Package 'NetInt'

May 3, 2025

Type Package

Title Methods for Unweighted and Weighted Network Integration

Version 1.0.1

Maintainer Jessica Gliozzo <jessica.gliozzo@unimi.it>

Description Implementation of network integration approaches comprising unweighted and weighted integration methods. Unweighted integration is performed considering the average, per-edge average, maximum and minimum of networks edges. Weighted integration takes into account a weight for each network during the fusion process, where the weights express the "predictiveness strength" of each network considering a specific predictive task. Weights can be learned using a machine learning algorithm able to associate the weights to the assessment of the accuracy of the learning algorithm trained on the network itself. The implemented methods can be applied to effectively integrate different biological networks modelling a wide range of problems in bioinformatics (e.g. disease gene prioritization, protein function prediction, drug repurposing, clinical outcome prediction).

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Giorgio Valentini [aut], Jessica Gliozzo [cre]

Contents

`align.networks` *Function to align the adjacency matrices of graphs/networks*

Description

It accepts a list of adjacency matrices (that is an arbitrary number of matrices separated by commas) and returns another list of adjacency matrices having as elements the union of the elements of all the matrices. Missed elements are replaced with null rows and columns. In this way the resulting matrices have the same number of rows/columns in the same order.

Usage

```
align.networks(fill = 0, ...)
```

Arguments

`fill`	value used for the missing elements (def: 0).
`...`	a list of numeric matrices. These must be named matrices, and corresponding elements in different matrices must have the same name.

Value

list of matrices : they correspond exactly and in the same order to the input matrices, but they are filled with rows and columns when they have missed values. The missing values are filled with fill.

Examples

```
# Create three example networks of different size
set.seed(123);
A1 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A1[lower.tri(A1)] = t(A1)[lower.tri(A1)];
diag(A1) <- 0;
rownames(A1) <- colnames(A1) <- sample(LETTERS, 10);

A2 <- matrix(runif(49, min = 0, max = 1), nrow = 7);
A2[lower.tri(A2)] = t(A2)[lower.tri(A2)];
diag(A2) <- 0;
rownames(A2) <- colnames(A2) <- rownames(A1)[1:7];

A3 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A3[lower.tri(A3)] = t(A3)[lower.tri(A3)];
diag(A3) <- 0;
rownames(A3) <- colnames(A3) <- c(rownames(A1)[1:5], c("A", "B", "Z", "K", "Q"));

# Align networks
A_aligned <- align.networks(fill = 0, A1, A2, A3);
```

ATLEASTK.int *ATLEASTK network integration*

Description

It performs the ATLEAST integration between networks: only edges present in at least k networks are preserved, the others are eliminated. The resulting network is a binary network: the edge is 1 if preserved, otherwise 0. An edge is considered "present" if its value is larger than 0.

Usage

```
ATLEASTK.int(k = 1, ...)
```

Arguments

k	the minimum number of the networks in which each edge must be present to be preserved (k=1).
...	a list of numeric matrices. These must be named matrices representing adjacency matrices of the networks. Matrices may have different dimensions, but corresponding elements in different matrices must have the same name.

Value

the integrated matrix : the matrix resulting from ATLEASTK.

Examples

```
# Create three example networks of different size
set.seed(123);
A1 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A1[lower.tri(A1)] = t(A1)[lower.tri(A1)];
diag(A1) <- 0;
rownames(A1) <- colnames(A1) <- sample(LETTERS, 10);

A2 <- matrix(runif(49, min = 0, max = 1), nrow = 7);
A2[lower.tri(A2)] = t(A2)[lower.tri(A2)];
diag(A2) <- 0;
rownames(A2) <- colnames(A2) <- rownames(A1)[1:7];

A3 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A3[lower.tri(A3)] = t(A3)[lower.tri(A3)];
diag(A3) <- 0;
rownames(A3) <- colnames(A3) <- c(rownames(A1)[1:5], c("A", "B", "Z", "K", "Q"));

# Integrate networks using ATLEASTK method
A_int <- ATLEASTK.int(k=2, A1, A2, A3);
```

`lalign.networks` *Function to align a list of adjacency matrices of graphs/networks*

Description

It accepts a list of adjacency matrices (that is an arbitrary number of matrices separated by commas) and returns another list of adjacency matrices having as elements the union of the elements of all the matrices. Missed elements are replaced with null rows and columns. In this way the resulting matrices have the same number of rows/columns in the same order. NOTE: It is equal to align.networks with a list of matrices as argument instead of the ... generic argument.

Usage

```
lalign.networks(fill = 0, networks)
```

Arguments

`fill`	value used for the missing elements (def: 0).
`networks`	a list of numeric matrices. These must be named matrices, and corresponding elements in different matrices must have the same name.

Value

list of matrices : they correspond exactly and in the same order to the input matrices, but they are filled with rows and columns when they have missed values. The missing values are filled with fill.

Examples

```
# Create three example networks of different size
set.seed(123);
A1 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A1[lower.tri(A1)] = t(A1)[lower.tri(A1)];
diag(A1) <- 0;
rownames(A1) <- colnames(A1) <- sample(LETTERS, 10);

A2 <- matrix(runif(49, min = 0, max = 1), nrow = 7);
A2[lower.tri(A2)] = t(A2)[lower.tri(A2)];
diag(A2) <- 0;
rownames(A2) <- colnames(A2) <- rownames(A1)[1:7];

A3 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A3[lower.tri(A3)] = t(A3)[lower.tri(A3)];
diag(A3) <- 0;
rownames(A3) <- colnames(A3) <- c(rownames(A1)[1:5], c("A", "B", "Z", "K", "Q"));

# Align networks
A_aligned <- lalign.networks(fill = 0, list(A1, A2, A3));
```

MAX.int *Maximum (MAX) network integration*

Description

It performs the Max integration between networks:

$$\bar{w}_{ij} = \max_d w_{ij}^d$$

Usage

```
MAX.int(...)
```

Arguments

...	a list of numeric matrices. These must be named matrices representing adjacency matrices of the networks. Matrices may have different dimensions, but corresponding elements in different matrices must have the same name.

Value

the integrated matrix : the matrix resulting from MAX.

Examples

```
# Create three example networks of different size
set.seed(123);
A1 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A1[lower.tri(A1)] = t(A1)[lower.tri(A1)];
diag(A1) <- 0;
rownames(A1) <- colnames(A1) <- sample(LETTERS, 10);

A2 <- matrix(runif(49, min = 0, max = 1), nrow = 7);
A2[lower.tri(A2)] = t(A2)[lower.tri(A2)];
diag(A2) <- 0;
rownames(A2) <- colnames(A2) <- rownames(A1)[1:7];

A3 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A3[lower.tri(A3)] = t(A3)[lower.tri(A3)];
diag(A3) <- 0;
rownames(A3) <- colnames(A3) <- c(rownames(A1)[1:5], c("A", "B", "Z", "K", "Q"));

# Integrate networks using Maximum (MAX) method
A_int <- MAX.int(A1, A2, A3);
```

MIN.int *Minimum (MIN) network integration*

Description

It performs the Min integration between networks:

$$\bar{w}_{ij} = \min_d w_{ij}^d$$

Note that this function consider the minimum between existing edges, that is if an edge (i,j) is not present in a network, since one of the nodes i or j is not present in the network, then the edge is not considered in the computation. If the edge (i,j) is not present in any of the available networks, that its value is 0. If drastic=TRUE the minimum is zero if at least one edge is not present in a network.

Usage

```
MIN.int(drastic = FALSE, ...)
```

Arguments

drastic	if TRUE the minimum is zero if at least one edge is not present in a network (def: FALSE).
...	a list of numeric matrices. These must be named matrices representing adjacency matrices of the networks. Matrices may have different dimensions, but corresponding elements in different matrices must have the same name.

Value

the integrated matrix : the matrix resulting from MIN.

Examples

```
# Create three example networks of different size
set.seed(123);
A1 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A1[lower.tri(A1)] = t(A1)[lower.tri(A1)];
diag(A1) <- 0;
rownames(A1) <- colnames(A1) <- sample(LETTERS, 10);

A2 <- matrix(runif(49, min = 0, max = 1), nrow = 7);
A2[lower.tri(A2)] = t(A2)[lower.tri(A2)];
diag(A2) <- 0;
rownames(A2) <- colnames(A2) <- rownames(A1)[1:7];

A3 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A3[lower.tri(A3)] = t(A3)[lower.tri(A3)];
diag(A3) <- 0;
rownames(A3) <- colnames(A3) <- c(rownames(A1)[1:5], c("A", "B", "Z", "K", "Q"));

# Integrate networks using Minimum (MIN) method
A_int.noDrastic <- MIN.int(drastic = FALSE, A1, A2, A3);

# Integrate networks using Minimum (MIN) method (drastic integration)
A_int.drastic <- MIN.int(drastic = TRUE, A1, A2, A3);
```

MS.UA.int *"Memory Saving" Unweighted Average (UA) network integration*

Description

It performs the unweighted average integration between networks:

$$\bar{w}_{ij} = \frac{1}{n} \sum_{d=1}^{n} w_{ij}^{d}$$

The matrices are read from files and loaded one at time in memory.

Usage

```
MS.UA.int(nets.files, example.names = NULL)
```

Arguments

nets.files	a list with the names of the .rda files storing the matrices representing the weighted adjacency matrices of the nets.
example.names	a list with the names of the examples stored in each net. If NULL (def.) it is assumed that the matrices have exactly the same examples in the same order, otherwise the matrices are arranged to be correctly aligned.

Value

the integrated matrix : the matrix resulting from UA.

PUA.int *Per-edge Unweighted Average (PUA) network integration*

Description

It performs the per-edge unweighted average integration between networks:

$$\bar{w}_{ij} = \frac{1}{|D(i,j)|} \sum_{d \in D(i,j)} w_{ij}^{d}$$

where:

$$D(i,j) = \{d | v_i \in V^d \wedge v_j \in V^d\}$$

Usage

```
PUA.int(...)
```

Arguments

...	a list of numeric matrices. These must be named matrices representing adjacency matrices of the networks. Matrices may have different dimensions, but corresponding elements in different matrices must have the same name.

Value

the integrated matrix : the matrix resulting from PUA.

Examples

```
# Create three example networks of different size
set.seed(123);
A1 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A1[lower.tri(A1)] = t(A1)[lower.tri(A1)];
diag(A1) <- 0;
rownames(A1) <- colnames(A1) <- sample(LETTERS, 10);

A2 <- matrix(runif(49, min = 0, max = 1), nrow = 7);
A2[lower.tri(A2)] = t(A2)[lower.tri(A2)];
diag(A2) <- 0;
rownames(A2) <- colnames(A2) <- rownames(A1)[1:7];

A3 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A3[lower.tri(A3)] = t(A3)[lower.tri(A3)];
diag(A3) <- 0;
rownames(A3) <- colnames(A3) <- c(rownames(A1)[1:5], c("A", "B", "Z", "K", "Q"));

# Integrate networks using Per-edge Unweighted Average (PUA) method
A_int <- PUA.int(A1, A2, A3);
```

UA.int *Unweighted Average (UA) network integration*

Description

It performs the unweighted average integration between networks:

$$\bar{w}_{ij} = \frac{1}{n} \sum_{d=1}^{n} w_{ij}^{d}$$

Usage

```
UA.int(align = TRUE, ...)
```

Arguments

align	logical. If TRUE (def.) the matrices are aligned using align.networks, otherwise they are directly summed without any previous alignment.
...	a list of numeric matrices. These must be named matrices representing adjacency matrices of the networks. Matrices may have different dimensions, but corresponding elements in different matrices must have the same name.

Value

the integrated matrix : the matrix resulting from UA.

Examples

```
# Create three example networks of different size
set.seed(123);
A1 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A1[lower.tri(A1)] = t(A1)[lower.tri(A1)];
diag(A1) <- 0;
rownames(A1) <- colnames(A1) <- sample(LETTERS, 10);

A2 <- matrix(runif(49, min = 0, max = 1), nrow = 7);
A2[lower.tri(A2)] = t(A2)[lower.tri(A2)];
diag(A2) <- 0;
rownames(A2) <- colnames(A2) <- rownames(A1)[1:7];

A3 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A3[lower.tri(A3)] = t(A3)[lower.tri(A3)];
diag(A3) <- 0;
rownames(A3) <- colnames(A3) <- c(rownames(A1)[1:5], c("A", "B", "Z", "K", "Q"));

# Integrate networks using Unweighted Average (UA) method
A_int <- UA.int(align=TRUE, A1, A2, A3);
```

WA.int *Weighted Average (WA) network integration*

Description

It performs the WA integration between networks.

Note that this function puts more weight (alpha parameter) for networks with associated larger M. The alphas are computed by averaging across the alpha of each class, and hence a unique integrated network is available for all the considered classes.

Usage

```
WA.int(M, logint = FALSE, ...)
```

Arguments

M	a numeric matrix with the values of the metric used to compute the alpha coefficients for each class and for each network. Rows correspond to networks and columns to classes. Element (i,j) of the matrix corresponds to the value of the metric (e.g. AUC) for the ith network and the jth class.
logint	logic. If TRUE the mean values m are log transformed: -log(1-m), otherwise a linear integration is performed (def: FALSE).
...	a list of numeric matrices. These must be named matrices representing adjacency matrices of the networks. Matrices may have different dimensions, but corresponding elements in different matrices must have the same name.

Value

A list with two elements:

- WA : the matrix resulting from WA
- alpha : a numeric vector with the weight coefficients of the networks

Examples

```
# Create three example networks of different size
set.seed(123);
A1 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A1[lower.tri(A1)] = t(A1)[lower.tri(A1)];
diag(A1) <- 0;
rownames(A1) <- colnames(A1) <- sample(LETTERS, 10);

A2 <- matrix(runif(49, min = 0, max = 1), nrow = 7);
A2[lower.tri(A2)] = t(A2)[lower.tri(A2)];
diag(A2) <- 0;
rownames(A2) <- colnames(A2) <- rownames(A1)[1:7];

A3 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A3[lower.tri(A3)] = t(A3)[lower.tri(A3)];
diag(A3) <- 0;
rownames(A3) <- colnames(A3) <- c(rownames(A1)[1:5], c("A", "B", "Z", "K", "Q"));

# Create random matrix of accuracy metrics (considering 3 classes)
M <- matrix(runif(9, min = 0, max = 1), ncol = 3);

# Integrate networks using Weighted Average (WA) method
A_int <- WA.int(M, logint=TRUE, A1, A2, A3);
```

`WAP.int` *Weighted Average Per-class (WAP) network integration*

Description

It performs the WAP integration between networks:

$$\bar{w}_{ij}(k) = \sum_{d=1}^{n} \alpha^d(k) w_{ij}^d$$

where

$$\alpha^d(k) = \frac{1}{\sum_{j=1}^{n} M^j(k)} M^d(k)$$

and $M^d(k)$ is a suitable accuracy metrics for class k on network d. The metrics could be, e.g. the AUC or the precision at a given recall. Note that this function puts more weight (alpha parameter) for networks with associated larger M.

Usage

```
WAP.int(m, align = FALSE, logint = FALSE, ...)
```

Arguments

`m`	a numeric vector with the values of the metric used to compute the alpha coefficients. It could be e.g. AUC values.
`align`	logic. If TRUE the numeric matrices passed as arguments are aligned according to the function align.networks (def: FALSE).

logint	logic. If TRUE m is log transformed: -log(1-m), otherwise a linear integration is performed (def: FALSE).
...	a list of numeric matrices. These must be named matrices representing adjacency matrices of the networks. Matrices may have different dimensions, but corresponding elements in different matrices must have the same name.

Value

A list with two elements:

- WAP : the matrix resulting from WAP
- alpha : a numeric vector with the weight coefficients of the networks

Examples

```
# Create three example networks of different size
set.seed(123);
A1 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A1[lower.tri(A1)] = t(A1)[lower.tri(A1)];
diag(A1) <- 0;
rownames(A1) <- colnames(A1) <- sample(LETTERS, 10);

A2 <- matrix(runif(49, min = 0, max = 1), nrow = 7);
A2[lower.tri(A2)] = t(A2)[lower.tri(A2)];
diag(A2) <- 0;
rownames(A2) <- colnames(A2) <- rownames(A1)[1:7];

A3 <- matrix(runif(100, min = 0, max = 1), nrow = 10);
A3[lower.tri(A3)] = t(A3)[lower.tri(A3)];
diag(A3) <- 0;
rownames(A3) <- colnames(A3) <- c(rownames(A1)[1:5], c("A", "B", "Z", "K", "Q"));

# Create random vector of accuracy metrics
m <- runif(3, min = 0, max = 1);

# Integrate networks using Weighted Average Per-class (WAP) method
A_int <- WAP.int(m, align=TRUE, logint=FALSE, A1, A2, A3);
```

Index