as the absolute difference. The default is "lv". Fields compared with the "lv" option are first transformed to character class. Factors with different levels compared using the "bi" option are transformed to factors with the union of the levels. Fields compared with the "nu" option need to be of class numeric.
breaks	break points for the comparisons to obtain levels of disagreement. It can be a list of length equal to the number of comparison fields, containing one numeric vector with the break points for each comparison field, where entries corresponding to comparison type "bi" are ignored. It can also be a named list of length two with elements 'lv' and 'nu' containing numeric vectors with the break points for all Levenshtein-based and numeric comparisons, respectively. Finally, it can be a numeric vector with the break points for all comparison fields of type "lv" and "nu", which might be meaningful only if all the non-binary comparisons are of a single type, either "lv" or "nu". For comparisons based on the normalized Levenshtein distance, a vector of length L of break points for the interval $[0, 1]$ leads to $L + 1$ levels of disagreement. Similarly, for comparisons based on the absolute difference, the break points are for the interval $[0, \infty)$. The default is breaks=c(0,.25,.5), which might be meaningful only for comparisons of type "lv".
nIter	number of iterations of Gibbs sampler.
burn	number of iterations to discard as part of the burn-in period.
a, b	hyper-parameters of the Dirichlet priors for the m and u parameters in the model for the comparison data among matches and non-matches, respectively. These can be vectors with as many entries as disagreement levels among all comparison fields. If specified as positive constants, they get recycled to the required length. If not specified, flat priors are taken.

aBM, bBM	hyper-parameters of beta prior on bipartite matchings. Default is aBM=bBM=1.
seed	seed to be used for pseudo-random number generation. By default it sets seed=0.
lFNM	individual loss of a false non-match in the loss functions of Sadinle (2017), default lFNM=1.
lFM1	individual loss of a false match of type 1 in the loss functions of Sadinle (2017), default lFM1=1.
lFM2	individual loss of a false match of type 2 in the loss functions of Sadinle (2017), default lFM2=2.
lR	individual loss of 'rejecting' to make a decision in the loss functions of Sadinle (2017), default lR=Inf.

Details

Beta record linkage (BRL, Sadinle, 2017) is a methodology for probabilistic bipartite record linkage, that is, the task of merging two duplicate-free datafiles that lack unique identifiers. This is accomplished by using the common partially identifying information of the entities contained in the datafiles. The duplicate-free requirement means that we expect each entity to be represented maximum once in each datafile. This methodology should not be used with datafiles that contain duplicates nor should it be used for deduplicating a single datafile.

The first step of BRL, accomplished by the function compareRecords, consists of constructing comparison vectors for each pair of records from the two datafiles. The current implementation allows binary comparisons (agree/disagree), numerical comparisons based on the absolute difference, and Levenshtein-based comparisons. This can be easily extended to other comparison types, so a resourceful user should be able to construct an object that recreates the output of compareRecords for other types of comparisons (so long as they get transformed to levels of disagreement), and still be able to run the next step outside the function BRL.

The second step of BRL, accomplished by the function bipartiteGibbs, consists of running a Gibbs sampler that explores the space of bipartite matchings representing the plausible ways of linking the datafiles. This sampler is derived from a model for the comparison data and a *beta* prior distribution on the space of bipartite matchings. See Sadinle (2017) for details.

The third step of BRL, accomplished by the function linkRecords, consists of deriving a point estimate of the bipartite matching (which gives us the optimal way of linking the datafiles) by minimizing the expected value of a loss function that uses different penalties for different types of linkage errors. The current implementation only supports the Bayes point estimates of bipartite matchings that can be obtained in closed form according to Theorems 1, 2 and 3 of Sadinle (2017). The losses have to be positive numbers and satisfy one of three conditions:

1. Conditions of Theorem 1 of Sadinle (2017): `(lR == Inf) & (lFNM <= lFM1) & (lFNM + lFM1 <= lFM2)`
2. Conditions of Theorem 2 of Sadinle (2017): `((lFM2 >= lFM1) & (lFM1 >= 2*lR)) | ((lFM1 >= lFNM) & (lFM2 >= lFM1 + lFNM))`
3. Conditions of Theorem 3 of Sadinle (2017): `(lFM2 >= lFM1) & (lFM1 >= 2*lR) & (lFNM >= 2*lR)`

If one of the last two conditions is satisfied, the point estimate might be partial, meaning that there might be some records in datafile 2 for which the point estimate does not include a linkage decision. For combinations of losses not supported here, the linear sum assignment problem outlined by Sadinle (2017) needs to be solved.

Value

A vector containing the point estimate of the bipartite matching, as in the output of `linkRecords`. If `lR != Inf` the output might be a partial estimate. A number smaller or equal to `n1` in entry `j` indicates the record in datafile 1 to which record `j` in datafile 2 gets linked, a number `n1+j` indicates that record `j` does not get linked to any record in datafile 1, and the value `-1` indicates a 'rejection' to link, meaning that the correct linkage decision is not clear.

References

Mauricio Sadinle (2017). Bayesian Estimation of Bipartite Matchings for Record Linkage. *Journal of the American Statistical Association* 112(518), 600-612. [Published] [arXiv]

See Also

`compareRecords` for examples on how to work with different types of comparison data, `bipartiteGibbs` for Gibbs sampler on bipartite matchings, and `linkRecords` for examples on full and partial point estimates of the true bipartite matching that indicates which records to link.

Examples

```
data(twoFiles)

(Zhat <- BRL(df1, df2, flds=c("gname", "fname", "age", "occup"),
    types=c("lv","lv","bi","bi")))

n1 <- nrow(df1)

Ztrue <- df2ID

## number of links (estimated matches)
nLinks <- sum(Zhat <= n1)

## number of actual matches according to the ground truth
nMatches <- sum(Ztrue <= n1)

## number of links that are actual matches
nCorrectLinks <- sum(Zhat[Zhat<=n1]==Ztrue[Zhat<=n1])

## compute measures of performance

## precision
nCorrectLinks/nLinks

## recall
nCorrectLinks/nMatches

## the linked record pairs
cbind( df1[Zhat[Zhat<=n1],], df2[Zhat<=n1,] )

## finally, note that we could run BRL step by step as follows
```

```
## create comparison data
myCompData <- compareRecords(df1, df2,
                             flds=c("gname", "fname", "age", "occup"),
                             types=c("lv","lv","bi","bi"))

## Gibbs sampling from posterior of bipartite matchings
chain <- bipartiteGibbs(myCompData)

## bipartite matching Bayes estimate derived from the loss functions of Sadinle (2017)
Zhat2 <- linkRecords(chain$Z, n1=n1)

identical(Zhat, Zhat2)
```

`compareRecords` *Creation of Comparison Data*

Description

Create comparison vectors for all pairs of records coming from two datafiles to be linked.

Usage

```
compareRecords(
  df1,
  df2,
  flds = NULL,
  flds1 = NULL,
  flds2 = NULL,
  types = NULL,
  breaks = c(0, 0.25, 0.5)
)
```

Arguments

`df1`, `df2`	two datasets to be linked, of class `data.frame`, with rows representing records and columns representing fields. Without loss of generality, `df1` is assumed to have no less records than `df2`.
`flds`	a vector indicating the fields to be used in the linkage. Either a `character` vector, in which case all entries need to be names of columns of `df1` and `df2`, or a `numeric` vector indicating the columns in `df1` and `df2` to be used in the linkage. If provided as a `numeric` vector it is assumed that the columns of `df1` and `df2` are organized such that it makes sense to compare the columns `df1[,flds]` and `df2[,flds]` in that order.
`flds1`, `flds2`	vectors indicating the fields of `df1` and `df2` to be used in the linkage. Either `character` vectors, in which case all entries need to be names of columns of `df1` and `df2`, respectively, or `numeric` vectors indicating the columns in `df1`

and df2 to be used in the linkage. It is assumed that it makes sense to compare the columns df1[,flds1] and df2[,flds2] in that order. These arguments are ignored if flds is specified. If none of flds,flds1,flds2 are specified, the columns with the same names in df1 and df2 are compared, if any.

types — a vector of characters indicating the comparison type per comparison field. The options are: "lv" for comparisons based on the Levenshtein edit distance normalized to $[0, 1]$, with 0 indicating no disagreement and 1 indicating maximum disagreement; "bi" for binary comparisons (agreement/disagreement); "nu" for numeric comparisons computed as the absolute difference. The default is "lv". Fields compared with the "lv" option are first transformed to character class. Factors with different levels compared using the "bi" option are transformed to factors with the union of the levels. Fields compared with the "nu" option need to be of class numeric.

breaks — break points for the comparisons to obtain levels of disagreement. It can be a list of length equal to the number of comparison fields, containing one numeric vector with the break points for each comparison field, where entries corresponding to comparison type "bi" are ignored. It can also be a named list of length two with elements 'lv' and 'nu' containing numeric vectors with the break points for all Levenshtein-based and numeric comparisons, respectively. Finally, it can be a numeric vector with the break points for all comparison fields of type "lv" and "nu", which might be meaningful only if all the non-binary comparisons are of a single type, either "lv" or "nu". For comparisons based on the normalized Levenshtein distance, a vector of length L of break points for the interval $[0, 1]$ leads to $L + 1$ levels of disagreement. Similarly, for comparisons based on the absolute difference, the break points are for the interval $[0, \infty)$. The default is breaks=c(0,.25,.5), which might be meaningful only for comparisons of type "lv".

Value

a list containing:

comparisons matrix with n1*n2 rows, where the comparison pattern for record pair (i, j) appears in row (j-1)*n1+i, for i in $1, \ldots, n1$, and j in $1, \ldots, n2$. A comparison field with $L + 1$ levels of disagreement, is represented by $L + 1$ columns of TRUE/FALSE indicators. Missing comparisons are coded as FALSE, which is justified under an assumption of ignorability of the missing comparisons, see Sadinle (2017).

n1,n2 the datafile sizes, n1 = nrow(df1) and n2 = nrow(df2).

nDisagLevs a vector containing the number of levels of disagreement per comparison field.

compFields a data frame containing the names of the fields in the datafiles used in the comparisons and the types of comparison.

References

Mauricio Sadinle (2017). Bayesian Estimation of Bipartite Matchings for Record Linkage. *Journal of the American Statistical Association* 112(518), 600-612. [Published] [arXiv]

Examples

```
data(twoFiles)

myCompData <- compareRecords(df1, df2,
                             flds=c("gname", "fname", "age", "occup"),
                             types=c("lv","lv","bi","bi"),
                             breaks=c(0,.25,.5))

## same as
myCompData <- compareRecords(df1, df2, types=c("lv","lv","bi","bi"))


## let's transform 'occup' to numeric to illustrate how to obtain numeric comparisons
df1$occup <- as.numeric(df1$occup)
df2$occup <- as.numeric(df2$occup)

## using different break points for 'lv' and 'nu' comparisons
myCompData1 <- compareRecords(df1, df2,
                              flds=c("gname", "fname", "age", "occup"),
                              types=c("lv","lv","bi","nu"),
                              breaks=list(lv=c(0,.25,.5), nu=0:3))

## using different break points for each comparison field
myCompData2 <- compareRecords(df1, df2,
                              flds=c("gname", "fname", "age", "occup"),
                              types=c("lv","lv","bi","nu"),
                              breaks=list(c(0,.25,.5), c(0,.2,.4,.6), NULL, 0:3))
```

`linkRecords` *Bayes Estimates of Bipartite Matchings*

Description

Bayes point estimates of bipartite matchings that can be obtained in closed form according to Theorems 1, 2 and 3 of Sadinle (2017).

Usage

```
linkRecords(Zchain, n1, lFNM = 1, lFM1 = 1, lFM2 = 2, lR = Inf)
```

Arguments

`Zchain`	matrix as the output `$Z` of the function `bipartiteGibbs`, with `n2` rows and `nIter` columns containing a chain of draws from a posterior distribution on bipartite matchings. Each column indicates the records in datafile 1 to which the records in datafile 2 are matched according to that draw.
`n1`	number of records in datafile 1.
`lFNM`	individual loss of a false non-match in the loss functions of Sadinle (2017), default `lFNM=1`.

lFM1	individual loss of a false match of type 1 in the loss functions of Sadinle (2017), default lFM1=1.
lFM2	individual loss of a false match of type 2 in the loss functions of Sadinle (2017), default lFM2=2.
lR	individual loss of 'rejecting' to make a decision in the loss functions of Sadinle (2017), default lR=Inf.

Details

Not all combinations of losses lFNM, lFM1, lFM2, lR are supported. The losses have to be positive numbers and satisfy one of three conditions:

1. Conditions of Theorem 1 of Sadinle (2017): (lR == Inf) & (lFNM <= lFM1) & (lFNM + lFM1 <= lFM2)
2. Conditions of Theorem 2 of Sadinle (2017): ((lFM2 >= lFM1) & (lFM1 >= 2*lR)) | ((lFM1 >= lFNM) & (lFM2 >= lFM1 + lFNM))
3. Conditions of Theorem 3 of Sadinle (2017): (lFM2 >= lFM1) & (lFM1 >= 2*lR) & (lFNM >= 2*lR)

If one of the last two conditions is satisfied, the point estimate might be partial, meaning that there might be some records in datafile 2 for which the point estimate does not include a linkage decision. For combinations of losses not supported here, the linear sum assignment problem outlined by Sadinle (2017) needs to be solved.

Value

A vector containing the point estimate of the bipartite matching. If lR != Inf the output might be a partial estimate. A number smaller or equal to n1 in entry j indicates the record in datafile 1 to which record j in datafile 2 gets linked, a number n1+j indicates that record j does not get linked to any record in datafile 1, and the value -1 indicates a 'rejection' to link, meaning that the correct linkage decision is not clear.

References

Mauricio Sadinle (2017). Bayesian Estimation of Bipartite Matchings for Record Linkage. *Journal of the American Statistical Association* 112(518), 600-612. [Published] [arXiv]

Examples

```
data(twoFiles)

myCompData <- compareRecords(df1, df2, flds=c("gname", "fname", "age", "occup"),
                             types=c("lv","lv","bi","bi"))

chain <- bipartiteGibbs(myCompData)

## discard first 100 iterations of Gibbs sampler

## full estimate of bipartite matching (full linkage)
fullZhat <- linkRecords(chain$Z[,-c(1:100)], n1=nrow(df1), lFNM=1, lFM1=1, lFM2=2, lR=Inf)
```

```
## partial estimate of bipartite matching (partial linkage), where
## lR=0.5, lFNM=1, lFM1=1 mean that we consider not making a decision for
## a record as being half as bad as a false non-match or a false match of type 1
partialZhat <- linkRecords(chain$Z[,-c(1:100)], n1=nrow(df1), lFNM=1, lFM1=1, lFM2=2, lR=.5)

## for which records the decision is not clear according to this set-up of the losses?
undecided <- which(partialZhat == -1)
df2[undecided,]

## compute frequencies of link options observed in the chain
linkOptions <- apply(chain$Z[undecided, -c(1:100)], 1, table)
linkOptions <- lapply(linkOptions, sort, decreasing=TRUE)
linkOptionsInds <- lapply(linkOptions, names)
linkOptionsInds <- lapply(linkOptionsInds, as.numeric)
linkOptionsFreqs <- lapply(linkOptions, function(x) as.numeric(x)/sum(x))

## first record without decision
df2[undecided[1],]

## options for this record; row of NAs indicates possibility that record has no match in df1
cbind(df1[linkOptionsInds[[1]],], prob = round(linkOptionsFreqs[[1]],3) )
```

`twoFiles` *Two Datasets for Record Linkage*

Description

Two data frames, `df1` and `df2`, containing 300 and 150 records of artificially created individuals, where 50 of them are included in both datafiles. In addition, the vector `df2ID` contains one entry per record in `df2` indicating the true matching between the datafiles, codified as follows: a number smaller or equal to `n1=300` in entry `j` indicates the record in `df1` to which record `j` in `df2` truly matches, and a number `n1+j` indicates that record `j` in `df2` does not match any record in `df1`.

Usage

```
data(twoFiles)
```

Source

Extracted from the datafiles used in the simulation studies of Sadinle (2017). The datafiles were originally generated using code provided by Peter Christen (https://users.cecs.anu.edu.au/~Peter.Christen/).

References

Mauricio Sadinle (2017). Bayesian Estimation of Bipartite Matchings for Record Linkage. *Journal of the American Statistical Association* 112(518), 600-612. [Published] [arXiv]

Examples

```
data(twoFiles)

n1 <- nrow(df1)

## the true matches
cbind( df1[df2ID[df2ID<=n1],], df2[df2ID<=n1,] )

## alternatively
df1$ID <- 1:n1
df2$ID <- df2ID
merge(df1, df2, by="ID")

## all the records in a merged file
merge(df1, df2, by="ID", all=TRUE)
```

Index