Package ‘lvm4net’

October 13, 2022

Title Latent Variable Models for Networks
Description Latent variable models for network data using fast inferential procedures. For more information please visit: <http://igollini.github.io/lvm4net/>.
Version 0.3
Depends R (>= 3.5), MASS, ergm, network
Imports igraph, grDevices, graphics, ellipse, stats, utils, glmmML, mvtnorm, corpcor
Suggests knitr, rmarkdown, manet, colorspace
VignetteBuilder knitr
License GPL (>= 2)
LazyData true
URL http://github.com/igollini/lvm4net
BugReports http://github.com/igollini/lvm4net/issues
Encoding UTF-8
RoxygenNote 6.1.1
NeedsCompilation no
Author Isabella Gollini [aut, cre] (<https://orcid.org/0000-0002-7738-5688>)
Maintainer Isabella Gollini <igollini.stats@gmail.com>
Repository CRAN
Date/Publication 2019-06-13 15:40:07 UTC

R topics documented:

lvm4net-package ...................................................... 2
boxroc ................................................................. 3
goflsm ................................................................. 4
lca ................................................................. 5
lift ................................................................. 6
lsjm ................................................................. 7
Description

`lv4mnet` provides a range of tools for latent variable models for network data. Most of the models are implemented using a fast variational inference approach.

Details

Latent space models for one-mode binary networks: the function `lsm` implements the latent space model (LSM) introduced by Hoff et al. (2002) using variational inference and squared Euclidian distance; the function `lsjm` implements latent space joint model (LSJM) for multiplex networks introduced by Gollini and Murphy (2016). These models assume that each node of a network has a latent position in a latent space: the closer two nodes are in the latent space, the more likely they are connected.

Latent variable models for binary bipartite networks: the function `lca` implements the latent class analysis (LCA) to find groups in the sender nodes (with the condition of independence within the groups); the function `lta` implements the latent trait analysis (LTA) to model the dependence in the receiver nodes by using a continuous latent variable; the function `mlta` implements the mixture of latent trait analyzers (MLTA) introduced by Gollini and Murphy (2014) and Gollini (in press) to identify groups assuming the existence of a latent trait describing the dependence structure between receiver nodes within groups of sender nodes and therefore capturing the heterogeneity of sender nodes’ behaviour within groups. `lta` and `mlta` use variational inference.

References


---

boxroc

Boxplot and ROC Curves

Description

Function to display boxplots and ROC curves to show model fit in terms of in-sample link prediction.

Usage

boxroc(Y, EZ, xiT, BOXPLOT = FALSE, ROC = FALSE, Lroc = 100, labelsPlot = NULL, powdist = 2, cexRocLeg = 0.8, colRoc = seq(2, Ndata + 1), ltyRoc = seq(2, Ndata + 1), lwdRoc = 2, ...)

Arguments

Y (N x N) binary adjacency matrix, or list containing the adjacency matrices.
EZ (N x D) matrix (or list of matrices) containing the posterior means of the latent positions
xiT vector of posterior means of the parameter α
BOXPLOT logical; if TRUE draws the boxplot. Default BOXPLOT = FALSE
ROC logical; if TRUE draws the ROC curve. Default ROC = FALSE
Lroc number of intervals in the ROC curve. Default Lroc = 100
labelsPlot main title for the boxplot. Default labelsPlot = NULL
powdist vector of power of the distance default powdist = 2, squared Euclidean distance, the alternative is 1, for the Euclidean distance
cexRocLeg cex for the ROC curve. Default cexRocLeg = .8
colRoc col for the ROC curve. Default colRoc = seq(2, Ndata + 1)
ltyRoc lty for the ROC curve. Default ltyRoc = seq(2, Ndata + 1)
lwdRoc lwd for the ROC curve. Default lwdRoc = 2
...

Value

The area under the ROC curve (AUC) and the selected plots. The closer the AUC takes values to 1 the better the fit.
References

See Also
lsm, lsjm

Examples

N <- 20
Y <- network(N, directed = FALSE)[,]
modLSM <- lsm(Y, D = 2)
bp <- boxroc(Y,
EZ = modLSM$lsmEZ,
xiT = modLSM$xiT,
Lroc = 150,
ROC = TRUE,
BOXPLOT = TRUE)

print(bp)

goflsm Goodness-of-Fit diagnostics for LSM model

Description
This function produces goodness-of-fit diagnostics for LSM model.

Usage
goflsm(object, Y, Ysim = NULL, nsim = 100, seed, directed = NULL,
stats = NULL, doplot = TRUE, parm = TRUE)

Arguments
object object of class 'lsm'
Y (N x N) binary adjacency matrix
Ysim list containing simulated (N x N) adjacency marices. Default Ysim = NULL
nsim number of simulations. Default nsim = 100
seed for simulations
directed if the network is directed or not. Default directed = NULL
stats statistics used. Default stats = NULL
doplot draw boxplot. Default doplot = TRUE
parm do all the plots in one window. Default parm = TRUE
**lca**

Latent Class Analysis

**See Also**

lsm, simulateLSM, plot.gofobj, print.gofobj

**Examples**

```r
Y <- network(15, directed = FALSE)[,]
modLSM <- lsm(Y, D = 2)
myGof <- goflsm(modLSM, Y = Y)
```

**Description**

Latent class analysis (LCA) can be used to find groups in the sender nodes (with the condition of independence within the groups). For more details see Gollini, I. (in press) and Gollini, I., and Murphy, T. B. (2014).

**Usage**

```r
lca(X, G, nstarts = 3, tol = 0.1^2, maxiter = 500)
```

**Arguments**

- `X` (N x M) binary incidence matrix
- `G` number of groups
- `nstarts` integer number of different starts for the EM algorithm. Default `nstarts = 3`.
- `tol` desired tolerance for convergence. Default `tol = 0.1^2`
- `maxiter` maximum number of iterations. Default `maxiter = 500`

**Value**

List containing the following information for each model fitted:

- `p` (G x M) matrix containing the conditional probability of observing a link to sender nodes if the receiver nodes are from group g.
- `eta` $\eta_g$ is the mixing proportion for the group $g = g(1, ..., G)$, that corresponds to the prior probability that a randomly chosen sender node is in the g-th group.
- `z` (N x G) matrix containing posterior probability for each sender node to belong to each group
- `LL` log likelihood
- `BIC` Bayesian Information Criterion (BIC) (Schwarz (1978))

If multiple models are fitted the output contains also a table to compare the BIC for all models fitted.
References


See Also

mlta

Examples

```r
### Simulate Bipartite Network
set.seed(1)
X <- matrix(rbinom(4 * 12, size = 1, prob = 0.4), nrow = 12, ncol = 4)
resLCA <- lca(X, G = 2:3)

lift
```

Description

The lift can be used to analyse the dependence within each groups found using the function mleta. The lift can be used to quantify the effect of the dependence on the probability of a sender nodes being liked to two receivers within each group compared to the probability of being liked to two receivers under an independence model. Two independent links to the receiver nodes have lift = 1: the more the links to receiver nodes are dependent, the further the value of the lift is from 1.

Usage

```r
lift(x, pdGH = 21)
```

Arguments

- `x`: object of class mleta
- `pdGH`: number of quadrature points for the Gauss-Hermite quadrature. Default `pdGH = 21`

Value

The function returns an (M x M x D) array.
lsjm

References


See Also

mlta

Examples

```r
### Simulate Bipartite Network
set.seed(1)
X <- matrix(rbinom(4 * 12, size = 1, prob = 0.4), nrow = 12, ncol = 4)
res <- mlta(X, G = 2, D = 1)
res_lift <- lift(res)
```

---

lsjm

**Latent Space Joint Model**

Description

Function to joint modelling of multiple network views using the Latent Space Joint Model (LSJM) Gollini and Murphy (2016). The LSJM merges the information given by the multiple network views by assuming that the probability of a node being connected with other nodes in each view is explained by a unique latent variable.

Usage

```r
lsjm(Y, D, sigma = 1, xi = rep(0, length(Y)), psi2 = rep(2, length(Y)), Niter = 500, tol = 0.1^2, preit = 20, randomZ = FALSE)
```

Arguments

- `Y` list containing a (N x N) binary adjacency matrix for each network view.
- `D` integer dimension of the latent space
- `sigma` (D x D) variance/covariance matrix of the prior distribution for the latent positions. Default `sigma = 1`
- `xi` vector of means of the prior distributions of $\alpha$. Default `xi = 0`
- `psi2` vector of variances of the prior distributions of $\alpha$. Default `psi2 = 2`
- `Niter` maximum number of iterations. Default `Niter = 500`
tol desired tolerance. Default tol = 0.1^2
preit Preliminary number of iterations default preit = 20
randomZ logical; If randomZ = TRUE random initialization for the latent positions is used.
If randomZ = FALSE and D = 2 or 3 the latent positions are initialized using the
Fruchterman-Reingold method and multidimensional scaling is used for D = 1
or D > 3. Default randomZ = FALSE

Value
List containing:
• EZ (N x D) matrix containing the posterior means of the latent positions
• VZ (D x D) matrix containing the posterior variance of the latent positions
• lsmEZ list containing a (N x D) matrix for each network view containing the posterior means
of the latent positions under each model in the latent space.
• lsmVZ list containing a (D x D) matrix for each network view containing the posterior variance
of the latent positions under each model in the latent space.
• xiT vector of means of the posterior distributions of α
• psi2T vector of variances of the posterior distributions of α
• Ell expected log-likelihood

References
Gollini, I., and Murphy, T. B. (2016), 'Joint Modelling of Multiple Network Views', Journal of

Examples
## Simulate Undirected Network
N <- 20
Ndata <- 2
Y <- list()
Y[[1]] <- network(N, directed = FALSE)[,]
### create a new view that is similar to the original
for(nd in 2:Ndata){
  Y[[nd]] <- Y[[nd - 1]] - sample(c(-1, 0, 1), N * N, replace = TRUE,
prob = c(.05, .85, .1))
  Y[[nd]] <- 1 * (Y[[nd]] > 0 )
  diag(Y[[nd]]) <- 0
}
par(mfrow = c(1, 2))
z <- plotY(Y[[1]], verbose = TRUE, main = 'Network 1')
plotY(Y[[2]], EZ = z, main = 'Network 2')
par(mfrow = c(1, 1))
modLSJM <- lsjm(Y, D = 2)
plot(modLSJM, Y, drawCB = TRUE)
plot(modLSJM, Y, drawCB = TRUE, plotZtilde = TRUE)
Latent space models (LSM) are a well known family of latent variable models for network data introduced by Hoff et al. (2002) under the basic assumption that each node has an unknown position in a D-dimensional Euclidean latent space: generally the smaller the distance between two nodes in the latent space, the greater the probability of them being connected. Unfortunately, the posterior distribution of the LSM cannot be computed analytically. For this reason we propose a variational inferential approach which proves to be less computationally intensive than the MCMC procedure proposed in Hoff et al. (2002) (implemented in the latentnet package) and can therefore easily handle large networks. Salter-Townshend and Murphy (2013) applied variational methods to fit the LSM with the Euclidean distance in the VBLPCM package. In this package, a distance model with squared Euclidean distance is used. We follow the notation of Gollini and Murphy (2016).

Usage

```r
lsm(Y, D, sigma = 1, xi = 0, psi2 = 2, Niter = 100, Miniter = 10,
   tol = 0.1^2, randomZ = FALSE, nstart = 1)
```

Arguments

- `Y` (N x N) binary adjacency matrix
- `D` integer dimension of the latent space
- `sigma` (D x D) variance/covariance matrix of the prior distribution for the latent positions. Default `sigma = 1`
- `xi` mean of the prior distribution of \( \alpha \). Default `xi = 0`
- `psi2` variance of the prior distribution of \( \alpha \). Default `psi2 = 2`
- `Niter` maximum number of iterations. Default `Niter = 100`
- `Miniter` minimum number of iterations. Default `Miniter = 10`
- `tol` desired tolerance. Default `tol = 0.1^2`
- `randomZ` logical; If `randomZ = TRUE` random initialization for the latent positions is used. If `randomZ = FALSE` and D = 2 or 3 the latent positions are initialized using the Fruchterman-Reingold method and multidimensional scaling is used for D = 1 or D > 3. Default `randomZ = FALSE`
- `nstart` number of starts

Value

List containing:

- `lsmEZ` (N x D) matrix containing the posterior means of the latent positions
- `lsmVZ` (D x D) matrix containing the posterior variance of the latent positions
Latent Trait Analysis

Description

Latent trait analysis (LTA) can be used to model the dependence in the receiver nodes by using a continuous D-dimensional latent variable. The function `lta` makes use of a variational inferential approach. For more details see Gollini, I. (in press) and Gollini, I., and Murphy, T. B. (2014).

Usage

```r
lta(X, D, nstarts = 3, tol = 0.1^2, maxiter = 250, pdGH = 21)
```

Arguments

- `X` (N x M) binary incidence matrix
- `D` dimension of the continuous latent variable
- `nstarts` number of starts. Default `nstarts = 3`
- `tol` desired tolerance for convergence. Default `tol = 0.1^2`
- `maxiter` maximum number of iterations. Default `maxiter = 500`
- `pdGH` number of quadrature points for the Gauss-Hermite quadrature. Default `pdGH = 21`

Examples

```r
### Simulate Undirected Network
N <- 20
Y <- network(N, directed = FALSE)[,]
modLSM <- lsm(Y, D = 2)
plot(modLSM, Y)
```

References


See Also

`plot.lsm`

References

Value

List containing the following information for each model fitted:

- \( \mathbf{b} \) intercepts for the logistic response function
- \( \mathbf{w} \) slopes for the logistic response function
- \( \mathbf{mu} \) (\( N \times D \)) matrix containing posterior means for the latent variable
- \( \mathbf{C} \) list of \( N \) (\( D \times D \)) matrices containing posterior variances for the latent variable
- \( \text{LL} \) log likelihood
- \( \text{BIC} \) Bayesian Information Criterion (BIC) (Schwarz (1978))

If multiple models are fitted the output contains also a table to compare the BIC for all models fitted.

References


See Also

mlta

Examples

```r
### Simulate Bipartite Network
set.seed(1)
X <- matrix(rbinom(4 * 12, size = 1, prob = 0.4), nrow = 12, ncol = 4)
resLTA <- lta(X, D = 1:2)
```

---

mlta  
Mixture of Latent Trait Analyzers

Description

Mixture of latent trait analyzers (MLTA) has been introduced by Gollini and Murphy (2014) and Gollini (in press) to identify groups assuming the existence of a latent trait describing the dependence structure between receiver nodes within groups of sender nodes and therefore capturing the heterogeneity of sender nodes' behaviour within groups. The function `mlta` makes use of a variational inferential approach. For more details see Gollini, I. (in press) and Gollini, I., and Murphy, T. B. (2014).
Usage

mlta(X, G, D, wfix = FALSE, nstarts = 3, tol = 0.1^2, maxiter = 250, pdGH = 21)

Arguments

X  
(N x M) binary incidence matrix

G  
number of groups

D  
dimension of the continuous latent variable

wfix  
Logical. Fit the parsimonious model with the w parameters equal across groups. Default wfix = FALSE

nstarts  
number of starts. Default nstarts = 3

tol  
desired tolerance for convergence. Default tol = 0.1^2

maxiter  
maximum number of iterations. Default maxiter = 500

pdGH  
number of quadrature points for the Gauss-Hermite quadrature. Default pdGH = 21

Value

List containing the following information for each model fitted:

- b matrix containing intercepts for the logistic response function
- w array containing slopes for the logistic response function
- eta \( \eta_g \) is the mixing proportion for the group \( g (g = 1, ..., G) \), that corresponds to the prior probability that a randomly chosen sender node is in the \( g \)-th group.
- mu (N x D x G) array containing posterior means for the latent variable
- C (D x D x N x G) array containing posterior variances for the latent variable
- z (N x G) matrix containing posterior probability for each sender node to belong to each group
- LL log likelihood
- BIC Bayesian Information Criterion (BIC) (Schwarz (1978))

If multiple models are fitted the output contains also tables to compare the log likelihood and BIC for all models fitted.

References


See Also

lta lca
Examples

```r
### Simulate Bipartite Network
set.seed(1)
X <- matrix(rbinom(4 * 12, size = 1, prob = 0.4), nrow = 12, ncol = 4)
resMLTA <- mlta(X, G = 2, D = 1)
```

plot.gofobj

Plot GoF object

Description

Function to plot an object of class 'gofobj'

Usage

```r
## S3 method for class 'gofobj'
plot(x, parm = TRUE, ...)
```

Arguments

- `x` object of class "gofobj"
- `parm` do all in one plots
- `...` other

Examples

```r
Y <- network(20, directed = FALSE)[,]
modLSM <- lsm(Y, D = 2)
myGof <- goflsm(modLSM, Y = Y, doplot = FALSE)
plot(myGof)
```

plot.lsjm

Two dimensional plot of Latent Space Joint Model output

Description

Function to plot an object of class 'lsjm'

Usage

```r
## S3 method for class 'lsjm'
plot(x, Y, drawCB = FALSE, dimZ = c(1, 2),
    plotZtilde = FALSE, colP1 = 1, colEll = rgb(0.6, 0.6, 0.6, alpha = 0.1),
    LEVEL = 0.95, pchplot = 20, pchEll = 19, pchP1 = 19,
    cexP1 = 1.1, mainZtilde = NULL, arrowhead = FALSE, curve = NULL,
    xlim = NULL, ylim = NULL, main = NULL, ...)
```
Arguments

- **x**: object of class 'lsjm'
- **Y**: list containing a (N x N) binary adjacency matrix for each network view.
- **drawCB**: logical if drawCB = TRUE draw confidence bounds
- **dimZ**: dimensions of the latent variable to be plotted. Default dimZ = c(1, 2)
- **plotZtilde**: if TRUE do the plot for the last step of LSM
- **colPl**: col for the points representing the nodes. Default colPl = NULL
- **colEll**: col for the ellipses. Default rgb(.6, .6, .6, alpha=.1)
- **LEVEL**: levels of confidence bounds shown when plotting the ellipses. Default LEVEL = .95
- **pchplot**: Default pchplot = 20
- **pchEll**: pch for the ellipses. Default pchEll = 19
- **pchPl**: pch for the points representing the nodes. Default pchPl = 19
- **cexPl**: cex for the points representing the nodes. Default cexPl = 1.1
- **mainZtilde**: title for single network plots TRUE do the plot for the last step of LSM
- **arrowhead**: logical, if the arrowed are to be plotted. Default arrowhead = FALSE
- **curve**: curvature of edges. Default curve = 0
- **xlim**: range for x
- **ylim**: range for y
- **main**: main title
- **...**: Arguments to be passed to methods, such as graphical parameters (see **par**).

Examples

```r
## Simulate Undirected Network
N <- 20
Ndata <- 2
Y <- list()
Y[[1]] <- network(N, directed = FALSE)

### create a new view that is similar to the original
for(nd in 2:Ndata){
  Y[[nd]] <- Y[[nd - 1]] - sample(c(-1, 0, 1), N * N, replace = TRUE, prob = c(.05, .85, .1))
  Y[[nd]] <- 1 * (Y[[nd]] > 0 )
  diag(Y[[nd]]) <- 0
}

par(mfrow = c(1, 2))
z <- plotY(Y[[1]], verbose = TRUE, main = 'Network 1')
plotY(Y[[2]], EZ = z, main = 'Network 2')
par(mfrow = c(1, 1))

modLSJM <- lsjm(Y, D = 2)
plot(modLSJM, Y, drawCB = TRUE)
plot(modLSJM, Y, drawCB = TRUE, plotZtilde = TRUE)
```
plot.lsm  Two dimensional plot of the Latent Space Model output

Description

Function to plot an object of class 'lsm'

Usage

## S3 method for class 'lsm'
plot(x, Y, drawCB = FALSE, dimZ = c(1, 2), colPl = 1,
colEll = rgb(0.6, 0.6, 0.6, alpha = 0.1), LEVEL = 0.95,
pchplot = 20, pchEll = 19, pchPl = 19, cexPl = 1.1,
arrowhead = FALSE, curve = NULL, xlim = NULL, ylim = NULL, ...)

Arguments

x  object of class 'lsm'
Y  (N x N) binary adjacency matrix
drawCB  draw confidence bounds
dimZ  dimensions of the latent variable to be plotted. Default dimZ = c(1, 2)
colPl  col for the points representing the nodes. Default colPl = NULL
colEll  col for the ellipses. Default rgb(.6, .6, .6, alpha=.1)
LEVEL  levels of confidence bounds shown when plotting the ellipses. Default LEVEL = .95
pchplot  Default pchplot = 20
pchEll  pch for the ellipses. Default pchEll = 19
pchPl  pch for the points representing the nodes. Default pchPl = 19
cexPl  cex for the points representing the nodes. Default cexPl = 1.1
arrowhead  logical, if the arrowed are to be plotted. Default arrowhead = FALSE
curve  curvature of edges. Default curve = 0
xlim  range for x
ylim  range for y
...  Arguments to be passed to methods, such as graphical parameters (see par).

Examples

N <- 20
Y <- network(N, directed = FALSE)[,]
modLSM <- lsm(Y, D = 2)
plot(modLSM, Y)
plotY

Plot the adjacency matrix of the network

Description

Function to plot the adjacency matrix of the network.

Usage

plotY(Y, Ndata = NULL, EZ = NULL, VZ = NULL, dimZ = c(1, 2),
labels = NULL, colPl = 1, colEll = rgb(0.6, 0.6, 0.6, alpha = 0.1),
LEVEL = 0.95, pchplot = 20, pchEll = 19, pchPl = 19,
cexPl = 1.1, arrowhead = FALSE, curve = NULL, lwdLine = 0.3,
xlim = NULL, ylim = NULL, verbose = FALSE, ...)

Arguments

Y list, or matrix containing a (N x N) binary adjacency matrix for each network view.
Ndata number of network views
EZ posterior mean latent positions
VZ posterior variance latent positions, if specified draw ellipse
dimZ dimensions of Z to be plotted, default dimZ = c(1, 2)
labels text to be added in the plot representing the labels of each node. Default labels = NULL, no labels are shown
colPl col for the points representing the nodes. Default colPl = NULL
colEll col for the ellipses. Default rgb(.6, .6, .6, alpha=.1)
LEVEL levels of confidence bounds shown when plotting the ellipses. Default LEVEL = .95
pchplot Default pchplot = 20
pchEll pch for the ellipses. Default pchEll = 19
pchPl pch for the points representing the nodes. Default pchPl = 19
cexPl cex for the points representing the nodes. Default cexPl = 1.1
arrowhead logical, if the arrowed are to be plotted. Default arrowhead = FALSE
curve curvature of edges. Default curve = 0
lwdLine lwd of edges. Default lwdLine = .3
xlim range for x
ylim range for y
verbose if verbose = TRUE save the nodal positions
... Arguments to be passed to methods, such as graphical parameters (see par).
Examples

```r
N <- 20
Y <- network(N, directed = FALSE)[,]
plotY(Y)
# Store the positions of nodes used to plot Y, in order to redraw the plot using
# the same positions
z <- plotY(Y, verbose = TRUE)
plotY(Y, EZ = z)
```

Description

The dataset contains a network formed by genetic protein-protein interactions (PPI) between 67 Saccharomyces cerevisiae proteins. The network is formed of 294 links. The data were downloaded from the Biological General Repository for Interaction Datasets (BioGRID) database [http://thebiogrid.org/](http://thebiogrid.org/).

Usage

`PPIgen`

Format

Binary adjacency matrix

Details

Binary adjacency matrix containing genetic interactions between 67 proteins.

References


See Also

`PPIphy`
PPInet

*PPI genetic and physical interactions data*

**Description**

The dataset contains two undirected networks formed by genetic and physical protein-protein interactions (PPI) between 67 Saccharomyces cerevisiae proteins. The genetic interactions network is formed of 294 links, and the physical interactions network is formed of 190 links. The data were downloaded from the Biological General Repository for Interaction Datasets (BioGRID) database [http://thebiogrid.org/](http://thebiogrid.org/)

**Format**

Two binary adjacency matrices

**Details**

- **PPIgen** Binary adjacency matrix containing genetic interactions between 67 proteins.
- **PPIphy** Binary adjacency matrix containing physical interactions between 67 proteins.

**Source**


**References**


**See Also**

PPIgen, PPIphy

---

**PPIphy**

*PPI physical interactions*

**Description**

The dataset contains a network formed by physical protein-protein interactions (PPI) between 67 Saccharomyces cerevisiae proteins. The network is formed of 190 links. The data were downloaded from the Biological General Repository for Interaction Datasets (BioGRID) database [http://thebiogrid.org/](http://thebiogrid.org/)

**Usage**

PPIphy
**Format**

Binary adjacency matrix

**Details**

Binary adjacency matrix containing physical interactions between 67 proteins.

**References**


**See Also**

PPIgen

---

**Description**

Function to print an object of class 'gofobj'

**Usage**

```r
## S3 method for class 'gofobj'
print(x, ...)
```

**Arguments**

- `x` object of class 'gofobj'
- `...` other

**Examples**

```r
Y <- network(20, directed = FALSE)[,]
modLSM <- lsm(Y, D = 2)
myGof <- goflsm(modLSM, Y = Y, doplot = FALSE)
print(myGof)
```
**rotXtoY**  
*Rotate X to match Y*

**Description**
Function to rotate X to match Y via singular value decomposition

**Usage**

```r
rotXtoY(X, Y)
```

**Arguments**

- **X**: matrix to be rotated
- **Y**: objective matrix

**Value**
rotated object \(X_{rot}\), and the rotation matrix \(R\)

**simulateLSM**  
*Simulate from LSM model*

**Description**
Function to simulate networks from the LSM model

**Usage**

```r
simulateLSM(object, Y = NULL, nsim = 100, seed, directed = NULL)
```

**Arguments**

- **object**: object of class `Var`\(lsm\`
- **Y**: \((N\times N)\) binary adjacency matrix
- **nsim**: number of simulations. Default \(nsim = 100\)
- **seed**: for simulations
- **directed**: if the network is directed or not. Default \(directed = \text{NULL}\)
Examples

```r
n <- 20
Y <- network(n, directed = FALSE)[,]

modLSM <- lsm(Y, D = 2)

Ysim <- simulateLSM(modLSM, Y = Y, nsim = 8)
# store EZ, to keep the nodes in the same positions
# and compare the networks
EZ <- modLSM$lsmEZ
par(mfrow = c(3,3))
plotY(Y, EZ = EZ, main = "Original Data")
for(i in 1:8) plotY(Ysim[[i]], EZ = EZ, main = paste("Simulation", i))
par(mfrow = c(1,1))
```
Index

* datasets
  PPIgen, 17
  PPInet, 18
  PPIphy, 18

boxroc, 3

gof1sm, 4

lca, 2, 5, 12
lift, 6
lsjm, 2, 4, 7
lsm, 2, 4, 5, 9
lta, 2, 10, 12
lvm4net (lvm4net-package), 2
lvm4net-package, 2

mlta, 2, 6, 7, 11, 11

par, 3, 14–16
plot.gofobj, 5, 13
plot.lsjm, 13
plot.lsm, 10, 15
plotY, 16
PPIgen, 17, 18, 19
PPInet, 18
PPIphy, 17, 18, 18
print.gofobj, 5, 19

rotXtoY, 20

simulateLSM, 5, 20