Package ‘INetTool’

June 19, 2025

Title Integration Network

Version 0.1.1

Maintainer Valeria Policastro <valeria.policastro@gmail.com>

Description It constructs a Consensus Network which identifies the general information of all the layers and Specific Networks for each layer with the information present only in that layer and not in all the others.The method is described in Policastro et al. (2024) ``INet for network integration" <doi:10.1007/s00180-024-01536-8>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports igraph, r2r, ggplot2, parallel, ggpubr, multinet, robin

VignetteBuilder knitr

Suggests knitr, rmarkdown

LazyData true

NeedsCompilation no

Author Valeria Policastro [aut, cre] (ORCID:
<https://orcid.org/0000-0003-1016-460X>),
Matteo Magnani [aut],
Claudia Angelini [aut],
Annamaria Carissimo [aut]

Repository CRAN

Date/Publication 2025-06-19 08:00:02 UTC

Contents

adjL_data 2
adj_rename 2
consensusNet 3
constructionGraph 4
densityNet 4

exampleL_data 5
graphL_data 5
JWmatrix 6
JWmean 6
measuresNet 7
plotC 8
plotINet 8
plotL 9
specificNet 10
thresholdNet 10
tryL_data 11

Index **12**

`adjL_data` *Adjacency Data*

Description

List of 2 adjacency matrices data type.

Usage

```
adjL_data
```

Format

‘adjL_data‘ A list of 2 objects:

AdjMatrix1 Adjacency matrix;

AdjMatrix2 Adjacency matrix.

`adj_rename` *adj_rename*

Description

This function constructs a list of adjacency matrices with the same row and column names for all the matrices. The output is the object needed for `consensusNet` function.

Usage

```
adj_rename(adjL)
```

Arguments

`adjL`	list of adjacency matrices

Value

a list of adjacency matrices with the same rows and columns name.

Examples

```
data("tryL_data")
adj_rename(tryL_data)
```

---

consensusNet *consensusNet*

Description

This function computes the INet Algorithm for the construction of a **Consensus Network**.

Usage

```
consensusNet(
  adjL,
  threshold = 0.5,
  tolerance = 0.1,
  theta = 0.04,
  nitermax = 50,
  ncores = 2,
  verbose = TRUE
)
```

Arguments

`adjL`	list of weighted adjacency matrix with the same name in rows and columns for all the matrices.
`threshold`	threshold for the construction of the Consensus (default 0.5). Used in the last step on the similar graphs.
`tolerance`	the tolerance of differences between similar graphs for the construction of the Consensus (default 0.1).
`theta`	importance to give to the neighbourhood part of the weight (default 0.04).
`nitermax`	maximum number of iteration before stopping the algorithm (default 50).
`ncores`	number of CPU cores to use (default is 2). We suggest to use ncores equal to the number of graphs to integrate.
`verbose`	flag for verbose output (default as TRUE).

Value

a list of 3 types: $graphConsensus the Consensus Network, $Comparison the Jaccard weighted distances between the graphs calculated in each iteration, $similarGraphs the similar graphs before the Thresholding

Examples

```
data("adjL_data")
consensusNet(adjL_data)
```

`constructionGraph` *constructionGraph*

Description

This function constructs graphs from data with pearson correlation and proportional thresholding (the data should be with the same names (the nodes) in columns for all the matrices).

Usage

```
constructionGraph(data, perc = 0.95)
```

Arguments

`data`	a list of datasets
`perc`	percentile (default 0.95 it takes the 5 percent of the highest weights)

Value

Threshold information (highest weight, number of edges, number of nodes, modularity with louvain method), graphs in a list for each layer and weighted adjacency matrices in a list for each layer.

Examples

```
data("exampleL_data")
constructionGraph(exampleL_data)
```

`densityNet` *densityNet*

Description

This function creates a density plot of the different graphs mean weights. It can be used to search the final Threshold for the Consensus Network starting from similar networks.

Usage

```
densityNet(graphL)
```

Arguments

`graphL`	the list of weighted graphs in igraph format.

Value

the quantile of the mean density distribution, the quantile of the mean density distribution without the zeros, plot density distribution without the zeros

Examples

```
data("graphL_data")
densityNet(graphL_data)
```

`exampleL_data` *Example Data*

Description

3 data types: Gene_Expression, Methy_Expression and Mirna_Expression data from patients with Glioblastoma

Usage

```
exampleL_data
```

Format

‘exampleL_data‘ A list of 3 objects:

Gene_Expression subset of Gene expression data;

Methy_Expression subset of Methylation data;

Mirna_Expression subset of Mirna data.

Source

<https://portal.gdc.cancer.gov/>

`graphL_data` *Graph Data*

Description

List of 2 graphs of igraph class type.

Usage

```
graphL_data
```

Format

‘graphL_data‘ A list of 2 objects:

Graph1 Graph firt layer;

Graph2 Graph second layer.

`JWmatrix` *JWmatrix*

Description

This function computes the Jaccard weighted matrix distance between all the pairs of graphs.

Usage

```
JWmatrix(graphL)
```

Arguments

`graphL`	list of graphs as igraph objects with the same nodes.

Value

weighted Jaccard distance matrix

Examples

```
data("graphL_data")
JWmatrix(graphL_data)
```

`JWmean` *JWmean*

Description

This function computes the Mean Weighted Jaccard Distance for Multilayer Networks.

Usage

```
JWmean(graphL)
```

Arguments

`graphL`	list of different graphs in igraph format with same nodes.

Value

a number: the mean distance

Examples

```
data("graphL_data")
JWmean(graphL_data)
```

---

measuresNet *measuresNet*

Description

This function computes graphs and nodes measures to analyse all the layers in one shot.

Usage

```
measuresNet(graphL, nodes.measures = TRUE)
```

Arguments

`graphL`	a list of graphs as igraphs objects.
`nodes.measures`	logical, if falso it computes only graph measures, if true it computes also nodes measures (default TRUE).

Value

list of measure for each layer.

Examples

```
data("graphL_data")
measuresNet(graphL_data)
```

plotC *plotC*

Description

The function plots the network without isolated nodes.

Usage

```
plotC(graph, ...)
```

Arguments

graph	a graph
...	other parameter

Value

plot

Examples

```
data("graphL_data")
plotC(graphL_data[[1]])
```

plotINet *plotINet*

Description

The function plots a beginning network and the consensus in one graph with different edge colours: red edges represent edges of the consensus already present in the beginning one, while light blue edges represent new edges constructed from the consensus.

Usage

```
plotINet(
  adj,
  graph.consensus,
  edge.width = 3,
  vertex.label.cex = 0.5,
  vertex.size = 10,
  edge.curved = 0.2,
  method = "NA",
  ...
)
```

Arguments

`adj`	one of the beginning adjacency matrices
`graph.consensus`	consensus network, output of the `consensusNet` function
`edge.width`	the edge width (default 3)
`vertex.label.cex`	the size of the vertex label (default 0.8)
`vertex.size`	the size of the vertex (default 10)
`edge.curved`	to make the edge curved (default 0.2)
`method`	community detection method to color the nodes one of "walktrap", "edgeBetweenness", "fastGreedy", "louvain", "spinglass", "leadingEigen", "labelProp", "infomap", "optimal" and "leiden" (default no method)
`...`	other parameter

Value

Union graph beginning and consensus edge coloured, green edges consensus already present in the beginning, blue edges new of the consensus. Community detection of the beggining graph if added.

Examples

```
data("adjL_data")
con <- consensusNet(adjL_data)
plotINet(adjL_data[[1]], con$graphConsensus)
```

`plotL` *plotL*

Description

This function plots all the layers in one plot.

Usage

```
plotL(graphL, ...)
```

Arguments

`graphL`	List of graphs
`...`	other parameter

Value

plot of graphs

Examples

```
data("graphL_data")
plotL(graphL_data)
```

`specificNet` *specificNet*

Description

The function creates Case Specific Networks one for each layer to give information of the peculiar layer not present in the Consensus.

Usage

```
specificNet(graphL, graph.consensus)
```

Arguments

`graphL`	a list of graphs as igraphs objects.
`graph.consensus`	graphConsensus output of the `consensusNet` function.

Value

Case Specific Networks one for each layer and percentage of specificity.

Examples

```
data("graphL_data")
data("adjL_data")
myConsensus <- consensusNet(adjL_data)
specificNet(graphL_data, myConsensus$graphConsensus)
```

`thresholdNet` *thresholdNet*

Description

The function reconstructs the Consensus Network with different thresholding after the `consensusNet` function starting from similar graphs.

Usage

```
thresholdNet(sim.graphL, threshold = 0.5)
```

Arguments

`sim.graphL`	a list of similarGraphs output of the `consensusNet` function.
`threshold`	different threshold to compute.

Value

a new consensus network igraph object.

Examples

```
data("adjL_data")
myConsensus <- consensusNet(adjL_data)
thresholdNet(myConsensus$similarGraphs)
```

`tryL_data` *try Data*

Description

Random data with different nodes name in a list of 2 adjacency matrices.

Usage

```
tryL_data
```

Format

`tryL_data` A list of 2 objects:

AdjMatrix1	Adjacency matrix;
AdjMatrix2	Adjacency matrix.

Index

∗ **datasets**
    adjL_data, 2
    exampleL_data, 5
    graphL_data, 5
    tryL_data, 11

adj_rename, 2
adjL_data, 2

consensusNet, *2*, 3, *9–11*
constructionGraph, 4

densityNet, 4

exampleL_data, 5

graphL_data, 5

JWmatrix, 6
JWmean, 6

measuresNet, 7

plotC, 8
plotINet, 8
plotL, 9

specificNet, 10

thresholdNet, 10
tryL_data, 11