Package ‘backbone’
March 22, 2022

Type Package
Title Extracts the Backbone from Graphs
Version 2.0.3
Description An implementation of methods for extracting an unweighted unipartite
graph (i.e. a backbone) from an unweighted unipartite graph (e.g., Hamann et al.,
2016 <doi:10.1007/s13278-016-0332-2>), a weighted unipartite graph (e.g.,
Serrano et al., 2009 <doi:10.1073/pnas.0808904106>), the projection of an unweighted
bipartite graph (e.g., Neal et al., <doi:10.1038/s41598-021-03238-3>, or the projection
of a weighted bipartite graph (e.g., Neal, 2017 <doi:10.1177/0308518X16631339>).
License GPL-3
Encoding UTF-8
RoxygenNote 7.1.2
Depends R (>= 2.10)
Imports igraph, Matrix, methods, network, MASS, PoissonBinomial,
  stats, Rcpp, utils,
Suggests knitr, rmarkdown, testthat (>= 3.0.0)
LinkingTo Rcpp
VignetteBuilder knitr
URL https://www.zacharyneal.com/backbone,
  https://github.com/zpneal/backbone
BugReports https://github.com/zpneal/backbone/issues
Config/testthat/edition 3
NeedsCompilation yes
Author Zachary Neal [aut, cre] (<https://orcid.org/0000-0003-3076-4995>),
  Rachel Domagalski [aut],
  Bruce Sagan [aut],
  Karl Godard [ctb]
Maintainer Zachary Neal <zpneal@msu.edu>
Repository CRAN
Date/Publication 2022-03-22 14:10:08 UTC
Description

Provides methods for extracting from an unweighted and sparse subgraph (i.e., a backbone) that contains only the most “important” edges in a weighted bipartite projection, a non-projection weighted network, or an unweighted network.

Available backbone extraction functions include:

- For weighted bipartite projections of weighted bipartite networks: osdsm().
- For weighted bipartite projections of binary bipartite networks: fixedfill(), fixedrow(), fixedcol(), sdsm(), and fdsm().
• For non-projection weighted networks: global(), disparity().
• For unweighted networks: sparsify(), sparsify.with.skeleton(), sparsify.with.gspar()
  sparsify.with.lspar(), sparsify.with.simmelian(), sparsify.with.jaccard(), sparsify.with.meetmin()
  sparsify.with.geometric(), sparsify.with.hypergeometric(), sparsify.with.localdegree(),
  sparsify.with.quadrilateral().
• For all networks: backbone.suggest() will examine the data and suggest an appropriate
  backbone function.

The package also includes some utility functions:

• fastball() - Fast marginal-preserving randomization of binary matrices
• bicm() - Compute probabilities under the bipartite configuration model
• bipartite.from.probability(), bipartite.from.sequence(), and bipartite.from.distribution() - Generate random bipartite networks with given edge probability, degree sequences, or degree distributions.

For additional documentation and background on the package functions, see vignette("backbone").
For updates, papers, presentations, and other backbone news, please see www.rbackbone.net

References


Sequence Model for Extracting the Backbone of Bipartite Projections. *Scientific Reports*, 11,
23929. doi: 10.1038/s41598021032383

backbone.extract

Extracts a backbone network from a backbone object

Description

backbone.extract returns a binary or signed adjacency matrix containing the backbone that re-
tains only the significant edges.

Usage

backbone.extract(  
  bb.object,
  signed = FALSE,
  alpha = 0.05,
  mtc = "none",
  class = "matrix"
)
Arguments

bb.object backbone: backbone S3 class object.
signed Boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)
alpha Real: significance level of hypothesis test(s)
mtc string: type of Multiple Test Correction to be applied; can be any method allowed by p.adjust.
class string: the class of the returned backbone graph, one of c("matrix", "sparseMatrix", "igraph", "network", "edgelist"), converted via tomatrix.

Details

The "backbone" S3 class object is composed of three matrices (the weighted graph, edges’ upper-tail p-values, edges’ lower-tail p-values), and a string indicating the null model used to compute p-values.

When signed = FALSE, a one-tailed test (is the weight stronger) is performed for each edge with a non-zero weight. It yields a backbone that preserves edges whose weights are significantly stronger than expected in the chosen null model. When signed = TRUE, a two-tailed test (is the weight stronger or weaker) is performed for each every pair of nodes. It yields a backbone that contains positive edges for edges whose weights are significantly stronger, and negative edges for edges whose weights are significantly weaker, than expected in the chosen null model. NOTE: Before v2.0.0, all significance tests were two-tailed and zero-weight edges were evaluated.

Value

backbone graph: Binary or signed backbone graph of class given in parameter class.

Examples

# A binary bipartite network of 30 agents & 75 artifacts; agents form three communities
B <- rbind(cbind(matrix(rbinom(250,1,.8),10),
            matrix(rbinom(250,1,.2),10),
            matrix(rbinom(250,1,.2),10)),
            cbind(matrix(rbinom(250,1,.2),10),
                  matrix(rbinom(250,1,.8),10),
                  matrix(rbinom(250,1,.2),10)),
            cbind(matrix(rbinom(250,1,.2),10),
                  matrix(rbinom(250,1,.2),10),
                  matrix(rbinom(250,1,.8),10)))

backbone.object <- fixedrow(B, alpha = NULL)
bb <- backbone.extract(backbone.object, alpha = 0.05)
backbone.suggest  Suggest a backbone model

Description

backbone.suggest suggests and optionally runs an appropriate backbone model for a graph object.

Usage

backbone.suggest(G, s = NULL)

Arguments

G  graph: A graph represented in an object of class matrix, sparse Matrix, dataframe, igraph, or network.

s  numeric: If provided, a backbone is extracted using this value as the significance level or sparsification parameter.

Value

If s == NULL: NULL, but a message is displayed with a suggested model. If 0 <= s <= 1: A binary backbone graph in the same class as G, obtained by extracting the backbone at the s significance level (if a statistical model is suggested) or using sparsification parameter s (if a sparsification model is suggested). The code used to perform the extraction, and suggested manuscript text are displayed.

Examples

M <- matrix(runif(100),10,10)  #A random weighted, directed graph
backbone <- backbone.suggest(M)
backbone <- backbone.suggest(M, s = 0.05)

bicm  Bipartite Configuration Model

Description

bicm estimates cell probabilities under the bipartite configuration model

Usage

bicm(M, tol = 1e-08, max_steps = 200, ...)
**Arguments**

- **M**: matrix: a binary matrix
- **tol**: numeric, tolerance of algorithm
- **max_steps**: numeric, number of times to run `loglikelihood_prime_bicm` algorithm
- ... optional arguments

**Details**

Given a binary matrix `M`, the Bipartite Configuration Model (BiCM; Saracco et al. 2015) returns a valued matrix `B` in which `B_{ij}` is the approximate probability that `M_{ij} = 1` in the space of all binary matrices with the same row and column marginals as `M`. The BiCM yields the closest approximations of the true probabilities compared to other estimation methods (Neal et al., 2021), and is used by `sdsm()` to extract the backbone of a bipartite projection using the stochastic degree sequence model.

**Value**

- `matrix`: a matrix of probabilities

**References**


**Examples**

```r
M <- matrix(rbinom(25,1,0.5),5,5) # A random bipartite graph
bicm(M)
```

**bipartite.add.blocks**: Adds a block structure to a bipartite network

**Description**

*bipartite.add.blocks* rewire a bipartite graph `B` to have a block structure such that edges are located within-block with density probability, while preserving both degree distributions.

**Usage**

```r
bipartite.add.blocks(B, blocks = 2, density = 0.5, max.tries = 1e+05)
```
Arguments

B  A bipartite network object of class "matrix", "sparseMatrix", igraph, matrix or dataframe edgelist, or network
blocks  integer: number of blocks to add (between 2 and 26)
density  numeric: desired within-block density
max.tries  numeric: number of ineligible re-wiring attempts before giving up

Details

Each row node and each column node are randomly assigned to one of blocks number of groups. Then degree-preserving checkerboard swaps are performed that increase the within-block density, until density is achieved. Eligible swaps are identified randomly, so the re-wiring can be slow when B is large. The process can get stuck when no eligible swaps remain but the target density has not been achieved; if this happens, increase max.tries to keep looking for eligible swaps or reduce the target density.

Examples

B <- bipartite.from.probability(R = 100, C = 100, P = .5)
B <- bipartite.add.blocks(B, blocks = 2, density = .7)

bipartite.from.distribution
Generates a bipartite network with given row and column degree distributions

Description

bipartite.from.distribution returns a bipartite graph, as an object of the requested class, with row and column degree distributions that approximately follow beta distributions with given parameters.

Usage

bipartite.from.distribution(
  R,
  C,
  P,
  rowdist = c(1, 1),
  coldist = c(1, 1),
  class = "matrix"
)
Arguments

\( R \)  
integer: number of rows

\( C \)  
integer: number of columns

\( P \)  
numeric: probability of an edge

rowdist  
vector length 2: Row degrees will approximately follow a Beta(a,b) distribution

coldist  
vector length 2: Column degrees will approximately follow a Beta(a,b) distribution

class  
string: the class of the returned backbone graph, one of c("matrix", "Matrix", "sparseMatrix", "igraph", "network")

Examples

\[
B \leftarrow \text{bipartite.from.distribution}(R = 100, C = 100, P = 0.1, 
\text{rowdist} = c(1,1), \text{coldist} = c(1,1)) \quad \# \text{Uniform}
\]

\[
B \leftarrow \text{bipartite.from.distribution}(R = 100, C = 100, P = 0.1, 
\text{rowdist} = c(1,10), \text{coldist} = c(1,10)) \quad \# \text{Right-tailed}
\]

\[
B \leftarrow \text{bipartite.from.distribution}(R = 100, C = 100, P = 0.1, 
\text{rowdist} = c(10,1), \text{coldist} = c(10,1)) \quad \# \text{Left-tailed}
\]

\[
B \leftarrow \text{bipartite.from.distribution}(R = 100, C = 100, P = 0.1, 
\text{rowdist} = c(10,10), \text{coldist} = c(10,10)) \quad \# \text{Normal}
\]

\[
B \leftarrow \text{bipartite.from.distribution}(R = 100, C = 100, P = 0.1, 
\text{rowdist} = c(10000,10000), \text{coldist} = c(10000,10000)) \quad \# \text{Constant}
\]

bipartite.from.probability

Generates a bipartite network with given edge probability

Description

bipartite.from.probability returns a bipartite graph, as an object of the requested class, in which each edge has a given probability and where no node is an isolate or maximally connected.

Usage

bipartite.from.probability(R, C, P = 0, class = "matrix")

Arguments

\( R \)  
integer: number of rows

\( C \)  
integer: number of columns

\( P \)  
numeric: probability of an edge; if \( P = 0 \) a probability will be chosen randomly

class  
string: the class of the returned backbone graph, one of c("matrix", "Matrix", "sparseMatrix", "igraph", "network").
bipartite.from.sequence

Examples

B <- bipartite.from.probability(R = 10, C = 10)
B <- bipartite.from.probability(R = 10, C = 10, P = .5)
B <- bipartite.from.probability(R = 10, C = 10, P = .5, class = "igraph")

bipartite.from.sequence

Generates a bipartite graph from row and column degree sequences

Description

bipartite.from.sequence returns a bipartite graph, as an object of the requested class, that has the given row and column degree sequences.

Usage

bipartite.from.sequence(R, C, class = "matrix")

Arguments

R numeric vector: requested row degree sequence of positive integers
C numeric vector: requested column degree sequence of positive integers
class string: the class of the returned backbone graph, one of c("matrix", "Matrix", "sparseMatrix", "igraph", "network")

Examples

B <- bipartite.from.sequence(R = c(1,1,2), C = c(1,1,2))
B <- bipartite.from.sequence(R = c(1,1,2), C = c(1,1,2), class = "igraph")

disparity

Extract backbone using the Disparity Filter

Description

disparity extracts the backbone of a weighted network using the Disparity Filter.

Usage

disparity(
    W,
    alpha = 0.05,
    signed = FALSE,
    mtc = "none",
    class = "original",
    narrative = FALSE
)


disparity

Arguments

\( W \)  
A weighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse Matrix; (2) an edgelist in the form of a three-column dataframe; (3) an igraph object; (4) a network object.

\( \alpha \)  
real: significance level of hypothesis test(s)

\( \text{signed} \)  
boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)

\( \text{mtc} \)  
string: type of Multiple Test Correction to be applied; can be any method allowed by \text{p.adjust}.

\( \text{class} \)  
string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as \( W \).

\( \text{narrative} \)  
boolean: TRUE if suggested text & citations should be displayed.

Details

The disparity function applies the disparity filter (Serrano et al., 2009), which compares an edge’s weight to its expected weight if a node’s total degree was uniformly distributed across all its edges. The graph may be directed or undirected, however the edge weights must be positive.

When \( \text{signed} = \text{FALSE} \), a one-tailed test (is the weight stronger) is performed for each edge with a non-zero weight. It yields a backbone that preserves edges whose weights are significantly stronger than expected in the chosen null model. When \( \text{signed} = \text{TRUE} \), a two-tailed test (is the weight stronger or weaker) is performed for each pair of nodes. It yields a backbone that contains positive edges for edges whose weights are significantly stronger, and negative edges for edges whose weights are significantly weaker, than expected in the chosen null model. \text{NOTE: Before } v2.0.0, \text{ all significance tests were two-tailed and zero-weight edges were evaluated}.

If \( W \) is an unweighted bipartite graph, any rows and columns that contain only zeros or only ones are removed, then the global threshold is applied to its weighted bipartite projection.

Value

If \( \alpha \neq \text{NULL} \): Binary or signed backbone graph of class \text{class}.

If \( \alpha = \text{NULL} \): An \text{S3} backbone object containing three matrices (the weighted graph, edges’ upper-tail p-values, edges’ lower-tail p-values), and a string indicating the null model used to compute p-values, from which a backbone can subsequently be extracted using \text{backbone.extract()}.

The signed, mtc, class, and narrative parameters are ignored.

References

Examples

#A network with heterogeneous (i.e. multiscale) weights
net <- matrix(c(0,10,10,10,75,0,0,0,0,
                             10,0,1,1,0,0,0,0,0,
                             10,1,0,1,0,0,0,0,0,
                             10,1,1,0,0,0,0,0,0,
                             10,1,1,1,0,0,0,0,0,
                             75,0,0,0,0,0,100,100,100,
                             0,0,0,0,0,100,0,10,10,
                             0,0,0,0,0,100,10,0,10,
                             0,0,0,0,0,100,10,10,0,
                             0,0,0,0,0,100,10,10,0),10)

net <- igraph::graph_from_adjacency_matrix(net, mode = "undirected", weighted = TRUE)
plot(net, edge.width = sqrt(igraph::E(net)$weight)) #A stronger clique & a weaker clique

strong <- igraph::delete.edges(net, which(igraph::E(net)$weight < mean(igraph::E(net)$weight)))
plot(strong) #A backbone of stronger-than-average edges ignores the weaker clique

bb <- disparity(net, alpha = 0.05, narrative = TRUE) #A disparity backbone...
plot(bb) #...preserves edges at multiple scales

Description

fastball randomizes a binary matrix, preserving the row and column sums

Usage

fastball(M, trades = 5 * nrow(M))

Arguments

M matrix: a binary matrix (see details)

trades integer: number of trades; the default is 5R trades (approx. mixing time)

Details

Given a matrix M, fastball randomly samples a new matrix from the space of all matrices with the same row and column sums as M.

Value

matrix: A random binary matrix with same row sums and column sums as M.
References


Examples

M <- matrix(rbinom(200,1,0.5),10,20) #A random 10x20 binary matrix
Mrand <- fastball(M) #Random matrix with same row and column sums

fdsm

Extract backbone using the Fixed Degree Sequence Model

Description

fdsm extracts the backbone of a bipartite projection using the Fixed Degree Sequence Model.

Usage

fdsm(
  B, 
  alpha = 0.05, 
  trials = NULL, 
  signed = FALSE, 
  mtc = "none", 
  class = "original", 
  narrative = FALSE, 
  ...
)

Arguments

B                 An unweighted bipartite graph, as: (1) an incidence matrix in the form of a matrix or sparse Matrix; (2) an edgelist in the form of a two-column dataframe; (3) an igraph object; (4) a network object. Any rows and columns of the associated bipartite matrix that contain only zeros are automatically removed before computations.
alpha            real: significance level of hypothesis test(s)
trials           numeric: the number of bipartite graphs generated to approximate the edge weight distribution. If NULL, the number of trials is selected based on alpha (see details)
signed           boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)
mtc               string: type of Multiple Test Correction to be applied; can be any method allowed by p.adjust.
class string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as B.

narrative boolean: TRUE if suggested text & citations should be displayed.

Details

The `fdsm` function compares an edge's observed weight in the projection $B^t B$ to the distribution of weights expected in a projection obtained from a random bipartite network where both the row vertex degrees and column vertex degrees are exactly fixed at their values in B. It uses the `fastball()` algorithm to generate random bipartite matrices with give row and column vertex degrees.

When `signed = FALSE`, a one-tailed test (is the weight stronger) is performed for each edge with a non-zero weight. It yields a backbone that preserves edges whose weights are significantly stronger than expected in the chosen null model. When `signed = TRUE`, a two-tailed test (is the weight stronger or weaker) is performed for each every pair of nodes. It yields a backbone that contains positive edges for edges whose weights are significantly stronger, and negative edges for edges whose weights are significantly weaker, than expected in the chosen null model. 

NOTE: Before v2.0.0, all significance tests were two-tailed and zero-weight edges were evaluated.

The p-values used to evaluate the statistical significance of each edge are computed using Monte Carlo methods. The number of trials performed affects the precision of these p-values, and the confidence that a given p-value is less than the desired alpha level. Because these p-values are proportions (i.e., the proportion of times an edge is weaker/stronger in the projection of a random bipartite graphs), evaluating the statistical significance of an edge is equivalent to comparing a proportion (the p-value) to a known proportion (alpha). When `trials = NULL`, the `power.prop.test` function is used to estimate the required number of trials to make such a comparison with a alpha type-I error rate, (1-alpha) power, and when the riskiest p-value being evaluated is at least 5% smaller than alpha. When any `mtc` correction is applied, for simplicity this estimation is based on a conservative Bonferroni correction.

Value

If `alpha != NULL`: Binary or signed backbone graph of class `class`.

If `alpha == NULL`: An S3 backbone object containing three matrices (the weighted graph, edges' upper-tail p-values, edges' lower-tail p-values), and a string indicating the null model used to compute p-values, from which a backbone can subsequently be extracted using `backbone.extract()`.

The `signed`, `mtc`, `class`, and `narrative` parameters are ignored.

References


`fastball`: Godard, Karl and Neal, Zachary P. 2022. fastball: A fast algorithm to sample bipartite graphs with fixed degree sequences. *arXiv:2112.04017*
Examples

# A binary bipartite network of 30 agents & 75 artifacts; agents form three communities
B <- rbind(cbind(matrix(rbinom(250,1,.8),10),
               matrix(rbinom(250,1,.2),10),
               matrix(rbinom(250,1,.2),10)),
     cbind(matrix(rbinom(250,1,.2),10),
               matrix(rbinom(250,1,.8),10),
               matrix(rbinom(250,1,.2),10)),
     cbind(matrix(rbinom(250,1,.2),10),
               matrix(rbinom(250,1,.2),10),
               matrix(rbinom(250,1,.8),10)))

P <- B%*%t(B) # An ordinary weighted projection...
plot(igraph::graph_from_adjacency_matrix(P, mode = "undirected",
     weighted = TRUE, diag = FALSE)) # ...is a dense hairball

bb <- fdsm(B, alpha = 0.05, trials = 1000, narrative = TRUE, class = "igraph") # An FDSM backbone...
plot(bb) # ...is sparse with clear communities

---

fixedcol

Extract backbone using the Fixed Column Model

Description

fixedcol extracts the backbone of a bipartite projection using the Fixed Column Model.

Usage

fixedcol(
  B,
  method = "RefinedNormal",
  alpha = 0.05,
  signed = FALSE,
  mtc = "none",
  class = "original",
  narrative = FALSE
)

Arguments

B

An unweighted bipartite graph, as: (1) an incidence matrix in the form of a matrix or sparse Matrix; (2) an edgelist in the form of a two-column dataframe; (3) an igraph object; (4) a network object. Any rows and columns of the associated bipartite matrix that contain only zeros are automatically removed before computations.

method

string: Specifies the method of the Poisson Binomial distribution computation used by the "ppbinom" function in PoissonBinomial-Distribution. "RefinedNormal" gives quick, very accurate approximations, while "DivideFFT" gives the quickest exact computations.
alpha real: significance level of hypothesis test(s)
signed boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)
mtc string: type of Multiple Test Correction to be applied; can be any method allowed by `p.adjust`.
class string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as B.
narrative boolean: TRUE if suggested text & citations should be displayed.

Details

This `fixedcol` function compares an edge's observed weight in the projection $B \ast t(B)$ to the distribution of weights expected in a projection obtained from a random bipartite graph where the column vertex degrees are fixed but the row vertex degrees are allowed to vary.

When signed = FALSE, a one-tailed test (is the weight stronger) is performed for each edge with a non-zero weight. It yields a backbone that preserves edges whose weights are significantly stronger than expected under the null model. When signed = TRUE, a two-tailed test (is the weight stronger or weaker) is performed for each every pair of nodes. It yields a backbone that contains positive edges for edges whose weights are significantly stronger, and negative edges for edges whose weights are significantly weaker, than expected in the chosen null model. NOTE: Before v2.0.0, all significance tests were two-tailed and zero-weight edges were evaluated.

Value

If alpha != NULL: Binary or signed backbone graph of class class.

If alpha == NULL: An S3 backbone object containing three matrices (the weighted graph, edges’ upper-tail p-values, edges’ lower-tail p-values), and a string indicating the null model used to compute p-values, from which a backbone can subsequently be extracted using `backbone.extract()`. The signed, mtc, class, and narrative parameters are ignored.

References


Examples

```r
# A binary bipartite network of 30 agents & 75 artifacts; agents form three communities
B <- rbind(cbind(matrix(rbinom(250,1,.8),10),
             matrix(rbinom(250,1,.2),10),
             matrix(rbinom(250,1,.2),10)),
          cbind(matrix(rbinom(250,1,.2),10),
                matrix(rbinom(250,1,.8),10),
                matrix(rbinom(250,1,.2),10)),
          cbind(matrix(rbinom(250,1,.2),10),
                matrix(rbinom(250,1,.2),10),
                matrix(rbinom(250,1,.2),10),
                matrix(rbinom(250,1,.2),10))
```
matrix(rbinom(250,1,.8),10))

P <- B%*%t(B)  # An ordinary weighted projection...
plot(igraph::graph_from_adjacency_matrix(P, mode = "undirected",
    weighted = TRUE, diag = FALSE))  # ...is a dense hairball

bb <- fixedcol(B, alpha = 0.05, narrative = TRUE, class = "igraph")  # A fixedcol backbone...
plot(bb)  # ...is sparse with clear communities

---

**fixedfill**

"Extract backbone using the Fixed Fill Model"

**Description**

`fixedfill` extracts the backbone of a bipartite projection using the Fixed Fill Model.

**Usage**

```r
fixedfill(
  B,
  alpha = 0.05,
  signed = FALSE,
  mtc = "none",
  class = "original",
  narrative = FALSE
)
```

**Arguments**

- **B**  
  An unweighted bipartite graph, as: (1) an incidence matrix in the form of a matrix or sparse `Matrix`; (2) an edgelist in the form of a two-column dataframe; (3) an `igraph` object; (4) a `network` object. Any rows and columns of the associated bipartite matrix that contain only zeros or only ones are automatically removed before computations.

- **alpha**  
  real: significance level of hypothesis test(s)

- **signed**  
  boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)

- **mtc**  
  string: type of Multiple Test Correction to be applied; can be any method allowed by `p.adjust`

- **class**  
  string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as `B`.

- **narrative**  
  boolean: TRUE if suggested text & citations should be displayed.
fixedfill

Details

The fixedfill function compares an edge's observed weight in the projection $B \ast t(B)$ to the distribution of weights expected in a projection obtained from a random bipartite graph where the number of edges present (i.e., the number of cells filled with a 1) is equal to the number of edges in $B$. When $B$ is large, this function may be impractically slow and may return a backbone object that contains NaN values.

When signed = FALSE, a one-tailed test (is the weight stronger) is performed for each edge with a non-zero weight. It yields a backbone that preserves edges whose weights are significantly stronger than expected under the null model. When signed = TRUE, a two-tailed test (is the weight stronger or weaker) is performed for each every pair of nodes. It yields a backbone that contains positive edges for edges whose weights are significantly stronger, and negative edges for edges whose weights are significantly weaker, than expected in the chosen null model. NOTE: Before v2.0.0, all significance tests were two-tailed and zero-weight edges were evaluated.

Value

If alpha != NULL: Binary or signed backbone graph of class class.

If alpha == NULL: An S3 backbone object containing three matrices (the weighted graph, edges’ upper-tail p-values, edges’ lower-tail p-values), and a string indicating the null model used to compute p-values, from which a backbone can subsequently be extracted using backbone.extract(). The signed, mtc, class, and narrative parameters are ignored.

References


Examples

# A binary bipartite network of 30 agents & 75 artifacts; agents form three communities
B <- rbind(cbind(matrix(rbinom(250,1,.8),10),
            matrix(rbinom(250,1,.2),10),
            matrix(rbinom(250,1,.2),10)),
            cbind(matrix(rbinom(250,1,.2),10),
            matrix(rbinom(250,1,.8),10),
            matrix(rbinom(250,1,.2),10)),
            cbind(matrix(rbinom(250,1,.2),10),
            matrix(rbinom(250,1,.2),10),
            matrix(rbinom(250,1,.8),10))

P <- B%*%t(B) # An ordinary weighted projection...
plot(igraph::graph_from_adjacency_matrix(P, mode = "undirected",
                           weighted = TRUE, diag = FALSE)) #...is a dense hairball

bb <- fixedfill(B, alpha = 0.05, narrative = TRUE, class = "igraph") # A fixedfill backbone...
plot(bb) #...is sparse with clear communities
**fixedrow**

*Extract backbone using the Fixed Row Model*

**Description**

`fixedrow` extracts the backbone of a bipartite projection using the Fixed Row Model.

**Usage**

```r
fixedrow(
  B,
  alpha = 0.05,
  signed = FALSE,
  mtc = "none",
  class = "original",
  narrative = FALSE
)
```

**Arguments**

- **B**: An unweighted bipartite graph, as: (1) an incidence matrix in the form of a matrix or sparse `Matrix`; (2) an edgelist in the form of a two-column dataframe; (3) an `igraph` object; (4) a `network` object. Any rows and columns of the associated bipartite matrix that contain only zeros are automatically removed before computations.
- **alpha**: real: significance level of hypothesis test(s)
- **signed**: boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)
- **mtc**: string: type of Multiple Test Correction to be applied; can be any method allowed by `p.adjust`.
- **class**: string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as `B`.
- **narrative**: boolean: TRUE if suggested text & citations should be displayed.

**Details**

The `fixedrow` function compares an edge's observed weight in the projection $B \ast t(B)$ to the distribution of weights expected in a projection obtained from a random bipartite graph where the row vertex degrees are fixed but the column vertex degrees are allowed to vary. When `signed = FALSE`, a one-tailed test (is the weight stronger) is performed for each edge with a non-zero weight. It yields a backbone that preserves edges whose weights are significantly *stronger* than expected under the null model. When `signed = TRUE`, a two-tailed test (is the weight stronger or weaker) is performed for each every pair of nodes. It yields a backbone that contains positive edges for edges whose weights are significantly *stronger*, and negative edges for edges whose weights are significantly *weaker*, than expected in the chosen null model. *NOTE: Before v2.0.0, all significance tests were two-tailed and zero-weight edges were evaluated.*
**Value**

If `alpha != NULL`: Binary or signed backbone graph of class `class`.

If `alpha == NULL`: An S3 backbone object containing three matrices (the weighted graph, edges' upper-tail p-values, edges' lower-tail p-values), and a string indicating the null model used to compute p-values, from which a backbone can subsequently be extracted using `backbone.extract()`. The signed, `mtc`, `class`, and `narrative` parameters are ignored.

**References**


**Examples**

```r
#A binary bipartite network of 30 agents & 75 artifacts; agents form three communities
B <- rbind(cbind(matrix(rbinom(250,1,.8),10),
            matrix(rbinom(250,1,.2),10),
            matrix(rbinom(250,1,.2),10)),
          cbind(matrix(rbinom(250,1,.2),10),
            matrix(rbinom(250,1,.8),10),
            matrix(rbinom(250,1,.2),10)),
          cbind(matrix(rbinom(250,1,.2),10),
            matrix(rbinom(250,1,.2),10),
            matrix(rbinom(250,1,.8),10)))

P <- B%*%t(B) #An ordinary weighted projection...
plot(igraph::graph_from_adjacency_matrix(P, mode = "undirected",
                                           weighted = TRUE, diag = FALSE)) #...is a dense hairball

bb <- fixedrow(B, alpha = 0.05, narrative = TRUE, class = "igraph") #A fixedrow backbone...
plot(bb) #...is sparse with clear communities
```

---

**global**

*Compute global threshold backbone*

**Description**

`global` extracts the backbone of a weighted network using a global threshold

**Usage**

```r
global(
  W,
  upper = 0,
  lower = NULL,
  keepzeros = TRUE,
  class = "original",
)```
Arguments

- `W`: A weighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse `Matrix`, or dataframe; (2) an edgelist in the form of a three-column dataframe; (3) an `igraph` object; (4) a `network` object.
- `upper`: real, FUN, or NULL: upper threshold value or function that evaluates to an upper threshold value.
- `lower`: real, FUN, or NULL: lower threshold value or function that evaluates to a lower threshold value.
- `keepzeros`: boolean: TRUE if zero-weight edges in `W` should be excluded from (i.e. also be zero in) the backbone.
- `class`: string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as `W`.
- `narrative`: boolean: TRUE if suggested text & citations should be displayed.

Details

The `global` function retains an edge in the backbone if its weight exceeds `upper`. If a `lower` threshold is also specified, it returns a signed backbone in which edge weights are set to 1 if above the given upper threshold, set to -1 if below the given lower threshold, and set to 0 otherwise. Prior to v2.0.0, the `global` function was called `universal()`, which is now deprecated.

If `W` is an unweighted bipartite graph, any rows and columns that contain only zeros or only ones are removed, then the global threshold is applied to its weighted bipartite projection.

Value

Binary or signed backbone graph of class given in parameter `class`.

References


Examples

```r
W <- matrix(sample(0:5, 100, replace = TRUE), 10) #Random weighted graph
diag(W) <- 0
W
global(W, narrative = TRUE) #Keep all non-zero edges
global(W, upper = 4, lower = 2, narrative = TRUE) #Signed with specified thresholds
global(W, upper = function(x)mean(x), lower = function(x)mean(x), narrative = TRUE) #Above-average --> positive edges
```
**hyperg**

*Wrapper for fixedrow()*

**Description**

Wrapper for fixedrow()

**Usage**

```r
hyperg(
  B,
  alpha = 0.05,
  signed = FALSE,
  mtc = "none",
  class = "original",
  narrative = FALSE
)
```

**Arguments**

- **B** An unweighted bipartite graph, as: (1) an incidence matrix in the form of a matrix or sparse `Matrix`; (2) an edgelist in the form of a two-column dataframe; (3) an `igraph` object; (4) a `network` object. Any rows and columns of the associated bipartite matrix that contain only zeros are automatically removed before computations.
- **alpha** Real: significance level of hypothesis test(s)
- **signed** Boolean: TRUE if signed backbone is to be returned, FALSE if binary backbone is to be returned
- **mtc** string: type of Multiple Test Correction to be applied; can be any method allowed by `p.adjust`.
- **class** string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as `B`.
- **narrative** Boolean: TRUE if suggested text for a manuscript is to be returned.

**osdsm**

*Extract backbone using the Ordinal Stochastic Degree Sequence Model*

**Description**

`osdsm` extracts the backbone of a bipartite projection using the Ordinal Stochastic Degree Sequence Model.
Usage

```r
osdsm(
  B,
  alpha = 0.05,
  trials = NULL,
  signed = FALSE,
  mtc = "none",
  class = "original",
  narrative = FALSE
)
```

Arguments

- **B**
  An ordinally weighted bipartite graph, as: (1) an incidence matrix in the form of a matrix or sparse `Matrix`; (2) an edgelist in the form of a three-column dataframe; (3) an `igraph` object; (4) a `network` object. Any rows and columns of the associated bipartite matrix that contain only zeros or only ones are automatically removed before computations.

- **alpha**
  real: significance level of hypothesis test(s)

- **trials**
  integer: the number of bipartite graphs generated to approximate the edge weight distribution. If NULL, the number of trials is selected based on `alpha` (see details)

- **signed**
  boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)

- **mtc**
  string: type of Multiple Test Correction to be applied; can be any method allowed by `p.adjust`.

- **class**
  string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as `B`.

- **narrative**
  boolean: TRUE if suggested text & citations should be displayed.

Details

The `osdsm` function compares an edge’s observed weight in the projection $B^t B$ to the distribution of weights expected in a projection obtained from a random bipartite network where both the rows and the columns contain approximately the same number of each value. The edges in $B$ must be integers, and are assumed to represent an ordinal-level measure such as a Likert scale.

When signed = FALSE, a one-tailed test (is the weight stronger) is performed for each edge with a non-zero weight. It yields a backbone that preserves edges whose weights are significantly stronger than expected in the chosen null model. When signed = TRUE, a two-tailed test (is the weight stronger or weaker) is performed for each pair of nodes. It yields a backbone that contains positive edges for edges whose weights are significantly stronger, and negative edges for edges whose weights are significantly weaker, than expected in the chosen null model. NOTE: Before v2.0.0, all significance tests were two-tailed and zero-weight edges were evaluated.

The p-values used to evaluate the statistical significance of each edge are computed using Monte Carlo methods. The number of trials performed affects the precision of these p-values, and the
confidence that a given p-value is less than the desired alpha level. Because these p-values are proportions (i.e., the proportion of times an edge is weaker/stronger in the projection of a random bipartite graph), evaluating the statistical significance of an edge is equivalent to comparing a proportion (the p-value) to a known proportion (alpha). When trials = NULL, the power.prop.test function is used to estimate the required number of trials to make such a comparison with a \( \alpha \) type-I error rate, (1-\( \alpha \)) power, and when the riskiest p-value being evaluated is at least 5% smaller than \( \alpha \). When any mtc correction is applied, for simplicity this estimation is based on a conservative Bonferroni correction.

Value

If \( \alpha \) != NULL: Binary or signed backbone graph of class class.

If \( \alpha \) == NULL: An S3 backbone object containing three matrices (the weighted graph, edges' upper-tail p-values, edges' lower-tail p-values), and a string indicating the null model used to compute p-values, from which a backbone can subsequently be extracted using backbone.extract(). The signed, mtc, class, and narrative parameters are ignored.

References


Examples

```
#A weighted binary bipartite network of 20 agents & 50 artifacts; agents form two communities
B <- rbind(cbind(matrix(sample(0:3, 250, replace = TRUE, prob = ((1:4)^2)),10),
            matrix(sample(0:3, 250, replace = TRUE, prob = ((4:1)^2)),10)),
            cbind(matrix(sample(0:3, 250, replace = TRUE, prob = ((4:1)^2)),10),
                  matrix(sample(0:3, 250, replace = TRUE, prob = ((1:4)^2)),10)))

P <- B%*%t(B) #An ordinary weighted projection...
plot(igraph::graph_from_adjacency_matrix(P, mode = "undirected",
                                           weighted = TRUE, diag = FALSE)) #...is a dense hairball

bb <- osdsm(B, alpha = 0.05, narrative = TRUE, #An oSDSM backbone...
               class = "igraph", trials = 1000)
plot(bb) #...is sparse with clear communities
```

sdsm

Extract backbone using the Stochastic Degree Sequence Model

Description

sdsm extracts the backbone of a bipartite projection using the Stochastic Degree Sequence Model.
Usage

```
sdsm(
    B,
    method = "RefinedNormal",
    alpha = 0.05,
    signed = FALSE,
    mtc = "none",
    class = "original",
    narrative = FALSE,
    ...
)
```

Arguments

- **B**: An unweighted bipartite graph, as: (1) an incidence matrix in the form of a matrix or sparse `Matrix`; (2) an edgelist in the form of a two-column dataframe; (3) an `igraph` object; (4) a `network` object. Any rows and columns of the associated bipartite matrix that contain only zeros are automatically removed before computations.

- **method**: string: Specifies the method of the Poisson Binomial distribution computation used by the "ppbinom" function in `PoissonBinomial-Distribution`. "RefinedNormal" gives quick, very accurate approximations, while "DivideFFT" gives the quickest exact computations.

- **alpha**: real: significance level of hypothesis test(s)

- **signed**: boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)

- **mtc**: string: type of Multiple Test Correction to be applied; can be any method allowed by `p.adjust`.

- **class**: string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as B.

- **narrative**: boolean: TRUE if suggested text & citations should be displayed.

... optional arguments

Details

The `sdsm` function compares an edge’s observed weight in the projection $B^tB$ to the distribution of weights expected in a projection obtained from a random bipartite network where both the row vertex degrees and column vertex degrees are approximately fixed at their values in B. It uses the Bipartite Configuration Model `bicm` to compute probabilities for the Poisson binomial distribution. When `signed = FALSE`, a one-tailed test (is the weight stronger) is performed for each edge with a non-zero weight. It yields a backbone that preserves edges whose weights are significantly stronger than expected in the chosen null model. When `signed = TRUE`, a two-tailed test (is the weight stronger or weaker) is performed for each pair of nodes. It yields a backbone that contains positive edges for edges whose weights are significantly stronger, and negative edges for edges whose weights are significantly weaker, than expected in the chosen null model. **NOTE: Before v2.0.0, all significance tests were two-tailed and zero-weight edges were evaluated.**
sparsify

**Value**

If alpha != NULL: Binary or signed backbone graph of class class.

If alpha == NULL: An S3 backbone object containing three matrices (the weighted graph, edges’ upper-tail p-values, edges’ lower-tail p-values), and a string indicating the null model used to compute p-values, from which a backbone can subsequently be extracted using backbone.extract(). The signed, mtc, class, and narrative parameters are ignored.

**References**


**Examples**

```r
# A binary bipartite network of 30 agents & 75 artifacts; agents form three communities
B <- rbind(
cbind(matrix(rbinom(250,1,.8),10),
  matrix(rbinom(250,1,.2),10),
  matrix(rbinom(250,1,.2),10)),
cbind(matrix(rbinom(250,1,.2),10),
  matrix(rbinom(250,1,.8),10),
  matrix(rbinom(250,1,.2),10)),
cbind(matrix(rbinom(250,1,.2),10),
  matrix(rbinom(250,1,.2),10),
  matrix(rbinom(250,1,.8),10)))

P <- B%*%t(B) # An ordinary weighted projection...
plot(igraph::graph_from_adjacency_matrix(P, mode = "undirected",
    weighted = TRUE, diag = FALSE)) #...is a dense hairball

bb <- sdsm(B, alpha = 0.05, narrative = TRUE, class = "igraph") # An SDSM backbone...
plot(bb) #...is sparse with clear communities
```

---

**sparsify**

*Extract the backbone from a network using a sparsification model*

**Description**

A generic function to extract the backbone of an undirected, unipartite network using a sparsification model described by a combination of an edge scoring metric, a edge score normalization, and an edge score filter.
Usage

```r
sparsify(
  U,
  s,
  escore = "original",
  normalize,
  filter,
  umst = FALSE,
  class = "original",
  narrative = FALSE
)
```

Arguments

- **U**: An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse `Matrix`; (2) an edgelist in the form of a two-column dataframe; (3) an `igraph` object; (4) a `network` object.
- **s**: numeric: Sparsification parameter
- **escore**: string: Method for scoring edges’ importance
- **normalize**: string: Method for normalizing edge scores
- **filter**: string: Type of filter to apply
- **umst**: boolean: TRUE if the backbone should include the union of minimum spanning trees, to ensure connectivity
- **class**: string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as `U`.
- **narrative**: boolean: TRUE if suggested text & citations should be displayed.

Details

The `escore` parameter determines how an unweighted edge’s importance is calculated. Unless noted below, scores are symmetric and larger values represent more important edges. There are 10 options for assigning an edge’s score; when `escore =`

- **random**: a random number drawn from a uniform distribution
- **betweenness**: edge betweenness
- **triangles**: number of triangles that include the edge
- **jaccard**: jaccard coefficient of the neighborhoods of an edge’s endpoints, or alternatively, triangles normalized by the size of the union of the endpoints neighborhoods
- **quadrangles**: number of quadrangles that include the edge
- **quadrilateral embeddedness**: geometric mean normalization of quadrangles
- **degree**: degree of neighbor to which an edge is adjacent (asymmetric)
- **meetmin**: triangles normalized by the smaller of the endpoints’ neighborhoods’ sizes
- **geometric**: triangles normalized by the product of the endpoints’ neighborhoods’ sizes
hypergeometric: probability of the edge being included at least as many triangles if edges were random, given the size of the endpoints’ neighborhoods (smaller is more important)

The normalize parameter determines whether edge scores are normalized. There are three options; when normalize =

- none: no normalization is performed
- rank: scores are normalized by neighborhood rank, such that the strongest edge in a node’s neighborhood is ranked 1 (asymmetric)
- embeddedness: scores are normalized using the maximum Jaccard coefficient of the top k-ranked neighbors of each endpoint, for all k

The filter parameter determines how edges are filtered based on their (normalized) edge scores. There are three options; when filter =

- threshold: Edges with scores more important than s are retained in the backbone
- proportion: Specifies the proportion of most important edges to retain in the backbone
- degree: Retains each node’s d-s most important edges, where d is the node’s degree (requires that normalize = "rank")

Specific combinations of escore, normalize, filter, and umst correspond to specific sparsification models in the literature, and are available via the following wrapper functions: sparsify.with.skeleton(), sparsify.with.gspar(), sparsify.with.lspar(), sparsify.with.simmelian(), sparsify.with.jaccard(), sparsify.with.meetmin(), sparsify.with.geometric(), sparsify.with.hypergeometric(), sparsify.with.localdegree(), sparsify.with.quadrilateral(). See the documentation for these wrapper functions for more details and the associated citation.

Value
An unweighted, undirected, unipartite graph of class class.

Examples
U <- igraph::sbm.game(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify(U, s = 0.6, escore = "jaccard", normalize = "rank",
filter = "degree", narrative = TRUE)
plot(sparse) #Clearly visible communities

sparsify.with.geometric

 Extract Goldberg and Roth’s (2003) Geometric backbone

Description
sparsify.with.geometric is a wrapper for sparsify() that extracts the geometric backbone described by Goldberg and Roth (2003). It is equivalent to sparsify(escore = "geometric", normalize = "none", filter = "threshold", umst = FALSE).
Usage

```
sparsify.with.geometric(U, s, class = "original", narrative = FALSE)
```

Arguments

- **U**: An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse `Matrix`; (2) an edgelist in the form of a two-column dataframe; (3) an `igraph` object; (4) a `network` object.
- **s**: numeric: Sparsification threshold, 0 < s < 1; larger values yield sparser graphs
- **class**: string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as U.
- **narrative**: boolean: TRUE if suggested text & citations should be displayed.

Value

An unweighted, undirected, unipartite graph of class class.

References


Examples

```
U <- igraph::sbm.game(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.geometric(U, s = 0.25, narrative = TRUE)
plot(sparse) #Clearly visible communities
```

Description

`sparsify.with.gspar` is a wrapper for `sparsify()` that extracts the G-spar backbone described by Satuluri et al. (2011). It is equivalent to `sparsify(escore = "jaccard", normalize = "none", filter = "proportion", umst = FALSE)`.

Usage

```
sparsify.with.gspar(U, s, class = "original", narrative = FALSE)
```
Arguments

- **U**: An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse Matrix; (2) an edgelist in the form of a two-column dataframe; (3) an igraph object; (4) a network object.

- **s**: numeric: Proportion of edges to retain, $0 < s < 1$; smaller values yield sparser graphs

- **class**: string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as U.

- **narrative**: boolean: TRUE if suggested text & citations should be displayed.

Value

An unweighted, undirected, unipartite graph of class class.

References


Examples

```r
U <- igraph::sbm.game(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.gspar(U, s = 0.4, narrative = TRUE)
plot(sparse) #Clearly visible communities
```

sparsify.with.hypergeometric

Extract Goldberg and Roth's (2003) Hypergeometric backbone

Description

sparsify.with.hypergeometric is a wrapper for sparsify() that extracts the hypergeometric backbone described by Goldberg and Roth (2003). It is equivalent to sparsify(escore = "hypergeometric",normalize = "none",filter = "threshold",umst = FALSE).

Usage

sparsify.with.hypergeometric(U, s, class = "original", narrative = FALSE)
sparsify.with.jaccard

Arguments

U  An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse Matrix; (2) an edgelist in the form of a two-column dataframe; (3) an igraph object; (4) a network object.

s  numeric: Sparsification threshold, 0 < s < 1; smaller values yield sparser graphs

class  string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as U.

narrative  boolean: TRUE if suggested text & citations should be displayed.

Value

An unweighted, undirected, unipartite graph of class class.

References


Examples

U <- igraph::sbm.game(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.hypergeometric(U, s = 0.3, narrative = TRUE)
plot(sparse) #Clearly visible communities

sparsify.with.jaccard  Extract Goldberg and Roth’s (2003) Jaccard backbone

Description

sparsify.with.jaccard is a wrapper for sparsify() that extracts the jaccard backbone described by Goldberg and Roth (2003). It is equivalent to sparsify(escore = "jaccard",normalize = "none",filter = "threshold",umst = FALSE).

Usage

sparsify.with.jaccard(U, s, class = "original", narrative = FALSE)

Arguments

U  An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse Matrix; (2) an edgelist in the form of a two-column dataframe; (3) an igraph object; (4) a network object.

s  numeric: Sparsification threshold, 0 < s < 1; larger values yield sparser graphs
sparsify.with.localdegree

```
  class string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as U.
  narrative boolean: TRUE if suggested text & citations should be displayed.
```

Value

An unweighted, undirected, unipartite graph of class class.

References


Examples

```
  U <- igraph::sbm.game(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
  plot(U) #A hairball
  sparse <- sparsify.with.jaccard(U, s = 0.3, narrative = TRUE)
  plot(sparse) #Clearly visible communities
```

Description

sparsify.with.localdegree is a wrapper for sparsify() that extracts the local degree backbone described by Hamann et al. (2016). It is equivalent to sparsify(escore = "degree",normalize = "rank",filter = "degree",umst = FALSE).

Usage

```
  sparsify.with.localdegree(U, s, class = "original", narrative = FALSE)
```

Arguments

```
  U An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse Matrix; (2) an edgelist in the form of a two-column dataframe; (3) an igraph object; (4) a network object.
  s numeric: Sparsification exponent, 0 < s < 1; smaller values yield sparser graphs
  class string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as U.
  narrative boolean: TRUE if suggested text & citations should be displayed.
```
sparsify.with.lspar

Value
An unweighted, undirected, unipartite graph of class `class`.

References

Examples
```r
U <- igraph::as.undirected(igraph::sample_pa(60, m = 3), mode = "collapse")
plot(U) # A hairball
sparse <- sparsify.with.localdegree(U, s = 0.3, narrative = TRUE)
plot(sparse) # Clearly visible hubs
```

sparsify.with.lspar  *Extract Satuluri et al's (2011) L-spar backbone*

Description
`sparsify.with.lspar` is a wrapper for `sparsify()` that extracts the L-spar backbone described by Satuluri et al. (2011). It is equivalent to `sparsify(escore = "jaccard", normalize = "rank", filter = "degree", umst = FALSE)`.

Usage
`sparsify.with.lspar(U, s, class = "original", narrative = FALSE)`

Arguments
- `U` An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse `Matrix`; (2) an edgelist in the form of a two-column dataframe; (3) an `igraph` object; (4) a `network` object.
- `s` numeric: Sparsification exponent, 0 < s < 1; smaller values yield sparser graphs
- `class` string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as `U`.
- `narrative` boolean: TRUE if suggested text & citations should be displayed.

Value
An unweighted, undirected, unipartite graph of class `class`.
References

Examples
plot(U) # A hairball
sparse <- sparsify.with.lspar(U, s = 0.6, narrative = TRUE)
plot(sparse) # Clearly visible communities

Description
sparsify.with.meetmin is a wrapper for sparsify() that extracts the meetmin backbone described by Goldberg and Roth (2003). It is equivalent to sparsify(escore = "meetmin", normalize = "none", filter = "threshold", umst = FALSE).

Usage
sparsify.with.meetmin(U, s, class = "original", narrative = FALSE)

Arguments
U An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse Matrix; (2) an edgelist in the form of a two-column dataframe; (3) an igraph object; (4) a network object.
s numeric: Sparsification threshold, 0 < s < 1; larger values yield sparser graphs
class string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as U.
narrative boolean: TRUE if suggested text & citations should be displayed.

Value
An unweighted, undirected, unipartite graph of class class.

References
Examples

```r
U <- igraph::sbm.game(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.meetmin(U, s = 0.5, narrative = TRUE)
plot(sparse) #Clearly visible communities
```

---

sparsify.with.quadrilateral

Extract Nocaj et al.'s (2015) Quadrilateral Simmelian backbone

Description

sparsify.with.quadrilateral is a wrapper for sparsify() that extracts the quadrilateral Simmelian backbone described by Nocaj et al. (2015). It is equivalent to sparsify(escore = "quadrilateral embeddedness", normalize = "embeddedness", filter = "threshold", umst = TRUE).

Usage

```r
sparsify.with.quadrilateral(U, s, class = "original", narrative = FALSE)
```

Arguments

- **U**: An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse Matrix; (2) an edgelist in the form of a two-column dataframe; (3) an igraph object; (4) a network object.
- **s**: numeric: Sparsification exponent, 0 < s < 1; larger values yield sparser graphs
- **class**: string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as U.
- **narrative**: boolean: TRUE if suggested text & citations should be displayed.

Value

An unweighted, undirected, unipartite graph of class class.

References


Examples

```r
U <- igraph::sbm.game(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.quadrilateral(U, s = 0.5, narrative = TRUE)
plot(sparse) #Clearly visible communities in a connected graph
```
sparsify.with.simmelian

Extract Nick et al’s (2013) Simmelian backbone

Description

sparsify.with.simmelian is a wrapper for sparsify() that extracts the simmelian backbone described by Nick et al. (2013). It is equivalent to sparsify(escore = "triangles",normalize = "embeddedness",filter = "threshold",umst = FALSE).

Usage

sparsify.with.simmelian(U, s, class = "original", narrative = FALSE)

Arguments

U An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse Matrix; (2) an edgelist in the form of a two-column dataframe; (3) an igraph object; (4) a network object.

s numeric: Sparsification threshold, 0 < s < 1; larger values yield sparser graphs

class string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as U.

narrative boolean: TRUE if suggested text & citations should be displayed.

Value

An unweighted, undirected, unipartite graph of class class.

References


Examples

U <- igraph::sbm.game(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.simmelian(U, s = 0.5, narrative = TRUE)
plot(sparse) #Clearly visible communities
**Description**

`sparsify.with.skeleton` is a wrapper for `sparsify()` that extracts the skeleton backbone described by Karger (1999), which preserves a specified proportion of random edges. It is equivalent to `sparsify(escore = "random", normalize = "none", filter = "proportion", umst = FALSE)

**Usage**

```r
sparsify.with.skeleton(U, s, class = "original", narrative = FALSE)
```

**Arguments**

- `U` An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse `Matrix`; (2) an edgelist in the form of a two-column dataframe; (3) an `igraph` object; (4) a `network` object.
- `s` numeric: Proportion of edges to retain, 0 < s < 1; smaller values yield sparser graphs
- `class` string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as `U`.
- `narrative` boolean: TRUE if suggested text & citations should be displayed.

**Value**

An unweighted, undirected, unipartite graph of class `class`.

**References**


**Examples**

```r
U <- igraph::erdos.renyi.game(60, .5)
plot(U) #A dense graph
sparse <- sparsify.with.skeleton(U, s = 0.25, narrative = TRUE)
plot(sparse) #A sparser graph
```
universal  Wrapper for global()

Description

Wrapper for global()

Usage

universal(
  M,
  upper = 0,
  lower = NULL,
  keepzeros = TRUE,
  class = "original",
  narrative = FALSE
)

Arguments

M  graph: Graph object of class matrix, sparse matrix, igraph, edgelist, or network object.
upper  Real, FUN, or NULL: upper threshold value or function that evaluates to an upper threshold value.
lower  Real, FUN, or NULL: lower threshold value or function that evaluates to a lower threshold value.
keepzeros  Boolean: TRUE if zero-weight edges in M should be missing in the backbone
class  string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"). If "original", the backbone graph returned is of the same class as B.
narrative  Boolean: TRUE if suggested text for a manuscript is to be returned
Index

backbone, 2
backbone.extract, 3
backbone.extract(), 10, 13, 15, 17, 19, 23, 25
backbone.suggest, 5
backbone.suggest(), 3
bicm, 5, 24
bicm(), 3
bipartite.add.blocks, 6
bipartite.from.distribution, 7
bipartite.from.distribution(), 3
bipartite.from.probability, 8
bipartite.from.probability(), 3
bipartite.from.sequence, 9
bipartite.from.sequence(), 3
disparity, 9
disparity(), 3
fastball, 11
fastball(), 3, 13
fdsm, 12
fdsm(), 2
fixedcol, 14
fixedcol(), 2
fixedfill, 16
fixedfill(), 2
fixedrow, 18
fixedrow(), 2
global, 19
global(), 3
hyperg, 21
igraph, 5, 7, 10, 12, 14, 16, 18, 20–22, 24, 26, 28–36
loglikelihood_prime_bicm, 6
Matrix, 5, 10, 12, 14, 16, 18, 20–22, 24, 26, 28–36
network, 5, 7, 10, 12, 14, 16, 18, 20–22, 24, 26, 28–36
osdsm, 21
osdsm(), 2
p.adjust, 4, 10, 12, 15, 16, 18, 21, 22, 24
PoissonBinomial-Distribution, 14, 24
sdsm, 23
sdsm(), 2, 6
sparsify, 25
sparsify(), 3, 27–36
sparsify.with.geometric, 27
sparsify.with.geometric(), 3, 27
sparsify.with.gspar, 28
sparsify.with.gspar(), 3, 27
sparsify.with.hypergeometric, 29
sparsify.with.hypergeometric(), 3, 27
sparsify.with.jaccard, 30
sparsify.with.jaccard(), 3, 27
sparsify.with.localdegree, 31
sparsify.with.localdegree(), 3, 27
sparsify.with.lspar, 32
sparsify.with.lspar(), 3, 27
sparsify.with.meetmin, 33
sparsify.with.meetmin(), 3, 27
sparsify.with.quadrilateral, 34
sparsify.with.quadrilateral(), 3, 27
sparsify.with.simmelian, 35
sparsify.with.simmelian(), 3, 27
sparsify.with.skeleton, 36
sparsify.with.skeleton(), 3, 27
tomatrix, 4
universal, 37
universal(), 20