Package 'NetPreProc'

January 20, 2025

Type Package

Title Network Pre-Processing and Normalization

Version 1.2

Date 2022-09-14

Author Giorgio Valentini [aut, cre]

Maintainer Giorgio Valentini <valentini@di.unimi.it>

Description Network Pre-Processing and normalization. Methods for normalizing graphs, including Chua normalization, Laplacian normalization, Binary magnification, min-max normalization and others. Methods to sparsify adjacency matrices. Methods for graph pre-processing and for filtering edges of the graph.

License GPL (>= 2)

LazyLoad yes

Depends methods

Imports graph

Suggests bionetdata

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-09-19 11:06:10 UTC

Contents

`NetPreProc-package` *NetPreProc*

Description

Network Pre-Processing and normalization.

Details

Package:	NetPreProc
Type:	Package
Version:	1.2
Date:	2022-09-14
License:	GPL (>= 2)
LazyLoad:	yes
Depends:	methods
Imports:	graph
Suggests:	bionedata

Network Pre-Processing and normalization. Methods for normalizing graphs, including Chua normalization, Laplacian normalization, Binary magnification, min-max normalization and oters. Methods to sparsify adjacency matrices. Methods for graph pre-processing and for filtering edges of the graph.

Author(s)

Giorgio Valentini – Universita' degli Studi di Milano

Maintainer: Giorgio Valentini<valentini@di.unimi.it>

`Binary.matrix.by.thresh-methods`
Transforming a real-valued network matrix into a binary matrix

---

Description

Methods to transform a a real-valued network matrix into a binary matrix. The binary matrix is obtained by thresholding: values above the given threshold are set to 1, otherwise to 0

Usage

```
Binary.matrix.by.thresh(W, thresh=0.5)
```

Arguments

W	an object representing the graph to be normalized
thresh	the threshold (def.=0.5)

Value

The normalized binary adjacency matrix of the network

Methods

`signature(W = "graph")` an object of the virtual class graph (hence including objects of class `graphAM` and `graphNEL` from the package **graph**)

`signature(W = "matrix")` a matrix representing the adjacency matrix of the graph

Examples

```
library(bionetdata);
data(DD.chem.data);
W <- Binary.matrix.by.thresh(DD.chem.data);

# Using both methods with both signatures "matrix" and "graph"
# reducing dimension of the graph
library(graph);
DD.chem.data.red <- DD.chem.data[1:100,1:100];
W.red <- Binary.matrix.by.thresh(DD.chem.data.red);
g <- new("graphAM", adjMat=DD.chem.data.red, values=list(weight=DD.chem.data.red));
Wg <- Binary.matrix.by.thresh(g);
any(W.red!=Wg);
```

---

check.network-methods *Graph checking*

Description

Method to check the characteristics of a graph. Check if its adjacency matrix is symmetric, if it has NA, NaN o Inf values, and some minimals statistics about nodes and edges.

Usage

```
check.network(W, name="Network matrix")
```

Arguments

W	an object representing the graph to be checked
name	a character vector that will be printed as heading

Value

It return a list of strings about the characteristics of the graph

Methods

`signature(W = "graph")` an object of the virtual class graph (hence including objects of class `graphAM` and `graphNEL` from the package **graph**)

`signature(W = "matrix")` a matrix representing the adjacency matrix of the graph

Examples

```
library(bionetdata);
data(DD.chem.data);
check.network(DD.chem.data);
W <- Prob.norm(DD.chem.data);
check.network(W, "prob. transition matrix");
WL <- Laplacian.norm(DD.chem.data);
check.network(WL, "Laplacian norm. matrix");

library(graph)
g1 = randomEGraph(LETTERS[1:15], edges = 40);
check.network(g1, "random graph");
```

Chua.norm-methods *Chua normalization*

Description

Normalization of graphs according to Chua et al., 2007. The normalized weigths between nodes are computed by taking into account their neighborhoods. This normalization is meaningful in particular with interaction data. More precisely, the normalized weigth W_{ij} between nodes i and j is computed by taking into account their neighborhods N_i and N_j :

$$W_{ij} = \frac{2|N_i \cap N_j|}{|N_i \setminus N_j| + 2|N_i \cap N_j| + 1} \times \frac{2|N_i \cap N_j|}{|N_j \setminus N_i| + 2|N_i \cap N_j| + 1}$$

where N_k is the set of the neighbors of gene k (k is included).

Usage

```
Chua.norm(W)
```

Arguments

`W`	an object representing the graph to be normalized

Value

The normalized adjacency matrix of the network

Methods

`signature(W = "graph")` an object of the virtual class graph (hence including objects of class `graphAM` and `graphNEL` from the package **graph**)

`signature(W = "matrix")` a matrix representing the adjacency matrix of the graph

References

Chua, H., Sung, W., & Wong, L. An efficient strategy for extensive integration of diverse biological data for protein function prediction. Bioinformatics, 23 , 3364–3373, 2007.

Examples

```
library(bionetdata);
data(Yeast.Biogrid.data);
W <- Chua.norm(Yeast.Biogrid.data);
```

Do.sim.matrix.Pearson *Construction of the Pearson correlation matrix*

Description

Function to obtain the Pearson correlation matrix between rows of a given matrix.

Usage

```
Do.sim.matrix.Pearson(M, cut = TRUE, remove.negatives = TRUE, min.thresh = 0)
```

Arguments

`M`	input matrix
`cut`	if TRUE (def.) at least one edge is maintained for each node, all the other edges are set to 0. If false no edgeis set to 0.
`remove.negatives`	if TRUE (def) negative values are replaced with 0 in the correlation matrix
`min.thresh`	minimum allowed threshold (def. 0). If a threshold lower than min.thresh is selected, thanit is substituted by min.thresh. Warning: setting min.thresh to large values may lead to highly disconneted network

Details

You can also "sparsify" the matrix, by putting to 0 all the weights, by setting a threshold such that at least one edge is maintained for each node. The diagonal values are set to 0.

Value

a square symmetric matrix of the Pearson correlation coefficients computed between the rows of M

Examples

```
# a gaussian random matrix
D <- matrix(rnorm(20000),nrow=200);
W <- Do.sim.matrix.Pearson (D);
# the same without default parameters
W2 <- Do.sim.matrix.Pearson (D, cut=FALSE, remove.negatives=FALSE, min.thresh=-20);
```

`Laplacian.norm-methods`

Laplacian graph normalization

Description

Methods to normalize weights of square symmetric adjacency matrices. A network matrix is normalized by dividing each entry W_{ij} by the square root of the product of the sum of elements of row i and the sum of the elemnts in column j. In other words if D is a diagonal matrix such that $D_{ii} = \sum_j W_{ij}$, then the normalize matrix is:

$$W_{norm} = D^{-1/2} W D^{-1/2}$$

Usage

```
Laplacian.norm(W)
```

Arguments

`W`	an object representing the graph to be normalized

Value

The normalized adjacency matrix of the network

Methods

`signature(W = "graph")` an object of the virtual class graph (hence including objects of class `graphAM` and `graphNEL` from the package **graph**)

`signature(W = "matrix")` a matrix representing the adjacency matrix of the graph

Examples

```
library(bionetdata);
# normalization of drug-drug similarity networks
data(DD.chem.data);
W <- Laplacian.norm(DD.chem.data);
# the same using an object of class graphAM
g <- new("graphAM", adjMat=DD.chem.data, values=list(weight=DD.chem.data));
Wg <- Laplacian.norm(g);
```

Magnify.binary.features.norm-methods
Normalization of binary matrices

Description

Normalization of binary matrices according to the procedure described in Mostafavi et al. 2008. Having a binary matrix M, for each feature, if b is the proportion of 1, then ones are replaced with -log(b) and zeros with log(1-b).

Usage

```
Magnify.binary.features.norm(M)
```

Arguments

M	an object representing the matrix to be normalized

Value

The normalized matrix

Methods

signature(M = "matrix")	Input binary matrix. Rows are examples, columns features

References

Mostafavi, S., Ray, D., Warde-Farley, D., Grouios, C., & Morris, Q. GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. Genome Biology, 9, 2008.

Examples

```
D <- matrix(ifelse(runif(40000)>0.9,1,0),nrow=100);
M <- Magnify.binary.features.norm(D);
```

`Max.Min.norm-methods` *Max-min graph normalization*

Description

Graph normalization with respect to the minimum and maximum value of its weights. Each entry of the normalized matrix is in the range $[0..1]$:

$$W_{norm} = \frac{(W - \min(W))}{(\max(W) - \min(W))}$$

Usage

```
Max.Min.norm(W)
```

Arguments

`W`	an object representing the graph to be normalized

Value

The normalized adjacency matrix of the network

Methods

`signature(W = "graph")` an object of the virtual class graph (hence including objects of class `graphAM` and `graphNEL` from the package **graph**)

`signature(W = "matrix")` a matrix representing the adjacency matrix of the graph

Examples

```
library(bionetdata);
# max-min normalization for a drug-drug similarity network
data(DD.chem.data);
W <- Max.Min.norm(DD.chem.data);
# the same using an object of class graphAM

g <- new("graphAM", adjMat=DD.chem.data, values=list(weight=DD.chem.data));
Wg <- Max.Min.norm(g);
```

Prob.norm-methods *Probabilistic normalization of a graph*

Description

Method to compute the transition probability matrix of network. A network matrix is normalized by dividing each entry W_{ij} by the the sum of elements of row i In other words if D is a diagonal matrix such that $D_{ii} = \sum_j W_{ij}$ then the normalize matrix is:

$$W_{norm} = D^{-1}W$$

Usage

```
Prob.norm(W)
```

Arguments

W	an object representing the graph to be normalized

Value

The normalized transition probability matrix of network

Methods

`signature(W = "graph")` an object of the virtual class graph (hence including objects of class graphAM and graphNEL from the package **graph**)

`signature(W = "matrix")` a matrix representing the adjacency matrix of the graph

Examples

```
library(bionetdata);
# making transition prob matrix for a drug-drug similarity network
data(DD.chem.data);
W <- Prob.norm(DD.chem.data);
# the same using an object of class graphAM and of class graphNEL

g <- new("graphAM", adjMat=DD.chem.data, values=list(weight=DD.chem.data));
Wg <- Prob.norm(g);
g2 <- as(g, "graphNEL");
Wg2 <- Prob.norm(g2);
```

Sparsify.matrix-methods *Sparsifying the graph*

Description

Methods to sparsify a network matrix. By this method a general threshold is et such that you a minimum of k edges is guranteed for each node

Usage

```
Sparsify.matrix(W, k=1)
```

Arguments

W	an object representing the graph to be sparsified
k	the number of guaranteed edges for each node (def.=1)

Value

The sparsified adjacency matrix of the network

Methods

`signature(W = "graph")` an object of the virtual class graph (hence including objects of class graphAM and graphNEL from the package **graph**)

`signature(W = "matrix")` a matrix representing the adjacency matrix of the graph

Examples

```
library(bionetdata);
data(FIN.data);
W <- Laplacian.norm(as.matrix(FIN.data));
# sparsification by maintaining at least one neighbour per node
W1 <- Sparsify.matrix(W);
# sparsification by maintaining at least 20 neighbours per node (if any)
W20 <- Sparsify.matrix(W, k=20);
```

Sparsify.matrix.fixed.neighbours-methods
Sparsifying the graph by a fixed number of edges per node

Description

Methods to sparsify a network matrix by fixing the number of edges for each node. It selects the first k neighbours for each node (by row) according to the weight of the edge By this function you select exactly k edges for each node (if there are at least k edges in the adjacency matrix). The resulting matrix is not symmetric.

Usage

```
Sparsify.matrix.fixed.neighbours(W, k=10)
```

Arguments

W	an object representing the graph to be normalized
k	the number of edges for each node (def.=10)

Value

a sparsified matrix (Warning: the matrix is not symmetric)

Methods

`signature(W = "graph")` an object of the virtual class graph (hence including objects of class `graphAM` and `graphNEL` from the package **graph**)

`signature(W = "matrix")` a matrix representing the adjacency matrix of the graph

Examples

```
library(bionetdata);
data(FIN.data);
W <- Laplacian.norm(as.matrix(FIN.data));
# sparsification with 10 neighbours per node
W10 <- Sparsify.matrix.fixed.neighbours(W);
# sparsification with 20 neighbours per node
W20 <- Sparsify.matrix.fixed.neighbours(W, k=20);
```

Index