Package ‘Perc’

December 5, 2016

Type Package
Title Using Percolation and Conductance to Find Information Flow Certainty in a Direct Network
Version 0.1.2
Date 2016-05-04
Description To find the certainty of dominance interactions with indirect interactions being considered.
Depends R (>= 2.14.0)
License GPL (>= 2)
Copyright Fushing Lab & McCowan Lab, University of California, Davis
LazyData true
Imports stats, grDevices
Suggests testthat, knitr, devtools, reshape2, lattice
VignetteBuilder knitr
RoxygenNote 5.0.1
NeedsCompilation no
Author Kevin Fujii [aut], 
     Jian Jin [aut, cre],
     Aaron Shev [aut],
     Brianne Beisner [aut],
     Brenda McCowan [aut, cph],
     Hsieh Fushing [aut, cph]
Maintainer Jian Jin <jinjian.pku@gmail.com>
Repository CRAN
Date/Publication 2016-12-05 18:28:46

R topics documented:

  as.conflictmat .......................................................... 2
  bradleyTerry .......................................................... 3
**as.conflictmat**

*convert to a matrix of conf.mat class*

**Description**

as.conflictmat convert an edgelist or a win-loss raw matrix to a matrix of conf.mat class

**Usage**

as.conflictmat(Data, weighted = FALSE, swap.order = FALSE)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>either a dataframe or a matrix, representing raw win-loss interactions using either an edgelist or a matrix. By default, winners are represented by IDs in the 1st column for an edgelist, and by row IDs for a matrix. Frequency of interactions for each dyad can be represented either by multiple occurrences of the dyad for a 2-column edgelist, or by a third column specifying the frequency of the interaction for a 3-column edgelist.</td>
</tr>
<tr>
<td>weighted</td>
<td>If the edgelist is a 3-column edgelist in which weight was specified by frequency, use weighted = TRUE.</td>
</tr>
<tr>
<td>swap.order</td>
<td>If the winner is placed in the 2nd column for an edgelist or as the column name for a matrix, specify as TRUE. By default, winners are placed in the first column of an edgelist or in the row names of a matrix.</td>
</tr>
</tbody>
</table>
Details

conf.mat is short for "Conflict Matrix". conf.mat is a class of R objects. It is required to use as.conflictmat to convert your raw edgelist or raw win-loss matrix into a matrix of conf.mat object before using other functions to find (in)direct pathways and computing dominance probabilities.

Note, when using a 3-column edgelist (e.g. a weighted edgelist) to represent raw win-loss interactions, each dyad must be unique. If more than one rows are found with the same initiator and recipient, sum of the frequencies will be taken to represent the frequency of interactions between this unique dyad. A warning message will prompt your attention to the accuracy of your raw data when duplicate dyads were found in a three-column edgelist.

Value

a named matrix with the \([i,j]th\) entry equal to the number of times \(i\) wins over \(j\).

See Also

findIDpaths, countPaths, transitivity, conductance

Examples

confmatrix <- as.conflictmat(sampleEdgelist, swap.order = FALSE)
confmatrix2 <- as.conflictmat(sampleRawMatrix, swap.order = FALSE)
confmatrix3 <- as.conflictmat(sampleWeightedEdgelist, weighted = TRUE, swap.order = FALSE)

bradleyTerry

Computes the MLE for the BT model using an MM algorithm

Description

bradleyTerry Computes the MLE for the BT model using an MM algorithm

Usage

bradleyTerry(conf.mat, initial = NA, baseline = NA, stop.dif = 0.001)

Arguments

conf.mat a matrix of conf.mat class. An N-by-N conflict matrix whose \((i,j)th\) element is the number of times \(i\) defeated \(j\).

initial initial values of dominance indices for the MM algorithm, if not supplied, the 0 vector will be the initial value.

baseline index for agent to represent baseline dominance index set to 0. If NA, the "sum-to-one" parameterization will be used.

stop.dif numeric value for difference in log likelihood value between iterations. Used as the convergence criterion for the algorithm.
Details

In order to meet Bradley-Terry assumption, each ID in confNmat should have at least one win AND one loss. bradleyTerry will return an error if no more than one win or loss was found.

@references Shev, A., Hsieh, F., Beisner, B., & McCowan, B. (2012). Using Markov chain Monte Carlo (MCMC) to visualize and test the linearity assumption of the Bradley-Terry class of models. Animal behaviour, 84(6), 1523-1531.


Value

A list of length 3.

- domInds: a vector of length N consisting of the MLE values of the dominance indices. Lower values represent lower ranks.
- probMat: an N-by-N numeric matrix of win-loss probabilities estimated by the BT model.
- logLik: the model fit.

Examples

```r
# create an edgelist
edgelist1 <- data.frame(col1 = sample(letters[1:15], 200, replace = TRUE),
                        col2 = sample(letters[1:15], 200, replace = TRUE),
                        stringsAsFactors = FALSE)
edgelist1 <- edgelist1[which(edgelist1$col1 == edgelist1$col2), ]
# convert an edgelist to conflict matrix
confmatrix_bt <- as.conflictmat(edgelist1)
# Computes the MLE for the BT model
bt <- bradleyTerry(confmatrix_bt)
```

---

**bt.test**  
Systemic test for the assumptions of the Bradley-Terry model

Description

bt.test Systemic test for the assumptions of the Bradley-Terry model, transitivity and monotonic win-loss relationship. That is, if $A > B$ and $B > C$ then $A > C$ and $Pr(AbeatsC) > Pr(BbeatsC)$.

Usage

```r
bt.test(conf.mat, baseline = 1, maxLength = 2, reps = 1000)
```
Arguments

conf.mat  an N-by-N matrix. The matrix should be a conflict matrix with element i,j representing the number of times i has beaten j.

baseline  an integer between 1 and N inclusive identifying the agent with dominance index equal to zero.

maxLength  an integer indicating maximum path length used in conductance

reps      an integer indicating number of conflict matrices simulated to estimate the sampling distribution under the BT model.

Details

The value of the test statistic should be within the estimated sampling distribution of the test statistics under the BT model. The p-value of the test indicates the probability of statistics in the estimated sampling distribution is larger than the test statistic. It is not appropriate to use Bradley-Terry model if value of the test statistic is higher than the estimated sampling distribution of the test statistics.

Value

A list of 3 elements.

- stat      value of the test statistic
- dist      estimated sampling distribution of the test statistics under the BT model.
- p.val     p-value of the test

References


Examples

```r
# create an edgelist
edgelist1 <- data.frame(col1 = sample(letters[1:15], 200, replace = TRUE),
                       col2 = sample(letters[1:15], 200, replace = TRUE),
                       stringsAsFactors = FALSE)
edgelist1 <- edgelist1[-which(edgelist1$col1 == edgelist1$col2), ]
# convert an edgelist to conflict matrix
confmatrix_bt <- as.conflictmat(edgelist1)
# test the assumptions of the Bradley-Terry model
# not run:
# condTestOutput <- bt.test(confmatrix_bt)
```
Description
conductance compute win-loss probabilities for all possible pairs based upon the combined information from directed wins/losses and indirect win/loss pathways from the network.

Usage
conductance(conf, maxLength, alpha = NULL, beta = 1, strict = FALSE)

Arguments
- **conf**
  a matrix of conf.mat class. An N-by-N conflict matrix whose (i,j)th element is the number of times i defeated j.
- **maxLength**
  an integer greater than 1 and less than 7, indicating the maximum length of paths to identify.
- **alpha**
  a positive integer that reflects the influence of an observed win/loss interaction on an underlying win-loss probability. It is used in the calculation of the posterior distribution for the win-loss probability of i over j: Beta(αc_{i,j} + β, c_{i,j} + β). In the absence of expertise to accurately estimate alpha, it is estimated from the data.
- **beta**
  a positive numeric value that, like alpha, reflects the influence of an observed win/loss interaction on an underlying win-loss probability. Both α and β are chosen such that ((α + β)/(α + 2β))^2 is equal to the order-1 transitivity of the observed network. Therefore, β is commonly set to 1.
- **strict**
  a logical vector of length 1. It is used in transitivity definition for alpha estimation. It should be set to TRUE when a transitive triangle is defined as all pathways in the triangle go to the same direction; it should be set to FALSE when a transitive triangle is defined as PRIMARY pathways in the triangle go to the same direction. Strict = FALSE by default.

Details
This function performs two major steps. First, repeated random walks through the empirical network identify all possible directed win-loss pathways between each pair of nodes in the network. Second, the information from both direct wins/losses and pathways of win/loss interactions are combined into an estimate of the underlying probability of i over j, for all ij pairs.

Value
a list of two elements.

- **imputed.conf**
  An N-by-N conflict matrix whose (i,j)th element is the 'effective' number of wins of i over j.
countPaths

p.hat An N-by-N numeric matrix whose $(i,j)$th element is the estimated win-loss probability. Three functions (valueConverter, individualDomProb, and dyadicLongConverter) are provided to convert win-loss probability into other formats that are easier for further analysis of win-loss probability.

References


See Also

as.conflictmat, findIDpaths, transitivity, simRankOrder

Examples

# convert an edgelist to conflict matrix
cconfmatrix <- as.conflictmat(sampleEdgelist)
# find win-loss probability matrix
perm2 <- conductance(confmatrix, 2, strict = FALSE)
perm2$imputed.conf
perm2$p.hat

countPaths Identifies the number of paths of length less than or equal to maxLength between all pairs

Usage

countPaths(conf, maxLength = 2)

Arguments

conf a matrix of conf.mat class. An N-by-N conflict matrix whose $(i,j)$th element is the number of times $i$ defeated $j$.

maxLength a positive numeric integer indicating the maximum length of paths to identify

Value

A list in which elements are number of paths between all pairs of a given length.

See Also

as.conflictmat, findIDpaths, transitivity, conductance
Examples

```r
# convert an edgelist to conflict matrix
confmatrix <- as.conflictmat(sampleEdgelist)
# find number of paths of length 3 or less
npaths <- countPaths(confmatrix, 3)
```

dyadicLongConverter  
*d.yadic long format converter*

Description

`dyadicLongConverter` convert win-loss probability matrix into long format for each dyad

Usage

```r
dyadicLongConverter(matrix)
```

Arguments

- `matrix`: the win-loss matrix which is the second output from `conductance`.

Details

Values on the diagonal of the matrix are not included in the converted long-format data.

Value

A dataframe of dyadic level win-loss probability and ranking certainty.

See Also

`conductance`, `valueConverter`, `individualDomProb`

Examples

```r
# convert an edgelist to conflict matrix
confmatrix <- as.conflictmat(sampleEdgelist)
# find win-loss probability matrix
perm2 <- conductance(confmatrix, 2)
perm2$imputed.conf
perm2$p.hat
dl <- dyadicLongConverter(perm2$p.hat)
```
**findAllPaths**

Identifies all paths between all pairs of less than or equal to a certain length

---

**Description**

findFirstPaths Identifies all paths length less than or equal to `maxLength` between all pairs of competitors

**Usage**

`findFirstPaths(conf, maxLength = 2)`

**Arguments**

- `conf`: a matrix of `conf.mat` class. An N-by-N conflict matrix whose `(i,j)`th element is the number of times i defeated j.
- `maxLength`: a positive numeric integer indicating the maximum length of paths to identify

**Value**

A list of two elements.

- **direct pathways**: direct pathways found in original matrix
- **indirect pathways**: a list of all paths from length 2 to the given length

**See Also**

`countPaths`, `findIDpaths`, `transitivity`

**Examples**

```r
# convert an edgelist to conflict matrix
confMatrix <- as.conflictMat(sampleEdgelist)
# find all paths of length 3
allp.3 <- findAllPaths(confMatrix, 3)
```
findIDpaths  

find all paths of a certain length for an individual

Description
findIDpaths identifies all unique win-loss pathways of order \((len - 1)\) beginning at selected ID

Usage
findIDpaths(conf, ID, len = 2)

Arguments
- **conf**: a matrix of conf.mat class. An N-by-N conflict matrix whose \((i,j)\)th element is the number of times i defeated j.
- **ID**: a numeric or character vector of length 1. It specifies the subject at the beginning of each pathway.
- **len**: a positive integer of length 1 greater than 2. The length of the win-loss paths to be identified \((len = order + 1)\)

Value
return all win-loss paths of length(len) beginning at ID

See Also
- as.conflictmat, findAllPaths, countPaths

Examples
```r
confmatrix <- as.conflictmat(sampleEdgelist)
path38891 <- findIDpaths(confmatrix, ID = "Kuai", len = 2)
```

getAllCosts  

Associate each costs with its corresponding simulated annealing runs

Description
Associate each costs with its corresponding simulated annealing runs

Usage
getAllCosts(costs_all, num)
Arguments

costs_all costs of all simAnnealing runs. It is the first element of the output from getSimOutput.
num number of simulated annealing runs

Value

a data.frame of all costs.

---

getAllRankOrder assign IDs to all best rank orders

Description

assign IDs to all best rank orders

Usage

getAllRankOrder(ID_index, allRankOrder)

Arguments

ID_index it depends on the inputed data from simRankOrder. It takes the colnames of data as ID, and index this ID by its position in the colname.
allRankOrder all rank orders found in all simulated annealing runs. It is the third output from getSimOutput.

Value

a data.frame of all costs.

---

getBestRankOrder assign IDs to the best rank order

Description

assign IDs to the best rank order

Usage

getBestRankOrder(ID_index, bestRankOrder)
### Arguments

- **ID_index**: it depends on the inputed data from `simRankOrder`. It takes the colnames of data as ID, and index this ID by its position in the colname.
- **bestRankOrder**: the best rank order found in all simulated annealing runs. It is the second output from `getSimOutput`.

### Value

a data.frame of all costs.

---

**getSimOutput**

get useful outputs from simulated annealing processes

### Description

get useful outputs from simulated annealing processes

### Usage

`getSimOutput(simAnnealList, num)`

### Arguments

- **simAnnealList**: the output from `simAnnealing` process
- **num**: number of simulated annealing runs

### Value

a list of three elements

- **costs_all**: costs of all simulated annealing runs.
- **bestRankOrder**: best rank order found in all simulated annealing processes
- **allRankOrder**: a dataframe, all best rank orders found in each simulated annealing processes
individualDomProb

individualDomProb  individual-level probability converter

Description

individualDomProb convert win-loss probability matrix into long format for each dyad

Usage

individualDomProb(matrix)

Arguments

matrix  the win-loss matrix which is the second output from conductance.

Value

a dataframe. Averaging probability of win-loss relationship with all other individuals.

See Also

conductance, valueConverter, dyadicLongConverter

Examples

# convert an edgelist to conflict matrix
confmatrix <- as.conflictmat(sampleEdgelist)
# find win-loss probability matrix
perm2 <- conductance(confmatrix, 2)
perm2$imputed.conf
perm2$p.hat
individualLevelOutput <- individualDomProb(perm2$p.hat)

Perc

Perc.
plotConfmat  

**generate heat map for a matrix**

**Description**
plotConfmat generate heat map for a matrix or a win-loss probability matrix

**Usage**
plotConfmat(conf.mat, ordering = NA, labels = FALSE, ...)

**Arguments**
- conf.mat: an N-by-N matrix. Either a conflict matrix or a win-loss probability matrix (the second element from conductance output)
- ordering: a reordering of the rows/columns, specified by a permutation of 1:N
- labels: if TRUE, displaying the agent names as specified in the rownames() of conf.mat() on the heatmap
- ...: Further argument may be supplied and processed by lattice::levelplot.

**Value**
A heatmap

**See Also**
as.conflictmat, conductance

**Examples**
# convert an edgelist to conflict matrix
cconfmatrix <- as.conflictmat(sampleEdgelist)
# find win-loss probability matrix
perm2 <- conductance(confmatrix, 2)
# plotting
plotConfmat(perm2$p.hat)
plotProbDiagnosis

Description

Diagnosis Plot plotProbDiagnosis generate heat map for dominance probability matrix

Usage

plotProbDiagnosis(prob.mat, cutoff = 0.75, ...)

Arguments

prob.mat  dominance probability matrix
cutoff  a numeric value between 0.5 to 1. A value that is equal or greater than the cutoff is considered of high certainty.
...  Further argument may be supplied and processed by levelplot.

See Also

plotConfmat

sampleEdgelist

Description

sampleEdgelist. social interactions among 11 monkeys

Usage

sampleEdgelist

Format

A data frame of edgelist with 174 rows and 2 variables: Iname, Rname

Iname  winner, animal ID
Rname  loser, animal ID ...

McCowan Lab sample data.
sampleRawMatrix

Description

sampleRawMatrix. dominance interactions between 39 monkeys

Usage

sampleRawMatrix

Format

A 39 x 39 matrix representing number of times that a row wins over a column
McCowan Lab sample data.

sampleWeightedEdgelist

Description

sampleWeightedEdgelist. dominance interactions among 29 monkeys

Usage

sampleWeightedEdgelist

Format

A data frame of edgelist with 181 rows and 3 variables: Initiator1, Recipient1, Freq

Initiator1 winner, monkey name
Recipient1 loser, monkey name
Freq Frequency, count of interaction ...

McCowan Lab sample data.
**simRankOrder**

*Find rank order using simulated annealing*

**Description**

simRankOrder find the rank order for the win-loss relationship

**Usage**

`simRankOrder(data, num = 10, alpha = NULL, kmax = 1000)`

**Arguments**

- `data`: a matrix, the win-loss probability matrix which is the second element of the output from `conductance`
- `num`: number of SimAnnealing (default is set at 10)
- `alpha`: a positive integer that reflects the influence of an observed win/loss interaction on an underlying win-loss probability. It is used in the calculation of the posterior distribution for the win-loss probability of i over j: $Beta(\alpha c_{i,j} + c_{i,j} + \beta)$. In the absence of expertise to accurately estimate alpha, it is estimated from the data.
- `kmax`: an integer between 2 to 1000, indicating the number of simulations in each SimAnnealing.

**Value**

a list of two dataframes.

- `BestSimulatedRankOrder`: a dataframe representing the best simulated rank order.
- `Costs`: the cost of each simulated annealing run
- `AllSimulatedRankOrder`: a dataframe representing all simulated rank orders.

**References**


**See Also**

`conductance transitivity`
transitivity

Examples

# convert an edgelist to conflict matrix
confmatrix <- as.conflictmat(sampleEdgelist)
# find dominance probability matrix
perm2 <- conductance(confmatrix, maxLength = 2)
### Not run:
# Note: It takes a while to run the simRankOrder example.
s.rank <- simRankOrder(perm2$p.hat, num = 10, kmax = 1000)
s.rank$BestSimulatedRankOrder
s.rank$Costs
s.rank$AllSimulatedRankOrder

### End(Not run)

---

transitivity calculate transitivity measurements for a matrix

Description

transitivity calculate transitivity measurements for a matrix

Usage

transitivity(conf, strict = FALSE)

Arguments

conf an N-by-N conflict matrix whose (i,j)th element is the number of times i defeated j

strict a logical vector of length 1 (TRUE or FALSE). It is used in transitivity definition for alpha estimation. It should be set to TRUE when a transitive triangle is defined as all pathways in the triangle go to the same direction; it should be set to FALSE when a transitive triangle is defined as PRIMARY pathways in the triangle go to the same direction. Strict = FALSE by default.

Details

transitivity is calculated as the proportion transitive triangles in the total of transitive and intransitive triangles. transitivity is used to estimate alpha, which is used in turn in imputing information from indirect pathways as to what degree we can trust information from indirect pathways. Greater transitivity is associated with assigning higher weight to information from indirect pathways.
**Value**

A list of four elements.

- **transitive**  The number of transitive triangles.
- **intransitive**  The number of intransitive triangles.
- **transitivity**  transitivity, the proportion of transitive triangles.
- **alpha**  The value of alpha corresponding to this value of transitivity.

**See Also**

`countPaths, findIDpaths, conductance`

**Examples**

```r
# convert an edgelist to conflict matrix
confmatrix <- as.conflictmat(sampleEdgelist)
# transitivity calculation
conftrans <- transitivity(confmatrix, strict = FALSE)
conftrans$transitive
conftrans$intransitive
conftrans$transitivity
conftrans$alpha
```

---

**Description**

`valueConverter` converts or transforms all values (which range from 0.0 to 1.0) in the win-loss probability matrix into 0.5 - 1.0

**Usage**

`valueConverter(matrix)`

**Arguments**

- `matrix`  the win-loss matrix which is the second output from `conductance`.

**Value**

A matrix of win-loss probability ranging from 0.5 - 1.0.

**See Also**

`conductance, individualDomProb, dyadicLongConverter`
Examples

# convert an edgelist to conflict matrix
confmatrix <- as.conflictmat(sampleEdgelist)

# find win-loss probability matrix
perm2 <- conductance(confmatrix, 2)
perm2$imputed.conf
perm2$p.hat
convertedValue <- valueConverter(perm2$p.hat)
Index

*Topic datasets
  sampleEdgelist, 15
  sampleRowMatrix, 16
  sampleWeightedEdgelist, 16

as.conflictmat, 2, 7, 10, 14

bradleyTerry, 3
bt.test, 4

conductance, 3, 6, 7, 8, 13, 14, 17, 19
countPaths, 3, 7, 9, 10, 19
dyadicLongConverter, 7, 8, 13, 19

findAllPaths, 9, 10
findIDpaths, 3, 7, 9, 10, 19

getAllCosts, 10
getAllRankOrder, 11
getBestRankOrder, 11
getSimOutput, 12

individualDomProb, 7, 8, 13, 19

Perc, 13
Perc-package (Perc), 13
plotConfmat, 14, 15
plotProbDiagnosis, 15

sampleEdgelist, 15
sampleRowMatrix, 16
sampleWeightedEdgelist, 16
simRankOrder, 7, 17

transitivity, 3, 7, 9, 17, 18

valueConverter, 7, 8, 13, 19