Package 'htna'

November 20, 2026

Title Heterogeneous Transition Network Analysis

Version 0.3.7

Description Implements the Heterogeneous Transition Network Analysis (HTNA) method described by López-Pernas et al. (2026) <doi:10.1002/jcal.70285>. The method is an extension of transition network analysis (TNA) where actions or events belong to two or more distinct actor types (e.g. Human and AI), preserving the actor type partition on the resulting network. Provides a thin, focused API on top of the 'Nestimate' estimation engine and the 'cograph' rendering engine, so downstream bootstrap, permutation, reliability, centrality, and plotting functions treat each actor's codes as a distinct node group.

License MIT + file LICENSE

URL https://sonsoles.me/htna/

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

Depends R (>= 4.1.0)

Imports Nestimate (>= 0.8.0), cograph, igraph

Suggests ggplot2, gridExtra, knitr, rmarkdown, testthat (>= 3.0.0), tna

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Sonsoles López-Pernas [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-9621-1392>), Kamila Misiejuk [aut, cph], Mohammed Saqr [aut, cph] (ORCID: <https://orcid.org/0000-0001-5881-3109>)

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

Repository CRAN

Date/Publication 2026-11-20 12:40:02 UTC

Contents

htna-package . . . 3
ai_simplified . . . 4
as.data.frame.htna . . . 5
as.data.frame.htna_stability . . . 6
as.igraph.htna . . . 8
association_rules_htna . . . 9
bayes_compare_htna . . . 10
bootstrap . . . 11
bootstrap_htna . . . 12
build_htna . . . 13
casedrop_reliability_htna . . . 16
centralities_htna . . . 17
centrality_stability_htna . . . 19
certainty_htna . . . 21
compare_htna . . . 22
deprune_htna . . . 23
edge_betweenness_htna . . . 24
extract_meta_paths . . . 25
frequencies_htna . . . 27
htna_comparison-methods . . . 28
htna_palette . . . 30
human_ai . . . 30
human_ai_codebook . . . 32
human_simplified . . . 32
markov_order_test_htna . . . 33
mosaic_plot_htna . . . 34
permutation_htna . . . 35
plot.htna_comparison_group . . . 36
plot.htna_stability . . . 37
plot_centralities . . . 38
plot_edge_betweenness_htna . . . 40
plot_frequencies_htna . . . 41
plot_htna . . . 42
plot_htna_bootstrap . . . 44
plot_htna_diff . . . 45
plot_sequences . . . 46
print.htna_bootstrap_group . . . 47
print.htna_paths . . . 48
print.htna_stability . . . 49
prune_htna . . . 50
pruning_details_htna . . . 51

reliability_htna 52
reprune_htna 53
sequence_compare_htna 54
sequence_plot_htna 55
state_distribution_htna 57
state_frequencies_htna 58
summary.htna 59

Index **61**

`htna-package` *htna: Heterogeneous Transition Network Analysis*

Description

`htna` estimates, describes, plots and tests heterogeneous transition networks of sequences of events produced by two or more types of actor, such as a learner and an AI assistant, a student and a teacher, or a clinician and a patient. Heterogeneous transition network analysis (HTNA) models such sequences as one transition network in which each node is a code belonging to one actor type and each edge is the transition from one code to the next (Lopez-Pernas, Misiejuk and Saqr, 2026). Keeping the actor partition on the network separates the dynamics within each actor from the hand-overs between actors, so the estimation, inference and visualisation tools of transition network analysis can show what the AI does after the learner asks for help, and how the learner responds.

`htna` builds the network from long event data, describes it by actor type, draws it with each actor type in its own colour, shape and arc of the layout, and keeps the partition on the results of every downstream analysis.

Building and describing

`build_htna()` estimates the network (or one network per cohort or cluster); `summary.htna()`, `as.data.frame.htna()` and `as.igraph.htna()` describe and convert it. `prune_htna()`, `deprune_htna()`, `reprune_htna()` and `pruning_details_htna()` remove and restore weak edges.

Plots

`plot_htna()` draws the network, `plot_htna_diff()` the difference between two networks, `plot_htna_bootstrap()` the bootstrap-significant edges, `sequence_plot_htna()` the sessions, `plot_frequencies_htna()` and `mosaic_plot_htna()` the code frequencies, `plot_centralities()` and `plot_edge_betweenness_htna()` node and edge importance.

Comparing networks

`compare_htna()` describes the differences between networks by actor-pair block; `permutation_htna()` and `bayes_compare_htna()` test them.

Model assessment

`bootstrap_htna()`, `certainty_htna()`, `reliability_htna()`, `casedrop_reliability_htna()`, `centrality_stability_htna()` and `markov_order_test_htna()` assess the stability of the edges and centralities and the adequacy of a first-order model.

Patterns and frequencies

`extract_meta_paths()` finds recurring paths at the level of codes or actor types, `association_rules_htna()` association rules, `sequence_compare_htna()` patterns that differ between cohorts, and `frequencies_htna()`, `state_frequencies_htna()` and `state_distribution_htna()` the frequencies of the codes.

Author(s)

Maintainer: Sonsoles López-Pernas <sonsoles.lopez@uef.fi> (ORCID) [copyright holder]

Authors:

- Sonsoles López-Pernas <sonsoles.lopez@uef.fi> (ORCID) [copyright holder]
- Kamila Misiejuk [copyright holder]
- Mohammed Saqr <saqr@saqr.me> (ORCID) [copyright holder]

References

Lopez-Pernas, S., Misiejuk, K., & Saqr, M. (2026). How AI-generated feedback hinders or helps learning: A heterogeneous TNA study of learning dynamics. *Journal of Computer Assisted Learning*, 42(4), e70285. doi:10.1002/jcal.70285

See Also

The bundled data human_ai; the 'tna' package for transition network analysis of sequences with a single actor type.

---

ai_simplified *AI turns of the vibe-coding sessions*

Description

`ai_simplified` is the AI part of human_ai: its AI rows, as one data frame for one actor type. Together with human_simplified it is the input for the named-list form of `build_htna()`, in which each actor type comes in its own data frame.

Usage

```
ai_simplified
```

Format

A data frame with 8,551 rows and 10 columns: the columns of human_ai without `phase`; `actor_type` is always `"AI"`.

Source

Derived from the `ai_long` data of the 'Nestimate' package (see human_ai).

See Also

human_ai, human_simplified, human_ai_codebook

Examples

```
data(human_simplified, ai_simplified)
net <- build_htna(list(Human = human_simplified, AI = ai_simplified))
summary(net)
```

`as.data.frame.htna` *Tabulate the edges and nodes of an htna network*

Description

`as.data.frame()` returns the edges or the nodes of a heterogeneous transition network as a tidy data frame in which every row carries the actor type of its codes. The edge table lists every non-zero transition with its weight and the actor types of its source and target, so the within-actor and between-actor transitions (for example Human to AI) can be read, counted or filtered directly. The node table lists every code with its actor type.

For a grouped network the tables of all cohorts are stacked with a leading `group` column. Self-transitions (the diagonal of the weight matrix) are kept. For an undirected network (`method = "co_occurrence"`) each pair of codes is listed once.

Usage

```
## S3 method for class 'htna'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  what = c("edges", "nodes")
)

## S3 method for class 'htna_group'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  what = c("edges", "nodes")
)
```

Arguments

x	An htna network from build_htna() or an htna_group.
row.names, optional	Ignored; present for the generic.
...	Unused.
what	"edges" (default) or "nodes".

Value

A data frame. For what = "edges": one row per non-zero edge, with columns from, to, weight, source_actor and target_actor (factors in the network's actor order), sorted by source then target in node order. For what = "nodes": one row per node, with columns node and actor. Grouped networks add a first column group.

See Also

build_htna(), summary.htna() for counts per actor pair.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
head(as.data.frame(net))
as.data.frame(net, what = "nodes")
```

---

as.data.frame.htna_stability

Tabulate centrality stability results

Description

as.data.frame() returns the results of centrality_stability_htna() as a tidy data frame; summary() of a grouped result returns the stacked correlation table. The CS coefficients (one row per measure) summarize how many sessions can be dropped before the centrality ranking changes; the correlation table (one row per measure and drop proportion) shows the trace behind each coefficient. For a grouped result the rows of every cohort are stacked with a leading group column.

Usage

```
## S3 method for class 'htna_stability'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  what = c("cs", "correlations")
```

```
)

## S3 method for class 'htna_stability_group'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  what = c("cs", "correlations")
)

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

Arguments

x	An htna_stability or htna_stability_group.
row.names, optional	Ignored; present for the generic.
...	Unused.
what	"cs" (default) for the CS coefficients, or "correlations" for the correlation table.
object	An htna_stability_group.

Value

A data frame. For what = "cs": columns measure and cs, one row per measure. For what = "correlations": columns measure, drop_prop, mean_cor, sd_cor and prop_above (the share of resamples with a correlation above the threshold), one row per measure and drop proportion. Grouped results add a first column group. summary() of a grouped result returns the stacked correlation table.

See Also

centrality_stability_htna(), plot.htna_stability()

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
cs  <- centrality_stability_htna(net, iter = 20, seed = 1)
as.data.frame(cs)
as.data.frame(cs, what = "correlations")
```

`as.igraph.htna` *Convert an htna network to an igraph graph*

Description

`as.igraph()` converts a heterogeneous transition network to an 'igraph' graph and keeps its actor partition. Many network measures and algorithms are available for 'igraph' graphs; because the partition is kept, a measure computed in 'igraph' can still be summarized by actor type, for example to compare the centrality of human and AI codes.

`as.igraph()` creates a weighted graph from the transition weights, stores the actor type of every vertex as the vertex attribute `actor_type`, the ordered actor types as the graph attribute `actor_levels` and the estimation method as `htna_method`. A grouped network is converted cohort by cohort; each graph also records its cohort name in the graph attribute `cohort`. It returns an `igraph` graph, or a list of graphs, one per cohort, for a grouped network.

For the edge and node tables of the network without 'igraph', use `as.data.frame.htna()`.

Usage

```
## S3 method for class 'htna'
as.igraph(x, mode = NULL, weighted = TRUE, diag = TRUE, ...)

## S3 method for class 'htna_group'
as.igraph(x, mode = NULL, weighted = TRUE, diag = TRUE, ...)
```

Arguments

`x`	An htna network from `build_htna()` or an `htna_group`.
`mode`	Graph mode. Default `NULL`: `"directed"` for a directed network, `"undirected"` otherwise.
`weighted`	Use the transition weights as edge weights. Default `TRUE`.
`diag`	Keep self-transitions as loops. Default `TRUE`.
`...`	Further arguments to `igraph::graph_from_adjacency_matrix()`.

Value

An `igraph` graph for an htna network. For an `htna_group`, a named list of `igraph` graphs, one per cohort, of class `c("htna_igraph_group", "list")`.

Conditions

`htna_bad_input` when `x` is not an htna network or an argument is invalid; `htna_unassigned_node` when a node has no actor type.

See Also

`build_htna()`, `as.data.frame.htna()`, `centralities_htna()`

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
graph <- igraph::as.igraph(net)
igraph::vertex_attr(graph, "actor_type")
```

`association_rules_htna` *Mine association rules from the sessions of a network*

Description

`association_rules_htna()` mines association rules from the sessions of an htna network: rules that describe which codes tend to occur together in the same session, regardless of their order. A rule "A, B => C" states that sessions containing codes A and B also tend to contain code C. Support is the share of sessions containing all codes of the rule, confidence the share of sessions with the antecedent that also contain the consequent, and lift the ratio of the confidence to the overall share of the consequent. In a heterogeneous network a rule can join codes of different actors, for example a human request and the AI actions that accompany it.

`association_rules_htna()` runs Nestimate's `association_rules()` on the network: it treats every session as one transaction (the set of codes it contains), finds the frequent itemsets with the Apriori algorithm, and keeps the rules that pass the support, confidence and lift thresholds. It returns a `net_association_rules` object whose rules table has one row per rule.

The rules ignore order. For ordered patterns use `extract_meta_paths()`; for associations between codes and sessions use `mosaic_plot_htna()`.

Usage

```
association_rules_htna(x, ...)
```

Arguments

`x`	An htna network from `build_htna()`, or a binary session-by-code matrix (declared with `format = "binary"` from Nestimate 1.0.9). A grouped network is not accepted; pass one cohort.
`...`	Thresholds: `min_support` (default 0.1), `min_confidence` (0.5), `min_lift` (1), `max_length` (largest itemset, 5) and `format`.

Value

An object of class `net_association_rules` with `print()`, `summary()` and `plot()` methods. Its rules table has one row per rule, with columns `antecedent`, `consequent`, `support`, `confidence`, `lift`, `conviction`, `count` and `n_transactions`.

See Also

extract_meta_paths() for ordered patterns, frequencies_htna(), mosaic_plot_htna().

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
association_rules_htna(net, max_length = 3L)
```

---

bayes_compare_htna *Compare networks edge by edge with a Bayesian model*

Description

bayes_compare_htna() compares two heterogeneous transition networks, for example of early and late sessions, edge by edge with a Bayesian model: how much more or less likely a human request is to be followed by an AI plan in one network than in the other, and how precisely the data estimate that difference. The model treats the transitions out of each code as a multinomial draw with a Dirichlet prior, so every transition probability has a Beta posterior. For each edge it reports the posterior mean difference, a credible interval, and the probability that the difference lies on its side of zero. An edge whose source code has few outgoing transitions gets a wide interval even when its probabilities look far apart.

bayes_compare_htna() runs Nestimate's bayes_compare() with the actor partition kept. It compares two htna networks built with method = "relative" or "frequency" on the same codes, or every pair of cohorts of a grouped network. An edge is flagged as credibly different when its interval excludes zero and its mean and nearest bound pass mean_threshold and bound_threshold; no correction is made for testing many edges, and every transition is treated as an independent draw, so the nesting of transitions in sessions is ignored. It returns a net_bayes object, or one per cohort pair for a grouped network.

The permutation test of permutation_htna() respects that nesting and asks whether a difference exceeds chance; compare_htna() gives descriptive similarity measures and the differences per actor pair. plot_htna_diff() draws the result.

Usage

```
bayes_compare_htna(x, y = NULL, ...)
```

Arguments

x, y	Two htna networks, or an htna_group in x with y = NULL, in which case every pair of cohorts is compared. Networks that do not hold the same codes are compared over the codes of both; a code a network lacks has no transitions in it.
...	Settings: prior (Dirichlet concentration, default 0.5), draws (Monte Carlo draws, 10000), ci (credible level, 0.95), mean_threshold (0.01), bound_threshold (0.001) and seed.

Value

For two networks, an object of class `c("net_bayes", "netdifference", "net_permutation")` with `print()`, `summary()` and `plot()` methods. For a grouped network, a named list of class `c("net_bayes_group", "list")` with one result per cohort pair (for example `"Early vs Late"`). `summary()` returns a data frame with one row per edge (and pair) and columns including `from`, `to`, `weight_x`, `weight_y`, `diff`, `effect_size`, `ci_lower`, `ci_upper`, `p_difference`, `p_value` and `sig`. Differences are `x - y`; in a grouped comparison `x` is the first cohort of the pair.

See Also

`permutation_htna()`, `compare_htna()`, `plot_htna_diff()`.

Examples

```
data(human_ai)
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
bc <- bayes_compare_htna(grp, draws = 200, seed = 1)
head(summary(bc))
plot_htna_diff(bc)
```

---

bootstrap *Bootstrap a network*

Description

`bootstrap()` bootstraps a fitted network: it is a generic for resampling-based uncertainty of a fitted network. Its htna method runs `bootstrap_htna()`: the sessions are resampled, the network is re-estimated on each resample, and the distribution of every edge weight is summarized. For an htna network it returns an `htna_bootstrap`.

The generic has the same name as the `tna` generic; when both packages are attached, the one attached last is found first. `bootstrap_htna()` is unambiguous.

Usage

```
bootstrap(x, ...)
```

Arguments

`x`	An object to bootstrap; for htna, a network from `build_htna()`.
`...`	Passed to the method; see `bootstrap_htna()`.

Value

The method's result; for an htna network an `htna_bootstrap` (see `bootstrap_htna()`).

See Also

`bootstrap_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
bootstrap(net, iter = 50, seed = 1)
```

`bootstrap_htna` *Bootstrap the edges of a heterogeneous transition network*

Description

`bootstrap_htna()` bootstraps the edges of a heterogeneous transition network, or of every cohort of a grouped network, to show which transitions would reappear in another sample of sessions. The non-parametric bootstrap resamples the sessions with replacement, re-estimates the network on each resample, and summarizes the distribution of every edge weight: its mean, standard deviation, percentile confidence interval, and a p-value. An edge weight estimated from a sample of sessions is uncertain, and a plot of the network alone does not show that uncertainty.

Under the default `inference = "stability"`, the p-value is the share of resamples (plus one, over the number of resamples plus one) whose weight falls outside a consistency range around the observed weight (by default 75 to 125 percent of it); an edge with a p-value below `ci_level` is reproduced by the resamples and is marked significant. The bootstrap works at the level of individual codes and runs Nestimate's `bootstrap_network()` with the actor partition kept, so `plot_htna_bootstrap()` draws the result with the actor layout and colours. `bootstrap()` is the generic, with a method for htna networks. It returns an `htna_bootstrap` object, or one per cohort for a grouped network.

For a closed-form estimate of the same quantities use `certainty_htna()`; to remove unstable edges use `prune_htna()` with `method = "bootstrap"`; for the stability of centralities use `centrality_stability_htna()`.

Usage

```
bootstrap_htna(x, ...)

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

Arguments

`x`	An htna network from `build_htna()`, or an `htna_group`, in which case every cohort is bootstrapped.
`...`	Settings: `iter` (resamples, default 1000), `ci_level` (0.05), `inference` (`"stability"` or `"threshold"`), `consistency_range` (`c(0.75, 1.25)`), `edge_threshold`, `seed`, `boundary`, `ci_method` and `actor` (resample whole actors instead of sessions).

Value

For a network, an object of class `c("htna_bootstrap", "net_bootstrap")` with `print()`, `summary()` and `plot()` methods. `summary()` returns a data frame with one row per edge and columns `from`, `to`, `weight`, `mean`, `sd`, `p_value`, `sig`, `ci_lower`, `ci_upper`, `cr_lower` and `cr_upper`. For an `htna_group`, a named list of such objects with class `c("htna_bootstrap_group", "list")`.

Conditions

An error of class `htna_bad_input` is raised when `x` is neither an htna network nor a list of them.

See Also

`plot_htna_bootstrap()`, `certainty_htna()`, `prune_htna()`, `reliability_htna()`.

Examples

```
data(human_ai)
net  <- build_htna(human_ai, actor_type = "actor_type")
boot <- bootstrap_htna(net, iter = 50, seed = 1)
boot
head(summary(boot))
```

`build_htna` *Build a heterogeneous transition network*

Description

`build_htna()` builds a heterogeneous transition network: one transition network over interaction sequences in which the events are produced by two or more types of actor, such as a learner and an AI assistant. In heterogeneous transition network analysis (HTNA) each node is a code (an action or event) and belongs to exactly one actor type, and each directed edge is the transition from one code to the next within a session (Lopez-Pernas, Misiejuk and Saqr, 2026). Because the actor partition is kept on the network, the within-actor transitions (Human to Human) and the hand-overs between actors (Human to AI, AI to Human) can be estimated, plotted and tested as distinct parts of one model.

`build_htna()` orders the events of every session in time, estimates the transition network over the combined sequence with the chosen `method` (relative transition probabilities by default, or frequencies, co-occurrences or attention weights), and attaches the actor type of every node. The actor partition can be given in three ways: a named list of one long data frame per actor type, a single data frame with a column naming the actor type of each row (`actor_type`), or a single data frame together with a code-to-actor lookup (`node_groups`). With `group`, one network is estimated per level of a session-level variable (for example an early and a late phase), in the order of the factor levels. A fitted sequence clustering (from `Nestimate::build_clusters()`, `Nestimate::cluster_mmm()` or `tna::cluster_sequences()`) is turned into one network per cluster without refitting, and a fitted `Nestimate::build_mcml()` model into one network whose actor types are its node clusters.

It returns an `htna` network, or an `htna_group` with one network per cohort or cluster. The grouped result is passed as it is to the comparison, assessment and plotting functions, which work through every cohort. Use `summary.htna()` to describe the result and `as.data.frame.htna()` for its edges and nodes.

Usage

```
build_htna(
  data,
  actor_type = NULL,
  actor = NULL,
  node_groups = NULL,
  action = "code",
  session = "session_id",
  order = "order_in_session",
  method = "relative",
  group = NULL,
  disambiguate = FALSE,
  source_data = NULL,
  time = NULL,
  ...
)
```

Arguments

data	The event data, in one of these forms: • a named list of long data frames, one per actor type, with the same columns, e.g. `list(Human = human_simplified, AI = ai_simplified)`; • one long data frame, with `actor_type` or `node_groups`; • a fitted clustering (`net_clustering`, `net_mmm`, `tna_clustering`) or a `netobject_group`, giving one network per cluster; • a fitted `mcml` model, whose node clusters become the actor types.
actor_type	Name of the column giving the actor type of each row, for a single data frame. A factor column sets the order of the actor types; otherwise they are ordered by first appearance.
actor	Name of an optional column identifying the individual who produced each event (for example a learner id). Independent of `actor_type`. Default `NULL`.
node_groups	Code-to-actor lookup for a single data frame without an actor-type column: a named list of code vectors, e.g. `list(Human = c("Specify", "Check"), AI = c("Plan", "Execute"))`, or a two-column data frame with the code column (named as `action`) and the actor-type column, such as human_ai_codebook. Every code must belong to exactly one actor type. For a clustering or `mcml` input it optionally supplies or overrides the partition. Use either `actor_type` or `node_groups`, not both.
action	Name of the code column. Default `"code"`.
session	Name of the session column. Default `"session_id"`.
order	Name of the within-session order column. Default `"order_in_session"`.
method	Network estimator: `"relative"` (default, transition probabilities), `"frequency"`, `"co_occurrence"` (undirected), `"attention"`, `"reverse"`, `"ngram"`, `"gap"` or `"latency"`. `"latency"` weights each reply by its speed and needs `time` (Nestimate 1.0.20 or later), with times that differ within a session; the bundled `human_ai` has one time per session and cannot be used.

group	Name of a session-level column defining cohorts; one network is built per cohort. Default NULL (one network).
disambiguate	If TRUE, a code used by more than one actor type is split into one node per actor type, prefixed with the actor label ("Human:Ask", "AI:Ask"). Default FALSE (a shared code is an error).
source_data	The original event data for an mcml input whose fitted object did not keep it. Default NULL.
time	Name of an optional column holding the time of each event. With it the events of a session are ordered by time, the order column is not required, and a session is split where two events are further apart than time_threshold seconds (900 by default; pass time_threshold = Inf to keep sessions whole). Required for method = "latency". Default NULL.
...	Further estimation arguments, such as threshold, scaling, time_threshold, or latency_floor, latency_cap, latency_aggregate and latency_scope for method = "latency". With start or end the network gains a start or an end state; these belong to no actor and form the actor type "Boundary", placed last. The plots draw them above and below the actor types.

Value

An object of class c("htna", "netobject", "cograph_network"): a transition network with its weight matrix, nodes, sequence data and the actor partition (the actor type of every node and the ordered actor types). With group or a clustering input, an htna_group: a named list with one htna network per cohort or cluster, which keeps the clustering assignments and diagnostics as attributes. An mcml input returns one htna. as.data.frame.htna() returns the edges (one row per non-zero transition, with the actor types of its source and target) or the nodes.

Conditions

Errors inherit from htna_error. Specific classes: htna_missing_column when the session, order, time or code column is absent; htna_actor_conflict when a code belongs to more than one actor type and disambiguate = FALSE; htna_unassigned_node when a code has no actor type in node_groups; htna_tied_times when method = "latency" is given times that are tied within sessions; htna_bad_input for other invalid arguments.

References

Lopez-Pernas, S., Misiejuk, K., & Saqr, M. (2026). How AI-generated feedback hinders or helps learning: A heterogeneous TNA study of learning dynamics. *Journal of Computer Assisted Learning*, 42(4), e70285. doi:10.1002/jcal.70285

See Also

summary.htna() and as.data.frame.htna() to describe the network; plot_htna() to draw it; compare_htna(), permutation_htna() and bootstrap_htna() for inference.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
net

# the same network from one data frame per actor type
build_htna(list(Human = human_simplified, AI = ai_simplified))

# or from a code-to-actor codebook
build_htna(human_ai, node_groups = human_ai_codebook)

# one network per phase, in the order of the factor levels
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
summary(grp)
```

---

casedrop_reliability_htna

Assess the case-dropping stability of the edge weights

Description

`casedrop_reliability_htna()` measures how far the estimated edge weights of an htna network depend on a few sessions, by case-dropping stability (Epskamp et al., 2018). Networks are re-estimated after dropping increasing proportions of the sessions, and the edge weights of each reduced network are correlated with those of the full network. The case-dropping stability (CS) coefficient is the largest proportion of sessions that can be dropped while the correlation stays above a threshold (0.7 by default) in a required share (95 percent) of the resamples; Epskamp et al. (2018) recommend a coefficient of at least 0.25 and preferably above 0.5.

`casedrop_reliability_htna()` applies the procedure to the edge-weight vector of an htna network, or to every cohort of a grouped network, and also reports the mean and maximum absolute weight differences at each drop proportion. It runs Nestimate's `casedrop_reliability()` on the network and returns a `net_casedrop_reliability` object (a `net_casedrop_reliability_group` for a grouped network).

For the stability of node centralities use `centrality_stability_htna()`; for split-half agreement use `reliability_htna()`; for edge-level confidence intervals use `bootstrap_htna()`.

Usage

```
casedrop_reliability_htna(x, ...)
```

Arguments

x	An htna network from `build_htna()` or an `htna_group`.

`...` Settings: `iter` (resamples per drop proportion, default 1000), `drop_prop` (default `seq(0.1, 0.9, by = 0.1)`), `threshold` (0.7), `certainty` (0.95), `method` (correlation, `"spearman"`), `include_diag` (`FALSE`), `seed`, `target` (the edges assessed: `"auto"`, `"existing"` or `"all"`; Nestimate 1.0.2 or later) and `actor` (drop whole actors instead of sessions; Nestimate 1.0.19 or later).

Value

An object of class `net_casedrop_reliability` (one network) or `net_casedrop_reliability_group` (a grouped network), with `print()`, `summary()` and `plot()` methods; `summary()` returns a data frame with one row per metric and drop proportion (columns `metric`, `drop_prop`, `mean`, `sd`, `median`, `mad`, `q025`, `q975`).

References

Epskamp, S., Borsboom, D., & Fried, E. I. (2018). Estimating psychological networks and their accuracy: A tutorial paper. *Behavior Research Methods*, 50(1), 195-212. doi:10.3758/s13428017-08621

See Also

`centrality_stability_htna()`, `reliability_htna()`, `bootstrap_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
casedrop_reliability_htna(net, iter = 20, seed = 1)
```

---

centralities_htna *Compute node centralities of an htna network*

Description

`centralities_htna()` computes the node centralities of an htna network, with the actor type of every code kept. Centrality measures rank the codes of a transition network by their position in it: how much transition probability a code sends and receives (out- and in-strength), how close it is to the other codes (closeness), how often it lies on the shortest paths between other codes (betweenness), and how far its influence spreads (diffusion). Because the actor type is kept, the table shows whether the human or the AI codes hold the central positions, for example which actor's codes broker the hand-overs.

`centralities_htna()` computes nine measures: `OutStrength`, `InStrength`, `ClosenessIn`, `ClosenessOut`, `Closeness`, `Betweenness`, `BetweennessRSP` (randomized shortest-path betweenness), `Diffusion` and `Clustering` (weighted clustering coefficient). Path-based measures treat a larger transition weight as a shorter distance (the weights are inverted); strength, diffusion and clustering use the weights as they are. For a grouped network the measures are computed per cohort. It returns a data frame with one row per code (per cohort and code for a grouped network) and one column per measure.

Draw the result with plot_centralities(). To check whether the ranking of the codes would survive dropping sessions use centrality_stability_htna().

Usage

```
centralities_htna(
  x,
 measures = c("OutStrength", "InStrength", "ClosenessIn", "ClosenessOut", "Closeness",
    "Betweenness", "BetweennessRSP", "Diffusion", "Clustering"),
  ...
)
```

Arguments

x	An htna network from build_htna() or an htna_group.
measures	Names of the measures to compute. Default: all nine.
...	Further arguments for the centrality computations.

Value

A data frame of class c("htna_centralities", "data.frame"), one row per node, with columns node, actor (its actor type) and one numeric column per measure. For an htna_group, the rows of all cohorts are stacked (one row per cohort and node) with a leading group column.

Conditions

htna_bad_input when x is not an htna network or a measure name is unknown; htna_empty_input for a grouped network without cohorts; htna_unnamed_nodes when the nodes of the network have no names.

See Also

plot_centralities(), centrality_stability_htna(), edge_betweenness_htna()

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
centralities_htna(net)
centralities_htna(net, measures = c("InStrength", "Betweenness"))
```

centrality_stability_htna *Assess the case-dropping stability of node centralities*

Description

`centrality_stability_htna()` computes the case-dropping stability (CS) coefficients of the node centralities of an htna network, or of every cohort of a grouped network (Epskamp et al., 2018). The procedure re-estimates the network after dropping increasing proportions of the sessions and correlates the centralities of each reduced network with those of the full network; the CS coefficient of a measure is the largest proportion of sessions that can be dropped while the correlation stays above `threshold` with probability `certainty`. Epskamp et al. (2018) recommend a coefficient of at least 0.25 and preferably above 0.5.

Centrality measures rank the codes of a network, for example by how much transition weight flows into them (in-strength) or how often they lie on shortest paths (betweenness); before such a ranking is interpreted, it should be shown not to depend on a few sessions. `centrality_stability_htna()` runs Nestimate's `centrality_stability()` on the network and returns an `htna_stability` object (an `htna_stability_group` for a grouped network). Printing the result shows the coefficients, `plot.htna_stability()` draws the correlation traces, and `as.data.frame.htna_stability()` returns the coefficients or the traces as data frames.

For the stability of the edge weights use `casedrop_reliability_htna()`; for edge-level intervals use `bootstrap_htna()`.

Usage

```
centrality_stability_htna(
  x,
  measures = c("InStrength", "OutStrength", "Betweenness"),
  iter = 1000L,
  drop_prop = seq(0.1, 0.9, by = 0.1),
  threshold = 0.7,
  certainty = 0.95,
  method = "pearson",
  centrality_fn = NULL,
  loops = FALSE,
  seed = NULL,
  ...
)
```

Arguments

`x`	An htna network from `build_htna()` or an `htna_group`.
`measures`	Centrality measures to assess (default `c("InStrength", "OutStrength", "Betweenness")`).
`iter`	Resamples per drop proportion (default 1000).
`drop_prop`	Proportions of sessions to drop (default `seq(0.1, 0.9, by = 0.1)`).

threshold	Correlation the reduced networks must reach (default 0.7).
certainty	Probability of reaching threshold required at a drop proportion (default 0.95).
method	Correlation: "pearson" (default), "spearman" or "kendall".
centrality_fn	A function computing the centralities from a weight matrix; NULL (default) uses the built-in measures.
loops	Keep self-transitions when computing centralities (default FALSE).
seed	Random seed (default NULL, no seed is set).
...	Further arguments passed to Nestimate::centrality_stability(), for example actor (drop whole actors instead of sessions; Nestimate 1.0.19 or later).

Value

For a network, an object of class c("htna_stability", "net_stability") holding the CS coefficients (cs, one per measure), the correlation traces and the settings. For an htna_group, a named list of such objects with class c("htna_stability_group", "list"). as.data.frame() returns the coefficients (one row per measure) or, with what = "correlations", the traces (one row per measure and drop proportion); summary() returns the traces.

Conditions

Errors of class htna_bad_input when x is not an htna network and htna_empty_input when a group is empty.

References

Epskamp, S., Borsboom, D., & Fried, E. I. (2018). Estimating psychological networks and their accuracy: A tutorial paper. *Behavior Research Methods*, 50(1), 195-212. doi:10.3758/s13428017-08621

See Also

plot.htna_stability(), as.data.frame.htna_stability(), casedrop_reliability_htna(), bootstrap_htna().

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
cs  <- centrality_stability_htna(net, iter = 30, seed = 1)
cs
as.data.frame(cs, what = "correlations")
```

certainty_htna *Estimate the closed-form certainty of the edge weights*

Description

`certainty_htna()` estimates the sampling uncertainty of the edge weights of an htna network in closed form, instead of by resampling. The edge weights of a relative transition network are estimated proportions, so their uncertainty can be derived analytically: for each edge the closed-form model gives a mean, a standard deviation, a confidence interval, a consistency range around the observed weight and a verdict on whether the edge is supported by the data. It serves the same purpose as a bootstrap, showing which edges are estimated precisely enough to interpret, at a fraction of its cost.

`certainty_htna()` computes these quantities for an htna network or for every cohort of a grouped network. With `inference = "stability"` an edge is supported when its interval lies within the consistency range; with `"threshold"` when it lies above `edge_threshold`. It runs Nestimate's `certainty()` on the network and returns a `net_certainty` object (a `net_certainty_group` for a grouped network).

Use `bootstrap_htna()` for the resampling estimate, which also covers frequency and co-occurrence networks, and `prune_htna()` with `method = "bootstrap"` to keep only supported edges.

Usage

```
certainty_htna(x, ...)
```

Arguments

`x`	An htna network from `build_htna()` or an `htna_group`.
`...`	Settings: `prior` (default 0.5), `ci_level` (0.05), `inference` (`"stability"` or `"threshold"`), `consistency_range` (`c(0.75, 1.25)`) and `edge_threshold`.

Value

An object of class `c("net_certainty", "net_bootstrap")` with `print()` and `summary()` methods, drawn by `plot_htna_bootstrap()`. `summary()` returns a data frame with one row per edge and columns `from`, `to`, `weight`, `mean`, `sd`, `p_value`, `sig`, `ci_lower`, `ci_upper`, `cr_lower` and `cr_upper`. A grouped network gives a `net_certainty_group`, a named list with one result per cohort.

See Also

`bootstrap_htna()` for the resampling estimate, `reliability_htna()`, `casedrop_reliability_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
cert <- certainty_htna(net)
head(summary(cert))
plot_htna_bootstrap(cert)
```

`compare_htna` *Describe the differences between heterogeneous transition networks*

Description

`compare_htna()` describes the differences between two heterogeneous transition networks, or between every pair of cohorts of a grouped network: overall, in how similar their weight matrices are, and in where the differences lie in the actor structure, within an actor (for example, how humans move between their own codes) or between actors (how the human hands over to the AI and back). It reports correlation, distance and similarity measures of the two weight matrices, the network-level metrics of each network, the difference of every edge, and a summary of the differences in each directed actor-pair block.

Networks that contain different subsets of the codes are aligned on the union of their codes, in actor order, with the missing codes treated as having no transitions; a code shared by both networks must belong to the same actor in both. Differences are `x - y`. `compare_htna()` runs Nestimate's `compare_model()` on the aligned networks and adds the actor-pair summary. It returns an `htna_comparison` object (an `htna_comparison_group` for a grouped network).

The comparison is descriptive and gives no test. Use `permutation_htna()` for a permutation test of the differences and `bayes_compare_htna()` for credible intervals; `plot_htna_diff()` draws the difference network. The tables of the result are returned by `summary()` and `as.data.frame()` (see htna_comparison-methods).

Usage

```
compare_htna(x, y = NULL, ...)
```

Arguments

`x`	An htna network, or an `htna_group`.
`y`	A second htna network; `NULL` (default) when `x` is an `htna_group`, in which case every pair of cohorts is compared.
`...`	Settings of the comparison: `scaling`, `measures` and `network`.

Value

For two networks, an object of class `c("htna_comparison", "net_comparison")`. For an `htna_group`, a named list of class `c("htna_comparison_group", "list")` with one comparison per cohort pair (for example `"Early vs Late"`). `summary()` returns the actor-pair table, one row per directed actor pair; `as.data.frame(x, what = )` returns the actor-pair, edge, metric or network table (see htna_comparison-methods).

Conditions

Errors of class `htna_bad_input` when `x` and `y` are not htna networks, when `y` is given with an `htna_group`, or when the networks use different actor types; `htna_empty_input` when a group has fewer than two cohorts; `htna_actor_conflict` when a shared code belongs to different actors.

See Also

htna_comparison-methods, `permutation_htna()`, `bayes_compare_htna()`, `plot_htna_diff()`.

Examples

```
data(human_ai)
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
comparison <- compare_htna(grp)
comparison
summary(comparison)
```

---

deprune_htna *Restore the edges removed by pruning*

Description

`deprune_htna()` restores the edges that `prune_htna()` removed: it returns the network with every removed edge put back at its original weight. Pruning sets weak edges aside rather than deleting them, so restoring them gives back the full network, for example to compare the pruned and the unpruned structure or to apply a different rule.

`deprune_htna()` runs Nestimate's `net_deprune()` with the actor partition kept. The rule stays recorded, so `reprune_htna()` can apply it again.

Usage

```
deprune_htna(x, ...)
```

Arguments

`x`	An htna network pruned with `prune_htna()`.
`...`	Unused.

Value

An htna network with the original weight matrix.

See Also

`prune_htna()`, `reprune_htna()`, `pruning_details_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
pruned <- prune_htna(net, threshold = 0.05)
deprune_htna(pruned)
```

`edge_betweenness_htna` *Compute the edge betweenness of an htna network*

Description

`edge_betweenness_htna()` computes the edge betweenness of an htna network. The betweenness of an edge is the number of shortest paths between pairs of codes that pass through it. Edges with high betweenness are the bridges of the interaction: transitions through which much of the flow between otherwise distant codes must pass. In a heterogeneous network the high-betweenness edges are often hand-overs between actors, which shows which exchanges hold the interaction together.

`edge_betweenness_htna()` computes the betweenness of every non-zero edge and returns a copy of the network whose edge weights are these scores, with the actor partition kept, so the result can be drawn with `plot_htna()` and tabulated with `as.data.frame.htna()`. By default a larger transition probability is a shorter distance (`invert = TRUE`), as in transition network analysis.

To draw the result directly use `plot_edge_betweenness_htna()`; for node-level betweenness use `centralities_htna()`.

Usage

```
edge_betweenness_htna(x, directed = TRUE, invert = TRUE)
```

Arguments

`x`	An htna network from `build_htna()`.
`directed`	Follow edge direction in shortest paths. Default `TRUE`.
`invert`	Use `1 / weight` as the length of an edge. Default `TRUE`.

Value

An htna network of class `c("htna_edge_betweenness", class(x))` whose edge weights are edge-betweenness scores at every position where `x` has a non-zero transition. `as.data.frame()` returns one row per edge with its score in `weight`.

Conditions

`htna_bad_input` when `x` is not an htna network.

See Also

`plot_edge_betweenness_htna()`, `centralities_htna()`, `as.data.frame.htna()`

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
eb  <- edge_betweenness_htna(net)
head(as.data.frame(eb))
```

`extract_meta_paths` *Count recurring paths and meta-paths in the sessions*

Description

`extract_meta_paths()` counts the recurring paths and meta-paths in the sessions of an htna network. In a heterogeneous network every code belongs to an actor, so each run of codes in a session has two readings: the concrete path (for example `Request->Plan->Execute`) and its meta-path, the sequence of actor types it passes through (`Human->AI->AI`). Meta-paths describe the turn-taking between the actors, such as how often the human intervenes between two AI actions, while the concrete paths say what is done in each turn. Counting both shows which interaction patterns recur and whether they occur more often than the frequencies of their codes would predict.

`extract_meta_paths()` counts every path of the requested lengths in the sessions of an htna network. With `level = "state"` (default) it returns concrete paths, each tagged with its meta-path; with `level = "type"` it returns the meta-paths, summed over their concrete instances. A `schema` restricts the search to one template whose elements can be actor types (any code of that actor), concrete codes, or `"*"` (any element), freely mixed: `"Human->Ask->*"` is any human code, then the AI code `Ask`, then anything. With `type = "gapped"`, the elements of a path are `gap + 1` positions apart instead of adjacent.

It returns a data frame with one row per path and gap. For each path the table gives its count, the number and share (`support`) of sessions that contain it, its share among the paths of the same length and gap (`frequency`), and its `lift`: the count divided by the count expected if the elements occurred independently with their overall frequencies.

To compare paths between groups of sessions use `sequence_compare_htna()`; for unordered co-occurrence use `association_rules_htna()`.

Usage

```
extract_meta_paths(
  x,
  level = c("state", "type"),
  length = 2:4,
  schema = NULL,
  type = c("contiguous", "gapped"),
  gap = 1L,
  min_count = 1L,
  min_support = 0,
  min_lift = 0,
  start = NULL,
```

```
  end = NULL,
  contain = NULL
)
```

Arguments

x	An htna network from build_htna().
level	"state" (default) for concrete paths with their meta-path, or "type" for meta-paths only.
length	Path lengths to count (default 2:4); ignored when schema is given.
schema	A template of elements separated by "->"; each element is an actor type, a code or "*". NULL (default) counts all paths.
type	"contiguous" (default) for adjacent elements, or "gapped" for elements gap + 1 positions apart.
gap	Gap or gaps for type = "gapped" (default 1); several values give one set of rows per length and gap.
min_count	Minimum number of occurrences (default 1).
min_support	Minimum share of sessions containing the path (default 0).
min_lift	Minimum lift (default 0, no filter).
start, end, contain	Keep only paths whose first element is in start, whose last element is in end, or that contain every element of contain (default NULL); elements are codes or, with level = "type", actor types.

Value

A data frame of class c("htna_meta_paths", "htna_paths", "data.frame") with one row per path and gap, sorted by length, gap and decreasing count, and columns schema (the path), meta_schema (its meta-path; only with level = "state"), length, gap, count, n_sequences, support, frequency and lift. The number of sessions and the level are kept as attributes and shown by print().

Conditions

Errors of class htna_bad_input for an unknown element in schema or a zero gap with type = "gapped", and htna_unassigned_node when x has no actor partition.

See Also

print.htna_paths(), sequence_compare_htna(), association_rules_htna().

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
extract_meta_paths(net, level = "type", length = 3)
extract_meta_paths(net, schema = "Human->AI->Human")
extract_meta_paths(net, schema = "Human->Ask->*", min_lift = 1)
extract_meta_paths(net, length = 3, type = "gapped", gap = 1:2)
```

frequencies_htna *Count the codes of each actor in a network*

Description

`frequencies_htna()` counts the codes of each actor in an htna network: how often each code occurs and how each actor distributes its activity over its own codes, such as which requests a human makes most often and which actions the AI takes. These counts describe a heterogeneous network before its transitions are read.

`frequencies_htna()` counts the occurrences of every code in the sessions of an htna network and reports each code under its actor type, with its share of that actor's events. The proportions therefore sum to one within each actor, not over all events. It runs Nestimate's `state_distribution()` on the network and returns a data frame with one row per actor and code.

`plot_frequencies_htna()` draws the table. `state_frequencies_htna()` gives the pooled table without actors, with shares of all events; `state_distribution_htna()` returns the same table as this function.

Usage

```
frequencies_htna(x)
```

Arguments

`x`	An htna network from `build_htna()`, or an `htna_group`, in which case the codes are counted in every cohort.

Value

A data frame with one row per actor and code, sorted by decreasing count, and columns `group` (the actor type), `state`, `count` and `proportion` (the code's share of its actor's events). For a grouped network, one row per cohort, actor and code, with columns `group` (the cohort), `actor` (the actor type), `state`, `count` and `proportion` (the share of that actor's events in the cohort).

Conditions

An error of class `htna_bad_input` when `x` is neither an htna network nor an `htna_group`.

See Also

`plot_frequencies_htna()`, `state_frequencies_htna()`, `mosaic_plot_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
frequencies_htna(net)
```

```
# one table for all cohorts
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
frequencies_htna(grp)
```

`htna_comparison-methods` *Print, summarize and tabulate an htna network comparison*

Description

`print()`, `summary()` and `as.data.frame()` read an `htna_comparison` object from `compare_htna()`, which describes how two heterogeneous transition networks differ edge by edge. Beyond the overall similarity and distance measures of the two weight matrices, the comparison splits the differences by actor-pair block: the transitions within each actor (for example Human to Human) and between actors (Human to AI, AI to Human). The actor-pair table shows where in the actor structure the two networks differ most.

`print()` shows the overall measures and the actor-pair table; for a grouped comparison it shows, for every cohort pair, the correlation and mean absolute difference of the edge weights and the actor-pair table. `summary()` returns the actor-pair table. `as.data.frame()` returns any of the comparison's tables, chosen with `what`; for a grouped comparison the tables of every pair are stacked with a leading `comparison` column.

Usage

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

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

## S3 method for class 'htna_comparison'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  what = c("actor_pairs", "edges", "metrics", "network")
)

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

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

## S3 method for class 'htna_comparison_group'
as.data.frame(
```

```
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  what = c("actor_pairs", "edges", "metrics", "network")
)
```

Arguments

x, object	An htna_comparison or htna_comparison_group from compare_htna().
...	Unused.
row.names, optional	Ignored; present for the generic.
what	Table to return: "actor_pairs" (default; one row per directed actor pair), "edges" (one row per edge of the aligned networks), "metrics" (one row per similarity or distance measure) or "network" (one row per network-level metric, with the values of both networks).

Value

print() returns x invisibly. summary() returns the actor-pair data frame. as.data.frame() returns a data frame:

- "actor_pairs": source_actor, target_actor, n_cells, n_nonzero_x, n_nonzero_y, n_changed, mean_difference, mean_absolute_difference, rms_difference, max_absolute_difference; differences are x - y.
- "edges": source, target, weight_x, weight_y, the difference measures of each edge, source_actor and target_actor.
- "metrics": category, metric, value.
- "network": metric, x, y.

For a grouped comparison every table has a first column comparison (for example "Early vs Late"), and the first-named cohort is x.

See Also

compare_htna(), plot.htna_comparison_group()

Examples

```
data(human_ai)
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
comparison <- compare_htna(grp)
comparison
summary(comparison)
as.data.frame(comparison, what = "metrics")
```

htna_palette *Colours of the actor types in htna plots*

Description

htna_palette gives the default fill colours of the actor types in the network plots of htna: the first actor type takes the first colour, the second the second, and so on. With more than six actor types the colours are recycled with a warning, so pass a longer vector to group_colors in that case. Each plot also gives the actor types different node shapes, so the actor types are not distinguished by colour alone.

Usage

```
htna_palette
```

Format

A character vector of six hexadecimal colours.

Value

A character vector of six hexadecimal colours.

See Also

plot_htna(), plot_htna_diff(), plot_htna_bootstrap()

Examples

```
htna_palette
```

human_ai *Human and AI turns in vibe-coding sessions*

Description

human_ai is a coded dataset of the turns in 429 human-AI pair-programming ("vibe coding") sessions in 34 projects, held between 5 February and 8 March 2026. The data are the standard example for a heterogeneous transition network with two actor types: six human codes (Check, Frustrate, Inquire, Refine, Request, Specify) and six AI codes (Ask, Delegate, Execute, Plan, Repair, Report). In each session a person directs an AI coding assistant; every turn is coded with one or more interaction codes, and the codes of both actors are interleaved in the order in which they occurred.

The codes are a simplified version of the original coding, with near-synonyms merged. AI codes: Investigate, Ask and Inquire became Ask; Explain and Report became Report. Human codes: Command and Request became Request; Correct and Verify became Check; Interrupt and Frustrate became Frustrate. The phase column splits the sessions into an early and a late half for grouped examples. The same rows are available split by actor type in human_simplified and ai_simplified, and the code-to-actor lookup in human_ai_codebook.

Usage

```
human_ai
```

Format

A data frame with 19,347 rows (10,796 human and 8,551 AI codes) and 11 columns, one row per code:

message_id Integer identifier of the turn; the codes of one turn share it.

project Project identifier, `"Project_1"` to `"Project_34"`.

session_id Session identifier (the session key for `build_htna()`).

timestamp Integer Unix time of the session date (day resolution).

session_date Date of the row, `"YYYY-MM-DD"`.

code Simplified interaction code.

cluster High-level cluster of the original code, kept for reference; because merged codes may come from different clusters, one simplified code can carry more than one cluster label.

code_order Order of the code within its turn.

order_in_session Position of the row in its session, counting both actors (the order key for `build_htna()`).

actor_type `"Human"` or `"AI"`.

phase Factor with levels `"Early"` and `"Late"`: sessions ordered by their first date (session id breaking ties) and split in half, 214 early and 215 late sessions.

Source

Saqr, M. (2026). Human-AI vibe coding interaction study. https://saqr.me/blog/2026/human-ai-interaction-cograp
Derived from the `human_long` and `ai_long` data of the 'Nestimate' package; the build script is `data-raw/human_ai.R` in the package sources.

See Also

human_simplified, ai_simplified, human_ai_codebook, `build_htna()`

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
plot_htna(net)
```

human_ai_codebook *Code-to-actor codebook for the vibe-coding data*

Description

human_ai_codebook is the codebook of human_ai: the actor type of each of its twelve codes. A codebook of this form declares the actor partition when the event data have no actor-type column: it is passed to build_htna() as node_groups.

Usage

```
human_ai_codebook
```

Format

A data frame with 12 rows, one per code, and 2 columns:

code Simplified interaction code.

actor_type "Human" or "AI".

Source

Derived from human_simplified and ai_simplified.

See Also

human_ai, build_htna()

Examples

```
data(human_ai, human_ai_codebook)
human_ai_codebook
net <- build_htna(human_ai, node_groups = human_ai_codebook)
summary(net)
```

human_simplified *Human turns of the vibe-coding sessions*

Description

human_simplified is the human part of human_ai: its human rows, as one data frame for one actor type. Together with ai_simplified it is the input for the named-list form of build_htna(), in which each actor type comes in its own data frame.

Usage

```
human_simplified
```

Format

A data frame with 10,796 rows and 10 columns: the columns of human_ai without phase; actor_type is always "Human".

Source

Derived from the human_long data of the 'Nestimate' package (see human_ai).

See Also

human_ai, ai_simplified, human_ai_codebook

Examples

```
data(human_simplified, ai_simplified)
net <- build_htna(list(Human = human_simplified, AI = ai_simplified))
plot_htna(net)
```

---

markov_order_test_htna

Test the Markov order of the sessions

Description

markov_order_test_htna() tests the Markov order of the session sequences of an htna network (or of raw sequence data). The Markov-order test compares the fit of models of increasing order (zero, one, two and so on) with likelihood-ratio statistics, judged against a permutation null, and reports the information criteria of each order. A transition network is a first-order Markov model: the next code is assumed to depend only on the current one. When the next code also depends on the codes before it, a first-order network misses part of the process and a higher order describes it better.

The orders are tested in turn: the selected optimal_order is the highest order reached before the first step that is not significant at alpha, and equals max_order when every step is significant, in which case the search should be widened. The orders preferred by BIC and AIC are reported alongside. markov_order_test_htna() runs Nestimate's markov_order_test() on the sequences and returns a net_markov_order object.

Use it before interpreting a first-order network, together with reliability_htna() and casedrop_reliability_htna() for the stability of the estimated weights.

Usage

```
markov_order_test_htna(data, ...)
```

Arguments

data	An htna network from build_htna(), or sequence data (a wide data frame or a list of character vectors).
...	Settings: max_order (default 3), n_perm (permutations, 500), alpha (0.05), parallel (FALSE), n_cores (2), seed and dof ("paths" or "observed").

Value

An object of class net_markov_order with print(), summary() and plot() methods. summary() returns a data frame with one row per tested order and columns order, loglik, AIC, BIC, df, g2, p_permutation, p_asymptotic and significant. The object also holds optimal_order, bic_order and aic_order.

See Also

reliability_htna(), casedrop_reliability_htna(), centrality_stability_htna().

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
# n_perm = 20 keeps the example fast; use the default 500 in practice.
mo <- markov_order_test_htna(net, max_order = 2, n_perm = 20, seed = 1)
mo
summary(mo)
```

---

mosaic_plot_htna *Plot a chi-square mosaic of the transitions of an htna network*

Description

mosaic_plot_htna() plots a chi-square mosaic of the transitions of an htna network. A mosaic plot of a transition table shows, for every pair of codes, the share of all transitions that go from the first code to the second (the area of the tile) and how far that count lies from what independence of the source and target codes would predict (the colour of the tile, from the standardized residual of a chi-square test). Over-represented transitions are shaded blue and under-represented ones red. In a heterogeneous network the plot shows which hand-overs between actors are more or less frequent than the overall frequencies of the codes imply.

mosaic_plot_htna() draws the mosaic of the transition counts of an htna network with Nestimate's mosaic_plot(), with the axis labels coloured by actor type. The chi-square test needs counts, so build the network with method = "frequency". With n_perm, p-values are obtained by permutation (set seed for a reproducible result). It returns a ggplot object.

For the marginal frequencies of the codes use plot_frequencies_htna().

Usage

```
mosaic_plot_htna(x, ...)
```

Arguments

x	An htna network built with method = "frequency".
...	Further arguments of Nestimate::mosaic_plot(), such as n_perm and seed. With Nestimate 1.0.13 or later, nested = TRUE draws residuals that allow for transitions nested in sessions, or in the actors named by actor; nested_method chooses the test and report = TRUE adds the overall test as the subtitle.

Value

A ggplot object (or a gtable for some inputs), which is drawn when printed.

See Also

plot_frequencies_htna(), frequencies_htna(), build_htna()

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type",
                  method = "frequency")
mosaic_plot_htna(net, n_perm = 50, seed = 1)
```

---

permutation_htna *Test differences between networks by permutation*

Description

permutation_htna() tests by permutation whether two heterogeneous transition networks, for example of two groups of learners or two phases of a study, differ more than chance would produce. The sessions behind both networks are pooled, the group labels are shuffled many times, both networks are re-estimated on every shuffle, and each observed difference is compared with the shuffled ones. Each edge gets a p-value and an effect size (the observed difference divided by the standard deviation of the shuffled differences), and the networks as a whole are tested with the sum and the maximum of the absolute differences. Because whole sessions are shuffled, the nesting of transitions in sessions is respected.

permutation_htna() runs Nestimate's permutation() with the actor partition kept. It tests two htna networks estimated with the same method on the same codes, or every pair of cohorts of a grouped network. With many edges some fall below alpha by chance; set adjust (for example "BH") to correct for multiple testing. With measures, node centralities are tested as well. It returns a net_permutation object, or one per cohort pair for a grouped network.

A small p-value says that a difference is larger than the shuffles produce, not how large or how probable it is, which bayes_compare_htna() estimates. compare_htna() gives descriptive differences per actor pair, and plot_htna_diff() draws the result.

Usage

```
permutation_htna(x, y = NULL, ...)
```

Arguments

x, y	Two htna networks, or an htna_group in x with y = NULL, in which case every pair of cohorts is tested. Networks that do not hold the same codes are compared over the codes of both; a code a network lacks has no transitions in it.
...	Settings: iter (shuffles, default 1000), alpha (0.05), paired (FALSE), adjust (multiplicity correction, "none"), measures (centralities to test), nlambda, seed and actor (shuffle whole actors).

Value

For two networks, an object of class net_permutation; for a grouped network, a named list of class c("net_permutation_group", "list") with one result per cohort pair (for example "Early vs Late"). Both have print() and summary() methods; summary() returns a data frame with one row per edge (and pair) and columns from, to, weight_x, weight_y, diff, effect_size, p_value and sig. Differences are x - y.

See Also

bayes_compare_htna(), compare_htna(), plot_htna_diff().

Examples

```
data(human_ai)
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
perm <- permutation_htna(grp, iter = 50, seed = 1)
perm
head(summary(perm))
```

---

plot.htna_comparison_group

Plot the comparisons of a grouped network

Description

plot() draws the plots of a descriptive network comparison for every cohort pair of a grouped comparison from compare_htna(): a scatter plot of the edge weights of the two networks, a heatmap of the edge differences, a histogram of the differences, or the weight distributions. It returns, invisibly, a named list with the plot of every cohort pair.

Usage

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

Arguments

x	An htna_comparison_group from compare_htna().
...	Plot settings: type ("scatter", "heatmap", "diff_hist", "weight_dist" or "all") and combined.

Value

Invisibly, a named list with the plot of every cohort pair.

Conditions

An error of class htna_empty_input when x is empty.

See Also

compare_htna(), plot_htna_diff() for the difference network.

Examples

```
data(human_ai)
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
plot(compare_htna(grp), type = "scatter")
```

---

plot.htna_stability *Plot centrality stability against the proportion of sessions dropped*

Description

plot() plots the case-dropping stability of an htna_stability result from centrality_stability_htna(): for each centrality measure, how closely the centralities of networks estimated from a subset of sessions correlate with those of the full network as more sessions are dropped. A measure whose mean correlation stays above the threshold until a large share of the sessions is dropped has a node ranking that does not depend on a few sessions; a measure whose line falls early cannot be interpreted with confidence. When the result was computed with actor, the horizontal axis is the proportion of actors dropped.

The plot draws one line per measure (distinguished by colour, line type and point shape), a ribbon of plus or minus one standard deviation of the correlations, and a dashed line at the stability threshold. The grouped method draws one such chart per cohort, titled with the cohort name. The charts are drawn with 'ggplot2', which must be installed, and are returned invisibly as ggplot objects. The CS coefficients themselves are reported by print.htna_stability() and returned as a table by as.data.frame.htna_stability().

Usage

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

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

Arguments

x	A result of centrality_stability_htna(): an htna_stability or, for a grouped network, an htna_stability_group.
...	Unused.

Value

Invisibly, the ggplot object (for htna_stability) or a named list of ggplot objects, one per cohort (for htna_stability_group). The charts are printed.

Conditions

An error of class htna_missing_package is raised when 'ggplot2' is not installed.

See Also

centrality_stability_htna(), as.data.frame.htna_stability()

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
cs  <- centrality_stability_htna(net, iter = 20, seed = 1)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(cs)
}
```

---

plot_centralities *Plot node centralities of an htna network*

Description

plot_centralities() plots the node centralities of an htna network: the value of each centrality measure for every code, one panel per measure, so the codes that hold the central positions of the interaction can be read at a glance and compared across measures. Colouring the bars by actor type shows whether the central codes belong to one actor or are shared between them.

plot_centralities() draws horizontal bars, one per code, in one facet per measure, sorted by value within each facet by default and labelled with the value. It accepts an htna network, a grouped network or a table from centralities_htna(). For a grouped network each panel shows one line-and-point profile per cohort over the same codes, so differences between cohorts read along one axis; by is then not used. The chart is a 'ggplot2' chart and needs that package; it is returned as a ggplot object.

The values themselves are returned by centralities_htna().

Usage

```
plot_centralities(
  x,
 measures = c("OutStrength", "InStrength", "ClosenessIn", "ClosenessOut", "Closeness",
    "Betweenness", "BetweennessRSP", "Diffusion", "Clustering"),
  by = c("state", "group"),
  reorder = TRUE,
  ncol = 3,
  scales = c("free_x", "fixed"),
  colors = NULL,
  labels = TRUE,
  ...
)
```

Arguments

x	An htna network, an htna_group, or a table from centralities_htna().
measures	Measures to plot. Default: all nine.
by	"state" (default) gives every code its own colour; "group" colours the bars by actor type.
reorder	Sort the codes by value within each measure. Default TRUE.
ncol	Number of facet columns. Default 3.
scales	Facet scales: "free_x" (default) or "fixed".
colors	Fill colours, recycled over codes or actor types. Default NULL (Okabe-Ito for codes, htna_palette for actor types).
labels	Print the value next to each bar. Default TRUE.
...	Further arguments for centralities_htna() when x is a network.

Value

A ggplot object, drawn when printed.

Conditions

htna_missing_package when 'ggplot2' is not installed; htna_bad_input when none of measures is in the table or by = "group" is asked for a table without an actor column.

See Also

centralities_htna(), centrality_stability_htna()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  data(human_ai)
  net <- build_htna(human_ai, actor_type = "actor_type")
  plot_centralities(net, measures = c("InStrength", "OutStrength",
```

```
                                        "Betweenness"), by = "group")

  grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
  plot_centralities(grp, measures = c("InStrength", "Betweenness"))
}
```

`plot_edge_betweenness_htna` *Plot the edge betweenness of an htna network*

Description

`plot_edge_betweenness_htna()` plots the edge-betweenness network of an htna network: each transition is drawn with a width given by the number of shortest paths that pass through it, in the layout and actor styling of `plot_htna()`. The widest edges are the bridges of the interaction, the transitions through which the flow between the codes of the actors is channeled.

`plot_edge_betweenness_htna()` computes the scores with `edge_betweenness_htna()` (unless `x` already holds them) and draws all edges with a non-zero score, unless `minimum` is given. It is called for the plot and returns, invisibly, the value of `plot_htna()`.

Usage

```
plot_edge_betweenness_htna(x, directed = TRUE, invert = TRUE, ...)
```

Arguments

`x`	An htna network, or a result of `edge_betweenness_htna()`.
`directed`, `invert`	Passed to `edge_betweenness_htna()` when `x` is an htna network; ignored otherwise. Defaults `TRUE`.
`...`	Further arguments of `plot_htna()`, such as `minimum`, `layout` or `group_colors`.

Value

Called for the plot. Invisibly, the value of `plot_htna()`.

Conditions

`htna_bad_input` when `x` is not an htna network.

See Also

`edge_betweenness_htna()`, `plot_htna()`

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
plot_edge_betweenness_htna(net)
plot_edge_betweenness_htna(net, minimum = 5)
```

---

plot_frequencies_htna *Plot the frequencies of the codes of an htna network*

Description

`plot_frequencies_htna()` plots the frequencies of the codes of an htna network: the number of occurrences of every code, coloured by its actor type. Knowing how often each code occurs helps before transitions are interpreted: a rare code can have extreme transition probabilities that rest on a few events, and the balance between the actor types (how much the human and the AI each contribute) frames every later comparison.

`plot_frequencies_htna()` draws one of three views: a treemap of the codes (`"treemap"`, the default, drawn with Nestimate's `plot_state_frequencies()`), a single bar chart of all codes sorted by count (`"bars"`), or one bar chart per actor type (`"facet"`). The bar views are 'ggplot2' charts and need that package. It returns the frequency object for the treemap and a `ggplot` object for the bar views.

The counts and proportions behind the plot are returned as a table by `frequencies_htna()`. For how the transitions from each code deviate from independence use `mosaic_plot_htna()`.

Usage

```
plot_frequencies_htna(x, view = c("treemap", "bars", "facet"), ...)
```

Arguments

`x`	An htna network from `build_htna()`.
`view`	`"treemap"` (default), `"bars"` or `"facet"`.
`...`	Further arguments of `Nestimate::plot_state_frequencies()` for the treemap; ignored by the bar views. An argument the treemap does not take raises a warning (Nestimate 1.0.12 or later).

Value

For `"treemap"`, invisibly, the frequency object of class `state_freq` (its table via `as.data.frame()`). For `"bars"` and `"facet"`, a `ggplot` object, which can be modified with `+`.

Conditions

`htna_missing_package` when 'ggplot2' is not installed for a bar view; `htna_bad_input` when a bar view is asked for an object that is not an htna network.

See Also

frequencies_htna(), state_frequencies_htna(), state_distribution_htna(), mosaic_plot_htna()

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
plot_frequencies_htna(net)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot_frequencies_htna(net, view = "facet")
}
```

---

plot_htna *Plot a heterogeneous transition network*

Description

plot_htna() plots a heterogeneous transition network: the codes of every actor type as nodes and the transitions between them as directed edges whose width and label give the transition weight. Each actor type has its own colour and node shape and occupies its own arc of the circular layout, so the transitions within an actor type run along its arc and the hand-overs between actor types cross the center. The plot shows which transitions dominate the interaction and how the actors pass control to each other.

plot_htna() draws one network, or one plot per cohort for a grouped network, each titled with its cohort name; arrange several plots with par(mfrow = ...). Edges with a weight below minimum are hidden. The first actor type is placed on the left of the circular layout. plot() on an htna network or htna_group calls plot_htna(). It is called for the plot and returns, invisibly, the value of cograph::plot_htna() for one network, or x for a grouped network.

A start and an end state (build_htna(start = , end =)) are not an arc of the circular layout. The actor types keep the positions they have without them, and the start state is placed at the top centre and the end state at the bottom centre, in the colour and shape of the "Boundary" type. plot_htna_bootstrap() and plot_htna_diff() place them the same way.

To show the difference between two networks use plot_htna_diff(); to show only the edges that survive bootstrapping use plot_htna_bootstrap().

Usage

```
plot_htna(
  x,
  layout = "circular",
  group_colors = htna_palette,
  group_shapes = htna_shape_palette,
  minimum = 0.05,
  ...
)
```

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

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

Arguments

x	An htna network from build_htna(), or an htna_group.
layout	Node layout. Default "circular"; "polygon" places three or more actor types on the sides of a polygon. Start and end states are placed as anchors in the circular layout only; in another layout they are laid out as one more actor type.
group_colors	Colours of the actor types, in actor order. Default htna_palette.
group_shapes	Node shapes of the actor types, in actor order. Default c("circle", "square", "diamond", "triangle", "pentagon", "hexagon", "star", "cross").
minimum	Smallest edge weight drawn. Default 0.05.
...	Further drawing arguments of cograph::plot_htna() and cograph::splot(), such as title, threshold or edge_labels.

Value

Called for the plot. Invisibly, the value of cograph::plot_htna() for one network, or x for a grouped network.

Conditions

htna_empty_input when a grouped network has no cohorts.

See Also

build_htna(), plot_htna_diff(), plot_htna_bootstrap(), htna_palette

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
plot_htna(net)
plot_htna(net, minimum = 0.1)

grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
plot(grp)
```

plot_htna_bootstrap *Plot the bootstrap-assessed edges of an htna network*

Description

plot_htna_bootstrap() plots the bootstrap-assessed edges of an htna network: it shows which transitions of a heterogeneous network are stable enough to interpret. Edges that are significant across resamples are drawn solid, the others dashed (or hidden), so the reader sees which within-actor transitions and which hand-overs between actors the data support. A bootstrap of a transition network resamples the sessions many times and records how often each edge is present and how much its weight varies.

plot_htna_bootstrap() draws the result of bootstrap_htna(), or the closed-form result of certainty_htna(), in the layout and actor styling of plot_htna(). With display = "significant" only the significant edges are drawn; show_ci = TRUE adds the confidence interval of each edge weight. A grouped bootstrap is drawn as one plot per cohort, titled with its cohort name. plot() on a bootstrap result calls this function. It is called for the plot and returns, invisibly, the value of the drawing call, or boot for a grouped bootstrap.

The table behind the plot (one row per edge, with its weight, interval and p-value) is returned by summary() of the bootstrap result.

Usage

```
plot_htna_bootstrap(boot, group_colors = htna_palette, ...)

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

Arguments

boot, x	A result of bootstrap_htna() (an htna_bootstrap or an htna_bootstrap_group) or of certainty_htna().
group_colors	Colours of the actor types, in actor order. Default htna_palette.
...	Further drawing arguments: display ("styled" (default), "significant" or "full"), show_ci (default FALSE), show_stars (default TRUE), title, and the arguments of cograph::splot().

Value

Called for the plot. Invisibly, the value of the drawing call for one bootstrap, or boot for a grouped bootstrap.

Conditions

htna_empty_input for a grouped bootstrap without cohorts; htna_bad_input when boot is not a result of bootstrap_htna() or certainty_htna() on an htna network.

See Also

bootstrap_htna(), plot_htna(), certainty_htna()

Examples

```
data(human_ai)
net  <- build_htna(human_ai, actor_type = "actor_type")
boot <- bootstrap_htna(net, iter = 50, seed = 1)
plot_htna_bootstrap(boot)
plot_htna_bootstrap(boot, display = "significant")
```

---

plot_htna_diff *Plot the difference between two htna networks*

Description

plot_htna_diff() plots the difference between two htna networks: for every transition, how much more (or less) likely it is in one network than in another, for example in the late sessions compared with the early ones. Drawn in the layout of the two heterogeneous networks, the difference network shows whether the change lies within an actor type or in the hand-overs between actors. When the difference comes from a permutation test or a Bayesian comparison, only the differences that the test supports are emphasized.

plot_htna_diff() accepts two htna networks and draws x - y, with positive differences in pos_color and negative differences in neg_color; networks whose node sets differ are aligned on the union of their nodes. It also accepts the result of permutation_htna() or bayes_compare_htna() for two networks, which adds the significance overlay (significant edges solid, the others dashed or hidden with show_nonsig), and the grouped results of either test, drawn as one plot per cohort pair titled with the pair name. It is called for the plot and returns, invisibly, the value of the drawing call, or x for a grouped test result.

For a table of the differences by actor pair use compare_htna().

Usage

```
plot_htna_diff(
  x,
  y = NULL,
  pos_color = "#009900",
  neg_color = "#C62828",
  group_colors = htna_palette,
  ...
)
```

Arguments

x	An htna network, or a result of permutation_htna() or bayes_compare_htna() (for two networks or for a grouped network).
y	A second htna network, when x is an htna network. Default NULL.
pos_color	Colour of positive differences. Default "#009900".
neg_color	Colour of negative differences. Default "#C62828".
group_colors	Colours of the actor types, in actor order. Default htna_palette.
...	Further drawing arguments of cograph::splot(), such as title, minimum or, for a test result, show_nonsig.

Value

Called for the plot. Invisibly, the value of the drawing call, or x for a grouped test result.

Conditions

htna_bad_input when x and y are not two htna networks with the same actor types, or a test result was not computed on htna networks; htna_empty_input for an empty grouped result.

See Also

compare_htna(), permutation_htna(), bayes_compare_htna(), plot_htna()

Examples

```
data(human_ai)
early <- build_htna(subset(human_ai, phase == "Early"),
                    actor_type = "actor_type")
late  <- build_htna(subset(human_ai, phase == "Late"),
                    actor_type = "actor_type")
plot_htna_diff(late, early)

# a permutation test on the grouped network: one plot per pair of phases
grp  <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
perm <- permutation_htna(grp, iter = 50, seed = 1)
plot_htna_diff(perm)
```

---

plot_sequences *Plot sequences*

Description

plot_sequences() plots the sequences behind a model; it is a generic. For an htna network it draws the sessions with sequence_plot_htna(), so plot_sequences(net) gives the same plot as sequence_plot_htna(net). The 'tna' package exports a generic of the same name; whichever package is attached last provides plot_sequences().

Usage

```
plot_sequences(x, ...)
```

Arguments

x	An object to plot; for an htna network, see sequence_plot_htna().
...	Arguments for the method.

Value

Called for the plot; the value is defined by the method (for an htna network, see sequence_plot_htna()).

See Also

sequence_plot_htna()

Examples

```
data(human_ai)
net <- build_htna(subset(human_ai, project == "Project_9"),
                  actor_type = "actor_type")
plot_sequences(net)
```

---

print.htna_bootstrap_group

Print and summarize the bootstraps of a grouped network

Description

print() and summary() read the bootstraps of a grouped network: print() shows each cohort's report in turn under its name, and summary() stacks the cohorts' edge tables into one data frame, so the edges of the cohorts can be read and compared together. A grouped network bootstrapped with bootstrap_htna() gives one bootstrap per cohort.

Usage

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

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

Arguments

x, object	An htna_bootstrap_group from bootstrap_htna().
...	Passed to the method of each cohort.

Value

print() returns x invisibly. summary() returns a data frame with one row per cohort and edge: a first column group, then the columns of the per-cohort summary (from, to, weight, mean, sd, p_value, sig, ci_lower, ci_upper, cr_lower, cr_upper).

See Also

bootstrap_htna(), plot_htna_bootstrap().

Examples

```
data(human_ai)
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
boots <- bootstrap_htna(grp, iter = 20, seed = 1)
boots
head(summary(boots))
```

---

print.htna_paths *Print a table of paths or meta-paths*

Description

print() prints the result of extract_meta_paths(): a header with the level (concrete paths or meta-paths), the template if one was given, the number of sessions, the lengths and gaps counted, and then the first n rows with support, frequency and lift rounded for reading. It returns x invisibly. The result is itself a data frame, so all rows and full precision are available with utils::head(), subset() or as.data.frame().

Usage

```
## S3 method for class 'htna_paths'
print(x, n = 10L, ...)
```

Arguments

x	A result of extract_meta_paths().
n	Number of rows to print (default 10).
...	Unused.

Value

x, invisibly.

See Also

extract_meta_paths().

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
paths <- extract_meta_paths(net, length = 3)
print(paths, n = 5)
```

`print.htna_stability` *Print centrality stability coefficients*

Description

`print()` prints the case-dropping stability (CS) coefficient of each centrality measure of an `htna_stability` result. The CS coefficient is the largest proportion of sessions that can be dropped while the correlation with the full-network centralities stays above the threshold with the required certainty. The header gives the number of resamples, the threshold and the drop proportions, and says when whole actors were dropped instead of sessions (`actor`). The grouped method prints one block per cohort. It returns `x` invisibly.

Usage

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

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

Arguments

`x`	A result of `centrality_stability_htna()`: an `htna_stability` or an `htna_stability_group`.
`...`	Unused.

Value

Invisibly, `x`, unchanged.

See Also

`as.data.frame.htna_stability()` for the coefficients and the correlation table as data frames, `plot.htna_stability()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
cs  <- centrality_stability_htna(net, iter = 20, seed = 1)
print(cs)
```

prune_htna *Remove weak edges from a heterogeneous transition network*

Description

prune_htna() removes weak edges from a heterogeneous transition network so that the plot and the descriptive measures focus on the transitions that carry the process. A relative transition network estimated from many sessions is usually dense: almost every code is followed by almost every other code at least once, and most of these transitions are rare. Because the weights are kept in the network and only set aside, pruning can be undone or re-applied.

prune_htna() sets to zero the edges that fail a rule: weights below a fixed threshold (method = "threshold", the default), the lowest share of the weights (method = "lowest"), or edges that are not supported by a bootstrap (method = "bootstrap", with boot from bootstrap_htna()). It runs Nestimate's net_prune() with the actor partition kept, so plot_htna() and the other functions work on the pruned network unchanged. It returns an htna network with the removed edges set to zero and the rule recorded.

deprune_htna() restores the removed edges, reprune_htna() removes them again, and pruning_details_htna() lists them.

Usage

```
prune_htna(x, ...)
```

Arguments

x	An htna network from build_htna().
...	Pruning rule: method ("threshold", default, "lowest" or "bootstrap"), threshold (default 0.1), lowest (default 0.05), level (default 0.5) and boot (a bootstrap result for method = "bootstrap").

Value

An htna network of the same class and structure as build_htna() returns, with the removed edges set to zero in its weight matrix and the rule recorded for deprune_htna(), reprune_htna() and pruning_details_htna().

See Also

deprune_htna(), reprune_htna(), pruning_details_htna(), bootstrap_htna().

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
pruned <- prune_htna(net, threshold = 0.05)
summary(pruned)
```

pruning_details_htna *List the edges removed by pruning*

Description

pruning_details_htna() lists the edges that prune_htna() removed: which transitions were set aside and how strong they were. Pruning changes what a network shows, so the removed edges should be reported alongside the pruned network.

pruning_details_htna() runs Nestimate's net_pruning_details() and returns a data frame with one row per removed edge and the weight each had before pruning. The pruning rule and the numbers of removed and retained edges are printed with the table.

Usage

```
pruning_details_htna(x, ...)
```

Arguments

x	An htna network pruned with prune_htna().
...	Unused.

Value

A data frame of class c("net_pruning_details", "data.frame") with one row per removed edge and columns from, to and weight. The rule is kept in the attributes method, threshold, lowest, cut_off, num_removed and num_retained, and is shown by print().

See Also

prune_htna(), deprune_htna(), reprune_htna().

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
pruned <- prune_htna(net, threshold = 0.05)
pruning_details_htna(pruned)
```

`reliability_htna` *Estimate the split-half reliability of a heterogeneous transition network*

Description

`reliability_htna()` estimates the split-half reliability of one or more htna networks: how far each network would look the same if it had been estimated from a different half of the sessions. The sessions are split at random into two halves many times, a network is estimated from each half, and the two networks are compared: the mean, median and maximum absolute difference of their edge weights and the correlation of their weights. High correlations and small deviations over the splits indicate that the estimated structure does not depend on which sessions happened to be observed.

When several networks are given (for example, built with different methods from the same data), each is assessed with the same splits. `reliability_htna()` runs Nestimate's `network_reliability()`, and the networks re-estimated on the splits keep the actor partition. It returns an `htna_reliability` object.

For the stability of the weights when increasing shares of the sessions are dropped use `casedrop_reliability_htna()`; for edge-level intervals use `bootstrap_htna()`.

Usage

```
reliability_htna(..., iter = 1000L, split = 0.5, scale = "none", seed = NULL)
```

Arguments

`...`	One or more htna networks from `build_htna()`.
`iter`	Number of random splits (default 1000).
`split`	Proportion of the sessions in the first half (default 0.5).
`scale`	Scaling of the weights before they are compared: `"none"` (default), `"minmax"`, `"standardize"` or `"proportion"`.
`seed`	Random seed (default `NULL`, no seed is set).

Value

An object of class `c("htna_reliability", "net_reliability")` with `print()`, `summary()` and `plot()` methods. `summary()` returns a data frame with one row per network and metric and columns `model`, `metric` (`mean_dev`, `median_dev`, `cor`, `max_dev`), `mean` and `sd`.

Conditions

An error of class `htna_empty_input` when no network is given, and of class `htna_bad_input` when an input is not an htna network.

See Also

`casedrop_reliability_htna()`, `bootstrap_htna()`, `centrality_stability_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
rel <- reliability_htna(net, iter = 30, seed = 1)
rel
summary(rel)
```

`reprune_htna` *Re-apply a recorded pruning rule*

Description

`reprune_htna()` re-applies the recorded pruning rule of a network and returns the pruned network with its actor partition. A network that has been pruned and then restored with `deprune_htna()` still carries the pruning rule; re-applying it removes the same weak edges again without restating the method and threshold.

`reprune_htna()` runs Nestimate's `net_reprune()` and returns an htna network with the edges that fail the recorded rule set to zero.

Usage

```
reprune_htna(x, ...)
```

Arguments

`x`	An htna network that was pruned with `prune_htna()` and restored with `deprune_htna()`.
`...`	Unused.

Value

An htna network with the edges that fail the recorded rule set to zero.

See Also

`prune_htna()`, `deprune_htna()`, `pruning_details_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
pruned <- prune_htna(net, threshold = 0.05)
reprune_htna(deprune_htna(pruned))
```

`sequence_compare_htna` *Compare short sequence patterns between groups of sessions*

Description

`sequence_compare_htna()` compares short sequence patterns between the cohorts of a grouped htna network, or between the groups of a single network given by `group`. It counts every k-gram (a run of k consecutive codes) in each group and finds the patterns that are more or less frequent in a group than expected if pattern and group were independent. Each pattern gets a standardized residual per group (Haberman, 1973) and, optionally, a permutation test (the session is the unit) or a chi-square test of whether its frequency differs between the groups, with p-values adjusted across all patterns. Two groups of sessions can differ not only in single transitions but in longer runs of codes, such as a human request followed by an AI plan and an AI execution.

With `level = "type"` every code is first replaced by its actor type, so the patterns are meta-paths such as `Human->AI->Human` and the comparison shows how the turn-taking between actors differs between the groups. `sequence_compare_htna()` runs Nestimate's `sequence_compare()` on the sequences and returns a `net_sequence_comparison` object.

For single transitions use `permutation_htna()` or `mosaic_plot_htna()`; for descriptive counts of patterns in one network use `extract_meta_paths()`.

Usage

```
sequence_compare_htna(x, group = NULL, level = c("state", "type"), ...)
```

Arguments

`x`	A grouped htna network (an `htna_group`), or an htna network together with `group`.
`group`	The grouping of the sessions when `x` is a single network: the name of a session-level column of the data the network was built from (for example `"phase"`), or a vector with one label per session (default `NULL`).
`level`	`"state"` (default) compares patterns of codes; `"type"` compares patterns of actor types.
`...`	Settings: `sub` (pattern lengths, default `3:5`), `min_freq` (minimum count in every group, 5), `test` (`"permutation"`, `"chisq"` or `"none"`), `iter` (permutations, 1000), `adjust` (multiplicity correction, `"fdr"`) and `actor` (reassign whole actors in the permutation test; needs Nestimate 1.0.13 or later).

Value

An object of class `net_sequence_comparison` with `print()`, `summary()` and `plot()` methods. `summary()` returns a data frame with one row per pattern and columns `pattern`, `length`, one `freq_<group>`, `prop_<group>` and `resid_<group>` column per group, `effect_size` and, when a test is run, `p_value`.

Conditions

An error of class `htna_bad_input` when `level = "type"` and `x` has no actor partition; `htna_missing_column` when `group` names a column the network does not hold.

References

Haberman, S. J. (1973). The analysis of residuals in cross-classified tables. *Biometrics*, 29(1), 205-220. doi:10.2307/2529686

See Also

`extract_meta_paths()`, `permutation_htna()`, `mosaic_plot_htna()`.

Examples

```
data(human_ai)
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
# iter = 50 keeps the example fast; use the default 1000 in practice.
sequence_compare_htna(grp, iter = 50)
sequence_compare_htna(grp, level = "type", iter = 50)
```

---

sequence_plot_htna *Plot the sessions of an htna network as sequences*

Description

`sequence_plot_htna()` plots the sessions of an htna network as sequences: every session is a row of coloured cells, one per event, so the reader sees how sessions begin and end, how long they run and whether some codes cluster early or late. A transition network pools the transitions of all sessions and so loses when each code occurs and how long the sessions are; the sequence plot restores that view. In a heterogeneous network the plot can separate the actor types, showing the course of each actor's own events or which actor acted at each step.

The plot is drawn with `Nestimate::sequence_plot()`. With `by = "state"` (the default) each cell is coloured by its code, and every session contributes one row per actor type holding that actor's events in order; the legend is split into one block per actor type. With `by = "group"` the sessions are kept whole and each cell is coloured by the actor type that acted at that step. `type` selects the display: a sequence index plot with one panel per actor type (`"index"`), a single carpet with a separator between the actor types (`"heatmap"`), or the share of each code at every position (`"distribution"`). A grouped network is drawn as one plot per cohort. `plot_sequences()` on an htna network calls this function. It is called for the plot and returns, invisibly, the description of the drawn plot (a list), or `net` for a grouped network.

Rows are ordered by clustering the sessions, which can take some seconds for several hundred sessions. For the network itself use `plot_htna()`; to test which sequence patterns differ between cohorts use `sequence_compare_htna()`.

Usage

```
sequence_plot_htna(
  net,
  by = c("state", "group"),
  type = c("index", "heatmap", "distribution"),
  grouped_legend = TRUE,
  ...
)

## S3 method for class 'htna'
plot_sequences(x, ...)
```

Arguments

net, x	An htna network from build_htna(), or an htna_group.
by	"state" (default) colours cells by code, "group" by actor type.
type	"index" (default), "heatmap" or "distribution".
grouped_legend	With by = "state", split the legend into one block per actor type. Default TRUE.
...	Further arguments of Nestimate::sequence_plot(), such as sort, legend, state_colors or main.

Value

Called for the plot. Invisibly, the description of the drawn plot returned by Nestimate::sequence_plot() (a list), or net for a grouped network.

Conditions

htna_bad_input when net is not an htna network with sequence data; htna_unassigned_node when it has no actor partition; htna_empty_input for a grouped network without cohorts.

See Also

plot_htna(), sequence_compare_htna(), extract_meta_paths(), build_htna()

Examples

```
data(human_ai)
net <- build_htna(subset(human_ai, project == "Project_9"),
                  actor_type = "actor_type")
sequence_plot_htna(net)
sequence_plot_htna(net, type = "heatmap")
sequence_plot_htna(net, by = "group", type = "distribution")
```

`state_distribution_htna` *Tabulate the distribution of the codes within each actor*

Description

`state_distribution_htna()` returns the state distribution of a network: how often each code occurs and its share of the events of its group. For an htna network the group is the actor type, so the table shows how each actor divides its activity over its own codes; for a network without an actor partition all codes form one group, `"all"`.

`state_distribution_htna()` runs Nestimate's `state_distribution()` and returns a data frame with one row per group and code. For an htna network it is the same table as `frequencies_htna()`; the function is provided under its general name for code written for other network types.

Usage

```
state_distribution_htna(x, ...)
```

Arguments

`x`	An htna network from `build_htna()`, or another network.
`...`	Unused for htna networks.

Value

A data frame with one row per group and code and columns `group`, `state`, `count` and `proportion` (proportions sum to one within each group).

See Also

`frequencies_htna()`, `state_frequencies_htna()` for the pooled table.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
state_distribution_htna(net)
```

`state_frequencies_htna` *Count the codes of a network over all actors*

Description

`state_frequencies_htna()` counts the codes of a network over all actors: every code in the sessions of an htna network, or in raw sequence data, in decreasing order of frequency. The pooled table shows which codes dominate the interaction as a whole, whichever actor produces them. Each code's proportion is its share of all events, so the shares of the different actors can be compared directly.

`state_frequencies_htna()` runs Nestimate's `state_frequencies()` and returns a data frame with one row per code.

The table has no actor column; `frequencies_htna()` reports each code under its actor, with shares within the actor.

Usage

```
state_frequencies_htna(data, ...)
```

Arguments

`data`	An htna network from `build_htna()`, or sequence data (a wide data frame or a list of character vectors). A grouped network is not accepted; use `state_distribution_htna()` for the counts of every cohort.
`...`	Unused.

Value

A data frame with one row per code, sorted by decreasing count, and columns `state`, `count` and `proportion` (the share of all events, rounded to four decimals).

See Also

`frequencies_htna()`, `plot_frequencies_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
state_frequencies_htna(net)
```

summary.htna *Summarize a heterogeneous transition network*

Description

summary() summarizes a heterogeneous transition network by its actor partition; print() shows the summary and as.data.frame() returns its tables. A heterogeneous transition network is described first by which codes belong to which actor type, and how the transitions between codes fall within and between the actors. The summary reports the network's size (sessions, nodes and non-zero edges), the nodes of each actor, the number of non-zero edges in every directed actor-pair block (for example Human to AI), and the standard network-level descriptive metrics (density, mean distance, strength, degree centralization and reciprocity). In an undirected network (method = "co_occurrence") every edge is counted once, so the blocks are the unordered actor pairs.

summary() returns the summary as a summary.htna object; printing it shows the report. For a grouped network, the summary of every cohort is returned and printed in turn. The tables of the summary are returned as data frames by as.data.frame().

Usage

```
## S3 method for class 'htna'
summary(object, max_nodes = 12L, ...)

## S3 method for class 'htna_group'
summary(object, max_nodes = 12L, ...)

## S3 method for class 'summary.htna'
print(x, max_nodes = attr(x, "max_nodes") %||% 12L, ...)

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

## S3 method for class 'summary.htna'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  what = c("actors", "actor_pairs", "metrics")
)

## S3 method for class 'summary.htna_group'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
```

```
  what = c("actors", "actor_pairs", "metrics")
)
```

Arguments

object	An htna network from build_htna() or an htna_group.
max_nodes	Maximum number of nodes listed per actor type when the summary is printed (default 12).
...	Unused.
x	A summary.htna or summary.htna_group object.
row.names, optional	Ignored; present for the generic.
what	Table to return: "actors" (default; one row per actor type with its number of nodes and their names), "actor_pairs" (one row per directed actor pair with the number of non-zero edges; one row per unordered pair for an undirected network) or "metrics" (one row per network metric).

Value

summary() returns an object of class summary.htna: a list with actors (data frame: actor, n_nodes, nodes), edges_by_actor (integer matrix of non-zero edge counts, source actors in rows; symmetric for an undirected network), network_metrics (data frame: metric, value), and the scalars n_nodes, n_edges, directed, n_sessions, n_timesteps and method. For an htna_group it returns a summary.htna_group, a named list of summary.htna objects. print() returns its input invisibly. as.data.frame() returns the table chosen by what; for a grouped summary the cohorts are stacked with a leading group column.

See Also

build_htna(), as.data.frame.htna() for the edge and node tables of the network itself.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
summary(net)
as.data.frame(summary(net), what = "actor_pairs")

grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
as.data.frame(summary(grp), what = "metrics")
```

Index

∗ **datasets**
ai_simplified, 4
human_ai, 30
human_ai_codebook, 32
human_simplified, 32

ai_simplified, 4, *30–33*
as.data.frame.htna, 5
as.data.frame.htna(), *3*, *8*, *13*, *15*, *24*, *60*
as.data.frame.htna_comparison *(*htna_comparison-methods*)*, 28
as.data.frame.htna_comparison_group *(*htna_comparison-methods*)*, 28
as.data.frame.htna_group *(*as.data.frame.htna*)*, 5
as.data.frame.htna_stability, 6
as.data.frame.htna_stability(), *19*, *20*, *37*, *38*, *49*
as.data.frame.htna_stability_group *(*as.data.frame.htna_stability*)*, 6
as.data.frame.summary.htna *(*summary.htna*)*, 59
as.data.frame.summary.htna_group *(*summary.htna*)*, 59
as.igraph.htna, 8
as.igraph.htna(), *3*
as.igraph.htna_group *(*as.igraph.htna*)*, 8
association_rules_htna, 9
association_rules_htna(), *4*, *25*, *26*

bayes_compare_htna, 10
bayes_compare_htna(), *3*, *22*, *23*, *35*, *36*, *45*, *46*
bootstrap, 11
bootstrap.htna *(*bootstrap_htna*)*, 12
bootstrap_htna, 12
bootstrap_htna(), *4*, *11*, *15–17*, *19–21*, *44*, *45*, *47*, *48*, *50*, *52*
build_htna, 13
build_htna(), *3*, *4*, *6*, *8*, *9*, *11*, *12*, *16*, *18*, *19*, *21*, *24*, *26*, *27*, *31*, *32*, *34*, *35*, *41*, *43*, *50*, *52*, *56–58*, *60*

casedrop_reliability_htna, 16
casedrop_reliability_htna(), *4*, *19–21*, *33*, *34*, *52*
centralities_htna, 17
centralities_htna(), *8*, *24*, *38*, *39*
centrality_stability_htna, 19
centrality_stability_htna(), *4*, *6*, *7*, *12*, *16–18*, *34*, *37–39*, *49*, *52*
certainty_htna, 21
certainty_htna(), *4*, *12*, *13*, *44*, *45*
cograph::plot_htna(), *42*, *43*
cograph::splot(), *43*, *44*, *46*
compare_htna, 22
compare_htna(), *3*, *10*, *11*, *15*, *28*, *29*, *35–37*, *45*, *46*

deprune_htna, 23
deprune_htna(), *3*, *50*, *51*, *53*

edge_betweenness_htna, 24
edge_betweenness_htna(), *18*, *40*
extract_meta_paths, 25
extract_meta_paths(), *4*, *9*, *10*, *48*, *54–56*

frequencies_htna, 27
frequencies_htna(), *4*, *10*, *35*, *41*, *42*, *57*, *58*

htna *(*htna-package*)*, 3
htna-package, 3
htna_comparison-methods, *22*, *23*, 28
htna_palette, 30, *39*, *43*, *44*, *46*
human_ai, *4*, *5*, 30, *32*, *33*
human_ai_codebook, *5*, *14*, *30*, *31*, 32, *33*
human_simplified, *4*, *5*, *30–32*, 32

igraph::graph_from_adjacency_matrix(), *8*

markov_order_test_htna, 33
markov_order_test_htna(), *4*
mosaic_plot_htna, 34
mosaic_plot_htna(), *3*, *9, 10*, *27*, *41, 42*, *54, 55*

permutation_htna, 35
permutation_htna(), *3*, *10, 11*, *15*, *22, 23*, *45, 46*, *54, 55*
plot.htna *(*plot_htna*)*, 42
plot.htna_bootstrap *(*plot_htna_bootstrap*)*, 44
plot.htna_comparison_group, 36
plot.htna_comparison_group(), *29*
plot.htna_group *(*plot_htna*)*, 42
plot.htna_stability, 37
plot.htna_stability(), *7*, *19, 20*, *49*
plot.htna_stability_group *(*plot.htna_stability*)*, 37
plot_centralities, 38
plot_centralities(), *3*, *18*
plot_edge_betweenness_htna, 40
plot_edge_betweenness_htna(), *3*, *24*
plot_frequencies_htna, 41
plot_frequencies_htna(), *3*, *27*, *34, 35*, *58*
plot_htna, 42
plot_htna(), *3*, *15*, *24*, *30*, *40*, *44–46*, *50*, *55, 56*
plot_htna_bootstrap, 44
plot_htna_bootstrap(), *3*, *12, 13*, *21*, *30*, *42, 43*, *48*
plot_htna_diff, 45
plot_htna_diff(), *3*, *10, 11*, *22, 23*, *30*, *35–37*, *42, 43*
plot_sequences, 46
plot_sequences.htna *(*sequence_plot_htna*)*, 55
print.htna_bootstrap_group, 47
print.htna_comparison *(*htna_comparison-methods*)*, 28
print.htna_comparison_group *(*htna_comparison-methods*)*, 28
print.htna_paths, 48
print.htna_paths(), *26*
print.htna_stability, 49
print.htna_stability(), *37*
print.htna_stability_group *(*print.htna_stability*)*, 49
print.summary.htna *(*summary.htna*)*, 59
print.summary.htna_group *(*summary.htna*)*, 59
prune_htna, 50
prune_htna(), *3*, *12, 13*, *21*, *23*, *51*, *53*
pruning_details_htna, 51
pruning_details_htna(), *3*, *23*, *50*, *53*

reliability_htna, 52
reliability_htna(), *4*, *13*, *16, 17*, *21*, *33, 34*
reprune_htna, 53
reprune_htna(), *3*, *23*, *50, 51*

sequence_compare_htna, 54
sequence_compare_htna(), *4*, *25, 26*, *55, 56*
sequence_plot_htna, 55
sequence_plot_htna(), *3*, *46, 47*
state_distribution_htna, 57
state_distribution_htna(), *4*, *27*, *42*, *58*
state_frequencies_htna, 58
state_frequencies_htna(), *4*, *27*, *42*, *57*
subset(), *48*
summary.htna, 59
summary.htna(), *3*, *6*, *13*, *15*
summary.htna_bootstrap_group *(*print.htna_bootstrap_group*)*, 47
summary.htna_comparison *(*htna_comparison-methods*)*, 28
summary.htna_comparison_group *(*htna_comparison-methods*)*, 28
summary.htna_group *(*summary.htna*)*, 59
summary.htna_stability_group *(*as.data.frame.htna_stability*)*, 6

utils::head(), *48*