Package 'mixggm'

November 14, 2018

Version 1.0

Date 2018-11-02

Title Mixtures of Gaussian Graphical Models

Description Mixtures of Gaussian graphical models for model-based clustering with sparse covariance and concentration matrices. See Fop, Murphy, and Scrucca (2018) <doi:10.1007/s11222-018-9838-y>.

Maintainer Michael Fop <michael.fop@ucd.ie>

Depends R (>= 3.3)

Imports foreach, GA (>= 3.1), mclust (>= 5.4), memoise, network, Rcpp

LinkingTo Rcpp, RcppArmadillo

License GPL (>= 2)

Repository CRAN

URL https://github.com/michaelfop/mixggm

BugReports https://github.com/michaelfop/mixggm/issues

ByteCompile true

LazyLoad yes

NeedsCompilation yes

Author Michael Fop [aut, cre] (<https://orcid.org/0000-0003-3936-2757>),
Luca Scrucca [ctb] (<https://orcid.org/0000-0003-3826-0484>),
Thomas Brendan Murphy [ctb] (<https://orcid.org/0000-0002-5668-7046>)

Date/Publication 2018-11-14 11:20:02 UTC

R topics documented:

`mixggm-package` *Mixtures of Gaussian graphical models*

Description

Mixtures of Gaussian graphical models for model-based clustering with sparse covariance and concentration matrices.

Details

A package implementing mixtures of Gaussian graphical models for model-based clustering with sparse covariance and concentration matrices. Model fitting is carried out by means of a structural-EM algorithm for parameter estimation and graph structure search.

The main functions are `mixGGM`, `searchGGM` and `fitGGM`.

Author(s)

Michael Fop, Luca Scrucca and Thomas Brendan Murphy.

Maintainer: Michael Fop <michael.fop@ucd.ie>

References

Fop, M., Murphy, T.B., and Scrucca, L. (2018). Model-based clustering with sparse covariance matrices. *Statistics and Computing*. To appear.

`control-parameters` *Set control parameters for various purposes*

Description

Set control parameters for graphical model estimation, graph structure search via stepwise or genetic algorithm, mixture model fitting via structural-EM algorithm, and Bayesian regularization.

Usage

```
ctrlICF(tol = 1e-04, maxiter = 1e03)

ctrlSTEP(occamAdd = Inf, occamRem = Inf, start = NULL)

ctrlGA(popSize = 50, pcrossover = 0.8, pmutation = 0.1,
       maxiter = 100, run = maxiter/2,
       elitism = base::max(1, round(popSize*0.05)))

ctrlEM(tol = 1e-05, maxiter = 1e02, subset = NULL, printMsg = FALSE)
```

```
ctrlREG(data, K, scaleType = c("full", "fixed", "one", "diag"),
        scale = NULL, psi = NULL)
```

Arguments

`tol`	Tolerance value for judging when convergence has been reached. Used in estimation of a Gaussian graphical model and in the structural-EM algorithm.
`maxiter`	Maximum number of iterations in the Gaussian graphical model estimation algorithm, the genetic algorithm for structure search, and the structural-EM algorithm.
`occamAdd, occamRem`	Set the bounds of the Occam's window for stepwise search. See "Details". Default is `Inf`, corresponding to the case of no deletion of candidate graph structures through the search.
`start`	Provide in input a user-defined starting adjacency matrix for stepwise structure search.
`popSize`	Population size. This number corresponds to the number of different graph structures to be considered at each iteration of the genetic algorithm.
`run`	Number of consecutive generations without any improvement in the best fitness value of the structure search procedure before the genetic algorithm is stopped.
`pcrossover`	Probability of crossover between pairs of binary adjacency matrices.
`pmutation`	Probability of mutation in a parent adjacency matrix.
`elitism`	Number of best fitness graph structures to survive at each iteration of the genetic algorithm in the graph structure search procedure.
`subset`	A logical or numeric vector specifying a subset of the data to be used in the initial hierarchical clustering phase employed in the initialization of the structural-EM algorithm. By default no subset is used.
`printMsg`	A logical value indicating whether or not certain warnings (usually related to singularity) should be issued.
`data`	A matrix or data frame of observations. Categorical variables are not allowed. Rows correspond to observations and columns correspond to variables.
`K`	The number of mixture components.
`scaleType`	The type of scale hyperparameter for the prior on the covariance matrix in the case of Bayesian regularization for Gaussian covariance graph model. See "Details". Default is `"full"`.
`scale`	The scale hyperparameter for the prior on the covariance matrix in the case of Bayesian regularization for Gaussian covariance graph model.
`psi`	The degrees of freedom hyperparameter for the prior on the covariance matrix in the case of Bayesian regularization for Gaussian covariance graph model.

Details

Function `ctrlICF` is used to set control parameters of the algorithms employed to estimate a Gaussian covariance or concentration graph model.

Function `ctrlSTEP` mainly controls the Occam's window used in the stepwise graph structure search. Default is `Inf`, corresponding to no Occam's window reduction is implemented. The rationale of the Occam's window is too reduce the space of candidate adjacency matrices during the search by discarding those with a value of the penalized objective function that is too distant from the current optimal value. Graph candidate structures whose penalized objective function value is within `occamAdd` from the current optimal value are considered in the next edge-add step of the search. Likewise, graph candidate structures whose penalized objective function value is within `occamRem` from the current optimal value are considered in the next edge-remove step of the search. Small values for `occamRem` and `occamAdd` significantly reduce the space of candidate solutions and the computational cost of the greedy search.

Function `ctrlGA` sets parameters of the genetic algorithm used for graph structure search. Arguments correspond to those of function `ga` in the `GA` package.

Function `ctrlEM` controls standard parameters of the structural-EM algorithm.

Function `ctrlREG` is used to set hyperparameters of the conjugate prior on the covariance matrix for Bayesian regularization in Gaussian covariance graph models. The function creates a list of hyperparameters to be given in input in argument `regHyperPar` of functions `fitGGM`, `searchGGM` and `mixGGM`. If not provided in input, the scale hyperparameter is computed on the data via the sample covariance matrix according to the argument `scaleType`. If `scaleType = "full"`, the scale matrix is proportional to the data covariance matrix; if `scaleType = "fixed"`, the scale matrix has determinant equal to $(\frac{0.001}{K})^{(1/V)}$ (Baudry, Celeux, 2015; Fop et al. 2018); if `scaleType = "one"` the scale matrix has determinant 1; if `scaleType = "diag"` the scale matrix is diagonal. Note that in the case `V > N`, if not provided in input the scale matrix is forced to be diagonal. The hyperparameter `psi` controlling the degrees of freedom is set to `V + 2` by default.

Value

A list of parameters values.

References

Baudry, J.P. and Celeux, G. (2015). EM for mixtures: Initialization requires special care. *Statistics and Computing*, 25(4):713-726.

Fop, M., Murphy, T.B., and Scrucca, L. (2018). Model-based clustering with sparse covariance matrices. *Statistics and Computing*. To appear.

Examples

```
## Not run:

# ga search with increased mutation probability
data(banknote, package = "mclust")
mod1 <- searchGGM(banknote[,-1], model = "concentration", search = "ga",
                  ctrlGa = ctrlGA(pmutation = 0.3))
```

```
# regularization
library(MASS)
V <- 10
N <- 20
mu <- rep(0, V)
sigma <- matrix(0.9, V,V)
diag(sigma) <- 1
x <- cbind( MASS::mvrnorm(N, mu, sigma),
            MASS::mvrnorm(N, mu, sigma),
            MASS::mvrnorm(N, mu, sigma))  # high-dimensional data V = 30, N = 20
#
hyperPar <- ctrlREG(x, K = 1, scaleType = "diag")
mod2 <- searchGGM(x, model = "covariance", penalty = "ebic")   # throws an error
mod2 <- searchGGM(x, model = "covariance", penalty = "ebic",   # regularization
                  regularize = TRUE, regHyperPar = hyperPar)
plot(mod2, "adjacency")


# occam's window
library(MASS)
V <- 20
N <- 500
mu <- rep(0, V)
sigma <- matrix(0.9, V,V)
diag(sigma) <- 1
edges <- rbinom(choose(V,2), 1, 0.3)
A <- matrix(0, V,V)
A[lower.tri(A)] <- edges
A <- A + t(A)
fit <- fitGGM(S = sigma, N = N, graph = A, model = "concentration",
              ctrlIcf = ctrlICF(tol = 1e-06))
sigma <- fit$sigma
#
x <- MASS:::mvrnorm(N, mu, sigma)
#
mod3 <- searchGGM(x, model = "concentration", search = "step-back",
                  ctrlStep = ctrlSTEP(occamAdd = 5, occamRem = 5))
par(mfrow = c(1,2))
plot(fit, what = "adjacency")
plot(mod3, what = "adjacency")


## End(Not run)
```

`fitGGM` *Fit a Gaussian graphical model*

Description

Estimation of a Gaussian graphical model given the graph structure corresponding to marginal or conditional independence restrictions.

Usage

```
fitGGM(data = NULL,
       S = NULL, N = NULL,
       graph,
       model = c("covariance", "concentration"),
       start = NULL,
       ctrlIcf = ctrlICF(),
       regularize = FALSE,
       regHyperPar = NULL,
       verbose = FALSE, ...)
```

Arguments

data	A dataframe or matrix, where rows correspond to observations and columns to variables. Categorical variables are not allowed.
S	The sample covariance matrix of the data. If S = NULL, the maximum likelihood estimate of the covariance matrix is used in the estimation of the graphical model.
N	The number of observations. If data = NULL and S is provided in input, N must be provided in input as well.
graph	A square symmetric binary adjacency matrix corresponding to the association structure of the graph. See "Details".
model	The type of Gaussian graphical model. Default is "covariance". See "Details".
start	A starting matrix for the estimation algorithm. If NULL, the starting value is the diagonal sample covariance matrix. Used only when model = "covariance".
ctrlIcf	A list of control parameters for the numerical algorithm for estimation of graphical model parameters; see also ctrlICF.
regularize	A logical argument indicating if Bayesian regularization should be performed. Default to FALSE. Used only when model = "covariance".
regHyperPar	A list of hyper parameters for Bayesian regularization. Only used when regularization = TRUE; see also ctrlREG.
verbose	A logical argument controlling whether iterations of the estimation procedure need to be shown or not.
...	Additional internal arguments not to be provided by the user.

Details

The function estimates a Gaussian graphical model given the graph association structure provided in input by the binary adjacency matrix. In the adjacency matrix, a zero entry corresponds to two variables being independent, marginally or conditionally according to the model.

If `model = "covariance"`, a **Gaussian covariance graph model** is estimated, and the joint distribution of the V dimensional vector of variables X is parameterized in terms of the covariance matrix Σ. It is assumed:

$$X \sim \mathcal{N}(\mu, \Sigma) \quad \Sigma \in C_G^+(A)$$

where $C_G^+(A)$ is the collection of sparse positive definite matrices whose zero patterns are given by graph G represented by the adjacency matrix A. In this type of model, the graph/adjacency matrix corresponds to *marginal independence* constraints among the variables, i.e. the variables associated to two non-connected edges in the graph are marginally independent. As a result, the covariance matrix `sigma` is estimated to be sparse according to the graph.

If `model = "concentration"`, a **Gaussian concentration graph model** is estimated, and the joint distribution of the V dimensional vector of variables X is parameterized in terms of the concentration matrix (inverse covariance or precision matrix) Ω. It is assumed:

$$X \sim \mathcal{N}(\mu, \Omega) \quad \Omega \in C_G^+(A)$$

where $C_G^+(A)$ is the collection of sparse positive definite matrices whose zero patterns are given by graph G embedded in the adjacency matrix A. For this type of model, the graph/adjacency matrix corresponds to *conditional independence* constraints among the variables, i.e. the variables associated to two non-adjacent edges in the graph are conditionally independent given their common neighbors. It results in the concentration matrix `omega` being estimated to be sparse according to the structure of the graph.

Note that conditional independence does not imply marginal independence, and marginal independence does not imply conditional independence, therefore a sparse concentration matrix and a sparse covariance matrix do not necessarily match; See Whittaker (1990).

The Gaussian covariance graph model is estimated using the *Iterative Conditional Fitting* algorithm by Chaudhuri et al. (2007), while the Gaussian concentration graph model is estimated using the algorithm by Hastie et al. (2009).

Bayesian regularization is performed by means of a conjugate prior on the covariance/concentration matrix, similarly to what described in Fop et al. (2018). In the case of covariance graph model, an Inverse-Wishart distribution is used as a prior for Σ, while a Wishart distribution is used for Ω in the case of a concentration graph model. Regularization can be useful when the number of variables is larger than the number of observations.

Value

An object of class `'fitGGM'` containing the estimated Gaussian graphical model.

The output is a list containing:

`sigma`	The estimated covariance matrix.
`omega`	The estimated concentration (inverse covariance) matrix.
`graph`	The adjacency matrix given in input corresponding to the marginal or conditional independence graph.
`model`	Estimated model type, whether `"covariance"` or `"concentration"`.
`loglik`	Value of the maximized log-likelihood.
`nPar`	Number of estimated parameters.
`N`	Number of observations.

V	Number of variables, corresponding to the number of nodes in the graph.
iter	Number of iterations for the algorithm to converge.

References

Chaudhuri, S., Drton M., and Richardson, T. S. (2007). Estimation of a covariance matrix with zeros. *Biometrika*, 94(1), 199-216.

Fop, M., Murphy, T.B., and Scrucca, L. (2018). Model-based clustering with sparse covariance matrices. *Statistics and Computing*. To appear.

Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer.

Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics*. Wiley.

See Also

plot.fitGGM

Examples

```
# Gaussian covariance graph model
data(mtcars)
x <- mtcars[,c(1,3:7)]
R <- cor(x)
#
# model where variables with correlation less than 0.5 are marginally independent
graph <- ( abs(R) < 0.5 )*1
diag(graph) <- 0
fit1 <- fitGGM(data = x, graph = graph)
plot(fit1)


# Gaussian concentration graph model
data(swiss)
#
# fit a conditional independence model:
V <- ncol(swiss)
graph <- matrix( c(0,1,0,1,1,1,
                   1,0,1,1,0,0,
                   0,1,0,1,1,0,
                   1,1,1,0,1,0,
                   1,0,1,1,0,0,
                   1,0,0,0,0,0), V,V, byrow = TRUE )
fit2 <- fitGGM(swiss, graph = graph, model = "concentration")
plot(fit2)


## Not run:

data(marks, package = "ggm")
#
# the conditional independence model of Whittaker (1990), pag. 6
```

```
V <- ncol(marks)
graph <- matrix( c(0,1,1,0,0,
                   1,0,1,0,0,
                   1,1,0,1,1,
                   0,0,1,0,1,
                   0,0,1,1,0), V,V, byrow = TRUE )
fit3 <- fitGGM(marks, graph = graph, model = "concentration")
plot(fit3)

## End(Not run)
```

`mixGGM` *Mixture of Gaussian graphical models*

Description

Estimation of a mixture of Gaussian covariance or concentration graph models using structural-EM algorithm. The mixture model returned is the optimal model according to BIC.

Usage

```
mixGGM(data, K = 1:3,
       model = c("covariance", "concentration"),
       search = c("step-forw", "step-back", "ga"),
       penalty = c("bic", "ebic", "erdos", "power"),
       beta = NULL,
       regularize = FALSE, regHyperPar = NULL,
       ctrlEm = ctrlEM(),
       ctrlStep = ctrlSTEP(), ctrlGa = ctrlGA(),
       ctrlIcf = ctrlICF(),
       keepAll = FALSE,
       parallel = FALSE,
       verbose = TRUE)
```

Arguments

`data`	A dataframe or matrix, where rows correspond to observations and columns to variables. Categorical variables are not allowed.
`K`	An integer vector specifying the numbers of mixture components (clusters) for which the BIC is to be calculated.
`model`	The type of Gaussian graphical model. Default is `"covariance"`. See "Details".
`search`	The type of structure search algorithm. If `search = "step-forw"`, a greedy forward-stepwise search is used to find the optimal graph association structure. If `search = "step-back"`, a greedy backward-stepwise search is implemented. If `search = "ga"` a stochastic search based on a genetic algorithm is employed. Default is `"step-forw"`.

penalty	The penalty function used to define a criterion for scoring the candidate graph configurations. Default is "bic". See "Details" and penalty.
beta	The hyperparameter of the penalty function. See "Details" and penalty.
regularize	A logical argument indicating if Bayesian regularization should be performed. Default to FALSE. Used only when model = "covariance".
regHyperPar	A list of hyper parameters for Bayesian regularization. Only used when regularization = TRUE; see also ctrlREG.
ctrlEm	A list of control parameters used in the structural-EM algorithm; see also ctrlEM.
ctrlStep	A list of control parameters used in the stepwise search; see also ctrlSTEP.
ctrlGa	A list of control parameters for the genetic algorithm; see also ctrlGA.
ctrlIcf	A list of control parameters employed in the algorithm for estimation of graphical model parameters; see also ctrlICF.
keepAll	A logical argument. If TRUE, also all the mixture models estimated for the values of K given in input are returned
parallel	A logical argument indicating if parallel computation should be used for structure search in the M step of the structural-EM algorithm. If TRUE, all the available cores are used. The argument could also be set to a numeric integer value specifying the number of cores to be employed.
verbose	If TRUE a progress bar will be shown.

Details

Estimation of a mixture of Gaussian graphical models by means of maximization of a penalized log-likelihood via structural-EM algorithm. The mixture model in output is the optimal model selected by BIC.

If model = "covariance", a **mixture of Gaussian covariance graph models** is estimated. The Gaussian mixture is parameterized in terms of the components covariance matrices and the component adjacency matrices correspond to *marginal independence* constraints among the variables:

$$X \sim \sum_{k}^{K} \tau_k \mathcal{N}(\mu_k, \Sigma_k) \quad \Sigma_k \in C_G^+(A_k)$$

Variables associated to two non-connected edges in the graphs are marginally independent and have different marginal association patterns across the mixture components. As a result, the covariance matrices sigma are estimated to be sparse according to the inferred graph structures.

If model = "concentration", estimation of a **mixture of Gaussian concentration graph model** is performed. The Gaussian mixture is parameterized in terms of the components concentration matrices and the component adjacency matrices correspond to *conditional independence* constraints among the variables:

$$X \sim \sum_{k}^{K} \tau_k \mathcal{N}(\mu_k, \Omega_k) \quad \Omega_k \in C_G^+(A_k)$$

Variables associated to two non-adjacent edges in the graph are conditionally independent given their common neighbors and have different conditional dependence patterns across the mixture

components. It results in the concentration matrices omega being estimated to be sparse according to the inferred graph structures.

Arguments penalty and search are used to define the type of penalty on the graph configuration and the structure search method in the structural-EM algorithm. The penalization term depends on the hyperparameter beta according to the type of penalty function. See searchGGM and penalty for more details.

Value

An object of class 'mixGGM' containing the optimal estimated mixture of Gaussian graphical models.

The output is a list containing:

parameters	A list with the following components: tau A vector containing the estimated mixing proportions. mu The mean for each mixture component. Columns denote the mixture components. sigma An array containing the components covariance matrices. omega An array containing the components concentration (inverse covariance) matrices.
graph	An array with the adjacency matrices corresponding to the optimal marginal or conditional independence graphs for each mixture component.
N	Number of observations in the data.
V	Number of variables in the data, corresponding to the number of nodes in the graphs.
K	Number of selected mixture components.
loglik	Value of the maximized log-likelihood.
loglikPen	Value of the maximized penalized log-likelihood.
loglikReg	Value of the maximized regularized log-likelihood. If regularize = FALSE, this value is equal to loglik
nPar	A vector with two entries: depPar Total number of dependence parameters. If model = "covariance", this is the total number of non-zero covariance parameters, while if model = "concentration", it corresponds to the total number of non-zero concentration parameters. totPar Total number of mixture parameters.
z	A matrix whose [i,k]th entry is the probability that observation i of the data belongs to the kth class.
classification	Classification corresponding to the maximum a posteriori of matrix z.
bic	Optimal BIC value.
BIC	All BIC values.
data	The data matrix provided in input.
model	Estimated model type, whether "covariance" or "concentration".

penalty	The type of penalty on the graph structure.
search	The search method used for graph structure search.
keepAll	A list containing all the estimated models. Provided in output only when keepAll = TRUE.

References

Fop, M., Murphy, T.B., and Scrucca, L. (2018). Model-based clustering with sparse covariance matrices. *Statistics and Computing*. To appear.

Examples

```
# fit a mixture of concentration graph models
data(iris)
mod1 <- mixGGM(iris[,-5], model = "concentration")
plot(mod1, what = "graph")
plot(mod1, what = "classification")


## Not run:

# a simple simulated data example
library(MASS)
N <- 200
tau <- c(0.3, 0.7)
Nk <- rowSums( rmultinom(N, 1, tau) )
class <- rep(1:2, Nk)
sigma1 <- diag(2)                          # independent variables
sigma2 <- matrix( c(1,0.9,0.9,1), 2,2 )    # correlated variables
mu1 <- c(0, 0)
mu2 <- c(2, 3)
x <- rbind( MASS::mvrnorm(Nk[1], mu1, sigma1),
            MASS::mvrnorm(Nk[2], mu2, sigma2)
)
mod2 <- mixGGM(x)
plot(mod2)
plot(mod2, what = "classification")

# fit a mixture of covariance graph models
data(wine, package = "gclus")
mod3 <- mixGGM(wine[,-1], K = 1:4, model = "covariance",
               penalty = "erdos", beta = 0.01)
plot(mod3, what = "graph")
plot(mod3, what = "classification", dimens = 1:4)


# complex simulated data example
N <- 500
V <- 20
tau <- c(0.3, 0.7)
Nk <- rowSums( rmultinom(N, 1, tau) )
class <- rep(1:2, Nk)
```

```
sigma1 <- rWishart(1, V+1, diag(V))[,,1]
mu1 <- rep(0, V)
mu2 <- rnorm(V, 0.5, 2)
x1 <- MASS::mvrnorm(Nk[1], mu1, sigma1)
x2 <- matrix(NA, Nk[2], V)
x2[,1] <- rnorm(Nk[2])
for ( j in 2:V ) x2[,j] <- x2[,j-1] + rnorm(Nk[2], mu2[j], sd = 0.5)
x <- rbind(x1, x2)
#
mod4 <- mixGGM(x, K = 1:4, model = "concentration",
               penalty = "ebic", beta = 0.5)
plot(mod4, what = "classification", dimens = c(1,5,10,15,20) )
plot(mod4, what = "graph")
plot(mod4, what = "adjacency")
table(class, mod4$classification)
#
mc <- mclust::Mclust(x, G = 1:4)
mc$bic
mod4$bic


## End(Not run)
```

`penalty` *Penalty functions for graph structure search*

Description

Collection of penalty functions employed in the structural-EM algorithm and penalized maximum likelihood estimation for graph structure search.

Details

The choice of the penalty function is via argument `penalty` in the functions `searchGGM` and `mixGGM`. Possible options are `"bic"` (default), `"ebic"`, `"erdos"`, and `"power"`. Functions `"ebic"`, `"erdos"`, and `"power"` depend also on a hyperparameter `beta` which can be set using the corresponding argument in `searchGGM` and `mixGGM`.

Let denote with `E` the number of non-zero entries in the adjacency matrix corresponding to the graph structure of a covariance/concentration graph model (i.e. the number of edges); `N` and `V` denote number of observations and number of variables (or nodes). The above options correspond to the following penalty functions:

- `"bic"` – A BIC-like penalty term is placed on the structure of a graph. This penalty is given by:

$$0.5E \log N$$

  The hyperparameter `beta` is not used.

- "ebic" – An EBIC-like penalty term for graphical models is placed on the structure of a graph. This penalty is given by:

$$0.5E \log N + 2\beta E \log V$$

For this penalty function, beta is a value in the range [0,1]. Default is beta = 1, encouraging sparser models. Clearly the case beta = 0 corresponds to "bic".

- "erdos" – Let denote by T the number of all possible edges in a graph, i.e. $T = \binom{V}{2}$. The penalty function is given by:

$$-E \log \beta - (T - E) \log(1 - \beta)$$

For this penalty function, beta is a value in the range (0,1). For small values of beta the penalization tends to favor situations where the graph decomposes into disjoint blocks. Default is beta = log(V)/T, a value for which the expected number of arcs is equal to log(V) and such that the graph will almost surely have disconnected components.

- "power" – Let denote with d_j the degree of node j, i. e. the number of nodes connected to it. This penalty function is defined as:

$$\beta \sum_{j}^{V} \log(d_j + 1)$$

In this case, beta is a positive value. Default is beta = log(NV), a value which place the penalty term on a similar magnitude of "bic" and "ebic", but denser graphs will tend to be less penalized.

An user-defined penalty function can be also provided in input of argument penalty in the functions searchGGM and mixGGM. In this case, the penalty must be an object of class "function" and have as argument graph, like for example "f <- function(graph, beta)"; see "Examples".

See also searchGGM and mixGGM for some examples.

References

Fop, M., Murphy, T.B., and Scrucca, L. (2018). Model-based clustering with sparse covariance matrices. *Statistics and Computing*. To appear.

Examples

```
# fit concentration graph model with power law penalty
data(ability.cov)
N <- ability.cov$n.obs
mod1 <- searchGGM(S = ability.cov$cov, N = ability.cov$n.obs,
                  model = "concentration", penalty = "power", beta = 2*log(N))
mod1
plot(mod1)


## Not run:
```

```
# two disconnected blocks of correlated variables
library(MASS)
V <- 10
N <- 500
mu <- rep(0, V)
sigma <- matrix(0.9, V,V)
diag(sigma) <- 1
x <- cbind( MASS::mvrnorm(N, mu, sigma),
            MASS::mvrnorm(N, mu, sigma) )
#
# fit a covariance graph with erdos penalty
mod2 <- searchGGM(x, model = "covariance",
                  penalty = "erdos")
plot(mod2, "adjacency")


# user defined penalty function
data(iris)
x <- iris[,-5]
N <- nrow(x)
V <- ncol(x)
ref <- matrix(0, V, V)
#
# penalize graphs different from a reference graph structure
myPenalty <- function(graph, beta)
{
  beta * sum( abs(graph - ref) )
}
#
mod3 <- mixGGM(x, K = 3, model = "covariance",
               penalty = myPenalty, beta = 2*V*log(N))
plot(mod3)


## End(Not run)
```

`plotting-functionalities` *Plotting functionalities for Gaussian covariance and concentration graph models and their mixture*

Description

Plotting functionalities for objects of class `fitGGM` or `mixGGM`.

Usage

```
## S3 method for class 'fitGGM'
plot(x, what = c("graph", "adjacency"),
```

```
    layout = c("circle", "random"), ...)

## S3 method for class 'mixGGM'
plot(x, what = c("graph", "classification", "adjacency", "common"),
    layout = c("circle", "random"),
    colors = NULL, symb = NULL, dimens = NULL, ...)
```

Arguments

x	An object of class fitGGM or mixGGM.
what	The type of plot to be produced. If what = "graph" (default), the graph(s) corresponding to the association structure(s) among the variables is displayed; if what = "adjacency" and heat-map representing the binary entries of the adjacency matrix is produced; if what = "classification" the function produces a scatterplot showing the clustering of the observations; if what = "common" the graph intersection of the mixture components graphs is produced, which display the edges common across the clusters. See "Details".
layout	Layout of the graph, either circular (default) or random.
colors	A vector of user defined colors
symb	A vector of user defined symbols
dimens	A vector giving the integer dimensions of the desired variables for multivariate data in case of what = "classification".
...	Other arguments.

Details

These functions are used to visualize graph association structures and clustering results for single and mixtures of Gaussian covariance and concentration models.

In the case of what = "graph", the graph of a Gaussian covariance graph model is bi-directed, while the graph of a Gaussian concentration model is un-directed. Thickness of the edges is proportional to the estimated association parameters.

See "Examples" for various cases.

Examples

```
# covariance graph
data(mtcars)
x <- mtcars[,c(1,3:7)]
R <- cor(x)
graph <- ( abs(R) < 0.5 )*1
diag(graph) <- 0
fit1 <- fitGGM(data = x, graph = graph)
plot(fit1)
plot(fit1, what = "adjacency")
```

```
# concentration graph
data(swiss)
V <- ncol(swiss)
graph <- matrix( c(0,1,0,1,1,1,
                   1,0,1,1,0,0,
                   0,1,0,1,1,0,
                   1,1,1,0,1,0,
                   1,0,1,1,0,0,
                   1,0,0,0,0,0), V,V, byrow = TRUE )
fit2 <- fitGGM(swiss, graph = graph, model = "concentration")
plot(fit2)
plot(fit2, layout = "random")
plot(fit2, what = "adjacency")


## Not run:

# mixture of Gaussian concentration graph models
data(banknote, package = "mclust")
mod3 <- mixGGM(banknote[,-1], model = "concentration", K = 2)
plot(mod3, what = "graph")
plot(mod3, what = "adjacency")
plot(mod3, what = "classification")
plot(mod3, what = "classification", dimens = c(1,4,5))
plot(mod3, what = "common")


# mixture of Gaussian covariance graph models
data(wine, package = "gclus")
mod4 <- mixGGM(wine[,-1], model = "covariance", K = 3)
clb <- c("#999999", "#E69F00", "#56B4E9")                # colorblind friendly palette
plot(mod4, what = "graph", colors = clb)
plot(mod4, what = "adjacency", colors = clb)
plot(mod4, what = "classification", colors = clb, dimens = c(1,7,8,12))
plot(mod4, what = "common")


## End(Not run)
```

---

searchGGM *Structure search and estimation for Gaussian graphical models*

Description

Graph structure search and estimation for Gaussian covariance and concentration graph models.

Usage

```
searchGGM(data = NULL,
```

```
S = NULL, N = NULL,
model = c("covariance", "concentration"),
search = c("step-forw", "step-back", "ga"),
penalty = c("bic", "ebic", "erdos", "power"),
beta = NULL,
start = NULL,
regularize = FALSE, regHyperPar = NULL,
ctrlStep = ctrlSTEP(), ctrlGa = ctrlGA(), ctrlIcf = ctrlICF(),
parallel = FALSE,
verbose = FALSE, ...)
```

Arguments

data	A dataframe or matrix, where rows correspond to observations and columns to variables. Categorical variables are not allowed.
S	The sample covariance matrix of the data. If `S = NULL`, the maximum likelihood estimate of the covariance matrix is used in the estimation of the graphical model.
N	The number of observations. If `data = NULL` and `S` is provided in input, `N` must be provided in input as well.
model	The type of Gaussian graphical model. Default is `"covariance"`. See "Details".
search	The type of structure search algorithm. If `search = "step-forw"`, a greedy forward-stepwise search is used to find the optimal graph association structure. If `search = "step-back"`, a greedy backward-stepwise search is implemented. If `search = "ga"` a stochastic search based on a genetic algorithm is employed. Default is `"step-forw"`.
penalty	The penalty function used to define a criterion for scoring the candidate graph configurations. Default is `"bic"`. See "Details" and `penalty`.
beta	The hyperparameter of the penalty function. See "Details" and `penalty`.
start	A starting matrix for the estimation algorithm. If `NULL`, the starting value is the diagonal sample covariance matrix. Used only when `model = "covariance"`.
regularize	A logical argument indicating if Bayesian regularization should be performed. Default to `FALSE`. Used only when `model = "covariance"`.
regHyperPar	A list of hyper parameters for Bayesian regularization. Only used when `regularization = TRUE`; see also `ctrlREG`.
ctrlStep	A list of control parameters used in the stepwise search; see also `ctrlSTEP`.
ctrlGa	A list of control parameters for the genetic algorithm; see also `ctrlGA`.
ctrlIcf	A list of control parameters employed in the algorithm for estimation of graphical model parameters; see also `ctrlICF`.
parallel	A logical argument indicating if parallel computation should be used for structure search. If TRUE, all the available cores are used. The argument could also be set to a numeric integer value specifying the number of cores to be employed.
verbose	A logical argument controlling whether iterations of the structure searching and estimation procedure need to be shown or not.
...	Additional internal arguments not to be provided by the user.

Details

The function performs graph association structure search and maximum penalized likelihood estimation of the optimal Gaussian graphical model given the data provided in input.

A Gaussian covariance graph model is estimated if `model = "covariance"`, while estimation of a Gaussian covariance graph model is performed if `model = "concentration"`. A Gaussian covariance graph model postulates that some variables are marginally independent according to the inferred graph structure. On the other hand, in a Gaussian concentration graph model, variables are conditionally independent given their neighbors in the inferred graph. See also `fitGGM`.

Search for the optimal graph structure and parameter estimation is carried out by maximization of a Gaussian penalized likelihood, given as follows:

$$\text{Covariance:} \quad \text{argmax}_{\Sigma, A} \ \ell(X|\Sigma, A) - P(A, \beta) \quad \Sigma \in C_G^+(A)$$

$$\text{Concentration:} \quad \text{argmax}_{\Omega, A} \ \ell(X|\Omega, A) - P(A, \beta) \quad \Omega \in C_G^+(A)$$

where $C_G^+(A)$ is the collection of sparse positive definite matrices whose zero patterns are given by graph G represented by the adjacency matrix A.

The penalty function $P(A, \beta)$ depends on the structure of graph G through the adjacency matrix A and a parameter β; see `penalty` on how to specify the penalization term and for further information.

For this type of penalized log-likelihood, graph structure search and parameter estimation is a maximization combinatorial problem. For a given candidate structure (i.e. adjacency matrix), association parameters in the covariance or concentration matrix are estimated using the estimation algorithms implemented in `fitGGM`. Regarding structure search, this can be carried out either using a greedy forward-stepwise or a greedy backward-stepwise algorithm, by setting `search = "step-forw"` or `search = "step-back"` respectively. Alternatively, a stochastic search via genetic algorithm can be used by setting `search = "ga"`. The procedure for the forward stepwise search is described in Fop et al. (2018), and the backward is implemented in a similar way; the genetic algorithm procedure relies on the `GA` package. All the structure searching methods can be run in parallel on a multi-core machine by setting the argument `parallel = TRUE`.

Value

An object of class `'fitGGM'` containing the optimal estimated marginal or conditional independence Gaussian graphical model.

The output is a list containing:

`sigma`	The estimated covariance matrix.
`omega`	The estimated concentration (inverse covariance) matrix.
`graph`	The adjacency matrix corresponding to the optimal marginal or conditional independence graph.
`model`	Estimated model type, whether `"covariance"` or `"concentration"`.
`loglikPen`	Value of the maximized penalized log-likelihood.
`loglik`	Value of the maximized log-likelihood.
`nPar`	Number of estimated parameters.
`N`	Number of observations.

V	Number of variables, corresponding to the number of nodes in the graph.
penalty	The type of penalty on the graph structure.
search	The search method used for graph structure search.
GA	An object of class 'ga-class' with information about the genetic algorithm. Only present when search = "ga". See ga.

References

Fop, M., Murphy, T.B., and Scrucca, L. (2018). Model-based clustering with sparse covariance matrices. *Statistics and Computing*. To appear.

Scrucca, L. (2017). On some extensions to GA package: Hybrid optimisation, parallelisation and islands evolution. *The R Journal*, 9(1), 187-206.

Scrucca, L. (2013). GA: A package for genetic algorithms in R. *Journal of Statistical Software*, 53(4), 1-3.

Examples

```
# fit covariance graph model with default forward-stepwise search
data(mtcars)
x <- mtcars[,c(1,3:7)]
mod1 <- searchGGM(x, model = "covariance")
mod1
plot(mod1)
#
# prefer a sparser model
mod2 <- searchGGM(x, model = "covariance", penalty = "ebic")
mod2
plot(mod2)


# fit concentration graph model with backward-stepwise structure search
# with a covariance matrix in input
data(ability.cov)
mod3 <- searchGGM(S = ability.cov$cov, N = ability.cov$n.obs,
                  model = "concentration", search = "step-back")
mod3
mod3$graph
mod3$omega
plot(mod3)

## Not run:

# generate data from a Markov model
N <- 1000
V <- 20
dat <- matrix(NA, N, V)
dat[,1] <- rnorm(N)
for ( j in 2:V ) dat[,j] <- dat[,j-1] + rnorm(N, sd = 0.5)
mod4 <- searchGGM(data = dat, model = "concentration")        # recover the model
```

```
plot(mod4, what = "adjacency")


## End(Not run)
```

Index