Package 'backbone'

September 17, 2021

Type Package

Title Extracts the Backbone from Weighted Graphs

Version 1.5.1

Description Provides methods for extracting from a weighted graph
a binary or signed backbone that retains only the significant edges.
The user may input a weighted graph, or a bipartite graph
from which a weighted graph is first constructed via projection.
Backbone extraction methods include the stochastic degree sequence model (SDSM; Neal, Z. P. (2014). <doi:10.1016/j.socnet.2014.06.001>),
the fixed degree sequence model (FDSM; Zweig, K. A., and Kaufmann, M. (2011). <doi:10.1007/s13278-011-0021-0>),
the fixed row model (FRM; Neal, Z. P. (2013). <doi:10.1007/s13278-013-0107-y>),
the fixed column model (FCM; Neal, Domagalski, and Sagan (2021). <arXiv:2105.13396>),
the fixed fill model (FFM; Neal, Domagalski, and Sagan (2021). <arXiv:2105.13396>),
and a universal threshold method.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 2.10)

Imports Matrix, methods, stats, utils, igraph, network,
PoissonBinomial

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

URL https://www.zacharyneal.com/backbone,
https://github.com/domagal9/backbone

BugReports https://github.com/domagal9/backbone/issues

NeedsCompilation no

Author Zachary Neal [aut, cre],
Rachel Domagalski [aut],
Bruce Sagan [aut]

Maintainer Zachary Neal <zpneal@msu.edu>

Repository CRAN

Date/Publication 2021-09-17 15:30:09 UTC

R topics documented:

---

backbone *backbone: Extracts the Backbone from Weighted Graphs*

Description

Provides methods for extracting from a weighted graph a binary or signed backbone that retains only the significant edges. The user may input a weighted graph, or a bipartite graph from which a weighted graph is first constructed via projection. Backbone extraction methods include:

- the stochastic degree sequence model (SDSM; Neal, Z. P. (2014). The backbone of bipartite projections: Inferring relationships from co-authorship, co-sponsorship, co-attendance, and other co-behaviors. Social Networks, 39, Elsevier: 84-97.doi: 10.1016/j.socnet.2014.06.001),
- the fixed degree sequence model (FDSM; Zweig, Katharina Anna, and Michael Kaufmann. 2011. "A Systematic Approach to the One-Mode Projection of Bipartite Graphs." Social Network Analysis and Mining 1 (3): 187–218.doi: 10.1007/s1327801100210),
- the fixed row model (FRM; Neal, Zachary. 2013. "Identifying Statistically Significant Edges in One-Mode Projections." Social Network Analysis and Mining 3 (4). Springer: 915–24.doi: 10.1007/s132780130107y),
- the fixed column model (FCM; Neal, Domagalski, and Sagan. 2021. "Comparing Models for Extracting the Backbone of Bipartite Projections." arXiv:2105.13396 cs.SI),

- the fixed fill model (FFM; Neal, Domagalski, and Sagan. 2021. "Comparing Models for Extracting the Backbone of Bipartite Projections." arXiv:2105.13396 cs.SI),
- and a universal threshold method.

Details

To decide whether an edge of a bipartite projection $B * t(B)$ is statistically significant, we compare the edge's observed weight to the distribution of weights expected in a projection obtained from a random bipartite graph under a null model. Given a null model that can generate a distribution of an edge's weights, we can then compute the probability that the observed weight is in the upper or lower tail of that distribution. The 'backbone' package provides several different null models to use, all of which define different constraints on the row degrees and column degrees of a random bipartite graph in their corresponding distribution.

- '`sdsm`': computes the probability of edge weights being above or below the observed edge weights in a bipartite projection using the stochastic degree sequence model. Here the expected row and column degrees of a random bipartite graph in the distribution match that of the original bipartite graph.
- '`fixedrow`': computes the probability of edge weights being above or below the observed edge weights in a bipartite projection using the hypergeometric model. Here the row degrees of a random bipartite graph in the distribution exactly match that of the original bipartite graph. The column degrees are allowed to vary.
- '`fixedcol`': computes the probability of edge weights being above or below the observed edge weights in a bipartite projection using the Poisson binomial model. Here the column degrees of a random bipartite graph in the distribution exactly match that of the original bipartite graph. The row degrees are allowed to vary.
- '`fdsm`': computes the proportion of edge weights above or below the observed edge weights in a bipartite projection using the fixed degree sequence model. Here the row and column degrees of a random bipartite graph in the distribution exactly match that of the original bipartite graph.
- '`fixedfill`':computes the probability of edge weights being above or below the observed edge weights in a bipartite projection where the number of edges in a random bipartite graph of the distribution exactly match the number of edges of the original bipartite graph.

Once one of these models is computed, use `backbone.extract` to return the backbone graph for a chosen alpha value.

In addition to these methods, one can also use `universal` to return a backbone graph in which edge weights are set to 1 if above the given upper parameter threshold, and set to -1 if below the given lower parameter threshold, and are 0 otherwise.

Additional functions that aid in the use of the above models are exported:

- '`bicm`': finds a matrix that maximizes the entropy function, used in `sdsm`.
- '`curveball`': generates a random 0/1 matrix with the same row and column sums as the input, used in `fdsm`.

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

References

Domagalski, R., Neal, Z. P., and Sagan, B. (2021). backbone: An R Package for Backbone Extraction of Weighted Graphs. PLoS ONE. doi: 10.1371/journal.pone.0244363

`backbone.extract` *Extracts the backbone of a weighted network using results from a null model*

Description

'backbone.extract' returns a binary or signed adjacency matrix containing the backbone that retains only the significant edges.

Usage

```
backbone.extract(
  backbone,
  signed = FALSE,
  alpha = 0.05,
  fwer = "none",
  class = "original",
  narrative = FALSE
)
```

Arguments

`backbone`	backbone: backbone S3 class object.
`signed`	Boolean: TRUE if signed backbone is to be returned, FALSE if binary backbone is to be returned
`alpha`	Real: significance level of hypothesis test(s)
`fwer`	string: type of familywise error rate correction to be applied; c("none","bonferroni","holm"). If "holm", Holm Bonferroni Family-wise Error Rate test is used, if "bonferroni", Bonferroni Family-wise Error Rate test should be used. By default, the given 'alpha' value is used for all tests with no correction for family-wise error rates.
`class`	string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"), converted via tomatrix. If "original", the backbone graph returned is of the same class as the data used to generate the backbone object.
`narrative`	Boolean: TRUE if suggested text for a manuscript is to be returned.

Details

The "backbone" S3 class object is composed of two matrices, a summary dataframe and (optionally, if generated by using fdsm) a 'dyad_values' vector. This object is returned by sdsm, fdsm, fixedrow, fixedcol, fixedfill.

The Holm Bonferroni correction was originally a port from python code written by Dr. Samin Aref. The authors thank Dr. Aref greatly for his contribution to this package!

Value

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

Examples

```
probs <- sdsm(davis)
bb <- backbone.extract(probs, alpha = .2, signed = TRUE, fwer = "none")
```

`bicm` *bicm: Bipartite Configuration Model.*

Description

bicm: Bipartite Configuration Model.

Usage

```
bicm(graph, tol = 1e-08, max_steps = 200, ...)
```

Arguments

`graph`	matrix, a bipartite adjacency matrix of a graph
`tol`	numeric, tolerance of algorithm
`max_steps`	numeric, number of times to run loglikelihood_prime_bicm algorithm
`...`	optional arguments

Details

The Bipartite Configuration Model (Saracco et. al. 2015, 2017) produces a matrix of edge specific probabilities which are used in sdsm to find the p-values of the edges in the bipartite projection. This R code is adapted from the python BiCM package by Matteo Bruno under the MIT license.

Value

matrix containing probabilities

References

python bicm: Matteo Bruno, matteo.bruno<at>imtlucca.it, https://github.com/mat701/BiCM

bicm: Saracco, F., Straka, M. J., Clemente, R. D., Gabrielli, A., Caldarelli, G., & Squartini, T. (2017). Inferring monopartite projections of bipartite networks: An entropy-based approach. New Journal of Physics, 19(5), 053022. doi: 10.1088/13672630/aa6b38

bicm: Saracco, F., Di Clemente, R., Gabrielli, A., & Squartini, T. (2015). Randomizing bipartite networks: The case of the World Trade Web. Scientific Reports, 5(1), 10595. doi: 10.1038/srep10595

Examples

```
bicm(davis)
```

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

Description

'bipartite.add.blocks' rewires 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

```
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 = 10, C = 10, 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 <- bipartite.from.distribution(R = 100, C = 100, P = 0.1,
rowdist = c(1,1), coldist = c(1,1))  #Uniform
B <- bipartite.from.distribution(R = 100, C = 100, P = 0.1,
rowdist = c(1,10), coldist = c(1,10))  #Right-tailed
B <- bipartite.from.distribution(R = 100, C = 100, P = 0.1,
rowdist = c(10,1), coldist = c(10,1))  #Left-tailed
B <- bipartite.from.distribution(R = 100, C = 100, P = 0.1,
rowdist = c(10,10), coldist = c(10,10))  #Normal
B <- bipartite.from.distribution(R = 100, C = 100, P = 0.1,
rowdist = c(10000,10000), coldist = c(10000,10000))  #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").

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

---

curveball *curveball algorithm*

Description

curveball algorithm

Usage

```
curveball(M)
```

Arguments

M	matrix

Value

rm, a matrix with same row sums and column sums as M, but randomized 0/1 entries.

References

Algorithm and R implementation: Strona, Giovanni, Domenico Nappo, Francesco Boccacci, Simone Fattorini, and Jesus San-Miguel-Ayanz. 2014. "A Fast and Unbiased Procedure to Randomize Ecological Binary Matrices with Fixed Row and Column Totals." Nature Communications 5 (June). Nature Publishing Group: 4114. doi: 10.1038/ncomms5114

Examples

```
curveball(davis)
```

davis *Davis Southern Women Data Set*

Description

A two mode matrix of 18 women and attendance of 14 social events.

Usage

```
data(davis)
```

Format

An object of class `matrix` (inherits from `array`) with 18 rows and 14 columns.

Source

UCI Network Data Repository

References

Davis, A., Gardner, B. B. and M. R. Gardner (1941) Deep South, Chicago: The University of Chicago Press.

fdsm *The fixed degree sequence model (fdsm) for backbone probabilities*

Description

'fdsm' computes the proportion of generated edges above or below the observed value using the fixed degree sequence model. Once computed, use `backbone.extract` to return the backbone matrix for a given alpha value.

Usage

```
fdsm(B, trials = 1000, dyad = NULL, progress = FALSE, ...)
```

Arguments

`B`	graph: An unweighted bipartite graph object of class matrix, sparse matrix, igraph, edgelist, or network object. Any rows and columns of the associated bipartite matrix that contain only zeros are automatically removed before computations.
`trials`	numeric: The number of bipartite graphs generated to approximate the edge weight distribution.

dyad	vector length 2: two row entries i,j. Saves each value of the i-th row and j-th column in each projected B* matrix. This is useful for visualizing an example of the empirical null edge weight distribution generated by the model. These correspond to the row and column indices of a cell in the projected matrix, and can be written as their string row names or as numeric values.
progress	Boolean: If txtProgressBar should be used to measure progress
...	optional arguments

Details

During each iteration, fdsm computes a new B* matrix using the curveball algorithm. This is a random bipartite matrix with the same row and column sums as the original matrix B. If a value is supplied for the dyad parameter, when the B* matrix is projected (multiplied by its transpose), the value in the corresponding row and column will be saved. This allows the user to see the distribution of the edge weights for desired row and column.

The "backbone" S3 class object returned is composed of two matrices, a summary dataframe and (if specified) a 'dyad_values' vector.

Value

backbone, a list(positive, negative, dyad_values, summary). Here 'positive' is a matrix of proportion of times each entry of the projected matrix B is above the corresponding entry in the generated projection, 'negative' is a matrix of proportion of times each entry of the projected matrix B is below the corresponding entry in the generated projection, 'dyad_values' is a list of edge weight for i,j in each generated projection, and 'summary' is a data frame summary of the inputted matrix and the model used including: class, model name, number of rows, number of columns, and running time.

References

fixed degree sequence model: Zweig, Katharina Anna, and Michael Kaufmann. 2011. "A Systematic Approach to the One-Mode Projection of Bipartite Graphs." Social Network Analysis and Mining 1 (3): 187–218. doi: 10.1007/s13278011002-10

curveball algorithm: Strona, Giovanni, Domenico Nappo, Francesco Boccacci, Simone Fattorini, and Jesus San-Miguel-Ayanz. 2014. "A Fast and Unbiased Procedure to Randomize Ecological Binary Matrices with Fixed Row and Column Totals." Nature Communications 5 (June). Nature Publishing Group: 4114. doi: 10.1038/ncomms5114

Examples

```
fdsm_props <- fdsm(davis, trials = 100, dyad=c(3,6))
```

`fixedcol` *Compute fixed column sums / Poisson binomial backbone probabilities*

Description

'fixedcol' computes the probability of observing a higher or lower edge weight using the Poisson binomial distribution. Once computed, use `backbone.extract` to return the backbone matrix for a given alpha value.

Usage

```
fixedcol(B, method = "RefinedNormal")
```

Arguments

`B`	graph: An unweighted bipartite graph object of class matrix, sparse matrix, igraph, edgelist, or 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.

Details

This fixedcol 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 graph where the column vertex degrees are fixed but the row vertex degrees are allowed to vary.

Value

backbone, a list(positive, negative, summary). Here 'positive' is a matrix of probabilities of edge weights being equal to or above the observed value in the projection, 'negative' is a matrix of probabilities of edge weights being equal to or below the observed value in the projection, and 'summary' is a data frame summary of the inputted matrix and the model used including: class, model name, number of rows, number of columns, and running time.

References

Neal, Domagalski, and Sagan. 2021. "Comparing Models for Extracting the Backbone of Bipartite Projections." arXiv:2105.13396 cs.SI)

Examples

```
fixedcol(davis)
```

`fixedfill` *Compute fixed fill backbone probabilities*

Description

'fixedfill' computes the probability of observing a higher or lower edge weight. Once computed, use `backbone.extract` to return the backbone matrix for a given alpha value.

Usage

```
fixedfill(B)
```

Arguments

`B`	graph: An unweighted bipartite graph object of class matrix, sparse matrix, igraph, edgelist, or network object. Any rows and columns of the associated bipartite matrix that contain only zeros are automatically removed before computations.

Details

The fixedfill 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 graph where the number of edges present 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.

Value

backbone, a list(positive, negative, summary). Here 'positive' is a matrix of probabilities of edge weights being equal to or above the observed value in the projection, 'negative' is a matrix of probabilities of edge weights being equal to or below the observed value in the projection, and 'summary' is a data frame summary of the inputted matrix and the model used including: class, model name, number of rows, number of columns, and running time.

References

Neal, Domagalski, and Sagan. 2021. "Comparing Models for Extracting the Backbone of Bipartite Projections." arXiv:2105.13396 cs.SI)

Examples

```
fixed_probs <- fixedfill(davis)
```

`fixedrow` *Compute fixed row sums / hypergeometric backbone probabilities*

Description

'fixedrow' computes the probability of observing a higher or lower edge weight using the hypergeometric distribution. Once computed, use `backbone.extract` to return the backbone matrix for a given alpha value.

Usage

```
fixedrow(B)
```

Arguments

`B`	graph: An unweighted bipartite graph object of class matrix, sparse matrix, igraph, edgelist, or network object. Any rows and columns of the associated bipartite matrix that contain only zeros are automatically removed before computations.

Details

The fixedrow 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 graph where the row vertex degrees are fixed but the column vertex degrees are allowed to vary.

Value

backbone, a list(positive, negative, summary). Here 'positive' is a matrix of probabilities of edge weights being equal to or above the observed value in the projection, 'negative' is a matrix of probabilities of edge weights being equal to or below the observed value in the projection, and 'summary' is a data frame summary of the inputted matrix and the model used including: class, model name, number of rows, number of columns, and running time.

References

Tumminello, Michele and Miccichè, Salvatore and Lillo, Fabrizio and Piilo, Jyrki and Mantegna, Rosario N. 2011. "Statistically Validated Networks in Bipartite Complex Systems." PLOS ONE, 6(3), doi: 10.1371/journal.pone.0017994

Neal, Zachary. 2013. "Identifying Statistically Significant Edges in One-Mode Projections." Social Network Analysis and Mining 3 (4). Springer: 915–24. doi: 10.1007/s132780130107y

Examples

```
fixedrow_probs <- fixedrow(davis)
```

hyperg *A wrapper for the fixedrow function*

Description

'hyperg' computes the probability of observing a higher or lower edge weight using the hypergeometric distribution. Once computed, use `backbone.extract` to return the backbone matrix for a given alpha value.

Usage

```
hyperg(B)
```

Arguments

B	graph: An unweighted bipartite graph object of class matrix, sparse matrix, igraph, edgelist, or network object. Any rows and columns of the associated bipartite matrix that contain only zeros are automatically removed before computations.

Details

Specifically, this 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 graph where the row vertex degrees are fixed but the column vertex degrees are allowed to vary.

Value

fixedrow

References

Tumminello, Michele and Miccichè, Salvatore and Lillo, Fabrizio and Piilo, Jyrki and Mantegna, Rosario N. 2011. "Statistically Validated Networks in Bipartite Complex Systems." PLOS ONE, 6(3), doi: 10.1371/journal.pone.0017994

Neal, Zachary. 2013. "Identifying Statistically Significant Edges in One-Mode Projections." Social Network Analysis and Mining 3 (4). Springer: 915–24. doi: 10.1007/s132780130107y

Examples

```
hyperg_probs <- hyperg(davis)
```

`sdsm` *The stochastic degree sequence model (sdsm) for backbone probabilities*

Description

'sdsm' computes the probability of edge weights being above or below the observed edge weights in a bipartite projection using the stochastic degree sequence model. Once computed, use `backbone.extract` to return the backbone matrix for a given alpha value.

Usage

```
sdsm(B, method = "RefinedNormal", ...)
```

Arguments

`B`	graph: An unweighted bipartite graph object of class matrix, sparse matrix, igraph, edgelist, or 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.
`...`	optional arguments

Details

The sdsm 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 approximately fixed. It uses the Bipartite Configuration Model bicm (Saracco et. al (2015, 2017)) to compute probabilities for the Poisson binomial distribution.

The "backbone" S3 class object returned is composed of two matrices, and a summary dataframe.

Value

backbone, a list(positive, negative, summary). Here 'positive' is a matrix of probabilities of edge weights being equal to or above the observed value in the projection, 'negative' is a matrix of probabilities of edge weights being equal to or below the observed value in the projection, and 'summary' is a data frame summary of the inputted matrix and the model used including: class, model name, number of rows, number of columns, and running time.

References

sdsm: Neal, Z. P. (2014). The backbone of bipartite projections: Inferring relationships from co-authorship, co-sponsorship, co-attendance, and other co-behaviors. Social Networks, 39, Elsevier: 84-97. doi: 10.1016/j.socnet.2014.06.001

bicm: Saracco, F., Straka, M. J., Clemente, R. D., Gabrielli, A., Caldarelli, G., & Squartini, T. (2017). Inferring monopartite projections of bipartite networks: An entropy-based approach. New Journal of Physics, 19(5), 053022. doi: 10.1088/13672630/aa6b38

bicm: Saracco, F., Di Clemente, R., Gabrielli, A., & Squartini, T. (2015). Randomizing bipartite networks: The case of the World Trade Web. Scientific Reports, 5(1), 10595. doi: 10.1038/srep10595

Examples

```
sdsm_probs <- sdsm(davis)
```

`universal` *Compute universal threshold backbone*

Description

'universal' returns a backbone graph in which edge weights are set to 1 if above the given upper parameter threshold, set to -1 if below the given lower parameter threshold, and are 0 otherwise.

Usage

```
universal(M, upper = NULL, lower = NULL, bipartite = NULL, narrative = FALSE)
```

Arguments

`M`	graph: Graph object of class matrix, sparse matrix, igraph, edgelist, or network object. Any rows and columns of the associated bipartite matrix that contain only zeros are automatically removed before computations.
`upper`	Real, FUN, or NULL: upper threshold value or function to be applied to the edge weights. Default is NULL.
`lower`	Real, FUN, or NULL: lower threshold value or function to be applied to the edge weights. Default is NULL.
`bipartite`	Boolean: TRUE if bipartite graph, FALSE if weighted graph. Default is NULL. If TRUE, input graph should be unweighted.
`narrative`	Boolean: TRUE if suggested text for a manuscript is to be returned

Details

If both 'upper' and 'lower' are 'NULL', a weighted projection is returned.

If 'bipartite' is 'NULL', the function tries to guess at whether the data is bipartite or unipartite based on its shape.

Value

backbone, a list(backbone, summary). The ‘backbone‘ object is a graph object of the same class as M. The ‘summary‘ contains a data frame summary of the inputted matrix and the model used including: model name, number of rows, number of columns, and running time.

Examples

```
test <- universal(davis%*%t(davis), upper = function(x)mean(x)+sd(x), lower=function(x)mean(x))
test2 <- universal(davis, upper = function(x)mean(x)+2*sd(x), lower = 2, bipartite = TRUE)
test3 <- universal(davis, upper = 4, lower = 2, bipartite = TRUE)
```

Index