Package ‘SemNeT’

September 4, 2021

Title  Methods and Measures for Semantic Network Analysis
Version  1.4.3
Date  2021-08-27
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>), random walk simulation (Kenett & Austerweil, 2016 <http://alab.psych.wisc.edu/papers/files/Kenett16CreativityRW.pdf>), and a function to compute global network measures. Significance tests and plotting features are also implemented.
Depends  R (>= 3.6.0)
License  GPL (>= 3.0)
Encoding  UTF-8
LazyData  true
Imports  pbapply, dplyr, plyr, RColorBrewer, purrr, magrittr, ggplot2, grid, igraph, qgraph, scales, car, broom, effects, methods, philentropy
URL  https://github.com/AlexChristensen/SemNeT
BugReports  https://github.com/AlexChristensen/SemNeT/issues
NeedsCompilation  no
Suggests  shiny, shinyjs, shinyalert, shinyMatrix, shinyBS, spreadr, animation, R.matlab, foreign, readxl, data.table
RoxygenNote  7.1.1
Author  Alexander P. Christensen [aut, cre]
         (https://orcid.org/0000-0002-9798-7037),
         Yoed N. Kenett [aut, ctb] (https://orcid.org/0000-0003-3872-7689)
Repository  CRAN
Date/Publication  2021-09-04 04:40:02 UTC
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 <yoedkenett@gmail.com>

References


See Also

Useful links:

- [https://github.com/AlexChristensen/SemNeT](https://github.com/AlexChristensen/SemNeT)
- Report bugs at [https://github.com/AlexChristensen/SemNeT/issues](https://github.com/AlexChristensen/SemNeT/issues)

---

*animals.freq*  

**Frequency of Animal Responses**

**Description**

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

**Usage**

```r
data(animals.freq)
```

**Format**

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

**Examples**

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

*Average Shortest Path Length*

Description

Computes the global average shortest path length of the network

Usage

\[ \text{ASPL}(A, \text{weighted} = \text{FALSE}) \]

Arguments

- \( A \): An adjacency matrix of network data
- \( \text{weighted} \): Is the network weighted? Defaults to \text{FALSE}. Set to \text{TRUE} for weighted measures

Value

Returns the ASPL of the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References


Examples

```r
# Pearson's correlation only for CRAN checks
A <- TMFG(similarity(sim.fluency(100), method = "cor"))

# Unweighted
aspl <- ASPL(A)
```
bootSemNet  

Description

Bootstrap techniques to generate semantic networks and compute global network characteristics

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. Cleaned response matrices (e.g., `responses$clean` from `textcleaner`) or binary response matrices (e.g., binary output from `textcleaner`)
- `method` Character. Network estimation method to use. Current options include:
  - `TMFG` Triangulated Maximally Filtered Graph
  - `CN` Community Network
  - `NRW` Naive Random Walk
  - `PF` Pathfinder
- `methodArgs` List. A list of additional arguments for the network estimation function. See links in argument `method` for additional arguments (see also Examples)
- `type` Character. Type of bootstrap to perform
  - `node` Generates partial networks based on dropping a certain proportion of nodes (see argument `prop`)
  - `case` Samples with replacement the same number of participants as in the original dataset
- `prop` Numeric. Only for type = "node". Proportion of nodes to remain in the network. Defaults to .50
- `sim` Character. Similarity measure to use. Defaults to "cosine". See `similarity` for other options
- `weighted` Boolean. Should weighted ASPL and CC be used? Defaults to FALSE. Set to TRUE for weighted ASPL and CC
iter  Numeric. Number of iterations in bootstrap. Defaults to 1000
cores Numeric. Number of computer processing cores to use for bootstrapping samples. Defaults to $n/2$ total number of cores. Set to any number between 1 and maximum amount of cores on your computer (see parallel::detectCores())

Value

Returns a list containing:

dataMeas A matrix for the network input in the data argument, where columns are the semantic network measures from semnetmeas and rows are their values from each bootstrapped sample (results in a matrix with the dimensions iter by 3)
dataSumm Summary statistics across the bootstrapped samples for the network input in the data argument
prop Outputs the proportion used from the prop argument
iter Outputs the number of bootstrapped samples used from the iter argument

If a paired network is input, then also returns:

pairedMeas A matrix for the network input in the paired argument, where columns are the semantic network measures from semnetmeas and rows are their values from each bootstrapped sample (results in a matrix with the dimensions iter by 3)
pairedSumm Summary statistics across the bootstrapped samples for the network input in the paired argument

Author(s)
Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

# Simulate Dataset
one <- sim.fluency(20)

# Run bootstrap node-drop (partial) networks
one.result <- bootSemNeT(one, prop = .50, iter = 100, sim = "cosine", cores = 2, method = "TMFG", type = "node")

# Run bootstrap case-drop networks
## Includes additional equating argument: minCase
one.result <- bootSemNeT(one, iter = 100, sim = "cosine", cores = 2, method = "TMFG", type = "case", methodArgs = list(minCase = 2))

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

## Run
### Inclitudes additional NRW argument: threshold
open <- bootSemNeT(low, high, iter = 100, cores = 2, method = "NRW", type = "case", methodArgs = list(type = "num", threshold = 3))

---

**CC**

*Clustering Coefficient*

---

**Description**

Computes global clustering coefficient \( \text{CC} \)

**Usage**

\[
\text{CC}(A, \text{weighted} = \text{FALSE})
\]

**Arguments**

- **A**: An adjacency matrix of network data
- **weighted**: Is the network weighted? Defaults to `FALSE`. Set to `TRUE` for weighted measures of \( \text{CC} \) and \( \text{CCI} \)

**Value**

Returns the network’s CC

**Author(s)**

Alexander Christensen <alexpaulchristensen@gmail.com>

**References**


**Examples**

```r
# Pearson's correlation only for CRAN checks
A <- TMFG(similarity(sim.fluency(100), method = "cor"))

# Unweighted
cc <- CC(A)
```
Community Network Estimation

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)

Arguments

- data: Matrix or data frame. A preprocessed verbal fluency matrix where rows are participants and columns are verbal fluency responses
- window: Numeric. Size of window to look for co-occurrences in. Defaults to 2
- alpha: Numeric. Significance value. Defaults to .05
- enrich: Boolean. Should the network be enriched by connecting all nodes in their respective modules? Defaults to FALSE

Value

Returns a undirected semantic network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References


Examples

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

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

## Low and high openness to experience groups
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

#### Plots Networks for Comparison

**Description**

Uses `qgraph` to plot networks. Accepts any number of networks and will organize the plots in the number of side-by-side columns using the heuristic of taking the square root of the number of input and rounding down to the nearest integer (i.e., `floor(sqrt(length(input)))`).

**Examples**

- 3 networks: 1 x 3
- 6 networks: 2 x 3
- 9 networks: 3 x 3

#### Usage

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

#### Arguments

- `...`: Matrices or data frames of network adjacency matrices
- `title`: List. Characters denoting titles of plots
- `config`: Character. Defaults to "spring". See `qgraph` for more options
- `placement`: Character. How should nodes be placed when comparing groups? Defaults to "default"
  - "match" places nodes in the same position for all networks
  - "default" places nodes in the default `config` positions
- `weighted`: Boolean. Should networks be plotted with weights? Defaults to FALSE. Set to TRUE to plot networks with weights corresponding to association strength. Often, unweighted networks are more aesthetically representational of the networks
- `qgraph.args`: List. An argument list to be passed onto `qgraph`. See `qgraph` for possible arguments
Value
Plots networks using \texttt{qgraph}

Author(s)
Alexander Christensen <alexpaulchristensen@gmail.com>

References

Examples
\begin{verbatim}
# Simulate Datasets
one <- sim.fluency(10)
two <- sim.fluency(10)

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

# Compute networks
net1 <- TMFG(cos1)
et2 <- TMFG(cos2)

# Compare networks
compare_nets(net1, net2, title = list("One", "Two"), config = "spring")

# Change edge colors
compare_nets(net1, net2, title = list("One", "Two"),
config = "spring", qgraph.args = list(edge.color = "blue"))
\end{verbatim}

\textbf{convert2cytoscape} \hspace{1cm} \textit{Convert Adjacency Matrix to Cytoscape Format}

Description
Converts an adjacency matrix to Cytoscape’s sparse matrix format

Usage
\texttt{convert2cytoscape(A)}
Arguments

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

Value

A sparse matrix formatted for Cytoscape

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References


Examples

# Simulate Datasets
one <- sim.fluency(10)
two <- sim.fluency(10)

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

# Compute networks
net1 <- TMFG(cos1)
net2 <- 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(A, neural = FALSE)
Description

A function to "equate" multiple response matrices to one another. \( N \) number of groups are matched based on their responses so that every group has the same responses in their data

Usage

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

Arguments

`...` Matrices, data frames or a list of matrices and data frames. Binary response matrices to be equated

Value

This function returns a list containing the equated binary response matrices in the order they were input. The response matrices are labeled as the object name they were entered with

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>
Examples

# Obtain binary data
bin <- open.binary

# Finalize mat1
mat1 <- finalize(bin[c(1:5),])

# Finalize mat2
mat2 <- finalize(bin[c(6:10),])

# Equate mat1 and mat1
eq <- equate(mat1, mat2)

# Obtain respective equated response matrices
eq.mat1 <- eq$mat1 # list objects are named with the names
eq.mat2 <- eq$mat2 # they were entered with

**finalize**

### Finalize Response Matrix

**Description**

Finalizes the response matrix by keeping responses that are given by a certain number of people.

**Usage**

```r
finalize(rmat, minCase = 2)
```

**Arguments**

- `minCase` Numeric. Minimum number of cases to produce a response.

**Value**

A binary response matrix with responses given by at least `minCase` people.

**Author(s)**

Alexander Christensen <alexpaulchristensen@gmail.com>

**Examples**

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

# Finalize mat1
mat1 <- finalize(bin)
```
net.high  

*High Openness to Experience Network*

**Description**

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

**Usage**

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

**Format**

`net.high` (matrix, 160 x 160)

**References**


**Examples**

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

---

net.low  

*Low Openness to Experience Network*

**Description**

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

**Usage**

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

**Format**

`net.low` (matrix, 160 x 160)

**References**


**Examples**

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

Arguments

data Matrix or data frame. A preprocessed verbal fluency matrix where rows are participants and columns are verbal fluency responses

type Character. Type of threshold to apply.

• "num" Minimum number of co-occurrences

• "prop" Minimum proportion of co-occurrences

threshold Numeric. Value of the minimum number or proportion of co-occurrences. Defaults to 0 for both "num" and "prop"

Value

Returns a undirected semantic network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References


Examples

# Get data
data <- open.clean

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

## Low and high openness to experience groups
low <- data[which(group == "Low"),]
high <- data[which(group == "High"),]

# Compute networks
low.net <- NRW(low)
high.net <- NRW(high)

one.result

### Simulated Result for Dataset One

**Description**

A result of `bootSemNeT` from a simulated dataset

**Usage**

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

**Format**

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

**Examples**

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

---

open.binary

### Binary response Matrices (Openness and Verbal Fluency)

**Description**

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

**Usage**

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

**Format**

`open.binary` (matrix, 516 x 367)

**References**

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


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


Examples

data("open.group")

---

PF

Pathfinder Network

Description

Estimates a pathfinder network using the MST-Pathfinder Network method from Quirin et al. (2008; see also Schvaneveldt, 1990)

Usage

PF(data)

Arguments

data Matrix or data frame. A binary response matrix

Value

An adjacency matrix

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References


Examples

# Obtain data
data <- open.binary

# Estimate network
pf.net <- PF(data)
## plot.animateShiny

### Animate Networks for Spreading Activation from Shiny

#### Description

Uses `qgraph` and `ani.record` to animate networks. Accepts only one network animation at a time.

#### Usage

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

#### Arguments

- **x**: Shiny result `resultShiny$spreadingActivationPlot`
- **...**: Additional arguments for `ani.record`

#### Value

Plots animated networks using `qgraph` and `ani.record`

#### Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

#### References


#### Examples

```r
if(interactive())
{SemNeTShiny()}
## Not run:
plot(resultShiny$spreadingActivationPlot[[1]])
## End(Not run)```
Description

Plots output from `bootSemNeT`

Usage

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

Arguments

- `...`: Object(s) from `bootSemNeT`
- `groups`: Character. Labels for groups in the order they were entered in `bootSemNeT`
- `measures`: Character. Measures to be plotted

Value

Returns plots for the specified measures

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```r
# Simulate Dataset
one <- sim.fluency(20)

# Run partial bootstrap networks
one.result <- bootSemNeT(one, prop = .50, iter = 1000,
                         sim = "cosine", cores = 2, type = "node", method = "TMFG")

# Plot
plot(one.result, groups = c("One"))
```
plot.compareShiny

Plots Networks for Comparison from Shiny

Description

Uses `qgraph` to plot networks. Accepts any number of networks and will organize the plots in the number of side-by-side columns using the heuristic of taking the square root of the number of input and rounding down to the nearest integer (i.e., `floor(sqrt(length(input)))`). Performs the same operations as `compare_nets`.

Examples

- 3 networks: 1 x 3
- 6 networks: 2 x 3
- 9 networks: 3 x 3

Usage

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

Arguments

- `x`: Shiny result `resultShiny$comparePlot`
- `...`: Additional arguments

Value

Plots networks using `qgraph`

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References


Examples

```r
# Simulate Datasets
one <- sim.fluency(10)
two <- sim.fluency(10)

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

# Compute networks
net1 <- TMFG(cos1)
net2 <- TMFG(cos2)

# Compare networks
compare_nets(net1, net2, title = list("One", "Two"), config = "spring")

# Change edge colors
compare_nets(net1, net2, title = list("One", "Two"),
config = "spring", qgraph.args = list(edge.color = "blue"))
```

---

**Q**

### Modularity

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

**Usage**

```r
Q(A)
```

**Arguments**

- `A` An adjacency matrix of network data

**Value**

Returns Q or a measure of how well the communities in the network are compartmentalized.

**Author(s)**

Alexander Christensen <alexpaulchristensen@gmail.com>

**References**


Examples

# Pearson's correlation only for CRAN checks
A <- TMFG(similarity(sim.fluency(100), method = "cor"))

modularity <- Q(A)

Description

Performs significance tests for global measures of semantic networks against the global measures of equivalent size (and density) random networks

Usage

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

Arguments

... Matrices or data frames. Semantic networks to be compared against random networks
iter Numeric. Number of iterations in bootstrap. Defaults to 1000
cores Number of computer processing cores to use for bootstrapping samples. Defaults to n - 1 total number of cores. Set to any number between 1 and maximum amount of cores on your computer

Value

Returns a matrix containing p-values for the network measures of the input networks against the distribution of equivalent random networks. The last two columns contain the mean ("M.rand") and standard deviation ("SD.rand") of the network measures for the random network distribution

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Examples

```r
# 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
randnet.test(net.one, net.two, iter = 100, cores = 2)
```

---

**randwalk**

*Random Walk Simulation*

**Description**

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

**Usage**

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

**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
- **reps**: Numeric. Number of repetitions of increments in 10 steps. Defaults to 20
- **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 for bootstrapping samples. Defaults to n - 1 total number of cores. Set to any number between 1 and maximum amount of cores on your computer

**Value**

A result matrix containing the means and standard deviations for several measures as well as \( p \)-values for a Mann-Whitney U test

**Author(s)**

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


Examples

```r
# Simulate Datasets
one <- sim.fluency(10)
two <- sim.fluency(10)

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

# Compute networks
net1 <- TMFG(cos1)
net2 <- TMFG(cos2)

# Run random walk analysis
rw.results <- randwalk(net1, net2, iter = 100, cores = 2)
```

---

response.analysis  

**Response Analysis**

**Description**

Computes the difference in the total and unique number of responses between two groups (follows Christensen et al., 2018)

**Usage**

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

**Arguments**

```r
...
```

Matrix or data frame. Responses matrices for two different groups

**Value**

A list containing objects:

```r
total  
```

A vector with the total responses given by each participant. A t-test is used to compare, on average, whether one group provides more response than the other
unique A vector with the number of unique responses provided by both groups (Total Across Groups), the number of unique responses provided by each group (Total), and the number of unique responses provided by each group that were not provided by the other group (Unique). A McNemar’s test is used to compare whether the number of unique responses are different between groups.

Author(s)
Alexander Christensen <alexpaulchristensen@gmail.com>

References

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

# Perform analysis
response.analysis(data)
```

---

### semnetmeas

**Semantic Network Measures**

**Description**
Computes the average shortest path length (ASPL), clustering coefficient (CC), and modularity (Q) of the network.

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

**Arguments**
- **A**
  Matrix or data frame. An adjacency matrix of a network
- **meas**
  Character. Global network measures to compute. By default, computes ASPL, CC, and Q. Individual measures can be selected
- **weighted**
  Boolean. Should weighted measures be computed? Defaults to FALSE. Set to TRUE for weighted measures

**Value**
Returns a values for ASPL, CC, and Q.
**SemNeTShiny**

**Author(s)**
Alexander Christensen <alexpaulchristensen@gmail.com>

**Examples**

```r
# Simulate Datasets
one <- sim.fluency(10)

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

# Compute networks
net <- TMFG(cos)

# Compute global network measures
globmeas <- semnetmeas(net)
```

**Description**

An interactive Shiny application for running SemNet analysis.

**Usage**

```r
SemNeTShiny()
```

**Value**

A list called `resultShiny` containing:

- **data** The data imported into *SemNeTShiny*
- **group** The grouping variable imported into *SemNeTShiny*
- **network** The networks generated during *SemNeTShiny* session. The networks are labelled using the provided grouping variable
- **measures** Network measures ASPL (Average Shortest Path Lengths), CC (Clustering Coefficient), and Q (Modularity) for the networks generated during *SemNeTShiny* (see `semnetmeas`)
- **comparePlot** A visualization of the networks generated during *SemNeTShiny* (see `compare_nets`)
- **randomTest** Statistical results from the Random Network Test in *SemNeTShiny* (see `randnet.test`)
- **bootstrap** Results from the Bootstrap Network Analysis in *SemNeTShiny* (see `bootSemNet`
sim.fluency

- **bootstrapTest**: Statistical results from the Bootstrap Network Analysis (see `test.bootSemNeT`)
- **bootstrapPlot**: Plots of the statistical results from the Bootstrap Network Analysis (see `plot.bootSemNeT`)
- **randomWalk**: Results from the Random Walk Analysis (see `randwalk`)
- **bootstrapPlot**: Results from the Spreading Activation Analysis (see `spreadr`)

**Examples**

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

---

**Description**

Simulates verbal fluency data based on the number of nodes in the desired network. The summed total of each response is simulated from a poisson distribution (see `rpois`), using frequencies from the `animals.freq` data. Using these sums, participants responses are simulated with a probability of giving a response as the total of the summed response over the number of participants.

**Usage**

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

**Arguments**

- **nodes**: Numeric. Number of nodes to simulate in data. Defaults to 100
- **cases**: Numeric. Number of participants to simulate in data. Defaults to 500
- **random**: Boolean. Should the frequencies be randomly sampled from? Defaults to FALSE.

**Value**

A binary matrix with p (participants) by n (nodes)

**Author(s)**

Alexander Christensen <alexpaulchristensen@gmail.com>

**Examples**

```r
# Simulate data for 50 nodes and 200 participants
sim.fluency(nodes = 50, cases = 200)
```
similarity
Measures of Similarity

Description

Computes several measures of similarity (see Choi, Cha, & Tappert, 2010 for additional measures)

Usage

```r
similarity(
  data,
  method = c("angular", "cor", "cosine", "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.

Below are the definitions for each bin:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>a</th>
<th>b</th>
<th>a+b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>(R1)</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>b</td>
<td></td>
<td>(R2)</td>
</tr>
<tr>
<td>0</td>
<td>c</td>
<td>d</td>
<td></td>
<td>(N)</td>
</tr>
<tr>
<td>a+c</td>
<td>b+d</td>
<td>a+b+c+d</td>
<td>(C1) (C2) (N)</td>
<td></td>
</tr>
</tbody>
</table>

Options include:

- "angular" = $1 - \frac{2 \times \cos(\text{cosinesimilarity})}{\pi}$
- "cosine" = $a/\sqrt{(a+b)(a+c)}$
- "faith" = $a + 0.5d/a + b + c + d$
- "jaccard" = $a/a + b + c$
- "phi" and "cor" = $ad - bc/\sqrt{(R1 \times R2 \times C1 \times C2)}$
- "rr" = $a/a + b + c + d$

Value

A symmetric similarity matrix

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Examples

```
# Simulate Datasets
one <- sim.fluency(10)

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

---

**test.bootSemNeT**

Statistical tests for *bootSemNeT*

**Description**

Computes statistical tests for bootstrapped networks from *bootSemNeT*

**Usage**

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

**Arguments**

- `...` Object(s) from *bootSemNeT*
- `test` Character. Type of statistical test to be used.
  - "t-test" Computes t-tests for all measures
  - "ANOVA" Computes ANOVAs and includes Tukey’s HSD for pairwise comparisons (*TukeyHSD*)
  - "ANCOVA" Computes ANCOVAs that control for the number of nodes and edges in the networks and includes adjusted means and Tukey’s HSD for pairwise comparisons (*TukeyHSD*)
- `measures` Character. Network measures to be tested.
  - `ASPL` Average shortest path length
  - `CC` Clustering coefficient
  - `Q` Modularity statistic
- `formula` Character. A formula for specifying an ANOVA structure. The formula should have the predictor variable as "y" and include the names the variables are grouped by (e.g., `formula = "y ~ group_var1 * group_var2"`). See Two-way ANOVA example in examples
- `groups` Data frame. A data frame specifying the groups to be input into the formula. The column names should be the variable names of interest. The groups should be in the same order as the groups input into *bootSemNeT*
Value

Returns a list containing the objects:

- **ASPL**: Test statistics for each proportion of nodes remaining for ASPL
- **CC**: Test statistics for each proportion of nodes remaining for CC
- **Q**: Test statistics for each proportion of nodes remaining for Q

If two groups:

A matrix in each object has the following columns:

- **t-statistic**: Statistic from the \texttt{t.test}
- **df**: Degrees of freedom
- **p-value**: \(p\)-value with values equal to 0 being \(p < .001\)
- **d**: Cohen’s \(d\)
- **CI95.lower**: Lower bound of the 95 percent confidence interval
- **CI95.upper**: Upper bound of the 95 percent confidence interval

\textbf{Direction}: Direction of the effect. The argument \texttt{groups} will specify specifically which group is higher or lower on the measure. If no groups are input, then "d" and "p" are used to represent \texttt{data} and \texttt{paired} samples from \texttt{bootSemNeT}, respectively.

Row names refer to the proportion of nodes remaining in bootstrapped networks

If three or more groups:

A list containing two objects:

- **ANOVA**: A matrix containing the \(F\)-statistic, group degrees of freedom, residual degrees of freedom, \(p\)-value, and partial eta squared \(\eta^2\)
- **HSD**: A matrix containing the differences between each group (\texttt{diff}), lower (\texttt{lwr}) and upper (\texttt{upr}) bounds of the 95\% confidence interval, and the adjusted \(p\)-value \(p_{\text{adj}}\)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```r
# Simulate Dataset
one <- sim.fluency(20)
two <- sim.fluency(20)

two.result <- bootSemNeT(one, two, prop = .50, iter = 100, sim = "cosine", cores = 2, type = "node", method = "TMFG")

test.bootSemNeT(two.result)
```
# Two-way ANOVA example

## Simulated data

```r
hihi <- sim.fluency(50, 500)
hilo <- sim.fluency(50, 500)
lohi <- sim.fluency(50, 500)
lolo <- sim.fluency(50, 500)
```

## Create groups

```r
groups <- matrix(
  c("high", "high",
    "high", "low",
    "low", "high",
    "low", "low" ),
  ncol = 2, byrow = TRUE)
```

## Change column names (variable names)

```r
colnames(groups) <- c("gf","caq")
```

## Run partial bootstrap networks

```r
boot.fifty <- bootSemNeT(hihi, hilo, lohi, lolo, prop = .50,
  type = "node", method = "TMFG", cores = 2, iter = 100)
boot.sixty <- bootSemNeT(hihi, hilo, lohi, lolo, prop = .60,
  type = "node", method = "TMFG", cores = 2, iter = 100)
```

## Compute tests

```r
test.bootSemNeT(boot.fifty, boot.sixty,
  test = "ANOVA", formula = "y ~ gf*caq", groups = groups)
```

---

**TMFG**

**Triangulated Maximally Filtered Graph**

**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 \((3n - 6); \text{where } n \text{ is the number of variables}\). The TMFG algorithm begins by identifying four variables which have the largest sum of correlations to all other variables. Then, it iteratively adds each variable with the largest sum of three correlations to nodes already in the network until all variables have been added to the network. This structure can be associated with the inverse correlation matrix (i.e., precision matrix) to be turned into a GGM (i.e., partial correlation network) by using LoGo. See Details for more information on this network estimation method.

**Usage**

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

data  Matrix or data frame. Must be a square matrix
depend  Boolean. Is network a dependency (or directed) network? Defaults to FALSE. Set to TRUE to generate a TMFG-filtered dependency network (output obtained from the depend function)

Details

The TMFG method applies a structural constraint on the network, which restrains the network to retain a certain number of edges (3n - 6, where n is the number of nodes; Massara et al., 2016). The network is also composed of 3- and 4-node cliques (i.e., sets of connected nodes; a triangle and tetrahedron, respectively). The TMFG method constructs a network using zero-order correlations and the resulting network can be associated with the inverse covariance matrix (yielding a GGM; Barfuss, Massara, Di Matteo, & Aste, 2016). Notably, the TMFG can use any association measure and thus does not assume the data is multivariate normal.

Construction begins by forming a tetrahedron of the four nodes that have the highest sum of correlations that are greater than the average correlation in the correlation matrix. Next, the algorithm iteratively identifies the node that maximizes its sum of correlations to a connected set of three nodes (triangles) already included in the network and then adds that node to the network. The process is completed once every node is connected in the network. In this process, the network automatically generates what’s called a planar network. A planar network is a network that could be drawn on a sphere with no edges crossing (often, however, the networks are depicted with edges crossing; Tumminello, Aste, Di Matteo, & Mantegna, 2005).

Value

Returns an TMFG filtered adjacency matrix

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References


Examples

# Pearson's correlation only for CRAN checks
A <- TMFG(similarity(sim.fluency(100), method = "cor"))
two.result  

*Simulated Result for Dataset One and Two*

**Description**

A result of `bootSemNeT` from two simulated datasets

**Usage**

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

**Format**

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

**Examples**

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

---

vignette.plots  

*Plots for Vignette*

**Description**

Plots for vignette taken from Christensen & Kenett (2019)

**Usage**

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

**Format**

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

**References**


**Examples**

```r
data("vignette.plots")```
Index

* datasets
  animals.freq, 3
  net.high, 14
  net.low, 14
  one.result, 16
  open.binary, 16
  open.clean, 17
  open.group, 17
  two.result, 34
  vignette.plots, 34

ani.record, 19
animals.freq, 3, 28
ASPL, 4, 30

bootSemNeT, 5, 16, 20, 27, 30, 31, 34

CC, 7, 30
CN, 5, 8
compare_nets, 9, 21, 27
convert2cytoscape, 10
convert2igraph, 11
convertConnBrainMat, 12

depend, 33
equate, 12
finalize, 13
igraph, 11, 12
LoGo, 32

net.high, 14
net.low, 14
NRW, 5, 15

one.result, 16
open.binary, 16
open.clean, 17, 17

open.group, 17
PF, 5, 18
plot.animateShiny, 19
plot.bootSemNet, 20, 28
plot.compareShiny, 21
Q, 22, 30
qgraph, 9, 10, 19, 21

randnet.test, 23, 27
randwalk, 24, 28
response.analysis, 25
rpois, 28

SemNet, 27
SemNet(SemNet-package), 2
SemNet-package, 2
semnetmeas, 6, 26, 27
SemNetShiny, 27, 27
sim.fluency, 3, 28
similarity, 5, 29
spreadr, 28

t.test, 31
test.bootSemNet, 28, 30
textcleaner, 5, 13
TMFG, 5, 32
TukeyHSD, 30
two.result, 34

vignette.plots, 34