Package ‘MixMashNet’

March 3, 2026

Title Tools for Multilayer and Single Layer Network Modeling

Version 0.6.0

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>).

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,
tidyr, 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]

Repository CRAN

Date/Publication 2026-03-03 10:40:09 UTC

Contents

`MixMashNet-package` *Tools for Multilayer and Single Layer Network Modeling*

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). MixMashNet: 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 dataset*

Description

Example dataset used in MixMashNet examples. 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

`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`).

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. (2025). **networktools**: Tools for identifying important nodes in networks. R package version 1.6.1. `https://github.com/paytonjjones/networktools`

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. 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("mixmashnet","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 layers and returned as a named list.
`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.

Value

A list with class `c("mixmashnet","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 `c("mixmashnet","community_scores")` object. If `fit` is a `multimixMN_fit` and `layer = NULL`, returns a named list of `community_scores` objects (one per layer).

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.

Usage

```
find_bridge_communities(fit, node)
```

Arguments

`fit`	An object of class `mixMN_fit`.
`node`	Character scalar: node of interest; must belong to `fit$graph$keep_nodes_graph`.

Details

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

Notes:

- 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.

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`).

`membershipStab` *Node stability from bootstrap community assignments*

Description

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

Usage

```
membershipStab(fit, IS.plot = FALSE)
```

Arguments

`fit`	An object returned by `mixMN()` (class `mixMN_fit`), containing `$communities$original_membership` and `$communities$boot_memberships`. Bootstrap memberships must be available, i.e. `reps` > `0` and `"community"` `%in%` `boot_what`.
`IS.plot`	Logical; if `TRUE`, prints a stability plot via the internal helper `membershipStab_plot()`.

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.

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 (`reps` × `p`)

`membership.stability` List with:

- `empirical.dimensions` Named numeric vector of node-level stability (proportion assigned to empirical community)
- `all.dimensions` Matrix (`p` × `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", "fast_greedy", "infomap", "walktrap", "edge_betweenness"),
  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 sup-

	ported 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 (`type == "g"`) are z-standardized internally by `mgm()`. Use `scale = FALSE` if your data are already standardized.
`lambdaSel`	Method for lambda selection: `"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. 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 algorithm used on the network: `"louvain"`, `"fast_greedy"`, `"infomap"`, `"walktrap"`, or `"edge_betweenness"`.
`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 `c("mixmashnet", "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`, `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 graphicalmodels: 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,
  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,
  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", "fast_greedy", "infomap", "walktrap", "edge_betweenness"),
  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 × 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 algorithm applied within each layer. One of `"louvain"`, `"fast_greedy"`, `"infomap"`, `"walktrap"`, or `"edge_betweenness"`.
`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 `c("mixmashnet", "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`, `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")
```

---

nhanes *NHANES dataset*

Description

Example dataset used in MixMashNet examples. 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/

`nhgh_data` *NHGH dataset*

Description

Example dataset used in MixMashNet examples. This dataset contains 15 variables derived from the National Health and Nutrition Examination Survey (NHANES)

Usage

```
data(nhgh_data)
```

Format

A data frame with 5621 rows and 15 variables:

wt Weight (numeric).

ht Height (numeric).

bmi Body mass index (numeric).

leg Leg length (numeric).

arml Arm length (numeric).

armc Arm circumference (numeric).

tri Triceps skinfold (numeric).

sub Subscapular skinfold (numeric).

gh Glychohemoglobin (numeric).

albumin Albumin (numeric).

bun Blood urea nitrogen (numeric).

SCr Serum creatinine (numeric).

age Age (numeric).

sex Sex with 0=woman and 1=man.

re Race with 1=Mexican American, 2=Other Hispanic, 3=Non-Hispanic White, 4=Non-Hispanic Black, 5=Other Race Including Multi-Racial.

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/

plot.mixmashnet *Plot method for MixMashNet objects*

Description

Unified plotting interface for objects returned by `mixMN()` and `multimixMN()`. Depending on `what`, it can:

- `what = "network"`: plot the estimated network (single layer or multilayer);
- `what = "intra"`: plot intralayer node/edge statistics with bootstrap quantile regions at the level stored in the object (centrality and bridge metrics);
- `what = "inter"`: plot interlayer node metrics or interlayer edge weights with bootstrap quantile regions at the level stored in the object (multilayer only), and the chosen `statistics`;
- `what = "stability"`: plot node stability within communities based on bootstrap community assignments.

Usage

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

Arguments

`x`	An object of class `mixmashnet`, as returned by `mixMN()` or `multimixMN()`.
`what`	Type of plot to produce. One of `c("network","intra","inter","stability")`.
`layer`	Optional layer name. For `what = "intra"` or `what = "stability"` on a `multimixMN_fit` object, this selects which layer-specific fit to use.
`...`	Additional arguments. Supported arguments depend on `what`: see the details below.

Details

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

`color_by` Node coloring. Single layer: `c("community","none")`. Multilayer: `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. For multilayer objects:

- if `layer` is provided, plots that layer only;
- if `layer` is `NULL`, plots all layers (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_exc` and `"edges"`. Note: 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. In multilayer mode, if not provided a layer-specific title is added automatically.

Interlayer summaries (`what = "inter"`**; multilayer only):** 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. For multilayer objects, `layer` selects a specific layer; if `layer` is `NULL`, 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.

`print.mixmashnet` *Print method for MixMashNet objects*

Description

Compact textual summary for objects returned by `mixMN()` and `multimixMN()`. The method reports:

- whether the fit is single layer (`mixMN`) or multilayer (`multimixMN`);
- number of subjects (if available) and variables;
- for multilayer fits, the number of nodes and non-zero edges per layer and, if present, per interlayer pair;
- size of the global graph (nodes and edges);
- number of communities (single layer) or communities per layer (multilayer);
- covariates used for adjustment and nodes excluded from the graph and/or clustering;
- main settings for community detection and bootstrap;
- data info.

Usage

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

Arguments

`x`	An object of class `mixmashnet`, returned by `mixMN()` or `multimixMN()`.
`...`	Additional arguments.

Value

The input object `x`, returned invisibly.

See Also

`mixMN`, `multimixMN`

summary.mixmashnet *Summarize MixMashNet fits (single and multilayer) in long format*

Description

Summarizes fitted MixMashNet objects (single and multilayer). The summary includes the original estimates and, when available, bootstrap means, standard errors, and quantile regions.

Usage

```
## S3 method for class 'mixmashnet'
summary(
  object,
  what = c("intra", "inter"),
  statistics = NULL,
  layer = NULL,
  pairs = NULL,
  digits = 3,
  ...
)
```

Arguments

`object` An object of class `"mixmashnet"` returned by `mixMN()` or `multimixMN()`.

`what` Character string indicating which part of the model to summarize:

- `"intra"`: intralayer quantities (node-level indices and/or intralayer edges);
- `"inter"`: interlayer quantities (node-level indices on the interlayer-only graph and/or cross-layer edges; multilayer fits only).

`statistics` Character vector specifying which statistics to include. For `what = "intra"`, valid values are: `c("edges", "strength", "expected_influence", "closeness", "betweenness", "bridge_strength", "bridge_closeness", "bridge_betweenness", "bridge_ei1", "bridge_ei2", "bridge_strength_excluded", "bridge_betweenness_excluded" "bridge_closeness_excluded", "bridge_ei1_excluded", "bridge_ei2_excluded")`.

For `what = "inter"`, valid values are: `c("edges", "strength", "expected_influence", "closeness", "betweenness")`.

If `statistics = NULL`, then:

- for `what = "intra"`, all available intralayer statistics (including `"edges"`) are returned;
- for `what = "inter"`, all available interlayer statistics (including `"edges"`) are returned.

`layer` Optional character vector of layer names to subset. Used for `what = "intra"` in multilayer fits. Ignored for single layer fits.

`pairs` Optional character vector of layer-pair names (e.g. `"bio_dis"`) used for `what = "inter"` when summarizing interlayer edges. If `NULL`, all available layer pairs are included. For interlayer node indices, `pairs` can be used to restrict the summary to nodes belonging to one of the layers in the given pair.

`digits`	Number of digits to round numeric summaries.
`...`	Not used (for S3 compatibility).

Value

A list (class `"summary.mixmashnet"`) with up to four data frames (`$index`, `$edges`, `$interlayer_index`, `$interlayer_edges`) and the quantile region used to compute the bootstrap quantile regions (`$quantile_level`).

Depending on `what` and `statistics`, some of these elements may be `NULL`.

`update_palette` *Update community and layer color palettes in MixMashNet objects*

Description

Updates the color palettes associated with communities and/or layers in `mixMN_fit` and `multimixMN_fit` objects. The function replaces only the colors corresponding to the provided names, leaving all other colors unchanged.

If colors are provided for names that do not exist in the object (e.g., unknown community labels or layer names), a warning is issued and those entries are ignored. If some communities or layers are not specified, their original colors are preserved.

Usage

```
update_palette(fit, community_colors = NULL, layer_colors = NULL)
```

Arguments

`fit`	An object of class `mixMN_fit` or `multimixMN_fit`.
`community_colors`	Optional named character vector specifying new colors for communities. Names must correspond to existing community labels (as stored in `communities$palette`). Missing names are ignored.
`layer_colors`	Optional named character vector specifying new colors for layers. Names must correspond to existing layer names (as stored in `layers$palette`). Only applicable to `multimixMN_fit` objects.

Details

For `mixMN_fit` objects, community colors are updated in `fit$communities$palette`.

For `multimixMN_fit` objects, community colors are updated separately within each layer (i.e., in `fit$layer_fits[[L]]$communities$palette`), while layer colors are updated in `fit$layers$palette`.

The function performs in-place modification of the palettes and returns the updated object.

Value

The input object `fit`, 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
)

# View original community palette
fit$communities$palette

# Update colors for communities 1 and 2
fit2 <- update_palette(
  fit,
  community_colors = c("1" = "red", "2" = "blue")
)

fit2$communities$palette

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

Index