

Package ‘backbone’

May 20, 2026

Type Package

Title Extracts the Backbone from Networks

Version 3.0.4

Description

An implementation of methods for extracting a sparse unweighted network (i.e. a backbone) from an unweighted network (e.g., Hamann et al., 2016 <[doi:10.1007/s13278-016-0332-2](https://doi.org/10.1007/s13278-016-0332-2)>), a weighted network (e.g., Serrano et al., 2009 <[doi:10.1073/pnas.0808904106](https://doi.org/10.1073/pnas.0808904106)>), or a weighted projection (e.g., Neal et al., 2021 <[doi:10.1038/s41598-021-03238-3](https://doi.org/10.1038/s41598-021-03238-3)>).

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.5)

Imports graphics, igraph, Matrix, methods, stats, Rcpp, utils,

Suggests knitr, rmarkdown, tinytest

LinkingTo Rcpp

VignetteBuilder knitr

URL <https://rbackbone.net>

BugReports <https://github.com/zpneal/backbone/issues>

LazyData true

NeedsCompilation yes

Author Zachary Neal [aut, cre] (ORCID: <<https://orcid.org/0000-0003-3076-4995>>),
Rachel Domagalski [ctb],
Bruce Sagan [ctb],
Karl Godard [ctb]

Maintainer Zachary Neal <zpneal@msu.edu>

Repository CRAN

Date/Publication 2026-05-20 19:30:02 UTC

Contents

backbone	2
backbone_from_projection	3
backbone_from_unweighted	5
backbone_from_weighted	8
bicm	10
fastball	12
senate108	12

Index	14
--------------	-----------

backbone	<i>Extract the backbone from a network</i>
----------	--------------------------------------------

Description

backbone() extracts the backbone from a weighted or unweighted network

Usage

```
backbone(N, narrative = TRUE, ...)
```

Arguments

N	A network represented as a matrix, Matrix , or igraph object
narrative	logical: display suggested text & citations
...	Optional arguments

Details

Given a weighted and/or dense network, the backbone is an sparse and unweighted subgraph that contains only the most "important" edges.

backbone() is a wrapper that detects the type of network in N, then extracts the backbone using the appropriate backbone_from_*() function:

- If N is a weighted network, [backbone_from_weighted\(\)](#)
- If N is a bipartite network or hypergraph, [backbone_from_projection\(\)](#)
- If N is an unweighted network, [backbone_from_unweighted\(\)](#)

Designed as a user-friendly wrapper, backbone() uses the defaults for the underlying functions, which should work in many cases, and which can be modified by passing optional arguments. Unlike the underlying functions, by default backbone() displays narrative text describing what it did so that you know.

For details about the backbone models, see the documentation for the underlying functions above. For an overview of the package with examples, please see the [Introduction to Backbone](#) using vignette("backbone"). For a detailed empirical example, please see the [U.S. Senate Example](#) using vignette("senate108").

Value

A backbone in the same class as N

References

package: Neal, Z. P. (2026). backbone: An R Package to Extract Network Backbones. PLOS One, 21, e0349258. doi:10.1371/journal.pone.0349258

Examples

```
N <- igraph::sample_gnp(100, .3) #A random unweighted network
backbone(N)
```

backbone_from_projection

Extract the backbone from a weighted bipartite or hypergraph projection

Description

backbone_from_projection() extracts the unweighted backbone from the weighted projection of a bipartite network or hypergraph

Usage

```
backbone_from_projection(
  B,
  alpha = 0.05,
  model = "sds",
  signed = FALSE,
  mtc = "none",
  missing_as_zero = FALSE,
  narrative = FALSE,
  trials = NULL,
  backbone_only = TRUE
)
```

Arguments

B	An unweighted bipartite network or hypergraph as an incidence matrix or Matrix , or as a bipartite igraph object
alpha	real: significance level of hypothesis test(s)
model	string: backbone model, one of: "sds", "fds", "fixedrow", "fixedcol", or "fixedfill"
signed	logical: return a signed backbone

mtc	string: type of Multiple Test Correction, either "none" or a method allowed by <code>p.adjust()</code> .
missing_as_zero	logical: treat missing edges as edges with zero weight and test them for significance
narrative	logical: display suggested text & citations
trials	numeric: if <code>model = "fdsm"</code> , the number of graphs generated using fastball to approximate the edge weight distribution
backbone_only	logical: return just the backbone (default), or a detailed backbone object

Details

The `backbone_from_projection` function extracts the backbone from the weighted projection of a bipartite network or hypergraph. The backbone is an unweighted unipartite network of agents that contains only edges whose weights in the projection are statistically significant. When `signed = FALSE`, the backbone contains edges that are statistically significantly strong under a one-tailed test. When `signed = TRUE`, the backbone contains positive edges that are statistically significantly strong, and negative edges that are statistically significantly weak, under a two-tailed test.

The `model` parameter controls the null model used to evaluate the statistical significance of edge weights. All available models are *statistical models* that are controlled by `alpha`, and differ in the constraints they impose on B:

- `sds`m (default) - The "Stochastic Degree Sequence Model" (SDSM; Neal et al., 2021) approximately constrains the agent and artifact degrees, and exactly constrains edges that are prohibited (`weight = 10`) or required (`weight = 11`; Neal & Neal, 2023)
- `fd`sm - The "Fixed Degree Sequence Model" (Neal et al., 2021) exactly constrains the agent and artifact degrees
- `fixedfill` - The "fixed fill" model (Neal et al., 2021) exactly constrains the total number of edges (i.e., sum)
- `fixedrow` - The "fixed row" model (Neal et al., 2021) exactly constrains the agent degrees (i.e., row sums)
- `fixedcol` - The "fixed column" model (Neal et al., 2021) exactly constrains the artifact degrees (i.e., column sums)

Although `backbone_from_projection` extracts the backbone from the weighted projection of a bipartite network or hypergraph, the input B *must be the bipartite network or hypergraph itself, and not the weighted projection*. This is necessary because these backbone models use information in the bipartite network that is missing from the projection. The "agent" nodes that appear in the projection must be represented by rows if B is an incidence matrix, or by `type = FALSE` nodes if B is a bipartite igraph object. In either case, the source network must be binary (i.e., unweighted), unless `model = "sds`m", when "prohibited" edges can be represented with `weight = 10` and "required" edges can be represented with `weight = 11`.

Value

A backbone in the same class as B, or if `backbone_only = FALSE`, then a backbone object.

References

package: Neal, Z. P. (2026). backbone: An R Package to Extract Network Backbones. PLOS One, 21, e0349258. doi:10.1371/journal.pone.0349258

sds-sm-ec model: Neal, Z. P. and Neal, J. W. (2023). Stochastic Degree Sequence Model with Edge Constraints (SDSM-EC) for Backbone Extraction. *International Conference on Complex Networks and Their Applications*, 12, 127-136. doi:10.1007/9783031534683_11

all other models: Neal, Z. P., Domagalski, R., and Sagan, B. (2021). Comparing Alternatives to the Fixed Degree Sequence Model for Extracting the Backbone of Bipartite Projections. *Scientific Reports*, 11, 23929. doi:10.1038/s41598021032383

Examples

```
#A binary bipartite network of 30 agents & 75 artifacts
#The 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)))
B <- igraph::graph_from_biadacency_matrix(B)

P <- igraph::bipartite_projection(B, which = "true") #An ordinary weighted projection...
plot(P)                                           #...is a dense hairball

bb <- backbone_from_projection(B) #A backbone...
plot(bb)                                         #...is sparse with clear communities
```

backbone_from_unweighted

Extract the backbone from an unweighted, undirected network

Description

backbone_from_unweighted() extracts the unweighted backbone from an unweighted, undirected network

Usage

```
backbone_from_unweighted(
  U,
  model = "lspar",
  parameter = 0.5,
  escore,
```

```

normalize,
filter,
umst,
narrative = FALSE,
backbone_only = TRUE
)

```

Arguments

U	An unweighted, undirected network as an adjacency matrix or Matrix , or an unweighted unipartite igraph object
model	string: backbone model
parameter	real: filtering parameter
escore	string: Method for scoring edges' importance
normalize	string: Method for normalizing edge scores
filter	string: Type of filter to apply
umst	logical: TRUE if the backbone should include the union of maximum spanning trees, to ensure connectivity
narrative	logical: display suggested text & citations
backbone_only	logical: return just the backbone (default), or a detailed backbone object

Details

The `backbone_from_unweighted` function extracts the backbone from an unweighted unipartite network. The backbone is an unweighted unipartite network that contains only edges preserved by a backbone model.

The following backbone models are available using the `model` parameter:

- `skeleton` - Skeleton backbone (Karger, 1999)
- `gspar` - Global Sparsification (Satuluri et al., 2011)
- `lspar` - Local Sparsification (Satuluri et al., 2011)
- `simmelian` - Simmelian backbone (Nick et al., 2013)
- `jaccard` - Jaccard backbone (Goldberg and Roth, 2003)
- `meetmin` - MeetMin backbone (Goldberg and Roth, 2003)
- `geometric` - Geometric backbone (Goldberg and Roth, 2003)
- `hyper` - Hypergeometric backbone, (Goldberg and Roth, 2003)
- `degree` - Local Degree backbone (Hamann et al, 2016)
- `quadrilateral` - Quadrilateral Simmelian backbone (Nocaj et al, 2015)
- `custom` - A custom backbone model specified by `escore`, `normalize`, `filter`, and `umst`

The `escore` parameter determines how an unweighted edge's importance is calculated.

- `random`: a random number drawn from a uniform distribution
- `betweenness`: edge betweenness

- triangles: number of triangles that include the edge
- jaccard: jaccard similarity coefficient of the neighborhoods of an edge's endpoints, or alternatively, triangles normalized by the size of the union of the endpoints neighborhoods
- dice: dice similarity coefficient of the neighborhoods of an edge's endpoints
- quadrangles: number of quadrangles that include the edge
- quadrilateral: 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 (inverted, so that larger is more important)

The `normalize` parameter determines whether edge scores are normalized.

- 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 (requires that `filter = degree`)
- 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.

- threshold: Edges with scores $>$ parameter are retained in the backbone
- proportion: Specifies the approximate proportion of edges to retain in the backbone
- degree: Retains each node's $d^{\text{parameter}}$ most important edges, where d is the node's degree (requires that `normalize = "rank"`)
- disparity: Applies the disparity filter using `backbone_from_weighted()`
- lans: Applies locally adaptive network sparsification using `backbone_from_weighted()`
- mlf: Applies the marginal likelihood filter using `backbone_from_weighted()`

Value

A backbone in the same class as `U`, or if `backbone_only = FALSE`, then a backbone object.

References

- package: Neal, Z. P. (2026). backbone: An R Package to Extract Network Backbones. PLOS One, 21, e0349258. doi:10.1371/journal.pone.0349258
- skeleton: Karger, D. R. (1999). Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 24, 383-413. doi:10.1287/moor.24.2.383
- gspar and lspar: Satuluri, V., Parthasarathy, S., & Ruan, Y. (2011, June). Local graph sparsification for scalable clustering. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (pp. 721-732). doi:10.1145/1989323.1989399

simmelian: Nick, B., Lee, C., Cunningham, P., & Brandes, U. (2013, August). Simmelian backbones: Amplifying hidden homophily in facebook networks. In Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining (pp. 525-532). doi:10.1145/2492517.2492569

jaccard, meetmin, geometric, hyper: Goldberg, D. S., & Roth, F. P. (2003). Assessing experimentally derived interactions in a small world. *Proceedings of the National Academy of Sciences*, 100, 4372-4376. doi:10.1073/pnas.0735871100

degree: Hamann, M., Lindner, G., Meyerhenke, H., Staudt, C. L., & Wagner, D. (2016). Structure-preserving sparsification methods for social networks. *Social Network Analysis and Mining*, 6, 22. doi:10.1007/s1327801603322

quadrilateral: Nocaj, A., Ortmann, M., & Brandes, U. (2015). Untangling the hairballs of multi-centered, small-world online social media networks. *Journal of Graph Algorithms and Applications*, 19, 595-618. doi:10.7155/jgaa.00370

Examples

```
#A dense, unweighted network with three embedded communities
U <- igrph::sample_sbm(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #Communities are not obvious

#Extract backbone using the built-in "Local Sparsification" model
bb <- backbone_from_unweighted(U, model = "lspar", parameter = 0.5)
plot(bb) #Communities are clearly visible

#Extract backbone using local sparification, but explicitly specifying the model steps
bb <- backbone_from_unweighted(U, model = "custom", escore = "jaccard",
                               normalize = "rank", filter = "degree",
                               umst = FALSE, parameter = 0.5)
plot(bb) #Communities are clearly visible
```

backbone_from_weighted

Extract the backbone from a weighted network

Description

backbone_from_weighted() extracts the unweighted backbone from a weighted network

Usage

```
backbone_from_weighted(
  W,
  model = "disparity",
  alpha = 0.05,
  signed = FALSE,
  mtc = "none",
  parameter = 0,
```

```

missing_as_zero = FALSE,
narrative = FALSE,
backbone_only = TRUE
)

```

Arguments

W	A weighted network as a valued adjacency matrix or Matrix , or a weighted unipartite igraph object
model	string: backbone model, one of: "disparity", "lans", "mlf", or "global"
alpha	real: significance level of hypothesis test(s) in statistical models
signed	logical: return a signed backbone from a statistical model
mtc	string: type of Multiple Test Correction, either "none" or a method allowed by p.adjust() .
parameter	real: parameter used to control structural backbone models
missing_as_zero	logical: treat missing edges as edges with zero weight and consider them for inclusion/exclusion in backbone
narrative	logical: display suggested text & citations
backbone_only	logical: return just the backbone (default), or a detailed backbone object

Details

The `backbone_from_weighted` function extracts the backbone from a weighted unipartite network. The backbone is an unweighted unipartite network that contains only edges whose weights are statistically significant (based on `alpha` for statistical models), or which exhibit certain structural properties (based on `parameter` for structural models). For statistical models, when `signed = FALSE`, the backbone contains edges that are statistically significantly strong under a one-tailed test. When `signed = TRUE`, the backbone contains positive edges that are statistically significantly strong, and negative edges that are statistically significantly weak, under a two-tailed test.

The `model` parameter controls the model used to evaluate the edge weights. The available models include:

Statistical Models (controlled by `alpha`, `signed`, and `mtc`)

- `disparity` (default) - The disparity filter (Serrano et al., 2009)
- `lans` - Locally adaptive network sparsification (Foti et al., 2011)
- `mlf` - Marginal likelihood filter (Dianati, 2016)

Structural Models (controlled by `parameter`)

- `global` - `parameter` is a numeric vector of length 1 or 2. If `length(parameter)==1`, then edges with weights above `parameter` are preserved. If `length(parameter)==2`, then edges with weights above `max(parameter)` are preserved as positive, and edges with weights above `min(parameter)` are preserved as negative.

The models implemented in `backbone_from_weighted()` can be applied to a weighted network that was obtained by projecting a bipartite network or hypergraph. However, if the original bipartite network or hypergraph is available, it is better to use [backbone_from_projection\(\)](#).

Value

A backbone in the same class as `W`, or if `backbone_only = FALSE`, then a backbone object.

References

package: Neal, Z. P. (2026). backbone: An R Package to Extract Network Backbones. *PLOS One*, 21, e0349258. doi:10.1371/journal.pone.0349258

disparity: Serrano, M. A., Boguna, M., & Vespignani, A. (2009). Extracting the multiscale backbone of complex weighted networks. *Proceedings of the National Academy of Sciences*, 106, 6483-6488. doi:10.1073/pnas.0808904106

lans: Foti, N. J., Hughes, J. M., & Rockmore, D. N. (2011). Nonparametric sparsification of complex multiscale networks. *PLOS One*, 6, e16431. doi:10.1371/journal.pone.0016431

mlf: Dianati, N. (2016). Unwinding the hairball graph: Pruning algorithms for weighted complex networks. *Physical Review E*, 93, 012304. doi:10.1103/PhysRevE.93.012304

Examples

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

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

mean_weight <- mean(igraph::E(W)$weight) #Find average edge weight
bb <- backbone_from_weighted(W, model = "global", #A backbone with stronger-than-average edges...
                           parameter = mean_weight)
plot(bb) #...ignores the weaker clique

bb <- backbone_from_weighted(W, model = "disparity") #A disparity filter backbone...
plot(bb) #...preserves edges at multiple scales
```

 bicm

Bipartite Configuration Model

Description

bicm estimates cell probabilities under the bipartite configuration model

Usage

```
bicm(M, fitness = FALSE, tol = 1e-08, max_steps = 200, ...)
```

Arguments

<code>M</code>	matrix: a binary matrix
<code>fitness</code>	logical: FALSE returns a matrix of probabilities, TRUE returns a list of row and column fitnesses only
<code>tol</code>	numeric, tolerance of algorithm
<code>max_steps</code>	numeric, number of times to run <code>.loglikelihood_prime_bicm</code> algorithm
<code>...</code>	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 to extract the backbone of a bipartite projection with the stochastic degree sequence model.

Matrix **M** is "conforming" if no rows and no columns contain only zeros or only ones. If **M** is conforming, then `bicm()` is faster. Additionally, if `fitness = TRUE`, then `bicm()` returns a list of row and column fitnesses, which requires less memory. Given the *i*th row's fitness R_i and the *j*th column's fitness R_j , the entry B_{ij} in the probability matrix can be computed as $R_i \times R_j / (1 + (R_i \times R_j))$.

Matrix **M** is "non-conforming" if any rows or any columns contain only zeros or only ones. If **M** is non-conforming, then `bicm()` is slower and will only return a probability matrix.

Value

a matrix of probabilities or a list of fitnesses

References

package: Neal, Z. P. (2025). backbone: An R Package to Extract Network Backbones. CRAN. doi:10.32614/CRAN.package.backbone

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

Examples

```
M <- matrix(c(0,0,1,0,1,0,1,0,1),3,3) #A binary matrix
bicm(M)
```

fastball	<i>Randomize a binary matrix using the fastball algorithm</i>
----------	---------------------------------------------------------------

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

fastball: Godard, Karl and Neal, Zachary P. 2022. fastball: A fast algorithm to sample bipartite graphs with fixed degree sequences. *Journal of Complex Networks* [doi:10.1093/comnet/cnac049](https://doi.org/10.1093/comnet/cnac049)

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
```

senate108	<i>Bill sponsorship in the 108th U.S. Senate</i>
-----------	--------------------------------------------------

Description

A bipartite network describing bill sponsorship in the 108th U.S. Senate

Usage

```
senate108
```

Format

senate108:

An [igraph](#) object containing a bipartite network with 100 senators and 3035 bills, where a senator is connected to a bill they sponsored or co-sponsored. Senator nodes include attributes for name, party affiliation, state, Govtrack ID, color (red for Republican, blue for Democrat, green for independent). Bill nodes include attributes for date introduced, title, legislative policy area, party affiliation of the sponsor, partisanship, and most recent status.

Source

These data were generated using `incidentally::incidence.from.congress(session = 108, types = "s", format = "igraph")`

Index

* datasets

senate108, [12](#)

backbone, [2](#)

backbone_from_projection, [3](#)

backbone_from_projection(), [2](#), [9](#)

backbone_from_unweighted, [5](#)

backbone_from_unweighted(), [2](#)

backbone_from_weighted, [8](#)

backbone_from_weighted(), [2](#), [7](#)

bicm, [10](#)

fastball, [12](#)

igraph, [2](#), [3](#), [6](#), [9](#), [13](#)

Matrix, [2](#), [3](#), [6](#), [9](#)

p.adjust(), [4](#), [9](#)

senate108, [12](#)