Package 'tna'

March 1, 2025

Title Transition Network Analysis (TNA)

Version 0.4.0

Description Provides tools for performing Transition Network Analysis (TNA) to study relational dynamics, including functions for building and plotting TNA models, calculating centrality measures, and identifying dominant events and patterns. TNA statistical techniques (e.g., bootstrapping and permutation tests) ensure the reliability of observed insights and confirm that identified dynamics are meaningful. See (Saqr et al., 2025) <doi:10.1145/3706468.3706513> for more details on TNA.

License MIT + file LICENSE

URL https://github.com/sonsoleslp/tna/, http://sonsoles.me/tna/

BugReports https://github.com/sonsoleslp/tna/issues/

Depends R (>= 4.1.0)

Imports checkmate, cli, colorspace, dplyr, ggplot2, graphics, igraph, qgraph, RColorBrewer, rlang, stats, tibble, tidyr

Suggests gt, knitr, rmarkdown, seqHMM, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

LazyData true

NeedsCompilation no

Author Mohammed Saqr [aut],
Santtu Tikka [aut],
Sonsoles López-Pernas [aut, cre]

Maintainer Sonsoles López-Pernas <sonsoles.lopez@uef.fi>

Repository CRAN

Date/Publication 2025-03-01 16:30:02 UTC

Contents

tna-package . . . 3
as.igraph.group_tna . . . 4
as.igraph.matrix . . . 5
as.igraph.tna . . . 5
betweenness_network . . . 6
bootstrap . . . 6
build_model . . . 9
centralities . . . 12
cliques . . . 14
communities . . . 15
compare . . . 17
compare.group_tna . . . 19
deprune . . . 19
engagement . . . 20
engagement_mmm . . . 21
estimate_cs . . . 22
group_model . . . 25
group_regulation . . . 28
hist.group_tna . . . 29
hist.tna . . . 30
mmm_stats . . . 31
permutation_test . . . 32
plot.group_tna . . . 33
plot.group_tna_centralities . . . 34
plot.group_tna_cliques . . . 35
plot.group_tna_communities . . . 36
plot.group_tna_stability . . . 37
plot.tna . . . 38
plot.tna_centralities . . . 39
plot.tna_cliques . . . 40
plot.tna_communities . . . 42
plot.tna_comparison . . . 43
plot.tna_permutation . . . 44
plot.tna_stability . . . 45
plot_compare . . . 46
plot_compare.group_tna . . . 47
plot_mosaic . . . 48
plot_mosaic.group_tna . . . 48
plot_mosaic.tna_data . . . 49
prepare_data . . . 50
print.group_tna . . . 53
print.group_tna_bootstrap . . . 54
print.group_tna_centralities . . . 55
print.group_tna_cliques . . . 56
print.group_tna_communities . . . 57
print.group_tna_stability . . . 58

print.summary.group_tna . . . 59
print.summary.group_tna_bootstrap . . . 60
print.summary.tna . . . 61
print.summary.tna_bootstrap . . . 61
print.tna . . . 62
print.tna_bootstrap . . . 63
print.tna_centralities . . . 63
print.tna_cliques . . . 64
print.tna_communities . . . 65
print.tna_comparison . . . 65
print.tna_data . . . 66
print.tna_permutation . . . 67
print.tna_stability . . . 67
prune . . . 68
pruning_details . . . 70
rename_groups . . . 71
reprune . . . 71
summary.group_tna . . . 72
summary.group_tna_bootstrap . . . 74
summary.tna . . . 75
summary.tna_bootstrap . . . 76

Index **78**

`tna-package` *The* `tna` *package.*

Description

Provides tools for performing transition network analysis (TNA), including functions for building TNA models, plotting transition networks, and calculating centrality measures. The package relies on the `qgraph` and `igraph` for network plotting and centrality measure calculations.

Author(s)

Sonsoles López-Pernas, Santtu Tikka, Mohammed Saqr

References

Saqr M., López-Pernas S., Törmänen T., Kaliisa R., Misiejuk K., Tikka S. (2025). Transition Network Analysis: A Novel Framework for Modeling, Visualizing, and Identifying the Temporal Patterns of Learners and Learning Processes. In *Proceedings of the 15th International Learning Analytics and Knowledge Conference (LAK '25)*, 351-361.

Banerjee A., Chandrasekhar A., Duflo E., Jackson M. (2014). Gossip: Identifying Central Individuals in a Social Network. Working Paper.

Kivimaki, I., Lebichot, B., Saramaki, J., Saerens, M. (2016). Two betweenness centrality measures based on Randomized Shortest Paths. *Scientific Reports*, 6, 19668.

Serrano, M. A., Boguna, M., Vespignani, A. (2009). Extracting the multiscale backbone of complex weighted networks. *Proceedings of the National Academy of Sciences*, 106, 6483-6488.

Zhang, B., Horvath, S. (2005). A general framework for weighted gene co-expression network analysis. *Statistical Applications in Genetics and Molecular Biology*, 4(1).

See Also

Useful links:

- https://github.com/sonsoleslp/tna/
- http://sonsoles.me/tna/
- Report bugs at https://github.com/sonsoleslp/tna/issues/

`as.igraph.group_tna` *Coerce a specific group from a* `group_tna` *object to an* `igraph` *object.*

Description

Coerce a specific group from a `group_tna` object to an `igraph` object.

Usage

```
## S3 method for class 'group_tna'
as.igraph(x, which, ...)
```

Arguments

`x`	The object to convert.
`which`	The number or name of the group.
`...`	Additional arguments. None currently.

Value

An `igraph` object.

as.igraph.matrix *Coerce a weight matrix to an* igraph *object.*

Description

Coerce a weight matrix to an igraph object.

Usage

```
## S3 method for class 'matrix'
as.igraph(x, directed = TRUE, ...)
```

Arguments

x	The object to convert.
directed	A logical value. If TRUE, assumes that the graph is directed and undirected otherwise.
...	Additional arguments. None currently.

Value

An igraph object.

as.igraph.tna *Coerce a* tna *object to an* igraph *object.*

Description

Coerce a tna object to an igraph object.

Usage

```
## S3 method for class 'tna'
as.igraph(x, ...)
```

Arguments

x	The object to convert.
...	Additional arguments. None currently.

Value

An igraph object.

betweenness_network *Build and Visualize a Network with Edge Betweenness*

Description

This function builds a network from a transition matrix in a tna object and computes edge betweenness for the network.

Usage

```
betweenness_network(x, ...)

## S3 method for class 'tna'
betweenness_network(x, ...)
```

Arguments

x	A tna object.
...	Ignored.

Value

A tna object where the edge weights are edge betweenness values.

Examples

```
model <- tna(group_regulation)
betweenness_network(model)
```

bootstrap *Bootstrap Transition Networks from Sequence Data*

Description

Perform bootstrapping on transition networks created from sequence data stored in a tna object. Bootstrapped estimates of edge weights are returned with confidence intervals and significance testing.

Usage

```
bootstrap(x, ...)

## S3 method for class 'tna'
bootstrap(
  x,
  iter = 1000,
  level = 0.05,
  method = "stability",
  threshold,
  consistency_range = c(0.75, 1.25),
  ...
)

## S3 method for class 'group_tna'
bootstrap(x, ...)
```

Arguments

x	A tna or a group_tna object created from sequence data.
...	Ignored.
iter	An integer specifying the number of bootstrap samples to draw. Defaults to 1000.
level	A numeric value representing the significance level for hypothesis testing and confidence intervals. Defaults to 0.05.
method	A character string. This argument defines the bootstrap test statistic. The "stability" option (the default) compares edge weights against a range of "consistent" values defined by consistency_range. Weights that fall outside this range are considered insignificant. In other words, an edge is considered significant if its value is within the range in (1 - level) * 100% of the bootstrap samples. The "threshold" option instead compares the edge weights against a user-specified threshold value.
threshold	A numeric value to compare edge weights against. The default is the 10th percentile of the edge weights. Used only when method = "threshold".
consistency_range	A numeric vector of length 2. Determines how much the edge weights may deviate (multiplicatively) from their observed values (below and above) before they are considered insignificant. The default is c(0.75, 1.25) which corresponds to a symmetric 25% deviation range. Used only when method = "stability".

Details

The function first computes the original edge weights for the specified cluster from the tna object. It then performs bootstrapping by resampling the sequence data and recalculating the edge weights for each bootstrap sample. The mean and standard deviation of the transitions are computed, and confidence intervals are derived. The function also calculates p-values for each edge and identifies

significant edges based on the specified significance level. A matrix of significant edges (those with p-values below the significance level) is generated. Additional statistics on removed edges (those not considered significant) are provided.

All results, including the original transition matrix, bootstrapped estimates, and summary statistics for removed edges, are returned in a structured list.

Value

A `tna_bootstrap` object which is a `list` containing the following elements:

- `weights_orig`: The original edge weight `matrix`.
- `weights_sig`: The `matrix` of significant transitions (those with p-values below the significance level).
- `weights_mean`: The mean weight `matrix` from the bootstrap samples.
- `weights_sd`: The standard deviation `matrix` from the bootstrap samples.
- `ci_lower`: The lower bound `matrix` of the confidence intervals for the edge weights.
- `ci_upper`: The upper bound `matrix` of the confidence intervals for the edge weights.
- `p_values`: The `matrix` of p-values for the edge weights.
- `summary`: A `data.frame` summarizing the edges, their weights, p-values, statistical significance and confidence intervals.

If `x` is a `group_tna` object, the output is a `group_tna_bootstrap` object, which is a `list` of `tna_bootstrap` objects.

See Also

Evaluation and validation functions `permutation_test()`, `prune()`, `pruning_details()`

Cluster-related functions `centralities()`, `cliques()`, `communities()`, `deprune()`, `estimate_cs()`, `group_model()`, `hist.group_tna()`, `mmm_stats()`, `plot.group_tna()`, `plot.group_tna_centralities()`, `plot.group_tna_cliques()`, `plot.group_tna_communities()`, `plot.group_tna_stability()`, `plot_compare.group_tna()`, `plot_mosaic.group_tna()`, `plot_mosaic.tna_data()`, `print.group_tna()`, `print.group_tna_bootstrap()`, `print.group_tna_centralities()`, `print.group_tna_cliques()`, `print.group_tna_communities()`, `print.group_tna_stability()`, `print.summary.group_tna()`, `print.summary.group_tna_bootstrap()`, `prune()`, `pruning_details()`, `rename_groups()`, `reprune()`, `summary.group_tna()`, `summary.group_tna_bootstrap()`

Examples

```
model <- tna(group_regulation)
# Small number of iterations for CRAN
bootstrap(model, iter = 10)
```

build_model *Build a Transition Network Analysis Model*

Description

Construct a transition network analysis (TNA) model from sequence data. The function takes a data set of sequence of events or states as input and builds a TNA model. It extracts the edge weights and initial probabilities from the data along with the state labels. THe function also accepts weight matrices and initial state probabilities directly.

Usage

```
build_model(x, type = "relative", scaling = character(0L), ...)

## Default S3 method:
build_model(
  x,
  type = "relative",
  scaling = character(0L),
  inits,
  params = list(),
  ...
)

## S3 method for class 'matrix'
build_model(x, type = "relative", scaling = character(0L), inits, ...)

## S3 method for class 'stslist'
build_model(
  x,
  type = "relative",
  scaling = character(0L),
  cols = seq(1, ncol(x)),
  params = list(),
  ...
)

## S3 method for class 'data.frame'
build_model(
  x,
  type = "relative",
  scaling = character(0L),
  cols = seq(1, ncol(x)),
  params = list(),
  ...
)
```

```
## S3 method for class 'tna_data'
build_model(x, type = "relative", scaling = character(0), params = list(), ...)

tna(x, ...)

ftna(x, ...)

ctna(x, ...)
```

Arguments

x	A stslist (from TraMineR), data.frame, a matrix, or a tna_data object (see prepare_data()). For stslist and data.frame objects x should describe a sequence of events or states to be used for building the Markov model. If x is a matrix, it is assumed that the element on row i and column j is the weight of the edge representing the transition from state i to state j. If x is a data.frame, then it must be in wide format (see cols on how to define columns for the time points).
type	A character string describing the weight matrix type. Currently supports the following types: • "relative" for relative frequencies (probabilities, the default) • "frequency" for frequencies. • "co-occurrence" for co-occurrences. • "n-gram" for n-gram transitions. Captures higher-order transitions by considering sequences of n states, useful for identifying longer patterns. • "gap" allows transitions between non-adjacent states, with transitions weighted by the gap size. • "window" creates transitions between all states within a sliding window, capturing local relationships (several sequences together). • "reverse" considers the sequences in reverse order (resulting in what is called a reply network in some contexts). The resulting weight matrix is the transpose of the "frequency" option.
scaling	A character vector describing how to scale the weights defined by type. When a vector is provided, the scaling options are applied in the respective order. For example, c("rank", "minmax") would first compute the ranks, then scale them to the unit interval using min-max normalization. An empty vector corresponds to no scaling. Currently supports the following options: • "minmax" performs min-max normalization to scale the weights to the unit interval. Note that if the smallest weight is positive, it will be zero after scaling. • "max" Multiplies the weights by the reciprocal of the largest weight to scale the weights to the unit interval. This options preserves positive ranks, unlike "minmax" when all weights are positive. • "rank" Computes the ranks of the weights using base::rank() with ties.method = "average".
...	Ignored.

inits — An optional numeric vector of initial state probabilities for each state. Can be provided only if x is a matrix. The vector will be scaled to unity.

params — A list of additional arguments for models of specific type. The potential elements of this list are:

- n_gram: An integer for n-gram transitions specifying the number of adjacent events. The default value is 2.
- max_gap: An integer for the gap-allowed transitions specifying the largest allowed gap size. The default is 1.
- window_size: An integer for the sliding window transitions specifying the window size. The default is 2.
- weighted: A logical value. If TRUE, the transitions are weighted by the inverse of the sequence length. Can be used for frequency, co-occurrence and reverse model types. The default is FALSE.

cols — An integer/character vector giving the indices/names of the columns that should be considered as sequence data. Defaults to all columns, i.e., seq(1, ncol(x)).

Value

An object of class tna which is a list containing the following elements:

- weights: An adjacency matrix of the model (weight matrix).
- inits: A numeric vector of initial values for each state. For matrix type x, this element will be NULL if inits is not directly provided
- labels: A character vector of the state labels, or NULL if there are no labels.
- data: The original sequence data that has been converted to an internal format used by the package when x is a stslist or a data.frame object. Otherwise NULL.

An object of class tna which is a list containing the following elements:

- weights: An adjacency matrix of the model (weight matrix).
- inits: A numeric vector of initial values for each state. For matrix type x, this element will be NULL if inits is not directly provided
- labels: A character vector of the state labels, or NULL if there are no labels.
- data: The original sequence data that has been converted to an internal format used by the package when x is a stslist or a data.frame object. Otherwise NULL.

An object of class tna which is a list containing the following elements:

- weights: An adjacency matrix of the model (weight matrix).
- inits: A numeric vector of initial values for each state. For matrix type x, this element will be NULL if inits is not directly provided
- labels: A character vector of the state labels, or NULL if there are no labels.
- data: The original sequence data that has been converted to an internal format used by the package when x is a stslist or a data.frame object. Otherwise NULL.

An object of class tna which is a list containing the following elements:

- `weights`: An adjacency `matrix` of the model (weight matrix).
- `inits`: A `numeric` vector of initial values for each state. For `matrix` type `x`, this element will be `NULL` if `inits` is not directly provided
- `labels`: A `character` vector of the state labels, or `NULL` if there are no labels.
- `data`: The original sequence data that has been converted to an internal format used by the package when `x` is a `stslist` or a `data.frame` object. Otherwise `NULL`.

See Also

Core functions `centralities()`, `plot.tna()`, `plot.tna_centralities()`, `plot_compare()`

Examples

```
model <- build_model(group_regulation)
print(model)

model <- tna(group_regulation)

model <- ftna(group_regulation)

model <- ctna(group_regulation)
```

`centralities` *Calculate Centrality Measures for a Transition Matrix*

Description

Calculates several centrality measures. See 'Details' for information about the measures.

Usage

```
centralities(x, loops = FALSE, normalize = FALSE, measures, ...)

## S3 method for class 'tna'
centralities(x, loops = FALSE, normalize = FALSE, measures, ...)

## S3 method for class 'matrix'
centralities(x, loops = FALSE, normalize = FALSE, measures, ...)

## S3 method for class 'group_tna'
centralities(x, loops = FALSE, normalize = FALSE, measures, ...)
```

Arguments

x	A tna object, a group_tna object, or a square matrix representing edge weights.
loops	A logical value indicating whether to include loops in the network when computing the centrality measures (default is FALSE).
normalize	A logical value indicating whether the centralities should be normalized (default is FALSE).
measures	A character vector indicating which centrality measures should be computed. If missing, all available measures are returned. See 'Details' for available measures. The elements are partially matched ignoring case.
...	Ignored.

Details

The following measures are provided:

- OutStrength: Outgoing strength centrality, calculated using igraph::strength() with mode = "out". It measures the total weight of the outgoing edges from each node.
- InStrength: Incoming strength centrality, calculated using igraph::strength() with mode = "in". It measures the total weight of the incoming edges to each node.
- ClosenessIn: Closeness centrality (incoming), calculated using igraph::closeness() with mode = "in". It measures how close a node is to all other nodes based on the incoming paths.
- ClosenessOut: Closeness centrality (outgoing), calculated using igraph::closeness() with mode = "out". It measures how close a node is to all other nodes based on the outgoing paths.
- Closeness: Closeness centrality (overall), calculated using igraph::closeness() with mode = "all". It measures how close a node is to all other nodes based on both incoming and outgoing paths.
- Betweenness: Betweenness centrality defined by the number of geodesics calculated using igraph::betweenness().
- BetweennessRSP: Betweenness centrality based on randomized shortest paths (Kivimäki et al. 2016). It measures the extent to which a node lies on the shortest paths between other nodes.
- Diffusion: Diffusion centrality of Banerjee et.al. (2014). It measures the influence of a node in spreading information through the network.
- Clustering: Signed clustering coefficient of Zhang and Horvath (2005) based on the symmetric adjacency matrix (sum of the adjacency matrix and its transpose). It measures the degree to which nodes tend to cluster together.

Value

A tna_centralities object which is a tibble (tbl_df). containing centrality measures for each state.

See Also

Core functions build_model(), plot.tna(), plot.tna_centralities(), plot_compare()

Cluster-related functions bootstrap(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralities(), plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Examples

```
model <- tna(group_regulation)

# Centrality measures including loops in the network
centralities(model)

# Centrality measures excluding loops in the network
centralities(model, loops = FALSE)

# Centrality measures normalized
centralities(model, normalize = TRUE)
```

---

cliques *Identify Cliques in a Transition Network*

Description

This function identifies cliques of a specified size in a transition network. It searches for cliques—complete subgraphs where every pair of nodes is connected—of size n in the transition matrix for the specified cluster in the tna object.

Usage

```
cliques(x, ...)

## S3 method for class 'tna'
cliques(x, size = 2, threshold = 0, sum_weights = FALSE, ...)

## S3 method for class 'group_tna'
cliques(x, size = 2, threshold = 0, sum_weights = FALSE, ...)
```

Arguments

x	A tna or a group_tna object.
...	Ignored.
size	An integer specifying the size of the cliques to identify. Defaults to 2 (dyads).
threshold	A numeric value that sets the minimum edge weight for an edge to be considered in the clique. Edges below this value are ignored. Defaults to 0.
sum_weights	A logical value specifying whether the sum of the weights should be above the threshold instead of individual weights of the directed edges. Defaults to FALSE.

Value

A tna_cliques object which is a list of two elements:

- weights is a matrix of the edge weights in the clique.
- inits is a numeric vector of initial weights for the clique.

If x is a group_tna object, a group_tna_cliques object is returned instead, which is a list or tna_cliques objects.

See Also

Cluster-related functions bootstrap(), centralities(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralities(), plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Examples

```
model <- tna(group_regulation)

# Find  2-cliques (dyads)
cliq <- cliques(model, size = 2)
```

---

communities *Community Detection for Transition Networks*

Description

This function detects communities within the transition networks (represented by the `tna` object). It uses various algorithms to find communities in the graph representation of transitions and returns a `list` of communities for each cluster or a specified cluster. If multiple transition matrices exist, the function iterates over each cluster in the `tna` object to find communities using different algorithms. The function uses the `igraph` package to convert the transition matrices into graphs and then applies community detection algorithms (e.g., Walktrap, Fast Greedy, Label Propagation, Infomap, Edge Betweenness, Leading Eigenvector, and Spin Glass).

Usage

```
communities(x, ...)

## S3 method for class 'tna'
communities(x, methods, gamma = 1, ...)

## S3 method for class 'group_tna'
communities(x, methods, ...)
```

Arguments

`x`	A `tna` or a `group_tna` object.
`...`	Ignored.
`methods`	A `character` vector of community detection algorithms to apply to the network. The supported options are: • `"walktrap"`: A community detection method using short random walks. • `"fast_greedy"`: A method based on modularity optimization. • `"label_prop"`: A method that uses label propagation. • `"infomap"`: A method that uses information flow to detect communities. • `"edge_betweenness"`: A method that uses edge betweenness to find communities. • `"leading_eigen"`: A method using the leading eigenvector of the modularity matrix. • `"spinglass"`: A method based on the spinglass model. If not provided, all methods are applied.
`gamma`	A `numeric` value depicting a parameter that affects the behavior of certain algorithms like the Spin Glass method. Defaults to `1`.

Value

An object of class `tna_communities` which is a `list` with an element for each cluster containing:

- `counts`: A `list` with the number of communities found by each algorithm.
- `assignments`: A `data.frame` where each row corresponds to a node and each column to a community detection algorithm, with color-coded community assignments.

If `x` is a `group_tna` object, a `group_tna_communities` object is returned instead, which is a `list` of `tna_communities` objects.

See Also

Pattern-finding functions plot.tna_communities(), plot.tna_comparison()

Cluster-related functions bootstrap(), centralities(), cliques(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralities(), plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Examples

```
model <- tna(group_regulation)
comm <- communities(model)
```

---

compare *Compare Two Matrices or TNA Models with Comprehensive Metrics*

Description

Various distances, measures of dissimilarity and similarity, correlations and other metrics are computed to compare the models. Optionally, the weight matrices of the models can be scaled before comparison. The resulting object can be used to produce heatmap plots and scatterplots to further illustrate the differences.

Usage

```
compare(x, ...)

## S3 method for class 'tna'
compare(x, y, scaling = "none", ...)

## S3 method for class 'matrix'
compare(x, y, scaling = "none", ...)
```

Arguments

x	A tna object or a matrix of weights.
...	Ignored.
y	A tna object or a matrix of weights.
scaling	A character string naming a scaling method to apply to the weights before comparing them. The supported options are: • "none": No scaling is performed. The weights are used as is.

- "minmax": Performs min-max normalization, i.e., the minimum value is subtracted and the differences are scaled by the range.
- "rank": Applies min-max normalization to the ranks of the weights (computed with ties.method = "average").
- "zscore": Computes the standard score, i.e. the mean weight is subtracted and the differences are scaled by the standard deviation.
- "robust": Computes the robust z-score, i.e. the median weight is subtracted and the differences are scaled by the median absolute deviation (using stats::mad).
- "log": Simply the natural logarithm of the weights.
- "log1p": As above, but adds 1 to the values before taking the logarithm. Useful for scenarios with zero weights.
- "softmax": Performs softmax normalization.
- "quantile": Uses the empirical quantiles of the weights via stats::ecdf.

Value

A tna_comparison object, which is a list containing the following elements:

- matrices: A list containing the scaled matrices of the input tna objects or the scaled inputs themselves in the case of matrices.
- difference_matrix: A matrix of differences x - y.
- edge_metrics: A data.frame of edge-level metrics about the differences.
- summary_metrics: A data.frame of summary metrics of the differences across all edges.
- network_metrics: A data.frame of network metrics for both x and y.
- centrality_differences: A data.frame of differences in centrality measures computes from x and y.
- centrality_correlations: A numeric vector of correlations of the centrality measures between x and y.

Examples

```
# Comparing TNA models
model_x <- tna(group_regulation[1:200, ])
model_y <- tna(group_regulation[1001:1200, ])
comp1 <- compare(model_x, model_y)

# Comparing matrices
mat_x <- model_x$weights
mat_y <- model_y$weights
comp2 <- compare(mat_x, mat_y)

# Comparing a matrix to a TNA model
comp3 <- compare(mat_x, model_y)
```

compare.group_tna *Compare TNA Clusters with Comprehensive Metrics*

Description

Compare TNA Clusters with Comprehensive Metrics

Usage

```
## S3 method for class 'group_tna'
compare(x, i = 1L, j = 2L, scaling = "none", ...)
```

Arguments

x	A group_tna object.
i	An integer index or the name of the principal cluster as a character string.
j	An integer index or the name of the secondary cluster as a character string.
scaling	See compare.tna().
...	Additional arguments passed to compare.tna().

Value

A tna_comparison object. See compare.tna() for details.

Examples

```
model <- group_model(engagement_mmm)
compare(model, i = 1, j = 2)
```

deprune *Restore a Pruned Transition Network Analysis Model*

Description

Restore a Pruned Transition Network Analysis Model

Usage

```
deprune(x, ...)

## S3 method for class 'tna'
deprune(x, ...)

## S3 method for class 'tna'
reprune(x, ...)

## S3 method for class 'group_tna'
deprune(x, ...)
```

Arguments

x	A tna or group_tna object.
...	Ignored.

Value

A tna or group_tna object that has not been pruned.

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralities(), plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Examples

```
model <- tna(group_regulation)
pruned_model <- prune(model, method = "threshold", threshold = 0.1)
depruned_model <- deprune(pruned_model) # restore original model
```

---

engagement *Example data on student engagement*

Description

Students' engagement states (Active / Average / Disengaged) throughout a whole study program. The data was generated synthetically based on the article "The longitudinal association between engagement and achievement varies by time, students' profiles, and achievement state: A full program study"

Usage

```
engagement
```

Format

A `stslist` object (sequence data).

Source

doi:10.1016/j.compedu.2023.104787

See Also

Other examples: `engagement_mmm`, `group_regulation`

`engagement_mmm` *Example mixed Markov model fitted to the* `engagement` *data*

Description

Example mixed Markov model fitted to the `engagement` data

Usage

```
engagement_mmm
```

Format

A `mhmm` object.

Source

The data was generated via `mixed_markov_model.R` in https://github.com/sonsoleslp/tna/tree/main/data-raw/

See Also

Other examples: `engagement`, `group_regulation`

`estimate_cs` *Estimate Centrality Stability*

Description

Estimates the stability of centrality measures in a network using subset sampling without replacement. It allows for dropping varying proportions of cases and calculates correlations between the original centralities and those computed using sampled subsets.

Usage

```
estimate_cs(x, ...)

estimate_centrality_stability(x, ...)

## S3 method for class 'tna'
estimate_cs(
  x,
  loops = FALSE,
  normalize = FALSE,
  measures = c("InStrength", "OutStrength", "Betweenness"),
  iter = 1000,
  method = "pearson",
  drop_prop = seq(0.1, 0.9, by = 0.1),
  threshold = 0.7,
  certainty = 0.95,
  detailed = FALSE,
  progressbar = FALSE,
  ...
)

## S3 method for class 'tna'
estimate_centrality_stability(
  x,
  loops = FALSE,
  normalize = FALSE,
  measures = c("InStrength", "OutStrength", "Betweenness"),
  iter = 1000,
  method = "pearson",
  drop_prop = seq(0.1, 0.9, by = 0.1),
  threshold = 0.7,
  certainty = 0.95,
  detailed = FALSE,
  progressbar = FALSE,
  ...
)
```

```
## S3 method for class 'group_tna'
estimate_cs(
  x,
  loops = FALSE,
  normalize = FALSE,
  measures = c("InStrength", "OutStrength", "Betweenness"),
  iter = 1000,
  method = "pearson",
  drop_prop = seq(0.1, 0.9, by = 0.1),
  threshold = 0.7,
  certainty = 0.95,
  detailed = FALSE,
  progressbar = FALSE,
  ...
)

## S3 method for class 'group_tna'
estimate_centrality_stability(
  x,
  loops = FALSE,
  normalize = FALSE,
  measures = c("InStrength", "OutStrength", "Betweenness"),
  iter = 1000,
  method = "pearson",
  drop_prop = seq(0.1, 0.9, by = 0.1),
  threshold = 0.7,
  certainty = 0.95,
  detailed = FALSE,
  progressbar = FALSE,
  ...
)
```

Arguments

x	A tna or a group_tna object representing the temporal network analysis data. The object should be created from a sequence data object.
...	Ignored.
loops	A logical value indicating whether to include loops in the network when computing the centrality measures (default is FALSE).
normalize	A logical value indicating whether to normalize the centrality measures. The default is FALSE.
measures	A character vector of centrality measures to estimate. The default measures are "InStrength", "OutStrength", and "Betweenness". See centralities() for a list of available centrality measures.
iter	An integer specifying the number of resamples to draw. The default is 1000.
method	A character string indicating the correlation coefficient type. The default is "pearson". See stats::cor() for details.

drop_prop	A numeric vector specifying the proportions of cases to drop in each sampling iteration. Default is a sequence from 0.1 to 0.9 in increments of 0.1.
threshold	A numeric value specifying the correlation threshold for calculating the CS-coefficient. The default is 0.7.
certainty	A numeric value specifying the desired level of certainty for the CS-coefficient. Default is 0.95.
detailed	A logical value specifying whether to return detailed sampling results. If TRUE, detailed results are included in the output. The default is FALSE.
progressbar	A logical value. If TRUE, a progress bar is displayed Defaults to FALSE

Details

The function works by repeatedly resampling the data, dropping varying proportions of cases, and calculating centrality measures on the subsets. The correlation between the original centralities and the resampled centralities is calculated for each drop proportion. The stability of each centrality measure is then summarized using a centrality stability (CS) coefficient, which represents the proportion of dropped cases at which the correlations drop below a given threshold (default 0.7).

The results can be visualized by plotting the output object showing the stability of the centrality measures across different drop proportions, along with confidence intervals. The CS-coefficients are displayed in the subtitle.

Value

A tna_stability object which is a list with an element for each measure with the following elements:

- cs_coefficient: The centrality stability (CS) coefficient of the measure.
- correlations: A matrix of correlations between the original centrality and the resampled centralities for each drop proportion.
- detailed_results: A detailed data frame of the sampled correlations, returned only if return_detailed = TRUE

If x is a group_tna object, a group_tna_stability object is returned instead, which is a list of tna_stability objects.

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralities(), plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralities(),

plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Examples

```
model <- tna(group_regulation)
# Small number of iterations and drop proportions for CRAN
estimate_cs(
  model,
  drop_prop = seq(0.3, 0.9, by = 0.2),
  measures = c("InStrength", "OutStrength"),
  iter = 10
)
```

---

group_model *Build a grouped Transition Network Analysis Model*

Description

This function constructs a transition network analysis (TNA) model for each cluster from a given sequence, wide-formatted dataframe, or a mixture Markov model.

Usage

```
group_model(x, ...)

## Default S3 method:
group_model(
  x,
  group,
  type = "relative",
  scaling = character(0L),
  cols,
  params = list(),
  na.rm = TRUE,
  ...
)

## S3 method for class 'mhmm'
group_model(x, ...)

group_tna(x, ...)
```

```
group_ftna(x, ...)

group_ctna(x, ...)
```

Arguments

x	An stslist object describing a sequence of events or states to be used for building the Markov model. The argument x also accepts data.frame objects in wide format, and tna_data objects. Alternatively, the function accepts a mixture Markov model from seqHMM.
...	Ignored.
group	A vector indicating the cluster assignment of each row of the data / sequence. Must have the same length as the number of rows/sequences of x. Alternatively, a single character string giving the column name of the data that defines the group when x is a wide format data.frame or a tna_data object.
type	A character string describing the weight matrix type. Currently supports the following types: • "relative" for relative frequencies (probabilities, the default) • "frequency" for frequencies. • "co-occurrence" for co-occurrences. • "n-gram" for n-gram transitions. Captures higher-order transitions by considering sequences of n states, useful for identifying longer patterns. • "gap" allows transitions between non-adjacent states, with transitions weighted by the gap size. • "window" creates transitions between all states within a sliding window, capturing local relationships (several sequences together). • "reverse" considers the sequences in reverse order (resulting in what is called a reply network in some contexts). The resulting weight matrix is the transpose of the "frequency" option.
scaling	A character vector describing how to scale the weights defined by type. When a vector is provided, the scaling options are applied in the respective order. For example, c("rank", "minmax") would first compute the ranks, then scale them to the unit interval using min-max normalization. An empty vector corresponds to no scaling. Currently supports the following options: • "minmax" performs min-max normalization to scale the weights to the unit interval. Note that if the smallest weight is positive, it will be zero after scaling. • "max" Multiplies the weights by the reciprocal of the largest weight to scale the weights to the unit interval. This options preserves positive ranks, unlike "minmax" when all weights are positive. • "rank" Computes the ranks of the weights using base::rank() with ties.method = "average".
cols	An integer/character vector giving the indices/names of the columns that should be considered as sequence data. Defaults to all columns, i.e., seq(1, ncol(x)). The columns are automatically determined for tna_data objects.

params — A list of additional arguments for models of specific type. The potential elements of this list are:

- n_gram: An integer for n-gram transitions specifying the number of adjacent events. The default value is 2.
- max_gap: An integer for the gap-allowed transitions specifying the largest allowed gap size. The default is 1.
- window_size: An integer for the sliding window transitions specifying the window size. The default is 2.
- weighted: A logical value. If TRUE, the transitions are weighted by the inverse of the sequence length. Can be used for frequency, co-occurrence and reverse model types. The default is FALSE.

na.rm — A logical value that determines if observations with NA value in group be removed. If FALSE, an additional category for NA values will be added. The default is FALSE and a warning is issued if NA values are detected.

Value

An object of class group_tna which is a list containing one element per cluster. Each element is a tna object.

An object of class group_tna which is a list containing one element per cluster. Each element is a tna object.

An object of class group_tna which is a list containing one element per cluster. Each element is a tna object.

An object of class group_tna which is a list containing one element per cluster. Each element is a tna object.

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralities(), plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralities(), plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralities(),

plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Examples

```
group <- c(rep("High", 1000), rep("Low", 1000))
model <- group_model(group_regulation, group = group)

model <- group_model(engagement_mmm)

model <- group_tna(group_regulation, group = gl(2, 1000))

model <- group_ftna(group_regulation, group = gl(2, 1000))

model <- group_ctna(group_regulation, group = gl(2, 1000))
```

---

group_regulation *Example data on group regulation*

Description

Students' regulation during collaborative learning. Students' interactions were coded as: "adapt", "cohesion", "consensus", "coregulate", "discuss", "emotion", "monitor", "plan", "synthesis"

Usage

```
group_regulation
```

Format

A data.frame object.

Source

The data was generated synthetically.

See Also

Other examples: engagement, engagement_mmm

hist.group_tna *Plot a Histogram of Edge Weights for a* group_tna *Object.*

Description

Plot a Histogram of Edge Weights for a group_tna Object.

Usage

```
## S3 method for class 'group_tna'
hist(x, ...)
```

Arguments

x	A group_tna object.
...	Additional arguments passed to graphics::hist().

Value

A list (invisibly) of histogram objects of the edge weights of each cluster.

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), mmm_stats(), plot.group_tna(), plot.group_tna_centralities(), plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Examples

```
model <- group_model(engagement_mmm)
hist(model)
```

`hist.tna` *Plot a Histogram of Edge Weights in the Network*

Description

Plot a Histogram of Edge Weights in the Network

Usage

```
## S3 method for class 'tna'
hist(x, breaks, col = "lightblue", main, xlab, border = "white", ...)
```

Arguments

`x`	a vector of values for which the histogram is desired.
`breaks`	one of: • a vector giving the breakpoints between histogram cells, • a function to compute the vector of breakpoints, • a single number giving the number of cells for the histogram, • a character string naming an algorithm to compute the number of cells (see 'Details'), • a function to compute the number of cells. In the last three cases the number is a suggestion only; as the breakpoints will be set to `pretty` values, the number is limited to `1e6` (with a warning if it was larger). If `breaks` is a function, the `x` vector is supplied to it as the only argument (and the number of breaks is only limited by the amount of available memory).
`col`	a colour to be used to fill the bars.
`main`	A `character` string defining the title of the plot.
`xlab`	A `character` string defining the vertical axis label.
`border`	the color of the border around the bars. The default is to use the standard foreground color.
`...`	Additional arguments passed to `graphics::hist()`.

Value

A `histogram` object of edge weights.

Examples

```
model <- tna(group_regulation)
hist(model)
```

`mmm_stats` *Retrieve statistics from a mixture Markov model (MMM)*

Description

Retrieve statistics from a mixture Markov model (MMM)

Usage

```
mmm_stats(x, use_t_dist = TRUE, level = 0.05)
```

Arguments

`x`	A `mhmm` object.
`use_t_dist`	A `logical` value. If `TRUE` (the default), the t-distribution is used to compute confidence intervals.
`level`	A `numeric` value representing the significance level for hypothesis testing and confidence intervals. Defaults to `0.05`.

Value

A `data.frame` object.

See Also

Cluster-related functions `bootstrap()`, `centralities()`, `cliques()`, `communities()`, `deprune()`, `estimate_cs()`, `group_model()`, `hist.group_tna()`, `plot.group_tna()`, `plot.group_tna_centralities()`, `plot.group_tna_cliques()`, `plot.group_tna_communities()`, `plot.group_tna_stability()`, `plot_compare.group_tna()`, `plot_mosaic.group_tna()`, `plot_mosaic.tna_data()`, `print.group_tna()`, `print.group_tna_bootstrap()`, `print.group_tna_centralities()`, `print.group_tna_cliques()`, `print.group_tna_communities()`, `print.group_tna_stability()`, `print.summary.group_tna()`, `print.summary.group_tna_bootstrap()`, `prune()`, `pruning_details()`, `rename_groups()`, `reprune()`, `summary.group_tna()`, `summary.group_tna_bootstrap()`

Examples

```
mmm_stats(engagement_mmm)
```

permutation_test *Compare Two Networks from Sequence Data Using Permutation Tests*

Description

This function compares two networks built from sequence data using permutation tests. The function builds Markov models for two sequence objects, computes the transition probabilities, and compares them by performing permutation tests. It returns the differences in transition probabilities, effect sizes, p-values, and confidence intervals.

Usage

```
permutation_test(
  x,
  y,
  iter = 1000,
  paired = FALSE,
  level = 0.05,
  measures = character(0),
  ...
)
```

Arguments

x	A tna object containing sequence data for the first tna model.
y	A tna object containing sequence data for the second tna model.
iter	An integer giving the number of permutations to perform. The default is 1000.
paired	A logical value. If TRUE, perform paired permutation tests; if FALSE, perform unpaired tests. The default is FALSE.
level	A numeric value giving the significance level for the permutation tests. The default is 0.05.
measures	A character vector of centrality measures to test. See centralities() for a list of available centrality measures.
...	Additional arguments passed to centralities().

Value

A tna_permutation object which is a list with two elements: edges and centralities, both containing the following elements

- stats: A data.frame of original differences, effect sizes, and p-values for each edge or centrality measure. The effect size is computed as the observed difference divided by the standard deviation of the differences of the permuted samples.
- diffs_true: A matrix of differences in the data.
- diffs_sig: A matrix showing the significant differences.

See Also

Evaluation and validation functions bootstrap(), prune(), pruning_details()

Examples

```
model_x <- tna(group_regulation[1:200, ])
model_y <- tna(group_regulation[1001:1200, ])
# Small number of iterations for CRAN
permutation_test(model_x, model_y, iter = 20)
```

---

plot.group_tna *Plot a grouped Transition Network Analysis Model*

Description

Plots a transition network of each cluster using qgraph.

Usage

```
## S3 method for class 'group_tna'
plot(x, title, ...)
```

Arguments

x	A group_model object.
title	A title for each plot. It can be a single string (the same one will be used for all plots) or a list (one per group)
...	Same as plot.tna().

Value

NULL (invisibly).

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna_centralities(), plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Examples

```
model <- group_model(engagement_mmm)
plot(model)
```

`plot.group_tna_centralities` *Plot Centrality Measures*

Description

Plot Centrality Measures

Usage

```
## S3 method for class 'group_tna_centralities'
plot(
  x,
  reorder = TRUE,
  ncol = 3,
  scales = c("free_x", "fixed"),
  colors,
  labels = TRUE,
  ...
)
```

Arguments

`x`	A `group_tna_centralities` object.
`reorder`	A `logical` value indicating whether to reorder the values for each centrality in a descending order. The default is `TRUE`.
`ncol`	Number of columns to use for the facets. The default is 3.
`scales`	Either `"fixed"` or `"free_x"` (the default). If `"free_x"`, the horizontal axis is scaled individually in each facet. If `"fixed"`, the same values are used for all axes.
`colors`	The colors for each node (default is the model colors if the `tna` model object is passed, otherwise `"black"`).
`labels`	A `logical` value indicating whether to show the centrality numeric values. The default is `TRUE`.
`...`	Ignored.

Value

A `ggplot` object displaying a line chart for each centrality with one line per cluster.

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap(

Examples

```
model <- group_model(engagement_mmm)
cm <- centralities(model)
plot(cm)
```

---

plot.group_tna_cliques

Plot Found Cliques

Description

Plot Found Cliques

Usage

```
## S3 method for class 'group_tna_cliques'
plot(x, title, ...)
```

Arguments

x	A group_tna_cliques object.
title	A character vector of titles to use for each plot.
...	Arguments passed to plot.tna_cliques().

Value

A list (invisibly) with one element per cluster. Each element contains a qgraph plot when only one clique is present per cluster, otherwise the element is NULL.

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap(

Examples

```
model <- group_model(engagement_mmm)
cliq <- cliques(model, size = 2)
plot(cliq, ask = F)
```

`plot.group_tna_communities` *Plot Found Communities*

Description

Plot Found Communities

Usage

```
## S3 method for class 'group_tna_communities'
plot(x, title = names(x), colors, ...)
```

Arguments

`x`	A `group_tna_communities` object.
`title`	A `character` vector of titles to use for each plot.
`colors`	A `character` vector of colors to use.
`...`	Arguments passed to `plot.tna_communities()`.

Value

A `list` (invisibly) of `qgraph` objects in which the nodes are colored by community for each cluster.

See Also

Cluster-related functions `bootstrap()`, `centralities()`, `cliques()`, `communities()`, `deprune()`, `estimate_cs()`, `group_model()`, `hist.group_tna()`, `mmm_stats()`, `plot.group_tna()`, `plot.group_tna_centralitie` `plot.group_tna_cliques()`, `plot.group_tna_stability()`, `plot_compare.group_tna()`, `plot_mosaic.group_tna()` `plot_mosaic.tna_data()`, `print.group_tna()`, `print.group_tna_bootstrap()`, `print.group_tna_centralities()`, `print.group_tna_cliques()`, `print.group_tna_communities()`, `print.group_tna_stability()`, `print.summary.group_tna()`, `print.summary.group_tna_bootstrap()`, `prune()`, `pruning_details()`, `rename_groups()`, `reprune()`, `summary.group_tna()`, `summary.group_tna_bootstrap()`

Examples

```
model <- group_model(engagement_mmm)
comm <- communities(model)
plot(comm)
```

plot.group_tna_stability *Plot Centrality Stability Results*

Description

Plot Centrality Stability Results

Usage

```
## S3 method for class 'group_tna_stability'
plot(x, ...)
```

Arguments

x	A group_tna_stability object.
...	Arguments passed to plot.tna_stability().

Value

A list (invisibly) of ggplot objects displaying the stability analysis plot.

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap(

Examples

```
model <- group_model(engagement_mmm)
# Low number of iterations for CRAN
stability <- estimate_cs(
  model,
  drop_prop = c(0.3, 0.5, 0.7, 0.9),
  iter = 10
)
plot(stability)
```

plot.tna *Plot a Transition Network Analysis Model*

Description

This function plots a transition network analysis (TNA) model using the qgraph package. The nodes in the graph represent states, with node sizes corresponding to initial state probabilities. Edge labels represent the edge weights of the network.

Usage

```
## S3 method for class 'tna'
plot(
  x,
  labels,
  colors,
  pie,
  show_pruned = TRUE,
  pruned_edge_color = "red",
  edge.color = NA,
  edge.labels = TRUE,
  edge.label.position = 0.65,
  layout = "circle",
  layout_args = list(),
  mar = rep(5, 4),
  theme = "colorblind",
  ...
)
```

Arguments

x	A tna object from tna().
labels	See qgraph::qgraph().
colors	See qgraph::qgraph().
pie	See qgraph::qgraph().
show_pruned	A logical value indicating if pruned edges removed by prune() should be shown in the plot. The default is TRUE, and the edges are drawn as dashed with a different color to distinguish them.
pruned_edge_color	A character string for the color to use for pruned edges when show_pruned = TRUE. The default is "red".
edge.color	See qgraph::qgraph().
edge.labels	See qgraph::qgraph().
edge.label.position	See qgraph::qgraph().

layout	One of the following: • A character string describing a qgraph layout. • A matrix of node positions to use, with a row for each node and x and y columns for the node positions. • A layout function from igraph.
layout_args	A list of arguments to pass to the igraph layout function when layout is a function.
mar	See qgraph::qgraph().
theme	See qgraph::qgraph().
...	Additional arguments passed to qgraph::qgraph().

Value

A qgraph plot of the transition network.

See Also

Core functions build_model(), centralities(), plot.tna_centralities(), plot_compare()

Examples

```
model <- tna(group_regulation)
plot(model)
```

---

plot.tna_centralities *Plot Centrality Measures*

Description

Plots the centrality measures of a tna_centralities object as a lollipop chart. The resulting plot includes facets for each centrality measure, showing the values for each state. The returned plot is a ggplot2 object, so it can be easily modified and styled. See centralities() for details on the centrality measures.

Usage

```
## S3 method for class 'tna_centralities'
plot(
  x,
  reorder = TRUE,
  ncol = 3,
  scales = c("free_x", "fixed"),
  colors,
  labels = TRUE,
  ...
)
```

Arguments

x	An object of class tna_centralities.
reorder	A logical value indicating whether to reorder the values for each centrality in a descending order. The default is TRUE.
ncol	Number of columns to use for the facets. The default is 3.
scales	Either "fixed" or "free_x" (the default). If "free_x", the horizontal axis is scaled individually in each facet. If "fixed", the same values are used for all axes.
colors	The colors for each node (default is the model colors if the tna model object is passed, otherwise "black").
labels	A logical value indicating whether to show the centrality numeric values. The default is TRUE.
...	Ignored.

Value

A ggplot object displaying the lollipop charts for each centrality measure.

See Also

Core functions build_model(), centralities(), plot.tna(), plot_compare()

Examples

```
tna_model <- tna(group_regulation)
cm <- centralities(tna_model)
plot(cm, ncol = 3, reorder = TRUE)
```

---

plot.tna_cliques *Plot Cliques of a TNA Network*

Description

Plot Cliques of a TNA Network

Usage

```
## S3 method for class 'tna_cliques'
plot(
  x,
  n = 6,
  first = 1,
  show_loops = FALSE,
  edge.labels = TRUE,
```

```
  edge.label.position = 0.65,
  minimum = 0.00001,
  mar = rep(5, 4),
  layout = "circle",
  layout_args = list(),
  cut = 0.01,
  normalize = TRUE,
  ask = TRUE,
  colors,
  theme = "colorblind",
  ...
)
```

Arguments

`x`	A `tna_cliques` object.
`n`	An `integer` defining the maximum number of cliques to show. The defaults is `6`.
`first`	An `integer` giving the index of the first clique to show. The default index is `1`.
`show_loops`	A `logical` value indicating whether to include loops in the plots or not.
`edge.labels`	See `qgraph::qgraph()`.
`edge.label.position`	See `qgraph::qgraph()`.
`minimum`	See `qgraph::qgraph()`.
`mar`	See `qgraph::qgraph()`.
`layout`	One of the following: • A `character` string describing a `qgraph` layout. • A `matrix` of node positions to use, with a row for each node and `x` and `y` columns for the node positions. • A layout function from `igraph`.
`layout_args`	A `list` of arguments to pass to the `igraph` layout function when `layout` is a function.
`cut`	See `qgraph::qgraph()`.
`normalize`	See `qgraph::qgraph()`.
`ask`	A `logical` value. When `TRUE`, show plots one by one and asks to plot the next plot in interactive mode.
`colors`	See `qgraph::qgraph()`.
`theme`	See `qgraph::qgraph()`.
`...`	Ignored.

Value

`NULL` (invisibly).

Examples

```
model <- tna(group_regulation)
cliq <- cliques(model, size = 2)
plot(cliq, n = 1, ask = FALSE)
```

`plot.tna_communities` *Plot Communities*

Description

This function visualizes the communities detected within a `tna` object based on different community detection algorithms and their corresponding color mappings.

Usage

```
## S3 method for class 'tna_communities'
plot(x, colors, method = "spinglass", ...)
```

Arguments

`x`	A `communities` object generated by the `find_communities` method. Each community detection method maps nodes or points in to a specific communities.
`colors`	A `character` vector of color values used for visualizing community assignments.
`method`	A `character` string naming a community detection method to use for coloring the plot. See `communities()` for details.
`...`	Additional arguments passed to `qgraph::qgraph()`.

Value

A `qgraph` object in which the nodes are colored by community.

See Also

Pattern-finding functions `communities()`, `plot.tna_comparison()`

Examples

```
model <- tna(group_regulation)
comm <- communities(model)
plot(comm, method = "leading_eigen")
```

plot.tna_comparison *Plot the results of comparing two* tna *models or matrices*

Description

Plot the results of comparing two tna models or matrices

Usage

```
## S3 method for class 'tna_comparison'
plot(
  x,
  type = "heatmap",
  population = "difference",
  method = "pearson",
  name_x = "x",
  name_y = "y",
  ...
)
```

Arguments

x	A tna_comparison object.
type	A character string naming the type of plot to produce. The available options are "heatmap" (the default), "scatterplot", "centrality_heatmap", and "weight_density".
population	A "character" string naming the population for which to produce the heatmaps, i.e, one of "x", "y", or "difference" for the differences. Ignored for type = "scatterplot". Defaults to "diff".
method	A character string naming the correlation coefficient to use when plotting a scatterplot. The available options are "pearson" (the default), "kendall", "spearman", and "distance". The final option is the distance correlation coefficient of Szekely, Rizzo, and Bakirov (2007). See also the energy package for further information on this measure.
name_x	An optional character string to use as the name of the first population in the plots. The default is "x".
name_y	An optional character string to use as the name of the second population in the plots. The default is "y".
...	Ignored.

Value

A ggplot object.

References

Szekely, G.J., Rizzo, M.L., and Bakirov, N.K. (2007), Measuring and Testing Dependence by Correlation of Distances, *Annals of Statistics*, Vol. 35 No. 6, pp. 2769-2794. doi:10.1214/009053607000000505

See Also

Pattern-finding functions `communities()`, `plot.tna_communities()`

Examples

```
model_x <- tna(group_regulation[1:200, ])
model_y <- tna(group_regulation[1001:1200, ])
comp <- compare(model_x, model_y)
plot(comp)
```

---

plot.tna_permutation *Plot the Significant Differences from a Permutation Test*

Description

Plot the Significant Differences from a Permutation Test

Usage

```
## S3 method for class 'tna_permutation'
plot(x, colors, ...)
```

Arguments

`x`	A `tna_permutation` object.
`colors`	See `qgraph::qgraph()`.
`...`	Arguments passed to `plot_model()`.

Value

A `qgraph` object containing only the significant edges according to the permutation test.

Examples

```
model_x <- tna(group_regulation[1:200, ])
model_y <- tna(group_regulation[1001:1200, ])
# Small number of iterations for CRAN
perm <- permutation_test(model_x, model_y, iter = 20)
plot(perm)
```

`plot.tna_stability` *Plot Centrality Stability Results*

Description

This function visualizes the centrality stability results produced by the `estimate_centrality_stability` function. It shows how different centrality measures' correlations change as varying proportions of cases are dropped, along with their confidence intervals (CIs).

Usage

```
## S3 method for class 'tna_stability'
plot(x, level = 0.05, ...)
```

Arguments

`x`	A `tna_stability` object produced by `estimate_cs`.
`level`	A `numeric` value representing the significance level for the confidence intervals. Defaults to `0.05`.
`...`	Ignored.

Details

The function aggregates the results for each centrality measure across multiple proportions of dropped cases (e.g., 0.1, 0.2, ..., 0.9) and calculates the mean and the desired quantiles for each proportion. The confidence intervals (CIs) are computed based on the quantiles and displayed in the plot.

If no valid data is available for a centrality measure (e.g., missing or NA values), the function skips that measure with a warning.

The plot includes:

- The mean correlation for each centrality measure as a function of the proportion of dropped cases.
- Shaded confidence intervals representing CIs for each centrality measure.
- A horizontal dashed line at the threshold value used for calculating the CS-coefficient.
- A subtitle listing the CS-coefficients for each centrality measure.

Value

A `ggplot` object displaying the stability analysis plot.

Examples

```
model <- tna(group_regulation)
cs <- estimate_cs(model, iter = 10)
plot(cs)
```

plot_compare *Plot the difference network between two models*

Description

Plots the difference network between model x and model y. The edges are computed from subtracting the two models. The pie chart is the difference in initial probabilities between model x and model y. Green color indicates that xis greater than yand red indicates otherwise.

Usage

```
plot_compare(x, ...)

## S3 method for class 'tna'
plot_compare(x, y, theme = NULL, palette = "colorblind", ...)
```

Arguments

x	A tna object. It will be the principal model.
...	Additional arguments passed to qgraph::qgraph().
y	A tna object. It will be the model subtracted from the principal model.
theme	See qgraph::qgraph().
palette	See qgraph::qgraph().

Value

A qgraph object displaying the difference network between the two models.

See Also

Core functions build_model(), centralities(), plot.tna(), plot.tna_centralities()

Examples

```
model_x <- tna(group_regulation[group_regulation[, 1] == "plan", ])
model_y <- tna(group_regulation[group_regulation[, 1] != "plan", ])
plot_compare(model_x, model_y)
```

plot_compare.group_tna *Plot the difference network between two clusters*

Description

Plot the difference network between two clusters

Usage

```
## S3 method for class 'group_tna'
plot_compare(x, i = 1L, j = 2L, ...)
```

Arguments

x	A group_tna object.
i	An integer index or the name of the principal cluster as a character string.
j	An integer index or the name of the secondary cluster as a character string.
...	Additional arguments passed to plot_compare.tna().

Value

A qgraph object displaying the difference network between the two clusters

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap(

Examples

```
model <- group_model(engagement_mmm)
plot_compare(model)
```

plot_mosaic *Create a mosaic plot of transitions*

Description

Create a mosaic plot of transitions

Usage

```
plot_mosaic(x, ...)

## S3 method for class 'tna'
plot_mosaic(x, digits = 1, ...)
```

Arguments

x	A tna or a group_tna object.
...	Ignored.
digits	An integer that determines the number of digits to use for the chi-square test statistic and the p-value in the plot.

Value

A ggplot object.

Examples

```
ftna_model <- ftna(group_regulation)
plot_mosaic(ftna_model)
```

plot_mosaic.group_tna *Plot state frequencies as a mosaic between two groups*

Description

Plot state frequencies as a mosaic between two groups

Usage

```
## S3 method for class 'group_tna'
plot_mosaic(x, label, digits = 1, ...)
```

Arguments

x	A group_tna object.
label	An optional character string that can be provided to specify the grouping factor name if x was not constructed using a column name of the original data.
digits	An integer that determines the number of digits to use for the chi-square test statistic and the p-value in the plot.
...	Ignored.

Value

A ggplot object.

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap(

Examples

```
model <- group_model(engagement_mmm)
plot_mosaic(model)
```

---

plot_mosaic.tna_data *Plot state frequencies as a mosaic between two groups*

Description

Plot state frequencies as a mosaic between two groups

Usage

```
## S3 method for class 'tna_data'
plot_mosaic(x, group, label = "Group", digits = 1, ...)
```

Arguments

x	A tna_data object.
group	A character string giving the column name of the (meta) data to contrast the frequencies with or a vector of group indicators with the the same length as the number of rows in the sequence data.

label	An optional character string that specifies a label for the grouping variable when group is not a column name of the data.
digits	An integer that determines the number of digits to use for the chi-square test statistic and the p-value in the plot.
...	Ignored.

Value

A ggplot object.

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap(

Examples

```
d <- data.frame(
  time = rep(1:5, rep = 4),
  group = rep(1:4, each = 5),
  event = sample(LETTERS[1:3], 20, replace = TRUE)
)
sequence_data <- prepare_data(
  d,
  time = "time",
  actor = "group",
  action = "event"
)
plot_mosaic(sequence_data, group = "group")
```

---

prepare_data *Compute User Sessions from Event Data*

Description

Processes a dataset to create user sessions based on time gaps, ordering columns, or actor groupings. It supports different ways to understand order in user behavior and provides flexibility when widening the data.

Usage

```
prepare_data(
  data,
  actor,
  time,
  action,
  order,
  time_threshold = 900,
  custom_format = NULL,
  is_unix_time = FALSE,
  unix_time_unit = "seconds",
  unused_fn = dplyr::first
)
```

Arguments

data	A data.frame or containing the action/event data.
actor	A character string giving the name of the column that represents a user/actor identifier. If not provided and neither time nor order is specified, the entire dataset is treated as a single session.
time	A character string giving the name of the column representing timestamps of the action events.
action	A character string giving the name of the column holding the information about the action taken.
order	A character string giving the name of a column with sequence numbers or non-unique orderable values that indicate order within an actor group, if not present it will be ordered with all the data if no actor is available, used when widening the data. If both actor and time are specified, then the sequence order should be specified such that it determines the order of events within actor and each session.
time_threshold	An integer specifying the time threshold in seconds for creating new time-based sessions. Defaults to 900 seconds.
custom_format	A character string giving the format used to parse the time column.
is_unix_time	A logical value indicating whether the time column is in Unix time. The default is FALSE.
unix_time_unit	A character string giving the Unix time unit when is_unix_time is TRUE. The default is "seconds". Valid options are "seconds", "milliseconds", or "microseconds".
unused_fn	How to handle extra columns when pivoting to wide format. See tidyr::pivot_wider(). The default is to keep all columns and to use the first value.

Value

A tna_data object, which is a list with the following elements:

- long_data: The processed data in long format.

- sequence_data: The processed data on the sequences in wide format, with actions/events as different variables structured with sequences.
- meta_data: Other variables from the original data in wide format.
- statistics: A list containing summary statistics: total sessions, total actions, unique users, time range (if applicable), and top sessions and user by activities.

Examples

```
data <- tibble::tibble(
  user = c("A", "A", "A", "B", "B", "C", "C", "C"),
  time = c(
    "2023-01-01 10:00:00", "2023-01-01 10:05:00",
    "2023-01-01 10:20:00", "2023-01-01 12:00:00",
    "2023-01-01 12:02:00", "2023-01-01 14:00:00",
    "2023-01-01 14:05:00", "2023-01-01 14:10:00"
  ),
  action = c(
    "view", "click", "add_cart", "view",
    "checkout", "view", "click", "share"
   )
)
results <- prepare_data(
  data, actor = "user", time = "time", action = "action"
)
print(results$sequence_data)
print(results$meta_data)
print(results$statistics)

data_ordered <- tibble::tibble(
   user = c("A", "A", "A", "B", "B", "C", "C", "C"),
   order = c(1, 2, 3, 1, 2, 1, 2, 3),
   action = c(
     "view", "click", "add_cart", "view",
     "checkout", "view", "click", "share"
   )
)
results_ordered <- prepare_data(
  data_ordered, actor = "user", order = "order", action = "action"
)
print(results_ordered$sequence_data)
print(results_ordered$meta_data)
print(results_ordered$statistics)

data_single_session <- tibble::tibble(
  action = c(
    "view", "click", "add_cart", "view", "checkout", "view", "click", "share"
   )
)
results_single <- prepare_data(data_single_session, action = "action")
print(results_single$sequence_data)
print(results_single$meta_data)
print(results_single$statistics)
```

print.group_tna *Print a* group_tna *Object*

Description

Print a group_tna Object

Usage

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

Arguments

x	A group_tna object.
...	Arguments passed to print.tna().

Value

x (invisibly).

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap(

Examples

```
model <- group_model(engagement_mmm)
print(model)
```

print.group_tna_bootstrap *Print* group_tna *Bootstrap Results*

Description

Print group_tna Bootstrap Results

Usage

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

Arguments

x	A group_tna_bootstrap object.
...	Arguments passed to print.tna_bootstrap().

Value

x (invisibly).

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap(

Examples

```
model <- group_model(engagement_mmm)
# Low number of iteration for CRAN
boot <- bootstrap(model, iter = 10)
print(boot)
```

print.group_tna_centralities *Print Centrality Measures*

Description

Print Centrality Measures

Usage

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

Arguments

x	A group_tna_centralities object.
...	Ignored.

Value

x (invisibly).

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap(

Examples

```
model <- group_model(engagement_mmm)
cm <- centralities(model)
print(cm)
```

print.group_tna_cliques *Print Found Cliques*

Description

Print Found Cliques

Usage

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

Arguments

x	A group_tna_cliques object.
...	Arguments passed to print.tna_cliques().

Value

x (invisibly).

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap(

Examples

```
model <- group_model(engagement_mmm)
cliq <- cliques(model, size = 2)
print(cliq)
```

print.group_tna_communities *Print Detected Communities*

Description

Print Detected Communities

Usage

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

Arguments

x	A group_tna_communities object.
...	Arguments passed to print.tna_communities().

Value

x (invisibly).

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap(

Examples

```
model <- group_model(engagement_mmm)
comm <- communities(model)
print(comm)
```

print.group_tna_stability *Print Centrality Stability Results*

Description

Print Centrality Stability Results

Usage

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

Arguments

x	A group_tna_stability object.
...	Arguments passed to print.tna_stability().

Value

x (invisibly).

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap(

Examples

```
model <- group_model(engagement_mmm)
# Low number of iterations for CRAN
stability <- estimate_cs(
  model,
  drop_prop = c(0.3, 0.5, 0.7, 0.9),
  iter = 10
)
print(stability)
```

print.summary.group_tna *Print the summary of a grouped Transition Network Analysis Model*

Description

Print the summary of a grouped Transition Network Analysis Model

Usage

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

Arguments

x	A summary.group_tna object.
...	Arguments passed to print.summary.tna().

Value

x (invisibly).

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap(

Examples

```
model <- group_model(engagement_mmm)
print(summary(model))
```

`print.summary.group_tna_bootstrap` *Print Bootstrap Summary for a Grouped Transition Network*

Description

Print Bootstrap Summary for a Grouped Transition Network

Usage

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

Arguments

`x`	A `summary.group_tna_bootstrap` object.
`...`	Arguments passed to the generic `print` method.

Value

`x` (invisibly).

See Also

Cluster-related functions `bootstrap()`, `centralities()`, `cliques()`, `communities()`, `deprune()`, `estimate_cs()`, `group_model()`, `hist.group_tna()`, `mmm_stats()`, `plot.group_tna()`, `plot.group_tna_centralitie` `plot.group_tna_cliques()`, `plot.group_tna_communities()`, `plot.group_tna_stability()`, `plot_compare.group_tna()`, `plot_mosaic.group_tna()`, `plot_mosaic.tna_data()`, `print.group_tna()`, `print.group_tna_bootstrap()`, `print.group_tna_centralities()`, `print.group_tna_cliques()`, `print.group_tna_communities()`, `print.group_tna_stability()`, `print.summary.group_tna()`, `prune()`, `pruning_details()`, `rename_groups()`, `reprune()`, `summary.group_tna()`, `summary.group_tna_bootstrap(`

Examples

```
model <- group_model(engagement_mmm)
# Low number of iteration for CRAN
boot <- bootstrap(model, iter = 10)
print(summary(boot))
```

print.summary.tna *Print a TNA Summary*

Description

Print a TNA Summary

Usage

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

Arguments

x	A summary.tna object.
...	Ignored.

Value

A summary.tna object (invisibly) containing the TNA model network metrics and values.

Examples

```
model <- tna(group_regulation)
print(summary(model))
```

print.summary.tna_bootstrap *Print Bootstrap Summary*

Description

Print Bootstrap Summary

Usage

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

Arguments

x	A summary.tna_bootstrap object.
...	Arguments passed to the generic print method.

Value

A summary.tna_bootstrap object (invisibly) containing the weight, p-value and confidence interval of each edge.

Examples

```
model <- tna(group_regulation)
# Small number of iterations for CRAN
boot <- bootstrap(model, iter = 10)
print(summary(boot))
```

---

print.tna *Print a* tna *object*

Description

Print a tna object

Usage

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

Arguments

x	A tna object.
digits	An integer giving the number of *significant* digits to print.
generic	A logical value. If TRUE, use generic print method instead. Defaults to FALSE.
...	Ignored.

Value

The tna object passed as argument x (invisibly).

Examples

```
model <- tna(group_regulation)
print(model)
```

print.tna_bootstrap *Print Bootstrap Results*

Description

Print Bootstrap Results

Usage

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

Arguments

x	A tna_bootstrap object.
digits	An integer giving the minimal number of *significant* digits to print.
type	A character vector giving the type of edges to print. The default option "both" prints both statistically significant and non-significant edges, "sig" prints only significant edges, and "nonsig" prints only the non-significant edges.
...	Ignored.

Value

x (invisibly).

Examples

```
model <- tna(group_regulation)
# Small number of iterations for CRAN
boot <- bootstrap(model, iter = 10)
print(boot)
```

print.tna_centralities *Print Centrality Measures*

Description

Print Centrality Measures

Usage

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

Arguments

x	A centralities object.
...	Ignored.

Value

x (invisibly).

Examples

```
model <- tna(group_regulation)
cm <- centralities(model)
print(cm)
```

---

print.tna_cliques *Print Found Cliques of a TNA Network*

Description

Print Found Cliques of a TNA Network

Usage

```
## S3 method for class 'tna_cliques'
print(x, n = 6, first = 1, digits = getOption("digits"), ...)
```

Arguments

x	A tna_cliques object.
n	An integer defining the maximum number of cliques to show. The defaults is 6.
first	An integer giving the index of the first clique to show. The default index is 1.
digits	An integer giving the minimal number of *significant* digits to print.
...	Ignored.

Value

x (invisibly).

Examples

```
model <- tna(group_regulation)
cliq <- cliques(model, size = 2)
print(cliq)
```

print.tna_communities *Print Detected Communities*

Description

Print Detected Communities

Usage

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

Arguments

x	A tna_communities object.
...	Ignored.

Value

x (invisibly).

Examples

```
model <- tna(group_regulation)
comm <- communities(model)
print(comm)
```

print.tna_comparison *Print Comparison Results*

Description

Print Comparison Results

Usage

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

Arguments

x	A tna_comparison object.
...	Additional arguments passed to the tibble print method.

Value

x (invisibly).

Examples

```
model_x <- tna(group_regulation[1:200, ])
model_y <- tna(group_regulation[1001:1200, ])
comp <- compare(model_x, model_y)
print(comp)
```

---

print.tna_data *Print a TNA data object*

Description

Print a TNA data object

Usage

```
## S3 method for class 'tna_data'
print(x, data = "sequence", ...)
```

Arguments

x	A tna_data object.
data	A character string that defines the data to be printed tibble. Accepts either "sequence" (default) for wide format sequence data, "meta", for the wide format metadata, or "long" for the long format data.
...	Arguments passed to the tibble print method.

Value

x (invisibly).

Examples

```
data_single_session <- tibble::tibble(
  action = c(
    "view", "click", "add_cart", "view", "checkout", "view", "click", "share"
   )
)
results_single <- prepare_data(data_single_session, action = "action")
print(results_single, which = "sequence")
print(results_single, which = "meta")
print(results_single, which = "long")
```

print.tna_permutation *Print Permutation Test Results*

Description

Print Permutation Test Results

Usage

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

Arguments

x	A tna_permutation object.
...	Additional arguments passed to the tibble print method.

Value

x (invisibly).

Examples

```
model_x <- tna(group_regulation[1:200, ])
model_y <- tna(group_regulation[1001:1200, ])
# Small number of iterations for CRAN
perm <- permutation_test(model_x, model_y, iter = 20)
print(perm)
```

print.tna_stability *Print Centrality Stability Results*

Description

Print Centrality Stability Results

Usage

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

Arguments

x	A tna_stability object.
...	Ignored.

Value

x (invisibly).

Examples

```
model <- tna(group_regulation)
# Small number of iterations and drop proportions for CRAN
cs <- estimate_cs(
  model,
  measures = c("InStrength", "OutStrength"),
  drop_prop = seq(0.3, 0.9, by = 0.2),
  iter = 10
)
print(cs)
```

---

prune *Prune a* tna *network based on transition probabilities*

Description

Prunes a network represented by a tna object by removing edges based on a specified threshold, lowest percent of non-zero edge weights, or the disparity filter algorithm (Serrano et al., 2009). It ensures the network remains weakly connected.

Prunes a network represented by a tna object by removing edges based on a specified threshold, lowest percent of non-zero edge weights, or the disparity filter algorithm (Serrano et al., 2009). It ensures the network remains weakly connected.

Usage

```
prune(x, ...)

## S3 method for class 'tna'
prune(
  x,
  method = "threshold",
  threshold = 0.1,
  lowest = 0.05,
  level = 0.5,
  boot = NULL,
  ...
)

## S3 method for class 'group_tna'
prune(x, ...)
```

Arguments

x	An object of class tna or group_tna
...	Arguments passed to bootstrap() when using method = "bootstrap" and when a tna_bootstrap is not supplied.
method	A character string describing the pruning method. The available options are "threshold", "lowest", "bootstrap" and "disparity", corresponding to the methods listed in Details. The default is "threshold".
threshold	A numeric value specifying the edge weight threshold. Edges with weights below or equal to this threshold will be considered for removal.
lowest	A numeric value specifying the lowest percentage of non-zero edges. This percentage of edges with the lowest weights will be considered for removal. The default is 0.05.
level	A numeric value representing the significance level for the disparity filter. Defaults to 0.5.
boot	A tna_bootstrap object to be used for pruning with method "boot". The method argument is ignored if this argument is supplied.

Value

A pruned tna or group_tna object. Details on the pruning can be viewed with pruning_details(). The original model can be restored with deprune().

See Also

Evaluation and validation functions bootstrap(), permutation_test(), pruning_details()

Evaluation and validation functions bootstrap(), permutation_test(), pruning_details()

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), pruning_details(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Examples

```
model <- tna(group_regulation)
pruned_threshold <- prune(model, method = "threshold", threshold = 0.1)
pruned_percentile <- prune(model, method = "lowest", lowest = 0.05)
pruned_disparity <- prune(model, method = "disparity", level = 0.5)
```

pruning_details *Print Detailed Information on the Pruning Results*

Description

Print Detailed Information on the Pruning Results

Usage

```
pruning_details(x, ...)

## S3 method for class 'tna'
pruning_details(x, ...)

## S3 method for class 'group_tna'
pruning_details(x, ...)
```

Arguments

x	A tna or group_tna object.
...	Ignored.

Value

A data.frame containing the removed edges if x is a tna object, or a list of data.frame objects in the case of group_tna object.

See Also

Evaluation and validation functions bootstrap(), permutation_test(), prune()

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), rename_groups(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Examples

```
model <- tna(group_regulation)
pruned_threshold <- prune(model, method = "threshold", threshold = 0.1)
pruning_details(pruned_threshold)
```

rename_groups *Rename clusters*

Description

Rename clusters

Usage

```
rename_groups(x, new_names)
```

Arguments

x	A group_tna object.
new_names	A character vector containing one name per cluster.

Value

A renamed group_tna object.

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), reprune(), summary.group_tna(), summary.group_tna_bootstrap()

Examples

```
model <- group_model(engagement_mmm)
model_renamed <- rename_groups(model, c("A", "B", "C"))
```

reprune *Restore Previous Pruning of a Transition Network Analysis Model*

Description

Restore Previous Pruning of a Transition Network Analysis Model

Usage

```
reprune(x, ...)

## S3 method for class 'group_tna'
reprune(x, ...)
```

Arguments

x	A tna or group_tna object.
...	Ignored.

Value

A tna or group_tna object that has not been pruned. The previous pruning result can be reactivated with reprune().

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), summary.group_tna(), summary.group_tna_bootstrap()

Examples

```
model <- tna(group_regulation)
pruned_model <- prune(model, method = "threshold", threshold = 0.1)
depruned_model <- deprune(pruned_model) # restore original model
repruned_model <- reprune(depruned_model) # reapply the previous pruning
```

---

summary.group_tna *Calculate Summary of Network Metrics for a grouped Transition Network*

Description

This function calculates a variety of network metrics for a tna object. It computes key metrics such as node and edge counts, network density, mean distance, strength measures, degree centrality, and reciprocity.

Usage

```
## S3 method for class 'group_tna'
summary(object, combined = TRUE, ...)
```

Arguments

object	A group_tna object.
combined	A logical indicating whether the summary results should be combined into a single data frame for all clusters (defaults to TRUE)
...	Ignored

Details

The function extracts the igraph network for each cluster and computes the following network metrics:

- Node count: Total number of nodes in the network.
- Edge count: Total number of edges in the network.
- Network density: Proportion of possible edges that are present in the network.
- Mean distance: The average shortest path length between nodes.
- Mean and standard deviation of out-strength and in-strength: Measures of the total weight of outgoing and incoming edges for each node.
- Mean and standard deviation of out-degree: The number of outgoing edges from each node.
- Centralization of out-degree and in-degree: Measures of how centralized the network is based on the degrees of nodes.
- Reciprocity: The proportion of edges that are reciprocated (i.e., mutual edges between nodes).

Value

A summary.group_tna object which is a list of lists or a combined data.frame containing the following network metrics:

- node_count: The total number of nodes.
- edge_count: The total number of edges.
- network_Density: The density of the network.
- mean_distance: The mean shortest path length.
- mean_out_strength: The mean out-strength of nodes.
- sd_out_strength: The standard deviation of out-strength.
- mean_in_strength: The mean in-strength of nodes.
- sd_in_strength: The standard deviation of in-strength.
- mean_out_degree: The mean out-degree of nodes.
- sd_out_degree: The standard deviation of out-degree.
- centralization_out_degree: The centralization of out-degree.
- centralization_in_degree: The centralization of in-degree.
- reciprocity: The reciprocity of the network.

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(), print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna_bootstrap()

Examples

```
group <- c(rep("High", 1000), rep("Low", 1000))
model <- group_model(group_regulation, group = group)
summary(model)
```

---

summary.group_tna_bootstrap

Summarize Bootstrap Results for a Grouped Transition Network

Description

Summarize Bootstrap Results for a Grouped Transition Network

Usage

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

Arguments

object	A group_tna_bootstrap object from bootstrap().
...	Ignored.

Value

A summary.group_tna_bootstrap object containing the weight, p-value and confidence interval of each edge for each cluster.

See Also

Cluster-related functions bootstrap(), centralities(), cliques(), communities(), deprune(), estimate_cs(), group_model(), hist.group_tna(), mmm_stats(), plot.group_tna(), plot.group_tna_centralitie plot.group_tna_cliques(), plot.group_tna_communities(), plot.group_tna_stability(), plot_compare.group_tna(), plot_mosaic.group_tna(), plot_mosaic.tna_data(), print.group_tna(), print.group_tna_bootstrap(), print.group_tna_centralities(), print.group_tna_cliques(),

print.group_tna_communities(), print.group_tna_stability(), print.summary.group_tna(), print.summary.group_tna_bootstrap(), prune(), pruning_details(), rename_groups(), reprune(), summary.group_tna()

Examples

```
model <- group_tna(engagement_mmm)
# Small number of iterations for CRAN
boot <- bootstrap(model, iter = 10)
summary(boot)
```

---

summary.tna *Calculate Summary of Network Metrics for a Transition Network*

Description

This function calculates a variety of network metrics for a tna object. It computes key metrics such as node and edge counts, network density, mean distance, strength measures, degree centrality, and reciprocity.

Usage

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

Arguments

object	A tna object.
...	Ignored

Details

The function extracts the igraph network and computes the following network metrics:

- Node count: Total number of nodes in the network.
- Edge count: Total number of edges in the network.
- Network density: Proportion of possible edges that are present in the network.
- Mean distance: The average shortest path length between nodes.
- Mean and standard deviation of out-strength and in-strength: Measures of the total weight of outgoing and incoming edges for each node.
- Mean and standard deviation of out-degree: The number of outgoing edges from each node.
- Centralization of out-degree and in-degree: Measures of how centralized the network is based on the degrees of nodes.
- Reciprocity: The proportion of edges that are reciprocated (i.e., mutual edges between nodes).

A summary of the metrics is printed to the console.

Value

A named `list` containing the following network metrics (invisibly):

- `node_count`: The total number of nodes.
- `edge_count`: The total number of edges.
- `network_Density`: The density of the network.
- `mean_distance`: The mean shortest path length.
- `mean_out_strength`: The mean out-strength of nodes.
- `sd_out_strength`: The standard deviation of out-strength.
- `mean_in_strength`: The mean in-strength of nodes.
- `sd_in_strength`: The standard deviation of in-strength.
- `mean_out_degree`: The mean out-degree of nodes.
- `sd_out_degree`: The standard deviation of out-degree.
- `centralization_out_degree`: The centralization of out-degree.
- `centralization_in_degree`: The centralization of in-degree.
- `reciprocity`: The reciprocity of the network.

Examples

```
model <- tna(group_regulation)
summary(model)
```

`summary.tna_bootstrap` *Summarize Bootstrap Results*

Description

Summarize Bootstrap Results

Usage

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

Arguments

`object`	A `tna_bootstrap` object from `bootstrap()`.
`...`	Ignored.

Value

A `summary.tna_bootstrap` object containing the weight, p-value and confidence interval of each edge.

Examples

```
model <- tna(group_regulation)
# Small number of iterations for CRAN
boot <- bootstrap(model, iter = 50)
summary(boot)
```

Index

∗ **clusters**
bootstrap, 6
centralities, 12
cliques, 14
communities, 15
deprune, 19
estimate_cs, 22
group_model, 25
hist.group_tna, 29
mmm_stats, 31
plot.group_tna, 33
plot.group_tna_centralities, 34
plot.group_tna_cliques, 35
plot.group_tna_communities, 36
plot.group_tna_stability, 37
plot_compare.group_tna, 47
plot_mosaic.group_tna, 48
plot_mosaic.tna_data, 49
print.group_tna, 53
print.group_tna_bootstrap, 54
print.group_tna_centralities, 55
print.group_tna_cliques, 56
print.group_tna_communities, 57
print.group_tna_stability, 58
print.summary.group_tna, 59
print.summary.group_tna_bootstrap, 60
prune, 68
pruning_details, 70
rename_groups, 71
reprune, 71
summary.group_tna, 72
summary.group_tna_bootstrap, 74

∗ **core**
build_model, 9
centralities, 12
plot.tna, 38
plot.tna_centralities, 39
plot_compare, 46

∗ **datasets**
engagement, 20
engagement_mmm, 21
group_regulation, 28

∗ **evaluation**
bootstrap, 6
permutation_test, 32
prune, 68
pruning_details, 70

∗ **examples**
engagement, 20
engagement_mmm, 21
group_regulation, 28

∗ **patterns**
communities, 15
plot.tna_communities, 42
plot.tna_comparison, 43

as.igraph.group_tna, 4
as.igraph.matrix, 5
as.igraph.tna, 5

base::rank(), *10*, *26*
betweenness_network, 6
bootstrap, 6, *14, 15*, *17*, *20*, *24*, *27*, *29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–60*, *69–72*, *74*
bootstrap(), *69*, *74*, *76*
build_model, 9, *14*, *39, 40*, *46*

centralities, *8*, *12*, 12, *15*, *17*, *20*, *24*, *27*, *29*, *31*, *33*, *35–37*, *39, 40*, *46, 47*, *49, 50*, *53–60*, *69–72*, *74*
centralities(), *23*, *32*, *39*
cliques, *8*, *14*, 14, *17*, *20*, *24*, *27*, *29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–60*, *69–72*, *74*
communities, *8*, *14, 15*, 15, *20*, *24*, *27*, *29*, *31*, *33*, *35–37*, *42*, *44*, *47*, *49, 50*, *53–60*, *69–72*, *74*
communities(), *42*

compare, 17
compare.group_tna, 19
compare.tna(), *19*
ctna *(*build_model*)*, 9

deprune, *8*, *14, 15*, *17*, 19, *24*, *27*, *29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–60*, *69–72*, *74*
deprune(), *69*

engagement, 20, *21*, *28*
engagement_mmm, *21*, 21, *28*
estimate_centrality_stability *(*estimate_cs*)*, 22
estimate_cs, *8*, *14, 15*, *17*, *20*, 22, *27*, *29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–60*, *69–72*, *74*

ftna *(*build_model*)*, 9

graphics::hist(), *29, 30*
group_ctna *(*group_model*)*, 25
group_ftna *(*group_model*)*, 25
group_model, *8*, *14, 15*, *17*, *20*, *24*, 25, *29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–60*, *69–72*, *74*
group_regulation, *21*, 28
group_tna *(*group_model*)*, 25

hist.group_tna, *8*, *14, 15*, *17*, *20*, *24*, *27*, 29, *31*, *33*, *35–37*, *47*, *49, 50*, *53–60*, *69–72*, *74*
hist.tna, 30

igraph::betweenness(), *13*
igraph::closeness(), *13*
igraph::strength(), *13*

mmm_stats, *8*, *14, 15*, *17*, *20*, *24*, *27*, *29*, 31, *33*, *35–37*, *47*, *49, 50*, *53–60*, *69–72*, *74*

permutation_test, *8*, 32, *69, 70*
plot.group_tna, *8*, *14, 15*, *17*, *20*, *24*, *27*, *29*, *31*, 33, *35–37*, *47*, *49, 50*, *53–60*, *69–72*, *74*
plot.group_tna_centralities, *8*, *14, 15*, *17*, *20*, *24*, *27*, *29*, *31*, *33*, 34, *35–37*, *47*, *49, 50*, *53–60*, *69–72*, *74*
plot.group_tna_cliques, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35*, 35, *36, 37*, *47*, *49, 50*, *53–60*, *69–72*, *74*
plot.group_tna_communities, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35*, 36, *37*, *47*, *49, 50*, *53–60*, *69–72*, *74*
plot.group_tna_stability, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35, 36*, 37, *47*, *49, 50*, *53–60*, *69–72*, *74*
plot.tna, *12*, *14*, 38, *40*, *46*
plot.tna(), *33*
plot.tna_centralities, *12*, *14*, *39*, 39, *46*
plot.tna_cliques, 40
plot.tna_cliques(), *35*
plot.tna_communities, *17*, 42, *44*
plot.tna_communities(), *36*
plot.tna_comparison, *17*, *42*, 43
plot.tna_permutation, 44
plot.tna_stability, 45
plot.tna_stability(), *37*
plot_compare, *12*, *14*, *39, 40*, 46
plot_compare.group_tna, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, 47, *49, 50*, *53–60*, *69–72*, *74*
plot_compare.tna(), *47*
plot_model(), *44*
plot_mosaic, 48
plot_mosaic.group_tna, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, 48, *50*, *53–60*, *69–72*, *74*
plot_mosaic.tna_data, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49*, 49, *53–60*, *69–72*, *74*
prepare_data, 50
prepare_data(), *10*
pretty, *30*
print.group_tna, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49, 50*, 53, *54–60*, *69–72*, *74*
print.group_tna_bootstrap, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53*, 54, *55–60*, *69–72*, *74*
print.group_tna_centralities, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53, 54*, 55, *56–60*, *69–72*, *74*
print.group_tna_cliques, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–55*, 56, *57–60*, *69–72*, *74*
print.group_tna_communities, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*,

47, *49, 50*, *53–56*, 57, *58–60*, *69–72*, *74, 75*
print.group_tna_stability, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–57*, 58, *59, 60*, *69–72*, *74, 75*
print.summary.group_tna, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–58*, 59, *60*, *69–72*, *74, 75*
print.summary.group_tna_bootstrap, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–59*, 60, *69–72*, *74, 75*
print.summary.tna, 61
print.summary.tna(), *59*
print.summary.tna_bootstrap, 61
print.tna, 62
print.tna(), *53*
print.tna_bootstrap, 63
print.tna_bootstrap(), *54*
print.tna_centralities, 63
print.tna_cliques, 64
print.tna_cliques(), *56*
print.tna_communities, 65
print.tna_communities(), *57*
print.tna_comparison, 65
print.tna_data, 66
print.tna_permutation, 67
print.tna_stability, 67
print.tna_stability(), *58*
prune, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–60*, 68, *70–72*, *74, 75*
prune(), *38*
pruning_details, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–60*, *69*, 70, *71, 72*, *74, 75*
pruning_details(), *69*

qgraph::qgraph(), *38, 39*, *41, 42*, *44*, *46*

rename_groups, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–60*, *69, 70*, 71, *72*, *74, 75*
reprune, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–60*, *69–71*, 71, *74, 75*
reprune(), *72*
reprune.tna *(*deprune*)*, 19

stats::cor(), *23*
stats::ecdf, *18*
stats::mad, *18*
summary.group_tna, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–60*, *69–72*, 72, *75*
summary.group_tna_bootstrap, *8*, *14, 15*, *17*, *20*, *24, 25*, *27–29*, *31*, *33*, *35–37*, *47*, *49, 50*, *53–60*, *69–72*, *74*, 74
summary.tna, 75
summary.tna_bootstrap, 76

tidyr::pivot_wider(), *51*
tna *(*build_model*)*, 9
tna(), *38*
tna-package, 3