Package ‘lvm4net’

October 7, 2018

Title  Latent Variable Models for Networks
Description  Latent variable models for network data using fast inferential procedures.
Version  0.2.1
Depends  R (>= 3.1.1), MASS, ergm, network
Imports  igraph, grDevices, graphics, ellipse, stats, utils
License  GPL (>= 2)
LazyData true
URL  http://github.com/igollini/lvm4net
BugReports  http://github.com/igollini/lvm4net/issues
Encoding  UTF-8
RoxygenNote  6.1.0
NeedsCompilation no
Author  Isabella Gollini [aut, cre]
Maintainer  Isabella Gollini <igollini.stats@gmail.com>
Repository  CRAN
Date/Publication  2018-10-07 20:30:03 UTC

R topics documented:

lvm4net-package ......................................................... 2
boxroc ................................................................. 2
goflsm ............................................................... 4
lsjm ................................................................. 5
lsm ................................................................. 6
plot.gofobj ......................................................... 8
plot.lsjm .......................................................... 8
plot.lsm .......................................................... 10
plotY ........................................................... 11
PPIgen ............................................................ 12
lvm4net-package  

**Description**

lvm4net provides a range of tools for latent variable models for network data. Most of the models are implemented using a fast variational inference approach. Latent space models for binary networks: the function `lsm` implements the latent space model (LSM) introduced by Hoff et al. (2002) using a 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. Functions for binary bipartite networks will be added soon.

**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**

```r
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, ...)```
boxroc

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 \( \alpha \)
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
labelPlot  main title for the boxplot. Default labelPlot = 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
...  Arguments to be passed to methods, such as graphical parameters (see par).

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, lsmj

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 matrices. 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

See Also

lsm, simulatelSM, plot.gofobj, print.gofobj

Examples

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

\[
\text{lsjm}(y, \sigma, \xi, \psi_\alpha, niter, tol, \text{preit, randomZ})
\]

Arguments

- \(Y\) list containing a \((n \times n)\) binary adjacency matrix for each network view.
- \(D\) integer dimension of the latent space
- \(\sigma\) \((d \times d)\) 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 \(\xi = 0\)
- \(\psi_\alpha\) vector of variances of the prior distributions of \(\alpha\). Default \(\psi_\alpha = 2\)
- \(niter\) maximum number of iterations. Default \(niter = 500\)
- \(tol\) desired tolerance. Default \(tol = 0.1^2\)
- \(\text{preit}\) Preliminary number of iterations default \(\text{preit} = 20\)
- \(\text{randomZ}\) logical; If \(\text{randomZ} = \text{TRUE}\) random initialization for the latent positions is used. If \(\text{randomZ} = \text{FALSE}\) and \(D = 2\) or \(D = 3\) the latent positions are initialized using the Fruchterman-Reingold method and multidimensional scaling is used for \(D = 1\) or \(D > 3\). Default \(\text{randomZ} = \text{FALSE}\)

@return List containing:
- \(\mathbf{EZ}\) \((n \times D)\) matrix containing the posterior means of the latent positions
- \(\mathbf{VZ}\) \((D \times D)\) matrix containing the posterior variance of the latent positions
- \(1\text{smEZ}\) list containing a \((n \times D)\) matrix for each network view containing the posterior means of the latent positions under each model in the latent space.
- \(1\text{smVZ}\) list containing a \((D \times D)\) matrix for each network view containing the posterior variance of the latent positions under each model in the latent space.
- \(\mathbf{xiT}\) vector of means of the posterior distributions of \(\alpha\)
- \(\psi_\alpha T\) vector of variances of the posterior distributions of \(\alpha\)
- \(\text{Ell}\) expected log-likelihood
References


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)
```

### lsm

**Latent Space Model**

**Description**

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 α. Default xi = 0
psi2 variance of the prior distribution of α. 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 @return 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
  • xIt mean of the posterior distribution of α
  • psi2T variance of the posterior distribution of α
  • Ell expected log-likelihood

References


See Also

plot.lsm

Examples

### Simulate Undirected Network
N <- 20
Y <- network(N, directed = FALSE)[,]

modLSM <- lsm(Y, D = 2)
plot(modLSM, Y)
### 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, 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, mainZtilde = NULL, arrowhead = FALSE, curve = NULL,
xlim = NULL, ylim = NULL, main = NULL, ...)
```
plot.lsjm

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)
```

## Notes

- **lsvm**: Library for simulating and analyzing directed and undirected networks.
- **network**: Function for creating and manipulating networks.
- **plot**: Function for plotting networks.
- **sample**: Function for generating random samples from a given distribution.
- **network**: Function for creating and manipulating networks.
- **plot**: Function for plotting networks.
- **sample**: Function for generating random samples from a given distribution.
plot.lsm

Two dimensional plot of the Latent Space Model output

Description

Function to plot an object of class 'lsm'

Usage

```r
## 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, 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

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

modLSM <- lsm(Y, D = 2)
plot(modLSM, Y)

# Plot with 95% CB
plot(modLSM, Y, drawCB = TRUE)
# Plot with 99% CB
plot(modLSM, Y, drawCB = TRUE, LEVEL = .99)
```
**plotY**  
*Plot the adjacency matrix of the network*

---

**Description**

Function to plot the adjacency matrix of the network.

**Usage**

```r
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`).
PPIgen

**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)
```

---

**PPIgen**  

**PPI genetic interactions**

**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/).

**Format**

Binary adjacency matrix

**Details**

Binary adjacency matrix containing genetic interactions between 67 proteins.

**References**


**See Also**

PPIphy
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/

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
http://thebiogrid.org/

References

See Also
ppigen, ppiphy

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/

Format
Binary adjacency matrix
Details

Binary adjacency matrix containing physical interactions between 67 proteins.

References


See Also

PPIgen

print.gofobj

Print GoF object

Description

Function to print an object of class 'gofobj'

Usage

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

Arguments

x          object of class 'gofobj'

...         other

Examples

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**

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

**Arguments**

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

**Value**

rotated object `Xrot`, and the rotation matrix `R`

---

simulatelsm  

*Simulate from LSM model*

**Description**

Function to simulate networks from the LSM model

**Usage**

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

**Arguments**

- **object**: object of class `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 = NULL`
Examples

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

*Topic **datasets**
  - PPIgen, 12
  - PPInet, 13
  - PPIphy, 13

boxroc, 2

gof1sm, 4

lsjm, 2, 3, 5
lsm, 2–4, 6
lvm4net (lvm4net-package), 2
lvm4net-package, 2

par, 3, 9–11
plot.gofobj, 4, 8
plot.lsjm, 8
plot.lsm, 7, 10
plotY, 11
PPIgen, 12, 13, 14
PPInet, 13
PPIphy, 12, 13, 13
print.gofobj, 4, 14

rotXtoY, 15

simulateLSM, 4, 15