

Package ‘SemNeT’

October 14, 2026

Title Methods and Measures for Semantic Network Analysis

Version 2.1.0

Date 2026-10-14

Maintainer Alexander P. Christensen <alexpaulchristensen@gmail.com>

Description Implements several functions for the analysis of semantic networks including different network estimation algorithms, partial node bootstrapping (Kenett, Anaki, & Faust, 2014 <[doi:10.3389/fnhum.2014.00407](https://doi.org/10.3389/fnhum.2014.00407)>), random walk simulation, and a function to compute global network measures. Significance tests, including permutation testing, and plotting features are also implemented.

Depends R (>= 3.6.0)

License GPL (>= 3.0)

Encoding UTF-8

LazyData true

Imports methods, utils, stats, graphics, future, future.apply, progressr, igraph, qgraph, car, broom, effects

Suggests testthat (>= 3.0.0), shiny (>= 1.8.0), shinyjs, shinyalert, DT, promises, spreadr, shinyMatrix, shinyBS, readxl, foreign, R.matlab, tools, ggplot2

URL <https://github.com/AlexChristensen/SemNeT>

BugReports <https://github.com/AlexChristensen/SemNeT/issues>

NeedsCompilation yes

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

Author Alexander P. Christensen [aut, cre] (ORCID: <<https://orcid.org/0000-0002-9798-7037>>),
Yoed N. Kenett [aut, ctb] (ORCID: <<https://orcid.org/0000-0003-3872-7689>>)

Repository CRAN

Date/Publication 2026-10-14 14:12:14 UTC

Contents

SemNeT-package	3
animals.freq	3
aspl	4
binary2responses	5
bootSemNeT	6
bootstrap_SemNeT	7
bootstrap_test_SemNeT	10
cc	13
cn	14
compare_nets	16
compare_networks	17
convert2cytoscape	18
convert2igraph	19
equate	20
finalize	21
find_textcleaner_objects	22
forward_flow	22
net.high	24
net.low	25
NRW	26
one.result	27
open.binary	27
open.clean	28
open.group	29
permutation_SemNeT	29
PF	33
plot.bootSemNeT	34
plot.bootstrap_SemNeT	35
Q	36
randnet.test	37
random_network_test	38
random_walk	39
randwalk	41
read_uploaded_file	42
response.analysis	43
responses2binary	44
response_analysis	45
semantic_network_measures	46
semnetmeas	48
SemNeTShiny	48
sim.fluency	49
similarity	50
simulate_fluency	51
test.bootSemNeT	52
TMFG	53
two.result	54

SemNeT-package	<i>SemNeT-package</i>
----------------	-----------------------

Description

Implements several functions for the analysis of semantic networks including partial node bootstrapping (Kenett, Anaki, & Faust, 2014), random walk simulation (Kenett & Austerweil, 2016), and a function to compute global network measures. Significance tests and plotting features are also implemented.

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com> & Yoed N. Kenett <yokedkenett@gmail.com>

References

Christensen, A. P., Kenett, Y. N., Cotter, K. N., Beaty, R. E., & Silvia, P. J. (2018). Remotely close associations: Openness to experience and semantic memory structure. *European Journal of Personality*, 32, 480-492.

Kenett, Y. N., Anaki, D., & Faust, M. (2014). Investigating the structure of semantic networks in low and high creative persons. *Frontiers in Human Neuroscience*, 8, 407.

Kenett, Y. N., & Austerweil, J. L. (2016). Examining search processes in low and high creative individuals with random walks. In *Paper presented at the proceedings of the 38th annual meeting of the cognitive science society*. Austin, TX.

See Also

- Useful links:
- <https://github.com/AlexChristensen/SemNeT>
 - Report bugs at <https://github.com/AlexChristensen/SemNeT/issues>

animals.freq	<i>Frequency of Animal Responses</i>
--------------	--------------------------------------

Description

Frequency of animal responses from Christensen & Kenett (2019). These frequencies are used to generate data in the [sim.fluency](#) function.

Usage

data(animals.freq)

Format

```
animals.freq (vector, length = 367)
```

Examples

```
data("animals.freq")
```

aspl	<i>Average Shortest Path Length</i>
------	-------------------------------------

Description

Computes the global average shortest path length of a network

Usage

```
aspl(A, weighted = FALSE, type = c("arithmetic", "harmonic"))
```

```
ASPL(A, weighted = FALSE)
```

Arguments

A	Matrix or data frame. An adjacency matrix of network data
weighted	Boolean. Is the network weighted? A weighted edge's length is the inverse of its weight (weights are strengths), so the value depends on the scale of the weights and negative weights are not allowed. Defaults to FALSE. Set to TRUE for weighted measures
type	Character (length = 1). How the shortest-path distances are averaged. Defaults to "arithmetic". Available options: "arithmetic" — The mean of every node's mean distance to the nodes it can reach. Unreachable pairs are left out of the average "harmonic" — The harmonic mean of the distances between all pairs of nodes, the inverse of the global efficiency (Latora & Marchiori, 2001): $1/\text{mean}(1/d_{ij})$ over all pairs of distinct nodes. Unreachable pairs have an infinite distance, whose inverse is 0, so they are counted instead of left out

Details

For a connected network, "arithmetic" is the usual average shortest path length (e.g., `igraph::mean_distance()`), and "harmonic" is smaller than it. They differ most for a disconnected network: "arithmetic" ignores unreachable pairs, which makes a fragmented network look better connected than it is and weights each node by the size of its component, whereas "harmonic" counts them. A network with no connected pair has an Inf harmonic ASPL.

Value

Returns the network's ASPL. The per-node average distance (type-matched; NaN for an isolated node when "arithmetic", Inf when "harmonic") and the eccentricity (the longest finite shortest path) are attached as the "node" and "eccentricity" attributes, rather than being computed and discarded

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Latora, V., & Marchiori, M. (2001). Efficient behavior of small-world networks. *Physical Review Letters*, 87, 198701.

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, 52, 1059-1069.

Examples

```
# Pearson's correlation only for CRAN checks
fluency_data <- matrix(sample(0:1, 1000, replace = TRUE), nrow = 100, ncol = 10)
A <- suppressWarnings(tmfg(similarity(fluency_data, method = "cor")))

# Unweighted
network_aspl <- aspl(A)
```

binary2responses

Convert a Binary Response Matrix to Raw Responses

Description

Converts a binary (participant by vocabulary) response matrix, or the result of [responses2binary](#), back into a raw word-response matrix (participants by response order, one word per cell)

Usage

```
binary2responses(binary_matrix)
```

Arguments

binary_matrix Matrix, data frame, or "responses2binary" object. A participant by vocabulary matrix of 0s (never said) and 1s (said), with words as column names. When a [responses2binary](#) result is provided, the stored response order is used to place each participant's words in the order they were said. A numeric matrix with values above 1 is read the same way (each value is the rank of the response, as in `responses2binary(...)$order`); a strictly 0/1 matrix has no order, so its words are placed in column order

Value

A matrix of words with participants as rows and response order as columns (Response_01, Response_02, ...). Rows are left-aligned and padded with NA

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Raw fluency responses, 2 participants, up to 3 responses each
responses <- matrix(
  c("dog", "cat", NA, "cat", "wolf", "dog"),
  nrow = 2, byrow = TRUE
)

# Round trip: raw -> binary -> raw
binary2responses(responses2binary(responses))
```

bootSemNeT

Bootstrapped Semantic Network Analysis (Legacy)

Description

Backward-compatible wrapper for [bootstrap_SemNeT](#) – `bootSemNeT()` was renamed `bootstrap_SemNeT()` in SemNeT 2.0.0. The returned object gets both the old class ("bootSemNeT", so `plot()` still dispatches to `plot.bootSemNeT`) and the new one ("bootstrap_SemNeT", so it can also be passed straight into `bootstrap_test_SemNeT` or `bootstrap_test_SemNeT`'s own legacy wrapper, `test.bootSemNeT`)

Usage

```
bootSemNeT(
  ...,
  method = c("CN", "NRW", "PF", "TMFG"),
  methodArgs = list(),
  type = c("case", "node"),
  prop = 0.5,
  sim,
  weighted = FALSE,
  iter = 1000,
  cores
)
```

Arguments

...	Matrices or data frames. Response matrices, one per group, e.g. <code>bootSemNeT(low, high)</code> – names are captured from the given expressions when not passed as named arguments, matching the original's own <code>substitute()</code> -based name capture
method	Character. See bootstrap_SemNeT . Unlike the original – which had no working default for this argument at all (<code>switch()</code> on an unresolved multi-value default silently required a real value every call) – omitting it now falls through to <code>bootstrap_SemNeT()</code> 's own default, "tmfg"
methodArgs	List. See bootstrap_SemNeT 's <code>method_args</code> . <code>minCase</code> inside <code>methodArgs</code> is pulled out into <code>minimum_cases</code> , matching where the original actually used it
type	Character. See bootstrap_SemNeT . Same no-working-default situation as <code>method</code> above; omitting it now defaults to "case"
prop	Numeric. See bootstrap_SemNeT
sim	Character. See bootstrap_SemNeT
weighted	Boolean. Unused – was already dead code in the original <code>bootSemNeT()</code>
iter	Numeric. See bootstrap_SemNeT
cores	Numeric. See bootstrap_SemNeT . Unlike the original, defaults to sequential processing rather than auto-detecting cores (see randwalk 's equivalent note)

Value

See [bootstrap_SemNeT](#)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

bootstrap_SemNeT

Bootstrapped Semantic Network Analysis

Description

Estimates each group's semantic network from `iter` resampled datasets (participants resampled with replacement or a proportion of nodes dropped) to build bootstrap distributions of [aspl](#), [cc](#), and [q](#)

Usage

```
bootstrap_SemNeT(
  ...,
  method = c("tmfg", "cn", "nrw", "pf"),
  method_args = list(),
  type = c("case", "node"),
```

```

prop = 0.5,
sim = "cosine",
minimum_cases = 2,
iter = 1000,
cores = 1,
keep_networks = FALSE,
seed = NULL
)

```

Arguments

...	Named matrices or data frames. One response matrix per group (at least one), e.g., <code>bootstrap_SemNeT(low = low_data, high = high_data)</code>
method	Character (length = 1). Network estimation method. Defaults to "tmfg". Available options: • "tmfg" — See tmfg • "cn" — See cn • "nrw" — See nrw • "pf" — See pf
method_args	List. Additional arguments passed to method's function. Defaults to <code>list()</code> (the method's defaults)
type	Character (length = 1). Type of bootstrap. Defaults to "case". Available options: • "case" — Resamples participants with replacement • "node" — Keeps every participant and randomly drops nodes (see <code>prop</code>). Requires every group to share the same nodes. Each iteration drops the <i>same</i> nodes from every group, so a group contrast within an iteration is not affected by which nodes happened to be kept
prop	Numeric (length = 1). Only for type = "node". Proportion of nodes to keep in each resampled network. Defaults to .50
sim	Character (length = 1). Similarity measure for method = "tmfg". Defaults to "cosine". See similarity for other options
minimum_cases	Numeric (length = 1). Only for method = "tmfg" and type = "case". Minimum number of participants who gave a response for it to be kept (finalize) before the groups' vocabularies are equated (equate). It is applied to each resample, in which a participant can appear more than once, so a response only one participant gave can survive in a resample (when that participant is drawn twice or more) even though the full data would drop it. Resampled networks therefore tend to contain somewhat different vocabularies than the observed network. Defaults to 2
iter	Numeric (length = 1). Number of bootstrap samples per group. Defaults to 1000
cores	Numeric (length = 1). Number of processing cores. Defaults to 1 (sequential)
keep_networks	Boolean (length = 1). Should every bootstrapped network be kept? These can be large (iter dense networks per group) and are not needed for testing. Defaults to FALSE

seed Numeric (length = 1). Defaults to NULL or random results (0 is random too). Set a nonzero value for reproducible results. Each bootstrap sample (and its **q** community detection) gets its own seed, so results do not depend on `set.seed()`, `cores`, or the order workers finish

Value

Returns a list containing:

groups A named list with one list per group containing networks (the bootstrapped networks; NULL unless `keep_networks = TRUE`), measures (a 3 x `iter` matrix of `aspl`, `cc`, and `q`), edges and nodes (each bootstrapped network's counts), observed (the measures, edges, and nodes of the network estimated from the full data; NA if it could not be estimated), and summary (each measure's bootstrap mean, standard error, and 95% percentile range of the bootstrap values; this describes the spread of the resampled statistics, which for a case bootstrap is not centred on, and is not a confidence interval for, the observed value – see observed and [bootstrap_test_SemNeT](#))

meta A list containing `prop` (type = "node" only), `iter`, `type`, `method`, and `seed`

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

See Also

[bootstrap_test_SemNeT](#) for testing group differences and [plot](#) for plotting the distributions

Examples

```
# Simulate datasets
one <- simulate_fluency(20)
two <- simulate_fluency(20)

# Bootstrap node-drop (partial) networks
boot_result <- bootstrap_SemNeT(
  one = one, two = two, prop = .50,
  iter = 100, # increase for more precise results
  cores = 2, method = "tmfg", type = "node", seed = 1234
)

# Bootstrap case-resampled networks -- equating (see `equate()`) requires
# named responses, so this uses real word responses instead
low <- open.clean[which(open.group == "Low"), ]
high <- open.clean[which(open.group == "High"), ]
boot_result <- bootstrap_SemNeT(
  low = low, high = high,
  iter = 100, # increase for more precise results
  cores = 2, method = "tmfg", type = "case", minimum_cases = 2
)
```

bootstrap_test_SemNeT *Statistical Tests for* [bootstrap_SemNeT](#)

Description

Compares groups' bootstrapped network measures using Wald tests and confidence intervals of the within-iteration group contrasts. The bootstrap samples are never treated as independent observations

Usage

```
bootstrap_test_SemNeT(
  ...,
  measures = c("aspl", "cc", "q"),
  groups = NULL,
  formula = NULL,
  ci_level = 0.95,
  power = 0.8,
  adjust = c("none", "bonferroni"),
  ci_type = c("normal", "bc", "percentile"),
  test = NULL,
  legacy = FALSE
)
```

Arguments

...	One or more bootstrap_SemNeT objects of the same groups. More than one is useful for type = "node" results at different prop values or for type = "case" results from different seeds. Only type = "case" results support inference about groups of participants (see Details)
measures	Character. One or more of "aspl", "cc", and "q". Defaults to all three
groups	Matrix or data frame. One row per group (matched to the group names by row name if present, otherwise by the order groups were entered), one column per design variable. With one variable, every pair of its levels is compared; with several, the main effect of each variable is compared (the unweighted average of the groups at each level). Defaults to NULL (one variable using the group names)
formula	Character (length = 1). Only for multiple variables in groups. Outcome written as "y", e.g., "y ~ gf * caq", to add two-way interaction contrasts (a difference of differences). Requires a fully crossed design; higher-order interactions are not supported. Defaults to NULL (main effects only)
ci_level	Numeric (length = 1). Confidence level between 0 and 1 (exclusive). Defaults to .95. At least 1000 bootstrap iterations are recommended

power	Numeric (length = 1). Power for the minimal detectable effect (mde) between 0 and 1 (exclusive). Defaults to .80
adjust	Character (length = 1). Multiplicity adjustment across the comparisons within each measure. Nothing is ever adjusted across the measures or across the objects passed in . . . ; with several comparisons and "none", each is an unadjusted test. Defaults to "none". Available options: • "none" — No adjustment • "bonferroni" — Widens each interval to the $1 - (1 - \text{ci_level}) / m$ level and multiplies each p-value by m, the number of comparisons
ci_type	Character (length = 1). Type of test and interval (not used for node results). Defaults to "normal". Available options: • "normal" — Wald test and interval: the observed difference plus and minus the normal quantile times the bootstrap standard error. Assumes the bootstrap distribution is roughly normal and needs fewer iterations than the percentile-based intervals. Requires the groups' observed statistics (NA without them, with a message) • "bc" — Bias-corrected percentile interval (Efron & Tibshirani, 1993). Requires the groups' observed statistics (falls back to "percentile" without them, with a message) • "percentile" — Percentile interval (e.g., the 2.5th and 97.5th percentiles of the within-iteration differences at $\text{ci_level} = .95$)
test	Character (length = 1). Only for legacy = TRUE: "t-test", "ANOVA", or "ANCOVA". Defaults to NULL ("t-test" when legacy = TRUE). Supplying it with legacy = FALSE is an error
legacy	Boolean (length = 1). Should the parametric tests on bootstrap replicates from versions before 2.1.0 be reproduced? These tests are statistically invalid: they treat bootstrap samples as independent observations and divide the bootstrap standard deviation (already a standard error) by $\text{sqrt}(\text{iter})$, inflating test statistics by about $\text{sqrt}(\text{iter})$. A warning is always issued. Defaults to FALSE

Details

Always run [permutation_SemNeT](#) alongside this function and report both: the bootstrap shows how precisely each group's networks and their difference are estimated, whereas the permutation test asks whether group membership matters.

Groups are bootstrapped independently, so the difference between two groups' measures within a bootstrap iteration is a draw from the bootstrap distribution of the group difference.

With $\text{ci_type} = \text{"normal"}$ (default), the test statistic is the Wald statistic $z = \hat{\Delta} / \text{SD}^*$, where $\hat{\Delta}$ is the observed contrast and SD^* is the standard deviation of the within-iteration contrasts (the bootstrap standard error). The p -value is $2\{1 - \Phi(|z|)\}$ and matches the interval $\hat{\Delta} \pm z_{1-\alpha/2} \text{SD}^*$. The null hypothesis is that the contrast *at the study's sample sizes* is zero, not the large-sample contrast, because network measures depend on sample size. The test relies on a normal approximation and was conservative in permuted-data checks. When group labels can be permuted, [permutation_SemNeT](#) is the exact test and should be preferred.

For $\text{ci_type} = \text{"bc"}$ and "percentile", a difference is reliable if the interval excludes 0 (Tu et al., under review). No p -value is reported (p_value is NA): the smallest level at which such an

interval excludes 0 is not a null-distribution p -value and was easily mistaken for one. Use `ci_type = "normal"` for a Wald p -value, or `permutation_SemNeT` for a test. Case resampling changes each resampled network's size, so the bootstrap distribution is not centred on the full-data statistic (reported as bias).

The minimal detectable effect (mde) is $(z_{1-\alpha/2} + z_{\text{power}}) \text{SD}^*$: the smallest true contrast a two-sided Wald test would detect with probability power (α is divided by the number of comparisons for `adjust = "bonferroni"`). It depends only on the bootstrap standard error, so a non-significant difference smaller than mde is inconclusive. The bootstrap standard error can exceed the true sampling standard deviation, so mde errs large. It also assumes the contrast is roughly normally distributed, for every `ci_type`. When that is doubtful (check the differences of each measure), `permutation_SemNeT` reports an mde whose critical value comes from the permutation null instead of a normal quantile.

Node results (`type = "node"`) subsample words, not participants, and the subsample sizes change the measures themselves, so they are summarized descriptively instead of tested.

Value

Returns a list containing:

<code>test</code>	"bootstrap CI" or the legacy test
<code>measures</code>	The measures that were tested
<code>groups</code>	The groups design matrix that was used
<code>results</code>	A named list with one list per object in ... ("Case" or "Proportion (<prop>)", each with one list per measure containing table and differences (an iter x comparisons matrix of the within-iteration contrasts). The table's numbers are not rounded (the printed tables format them), so <code>z</code>, <code>p_value</code> and the interval agree exactly with <code>observed_diff</code> and <code>se</code>; for <code>legacy = TRUE</code>, <code>table</code>, <code>post_hoc</code>, and <code>adjusted_means</code>. <code>table</code> has one row per comparison: <code>term</code>, <code>contrast</code>, <code>mean_1</code>, <code>se_1</code>, <code>mean_2</code>, and <code>se_2</code> (the bootstrap mean and standard error of each side; NA for interactions), <code>mean_diff</code>, <code>observed_diff</code>, <code>bias</code> (<code>mean_diff - observed_diff</code>), <code>se</code> (of the difference), <code>bias_se</code> (<code> bias / se</code>), <code>z</code> (Wald statistic; <code>ci_type = "normal"</code> only), <code>mde</code>, <code>ci_lower</code>, <code>ci_upper</code>, <code>p_value</code> (Wald test; <code>ci_type = "normal"</code> only, otherwise NA), <code>n_boot</code>, and <code>direction</code> ("n.s." when the interval includes 0). For node results, <code>table</code> has <code>term</code>, <code>contrast</code>, <code>observed_diff</code>, <code>mean_diff</code>, <code>sd_diff</code>, <code>q_lower</code> and <code>q_upper</code> (central <code>ci_level</code> range of the subsample differences), <code>prop_same_sign</code> (share with the observed difference's sign), and <code>n_boot</code>: a robustness summary with no p-value or interval
<code>meta</code>	A named list like <code>results</code> containing <code>type</code> , <code>method</code> , <code>iter</code> , <code>seed</code> (NA if none), <code>prop</code> (node results only), and <code>networks</code> (each group's mean nodes and edges across bootstrapped networks and its <code>observed_nodes</code> and <code>observed_edges</code>)
<code>ci_level</code>	The confidence level used
<code>power</code>	The power used for mde
<code>adjust</code>	The multiplicity adjustment used
<code>ci_type</code>	The test and interval type used
<code>legacy</code>	Whether the legacy (invalid) tests were run

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References**Bootstrap**

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. Chapman & Hall.

Inflated bootstrap statistics

Tu, C.-Y., Chodorow, M., Brooks, P. J., Ezrina, E., & Valian, V. (under review). Inflated statistics lead to incorrect inference: How (not) to make cognitive network group comparisons.

See Also

[permutation_SemNeT](#) for a permutation test and [bootstrap_SemNeT](#) for generating the bootstrap samples

Examples

```
# Get openness data
low <- open.clean[which(open.group == "Low"), ]
high <- open.clean[which(open.group == "High"), ]

# Bootstrap networks
boot_result <- bootstrap_SemNeT(
  low = low, high = high, iter = 1000, cores = 2,
  type = "case", method = "tmfg"
)

# Wald tests of the group differences
bootstrap_test_SemNeT(boot_result)

# Percentile intervals
bootstrap_test_SemNeT(boot_result, ci_type = "percentile")
```

cc

Clustering Coefficient

Description

Computes the global clustering coefficient (CC) of a network

Usage

```
cc(A, weighted = FALSE, na.rm = FALSE)
```

```
CC(A, weighted = FALSE)
```

Arguments

<code>A</code>	Matrix or data frame. An adjacency matrix of network data
<code>weighted</code>	Boolean. Is the network weighted? The weighted coefficient (Rubinov & Sporns, 2010) is defined for weights scaled by the network's largest absolute weight, so the weights are divided by it first: CC does not depend on the scale of the weights, and for a signed network a node's coefficient can be negative. Defaults to FALSE. Set to TRUE for weighted measures of CC
<code>na.rm</code>	Boolean. A node with fewer than two neighbors has no possible triangle, so its clustering coefficient is undefined. Should those nodes be left out of the average (TRUE), or counted as having a clustering coefficient of 0 (FALSE)? Counting them as 0 lowers CC in sparse networks (e.g., <code>cn</code> , <code>nw</code>) in proportion to how many such nodes there are. Defaults to FALSE

Value

Returns the network's CC. The per-node clustering coefficient is attached as the "node" attribute (NA for a node with fewer than two neighbors when `na.rm = TRUE`), rather than being computed and discarded

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, 52, 1059-1069.

Examples

```
# Pearson's correlation only for CRAN checks
fluency_data <- matrix(sample(0:1, 1000, replace = TRUE), nrow = 100, ncol = 10)
A <- suppressWarnings(tmfg(similarity(fluency_data, method = "cor")))

# Unweighted
network_cc <- cc(A)
```

Description

Estimates a semantic network using the Community Network method described in Goni et al. (2011)

Usage

```
cn(data, window = 2, alpha = 0.05, enrich = FALSE, cutoff = 0.5)
```

```
CN(data, window = 2, alpha = 0.05, enrich = FALSE)
```

Arguments

data	Matrix or data frame. A preprocessed verbal fluency matrix where rows are participants and columns are the order of their verbal fluency responses. Content may be the raw word responses or an ordered numeric response matrix (see <code>responses2binary()</code>)
window	Numeric. Size of window to look for co-occurrences in. Defaults to 2
alpha	Numeric. Significance level of the test of each pair of responses' co-occurrence against chance, following Goni et al. (2011): the pair's co-occurrence rate gets a two-sided $1 - \alpha$ Clopper-Pearson confidence interval, and an edge is kept when the rate expected by chance falls below the interval's lower bound (so each tail holds $\alpha / 2$). Must be strictly between 0 and 1. The tests are not adjusted for the number of pairs tested. Defaults to .05
enrich	Boolean. Should the network be enriched by connecting all nodes within their respective modules? Defaults to FALSE
cutoff	Numeric. Height at which the hierarchical clustering of the generalized topological overlap dissimilarity is cut into modules when <code>enrich = TRUE</code> . Goni et al. (2011) chose their cutoff (0.58 in their data) so that it separated the blocks of the overlap matrix as well as possible, so there is no established value : it depends on the data and should be inspected rather than assumed. A warning is given whenever <code>enrich = TRUE</code> . Defaults to 0.50

Value

Returns an undirected semantic network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Goni, J., Arrondo, G., Sepulcre, J., Martincorena, I., de Mendizabal, N. V., Corominas-Murtra, B., ... & Villoslada, P. (2011). The semantic organization of the animal category: Evidence from semantic verbal fluency and network theory. *Cognitive Processing*, 12, 183-196.

Examples

```
# Get data
data <- open.clean

# Organize group data
group <- open.group
```

```

low <- data[which(group == "Low"), ]
high <- data[which(group == "High"), ]

## Not run:
# Compute networks
low_net <- cn(low)
high_net <- cn(high)

## End(Not run)

```

compare_nets

Compare Graphs (Legacy)

Description

Backward-compatible wrapper for [compare_networks](#) – `compare_nets()` was renamed `compare_networks()` in SemNeT 2.0.0

Usage

```

compare_nets(
  ...,
  title,
  config,
  placement = c("match", "default"),
  weighted = FALSE,
  qgraph.args = list()
)

```

Arguments

<code>...</code>	Matrices or data frames. Networks to plot, e.g. <code>compare_nets(low, high)</code> – names are captured from the given expressions when not passed as named arguments, matching the original’s own <code>substitute()</code> -based name capture
<code>title</code>	List. See compare_networks
<code>config</code>	Character. See compare_networks
<code>placement</code>	Character. See compare_networks
<code>weighted</code>	Boolean. See compare_networks
<code>qgraph.args</code>	List. See compare_networks ’s <code>qgraph_args</code>

Value

See [compare_networks](#)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Description

Uses [qgraph](#) to plot networks side by side. Accepts any number of networks and arranges the plots using $\text{floor}(\sqrt{\text{length}(\dots)})$ rows (e.g. 3 networks: 1 row of 3; 6 networks: 2 rows of 3; 9 networks: 3 rows of 3)

Usage

```
compare_networks(
  ...,
  title,
  config = "spring",
  placement = c("default", "match"),
  weighted = FALSE,
  qgraph_args = list()
)
```

Arguments

<code>...</code>	Named matrices or data frames. At least two network adjacency matrices, e.g. <code>compare_networks(low = low_net, high = high_net)</code>
<code>title</code>	List. Titles for each plot, in the same order as <code>...</code> . Defaults to the names given in <code>...</code>
<code>config</code>	Character. Defaults to "spring". See qgraph for more options
<code>placement</code>	Character. How should nodes be placed when comparing networks? "default" — places nodes in the default <code>config</code> positions "match" — places nodes in the same position across every network (requires every network to have the same nodes) Defaults to "default"
<code>weighted</code>	Boolean. Should networks be plotted with weights? Defaults to FALSE. Unweighted networks are often more aesthetically representative of the network's structure
<code>qgraph_args</code>	List. Additional arguments passed on to qgraph . Defaults to <code>list()</code>

Value

Plots networks side by side using [qgraph](#); returns NULL invisibly

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Epskamp, S., Cramer, A. O. J., Waldorp, L. J., Schmittmann, V. D., & Borsboom, D. (2012). qgraph: Network visualizations of relationships in psychometric data. *Journal of Statistical Software*, 48, 1-18.

Examples

```
# Simulate datasets
one <- simulate_fluency(10)
two <- simulate_fluency(10)

# Compute similarity matrices
cos1 <- similarity(one, method = "cosine")
cos2 <- similarity(two, method = "cosine")

# Compute networks
net1 <- suppressWarnings(tmfg(cos1))
net2 <- suppressWarnings(tmfg(cos2))

# Compare networks
compare_networks(one = net1, two = net2, config = "spring")

# Change edge colors
compare_networks(
  one = net1, two = net2, config = "spring",
  qgraph_args = list(edge.color = "blue")
)
```

convert2cytoscape

Convert an Adjacency Matrix to Cytoscape Format

Description

Converts an adjacency matrix to Cytoscape's sparse edge-list format

Usage

```
convert2cytoscape(A)
```

Arguments

A Matrix or data frame. A cleaned, finalized network matrix ready to be visualized

Value

A sparse matrix formatted for Cytoscape, with columns nodeTO, nodeFROM, and nodeWEIGHT

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., ... & Ideker, T. (2003). Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13, 2498-2504.

Examples

```
# Simulate toy binarized datasets
one <- matrix(sample(0:1, 100, replace = TRUE), nrow = 10, ncol = 10)
two <- matrix(sample(0:1, 100, replace = TRUE), nrow = 10, ncol = 10)

# Compute similarity matrices
cos1 <- similarity(one, method = "cosine")
cos2 <- similarity(two, method = "cosine")

# Compute networks
net1 <- suppressWarnings(tmfg(cos1))
net2 <- suppressWarnings(tmfg(cos2))

# Convert to Cytoscape format
cyto1 <- convert2cytoscape(net1)
cyto2 <- convert2cytoscape(net2)

# Write to .csv
write.csv(cyto1, file.path(tempdir(), "cyto1.csv"), row.names = FALSE)
write.csv(cyto2, file.path(tempdir(), "cyto2.csv"), row.names = FALSE)
```

convert2igraph

Convert a Network to igraph's Format

Description

Converts an adjacency matrix into igraph's network format

Usage

```
convert2igraph(A)
```

Arguments

A Matrix or data frame. An adjacency (network) matrix

Value

Returns a network in igraph's format

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Pearson's correlation only for CRAN checks
fluency_data <- matrix(sample(0:1, 500, replace = TRUE), nrow = 50, ncol = 10)
A <- suppressWarnings(tmfg(similarity(fluency_data, method = "cor")))

igraph_network <- convert2igraph(A)
```

equate

Equate Groups

Description

Equates multiple binary response matrices to one another, so that every group retains exactly the responses common to *all* of them

Usage

```
equate(...)
```

Arguments

... Named binary response matrices or data frames. At least two, e.g. `equate(low = low_data, high = high_data)`

Value

A named list of the equated binary response matrices, one per input, each containing the same responses (columns) in the same order

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Obtain binary data
bin <- open.binary

# Finalize each group
mat1 <- finalize(bin[c(1:5), ])
mat2 <- finalize(bin[c(6:10), ])

# Equate the groups
eq <- equate(mat1 = mat1, mat2 = mat2)
```

```
# Obtain respective equated response matrices
eq_mat1 <- eq$mat1
eq_mat2 <- eq$mat2
```

finalize*Finalize a Response Matrix*

Description

Finalizes a binary response matrix by keeping only responses given by at least a minimum number of participants

Usage

```
finalize(data, minimum_cases = 2)
```

Arguments

<code>data</code>	Matrix or data frame. A binary response matrix (participants x responses)
<code>minimum_cases</code>	Numeric. Minimum number of participants required to have given a response for that response to be kept. Defaults to 2

Value

A binary response matrix containing only responses given by at least `minimum_cases` participants

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Obtain binary data
bin <- open.binary

# Finalize
final <- finalize(bin)
```

```
find_textcleaner_objects
```

Find Usable textcleaner Objects in the Global Environment

Description

Scans `.GlobalEnv` for objects inheriting class `"textcleaner"` and `"fluency"` (from `SemNet-Cleaner`'s own `textcleaner()/textcleaner_shiny()` – a sibling package) – used by the Shiny app's Load Data tab to offer loading one directly, as a second data source alongside its own file-upload widget.

Restricted to `type = "fluency"` results: a `type = "free"` result's `$data$clean` is a long-format response/cue/ID table, a different shape than the participant x response-slot word matrix this package's own `responses2binary()` (and every bundled example dataset) expects – not something to silently reshape without a real design decision, so it's excluded here rather than offered and failing downstream.

Usage

```
find_textcleaner_objects()
```

Value

A character vector of object names in `.GlobalEnv` that are usable `textcleaner` results (possibly empty)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
find_textcleaner_objects()
```

```
forward_flow
```

Forward Flow

Description

Computes the "forward flow" of a sequence of verbal fluency or free association responses (Gray et al., 2019): for each response after the first, the mean semantic distance ($1 - \text{cosine similarity}$) between that response and every response that preceded it, using word vectors from a user-supplied embedding matrix.

Unlike the pre-2.0.0 version of this function, no semantic space is downloaded automatically – CRAN policy prohibits a package silently authenticating with and downloading files from an external service at runtime. Bring your own pre-trained (e.g. GloVe, word2vec) or custom word embeddings instead

Usage

```
forward_flow(
  response_matrix,
  embeddings,
  min_response = 3,
  max_response = NULL,
  type = c("fluency", "free"),
  cores = 1
)
```

Arguments

<code>response_matrix</code>	Matrix or data frame. For type = "fluency": participants (rows) by responses (columns), one word per cell, in the order each participant gave them. Row names, if present, are used as participant IDs. For type = "free": long-format data with columns named "ID", "Cue", and "Response", one response per row, in the order each participant gave them for each cue
<code>embeddings</code>	Matrix or data frame. A word-embedding matrix: one row per vocabulary term (row names are the terms themselves), one column per embedding dimension. Not bundled or downloaded by this package – supply your own pre-trained or custom vectors. A response with no matching row is treated as missing (excluded from the distances it would have contributed to, rather than corrupting every flow value it touches)
<code>min_response</code>	Numeric. Minimum number of responses a participant (type = "fluency") or participant-cue pair (type = "free") must have to compute forward flow for it (at least 2: the first response has no earlier response to be measured against). Blank-like responses (" ", " ", "NA") are treated as missing, and responses are counted before any not found in embeddings are dropped. A response a participant repeats has a distance of 0 to its earlier occurrence, and is kept. Defaults to 3
<code>max_response</code>	Numeric. Maximum number of responses to use, keeping only the first n. Useful for decoupling forward flow from fluency (participants who simply produced more responses). Defaults to NULL, which uses every response
<code>type</code>	Character. "fluency" or "free". Defaults to "fluency"
<code>cores</code>	Numeric. Number of computer processing cores to use. Defaults to 1

Value

For type = "fluency", a list with:

<code>mean_flow</code>	A data frame of each participant's average forward flow
<code>response_flow</code>	A list, one element per participant, of that participant's response-by-response forward flow

For type = "free", a list with:

<code>mean_flow</code>	A data frame of each participant's average forward flow, across all of their cues
------------------------	---

mean_response_flow A data frame of each participant-cue pair's average forward flow

response_flow A list, one element per participant (itself a list with one element per cue), of that cue's response-by-response forward flow

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com> & Brendan Baker <bsb5477@psu.edu>

References

Gray, K., Anderson, S., Chen, E. E., Kelly, J. M., Christian, M. S., Patrick, J., ... & Lewis, K. (2019). "Forward flow": A new measure to quantify free thought and predict creativity. *American Psychologist*, 74(5), 539-554.

Beatty, R. E., Zeitlen, D. C., Baker, B. S., & Kenett, Y. N. (2021). Forward flow and creative thought: Assessing associative cognition and its role in divergent thinking. *Thinking Skills and Creativity*, 100859.

Examples

```
# A tiny toy embedding space and toy fluency data -- illustrates the
# interface only; use real pre-trained vectors (GloVe, word2vec, etc.) in
# practice
set.seed(1)
words <- c("dog", "cat", "wolf", "car", "truck", "bus")
embeddings <- matrix(rnorm(length(words) * 5), nrow = length(words))
row.names(embeddings) <- words

responses <- matrix(
  c("dog", "wolf", "cat", "car", "truck", "bus"),
  nrow = 2, byrow = TRUE, dimnames = list(c("P1", "P2"), NULL)
)

result <- forward_flow(responses, embeddings, min_response = 2)
result$mean_flow
```

net.high

High Openness to Experience Network

Description

High openness to experience network from Christensen & Kenett (2019)

Usage

```
data(net.high)
```

Format

net.high (matrix, 160 x 160)

References

Christensen, A. P., & Kenett, Y. N. (2019) Semantic network analysis (SemNA): A tutorial on preprocessing, estimating, and analyzing semantic networks. *PsyArXiv*.

Examples

```
data("net.high")
```

net.low

Low Openness to Experience Network

Description

Low openness to experience network from Christensen & Kenett (2019)

Usage

```
data(net.low)
```

Format

net.low (matrix, 160 x 160)

References

Christensen, A. P., & Kenett, Y. N. (2019) Semantic network analysis (SemNA): A tutorial on preprocessing, estimating, and analyzing semantic networks. *PsyArXiv*.

Examples

```
data("net.low")
```

NRW

*Naive Random Walk Network Estimation***Description**

Estimates a semantic network using the Naive Random Walk method described in Lerner, Ogrocki, and Thomas (2009)

Usage

```
NRW(data, type = c("num", "prop"), threshold = 0)
```

```
nrw(data, type = c("num", "prop"), threshold = 0)
```

Arguments

data	Matrix or data frame. A preprocessed verbal fluency matrix where rows are participants and columns are the order of their verbal fluency responses. Content may be the raw word responses or an ordered numeric response matrix (see <code>responses2binary()</code>)
type	Character. Type of threshold to apply. One of: • "num" — Minimum number of participants in whose responses the two responses were adjacent • "prop" — Minimum proportion of participants in whose responses the two responses were adjacent Defaults to "num"
threshold	Numeric. Value of the minimum number or proportion of co-occurrences. Defaults to 0 for both "num" and "prop". With a threshold above 0, responses left with no link are dropped from the network; at 0 every response is kept (including one that was never adjacent to a different response)

Value

Returns an undirected semantic network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Lerner, A. J., Ogrocki, P. K., & Thomas, P. J. (2009). Network graph analysis of category fluency testing. *Cognitive and Behavioral Neurology*, 22, 45-52.

Examples

```
# Get data
data <- open.clean

# Organize group data
group <- open.group
low <- data[which(group == "Low"), ]
high <- data[which(group == "High"), ]

# Compute networks
low_net <- nrw(low)
high_net <- nrw(high)
```

one.result	<i>Simulated Result for Dataset One</i>
------------	---

Description

A result of [bootSemNeT](#) from a simulated dataset

Usage

```
data(one.result)
```

Format

```
one.result (list, length = 4)
```

Examples

```
data("one.result")
```

open.binary	<i>Binary response Matrices (Openness and Verbal Fluency)</i>
-------------	---

Description

Binary response matrices for the Animals verbal fluency data ($n = 516$) from Christensen et al. (2018).

Usage

```
data(open.binary)
```

Format

open.binary (matrix, 516 x 367)

References

Christensen, A. P., Kenett, Y. N., Cotter, K. N., Beaty, R. E., & Silvia, P. J. (2018). Remotely close associations: Openness to experience and semantic memory structure. *European Journal of Personality*, 32, 480-492.

Examples

```
data("open.binary")
```

open.clean

Cleaned response Matrices (Openness and Verbal Fluency)

Description

Cleaned response matrices for the Animals verbal fluency data ($n = 516$) from Christensen et al. (2018).

Usage

```
data(open.clean)
```

Format

open.clean (matrix, 516 x 35)

References

Christensen, A. P., Kenett, Y. N., Cotter, K. N., Beaty, R. E., & Silvia, P. J. (2018). Remotely close associations: Openness to experience and semantic memory structure. *European Journal of Personality*, 32, 480-492.

Examples

```
data("open.clean")
```

open.group

Groups for Openness and Verbal Fluency

Description

Groups for the Animals verbal fluency data ($n = 516$) from Christensen et al. (2018; see also [open.clean](#)).

Usage

```
data(open.group)
```

Format

```
open.group (vector, length = 516)
```

References

Christensen, A. P., Kenett, Y. N., Cotter, K. N., Beaty, R. E., & Silvia, P. J. (2018). Remotely close associations: Openness to experience and semantic memory structure. *European Journal of Personality*, 32, 480-492.

Examples

```
data("open.group")
```

permutation_SemNeT

Permutation Test for Network Measures

Description

Tests whether global network measures differ between two samples by comparing the observed difference with a null distribution from randomly re-dealing the pooled participants into two samples

Usage

```
permutation_SemNeT(
  sample_1 = NULL,
  sample_2 = NULL,
  method = c("tmfg", "cn", "nrw", "pf"),
  method_args = list(),
  sim = "cosine",
  minimum_cases = 2,
  measure = c("aspl", "cc", "q"),
  alternative = c("two_tailed", "two_tailed_centered", "two_tailed_doubled", "less",
    "greater"),
```

```

    iter = 1000,
    cores = 1,
    previous = NULL,
    keep_networks = TRUE,
    seed = NULL,
    alpha = 0.05,
    power = 0.8
  )

```

Arguments

sample_1	Matrix or data frame. Response matrix to be compared with sample_2. The two samples are pooled by rows, so their columns are lined up first: raw responses of different widths are padded with missing responses, and numeric matrices are matched by column (response) name
sample_2	Matrix or data frame. Response matrix to be compared with sample_1
method	Character (length = 1). Network estimation method. Defaults to "tmfg". Available options: • "tmfg" — See tmfg • "cn" — See cn • "nrw" — See nrw • "pf" — See pf
method_args	List. Additional arguments passed to method's function. Defaults to list() (the method's defaults)
sim	Character (length = 1). Similarity measure for method = "tmfg". Defaults to "cosine". See similarity for other options
minimum_cases	Numeric (length = 1). Only for method = "tmfg". Minimum number of participants who gave a response for it to be kept (finalize) before the samples' vocabularies are equated (equate). Defaults to 2
measure	Character. One or more of "aspl", "cc", and "q" (e.g., c("aspl", "cc", "q") for all three). Every measure is computed from the same permuted networks, so testing several costs little more than one. Defaults to "aspl"
alternative	Character (length = 1). Alternative hypothesis. Defaults to "two_tailed". Available options: • "two_tailed" — sample_1 differs from sample_2; the p-value is the proportion of permuted differences T^* with $T^* \geq T_{obs}$ (distance from 0) • "two_tailed_centered" — same hypothesis, but the distance is measured from the permutation null's mean $\bar{T}^*$: $T^* - \bar{T}^* \geq T_{obs} - \bar{T}^*$ • "two_tailed_doubled" — same hypothesis, with the p-value twice the smaller one-sided p-value (at most 1) • "less" — sample_1 is less than sample_2 • "greater" — sample_1 is greater than sample_2
iter	Numeric (length = 1). Number of permutations (the first is the observed data). Defaults to 1000

cores	Numeric (length = 1). Number of processing cores. Defaults to 1 (sequential)
previous	permutation_SemNeT object. Reuses the previous call's permuted networks to test other measures without re-estimating them (requires <code>keep_networks = TRUE</code> in that call). Defaults to NULL
keep_networks	Boolean (length = 1). Should every permuted network be kept? Only needed to test other measures later with <code>previous</code> and can be large. Defaults to TRUE
seed	Numeric (length = 1). Defaults to NULL or random results (previous's seed when it is used); 0 is random too. Set a nonzero value for reproducible results. Each permutation (and its <code>q</code> community detection) gets its own seed, so results do not depend on <code>set.seed()</code> or <code>cores</code>
alpha	Numeric (length = 1). Significance level for the critical value of the minimal detectable effect (mde) between 0 and 1 (exclusive). Does not change the <i>p</i> -value. Defaults to .05
power	Numeric (length = 1). Power for the minimal detectable effect (mde) between 0 and 1 (exclusive). Defaults to .80

Details

Always run [bootstrap_test_SemNeT](#) (on [bootstrap_SemNeT](#) results) alongside this function and report both: the permutation test asks whether group membership matters, whereas the bootstrap shows how precisely each group's networks and their difference are estimated.

The null hypothesis is that the participants' group labels are exchangeable: both groups' participants come from the same population of responses. A rejection therefore says the groups differ in their responses (their vocabularies, how many responses they give, and the size of the resulting networks as well as their structure); it does not by itself show a difference in semantic structure. `diagnostics` shows how the networks' sizes vary across permutations.

The permutation null answers a different question than a bootstrap interval, so their spreads need not agree: a permutation pools both groups' participants and re-deals them, so the permuted networks' node and edge counts (which ASPL and CC depend on) vary, whereas a case bootstrap ([bootstrap_SemNeT](#)) resamples within each group. Use `diagnostics` to see how the network sizes vary.

The three two-sided definitions agree when the null is centred on 0 and symmetric, which is the case when the two groups have (about) the same size. With unequal group sizes the permutation null is generally not centred on 0, because the measures (e.g., ASPL, CC) depend on a network's size. All three are valid tests of the same null (the participants' group labels are exchangeable) because the observed difference is one of the permutations. "two_tailed" also counts the size-induced offset as evidence; "two_tailed_centered" measures how far the observed difference is from what the group sizes alone produce, assuming the null is roughly symmetric about its mean; "two_tailed_doubled" makes no symmetry assumption. The one-sided tests are unaffected by centring.

When several measures are tested, each is tested against its own null: the *p*-values and mde are *not* adjusted for the number of measures.

The minimal detectable effect (mde) is the smallest true difference the test would detect with probability power, at the study's sample sizes (it does not transfer to other sample sizes and, apart from the observed difference being one of the `iter` permuted values, does not depend on the observed difference). The permutation null gives the critical value *c* without assuming normality:

the $1 - \alpha$ quantile of $|T^*|$ ("two_tailed"), of $|T^* - \bar{T}^*|$ ("two_tailed_centered"; the mde is then a difference from the null's mean), the larger of the $1 - \alpha/2$ quantile of T^* and the negated $\alpha/2$ quantile ("two_tailed_doubled"), T^* ("greater"), or $-T^*$ ("less") over the permuted differences. The distribution under an alternative is not identified by permuting labels, so it is approximated as normal with the null's standard deviation SD_0 : $MDE \approx c + z_{\text{power}} SD_0$. Use at least 1000 permutations for a stable critical value.

Value

Returns a list containing:

result	A data frame with one row per measure containing each sample's observed value, the permutation p -value (p_value), and the alternative tested (alternative)
networks	A list of network_1 and network_2, the networks of every permuted sample (the first entry is the observed data); NULL unless keep_networks = TRUE
method	The method used
seed	The seed used (NULL if none)
differences	The measure difference for every permutation: a vector for one measure, or a permutations x measures matrix for several
diagnostics	A data frame with a row per permutation containing each sample's measure value (value_1 and value_2, or <measure>_1 and <measure>_2 for several measures) and each network's nodes_1, nodes_2, edges_1, and edges_2
mde	A data frame with one row per measure containing alpha, power, critical_value, null_sd, and mde

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

See Also

[bootstrap_test_SemNeT](#) for bootstrap tests of group differences

Examples

```
# Get openness data -- equating (see `equate()`) requires named responses
low <- open.clean[which(open.group == "Low"), ]
high <- open.clean[which(open.group == "High"), ]

# Perform permutation test
perm_test <- permutation_SemNeT(
  low, high, method = "tmfg",
  iter = 100, # increase for more precise results
  cores = 2, keep_networks = TRUE, seed = 1234
)
```

PF *Pathfinder Network*

Description

Estimates a semantic network from verbal fluency data by applying the MST-Pathfinder Network algorithm (Quirin et al., 2008; see also Schvaneveldt, 1990) to the response-order distance of Paulsen et al. (1996)

Usage

PF(data)

pf(data)

Arguments

data Matrix or data frame. A preprocessed verbal fluency matrix where rows are participants and columns are the order of their verbal fluency responses. Content may be the raw word responses or an ordered numeric response matrix (see `responses2binary()`). A strictly binary (0/1) matrix is not accepted because it has discarded the response order the distance needs

Details

The distance between two responses i and j follows Paulsen et al. (1996):

$$D_{ij} = \frac{N}{T_{ij}^2} \sum_{k=1}^{T_{ij}} \frac{d_{ijk}}{n_k}$$

where d_{ijk} is the number of positions between the two responses in participant k 's list, n_k is that participant's number of responses (the largest position in their row), T_{ij} is the number of participants who gave both responses, and N is the number of participants. Responses that are close together in many participants' lists, relative to how many responses they gave, are close; a pair no participant gave together has an infinite distance and can only be linked through other responses. The positions are those of `responses2binary`'s `$order` (a response a participant repeats is positioned at its last occurrence), and responses nobody gave are not nodes.

The network keeps a link between two responses only if no indirect path between them has every one of its links shorter than the direct distance (Pathfinder parameters $r = \infty$ and $q = n - 1$), computed with the MST-Pathfinder algorithm: the union of all minimum spanning trees of the distance matrix. Pairs at exactly the same distance are all kept when none is bridged by shorter links. When some responses are never given together with the others the network is not connected (responses with no finite distance to any other are isolated nodes).

Value

An adjacency matrix. The distances (D_{ij}) of the retained links are attached as the "weighted" attribute ($\emptyset$ where there is no link)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Paulsen, J. S., Romero, R., Chan, A., Davis, A. V., Heaton, R. K., & Jeste, D. V. (1996). Impairment of the semantic network in schizophrenia. *Psychiatry Research*, 63, 109-121.

Quirin, A., Cordon, O., Guerrero-Bote, V. P., Vargas-Quesada, B., & Moya-Aneon, F. (2008) A quick MST-based algorithm to obtain Pathfinder networks (Inf, n-1). *Journal of the American Society for Information Science and Technology*, 59, 1912-1924.

Schvaneveldt, R. W. (1990). *Pathfinder associative networks: Studies in knowledge organization*. Norwood, NJ: Ablex Publishing.

Examples

```
# Obtain data
data <- open.clean[1:100, ]

# Estimate network
pf_net <- pf(data)
```

plot.bootSemNeT	Plot for bootSemNeT (Legacy)
-----------------	--

Description

Backward-compatible wrapper for [plot.bootstrap_SemNeT](#) – dispatches for objects returned by the legacy [bootSemNeT](#), which carry the old class alongside the new one

Usage

```
## S3 method for class 'bootSemNeT'
plot(..., groups = NULL, measures = c("ASPL", "CC", "Q"))
```

Arguments

...	An object from bootSemNeT (only the first element is used, matching the original)
groups	Character. Labels for groups in the order they were entered into bootSemNeT . Defaults to NULL (keep the names already on the object)
measures	Character. One or more of "ASPL", "CC", "Q". Defaults to all three

Value

See [plot.bootstrap_SemNeT](#)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

plot.bootstrap_SemNeT *Plot for bootstrap_SemNeT*

Description

Plots each measure's bootstrap distribution as a raincloud plot (Allen et al., 2018): a half violin, boxplot, and the jittered bootstrap values, with one panel per measure and the groups side by side

Usage

```
## S3 method for class 'bootstrap_SemNeT'
plot(x, measures = c("aspl", "cc", "q"), ...)
```

Arguments

x	A bootstrap_SemNeT object
measures	Character. One or more of "aspl", "cc", and "q". Defaults to all three
...	Not used

Value

Returns a ggplot object (printed at the console and compatible with shiny::renderPlot())

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References**Raincloud plots**

Allen, M., Poggiali, D., Whitaker, K., Marshall, T. R., & Kievit, R. A. (2018). Raincloud plots: a multi-platform tool for robust data visualization. *PeerJ Preprints*, 6, e27137v1. doi:[10.7287/peerj.preprints.27137v1](https://doi.org/10.7287/peerj.preprints.27137v1)

Examples

```
# Get openness data
low <- open.clean[which(open.group == "Low"), ]
high <- open.clean[which(open.group == "High"), ]

# Bootstrap networks
boot_result <- bootstrap_SemNeT(
  low = low, high = high, iter = 100, cores = 2,
  type = "case", method = "tmfg")
```

```
)

# Plot
plot(boot_result)
```

Q

*Modularity***Description**

Computes a global modularity measure (Q) using the Louvain community detection algorithm

Usage

```
Q(A)

q(A, resolution = 1, seed = NULL, signed = FALSE)
```

Arguments

A	Matrix or data frame. An adjacency matrix of network data. Must be symmetric
resolution	Numeric. Adjusts modularity to prefer smaller ($\text{resolution} > 1$) or larger ($0 < \text{resolution} < 1$) communities. Defaults to 1 (standard modularity)
seed	Numeric. A seed for the Louvain algorithm's internal random number generator, for reproducible community detection. Defaults to NULL (not reproducible); 0 also gives random results. Note this is independent of R's own <code>set.seed()</code> – see Details
signed	Boolean. Whether modularity should be computed on the signed network (Gomez et al., 2009), discounting negative edges instead of folding them in as positive. Defaults to FALSE (the original, <code>abs()</code> -based behavior)

Details

Community detection runs via a compiled Louvain implementation (Blondel et al., 2008) whose internal randomness depends only on `seed`, not on R's global random number generator – unlike `igraph::cluster_louvain()`, calling `set.seed()` beforehand has no effect on `q()`'s reproducibility; pass `seed` directly instead. A given nonzero `seed` always gives the same partition; `seed = 0` (like NULL) seeds from the clock.

With `signed = TRUE`, the partition (found on the absolute-value network) is scored as $Q = \frac{W^+}{W^+ + W^-} Q^+ - \frac{W^-}{W^+ + W^-} Q^-$ (Gomez et al., 2009), where W^+ and W^- are the total positive and absolute negative weights and Q^+ and Q^- are the modularity of the positive and of the absolute negative edges.

Value

Returns Q , a measure of how well the network partitions into distinct, densely-connected communities. A network without edges has no defined modularity (NaN)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008, P10008.

Gomez, S., Jensen, P., & Arenas, A. (2009). Analysis of community structure in networks of correlated data. *Physical Review E*, 80, 016114.

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, 52, 1059-1069.

Examples

```
# Pearson's correlation only for CRAN checks
fluency_data <- matrix(sample(0:1, 1000, replace = TRUE), nrow = 100, ncol = 10)
A <- suppressWarnings(tmfg(similarity(fluency_data, method = "cor")))

modularity <- q(A, seed = 1)
```

randnet.test	<i>Random Network Test (Legacy)</i>
--------------	-------------------------------------

Description

Backward-compatible wrapper for [random_network_test](#) – `randnet.test()` was renamed `random_network_test()` in SemNeT 2.0.0. **Output changed:** the p -values are now Monte Carlo p -values, computed for Erdos-Renyi and degree-preserving random networks (`p_value`, `p_value_degree`, ...), and a network's edge count is the number of nonzero cells (weights are no longer summed); see [random_network_test](#)

Usage

```
randnet.test(..., iter, cores)
```

Arguments

...	Matrices or data frames. Networks to test, e.g. <code>randnet.test(low, high)</code> – names are captured from the given expressions when not passed as named arguments, matching the original's own <code>substitute()</code> -based name capture
iter	Numeric. See random_network_test
cores	Numeric. See random_network_test . Unlike the original, defaults to sequential processing rather than auto-detecting cores (see randwalk 's equivalent note)

Value

See [random_network_test](#)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

random_network_test	<i>Test Against Random Networks</i>
---------------------	-------------------------------------

Description

Tests global measures of semantic networks against the distribution of the same measures in random networks of equivalent size and density (Erdos-Renyi) and in random networks that also keep every node's degree

Usage

```
random_network_test(..., iter = 1000, cores = 1, seed = NULL)
```

Arguments

...	Named matrices or data frames. Semantic networks to be compared against random networks, e.g. <code>random_network_test(low = low_net, high = high_net)</code>
iter	Numeric. Number of random networks to generate per input network. Defaults to 1000
cores	Numeric. Number of computer processing cores to use. Defaults to 1
seed	Numeric. A seed for reproducible results (including the community detection behind q), the same whatever cores is. Each random network gets its own seed derived from it. Defaults to NULL (not reproducible; 0 is random too)

Details

Every network is binarized (any nonzero edge becomes 1) before its measures are computed, and its edge count is the number of nonzero off-diagonal entries (a weighted network's weights are not summed).

Two null models are used, because they answer different questions. An Erdos-Renyi network has the observed node and edge counts but otherwise random edges, so any structure (including the degree distribution that nearly every semantic network has) distinguishes the observed network from it. A degree-preserving network is the observed network after many random edge swaps ([rewire](#)), so it keeps every node's degree and asks whether the measure differs beyond what the degrees imply.

p -values are Monte Carlo p -values, with no normality assumption: $(1 + \#\{|x_i - \bar{x}| \geq |x_{obs} - \bar{x}|\}) / (iter + 1)$, where x_i are the `iter` random-network values and $\bar{x}$ their mean. They cannot be smaller than $1 / (iter + 1)$ (and are reported to 4 significant digits, so that they are not rounded to 0). Neither null model guarantees a connected network, and `aspl` ignores unreachable pairs, so a sparse random network can differ from a connected observed one for that reason alone. The tests are not adjusted for the number of measures.

Value

Returns a named list (one entry per input network) of data frames with, for each of aspl, cc, and q, the Monte Carlo p -value and the mean and standard deviation of the random-network distribution: p_value, random_mean, and random_sd for Erdos-Renyi networks, and p_value_degree, random_mean_degree, and random_sd_degree for degree-preserving networks

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Erdos, P., & Renyi, A. (1960). On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5, 17-61.

Maslov, S., & Sneppen, K. (2002). Specificity and stability in topology of protein networks. *Science*, 296, 910-913.

Examples

```
# Get openness data
one <- open.clean[which(open.group == "Low"), ]
two <- open.clean[which(open.group == "High"), ]

# Compute networks
net_one <- cn(one)
net_two <- cn(two)

# Perform random networks test
random_network_test(
  low = net_one, high = net_two, cores = 2,
  iter = 100 # increase for more precise results
)
```

random_walk

Random Walk Simulation

Description

Simulates random walks over two networks to examine the characteristics of spontaneous spreading activation (see Kenett & Austerweil, 2016)

Usage

```
random_walk(
  A,
  B,
  name_a = "A",
  name_b = "B",
  reps = 20,
  steps = 10,
  iter = 10000,
  cores = 1
)
```

Arguments

A	Matrix or data frame. Adjacency matrix of a semantic network
B	Matrix or data frame. A comparison adjacency matrix of a semantic network
name_a	Character. Label for network A in the output. Defaults to "A"
name_b	Character. Label for network B in the output. Defaults to "B"
reps	Numeric. Sets the largest number of steps tested, $\text{reps} \times 10$: walks are run for steps, steps + 10, steps + 20, ... up to $\text{reps} \times 10$ steps (so steps cannot exceed $\text{reps} \times 10$). Defaults to 20 (up to 200 steps)
steps	Numeric. Number of random steps to begin with. Defaults to 10
iter	Numeric. Number of iterations for each random walk. Defaults to 10000
cores	Numeric. Number of computer processing cores to use. Defaults to 1

Details

The comparison is descriptive. Each network's walks are `iter` simulated draws, and `iter` is chosen by the user, so the p -value of a test on those draws would only reflect how many walks were simulated: any difference, however small, becomes "significant" with enough of them. No p -values are reported. Each statistic (number of unique nodes visited, similarity of the walk's end to its start, and the same at 5 unique nodes) is instead summarized by

- the difference in means (A minus B) with a Monte Carlo 95% interval, which describes the error from simulating a finite number of walks (and shrinks as `iter` grows), not uncertainty about the networks
- Cohen's d , which does not depend on `iter`

Neither accounts for the uncertainty in the estimated networks; use [bootstrap_SemNet](#) for that.

Every walk starts at a node drawn at random from the nodes with at least one connection in each network (an isolated node cannot be walked from), and the two networks' walks share each step's random draw. When the pooled standard deviation of a statistic is 0, d is 0 for no difference in means and Inf (signed) for a difference, since any difference is then infinitely large in standard deviation units. The 5-unique-node statistic is measured at each network's own 5th distinct node (or its last, if the walk visited fewer).

Value

A list with long (every simulated summary statistic, all numeric: each network's mean and standard deviation, and the difference, its Monte Carlo interval limits, and Cohen's d for each statistic) and short (a compact, correctly-typed summary with the difference, Cohen's d , and a direction that is the network with the larger mean when $|d| \geq .2$ (Cohen's small effect) and "negligible" otherwise)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com> and Yoed Kenett <yoadkenett@gmail.com>

References

Kenett, Y. N., & Austerweil, J. L. (2016). Examining search processes in low and high creative individuals with random walks. In *Paper presented at the proceedings of the 38th annual meeting of the cognitive science society*. Austin, TX.

Examples

```
# Simulate toy binarized datasets
one <- matrix(sample(0:1, 100, replace = TRUE), nrow = 10, ncol = 10)
two <- matrix(sample(0:1, 100, replace = TRUE), nrow = 10, ncol = 10)

# Compute similarity matrices
cos1 <- similarity(one, method = "cosine")
cos2 <- similarity(two, method = "cosine")

# Compute networks
net1 <- suppressWarnings(tmfg(cos1))
net2 <- suppressWarnings(tmfg(cos2))

# Run random walk analysis
rw_results <- random_walk(net1, net2, iter = 100, cores = 2)
```

Description

Backward-compatible wrapper for [random_walk](#) – `randwalk()` was renamed `random_walk()` in SemNeT 2.0.0. **Output changed:** the Wilcoxon p -values are gone (they only reflected how many walks were simulated) and `$short` is a data frame of differences, Cohen's d , and directions; see [random_walk](#)

Usage

```
randwalk(A, B, reps = 20, steps = 10, iter = 10000, cores)
```

Arguments

A, B	Matrix or data frame. Two networks to compare – names are captured from the given expressions, matching the original’s own <code>substitute()</code> -based name capture
reps	Numeric. See random_walk
steps	Numeric. See random_walk
iter	Numeric. See random_walk
cores	Numeric. See random_walk . Unlike the original, defaults to sequential processing rather than auto-detecting cores – matches <code>random_walk()</code> ’s own default and this rebuild’s dependency trim (no more direct <code>parallel::detectCores()</code> calls outside <code>parallel_process()</code>)

Value

See [random_walk](#)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

read_uploaded_file	<i>Read a Response or Group File in Any Supported Format</i>
--------------------	--

Description

Multi-format file reader dispatching on a file’s extension: `.rds`, `.RData`, `.csv`, `.txt`, `.xls/.xlsx` (via `readxl`), `.sav` (via `foreign`), `.mat` (via `R.matlab`), or no/unrecognized extension (read as plain delimited text). Used by the Shiny app’s upload widgets, and usable directly for the same purpose outside Shiny

Usage

```
read_uploaded_file(datapath, filename, header = TRUE, sep = ",")
```

Arguments

datapath	Character. Path to the file’s actual content on disk (e.g. Shiny’s <code>input\$file\$datapath</code>)
filename	Character. The file’s original name, used only to determine its format from its extension – kept separate from <code>datapath</code> since Shiny’s own upload temp files don’t necessarily carry a matching extension
header	Boolean. Does the file have a header row (for delimited-text and spreadsheet formats)? Defaults to <code>TRUE</code>
sep	Character. Field delimiter (for delimited-text formats). Defaults to <code>" , "</code>

Value

The file's contents as a data frame (or the single object it contains, for .rds/.RData)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
path <- tempfile(fileext = ".csv")
write.csv(data.frame(x = 1:3, y = c("a", "b", "c")), path, row.names = FALSE)

read_uploaded_file(path, "responses.csv")
```

response.analysis	<i>Response Pattern Analysis (Legacy)</i>
-------------------	---

Description

Backward-compatible wrapper for [response_analysis](#) – response.analysis() was renamed response_analysis() in SemNeT 2.0.0

Usage

```
response.analysis(...)
```

Arguments

...	Matrices or data frames. Response matrices, e.g. response.analysis(low, high) – names are captured from the given expressions when not passed as named arguments, matching the original's own substitute()-based name capture
-----	---

Value

See [response_analysis](#)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

responses2binary

*Convert Raw Responses to a Binary Response Matrix***Description**

Converts a raw word-response matrix (participants by response order, one word per cell) into a binary (participant by vocabulary) response matrix, alongside a matching matrix recording response order

Usage

```
responses2binary(responses)
```

Arguments

responses	Matrix or data frame. A raw word-response matrix: participants (rows) by response order (columns), one word per cell (e.g. verbal fluency data). Empty, missing, and placeholder cells (NA, "", whitespace-only) are treated as no response
-----------	---

Value

A list of class "responses2binary" with:

binary	A participant by vocabulary matrix of 0s (never said) and 1s (said)
order	A participant by vocabulary matrix of 0s (never said) and the response's 1-based order (first said, second said, ...) otherwise

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Raw fluency responses, 2 participants, up to 3 responses each
responses <- matrix(
  c("dog", "cat", NA, "cat", "wolf", "dog"),
  nrow = 2, byrow = TRUE
)

result <- responses2binary(responses)
result$binary
result$order
```

response_analysis	<i>Response Analysis</i>
-------------------	--------------------------

Description

Computes the difference in the total and unique number of responses between groups (follows Christensen et al., 2018). With more than two groups, every pairwise comparison is returned, since both underlying tests (a two-sample t -test and McNemar's test) are inherently pairwise

Usage

```
response_analysis(...)
```

Arguments

... Named matrices or data frames. At least two response matrices (raw word responses or already-binary), one per group, e.g. `response_analysis(low = low_data, high = high_data)`

Details

The number of responses unique to each group (unique) grows with a group's number of participants, and the responses are not independent observations, so the McNemar test on those counts should be read as a description, not as a test of the groups. The vocabulary statistics compare groups at the same number of participants instead.

McNemar's test only uses the two off-diagonal cells (the numbers of responses unique to each group), so here it is a continuity-corrected sign test on those two counts; it is NA when neither group has a unique response. `phi` is the correlation between the two groups' indicators of giving each response across the responses given by either group. Because no response is given by neither group, it is at most 0 by construction (and reported as an absolute value), so it describes the overlap of the vocabularies and not an association between the groups. The t -test is NA when it is undefined (e.g., every participant gave the same number of responses).

Value

With exactly two groups, a list with:

total	A named vector comparing the number of distinct responses given by each participant (a response repeated by a participant counts once) via a two-sample t -test and Cohen's d
unique	A named vector comparing the number of responses unique to each group via McNemar's test (and <code>phi</code> , see Details)
vocabulary	Descriptive statistics for the groups' vocabularies that account for the groups' sizes, unlike <code>unique</code> : <code>jaccard</code> (shared responses over all responses given by either group; it still grows with group size, because larger groups reveal more

rare responses); `n_participants_<group>`; `rarefied_n` (the sample size compared, the smaller group's); `rarefied_<group>` (the expected number of distinct responses in a random sample of `rarefied_n` participants from the group, Hurlbert, 1971; the smaller group's is its observed number); and `rarefied_diff` (the first group's minus the second's)

With more than two groups, a named list of such lists, one per pair of groups (named "<group1>_vs_<group2>")

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Christensen, A. P., Kenett, Y. N., Cotter, K. N., Beaty, R. E., & Silvia, P. J. (2018). Remotely close associations: Openness to experience and semantic memory structure. *European Journal of Personality*, 32, 480-492.

Hurlbert, S. H. (1971). The nonconcept of species diversity: A critique and alternative parameters. *Ecology*, 52, 577-586.

Examples

```
# Obtain data
low <- open.clean[which(open.group == "Low"), ]
high <- open.clean[which(open.group == "High"), ]

# Perform analysis
response_analysis(low = low, high = high)
```

semantic_network_measures

Semantic Network Measures

Description

Computes the average shortest path length ([aspl](#)), clustering coefficient ([cc](#)), and modularity ([q](#)) of a network

Usage

```
semantic_network_measures(
  A,
  measures = c("aspl", "cc", "q"),
  weighted = FALSE,
  seed = NULL,
  na.rm = FALSE,
  signed = FALSE,
  type = c("arithmetic", "harmonic")
)
```

Arguments

<code>A</code>	Matrix or data frame. An adjacency matrix of a network
<code>measures</code>	Character. Which global network measures to compute: any of "aspl", "cc", and "q". Defaults to all three
<code>weighted</code>	Boolean. Should weighted measures be computed? Defaults to FALSE. Set to TRUE for weighted measures
<code>seed</code>	Numeric. Passed on to <code>q</code> for reproducible community detection. Defaults to NULL (not reproducible; 0 is random too) – see <code>q</code> 's Details
<code>na.rm</code>	Boolean. Passed on to <code>cc</code> : should nodes with fewer than two neighbors (whose clustering coefficient is undefined) be left out of the average (TRUE) instead of counted as 0 (FALSE)? Only affects "cc". Defaults to FALSE
<code>signed</code>	Boolean. Passed on to <code>q</code> : should modularity be computed on the signed network (Gomez et al., 2009), discounting negative edges? Only affects "q", and only matters for a network with negative weights, so it needs <code>weighted = TRUE</code> (an unweighted network is binarized first, which removes the sign). Defaults to FALSE
<code>type</code>	Character. Passed on to <code>aspl</code> : "arithmetic" (the mean of each node's mean distance to the nodes it can reach; unreachable pairs are left out) or "harmonic" (the inverse of the mean inverse distance between all pairs, which counts unreachable pairs). Only affects "aspl". Defaults to "arithmetic"

Value

Returns a named numeric vector with one value per requested measure

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Gomez, S., Jensen, P., & Arenas, A. (2009). Analysis of community structure in networks of correlated data. *Physical Review E*, 80, 016114.

Examples

```
# Simulate a toy binarized dataset
one <- matrix(sample(0:1, 100, replace = TRUE), nrow = 10, ncol = 10)

# Compute a similarity matrix
cos <- similarity(one, method = "cosine")

# Compute a network
net <- suppressWarnings(tmfg(cos))

# Compute global network measures
global_measures <- semantic_network_measures(net)
```

semnetmeas

*Semantic Network Measures (Legacy)***Description**

Backward-compatible wrapper for [semantic_network_measures](#) – `semnetmeas()` was renamed `semantic_network_measures()` in SemNeT 2.0.0

Usage

```
semnetmeas(A, meas = c("ASPL", "CC", "Q"), weighted = FALSE)
```

Arguments

A	Matrix or data frame. See semantic_network_measures
meas	Character. One or more of "ASPL", "CC", "Q". Defaults to all three
weighted	Boolean. See semantic_network_measures

Value

A named numeric vector, named to match meas (uppercase), as in the original

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

SemNeTShiny

*Shiny App for [SemNeT](#)***Description**

Launches an interactive Shiny application for semantic network analysis, covering network estimation, comparison, bootstrap and permutation testing, random walk simulation, and spreading activation. Results are retrieved via the app's own download buttons (as `.rds/` `.csv`/image files), not returned to the R session that launched it – each browser session keeps its own results independently, so the app can be run locally or shared with multiple simultaneous users

Usage

```
SemNeTShiny()
```

Value

Invisibly returns NULL. Launches the app as a side effect; see the app's own "Export" tab for how to retrieve results

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
if(interactive()){  
  SemNeTShiny()  
}
```

sim.fluency

Simulate Verbal Fluency Data (Legacy)

Description

Backward-compatible wrapper for [simulate_fluency](#) – `sim.fluency()` was renamed `simulate_fluency()` in SemNeT 2.0.0

Usage

```
sim.fluency(nodes, cases, random = FALSE)
```

Arguments

nodes	Numeric. See simulate_fluency . Unlike <code>simulate_fluency()</code> , has no default – matches the original, which always required this argument
cases	Numeric. See simulate_fluency . Unlike <code>simulate_fluency()</code> , has no default – matches the original, which always required this argument
random	Boolean. See simulate_fluency

Value

See [simulate_fluency](#)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

similarity	<i>Measures of Similarity</i>
------------	-------------------------------

Description

Computes a similarity matrix between the columns of a binarized verbal fluency or linguistic dataset (see Choi, Cha, & Tappert, 2010 for additional measures)

Usage

```
similarity(  
  data,  
  method = c("cosine", "angular", "cor", "euclid", "faith", "jaccard", "phi", "rr")  
)
```

Arguments

data	Matrix or data frame. A binarized dataset of verbal fluency or linguistic data
method	Character. Type of similarity measure to compute. One of "cosine" (default), "angular", "cor", "euclid", "faith", "jaccard", "phi", or "rr". Defaults to "cosine". "euclid" is the Euclidean <i>distance</i> between columns converted to a similarity, $1/(1 + d)$, so that (like every other option) larger values mean more similar and the matrix can be used to estimate a network (e.g., tmfg)

Details

"angular", "faith", "jaccard" and "rr" are computed from the 2 x 2 table of two columns' agreements and disagreements across participants, which needs 0/1 data (other values are an error); missing or infinite values are an error for every measure. A response nobody gave (an all-zero column) makes "cosine", "angular" and "jaccard" undefined (NaN), and a response everybody gave (or nobody) makes "cor"/"phi" undefined; remove such responses first (see [finalize](#)). A participant who gave no response still counts as a case in which every response is absent, which matters for the measures that use joint absences ("cor", "phi", "faith", "rr"). "euclid" mostly reflects how often each response was given: two rare responses that never co-occur look similar.

Value

A symmetric similarity matrix. The diagonal is set to 1

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Choi, S. S., Cha, S. H., & Tappert, C. C. (2010). A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8, 43-48.

Examples

```
# Simulate a toy binarized dataset
one <- matrix(sample(0:1, 100, replace = TRUE), nrow = 10, ncol = 10)

# Compute a cosine similarity matrix
cos <- similarity(one, method = "cosine")
```

simulate_fluency	<i>Simulate a Verbal Fluency Binary Response Matrix</i>
------------------	---

Description

Simulates verbal fluency data based on the number of nodes desired in the resulting network. Each response's total endorsement count is simulated from a Poisson distribution (see [rpois](#)), using frequencies from the [animals.freq](#) data; participants' individual responses are then simulated with a probability of endorsing each response equal to that response's simulated total (plus 1) over 1.05 times the largest such total. The probability therefore does not depend on the number of participants, and every response is simulated independently of the others: the data contain no association between responses, so a network estimated from them (which still has its full complement of edges, e.g. [tmfg](#)) describes noise. Use them to try out functions, not as a model of semantic structure

Usage

```
simulate_fluency(nodes = 100, cases = 500, random = FALSE)
```

Arguments

nodes	Numeric. Number of nodes (unique responses) to simulate. Defaults to 100
cases	Numeric. Number of participants to simulate. Defaults to 500
random	Boolean. Should response frequencies be randomly sampled (with replacement) from animals.freq , rather than taken in their existing order? Defaults to FALSE

Value

A binary matrix with cases rows (participants) and nodes columns (responses)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Simulate data for 50 nodes and 200 participants
simulate_fluency(nodes = 50, cases = 200)
```

test.bootSemNeT	Statistical Tests for bootSemNeT (Legacy)
-----------------	---

Description

Backward-compatible wrapper for [bootstrap_test_SemNeT](#) – `test.bootSemNeT()` was renamed `bootstrap_test_SemNeT()` in SemNeT 2.0.0.

Legacy warning: before SemNeT 2.1.0, this function ran *t*-tests/ANOVAs/ANCOVAs on the bootstrap replicates themselves, which is statistically invalid (the replicates are not independent observations, and the bootstrap standard deviation – already a standard error – was divided by `sqrt(iter)`, inflating test statistics by roughly `sqrt(iter)`). It now always warns and returns the results of `bootstrap_test_SemNeT()`: by default a Wald test of each contrast using the bootstrap standard error (with normal intervals), or percentile intervals, which have no *p*-value. Results from earlier versions should be recomputed. Only accepts objects produced by this package's own [bootSemNeT](#) / [bootstrap_SemNeT](#), not a raw object saved from SemNeT < 2.0.0

Usage

```
test.bootSemNeT(
  ...,
  test = c("ANCOVA", "ANOVA", "t-test"),
  measures = c("ASPL", "CC", "Q"),
  formula = NULL,
  groups = NULL
)
```

Arguments

<code>...</code>	One or more bootSemNeT / bootstrap_SemNeT objects. See bootstrap_test_SemNeT
<code>test</code>	Character. "ANCOVA", "ANOVA", or "t-test" (the default). Kept only for backward compatibility: the parametric test is not run (it is invalid). The comparisons returned are determined by groups/formula (see bootstrap_test_SemNeT)
<code>measures</code>	Character. One or more of "ASPL", "CC", "Q". Defaults to all three
<code>formula</code>	Character. See bootstrap_test_SemNeT
<code>groups</code>	Matrix or data frame. See bootstrap_test_SemNeT

Value

See [bootstrap_test_SemNeT](#)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Description

Applies the Triangulated Maximally Filtered Graph (TMFG) filtering method (**please see and cite Massara et al., 2016**). The TMFG method uses a structural constraint that limits the number of zero-order correlations included in the network ($3n - 6$, where n is the number of variables). Construction begins by forming a tetrahedron from the four nodes with the largest strength (the sum of their absolute associations that exceed the matrix's average absolute association, diagonal included), then iteratively adds each remaining node to the connected set of three nodes (a triangle) it is most strongly associated with, until every node is included. Edges are chosen on absolute association (a strong negative association is a strong edge); the signed values are kept in the returned network.

Usage

```
TMFG(data, depend = FALSE)
```

```
tmfg(data, depend = FALSE)
```

Arguments

data	Matrix or data frame. A square association matrix (e.g. a correlation or similarity matrix)
depend	Boolean. Is data an asymmetric dependency matrix? Without it, data must be symmetric. Defaults to FALSE. With TRUE, each edge is placed using the larger of its two asymmetric values, but the returned network's links are still symmetric (both cells of a retained edge are filled, each with its own value from data), so it is not a directed network

Details

The TMFG method applies a structural constraint on the network, which restrains the network to retain a certain number of edges ($3n - 6$). The network is composed of 3- and 4-node cliques (a triangle and a tetrahedron, respectively). Notably, TMFG can use any association measure and does not assume the data is multivariate normal.

Value

Returns a TMFG-filtered adjacency matrix

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Christensen, A. P., Kenett, Y. N., Aste, T., Silvia, P. J., & Kwapil, T. R. (2018). Network structure of the Wisconsin Schizotypy Scales-Short Forms: Examining psychometric network filtering approaches. *Behavior Research Methods*, 50, 2531-2550.

Massara, G. P., Di Matteo, T., & Aste, T. (2016). Network filtering for big data: Triangulated maximally filtered graph. *Journal of Complex Networks*, 5, 161-178.

Examples

```
# Pearson's correlation only for CRAN checks
fluency_data <- matrix(sample(0:1, 1000, replace = TRUE), nrow = 100, ncol = 10)
A <- suppressWarnings(tmfg(similarity(fluency_data, method = "cor")))
```

two.result	<i>Simulated Result for Dataset One and Two</i>
------------	---

Description

A result of [bootSemNeT](#) from two simulated datasets

Usage

```
data(two.result)
```

Format

```
two.result (list, length = 6)
```

Examples

```
data("two.result")
```

vignette.plots	<i>Plots for Vignette</i>
----------------	---------------------------

Description

Plots for vignette taken from Christensen & Kenett (2019)

Usage

```
data(vignette.plots)
```

Format

`vignette.plots (list, length = 3)`

References

Christensen, A. P., & Kenett, Y. N. (2019) Semantic network analysis (SemNA): A tutorial on preprocessing, estimating, and analyzing semantic networks. *PsyArXiv*.

Examples

`data("vignette.plots")`

Index

* datasets

- animals.freq, [3](#)
- net.high, [24](#)
- net.low, [25](#)
- one.result, [27](#)
- open.binary, [27](#)
- open.clean, [28](#)
- open.group, [29](#)
- two.result, [54](#)
- vignette.plots, [54](#)

animals.freq, [3](#), [51](#)

ASPL (aspl), [4](#)

aspl, [4](#), [7](#), [46](#), [47](#)

binary2responses, [5](#)

bootSemNeT, [6](#), [27](#), [34](#), [52](#), [54](#)

bootstrap_SemNeT, [6](#), [7](#), [7](#), [10](#), [13](#), [31](#), [35](#), [40](#), [52](#)

bootstrap_test_SemNeT, [6](#), [9](#), [10](#), [31](#), [32](#), [52](#)

CC (cc), [13](#)

cc, [7](#), [13](#), [46](#), [47](#)

CN (cn), [14](#)

cn, [8](#), [14](#), [14](#), [30](#)

compare_nets, [16](#)

compare_networks, [16](#), [17](#)

convert2cytoscape, [18](#)

convert2igraph, [19](#)

equate, [8](#), [20](#), [30](#)

finalize, [8](#), [21](#), [30](#), [50](#)

find_textcleaner_objects, [22](#)

forward_flow, [22](#)

net.high, [24](#)

net.low, [25](#)

NRW, [26](#)

nrw, [8](#), [14](#), [30](#)

nrw (NRW), [26](#)

one.result, [27](#)

open.binary, [27](#)

open.clean, [28](#), [29](#)

open.group, [29](#)

permutation_SemNeT, [11–13](#), [29](#), [31](#)

PF, [33](#)

pf, [8](#), [30](#)

pf (PF), [33](#)

plot, [9](#)

plot.bootSemNeT, [6](#), [34](#)

plot.bootstrap_SemNeT, [34](#), [35](#)

Q, [36](#)

q, [7](#), [9](#), [31](#), [46](#), [47](#)

q (Q), [36](#)

qgraph, [17](#)

randnet.test, [37](#)

random_network_test, [37](#), [38](#), [38](#)

random_walk, [39](#), [41](#), [42](#)

randwalk, [7](#), [37](#), [41](#)

read_uploaded_file, [42](#)

response.analysis, [43](#)

response_analysis, [43](#), [45](#)

responses2binary, [5](#), [33](#), [44](#)

rewire, [38](#)

rpois, [51](#)

semantic_network_measures, [46](#), [48](#)

SemNeT, [48](#)

SemNeT (SemNeT-package), [3](#)

SemNeT-package, [3](#)

semnetmeas, [48](#)

SemNeTShiny, [48](#)

sim.fluency, [3](#), [49](#)

similarity, [8](#), [30](#), [50](#)

simulate_fluency, [49](#), [51](#)

test.bootSemNeT, [6](#), [52](#)

TMFG, [53](#)

tmfg, [8](#), [30](#), [50](#), [51](#)

tmfg (TMFG), [53](#)

two.result, [54](#)

vignette.plots, [54](#)