Package 'nethist'

October 25, 2026

Type Package

Title Network Histograms

Version 1.1.1

Date 2026-10-24

Maintainer Youngseok Song <youngseok.song@mail.wvu.edu>

Description Estimates network histograms, a blockmodel approximation to the graphon underlying a network's connectivity pattern, for both single-layer and multilayer networks. Implements graphon estimation methods including the profile-likelihood method of Olhede and Wolfe (2014) <doi:10.1073/pnas.1400374111> and the least-squares method of Gao, Lu, and Zhou (2015) <doi:10.1214/15-AOS1354> for single-layer networks, and the multilayer extension of Song and Olhede (2026) <doi:10.1093/comnet/cnag042>.

SystemRequirements Fortran compiler (for linking LAPACK/BLAS via 'RcppArmadillo')

Suggests testthat (>= 3.0.0), tinytest, igraphdata, xml2, knitr, rmarkdown, Matrix, network, ergm.multi, lifecycle, withr, mclust, plot3D

Encoding UTF-8

Imports lattice, methods, Rcpp(>= 1.0.9), stats, RSpectra, ggplot2, ggtext, graphics, reshape2, igraph, rlang

LinkingTo Rcpp, RcppArmadillo, testthat

Depends R (>= 3.5.0)

LazyData true

Config/testthat/edition 3

URL https://enigmasong.github.io/nethist/

BugReports https://github.com/EnigmaSong/nethist/issues

License MIT + file LICENSE

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

NeedsCompilation yes

Author Youngseok Song [aut, cre],
Sofia C. Olhede [aut]

Repository CRAN

Date/Publication 2026-10-25 14:20:02 UTC

Contents

covariate_plot 2
fitted.multinethist 4
fitted.nethist 5
hnethist 6
IndianVil 7
logLik.network_histogram 8
multinethist 9
nethist 11
nethist_control 13
netsummary_plot 14
network_histogram-class 16
plot.hnethist 17
plot.multinethist 18
plot.nethist 19
plot3d 21
polblog 22
polblog_edgelist 23
print.nethist 24
summary.network_histogram 25

Index **27**

`covariate_plot` *Bin summary by covariate*

Description

Drawing a bin summary plot of covariates given a `multinethist` object with a user-specified order.

Usage

```
covariate_plot(
  object,
  covariate,
  idx_order = 1:max(object$cluster),
  main = NA,
  xlab = NULL,
  ylab = NA,
  legend_title = NA,
```

```
  stat = "count",
  position = "stack"
)

summary_plot(object, covariate, ...)
```

Arguments

object	a multinethist object from multinethist().
covariate	a vector for univariate covariate. If it is a factor, a stacked bar chart is drawn. If it is numeric, a violin plot is drawn.
idx_order	A numeric vector for index label order, which must be a permutation of object$cluster. If NA, it uses 1:max(object$cluster).
main	title of summary plot. If NA, the plot has no title.
xlab	label of x-axis. If NULL, no label is shown.
ylab	label of y-axis. If NA, y-axis label is "covariate"
legend_title	title of legend. If NA, the legend title is "covariate"
stat	variables pass to ggplot2::geom_bar(). Only used for a factor covariate.
position	variables pass to ggplot2::geom_bar(). Only used for a factor covariate.
...	currently unused.

Details

When covariate is a factor, a stacked bar chart is drawn with bins ordered by idx_order. When covariate is numeric, a violin plot is drawn.

Value

A ggplot object. Printed as a side effect. Returns the plot invisibly.

Examples

```
set.seed(42)
data(polblog)
nethist_polblog <- multinethist(polblog)
x <- factor(c(rep("Liberal", 586), rep("Conservative", 638)))
covariate_plot(nethist_polblog, x)
```

`fitted.multinethist` *Fitted values for a multinethist model*

Description

Returns a subarray of the fitted block-probability or graphon matrix for a `multinethist` object.

Usage

```
## S3 method for class 'multinethist'
fitted(
  object,
  set1 = NULL,
  set2 = NULL,
  layer = NULL,
  type = "nethist",
  drop = TRUE,
  ...
)
```

Arguments

`object`	a `multinethist` object from `multinethist()`.
`set1`	integer vector of vertex indices for rows. `NULL` uses all vertices.
`set2`	integer vector of vertex indices for columns. `NULL` uses all vertices.
`layer`	integer vector of layer indices. `NULL` uses all layers.
`type`	one of `"nethist"` (default) or `"prob"`. `"nethist"` returns `thetahat[,,l] / rho_hat[l]` per layer; `"prob"` returns raw `thetahat`.
`drop`	logical. If `TRUE` (default) and a single layer is selected, the layer dimension is dropped and a matrix is returned. If `FALSE`, a 3-dimensional array is always returned.
`...`	currently unused.

Value

A numeric matrix of dimension `|set1| x |set2|` when a single layer is selected and `drop = TRUE`, otherwise a 3-dimensional array of dimension `|set1| x |set2| x |layer|`.

See Also

`multinethist()`, `nethist()`

Examples

```
data(IndianVil)
fit <- multinethist(IndianVil, h = 20L)
fitted(fit, set1 = 1:10, set2 = 1:10, layer = 1)
fitted(fit, layer = c(1, 2), drop = FALSE)
```

---

fitted.nethist *Fitted values for a nethist model*

Description

Returns a submatrix of the fitted block-probability or graphon matrix for a `nethist` or `hnethist` object.

Usage

```
## S3 method for class 'nethist'
fitted(object, set1 = NULL, set2 = NULL, type = "nethist", ...)
```

Arguments

`object`	a `nethist` or `hnethist` object from `nethist()` or `hnethist()`.
`set1`	integer vector of vertex indices for rows. `NULL` uses all vertices.
`set2`	integer vector of vertex indices for columns. `NULL` uses all vertices.
`type`	one of `"nethist"` (default) or `"prob"`. `"nethist"` returns the graphon estimate `thetahat / rho_hat`; `"prob"` returns the raw block probability matrix `thetahat`.
`...`	currently unused.

Value

A numeric matrix of dimension `|set1| x |set2|`.

See Also

`nethist()`, `hnethist()`, `multinethist()`

Examples

```
set.seed(1)
A <- igraph::as_adjacency_matrix(
  igraph::sample_gnp(60, 0.3), sparse = FALSE)
fit <- nethist(A, h = 10L)
fitted(fit, set1 = 1:10, set2 = 1:10)
fitted(fit, type = "prob")
```

hnethist *Hybrid Network histogram estimation*

Description

Estimating hybrid network histogram for single-layer networks and returning the indices of partitions.

Usage

```
hnethist(A, h = NA, method = "LSE", control = nethist_control(), ...)
```

Arguments

A	An adjacency matrix or an igraph object. It must be an undirected and simple graph.
h	A bandwidth parameter. If NA, the bandwidth is selected by Olhede and Wolfe (2014). If specified, the user-supplied value is used.
method	Type of loss function for network histogram. Must be one of LSE (default) or PLL for single-layer hybrid network histogram. See Details.
control	A control object from nethist_control. Governs max_itr, greedy_swap_rule, greedy_stop_threshold, and verbose.
...	Currently unused.

Details

Among the outputs, the best model is selected based on the BIC criterion.

The original reference provided theoretical guarantees for LSE, but we also allow PLL for the initial nethist fit. The block clustering and model selection steps are performed on the LSE loss regardless of the initial method, as the theoretical results pertain to LSE.

Both stages are stochastic. The initial nethist() fit uses a greedy search that swaps randomly chosen vertex pairs, and the merging step calls stats::kmeans(), which draws its starting centres at random. Neither stage seeds itself, so both consume the global random number stream and the selected number of shapes s depends on its state. Call set.seed() immediately before the fit to make results reproducible, and expect different seeds to give somewhat different values of s.

Value

A list of class c("hnethist", "nethist", "network_histogram") (see network_histogram-class) with the following fields:

- cluster an integer vector of length n with vertex-level block assignments from the initial nethist fit.
- thetahat a k-by-k probability matrix of the selected hybrid model, where k is the number of nethist blocks.

- rho_hat estimated sparsity parameter from the initial nethist fit.
- normalized_LL normalized log-likelihood of the selected model, divided by the number of edges.
- MSE mean squared error of the selected model.
- method loss function used ("LSE" or "PLL").
- h bandwidth used for the initial nethist fit.
- blockcluster a kmeans object describing how the k-by-k blocks were merged into s shapes.
- BIC BIC value of the selected model.
- s number of distinct shapes in the selected model.
- details list of all candidate models, one entry per number of shapes from 1 up to the maximum considered, each containing s, blockcluster, thetahat, MSE, normalized_LL, and BIC.
- initial the nethist object used as the starting point for block clustering.

References

Verdeyme, A. & Olhede, S. C. (2024). Hybrid of Node and Link Communities for Graphon Estimation. arXiv:2401.05088

Examples

```
set.seed(42)
data(polblog)
hfit <- hnethist(polblog)
hfit
plot(hfit)
```

IndianVil *Indian Village socioeconomic networks*

Description

A processed version of the Indian village dataset from Banerjee et al. (2013), representing socio-economic networks at the individual level for village ID 40.

Format

IndianVil is an array with size of 231 x 231 x 12.

Details

The array represents the socio-economic relationships of individuals from Village ID 40. Vertices with zero degrees across all layers were removed, reducing the total number of vertices from 241 to 231.

Source

https://doi.org/10.7910/DVN/U3BIHX

References

Banerjee, A., Chandrasekhar, A. G., Duflo, E., & Jackson, M. O. (2013). The diffusion of microfinance. Science, 341(6144), 1236498.

Song, Y. & Olhede, S. C. (2026). Joint Estimation of Sparse Multilayer Networks via Graph Limits. doi:10.1093/comnet/cnag042

Examples

```
data(IndianVil)

dim(IndianVil)
```

---

logLik.network_histogram

Log-likelihood of network histogram fits

Description

Extracts the Bernoulli log-likelihood of a fitted network histogram. The method is defined on the common base class, so it applies to `nethist`, `multinethist`, and `hnethist` objects alike; see network_histogram-class.

Usage

```
## S3 method for class 'network_histogram'
logLik(object, normalized = FALSE, ...)
```

Arguments

`object`	a `network_histogram` object, that is, a fit returned by `nethist`, `multinethist`, or `hnethist`.
`normalized`	logical. If `FALSE` (default), the unnormalized log-likelihood is returned as a `logLik` object. If `TRUE`, the per-edge normalized value stored in `object$normalized_LL` is returned as a plain numeric.
`...`	currently unused.

Details

The normalized value is not given class `logLik` on purpose. Passing a per-edge normalized value to `AIC()` or `BIC()` would return a meaningless number without any error.

The degrees of freedom count the free block probabilities: `k * (k + 1) / 2` for `nethist`, where `k` is the number of blocks; the same for `multinethist` fitted with `common_f = TRUE`, and `L` times that for `L` layers otherwise; and `s`, the number of selected shapes, for `hnethist`.

The number of observations is the number of Bernoulli edge variables, i.e. `n * (n - 1) / 2` for a single layer and `L` times that for `L` layers.

The log-likelihood is evaluated at the fitted probability matrix regardless of `method`, so it is also available for `method = "LSE"` fits.

Value

With `normalized = FALSE`, an object of class `logLik` carrying `df` and `nobs` attributes, so that `stats::AIC()` and `stats::BIC()` work directly on the fit. With `normalized = TRUE`, an unclassed numeric equal to `object$normalized_LL`.

See Also

network_histogram-class, `summary.network_histogram`

Examples

```
set.seed(42)
data(polblog)
fit <- suppressMessages(nethist(polblog))
logLik(fit)
logLik(fit, normalized = TRUE)
AIC(fit)
BIC(fit)
```

`multinethist` *Network histogram estimation*

Description

Estimating network histogram for multiplex networks and returning the indices of partitions.

Usage

```
multinethist(A, h = NA, common_f = FALSE, method = "PLL",
  control = nethist_control(), ...)
```

Arguments

A	Adjacency data for one or more network layers. Accepted formats: • **Single-layer**: a matrix, igraph object, or network object. • **Multilayer**: a 3D array of dimension $n \times n \times L$; a list of igraph or network objects (all sharing a common vertex set); or a combined_networks object from ergm.multi::Networks(). Plain matrix or sparse Matrix elements inside a list are not accepted because vertex ordering across layers cannot be verified. Use a 3D array instead. When a list of igraph or network objects is supplied, vertex names are used to align layers if present (via igraph::vertex_attr(g, "name") or network::network.vertex.names() If vertex names are absent, positional correspondence is assumed: vertex i in each layer is treated as the same vertex. All layers must have the same number of vertices, and when names are present they must form the same set.
h	A bandwidth parameter. If NA, the bandwidth is selected by Olhede and Wolfe (2014). If specified, the user-supplied value is used.
common_f	A logical; if TRUE, assumes a common network histogram function for all layers.
method	Type of loss function for network histogram. Must be one of PLL (default) or LSE. LSE is implemented only for single-layer networks. See Details.
control	A control object from nethist_control. Governs max_itr, greedy_swap_rule, greedy_stop_threshold, and verbose.
...	Currently unused.

Details

The lth layer's multi-network histogram is defined by thetahat/rho_hat. The multinetwork histogram can be plotted using plot() and plot3d().

If the number of layers is 1, multinethist() fits a single-layer network histogram, equivalent to nethist().

method = "PLL" is for Olhede and Wolfe (2014), and method = "LSE" is for Gao et al. (2015). method = "LSE" is implemented only for single-layer networks, and an error is raised for multilayer input.

Note that cluster only shows a partition of vertices, and the index labels are not ordered. For example, vertices in cluster 1 do not have to be more similar to vertices in cluster 2 than to vertices in cluster 10. Hence, users may specify a custom order in plot.multinethist().

The greedy search is stochastic. At each iteration it draws a pair of vertices uniformly at random and swaps their block labels if the objective improves, so it converges to a local optimum that depends on the state of the random number stream. Call set.seed() immediately before the fit to make results reproducible, and expect different seeds to give somewhat different partitions.

Value

If the number of layers is greater than 1, it returns an object of class c("multinethist", "network_histogram") (see network_histogram-class):

- cluster a vector of partition indices.

- `thetahat` a probability array from multinetwork histogram ordered by group labels.
- `rho_hat` a vector of estimated sparsity parameters.
- `normalized_LL` a normalized likelihood from the algorithm.
- `method` loss function used. Always `"PLL"` for multilayer fits.
- `homogeneous` a logical variable indicating homogeneous multinetwork histogram.
- `h` bandwidth used for estimation.

References

Song, Y. & Olhede, S. C. (2026). Joint Estimation of Sparse Multilayer Networks via Graph Limits. doi:10.1093/comnet/cnag042

Olhede, S. C. & Wolfe, P. J. (2014). Network Histograms and Universality of Blockmodel Approximation. Proceedings of the National Academy of Sciences, 111(41), 14722-14727. doi:10.1073/pnas.1400374111

Gao, C., Lu, Y., & Zhou, H. H. (2015). Rate-Optimal Graphon Estimation. The Annals of Statistics, 43(6), 2624-2652. doi:10.1214/15-AOS1354

See Also

`plot.multinethist()`, `plot.nethist()`, `nethist_control()`

Examples

```
#single-layer network histogram
set.seed(42)
data(polblog)
nethist(polblog)
nethist_polblog <- nethist(polblog)
nethist_polblog_with_h <- nethist(polblog, h = 72)

#multi-network histogram
set.seed(42)
data(IndianVil)
IndianVil
mnethist_Ind_Vil <- multinethist(IndianVil)
```

`nethist` *Network histogram estimation for single-layer networks*

Description

Estimating a network histogram for a single-layer network and returning the indices of partitions.

Usage

```
nethist(A, h = NA, method = "PLL", control = nethist_control(), ...)
```

Arguments

A	An adjacency matrix or graph object. Accepted formats: a matrix, sparse dgCMatrix, igraph object, or network object. Must be an undirected simple graph.
h	A bandwidth parameter. If NA, the bandwidth is selected by Olhede and Wolfe (2014). If specified, the user-supplied value is used.
method	Type of loss function. One of "PLL" (default, profile log-likelihood) or "LSE" (least squares). See Details.
control	A control object from nethist_control. Governs max_itr, greedy_swap_rule, greedy_stop_threshold, and verbose.
...	**[Deprecated]** Pass max_itr, greedy_swap_rule, greedy_stop_threshold, or verbose via control = nethist_control(...) instead.

Details

method = "PLL" is for Olhede and Wolfe (2014). method = "LSE" is for Gao et al. (2015).

Note that cluster labels are not ordered: vertices in cluster 1 are not necessarily more similar to cluster 2 than to cluster 10. Users may specify a custom display order in plot.nethist.

The greedy search is stochastic. At each iteration it draws a pair of vertices uniformly at random and swaps their block labels if the objective improves, so it converges to a local optimum that depends on the state of the random number stream. Call set.seed immediately before the fit to make results reproducible, and expect different seeds to give somewhat different partitions.

Value

An object of class c("nethist", "network_histogram") (see network_histogram-class) with the following fields:

- cluster an integer vector of length n with block assignments.
- thetahat a $k \times k$ probability matrix ordered by group labels.
- rho_hat estimated sparsity parameter.
- normalized_LL normalized log-likelihood.
- MSE mean squared error.
- method loss function used ("PLL" or "LSE").
- h bandwidth used for estimation.

References

Olhede, S. C. & Wolfe, P. J. (2014). Network Histograms and Universality of Blockmodel Approximation. PNAS, 111(41), 14722-14727. doi:10.1073/pnas.1400374111

Gao, C., Lu, Y., & Zhou, H. H. (2015). Rate-Optimal Graphon Estimation. The Annals of Statistics, 43(6), 2624-2652. doi:10.1214/15-AOS1354

See Also

multinethist, plot.nethist, nethist_control

Examples

```
set.seed(42)
data(polblog)
fit <- nethist(polblog)
fit
plot(fit)

fit_h <- nethist(polblog, h = 72)
```

`nethist_control` *Control parameters for network histogram algorithms*

Description

Constructs a control object for `nethist`, `multinethist`, and `hnethist`.

Usage

```
nethist_control(
  algorithm = "greedy",
  max_itr = 5e+06,
  greedy_swap_rule = "single_random",
  greedy_stop_threshold = 20000,
  verbose = FALSE,
  ...
)

## S3 method for class 'nethist_control'
print(x, ...)
```

Arguments

`algorithm`	character. Optimization algorithm. Currently only `"greedy"` is implemented.
`max_itr`	integer. Maximum number of iterations. Default is 5×10^6.
`greedy_swap_rule`	character. Vertex-pair selection rule for the greedy search. At each iteration, two vertices are drawn and their group labels are swapped if the move improves the objective. Currently only `"single_random"` (one pair drawn uniformly at random) is implemented.
`greedy_stop_threshold`	integer. Early stopping criterion for the greedy search; the algorithm terminates if the objective has not improved for this many consecutive iterations. Default is 20,000.
`verbose`	logical. Print progress messages during fitting.
`...`	Accepts deprecated argument names `swap_rule` and `consecutive_iter_threshold` with a warning.
`x`	a `nethist_control` object.

Value

An object of class "nethist_control".

See Also

nethist, multinethist, hnethist

Examples

```
# default control object
ctrl <- nethist_control()
print(ctrl)

# reduce iteration limit for quick testing
ctrl <- nethist_control(max_itr = 1e4, greedy_stop_threshold = 100)


data(polblog)
fit <- nethist(polblog, control = nethist_control(max_itr = 1e4))
```

---

netsummary_plot *Network summary plots*

Description

Draw a network summary plot proposed by Maugis et al. (2017). To count k-cycles, Alon et al. (1997) is used.

Usage

```
netsummary_plot(
  A,
  subsample_sizes = NA,
  max_cycle_order = 4,
  n_rep = NA,
  n_subsample_sizes = 11,
  alpha = 0.05,
  y_max = NA,
  save_plot = FALSE,
  filename = "myplot.pdf",
  width = 7,
  height = 5,
  max_subsample_size = 250,
  ...
)

violin_netsummary(A, ...)
```

Arguments

A	an adjacency matrix, igraph object, or network object to draw a network summary plot. It must be an undirected and simple graph.
subsample_sizes	a numeric vector of vertex subsample sizes. If NA, the subsample size is selected automatically.
max_cycle_order	an integer value of the maximum cycle size. Must be >=3 and <=7.
n_rep	an integer value of subsampling replication. If NA, n_rep is automatically selected by alpha.
n_subsample_sizes	number of different subsample sizes for automatic selection. It is only used when subsample_sizes = NA.
alpha	a pre-specified level used in determining n_rep and subsample_sizes when they are not specified. It must be in (0,1). Default is 0.05. Smaller alpha gives larger n_rep and subsample_sizes.
y_max	Upper limit of y-axis of the plot. Must be 0 < y_max <= 1. If NA, the upper limit is automatically selected.
save_plot	A logical indicating whether to save the generated figure. If TRUE, the plot is saved via ggplot2::ggsave() using the specified file name. Otherwise, the plot is displayed.
filename	file name to save the generated figure.
width	a numeric value of the width of the generated figure in inch. It is only used when save_plot = TRUE.
height	a numeric value of the height of the generated figure in inch. It is only used when save_plot = TRUE.
max_subsample_size	integer. Upper bound on the automatically selected subsample size. Larger values improve statistical accuracy but increase computation time. Default is 250.
...	**[Deprecated]** Pass R, Ns, y.max, or save.plot via the renamed arguments n_rep, n_subsample_sizes, y_max, and save_plot instead.

Details

Vertex sampling is done by simple random sampling without replacement.

The automatically selected subsample size is capped at max_subsample_size to limit computation time.

Each violin shows the distribution of the subsampled statistic, and a dot marks the mean.

The following input classes are supported: base::matrix, Matrix::dgCMatrix, igraph::igraph, network::network.

Value

A ggplot object. Printed as a side effect when save_plot = FALSE. Returns the plot invisibly when save_plot = TRUE.

References

Maugis et al. (2017). Topology reveals universal features for network comparison. arXiv: 1705.05677

Alon et al. (1997). Finding and counting given length cycles. Algorithmica 17, 209–223 (1997). https://doi.org/10.1007/BF02523189

Examples

```
{
set.seed(2022)
#Generating Erdos-Renyi graph
n <- 400
#igraph object
A <- igraph::sample_gnp(n, 0.05)
netsummary_plot(A)
}

#sparse adjacency matrix
A2 <- igraph::as_adjacency_matrix(A)
netsummary_plot(A2)

#dense adjacency matrix
A2 <- igraph::as_adjacency_matrix(A, sparse = FALSE)
netsummary_plot(A2)

#user-specified n_rep and subsample_sizes
netsummary_plot(A, n_rep = 500, subsample_sizes = 150)

#user-specified alpha
netsummary_plot(A, alpha = 0.1)

#network object
if (requireNamespace("network", quietly = TRUE)) {
  A3 <- network::as.network(igraph::as_adjacency_matrix(A, sparse = FALSE))
  netsummary_plot(A3)
}

#user-specified max_subsample_size
netsummary_plot(A, max_subsample_size = 100)

#saving the plot with user-specified file name
netsummary_plot(A, save_plot = TRUE,
                filename = file.path(tempdir(), "myfig.pdf"))
```

---

network_histogram-class

Network histogram class hierarchy

Description

All fitted objects returned by nethist, multinethist, and hnethist inherit from the common base class network_histogram:

Details

- nethist fits have class c("nethist", "network_histogram").
- multinethist fits have class c("multinethist", "network_histogram").
- hnethist fits have class c("hnethist", "nethist", "network_histogram").

Every network_histogram object carries cluster, thetahat, rho_hat, normalized_LL, method, and h. Methods that only depend on those fields are defined once on the base class, so logLik.network_histogram and summary.network_histogram are shared by all three. Methods whose behaviour depends on the layer dimension, such as plot(), plot3d(), and fitted(), remain specific to the subclasses.

See Also

logLik.network_histogram, summary.network_histogram, print.nethist

---

plot.hnethist *Plot an hnethist object*

Description

Plots a heatmap or a BIC curve for an hnethist object.

Usage

```
## S3 method for class 'hnethist'
plot(x, type = "nethist", at = NULL, ...)
```

Arguments

x	an hnethist object from hnethist().
type	One of nethist (default), prob, or BIC. When type = "BIC", plots BIC values against the number of shapes s. The selected model is marked with a dashed vertical line, and the rightmost point (largest s, corresponding to the initial nethist fit) is labelled. type = "bic" is also accepted.
at	A numeric vector of breakpoints for the color scale. Passed to plot.nethist(). See plot.nethist() for details.
...	Additional arguments passed to plot() when type = "BIC", or to plot.nethist() otherwise.

Value

Called for its side effects (plotting). Returns NULL invisibly.

See Also

plot.nethist(), hnethist()

Examples

```
set.seed(2022)
A <- igraph::as_adjacency_matrix(
  igraph::sample_gnp(100, 0.3), sparse = FALSE)
fit <- suppressMessages(hnethist(A))
plot(fit)
plot(fit, type = "BIC")
```

---

plot.multinethist *Network histogram plot*

Description

Drawing lattice::levelplot() using a multinethist object.

Usage

```
## S3 method for class 'multinethist'
plot(
  x,
  y = NA,
  type = "nethist",
  idx_order = 1:max(x$cluster),
  power = 0.25,
 col.regions = function(n) grDevices::hcl.colors(n, palette = "Reds 3", rev = TRUE),
  colorkey = FALSE,
  prob = FALSE,
  digits = 2,
  prob.cex = 0.1 + 0.5/log10(max(x$cluster)),
  prob.col = "white",
  layout = NULL,
  layer_titles = NULL,
  at = NULL,
  ...
)
```

Arguments

x	a multinethist object from multinethist().
y	A dummy variable for S3 dispatch. Never used.
type	One of nethist (default) or prob. "MNhist" is a deprecated alias for "nethist".

idx_order	A numeric vector for index label order, which must be a permutation of x$cluster. If NA, it uses 1:max(x$cluster).
power	A positive number for the power transform applied to the graphon estimate. Only used when type = "nethist". Default is 0.25.
col.regions	A function taking an integer n and returning n colors. Default is viridis via grDevices::hcl.colors().
colorkey	Logical. Whether to draw a color legend. Default FALSE.
prob	Logical. Whether to print block probabilities on the plot. Default FALSE.
digits	Integer. Number of decimal places for probabilities.
prob.cex	Numeric. cex for probability labels.
prob.col	Color for probability labels. Default "white".
layout	An integer vector c(nrows, ncols) specifying the panel grid for multi-layer plots, following the mfrow convention. If NULL (default), each layer is plotted as a separate figure.
layer_titles	A character vector of length equal to the number of layers plotted, giving each panel's title. If NULL (default), titles default to "Layer 1", "Layer 2", etc.
at	A numeric vector of breakpoints for the color scale. If NULL (default), breakpoints are computed automatically from the data range. Specify a fixed vector to compare plots from different fits on a common scale.
...	Additional arguments passed to lattice::levelplot().

Value

Called for its side effects (plotting). Returns NULL invisibly.

Examples

```
set.seed(42)
data(IndianVil)
mnhist_Ind_vil <- multinethist(IndianVil)
plot(mnhist_Ind_vil)
plot(mnhist_Ind_vil, power = 0.5)
plot(mnhist_Ind_vil, layout = c(3,4))
plot(mnhist_Ind_vil, layer_titles = paste0("Network ", seq_along(mnhist_Ind_vil$rho_hat)))
```

---

plot.nethist *Network histogram plot*

Description

Drawing lattice::levelplot() using a nethist object.

Usage

```
## S3 method for class 'nethist'
plot(
  x,
  type = "nethist",
  idx_order = 1:max(x$cluster),
  power = 0.25,
 col.regions = function(n) grDevices::hcl.colors(n, palette = "Reds 3", rev = TRUE),
  colorkey = FALSE,
  prob = FALSE,
  digits = 2,
  prob.cex = 0.1 + 0.5/log10(max(x$cluster)),
  prob.col = "white",
  at = NULL,
  y = NA,
  ...
)
```

Arguments

x	a nethist object from nethist().
type	One of nethist (default) or prob. "pmat" is a deprecated alias for "prob".
idx_order	A numeric vector for index label order, which must be a permutation of x$cluster. If NA, it uses 1:max(x$cluster).
power	A positive number for the power transform applied to the graphon estimate. Only used when type = "nethist". Default is 0.25.
col.regions	A function taking an integer n and returning n colors. Default is viridis via grDevices::hcl.colors().
colorkey	Logical. Whether to draw a color legend. Default FALSE.
prob	Logical. Whether to print block probabilities on the plot. Default FALSE.
digits	Integer. Number of decimal places for probabilities.
prob.cex	Numeric. cex for probability labels.
prob.col	Color for probability labels. Default "white".
at	A numeric vector of breakpoints for the color scale. If NULL (default), breakpoints are computed automatically from the data range. Specify a fixed vector to compare plots from different fits on a common scale.
y	A dummy variable for S3 dispatch. Never used.
...	Additional arguments passed to lattice::levelplot().

Value

Called for its side effects (plotting). Returns NULL invisibly.

Examples

```
set.seed(2022)
A <- igraph::sample_gnp(200, 0.05)
hist_A <- nethist(A)
plot(hist_A)
plot(hist_A, power = 0.5)
plot(hist_A, type = "prob", prob = TRUE)
```

`plot3d` *3D histogram plot for nethist objects*

Description

Drawing `plot3D::hist3D()` using a `nethist`, `multinethist`, or `hnethist` object with a user-specified order.

Usage

```
plot3d(x, idx_order = 1:max(x$cluster), type = "nethist", ...)

## S3 method for class 'nethist'
plot3d(x, idx_order = 1:max(x$cluster), type = "nethist", ...)

## S3 method for class 'hnethist'
plot3d(x, idx_order = 1:max(x$cluster), type = "nethist", ...)

## S3 method for class 'multinethist'
plot3d(x, idx_order = 1:max(x$cluster), type = "nethist", ...)
```

Arguments

`x`	a `nethist`, `multinethist`, or `hnethist` object.
`idx_order`	A numeric vector for index label order, which must be a permutation of `x$cluster`. If `NA`, it uses `1:max(x$cluster)`.
`type`	One of `"nethist"` (default) or `"prob"`. `"MNhist"` is a deprecated alias for `"nethist"`.
`...`	Other arguments passed to `plot3D::hist3D()`.

Value

Called for its side effects (plotting). Returns `NULL` invisibly.

Examples

```
set.seed(42)
data(IndianVil)
mnhist_Ind_vil <- multinethist(IndianVil)
plot3d(mnhist_Ind_vil)

data(polblog)
fit <- nethist(polblog)
plot3d(fit)
```

`polblog` *Political blog*

Description

A pre-processed version of the Political Blog dataset from Olhede and Wolfe (2014), based on the original data by Adamic and Glance (2005). The provided igraph object represents the graph after removing zero-degree nodes and simplifying the graph structure. The graph is derived from the edge list `polblog_edgelist`.

Usage

```
data(polblog)
```

Format

`polblog` is an igraph object, which is pre-processed data symmetrize and simplified without self-loops.

Details

The adjacency matrix from `polblog_edgelist` is sparse, asymmetric, so it needs to be symmetrized if a method requires a symmetric matrix. Nodes with zero degree need to be removed. All edges are considered as undirected edges. Then, we get the undirected version of dataset as `polblog`.

First 586 blogs are liberal; remaining 638 are conservative.

Source

https://www-personal.umich.edu/~mejn/netdata/

https://github.com/p-wolfe/network-histogram-code

References

Adamic, L. A., & Glance, N. (2005). The political blogosphere and the 2004 US election: divided they blog. In Proceedings of the 3rd international workshop on Link discovery (pp. 36-43)

Examples

```
data(polblog_edgelist)
data(polblog)

#From polblog_edgelist to polblog
G<- igraph::graph_from_edgelist(as.matrix(polblog_edgelist), directed = FALSE)
G<- igraph::delete_vertices(G, igraph::degree(G) == 0)
G<- igraph::simplify(G)

polblog
```

---

polblog_edgelist *Political blog (edgelist)*

Description

Political blog data set from Olhede and Wolfe (2014), which is part of the original data set from Adamic and Glance (2005). The provided edgelist is from GitHub repository of Prof. Patrick Wolfe.

Usage

```
data("polblog_edgelist")
```

Format

polblog_edgelist is a matrix for the edge list from https://github.com/p-wolfe/network-histogram-code.

`V1` a numeric vector of tail vertices.

`V2` a numeric vector of head vertices.

Details

First 586 blogs are liberal; remaining 638 are conservative. Nodes with zero degree need to be removed.

Source

https://www-personal.umich.edu/~mejn/netdata/

https://github.com/p-wolfe/network-histogram-code

References

Adamic, L. A., & Glance, N. (2005). The political blogosphere and the 2004 US election: divided they blog. In Proceedings of the 3rd international workshop on Link discovery (pp. 36-43)

Examples

```
data(polblog_edgelist)
data(polblog)

#From polblog_edgelist to polblog
G<- igraph::graph_from_edgelist(as.matrix(polblog_edgelist), directed = FALSE)
G<- igraph::delete_vertices(G, igraph::degree(G) == 0)
G<- igraph::simplify(G)

polblog
```

`print.nethist` *Print nethist objects*

Description

Compact one-screen description of `nethist`, `multinethist`, and `hnethist` objects. The estimated probabilities are reported by their dimensions only; use `summary.network_histogram` for the model fit statistics and `object$thetahat` for the values themselves.

Usage

```
## S3 method for class 'nethist'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'multinethist'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'hnethist'
print(x, digits = getOption("digits"), ...)
```

Arguments

`x`	a `nethist`, `multinethist`, or `hnethist` object.
`digits`	number of significant digits used when printing the loss.
`...`	additional arguments. Currently unused.

Value

The input object, invisibly.

See Also

summary.network_histogram

Examples

```
set.seed(42)
data(polblog)
fit <- suppressMessages(nethist(polblog))
print(fit)
summary(fit)
```

---

summary.network_histogram
Summarize network histogram fits

Description

Detailed summary of a fitted network histogram. The print methods of the fitted objects give a short identification only; the model fit statistics are reported here.

Usage

```
## S3 method for class 'network_histogram'
summary(object, ...)

## S3 method for class 'multinethist'
summary(object, ...)

## S3 method for class 'hnethist'
summary(object, ...)

## S3 method for class 'summary.network_histogram'
print(x, digits = getOption("digits"), ...)
```

Arguments

object	a network_histogram object, that is, a fit returned by nethist, multinethist, or hnethist.
...	additional arguments. Currently unused.
x	a summary.network_histogram object.
digits	number of significant digits used when printing numeric quantities.

Details

The summary method is defined on the common base class, so it applies to nethist, multinethist, and hnethist objects alike. The subclass methods add the fields specific to them and are otherwise deferred to the base method via NextMethod(); see network_histogram-class.

The estimated probabilities themselves are not printed, since the matrix grows with the number of blocks. Use object$thetahat or fitted() to obtain them.

Value

summary() returns an object inheriting from summary.network_histogram: a list holding the fitted object together with the number of nodes, the number of blocks, the log-likelihood, the degrees of freedom, AIC, and BIC. Multilayer fits add n_layers, and hnethist fits add the selected number of shapes s and the number of candidate models n_candidate. The print method returns its input invisibly.

See Also

network_histogram-class, print.nethist, logLik.network_histogram

Examples

```
set.seed(42)
data(polblog)
fit <- suppressMessages(nethist(polblog))
fit
summary(fit)
```

Index

∗ **datasets**
IndianVil, 7
polblog, 22
polblog_edgelist, 23

base::matrix, *15*

covariate_plot, 2

fitted(), *26*
fitted.multinethist, 4
fitted.nethist, 5

ggplot2::geom_bar(), *3*
ggplot2::ggsave(), *15*
grDevices::hcl.colors(), *19, 20*

hnethist, 6, *8*, *13, 14*, *17*, *25*
hnethist(), *5*, *17, 18*

igraph::igraph, *15*
IndianVil, 7

lattice::levelplot(), *18–20*
logLik.network_histogram, 8, *17*, *26*

Matrix::dgCMatrix, *15*
multinethist, *8*, 9, *12–14*, *17*, *25*
multinethist(), *3–5*, *18*

nethist, *8*, 11, *13, 14*, *17*, *25*
nethist(), *4–6*, *20*
nethist_control, *6*, *10*, *12*, 13
nethist_control(), *11*
netsummary_plot, 14
network::network, *15*
network_histogram-class, *6*, *8–10*, *12*, 16, *26*

plot(), *10*, *17*
plot.hnethist, 17
plot.multinethist, 18
plot.multinethist(), *10, 11*
plot.nethist, *12*, 19
plot.nethist(), *11*, *17, 18*
plot3d, 21
plot3d(), *10*
plot3D::hist3D(), *21*
polblog, 22
polblog_edgelist, 23
print.hnethist *(*print.nethist*)*, 24
print.multinethist *(*print.nethist*)*, 24
print.nethist, *17*, 24, *26*
print.nethist_control *(*nethist_control*)*, 13
print.summary.network_histogram *(*summary.network_histogram*)*, 25

set.seed, *12*
set.seed(), *6*, *10*
stats::AIC(), *9*
stats::BIC(), *9*
stats::kmeans(), *6*
summary.hnethist *(*summary.network_histogram*)*, 25
summary.multinethist *(*summary.network_histogram*)*, 25
summary.network_histogram, *9*, *17*, *24, 25*, 25
summary_plot *(*covariate_plot*)*, 2

violin_netsummary *(*netsummary_plot*)*, 14