Package: EstimateGroupNetwork (via r-universe)

September 11, 2024

Type Package

Title Perform the Joint Graphical Lasso and Selects Tuning Parameters

Version 0.3.1

Author Giulio Costantini, Nils Kappelmann, Sacha Epskamp

Maintainer Giulio Costantini <giulio.costantini@unimib.it>

Description Can be used to simultaneously estimate networks (Gaussian Graphical Models) in data from different groups or classes via Joint Graphical Lasso. Tuning parameters are selected via information criteria (AIC / BIC / extended BIC) or cross validation.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports parallel, igraph, qgraph, dplyr, ggplot2, stats

Suggests mvtnorm, JGL, psych

NeedsCompilation no

Repository CRAN

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

Contents

BootTable 2
covNoBessel 3
EstimateGroupNetwork 3
GroupBootPlot 11
GroupNetworkBoot 12

Index **15**

BootTable *Summary table for a bootstrapped Joint Graphical Lasso model*

Description

Create a table of bootstrapped means and confidence intervals for all edges of a bootstrapped Joint Graphical Lasso model obtained through GroupNetworkBoot.

Usage

```
BootTable(BootOut)
```

Arguments

`BootOut`	The output from GroupNetworkBoot

Details

Summary table of the output of GroupNetworkBoot

Value

`Var1`	Nodes included in each edge
`Var2`	Nodes included in each edge
`edges`	Edge identifier
`sample`	sample value of each edge
`boot.mean`	mean of boostrapped values of each edge
`ci.lb`	lower bound of the .95 confidence interval
`ci.ub`	upper bound of the .95 confidence interval
`boot.zero`	proportion of bootstraps, in which an edge was estimated as equal to zero (i.e., 0= edge not estimated as zero throughout bootstraps; 1= edge estimated as zero in all bootstraps)
`boot.pos`	Proportion of bootstraps in which an edge was estimated as >0 (i.e., positive)
`boot.neg`	Proportion of bootstraps in which an edge was estimated as <0 (i.e., negative)
`g`	group in which the edge was estimated

Author(s)

Nils Kappelmann <n.kappelmann@gmail.com>, Giulio Costantini

References

Epskamp, S., Borsboom, D., & Fried, E. I. (2018). Estimating psychological networks and their accuracy: A tutorial paper. Behavior Research Methods, 50(1), 195–212. https://doi.org/10.3758/s13428-017-0862-1 Danaher, P., Wang, P., & Witten, D. M. (2014). The joint graphical lasso for inverse covariance estimation across multiple classes. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 76(2), 373–397. https://doi.org/10.1111/rssb.12033

See Also

JGL, **qgraph**, **parcor**

`covNoBessel` *Covariance matrix without Bessel's correction*

Description

Computes the Covariance matrix without Bessel's correction, for consistency with package **JGL**

Usage

```
covNoBessel(x,...)
```

Arguments

`x`	A dataframe of numeric values.
`...`	Arguments to be passed to cov

Value

A covariance matrix

Author(s)

Giulio Costantini

Examples

```
library(psych)
data(bfi)
covNoBessel(bfi, use = "complete.obs")
```

`EstimateGroupNetwork` *Estimate Joint Graphical Lasso model on data collected on observations from different groups.*

Description

The Joint Graphical lasso fits gaussian graphical models on data with the same variables observed on different groups or classes of interest (e.g., patients vs. controls; Danaher et al., 2014). The Joint Graphical Lasso relies on two tuning parameters, lambda1 and lambda2: This function performs tuning parameters selection relying on an information criterion (AIC / BIC / extended BIC) or k-fold cross validation and then fits the Joint Graphical Lasso model.

Usage

```
EstimateGroupNetwork(X,inputType = c("dataframe", "list.of.dataframes",
"list.of.covariance.matrices"),
                     n, covfun = covNoBessel, groupID, labels,
                     method = c("InformationCriterion", "crossvalidation"),
                     strategy = c("sequential", "simultaneous"),
                     nlambda1 = 100, lambda1.min.ratio = .01, logseql1 = TRUE,
                     nlambda2 = 100, lambda2.min.ratio = .01, logseql2 = TRUE,
                     k = 10, seed,
                     criterion = c("ebic", "bic", "aic"), count.unique = FALSE,
                     gamma = .5, dec = 5,
                     optimize = TRUE, optmethod = "CG",
              penalty = c("fused", "group"), weights = c("equal", "sample.size"),
               penalize.diagonal = FALSE, maxiter = 500, rho = 1, truncate = 1e-5,
                     ncores = 1, simplifyOutput = TRUE)
```

Arguments

Agruments describing input data

`X` Can be one of the following.

- A single dataframe including data from all groups, plus a group ID variable which must be specified as `groupID`.

- A list of dataframes, one by group. Each dataframe must be structured in the same way (the same variables for each group).

- A list of covariance or correlation matrices. Each matrix must be structured in the same way (the same variables for each group). For this type of input, a vector of sample sizes must be given in `n`.

`inputType` The type of data in input. If missing, the function will attempt to guess the type of input data. Can be one of the following:

- `"dataframe"`: A single dataframe including data from all groups, plus a group ID variable which must be specified as `groupID`.

- `"list.of.dataframes"`: A list of dataframes, one by group.

- `"list.of.covariance.matrices"`: A list of covariance or correlation matrices plus a vector of sample sizes `n`.

`n` Integer. Vector of sample sizes, one by group, in the same order in which the groups are included in the list of covariance matrices. This argument is relevant only if `inputType` is "list.of.covariance.matrices" and will be ignored otherwise (with a warning).

`covfun` The function used for computing the sample covariance matrix. The default, covNoBessel, computes the covariance matrix without Bessel's correction, for consistency with package **JGL**.

`groupID` a string. The name or number of the variable in the dataframe indicating a variable that identifies different groups. This argument is relevant only if `inputType` is "dataframe" and will be ignored otherwise.

`labels` Optional vector of strings. Name of each variable, in the same order in which they are included in the dataframe. If missing, column names will be used. If no column names are present, the variables will be simply named "V1", "V2", and so on.

Arguments connected to tuning parameter selection

`method` Methods for selecting tuning parameters. Can be one of the following:

- "InformationCriterion". Tuning parameters lambda 1 and lambda 2 are selected according to an information criterion. Argument `criterion` determines which information criterion is used. If the extended Bayes Information Criterion is used (see Foygel and Drton, 2010), the gamma parameter can be regulated through argument `gamma`. Argument `strategy` determines whether tuning parameter selection is performed simultaneously for lambda1 and lambda2, or separately for lambda 1 and lambda 2.

- "crossvalidation". Tuning parameters lambda 1 and lambda 2 are selected via k-fold crossvalidation. The cost function for the k-fold crossvalidation procedure is the average predictive negative loglikelihood, as defined in Guo et al. (2011, p.5). Parameter `k` regulates the number of sample splits for the crossvalidations (defaults to 10 splits), whereas parameter `seed` can be selected to ensure exact reproducibility of the results. Argument `strategy` determines whether crossvaliaditon is performed simultaneously for lambda1 and lambda2, or separately for lambda 1 and lambda 2.

`strategy` The strategy adopted for selecting tuning parameters. Can be one of the following:

- `"sequential"`: Tuning parameter selection is performed by first determining lambda 1 and then selecting lambda 2. This option is faster, but can return less accurate results than the next option.

- `"simultaneous"`: Tuning parameter selection is performed simultaneously for lambda 1 and lambda2. This option returns more accurate results, but it is also more computationally intensive and therefore slower.

General arguments that influence tuning parameter selection for all methods

`nlambda1` Integer. Number of candidate lambda 1 values. The candidate lambda 1 values will be spaced between the maximum value of lambda 1 (the one that results in at least one network being completely empty) and a minimum value, given by the maximum multiplied by lambda1.min.ratio

`lambda1.min.ratio`
Numeric. Ratio of lowest lambda 1 value compared to maximal lambda 1

`logseql1` Logical. If `FALSE`, the candidate lambda 1 values are equally spaced between a minimum and a maximum value; if `TRUE` the values are logarithmically spaced.

`nlambda2` Integer. Number of candidate lambda 2 values. The candidate lambda 2 values will be spaced between the maximum value of lambda 2 (the one that results in all groups having the same network) and a minimum value, given by the maximum multiplied by lambda1.min.ratio

`lambda2.min.ratio`
Numeric. Ratio of lowest lambda 2 value compared to maximal lambda 2

logseql2 — Logical. If FALSE, the candidate lambda 2 values are equally spaced between a minimum and a maximum value; if TRUE the values are logarithmically spaced.

Parameters for crossvalidation. The following arguments will be ignored if argument method *is not* "crossvalidation".

k — Integer. Number of splits for the k-fold cross-validation procedure.

seed — Integer. A seed for the random number generator, to include the exact reproducibility of the results obtained with the k-fold crossvalidation procedure.

Parameters for selecting tuning parameters via an information criterion. The following arguments will be ignored if argument method *is not* "InformationCriterion".

criterion — The Information criterion used for tuning parameter selection. Can be "aic", "bic" and "ebic" for Akaike information Criterion (Akaike, 1974), Bayes Information Criterion (Schwarz, 1978), and Extended Bayes Information Criterion (Foygel and Drton, 2010) respectively.

count.unique — Logical. Information criteria such as AIC, BIC and extended BIC include the number of model parameters in their formula. In Danaher et al (2014) an extension of the AIC is proposed in which each network edge is counted as a single parameter each time is different from zero in each group (up to a tolerance level, by default tol = 10^5, see parameter truncate). Therefore, even if the value of an edge is identical in two groups, it will be counted as two parameters. This option is implemented by selecting count.unique = FALSE. Here we implement an alternative possibility, which can be selected by setting argument count.unique = TRUE: If an edge is identical in two (or more) groups (up to a tolerance leve, see parameter dec), it will be counted as a single parameter.

gamma — Numeric. Parameter gamma for the extended Bayes Information Criterion (see Foygel and Drton, 2010).

dec — Integer. This is only relevant if count.unique = TRUE. Edges that are equal across groups up to the dec decimal place will be considered as one parameter in the information criteria.

optimize — Logical. If TRUE, after identifying the best tuning parameters (i.e., associated with the lowest value of an Information Criterion) among the candidate values, use an optimizer to try to further reduce the value of the information criterion. Since this is not a convex optimization problem, there is no guarantee that this step will lead to better results. However, it cannot do any harm either (if the optimization stage does not lead to improvements, the best value among the candidates will be returned). Be advised that setting this argument to TRUE results in longer computational time.

optmethod — If argument Strategy is set to "simultaneous" and argument optimize = TRUE, the optimization stage will consider simultaneous tuning parameters simultanously. Therefore, function optim will be used for the optimization stage. Argument optmethod can be used to set the optimization method. See parameter method in function optim.

Arguments that influence the Joint Graphical Lasso procedure. See also JGL

`penalty` Can be one of `"fused"` for Fused Graphical Lasso and `"group"` for Group Grahical Lasso. Fused is suggested. See Danaher et al. (2014) for details.

`weights` If `"equal"` all groups are equally weighted, if `"sample.size"` groups are weighted according to sample size.

`penalize.diagonal` Logical. If `TRUE`, the lambda 1 penalty is applied also the diagonal elements of the concentration matrix, otherwise the lambda 1 penalty is applied only to the off-diagonal elements. Notice that the lambda 2 penalty is always applied also to the diagonal elements.

`maxiter` Integer. Maximum number of iterations for the Joint Graphical Lasso procedure.

`rho` Numeric. A step size parameter for the Joint Graphical Lasso procedure. Large values decrease step size.

`truncate` Numeric. At convergence, all values of theta below this number will be set to zero.

Miscellaneous

`ncores` Numeric. Number of cores to use if working on a multicore system. `ncores = 1` implies no parallel processing

`simplifyOutput` Logical. If `TRUE`, only the estimated network will be returned. If `FALSE`, a much richer output will be returned. See section value.

Details

The code for the Joint Graphical Lasso procedure was adapted from the R package **JGL**. Some of the code for the cross-validation procedure was adapted from package **parcor**. Some of the code was inspired by package **qgraph**.

Value

If `simplifyOutput = TRUE`, a list corresponding to the networks estimated in each group is returned. If `simplifyOutput = FALSE`, a list is returned that includes including

`network` A list of matrices, each including the standardized partial correlation network for each group

`concentrationMatrix` A list of matrices, each including the unstandardized concentration matrix for each group

`correlationMatrix` A list of matrices, each including the correlation matrix for each group

`InformationCriteria` A vector including he information criteria AIC, BIC and extended BIC (eBIC), plus additional parameters that were used for their computation: the gamma value for eBIC and the values of parameters dec and count.unique

`Miscellaneous` A vector including several input parameters that could be important for replicating the results of the analysis

.

Author(s)

Giulio Costantini, Sacha Epskamp

References

Akaike, H. (1974), "A new look at the statistical model identification", IEEE Transactions on Automatic Control, 19 (6): 716-723, doi:10.1109/TAC.1974.1100705

Danaher, P (2013). JGL: Performs the Joint Graphical Lasso for sparse inverse covariance estimation on multiple classes. R package version 2.3. https://CRAN.R-project.org/package=JGL

Danaher, P., Wang, P., and Witten, D. M. (2014). The joint graphical lasso for inverse covariance estimation across multiple classes. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 76(2), 373-397. http://doi.org/10.1111/rssb.12033

Foygel, R., & Drton, M. (2010). Extended Bayesian Information Criteria for Gaussian Graphical Models. In NIPS (pp. 604-612). Chicago

Guo, J., Levina, E., Michailidis, G., & Zhu, J. (2011). Joint estimation of multiple graphical models. Biometrika, 98(1), 1-15. http://doi.org/10.1093/biomet/asq060

Schwarz, G. (1978). "Estimating the dimension of a model." The annals of statistics 6.2: 461-464.

See Also

JGL, **qgraph**, **parcor**

Examples

```
## Not run:
# Toy example, two identical networks with two nodes.
# This example is only meant to test the package. The number
# of candidate lambda1 and lambda2 values (nlambda1 and nlambda2) was
# reduced to 2 to speed up computations for CRAN checking.
Sigma <- list()
Sigma[[1]] <- Sigma[[2]] <- matrix(c(1, .5,
                                     .5, 1), nrow = 2)
recovered <- EstimateGroupNetwork(X = Sigma, n = c(100, 100),
                                  nlambda1 = 2, nlambda2 = 2, optimize = FALSE)


library("qgraph")
library("parallel")
library("psych")
library("mvtnorm")

ncores <- 1
# uncomment for parallel processing
# ncores <- detectCores() -1

# In this example, the BFI network of males and females are compared
# Load BFI data
data(bfi)
```

```
# remove observations with missing values
bfi2 <- bfi[rowSums(is.na(bfi[,1:26])) == 0,]

# Compute correlations:
CorMales <- cor_auto(bfi2[bfi2$gender == 1,1:25])
CorFemales <- cor_auto(bfi2[bfi2$gender == 2,1:25])

# Estimate JGL:
Res <- EstimateGroupNetwork(list(males = CorMales, females = CorFemales),
                            n = c(sum(bfi2$gender == 1),sum(bfi2$gender == 2)))

# Plot:
Layout <- averageLayout(Res$males,Res$females)
layout(t(1:2))
qgraph(Res$males, layout = Layout, title = "Males (JGL)")
qgraph(Res$females, layout = Layout, title = "Females (JGL)")



# Example with simluated data
# generate three network structures, two are identical and one is different
nets <- list()
nets[[1]] <- matrix(c(0, .3, 0, .3,
                      .3, 0, -.3, 0,
                      0, -.3, 0, .2,
                      .3, 0, .2, 0), nrow = 4)

nets[[2]] <- matrix(c(0, .3, 0, .3,
                      .3, 0, -.3, 0,
                      0, -.3, 0, .2,
                      .3, 0, .2, 0), nrow = 4)

nets[[3]] <- matrix(c(0, .3, 0, 0,
                      .3, 0, -.3, 0,
                      0, -.3, 0, .2,
                      0, 0, .2, 0), nrow = 4)

# optional: plot the original netwotk structures
par(mfcol = c(3, 1))
lapply(nets, qgraph, edge.labels = TRUE)

# generate nobs = 500 observations from each of the three networks
nobs <- 500
nvar <- ncol(nets[[1]])
set.seed(1)
X <- lapply(nets, function(x) as.data.frame(rmvnorm(nobs, sigma = cov2cor(solve(diag(nvar)-x)))))

# use EstimateGroupNetwork for recovering the original structures
recnets <- list()

# using EBICglasso
recnets$glasso <- list()
recnets$glasso[[1]] <- EBICglasso(S = cor(X[[1]]), n = nobs)
```

```
recnets$glasso[[2]] <- EBICglasso(S = cor(X[[2]]), n = nobs)
recnets$glasso[[3]] <- EBICglasso(S = cor(X[[3]]), n = nobs)

# Using Akaike information criterion without count.unique option
recnets$AIC1 <- EstimateGroupNetwork(X = X, method = "InformationCriterion",
criterion = "aic", ncores = ncores)
# Using Akaike information criterion with count.unique option
recnets$AIC2 <- EstimateGroupNetwork(X = X, method = "InformationCriterion",
criterion = "aic", ncores = ncores, count.unique = TRUE)
# Using Bayes information criterion without count.unique option
recnets$BIC1 <- EstimateGroupNetwork(X = X, method = "InformationCriterion",
criterion = "bic", ncores = ncores)
# Using Bayes information criterion with count.unique option
recnets$BIC2 <- EstimateGroupNetwork(X = X, method = "InformationCriterion",
criterion = "bic", ncores = ncores, count.unique = TRUE)
# Using extended Bayes information criterion (gamma = .5 by default)
# without count.unique option
recnets$eBIC1 <- EstimateGroupNetwork(X = X, method = "InformationCriterion",
ncores = ncores, criterion = "ebic")
# Using extended Bayes information criterion (gamma = .5 by default) with
# count.unique option
recnets$eBIC2 <- EstimateGroupNetwork(X = X, method = "InformationCriterion",
ncores = ncores, criterion = "ebic", count.unique = TRUE)
# Use a more computationally intensive search strategy
recnets$eBIC3 <- EstimateGroupNetwork(X = X, method = "InformationCriterion",
ncores = ncores, criterion = "ebic", count.unique = TRUE, strategy = "simultaneous")
# Add also the "optimization" stage, which may or may not improve the results
# (but cannot do any harm either)
recnets$eBIC3 <- EstimateGroupNetwork(X = X, method = "InformationCriterion",
ncores = ncores, criterion = "ebic", count.unique = TRUE, strategy = "simultaneous",
optimize = TRUE)
# Using k-fold crossvalidation (k = 10 by default)
recnets$cv <- EstimateGroupNetwork(X = X, method = "crossvalidation",
ncores = ncores, seed = 1)

# Compare each network with the data generating network using correlations
correl <- data.frame(matrix(nrow = length(recnets), ncol = length(nets)))
row.names(correl) <- names(recnets)

for(i in seq_along(recnets))
{
  for(j in seq_along(nets))
      {
    nt1 <- nets[[j]]
    nt2 <- recnets[[i]][[j]]
    correl[i, j] <-  cor(nt1[lower.tri(nt1)], nt2[lower.tri(nt2)])
  }
}
correl

# sort the methods in order of performance in recovering the original network
# notice that this is not a complete simulation and is not indicative of performance
# in settings other than this one
```

```
sort(rowMeans(correl))

## End(Not run)
```

`GroupBootPlot` *Create a plot of bootstrapped confidence intervals for all edges of a Joint Graphical Lasso model.*

Description

This function plots output from bootstrapped networks computed with GroupNetworkBoot.

Usage

```
GroupBootPlot(BootOut, GroupNames, edges.x, edges.y,
labels = TRUE, transparency = 0.15, point.size = 1.5, line.size = 1, scales = "fixed",
legend.position = "none", GroupNamesCheck = FALSE)
```

Arguments

`BootOut` The output from GroupNetworkBoot.

`GroupNames` A vector of optional group names that will be printed as facet labels in plot. By default, names of the networks are taken. If specified, GroupNames should match the alphabetical order of names of network groups. If unsure, you can check the matching of names by setting `GroupNamesCheck = TRUE`.

`edges.x` If only a subset of edge combinations is of interest for the plot, this subset can be specified by setting `edges.x` and `edges.y`. Specifically, node names can be specified as vectors for `edges.x` and `edges.y` and all unique combinations of `edges.x` and `edges.y` will be plotted. For example, `edges.x = c("a", "b")` and `edges.y = "c"` will plot edges a-c and b-c but not a-b.

`edges.y` See `edges.x`.

`labels` Logical, should edge labels be included in plots. Default is `labels = TRUE`.

`transparency` Set **ggplot2** alpha channel (transparency) for confidence interval ribbon in plot.

`point.size` Set point size.

`line.size` Set line size.

`scales` Set **ggplot2** facet scales. Default is `scale = "fixed"`. See ?facet_grid in ggplot2 for details.

`legend.position` Define legend position to indicate colour for sample and bootstrap means. See ?theme in ggplot2.

`GroupNamesCheck` Option to print match of indicated GroupNames to console. Only prints if GroupNames is specified. See `GroupNames` for details.

Details

The code for the Joint Graphical Lasso procedure was adapted from the R package **JGL**. Some of the code for the cross-validation procedure was adapted from package **parcor**. Some of the code was inspired by package **qgraph**. GroupBootPlot automatically calls BootTable to format GroupNetworkBoot output, so see BootTable for completely independent plotting.

Value

The output of GroupBootPlot returns a plot based on **ggplot2** with the bootstrapped confidence intervals of edges across groups.

Author(s)

Nils Kappelmann <n.kappelmann@gmail.com>, Giulio Costantini

References

Danaher, P., Wang, P., and Witten, D. M. (2014). The joint graphical lasso for inverse covariance estimation across multiple classes. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 76(2), 373-397. http://doi.org/10.1111/rssb.12033

See Also

JGL, **qgraph**, **parcor**

`GroupNetworkBoot` *Compute bootstrap networks for a Joint Graphical Lasso model on data collected on observations from different groups.*

Description

This bootstrapping function resamples from intial dataframes to compute bootstrapping intervals for edges estimated using EstimateGroupNetwork.

Usage

```
GroupNetworkBoot(data_list, groupNetwork, nboots = 100, bootSeed, ...)
```

Arguments

`data_list` A list of dataframes, one by group. Each dataframe must be structured in the same way (the same variables for each group). This needs to be the same input as was used for the original Joint Graphical Lasso network estimated with `EstimateGroupNetwork`.

`groupNetwork` The to-be-bootstrapped network estimated with the `EstimateGroupNetwork` function. Importantly, the initial Joint Graphical Lasso needs to be estimated with `simplifyOutput = FALSE`.

nboots	The number of bootstraps to-be-conducted.
bootSeed	An optional random seed for ensuring replicability of the results.
...	All further arguments need to be specified as done for the initial computation of the EstimateGroupNetwork function. Here all arguments apply and have the default values of function EstimateGroupNetwork, with the exceptions being the arguments inputType = "list.of.dataframes", simplifyOutput = FALSE, and labels, as node labels are taken directly from the original network. These arguments are set by default.

Details

Some of the code for the cross-validation procedure was adapted from package **parcor**. Some of the code was inspired by package **qgraph**.

Value

The output of GroupNetworkBoot returns a list with the following elements:

data	The original list of dataframes supplied to the function
sample	A list including the original output from EstimateGroupNetwork
boot	A list of matrices, each including a bootstrapped network

Author(s)

Nils Kappelmann <n.kappelmann@gmail.com>, Giulio Costantini

References

Danaher, P (2013). JGL: Performs the Joint Graphical Lasso for sparse inverse covariance estimation on multiple classes. R package version 2.3. https://CRAN.R-project.org/package=JGL
Danaher, P., Wang, P., and Witten, D. M. (2014). The joint graphical lasso for inverse covariance estimation across multiple classes. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 76(2), 373-397. http://doi.org/10.1111/rssb.12033

See Also

JGL, **qgraph**

Examples

```
## Not run:

## Load packages:
library("psych")
library("EstimateGroupNetwork")

# In this example, the BFI network of males and females are compared for the subset of
# Agreeableness items
# Load BFI data
data(bfi)
```

```
## The bfi data is subset to Agreeableness items only for the first 500 individuals to decrease
# computational time
bfi <- bfi[, c(paste("A", 1:5, sep = ""), "gender")]

# remove observations with missing values on items or gender
bfi <- na.omit(bfi)

# Create list split by gender
bfi_list <- list(males = bfi[bfi$gender == 1, 1:5],
                 females = bfi[bfi$gender == 2, 1:5])

# Estimate JGL:
bfi_net <- EstimateGroupNetwork(bfi_list, inputType = "list.of.dataframes", simplifyOutput = FALSE)

# Bootstrap network 10 times (this will take a few minutes)
boot_bfi_net <- GroupNetworkBoot(data_list = bfi_list, groupNetwork = bfi_net,
                                 nboots = 10, bootSeed = 1234, ncores = 1)

# use BootTable to obtain a table with information for each boostrapped edge
BootTable(boot_bfi_net)

## Use GroupBootPlot to obtain plots as a list with each group plot as one element
GroupBootPlot(boot_bfi_net)

## Get plot for a subset of edges (here: all edges including A1). Also check Groupnames
GroupBootPlot(boot_bfi_net, edges.x = "A1", edges.y = c("A2", "A3", "A4", "A5"),
         GroupNames = c("Females", "Males"), GroupNamesCheck = TRUE, legend.position = "top")


## End(Not run)
```

Index

∗ **graphs**
BootTable, 2
EstimateGroupNetwork, 3
GroupBootPlot, 11
GroupNetworkBoot, 12
∗ **multivariate**
BootTable, 2
covNoBessel, 3
EstimateGroupNetwork, 3
GroupBootPlot, 11
GroupNetworkBoot, 12

BootTable, 2, *12*

cov, *3*
covNoBessel, 3, *4*

EstimateGroupNetwork, 3

GroupBootPlot, 11
GroupNetworkBoot, *2*, *11, 12*, 12

optim, *6*