Package 'NetSci'

October 12, 2022

Type Package

Title Calculates Basic Network Measures Commonly Used in Network Medicine

Version 1.0.0

Author Deisy Gysi

Maintainer Deisy Morselli Gysi <deisy.ccnr@gmail.com>

Description Calculates network measures such as Largest Connected Component (LCC), Proximity, Separation, Jaccard Index,
along with permutation, when needed.

Imports igraph, magrittr, wTO, dplyr, Rfast, utils, binr, cubature

Suggests CoDiNA

License GPL-2

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2022-07-03 18:00:02 UTC

R topics documented:

`avr_proximity_multiple_target_sets`
avr_proximity_multiple_target_sets

Description

Calculates the average proximity from a set of targets to a set of source nodes. It is calculate using a degree preserving randomization. It is calculated as described in Guney, E. et al (2016) <doi.org:10.1038/ncomms10331>

Usage

```
avr_proximity_multiple_target_sets(
  set,
  G,
  ST,
  source,
  N = 1000,
  bins = 100,
  min_per_bin = 20,
  weighted = FALSE
)
```

Arguments

`set`	Name of the sets you have targets for. (In a drug-target setup, those would be the drugs of interest).
`G`	The original graph (often an interactome).
`ST`	Set-Target data. It is a data.frame with two columns. ID and Target.
`source`	The source nodes (disease genes).
`N`	Number of randomizations.
`bins`	the number os bins for the degree preserving randomization.
`min_per_bin`	the minimum size of each bin.
`weighted`	consider a weighted graph? TRUE/FALSE

Value

proximity and its significance based on the degree preserving randomization.

Examples

```
set.seed(666)
net  = data.frame(
Node.1 = sample(LETTERS[1:15], 15, replace = TRUE),
Node.2 = sample(LETTERS[1:10], 15, replace = TRUE))
net$value = 1
net =  CoDiNA::OrderNames(net)
net = unique(net)
net$weight = runif(nrow(net))

g <- igraph::graph_from_data_frame(net, directed = FALSE )
S = c("N", "A", "F", "I")
T1 = data.frame(ID = "T1", Target = c("H", "M"))
T2 = data.frame(ID = "T2", Target = c("G", "O"))

avr_proximity_multiple_target_sets(set = c('T1', 'T2'),
G = g,
 source = S,
 ST = rbind(T1,T2),
 bins = 1,
 min_per_bin = 2)

# In a weighted graph
# avr_proximity_multiple_target_sets(set = c('T1', 'T2'),
# G = g,
#  source = S,
#  ST = rbind(T1,T2),
#  bins = 1,
#  min_per_bin = 2,
#  weighted = TRUE)
```

`extract_LCC` *Extract LCC from a graph*

Description

Extract LCC from a graph

Usage

```
extract_LCC(g)
```

Arguments

g	is the graph you want to extract the largest connected component

Value

a graph (from igraph) with only the largest connected component

Examples

```
set.seed(12)
x = data.frame(n1 = sample(LETTERS[1:5]),
               n2 =  sample(LETTERS[1:20]))

g = igraph::graph_from_data_frame(x, directed = FALSE)
g = igraph::simplify(g)
LCC = extract_LCC(g)
```

Histogram_LCC *Histogram_LCC*

Description

Plots the histogram to evaluate the significance of the Largest Connected Component (LCC).

Usage

```
Histogram_LCC(LCC_L, Name = NULL)
```

Arguments

LCC_L	an output from the function LCC_Significance or LCC_Bipartide
Name	title of the plot

Value

An Histogram of the simulated LCC, and a red line of the actual LCC.

Examples

```
set.seed(666)
net  = data.frame(
Node.1 = sample(LETTERS[1:15], 15, replace = TRUE),
Node.2 = sample(LETTERS[1:10], 15, replace = TRUE))
net$value = 1
net =  CoDiNA::OrderNames(net)
net = unique(net)

g <- igraph::graph_from_data_frame(net, directed = FALSE )
targets = c("N", "A", "I", "F")
LCC_Out = LCC_Significance(N = 1000,
                Targets = targets,
                                G = g,
                                bins = 5,
                                min_per_bin = 2)
                                # in a real interactome, please use the default
```

```
Histogram_LCC(LCC_Out, "Example")
```

`Hypergeometric.test` *Hypergeometric.test*

Description

Calculates the significance of an overlap of two sets using an hypergeometric test. It is a wrapper of the ‘phyper‘ function.

Usage

```
Hypergeometric.test(
  success,
  universe_success,
  universe_failure,
  size_collected,
  lower.tail = FALSE
)
```

Arguments

`success`	Is the number of elements in the overlap of the sets.
`universe_success`	Is the number of elements of the set of interest.
`universe_failure`	Is the number of elements of the set of the other set.
`size_collected`	The total of elements in the universe
`lower.tail`	Should the test be calculated on the lower tail? (Hypothesis test is lower than)

Value

the p-value for the hypergeometric test.

Examples

```
require(magrittr)
s = 10; S = 15; f = 10; T = 30
Hypergeometric.test(success = s,
universe_success = S,
universe_failure = f,
size_collected = T
)
```

Jaccard *Jaccard*

Description

Calculates the Jaccard index between different sets.

Usage

```
Jaccard(Data)
```

Arguments

Data	A data.frame with 2 columns. The first refers to the set and the second the elements

Value

a data.frame with the set names and their Jaccard index

Examples

```
set.seed(123)
Data = data.frame(Class = sample(c("X", "Y", "Z"), replace = TRUE, size = 50),
                  Element = sample(LETTERS[1:15], replace = TRUE, size = 50))
Data = unique(Data)
Jaccard(Data)
```

LCC_Significance *LCC Significance*

Description

Calculates the Largest Connected Component (LCC) from a given graph, and calculates its significance using a degree preserving approach. Menche, J., et al (2015) <doi.org:10.1126/science.1065103>

Usage

```
LCC_Significance(
  N = N,
  Targets = Targets,
  G,
  bins = 100,
  hypothesis = "greater",
  min_per_bin = 20
)
```

Arguments

N	Number of randomizations.
Targets	Name of the nodes that the subgraph will focus on - Those are the nodes you want to know whether if forms an LCC.
G	The graph of interest (often, in NetMed it is an interactome - PPI).
bins	the number os bins for the degree preserving randomization. When bins = 1, assumes a uniform distribution for nodes.
hypothesis	are you expecting an LCC greater or smaller than the average?
min_per_bin	the minimum size of each bin.

Value

a list with the LCC - $LCCZ all values from the randomizations - $mean the average LCC of the randomizations - $sd the sd LCC of the randomizations - $Z The score - $LCC the LCC of the given targets - $emp_p the empirical p-value for the LCC - $rLCC the relative LCC

Examples

```
set.seed(666)
net  = data.frame(
Node.1 = sample(LETTERS[1:15], 15, replace = TRUE),
Node.2 = sample(LETTERS[1:10], 15, replace = TRUE))
net$value = 1
net =  CoDiNA::OrderNames(net)
net = unique(net)

g <- igraph::graph_from_data_frame(net, directed = FALSE )
plot(g)
targets = c("I", "H", "F", "E")
 LCC_Significance(N = 100,
                  Targets = targets,
                                G = g,
                               bins = 1,
                               min_per_bin = 2)
```

NetSci *Global Definition*

Description

Basic global variables to make sure the package runs.

proximity_average *Proximity from target to source*

Description

Calculates the proximity (average or closest) from source to targets.

Usage

```
proximity_average(G, source, targets)
```

Arguments

G	The original graph (often an interactome).
source	nodes from the network (in a drug repurpusing set-up those are the disease genes)
targets	targets in the network (in a drug repurpusing set-up those are the drug-targets)

Value

the proximity value for the source-targets

Examples

```
#' set.seed(666)
net  = data.frame(
Node.1 = sample(LETTERS[1:15], 15, replace = TRUE),
Node.2 = sample(LETTERS[1:10], 15, replace = TRUE))
net$value = 1
net =  CoDiNA::OrderNames(net)
net = unique(net)

g <- igraph::graph_from_data_frame(net, directed = FALSE )
T = c("G", "A", "D")
S = c("C", "M")
proximity_average(g, source = S, targets = T)
```

proximity_average_weighted *Proximity from target to source*

Description

Calculates the weighted average proximity from source to targets.

Usage

```
proximity_average_weighted(G, source, targets)
```

Arguments

G	The original graph (often a weighted interactome).
source	nodes from the network (in a drug repurpusing set-up those are the disease genes)
targets	targets in the network (in a drug repurpusing set-up those are the drug-targets)

Value

the proximity value for the source-targets

Examples

```
set.seed(666)
net  = data.frame(
Node.1 = sample(LETTERS[1:15], 15, replace = TRUE),
Node.2 = sample(LETTERS[1:10], 15, replace = TRUE))
net$value = 1
net =  CoDiNA::OrderNames(net)
net = unique(net)
net$weight = runif(nrow(net))
g <- igraph::graph_from_data_frame(net, directed = FALSE )
T = c("G", "A", "D")
S = c("C", "M")
proximity_average_weighted(g, source = S, targets = T)
```

---

proximity_close *Proximity from target to source*

Description

Calculates the proximity (average or closest) from source to targets.

Usage

```
proximity_close(G, source, targets)
```

Arguments

G	The original graph (often an interactome).
source	nodes from the network (in a drug repurpusing set-up those are the disease genes)
targets	targets in the network (in a drug repurpusing set-up those are the drug-targets)

Value

the proximity value for the source-targets

Examples

```
set.seed(666)
net  = data.frame(
Node.1 = sample(LETTERS[1:15], 15, replace = TRUE),
Node.2 = sample(LETTERS[1:10], 15, replace = TRUE))
net$value = 1
net =  CoDiNA::OrderNames(net)
net = unique(net)

g <- igraph::graph_from_data_frame(net, directed = FALSE )
T = c("G", "A", "D")
S = c("C", "M")
proximity_close(g, source = S, targets = T)
```

`separation` *Separation*

Description

Calculates the separation of two set of targets on a network. Often used to measure separation of disease modules in a interactome. Separation is calculated as in Menche, J. et al (2015) <doi:10.1126/science.1257601>.

Usage

```
separation(G, ST)
```

Arguments

`G`	The original graph (often an interactome).
`ST`	Set-Target data. It is a data.frame with two columns. ID and Target.

Value

the separation and distance of modules.

Examples

```
require(magrittr)
set.seed(12)
x = data.frame(n1 = sample(LETTERS[1:5]),
               n2 =  sample(LETTERS[1:20]))

D1 = data.frame(gene = c("H", "I", "S", "N", "A"), disease = "D1")
D2 = data.frame(gene = c("E", "C",  "R" , "J", "Q", "O"), disease = "D2")
D3 = data.frame(gene = c("E", "G", "T", "P"), disease = "D3")
```

```
D4 = data.frame(gene = c("A", "B", "E"), disease = "D4")

Diseases = rbind(D1, D2, D3, D4)
Diseases %<>% dplyr::select(disease, gene)
g = igraph::graph_from_data_frame(x, directed = FALSE)
g = igraph::simplify(g)

separation(G = g, ST = Diseases)
```

`separation_Significance` *Separation Significance*

Description

Calculates the separation of two set of targets on a network and assigns a p-value to it. Often used to measure separation of disease modules in a interactome. Separation is calculated as in Menche, J. et al (2015) <doi:10.1126/science.1257601>. p-values are calculates based on the permutation of nodes, you can set the full network to be in the set for permutation or can select the ones you include as input.

Usage

```
separation_Significance(G, ST, Threads = 2, N = 1000, correct_by_target = TRUE)
```

Arguments

`G`	The original graph (often an interactome / PPI).
`ST`	Set-Target data. It is a data.frame with two columns. ID and Target.
`Threads`	How many threads you'd like to use (for parallel computation).
`N`	default to 1000. The number of permutations
`correct_by_target`	TRUE by default. If you want to use the set of targets for the permutation or the full network.

Examples

```
set.seed(12)
require(magrittr)
x = data.frame(n1 = sample(LETTERS[1:5]),
               n2 =  sample(LETTERS[1:20]))

D1 = data.frame(gene = c("H", "I", "S", "N", "A"), disease = "D1")
D2 = data.frame(gene = c("E", "C",  "R" , "J", "Q", "O"), disease = "D2")
D3 = data.frame(gene = c("E", "G", "T", "P"), disease = "D3")
D4 = data.frame(gene = c("A", "B", "E"), disease = "D4")
D5 = data.frame(gene = c("D", "F", "L"), disease = "D5")
D6 = data.frame(gene = c("D", "F", "K"), disease = "D6")
```

```
D7 = data.frame(gene = c("A", "B" ,"F", "K"), disease = "D7")

Diseases = rbind(D1, D2, D3, D4, D5, D6, D7)
Diseases %<>% dplyr::select(disease, gene)
g = igraph::graph_from_data_frame(x, directed = FALSE)
g = igraph::simplify(g)

separation_Significance(G = g,
ST = Diseases,
correct_by_target = FALSE,
Threads = 2)
```

Index