

Package ‘tna’

May 26, 2025

Title Transition Network Analysis (TNA)

Version 0.5.0

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](https://doi.org/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, tidyselect

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

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

LazyData true

Config/Needs/website rmarkdown

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 2025-05-26 08:50:02 UTC

Contents

tna-package	3
as.igraph.group_tna	4
as.igraph.matrix	5
as.igraph.tna	6
betweenness_network	6
bootstrap	7
build_model	9
centralities	13
cliques	14
communities	16
compare	17
compare.group_tna	19
deprune	20
engagement	21
engagement_mmm	21
estimate_cs	22
group_model	25
group_regulation	28
group_regulation_long	28
hist.group_tna	29
hist.tna	30
import_data	31
mmm_stats	32
permutation_test	33
permutation_test.group_tna	34
plot.group_tna	36
plot.group_tna_bootstrap	37
plot.group_tna_centralities	37
plot.group_tna_cliques	39
plot.group_tna_communities	39
plot.group_tna_permutation	40
plot.group_tna_stability	41
plot.tna	42
plot.tna_bootstrap	44
plot.tna_centralities	44
plot.tna_cliques	46
plot.tna_communities	47
plot.tna_comparison	48
plot.tna_permutation	50
plot.tna_stability	51
plot_compare	52
plot_compare.group_tna	53
plot_frequencies	54
plot_frequencies.group_tna	55
plot_mosaic	56
plot_mosaic.group_tna	57

plot_mosaic.tna_data	58
prepare_data	59
print.group_tna	61
print.group_tna_bootstrap	62
print.group_tna_centralities	63
print.group_tna_cliques	63
print.group_tna_communities	64
print.group_tna_permutation	65
print.group_tna_stability	66
print.summary.group_tna	67
print.summary.group_tna_bootstrap	67
print.summary.tna	68
print.summary.tna_bootstrap	69
print.tna	70
print.tna_bootstrap	71
print.tna_centralities	72
print.tna_cliques	72
print.tna_communities	73
print.tna_comparison	74
print.tna_data	75
print.tna_permutation	76
print.tna_stability	77
prune	78
pruning_details	79
rename_groups	80
reprune	81
simulate.tna	82
summary.group_tna	83
summary.group_tna_bootstrap	84
summary.tna	85
summary.tna_bootstrap	87
Index	88

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.
- Kivimäki, I., Leichot, B., Saramäki, J., Særens, 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/>

Basic functions `build_model()`, `hist.group_tna()`, `hist.tna()`, `import_data()`, `plot.group_tna()`, `plot.tna()`, `plot_frequencies()`, `plot_frequencies.group_tna()`, `plot_mosaic()`, `plot_mosaic.group_tna()`, `plot_mosaic.tna_data()`, `prepare_data()`, `print.group_tna()`, `print.summary.group_tna()`, `print.summary.tna()`, `print.tna()`, `print.tna_data()`, `simulate.tna()`, `summary.group_tna()`, `summary.tna()`

as.igraph.group_tna	Coerce a Specific Group from a group_tna Object into an igraph Object.
---------------------	--

Description

Coerce a Specific Group from a group_tna Object into 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.

See Also

Helper functions [as.igraph.matrix\(\)](#), [as.igraph.tna\(\)](#)

as.igraph.matrix	<i>Coerce a Weight Matrix into an igraph Object.</i>
------------------	--

Description

Coerce a Weight Matrix into an igraph Object.

Usage

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

Arguments

x	A matrix of edge weights.
mode	Character scalar, specifies how igraph should interpret the supplied matrix. See also the weighted argument, the interpretation depends on that too. Possible values are: directed, undirected, upper, lower, max, min, plus. See details below.
...	Ignored.

Value

An igraph object.

See Also

Helper functions [as.igraph.group_tna\(\)](#), [as.igraph.tna\(\)](#)

<code>as.igraph.tna</code>	<i>Coerce a tna Object into an igraph Object.</i>
----------------------------	---

Description

Coerce a tna Object into an igraph Object.

Usage

```
## S3 method for class 'tna'
as.igraph(x, mode = "directed", ...)
```

Arguments

<code>x</code>	A tna object.
<code>mode</code>	Character scalar, specifies how igraph should interpret the supplied matrix. See also the weighted argument, the interpretation depends on that too. Possible values are: directed, undirected, upper, lower, max, min, plus. See details below.
<code>...</code>	Ignored.

Value

An igraph object.

See Also

Helper functions [as.igraph.group_tna\(\)](#), [as.igraph.matrix\(\)](#)

<code>betweenness_network</code>	<i>Build and Visualize a Network with Edge Betweenness</i>
----------------------------------	--

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.

See Also

Centrality measure functions `centralities()`, `plot.group_tna_centralities()`, `plot.tna_centralities()`, `print.group_tna_centralities()`, `print.tna_centralities()`

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,
  iter = 1000,
```

```

    level = 0.05,
    method = "stability",
    threshold,
    consistency_range = c(0.75, 1.25),
    ...
  )

```

Arguments

<code>x</code>	A tna or a group_tna object created from sequence data.
<code>...</code>	Ignored.
<code>iter</code>	An integer specifying the number of bootstrap samples to draw. Defaults to 1000.
<code>level</code>	A numeric value representing the significance level for hypothesis testing and confidence intervals. Defaults to 0.05.
<code>method</code>	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 <code>consistency_range</code> . 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 $(1 - \text{level}) * 100\%$ of the bootstrap samples. The "threshold" option instead compares the edge weights against a user-specified threshold value.
<code>threshold</code>	A numeric value to compare edge weights against. The default is the 10th percentile of the edge weights. Used only when <code>method = "threshold"</code> .
<code>consistency_range</code>	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 <code>c(0.75, 1.25)</code> which corresponds to a symmetric 25% deviation range. Used only when <code>method = "stability"</code> .

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 estimates p-values for each edge and identifies significant edges based on the specified significance level. A matrix of significant edges (those with estimated 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 estimated 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.
- `cr_lower`: The lower bound matrix of the consistency range for the edge weights.
- `cr_upper`: The upper bound matrix of the consistency range for the edge weights.
- `ci_lower`: The lower bound matrix of the bootstrap confidence intervals for the edge weights.
- `ci_upper`: The upper bound matrix of the bootstrap confidence intervals for the edge weights.
- `p_values`: The matrix of estimated p-values for the edge weights.
- `summary`: A `data.frame` summarizing the edges, their weights, p-values, statistical significance, consistency ranges, 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

Validation functions `deprune()`, `estimate_cs()`, `permutation_test()`, `permutation_test.group_tna()`, `plot.group_tna_bootstrap()`, `plot.group_tna_permutation()`, `plot.group_tna_stability()`, `plot.tna_bootstrap()`, `plot.tna_permutation()`, `plot.tna_stability()`, `print.group_tna_bootstrap()`, `print.group_tna_permutation()`, `print.group_tna_stability()`, `print.summary.group_tna_bootstrap()`, `print.summary.tna_bootstrap()`, `print.tna_bootstrap()`, `print.tna_permutation()`, `print.tna_stability()`, `prune()`, `pruning_details()`, `reprune()`, `summary.group_tna_bootstrap()`, `summary.tna_bootstrap()`

Examples

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

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, ...)  
  
atna(x, ...)
```

Arguments

x	A <code>stslist</code> (from <code>TraMineR</code>), <code>data.frame</code> , a <code>matrix</code> , or a <code>tna_data</code> object (see prepare_data()). For <code>stslist</code> and <code>data.frame</code> objects <code>x</code> should describe a sequence of events or states to be used for building the Markov model. If <code>x</code> is a <code>matrix</code> , it is assumed that the element on row <code>i</code> and column <code>j</code> is the weight of the edge representing the transition from state <code>i</code> to state <code>j</code> . If <code>x</code> is a <code>data.frame</code> , then it must be in wide format (see <code>cols</code> 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 <code>n</code> 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. • "attention" aggregates all downstream pairs of states with an exponential decay for the gap between states. The parameter <code>lambda</code> can be used to control the decay rate (the default is 1)-
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, <code>c("rank", "minmax")</code> 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 <code>base::rank()</code> with <code>ties.method = "average"</code>.
...	Ignored. For the <code>build_model</code> aliases (e.g., <code>tna</code>), this argument matches the actual arguments to <code>build_model</code> beside <code>x</code> .
inits	An optional numeric vector of initial state probabilities for each state. Can be provided only if <code>x</code> is a <code>matrix</code> . 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.
 - `lambda`: A numeric value for the decay rate. The default is 1.
- `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.

See Also

Basic functions `hist.group_tna()`, `hist.tna()`, `import_data()`, `plot.group_tna()`, `plot.tna()`, `plot_frequencies()`, `plot_frequencies.group_tna()`, `plot_mosaic()`, `plot_mosaic.group_tna()`, `plot_mosaic.tna_data()`, `prepare_data()`, `print.group_tna()`, `print.summary.group_tna()`, `print.summary.tna()`, `print.tna()`, `print.tna_data()`, `simulate.tna()`, `summary.group_tna()`, `summary.tna()`, `tna-package`

Examples

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

model <- tna(group_regulation)

model <- ftna(group_regulation)

model <- cttna(group_regulation)

model <- attna(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

Centrality measure functions `betweenness_network()`, `plot.group_tna_centralities()`, `plot.tna_centralities()`, `print.group_tna_centralities()`, `print.tna_centralities()`

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, i.e., 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

<code>x</code>	A <code>tna</code> or a <code>group_tna</code> object.
<code>...</code>	Ignored.
<code>size</code>	An integer specifying the size of the cliques to identify. Defaults to 2 (dyads).
<code>threshold</code>	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.
<code>sum_weights</code>	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 of `tna_cliques` objects.

See Also

Clique-related functions [plot.group_tna_cliques\(\)](#), [plot.tna_cliques\(\)](#), [print.group_tna_cliques\(\)](#), [print.tna_cliques\(\)](#)

Examples

```
model <- tna(group_regulation)

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

model <- group_tna(engagement_mmm)
cliques(model)
```

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, gamma = 1, ...)
```

Arguments

<code>x</code>	A <code>tna</code> or a <code>group_tna</code> object.
<code>...</code>	Ignored.
<code>methods</code>	A character vector of community detection algorithms to apply to the network. The supported options are: • <code>"walktrap"</code>: A community detection method using short random walks. • <code>"fast_greedy"</code>: A method based on modularity optimization. • <code>"label_prop"</code>: A method that uses label propagation. • <code>"infomap"</code>: A method that uses information flow to detect communities. • <code>"edge_betweenness"</code>: A method that uses edge betweenness to find communities. • <code>"leading_eigen"</code>: A method using the leading eigenvector of the modularity matrix. • <code>"spinglass"</code>: A method based on the spinglass model. If not provided, all methods are applied.
<code>gamma</code>	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

Community detection functions `plot.group_tna_communities()`, `plot.tna_communities()`, `print.group_tna_communities()`, `print.tna_communities()`

Cluster-related functions `group_model()`, `mmm_stats()`, `rename_groups()`

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

<code>x</code>	A tna object or a matrix of weights.
<code>...</code>	Ignored.
<code>y</code>	A tna object or a matrix of weights.
<code>scaling</code>	A character string naming a scaling method to apply to the weights before comparing them. The supported options are: • <code>"none"</code>: No scaling is performed. The weights are used as is. • <code>"minmax"</code>: Performs min-max normalization, i.e., the minimum value is subtracted and the differences are scaled by the range. • <code>"rank"</code>: Applies min-max normalization to the ranks of the weights (computed with <code>ties.method = "average"</code>). • <code>"zscore"</code>: Computes the standard score, i.e. the mean weight is subtracted and the differences are scaled by the standard deviation. • <code>"robust"</code>: Computes the robust z-score, i.e. the median weight is subtracted and the differences are scaled by the median absolute deviation (using <code>stats::mad</code>). • <code>"log"</code>: Simply the natural logarithm of the weights. • <code>"log1p"</code>: As above, but adds 1 to the values before taking the logarithm. Useful for scenarios with zero weights. • <code>"softmax"</code>: Performs softmax normalization. • <code>"quantile"</code>: Uses the empirical quantiles of the weights via <code>stats::ecdf</code>.

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 computed from `x` and `y`.
- `centrality_correlations`: A numeric vector of correlations of the centrality measures between `x` and `y`.

See Also

Model comparison functions `compare.group_tna()`, `plot.tna_comparison()`, `plot_compare()`, `plot_compare.group_tna()`, `print.tna_comparison()`

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	<i>Compare TNA Clusters with Comprehensive Metrics</i>
-------------------	--

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.

See Also

Model comparison functions [compare\(\)](#), [plot.tna_comparison\(\)](#), [plot_compare\(\)](#), [plot_compare.group_tna\(\)](#), [print.tna_comparison\(\)](#)

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

Validation functions [bootstrap\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_permutation\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.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](https://doi.org/10.1016/j.compedu.2023.104787)

See Also

Datasets [engagement_mmm](#), [group_regulation](#), [group_regulation_long](#)

`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

Datasets [engagement](#), [group_regulation](#), [group_regulation_long](#)

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,
  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,
  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,
  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,
  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.
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.

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

Validation functions `bootstrap()`, `deprune()`, `permutation_test()`, `permutation_test.group_tna()`, `plot.group_tna_bootstrap()`, `plot.group_tna_permutation()`, `plot.group_tna_stability()`, `plot.tna_bootstrap()`, `plot.tna_permutation()`, `plot.tna_stability()`, `print.group_tna_bootstrap()`, `print.group_tna_permutation()`, `print.group_tna_stability()`, `print.summary.group_tna_bootstrap()`, `print.summary.tna_bootstrap()`, `print.tna_bootstrap()`, `print.tna_permutation()`, `print.tna_stability()`, `prune()`, `pruning_details()`, `reprune()`, `summary.group_tna_bootstrap()`, `summary.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-format dataframe, or a mixture Markov model.

Usage

```
group_model(x, ...)

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

## S3 method for class 'mhmm'
group_model(
  x,
  type = "relative",
  scaling = character(0L),
  groupwise = FALSE,
  cols,
  params = list(),
  na.rm = TRUE,
  ...
)

group_tna(x, ...)

group_ftna(x, ...)

group_ctna(x, ...)

group_atna(x, ...)
```

Arguments

x	An <code>stslst</code> object describing a sequence of events or states to be used for building the Markov model. The argument <code>x</code> also accepts <code>data.frame</code> objects in wide format, and <code>tna_data</code> objects. Alternatively, the function accepts a mixture Markov model from <code>seqHMM</code> .
...	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 <code>x</code> . Alternatively, a single character string giving the column name of the data that defines the group when <code>x</code> is a wide format <code>data.frame</code> or a <code>tna_data</code> object. If not provided, each row of the data forms a cluster.
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 <code>n</code> 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. • "attention" aggregates all downstream pairs of states with an exponential decay for the gap between states. The parameter <code>lambda</code> can be used to control the decay rate (the default is 1)-
scaling	A character vector describing how to scale the weights defined by <code>type</code>. When a vector is provided, the scaling options are applied in the respective order. For example, <code>c("rank", "minmax")</code> 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 <code>base::rank()</code> with <code>ties.method = "average"</code>.
groupwise	A logical value that indicates whether scaling methods should be applied by group (TRUE) or globally (FALSE, the default).

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., <code>seq(1, ncol(x))</code> . The columns are automatically determined for <code>tna_data</code> objects.
params	A list of additional arguments for models of specific type. The potential elements of this list are: • <code>n_gram</code>: An integer for n-gram transitions specifying the number of adjacent events. The default value is 2. • <code>max_gap</code>: An integer for the gap-allowed transitions specifying the largest allowed gap size. The default is 1. • <code>window_size</code>: An integer for the sliding window transitions specifying the window size. The default is 2. • <code>weighted</code>: 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. • <code>lambda</code>: A numeric value for the decay rate. The default is 1.
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.

See Also

Cluster-related functions [communities\(\)](#), [mmm_stats\(\)](#), [rename_groups\(\)](#)

Examples

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

# Groups defined by a mixed Markov model
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))

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

group_regulation	<i>Example Wide Data on Group Regulation</i>
------------------	--

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

Datasets [engagement](#), [engagement_mmm](#), [group_regulation_long](#)

group_regulation_long	<i>Example Long Data on Group Regulation</i>
-----------------------	--

Description

Students’ regulation during collaborative learning. This is the same dataset as group_regulation but in long format. In addition to students’ actions (Action), it contains the student identifier (Actor), timestamp (Time), Course name, and collaboration Group. It also includes a column (Achiever) indicating whether the student is a high or low achiever.

Usage

group_regulation_long

Format

A data.frame object.

Source

The data was generated synthetically from group_regulation

See Also

Datasets [engagement](#), [engagement_mmm](#), [group_regulation](#)

hist.group_tna	<i>Plot a Histogram of Edge Weights for a group_tna Object.</i>
----------------	---

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

Basic functions [build_model\(\)](#), [hist.tna\(\)](#), [import_data\(\)](#), [plot.group_tna\(\)](#), [plot.tna\(\)](#), [plot_frequencies\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic\(\)](#), [plot_mosaic.group_tna\(\)](#), [plot_mosaic.tna_data\(\)](#), [prepare_data\(\)](#), [print.group_tna\(\)](#), [print.summary.group_tna\(\)](#), [print.summary.tna\(\)](#), [print.tna\(\)](#), [print.tna_data\(\)](#), [simulate.tna\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

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.

See Also

Basic functions [build_model\(\)](#), [hist.group_tna\(\)](#), [import_data\(\)](#), [plot.group_tna\(\)](#), [plot.tna\(\)](#), [plot_frequencies\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic\(\)](#), [plot_mosaic.group_tna\(\)](#), [plot_mosaic.tna_data\(\)](#), [prepare_data\(\)](#), [print.group_tna\(\)](#), [print.summary.group_tna\(\)](#), [print.summary.tna\(\)](#), [print.tna\(\)](#), [print.tna_data\(\)](#), [simulate.tna\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

Examples

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

import_data

Import Wide Format Sequence Data as Long Format Sequence Data

Description

This function transforms wide format data where features are in separate columns into a long format suitable for sequence analysis. It creates windows of data based on row order and generates sequence order within these windows.

Usage

```
import_data(data, cols, id_cols, window_size = 1, replace_zeros = TRUE)
```

Arguments

data	A <code>data.frame</code> in wide format.
cols	An expression giving a tidy selection of column names to be transformed into long format (actions). This can be a vector of column names (e.g., <code>c(feature1, feature2)</code>) or a range specified as <code>feature1:feature6</code> (without quotes) to include all columns from 'feature1' to 'feature6' in the order they appear in the data frame. For more information on tidy selections, see dplyr::select() .
id_cols	A character vector of column names that uniquely identify each observation (IDs).
window_size	An integer specifying the size of the window for sequence grouping. Default is 1 (each row is a separate window).
replace_zeros	A logical value indicating whether to replace 0s in cols with NA. The default is TRUE.

Value

A `data.frame` in long format with added columns for window and sequence order.

See Also

Basic functions [build_model\(\)](#), [hist.group_tna\(\)](#), [hist.tna\(\)](#), [plot.group_tna\(\)](#), [plot.tna\(\)](#), [plot_frequencies\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic\(\)](#), [plot_mosaic.group_tna\(\)](#), [plot_mosaic.tna_data\(\)](#), [prepare_data\(\)](#), [print.group_tna\(\)](#), [print.summary.group_tna\(\)](#), [print.summary.tna\(\)](#), [print.tna\(\)](#), [print.tna_data\(\)](#), [simulate.tna\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

Examples

```
data <- data.frame(
  ID = c("A", "A", "B", "B"),
  Time = c(1, 2, 1, 2),
  feature1 = c(10, 0, 15, 20),
  feature2 = c(5, 8, 0, 12),
  feature3 = c(2, 4, 6, 8),
  other_col = c("X", "Y", "Z", "W")
)

# Using a vector
long_data1 <- import_data(
  data = data,
  cols = c(feature1, feature2),
  id_cols = c("ID", "Time"),
  window_size = 2,
  replace_zeros = TRUE
)

# Using a column range
long_data2 <- import_data(
  data = data,
  cols = feature1:feature3,
  id_cols = c("ID", "Time"),
  window_size = 2,
  replace_zeros = TRUE
)
```

mmm_stats	<i>Retrieve Statistics from a Mixture Markov Model (MMM)</i>
-----------	--

Description

Retrieve Statistics from a Mixture Markov Model (MMM)

Usage

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

Arguments

x	A mhm 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 [communities\(\)](#), [group_model\(\)](#), [rename_groups\(\)](#)

Examples

```
mmm_stats(engagement_mmm)
```

permutation_test	<i>Compare Two Networks from Sequence Data using a Permutation Test</i>
------------------	---

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, estimated p-values, and confidence intervals.

Usage

```
permutation_test(x, ...)

## S3 method for class 'tna'
permutation_test(
  x,
  y,
  adjust = "none",
  iter = 1000,
  paired = FALSE,
  level = 0.05,
  measures = character(0),
  ...
)
```

Arguments

x	A tna object containing sequence data for the first tna model.
...	Additional arguments passed to centralities() .
y	A tna object containing sequence data for the second tna model.
adjust	A character string for the method to adjust p-values with for multiple comparisons. The default is "none" for no adjustment. See stats::p.adjust() for details and available adjustment methods.

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.

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 estimated 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

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_permutation\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.tna_bootstrap\(\)](#)

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)
```

`permutation_test.group_tna`

Compare Networks using a Permutation Test

Description

Test edge weight differences between all pairs or a subset of pairs of a `group_tna` object. See [permutation_test.tna\(\)](#) for more details.

Usage

```
## S3 method for class 'group_tna'
permutation_test(
  x,
  groups,
  adjust = "none",
  iter = 1000,
  paired = FALSE,
  level = 0.05,
  measures = character(0),
  ...
)
```

Arguments

x	A group_tna object
groups	An integer vector or a character vector of group indices or names, respectively, defining which groups to compare. When not provided, all pairs are compared (the default).
adjust	A character string for the method to adjust p-values with for multiple comparisons. The default is "none" for no adjustment. See stats::p.adjust() for details and available adjustment methods.
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() .

See Also

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_permutation\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.tna_bootstrap\(\)](#)

Examples

```
model <- group_model(engagement_mmm)
# Small number of iterations for CRAN
permutation_test(model, iter = 20)
```

plot.group_tna	<i>Plot a Grouped Transition Network Analysis Model</i>
----------------	---

Description

Plots a transition network of each cluster using qgraph.

Usage

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

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)
which	An optional integer vector of groups to plot. By default, all groups are plotted.
...	Same as plot.tna() .

Value

NULL (invisibly).

See Also

Basic functions [build_model\(\)](#), [hist.group_tna\(\)](#), [hist.tna\(\)](#), [import_data\(\)](#), [plot.tna\(\)](#), [plot_frequencies\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic\(\)](#), [plot_mosaic.group_tna\(\)](#), [plot_mosaic.tna_data\(\)](#), [prepare_data\(\)](#), [print.group_tna\(\)](#), [print.summary.group_tna\(\)](#), [print.summary.tna\(\)](#), [print.tna\(\)](#), [print.tna_data\(\)](#), [simulate.tna\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

Examples

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

plot.group_tna_bootstrap

Plot a Bootstrapped Grouped Transition Network Analysis Model

Description

Plot a Bootstrapped Grouped Transition Network Analysis Model

Usage

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

Arguments

`x` A group_tna_bootstrap object.
`...` Additional arguments passed to [plot.tna\(\)](#).

See Also

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_permutation\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.tna_bootstrap\(\)](#)

Examples

```
model <- group_model(engagement_mmm)
# Small number of iterations for CRAN
boot <- bootstrap(model, iter = 50)
plot(boot)
```

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

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

Value

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

See Also

Centrality measure functions `betweenness_network()`, `centralities()`, `plot.tna_centralities()`, `print.group_tna_centralities()`, `print.tna_centralities()`

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

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

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

Clique-related functions `cliques()`, `plot.tna_cliques()`, `print.group_tna_cliques()`, `print.tna_cliques()`

Examples

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

`plot.group_tna_communities`*Plot Detected Communities*

Description

Plot Detected Communities

Usage

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

Arguments

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

Value

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

See Also

Community detection functions `communities()`, `plot.tna_communities()`, `print.group_tna_communities()`, `print.tna_communities()`

Examples

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

```
plot.group_tna_permutation
```

Plot Permutation Test Results

Description

Plot Permutation Test Results

Usage

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

Arguments

<code>x</code>	A <code>group_tna_permutation</code> object.
<code>title</code>	An optional character vector of titles for each plot. When not provided, the title shows the names of the clusters being contrasted.
<code>...</code>	Arguments passed to <code>plot.tna_permutation()</code> .

Value

A list (invisibly) of `qgraph` objects depicting the significant difference between each pair.

See Also

Validation functions `bootstrap()`, `deprune()`, `estimate_cs()`, `permutation_test()`, `permutation_test.group_tna()`, `plot.group_tna_bootstrap()`, `plot.group_tna_stability()`, `plot.tna_bootstrap()`, `plot.tna_permutation()`, `plot.tna_stability()`, `print.group_tna_bootstrap()`, `print.group_tna_permutation()`, `print.group_tna_stability()`, `print.summary.group_tna_bootstrap()`, `print.summary.tna_bootstrap()`, `print.tna_bootstrap()`, `print.tna_permutation()`, `print.tna_stability()`, `prune()`, `pruning_details()`, `reprune()`, `summary.group_tna_bootstrap()`, `summary.tna_bootstrap()`

Examples

```
model <- group_tna(engagement_mmm)
# Small number of iterations for CRAN
perm <- permutation_test(model, iter = 20)
plot(perm)
```

```
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

Validation functions `bootstrap()`, `deprune()`, `estimate_cs()`, `permutation_test()`, `permutation_test.group_tna()`, `plot.group_tna_bootstrap()`, `plot.group_tna_permutation()`, `plot.tna_bootstrap()`, `plot.tna_permutation()`, `plot.tna_stability()`, `print.group_tna_bootstrap()`, `print.group_tna_permutation()`, `print.group_tna_stability()`, `print.summary.group_tna_bootstrap()`, `print.summary.tna_bootstrap()`, `print.tna_bootstrap()`, `print.tna_permutation()`, `print.tna_stability()`, `prune()`, `pruning_details()`, `reprune()`, `summary.group_tna_bootstrap()`, `summary.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,
  cut,
  show_pruned = TRUE,
  pruned_edge_color = "pink",
  edge.color = NA,
  edge.labels = TRUE,
  edge.label.position = 0.65,
  layout = "circle",
  layout_args = list(),
  mar = rep(5, 4),
  theme = "colorblind",
  ...
)
```

Arguments

<code>x</code>	A <code>tna</code> object from <code>tna()</code> .
<code>labels</code>	See <code>qgraph::qgraph()</code> .
<code>colors</code>	See <code>qgraph::qgraph()</code> .
<code>pie</code>	See <code>qgraph::qgraph()</code> .

cut	Edge color and width emphasis cutoff value. The default is the median of the edge weights. See <code>qgraph::qgraph()</code> for details.
show_pruned	A logical value indicating if pruned edges removed by <code>prune()</code> 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 <code>show_pruned = TRUE</code> . The default is "pink".
edge.color	See <code>qgraph::qgraph()</code> .
edge.labels	See <code>qgraph::qgraph()</code> .
edge.label.position	See <code>qgraph::qgraph()</code> .
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 <code>qgraph::qgraph()</code> .
theme	See <code>qgraph::qgraph()</code> .
...	Additional arguments passed to <code>qgraph::qgraph()</code> .

Value

A qgraph plot of the transition network.

See Also

Basic functions `build_model()`, `hist.group_tna()`, `hist.tna()`, `import_data()`, `plot.group_tna()`, `plot_frequencies()`, `plot_frequencies.group_tna()`, `plot_mosaic()`, `plot_mosaic.group_tna()`, `plot_mosaic.tna_data()`, `prepare_data()`, `print.group_tna()`, `print.summary.group_tna()`, `print.summary.tna()`, `print.tna()`, `print.tna_data()`, `simulate.tna()`, `summary.group_tna()`, `summary.tna()`, `tna-package`

Examples

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

plot.tna_bootstrap *Plot a Bootstrapped Transition Network Analysis Model*

Description

Plot a Bootstrapped Transition Network Analysis Model

Usage

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

Arguments

x A tna_bootstrap object.
 ... Additional arguments passed to [plot.tna\(\)](#).

See Also

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_permutation\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.tna_bootstrap\(\)](#)

Examples

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

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

Centrality measure functions [betweenness_network\(\)](#), [centralities\(\)](#), [plot.group_tna_centralities\(\)](#), [print.group_tna_centralities\(\)](#), [print.tna_centralities\(\)](#)

Examples

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

plot.tna_cliques	<i>Plot Cliques of a TNA Network</i>
------------------	--------------------------------------

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 = 1e-05,
  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).

See Also

Clique-related functions [cliques\(\)](#), [plot.group_tna_cliques\(\)](#), [print.group_tna_cliques\(\)](#), [print.tna_cliques\(\)](#)

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 = "spring", ...)
```

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 <code>communities()</code> for details.
...	Additional arguments passed to <code>qgraph::qgraph()</code> .

Value

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

See Also

Community detection functions `communities()`, `plot.group_tna_communities()`, `print.group_tna_communities()`, `print.tna_communities()`

Examples

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

plot.tna_comparison *Plot the Comparison of Two TNA Models or Matrices*

Description

Plot the Comparison of 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

Model comparison functions `compare()`, `compare.group_tna()`, `plot_compare()`, `plot_compare.group_tna()`, `print.tna_comparison()`

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, posCol = "#009900", negCol = "red", ...)
```

Arguments

x	A tna_permutation object.
colors	See qgraph::qgraph() .
posCol	Color for plotting edges the difference in edge weights is positive. See qgraph::qgraph() .
negCol	Color for plotting edges when the the difference in edge weights is negative. See qgraph::qgraph() .
...	Arguments passed to plot_model() .

Value

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

See Also

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_permutation\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.tna_bootstrap\(\)](#)

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	<i>Plot Centrality Stability Results</i>
--------------------	--

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

<code>x</code>	A <code>tna_stability</code> object produced by <code>estimate_cs</code> .
<code>level</code>	A numeric value representing the significance level for the confidence intervals. Defaults to 0.05.
<code>...</code>	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.

See Also

Validation functions `bootstrap()`, `deprune()`, `estimate_cs()`, `permutation_test()`, `permutation_test.group_tna()`, `plot.group_tna_bootstrap()`, `plot.group_tna_permutation()`, `plot.group_tna_stability()`, `plot.tna_bootstrap()`, `plot.tna_permutation()`, `print.group_tna_bootstrap()`, `print.group_tna_permutation()`, `print.group_tna_stability()`, `print.summary.group_tna_bootstrap()`, `print.summary.tna_bootstrap()`, `print.tna_bootstrap()`, `print.tna_permutation()`, `print.tna_stability()`, `prune()`, `pruning_details()`, `reprune()`, `summary.group_tna_bootstrap()`, `summary.tna_bootstrap()`

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 x is greater than y and red indicates otherwise.

Usage

```
plot_compare(x, ...)

## S3 method for class 'tna'
plot_compare(
  x,
  y,
  theme = NULL,
  palette = "colorblind",
  posCol = "#009900",
  negCol = "red",
  ...
)
```

Arguments

x	A tna object. This is the principal model.
...	Additional arguments passed to <code>qgraph::qgraph()</code> .
y	A tna object. This is the model subtracted from the principal model.
theme	See <code>qgraph::qgraph()</code> .
palette	See <code>qgraph::qgraph()</code> .

posCol Color for plotting edges and pie when the first group has a higher value. See [qgraph::qgraph\(\)](#).

negCol Color for plotting edges and pie when the second group has a higher value. See [qgraph::qgraph\(\)](#).

Value

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

See Also

Model comparison functions [compare\(\)](#), [compare.group_tna\(\)](#), [plot.tna_comparison\(\)](#), [plot_compare.group_tna\(\)](#), [print.tna_comparison\(\)](#)

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

Model comparison functions [compare\(\)](#), [compare.group_tna\(\)](#), [plot.tna_comparison\(\)](#), [plot_compare\(\)](#), [print.tna_comparison\(\)](#)

Examples

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

plot_frequencies	<i>Plot the Frequency Distribution of States</i>
------------------	--

Description

Plot the Frequency Distribution of States

Usage

```
plot_frequencies(x, ...)

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

Arguments

x	A tna object created from sequence data.
...	Ignored.

Value

A ggplot object.

See Also

Basic functions [build_model\(\)](#), [hist.group_tna\(\)](#), [hist.tna\(\)](#), [import_data\(\)](#), [plot.group_tna\(\)](#), [plot.tna\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic\(\)](#), [plot_mosaic.group_tna\(\)](#), [plot_mosaic.tna_data\(\)](#), [prepare_data\(\)](#), [print.group_tna\(\)](#), [print.summary.group_tna\(\)](#), [print.summary.tna\(\)](#), [print.tna\(\)](#), [print.tna_data\(\)](#), [simulate.tna\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

Examples

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

`plot_frequencies.group_tna`*Plot the Frequency Distribution of States*

Description

Plot the Frequency Distribution of States

Usage

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

Arguments

<code>x</code>	A <code>group_tna</code> object.
<code>label</code>	An optional character string that can be provided to specify the grouping factor name if <code>x</code> was not constructed using a column name of the original data.
<code>...</code>	Ignored.

Value

A ggplot object.

See Also

Basic functions `build_model()`, `hist.group_tna()`, `hist.tna()`, `import_data()`, `plot.group_tna()`, `plot.tna()`, `plot_frequencies()`, `plot_mosaic()`, `plot_mosaic.group_tna()`, `plot_mosaic.tna_data()`, `prepare_data()`, `print.group_tna()`, `print.summary.group_tna()`, `print.summary.tna()`, `print.tna()`, `print.tna_data()`, `simulate.tna()`, `summary.group_tna()`, `summary.tna()`, [tna-package](#)

Examples

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

plot_mosaic

Create a Mosaic Plot of Transitions or Events

Description

Create a Mosaic Plot of Transitions or Events

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.

See Also

Basic functions [build_model\(\)](#), [hist.group_tna\(\)](#), [hist.tna\(\)](#), [import_data\(\)](#), [plot.group_tna\(\)](#), [plot.tna\(\)](#), [plot_frequencies\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic.group_tna\(\)](#), [plot_mosaic.tna_data\(\)](#), [prepare_data\(\)](#), [print.group_tna\(\)](#), [print.summary.group_tna\(\)](#), [print.summary.tna\(\)](#), [print.tna\(\)](#), [print.tna_data\(\)](#), [simulate.tna\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

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

Basic functions [build_model\(\)](#), [hist.group_tna\(\)](#), [hist.tna\(\)](#), [import_data\(\)](#), [plot.group_tna\(\)](#), [plot.tna\(\)](#), [plot_frequencies\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic\(\)](#), [plot_mosaic.tna_data\(\)](#), [prepare_data\(\)](#), [print.group_tna\(\)](#), [print.summary.group_tna\(\)](#), [print.summary.tna\(\)](#), [print.tna\(\)](#), [print.tna_data\(\)](#), [simulate.tna\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

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

Basic functions [build_model\(\)](#), [hist.group_tna\(\)](#), [hist.tna\(\)](#), [import_data\(\)](#), [plot.group_tna\(\)](#), [plot.tna\(\)](#), [plot_frequencies\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic\(\)](#), [plot_mosaic.group_tna\(\)](#), [prepare_data\(\)](#), [print.group_tna\(\)](#), [print.summary.group_tna\(\)](#), [print.summary.tna\(\)](#), [print.tna\(\)](#), [print.tna_data\(\)](#), [simulate.tna\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

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.

<code>time_threshold</code>	An integer specifying the time threshold in seconds for creating new time-based sessions. Defaults to 900 seconds.
<code>custom_format</code>	A character string giving the format used to parse the time column.
<code>is_unix_time</code>	A logical value indicating whether the time column is in Unix time. The default is FALSE.
<code>unix_time_unit</code>	A character string giving the Unix time unit when <code>is_unix_time</code> is TRUE. The default is "seconds". Valid options are "seconds", "milliseconds", or "microseconds".
<code>unused_fn</code>	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.

See Also

Basic functions [build_model\(\)](#), [hist.group_tna\(\)](#), [hist.tna\(\)](#), [import_data\(\)](#), [plot.group_tna\(\)](#), [plot.tna\(\)](#), [plot_frequencies\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic\(\)](#), [plot_mosaic.group_tna\(\)](#), [plot_mosaic.tna_data\(\)](#), [print.group_tna\(\)](#), [print.summary.group_tna\(\)](#), [print.summary.tna\(\)](#), [print.tna\(\)](#), [print.tna_data\(\)](#), [simulate.tna\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

Examples

```
results <- prepare_data(
  group_regulation_long, actor = "Actor", 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	<i>Print a group_tna Object</i>
-----------------	---------------------------------

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

Basic functions [build_model\(\)](#), [hist.group_tna\(\)](#), [hist.tna\(\)](#), [import_data\(\)](#), [plot.group_tna\(\)](#), [plot.tna\(\)](#), [plot_frequencies\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic\(\)](#), [plot_mosaic.group_tna\(\)](#), [plot_mosaic.tna_data\(\)](#), [prepare_data\(\)](#), [print.summary.group_tna\(\)](#), [print.summary.tna\(\)](#), [print.tna\(\)](#), [print.tna_data\(\)](#), [simulate.tna\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

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

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_permutation\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.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

Centrality measure functions [betweenness_network\(\)](#), [centralities\(\)](#), [plot.group_tna_centralities\(\)](#), [plot.tna_centralities\(\)](#), [print.tna_centralities\(\)](#)

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

Clique-related functions `cliques()`, `plot.group_tna_cliques()`, `plot.tna_cliques()`, `print.tna_cliques()`

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

Community detection functions `communities()`, `plot.group_tna_communities()`, `plot.tna_communities()`, `print.tna_communities()`

Examples

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

```
print.group_tna_permutation
      Print Permutation Test Results
```

Description

Print Permutation Test Results

Usage

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

Arguments

x	A group_tna_permutation object.
...	Arguments passed to print.tna_permutation() .

Value

x (invisibly).

See Also

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.tna_bootstrap\(\)](#)

Examples

```
model <- group_model(engagement_mmm)
# Small number of iterations for CRAN
perm <- permutation_test(model, iter = 20)
print(perm)
```

```
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

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_permutation\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.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 a Summary of a Grouped Transition Network Analysis Model

Description

Print a 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

Basic functions `build_model()`, `hist.group_tna()`, `hist.tna()`, `import_data()`, `plot.group_tna()`, `plot.tna()`, `plot_frequencies()`, `plot_frequencies.group_tna()`, `plot_mosaic()`, `plot_mosaic.group_tna()`, `plot_mosaic.tna_data()`, `prepare_data()`, `print.group_tna()`, `print.summary.tna()`, `print.tna()`, `print.tna_data()`, `simulate.tna()`, `summary.group_tna()`, `summary.tna()`, `tna-package`

Examples

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

```
print.summary.group_tna_bootstrap
```

Print a Bootstrap Summary for a Grouped Transition Network Model

Description

Print a Bootstrap Summary for a Grouped Transition Network Model

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

Validation functions `bootstrap()`, `deprune()`, `estimate_cs()`, `permutation_test()`, `permutation_test.group_tna()`, `plot.group_tna_bootstrap()`, `plot.group_tna_permutation()`, `plot.group_tna_stability()`, `plot.tna_bootstrap()`, `plot.tna_permutation()`, `plot.tna_stability()`, `print.group_tna_bootstrap()`, `print.group_tna_permutation()`, `print.group_tna_stability()`, `print.summary.tna_bootstrap()`, `print.tna_bootstrap()`, `print.tna_permutation()`, `print.tna_stability()`, `prune()`, `pruning_details()`, `reprune()`, `summary.group_tna_bootstrap()`, `summary.tna_bootstrap()`

Examples

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

<code>print.summary.tna</code>	<i>Print a TNA Summary</i>
--------------------------------	----------------------------

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.

See Also

Basic functions [build_model\(\)](#), [hist.group_tna\(\)](#), [hist.tna\(\)](#), [import_data\(\)](#), [plot.group_tna\(\)](#), [plot.tna\(\)](#), [plot_frequencies\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic\(\)](#), [plot_mosaic.group_tna\(\)](#), [plot_mosaic.tna_data\(\)](#), [prepare_data\(\)](#), [print.group_tna\(\)](#), [print.summary.group_tna\(\)](#), [print.tna\(\)](#), [print.tna_data\(\)](#), [simulate.tna\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

Examples

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

```
print.summary.tna_bootstrap
```

Print a Bootstrap Summary

Description

Print a 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, estimated p-value and confidence interval of each edge.

See Also

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_permutation\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.tna_bootstrap\(\)](#)

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 <i>significant</i> 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).

See Also

Basic functions [build_model\(\)](#), [hist.group_tna\(\)](#), [hist.tna\(\)](#), [import_data\(\)](#), [plot.group_tna\(\)](#), [plot.tna\(\)](#), [plot_frequencies\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic\(\)](#), [plot_mosaic.group_tna\(\)](#), [plot_mosaic.tna_data\(\)](#), [prepare_data\(\)](#), [print.group_tna\(\)](#), [print.summary.group_tna\(\)](#), [print.summary.tna\(\)](#), [print.tna_data\(\)](#), [simulate.tna\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

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 <i>significant</i> 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).

See Also

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_permutation\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.tna_bootstrap\(\)](#)

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).

See Also

Centrality measure functions [betweenness_network\(\)](#), [centralities\(\)](#), [plot.group_tna_centralities\(\)](#), [plot.tna_centralities\(\)](#), [print.group_tna_centralities\(\)](#)

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 <i>significant</i> digits to print.
...	Ignored.

Value

x (invisibly).

See Also

Clique-related functions [cliques\(\)](#), [plot.group_tna_cliques\(\)](#), [plot.tna_cliques\(\)](#), [print.group_tna_cliques\(\)](#)

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).

See Also

Community detection functions `communities()`, `plot.group_tna_communities()`, `plot.tna_communities()`, `print.group_tna_communities()`

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).

See Also

Model comparison functions `compare()`, `compare.group_tna()`, `plot.tna_comparison()`, `plot_compare()`, `plot_compare.group_tna()`

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	<i>Print a TNA Data Object</i>
----------------	--------------------------------

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).

See Also

Basic functions [build_model\(\)](#), [hist.group_tna\(\)](#), [hist.tna\(\)](#), [import_data\(\)](#), [plot.group_tna\(\)](#), [plot.tna\(\)](#), [plot_frequencies\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic\(\)](#), [plot_mosaic.group_tna\(\)](#), [plot_mosaic.tna_data\(\)](#), [prepare_data\(\)](#), [print.group_tna\(\)](#), [print.summary.group_tna\(\)](#), [print.summary.tna\(\)](#), [print.tna\(\)](#), [simulate.tna\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

Examples

```
res <- prepare_data(group_regulation_long, action = "Action", actor = "Actor",  
  time = "Time")  
print(res, which = "sequence")  
print(res, which = "meta")  
print(res, 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).

See Also

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_permutation\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.tna_bootstrap\(\)](#)

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).

See Also

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_permutation\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.tna_bootstrap\(\)](#)

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 Transition 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 <code>bootstrap()</code> when using <code>method = "bootstrap"</code> and when a <code>tna_bootstrap</code> 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

Validation functions `bootstrap()`, `deprune()`, `estimate_cs()`, `permutation_test()`, `permutation_test.group_tna()`, `plot.group_tna_bootstrap()`, `plot.group_tna_permutation()`, `plot.group_tna_stability()`, `plot.tna_bootstrap()`, `plot.tna_permutation()`, `plot.tna_stability()`, `print.group_tna_bootstrap()`, `print.group_tna_permutation()`, `print.group_tna_stability()`, `print.summary.group_tna_bootstrap()`, `print.summary.tna_bootstrap()`, `print.tna_bootstrap()`, `print.tna_permutation()`, `print.tna_stability()`, `pruning_details()`, `reprune()`, `summary.group_tna_bootstrap()`, `summary.tna_bootstrap()`

Validation functions `bootstrap()`, `deprune()`, `estimate_cs()`, `permutation_test()`, `permutation_test.group_tna()`, `plot.group_tna_bootstrap()`, `plot.group_tna_permutation()`, `plot.group_tna_stability()`, `plot.tna_bootstrap()`, `plot.tna_permutation()`, `plot.tna_stability()`, `print.group_tna_bootstrap()`, `print.group_tna_permutation()`, `print.group_tna_stability()`, `print.summary.group_tna_bootstrap()`, `print.summary.tna_bootstrap()`, `print.tna_bootstrap()`, `print.tna_permutation()`, `print.tna_stability()`, `pruning_details()`, `reprune()`, `summary.group_tna_bootstrap()`, `summary.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	<i>Print Detailed Information on the Pruning Results</i>
-----------------	--

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

Validation functions `bootstrap()`, `deprune()`, `estimate_cs()`, `permutation_test()`, `permutation_test.group_tna()`, `plot.group_tna_bootstrap()`, `plot.group_tna_permutation()`, `plot.group_tna_stability()`, `plot.tna_bootstrap()`, `plot.tna_permutation()`, `plot.tna_stability()`, `print.group_tna_bootstrap()`, `print.group_tna_permutation()`, `print.group_tna_stability()`, `print.summary.group_tna_bootstrap()`, `print.summary.tna_bootstrap()`, `print.tna_bootstrap()`, `print.tna_permutation()`, `print.tna_stability()`, `prune()`, `reprune()`, `summary.group_tna_bootstrap()`, `summary.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 `communities()`, `group_model()`, `mmm_stats()`

Examples

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

reprune	<i>Restore Previous Pruning of a Transition Network Analysis Model</i>
---------	--

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

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_permutation\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [summary.group_tna_bootstrap\(\)](#), [summary.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
```

simulate.tna

*Simulate Data from a Transition Network Analysis Model***Description**

Simulate Data from a Transition Network Analysis Model

Usage

```
## S3 method for class 'tna'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  max_len = 100L,
  na_range = c(0L, 0L),
  ...
)
```

Arguments

object	A tna object. The edge weights must be transition probabilities, i.e., the model must have type = "relative".
nsim	An integer giving the number of sequences to simulate. The default is 1.
seed	Ignored. Please use set.seed() manually.
max_len	An integer giving the maximum length of the simulated sequences. When no missing values are generated, this is the length of all simulated sequences.
na_range	An integer vector of length 2 giving the minimum and maximum number of missing values to generate for each sequence. The number of missing values is drawn uniformly from this range. If both values are zero (the default), no missing values are generated.
...	Ignored.

Value

A data.frame of the simulated sequence data with nsim rows and max_len columns.

See Also

Basic functions [build_model\(\)](#), [hist.group_tna\(\)](#), [hist.tna\(\)](#), [import_data\(\)](#), [plot.group_tna\(\)](#), [plot.tna\(\)](#), [plot_frequencies\(\)](#), [plot_frequencies.group_tna\(\)](#), [plot_mosaic\(\)](#), [plot_mosaic.group_tna\(\)](#), [plot_mosaic.tna_data\(\)](#), [prepare_data\(\)](#), [print.group_tna\(\)](#), [print.summary.group_tna\(\)](#), [print.summary.tna\(\)](#), [print.tna\(\)](#), [print.tna_data\(\)](#), [summary.group_tna\(\)](#), [summary.tna\(\)](#), [tna-package](#)

Examples

```
model <- tna(group_regulation)
sim <- simulate(model, nsim = 10, max_len = 10)
```

summary.group_tna	<i>Calculate Summary of Network Metrics for a grouped Transition Network</i>
-------------------	--

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

Basic functions `build_model()`, `hist.group_tna()`, `hist.tna()`, `import_data()`, `plot.group_tna()`, `plot.tna()`, `plot_frequencies()`, `plot_frequencies.group_tna()`, `plot_mosaic()`, `plot_mosaic.group_tna()`, `plot_mosaic.tna_data()`, `prepare_data()`, `print.group_tna()`, `print.summary.group_tna()`, `print.summary.tna()`, `print.tna()`, `print.tna_data()`, `simulate.tna()`, `summary.tna()`, `tna-package`

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.
...	Ignored.

Value

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

See Also

Validation functions `bootstrap()`, `deprune()`, `estimate_cs()`, `permutation_test()`, `permutation_test.group_tna()`, `plot.group_tna_bootstrap()`, `plot.group_tna_permutation()`, `plot.group_tna_stability()`, `plot.tna_bootstrap()`, `plot.tna_permutation()`, `plot.tna_stability()`, `print.group_tna_bootstrap()`, `print.group_tna_permutation()`, `print.group_tna_stability()`, `print.summary.group_tna_bootstrap()`, `print.summary.tna_bootstrap()`, `print.tna_bootstrap()`, `print.tna_permutation()`, `print.tna_stability()`, `prune()`, `pruning_details()`, `reprune()`, `summary.tna_bootstrap()`

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.

See Also

Basic functions `build_model()`, `hist.group_tna()`, `hist.tna()`, `import_data()`, `plot.group_tna()`, `plot.tna()`, `plot_frequencies()`, `plot_frequencies.group_tna()`, `plot_mosaic()`, `plot_mosaic.group_tna()`, `plot_mosaic.tna_data()`, `prepare_data()`, `print.group_tna()`, `print.summary.group_tna()`, `print.summary.tna()`, `print.tna()`, `print.tna_data()`, `simulate.tna()`, `summary.group_tna()`, `tna-package`

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

<code>object</code>	A <code>tna_bootstrap</code> object.
<code>...</code>	Ignored.

Value

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

See Also

Validation functions [bootstrap\(\)](#), [deprune\(\)](#), [estimate_cs\(\)](#), [permutation_test\(\)](#), [permutation_test.group_tna\(\)](#), [plot.group_tna_bootstrap\(\)](#), [plot.group_tna_permutation\(\)](#), [plot.group_tna_stability\(\)](#), [plot.tna_bootstrap\(\)](#), [plot.tna_permutation\(\)](#), [plot.tna_stability\(\)](#), [print.group_tna_bootstrap\(\)](#), [print.group_tna_permutation\(\)](#), [print.group_tna_stability\(\)](#), [print.summary.group_tna_bootstrap\(\)](#), [print.summary.tna_bootstrap\(\)](#), [print.tna_bootstrap\(\)](#), [print.tna_permutation\(\)](#), [print.tna_stability\(\)](#), [prune\(\)](#), [pruning_details\(\)](#), [reprune\(\)](#), [summary.group_tna_bootstrap\(\)](#)

Examples

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

Index

* **basic**

- build_model, 9
- hist.group_tna, 29
- hist.tna, 30
- import_data, 31
- plot.group_tna, 36
- plot.tna, 42
- plot_frequencies, 54
- plot_frequencies.group_tna, 55
- plot_mosaic, 56
- plot_mosaic.group_tna, 57
- plot_mosaic.tna_data, 58
- prepare_data, 59
- print.group_tna, 61
- print.summary.group_tna, 67
- print.summary.tna, 68
- print.tna, 70
- print.tna_data, 75
- simulate.tna, 82
- summary.group_tna, 83
- summary.tna, 85
- tna-package, 3

* **centralities**

- betweenness_network, 6
- centralities, 13
- plot.group_tna_centralities, 37
- plot.tna_centralities, 44
- print.group_tna_centralities, 63
- print.tna_centralities, 72

* **cliques**

- cliques, 14
- plot.group_tna_cliques, 39
- plot.tna_cliques, 46
- print.group_tna_cliques, 63
- print.tna_cliques, 72

* **clusters**

- communities, 16
- group_model, 25
- mmm_stats, 32

- rename_groups, 80

* **communities**

- communities, 16
- plot.group_tna_communities, 39
- plot.tna_communities, 47
- print.group_tna_communities, 64
- print.tna_communities, 73

* **comparison**

- compare, 17
- compare.group_tna, 19
- plot.tna_comparison, 48
- plot_compare, 52
- plot_compare.group_tna, 53
- print.tna_comparison, 74

* **datasets**

- engagement, 21
- engagement_mmm, 21
- group_regulation, 28
- group_regulation_long, 28

* **helpers**

- as.igraph.group_tna, 4
- as.igraph.matrix, 5
- as.igraph.tna, 6

* **validation**

- bootstrap, 7
- deprune, 20
- estimate_cs, 22
- permutation_test, 33
- permutation_test.group_tna, 34
- plot.group_tna_bootstrap, 37
- plot.group_tna_permutation, 40
- plot.group_tna_stability, 41
- plot.tna_bootstrap, 44
- plot.tna_permutation, 50
- plot.tna_stability, 51
- print.group_tna_bootstrap, 62
- print.group_tna_permutation, 65
- print.group_tna_stability, 66
- print.summary.group_tna_bootstrap,

- 67
- print.summary.tna_bootstrap, 69
- print.tna_bootstrap, 71
- print.tna_permutation, 76
- print.tna_stability, 77
- prune, 78
- pruning_details, 79
- reprune, 81
- summary.group_tna_bootstrap, 84
- summary.tna_bootstrap, 87
- as.igraph.group_tna, 4, 5, 6
- as.igraph.matrix, 5, 5, 6
- as.igraph.tna, 5, 6
- atna (build_model), 9
- base::rank(), 11, 26
- betweenness_network, 6, 14, 38, 45, 63, 72
- bootstrap, 7, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
- bootstrap(), 78
- build_model, 4, 9, 29–31, 36, 43, 54–58, 60, 61, 67, 69, 70, 75, 82, 84, 86
- centralities, 7, 13, 38, 45, 63, 72
- centralities(), 23, 33–35, 44
- cliques, 14, 39, 47, 64, 73
- communities, 16, 27, 33, 40, 48, 64, 74, 80
- communities(), 48
- compare, 17, 19, 49, 53, 54, 74
- compare.group_tna, 18, 19, 49, 53, 54, 74
- compare.tna(), 19
- ctna (build_model), 9
- deprune, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
- deprune(), 79
- dplyr::select(), 31
- engagement, 21, 22, 28, 29
- engagement_mmm, 21, 21, 28, 29
- estimate centrality stability (estimate_cs), 22
- estimate_cs, 9, 20, 22, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
- ftna (build_model), 9
- graphics::hist(), 29, 30
- group_atna (group_model), 25
- group_ctna (group_model), 25
- group_ftna (group_model), 25
- group_model, 17, 25, 33, 80
- group_regulation, 21, 22, 28, 29
- group_regulation_long, 21, 22, 28, 28
- group_tna (group_model), 25
- hist.group_tna, 4, 12, 29, 30, 31, 36, 43, 54–58, 60, 61, 67, 69, 70, 75, 82, 84, 86
- hist.tna, 4, 12, 29, 30, 31, 36, 43, 54–58, 60, 61, 67, 69, 70, 75, 82, 84, 86
- igraph::betweenness(), 14
- igraph::closeness(), 13, 14
- igraph::strength(), 13
- import_data, 4, 12, 29, 30, 31, 36, 43, 54–58, 60, 61, 67, 69, 70, 75, 82, 84, 86
- mmm_stats, 17, 27, 32, 80
- permutation_test, 9, 20, 24, 33, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
- permutation_test.group_tna, 9, 20, 24, 34, 34, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
- permutation_test.tna(), 34
- plot.group_tna, 4, 12, 29–31, 36, 43, 54–58, 60, 61, 67, 69, 70, 75, 82, 84, 86
- plot.group_tna_bootstrap, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
- plot.group_tna_centralities, 7, 14, 37, 45, 63, 72
- plot.group_tna_cliques, 15, 39, 47, 64, 73
- plot.group_tna_communities, 17, 39, 48, 64, 74
- plot.group_tna_permutation, 9, 20, 24, 34, 35, 37, 40, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
- plot.group_tna_stability, 9, 20, 24, 34, 35, 37, 41, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
- plot.tna, 4, 12, 29–31, 36, 42, 54–58, 60, 61, 67, 69, 70, 75, 82, 84, 86
- plot.tna(), 36, 37, 44

`plot.tna_bootstrap`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
`plot.tna_centralities`, 7, 14, 38, 44, 63, 72
`plot.tna_cliques`, 15, 39, 46, 64, 73
`plot.tna_cliques()`, 39
`plot.tna_communities`, 17, 40, 47, 64, 74
`plot.tna_communities()`, 40
`plot.tna_comparison`, 18, 19, 48, 53, 54, 74
`plot.tna_permutation`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
`plot.tna_permutation()`, 40
`plot.tna_stability`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 51, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
`plot.tna_stability()`, 41
`plot_compare`, 18, 19, 49, 52, 54, 74
`plot_compare.group_tna`, 18, 19, 49, 53, 53, 74
`plot_compare.tna()`, 53
`plot_frequencies`, 4, 12, 29–31, 36, 43, 54, 55–58, 60, 61, 67, 69, 70, 75, 82, 84, 86
`plot_frequencies.group_tna`, 4, 12, 29–31, 36, 43, 54, 55, 56–58, 60, 61, 67, 69, 70, 75, 82, 84, 86
`plot_model()`, 50
`plot_mosaic`, 4, 12, 29–31, 36, 43, 54, 55, 56, 57, 58, 60, 61, 67, 69, 70, 75, 82, 84, 86
`plot_mosaic.group_tna`, 4, 12, 29–31, 36, 43, 54–56, 57, 58, 60, 61, 67, 69, 70, 75, 82, 84, 86
`plot_mosaic.tna_data`, 4, 12, 29–31, 36, 43, 54–57, 58, 60, 61, 67, 69, 70, 75, 82, 84, 86
`prepare_data`, 4, 12, 29–31, 36, 43, 54–58, 59, 61, 67, 69, 70, 75, 82, 84, 86
`prepare_data()`, 11
`pretty`, 30
`print.group_tna`, 4, 12, 29–31, 36, 43, 54–58, 60, 61, 67, 69, 70, 75, 82, 84, 86
`print.group_tna_bootstrap`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
`print.group_tna_centralities`, 7, 14, 38, 45, 63, 72
`print.group_tna_cliques`, 15, 39, 47, 63, 73
`print.group_tna_communities`, 17, 40, 48, 64, 74
`print.group_tna_permutation`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
`print.group_tna_stability`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
`print.summary.group_tna`, 4, 12, 29–31, 36, 43, 54–58, 60, 61, 67, 69, 70, 75, 82, 84, 86
`print.summary.group_tna_bootstrap`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 67, 69, 71, 76, 77, 79–81, 85, 87
`print.summary.tna`, 4, 12, 29–31, 36, 43, 54–58, 60, 61, 67, 68, 70, 75, 82, 84, 86
`print.summary.tna()`, 67
`print.summary.tna_bootstrap`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
`print.tna`, 4, 12, 29–31, 36, 43, 54–58, 60, 61, 67, 69, 70, 75, 82, 84, 86
`print.tna()`, 61
`print.tna_bootstrap`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
`print.tna_bootstrap()`, 62
`print.tna_centralities`, 7, 14, 38, 45, 63, 72
`print.tna_cliques`, 15, 39, 47, 64, 72
`print.tna_cliques()`, 64
`print.tna_communities`, 17, 40, 48, 64, 73
`print.tna_communities()`, 64
`print.tna_comparison`, 18, 19, 49, 53, 54, 74
`print.tna_data`, 4, 12, 29–31, 36, 43, 54–58, 60, 61, 67, 69, 70, 75, 82, 84, 86
`print.tna_permutation`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87
`print.tna_permutation()`, 65
`print.tna_stability`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87

`print.tna_stability()`, 66
`prune`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 78, 80, 81, 85, 87
`prune()`, 43
`pruning_details`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79, 79, 81, 85, 87
`pruning_details()`, 79

`qgraph::qgraph()`, 42, 43, 46–48, 50, 52, 53

`rename_groups`, 17, 27, 33, 80
`reprune`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79, 80, 81, 85, 87
`reprune()`, 81
`reprune.tna (deprune)`, 20

`set.seed()`, 82
`simulate.tna`, 4, 12, 29–31, 36, 43, 54–58, 60, 61, 67, 69, 70, 75, 82, 84, 86
`stats::cor()`, 23
`stats::ecdf`, 18
`stats::mad`, 18
`stats::p.adjust()`, 33, 35
`summary.group_tna`, 4, 12, 29–31, 36, 43, 54–58, 60, 61, 67, 69, 70, 75, 82, 83, 86
`summary.group_tna_bootstrap`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 84, 87
`summary.tna`, 4, 12, 29–31, 36, 43, 54–58, 60, 61, 67, 69, 70, 75, 82, 84, 85
`summary.tna_bootstrap`, 9, 20, 24, 34, 35, 37, 41, 44, 50, 52, 62, 65, 66, 68, 69, 71, 76, 77, 79–81, 85, 87

`tidyr::pivot_wider()`, 60
`tna (build_model)`, 9
`tna()`, 42
`tna-package`, 3