

Package ‘MixMashNet’

May 7, 2026

Title Tools for Multilayer and Single Layer Network Modeling

Version 1.0.0

Maintainer Maria De Martino <maria.demartino@uniud.it>

Description Estimation and bootstrap utilities for single layer and multilayer Mixed Graphical Models, including functions for centrality, bridge metrics, membership stability, and plotting (De Martino et al. (2026) <[doi:10.48550/arXiv.2602.05716](https://doi.org/10.48550/arXiv.2602.05716)>).

License AGPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 4.1)

Imports mgm, igraph, qgraph, colorspace, future.apply, stats, utils, ggplot2, EGAnet, networktools, dplyr, magrittr, rlang, tibble, tidy, patchwork, progressr

RoxygenNote 7.3.3

URL <https://arcbiostat.github.io/MixMashNet/>

BugReports <https://github.com/ARCbiostat/MixMashNet/issues>

NeedsCompilation no

Author Maria De Martino [aut, cre],
Caterina Gregorio [aut],
Adrien Perigord [ctb],
Hudson Golino [ctb],
Jonas Haslbeck [ctb]

Repository CRAN

Date/Publication 2026-04-30 19:00:02 UTC

Contents

MixMashNet-package	2
bacteremia	3
bridge_metrics	4

bridge_metrics_excluded	5
community_scores	6
find_bridge_communities	8
find_bridge_layers	9
get_centrality	10
get_edges	12
layer_slice	14
membershipStab	14
mixMN	15
multimixMN	20
nhanes	24
plot.mixMN_fit	26
plot.multimixMN_fit	27
print.mixMN_fit	29
print.multimixMN_fit	30
summary.mixMN_fit	30
summary.multimixMN_fit	31
update_palette	32

Index	34
--------------	-----------

MixMashNet-package	<i>Tools for Multilayer and Single Layer Network Modeling</i>
--------------------	---

Description

Tools for estimating and analyzing single layer and multilayer networks using Mixed Graphical Models (MGMs), accommodating continuous, count, and categorical variables. In the multilayer setting, layers may comprise different types and numbers of variables, and users can explicitly impose a predefined multilayer topology to constrain the estimation of inter and intralayer connections. The package implements bootstrap procedures to derive quantile regions for edge weights and node-level centrality and bridge metrics, and provides tools to assess the stability of node community membership. In addition, subject-level community scores can be computed to summarize the latent dimensions identified through network clustering.

Author(s)

Maria De Martino, Caterina Gregorio, Adrien Perigord
<maria.demartino@uniud.it>

References

- De Martino, M., Triolo, F., Perigord, A., Ornago, A. M., Vetrano, D. L., Gregorio, C. (2026). Mix-MashNet: An R Package for Single and Multilayer Networks. <https://arxiv.org/abs/2602.05716>
- Christensen, A. P., & Golino, H. (2021). Estimating the Stability of Psychological Dimensions via Bootstrap Exploratory Graph Analysis: A Monte Carlo Simulation and Tutorial. *Psych*, 3(3), 479–500. doi:10.3390/psych3030032

- Christensen, A. P., Golino, H., Abad, F. J., & Garrido, L. E. (2025). Revised network loadings. *Behavior Research Methods*, 57(4), 114. doi:10.3758/s13428025026403
- Epskamp, S., Borsboom, D., & Fried, E. I. (2018). Estimating psychological networks and their accuracy: A tutorial paper. *Behavior Research Methods*, 50(1), 195–212. doi:10.3758/s13428017-08621
- Haslbeck, J. M. B., & Waldorp, L. J. (2020). mgm: Estimating Time-Varying Mixed Graphical Models in High-Dimensional Data. *Journal of Statistical Software*, 93(8). doi:10.18637/jss.v093.i08
- Jones, P. J., Ma, R., & McNally, R. J. (2021). Bridge Centrality: A Network Approach to Understanding Comorbidity. *Multivariate Behavioral Research*, 56(2), 353–367. doi:10.1080/00273171.2019.1614898

See Also

Useful links:

- <https://arcbiostat.github.io/MixMashNet/>
- Report bugs at <https://github.com/ARCbiostat/MixMashNet/issues>

bacteremia

Bacteremia example dataset for single layer network analysis

Description

Clinical bacteremia dataset used to illustrate single-layer network estimation with MixMashNet. This dataset contains 7240 patients with clinical suspicion of bacteremia who underwent blood culture testing at the Vienna General Hospital.

Usage

```
data(bacteremia)
```

Format

A data frame with 7420 rows and 16 variables:

AGE Age (numeric).

WBC White blood cell (numeric).

NEU Neutrophil counts (numeric).

HGB Hemoglobin (numeric).

PLT Platelet count (numeric).

CRP C-reactive protein (numeric).

APTT Activated partial thromboplastin time (numeric).

FIB Fibrinogen (numeric).

CREA Creatinine (numeric).

BUN Blood urea nitrogen (numeric).
GLU Glucose (numeric).
ALAT High-sensitivity C-reactive protein (numeric).
GBIL Total bilirubin (numeric).
ALB Albumin (numeric).
SEX Sex with 0=male and 1=female.
BloodCulture Positive blood culture results with 0=no and 1=yes.

References

Ratzinger, F., Dedeyan, M., Rammerstorfer, M., Perkmann, T., Burgmann, H., Makristathis, A., Dorffner, G., Loetsch, F., Blacky, A., & Ramharter, M. (2014). A risk prediction model for screening bacteremic patients: A cross sectional study. *PLoS ONE*, 9(9), e106765. doi:10.1371/journal.pone.0106765

bridge_metrics

Bridge metrics for nodes across communities

Description

Computes bridge centrality measures for nodes with an assigned community. This function is used internally by `mixMN()` and `multimixMN()`. Specifically, the function computes bridge strength as the sum of absolute edge weights connecting a node to nodes in other communities; bridge expected influence of order one (EI1) as the signed sum of direct connections to nodes in other communities; bridge expected influence of order two (EI2) as the signed influence that propagates indirectly to nodes in other communities via one intermediate neighbor (i.e., through paths of length two); bridge betweenness as the number of times a node lies on shortest paths between nodes belonging to different communities; and bridge closeness as the inverse of the mean shortest-path distance to nodes in other communities.

Usage

```
bridge_metrics(g, membership)
```

Arguments

<code>g</code>	An igraph object with edge attribute weight.
<code>membership</code>	Named vector/factor of community labels for a subset of nodes (names must match <code>V(g)\$name</code>).

Details

Bridge betweenness and closeness are computed on the positive-weight subgraph only, with weights converted to distances as $d = 1/w$.

Value

A data.frame with columns: node, community, bridge_strength, bridge_ei1, bridge_ei2, bridge_betweenness, bridge_closeness.

References

Jones, P. J., Ma, R., & McNally, R. J. (2021). Bridge Centrality: A Network Approach to Understanding Comorbidity. *Multivariate Behavioral Research*, 56(2), 353–367. doi:10.1080/00273171.2019.1614898

bridge_metrics_excluded

Bridge metrics for nodes excluded from communities

Description

Computes bridge centrality measures for nodes that are not assigned to any community. This function is used internally by mixMN() and multimixMN(). For these excluded nodes, the function computes bridge strength, bridge closeness, bridge betweenness, and bridge expected influence of order one and two (EI1 and EI2), quantifying their role in connecting nodes across different communities.

Usage

```
bridge_metrics_excluded(g, membership)
```

Arguments

g	An igraph object with edge attribute weight.
membership	Named vector/factor of community labels for a subset of nodes (names must match V(g)\$name). Nodes not present here are treated as excluded.

Details

Bridge betweenness excluded and closeness excluded are computed on the positive-weight subgraph only, with weights converted to distances as $d = 1/w$.

Value

A data.frame with columns: node, bridge_strength, bridge_closeness, bridge_betweenness, bridge_ei1, bridge_ei2.

References

Jones, P. J. (2025). **networktools**: Tools for identifying important nodes in networks. R package version 1.6.1. <https://github.com/paytonjjones/networktools>

community_scores

*Compute community scores from a fitted MixMashNet model***Description**

Computes subject-level community scores. Community scores are obtained as weighted sums of the variables belonging to each detected community, where weights correspond to the standardized community loadings estimated via EGAnet::net.loads and stored in the fitted mixMN_fit object. Scores are computed using the dataset provided via the data argument. If data = NULL, the original dataset used to fit the model (fit\$model\$data) is used by default. Errors if both are NULL. Optionally, percentile bootstrap quantile regions for the community scores can be computed if bootstrap community loadings are available in fit\$community_loadings\$boot. Community scores are only available if community loadings were computed in the fitted model. This requires that all variables in the community subgraph are of MGM type Gaussian ("g"), Poisson ("p"), or binary categorical ("c" with level == 2).

Usage

```
community_scores(
  fit,
  data = NULL,
  layer = NULL,
  scale = TRUE,
  quantile_level = NULL,
  return_quantile_region = FALSE,
  na_action = c("stop", "omit")
)
```

Arguments

fit	A fitted object of class c("mixMN_fit", "multimixMN_fit") returned by mixMN() or multimixMN().
data	Optional data.frame with variables in columns. If NULL, uses fit\$model\$data. Errors if both are NULL.
layer	Optional. If fit is a multimixMN_fit, specify which layer to score (name or index). If NULL, scores are computed for all scoreable layers and returned as a named list. If no layer is scoreable, the function errors.
scale	Logical; if TRUE (default), z-standardize variables used for scoring, using the mean/SD computed from the dataset used for scoring.
quantile_level	Optional numeric from 0 to 1, e.g. 0.95 or 0.99. If provided, percentile bootstrap quantile regions are computed for community scores (requires fit\$community_loadings\$boot).
return_quantile_region	Logical; if TRUE, return quantile regions.
na_action	Character. How to handle missing values in the scoring data: "stop" (default) stops if any missing value is present in the required variables; "omit" computes scores using row-wise omission within each community (i.e., uses available variables only, re-normalizing weights within community for that row).

Details

The function requires that `fit$community_loadings$true` exists and that the input data contains all required variables in `fit$community_loadings$nodes`. It errors otherwise.

The returned object has class "community_scores" and implements `print()` and `summary()` methods for quick inspection and descriptive summaries of the scores.

Value

A list with class "community_scores" containing:

`call` The matched call.

`settings` List with `scale`, `quantile_level`, and `na_action`.

`ids` Character vector of subject IDs (rownames of data).

`communities` Character vector of community score names.

`scores` Numeric matrix of scores ($n \times K$).

`quantile_region` If requested and available, a list with lower and upper matrices ($n \times K$) for percentile bootstrap quantile regions; otherwise NULL.

`details` List containing `nodes_used`, `loadings_true`, `loadings_boot_available`, and scaling parameters (`center`, `scale`).

If `fit` is a `mixMN_fit` (or a `multimixMN_fit` with `layer` specified), returns a "community_scores" object. If `fit` is a `multimixMN_fit` and `layer = NULL`, returns a named list of community_scores objects for all scoreable layers. If no layer is scoreable, the function errors.

References

Christensen, A. P., Golino, H., Abad, F. J., & Garrido, L. E. (2025). Revised network loadings. *Behavior Research Methods*, 57(4), 114. doi:10.3758/s13428025026403

Examples

```
data(bacteremia)

vars <- c("WBC", "NEU", "HGB", "PLT", "CRP")
df <- bacteremia[, vars]

fit <- mixMN(
  data = df,
  lambdaSel = "EBIC",
  reps = 0,
  seed_model = 42,
  compute_loadings = TRUE,
  progress = FALSE,
  save_data = TRUE
)

# Compute community scores on the original data
scores <- community_scores(fit)
summary(scores)
```

 find_bridge_communities

Bridge profiles of a node across communities

Description

Identifies which communities contribute most to the bridge role of a given node, by decomposing its bridge connectivity into community-specific contributions, excluding its own community when assigned. The function is designed as an interpretative companion to `bridge_metrics()` and `bridge_metrics_excluded()`, providing the components underlying the corresponding overall bridge indices.

Bridge connectivity is summarized using five complementary profiles: bridge strength, bridge EI1, bridge EI2, bridge closeness, and bridge betweenness.

For single layer fits (`mixMN_fit`), profiles are computed directly on the supplied fitted object.

For multilayer fits (`multimixMN_fit`), profiles are computed within the selected layer only, by applying the same single layer procedure to the corresponding intralayer fit stored in `fit$layer_fits[[layer]]`.

Usage

```
find_bridge_communities(fit, node, layer = NULL)
```

Arguments

<code>fit</code>	An object of class <code>mixMN_fit</code> or <code>multimixMN_fit</code> .
<code>node</code>	Character scalar: node of interest.
<code>layer</code>	Character scalar giving the layer of interest for <code>multimixMN_fit</code> objects. Ignored for <code>mixMN_fit</code> objects.

Details

Bridge profiles are computed using only connections from the focal node to nodes in communities different from its own. If the focal node is not assigned to any community, i.e. excluded, connections to all assigned nodes in communities are considered.

Bridge betweenness is computed by counting all shortest paths between pairs of nodes in different communities that pass through the focal node as an intermediate vertex. When multiple shortest paths exist, each path is counted separately.

The returned object has class `"bridge_profiles"` and provides a dedicated `print()` method. By default, all bridge profiles are displayed; a specific profile can be selected through the `statistic` argument.

Value

An object of class `"bridge_profiles"` (a named list) with the following components:

`bridge_strength` Bridge strength. List with overall, the total value across all other communities, and `by_comm`, a tibble with community-specific contributions (`community`, `sum_abs_w`).

bridge_ei1 Bridge expected influence (order 1). List with overall and by_comm (community, sum_signed_w).

bridge_ei2 Bridge expected influence (order 2). List with overall and by_comm (community, sum_signed_w2).

bridge_closeness Bridge closeness. List with overall and by_comm (community, inv_mean_dist).

bridge_betweenness Bridge betweenness. List with overall and by_pair, a tibble with contributions by community pair (Ci, Cj, hits).

find_bridge_layers *Bridge profiles of a node across layers*

Description

Identifies which layers contribute most to the interlayer bridge role of a given node, by decomposing its interlayer connectivity into layer-specific contributions. The function is designed as an interpretative companion to the interlayer node-level indices returned by `multimixMN()`, providing the components underlying the corresponding overall interlayer indices.

Interlayer connectivity is summarized using four complementary profiles: interlayer strength, interlayer expected influence (order 1), interlayer closeness, and interlayer betweenness.

Usage

```
find_bridge_layers(fit, node, layer)
```

Arguments

fit	An object of class <code>multimixMN_fit</code> .
node	Character scalar: node of interest.
layer	Character scalar giving the layer of the focal node.

Details

The function operates on the interlayer-only graph, i.e. on the graph containing only edges between nodes belonging to different layers.

For a focal node in the selected layer, the function decomposes:

- interlayer strength into contributions toward each layer;
- interlayer expected influence (order 1) into signed contributions toward each layer;
- interlayer closeness into additive harmonic-distance contributions toward each layer;
- interlayer betweenness into additive contributions from shortest paths between layer pairs, using the standard fraction $\sigma_{st}(v)/\sigma_{st}$.

Contributions are defined so that they sum to the corresponding overall interlayer index.

The returned object has class `"bridge_layer_profiles"` and provides a dedicated `print()` method. By default, all interlayer profiles are displayed; a specific profile can be selected through the `statistic` argument.

Value

An object of class "bridge_layer_profiles" (a named list) with the following components:

bridge_strength List with overall and by_layer, where by_layer is a tibble with columns target_layer and sum_abs_w.

bridge_ei1 List with overall and by_layer, where by_layer is a tibble with columns target_layer and sum_signed_w.

bridge_closeness List with overall and by_layer, where by_layer is a tibble with columns target_layer and contribution.

bridge_betweenness List with overall and by_pair, where by_pair is a tibble with columns Li, Lj, and contribution.

<code>get_centrality</code>	<i>Extract node-level centrality indices</i>
-----------------------------	--

Description

Extracts node-level centrality indices from fitted objects returned by `mixMN()` and `multimixMN()` in a long-format data frame.

For single layer fits of class "mixMN_fit", only intralayer node-level indices are available.

For multilayer fits of class "multimixMN_fit", `what = "intra"` returns intralayer node-level indices, whereas `what = "inter"` returns node-level indices computed on the interlayer-only graph. If `what` is not specified for a multilayer fit, both intralayer and interlayer node-level indices are returned by default, unless `layer` or `pairs` imply a specific scope.

The function returns the original estimates and, when available, bootstrap means, standard errors, and bootstrap quantile regions.

Usage

```
get_centrality(object, ...)

## S3 method for class 'mixMN_fit'
get_centrality(
  object,
  what = "intra",
  statistics = NULL,
  digits = NULL,
  drop_na_boot = TRUE,
  ...
)

## S3 method for class 'multimixMN_fit'
get_centrality(
  object,
  what = c("intra", "inter"),
```

```

    statistics = NULL,
    layer = NULL,
    pairs = NULL,
    digits = NULL,
    drop_na_boot = TRUE,
    ...
  )

```

Arguments

object	A fitted object of class "mixMN_fit" or "multimixMN_fit".
...	Further arguments passed to methods.
what	Character string indicating which node-level indices to extract: <ul style="list-style-type: none"> • "intra": intralayer node-level indices; • "inter": interlayer-only node-level indices (available only for "multimixMN_fit" objects). <p>For single layer fits, only "intra" is allowed. For multilayer fits, if omitted, both scopes are returned by default unless layer or pairs imply a specific scope.</p>
statistics	Character vector specifying which node-level statistics to include.
digits	Optional number of digits used to round numeric columns.
drop_na_boot	Logical. If TRUE (default), bootstrap-related columns that are entirely NA are removed from the output.
layer	Optional character vector of layer names to subset. Relevant for intralayer output in multilayer fits.
pairs	Optional character vector of layer-pair names to subset. Relevant for interlayer output in multilayer fits.

Details

The returned data frame is in long format, with one row per node-statistic combination.

For single layer fits, only what = "intra" is available.

For multilayer fits, layer can be used to subset intralayer output, whereas pairs can be used to subset interlayer output.

The set of admissible statistics depends on what and on the class of object. In particular, bridge-related indices are available only for intralayer output.

Value

A tibble in long format with one row per node-statistic combination. It contains the columns:

- node
- layer
- scope
- metric

- estimated

When available, the output also contains bootstrap summary columns:

- mean.bootstrap
- SE.bootstrap
- quantile.lower.bootstrap
- quantile.upper.bootstrap

The quantile level used to compute the bootstrap quantile region is stored as the "quantile_level" attribute of the returned tibble.

get_edges

Extract edge-level summaries

Description

Extracts edge-level summaries from fitted objects returned by `mixMN()` and `multimixMN()` in a long-format data frame.

For single layer fits of class "mixMN_fit", only intralayer edges are available.

For multilayer fits of class "multimixMN_fit", `what = "intra"` returns intralayer edges, whereas `what = "inter"` returns interlayer edges. If `what` is not specified for a multilayer fit, both scopes are returned by default, unless `layer` or `pairs` imply a specific scope.

The function returns the original edge weights and, when available, bootstrap means, standard errors, and bootstrap quantile regions.

Usage

```
get_edges(object, ...)
```

```
## S3 method for class 'mixMN_fit'
```

```
get_edges(object, what = "intra", digits = NULL, drop_na_boot = TRUE, ...)
```

```
## S3 method for class 'multimixMN_fit'
```

```
get_edges(
  object,
  what = c("intra", "inter"),
  layer = NULL,
  pairs = NULL,
  digits = NULL,
  drop_na_boot = TRUE,
  ...
)
```

Arguments

object	A fitted object of class "mixMN_fit" or "multimixMN_fit".
...	Further arguments passed to methods.
what	Character string indicating which edge-level summaries to extract: <ul style="list-style-type: none"> • "intra": intralayer edges; • "inter": interlayer edges (available only for "multimixMN_fit" objects). For single layer fits, only "intra" is allowed. For multilayer fits, if omitted, both scopes are returned by default unless layer or pairs imply a specific scope.
digits	Optional number of digits used to round numeric columns.
drop_na_boot	Logical. If TRUE (default), bootstrap-related columns that are entirely NA are removed from the output.
layer	Optional character vector of layer names to subset. Relevant for intralayer output in multilayer fits.
pairs	Optional character vector of layer-pair names to subset. Relevant for interlayer output in multilayer fits.

Details

The returned data frame is in long format, with one row per edge.

For single layer fits, only what = "intra" is available.

For multilayer fits, layer can be used to subset intralayer output, whereas pairs can be used to subset interlayer output.

Value

A tibble in long format with one row per edge. It contains the columns:

- edge
- layer for intralayer edges
- pairs for interlayer edges
- scope
- estimated

When available, the output also contains bootstrap summary columns:

- mean.bootstrap
- SE.bootstrap
- quantile.lower.bootstrap
- quantile.upper.bootstrap

The quantile level used to compute the bootstrap quantile region is stored as the "quantile_level" attribute of the returned tibble.

layer_slice	<i>Extract a single layer from a multilayer MixMashNet object</i>
-------------	---

Description

Extracts one layer from a fitted multilayer `multimixMN_fit` object returned by `multimixMN()`. The selected layer is returned as the corresponding single layer `mixMN_fit` object stored in `layer_fits`.

Usage

```
layer_slice(object, ...)

## S3 method for class 'multimixMN_fit'
layer_slice(object, layer, ...)
```

Arguments

<code>object</code>	An object of class "multimixMN_fit" returned by <code>multimixMN()</code> .
<code>...</code>	Further arguments passed to methods.
<code>layer</code>	Character string giving the layer to extract.

Value

An object of class "mixMN_fit" corresponding to the selected layer.

membershipStab	<i>Node stability from bootstrap community assignments</i>
----------------	--

Description

Computes per-node stability given the empirical community structure and the homogenized bootstrap memberships contained in a `mixMN_fit` object. Stability is expressed as the proportion of bootstrap replications that assign each node to its empirical (original) community.

Usage

```
membershipStab(fit)
```

Arguments

<code>fit</code>	An object returned by <code>mixMN()</code> (class <code>mixMN_fit</code>), containing <code>\$communities\$original_membership</code> and <code>\$communities\$boot_memberships</code> . Bootstrap memberships must be available, i.e. <code>reps > 0</code> and "community" %in% <code>boot_what</code> .
------------------	--

Details

Bootstrap community labels are first aligned to the empirical solution using `EGAnet::community.homogenize()`. Stability is then computed node-wise as the proportion of bootstrap runs in which the node's community matches its empirical assignment.

The returned object has class "membershipStab" and provides `print()`, `summary()`, and `plot()` methods for quick inspection, descriptive summaries, and visualization of node stability.

Value

An object of class `c("membershipStab")`, with components:

`membership` List with:

`empirical` Named integer vector of empirical community labels

`bootstrap` Matrix of homogenized bootstrap labels ($\text{reps} \times p$)

`membership.stability` List with:

`empirical.dimensions` Named numeric vector of node-level stability (proportion assigned to empirical community)

`all.dimensions` Matrix ($p \times K$) with proportions of assignment to each community

`community_palette` Named vector of colors for communities, if available

References

Christensen, A. P., & Golino, H. (2021). Estimating the Stability of Psychological Dimensions via Bootstrap Exploratory Graph Analysis: A Monte Carlo Simulation and Tutorial. *Psych*, 3(3), 479–500. doi:10.3390/psych3030032

mixMN

Estimate single layer MGM network with bootstrap centrality, bridge metrics, clustering, and (optionally) community score loadings

Description

Estimates a single layer Mixed Graphical Model (MGM) network on the original data, using the estimation framework implemented in the **mgm** package, and performs non-parametric bootstrap (row resampling) to compute centrality indices, bridge metrics, clustering stability, and quantile regions for node metrics and edge weights. Optionally, the function computes community score loadings (for later prediction on new data) and can bootstrap the corresponding loadings.

Usage

```
mixMN(
  data,
  reps = 100,
  scale = TRUE,
  lambdaSel = c("CV", "EBIC"),
```

```

lambdaFolds = 5,
lambdaGam = 0.25,
alphaSeq = 1,
alphaSel = "CV",
alphaFolds = 5,
alphaGam = 0.25,
k = 2,
ruleReg = "AND",
threshold = "LW",
overparameterize = FALSE,
thresholdCat = TRUE,
quantile_level = 0.95,
covariates = NULL,
exclude_from_cluster = NULL,
treat_singletons_as_excluded = FALSE,
seed_model = NULL,
seed_boot = NULL,
cluster_method = c("louvain", "edge_betweenness", "fast_greedy", "infomap",
  "label_prop", "leading_eigen", "leiden", "optimal", "spinglass", "walktrap"),
cluster_args = list(),
compute_loadings = TRUE,
boot_what = c("general_index", "bridge_index", "excluded_index", "community",
  "loadings"),
save_data = FALSE,
progress = TRUE
)

```

Arguments

data	A data.frame (n x p) with variables in columns. Variables may be numeric, integer, logical, or factors. Character and Date/POSIXt variables are not supported and must be converted prior to model fitting. Variable types are internally mapped to MGM types as follows: continuous numeric (double) variables are treated as Gaussian; integer variables are treated as Poisson unless they take only values in {0,1}, in which case they are treated as binary categorical; factors and logical variables are treated as categorical. Binary categorical variables (two-level factors and logical variables) are internally recoded to {0,1} for model fitting. The original input data are not modified.
reps	Integer (≥ 0). Number of bootstrap replications.
scale	Logical; if TRUE (default) Gaussian variables (<code>type == "g"</code>) are z-standardized internally by <code>mgm()</code> . Use <code>scale = FALSE</code> if your data are already standardized.
lambdaSel	Method for lambda selection: "CV" or "EBIC".
lambdaFolds	Number of folds for CV (if <code>lambdaSel = "CV"</code>).
lambdaGam	EBIC gamma parameter (if <code>lambdaSel = "EBIC"</code>).
alphaSeq	Alpha parameters of the elastic net penalty (values between 0 and 1).
alphaSel	Method for selecting the alpha parameter: "CV" or "EBIC".

alphaFolds	Number of folds for CV (if alphaSel = "CV").
alphaGam	EBIC gamma parameter (if alphaSel = "EBIC").
k	Integer (≥ 1). Order of modeled interactions.
ruleReg	Rule to combine neighborhood estimates: "AND" or "OR".
threshold	Threshold below which edge-weights are set to zero: Available options are "LW", "HW", or "none". "LW" applies the threshold proposed by Loh & Wainwright; "HW" applies the threshold proposed by Haslbeck & Waldorp; "none" disables thresholding. Defaults to "LW".
overparameterize	Logical; controls how categorical interactions are parameterized in the neighborhood regressions. If TRUE, categorical interactions are represented using a fully over-parameterized design matrix (i.e., all category combinations are explicitly modeled). If FALSE, the standard glmnet parameterization is used, where one category serves as reference. For continuous variables, both parameterizations are equivalent. The default is FALSE. The over-parameterized option may be advantageous when distinguishing pairwise from higher-order interactions.
thresholdCat	Logical; if FALSE thresholds of categorical variables are set to zero.
quantile_level	Level of the central bootstrap quantile region (default 0.95). Must be a single number between 0 and 1.
covariates	Character vector. Variables used as adjustment covariates in model estimation.
exclude_from_cluster	Character vector. Nodes excluded from community detection (in addition to covariates).
treat_singletons_as_excluded	Logical; if TRUE, singleton communities (size 1) are treated as excluded nodes when computing bridge metrics.
seed_model	Optional integer seed for reproducibility of the initial MGM fit.
seed_boot	Optional integer seed passed to future.apply for reproducibility of bootstrap replications.
cluster_method	Community detection method used on the clustering graph. Either a character string naming one of the built-in methods "louvain", "edge_betweenness", "fast_greedy", "infomap", "label_prop", "leading_eigen", "leiden", "optimal", "spinglass", "walktrap", or a user-supplied function. If a function is supplied, it must accept a graph through argument graph and return either an igraph communities object, a list with component membership, or a membership vector of length equal to the number of nodes used for clustering.
cluster_args	Named list of additional arguments passed to the selected community detection method. For example, steps for "walktrap", nb.trials for "infomap", spins for "spinglass". Ignored if not relevant for the selected method.
compute_loadings	Logical; if TRUE (default), compute community loadings (EGAnet::net.loads). Only supported for Gaussian, Poisson, and binary categorical nodes; otherwise loadings are skipped and the reason is stored in community_loadings\$reason.

boot_what	Character vector specifying which quantities to bootstrap. Valid options are: "general_index" (centrality indices), "bridge_index" (bridge metrics for nodes in communities), "excluded_index" (bridge metrics for nodes treated as excluded), "community" (community memberships), "loadings" (community loadings, only if compute_loadings = TRUE), and "none" (skip all node-level bootstrap: only edge-weight bootstrap is performed if reps > 0).
save_data	Logical; if TRUE, store the original data in the output object.
progress	Logical; if TRUE (default), show a bootstrap progress bar.

Details

This function does **not** call `future::plan()`. To enable parallel bootstrap, set a plan (e.g. `future::plan(multisession)`) before calling `mixMN()`. If `boot_what` is "none" and `reps > 0`, node-level metrics are not bootstrapped but edge-weight bootstrap and corresponding quantile regions are still computed.

Value

An object of class "mixMN_fit", that is a list with the following top-level components:

`call` The matched function call.

`settings` List of main settings used in the call, including `reps`, `cluster_method`, `cluster_args`, `covariates`, `exclude_from_cluster`, `treat_singletons_as_excluded`, and `boot_what`.

`data_info` List with information derived from the input data used for model setup: `mgm_type_level` (data frame with one row per variable, reporting the original R class and the inferred MGM type and level, as used in the call to `mgm::mgm`), and `binary_recode_map` (named list describing the mapping from original binary labels to the internal {0,1} coding used for model fitting).

`model` List with: `mgm` (the fitted mgm object), `nodes` (character vector of all node names), `n` (number of observations), `p` (number of variables), and `data` (if `save_data = TRUE`).

`graph` List describing the graph: `igraph` (an **igraph** object built on `keep_nodes_graph`, with edge attributes `weight`, `abs_weight`, `sign` and vertex attribute `membership` for communities), `keep_nodes_graph` (nodes retained in the graph and all node-level metrics), and `keep_nodes_cluster` (nodes used for community detection).

`communities` List describing community structure with: `original_membership` (integer vector of community labels on `keep_nodes_cluster`), `groups` (factor of community labels actually used for bridge metrics, optionally with singletons treated as excluded), `palette` (named vector of colors per community), and `boot_memberships` (list of bootstrap memberships if "community" is requested in `boot_what`, otherwise an empty list).

`statistics` List with node- and edge-level summaries: `node` is a list with: `true` (data frame with one row per node in `keep_nodes_graph`, containing the node name and metrics `strength`, `ei1`, `closeness`, `betweenness`, `bridge_strength`, `bridge_betweenness`, `bridge_closeness`, `bridge_ei1`, `bridge_ei2`, and for nodes treated as excluded from communities also `bridge_strength_excluded`, `bridge_betweenness_excluded`, `bridge_closeness_excluded`, `bridge_ei1_excluded`, `bridge_ei2_excluded`); `boot` (list of bootstrap matrices for each metric, each of dimension `reps` x `length(keep_nodes_graph)`, possibly NULL if the metric was not requested or if `reps = 0`); and `quantile_region` (list of quantile regions for each node metric, one `p` x 2 matrix per metric, with columns corresponding to the lower and upper quantile bounds implied by `quantile_level`, or NULL if no bootstrap was performed).

edge is a list with: `true` (data frame with columns `edge` and `weight` for all unique undirected edges among `keep_nodes_graph`); `boot` (matrix of bootstrap edge weights of dimension `n_edges` x `reps`); and `quantile_region` (matrix of quantile regions for edge weights, `n_edges` x 2, with columns corresponding to the lower and upper bootstrap quantile bounds, or `NULL` if `reps = 0`).

`community_loadings` List containing community-loading information (based on `EGAnet::net.loads`) for later community-score computation on new data: `nodes` (nodes used for loadings), `wc` (integer community labels aligned with `nodes`), `true` (matrix of standardized loadings, `nodes` x `communities`, or `NULL` if loadings were not computed.), `boot` (list of bootstrap loading matrices, one per replication, or `NULL` if not bootstrapped), `available` (logical indicating whether loadings were computed), `reason` (character string explaining why loadings were not computed, or `NULL` if `available = TRUE`), `non_scorable_nodes` (character vector of nodes in the community subgraph that prevented loadings from being computed (e.g., categorical variables with >2 levels), otherwise empty).

References

- Haslbeck, J. M. B., & Waldorp, L. J. (2020). `mgm`: Estimating Time-Varying Mixed Graphical Models in High-Dimensional Data. *Journal of Statistical Software*, 93(8). doi:10.18637/jss.v093.i08
- Loh, P. L., & Wainwright, M. J. (2012). Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses. *NIPS*

Examples

```
data(bacteremia)
df <- bacteremia[, !names(bacteremia) %in% "BloodCulture"]

fit <- mixMN(
  data = df,
  lambdaSel = "EBIC",
  lambdaGam = 0.25,
  reps = 0,
  seed_model = 42,
  cluster_method = "louvain",
  covariates = c("AGE", "SEX"),
  compute_loadings = FALSE,
  progress = FALSE
)
fit

# Plot the estimated network
set.seed(1)
plot(fit)

fit_b <- mixMN(
  data = df,
  lambdaSel = "EBIC",
  lambdaGam = 0.25,
```

```

    reps = 5,
    seed_model = 42,
    seed_boot = 42,
    cluster_method = "louvain",
    covariates = c("AGE", "SEX"),
    boot_what = "community",
    compute_loadings = FALSE,
    progress = FALSE
  )

# Plot the membership stability
plot(fit_b, what = "stability", cutoff = 0.7)

```

multimixMN

Multilayer MGM with bootstrap, intra/interlayer metrics, and quantile regions

Description

Estimates a multilayer Mixed Graphical Model (MGM) using the estimation framework implemented in the **mgm** package, with a masking scheme that enforces which cross-layer edges are allowed according to `layer_rules`. Within each layer, the function computes community structure and performs non-parametric row-bootstrap to obtain node centrality indices, edge weights, and bridge metrics, including metrics for nodes treated as excluded. Optionally, within-layer community loadings can also be estimated and bootstrapped. The function additionally returns interlayer-only node metrics and summaries of cross-layer edge weights.

Usage

```

multimixMN(
  data,
  layers,
  layer_rules,
  scale = TRUE,
  reps = 100,
  lambdaSel = c("CV", "EBIC"),
  lambdaFolds = 5,
  lambdaGam = 0.25,
  alphaSeq = 1,
  alphaSel = "CV",
  alphaFolds = 5,
  alphaGam = 0.25,
  k = 2,
  ruleReg = "AND",
  threshold = "LW",
  overparameterize = FALSE,

```

```

thresholdCat = TRUE,
quantile_level = 0.95,
covariates = NULL,
exclude_from_cluster = NULL,
seed_model = NULL,
seed_boot = NULL,
treat_singletons_as_excluded = FALSE,
cluster_method = c("louvain", "edge_betweenness", "fast_greedy", "infomap",
  "label_prop", "leading_eigen", "leiden", "optimal", "spinglass", "walktrap"),
cluster_args = list(),
compute_loadings = TRUE,
boot_what = c("general_index", "interlayer_index", "bridge_index", "excluded_index",
  "community", "loadings"),
save_data = FALSE,
progress = TRUE
)

```

Arguments

data	A data.frame (n x p) with variables in columns. Variables may be numeric, integer, logical, or factors. Character and Date/POSIXt variables are not supported and must be converted prior to model fitting. Variable types are internally mapped to MGM types as follows: continuous numeric (double) variables are treated as Gaussian; integer variables are treated as Poisson unless they take only values in {0,1}, in which case they are treated as binary categorical; factors and logical variables are treated as categorical. Binary categorical variables (two-level factors and logical variables) are internally recoded to {0,1} for model fitting. The original input data are not modified.
layers	A named vector (names = variable names) assigning each node to a layer (character or factor). Must cover all columns of data except variables listed in covariates (treated as adjustment covariates).
layer_rules	A square matrix (L x L), where L is the number of layers. Row and column names must match the layer names. Entries equal to TRUE or 1 allow cross-layer edges between the corresponding pair of layers, while FALSE or 0 disallow them. The matrix is symmetrized internally. Diagonal entries are ignored (intralayer edges are always permitted).
scale	Logical; if TRUE (default) Gaussian variables (type == "g") are z-standardized internally by mgm(). Use scale = FALSE if your data are already standardized.
reps	Integer (>= 0). Number of bootstrap replications (row resampling with replacement). If reps = 0, no bootstrap is performed.
lambdaSel	Method for lambda selection in mgm: "CV" or "EBIC".
lambdaFolds	Number of folds for CV (if lambdaSel = "CV").
lambdaGam	EBIC gamma parameter (if lambdaSel = "EBIC").
alphaSeq	Alpha parameters of the elastic net penalty (values between 0 and 1).
alphaSel	Method for selecting the alpha parameter: "CV" or "EBIC".
alphaFolds	Number of folds for CV (if alphaSel = "CV").

alphaGam	EBIC gamma parameter (if alphaSel = "EBIC").
k	Integer (≥ 1). Order of modeled interactions.
ruleReg	Rule to combine neighborhood estimates: "AND" or "OR".
threshold	Threshold below which edge-weights are set to zero: Available options are "LW", "HW", or "none". "LW" applies the threshold proposed by Loh & Wainwright; "HW" applies the threshold proposed by Haslbeck & Waldorp; "none" disables thresholding. Defaults to "LW".
overparameterize	Logical; controls how categorical interactions are parameterized in the neighborhood regressions. If TRUE, categorical interactions are represented using a fully over-parameterized design matrix (i.e., all category combinations are explicitly modeled). If FALSE, the standard glmnet parameterization is used, where one category serves as reference. For continuous variables, both parameterizations are equivalent. The default is FALSE. The over-parameterized option may be advantageous when distinguishing pairwise from higher-order interactions.
thresholdCat	Logical; if FALSE thresholds of categorical variables are set to zero.
quantile_level	Level of the central bootstrap quantile region (default 0.95). Must be a single number between 0 and 1.
covariates	Character vector. Variables used as adjustment covariates in model estimation.
exclude_from_cluster	Character vector of node names. Nodes in this set are excluded from community detection in addition to covariates.
seed_model	Optional integer seed for reproducibility of the initial MGM fit.
seed_boot	Optional integer seed passed to future.apply for reproducibility of bootstrap replications.
treat_singletons_as_excluded	Logical; if TRUE, singleton communities (size 1) are treated as "excluded" when computing bridge metrics and related summaries.
cluster_method	Community detection method used within each layer. Either a character string naming one of the built-in methods "louvain", "edge_betweenness", "fast_greedy", "infomap", "label_prop", "leading_eigen", "leiden", "optimal", "spinglass", "walktrap", or a user-supplied function. If a function is supplied, it must accept a graph through argument graph and return either an igraph communities object, a list with component membership, or a membership vector of length equal to the number of nodes used for clustering.
cluster_args	Named list of additional arguments passed to the selected community detection method. For example, steps for "walktrap", nb.trials for "infomap", spins for "spinglass". Ignored if not relevant for the selected method.
compute_loadings	Logical; if TRUE (default), compute community loadings (EGAnet::net.loads). Only supported for Gaussian, Poisson, and binary categorical nodes; otherwise loadings are skipped and the reason is stored in community_loadings\$reason.

boot_what	Character vector specifying which quantities to bootstrap. Valid options are: "general_index" (intralayer centrality indices), "interlayer_index" (interlayer-only node metrics), "bridge_index" (bridge metrics for nodes in communities), "excluded_index" (bridge metrics for nodes treated as excluded), "community" (community memberships), "loadings" (within-layer community loadings, only if compute_loadings = TRUE), and "none" (skip all node-level bootstrap: only edge-weight bootstrap is performed if reps > 0).
save_data	Logical; if TRUE, store the original data in the output object.
progress	Logical; if TRUE (default), show a bootstrap progress bar.

Details

This function does **not** call `future::plan()`. To enable parallel bootstrap, set a plan (e.g. `future::plan(multisession)`) before calling `multimixMN()`. If "none" is the only element of `boot_what` and `reps > 0`, node-level metrics are not bootstrapped, but intra and interlayer edge-weight bootstrap and the corresponding quantile regions are still computed.

Value

An object of class "multimixMN_fit". The returned list contains at least the following components:

`call` The matched function call.

`settings` List of main settings used in the call, including `reps`, `cluster_method`, `cluster_args`, `covariates`, `exclude_from_cluster`, `treat_singletons_as_excluded`, `boot_what`.

`data_info` List with information derived from the input data used for model setup: `mgm_type_level` (data frame with one row per variable, reporting the original R class and the inferred MGM type and level, as used in the call to `mgm::mgm`), and `binary_recode_map` (named list describing the mapping from original binary labels to the internal {0,1} coding used for model fitting).

`model` List with: `mgm` (the fitted mgm object), `nodes` (character vector of all node names), `n` (number of observations), `p` (number of variables), and `data` (if `save_data = TRUE`)

`layers` List describing the multilayer structure (assignment of nodes to layers, `layer_rules` matrix used and color of each layer in `palette`).

`layer_fits` Named list (one element per layer) with single layer fits, including community structure, node-level statistics, edge-level statistics, bridge metrics, and (optionally) community loadings with bootstrap information.

`interlayer` List collecting interlayer-only node metrics (strength, expected influence, closeness, betweenness, with or without bootstrap) and cross-layer edge summaries for each allowed pair of layers.

`graph` List containing a global **igraph** object built on the retained nodes (`keep_nodes_graph`), with vertex attributes such as `name`, `layer`, `membership`, and edge attributes such as `weight`, `abs_weight`, `sign`, `type` (intra vs inter) and `layer_pair`.

References

Haslbeck, J. M. B., & Waldorp, L. J. (2020). `mgm`: Estimating Time-Varying Mixed Graphical Models in High-Dimensional Data. *Journal of Statistical Software*, 93(8). doi:10.18637/jss.v093.i08

Examples

```

data(nhanes)

bio_vars <- c("ALT", "AST", "HDL", "HbA1c")
ant_vars <- c("BMI", "Waist", "ArmCirc", "LegLength")
life_vars <- c("Smoke", "PhysicalActivity", "Drug")
covs      <- c("Age", "Gender", "MonInc")

df <- nhanes[, c(bio_vars, ant_vars, life_vars, covs)]

# Layer assignment (must cover all columns except covariates)
layers <- c(
  setNames(rep("bio", length(bio_vars)), bio_vars),
  setNames(rep("ant", length(ant_vars)), ant_vars),
  setNames(rep("life", length(life_vars)), life_vars)
)

# Allow cross-layer edges bio<->ant and ant<->life; disallow bio<->life
layer_rules <- matrix(0, nrow = 3, ncol = 3,
  dimnames = list(c("bio", "ant", "life"),
    c("bio", "ant", "life")))

layer_rules["bio", "ant"] <- 1
layer_rules["ant", "life"] <- 1

fitM <- multimixMN(
  data = df,
  layers = layers,
  layer_rules = layer_rules,
  covariates = covs,
  lambdaSel = "EBIC",
  lambdaGam = 0.25,
  reps = 5,
  seed_model = 42,
  seed_boot = 42,
  compute_loadings = FALSE,
  progress = FALSE
)
fitM

# Plot the estimated network
set.seed(1)
plot(fitM, color_by = "layer")

```

Description

Example dataset derived to illustrate multilayer network estimation in MixMashNet. This dataset contains 29 variables derived from the National Health and Nutrition Examination Survey (NHANES)

Usage

```
data(nhanes)
```

Format

A data frame with 2759 rows and 29 variables:

TotChol Total Cholesterol (numeric).

HDL High-density lipoprotein cholesterol (numeric).

Creatinine Creatinine (numeric).

UricAcid Uric acid (numeric).

ALT Alanine aminotransferase (numeric).

AST Aspartate aminotransferase (numeric).

GGT Gamma-glutamyl transferase (numeric).

Bilirubin Bilirubin (numeric).

Albumin Albumin (numeric).

TotProtein Total protein (numeric).

HbA1c Glycated hemoglobin (numeric).

hsCRP High-sensitivity C-reactive protein (numeric).

BMI Body mass index (numeric).

Waist waist circumference (numeric).

Height Height (numeric).

ArmCirc Arm circumference (numeric).

HipCirc Hip circumference (numeric).

LegLength Leg length (numeric).

ArmLength Arm Length (numeric).

TroubleSleep Trouble Sleep with 0=no and 1=yes.

PhysicalActivity Physical Activity with 0=no and 1=yes.

Smoke Smoking with 0=no and 1=yes.

Drug Drug use with 0=no and 1=yes.

Diet Dietary quality with 1=Poor, 2=Fair, 3=Good, 4=Very good, 5=Excellent.

Alcohol Alcohol consumption with 0=no and 1=yes.

Work Employment status with 1=working, 2=employed but absent, 3=seeking work, 4=not working.

MonInc Monthly income category (1–12), where higher values indicate higher income.

Gender Gender with 0=male and 1=female.

Age Age (numeric).

References

Centers for Disease Control and Prevention (CDC) & National Center for Health Statistics (NCHS). (2020). National Health and Nutrition Examination Survey data. <https://www.cdc.gov/nchs/nhanes/>

<code>plot.mixMN_fit</code>	<i>Plot method for single layer MixMashNet objects</i>
-----------------------------	--

Description

Plotting interface for objects returned by `mixMN()`.

Depending on what, the method can:

- `what = "network"`: plot the estimated single-layer network;
- `what = "intra"`: plot node-level metrics or edge weights with bootstrap quantile regions at the level stored in the object;
- `what = "stability"`: plot node stability within communities based on bootstrap community assignments.

Usage

```
## S3 method for class 'mixMN_fit'
plot(x, what = c("network", "intra", "stability"), ...)
```

Arguments

<code>x</code>	An object of class "mixMN_fit", as returned by <code>mixMN()</code> .
<code>what</code>	Type of plot to produce. One of <code>c("network", "intra", "stability")</code> .
<code>...</code>	Additional arguments. Supported arguments depend on what: see the details below.

Details

Network plots (`what = "network"`): Supported arguments (via `...`):

`color_by` Node coloring: `c("community", "none")`.

`edge_color_by` Edge coloring: `c("sign", "none")`.

`edge_scale` Numeric scaling factor for edge widths (multiplied by `abs(weight)`).

`graphics::plot.igraph` **arguments** e.g., `vertex.size`, `vertex.label.cex`, `edge.width`, `vertex.label.color`, etc.

Within-network statistics (`what = "intra"`): Plots node-level metrics or edge weights with bootstrap quantile regions.

Supported arguments (via `...`):

statistics Character vector of metrics. Options include: "strength", "expected_influence", "closeness", "betweenness", bridge metrics "bridge_strength", "bridge_ei1", "bridge_ei2", "bridge_closeness", "bridge_betweenness", excluded bridge metrics "bridge_strength_excluded", "bridge_ei1_excluded", "bridge_ei2_excluded", "bridge_closeness_excluded", "bridge_betweenness_excluded" and "edges". Different metric families cannot be mixed in the same call (e.g., "edges" cannot be combined with node metrics).

ordering Node ordering: c("value", "alphabetical", "community").

standardize Logical; if TRUE, z-standardize the displayed values within the panel.

exclude_nodes Optional character vector of node names to remove before plotting.

color_by_community Logical; if TRUE, color nodes by community when available.

edges_top_n Integer; when statistics = "edges", keep the top edges by absolute weight.

title Optional plot title.

Community membership stability (what = "stability"): Plots node stability by community.

Supported arguments (via ...):

title Plot title. Default: "Node Stability by Community".

cutoff Optional numeric threshold in [0, 1] shown as a dashed vertical line. Use NULL to hide the line. Default: 0.7.

The quantile region level is taken from the fitted object (x\$settings\$quantile_level); if missing or invalid, a default of 0.95 is used.

Value

If what != "network", the function returns a ggplot object. If what = "network", the network is plotted directly.

See Also

[mixMN](#)

plot.multimixMN_fit *Plot method for multilayer MixMashNet objects*

Description

Plotting interface for objects returned by multimixMN().

Depending on what, the method can:

- what = "network": plot the estimated multilayer network, or a single layer if layer is specified;
- what = "intra": plot intralayer node-level metrics or edge weights with bootstrap quantile regions;
- what = "inter": plot interlayer node metrics or interlayer edge weights with bootstrap quantile regions;
- what = "stability": plot node stability within communities based on bootstrap community assignments.

Usage

```
## S3 method for class 'multimixMN_fit'
plot(x, what = c("network", "intra", "inter", "stability"), layer = NULL, ...)
```

Arguments

x An object of class "multimixMN_fit", as returned by multimixMN().

what Type of plot to produce. One of c("network", "intra", "inter", "stability").

layer Optional layer name. For what = "intra" or what = "stability", this selects which layer-specific fit to use. For what = "network", if provided, the selected layer is plotted as a single layer network.

... Additional arguments. Supported arguments depend on what: see the details below.

Details

Network plots (what = "network"): Supported arguments (via ...):

color_by Node coloring: c("layer", "community", "none").

edge_color_by Edge coloring: c("sign", "none").

edge_scale Numeric scaling factor for edge widths (multiplied by abs(weight)).

graphics::plot.igraph arguments e.g., vertex.size, vertex.label.cex, edge.width, vertex.label.color, etc.

Intralayer statistics (what = "intra"): Plots node-level metrics or edge weights with bootstrap quantile regions. If layer is provided, only that layer is plotted. If layer is NULL, all layers are plotted, one panel per layer.

Supported arguments (via ...):

statistics Character vector of metrics. Options include: "strength", "expected_influence", "closeness", "betweenness", bridge metrics "bridge_strength", "bridge_ei1", "bridge_ei2", "bridge_closeness", "bridge_betweenness", excluded bridge metrics "bridge_strength_excluded", "bridge_ei1_excluded", "bridge_ei2_excluded", "bridge_closeness_excluded", "bridge_betweenness_excluded" and "edges". Different metric families cannot be mixed in the same call (e.g., "edges" cannot be combined with node metrics).

ordering Node ordering: c("value", "alphabetical", "community").

standardize Logical; if TRUE, z-standardize the displayed values within each panel.

exclude_nodes Optional character vector of node names to remove before plotting.

color_by_community Logical; if TRUE, color nodes by community when available.

edges_top_n Integer; when statistics = "edges", keep the top edges by absolute weight.

title Optional plot title. If omitted and multiple layers are shown, layer-specific titles are added automatically.

Interlayer summaries (what = "inter"): Plots interlayer node metrics or interlayer edge weights with bootstrap quantile regions.

Supported arguments (via ...):

statistics Character vector. Node metrics: c("strength", "expected_influence", "closeness", "betweenness"), or "edges" for interlayer edge weights. Node metrics and "edges" cannot be combined.

pairs Layer pairs to show. Either "*" (all available) or a character vector of pair keys like "bio_dis" (order-insensitive).

edges_top_n Integer; keep the top interlayer edges by absolute weight.

ordering Ordering within panels: c("value", "alphabetical").

standardize Logical; if TRUE, z-standardize values (node metrics by metric, edges by pair).

exclude_nodes Optional character vector; removes nodes and incident interlayer edges.

nodes_layer Optional layer name to restrict node metrics to nodes belonging to that layer.

title Optional plot title.

Community membership stability (what = "stability"): Plots node stability by community. If layer is provided, only that layer is shown. Otherwise, stability plots are shown for all layers.

Supported arguments (via ...):

title Plot title. Default: "Node Stability by Community".

cutoff Optional numeric threshold in [0, 1] shown as a dashed vertical line. Use NULL to hide the line. Default: 0.7.

The quantile region level is taken from the fitted object (x\$settings\$quantile_level); if missing or invalid, a default of 0.95 is used.

Value

If what != "network", the function returns a ggplot object. If what = "network", the network is plotted directly.

See Also

[multimixMN](#)

print.mixMN_fit *Print method for single layer MixMashNet objects*

Description

Compact textual summary for objects returned by mixMN().

Usage

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

Arguments

x An object of class "mixMN_fit".

... Additional arguments.

Value

The input object `x`, returned invisibly.

See Also

[mixMN](#)

`print.multimixMN_fit` *Print method for multilayer MixMashNet objects*

Description

Compact textual summary for objects returned by `multimixMN()`.

Usage

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

Arguments

`x` An object of class "multimixMN_fit".
`...` Additional arguments.

Value

The input object `x`, returned invisibly.

See Also

[multimixMN](#)

`summary.mixMN_fit` *Summarize top intralayer edges for single-layer MixMashNet fits*

Description

Returns the top 10 intralayer edges for fitted objects returned by `mixMN()`, in the same long-format structure used for edge summaries.

Usage

```
## S3 method for class 'mixMN_fit'
summary(object, top_n = 10L, digits = NULL, drop_na_boot = TRUE, ...)
```

Arguments

object	An object of class "mixMN_fit".
top_n	Number of top edges to retain. Default is 10.
digits	Optional number of digits used to round numeric columns.
drop_na_boot	Logical. If TRUE (default), bootstrap-related columns that are entirely NA are removed.
...	Further arguments for S3 compatibility.

Value

An object of class "summary.mixMN_fit" containing the top intralayer edges.

```
summary.multimixMN_fit
```

Summarize top interlayer edges for multilayer MixMashNet fits

Description

Returns the top 10 interlayer edges for fitted objects returned by multimixMN(), in the same long-format structure used for edge summaries.

Usage

```
## S3 method for class 'multimixMN_fit'
summary(object, top_n = 10L, digits = NULL, drop_na_boot = TRUE, ...)
```

Arguments

object	An object of class "multimixMN_fit".
top_n	Number of top edges to retain. Default is 10.
digits	Optional number of digits used to round numeric columns.
drop_na_boot	Logical. If TRUE (default), bootstrap-related columns that are entirely NA are removed.
...	Further arguments for S3 compatibility.

Value

An object of class "summary.multimixMN_fit" containing the top interlayer edges.

update_palette	<i>Update community and layer color palettes in MixMashNet objects</i>
----------------	--

Description

Updates the color palettes associated with communities and/or layers in fitted `mixMN_fit` and `multimixMN_fit` objects.

For `mixMN_fit` objects, `community_colors` must be a named character vector specifying colors for community labels in `object$communities$palette`.

For `multimixMN_fit` objects, `community_colors` must be a named list whose elements correspond to layer names. Each element must be a named character vector specifying colors for the community labels of that layer. The list may be partial, so only the specified layers are updated.

For `multimixMN_fit` objects, `layer_colors` updates the palette stored in `object$layers$palette`.

The function replaces only the colors corresponding to the provided names, leaving all other colors unchanged. Unknown layer names, community labels, or layer labels are ignored with a warning.

Usage

```
update_palette(object, ...)
```

```
## S3 method for class 'mixMN_fit'
```

```
update_palette(object, community_colors = NULL, layer_colors = NULL, ...)
```

```
## S3 method for class 'multimixMN_fit'
```

```
update_palette(object, community_colors = NULL, layer_colors = NULL, ...)
```

Arguments

`object` An object of class `mixMN_fit` or `multimixMN_fit`.

`...` Further arguments passed to methods.

`community_colors`

For `mixMN_fit` objects, an optional named character vector specifying new colors for communities.

For `multimixMN_fit` objects, an optional named list whose names are layer names and whose elements are named character vectors specifying new colors for communities within each layer.

`layer_colors` Optional named character vector specifying new colors for layers. Only applicable to `multimixMN_fit` objects.

Details

For single layer fits, only `community_colors` is used.

For multilayer fits:

- `community_colors` updates community palettes within the specified layers;

- layer_colors updates the palette of the layers themselves.

For multilayer fits, community_colors can be partial: layers not included in the list are left unchanged.

Value

The input object, with updated community and/or layer palettes.

Examples

```
data(bacteremia)

vars <- c("WBC", "NEU", "HGB", "PLT", "CRP")
df <- bacteremia[, vars]

fit <- mixMN(
  data = df,
  lambdaSel = "EBIC",
  reps = 0,
  seed_model = 42,
  compute_loadings = FALSE,
  progress = FALSE
)

fit$communities$palette

fit2 <- update_palette(
  fit,
  community_colors = c("1" = "red", "2" = "blue")
)

fit2$communities$palette

set.seed(1)
plot(fit2)
```

Index

* package

MixMashNet-package, 2

bacteremia, 3

bridge_metrics, 4

bridge_metrics_excluded, 5

community_scores, 6

find_bridge_communities, 8

find_bridge_layers, 9

get_centrality, 10

get_edges, 12

layer_slice, 14

membershipStab, 14

MixMashNet (MixMashNet-package), 2

MixMashNet-package, 2

mixMN, 15, 27, 30

multimixMN, 20, 29, 30

nhanes, 24

plot.mixMN_fit, 26

plot.multimixMN_fit, 27

print.mixMN_fit, 29

print.multimixMN_fit, 30

summary.mixMN_fit, 30

summary.multimixMN_fit, 31

update_palette, 32