Package 'GraphAlignment'

October 18, 2024

Version 1.69.0

Date 2013-01-15

Title GraphAlignment

Author Joern P. Meier <mail@ionflux.org>, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg.

Maintainer Joern P. Meier <mail@ionflux.org>

Description Graph alignment is an extension package for the R programming environment which provides functions for finding an alignment between two networks based on link and node similarity scores. (J. Berg and M. Laessig, ``Cross-species analysis of biological networks by Bayesian alignment", PNAS 103 (29), 10967-10972 (2006))

License file LICENSE

License_restricts_use yes

URL http://www.thp.uni-koeln.de/~berg/GraphAlignment/

biocViews GraphAndNetwork, Network

git_url https://git.bioconductor.org/packages/GraphAlignment

git_branch devel

git_last_commit ad491a3

git_last_commit_date 2024-04-30

Repository Bioconductor 3.20

Date/Publication 2024-10-17

Contents

`AlignedPairs` *Get aligned node pairs*

Description

Create a matrix of pairs of aligned nodes from networks A and B using the permutation vector P, where P is in the format returned by AlignNetworks().

Usage

```
AlignedPairs(A, B, P)
```

Arguments

`A`	adjacency matrix for network A
`B`	adjacency matrix for network B
`P`	permutation vector to be used as the alignment

Details

This function creates a matrix containing pairs of aligned nodes from networks A and B using the permutation vector P, where P is in the format returned by AlignNetworks.

Value

The return value is a matrix with two columns. The number of rows is equal to the number of aligned node pairs. Each row in the matrix denotes a pair of aligned nodes. In each row, the first element (index 1) is the label of a node in network A, and the second element (index 2) is the label of a node in network B.

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
ex<-GenerateExample(dimA=22, dimB=22, filling=.5, covariance=.6,
  symmetric=TRUE, numOrths=10, correlated=seq(1,18))

pinitial<-InitialAlignment(psize=34, r=ex$r, mode="reciprocal")

lookupLink<-seq(-2,2,.5)
linkParams<-ComputeLinkParameters(ex$a, ex$b, pinitial, lookupLink)

lookupNode<-c(-.5,.5,1.5)
nodeParams<-ComputeNodeParameters(dimA=22, dimB=22, ex$r,
  pinitial, lookupNode)

al<-AlignNetworks(A=ex$a, B=ex$b, R=ex$r, P=pinitial,
  linkScore=linkParams$ls,
  selfLinkScore=linkParams$ls,
  nodeScore1=nodeParams$s1, nodeScore0=nodeParams$s0,
  lookupLink=lookupLink, lookupNode=lookupNode,
  bStart=.1, bEnd=30,
  maxNumSteps=50)

alignedPairs<-AlignedPairs(A=ex$a, B=ex$b, al)
```

`AlignNetworks` *Align networks*

Description

Align networks A and B.

Usage

```
AlignNetworks(A, B, R, P, linkScore, selfLinkScore, nodeScore1,
  nodeScore0, lookupLink, lookupNode, bStart, bEnd, maxNumSteps, clamp=TRUE,
  directed=FALSE)
```

Arguments

`A`	adjacency matrix for network A
`B`	adjacency matrix for network B
`R`	node similarity matrix
`P`	permutation vector to be used as the initial alignment (see InitialAlignment)
`linkScore`	link score matrix (see ComputeLinkParameters)
`selfLinkScore`	self link score matrix (see ComputeLinkParameters)
`nodeScore1`	node score vector (s1) (see ComputeNodeParameters)
`nodeScore0`	node score vector for unaligned nodes (s0) (see ComputeNodeParameters)
`lookupLink`	link bin lookup table (see GetBinNumber)
`lookupNode`	node bin lookup table (see GetBinNumber)
`bStart`	start scaling value for simulated annealing

bEnd	end scaling value for simulated annealing
maxNumSteps	maximum number of steps
clamp	clamp values to range when performing bin lookups
directed	whether input networks should be treated as directed graphs

Details

This function finds an alignment between the two input networks, specified in the form of adjacency matrices, by repeatedly calling ComputeM and LinearAssignment, up to maxNumSteps times. Simulated annealing is performed if a range is specified in the bStart and bEnd arguments. This simple procedure is described in detail in [Berg, Laessig 2006]. Different procedures can easily be implemented by the user.

In each step, the matrix M is calculated from the scoring parameters and the current permutation vector P. The result is then normalized to the range [-1, 1] and, if simulated annealing is enabled, a random matrix depending on the current simulated annealing parameters is added. The linear assignment routine is used to calculate the value of P which is used to compute M in the next step.

If the flag directed is set, directed binary networks are encoded by suitable symmetric matrices using EncodeDirectedGraph. The corresponding 3x3 matrices of the link score are computed from the 2x2 matrices given as input.

Simulated annealing is enabled if bStart differs from bEnd. In this case, a value bStep = bEnd - bStart) / (maxNumSteps - 1) is calculated. In step n, the random matrix which is added to M is scaled by the factor 1 / [bStart + (n - 1) * bStep].

Value

The return value is a permutation vector p which aligns nodes from network a with nodes from network B (including dummy nodes). The returned permutation should be read in the following way: the node i in the network A is aligned to that node in the network B which label is at the i-th position of the permutation vector p. If the label at this position is larger than the size of the network B, the node i is not aligned.

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

References

Berg, J. & Laessig, M. (2006) Proc. Natl. Acad. Sci. USA 103, 10967-10972.

Examples

```
ex<-GenerateExample(dimA=22, dimB=22, filling=.5, covariance=.6,
  symmetric=TRUE, numOrths=10, correlated=seq(1,18))

pinitial<-InitialAlignment(psize=34, r=ex$r, mode="reciprocal")

lookupLink<-seq(-2,2,.5)
linkParams<-ComputeLinkParameters(ex$a, ex$b, pinitial, lookupLink)

lookupNode<-c(-.5,.5,1.5)
nodeParams<-ComputeNodeParameters(dimA=22, dimB=22, ex$r,
  pinitial, lookupNode)
```

```
al<-AlignNetworks(A=ex$a, B=ex$b, R=ex$r, P=pinitial,
  linkScore=linkParams$ls,
  selfLinkScore=linkParams$ls,
  nodeScore1=nodeParams$s1, nodeScore0=nodeParams$s0,
  lookupLink=lookupLink, lookupNode=lookupNode,
  bStart=.1, bEnd=30,
  maxNumSteps=50)
```

`AnalyzeAlignment` *Analyze an alignment*

Description

Analyze an alignment between two networks.

Usage

```
AnalyzeAlignment(A, B, R, P, lookupNode, epsilon, clamp)
```

Arguments

`A`	adjacency matrix for network A
`B`	adjacency matrix for network B
`R`	node similarity matrix
`P`	permutation vector
`lookupNode`	node bin lookup vector
`epsilon`	node similarity threshold
`clamp`	clamp values to range when performing bin lookups

Details

This function analyzes an alignment and returns various characteristics.

Value

The return value is a list containing the results. Defined values are:

na - number of aligned node pairs

nb - number of aligned node pairs where neither partner has appreciable sequence similarity with any node in the other network. Appreciable sequence similarity means that r>epsilon. Formally, this is the number of aligned node pairs (ia, ib), where no jb exists such that R[ia, jb] > epsilon and no ja such that R[ja, ib] > epsilon.

nc - number of aligned node pairs, where the partners have no appreciable sequence similarity, but one or both of them has appreciable sequence similarity with some other node in the other network. Formally, this is the number of aligned node pairs (ia, ib) with R[ia, ib] < epsilon but jb or ja exists, such that R[ia, jb] > epsilon or R[ja, ib] > epsilon.

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
ex<-GenerateExample(dimA=22, dimB=22, filling=.5, covariance=.6,
  symmetric=TRUE, numOrths=10, correlated=seq(1,18))

pinitial<-InitialAlignment(psize=34, r=ex$r, mode="reciprocal")

lookupLink<-seq(-2,2,.5)
linkParams<-ComputeLinkParameters(ex$a, ex$b, pinitial, lookupLink)

lookupNode<-c(-.5,.5,1.5)
nodeParams<-ComputeNodeParameters(dimA=22, dimB=22, ex$r,
  pinitial, lookupNode)

al<-AlignNetworks(A=ex$a, B=ex$b, R=ex$r, P=pinitial,
  linkScore=linkParams$ls,
  selfLinkScore=linkParams$ls,
  nodeScore1=nodeParams$s1, nodeScore0=nodeParams$s0,
  lookupLink=lookupLink, lookupNode=lookupNode,
  bStart=.1, bEnd=30,
  maxNumSteps=50)

AnalyzeAlignment(A=ex$a, B=ex$b, R=ex$r, P=al, lookupNode,
  epsilon=.5)
```

ComputeLinkParameters *Compute link parameters*

Description

Compute the optimal scoring parameters (link score) for a given alignment.

Usage

```
ComputeLinkParameters(A, B, P, lookupLink, clamp=TRUE)
```

Arguments

A	adjacency matrix for network A
B	adjacency matrix for network B
P	permutation vector (see InitialAlignment, AlignNetworks)
lookupLink	link bin lookup table (see GetBinNumber)
clamp	clamp values to range when performing bin lookups

Details

This function computes optimal link score parameters for use with ComputeM and AlignNetworks. It takes two matrices as well as an initial alignment P and the lookup table for link binning, lookupLink, as parameters.

Value

The return value is a list containing the self link score matrix (lsSelf) and the link score matrix (ls).

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
ex<-GenerateExample(dimA=22, dimB=22, filling=.5, covariance=.6,
  symmetric=TRUE, numOrths=10, correlated=seq(1,18))

pinitial<-InitialAlignment(psize=34, r=ex$r, mode="reciprocal")

lookupLink<-seq(-2,2,.5)
linkParams<-ComputeLinkParameters(ex$a, ex$b, pinitial, lookupLink)
```

`ComputeM` *Compute score matrix M*

Description

Compute the score matrix M.

Usage

```
ComputeM(A, B, R, P, linkScore, selfLinkScore, nodeScore1,
  nodeScore0, lookupLink, lookupNode, clamp=TRUE)
```

Arguments

`A`	adjacency matrix for network A
`B`	adjacency matrix for network B
`R`	node similarity matrix
`P`	permutation vector to be used as the initial alignment (see InitialAlignment)
`linkScore`	link score matrix (see ComputeLinkParameters)
`selfLinkScore`	self link score matrix (see ComputeLinkParameters)
`nodeScore1`	node score vector (s1) (see ComputeNodeParameters)
`nodeScore0`	node score vector for unaligned nodes (s0) (see ComputeNodeParameters)
`lookupLink`	link bin lookup table (see GetBinNumber)
`lookupNode`	node bin lookup table (see GetBinNumber)
`clamp`	clamp values to range when performing bin lookups

Details

This function computes the score Matrix M from the network adjacency matrices A and B, the node similarity matrix R, an alignment P (given as a permutation vector) and the node and link scores with their associated binning information. The alignment P is either generated by the previous iterative step, or, initially, by using InitialAlignment. The matrix M is then given to the linear assignment solver to compute the new alignment.

Value

The return value is the score matrix M.

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
ex<-GenerateExample(dimA=22, dimB=22, filling=.5, covariance=.6,
  symmetric=TRUE, numOrths=10, correlated=seq(1,18))

pinitial<-InitialAlignment(psize=34, r=ex$r, mode="reciprocal")

lookupLink<-seq(-2,2,.5)
linkParams<-ComputeLinkParameters(ex$a, ex$b, pinitial, lookupLink)

lookupNode<-c(-.5,.5,1.5)
nodeParams<-ComputeNodeParameters(dimA=22, dimB=22, ex$r,
  pinitial, lookupNode)

ComputeM(A=ex$a, B=ex$b, R=ex$r, P=pinitial,
  linkScore=linkParams$ls,
  selfLinkScore=linkParams$ls,
  nodeScore1=nodeParams$s1, nodeScore0=nodeParams$s0,
  lookupLink=lookupLink, lookupNode=lookupNode)
```

ComputeNodeParameters *Compute node parameters*

Description

Compute the optimal scoring parameters (node score) for a given alignment.

Usage

```
ComputeNodeParameters(dimA, dimB, R, P, lookupNode, clamp=TRUE)
```

Arguments

dimA	size of network A
dimB	size of network B
R	node similarity score matrix
P	permutation vector (see InitialAlignment, AlignNetworks)
lookupNode	node bin lookup table (see GetBinNumber)
clamp	clamp values to range when performing bin lookups

Details

This function computes optimal node score parameters for use with ComputeM and AlignNetworks. It takes the size of the networks, a matrix of node similarities R, an initial alignment P, and the lookup table for node binning, lookupNode, as parameters.

Value

The return value is list containing the node score vectors s0 and s1.

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
ex<-GenerateExample(dimA=22, dimB=22, filling=.5, covariance=.6,
  symmetric=TRUE, numOrths=10, correlated=seq(1,18))

pinitial<-InitialAlignment(psize=34, r=ex$r, mode="reciprocal")

lookupNode<-c(-.5,.5,1.5)
nodeParams<-ComputeNodeParameters(dimA=22, dimB=22, ex$r,
  pinitial, lookupNode)
```

`ComputeScores` *Compute scores*

Description

Compute scores.

Usage

```
ComputeScores(A, B, R, P, linkScore, selfLinkScore, nodeScore1,
  nodeScore0, lookupLink, lookupNode, symmetric=TRUE, clamp=TRUE)
```

Arguments

`A`	adjacency matrix for network A
`B`	adjacency matrix for network B
`R`	node similarity matrix
`P`	permutation vector to be used as the initial alignment (see InitialAlignment)
`linkScore`	link score matrix (see ComputeLinkParameters)
`selfLinkScore`	self link score matrix (see ComputeLinkParameters)
`nodeScore1`	node score vector (s1) (see ComputeNodeParameters)
`nodeScore0`	node score vector for unaligned nodes (s0) (see ComputeNodeParameters)
`lookupLink`	link bin lookup table (see GetBinNumber)
`lookupNode`	node bin lookup table (see GetBinNumber)
`symmetric`	network symmetry flag
`clamp`	clamp values to range when performing bin lookups

Details

This function computes log-likelihood scores for an alignment using the specified scoring tables, two networks A and B and their alignment P. The total score of the alignment has two contributions, the first coming from the sequence homology (node similarity, sn) and the second from the similarity of interaction networks (sl).

Value

The return value is a list containing the link score (sl) and the node score (sn).

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
ex<-GenerateExample(dimA=22, dimB=22, filling=.5, covariance=.6,
  symmetric=TRUE, numOrths=10, correlated=seq(1,18))

pinitial<-InitialAlignment(psize=34, r=ex$r, mode="reciprocal")

lookupLink<-seq(-2,2,.5)
linkParams<-ComputeLinkParameters(ex$a, ex$b, pinitial, lookupLink)

lookupNode<-c(-.5,.5,1.5)
nodeParams<-ComputeNodeParameters(dimA=22, dimB=22, ex$r,
  pinitial, lookupNode)

al<-AlignNetworks(A=ex$a, B=ex$b, R=ex$r, P=pinitial,
  linkScore=linkParams$ls,
  selfLinkScore=linkParams$ls,
  nodeScore1=nodeParams$s1, nodeScore0=nodeParams$s0,
  lookupLink=lookupLink, lookupNode=lookupNode,
  bStart=.1, bEnd=30,
  maxNumSteps=50)

ComputeScores(A=ex$a, B=ex$b, R=ex$r, P=al,
  linkScore=linkParams$ls,
  selfLinkScore=linkParams$ls,
  nodeScore1=nodeParams$s1, nodeScore0=nodeParams$s0,
  lookupLink=lookupLink, lookupNode=lookupNode,
  symmetric=TRUE)
```

CreateScoreMatrix *Create score matrix*

Description

Create a score matrix from two lookup vectors.

Usage

```
CreateScoreMatrix(lookupX, lookupY)
```

Arguments

lookupX	x lookup vector
lookupY	y lookup vector

Details

This function creates a very simple score matrix containing the product of lookup table values for each row and column as its elements. This can be used for testing purposes.

Value

The return value is a matrix of link scores.

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
lookupLink <- 0.1*(-10:10)
CreateScoreMatrix(lookupLink, lookupLink)
```

EncodeDirectedGraph *Encode directed graph*

Description

Encode an adjacency matrix for a directed graph into a symmetric matrix.

Usage

```
EncodeDirectedGraph(matrix, P)
```

Arguments

matrix	square matrix
P	permutation vector

Details

This function encodes an adjacency matrix for a directed graph into a symmetric matrix. Currently only binary directed graphs are implemented. The adjacency matrix of a binary directed graph has elements 0, 1. The same graph can be represented by a symmetric adjacency matrix with elements -1, 0, 1, with the sign of the entry indicating the direction of the link. The result is obtained by setting entries (i, j) and (j, i) of the target matrix m' to 1 if entry m[i, j] = 1 and P[i] > P[j] and to -1 if m[i, j] = 1 and P[j] > P[i].

Value

The return value is a symmetric matrix representing the encoded input matrix.

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
dg <- matrix(c(0, 1, 0, 0, 1, 1, 1, 0, 1), 3, 3)
EncodeDirectedGraph(dg, 1:dim(dg)[1])
```

GenerateExample *Generate example input data*

Description

Generate input data which can be used as an example.

Usage

```
GenerateExample(dimA, dimB, filling, covariance, symmetric = FALSE,
numOrths = 0, correlated = NA, distribution = "normal")
```

Arguments

dimA	size of network A
dimB	size of network B
filling	fraction of entries which are not explicitly set to zero
covariance	covariance of correlated rows and columns
symmetric	generate symmetric matrices
numOrths	number of diagonal elements to set to 1 in example matrix R
correlated	indices of correlated rows and columns
distribution	from which distribution are the values drawn ("normal" or "uniform")

Details

Symmetric example matrices A, B can be generated by specifying TRUE as the 'symmetric' argument. If the 'numOrths' argument is specified, a number of diagonal elements of R equal to the value will be set to 1. If a vector is specified as the 'correlated' argument, the specified rows and columns will be set to correlated values. Leaving this argument blank will result in pairwise correlations of all entries in A, B (or, if the matrices are of different rank, all elements of the smaller on will be correlated with the corresponding parts of the larger matrix).

Value

The return value is a list containing example matrices A (a), B (b) and R (r).

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
ex <- GenerateExample(dimA = 170, dimB = 140, filling = 0.75,
                      covariance = 0.4, symmetric = TRUE, numOrths = 90,
                      correlated = 1:90, distribution = "normal")

##-- edge weight correlation
image(cor(ex$a[1:140, 1:140], ex$b))
hist(diag(cor(ex$a[1:140, 1:140], ex$b))[1:90])

##-- edge weight distribution
hist(ex$a[ex$a != 0]);
hist(ex$b[ex$b != 0]);
```

`GetBinNumber` *Get bin number*

Description

Get the bin number for the argument using the specified lookup vector.

Usage

```
GetBinNumber(x, lookup, clamp=TRUE)
```

Arguments

`x`	value to be binned
`lookup`	lookup vector
`clamp`	clamp value to lookup range

Details

The lookup vector defines the lower and upper boundaries for each bin. The first entry in the lookup vector is the lower boundary of the first bin, while the last value in the lookup vector is the upper boundary of the last bin. For all other entries, entry i of the lookup vector defines the upper boundary of the (i-1)-th bin and the lower boundary of the i-th bin. The number of bins is therefore n-1, where n is the length of the lookup vector. A lookup vector must have at least two elements.

If clamping is enabled (clamp=TRUE), arguments which fall below the lower boundary of the first bin are treated as if they are actually in the first bin. Likewise, values which are above the upper boundary of the last bin are treated as if they are actually in the last bin. If clamping is disabled (clamp=FALSE), values outside the lookup range cause an error.

Value

This function returns the bin number corresponding to the input value. The bin number is obtained by performing a lookup in the specified lookup vector.

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
lookup <- c(-1, 0, 1)
GetBinNumber(-0.5, lookup)
GetBinNumber(0.5, lookup)
```

`InitialAlignment` *Create initial alignment*

Description

Create a permutation for an initial alignment. Either "random" or "reciprocal" may be specified as the mode argument.

Usage

```
InitialAlignment(psize, r=NA, mode="random")
```

Arguments

`psize`	size of the alignment
`r`	node similarity score matrix (required for mode 'reciprocal')
`mode`	type of initial alignment

Details

To create a random initial alignment of size psize, the InitialAlignment function can be used with the mode argument set to "random". If mode is set to "reciprocal", a reciprocal best match algorithm is applied to the input matrix R to find an initial alignment. This mode requires that the psize argument is sufficiently large to allow for the addition of dummy nodes to which unaligned nodes can formally be aligned.

Value

The return value is a permutation vector of the specified size.

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
ex<-GenerateExample(dimA=22, dimB=22, filling=.5, covariance=.6,
  symmetric=TRUE, numOrths=10, correlated=seq(1,18))

pinitial<-InitialAlignment(psize=34, r=ex$r, mode="reciprocal")
```

InvertPermutation *Invert permutation*

Description

Invert the permutation defined by the specified vector.

Usage

```
InvertPermutation(p)
```

Arguments

p	permutation vector

Details

The return value is a permutation vector equal to the inverse of the input permutation.

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
p <- rank(rnorm(10))
pInv <- InvertPermutation(p)
```

LinearAssignment *Solve linear assignment problem*

Description

Solve the linear assignment problem specified by the cost matrix.

Usage

```
LinearAssignment(matrix)
```

Arguments

matrix	cost matrix

Details

This function solves the linear assignment problem defined by the input matrix. [Jonker, Volgenant 1987]

Value

The return value is a permutation vector equal to the solution of the linear assignment problem specified by the cost matrix. The result is the permutation P for which MP is minimal (where M is the cost matrix).

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

References

Jonker, R & Volgenant, A. (1987) Computing 38, 325–340.

Examples

```
m <- matrix(rnorm(25), 5, 5)
px <- LinearAssignment(round(-1000 * (m / max(abs(m)))))
m[px,]
```

`MatrixToBin` *Convert matrix to bin representation*

Description

Convert a matrix of real numbers to a vector of integer bin numbers according to the specified lookup vector.

Usage

```
MatrixToBin(M, lookup, clamp=TRUE)
```

Arguments

`M`	input matrix
`lookup`	lookup vector
`clamp`	clamp values to lookup range

Details

This function transforms a matrix of arbitrary values into a matrix of bin numbers corresponding to the data in the input matrix. Bin numbers are found using the specified lookup table (see GetBinNumber).

Value

The return value is a matrix containing the bin numbers for the elements of the input matrix.

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
m <- matrix(rnorm(25), 5, 5)
lookup <- 0.2*(-5:5)
MatrixToBin(m, lookup)
```

`Permute` *Permute rows and columns of a matrix*

Description

Permute rows and columns of the input matrix.

Usage

```
Permute(m, p, invertp=FALSE)
```

Arguments

`m`	matrix
`p`	permutation vector
`invertp`	apply inverse permutation

Details

This function permutes rows and columns of a matrix using the specified permutation vector. The inverse of the permutation will be applied if the invertp argument is set to TRUE.

Value

The return value is the row- and columnwise Permutation of the elements of the input matrix, so Permute(m, p)[i, j] is equal to m[p[i], p[j]].

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
example <- GenerateExample(10, 10, 1, 0.5, TRUE, 4)
b <- example$b
p <- rank(rnorm(10))
Permute(b, p, TRUE)
```

Trace *Calculate trace of a matrix*

Description

Calculate the trace of a square matrix.

Usage

```
Trace(m)
```

Arguments

m	square matrix

Details

The return value is the trace of the input matrix.

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
m <- matrix(c(1, 0, 0, 1), 2, 2)
Trace(m)
```

VectorToBin *Convert vector to bin representation*

Description

Convert a vector of real numbers to a vector of integer bin numbers according to the specified lookup vector.

Usage

```
VectorToBin(v, lookup, clamp=TRUE)
```

Arguments

v	input vector
lookup	lookup vector
clamp	clamp values to lookup range

Details

This function transforms a vector of arbitrary values into a vector of bin numbers corresponding to the data in the input vector. Bin numbers are found using the specified lookup table (see GetBinNumber).

Value

The return value is a vector containing the bin numbers for the elements of the input vector.

Author(s)

Joern P. Meier, Michal Kolar, Ville Mustonen, Michael Laessig, and Johannes Berg

Examples

```
x <- rnorm(10)
lookup <- 0.2*(-5:5)
VectorToBin(x, lookup)
```

Index