Package ‘bootnet’

February 7, 2026

Type Package

Title Bootstrap Methods for Various Network Estimation Routines

Version 1.7.1

Maintainer Sacha Epskamp <`mail@sachaepskamp.com`>

Depends ggplot2, R (>= 3.0.0)

Imports methods, igraph, IsingFit (>= 0.4), qgraph, dplyr (>= 0.3.0.2), tidyr, gtools, corpcor, IsingSampler (>= 0.2.3), mvtnorm, abind, Matrix, snow, mgm (>= 1.2), NetworkToolbox (>= 1.1.0), pbapply, networktools, rlang, tibble, tidyselect

Suggests glasso, BDgraph, graphicalVAR, relaimpo, lavaan, psychTools

Description Bootstrap methods to assess accuracy and stability of estimated network structures and centrality indices <doi:10.3758/s13428-017-0862-1>. Allows for flexible specification of any undirected network estimation procedure in R, and offers default sets for various estimation routines.

License GPL-2

BugReports `https://github.com/SachaEpskamp/bootnet/issues`

URL `https://github.com/SachaEpskamp/bootnet`

RoxygenNote 7.3.3

NeedsCompilation no

Author Sacha Epskamp [aut, cre],
Eiko I. Fried [ctb]

Repository CRAN

Date/Publication 2026-02-07 06:20:08 UTC

Contents

`bootnet-package` *Bootstrap Methods for Various Network Estimation Routines*

Description

Bootstrap standard errors on various network estimation routines, such as EBICglasso from the qgraph package and IsingFit from the IsingFit package. See `bootnet`

Author(s)

Sacha Epskamp

Maintainer: Sacha Epskamp <mail@sachaepskamp.com>

See Also

`bootnet`

`binarize` *Binarizes a dataset*

Description

This function will transform data into binary data (0,1). If the data is already binary, this function does nothing.

Usage

```
binarize(x, split = "median", na.rm = TRUE, removeNArows = TRUE, verbose = TRUE)
```

Arguments

`x`	A data frame or matrix
`split`	Either a function to split on (as character or as function) or a vector. e.g., `split = "mean"` will split every variable on the mean of that variable, `split=2` will make every value above 2 a 1 and every value below 2 a 0 and a vector of the same length as each variable in the dataset will use those elements to split.
`na.rm`	The `na.rm` argument used in the split function.
`removeNArows`	Logical, should rows with `NA` be removed?
`verbose`	Output progress to the console?

Value

A binarized data frame

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

`bootInclude` *Inclusion proportion graph*

Description

This function takes bootstrap results and returns a inclusion probability network (edge weights indicate how often a certain edge was included in the model). Note that the plotting method automatically uses a black-white color scheme (as edges are not signed and always positive).

Usage

```
bootInclude(bootobject, verbose = TRUE)
```

Arguments

`bootobject`	Nonparametric bootstrap results from `bootnet`
`verbose`	Logical, should progress be reported to the console?

Value

A `bootnetResult` object with the following elements:

`graph`	The weights matrix of the network
`intercepts`	The intercepts
`results`	The results of the estimation procedure
`labels`	A vector with node labels
`nNodes`	Number of nodes in the network
`nPerson`	Number of persons in the network
`input`	Input used, including the result of the default set used

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

See Also

`bootnet`, `estimateNetwork`

Examples

```
## Not run:
# BFI Extraversion data from psychTools package:
library("psychTools")
data(bfi)
# Subset of data:
bfiSub <- bfi[1:250,1:25]

# Estimate ggmModSelect networks (not stepwise to increase speed):
Network <- estimateNetwork(bfiSub], default = "ggmModSelect", corMethod = "cor",
              stepwise = FALSE)

# Bootstrap 100 values, using 8 cores (100 to incease speed, preferably 1000+):
boots <- bootnet(Network, nBoots = 100, nCores = 8)

# Threshold network:
Network_inclusion <- bootInclude(boots)

# Plot:
plot(Network_inclusion)

## End(Not run)
```

`bootnet` *Bootstrapped network estimation*

Description

This function can be used to bootstrap network estimation methods so that the spread of parameter and centrality estimates can be assessed. Most important methods are `type = 'nonparametric'` for the non-parametric bootstrap and `type = 'case'` for the case-dropping bootstrap. See also Epskamp, Borsboom and Fried (2016) for more details.

Usage

```
bootnet(data, nBoots = 1000, default = c("none", "EBICglasso",
                  "ggmModSelect", "pcor", "IsingFit", "IsingSampler",
                  "huge", "adalasso", "mgm", "relimp", "cor", "TMFG",
                  "ggmModSelect", "LoGo", "SVAR_lavaan", "GGMncv"), type
                  = c("nonparametric", "parametric", "node", "person",
                  "jackknife", "case"), nCores = 1, statistics =
                  c("edge", "strength", "outStrength", "inStrength"),
                  model = c("detect", "GGM", "Ising", "graphicalVAR"),
                  fun, verbose = TRUE, labels, alpha = 1, caseMin =
                  0.05, caseMax = 0.75, caseN = 10, subNodes, subCases,
                  computeCentrality = TRUE, propBoot = 1, replacement =
                  TRUE, graph, sampleSize, intercepts, weighted, signed,
                  directed, includeDiagonal = FALSE, communities,
                  useCommunities, bridgeArgs = list(), library =
                  .libPaths(), memorysaver = TRUE, ...)
```

Arguments

`data`	A data frame or matrix containing the raw data. Must be numeric, integer or ordered factors.
`nBoots`	Number of bootstraps
`default`	A string indicating the method to use. See documentation at `estimateNetwork`.
`type`	The kind of bootstrap method to use.
`nCores`	Number of cores to use in computing results. Set to 1 to not use parallel computing.
`statistics`	Vector indicating which statistics to store. Options are: `"edge"` Edge-weight `"strength"` Degree or node-strength `"outStrength"` Out-degree or Out-strength `"inStrength"` In-degree or In-strength `"expectedInfluence"` Expected Influence

	`"outExpectedInfluence"` Outgoing expected influence `"inExpectedInfluence"` Incoming expected influence `"bridgeInDegree"` Bridge in-degree (see `bridge`) `"bridgeOutnDegree"` Bridge out-degree (see `bridge`) `"bridgeStrength"` Bridge-strength (see `bridge`) `"bridgeCloseness"` Bridge-closeness (see `bridge`) `"bridgeBetweenness"` Bridge-betweenness (see `bridge`) `"rspbc"` Randomized shortest paths betweenness centrality (see `rspbc`) `"hybrid"` Hybrid centrality (see `hybrid`) `"eigenvector"` Eigenvector centrality (see `eigenvector`) Can contain `"edge"`, `"strength"`, `"closeness"`, `"betweenness"`, `"length"`, `"distance"`, `"expectedInfluence"`, `"inExpectedInfluence"`, `"outExpectedInfluence"`. By default, length and distance are not stored.
`model`	The modeling framework to use. Automatically detects if data is binary or not.
`fun`	A custom estimation function, when no default set is used. This must be a function that takes the data as input (first argument) and returns either a weights matrix or a list containing the elements `"graph"` for the weights matrix, `"intercepts"` for the intercepts (optional) and `"results"` for the full estimation results (optional).
`verbose`	Logical. Should progress of the function be printed to the console?
`labels`	A character vector containing the node labels. If omitted the column names of the data are used.
`alpha`	The centrality tuning parameter as used in `centrality`.
`subNodes`	Range of nodes to sample in node-drop bootstrap
`caseMin`	Minimum proportion of cases to drop when `type = "case"`.
`caseMax`	Maximum proportion of cases to drop when `type = "case"`.
`caseN`	Number of sampling levels to test when `type = "case"`.
`subCases`	Range of persons to sample in person-drop bootstrap
`computeCentrality`	Logical, should centrality be computed?
`propBoot`	Proportion of persons to sample in bootstraps. Set to lower than 1 for m out of n bootstrap
`replacement`	Logical, should replacement be used in bootstrap sampling?
`graph`	A given network structure to use in parametric bootstrap.
`sampleSize`	The samplesize to use in parametric bootstrap.
`intercepts`	Intercepts to use in parametric bootstrap.
`weighted`	Logical, should the analyzed network be weighted?
`signed`	Logical, should the analyzed network be signed?
`directed`	Logical, is the analyzed network directed? Usually does not have to be set and is detected automatically.

`includeDiagonal`	Logical, should diagonal elements (self-loops) be included in the bootstrap? Only used when `directed = TRUE`.
`communities`	Used for bridge centrality measures (see `bridge`).
`useCommunities`	Used for bridge centrality measures (see `bridge`).
`library`	Library location to be used in parallel computing.
`memorysaver`	Logical. If TRUE (recommended) then raw bootstrapped data and results are not stored in the output object. This saves a lot of memory. Set this only to TRUE if you need the raw results or bootstrap data.
`bridgeArgs`	List of arguments used in the 'bridge' function for computing bridge centrality
`...`	Additional arguments used in the estimator function.

Value

A `bootnet` object with the following elements:

`sampleTable`	A data frame containing all estimated values on the real sample.
`bootTable`	A data frame containing all estimated values on all bootstrapped samples.
`sample`	A `bootnetResult` object with plot and print method containing the estimated network of the real sample.
`boots`	A list of `bootnetResult` objects containing the raw bootstrap results.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

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

See Also

`estimateNetwork`, `differenceTest`, `corStability`, `plot.bootnet`, `summary.bootnet`

Examples

```
# BFI Extraversion data from psychTools package:
library("psychTools")
data(bfi)
bfiSub <- bfi[,1:25]

# Estimate network:
Network <- estimateNetwork(bfiSub, default = "EBICglasso")

# Centrality indices:
library("qgraph")
centralityPlot(Network)
```

```
# Estimated network:
plot(Network, layout = 'spring')

### Non-parametric bootstrap ###
# Bootstrap 1000 values, using 8 cores:
Results1 <- bootnet(Network, nBoots = 1000, nCores = 8)

# Plot bootstrapped edge CIs:
plot(Results1, labels = FALSE, order = "sample")

# Plot significant differences (alpha = 0.05) of edges:
plot(Results1, "edge", plot = "difference",onlyNonZero = TRUE,
     order = "sample")

# Plot significant differences (alpha = 0.05) of node strength:
plot(Results1, "strength", plot = "difference")

# Test for difference in strength between node "A1" and "C2":
differenceTest(Results1,  "A1", "C2", "strength")

### Case-drop bootstrap ###
# Bootstrap 1000 values, using 8 cores:
Results2 <- bootnet(Network, nBoots = 1000, nCores = 8,
              type = "case")

# Plot centrality stability:
plot(Results2)

# Compute CS-coefficients:
corStability(Results2)
```

`bootThreshold` *Threshold network based on bootstrapped intervals*

Description

This function takes the output of `bootnet` and returns a network as if it had been estimated using `estimateNetwork`, but with edges removed (set to zero) based on some significance level.

Usage

```
bootThreshold(bootobject, alpha = 0.05, verbose = TRUE, thresholdIntercepts = FALSE)
```

Arguments

`bootobject`	Nonparametric bootstrap results from `bootnet`
`alpha`	Significance level

`verbose`	Logical, should progress be reported to the console?
`thresholdIntercepts`	Logical, should intercepts also be thresholded?

Value

A `bootnetResult` object with the following elements:

`graph`	The weights matrix of the network
`intercepts`	The intercepts
`results`	The results of the estimation procedure
`labels`	A vector with node labels
`nNodes`	Number of nodes in the network
`nPerson`	Number of persons in the network
`input`	Input used, including the result of the default set used

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

See Also

`bootnet`, `estimateNetwork`

Examples

```
## Not run:
# BFI Extraversion data from psychTools package:
library("psychTools")
data(bfi)
bfiSub <- bfi[,1:25]

# Estimate unregularized network:
Network <- estimateNetwork(bfiSub, default = "pcor", corMethod = "cor")

# Bootstrap 1000 values, using 8 cores:
boots <- bootnet(Network, nBoots = 1000, nCores = 8)

# Threshold network:
Network_thresholded <- bootThreshold(boots)

# Plot:
plot(Network_thresholded)

## End(Not run)
```

`corStability` *Correlation stability coefficient*

Description

This coefficient denotes the estimated maximum number of cases that can be dropped from the data to retain, with 95% probability, a correlation of at least 0.7 (default) between statistics based on the original network and statistics computed with less cases. This coefficient should not be below 0.25 and is preferably above 0.5. See also Epskamp, Borsboom and Fried (2016) for more details.

Usage

```
corStability(x, cor = 0.7, statistics = "all", verbose = TRUE)
```

Arguments

`x`	Output of `bootnet`. Must be case-drop bootstrap.
`cor`	The correlation level tot est at.
`statistics`	The statistic(s) to test for. Can also be `"all"`.
`verbose`	Logical, should information on the progress be printed to the console?

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Epskamp, S., Borsboom, D., & Fried, E. I. (2016). Estimating psychological networks and their accuracy: a tutorial paper. arXiv preprint, arXiv:1604.08462.

See Also

`bootnet`

Examples

```
## Not run:
# BFI Extraversion data from psychTools package:
library("psychTools")
data(bfi)
bfiSub <- bfi[,1:25]

# Estimate network:
Network <- estimateNetwork(bfiSub, default = "EBICglasso")

# Bootstrap 1000 values, using 8 cores:
# Bootstrap 1000 values, using 8 cores:
Results2 <- bootnet(Network, nBoots = 1000, nCores = 8,
```

```
                type = "case")

# Compute CS-coefficients:
corStability(Results2)

## End(Not run)
```

`differenceTest` *Bootstrapped difference test*

Description

This function computes the bootstrapped difference test for edge-weights and centrality indices. A confidence interval is constructured on the difference of two values, and the test is deemed significant if zero is not in this confidence interval. See also Epskamp, Borsboom and Fried (2016) for more details.

Usage

```
differenceTest(bootobject, x, y, measure = c("strength", "closeness", "betweenness"),
                  alpha = 0.05, x2, y2, verbose = TRUE)
```

Arguments

`bootobject`	Output of `bootnet`. Must be nonparametric or parametric bootstrap.
`x`	A character string indicating the ID of a node or an edge, or a number indiacting the node or edge. For an edge, can be e.g., `"1--2"` or `"x = 1, x2 = 2"`.
`y`	A character string indicating the ID of a node or an edge, or a number indiacting the node or edge. For an edge, can be e.g., `"1--2"` or `"y = 1, y2 = 2"`.
`measure`	Measure to test. Can be `"strength"`, `"closeness"`, `"betweenness"`, `"edge"` or `"distance"`.
`alpha`	Signifiance level to test at. Note that the actual signifiance level is influenced by the number of bootstrap samples, and is returned in a message.
`x2`	Second node in an edge. optional.
`y2`	Second node in an edge. optional.
`verbose`	Logical, should the message indiacting actual signifiance level be printed?

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Epskamp, S., Borsboom, D., & Fried, E. I. (2016). Estimating psychological networks and their accuracy: a tutorial paper. arXiv preprint, arXiv:1604.08462.

See Also

bootnet

Examples

```
## Not run:
# BFI Extraversion data from psychTools package:
library("psychTools")
data(bfi)
bfiSub <- bfi[,1:25]

# Estimate network:
Network <- estimateNetwork(bfiSub, default = "EBICglasso")

# Bootstrap 1000 values, using 8 cores:
Results1 <- bootnet(Network, nBoots = 1000, nCores = 8)

# Test for difference in strength between node "A1" and "C2":
differenceTest(Results,  "A1", "C2", "strength")

# Test for difference between edge N1--N2 and N3--N4:
differenceTest(Results,  "N1--N2", "N3--N4", "edge")

# Alternative:
differenceTest(Results,  x = "N1", x2 = "N2", y = "N3",
                y2 = "N4", measure = "edge")

## End(Not run)
```

`estimateNetwork` *Estimate a network structure*

Description

This function allows for flexible estimation of a network structure using various R packages and model frameworks. This is typically done by using one of the default sets. See details for manual specification. See also Epskamp, Borsboom and Fried (2016) for more details. IMPORTANT: THE ESTIMATOR FUNCTIONS (e.g., `fun = bootnet_pcor`) ARE NOT INTENDED TO BE USED MANUALY (see details).

Usage

```
estimateNetwork(data, default = c("none", "EBICglasso", "pcor",
                   "IsingFit", "IsingSampler", "huge", "adalasso", "mgm",
                   "relimp", "cor", "TMFG", "ggmModSelect", "LoGo",
                   "graphicalVAR", "piecewiseIsing", "SVAR_lavaan", "GGMncv"),
                   fun, labels, verbose = TRUE, .dots = list(), weighted = TRUE,
                   signed = TRUE, directed, datatype, checkNumeric = FALSE, ...,
                   .input, memorysaver = FALSE)
```

```
bootnet_EBICglasso(data, tuning = 0.5, corMethod = c("cor", "cov",
                    "cor_auto", "npn", "spearman"), missing =
                    c("pairwise", "listwise", "fiml", "stop"), sampleSize
                    = c("pairwise_average", "maximum", "minimum",
                    "pairwise_maximum", "pairwise_minimum",
                    "pairwise_average_v1.5", "pairwise_maximum_v1.5",
                    "pairwise_minimum_v1.5"), verbose = TRUE, corArgs =
                    list(), refit = FALSE, principalDirection = FALSE,
                    lambda.min.ratio = 0.01, nlambda = 100, threshold =
                    FALSE, unlock = FALSE, nonPositiveDefinite = c("stop",
                    "continue"), transform = c("none", "rank",
                    "quantile"), ...)

bootnet_pcor(data, corMethod = c("cor", "cov", "cor_auto", "npn",
                    "spearman"), missing = c("pairwise", "listwise",
                    "fiml", "stop"), sampleSize = c("pairwise_average",
                    "maximum", "minimum", "pairwise_maximum",
                    "pairwise_minimum", "pairwise_average_v1.5",
                    "pairwise_maximum_v1.5", "pairwise_minimum_v1.5"),
                    verbose = TRUE, corArgs = list(), threshold = 0, alpha
                    = 0.05, adjacency, principalDirection = FALSE, unlock
                    = FALSE, nonPositiveDefinite = c("stop", "continue"),
                    transform = c("none", "rank", "quantile"))

bootnet_cor(data, corMethod = c("cor", "cov", "cor_auto", "npn",
                    "spearman"), missing = c("pairwise", "listwise",
                    "fiml", "stop"), sampleSize = c("pairwise_average",
                    "maximum", "minimum", "pairwise_maximum",
                    "pairwise_minimum", "pairwise_average_v1.5",
                    "pairwise_maximum_v1.5", "pairwise_minimum_v1.5"),
                    verbose = TRUE, corArgs = list(), threshold = 0, alpha
                    = 0.05, principalDirection = FALSE, unlock = FALSE,
                    nonPositiveDefinite = c("stop", "continue"), transform
                    = c("none", "rank", "quantile"))

bootnet_IsingFit(data, tuning = 0.25, missing = c("listwise", "stop"),
                    verbose = TRUE, rule = c("AND", "OR"), split =
                    "median", principalDirection = FALSE,
                    min_sum = -Inf, unlock = FALSE)

bootnet_IsingSampler(data, missing = c("listwise", "stop"), verbose = TRUE,
                    split = "median", method = c("uni", "ll", "pl", "bi"),
                    principalDirection = FALSE, unlock = FALSE, threshold
                    = FALSE, alpha = 0.01, min_sum = -Inf, rule = c("AND",
                    "OR"))

bootnet_adalasso(data, missing = c("listwise", "stop"), verbose = TRUE,
```

```
                   nFolds = 10, principalDirection = FALSE, unlock =
                   FALSE, transform = c("none", "rank", "quantile"), ...)

bootnet_huge(data, tuning = 0.5, missing = c("listwise", "stop"),
                   verbose = TRUE, npn = TRUE, criterion = c("ebic",
                   "ric", "stars"), principalDirection = FALSE,
                   lambda.min.ratio = 0.01, nlambda = 100, unlock =
                   FALSE, transform = c("none", "rank", "quantile"), ...)

bootnet_mgm(data, type, level, tuning = 0.25, missing =
                   c("listwise", "stop"), verbose = TRUE, criterion =
                   c("EBIC", "CV"), nFolds = 10, order = 2, rule =
                   c("AND", "OR"), binarySign, unlock = FALSE, transform
                   = c("none", "rank", "quantile"), ...)

bootnet_relimp(data, normalized = TRUE, type = "lmg",
                   structureDefault = c("none", "custom", "EBICglasso",
                   "pcor", "IsingFit", "IsingSampler", "huge",
                   "adalasso", "mgm", "cor", "TMFG", "ggmModSelect",
                   "LoGo"), missing = c("listwise", "stop"), ..., verbose
                   = TRUE, threshold = 0, unlock = FALSE, transform =
                   c("none", "rank", "quantile"))

bootnet_TMFG(data, graphType = c("cor", "pcor"), corMethod =
                   c("cor", "cov", "cor", "npn", "cor_auto"), missing =
                   c("pairwise", "listwise", "fiml", "stop"), verbose =
                   TRUE, corArgs = list(), principalDirection = FALSE,
                   unlock = FALSE, transform = c("none", "rank",
                   "quantile"), ...)

bootnet_LoGo(data, corMethod = c("cor", "cov", "cor", "npn",
                   "cor_auto"), missing = c("pairwise", "listwise",
                   "fiml", "stop"), verbose = TRUE, corArgs = list(),
                   principalDirection = FALSE, unlock = FALSE, transform
                   = c("none", "rank", "quantile"), ...)

bootnet_graphicalVAR(data, tuning = 0.5, verbose = TRUE, principalDirection
                   = FALSE, missing = c("listwise", "stop"), unlock =
                   FALSE, transform = c("none", "rank", "quantile"), ...)

bootnet_ggmModSelect(data, tuning = 0, corMethod = c("cor", "cov",
                    "cor_auto", "npn", "spearman"), missing =
                    c("pairwise", "listwise", "fiml", "stop"), sampleSize
                    = c("pairwise_average", "maximum", "minimum",
                    "pairwise_maximum", "pairwise_minimum",
                    "pairwise_average_v1.5", "pairwise_maximum_v1.5",
                    "pairwise_minimum_v1.5"), verbose = TRUE, corArgs =
                    list(), principalDirection = FALSE, start =
```

```
                    c("glasso", "empty", "full"), stepwise = TRUE, nCores
                    = 1, unlock = FALSE, nonPositiveDefinite = c("stop",
                    "continue"), transform = c("none", "rank",
                    "quantile"), ...)

bootnet_piecewiseIsing(data, cutoff, missing = c("listwise", "stop"), verbose
                    = TRUE, IsingDefault = c("IsingSampler", "IsingFit",
                    "custom"), zeroThreshold = 1, minimalN = ncol(data) +
                    1, unlock = FALSE, ...)

bootnet_SVAR_lavaan(data, verbose = TRUE, principalDirection = FALSE,
                   missing = c("listwise", "stop"), criterion = "bic",
                   eqThreshold = 1e-04, tempWhitelist, tempBlacklist,
                   contWhitelist, contBlacklist, minimalModInd = 10,
                   unlock = FALSE, transform = c("none", "rank",
                   "quantile"), ...)

bootnet_GGMncv(data, penalty = c("atan", "selo", "exp", "log",
                    "sica", "scad", "mcp", "lasso"), corMethod = c("cor",
                    "cov", "cor_auto", "npn", "spearman"), missing =
                    c("pairwise", "listwise", "fiml", "stop"), sampleSize
                    = c("pairwise_average", "maximum", "minimum",
                    "pairwise_maximum", "pairwise_minimum",
                    "pairwise_average_v1.5", "pairwise_maximum_v1.5",
                    "pairwise_minimum_v1.5"), verbose = TRUE, corArgs =
                    list(), principalDirection = FALSE, unlock = FALSE,
                    nonPositiveDefinite = c("stop", "continue"), transform
                    = c("none", "rank", "quantile"), ...)
```

Arguments

`data`	A data frame or matrix containing the raw data. Must be numeric, integer or ordered factors.
`default`	A string indicating the method to use. Specifying a `default` sets default values to `prepFun`, `prepArgs`, `estFun`, `estArgs`, `graphFun`, `graphArgs`, `intFun` and `intArgs`. Setting a `default` can be omitted but that does require specifying all above mentioned arguments. Current options are: `"EBICglasso"` Gaussian Markov random field estimation using graphical LASSO and extended Bayesian information criterion to select optimal regularization parameter. Using `EBICglasso` from the qgraph package. Calls `bootnet_EBICglasso`. `"IsingFit"` Ising model estimation using LASSO regularized nodewise logistic regression and extended Bayesian information criterion to select optimal regularization parameter. Using `IsingFit` from the IsingFit package. Calls `bootnet_IsingFit`. `"IsingSampler"` Calls the `EstimateIsing` function from the IsingSampler package. `"pcor"` Partial correlation network (non-regularized Gaussian Markov random field), using `cor2pcor` from the corpcor package. Calls `bootnet_pcor`.

	`"cor"` Correlation network. `"huge"` Uses EBIC model selection of GGM networks estimated via the glasso algorithm as implemented in the huge package (as opposed to glasso and qgraph packages used in `default = "EBICglasso"`). Uses nonparanormal transformation in preparing the data and does not use polychoric correlations. Calls `bootnet_huge`. `"mgm"` Estimates a Mixed graphical model by using the the mgm (or mgmfit in older versions) function of the mgm package. Calls `bootnet_mgm`. `"TMFG"` Estimates a Triangulated Maximally Filtered Graph, using the function `TMFG` of the NetworkToolbox package. Calls `bootnet_TMFG`. Note that this estimates a *correlation network* by default (use the 'graphType' argument to estimate a partial correlation network instead). `"LoGo"` Estimates a Local/Global Sparse Inverse Covariance Matrix, using the function `LoGo` of the NetworkToolbox package. Calls `bootnet_LoGo`. `"relimp"` Estimates a (directed) relative importance network, using the function 'calc.relimp' of the 'relaimpo' package. The 'structureDefault' argument can be used to use a different default set for estimating the structure of the graph. Calls `bootnet_relimp`. `"ggmModSelect"` Estimates an unregularized GGM using the glasso algorithm and stepwise model selection, using the 'ggmModSelect' function from the qgraph package. Calls `bootnet_ggmModSelect`. `"graphicalVAR"` Estimates a graphical VAR model using the graphicalVAR package. This results in two networks which can be plotted using the 'graph' argument in the plot method. Calls `bootnet_graphicalVAR`. See details section for a more detailed description.
`fun`	A custom estimation function, when no default set is used. This must be a function that takes the data as input (first argument) and returns either a weights matrix or a list containing the elements `"graph"` for the weights matrix, `"intercepts"` for the intercepts (optional) and `"results"` for the full estimation results (optional).
`tuning`	EBIC tuning parameter, used in 'EBICglasso', 'IsingFit', 'huge', 'mgm' and 'ggmModSelect' default sets. Note that the default value differs: 'EBICglasso', 'huge' and 'mgm' use 0.5, 'IsingFit' uses 0.25 and 'ggmModSelect uses 0.
`corMethod`	Correlation method, used in 'EBICglasso' and 'pcor' default sets. `"cor_auto"` uses `cor_auto` for polychoric and polyserial correlations, `"cov"` uses the `cov` function for covariances, `"cor"` will use the `cor` function for correlations and `"npn"` will apply the nonparanormal transformation (via `huge::huge.npn`) and then compute correlations.
`missing`	How to handle missing data? `"pairwise"` for pairwise deletion, `"listwise"` for listwise deletion, `"fiml"` for full-information maximum likelihood and `"stop"` to stop with an error.
`sampleSize`	How will sample size be computed in EBICglasso default set? The default `"pairwise_average"` will set the sample size to the average of sample sizes used for each individual correlation. Other options are `"pairwise_maximum"` (largest sample sized used for each individual correlation), `"pairwise_minimum"` (smallest sample sized used for each individual correlation), `"maximum"` (takes

	total number of rows including rows with NA), and `"minimum"` (takes total number of rows that contain no NA). The arguments "pairwise_average_v1.5", "pairwise_minimum_v1.5", and "pairwise_maximum_v1.5" can be used to mimic bootnet's behavior in version 1.5 and earlier (which also computed the sample size based on the sample sizes for the variances).
`corArgs`	A list with arguments for the function used defined by `corMethod`.
`threshold`	Thresholding to use in partial correlation networks. Can be a fixed number to threshold all absolute edges below this value, `'locfdr'` for local FDR, or any option corresponding to adjustments in `corr.p` (`'none'`, `'sig'`, `'holm'`, `'hochberg'`, `'hommel'`, `'bonferroni'`, `'BH'`, `'BY'` or `'fdr'`). Can also be used for `default = "IsingSampler"` but can only be set to a logical enabling or disabling significance thresholding.
`refit`	Logical used in EBICglasso default set: should the estimated model structure be refitted without LASSO regularization?
`rule`	The rule to use to select an edge in nodewise estimation. `"AND"` to only select in edge if both regression coefficients are nonzero and `"OR"` if only one is nonzero. Used in 'IsingFit' and 'mgm' default sets.
`split`	A function or character string (`"median"` or `"mean"`) indicating how to binarize values when estimating an Ising model.
`method`	The estimation method used in the IsingSampler default set (see `EstimateIsing`).
`npn`	Logical, should nonparanormal be used in huge default set?
`criterion`	The criterion used in model selection. `"ebic"`, `"ric"` or `"stars"` in the huge default set or `"EBIC"` or `"CV"` in the mgm default set.
`nFolds`	Number of folds used in k-fold cross-validation.
`type`	For mgm, see mgm or mgmfit; for relative importance networks, see `calc.relimp`
`level`	See `mgm`. Automatically set if not assigned.
`order`	Order up until including which interactions are included in the model. See `mgm`. Automatically set if not assigned.
`binarySign`	See `mgm`. Automatically set if not assigned.
`normalized`	Should normalized relative importance be used in relative importance networks?
`structureDefault`	In relative importance networks, default set used to compute the graph structure. Any other arguments used (using ...) are sent to the graph estimator function as well.
`graphType`	`"cor"` to estimate a correlation network and `"pcor"` to estimate a partial correlation network (GGM)
`alpha`	Significance level to test at.
`principalDirection`	Rescales variables according to the sign of the first eigen-vector. This will lead to most correlations to be positive (positive manifold), leading to negative edges to be substantively interpretable.
`stepwise`	Logical indicating if 'ggmModSelect' should use stepwise estimation.
`start`	See `ggmModSelect`

`labels`	A character vector containing the node labels. If omitted the column names of the data are used.
`verbose`	Logical, currently only used when `default = "EBICglasso"` in the `cor_auto` function.
`weighted`	Logical, should the analyzed network be weighted?
`signed`	Logical, should the analyzed network be signed?
`directed`	Logical, is the analyzed network directed? Usually does not have to be set and is detected automatically.
`datatype`	`"normal"` if the data argument is a data frame, or `"graphicalVAR"` if the data argument is a data list that can be used as input to the graphicalVAR package.
`checkNumeric`	Logical: should the data be checked to be numeric?
`lambda.min.ratio`	Minimal lambda ratio (LASSO tuning parameter)
`nlambda`	Number of LASSO tuning parameters to test
`nCores`	Number of cores to use in estimating networks
`.dots`	A list of arguments used in the estimation function set by a default set or by the `fun` argument.
`...`	A list of arguments used in the estimation function set by a default set or by the `fun` argument.
`.input`	Used internally in the bootnet function. Do not use.
`memorysaver`	Logical. If TRUE attempts to save memory (RAM) by removing some objects from the output. Used by bootnet by default for bootstraps.
`cutoff`	Cutoff score for sum-score to condition on when using `default = "piecewiseIsing"`. This is *experimental*!
`IsingDefault`	Default set for Ising model structure estimation in piecewise Ising estimation. This is *experimental*!
`zeroThreshold`	Used in piecewise Ising estimation. Proportion of edges needed to be exactly 0 in pieces to set edge to zero in final network. This is *experimental*!
`minimalN`	Used in piecewise Ising estimation. Minimal sample sizes needed in piece estimation. This is *experimental*!
`eqThreshold`	Used in SVAR_lavaan estimation (stepup SVAR estimation). This is *experimental*! Maximum difference in criterion to decide if two models are equivalent (and select one at random).
`tempWhitelist`	Used in SVAR_lavaan estimation (step up SVAR estimation). This is *experimental*! Matrix with edges to be whitelisted in the temporal model. The matrix should contain two columns and a row for each edge. The elements should be characters indicating the variable names of each edge (from, to).
`tempBlacklist`	Used in SVAR_lavaan estimation (step up SVAR estimation). This is *experimental*! Matrix with edges to be blacklisted in the temporal model. The matrix should contain two columns and a row for each edge. The elements should be characters indicating the variable names of each edge (from, to).

`contWhitelist`	Used in SVAR_lavaan estimation (step up SVAR estimation). This is *experimental*! Matrix with edges to be whitelisted in the contemporaneous model. The matrix should contain two columns and a row for each edge. The elements should be characters indicating the variable names of each edge (from, to).
`contBlacklist`	Used in SVAR_lavaan estimation (step up SVAR estimation). This is *experimental*! Matrix with edges to be blacklisted in the contemporaneous model. The matrix should contain two columns and a row for each edge. The elements should be characters indicating the variable names of each edge (from, to).
`minimalModInd`	Minimal modification index to consider when adding parameters in SVAR search.
`adjacency`	An 'adjacency' matrix indicating the graph structure (zeroes indicate a missing edge).
`nonPositiveDefinite`	Set to `"stop"` to stop with an error when the input matrix is not positive definite, and to `"continue"` (old behavior) to continue anyway.
`unlock`	Set to `TRUE` to not result in a standard error. This is to prevent using the inner functions seperatly (see details below).
`transform`	Should data be transformed before estimate the network? `"rank"` will call `rank_transformation` and `"quantile""` will call `quantile_transformation`.
`penalty`	Penalty to use in the GGMncv methods.
`min_sum`	The minimum sum score that is artifically possible in the dataset. Defaults to -Inf. Set this only if you know a lower sum score is not possible in the data, for example due to selection bias.
`AND`	Logical, should an AND-rule (both regressions need to be significant) or OR-rule (one of the regressions needs to be significant) be used?

Details

The user can manually specify an estimation method by assigning a custom function to the `'fun'` argument. This function must take data as input and output an estimated network. Fhe functions `bootnet_` correspond to the functions used when using default sets. E.g, `default = "pcor"` sets `fun = bootnet_pcor`. As the `...` leads to any argument to `estimateNetwork` to be passed to the estimator function, the arguments described above in these estimator functions can be used for the appropriate default method. For example, if `default = "pcor"`, the arguments of `fun = bootnet_pcor` can be used in `estimateNetwork`. IMPORTANT NOTE: DO NOT USE THE ESTIMATOR FUNCTIONS (e.g., `fun = bootnet_pcor`) YOURSELF, THEY ARE ONLY INCLUDED HERE TO SHOW WHICH ARGUMENTS YOU CAN USE IN `estimateNetwork`.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Epskamp, S., Borsboom, D., & Fried, E. I. (2016). Estimating psychological networks and their accuracy: a tutorial paper. arXiv preprint, arXiv:1604.08462.

See Also

bootnet

Examples

```
# BFI Extraversion data from psychTools package:
library("psychTools")
data(bfi)
bfiSub <- bfi[,1:25]

# Estimate network:
Network <- estimateNetwork(bfiSub, default = "EBICglasso")

## Not run:
# Some pointers:
print(Network)

# Estimated network:
plot(Network, layout = 'spring')

# Centrality indices:
library("qgraph")
centralityPlot(Network)

# BIC model selection:
Network_BIC <- estimateNetwork(bfiSub, default = "EBICglasso", tuning = 0)

# Ising model:
Network_BIC <- estimateNetwork(bfiSub, default = "IsingFit")

## End(Not run)
```

`genGGM` *Generates a GGM small-world network.*

Description

Simulates a GGM as described by Yin and Li (2011), using the Watts and Strogatz (1998) algorithm for generating the graph structure (see `watts.strogatz.game`).

Usage

```
genGGM(Nvar, p = 0, nei = 1, parRange = c(0.5,1), constant = 1.5, propPositive = 0.5,
  clusters = NULL, graph = c("smallworld","random", "scalefree", "hub", "cluster"))
```

Arguments

`Nvar`	Number of nodes
`p`	Rewiring probability if graph = "smallworld" or "cluster", or connection probability if graph = "random". If cluster, can add multiple p's for each cluster, e.g., "c(.1, .5)"
`nei`	Neighborhood (see `watts.strogatz.game`).
`parRange`	Range of partial correlation coefficients to be originally sampled.
`constant`	A constant as described by Yin and Li (2011).
`propPositive`	Proportion of edges to be set positive.
`clusters`	Number of clusters if graph = "cluster"
`graph`	Type of graph to simulate

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Yin, J., and Li, H. (2011). A sparse conditional gaussian graphical model for analysis of genetical genomics data. The annals of applied statistics, 5(4), 2630.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of ‘small-world’ networks. nature, 393(6684), 440-442.

`ggmGenerator` *Generates a function that simulates data from the Gaussian graphical model (GGM)*

Description

Generates data given a partial correlation network. Data can be made ordinal by using a threshold model with equally spaced thresholds.

Usage

```
ggmGenerator(ordinal = FALSE, nLevels = 4, skewFactor = 1, type =
                    c("uniform", "random"), missing = 0)
```

Arguments

`ordinal`	Logical, should ordinal data be generated?
`nLevels`	Number of levels used in ordinal data.
`skewFactor`	How skewed should ordinal data be? 1 indicates uniform data and higher values increase skewedness.
`type`	Should thresholds for ordinal data be sampled at random or determined uniformly?
`missing`	Proportion of data that should be simulated to be missing.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

`IsingGenerator` *Generates a function that simulates data from the Ising model*

Description

Uses `IsingSampler` to generate the data.

Usage

```
IsingGenerator(...)
```

Arguments

`...`	Arguments passed to `IsingSampler`

Value

A function with as first argument the sample size and as second argument a named list, with element `graph` encoding a weights matrix and element `intercepts` encoding thresholds.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

`multiverse` *Multiverse plot of bootnet results*

Description

This function makes a 'multiverse' plot of bootstrap results. Every row indicates an edge and every column a bootstrap; colors are in line of the edge strength as drawn with `plot.bootnetResult`.

Usage

```
multiverse(x, labels = FALSE)
```

Arguments

`x`	Results from `bootnet`
`labels`	Logical, should labels be printed next to the plot?

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

`netSimulator` *Network Estimation Performance*

Description

This function can be used to run a simulation study on the performance of network estimation by varying sample size or any argument used as input to `estimateNetwork`. The purpose of this function is to provide a way to assess the required sample size given a network structure, as well as to easily perform simulation studies. By default, the function uses `genGGM` to simulate a chain graph or small-world network. See details for more information. The `replicationSimulator` function instead assesses how well a network based on a second independent sample would replicate the network based on the first independent sample.

Usage

```
netSimulator(
      input = genGGM(Nvar = 10),
      nCases = c(50, 100, 250, 500, 1000, 2500),
      nReps = 100,
      nCores = 1,
      default,
      dataGenerator,
      ...,
      moreArgs = list(),
      moreOutput = list())

replicationSimulator(
      input = genGGM(Nvar = 10),
      nCases = c(50, 100, 250, 500, 1000, 2500),
      nReps = 100,
      nCores = 1,
      default,
      dataGenerator,
      ...,
      moreArgs = list())
```

Arguments

`input`	Either a weights matrix, a list containing elements `graph` (encoding the weights matrix) and `intercepts` (encoding the intercepts), or a function generating such objects. By default, `genGGM` is used to generate a Gaussian graphical model. However, it is reccomended to replace this with a prior expected graph structure.
`nCases`	The sample sizes to test for.
`nReps`	Number of repetitions per sampling level.
`nCores`	Number of cores to use. Set to more than 1 to use parallel computing.

`default`	Default set used (see `estimateNetwork`). In most cases, this will set `dataGenerator` to the relevant generator.
`dataGenerator`	A function that generates data. The first argument must be the sample size, the second argument must be the output of `input`. Can often be ignored if `default` is set.
`moreArgs`	A named list of arguments to be used when estimating the network, but which should not be interpreted as different conditions. Use this argument to assign arguments that require vectors.
`moreOutput`	List with functions that take the estimated weights matrix as first argument and the true weights matrix as second argument to produce some output.
`...`	Arguments used by `estimateNetwork` to estimate the network structure. Providing a vector for any argument will simulate under each value. This way, any argument in `estimateNetwork` can be used in a simulation study.

Details

any argument to `estimateNetwork` can be used in a simulation study, with a vector (e.g., `rule = c("AND","OR")`) specifying that both conditions are tested. Adding too many conditions can quickly make any simulation study intractible, so only vary some arguments! The `dataGenerator` argument can be any function that generates data. Currently, only `ggmGenerator` and `IsingGenerator` are implemented in bootnet itself, which generates data given a Gaussian graphical model.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

Examples

```
# 5-node GGM chain graph:
trueNetwork <- genGGM(5)

# Simulate:
Res <- netSimulator(trueNetwork, nReps = 10)

# Results:
Res



# Plot:
plot(Res)

# BFI example:
# Load data:
library("psychTools")
data(bfi)
bfiData <- bfi[,1:25]

# Estimate a network structure, with parameters refitted without LASSO regularization:
library("qgraph")
```

```
Network <- EBICglasso(cor_auto(bfiData), nrow(bfiData), refit = TRUE)

# Simulate 100 repititions in 8 cores under different sampling levels:
Sim1 <- netSimulator(Network,
                     default = "EBICglasso",
                     nCases = c(100,250,500),
                     nReps = 100,
                     nCores = 8)

# Table of results:
Sim1

# Plot results:
plot(Sim1)

# Compare different default set at two sampling levels:
Sim2_EBICglasso <- netSimulator(Network,
                     default = c("EBICglasso"),
                     nCases = c(100,250,500),
                     nReps = 100,
                     nCores = 8)

Sim2_pcor <- netSimulator(Network,
                                default = c("pcor"),
                                nCases = c(100,250,500),
                                nReps = 100,
                                nCores = 8)

Sim2_huge <- netSimulator(Network,
                          default = c("huge"),
                          nCases = c(100,250,500),
                          nReps = 100,
                          nCores = 8)

Sim2 <- rbind(Sim2_EBICglasso, Sim2_pcor, Sim2_huge)

# Print results:
Sim2

# Plot results:
plot(Sim2, xfacet = "default", yvar = "correlation")

# Difference using polychoric or pearson correlations in ordinal data:
Sim3 <- netSimulator(Network,
                     dataGenerator = ggmGenerator(ordinal = TRUE, nLevels = 4),
                     default = "EBICglasso",
                     corMethod = c("cor","cor_auto"),
                     nCases = c(100,250, 500),
                     nReps = 100,
                     nCores = 8)

# Print results:
Sim3
```

```
# Plot results:
plot(Sim3, color = "corMethod")

# Ising model:
trueNetwork <- read.csv('http://sachaepskamp.com/files/weiadj.csv')[,-1]
trueNetwork <- as.matrix(trueNetwork)
Symptoms <- rownames(trueNetwork) <- colnames(trueNetwork)
Thresholds <- read.csv('http://sachaepskamp.com/files/thr.csv')[,-1]

# Create an input list (intercepts now needed)
input <- list(graph=trueNetwork,intercepts=Thresholds)

# Simulate under different sampling levels:
Sim4 <- netSimulator(
  input = input,
  default = "IsingFit",
  nCases = c(250,500,1000),
  nReps = 100,
  nCores = 8)

# Results:
Sim4

# Plot:
plot(Sim4)

# Compare AND and OR rule:
Sim5 <- netSimulator(
  input = input,
  default = "IsingFit",
  nCases = c(250,500,1000),
  rule = c("AND","OR"),
  nReps = 100,
  nCores = 8)

# Print:
Sim5

# Plot:
plot(Sim5, yfacet = "rule")
```

`netSimulator and replicationSimulator methods` *netSimulator S3 methods*

Description

Plot, print and summary methods for `netSimulator` output.

Usage

```
## S3 method for class 'netSimulator'
plot(x, xvar = "factor(nCases)", yvar = c("sensitivity",
                 "specificity", "correlation"), xfacet = "measure",
                 yfacet = ".", color = NULL, ylim = c(0, 1), print =
                 TRUE, xlab = "Number of cases", ylab, outlier.size =
                 0.5, boxplot.lwd = 0.5, style = c("fancy", "basic"),
                 ...)
## S3 method for class 'netSimulator'
print(x, digits = 2, ...)
## S3 method for class 'netSimulator'
summary(object, digits = 2, ...)

## S3 method for class 'replicationSimulator'
plot(x, yvar = c("correlation", "jaccard",
                 "replicatedEdges", "replicatedZeroes"), ...)
## S3 method for class 'replicationSimulator'
print(x, digits = 2, ...)
## S3 method for class 'replicationSimulator'
summary(object, digits = 2, ...)
```

Arguments

`x`	Output of `netSimulator`.
`object`	Output of `netSimulator`.
`xvar`	String indicating the variable to be used on the x-axis.
`yvar`	String vector indicating the variable(s) to be used on the y-axis.
`xfacet`	String indicating the variable to be used on the horizontal facets (or `"."` to omit).
`yfacet`	String indicating the variable to be used on the vertical facets (or `"."` to omit).
`color`	String indicating the variable to be used in coloring boxplots.
`ylim`	Y-axis limits.
`print`	Logical, should the plot be printed? This helps printing the plots to PDF files.
`digits`	Number of digits to be used in print and summary method.
`xlab`	X-axis label
`ylab`	Y-axis label. Defaults to combining the values in `yvar`. Is hidden when `xfacet = "measure"`, as then it is clear what the y-axis represent from the facet labels.
`outlier.size`	Size of the outliers as plotted in boxplots.
`boxplot.lwd`	Line width of the boxplots
`style`	`"fance"` for a style including several aesthethic enhancements, and `"basic"` for a as simple as possible style.
`...`	Arguments sent to `"plot.netSimulator"` from `"plot.replicationSimulator"`

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

`null` *Returns NULL*

Description

This function simply returns `NULL`.

Usage

```
null(...)
```

Arguments

`...`	Anything

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

Examples

```
null("Not NULL")
```

`plot.bootnet` *Plots bootnet results*

Description

This function can be used to plot bootnet results by plotting all bootstrapped statistics as line or by plotting confidence intervals.

Usage

```
## S3 method for class 'bootnet'
plot(x, statistics, plot, graph, CIstyle = c(
                 "quantiles", "SE"), rank = FALSE, sampleColor =
                 "darkred", samplelwd = 1, meanColor = "black", meanlwd
                 = 0.5, bootColor = "black", bootAlpha = 0.01, bootlwd
                 = 0.9, areaAlpha = 0.1, order = c("id", "sample",
                 "mean"), decreasing = TRUE, perNode = FALSE,
                 legendNcol = 2, labels = TRUE, legend = TRUE,
                 subsetRange = c(100, 0), area = !perNode, alpha =
                 0.05, onlyNonZero = FALSE, differenceShowValue,
                 differenceEdgeColor = TRUE, verbose = TRUE, panels =
                 TRUE, split0 = FALSE, prop0 = ifelse(split0, TRUE,
                 FALSE), prop0_cex = 1, prop0_alpha = 0.8,
                 prop0_minAlpha = 0.25, subset, ...)
```

Arguments

`x`	A `bootnet` object
`statistics`	The types of statistics to plot.Defaults to `"edge"` for regular bootstrap and `c("strength", "outStrength", "inStrength")` for node and person drop bootstrap. Use `"all"` to obtain all pairwise statistics tested for regular bootstraps and all node-wise statistics tested for person and node drop bootstarps.
`plot`	Character string indicating what to plot. Can be `"area"` to produce a graph with the area indicating the confidence region, or `"difference"` producing a plot showing significant differences. Other options are `"line"` and `"interval"`, which are currently unstable and not recommended to use.
`graph`	If multipe graphs are estimated, which graph should be plotted? Currently used for `default = "graphicalVAR"` to plot a temporal network using `graph = "temporal"` or a contemporaneous network using `graph = "contemporaneous"`
`CIstyle`	Style of CIs to construct. `"SE"` shows the sample statistic plus and minus two times the standard deviation of bootstraps, and `"quantiles"` the area between the 2.5th and 97.5th quantile. Defaults to `"quantiles"`.
`rank`	Logical, should plots show rank of statistics instead of statistics?
`sampleColor`	Color of the original sample line
`samplelwd`	Line width of the original sample line
`bootColor`	Color of the bootstrap lines
`bootAlpha`	Alpha of the bootstrap lines
`bootlwd`	Line width of the bootstrap lines
`areaAlpha`	Alpha of the area
`order`	String indicating how to order nodes. `"id"` will order nodes based on their name, `"mean"` will order nodes based on the average bootstrapped value of the first statistic in `statistics`, and `"sample"` will order the nodes as done in `"mean"` but orders ties based on their sample value.
`decreasing`	Logical indicating if the ordering is decreasing or increasing.
`perNode`	Logical, should centrality estimates per node be plotted instead of correlation with original parameter. Only used in node and person drop bootstrap.
`legendNcol`	Number of columns in the legend if `perNode = TRUE`.
`labels`	Logical, should labels be plotted?
`legend`	Logical, should the legend be plotted?
`subsetRange`	Range in percentages of the x-axis in node and person drop plots.
`area`	Logical, should the confidence area be plotted?
`alpha`	Signifiance level used in `plot = "difference"`.
`onlyNonZero`	Logical used when `plot = "difference"`, `statistics = "edge"`, should only edges be included that were nonzero in the estimated network structure?
`differenceShowValue`	Logical used when `plot = "difference"`. Should values be shown in the diagonal of the difference plot?

`differenceEdgeColor`	Logical used when `plot = "difference"`, `statistics = "edge"`. Should diagonal blocks be colored according to default edge colors?
`verbose`	Should expected alpha be printed?
`panels`	Logical, should panel titles be printed?
`meanColor`	Color of the bootstrap means.
`meanlwd`	Line width of the bootstrap means
`split0`	Logical. When set to TRUE, the displayed intervals are based on occasions when the parameter was not estimated to be zero, and an extra box is added indicating the number of times a parameter is estimated to be zero.
`prop0`	Logical, should boxes indicating the proportion of times parameters were estimated to be zero be added to the plot?
`prop0_cex`	Size of the boxes indicating number of times a parameter was set to zero.
`prop0_alpha`	Transparency of the boxes indicating number of times a parameter was set to zero.
`prop0_minAlpha`	Minimal transparency of the *lines* of plotted intervals as the proportion of times an edge was not included goes to 0.
`subset`	Vector indicating labels of nodes to include in the plot. This can be used to show, for example, only edges related to one particular node.
`...`	Not used.

Value

A `ggplot2` object.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

`plot.bootnetResult` *Plot method for bootnetResult objects*

Description

Plots the graph using the qgraph package and the qgraph function. Defined as `qgraph::qgraph(x[['graph']],labels=x[[`

Usage

```
## S3 method for class 'bootnetResult'
plot(x, graph, weighted, signed, directed, labels, layout =
                 "spring", parallelEdge = TRUE, cut = 0, theme =
                 "colorblind", bootIncludeOverwrite = TRUE, ...)
```

Arguments

`x`	A `bootnetResult` object
`graph`	Numeric or string indicating which graph to plot. Only needed when multiple graphs are estimated. For example, when using `default = "graphicalVAR"`, `graph = "temporal"` plots the temporal network and `graph = "contemporaneous"` plots the contemporaneous network.
`weighted`	Logical, should the analyzed network be weighted?
`signed`	Logical, should the analyzed network be signed?
`directed`	Logical, is the analyzed network directed? Usually does not have to be set and is detected automatically.
`labels`	Labels of the nodes. Defaults to the column names of the data if missing.
`layout`	Placement of the nodes. See `qgraph`. Always defaults to `"spring"`.
`parallelEdge`	Should edges in directed networks be plotted parallel? See `qgraph`. Defaults to `TRUE` instead of `FALSE` (as in qgraph).
`cut`	Should scaling in width and saturation of edges be split? See `qgraph`. Defaults to `0` to disable cut (qgraph chooses a cutoff with at least 20 nodes).
`theme`	Theme of the edge and node colors. See `qgraph`. Defaults to `"colorblind"` rather than the default used in qgraph (`"classic"`).
`bootIncludeOverwrite`	Logical. If `TRUE`, several plot defaults are overwritten when the input is the result of `bootInclude` (e.g., edge colors are set to black and white).
`...`	Arguments sent to `qgraph`.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

`print.bootnet` *Print method for bootnet and bootnetResult objects*

Description

Prints a short overview of the results of `bootnet`

Usage

```
## S3 method for class 'bootnet'
print(x, ...)
## S3 method for class 'bootnetResult'
print(x, ...)
## S3 method for class 'bootnetResult'
summary(object, ...)
```

Arguments

`x`	A `bootnet` or `bootnetResult` object
`object`	A `bootnetResult` object
`...`	Not used.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

`summary.bootnet` *Summarize bootnet results*

Description

Creates a data frame (wrapped as `tbl_df`) containing summarized results of the bootstraps.

Usage

```
## S3 method for class 'bootnet'
summary(object, graph, statistics = c("edge", "intercept",
                 "strength", "closeness", "betweenness", "distance"),
                 perNode = FALSE, rank = FALSE, tol =
                 sqrt(.Machine$double.eps), ...)
```

Arguments

`object`	A `bootnet` object
`graph`	Numeric or string indicating which graph to summarize. Only needed when multiple graphs are estimated. For example, when using `default = "graphicalVAR"`, `graph = "temporal"` plots the temporal network and `graph = "contemporaneous"` plots the contemporaneous network.
`statistics`	The types of statistics to include in the summary table
`perNode`	Logical, should centrality estimates per node be plotted instead of correlation with original parameter. Only used in node and person drop bootstrap.
`rank`	Logical, should plots show rank of statistics instead of statistics?
`tol`	Tolerance level for assuming an edge is set to zero.
`...`	Not used.

Value

A `tbl_df` (data frame) containing summarized statistics.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

`transformation` *Data transformation functions*

Description

Functions to transform data

Usage

```
quantile_transformation(x)
rank_transformation(x, ties.method = c("average", "first",
    "last", "random", "max", "min"))
```

Arguments

`x`	A dataset
`ties.method`	See `rank`.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

Index