Package 'modnets'

October 1, 2021

Title Modeling Moderated Networks

Version 0.9.0

Maintainer Trevor Swanson <trevorswanson222@gmail.com>

Description Methods for modeling moderator variables in cross-sectional, temporal, and multilevel networks. Includes model selection techniques and a variety of plotting functions. Implements the methods described by Swanson (2020) <https://www.proquest.com/openview/d151ab6b93ad47e3f0d5e59d7b6fd3d3>.

License GPL (>= 3)

URL https://github.com/tswanson222/modnets

BugReports https://github.com/tswanson222/modnets/issues

Imports abind, corpcor, ggplot2, glinternet, glmnet, gridExtra, gtools, igraph, interactionTest, leaps, lme4, lmerTest, Matrix, methods, mvtnorm, parallel, pbapply, plyr, psych, qgraph, reshape2, systemfit

Suggests sn, arm, hierNet, psychTools

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 2.10)

NeedsCompilation no

Author Trevor Swanson [aut, cre]

Repository CRAN

Date/Publication 2021-10-01 08:20:02 UTC

R topics documented:

bfiDat *bfi data*

Description

Composite scores on the Big 5 personality dimensions and gender. Adapted from the `bfi` data in the `psychTools` package.

Usage

```
bfiDat
```

Format

A data frame with six variables representing composite scores forparticipants across the Big 5 personality dimensions as well as gender. `A` is Agreeableness, `C` is conscientiousness, `E` is extraversion, `N` is neuroticism, and `O` is openness to experience. For a given participant, each value represents the mean of 5 self report items from the associated scale. Response values range from `0` to `6`. In total, 7 negatively-worded items were reverse scored before calculating scale composites. `gender` is a binary variable coded such that `0` = `Males`, and `1` = `Females`.

Source

The items are from the IPIP (Goldberg, 1999). The data are from the SAPA project (Revelle, Wilt and Rosenthal, 2010) , collected Spring, 2010 (https://www.sapa-project.org/).

References

Goldberg, L.R. (1999) A broad-bandwidth, public domain, personality inventory measuring the lower-level facets of several five-factor models. In Mervielde, I. and Deary, I. and De Fruyt, F. and Ostendorf, F. (eds) Personality psychology in Europe. 7. Tilburg University Press. Tilburg, The Netherlands.

Revelle, W., Wilt, J., and Rosenthal, A. (2010) Individual Differences in Cognition: New Methods for examining the Personality-Cognition Link In Gruszka, A. and Matthews, G. and Szymura, B. (Eds.) Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, Springer.

bootNet *Bootstrapping network estimation for moderated networks*

Description

Follows closely to the methods of bootstrapping found in the `bootnet` package. An essential goal behind this function is to expand the methods in `bootnet` to encompass moderated networks.

Usage

```
bootNet(
  data,
  m = NULL,
  nboots = 10,
  lags = NULL,
  caseDrop = FALSE,
  rule = "OR",
  ci = 0.95,
  caseMin = 0.05,
  caseMax = 0.75,
  caseN = 10,
  threshold = FALSE,
  fits = NULL,
  type = "g",
  saveMods = TRUE,
  verbose = TRUE,
  fitCoefs = FALSE,
  size = NULL,
  nCores = 1,
  cluster = "mclapply",
  block = FALSE,
  maxiter = 10,
  directedDiag = FALSE,
  beepno = NULL,
  dayno = NULL,
  ...
)
```

Arguments

data	Dataframe or matrix.
m	Numeric or character string. Indicates which variable should be treated as a moderator (if any).
nboots	Number of bootstrapped samples.
lags	Numeric or logical, to indicate whether or not a temporal network is being estimated. Maximum of 1 lag – meaningful values are either 1 or TRUE.
caseDrop	Logical. Determines whether to do a caseDrop bootstrap procedure or not.
rule	Only applies to GGMs (including between-subjects networks) when a threshold is supplied. The "AND" rule will only preserve edges when both corresponding coefficients have p-values below the threshold, while the "OR" rule will preserve an edge so long as one of the two coefficients have a p-value below the supplied threshold.
ci	Numeric, between 0 and 1. The level of the confidence intervals estimated. Defaults at .95

caseMin — Numeric. The minimum proportion of the sample that should be taken when caseDrop = TRUE. Provide a value between 0 and 1. The value indicates the smallest proportion of the total sample size to test in the case-dropping procedure,

caseMax — Numeric. The maximum proportion of the sample that should be taken when caseDrop = TRUE. Provide a value between 0 and 1. The value indicates the largest proportion of the total sample size to test in the case-dropping procedure,

caseN — Numeric. The number of samples to draw at each sample size tested when caseDrop = TRUE.

threshold — Logical or numeric. If TRUE, then a default value of .05 will be set. Indicates whether a threshold should be placed on the bootstrapped samples. A significant choice by the researcher. Only applies when a variable selection procedure is applied, or whether a resample object is used as input.

fits — A list of all fitted models, if available. Not likely to be used.

type — See type argument in fitNetwork function. This is where a variable selection model can be provided. This will fit the same selected model across all iterations of the bootstrapping procedure.

saveMods — Logical. Determines whether or not to return all of the fitted models – that is, all the models fit to each bootstrapped sample. Defaults to TRUE, but if FALSE then models will not be returned which can save memory.

verbose — Logical. Determines whether a progress bar should be shown, as well as whether messages should be shown.

fitCoefs — Logical, refers to the argument in the fitNetwork function. Most likely this should always be FALSE.

size — Numeric. Size of sample to use for bootstrapping. Not recommended.

nCores — If a logical or numeric value is provided, then the bootstrapping procedure will be parallelized across multiple CPUs. If numeric, this will specify the number of cores to use for the procedure. If TRUE, then the parallel::detectCores function of the parallel package will be run to maximize the number of cores available. Defaults to 1, which does not run any parallelization functions.

cluster — Character string to indicate which type of parallelization function to use, if nCores > 1. Options are "mclapply" or "SOCK".

block — Logical or numeric. If specified, then this indicates that lags != 0 or lags != NULL. If numeric, then this indicates that block bootstrapping will be used, and the value specifies the block size. If TRUE then an appropriate block size will be estimated automatically.

maxiter — The maximum number of iterations for the algorithm to go through before stopping. In some circumstances, iterated versions of the model based on subsamples of the data may not be possible to fit. In these cases, maxiter specifies the number of attempts that are made with different versions of the sample before stopping the algorithm.

directedDiag — logical

beepno — Character string or numeric value to indicate which variable (if any) encodes the survey number within a single day. Must be used in conjunction with dayno argument.

dayno	Character string or numeric value to indicate which variable (if any) encodes the survey number within a single day. Must be used in conjunction with beepno argument.
...	Additional arguments.

Details

Can be used to perform bootstrapped network estimation, as well as perform a case-drop bootstrap. Details on these two methods can be found in the help page for the `bootnet::bootnet` function.

The defining feature of `bootNet` that differentiates it from the `resample` function when `sampMethod = "bootstrap"` is that the *same model is fit at every iteration* in `bootNet`. The only time that models may differ across iterations is if a `threshold` is specified. When `threshold = FALSE`, then the saturated model is fit to each bootstrapped sample. Alternatively, bootstrapping can be performed with respect to a specific constrained model. In this case, the constrained model (variable selection model; output of `varSelect` or `resample`) can be supplied to the `type` argument, and thus this function provides a way to estimate the posterior distributions of the nodes based on a constrained model.

In addition to expanding `bootnet` to handle moderated networks, there are also some additional features such as the capacity to perform the block bootstrap for temporal networks via the `block` argument. The block bootstrap is **highly** recommended for resampling temporal networks.

Another feature of this function is that it can be used on outputs from the `resample` function. This can be used as a way to evaluate the iterations of `resample` beyond just using it for variable selection.

Value

A `bootNet` object

Warning

Importantly, if output from the `resample` function is used as input for the `bootNet` function, and the user wishes to use the model selected by the `resample` function as the comparison to the bootstrapped results, you must add the `fit0` argument to this function. Use the fitted object in the `resample` output as the input for the undocumented `fit0` argument for the `bootNet` function.

See Also

`summary.bootNet`, `fitNetwork`, `varSelect`, `resample`, `plotBoot`, `plotNet`, `net`, `netInts`

Examples

```
boot1 <- bootNet(ggmDat, 'M')
summary(boot1)

boot2 <- bootNet(gvarDat, 'M', lags = 1)

mod1 <- varSelect(gvarDat, 'M', lags = 1)
boot3 <- bootNet(gvarDat, 'M', lags = 1, type = mod1, caseDrop = TRUE)
```

```
summary(boot3)
```

---

bootNetDescriptives *Descriptive statistics for* bootNet *objects*

Description

Currently only works for GGMs, including the between-subjects network returned in the mlGVAR output.

Usage

```
## S3 method for class 'bootNet'
summary(object, centrality = TRUE, ...)

cscoef(object, cor = 0.7, ci = 0.95, first = TRUE, verbose = TRUE)
```

Arguments

object	bootNet output
centrality	Logical. Determines whether or not strength centrality and expected influence should be computed for output.
...	Additional arguments.
cor	Numeric value to indicate the correlation stability value to be computed.
ci	Numeric. Confidence interval level for CS coefficient.
first	Logical. Whether or not to count the first instance that a CS statistic dips below the requisite threshold. Often times the value of this will not affect the results. When it does, if first = TRUE then the calculation will be more conservative.
verbose	Logical. Whether to write out the full statement of the CS coefficient in output. Set to FALSE if you want the details about the CS coefficient saved as attributes on the output.

Details

Outputs correlation-stability (CS) coefficients for the case-drop bootstrap.

Value

A table of descriptives for bootNet objects, or correlation-stability coefficients for the case-drop bootstrap.

See Also

bootNet

Examples

```
boot1 <- bootNet(ggmDat, 'M')
summary(boot1)

boot2 <- bootNet(gvarDat, 'M', lags = 1)

mod1 <- varSelect(gvarDat, 'M', lags = 1)
boot3 <- bootNet(gvarDat, 'M', lags = 1, type = mod1, caseDrop = TRUE)
summary(boot3)
```

`CentClust` *Node centrality, clustering coefficients, and shortest path lengths*

Description

Mimics the `qgraph::centrality_auto` and `qgraph::clustcoef_auto` functions. The purpose of amending these functions was to make them compatible with outputs from the `modnets` package. The main use of these functions is as the engines for the `centTable` and `clustTable` functions.

Usage

```
centAuto(x, which.net = "temporal", weighted = TRUE, signed = TRUE)

clustAuto(x, thresholdWS = 0, thresholdON = 0)
```

Arguments

`x`	Output from one of the primary `modnets` functions. Can also supply a list of network models, and the function will be applied to all models in the list.
`which.net`	Only applies to SUR networks, as well as those fit with the `mlGVAR` function. Character string to indicate which type of network to compute centrality values for. Options are `"temporal"` for the temporal network, `"contemporaneous"` for the contemporaneous network, `"PDC"` for the partial directed correlation network, and `"interactions"` for the temporal interaction network.
`weighted`	Logical. If `TRUE` then results are converted to an unweighted network.
`signed`	Logical. Determines whether to ignore the signs of edges or not. Primarily affects the output for expected influence statistics.
`thresholdWS`	Numeric threshold for the WS values.
`thresholdON`	Numeric threshold for the Zhang values.

Details

Returns several node centrality statistics, edge-betweenness centrality, and shortest path lengths. Betweenness and Closeness centrality are computed for all types of networks, as well as edge-betweenness values and shortest path lengths. For GGMs, Strength centrality and Expected Influence are also computed. For SUR networks, InStrength, OutStrength, InExpectedInfluence, and OutExpectedInfluence are computed instead.

The key distinction between these functions and the `qgraph::centrality_auto` and `qgraph::clustcoef_auto` functions is that centrality and clustering values can be computed for the matrix of interactions within a temporal network.

Value

A list containing node centrality statistics, edge-betweenness values, and shortest path lengths.

See Also

`centTable`,`clustTable`,`centPlot`,`clustPlot`,`plotCentrality`,`qgraph::centrality_auto`,`qgraph::clustcoef_au`

Examples

```
x <- fitNetwork(ggmDat, 'M')

clustAuto(x)
centAuto(x, 'interactions')
```

`CentralityAndClustering` *Create table of centrality values or clustering coefficients*

Description

Mimics the output of the `qgraph::centralityTable` and `qgraph::clusteringTable` functions. The purpose of revising these function was to make them compatible with outputs from the `modnets` package.

Usage

```
centTable(
  Wmats,
  scale = TRUE,
  which.net = "temporal",
  labels = NULL,
  relative = FALSE,
  weighted = TRUE,
  signed = TRUE
)

clustTable(Wmats, scale = TRUE, labels = NULL, relative = FALSE, signed = TRUE)
```

Arguments

Wmats	Output from one of the primary modnets functions.
scale	Logical. Determines whether to standardize values within each measure (i.e., convert to z-scores).
which.net	Only applies to SUR networks, as well as those fit with the mlGVAR function. Character string to indicate which type of network to compute centrality values for. Options are "temporal" for the temporal network, "contemporaneous" for the contemporaneous network, "PDC" for the partial directed correlation network, and "interactions" for the temporal interaction network.
labels	Character vector to input the names of the nodes. If left NULL, the function defaults to the node names specified by the model.
relative	Logical. Determines whether to scale values within each measure relative to the largest value within that measure.
weighted	Logical. If TRUE then results are converted to an unweighted network.
signed	Logical. Determines whether to ignore the signs of edges or not. Primarily affects the output for expected influence statistics.

Details

For centTable, centrality values can be computed for the matrix of interactions within a temporal network.

Value

A table containing the names of nodes, measures of node centrality, and their corresponding centrality values or clustering coefficients.

See Also

centAuto, clustAuto, centPlot, clustPlot, plotCentrality, qgraph::centralityTable, qgraph::clusteringTabl

Examples

```
x <- fitNetwork(gvarDat, 'M', lags = TRUE)

clustTable(x)
centTable(x, which.net = 'interactions')
```

CentralityAndClusteringPlots

Plots for node centrality values or clustering coefficients

Description

Mimics the qgraph::centralityPlot and qgraph::clusteringPlot functions. The purpose of revising this function was to make it compatible with outputs from the modnets package.

Usage

```
centPlot(
  Wmats,
  scale = c("z-scores", "raw", "raw0", "relative"),
  which.net = "temporal",
  include = "all",
  labels = NULL,
  orderBy = NULL,
  decreasing = FALSE,
  plot = TRUE,
  verbose = TRUE,
  weighted = TRUE,
  signed = TRUE
)

clustPlot(
  Wmats,
  scale = c("z-scores", "raw", "raw0", "relative"),
  include = "all",
  labels = NULL,
  orderBy = NULL,
  decreasing = FALSE,
  plot = TRUE,
  signed = TRUE,
  verbose = TRUE
)

plotCentrality(
  Wmats,
  which.net = "temporal",
  scale = TRUE,
  labels = NULL,
  plot = TRUE,
  centrality = "all",
  clustering = "Zhang"
)
```

Arguments

Wmats	Output from one of the primary modnets functions.
scale	If "z-scores", then standardized values will be plotted. If "relative", then values will be scaled relative to the largest value on each measure. "raw" can be used to plot raw values.
which.net	Only applies to SUR networks, as well as those fit with the mlGVAR function. Character string to indicate which type of network to compute centrality values for. Options are "temporal" for the temporal network, "contemporaneous" for the contemporaneous network, "PDC" for the partial directed correlation network, and "interactions" for the temporal interaction network.

include	Character vector of which centrality measures to plot. "Betweenness" and "Closeness" are available for all types of network. "Strength" and "ExpectedInfluence" are only available for GGMs. And "InStrength","OutStrength","InExpectedInfluence","OutExpe are only available for SUR networks. Defaults to "all"
labels	Character vector listing the node names. If NULL, then the names specified by the model are used.
orderBy	Character string specifying which measure to order values by.
decreasing	Logical. Only relevant if orderBy is specified. Determines whether values are organized from highest to lowest, or vice versa.
plot	Logical. Determines whether to plot the output or not.
verbose	Logical. Determines whether to return a message about the plot (messages are only shown if values are scaled).
weighted	See centTable or clustTable.
signed	See centTable or clustTable.
centrality	Character vector of centrality measures to plot. Defaults to "all".
clustering	Character vector of clustering measures to plot. Defaults to "Zhang".

Details

The only utility of the plotCentrality function is as an easy way to combine centrality measures and clustering coefficients into a single plot.

Value

A plot of centrality values or clustering coefficients for several measures.

See Also

centTable,clustTable,centAuto,clustAuto,qgraph::centralityPlot,qgraph::clusteringPlot

Examples

```
x <- fitNetwork(ggmDat)

centPlot(x)
clustPlot(x)
plotCentrality(x)
```

compareVAR *Compare two to three* lmerVAR *models*

Description

Affords ANOVAs to compare two or three lmerVAR models. It is necessary to supply at least two different models for comparison, although a third can also be supplied if desired.

Usage

```
compareVAR(m1, m2, m3 = NULL, anova = NULL, type = "tempMods")
```

Arguments

m1	Output from lmerVAR.
m2	Output from another run of lmerVAR. Necessary to supp
m3	Output from a third run of lmerVAR. This is optional.
anova	If NULL, then the results of each nodewise comparison will be displayed. If numeric, then this indicates which nodewise comparison to home in on. anova = 1 will show the full ANOVA results for the first predictor. anova = 2 will show the full ANOVA results for the second predictor, etc.
type	Character string. Either "tempMods" or "contempMods". Determines whether to compare the temporal network outputs or the contemporaneous network outputs with ANOVA.

Details

Performs individual nodewise model comparisons across multiple lmerVAR models.

Value

Table of ANOVA results comparing two or three models.

See Also

lmerVAR

Examples

```
fit1 <- lmerVAR(mlgvarDat, temporal = "fixed", contemp = "orthogonal")
fit2 <- lmerVAR(mlgvarDat, temporal = "orthogonal", contemp = "orthogonal")

compareVAR(fit1, fit2)
```

condPlot *Conditional effects plot*

Description

Creates a plot of the relationships between two variables at different levels of the moderator. Only works for relationships that include an interaction.

Usage

```
condPlot(
  out,
  to,
  from,
  swap = FALSE,
  avg = FALSE,
  compare = NULL,
  hist = FALSE,
  xlab = NULL,
  mods = NULL,
  nsims = 500,
  xn = NULL,
  getCIs = FALSE,
  discrete = FALSE,
  ylab = NULL,
  main = NULL,
  midline = TRUE
)
```

Arguments

out	Output from fitNetwork or resample. Can also provide the fixedNets or betweenNet element of the mlGVAR output.
to	Outcome variable, specified with character string or numeric value.
from	Predictor variable, specified with character string or numeric value.
swap	Logical. Serves to switch the arguments for to and from.
avg	Logical. If TRUE then the average relationship between the two variables is displayed. Only works for GGMs.
compare	Two values can be supplied to indicate levels of the moderator to be compared.
hist	Logical. Determines whether to show a histogram of the data distribution at the bottom of the plot.
xlab	Character string for labeling the x-axis.
mods	This argument will be removed. Model output is automatically detected based on fit argument.

nsims	Number of iterations to simulate the posterior distribution.
xn	Numeric value to indicate how many values of the moderator should be evaluated.
getCIs	Logical. Only applies when avg = TRUE. If getCIs = TRUE, then the confidence intervals for the average difference between the maximum and minimum of the moderator will be returned.
discrete	Logical. Determines whether to treat the moderator as a discrete or continuous variable.
ylab	Character string for labeling the y-axis.
main	Character string for labeling the title of the plot.
midline	Logical. Only applies when discrete = TRUE. Shows a line at the average level of the outcome.

Value

A plot of the conditional effects of one variable on another given different levels of the moderator.

See Also

fitNetwork,resample

Examples

```
fit <- fitNetwork(ggmDat, 'M')
condPlot(fit, to = 'V5', from = 'V4')
condPlot(fit, to = 2, from = 3, avg = TRUE)
```

---

fitNetwork *Fit cross-sectional and idiographic moderated network models*

Description

The main function that ties everything together for both cross-sectional and idiographic (temporal) network models, moderated or otherwise.

Usage

```
fitNetwork(
  data,
  moderators = NULL,
  type = "gaussian",
  lags = NULL,
  seed = NULL,
  folds = 10,
```

```
  gamma = 0.5,
  which.lam = "min",
  rule = "OR",
  threshold = FALSE,
  scale = FALSE,
  std = TRUE,
  center = TRUE,
  covariates = NULL,
  verbose = FALSE,
  exogenous = TRUE,
  mval = NULL,
  residMat = "sigma",
  medges = 1,
  pcor = FALSE,
  maxiter = 100,
  getLL = TRUE,
  saveMods = TRUE,
  binarize = FALSE,
  fitCoefs = FALSE,
  detrend = FALSE,
  beepno = NULL,
  dayno = NULL,
  ...
)
```

Arguments

data	n x k dataframe or matrix.
moderators	Numeric or character vector indicating which variables (if any) to use as moderators.
type	Primarily used to supply a variable selection object, such as those created with varSelect or modSelect, or to indicate that a variable selection method should be employed by setting the value to "varSelect". Currently doesn't support setting the value to "resample", although this will be implemented in the future. Alternatively, this can be used to specify the type of variable for each node. In this case it should be either a single value – "gaussian" or "binomial" – or can be a vector of length k to specify which of those two types apply to each variable. These dictate which family to use for the call to stats::glm. Cannot use binomial models for SUR networks.
lags	Logical or numeric, to indicate whether to fit a SUR model or not. Set to TRUE or 1 for a SUR model fit to temporal data for a single subject.
seed	Only useful if type = "varSelect", and if the varSeed argument is not specified in the ...
folds	Can be used to specify the number of folds in cross-validation when type = "varSelect" and criterion = "CV". Overwritten if nfolds argument is provided.
gamma	Only useful if type = "varSelect" and the criterion is set to "EBIC". This is the hyperparameter for the calculation of EBIC.

`which.lam` Only useful if `criterion = "CV"`, or if a variable selection object based on cross-validation is supplied for `type`. Options include `"min"`, which uses the lambda value that minimizes the objective function, or `"1se"` which uses the lambda value at 1 standard error above the value that minimizes the objective function.

`rule` Only applies to GGMs (including between-subjects networks) when a threshold is supplied. The `"AND"` rule will only preserve edges when both corresponding coefficients have p-values below the threshold, while the `"OR"` rule will preserve an edge so long as one of the two coefficients have a p-value below the supplied threshold.

`threshold` Determines whether to employ a p-value threshold on the model. If `TRUE` then this defaults to .05. Not recommended, as thresholds can be applied post-hoc through the plotting functions, or via the `net` and `netInts` functions. Recommended to leave as `FALSE`.

`scale` Determines whether to standardize all variables or not.

`std` Only applies to SUR networks. Logical. Provides input to the `method` argument of the `systemfit::systemfit` function. If `TRUE`, then the `method` will be `"SUR"`. If `FALSE`, then the `method` will be `"OLS"`. These two methods only differ when constraints are applied. When a saturated model is fit, both methods produce the same results.

`center` Determines whether to mean-center variables or not.

`covariates` Either a numeric value or character string – this could also be a vector – to indicate which variables (if any) should be treated as covariates in the model.

`verbose` Logical. Determines whether to return information about the progress of the model fitting – especially when variable selection is employed – as well as prints the amount of time it takes to fit the model to the console.

`exogenous` Logical. Indicates whether moderator variables should be treated as exogenous or not. If they are exogenous, they will not be modeled as outcomes/nodes in the network. If the number of moderators reaches `k - 1` or `k`, then `exogenous` will automatically be `FALSE`.

`mval` Numeric value to set the moderator variable to when computing model coefficients. Useful to create conditional networks – i.e., those whose values are conditioned on specific values of the moderator. Excellent when the moderator is a categorical variable, or when it's desired to have model estimates at +/- 1 SD around the mean of the moderator. These values must be supplied explicitly. Can only specify a single value for a given model.

`residMat` Character string indicating which type of residual covariance matrix to compute for SUR models. Options include `"res","dfres","sigma"`. `"sigma"` uses the residual covariance matrix as computed by the `systemfits` package. `"res"` and `"dfres"` compute the matrix based directly on the residual values. `"dfres"` is the sample estimator that uses `N - 1` in the denominator, while `"res"` just uses `N`. Input for `SURnet` function.

`medges` Only relevant when `lags = 1` and `exogenous = FALSE`. Determines the linetype of moderated edges (corresponds to the lty argument of `plot()`).

pcor	Logical. Determines whether to operationalize the adjacency matrix as the partial correlation matrix of the data, or to use nodewise estimation. Only relevant for unmoderated networks.
maxiter	See argument of SURfit() function.
getLL	Logical. Determines whether to return log-likelihood statistics with model results. Recommended to keep TRUE.
saveMods	Logical. Determines whether to save the fitobj element of the output, which contains the nodewise models, or the SUR model output of systemfit::systemfit.
binarize	Logical. Determines whether to convert the output to a binary, unweighted network. Only relevant for GGMs.
fitCoefs	Determines whether to use the getFitCIs function on the output. Not recommended to use. The downside is that this will overwrite the fitobj element of the output which contains the actual models. Better to leave this as FALSE, and then use the getFitCIs function on the object separately.
detrend	Logical. Determines whether to remove linear trends from time series variables. Only applies to temporal networks.
beepno	Character string or numeric value to indicate which variable (if any) encodes the survey number within a single day. Must be used in conjunction with dayno argument. Only relevant to temporal data.
dayno	Character string or numeric value to indicate which variable (if any) encodes the survey number within a single day. Must be used in conjunction with beepno argument. Only relevant to temporal data.
...	Additional arguments.

Details

For GGMs, nodewise estimation is utilized to fit models to each node, and then aggregate results into the final network. For temporal networks that represent data for a single subject, SUR estimation based on feasible generalized least squares (FGLS) is used. Also incorporates the variable selection functions to integrate model selection and estimation. Nodewise estimation is used for all GGMs, and SUR estimation is used for temporal networks. See systemfit package for more information on the latter, particularly via the systemfit::systemfit function.

Value

A ggm or SUR network

Examples

```
fit1 <- fitNetwork(ggmDat)


fit2 <- fitNetwork(ggmDat, 'M', type = 'varSelect', criterion = 'BIC')


fit3 <- fitNetwork(gvarDat, 'M', lags = 1)
```

getFitCIs *Provides model coefficients with confidence intervals*

Description

Requires that either `fitobj` or `SURfit` is included in the object from `fitNetwork`. Returns a list of nodewise model coefficients, including confidence intervals computed from the estimated standard errors.

Usage

```
getFitCIs(fit, allNames = NULL, alpha = 0.05)
```

Arguments

`fit`	Output from `fitNetwork`, or either the `fixedNets` or `betweenNet` element of the output from `mlGVAR`
`allNames`	Character vector containing all the predictor names. Do not change, as these are automatically detected.
`alpha`	Type 1 error rate. The complement of the confidence level.

Details

The `select` column in the output indicates whether the variable would be selected given the supplied alpha level.

Value

List of tables containing model coefficients along with confidence intervals

See Also

`fitNetwork`,`plotCoefs`

Examples

```
x <- fitNetwork(ggmDat)
getFitCIs(x)
```

ggmDat *Simulated GGM data*

Description

Data generated from simNet, with five variables that serve as nodes in the GGM and a sixth that serves as a moderator.

Usage

```
ggmDat
```

Format

A 5000 x 6 data frame with five variables as nodes in a GGM and a sixth as a moderator. Attributes contain values for the data generating model. b1 contains the pairwise network, while b2 contains the interaction matrix. intercepts contain the population means of the five node variables. m contains the population mean of the moderator. m1 contains the main effect coefficients for the moderator predicting each of the five nodes.

gvarDat *Simulated temporal network data*

Description

Data generated from simNet, with five variables that serve as nodes in the GVAR model and a sixth that serves as a moderator. The data were generated by setting lag = 1 to specify a single-subject lag-1 model. The data were simulated from a graphical vector autoregressive model (GVAR).

Usage

```
gvarDat
```

Format

A 5000 x 6 data frame with five variables as nodes in a GVAR and a sixth as a moderator. Attributes contain values for the data generating model.

Details

In the case of using simNet with lags = 1, that function essentially serves as a wrapper for mlGVARsim which automatically sets nPerson = 1.

intsPlot *Plot confidence intervals for interaction terms*

Description

Allows one to plot the confidence intervals associated with interaction terms. Provides an easy way to look at whether there are any significant interactions, and if so which interactions are important.

Usage

```
intsPlot(out, y = "all", nsims = 500, alpha = 0.05)
```

Arguments

out	GGM moderated network output from fitNetwork, or output from a moderated between-subjects network fit with mlGVAR (e.g., when bm = TRUE).
y	Character string. The name of the outcome variable for which to create the plot. If y = "all", then all interaction terms associated with all outcomes will be plotted.
nsims	The number of simulations to estimate the posterior distribution of the difference between high and low levels of the confidence interval.
alpha	Alpha level that is used to compute confidence intervals.

Details

The default setting y = "all" shows all interaction terms associated with the model. But the user can also home-in on specific variables to see what interactions might be relevant. When y = "all", the axis labels should be explained. These follow the format of predictor:outcome. The title reflects the name of the moderator variable. For instance, if a variable named "M" moderates the relationship between "X" and "Y", where "X" predicts "Y", the title of the plot will list the variable "M" as the moderator, and the label (shown on the y-axis), will read "X:Y". When y != "all" (that is, a specific value for y is provided), then the title will still reflect the moderator, but the labels will simply show which predictor interacts with that moderator to predict the outcome.

Value

A plot showing the spread of different interactions.

See Also

fitNetwork, plotNet, mlGVAR

Examples

```
fit <- fitNetwork(ggmDat, 'M')
plot(fit, 'ints', y = 'all')
```

lmerVAR *Mixed-effects modeling for the GVAR in multilevel data*

Description

Proper estimation of mixed-effects GVAR models. This is an alternative fitting procedure to that provided by the mlGVAR function. The key differences are that this function can take significantly longer to fit, and it may fail when trying to fit especially large models.

Usage

```
lmerVAR(
  data,
  m = NULL,
  temporal = "default",
  contemp = "default",
  idvar = "ID",
  intvars = NULL,
  center = TRUE,
  scale = TRUE,
  centerWithin = TRUE,
  scaleWithin = FALSE,
  exogenous = TRUE,
  covariates = NULL,
  fix = NULL,
  verbose = TRUE,
  beepno = NULL,
  dayno = NULL,
  deleteMissing = TRUE
)
```

Arguments

data	n x k dataframe or matrix.
m	Character vector or numeric vector indicating the moderator(s), if any. Can also specify "all" to make every variable serve as a moderator, or 0 to indicate that there are no moderators. If the length of m is k -1 or longer, then it will not be possible to have the moderators as exogenous variables. Thus, exogenous will automatically become FALSE.
temporal	Only affects the model for the temporal network and between-subjects network (which is derived from the temporal network). Options are "default","correlated","orthogonal","f "correlated" makes it so that all random-effect terms are correlated, and "orthogonal" makes it so they are not. "fixed" makes it so that there is only a random intercept, but no other random-effect terms related to the individual predictors. "intfixed" essentially mimics "orthogonal", with the exception that no interaction terms have random slopes. "default" will automatically set the value to

	"correlated" if there are 6 or fewer nodes in the network, and "orthogonal" otherwise. The reason for this is that models with correlated random effects take substantially longer to fit than those with orthogonal effects. The "default" option is designed to strike a balance between comprehensiveness and efficiency for the average user. It is recommended to set this value manually in order to produce results according to one's individual specifications.
contemp	Options are "default","correlated","orthogonal". "correlated" makes it so that random-effect terms are correlated, and "orthogonal" makes it so they are not. "default" will automatically set the value to "correlated" if there are 6 or fewer nodes in the network, and "orthogonal" otherwise. The reason for this is that models with correlated random effects take substantially longer to fit than those with orthogonal effects. The "default" option is designed to strike a balance between comprehensiveness and efficiency for the average user. It is recommended to set this value manually in order to produce results according to one's individual specifications.
idvar	Character string to indicate which variable contains the participant identification numbers.
intvars	Character vector to indicate which interaction terms to include in the model. Not necessary, but useful to add significant customization and explicitly state which interactions to include in the model.
center	Logical. Determines whether to mean-center the variables.
scale	Logical. Determines whether to standardize the variables.
centerWithin	Following the application of center and scale, this determines whether to center variables within individual subjects to create subject-centered values.
scaleWithin	Following the application of center and scale, this determines whether to scale variables within individual subjects to create subject-standardized values.
exogenous	Logical. Indicates whether moderator variables should be treated as exogenous or not. If they are exogenous, they will not be modeled as outcomes/nodes in the network. If the number of moderators reaches k - 1 or k, then exogenous will automatically be FALSE.
covariates	See corresponding argument in fitNetwork function. Can supply a numeric value or vector to indicate which variables are covariates, or can supply a list containing the individual covariates separately from the dataset.
fix	Character vector to indicate which variables to only create fixed effects terms for.
verbose	Logical. Determines whether to output progress bars and messages in the console during the fitting process.
beepno	Character string or numeric value to indicate which variable (if any) encodes the survey number within a single day. Must be used in conjunction with dayno argument.
dayno	Character string or numeric value to indicate which variable (if any) encodes the survey number within a single day. Must be used in conjunction with beepno argument.
deleteMissing	Logical. Determines whether to automatically perform listwise deletion if there are any missing values in the dataset.

Details

In the process of adding further documentation. More details to come. The method is referred to as the "two-step multilevel VAR" (Epskamp et al., 2018).

Value

A `lmerVAR` mixed-effects model with corresponding networks.

References

Epskamp, S., Waldorp, L. J., Mottus, R., & Borsboom, B. (2018). The gaussian graphical model in cross-sectional and time-series data. Multivariate Behavioral Research. 53, 453-580.

See Also

`compareVAR`,`mlGVAR`,`mlGVARsim`

Examples

```
# The options were chosen so that the function would take less time to run
x <- lmerVAR(mlgvarDat, 'M', temporal = "fixed", contemp = "orthogonal")
```

LogLikelihood *Log-likelihood functions and Likelihood Ratio Tests for moderated networks*

Description

Computes log-likelihood, AIC, and BIC for a whole network, or for each node in the network. Also compares two or more networks using a likelihood ratio test (LRT).

Usage

```
modLL(
  net0,
  net1 = NULL,
  nodes = FALSE,
  lrt = NULL,
  all = FALSE,
  d = 4,
  alpha = 0.05,
  orderBy = NULL,
  decreasing = TRUE
)

SURll(
```

```
  net0,
  net1 = NULL,
  nodes = FALSE,
  lrt = NULL,
  all = FALSE,
  d = 4,
  alpha = 0.05,
  s = "res",
  orderBy = NULL,
  decreasing = TRUE,
  sysfits = FALSE
)

modTable(
  net0,
  nodes = FALSE,
  orderBy = TRUE,
  d = 4,
  alpha = 0.05,
  decreasing = TRUE,
  names = NULL,
  rmsea = FALSE
)

SURtable(
  net0,
  nodes = FALSE,
  orderBy = TRUE,
  d = 4,
  alpha = 0.05,
  decreasing = TRUE,
  names = NULL,
  rmsea = FALSE,
  s = "res"
)
```

Arguments

net0	Output from one of the main modnets functions. Alternatively, a list of network models can be provided. This list should be named for the easiest interpretation of results.
net1	For modLL and SURll, can be used to supply a second network to compare to net0 via an LRT. Or if lrt = FALSE, then relevant statistics will be returned for both net0 and net1. Importantly, if one network is provided for net0, and another is provided for net1, then the names in the output will reflect these argument names. This can be somewhat confusing at times, so ultimately it is not recommended to use this argument. Instead, try supplying both networks (or more) as a named list to the net0 argument for the most customization.

nodes	Logical. Determines whether to compute omnibus or nodewise statistics and tests. If TRUE, then LL values for nodewise models will be returned, and any LRTs requested will reflect nodewise tests.
lrt	Logical. Determines whether to conduct an LRT or not. If FALSE, then only LL-related statistics will be returned for all models supplied. Only relevant for modLL and SURll
all	Logical. If TRUE, then omnibus LL statistics as well as nodewise statistics are returned for either LL function.
d	Number of decimal places to round outputted statistics to.
alpha	Alpha level for LRTs. Defaults to .05.
orderBy	Can be one of "LL","df","AIC","BIC" to indicate which statistic to order the table by. If using modTable or SURtable, then a value of TRUE will organize the output by the LRT column, which indicates the number of times that a particular model performed better than another based on the pairwise LRTs. Higher values indicate that the model was selected more often in comparison with other models in the list.
decreasing	Logical. Determines whether to organize output from highest to lowest, or vice versa, in accordance with the value of orderBy.
s	Character string indicating which type of residual covariance matrix to compute for SUR models. Options include "res","dfres","sigma". "sigma" uses the residual covariance matrix as computed by the systemfit::systemfit function. "res" and "dfres" compute the matrix based directly on the residual values. "dfres" is the sample estimator that uses N -1 in the denominator, while "res" just uses N.
sysfits	Logical, only relevant to SURll when multiple networks are included in a list. Does not currently work when there are two networks in the list, but does work with 3 or more. Returns the omnibus model statistics based on functions available to output from the systemfit::systemfit function. This allows for some additional statistics such as SSR,detSigma,OLS.R2,McElroy.R2.
names	Character vector containing the names of the models being compared. Only relevant to the modTable and SURtable. Alternatively, models can be named by supplying a named list to the net0 argument.
rmsea	Logical. Relevant to modTable and SURtable. Determines whether to return RMSEA values, as well as tests comparing RMSEA across each pair of models.

Details

Fits LRT to a list of network models to compare them all against each other. Obtain all possible LRTs comparing a list of SUR models. Can include tests comparing RMSEA values. The nodes argument determines whether to perform these computations in an omnibus or nodewise fashion.

One key thing to note is that when using modTable or SURtable, the LRT column indicates the number of times that each network was selected over others with respect to the pairwise LRTs.

Value

A table or list of results depending on which function is used.

See Also

fitNetwork,mlGVAR

Examples

```
data <- na.omit(psychTools::msq[, c('hostile', 'lonely', 'nervous', 'sleepy', 'depressed')])

##### Use modLL() for GGMs
ggm1 <- fitNetwork(data[, -5])
ggm2 <- fitNetwork(data, covariates = 5)
ggm3 <- fitNetwork(data, moderators = 5)

modLL(ggm1)
modLL(ggm2)

modLL(ggm1, ggm2)
modLL(ggm1, ggm2, nodes = TRUE)

modLL(list(ggm1 = ggm1, ggm2 = ggm2))
modLL(list(GGM1 = ggm1, GGM2 = ggm2), nodes = TRUE)

ggms <- list(ggm1, ggm2, ggm3)

modLL(ggms)
modTable(ggms)
modTable(ggms, names = c("GGM1", "GGM2", "GGM3"))

names(ggms) <- c("GGM1", "GGM2", "GGM3")
modTable(ggms)
modLL(ggms)

##### Use SURll() for SUR networks
sur1 <- fitNetwork(data[, -5], lags = TRUE)
sur2 <- fitNetwork(data, covariates = 5, lags = TRUE)
sur3 <- fitNetwork(data, moderators = 5, lags = TRUE)

SURll(sur1)
SURll(sur2)

SURll(sur1, sur2)
SURll(sur1, sur2, nodes = TRUE)
SURll(list(SUR1 = sur1, SUR2 = sur2), nodes = TRUE)

surs <- list(sur1, sur2, sur3)

SURll(surs)
SURtable(surs, names = c('SUR1', "SUR2", "SUR3"))

names(surs) <- c("SUR1", "SUR2", "SUR3")
SURll(surs)
SURtable(surs)
```

mlGVAR *Fit GVAR models with multilevel data*

Description

Fits a graphical vector autoregressive model to data containing multiple time points measured for multiple individuals.

Usage

```
mlGVAR(
  data,
  m = NULL,
  selectFUN = NULL,
  subjectNets = FALSE,
  idvar = "ID",
  exogenous = TRUE,
  center = TRUE,
  scale = TRUE,
  fixedType = "g",
  betweenType = "g",
  centerWithin = TRUE,
  scaleWithin = FALSE,
  rule = "OR",
  threshold = "none",
  verbose = TRUE,
  pcor = FALSE,
  fixedArgs = NULL,
  betweenArgs = NULL,
  bm = FALSE,
  beepno = NULL,
  dayno = NULL,
  deleteMissing = TRUE,
  ...
)
```

Arguments

data	n x k dataframe or matrix
m	Character vector or numeric vector indicating the moderator(s), if any. Can also specify "all" to make every variable serve as a moderator, or 0 to indicate that there are no moderators. If the length of m is k -1 or longer, then it will not be possible to have the moderators as exogenous variables. Thus, exogenous will automatically become FALSE.
selectFUN	Choose a variable selection function. Can specify either "varSelect" or "resample" to use the corresponding functions. If you want to use the resample function

though, then it is recommended to specify `selectFUN` as one of: `"stability"`,`"split"`,`"bootstrap"` in order to identify the specific method. If `selectFUN = "resample"`, then it is recommended to add the `sampMethod` argument to the call to `mlGVAR`.

`subjectNets` If `TRUE`, then subject-specific networks are fit for all subjects and returned in the final output. Otherwise, can specify a single value or a vector of values to represent which subjects to return individual networks for – specifically, the SUR network. One caveat is that variable selection methods are not applied to these subject-specific networks. Further modeling could be done using the output, however.

`idvar` Character string to indicate which variable contains the participant identification numbers.

`exogenous` Logical. Indicates whether moderator variables should be treated as exogenous or not. If they are exogenous, they will not be modeled as outcomes/nodes in the network. If the number of moderators reaches `k - 1` or `k`, then `exogenous` will automatically be `FALSE`.

`center` Logical. Determines whether to mean-center the variables.

`scale` Logical. Determines whether to standardize the variables.

`fixedType` If logical, then any variable selection procedure specified by `selectFUN` will not be applied to the SUR network. Alternatively, a variable selection result, such as the output from either `varSelect` or `modSelect`, can be supplied to choose a specific constrained model in advance.

`betweenType` If logical, then any variable selection procedure specified by `selectFUN` will not be applied to the SUR network. Alternatively, a variable selection result, such as the output from either `varSelect` or `modSelect`, can be supplied to choose a specific constrained model in advance.

`centerWithin` Following the application of `center` and `scale`, this determines whether to center variables within individual subjects to create subject-centered values.

`scaleWithin` Following the application of `center` and `scale`, this determines whether to scale variables within individual subjects to create subject-standardized values.

`rule` Only applies to the between-subject network when a threshold is supplied. The `"AND"` rule will only preserve edges when both corresponding coefficients have p-values below the threshold, while the `"OR"` rule will preserve an edge so long as one of the two coefficients have a p-value below the supplied threshold.

`threshold` Logical or numeric. If `TRUE`, then a default value of .05 will be set. Indicates whether a threshold should be placed on the models at each iteration of the sampling. A significant choice by the researcher.

`verbose` Logical. Determines whether to output progress bars and messages in the console during the fitting process.

`pcor` See corresponding argument in the `fitNetwork` function

`fixedArgs` A named list of arguments for the variable selection function can be provided here, specifically those that are meant to be applied to the SUR network estimation.

`betweenArgs` A named list of arguments for the variable selection function can be provided for the between-subjects network.

bm	Logical. Determines whether the same moderators are applied in the between-subjects network. By default, the value of m only applies to the SUR network. This allows one to decide whether or not to apply those moderators in the between-subject network.
beepno	Character string or numeric value to indicate which variable (if any) encodes the survey number within a single day. Must be used in conjunction with dayno argument.
dayno	Character string or numeric value to indiciate which variable (if any) encodes the survey number within a single day. Must be used in conjunction with beepno argument.
deleteMissing	Logical. Determines whether to automatically perform listwise deletion if there are any missing values in the dataset.
...	Additional arguments.

Details

Uses a pseudo-mixed effects approach, wherein fixed effects are estimated and random effects are approximated. See the work of Epskamp et al. (2018) for more details on how these types of effects are estimated.

Value

mlGVAR objects

References

Epskamp, S., Waldorp, L. J., Mottus, R., & Borsboom, B. (2018). The gaussian graphical model in cross-sectional and time-series data. Multivariate Behavioral Research. 53, 453-580.

See Also

mlGVARsim, lmerVAR, fitNetwork, varSelect, resample, net, netInts, plotNet

Examples

```
fit1 <- mlGVAR(mlgvarDat, 'M')

fit2 <- mlGVAR(mlgvarDat, 'M', bm = TRUE) # Fit the same moderator in the between-subjects network

fit3 <- mlGVAR(mlgvarDat, 'M', selectFUN = 'varSelect')
```

mlgvarDat *Simulated multi-level network data*

Description

Data generated from `mlGVARsim`, with five variables that serve as nodes in the multi-level GVAR model, one moderator variable, and an ID variable that distinguishes between subjects.

Usage

```
mlgvarDat
```

Format

A `50000` x `7` data frame with five variables to serve as nodes in the networks, one variable `"M"` to serve as the time-lagged moderator, and an `ID` variable that labels responses from each of `100` simulated individuals. For each ID number, there are 500 time-ordered responses.

mlGVARsim *Main workhorse for simulating VAR and mlGVAR data*

Description

Affords the generation of simulated data containing multiple timepoint measurements for a number of subjects. Can simulate data with a single moderator as well.

Usage

```
mlGVARsim(
  nTime = 50,
  nPerson = 10,
  nNode = 3,
  m = NULL,
  m2 = 0.25,
  m1 = 0.7,
  m0 = 1,
  lag = 1,
  thetaVar = NULL,
  mu_SD = NULL,
  init_beta_SD = NULL,
  fixedMuSD = 1,
  shrink_fixed = 0.9,
  propPos = 0.5,
  m1SD = 0.1,
  m2SD = 0.1,
```

```
  m1_range = NULL,
  m2_range = NULL,
  shrink_deviation = 0.9,
  getM = FALSE,
  contemporaneous = "wishart",
  GGMsparsity = 0.5,
  mcenter = TRUE,
  skew = FALSE,
  skewErr = FALSE,
  ordinal = FALSE,
  nLevels = 5,
  ordWithin = TRUE,
  minOrd = 3,
  thresholds = NULL,
  mseed = NULL,
  onlyNets = FALSE,
  modType = "none"
)
```

Arguments

nTime	Numeric value. The number of timepoints to simulate for each individual.
nPerson	The number of subjects to create data for. Can set to 1 to just simulate a single graphical VAR network.
nNode	The number of nodes/variables to simulate. Does not include a moderator if one is specified.
m	Logical. If TRUE, then a moderator variable will be simulated. Various options also available for highly specific moderator specification: "fixed","random","mixed1","mixed2","ar
m2	Numeric. If m2 >= 1, then this will determine the number of interaction effects between the moderator and some node in the network. If a value between 0 and 1 is provided, then this determines the probability of any given edge being moderated by the moderator.
m1	Functions similarly to m2, except that this argument refers to the number/probability of main effects of the moderator on any given node.
m0	Only relevant when m = "ar". Determines the autoregressive coefficient in the estimated models. Defaults to .3
lag	Numeric value, supposed to indicate the number of lags to simulate models parameters for. Recommended to leave at 1.
thetaVar	Numeric vector containing the variance associated with each node (excluding the moderator) in the contemporaneous network. If NULL, then it is assumed that the variance for each term is 1.
mu_SD	Numeric vector of length 2. The first value determines the standard deviation of the means associated with the temporal data, and the second value determines the standard deviations of the means associated with the between-subjects network.
init_beta_SD	Similar to mu_SD except that it applies to the coefficient estimates.

`fixedMuSD` Standard deviation of the random values for the means of the fixed effects.

`shrink_fixed` Numeric value to determine the factor by which to shrink sampled beta coefficients for fixed effects. Value between 0 and 1, where higher values are recommended.

`propPos` The proportion of edges with a positive sign.

`m1SD` Standard deviation of the moderator main effect coefficients.

`m2SD` Standard deviation of the moderator interaction effect coefficients.

`m1_range` Numeric vector of length 2. The range of values for moderator main effect coefficients.

`m2_range` Numeric vector of length 2. The range of values for moderator interaction effect coefficients.

`shrink_deviation` Numeric value to determine the factor by which to shrink contemporaneous coefficients. Value between 0 and 1, where higher values are recommended.

`getM` If `TRUE`, only the data for the moderator, the moderator main effects, and interaction effects are returned.

`contemporaneous` Options include `"wishart"`, `"randomGGM"`, `"fixed"`. Determines how the contemporaneous network is sampled. The former two options sample different matrices for each subject, whereas `"fixed"` only samples one matrix and uses it for all subject contemporaneous networks.

`GGMsparsity` Numeric value between 0 and 1. Determines the sparsity of sampled network matrices.

`mcenter` If `TRUE` then the moderator variable is mean-centered.

`skew` If `TRUE` then random values will be generated to represent the skewness of the node distributions. Alternatively, a numeric vector of length `nNode` can be provided to specify the skewness of each variable.

`skewErr` The skewness parameter for the `alpha` argument in the `sn::rmsn` function.

`ordinal` Logical. Determines whether to sample ordinal variables. If a numeric value is provided, then this will automatically be assigned to the `nLevels` argument.

`nLevels` Number of levels for the ordinal variables. Only relevant if `ordinal` is not `FALSE`.

`ordWithin` If `TRUE`, then variables will ordinalized within subjects, rather than across subjects.

`minOrd` The minimum number of unique values allowed for each variable.

`thresholds` List of length `k`, where each element is a numeric vector of length `(nLevels -1)` containing the splitpoints for grouping each variable into ordered categories.

`mseed` Numeric value for the seed to be set when

`onlyNets` If `TRUE` then only the network models are returned, without the data. Could be used to create random models and then simulate data by another method.

`modType` Determines the type of moderation to employ, such as `"none"`, `"full"`, `"partial"`. See `simNet`

Details

Made to simulate data based on pre-specified parameters, possibly for power simulations or other analyses. Output can be used to fit models with either mlGVAR or lmerVAR.

Value

Simulated mlGVAR or VAR data.

See Also

mlGVAR,lmerVAR,simNet,plotNet,net,netInts

Examples

```
set.seed(1)
x <- mlGVARsim(nTime = 50, nPerson = 10, nNode = 3, m = TRUE)
```

---

mnetPowerSim *Power simulator for cross-sectional and idiographic networks*

Description

Samples data based on several parameters, mainly used to see how different sample sizes perform given various parameterizations when simulating from network models, especially moderated networks. See simNet for more details about arguments as well as the warning about simulations that fail.

Usage

```
mnetPowerSim(
  niter = 10,
  N = 100,
  p = 5,
  m = FALSE,
  m1 = 0,
  m2 = 0.1,
  sparsity = 0.5,
  lags = NULL,
  trueNet = NULL,
  threshold = TRUE,
  rule = "OR",
  avg = TRUE,
  maxiter = 100,
  saveFits = TRUE,
  saveData = FALSE,
  intercepts = NULL,
  mbinary = FALSE,
```

```
  select = NULL,
  vargs = list(),
  type = "g",
  gibbs = TRUE,
  ordinal = FALSE,
  mord = FALSE,
  nLevels = 5,
  minOrd = 3,
  div = 1000,
  modType = "none",
  m1_range = NULL,
  m2_range = c(0.1, 0.3),
  time = TRUE,
  skewErr = FALSE,
  nCores = 1,
  cluster = "mclapply",
  fixedPar = NULL,
  V2 = 1,
  ...
)
```

Arguments

niter	Number of iterations/samples to take for each combination of parameters.
N	Numeric value, or vector of sample sizes to generate data with.
p	Numeric value, or vector of network sizes.
m	If a value is provided then a moderated network will be simulated. See simNet for details.
m1	Functions similarly to m2, except that this argument refers to the number/probability of main effects of the moderator on any given node.
m2	Numeric. If m2 >= 1, then this will determine the number of interaction effects between the moderator and some node in the network. If a value between 0 and 1 is provided, then this determines the probability of any given edge being moderated by the moderator.
sparsity	Numeric value between 0 and 1. Determines the sparsity of sampled network matrices.
lags	Determines whether the network should be a temporal network or not. If simulating a temporal network, set to TRUE or 1.
trueNet	The adjacency matrix of the data-generating network model, or a list containing the adjacency matrix as the first element, and the interaction matrix as the second element.
threshold	See corresponding argument in fitNetwork. Automatically set to TRUE if select is not NULL.
rule	Only applies to GGMs (including between-subjects networks) when a threshold is supplied. The "AND" rule will only preserve edges when both corresponding coefficients have p-values below the threshold, while the "OR" rule will preserve

an edge so long as one of the two coefficients have a p-value below the supplied threshold.

`avg` See corresponding argument of `netInts`

`maxiter` If a model fails to be fit, this determines the maximum number of iterations to re-try it before giving up. Will also simulate new datasets at each iteration.

`saveFits` Logical. Determines whether to save the models fit to each dataset at each iteration.

`saveData` Logical. Determines whether to save the datasets generated at each iteration.

`intercepts` A vector of means for sampling node values.

`mbinary` Logical. Determines whether the moderator should be a binary variable.

`select` Identifies a variable selection function – either `varSelect` or `resample` – to use for introducing variable selection at each iteration. The usefulness of this is to mimic a real-world situation, wherein the researcher may be interested in seeing how well datasets of different sizes afford models that approximate a true model after employing iterated variable selection. If `TRUE` then this defaults to `"varSelect"`. Highly recommended to use the `vargs` argument to supply necessary information about the parameters of the variable selection process, such as `sampMethod`, `criterion`, etc.

`vargs` A named list of arguments relevant to the variable selection procedure specified by the `select` argument.

`type` Can supply a variable selection object, such as the output from either `varSelect` or `modSelect`, can be supplied to choose a specific constrained model to fit on all iterations. This is essentially an alternative to `select`, in that `select` performs variable selection at each iteration, whereas this argument defines a constrained model that is applied at every iteration.

`gibbs` If `TRUE`, then Gibbs sampling will be used. Otherwise, data are generated from the `mvtnorm::rmvnorm` function based on the partial correlation matrix that is created.

`ordinal` Logical. Determines whether to generate ordinal values or not.

`mord` Logical. Determines whether the moderator variable should be simulated as ordinal.

`nLevels` Number of levels for the ordinal variables. Only relevant if `ordinal` is not `FALSE`.

`minOrd` The minimum number of unique values allowed for each variable.

`div` A value to use as a sign that the sampler diverged. Can be increased based on expected range of values. If a datapoint is larger than `div`, then the sampler will stop.

`modType` Determines the type of moderation to employ, such as `"none"`, `"full"`, `"partial"`. See `simNet` for details.

`m1_range` Numeric vector of length 2. The range of values for moderator main effect coefficients.

`m2_range` Numeric vector of length 2. The range of values for moderator interaction effect coefficients.

time	If TRUE then the time it takes to simulate the data is printed to screen at the end of the sampling.
skewErr	The skewness parameter for the alpha argument in the sn::rmsn function. Only relevant when gibbs = FALSE and no moderator is specified.
nCores	Numeric value indicating the number of CPU cores to use for the resampling. If TRUE, then the parallel::detectCores function will be used to maximize the number of cores available.
cluster	Character vector indicating which type of parallelization to use, if nCores > 1. Options include "mclapply" and "SOCK".
fixedPar	Numeric. If provided, then this will be set as the coefficient value for all edges in the network. Provides a way to standardize the parameter values while varying the sparsity of the network. If length(fixedPar) == 1, then the same value will be used for all parameters. If length(fixedPar) == 2, then the first value will be for pairwise relationships, and the second value will be for interaction terms.
V2	If V2 = 1 and m2 is between 0 and 1, the number of interaction terms in the model will be determined by multiplying m2 with the number of elements in the interaction matrix and taking the ceiling.
...	Additional arguments.

Details

Evaluates how closely an estimated network is with the true network with regards to metrics such as sensitivity, specificity, and precision, among others. Doesn't calculate values for power, but can be used to serve a similar function as a traditional power analysis based on simulated datasets.

Value

Power simulation results

See Also

summary.mnetPower, plotPower, simNet, mlGVARsim

Examples

```
x <- mnetPowerSim(niter = 10, N = c(100, 200))
summary(x)
plot(x)
```

`modnets` *modnets: Modeling Moderated Networks*

Description

Methods for modeling and plotting various types of moderated networks, including tools for model selection. Model selection tools can be employed for any type of moderated network, and include methods based on the LASSO as well as resampling techniques such as bootstrapping, multi-sample splitting, and stability selection. The primary model types include:

- Cross-section moderated networks
- Temporal (idiographic) moderated networks
- Multi-level moderated networks

Core Package Functions

Model fitting:	`fitNetwork`, `mlGVAR`, `lmerVAR`
Variable selection:	`varSelect`, `resample`, `modSelect`
Model comparison:	`modLL`, `modTable`, `SURll`, `SURtable`
Simulating network data:	`simNet`, `mlGVARsim`, `mnetPowerSim`
Centrality and clustering:	`centTable`, `clustTable`
Adjacency matrices:	`net`, `netInts`
Bootstrapping and stability:	`bootNet`

Plotting functions are available for all model objects and can be accessed through the `plot()` S3 generic.

Details

Package:	modnets
Title:	Modeling Moderated Networks
Version:	0.9.0
Author:	Trevor Swanson
Maintainer:	trevorswanson222@gmail.com
URL:	https://github.com/tswanson222/modnets
BugReports:	https://github.com/tswanson222/modnets/issues
License:	GPL-3
Imports:	abind, corpcor, ggplot2, glinternet, glmnet, gridExtra, gtools, igraph, interactionTest, leaps, lme4, lmerTest, Ma
Suggests:	sn, arm, hierNet, psychTools
Date:	2021-09-26

Author(s)

Trevor Swanson

Maintainer: Trevor Swanson trevorswanson222@gmail.com

References

Swanson, T. J. (2020). *Modeling moderators in psychological networks* (Publication No. 28000912) [Doctoral dissertation, University of Kansas]. ProQuest Dissertations Publishing.

`modSelect` *Select a model based on output from* `resample`

Description

Creates the necessary input for fitNetwork when selecting variables based on the `resample` function. The purpose of making this function available to the user is to that different decisions can be made about how exactly to use the `resample` output to select a model, as sometimes there is more than one option for choosing a final model.

Usage

```
modSelect(
  obj,
  data = NULL,
  fit = FALSE,
  select = "select",
  thresh = NULL,
  ascall = TRUE,
  type = "gaussian",
  ...
)
```

Arguments

`obj`	`resample` output
`data`	The dataframe used to create the `resample` object. Necessary if `ascall = TRUE` or `fit = TRUE`.
`fit`	Logical. Determines whether to fit the selected model to the data or just return the model specifications. Must supply a dataset in the `data` argument as well.
`select`	Character string, referring to which variable of the output should be used as the basis for selecting variables. If the resampling method was either `"bootstrap"` or `"split"`, then setting `select = "select"` will select variables based on the aggregated p-values being below a pre-specified threshold. Setting `select = "select_ci"`, however, will use the adjusted confidence intervals rather than p-values to select variables. Alternatively, if `select = "freq"` then the `thresh`

argument can be used to indicate the minimum selection frequency across iterations. In this case, variables are selected based on how frequently they were selected in the resampling procedure. This also works if `select` is simply set a numeric value (this value will serve as the value for `thresh`).

When the resampling method was `"stability"`, the default option of `select = "select"` chooses variables based on the original threshold provided to the `resample` function, and relies on the simultaneous selection proportion (the `"freq"` column in the `"results"` element). Alternatively, if `select` is a numeric value, or a value for `thresh` is provided, that new frequency selection threshold will determine the choice of variables. Alternatively, one can specify `select = "split1"` or `select = "split2"` to base the threshold on the selection frequency in one of the two splits rather than on the simultaneous selection frequency which is likely to be the most conservative.

For all types of `resample` objects, when `select = "Pvalue"` then `thresh` can be set to a numeric value in order to select variables based on aggregated p-values. For the `"bootstrapping"` and `"split"` methods this allows one to override the original threshold (set as part of `resample`) if desired.

`thresh` Numeric value. If `select = "Pvalue"`, then this value will be the p-value threshold. Otherwise, this value will determine the minimum frequency selection threshold.

`ascall` Logical. Determines whether to return a list with arguments necessary for fitting the model with `do.call` to `fitNetwork`. Only possible if a dataset is supplied.

`type` Should just leave as-is. Automatically taken from the `resample` object.

`...` Additional arguments.

Value

A call ready for `fitNetwork`, a fitted network model, or a list of selected variables for each node along with relevant attributes. Essentially, the output is either the selected model itself or a list of the necessary parameters to fit it.

See Also

`resample`

Examples

```
res1 <- resample(ggmDat, m = 'M', niter = 10)
mods1 <- modSelect(res1)
fit1 <- fitNetwork(ggmDat, morderators = 'M', type = mods1)

res2 <- resample(ggmDat, m = 'M', sampMethod = 'stability')
fit2 <- modSelect(res2, data = ggmDat, fit = TRUE, thresh = .7)
```

net *Get adjacency matrices from fit objects*

Description

net returns the adjacency matrix for any network model fit using functions from the modnets package. netInts returns a matrix of interaction terms associated with a moderated network.

Usage

```
net(
  fit,
  n = "beta",
  threshold = FALSE,
  rule = "OR",
  binary = FALSE,
  nodewise = FALSE,
  d = 14,
  r = NULL
)

netInts(
  fit,
  n = "temporal",
  threshold = FALSE,
  avg = FALSE,
  rule = "none",
  r = NULL,
  empty = TRUE,
  mselect = NULL
)
```

Arguments

fit	A fitted network model. Can be the output from fitNetwork, mlGVAR, lmerVAR, bootNet, resample, simNet, or mlGVARsim.
n	When multiple networks exist for a single object, this allows the user to indicate which adjacency matrix to return. For a GGM, all values of this argument return the same adjacency matrix. For a SUR network, "beta" and "temporal" return the coefficients associated with the temporal network, while "pdc" returns the Partial Directed Correlations, or the standardized temporal network. "contemporaneous" and "pcc" return the standardized contemporaneous network (Partial Contemporaneous Correlations). "kappa" returns the unstandardized residual covariance matrix. All of these terms apply for multilevel networks, but "between" can also return the between-subjects network. If a numeric or logical value is supplied, however, this argument will function as the

	`threshold` argument. A numeric value will set a threshold at the supplied value, while `TRUE` will set a threshold of .05.
`threshold`	A numeric or logical value to set a p-value threshold. `TRUE` will automatically set the threshold at .05.
`rule`	Only applies to GGMs (including between-subjects networks) when a threshold is supplied. The `"AND"` rule will only preserve edges when both corresponding coefficients have p-values below the threshold, while the `"OR"` rule will preserve an edge so long as one of the two coefficients have a p-value below the supplied threshold.
`binary`	Logical. If `TRUE` then the weighted adjacency matrix will be converted into an unweighted adjacency matrix.
`nodewise`	Logical, only applies to GGMs (including between-subjects networks). If `TRUE` then the adjacency matrix will retain all coefficients in their original form. In this case, values in rows represent the coefficients predicting the columns.
`d`	Numeric. Only used for output of `mlGVARsim`, or `simNet` when `lags = 1`. Sets the number of decimal places to round the output to.
`r`	Numeric. Chooses which rows/columns to remove from the output, if desired.
`avg`	Logical. For `netInts`, determines whether to take the average two corresponding interaction terms.
`empty`	Logical. Determines the output of `netInts` when `fit` is not a moderated network. If `TRUE` then an empty list will be returned. If `FALSE` then a matrix of zeros will be returned.
`mselect`	Only used for `netInts` when there is more than one exogenous moderator. Allows the user to indicate which moderator should be used to construct the interaction matrix.

Details

For GGMs when a non-symmetric matrix is requested, columns will represent outcomes and rows will represent predictors. For temporal networks, columns represent predictors and rows represent outcomes.

Can also be used with output from the `resample` and `bootNet` functions.

Value

An adjacency matrix representing a network or a matrix of interaction terms.

See Also

`fitNetwork`, `mlGVAR`, `lmerVAR`, `bootNet`, `resample`, `simNet`, `mlGVARsim`

Examples

```
x <- fitNetwork(ggmDat, 'M')

net(x, threshold = .05)
netInts(x, threshold = TRUE)
```

```
y <- mlGVAR(mlgvarDat, 'M')

net(y, n = 'beta')
net(y, n = 'pcc')
net(y, n = 'between')

netInts(y)
```

`ordinalize` *Convert continuous variables into ordinal variables*

Description

Allows for easy conversion of continuous variables into ordinal variables.

Usage

```
ordinalize(
  data,
  m = NULL,
  nLevels = 5,
  thresholds = NULL,
  mthresh = NULL,
  mord = TRUE,
  minOrd = 3
)
```

Arguments

`data`	An `n` x `k` dataframe or matrix containing only numeric values. Can also be a numeric vector.
`m`	The column number or name of the moderator variable, if applicable. Leave as `NULL` if there is no moderator, and set to `TRUE` if the moderator is the last column in the matrix or dataframe.
`nLevels`	Number of levels for the ordinal variables.
`thresholds`	List of length `k`, where each element is a numeric vector of length `(nLevels -1)` containing the splitpoints for grouping each variable into ordered categories.
`mthresh`	Vector of length `(nLevels -1)` containing thresholds to group values of the moderator into ordered categories.
`mord`	if `FALSE`, then the moderator will not be converted into an ordinal variable (if applicable).
`minOrd`	The minimum number of unique values allowed for each variable.

Details

If a moderator value is specified via the `m` argument, that variable will automatically be relegated to the last column of the resultant dataframe or matrix. It will also be renamed "M"

Value

A dataframe or matrix containing the ordinalized data.

Examples

```
dat <- data.frame(sapply(1:5, function(z) rnorm(100)))
ord_dat <- ordinalize(dat)

# Including a moderator, without converting the moderator into an ordinal variable
ord_dat <- ordinalize(dat, m = 5, mord = FALSE)

colnames(dat)[5] <- 'M'
ord_dat <- ordinalize(dat, m = 'M', mord = FALSE)

# Use thresholds to break each variable into quartiles
thresh <- lapply(dat, function(z) quantile(z, probs = c(.25, .5, .75)))
ord_dat <- ordinalize(dat, thresholds = thresh)
```

---

plot.resample *Plot method for output of resample function*

Description

Allows one to plot results from the `resample` function based on a few different options.

Usage

```
## S3 method for class 'resample'
plot(x, what = "network", ...)
```

Arguments

`x`	Output from the `resample` function.
`what`	Can be one of three options for all `resample` outputs: `what = "network"` will plot the final network model selected from resampling. `what = "bootstrap"` will run `bootNet` based on the final model to create bootstrapped estimates of confidence bands around each edge estimate. `what = "coefs"` will plot the confidence intervals based on the model parameters in the final network. Additionally, if the object was fit with `sampMethod = "stability"`, a stability plot can be created with the `"stability"` option. Otherwise, if `sampMethod = "bootstrap"` or `sampMethod = "split"`, a plot of the empirical distribution function of p-values can be displayed with the `"pvals"` option.
`...`	Additional arguments.

Details

For the what argument, the options correspond with calls to the following functions:

- "network": plotNet
- "bootstrap": plotBoot
- "coefs": plotCoefs
- "stability": plotStability
- "pvals": plotPvals

"bootstrap" and "pvals" only available for bootstrapped and multi-sample split resampling. "stability" only available for stability selection.

Value

Plots various aspects of output from the resample function.

Examples

```
fit1 <- resample(ggmDat, m = 'M', niter = 10)

net(fit1)
netInts(fit1)

plot(fit1)
plot(fit1, what = 'coefs')
plot(fit1, what = 'bootstrap', multi = TRUE)
plot(fit1, what = 'pvals', outcome = 2, predictor = 4)

fit2 <- resample(gvarDat, m = 'M', niter = 10, lags = 1, sampMethod = 'stability')

plot(fit2, what = 'stability', outcome = 3)
```

---

plotBoot *Plot* bootNet *outputs*

Description

Creates various types of plot to visualize bootNet objects.

Usage

```
plotBoot(
  x,
  type = "edges",
  net = "temporal",
  plot = "all",
```

```
  cor = 0.7,
  order = "mean",
  ci = 0.95,
  pairwise = TRUE,
  interactions = TRUE,
  labels = NULL,
  title = NULL,
  cis = "quantile",
  true = NULL,
  errbars = FALSE,
  vline = FALSE,
  threshold = FALSE,
  difference = FALSE,
  color = FALSE,
  text = FALSE,
  textPos = "value",
  multi = NULL,
  directedDiag = FALSE,
  ...
)

## S3 method for class 'bootNet'
plot(
  x,
  type = "edges",
  net = "temporal",
  plot = "all",
  cor = 0.7,
  order = "mean",
  ci = 0.95,
  pairwise = TRUE,
  interactions = TRUE,
  labels = NULL,
  title = NULL,
  cis = "quantile",
  true = NULL,
  errbars = FALSE,
  vline = FALSE,
  threshold = FALSE,
  difference = FALSE,
  color = FALSE,
  text = FALSE,
  textPos = "value",
  multi = NULL,
  directedDiag = FALSE,
  ...
)
```

Arguments

x	Output from bootNet. Also some compatiblity with resample objects (when sampMethod != 'stability').
type	The outcome measure to plot. Options include: "edges","strength","ei","outstrength","instren The "out-" and "in-" options are only available for temporal networks. Moreover, both related options can be used together in temporal networks, by setting either type = c("outstrength","instrength") or type = c("outei","inei").
net	Determines which network to plot coefficients for. Options include: "ggm","temporal","contemporane Only relevant to SUR networks or mlGVAR objects.
plot	Primary use is to set as "none" or FALSE in order to return a table containing the constituents of the plot rather than create the plot itself. The options "all" and "both" each essentially indicate that both pairwise and interaction terms are plotted. Can also specify "pairwise" to only plot the pairwise terms, or "interactions" to only plot the interaction terms.
cor	Numeric value to indicate the correlation stability value to be plotted. Only applies to the case-drop bootstrapping output.
order	Determines how to arrange the predictors displayed in the plot. If TRUE, then defaults to "mean". If FALSE then defaults to "id". The "mean" option will arrange the values by the bootstrapped sample means. The "sample" option will arrange the values according to the statistics from the model fitted to the full sample. The "id" option will keep the variables in the same order that they appear in the dataframe. Not relevant to the case-drop bootstrap.
ci	Numeric value between 0 and 1 to specify the confidence level.
pairwise	Logical. Whether to plot pairwise relationships. Defaults to TRUE. If FALSE, this will override the "all" option of the plot argument.
interactions	Logical. Whether to plot interactions. Defaults to TRUE. If FALSE, this will override the "all" option of the plot argument. Only relevant to moderated networks.
labels	Logical. Determines whether to plot names of the predictors.
title	Character vector the title label.
cis	Either "quantile" or "se". If "quantile", then confidence bands will be computed based on quantiles (specified by the ci argument) of the bootstrapped resamples. If "se", then the confidence bands will be computed based on the standard errors associated with the sample statistics. Thus, the "se" argument will always produce a symmetric confidence band, whereas for "quantile" argument this is not necessary. Not relevant to outputs for the case-drop bootstrap.
true	Defaults to NULL, not relevant for the case-drop bootstrap. Can supply another output from fitNetwork, or an adjacency matrix, to serve as the true network in the plot. If there are interactions in the model, then a fitNetwork object is recommended. Alternatively, this argument can be extremely useful for simulated data – especially anything created with simNet. For whatever outcome (e.g., edges,strength,EI) is plotted, supplying another object to true will plot the values related to the true network, i.e., the data-generating model.

`errbars` Logical. Not relevant to the case-drop bootstrap. If `TRUE`, then error bars are used rather than confidence bands. Can be useful to home in on specific variables and see their confidence interval.

`vline` Logical or numeric. Not relevant to the case-drop bootstrap. If `TRUE`, then a dashed vertical line will be plotted at 0. If numeric, then the line will be plotted at the supplied intercept on the x-axis.

`threshold` Numeric or logical. Not relevant to the case-drop bootstrap. Has a significant effect on the bootstrapped coefficient distributions. If `TRUE`, then the default p-value threshold is set to .05. A numeric value can specify a different threshold. Causes the `bootNet` function to run the object again, only to re-compute the bootstrapped distributions after applying a p-value threshold to the results of each model iteration. If `NULL`, all coefficient estimates are used in estimating the posterior distribution of each parameter.

`difference` Logical. Not relevant to the case-drop bootstrap. If `TRUE`, then a difference plot is provided rather than a coefficient plot. In the difference plot, the diagonal squares reflect the fitted network coefficients for the the original sample. Black boxes indicate that the difference between the two edges, coefficients, or centrality values being compared is significantly different from 0. The significance level will have already been determined by the parameters used to fit the `bootNet` object. Gray boxes indicate the difference is not significantly different from 0.

`color` Logical. Only applies when `difference = TRUE`. Determines whether to add colors that reflect the sign of the sample values. Reflected in the diagonal of the difference plot.

`text` Logical. For difference plots, if `TRUE` then the statistics based on the full sample will be labeled in the diagonal boxes. For coefficient plots, setting this to `TRUE` will plot a label for each variable to reflect the proportion of times that it was selected across all bootstrapped iterations. Only relevant if a threshold was set for the fitted bootstrap models, either specified in the current function or was specified in creating the `bootNet` object. If a numeric value is provided, this will determine the size of the text label. Defaults to 1.2 when `text = TRUE`.

`textPos` Supports the `text` argument for coefficient plots. Indicates the x-axis position of where to plot the coefficient labels. Generally will be numeric, but defaults to `"value"`, which means that the text will be labeled on top each point on the plot.

`multi` Useful when there are interactions in a model. If `TRUE`, the single plot with a facet for both pairwise and interaction terms is split into two separate plots. Allows for a more elegant side-by-side plot, and allows arguments that are restricted for plots of either pairwise or interactions (such as `text`) are plotted. This argument will eventually be expanded to allow one to plot combinations of edge and centrality plots.

`directedDiag` See corresponding argument in the `bootNet`. function.

`...` Additional arguments.

Value

A coefficient plot, difference plot, or correlation-stability plot. When `plot %in% c('none',FALSE)`, the table used to construct the relevant plot will be returned as output instead.

See Also

bootNet,resample

Examples

```
boot1 <- bootNet(ggmDat, caseDrop = TRUE)

plot(boot1)
plotBoot(boot1) # This functions the same as the command above

boot2 <- bootNet(ggmDat)

plot(boot2)
plot(boot2, difference = TRUE)
```

---

plotCoefs *Plot model coefficients with confidence intervals*

Description

Return a plot or dataframe showing the point estimates from each model, along with confidence intervals based on the estimated standard errors.

Usage

```
plotCoefs(
  fit,
  true = FALSE,
  alpha = 0.05,
  plot = TRUE,
  col = "blue",
  flip = TRUE,
  data = NULL,
  select = TRUE,
  size = 1,
  labels = TRUE,
  title = NULL,
  vars = "all"
)
```

Arguments

fit	Output from fitNetwork, bootNet, or resample. Can also be the fixedNets or betweenNet elements of the mlGVAR output.
true	An adjacency matrix containing the true parameter values, if known. This can be used in conjunction with a simulated network, in that the user can supply the true network and plot those values against the estimated values.
alpha	Alpha level that is used to compute confidence intervals.
plot	Logical. If FALSE, a dataframe containing all of the confidence interval data will be returned.
col	Character string. Color of the points associated with the true values.
flip	Logical. If FALSE, the facets will be turned 90 degrees.
data	Supply the original dataset if not already included in the fit object.
select	Relevant to the resample output. Determines whether all variables should be plotted, or only those that were selected according to the resampling or variable selection procedure.
size	Numeric. Size of the point estimates.
labels	If logical, determines whether or not variable labels should be included. If a character vector, can be used to customize variable labels.
title	Custom plot title.
vars	Defaults to "all". Determines which variables should be plotted.

Details

This is differentiated from the output of bootNet and plotBoot in that the confidence intervals are computed directly from model parameters rather than estimated from bootstrapping.

Value

Plot displaying estimated model coefficients and confidence intervals.

See Also

fitNetwork, resample, getFitCIs, plot.resample, plotNet

Examples

```
x <- fitNetwork(ggmDat)
plot(x, which.net = 'coefs')
plotCoefs(x) # This is the same as the above command
```

plotMods *Plot conditional networks at different levels of the moderator*

Description

An easy wrapper for plotting the same network at different levels of a moderator. Using the mval argument of the fitNetwork function, you can create multiple models—conditional networks—wherein the same model is fit at different values of the moderator.

Usage

```
plotMods(
  nets,
  nodewise = FALSE,
  elsize = 2,
  vsize = NULL,
  elabs = TRUE,
  predict = NULL,
  layout = NULL,
  which.net = "temporal",
  ...
)
```

Arguments

nets	List of network models fit with fitNetwork, where mval has been specified.
nodewise	See corresponding argument in plotNet.
elsize	Numeric value to indicate the size of the edge labels.
vsize	Numeric value to indicate the size of the nodes. If NULL, then a default value will be determined based on the number of nodes in the network.
elabs	If TRUE, then edges will be labeled with their numeric values.
predict	See corresponding argument in plotNet.
layout	Can be a character string, corresponding to the options in qgraph::qgraph, or can be a matrix that defines the layout (e.g., based on the qgraph::averageLayout function). Recommended to leave as NULL, so that the layout will be based on the list of networks provided.
which.net	See corresponding argument in plotNet.
...	Additional arguments.

Details

Importantly, this function will fix a common layout across all conditional networks so that the network can be easily compared (visually) at different levels of the moderator.

Value

Returns a plot where multiple conditional networks are plotted side by side.

See Also

fitNetwork

Examples

```
data <- na.omit(psychTools::msq[, c('hostile', 'lonely', 'nervous', 'sleepy', 'depressed')])

fit0 <- fitNetwork(data, moderators = 'depressed', mval = 0)
fit1 <- fitNetwork(data, moderators = 'depressed', mval = 1)
fit2 <- fitNetwork(data, moderators = 'depressed', mval = 2)

fits <- list(fit0, fit1, fit2)
plotMods(fits)
```

---

plotNet *Plot moderated and unmoderated network models*

Description

Core function for plotting various types of network models. Accessible through the `plot()` S3 generic function.

Usage

```
plotNet(
  x,
  which.net = "temporal",
  threshold = FALSE,
  layout = "spring",
  predict = FALSE,
  mnet = FALSE,
  names = TRUE,
  nodewise = FALSE,
  scale = FALSE,
  lag = NULL,
  con = "R2",
  cat = "nCC",
  covNet = FALSE,
  plot = TRUE,
  elabs = FALSE,
  elsize = 1,
  rule = "OR",
  binarize = FALSE,
```

```
  mlty = TRUE,
  mselect = NULL,
  ...
)

## S3 method for class 'ggm'
plot(
  x,
  which.net = "temporal",
  threshold = FALSE,
  layout = "spring",
  predict = FALSE,
  mnet = FALSE,
  names = TRUE,
  nodewise = FALSE,
  scale = FALSE,
  lag = NULL,
  con = "R2",
  cat = "nCC",
  covNet = FALSE,
  plot = TRUE,
  elabs = FALSE,
  elsize = 1,
  rule = "OR",
  binarize = FALSE,
  mlty = TRUE,
  mselect = NULL,
  ...
)

## S3 method for class 'SURnet'
plot(
  x,
  which.net = "temporal",
  threshold = FALSE,
  layout = "spring",
  predict = FALSE,
  mnet = FALSE,
  names = TRUE,
  nodewise = FALSE,
  scale = FALSE,
  lag = NULL,
  con = "R2",
  cat = "nCC",
  covNet = FALSE,
  plot = TRUE,
  elabs = FALSE,
  elsize = 1,
```

```
  rule = "OR",
  binarize = FALSE,
  mlty = TRUE,
  mselect = NULL,
  ...
)

## S3 method for class 'mlGVAR'
plot(
  x,
  which.net = "temporal",
  threshold = FALSE,
  layout = "spring",
  predict = FALSE,
  mnet = FALSE,
  names = TRUE,
  nodewise = FALSE,
  scale = FALSE,
  lag = NULL,
  con = "R2",
  cat = "nCC",
  covNet = FALSE,
  plot = TRUE,
  elabs = FALSE,
  elsize = 1,
  rule = "OR",
  binarize = FALSE,
  mlty = TRUE,
  mselect = NULL,
  ...
)

## S3 method for class 'lmerVAR'
plot(
  x,
  which.net = "temporal",
  threshold = FALSE,
  layout = "spring",
  predict = FALSE,
  mnet = FALSE,
  names = TRUE,
  nodewise = FALSE,
  scale = FALSE,
  lag = NULL,
  con = "R2",
  cat = "nCC",
  covNet = FALSE,
  plot = TRUE,
```

```
  elabs = FALSE,
  elsize = 1,
  rule = "OR",
  binarize = FALSE,
  mlty = TRUE,
  mselect = NULL,
  ...
)

## S3 method for class 'ggmSim'
plot(
  x,
  which.net = "temporal",
  threshold = FALSE,
  layout = "spring",
  predict = FALSE,
  mnet = FALSE,
  names = TRUE,
  nodewise = FALSE,
  scale = FALSE,
  lag = NULL,
  con = "R2",
  cat = "nCC",
  covNet = FALSE,
  plot = TRUE,
  elabs = FALSE,
  elsize = 1,
  rule = "OR",
  binarize = FALSE,
  mlty = TRUE,
  mselect = NULL,
  ...
)

## S3 method for class 'mlGVARsim'
plot(
  x,
  which.net = "temporal",
  threshold = FALSE,
  layout = "spring",
  predict = FALSE,
  mnet = FALSE,
  names = TRUE,
  nodewise = FALSE,
  scale = FALSE,
  lag = NULL,
  con = "R2",
  cat = "nCC",
```

```
  covNet = FALSE,
  plot = TRUE,
  elabs = FALSE,
  elsize = 1,
  rule = "OR",
  binarize = FALSE,
  mlty = TRUE,
  mselect = NULL,
  ...
)

## S3 method for class 'GVARsim'
plot(
  x,
  which.net = "temporal",
  threshold = FALSE,
  layout = "spring",
  predict = FALSE,
  mnet = FALSE,
  names = TRUE,
  nodewise = FALSE,
  scale = FALSE,
  lag = NULL,
  con = "R2",
  cat = "nCC",
  covNet = FALSE,
  plot = TRUE,
  elabs = FALSE,
  elsize = 1,
  rule = "OR",
  binarize = FALSE,
  mlty = TRUE,
  mselect = NULL,
  ...
)
```

Arguments

x	Output from any of the `modnets` model fitting or simulation functions.
which.net	When multiple networks exist for a single object, this allows the user to indicate which network to plot. For a GGM, all values of this argument return the same adjacency matrix. For a SUR network, `"beta"` and `"temporal"` plot the temporal network, while `"pdc"` plots the Partial Directed Correlations, or the standardized temporal network. `"contemporaneous"` and `"pcc"` plot the standardized contemporaneous network (Partial Contemporaneous Correlations). All of these terms apply for multilevel networks, but `"between"` can also plot the between-subjects network. Additionally, the value `"coef"` will plot the model coefficients and confidence intervals, defaulting to the `plotCoefs` function. Moreover, with

GGMs or outputs from mlGVAR with a moderated between-subjects network, the value "ints" will call the intsPlot function. If a numeric or logical value is supplied, however, this argument will function as the threshold argument. A numeric value will set a threshold at the supplied value, while TRUE will set a threshold of .05.

threshold — A numeric or logical value to set a p-value threshold. TRUE will automatically set the threshold at .05.

layout — Character. Corresponds to the layout argument in the qgraph::qgraph function.

predict — If TRUE, then prediction error associated with each node will be plotted as a pie graph around the nodes. For continuous variables, the type of prediction error is determined by the con argument. For categorical variables, the type of error is determined by the cat argument. The desired value of con or can can be supplied directly into the present argument as well. Alternatively, another network model constituted by the same nodes can be supplied in order to plot the difference in prediction error, such as R-squared change.

mnet — Logical. If TRUE, the moderator will be plotted as a square "node" in the network, along with main effects represented as directed edges.

names — If TRUE, then the variable names associated with the model will be plotted as labels on the nodes. If FALSE, then nodes will be labeled with numbers rather than names. Alternatively, a character vector can be provided to serve as custom labels for the nodes.

nodewise — Only applies to GGMs. If TRUE, then nodewise edges will be plotted rather than the undirected averages of corresponding edges.

scale — Logical. Only applies when predict does not equal FALSE. The value of this argument is sent to the predictNet function. This argument will be removed.

lag — This argument will be removed. The function will automatically detect whether the network is based on time-lagged data.

con — Character string indicating which type of prediction error to plot for continuous variables, if predict does not equal FALSE. Options are: "R2", "adjR2", "MSE", "RMSE"

cat — Character string indicating which type of prediction error to plot for categorical variables, if predict does not equal FALSE. Options are: "nCC", "CC", "CCmarg"

covNet — Logical. Only applies when a covariate is modeled. Allows the covariate to be plotted as a separate square "node".

plot — Logical. If FALSE, then a qgraph object will be returned rather than plotted.

elabs — Logical. If TRUE, the values of the edges will be plotted as labels on the edges.

elsize — numeric

rule — Only applies to GGMs (including between-subjects networks) when a threshold is supplied. The "AND" rule will only preserve edges when both corresponding coefficients have p-values below the threshold, while the "OR" rule will preserve an edge so long as one of the two coefficients have a p-value below the supplied threshold.

binarize — Logical. If TRUE, the network will be plotted as an unweighted network. Only applies to GGMs.

mlty	Logical. If FALSE, then moderated edges are displayed as solid lines. If TRUE, then moderated edges are shown as dashed lines.
mselect	If the model contains more than one moderator, input the character string naming which moderator you would like the plot to reflect. Only affects which lines are dashed or solid. Not compatible with the mnet argument.
...	Additional arguments.

Value

Displays a network plot, or returns a qgraph object if plot = FALSE.

See Also

fitNetwork, predictNet, mlGVAR, lmerVAR, simNet, mlGVARsim, plotCoefs, intsPlot, resample

Examples

```
fit1 <- fitNetwork(ggmDat)

plot(fit1)
plotNet(fit1) # This and the command above produce the same result

fit2 <- fitNetwork(gvarDat, moderators = 'M', lags = 1)

plot(fit2, 'pdc') # Partial Directed Correlations
plot(fit2, 'pcc') # Partial Contemporaneous Correlations
```

---

plotNet2 *Plot temporal and contemporaneous networks in the same window*

Description

Designed for easy-to-use plotting with temporal networks. Essentially just a wrapper for running plotNet twice—once for a temporal network, and again for a contemporaneous network—and plotting the two networks in the same window. Good for a quick glance at results from a SUR network. Also compatible with mlGVAR and lmerVAR outputs, although can only plot two networks in the same window. plotNet3 can be used to plot 3 networks.

Usage

```
plotNet2(
  object,
  whichNets = NULL,
  whichTemp = c("temporal", "PDC"),
  titles = c("PDC ", "PCC "),
  ...
)
```

Arguments

object	Output from fitNetwork, specifically with a SUR model.
whichNets	Vector of length 2 indicating which networks to plot. "beta" and "temporal" both refer to the unstandardized temporal network coefficients, while "PDC" refers to the standardized temporal network. "PCC" and "contemporaneous" both refer to the standardized residual covariance matrix (the contemporaneous network). If the object is fitted with mlGVAR or lmerVAR, then "between" is also an option for plotting the between-subjects network.
whichTemp	Which version of the temporal network should be plotted, either "temporal" or "PDC". This argument is ignored if whichNets is not NULL.
titles	Character vector of length 2 where custom names for the two network plots can be supplied.
...	Additional arguments.

Value

Returns two network plots side by side.

See Also

fitNetwork

Examples

```
x <- fitNetwork(gvarDat, lags = TRUE)
plotNet2(x)
```

---

plotNet3 *Plot temporal, contemporaneous, and between-subject networks*

Description

Quick, easy plotting for mlGVAR and lmerVAR output. Allows one to plot three networks in the same window: temporal, contemporaneous, and between-subject.

Usage

```
plotNet3(
  object,
  ...,
  nets = c("temporal", "contemporaneous", "between"),
  titles = TRUE,
  l = 3,
  label = NULL,
  xpos = 0,
  ypos = 0.5
)
```

Arguments

object	Output from mlGVAR or lmerVAR.
...	Additional arguments.
nets	Character vector of length 3 indicating which networks to plot, and in what order. Same options as for which.net in plotNet.
titles	If TRUE, then uses default titles for temporal, contemporaneous, and between-subject networks. If FALSE, then no titles will be used. Can also be a character vector to provide custom plot titles.
l	A numeric value to indicate a type of pane layout.
label	Can include a character string for text annotation.
xpos	Numeric, x-axis value for where the text annotation should go. Between 0 and 1.
ypos	numeric, y-axis value for where the text annotation should go. Between 0 and 1.

Value

Returns 3 network plots.

See Also

mlGVAR, lmerVAR

Examples

```
x <- mlGVAR(mlgvarDat, 'M')
plotNet3(x)
```

---

plotPower *Plot results of power simulations*

Description

Plots the output from the mnetPowerSim function.

Usage

```
plotPower(
  x,
  by = "type",
  yvar = "default",
  yadd = NULL,
  hline = 0.8,
  xlab = "Number of cases",
```

```
  title = NULL,
  ...
)

## S3 method for class 'mnetPower'
plot(
  x,
  by = "type",
  yvar = "default",
  yadd = NULL,
  hline = 0.8,
  xlab = "Number of cases",
  title = NULL,
  ...
)
```

Arguments

x	mnetPowerSim output
by	In development. Currently only supports "type" for creating different facets for Pairwise and Interaction effects. "network" for creating facets based on different networks (e.g., temporal, contemporaneous). "p" for creating facets based on the number of nodes in the network.
yvar	The performance metrics to plot. Options include: "sensitivity","specificity","correlation","p The option "default" automatically sets this to sensitivity, specificity, and correlation.
yadd	Specify additional performance metrics to plot. The final performance metrics that end up being plotted are simply: c(yvar,yadd). Thus, this argument is only useful as a shortcut for keeping the default values of yvar, but adding more metrics to plot.
hline	Numeric value between 0 and 1 for where to plot a horizontal line of interest. Can set to FALSE to remove line.
xlab	Character string for the x-axis label.
title	Character string for the title of the plot.
...	Additional arguments.

Details

The options of what performance metrics to plot include:

- Sensitivity
- Specificity
- Correlation
- MAE (Mean Absolute Error)
- Precision
- Accuracy
- FDR (False Discovery Rate)

Value

Plots the results of a power simulation according to a variety of performance metrics.

See Also

mnetPowerSim

Examples

```
x <- mnetPowerSim(niter = 10, N = c(100, 200))
summary(x)
plot(x)
```

---

plotPvals *Plot the ECDF of p-values from resampling*

Description

Plots the empirical cumulative distribution function of the p-values related to iterated resampling via bootstrapping or multi-sample splitting.

Usage

```
plotPvals(x, outcome = 1, predictor = 1, title = TRUE, alpha = 0.05)
```

Arguments

x	Output from resample, given that sampMethod = "bootstrap" or sampMethod = "split".
outcome	Character string or numeric value (in terms of columns in the dataset) to indicate which outcome to plot the p-value distribution for.
predictor	Character string or numeric value (in terms of columns in the dataset) to indicate which predictor to plot the p-value distribution for.
title	If TRUE, then a default title will be given according to the outcome and predictor that are specified. If FALSE, then no title will be plotted. A custom title may also be supplied by the user.
alpha	The false discovery rate. Defaults to .05

Details

See Meinshausen, Meier, & Buhlmann (2009) for details.

Value

Returns a plot based on the relationship between a particular outcome and predictor.

References

Meinshausen, N., Meier, L., & Buhlmann, P. (2009). P-values for high-dimensional regression. Journal of the American Statistical Association. 104, 1671-1681.

See Also

resample

Examples

```
x <- resample(ggmDat, sampMethod = "bootstrap")
plot(x, what = 'pvals')
plot(x, 'pvals', outcome = 'V2', predictor = 'V1')
```

---

plotStability *Plot stability selection paths for a given outcome*

Description

Creates a plot to show the stability path for a particular variable in terms of how frequently it was chosen in stability selection.

Usage

```
plotStability(
  x,
  outcome = 1,
  s = c("simult", "split1", "split2"),
  thresh = 0.5,
  typeLegend = TRUE
)
```

Arguments

x	Output of resample where sampMethod = "stability".
outcome	Character string or numeric value (in terms of columns in the dataset) to indicate which outcome to plot the stability path for.
s	Character string or numeric value. This indicates which stability path to return a plot for. Either the first sample split "split1", the second sample split "split2", or the path for simultaneous selection "simult", which is the default.
thresh	The selection threshold, which is represented as a horizontal red line on the plot. Defaults to .5
typeLegend	Logical. If FALSE, linetype legend is removed. Only applies if there is a moderator in the model.

Details

See Meinshausen & Buhlmann (2010) for details on stability selection. Cannot be used when the criterion for stability selection was set as cross-validation.

Value

Plot of the stability path associated with a given outcome.

References

Meinshausen, N., & Buhlmann, P. (2010). Stability selection. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 72, 417-423

See Also

resample

Examples

```
x <- resample(ggmDat, sampMethod = "stability")
plot(x, what = "stability")
plot(x, 'stability', outcome = 'V3')
```

---

predictNet *Calculate prediction error from network models*

Description

See the prediction error based on different statistics for either GGMs or SURs. Also can compare and find the change values (such as R-squared change) between two networks of the same size (i.e., with the same nodes).

Usage

```
predictNet(object, data = NULL, all = FALSE, scale = FALSE)
```

Arguments

object	Output from fitNetwork or mlGVAR. If using output from mlGVAR, then one of the two networks must be provided (i.e., either fixedNets or betweenNet).
data	The dataset used to fit the network model, or another network of the same type and size to be compared with the network specified in the first argument. If the prediction error for only one network is desired, and the dataset is included as an element of the relevant object, then this can be left as NULL.
all	if TRUE then returns a list containing the observed outcomes used to fit the models, their predicted values, and the prediction error for each outcome.

`scale` Logical; determines whether or not to standardize the data before computing prediction error. This argument will be removed.

Value

A table showing different measures of prediction error associated with each node of the network. Or, if two networks are provided, a table that shows the difference in prediction error for each node across the two networks. Specifically, this is computed by taking the statistics for `data` and subtracting them from those for `object`.

If `all = TRUE`, then the following output is returned:

Y The observed values of the outcome variables based on the data provided.

preds The predicted values of the outcomes based on the models provided.

errors Table containing prediction error statistics for each node.

See Also

`fitNetwork`

Examples

```
fit1 <- fitNetwork(ggmDat, covariates = 'M')
fit2 <- fitNetwork(ggmDat, moderators = 'M')

predictNet(fit1)
predictNet(fit1, all = TRUE)

predictNet(fit2, fit1) # Find the differences in prediction error across the two models
```

---

resample *Bootstrapping or multi-sample splits for variable selection*

Description

Multiple resampling procedures for selecting variables for a final network model. There are three resampling methods that can be parameterized in a variety of different ways. The ultimate goal is to fit models across iterated resamples with variable selection procedures built in so as to home in on the best predictors to include within a given model. The methods available include: bootstrapped resampling, multi-sample splitting, and stability selection.

Usage

```
resample(
  data,
  m = NULL,
  niter = 10,
  sampMethod = "bootstrap",
  criterion = "AIC",
```

```
  method = "glmnet",
  rule = "OR",
  gamma = 0.5,
  nfolds = 10,
  nlam = 50,
  which.lam = "min",
  threshold = FALSE,
  bonf = FALSE,
  alpha = 0.05,
  exogenous = TRUE,
  split = 0.5,
  center = TRUE,
  scale = FALSE,
  varSeed = NULL,
  seed = NULL,
  verbose = TRUE,
  lags = NULL,
  binary = NULL,
  type = "g",
  saveMods = TRUE,
  saveData = FALSE,
  saveVars = FALSE,
  fitit = TRUE,
  nCores = 1,
  cluster = "mclapply",
  block = FALSE,
  beepno = NULL,
  dayno = NULL,
  ...
)
```

Arguments

data	n x k dataframe. Cannot supply a matrix as input.
m	Character vector or numeric vector indicating the moderator(s), if any. Can also specify "all" to make every variable serve as a moderator, or 0 to indicate that there are no moderators. If the length of m is k -1 or longer, then it will not be possible to have the moderators as exogenous variables. Thus, exogenous will automatically become FALSE.
niter	Number of iterations for the resampling procedure.
sampMethod	Character string indicating which type of procedure to use. "bootstrap" is a standard bootstrapping procedure. "split" is the multi-sample split procedure where the data are split into disjoint training and test sets, the variables to be modeled are selected based on the training set, and then the final model is fit to the test set. "stability" is stability selection, where models are fit to each of two disjoint subsamples of the data, and it is calculated how frequently each variable is selected in each subset, as well how frequently they are simultaneously selected in both subsets at each iteration.

criterion — The criterion for the variable selection procedure. Options include: "cv","aic","bic","ebic","cp","r "CV" refers to cross-validation, the information criteria are "AIC","BIC","EBIC", and "Cp", which refers to Mallow's Cp. "RSS" is the residual sum of squares, "adjR2" is adjusted R-squared, and "Rsq" or "R2" is R-squared. Capitalization is ignored. For methods based on the LASSO, only "CV","AIC","BIC","EBIC" are available. For methods based on subset selection, only "Cp","BIC","RSS","adjR2","R2" are available.

method — Character string to indicate which method to use for variable selection. Options include "lasso" and "glmnet", both of which use the LASSO via the glmnet package (either with glmnet::glmnet or glmnet::cv.glmnet, depending upon the criterion). "subset","backward","forward","seqrep", all call different types of subset selection using the leaps::regsubsets function. Finally "glinternet" is used for applying the hierarchical lasso, and is the only method available for moderated network estimation (either with glinternet::glinternet or glinternet::glinternet.cv, depending upon the criterion). If one or more moderators are specified, then method will automatically default to "glinternet".

rule — Only applies to GGMs (including between-subjects networks) when a threshold is supplied. The "AND" rule will only preserve edges when both corresponding coefficients have p-values below the threshold, while the "OR" rule will preserve an edge so long as one of the two coefficients have a p-value below the supplied threshold.

gamma — Numeric value of the hyperparameter for the "EBIC" criterion. Only relevant if criterion = "EBIC". Recommended to use a value between 0 and .5, where larger values impose a larger penalty on the criterion.

nfolds — Only relevant if criterion = "CV". Determines the number of folds to use in cross-validation.

nlam — if method = "glinternet", determines the number of lambda values to evaluate in the selection path.

which.lam — Character string. Only applies if criterion = "CV". Options include "min", which uses the lambda value that minimizes the objective function, or "1se" which uses the lambda value at 1 standard error above the value that minimizes the objective function.

threshold — Logical or numeric. If TRUE, then a default value of .05 will be set. Indicates whether a threshold should be placed on the models at each iteration of the sampling. A significant choice by the researcher.

bonf — Logical. Determines whether to apply a bonferroni adjustment on the distribution of p-values for each coefficient.

alpha — Type 1 error rate. Defaults to .05.

exogenous — Logical. Indicates whether moderator variables should be treated as exogenous or not. If they are exogenous, they will not be modeled as outcomes/nodes in the network. If the number of moderators reaches k -1 or k, then exogenous will automatically be FALSE.

split — If sampMethod == "split" or sampMethod = "stability" then this is a value between 0 and 1 that indicates the proportion of the sample to be used for the

training set. When sampMethod = "stability" there isn't an important distinction between the labels "training" and "test", although this value will still cause the two samples to be taken of complementary size.

center — Logical. Determines whether to mean-center the variables.

scale — Logical. Determines whether to standardize the variables.

varSeed — Numeric value providing a seed to be set at the beginning of the selection procedure. Recommended for reproducible results. Importantly, this seed will be used for the variable selection models at each iteration of the resampler. Caution this means that while each model is run with a different sample, it will always have the same seed.

seed — Can be a single value, to set a seed before drawing random seeds of length niter to be used across iterations. Alternatively, one can supply a vector of seeds of length niter. It is recommended to use this argument for reproducibility over the varSeed argument.

verbose — Logical. Determines whether information about the modeling progress should be displayed in the console.

lags — Numeric or logical. Can only be 0, 1 or TRUE or FALSE. NULL is interpreted as FALSE. Indicates whether to fit a time-lagged network or a GGM.

binary — Numeric vector indicating which columns of the data contain binary variables.

type — Determines whether to use gaussian models "g" or binomial models "c". Can also just use "gaussian" or "binomial". Moreover, a vector of length k can be provided such that a value is given to every variable. Ultimately this is not necessary, though, as such values are automatically detected.

saveMods — Logical. Indicates whether to save the models fit to the samples at each iteration or not.

saveData — Logical. Determines whether to save the data from each subsample across iterations or not.

saveVars — Logical. Determines whether to save the variable selection models at each iteration.

fitit — Logical. Determines whether to fit the final selected model on the original sample. If FALSE, then this can still be done with fitNetwork and modSelect.

nCores — Numeric value indicating the number of CPU cores to use for the resampling. If TRUE, then the parallel::detectCores function will be used to maximize the number of cores available.

cluster — Character vector indicating which type of parallelization to use, if nCores > 1. Options include "mclapply" and "SOCK".

block — Logical or numeric. If specified, then this indicates that lags != 0 or lags != NULL. If numeric, then this indicates that block bootstrapping will be used, and the value specifies the block size. If TRUE then an appropriate block size will be estimated automatically.

beepno — Character string or numeric value to indicate which variable (if any) encodes the survey number within a single day. Must be used in conjunction with dayno argument.

dayno — Character string or numeric value to indiciate which variable (if any) encodes the survey number within a single day. Must be used in conjunction with beepno argument.

... — Additional arguments.

Details

Sampling methods can be specified via the sampMethod argument.

Bootstrapped resampling Standard bootstrapped resampling, wherein a bootstrapped sample of size n is drawn with replacement at each iteration. Then, a variable selection procedure is applied to the sample, and the selected model is fit to obtain the parameter values. P-values and confidence intervals for the parameter distributions are then estimated.

Multi-sample splitting Involves taking two disjoint samples from the original data – a training sample and a test sample. At each iteration the variable selection procedure is applied to the training sample, and then the resultant model is fit on the test sample. Parameters are then aggregated based on the coefficients in the models fit to the test samples.

Stability selection Stability selection begins the same as multi-sample splitting, in that two disjoint samples are drawn from the data at each iteration. However, the variable selection procedure is then applied to each of the two subsamples at each iteration. The objective is to compute the proportion of times that each predictor was selected in each subsample across iterations, as well as the proportion of times that it was simultaneously selected in both disjoint samples. At the end of the resampling, the final model is selected by setting a frequency threshold between 0 and 1, indicating the minimum proportion of samples that a variable would have to have been selected to be retained in the final model.

For the bootstrapping and multi-sample split methods, p-values are aggregated for each parameter using a method developed by Meinshausen, Meier, & Buhlmann (2009) that employs error control based on the false-discovery rate. The same procedure is employed for creating adjusted confidence intervals.

A key distinguishing feature of the bootstrapping procedure implemented in this function versus the bootNet function is that the latter is designed to estimate the parameter distributions of a single model, whereas the version here is aimed at using the bootstrapped resamples to select a final model. In a practical sense, this boils down to using the bootstrapping method in the resample function to perform variable selection at each iteration of the resampling, rather than taking a single constrained model and applying it equally at all iterations.

Value

resample output

References

Meinshausen, N., Meier, L., & Buhlmann, P. (2009). P-values for high-dimensional regression. Journal of the American Statistical Association. 104, 1671-1681.

Meinshausen, N., & Buhlmann, P. (2010). Stability selection. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 72, 417-423

See Also

plot.resample, modSelect, fitNetwork, bootNet, mlGVAR, plotNet, plotCoefs, plotBoot, plotPvals, plotStabilit

Examples

```
fit1 <- resample(ggmDat, m = 'M', niter = 10)

net(fit1)
netInts(fit1)

plot(fit1)
plot(fit1, what = 'coefs')
plot(fit1, what = 'bootstrap', multi = TRUE)
plot(fit1, what = 'pvals', outcome = 2, predictor = 4)

fit2 <- resample(gvarDat, m = 'M', niter = 10, lags = 1, sampMethod = 'stability')

plot(fit2, what = 'stability', outcome = 3)
```

---

sampleSize *Reports the minimum sample size required to fit a network model*

Description

Indicates the minimum sample size required to fit a moderated or unmoderated network model based on the number of nodes p, number of moderators m, and the number of lags.

Usage

```
sampleSize(p, m = 0, lags = 0, print = TRUE)
```

Arguments

p	Number of nodes
m	Number of moderator variables (defaults to 0)
lags	Number of lags (currently only supports 0 and 1)
print	if FALSE, then the minimum sample size is returned and can be assigned to an object.

Details

When lags = 0, the minimum sample size *N* refers to the number of subjects, whereas when lags = 1 it is assumed that a single subject is being measured at multiple time points, where *N* refers to the number of time points.

Value

Minimum sample size to fit a network model according to the specified parameters.

Examples

```
sampleSize(p = 10)

sampleSize(p = 10, m = 1)

sampleSize(p = 10, m = 1, lags = 1)

minSamp <- sampleSize(p = 10, m = 1, lags = 1, print = FALSE)
```

---

selected *Shows which variables were selected for each node of a network*

Description

Provides a quick representation showing which variables were selected as predictors of each node in a network, both for unmoderated and moderated networks. Especially useful as a way to see which variables were selected in a variable selection procedure, such as through the `varSelect` and `resample` functions.

Usage

```
selected(object, threshold = FALSE, mod = c("temporal", "between"))
```

Arguments

`object`	Output from either `fitNetwork` or `mlGVAR`
`threshold`	Can be a numeric value between `0` and `1`, or defaults to `.05` when set to `TRUE`
`mod`	Only relevant to models fit with `mlGVAR`

Details

The `threshold` argument allows the user to set a threshold for p-values, such that the output only reflects the predictors that are significant at that threshold. This argument can be utilized whether or not a variable selection procedure has been employed.

Value

A table where the columns represent nodes, and the rows show which variables were selected in predicting each node. For moderated networks, the output is a list that separates main effects (`mods`) from interaction effects (`ints`).

See Also

`fitNetwork`,`mlGVAR`

Examples

```
fit1 <- fitNetwork(ggmDat)
selected(fit1)

fit2 <- mlGVAR(mlgvarDat, m = 'M', verbose = FALSE)
selected(fit2, threshold = TRUE, mod = 'temporal') # Can also set to 'between'

fit3 <- fitNetwork(gvarDat, moderators = 'M', type = 'varSelect', lags = 1)
selected(fit3)
```

---

simNet *Simulate network structure and data*

Description

Used for generating moderated and unmoderated adjacency matrices, along with data based on those model structures.

Usage

```
simNet(
  N = 100,
  p = 5,
  m = FALSE,
  m2 = 0.1,
  b1 = NULL,
  b2 = NULL,
  sparsity = 0.5,
  intercepts = NULL,
  nIter = 250,
  msym = FALSE,
  onlyDat = FALSE,
  pbar = TRUE,
  div = 10,
  gibbs = TRUE,
  ordinal = FALSE,
  nLevels = 5,
  mord = FALSE,
  time = TRUE,
  mbinary = FALSE,
  minOrd = 3,
  m1 = NULL,
  m1_range = NULL,
  m2_range = c(0.1, 0.3),
  modType = "none",
```

```
  lags = NULL,
  V = 2,
  skewErr = FALSE,
  onlyNets = FALSE,
  netArgs = NULL,
  nCores = 1,
  cluster = "SOCK",
  getChains = FALSE,
  const = 1.5,
  fixedPar = NULL,
  V2 = 1,
  ...
)
```

Arguments

N	Numeric value. Total number of subjects.
p	Numeric value. Total number of nodes (excluding moderator).
m	If a value is provided, a moderator is generated and named M in the resultant data. If TRUE, then a normal distribution with a mean of 0 will be used to generate the initial value of m, which will serve as the population mean for m throughout the simulation. If a numeric value is provided, then this will serve as the population mean, and all subsequent draws will be taken from a normal distribution with that mean. If m = "binary", then this will simply set the argument mbinary = TRUE. If m = "ordinal", this will set mord = TRUE. To simulate m from a skewed distribution, there are two options: if m = "skewed", then the alpha parameter of the sn::rmsn will automatically be set to 3. Alternatively, a vector of length two can be supplied, containing the element "skewed" as well as the desired value of alpha. Lastly, a function can be provided for m if the user wishes to sample m from another distribution. The requirement is that the function have only one argument, and only returns a single numeric value. The input of the argument should be the location parameter of the desired sampling distribution.
m2	Numeric. If m2 >= 1, then this will determine the number of interaction effects between the moderator and some node in the network. If a value between 0 and 1 is provided, then this determines the probability of any given edge being moderated by the moderator.
b1	Can provide an adjacency matrix to use for generating data.
b2	Can provide an interaction matrix for generated moderated data.
sparsity	Numeric value between 0 and 1. Determines the sparsity of sampled network matrices.
intercepts	A vector of means for sampling node values.
nIter	Number of iterations for generating each instance of a datapoint with the Gibbs sampler.
msym	If TRUE then will force the interaction matrix to be symmetric.
onlyDat	If TRUE then the function only returns the simulated data.
pbar	If TRUE then a progress bar will be shown as samples are generated.

div	A value to use as a sign that the sampler diverged. Can be increased based on expected range of values. If a datapoint is larger than div, then the sampler will stop.
gibbs	If TRUE, then Gibbs sampling will be used. Otherwise, data are generated from the mvtnorm::rmvnorm function based on the partial correlation matrix that is created.
ordinal	Logical. Determines whether to generate ordinal values or not.
nLevels	Number of levels for the ordinal variables. Only relevant if ordinal is not FALSE.
mord	Logical. Determines whether the moderator variable should be simulated as ordinal.
time	If TRUE then the time it takes to simulate the data is printed to screen at the end of the sampling.
mbinary	Logical. Determines whether the moderator should be a binary variable.
minOrd	The minimum number of unique values allowed for each variable.
m1	Functions similarly to m2, except that this argument refers to the number/probability of main effects of the moderator on any given node.
m1_range	Numeric vector of length 2. The range of values for moderator main effect coefficients.
m2_range	Numeric vector of length 2. The range of values for moderator interaction effect coefficients.
modType	Determines the type of moderation to employ, such as "none","full","partial". If modType = "full", then for any interaction terms there will be full moderation, such that all pairwise relationships for moderated paths will be set to zero. If modType = "partial", then pairwise edges for moderated paths will always be nonzero. If modType = "none", no constraints will be applied (e.g., could produce a mix between full and partial moderation).
lags	If TRUE or 1, then arguments are rerouted to the mlGVARsim function to simulate temporal data for a single individual.
V	Numeric, either 1 or 2. Determines whether to randomize the order of simulating node values at each iteration of the Gibbs sampler. If V = 2, then the order is randomized at each iteration. If V = 1, then the sampler moves through the nodes from the first to the last in order at each iteration.
skewErr	The skewness parameter for the alpha argument in the sn::rmsn function. Only relevant when gibbs = FALSE and no moderator is specified.
onlyNets	If TRUE then only the network models are returned, without the data. Could be used to create random models and then simulate data by another method.
netArgs	Only for use by the internal function modnets:::simNet2, which serves as a wrapper for the current function to prevent it from failing.
nCores	Numeric value indicating the number of CPU cores to use for the resampling. If TRUE, then the parallel::detectCores function will be used to maximize the number of cores available.
cluster	Character vector indicating which type of parallelization to use, if nCores > 1. Options include "mclapply" and "SOCK".

getChains	Logical. Determines whether to return the data-generating chains from the Gibbs sampler.
const	Numeric. The constant to be used by the internal modnets:::simPcor function.
fixedPar	Numeric. If provided, then this will be set as the coefficient value for all edges in the network. Provides a way to standardize the parameter values while varying the sparsity of the network. If length(fixedPar) == 1, then the same value will be used for all parameters. If length(fixedPar) == 2, then the first value will be for pairwise relationships, and the second value will be for interaction terms.
V2	If V2 = 1 and m2 is between 0 and 1, the number of interaction terms in the model will be determined by multiplying m2 with the number of elements in the interaction matrix and taking the ceiling.
...	Additional arguments.

Details

If no moderator is specified then data can be generated directly from a partial correlation matrix by setting gibbs = FALSE, which produces fast simulation results. Alternatively, a Gibbs sampler is used to generate data, which is the default option. For moderated networks, Gibbs sampling is the only method available.

Value

Simulated network models as well as data generated from those models. For GGMs, model matrices are always symmetric. For temporal networks (when lags = 1), columns predict rows.

Warning

Importantly, the Gibbs sampler can easily diverge given certain model parameters. Generating network data based on moderator variables can produce data that quickly take on large values due to the presence of multiplicative terms. If the simulation fails, first simply try re-running the function with a different seed; this will often be sufficient to solve the problem when default parameters are specified. Additionally, one can increase the value of div, in case the sampler only diverges slightly or simply produced an anomalous value. This raises the threshold of tolerated values before the sampler stops. If supplying user-generated model matrices (for the b1 and/or b2 arguments) and the function continues to fail, you will likely need to change the parameter values in those matrices, as it may not be possible to simulate data under the given values. If simulating the model matrices inside the function (as is the default) and the function continues to fail, try adjusting the following parameters:

1. Try reducing the value of m2 to specify fewer interactions.
2. Try reducing a range with a smaller maximum for m2_range, to adjust the range of interaction coefficients.
3. Try adjusting the corresponding main effect parameters for the moderator, m1 and m1_range.
4. Try setting modType = "full" to reduce the number of main effect parameters.
5. Try setting a low value(s) for fixedPar, in order to provide parameter values that are known to be lower

An alternative approach could be to use the internal function `simNet2`, which is a wrapper designed to re-run `simNet` when it fails and automatically adjust simulation parameters such as `div` to thoroughly test a given parameterization scheme. This function can be accessed via `modnets:::simNet2`. There is not documentation for this function, so it is recommended to look at the source code if one wishes to use it This wrapper is also used inside the `mnetPowerSim` function.

See Also

`mlGVARsim`,`mnetPowerSim`,`plotNet`,`net`,`netInts`,`plotBoot`,`plotCoefs`

Examples

```
# Generate a moderated GGM along with data
set.seed(1)
x <- simNet(N = 100, p = 3, m = TRUE)

net(x) # Get data-generating adjacency matrix
netInts(x) # Get data-generating interaction matrix

plot(x) # Plot the moderated network that generated the data

# Generate a single-subject GVAR model with data
set.seed(1)
x <- simNet(N = 500, p = 3, m = TRUE, lags = 1)

net(x, n = 'temporal') # Get the data-generating time-lagged adjacency matrix
net(x, n = 'contemporaneous') # Get the data-generating standardized residual covariance matrix

plot(x, which.net = 'beta') # 'beta' is another way of referring to the temporal network
plot(x, which.net = 'pcc') # 'pcc' is another way of referring to the contemporaneous network
```

---

summary.mnetPower *Descriptive statistics for power simulation results*

Description

A quick way to view the results of power simulations conducted with `mnetPowerSim`.

Usage

```
## S3 method for class 'mnetPower'
summary(object, ind = "all", order = NULL, decreasing = FALSE, ...)
```

Arguments

`object`	Output from `mnetPowerSim` function.

ind	Character string or vector to indicate which aspects of the results to view. If "means", then only the means will be returned for all performance indices. "sds" returns the standard deviations, "ses" returns the standard errors, and "medians" returns the medians. These statistics describe the sample distributions according to each combination of input parameters, and with regard to all performance indices. Any combination of these options will return a list with each table as a separate element. "all" returns a list of length 4 with tables for all 4 types of statistic.
order	Character string referring to which output index to organize output by.
decreasing	Logical. Determines whether to organize values from highest to lowest or vice versa according to the value of the order argument.
...	Additional arguments.

Value

Summary table, or list of summary tables.

See Also

mnetPowerSim

Examples

```
x <- mnetPowerSim(niter = 10, N = c(100, 200))
summary(x)
plot(x)
```

SURfit *Fit SUR models with or without constraints*

Description

A wrapper for the systemfit::systemfit function that will construct formulas for all equations based on specified moderators. This function was NOT designed for user-level functionality, but rather exists to be embedded within fitNetwork. The purpose for making it available to the user is for allowing the exact fitted model to be highly customizable.

Usage

```
SURfit(
  data,
  varMods = NULL,
  mod = "min",
  maxiter = 100,
  m = NULL,
```

```
  type = "g",
  center = TRUE,
  scale = FALSE,
  exogenous = TRUE,
  covs = NULL,
  sur = TRUE,
  consec = NULL,
  ...
)
```

Arguments

data	Dataframe or matrix containing idiographic temporal data.
varMods	Output of varSelect or modSelect. The latter must be applied to resample results in order for it to work as input for this argument.
mod	Character string. Only applies if output from varSelect or modSelect is used to constrain the model, and cross-validation "CV" was set as the criterion for model/variable selection. Options include "min", which uses the lambda value that minimizes the objective function, or "1se" which uses the lambda value at 1 standard error above the value that minimizes the objective function.
maxiter	Numeric. The maximum number of iterations to attempt before stopping the function.
m	Character string or numeric value to specify the moderator (if any).
type	Indicates the type of model to use, either "g" for gaussian, or "c" for categorical (i.e., binary, at present). This argument should not be edited by the user, as the appropriate input will automatically be detected.
center	Logical. Determines whether to mean-center the variables.
scale	Logical. Determines whether to standardize the variables.
exogenous	Logical. See fitNetwork function for details.
covs	something
sur	Logical. Provides input to the method argument of the systemfit::systemfit function. If TRUE, then the method will be "SUR". If FALSE, then the method will be "OLS". These two methods only differ when constraints are applied. When a saturated model is fit, both methods produce the same results.
consec	A logical vector that identifies which values to include in accordance with the beepno and dayno arguments in the fitNetwork function.
...	Additional arguments.

Details

See the systemfit package for details on customizing systemfit::systemfit objects. Constraints can be applied via the varMods argument, which is intended to facilitate the output of the varSelect and resample functions. These objects can be further edited to apply constraints not specified by these automated functions. Moreover, there are a variety of additional arguments that can be supplied to the systemfit::systemfit function if desired.

If the variable selection results from resample are intended to be used as input for the varMods argument, then these results must be fed into the modSelect function.

Value

A SUR model, as fit with the systemfit::systemfit function.

See Also

SURnet, fitNetwork, systemfit::systemfit

SURnet *Creates temporal and contemporaneous network of SUR results*

Description

A method for converting outputs from the systemfit::systemfit function into temporal and contemporaneous networks. Intended as an internal function of fitNetwork. Not intended for use by the user. The only purpose of making it available is to allow for extreme customization, and the capacity to convert any systemfit::systemfit output into a pair of network models compatible with the modnets package.

Usage

```
SURnet(
  fit,
  dat,
  s = "sigma",
  m = NULL,
  threshold = FALSE,
  mval = NULL,
  medges = 1,
  pcor = "none"
)
```

Arguments

fit	Output from SURfit
dat	A list containing elements "Y" and "X" elements, to reflect the outcome and predictor matrices. These are lagged data matrices, and can be automatically created through the internal modnets:::lagMat function. These transformed matrices must be supplied in conjunction with the SURfit output in order to construct network models.
s	Character string indicating which type of residual covariance matrix to compute for SUR models. Options include "res","dfres","sigma". "sigma" uses the residual covariance matrix as computed by the systemfit::systemfit function. "res" and "dfres" compute the matrix based directly on the residual values. "dfres" is the sample estimator that uses N -1 in the denominator, while "res" just uses N.
m	Character string or numeric value to specify the moderator (if any).

threshold	See corresponding argument of fitNetwork
mval	Numeric. See corresponding argument of fitNetwork
medges	Numeric. See corresponding argument of fitNetwork
pcor	See corresponding argument of fitNetwork

Value

Temporal and contemporaneous networks

See Also

SURfit, fitNetwork, systemfit::systemfit

---

varSelect *Variable selection for moderated networks*

Description

Perform variable selection via the LASSO, best subsets selection, forward selection, backward selection, or sequential replacement on unmoderated networks. Or, perform variable selection via the hierarchical LASSO for moderated networks. Can be used for both GGMs and SUR networks.

Usage

```
varSelect(
  data,
  m = NULL,
  criterion = "AIC",
  method = "glmnet",
  lags = NULL,
  exogenous = TRUE,
  type = "g",
  center = TRUE,
  scale = FALSE,
  gamma = 0.5,
  nfolds = 10,
  varSeed = NULL,
  useSE = TRUE,
  nlam = NULL,
  covs = NULL,
  verbose = TRUE,
  beepno = NULL,
  dayno = NULL
)
```

Arguments

data	n x k dataframe or matrix.
m	Character vector or numeric vector indicating the moderator(s), if any. Can also specify "all" to make every variable serve as a moderator, or 0 to indicate that there are no moderators. If the length of m is k -1 or longer, then it will not be possible to have the moderators as exogenous variables. Thus, exogenous will automatically become FALSE.
criterion	The criterion for the variable selection procedure. Options include: "cv","aic","bic","ebic","cp","r "CV" refers to cross-validation, the information criteria are "AIC","BIC","EBIC", and "Cp", which refers to Mallow's Cp. "RSS" is the residual sum of squares, "adjR2" is adjusted R-squared, and "Rsq" or "R2" is R-squared. Capitalization is ignored. For methods based on the LASSO, only "CV","AIC","BIC","EBIC" are available. For methods based on subset selection, only "Cp","BIC","RSS","adjR2","R2" are available.
method	Character string to indicate which method to use for variable selection. Options include "lasso" and "glmnet", both of which use the LASSO via the glmnet package (either with glmnet::glmnet or glmnet::cv.glmnet, depending upon the criterion). "subset","backward","forward","seqrep", all call different types of subset selection using the leaps::regsubsets function. Finally "glinternet" is used for applying the hierarchical lasso, and is the only method available for moderated network estimation (either with glinternet::glinternet or glinternet::glinternet.cv, depending upon the criterion). If one or more moderators are specified, then method will automatically default to "glinternet".
lags	Numeric or logical. Can only be 0, 1 or TRUE or FALSE. NULL is interpreted as FALSE. Indicates whether to fit a time-lagged network or a GGM.
exogenous	Logical. Indicates whether moderator variables should be treated as exogenous or not. If they are exogenous, they will not be modeled as outcomes/nodes in the network. If the number of moderators reaches k -1 or k, then exogenous will automatically be FALSE.
type	Determines whether to use gaussian models "g" or binomial models "c". Can also just use "gaussian" or "binomial". Moreover, a vector of length k can be provided such that a value is given to every variable. Ultimately this is not necessary, though, as such values are automatically detected.
center	Logical. Determines whether to mean-center the variables.
scale	Logical. Determines whether to standardize the variables.
gamma	Numeric value of the hyperparameter for the "EBIC" criterion. Only relevant if criterion = "EBIC". Recommended to use a value between 0 and .5, where larger values impose a larger penalty on the criterion.
nfolds	Only relevant if criterion = "CV". Determines the number of folds to use in cross-validation.
varSeed	Numeric value providing a seed to be set at the beginning of the selection procedure. Recommended for reproducible results.
useSE	Logical. Only relevant if method = "glinternet" and criterion = "CV". Indicates whether to use the standard error of the estimates across folds, if TRUE, or to use the standard deviation, if FALSE.

nlam	if method = "glinternet", determines the number of lambda values to evaluate in the selection path.
covs	Numeric or character string indicating a variable to be used as a covariate. Currently not working properly.
verbose	Logical. Determines whether to provide output to the console about the status of the procedure.
beepno	Character string or numeric value to indicate which variable (if any) encodes the survey number within a single day. Must be used in conjunction with dayno argument.
dayno	Character string or numeric value to indicate which variable (if any) encodes the survey number within a single day. Must be used in conjunction with beepno argument.

Details

The primary value of the output is to be used as input when fitting the selected model with the fitNetwork function. Specifically, the output of varSelect can be assigned to the type argument of fitNetwork in order to fit the constrained models that were selected across nodes.

For moderated networks, the only variable selection approach available is through the glinternet package, which implements the hierarchical LASSO. The criterion for model selection dictates which function from the package is used, where information criteria use the glinternet::glinternet function to compute models, and cross-validation calls the glinternet::glinternet.cv function.

Value

List of all models, with the selected variables for each along with model coefficients and the variable selection models themselves. Primarily for use as input to the type argument of the fitNetwork function.

See Also

resample, fitNetwork, bootNet, mlGVAR, glinternet::glinternet, glinternet::glinternet.cv, glmnet::glmnet,

Examples

```
vars1 <- varSelect(ggmDat, criterion = 'BIC', method = 'subset')
fit1 <- fitNetwork(ggmDat, type = vars1)

vars2 <- varSelect(ggmDat, criterion = 'CV', method = 'glmnet')
fit2 <- fitNetwork(ggmDat, type = vars2, which.lam = 'min')

# Add a moderator
vars3 <- varSelect(ggmDat, m = 'M', criterion = 'EBIC', gamma = .5)
fit3 <- fitNetwork(ggmDat, moderators = 'M', type = vars3)
```

Index