Package 'cglasso'

July 21, 2020

Version 1.1.2

Date 2020-07-20

Type Package

Title L1-Penalized Censored Gaussian Graphical Models

Author Luigi Augugliaro

Maintainer Luigi Augugliaro <luigi.augugliaro@unipa.it>

Depends R (>= 3.4), igraph

Description The l1-penalized censored Gaussian graphical model is an extension of the graphical lasso estimator developed to handle datasets with censored observations. An EM-like algorithm is implemented to estimate the parameters of the censored Gaussian graphical models.

Imports methods, MASS

License GPL (>= 2)

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-07-21 07:10:02 UTC

R topics documented:

`cglasso-package` *L1-Penalized Censored Gaussian Graphical Model*

Description

The ℓ_1-penalized censored Gaussian graphical model (Augugliaro *and other*, 2018) is an extension of the graphical lasso estimator (Yuan *and other*, 2007) developed to handle datasets from a censored Gaussian graphical model. An EM-like algorithm is implemented to fit the model. The graphical lasso algorithm (Friedman *and other*, 2008) is used to solve the maximization problem in the M-step.

Details

Package:	cglasso
Type:	Package
Version:	1.1.2
Date:	2020-07-20
License:	GPL (>=2)

Author(s)

Luigi Augugliaro
Maintainer: Luigi Augugliaro <luigi.augugliaro@unipa.it>

References

Augugliaro, L., Abbruzzo, A., and Vinciotti, V. (2018) <DOI:10.1093/biostatistics/kxy043>. ℓ_1-Penalized gaussian graphical model. *Biostatistics* (to appear).

Friedman, J.H., Hastie, T., and Tibshirani, R. (2008) <DOI:10.1093/biostatistics/kxm045>. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* **9**, 432–441.

Yuan, M., and Lin, Y. (2007) <DOI:10.1093/biomet/asm018>. Model selection and estimation in the Gaussian graphical model. *Biometrika* **94**, 19–35.

aic *Akaike's An Information Criterion*

Description

'aic' computes the 'Akaike Information Criterion' whereas 'bic' computes the 'Bayesian Information Criterion'.

Usage

```
aic(object, k = 2)

bic(object)
```

Arguments

object	an object with class 'glasso', 'ggm', 'mglasso' or 'mggm' 'cglasso' or 'cggm'.
k	the *penalty* per parameter to be used; the default k = 2 is the classical AIC.

Details

The measure of goodness-of-fit (gof) returned by the functions 'aic' and 'bic' depends on the class of the fitted model.

If 'object' has class 'glasso' or 'ggm', then 'aic' computes the following measure of goodness-of-fit:

$$-2\,\text{log-likelihood} + k\,\text{df},$$

where k is the *penalty* per parameter and df represents the number of parameters in the fitted model. The values of the log-likelihood function are computed using the function loglik. The usual Akaike Information Criterion (AIC) is computed letting $k = 2$ (default value of the function 'aic') whereas the 'Bayesian Information Criterion' (BIC) is computed letting $k = \log(n)$, where n is the sample size.

If 'object' has class 'mglasso' or 'mggm' 'cglasso' or 'cggm', then 'aic' computes the following measure of goodness-of-fit:

$$-2\,Q\text{-function} + k\,df,$$

in other words the log-likelihood is replaced with the Q-function maximized in the M-step of the EM-like algorithm describted in cglasso, mglasso and mle. This measure of goodness-of-fit was proposed in Ibrahim *and others* (2008) for statistical model with missing-data.

'aic' and 'bic' return an object with S3 class 'gof' for which are available the method functions 'print.gof' and 'plot.gof'. These method functions are developed with the aim of helping the user in finding the optimal value of the tuning parameter, defined as the ρ-value minimizing the chosen measure of goodness-of-fit. For this reason, 'print.gof' shows also the ranking of the fitted models (the best model is pointed out with an arrow) whereas 'plot.gof' point out the optimal ρ-value by a vertical dashed line (see below for some examples).

Value

'aic' and 'bic' return an object with S3 class "gof", i.e. a list containing the following components:

value_gof	the values of the measure of goodness-of-fit used to evaluate the fitted models.
rho	the values of the tuning parameter used to fit the model.
value	the values of the log-likelihood function or the Q-function.
df	the number of the estimated non-zero parameters, i.e. the number of non-zero partial correlations plus $2p$.
n	the sample size.
p	the number of variables.
model	the name of the fitted models.
type	the measure of goodness-of-fit used to evaluate the fitted models.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

References

Ibrahim, J.G., Zhu, H. and Tang, N. (2008). Model selection criteria for missing-data problems using the EM algorithm. *Journal of the American Statistical Association* **103**, 1648–1658.

Sakamoto, Y., Ishiguro, M., and Kitagawa, G. (1986). *Akaike Information Criterion Statistics*. D. Reidel Publishing Company.

See Also

loglik, cglasso, mglasso, glasso, mle, ebic and the method funtions 'plot' and summary.

Examples

```
library("cglasso")
set.seed(123)

#################
# cglasso model #
#################
n <- 100L
p <- 5L
mu <- rep.int(0L, times = p)
X <- rdatacggm(n = n, mu = mu, probr = 0.05)
out <- cglasso(X = X)
out_aic <- aic(out)
out_aic
plot(out_aic)

out_bic <- bic(out)
out_bic
plot(out_bic)
```

```
##############
# cggm model #
##############
out_mle <- mle(out)
out_aic <- aic(out_mle)
out_aic
plot(out_aic)

out_bic <- bic(out_mle)
out_bic
plot(out_bic)

#################
# mglasso model #
#################
X <- rnorm(n * p)
na.id <- sample(n * p, size = n * p * 0.05, replace = TRUE)
X[na.id] <- NA
dim(X) <- c(n, p)
out <- mglasso(X)
out_aic <- aic(out)
out_aic
plot(out_aic)

out_bic <- bic(out)
out_bic
plot(out_bic)

##############
# mggm model #
##############
out_mle <- mle(out)
out_aic <- aic(out_mle)
out_aic
plot(out_aic)

out_bic <- bic(out_mle)
out_bic
plot(out_bic)

#################
# glasso model #
#################
X <- matrix(rnorm(n * p), nrow = n, ncol = p)
out <- glasso(X)
out_aic <- aic(out)
out_aic
plot(out_aic)

out_bic <- bic(out)
out_bic
plot(out_bic)
```

```
#############
# ggm model #
#############
out_mle <- mle(out)
out_aic <- aic(out_mle)
out_aic
plot(out_aic)

out_bic <- bic(out_mle)
out_bic
plot(out_bic)
```

`cglasso` *Censored Graphical Lasso Estimator*

Description

'`cglasso`' function is used to fit an l1-penalized censored Gaussian graphical model.

Usage

```
cglasso(X, lo, up, weights, pendiag = FALSE, nrho = 50L, rho.min.ratio,
        rho, maxR2, maxit_em = 1.0e+3, thr_em = 1.0e-4, maxit_bcd = 1.0e+4,
        thr_bcd = 1.0e-4, trace = 0L)
```

Arguments

`X`	an object with S3 class '`datacggm`', usually the output of the function `datacggm`. Optionally, this argument can be a matrix of dimension $n \times p$; in this case, the matrix '`X`' and the arguments '`lo`' and '`up`' are passed to `datacggm` to create the object with class '`datacggm`'.
`lo`	optional argument. If the argument '`X`' is a matrix then '`lo`' is used to create an object with class '`datacggm`'.
`up`	optional argument. If the argument '`X`' is a matrix then '`up`' is used to create an object with class '`datacggm`'.
`weights`	an optional symmetric matrix of non-negative weights. This matrix can be used to specify the unpenalized partial correlation coefficients ('`weights[i,j] = 0`') or the structural zeros in the precision matrix ('`weights[i,j] = +Inf`'). See below for an example. By default, cglasso model is fitted without weights.
`pendiag`	flag used to specify if the diagonal elements of the concentration matrix are penalized ('`pendiag = TRUE`') or unpenalized ('`pendiag = FALSE`').
`nrho`	the integer specifying the number of tuning parameters used to fit the cglasso model. Default is '`nrho = 50`'.

`rho.min.ratio` the smallest value for the tuning parameter ρ, as a fraction of the smallest tuning parameter for which all the estimated partial correlation coefficients are zero. The default depends on the sample size 'n' relative to the number of variables 'p'. If '$p < n$', the default is '1.0E-4' otherwise the value '1.0E-2' is used as default. A very small value of '`rho.min.ratio`' will lead to a saturated fitted model in the '$p < n$' case.

`rho` optional argument. A user supplied `rho` sequence. WARNING: avoid supplying a single value for the tuning parameter; supply instead a decreasing sequence of ρ-values.

`maxR2` a value belonging to the interval $[0, 1]$ specifying the largest value of the pseudo R-squared measure (see Section **Details**). The regularization path is stopped when R^2 exceeds '`maxR2`'. Default depends on the sample size 'n' relative to the number of variables 'p'. If '$p < n$', the default is '1' otherwise the value '0.9' is used as default.

`maxit_em` maximum number of iterations of the EM algorithm. Default is `1.0E+3`.

`thr_em` threshold for the convergence of the EM algorithm. Default value is `1.0E-4`.

`maxit_bcd` maximum number of iterations of the glasso algorithm. Default is `1.0E+4`.

`thr_bcd` threshold for the convergence of the glasso algorithm. Default is `1.0E-4`.

`trace` integer for printing out information as iterations proceed: `trace = 0` no information is printed out on video; `trace = 1` basic information is printed out on video; `trace = 2` detailed information is printed out on video.

Details

The censored graphical lasso (cglasso) estimator (Augugliaro *and other*, 2018) is an extension of the classical graphical lasso (glasso) estimator (Yuan *and other*, 2007) developed to fit a sparse censored Gaussian graphical model (see Section 2 in Augugliaro *and other* (2018) for a formal definition).

`cglasso` function fits the model using the following EM-like algorithm:

Step	Description
1.	Let $\{\hat{\mu}^{\rho}_{ini}; \hat{\Theta}^{\rho}_{ini}\}$ be initial estimates;
2.	**E-step** use the moments of the truncated normal distribution to compute the current estimates of the marginal means, denoted by $\bar{x}^{\rho}$, and to complete the empirical covariance matrix S^{ρ};
3.	**M-step** let $\hat{\mu}^{\rho} = \bar{x}^{\rho}$; compute $\hat{\Theta}^{\rho}$ using S^{ρ} and the glasso algorithm (Friedman *and other*, 2008);
4.	repeat steps 2. and 3. until a convergence criterion is met.

In order to reduce the computational burdern of the algorithm, in Step 2. the matrix S^{ρ} is approximated using the method proposed in Guo *and others* (2015).

In order to avoid the overfitting of the model, we use the following pseudo R-squared measure:

$$R^2 = 1 - \frac{\|S^{\rho} - \hat{\Sigma}^{\rho}\|_F}{\|S^{\rho_{\max}} - \hat{\Sigma}^{\rho_{\max}}\|_F},$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\rho_{\max}$ denotes the smallest value of the tuning parameter for which all the estimated partial correlation coefficients are zero. By straightforward algebra, it is easy to show that the proposed pseudo R-squared belongs to the closed interval $[0, 1]$: $R^2 = 0$ when the tuning parameter is equal to $\rho_{\max}$ and $R^2 = 1$ when $\rho = 0$. The regularization path is stopped when R^2 exceeds the threshold specify by '`maxR2`'.

Value

`cglasso` returns an object with S3 class "`cglasso`", i.e., a list containing the following components:

`call`	the call that produced this object.
`X`	the object with S3 class '`datacggm`' used to fit the cglasso model.
`weights`	the weights used to fit the cglasso model.
`pendiag`	flag used to specify if the diagonal elements of the concentration matrix are penalized.
`xm`	the p-dimensional vector reporting the estimates of the marginal expected values under the assumption that the precision matrix is diagonal.
`vm`	the p-dimensional vector reporting the estimates of the marginal variances under the assumption that the precision matrix is diagonal.
`nrho`	the number of fitted cglasso model.
`rho.min.ratio`	the scale factor used to compute the smallest value of the tuning parameter.
`rho`	the p-dimensional vector reporting the values of the tuning parameter used to fit the cglasso model.
`maxR2`	the threshold value used to stop the regularization path.
`maxit_em`	the maximum number of iterations of the EM algorithm.
`thr_em`	the threshold for the convergence of the EM algorithm.
`maxit_bcd`	the maximum number of iterations of the glasso algorithm.
`thr_bcd`	the threshold for the convergence of the glasso algorithm.
`Xipt`	an array of dimension $n \times p \times$ `nrho`. `Xipt[,,k]` is the matrix where the censored values are replaced with the conditional expected values computed in the E-step of the algorithm describted in section **Details**.
`S`	an array of dimension $p \times p \times$ `nrho`. `S[,,k]` is the matrix S^{ρ} used to fit the glasso model in the M-step of the algorithm describted in section **Details**.
`mu`	a matrix of dimension $p \times$ `nrho`. The kth column is the estimate of the expected values of the cglasso model fitted using `rho[k]`.
`Sgm`	an array of dimension $p \times p \times$ `nrho`. `Sgm[,,k]` is the estimate of the covariance matrix of the cglasso model fitted using `rho[k]`.
`Tht`	an array of dimension $p \times p \times$ `nrho`. `Tht[,,k]` is the estimate of the precision matrix of the cglasso model fitted using `rho[k]`.
`Adj`	an array of dimension $p \times p \times$ `nrho`. `Adj[,,k]` is the adjacency matrix associated to `Tht[,,k]`, i.e. `Adj[i,j,k]` $= 1$ iff `Tht[i,j,k]` $\neq 0$ and `0` otherwise.
`df`	the $nrho$-dimensional vector reporting the number of non-zero partial correlation coefficients.

R2	the $nrho$-dimensional vector reporting the values of the measure R^2 described in section **Details**.
ncomp	the $nrho$-dimensional vector reporting the number of connected components (for internal purposes only).
Ck	the $(p \times \texttt{nrho})$-dimensional matrix encoding the connected components (for internal purposes only).
pk	the $(p \times \texttt{nrho})$-dimensional matrix reporting the number of vertices per connected component (for internal purposes only).
nit	the $(\texttt{nrho} \times 2)$-dimensional matrix reporting the number of iterations.
conv	a description of the error that has occurred.
subrout	the name of the Fortran subroutine where the error has occurred (for internal debug only).
trace	the integer used for printing out information.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

References

Augugliaro, L., Abbruzzo, A. and Vinciotti, V. (2018). ℓ_1-Penalized gaussian graphical model. *Biostatistics* (to appear).

Friedman, J.H., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* **9**, 432–441.

Guo, J., Levina, E., Michailidis, G. and Zhu, J. (2015). Graphical models for ordinal data. *Journal of Computational and Graphical Statistics* **24**, 183–204.

Yuan, M. and Lin, Y. (2007). Model selection and estimation in the Gaussian graphical model. *Biometrika* **94**, 19–35.

See Also

datacggm, glasso, to_graph, mle and the method functions summary, coef, plot, aic, bic, ebic.

Examples

```
library("cglasso")
set.seed(123)

p <- 5L
n <- 100L
mu <- rep(0L, p)
Tht <- diag(p)
diag(Tht[-1L, -p]) <- diag(Tht[-p, -1L]) <- 0.3
Sgm <- solve(Tht)
X <- rdatacggm(n = n, mu = mu, Sigma = Sgm, probr = 0.05)
out <- cglasso(X = X)
out
```

```
# in this example we use the argument 'weights' to specify
# the unpenalized partial correlation coefficients and the
# structural zeros in the precision matrix

w <- rep(1L, p * p)
dim(w) <- c(p, p)

# specifing the unpenalized partial correlation coefficients
diag(w) <- diag(w[-1L, -p]) <- diag(w[-p, -1L]) <- 0L

# specifing the structural zeros
w[1L, 4L:5L] <- w[4L:5L, 1L] <- +Inf
w[2L, 5L] <- w[5L, 2L] <- +Inf
w

out <- cglasso(X = X, weights = w)

# checking structural zeros
out$Tht[, , out$nrho][w == +Inf]

# checking stationarity conditions of the MLE estimators
# (the unpenalized partial correlation coefficients)
(out$Sgm[, , out$nrho] - out$S[, , out$nrho])[w == 0]
```

---

coef *Extract Model Coefficients*

Description

'coef' extracts model coefficients from a fitted model.

Usage

```
## S3 method for class 'glasso'
coef(object, ..., nrho = 1L, type = c("theta", "sigma"),
     print.info = FALSE, digits = 3L)
## S3 method for class 'mglasso'
coef(object, ..., nrho = 1L, type = c("theta", "sigma", "mu"),
     print.info = FALSE, digits = 3L)
```

Arguments

object	an object with class 'glasso', 'ggm', 'mglasso' or 'mggm' 'cglasso' or 'cggm'.
nrho	integer used to specify the model from which to extract the coefficients. Default is nrho = 1.

type	a string specifying the returned parameters. If 'object' has class 'glasso' or 'ggm', the user can choice between the precision matrix ('type = "theta"') and the covariance matrix ('type = "sigma"'). In the other fitted models,the user can also extract the estimates of the expected values ('type = "mu"'). Default is "theta"
print.info	flag specifying if information about the model is printed out. Default is FALSE.
digits	the minimum number of significant digits to be used. Default is 3L.
...	additional argument added for backward compatibility with the generic function coef.

Details

By default, the method functions 'coef.glasso' and 'coef.mglasso' return the parameters specified by the argument 'type'.

If 'print.info = TRUE' then the estimated parameters are silently returned and information about the chosen model is printed out, i.e. the value of the tuning parameter, the value of the pseudo R-squared, the number of connected components and the number of vertices per connected component. Furthermore, to improve the readability of the results the estimates are printed out taken into account the connected components (see the examples below).

Value

Coefficients extracted from 'object' are returned.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

See Also

glasso, mglasso, cglasso and mle.

Examples

```
library("cglasso")

#################
# cglasso model #
#################
set.seed(123)
p <- 5L
n <- 100L
mu <- rep(0L, p)
Tht <- diag(p)
diag(Tht[-1L, -p]) <- diag(Tht[-p, -1L]) <- 0.3
Sgm <- solve(Tht)
X <- rdatacggm(n = n, mu = mu, Sigma = Sgm, probr = 0.05)
out <- cglasso(X = X)

coef(out, nrho = 3L, type = "theta", print.info = TRUE)
```

```
Tht_hat <- coef(out, nrho = 3L, type = "theta")
Tht_hat

coef(out, nrho = 3L, type = "sigma", print.info  = TRUE)
Sgm_hat <- coef(out, nrho = 3L, type = "sigma")
Sgm_hat

coef(out, nrho = 3L, type = "mu", print.info = TRUE)
mu_hat <- coef(out, nrho = 3L, type = "mu")
mu_hat

##################
# mglasso model #
##################
R <- event(X)
X <- as.matrix(X)
X[R == 1L] <- NA
out <- mglasso(X = X)

coef(out, nrho = 3L, type = "theta", print.info = TRUE)
Tht_hat <- coef(out, nrho = 3L, type = "theta")
Tht_hat

coef(out, nrho = 3L, type = "sigma", print.info  = TRUE)
Sgm_hat <- coef(out, nrho = 3L, type = "sigma")
Sgm_hat

coef(out, nrho = 3L, type = "mu", print.info = TRUE)
mu_hat <- coef(out, nrho = 3L, type = "mu")
mu_hat

################
# glasso model #
################
X <- MASS::mvrnorm(n = n, mu = mu, Sigma = Sgm)
out <- glasso(X = X)

coef(out, nrho = 3L, type = "theta", print.info = TRUE)
Tht_hat <- coef(out, nrho = 3L, type = "theta")
Tht_hat

coef(out, nrho = 3L, type = "sigma", print.info  = TRUE)
Sgm_hat <- coef(out, nrho = 3L, type = "sigma")
Sgm_hat
```

---

datacggm *Create a Dataset from a Censored Gaussian Graphical Model*

Description

'datacggm' function is used to create a dataset from a censored Gaussian graphical model.

Usage

```
datacggm(X, lo, up)
```

Arguments

X	a $(n \times p)$-dimensional matrix; each row is an observation from a censored Gaussian graphical model with censoring vectors lo and up.
lo	the lower censoring vector; lo[j] is used to specify the lower censoring value for the random variable X_j.
up	the upper censoring vector; up[j] is used to specify the upper censoring value for the random variable X_j.

Details

The function 'datacggm' returns a named list with class 'datacggm' containing the elements needed to fit a censored graphical lasso (cglasso) model. In output, the matrix X is ordered according to the patter of censoring values.

There are specific method functions developed to help the user to deal with the censored values. The 'print.datacggm' method function print out the left and right-censored values using the following rules: a right-censored value is labeld adding the symbol '+' at the end of the value, whereas the symbol '-' is used for the left-censored values (see examples bellow). The summary statistics about the censored values can be obtained using the method function 'summary.datacggm'. The original X matrix is returned using the method function 'as.matrix'.

Finally, the status indicator matrix, denoted by R, can be obtained by the function event. The elements of this matrix specify the status of an observation as follows:

- 'R[i,j] = 0' means that the ith observation of the jth random variable is observed;
- 'R[i,j] = -1' means that the ith observation of the jth random variable is left-censored;
- 'R[i,j] = +1' means that the ith observation of the jth random variable is right-censored.

Value

'datacggm' returns an object with S3 class "datacggm", i.e. a list containing the following components:

X	the $(n \times p)$-dimensional matrix X ordered according to the patterns of censored values.
lo	the lower censoring vector.
up	the upper censoring vector.
R	the augmented status indicator matrix encoding the patterns of censored values (for internal purposes only); the status indicator matrix is returned by function event.
startmis	the row of the matrix X where are starting the patterns of censored values (for internal purposes only).

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

References

Augugliaro, L., Abbruzzo, A. and Vinciotti, V. (2018). ℓ_1-Penalized gaussian graphical model. *Biostatistics* (to appear).

See Also

event, rdatacggm, cglasso and the method functions scale.datacggm and summary.datacggm.

Examples

```
set.seed(123)
library("cglasso")

# a dataset from a left-censored Gaussian graphical model
n <- 100L
p <- 5L
X <- matrix(rnorm(n * p), n, p)
lo <- -1
X[X <= lo] <- lo
X <- datacggm(X, lo = lo)
X
as.matrix(X)

# a dataset from a right-censored Gaussian graphical model
n <- 100L
p <- 5L
X <- matrix(rnorm(n * p), n, p)
up <- 1
X[X >= up] <- up
X <- datacggm(X, up = up)
X
as.matrix(X)

# a dataset from a censored Gaussian graphical model
n <- 100L
p <- 5L
X <- matrix(rnorm(n * p), n, p)
up <- 1
lo <- -1
X[X >= up] <- up
X[X <= lo] <- lo
X <- datacggm(X, lo = lo, up = up)
X
as.matrix(X)
```

ebic *Extended Bayesian Information Criterion*

Description

'ebic' function computes the extended Bayesian Information Criterion.

Usage

```
ebic(object, g)

## S3 method for class 'glasso'
ebic(object, g = 0.5)

## S3 method for class 'mglasso'
ebic(object, g = 0.5)

## S3 method for class 'cglasso'
ebic(object, g = 0.5)
```

Arguments

object	a fitted model object.
g	the parameter indexing the extended BIC: a value belonging to the interval $[0, 1]$. Default is 0.5

Details

The measure of goodness-of-fit (gof) returned by the function 'ebic' depends on the class of the fitted model.

If 'object' has class 'glasso' or 'ggm', then 'ebic' computes the extended Bayesian Information Criterion (eBIC) proposed in Foygel *and others* (2010):

$$\text{eBIC} = -2\,\text{log-likelihood} + a(\rho)(\log n + 4\gamma logp),$$

where $a(\rho)$ denotes the number of non-zero off-diagonal elements in $\hat{\Theta}^{\rho}$ and γ is a value belonging to the interval $[0, 1]$ indexing the measure of goodness-of-fit. As explained in Foygel *and others* (2010), the log-likelihood function is evaluated using the maximum likelihood estimates of the model select by glasso. For this reason, 'ebic' calls the generic function mle to fit the Gaussian graphical model (GGM) selected by glasso.

For the remaining models, eBIC is defined as:

$$\text{eBIC} = -2\,Q\text{-function} + a(\rho)(\log n + 4\gamma logp),$$

where the Q-function is evaluated at the M-step of the EM-like algorithm describted in mle.

'ebic' returns an object with S3 class 'gof' for which are available the method functions 'print.gof' and 'plot.gof'. These method functions are developed with the aim of helping the user in finding

the optimal value of the tuning parameter, defined as the ρ-value minimizing the eBIC meaure. For this reason, '`print.gof`' shows also the ranking of the fitted models (the best model is pointed out with an arrow) whereas '`plot.gof`' points out the optimal ρ-value by a vertical dashed line (see below for some examples).

Value

'`ebic`' returns an object with S3 class "`gof`", i.e. a list containing the following components:

`value_gof`	the values of the measure of goodness-of-fit used to evaluate the fitted models.
`rho`	the values of the tuning parameter used to fit the models.
`value`	the values of the log-likelihood or Q-function.
`df`	the number of the estimated non-zero parameters, i.e. the number of non-zero partial correlations plus $2p$.
`n`	the sample size.
`p`	the number of variables.
`model`	the name of the fitted model.
`type`	the measure of goodness-of-fit used to evaluate the fitted models.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

References

Foygel, R. and Drton, M. (2010). Extended Bayesian Information Criteria for Gaussian Graphical Models. In: Lafferty, J., Williams, C., Shawe-taylor, J., Zemel, R.s. and Culott, A. (editors), *Advances in Neural Information Processing Systems 23*. pp. 604–612.

See Also

`loglik`, `cglasso`, `mglasso`, `glasso`, `mle`, `aic`, `bic` and the method funtions `plot` and `summary`.

Examples

```
library("cglasso")
set.seed(123)

#################
# cglasso model #
#################
n <- 100L
p <- 5L
mu <- rep.int(0L, times = p)
X <- rdatacggm(n = n, mu = mu, probr = 0.05)
out <- cglasso(X = X)
out_ebic <- ebic(out)
out_ebic
plot(out_ebic)
```

```
##############
# cggm model #
##############
out_mle <- mle(out)
out_ebic <- ebic(out_mle)
out_ebic
plot(out_ebic)

#################
# mglasso model #
#################
X <- rnorm(n * p)
id.na <- sample.int(n = n * p, size = n * p * 0.05)
X[id.na] <- NA
dim(X) <- c(n, p)
out <- mglasso(X = X)
out_ebic <- ebic(out)
out_ebic
plot(out_ebic)

##############
# mggm model #
##############
out_mle <- mle(out)
out_ebic <- ebic(out_mle)
out_ebic
plot(out_ebic)

#################
# glasso model #
#################
X <- rnorm(n * p)
dim(X) <- c(n, p)
out <- glasso(X)
out_ebic <- ebic(out)
out_ebic
plot(out_ebic)

#############
# ggm model #
#############
out_mle <- mle(out)
out_ebic <- ebic(out_mle)
out_ebic
plot(out_ebic)
```

`event` *Return the Indicator Matrix from an Object with class* '`datacggm`'

Description

The 'event' function is used to create a status indicator matrix from an object with class 'datacggm'. The elements of the matrix, denoted by R, are used to specify the status of an observation:

- 'R[i,j] = 0' means that the ith observation of the jth random variable is observed;
- 'R[i,j] = -1' means that the ith observation of the jth random variable is left-censored;
- 'R[i,j] = +1' means that the ith observation of the jth random variable is right-censored.

See examples below.

Usage

```
event(x)
```

Arguments

x	an object with class 'datacggm'.

Value

event returns a $(n \times p)$-dimensional matrix.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

References

Augugliaro, L., Abbruzzo, A. and Vinciotti, V. (2018). ℓ_1-Penalized gaussian graphical model. *Biostatistics* (to appear).

See Also

datacggm, rdatacggm and the method function summary.datacggm.

Examples

```
set.seed(123)
library("cglasso")

# dataset from a left-censored Gaussian graphical model
n <- 100L
p <- 5L
X <- matrix(rnorm(n * p), n, p)
lo <- -1
X[X <= lo] <- lo
X <- datacggm(X, lo = lo)
event(X)

# dataset from a right-censored Gaussian graphical model
n <- 100L
```

```
p <- 5L
X <- matrix(rnorm(n * p), n, p)
up <- 1
X[X >= up] <- up
X <- datacggm(X, up = up)
event(X)

# dataset from a censored Gaussian graphical model
n <- 100L
p <- 5L
X <- matrix(rnorm(n * p), n, p)
up <- 1
lo <- -1
X[X >= up] <- up
X[X <= lo] <- lo
X <- datacggm(X, lo = lo, up = up)
event(X)
```

`glasso` *Lasso Estimator for Gaussian Graphical Models*

Description

'`glasso`' fits the l1-penalized Gaussian graphical model.

Usage

```
glasso(X, weights, pendiag = FALSE, nrho = 50L, rho.min.ratio, rho, maxR2,
       maxit = 1.0e+4, thr = 1.0e-04, trace = 0L)
```

Arguments

`X`	the $(n \times p)$-dimensional matrix used to compute the covariance matrix.
`weights`	an optional symmetric matrix of non-negative weights. This matrix can be used to specify the unpenalized partial correlation coefficients ('`weights[i,j] = 0`') or the structural zeros in the precision matrix ('`weights[i,j] = +Inf`'). See below for an example. By default, glasso model is fitted without weights.
`pendiag`	flag used to specify if the diagonal elements of the concentration matrix are penalized ('`pendiag = TRUE`') or unpenalized ('`pendiag = FALSE`').
`nrho`	the integer specifying the number of tuning parameters used to fit the glasso model. Default is '`nrho = 50L`'.
`rho.min.ratio`	the smallest value for the tuning parameter ρ, as a fraction of the smallest tuning parameter for which all the estimated partial correlation coefficients are zero. The default depends on the sample size 'n' relative to the number of variables 'p'. If '$p < n$', the default is '1.0E-4' otherwise the value '1.0E-2' is used as default. A very small value of '`rho.min.ratio`' will lead to a saturated fitted model in the '$p < n$' case.

rho	optional argument. A user supplied rho sequence. WARNING: avoid supplying a single value for the tuning parameter; supply instead a decreasing sequence of ρ-values.
maxR2	a value belonging to the interval $[0, 1]$ specifying the largest value of the pseudo R-squared measure (see Section **Details**). The regularization path is stopped when R^2 exceeds 'maxR2'. The default depends on the sample size 'n' relative to the number of variables 'p'. If '$p < n$', the default is '1' otherwise the value '0.9' is used as default.
maxit	maximum number of iterations of the glasso algorithm. Default is 1.0E+4.
thr	threshold for the convergence of the glasso algorithm. Default is 1.0E-4.
trace	integer for printing out information as iterations proceed: trace = 0 no information is printed out on video; trace = 1 basic information is printed out on video; trace = 2 detailed information is printed out on video.

Details

For a fixed value of the tuning parameter, glasso solves the following maximization problem:

$$\max_{\Theta} \log det\Theta - tr\{S\Theta\} - \rho \sum_{h,k} w_{hk}|\theta_{hk}|,$$

where w_{hk} is the non-negative weight for θ_{hk}. The previous maximization problem is solved effeciently combining the block-coordinate descent algorithm (Friedman *and others*, 2008) with the screening rule proposed in Witten *and others* (2011).

In order to avoid the overfitting of the model, we use the following pseudo R-squared measure:

$$R^2 = 1 - \frac{\|S - \hat{\Sigma}^{\rho}\|_F}{\|S - \hat{\Sigma}^{\rho_{\max}}\|_F},$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\rho_{\max}$ denotes the smallest value of the tuning parameter for which all the estimated partial correlation coefficients are zero. By straightforward algebra, it is easy to show that the proposed pseudo R-squared belongs to the closed interval $[0, 1]$: $R^2 = 0$ when the tuning parameter is equal to $\rho_{\max}$ and $R^2 = 1$ when $\rho = 0$. The regularization path is stopped when R^2 exceeds 'maxR2'.

Value

'glasso' returns an object with S3 class "glasso", i.e. a list containing the following components:

call	the call that produced this object.
X	the matrix used to compute the covariance matrix.
S	the covariance matrix used to fit the glasso model.
weights	the used weights.
pendiag	the flag specifying if the diagonal elements of the precisione matrix are penalized.
nrho	the number of fitted glasso model.
rho.min.ratio	the scale factor used to compute the smallest value of the tuning parameter.

`rho`	the p-dimensional vector reporting the values of the tuning parameter used to fit the glasso model.
`maxR2`	the threshold value used for the pseudo R-squared measure.
`maxit`	the maximum number of iterations of the glasso algorithm.
`thr`	the threshold for the convergence of the glasso algorithm.
`Sgm`	an array of dimension $(p \times p \times$ `nrho`$)$. `Sgm[,,k]` is the estimate of the covariance matrix of the glasso model fitted using `rho[k]`.
`Tht`	an array of dimension $(p \times p \times$ `nrho`$)$. `Tht[,,k]` is the estimate of the precision matrix of the glasso model fitted using `rho[k]`.
`Adj`	an array of dimension $(p \times p \times$ `nrho`$)$. `Adj[,,k]` is the adjacency matrix associated to `Tht[,,k]`, i.e. `Adj[i,j,k]` $= 1$ iff `Tht[i,j,k]` $\neq 0$ and `0` otherwise.
`df`	the $nrho$-dimensional vector reporting the number of non-zero partial correlation coefficients.
`R2`	the $nrho$-dimensional vector reporting the values of the measure R^2 described in the section **Details**.
`ncomp`	the $nrho$-dimensional vector reporting the number of connected components (for internal purposes only).
`Ck`	the $(p \times$ `nrho`$)$-dimensional matrix encoding the connected components (for internal purposes only).
`pk`	the $(p \times$ `nrho`$)$-dimensional matrix reporting the number of vertices per connected component (for internal purposes only).
`nit`	the p-dimensional vector reporting the number of iterations.
`conv`	a description of the error that has occurred.
`trace`	the integer used for printing out information.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

References

Friedman, J.H., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* **9**, 432–441.

Witten, D.M., Friedman, J.H, and Simon, N. (2011). New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics* **20**, 892–900.

See Also

`mle`, `to_graph` and the method functions `summary`, `coef`, `plot`, `aic`, `bic` and `ebic`.

Examples

```
library("cglasso")
set.seed(123)

p <- 5L
n <- 100L
mu <- rep(0L, p)
Tht <- diag(p)
diag(Tht[-1L, -p]) <- diag(Tht[-p, -1L]) <- 0.3
Sgm <- solve(Tht)
X <- MASS::mvrnorm(n = n, mu = mu, Sigma = Sgm)
out <- glasso(X)
out

# in this example we use the argument 'weights' to specify
# the unpenalized partial correlation coefficients and the
# structural zeros in the precision matrix

w <- rep(1, p * p)
dim(w) <- c(p, p)

# specifing the unpenalized partial correlation coefficients
diag(w) <- diag(w[-1L, -p]) <- diag(w[-p, -1L]) <- 0

# specifing the structural zero
w[1L, 4L:5L] <- w[4L:5L, 1L] <- +Inf
w[2L, 5L] <- w[5L, 2L] <- +Inf
w

out <- glasso(X = X, weights = w)

# checking structural zeros
out$Tht[, , out$nrho][w == +Inf]

# checking stationarity conditions of the MLE estimators
# (the unpenalized partial correlation coefficients)
(out$Sgm[, , out$nrho] - out$S)[w == 0L]
```

---

loglik *Extract Log-Likelihood or Q-Function*

Description

'loglik' function extracts the values of the log-liklihood function from an object with class 'glasso' or 'ggm', otherwise the values of the Q-function are returned.

Usage

```
loglik(object)
```

Arguments

`object`	a fitted model object.

Details

If '`object`' has class '`glasso`' or '`ggm`', the function '`loglik()`' returns the value of the log-likelihood function:

$$\frac{n}{2}\left\{\log det\Theta - tr(S\Theta) - p\log(2\pi)\right\},$$

where Θ is estimated using the function glasso or mle.glasso.

For the other models, '`loglik()`' returns the value of the Q-function, i.e. the function maximaxed in the M-step of the EM-like algorithm. The Q-function is defined as follows:

$$\frac{n}{2}\left\{\log det\Theta - tr(S'\Theta) - p\log(2\pi)\right\},$$

where S' is computed in the E-step.

The method function '`print.loglik`' is used the improve the readability of the results.

Value

'`loglik`' returns an object with S3 class "`loglik`", i.e. a list containing the following components:

`value`	the values of the log-likelihood or Q-function.
`df`	the number of the estimated non-zero parameters, i.e. the number of non-zero partial correlations plus $2p$.
`n`	the sample size.
`p`	the number of variables.
`rho`	the values of the tuning parameter used to fit the model.
`model`	the name of the fitted model.
`fun`	the name of the used function, i.e. the log-likelihood or the Q-function.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

See Also

`cglasso`, `mglasso`, `glasso`, `mle` and the method functions, `plot`, `aic`, `bic` and `ebic`.

Examples

```
library("cglasso")
set.seed(123)

#################
# cglasso model #
#################
p <- 5L
```

```
n <- 100L
mu <- rep(0L, p)
Tht <- diag(p)
diag(Tht[-1L, -p]) <- diag(Tht[-p, -1L]) <- 0.3
Sgm <- solve(Tht)
X <- rdatacggm(n = n, mu = mu, Sigma = Sgm, probr = 0.05)
out <- cglasso(X = X)
out_loglik <- loglik(out)
out_loglik

##############
# cggm model #
##############
out_mle <- mle(out)
out_loglik <- loglik(out_mle)
out_loglik

#################
# mglasso model #
#################
library(MASS)
X <- mvrnorm(n = n, mu = mu, Sigma = Sgm)
id.na <- sample.int(n = n * p, size = n * p * 0.05)
X[id.na] <- NA
out <- mglasso(X = X)
out_loglik <- loglik(out)
out_loglik

##############
# mggm model #
##############
out_mle <- mle(out)
out_loglik <- loglik(out_mle)
out_loglik

################
# glasso model #
################
X <- mvrnorm(n = n, mu = mu, Sigma = Sgm)
out <- glasso(X)
out_loglik <- loglik(out)
out_loglik

#############
# ggm model #
#############
out_mle <- mle(out)
out_loglik <- loglik(out_mle)
out_loglik
```

`mglasso` *Graphical Lasso Estimator with Missing-at-Random Data*

Description

'mglasso' function is used to fit an l1-penalized Gaussian graphical model with missing-at-random data.

Usage

```
mglasso(X, weights, pendiag = FALSE, nrho = 50L, rho.min.ratio, rho,
        maxR2, maxit_em = 1.0e+3, thr_em = 1.0e-4, maxit_bcd = 1.0e+4,
        thr_bcd = 1.0e-4, trace = 0L)
```

Arguments

X	the $(n \times p)$-dimensional matrix used to fit the model.
weights	an optional symmetric matrix of non-negative weights. This matrix can be used to specify the unpenalized partial correlation coefficients ('weights[i,j] = 0') or the structural zeros in the precision matrix ('weights[i,j] = +Inf'). See below for an example. By default, mglasso model is fitted without weights.
pendiag	flag used to specify if the diagonal elements of the concentration matrix are penalized ('pendiag = TRUE') or unpenalized ('pendiag = FALSE').
nrho	the integer specifying the number of tuning parameters used to fit the mglasso model. Default is 'nrho = 50'.
rho.min.ratio	the smallest value for the tuning parameter ρ, as a fraction of the smallest tuning parameter for which all the estimated partial correlation coefficients are zero. The default depends on the sample size 'n' relative to the number of variables 'p'. If '$p < n$', the default is '1.0E-4' otherwise the value '1.0E-2' is used as default. A very small value of 'rho.min.ratio' will lead to a saturated fitted model in the '$p < n$' case.
rho	optional argument. A user supplied rho sequence. WARNING: avoid supplying a single value for the tuning parameter; supply instead a decreasing sequence of ρ-values.
maxR2	a value belonging to the interval $[0, 1]$ specifying the largest value of the pseudo R-squared measure (see Section **Details**). The regularization path is stopped when R^2 exceeds 'maxR2'. Default depends on the sample size 'n' relative to the number of variables 'p'. If '$p < n$', the default is '1' otherwise the value '0.9' is used as default.
maxit_em	maximum number of iterations of the EM algorithm. Default is 1.0E+3.
thr_em	threshold for the convergence of the EM algorithm. Default value is 1.0E-4.
maxit_bcd	maximum number of iterations of the glasso algorithm. Default is 1.0E+4.
thr_bcd	threshold for the convergence of the glasso algorithm. Default is 1.0E-4.
trace	integer for printing out information as iterations proceed: trace = 0 no information is printed out on video; trace = 1 basic information is printed out on video; trace = 2 detailed information is printed out on video.

Details

The missglasso estimator (Stadler *and other*, 2012) is an extension of the classical graphical lasso (glasso) estimator (Yuan *and other*, 2007) developed to fit a sparse Gaussian graphical model under the assumption that data are missing-at-random.

`mglasso` function fits the model using the following EM algorithm:

Step	Description
1.	Let $\{\hat{\mu}^{\rho}_{ini}; \hat{\Theta}^{\rho}_{ini}\}$ be initial estimates;
2.	**E-step** use the expected values of the conditional normal distribution to impute the missing data let X^{ρ} the completed data and S^{ρ} the corresponding empirical covariance matrix
3.	**M-step** let $\hat{\mu}^{\rho}_{h} = \sum_{i=1}^{n} x^{\rho}_{ih}$; compute $\hat{\Theta}^{\rho}$ using S^{ρ} and the glasso algorithm (Friedman *and other*, 2008);
4.	repeat steps 2. and 3. until a convergence criterion is met.

In order to avoid the overfitting of the model, we use the following pseudo R-squared measure:

$$R^2 = 1 - \frac{\|S^{\rho} - \hat{\Sigma}^{\rho}\|_F}{\|S^{\rho_{\max}} - \hat{\Sigma}^{\rho_{\max}}\|_F},$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\rho_{\max}$ denotes the smallest value of the tuning parameter for which all the estimated partial correlation coefficients are zero. By straightforward algebra, it is easy to show that the proposed pseudo R-squared belongs to the closed interval $[0, 1]$: $R^2 = 0$ when the tuning parameter is equal to $\rho_{\max}$ and $R^2 = 1$ when $\rho = 0$. The regularization path is stopped when R^2 exceeds the threshold specify by '`maxR2`'.

Value

`mglasso` returns an object with S3 class "`mglasso`", i.e., a list containing the following components:

`call`	the call that produced this object.
`X`	the original matrix used to fit the missglasso model.
`weights`	the weights used to fit the missglasso model.
`pendiag`	the flag specifying if the diagonal elements of the precisione matrix are penalized.
`nrho`	the number of fitted missglasso model.
`rho.min.ratio`	the scale factor used to compute the smallest value of the tuning parameter.
`rho`	the p-dimensional vector reporting the values of the tuning parameter used to fit the missglasso model.
`maxR2`	the threshold value used to stop the regularization path.
`maxit_em`	the maximum number of iterations of the EM algorithm.
`thr_em`	the threshold for the convergence of the EM algorithm.
`maxit_bcd`	the maximum number of iterations of the glasso algorithm.
`thr_bcd`	the threshold for the convergence of the glasso algorithm.

`Xipt`	an array of dimension $n \times p \times$ `nrho`. `Xipt[,,k]` is the matrix where the missing values are replaced with the conditional expected values computed in the E-step of the algorithm describted in section **Details**.
`S`	an array of dimension $p \times p \times$ `nrho`. `S[,,k]` is the matrix S^{ρ} used to fit the glasso model in the M-step of the algorithm describted in section **Details**.
`mu`	a matrix of dimension $p \times$ `nrho`. The kth column is the estimate of the expected values of the missglasso model fitted using `rho[k]`.
`Sgm`	an array of dimension $p \times p \times$ `nrho`. `Sgm[,,k]` is the estimate of the covariance matrix of the missglasso model fitted using `rho[k]`.
`Tht`	an array of dimension $p \times p \times$ `nrho`. `Tht[,,k]` is the estimate of the precision matrix of the missglasso model fitted using `rho[k]`.
`Adj`	an array of dimension $p \times p \times$ `nrho`. `Adj[,,k]` is the adjacency matrix associated to `Tht[,,k]`, i.e. `Adj[i,j,k]` $= 1$ iff `Tht[i,j,k]` $\neq 0$ and `0` otherwise.
`df`	the $nrho$-dimensional vector reporting the number of non-zero partial correlation coefficients.
`R2`	the $nrho$-dimensional vector reporting the values of the measure R^2 described in section **Details**.
`ncomp`	the $nrho$-dimensional vector reporting the number of connected components (for internal purposes only).
`Ck`	the $(p \times$ `nrho`$)$-dimensional matrix encoding the connected components (for internal purposes only).
`pk`	the $(p \times$ `nrho`$)$-dimensional matrix reporting the number of vertices per connected component (for internal purposes only).
`nit`	the $($`nrho` $\times 2)$-dimensional matrix reporting the number of iterations.
`conv`	a description of the error that has occurred.
`subrout`	the name of the Fortran subroutine where the error has occurred (for internal debug only).
`trace`	the integer used for printing out information.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

References

Friedman, J.H., Hastie, T., and Tibshirani, R. (2008) <DOI:10.1093/biostatistics/kxm045>. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* **9**, 432–441.

Stadler N., and Buhlmann P. (2012) <DOI:10.1007/s11222-010-9219-7>. Missing values: sparse inverse covariance estimation and an extension to sparse regression. *Statistics and Computing* **22**, 219–235.

Yuan, M., and Lin, Y. (2007) <DOI:10.1093/biomet/asm018>. Model selection and estimation in the Gaussian graphical model. *Biometrika* **94**, 19–35.

See Also

glasso, to_graph, mle and the method functions summary, coef, plot, aic, bic, ebic.

Examples

```
library("cglasso")
set.seed(123)

p <- 5L
n <- 100L
mu <- rep(0L, p)
Tht <- diag(p)
diag(Tht[-1L, -p]) <- diag(Tht[-p, -1L]) <- 0.3
Sgm <- solve(Tht)
X <- MASS::mvrnorm(n = n, mu = mu, Sigma = Sgm)
X <- as.vector(X)
id.na <- sample.int(n = n * p, size = n * p * 0.05)
X[id.na] <- NA
dim(X) <- c(n, p)
out <- mglasso(X = X)
out

# in this example we use the argument 'weights' to specify
# the unpenalized partial correlation coefficients and the
# structural zeros in the precision matrix

w <- rep(1, p * p)
dim(w) <- c(p, p)

# specifing the unpenalized partial correlation coefficients
diag(w) <- diag(w[-1L, -p]) <- diag(w[-p, -1L]) <- 0

# specifing the structural zero
w[1L, 4L:5L] <- w[4L:5L, 1L] <- +Inf
w[2L, 5L] <- w[5L, 2L] <- +Inf
w

out <- mglasso(X = X, weights = w)

# checking structural zeros
out$Tht[, , out$nrho][w == +Inf]

# checking stationarity conditions of the MLE estimators
# (the unpenalized partial correlation coefficients)
(out$Sgm[, , out$nrho] - out$S[, , out$nrho])[w == 0]
```

MKMEP *Megakaryocyte-Erythroid Progenitors*

Description

In a study about the formation of blood cells, Psaila *and others* (2016) have recently identified three distinct subpopulations of cells, which are all derived from hematopoietic stem cells through cell differentiation. One of these sub-populations, denoted by MK-MEP, is a previously unknown, rare population of cells that are bipotent but primarily generate megakaryocytic progeny.

'MKMEP' in an object of class 'datacggm' containing a subset of the data available from Psaila *and others* (2016).

Multiplex RT-qPCR is used to profile 63 genes and 48 single human MK-MEP cells. RT-qPCR data are typically right-censored with a limit of detection fixed by the manufacturer to 40. Raw data have been mean normalized using the method proposed in Pipelers *and others* (2017). See Section 5 in Augugliaro *and others* (2018) for more details.

Usage

```
data("MKMEP")
```

References

Augugliaro, L., Abbruzzo, A. and Vinciotti, V. (2018). ℓ_1-Penalized gaussian graphical model. *Biostatistics* (to appear).

Pipelers, P., Clement, L., Vynck, M., Hellemans, J., Vandesompele, J. and Thas, O. (2017). A unified censored normal regression model for qPCR differential gene expression analysis. *PLoS One* **12**, e0182832.

Psaila, B., Barkas, N., Iskander, D., Roy, A., Anderson, S., Ashley, N., Caputo, V. S., Lichtenberg, J., Loaiza, S., Bodine, D. M. *and others*. (2016). Single-cell profiling of human megakaryocyte-erythroid progenitors identifies distinct megakaryocyte and erythroid differentiation pathways. *Genome Biology* **17**, 83–102.

See Also

cglasso, to_graph, and the method functions summary, coef, plot, aic, bic and ebic.

Examples

```
data("MKMEP")
out_cglasso <- cglasso(MKMEP, pendiag = TRUE, nrho = 200L, rho.min.ratio = 0.35)
out_ebic <- ebic(out_cglasso)
plot(out_ebic, type = "l")

out_graph <- to_graph(out_cglasso, nrho = which.min(out_ebic$value_gof))
V(out_graph)$color <- "white"
V(out_graph)$frame.color <- NA
V(out_graph)$label.color <- "black"
E(out_graph)$color <- "gray75"

plot(out_graph, layout = layout_with_lgl(out_graph))
mtext(text = "Megakaryocytic MEP population", cex = 1.5, line = 1)
```

mle *Maximum Likelihood Estimation*

Description

The generic function 'mle' fits the graphical model selected by glasso, mglasso or cglasso.

Usage

```
mle(object, ...)

## S3 method for class 'glasso'
mle(object, ..., maxit = 1.0e+04, thr = 1.0e-04, trace = 0L)

## S3 method for class 'mglasso'
mle(object, ..., maxit_em = 1.0e+03, thr_em = 1.0e-4, maxit_bcd = 1.0e+4,
    thr_bcd = 1.0e-4, trace = 0L)

## S3 method for class 'cglasso'
mle(object, ..., maxit_em = 1.0e+03, thr_em = 1.0e-4, maxit_bcd = 1.0e+4,
    thr_bcd = 1.0e-4, trace = 0L)
```

Arguments

object	an object of class 'glasso', 'mglasso' or 'cglasso'.
maxit	maximum number of iterations of the glasso algorithm. Default is 1.0E+4.
thr	threshold for the convergence of the glasso algorithm. Default is 1.0E-4.
maxit_em	maximum number of iterations of the EM algorithm. Default is 1.0E+03.
thr_em	threshold for the convergence of the EM algorithm. Default is 1.0E-4.
maxit_bcd	maximum number of iterations of the glasso algorithm. Default is 1.0E+4.
thr_bcd	threshold for the convergence of the glasso algorithm. Default is 1.0E-4.
trace	integer for printing out information as iterations proceed: trace = 0 no information is printed out on video; trace = 1 basic information is printed out on video; trace = 2 detailed information is printed out on video.
...	additional argument added for backward compatibility with the generic function 'mle'.

Details

The generic function 'mle' computes the maximum likelihood estimates of the graphical model selected by the function glasso, mglasso or cglasso.

If 'object' has class 'glasso', the method function 'mle.glasso' computes the maximum likelihood estimates of the parameters of the Gaussian graphical models (ggm) associated to the sequence

of glasso estimates. Formally, for a given value of the tuning parameter let $\hat{\Theta}^\rho$ be the glasso estimate of the precision matrix, then '`mle.glasso`' solves the following maximization problem:

$$\max_{\bar{\Theta}} \log det\bar{\Theta} - tr\{S\bar{\Theta}\},$$

where $\bar{\Theta} = \{\bar{\theta}_{hk}\}$ and $\bar{\theta}_{hk} = 0$ if $\hat{\theta}^\rho_{hk} = 0$ otherwise it is estimated.

If '`object`' has class '`mglasso`', the method function '`mle.mglasso`' computes the maximum likelihood estimates of the parameters of the Gaussian graphical models with missing-at-random data (mggm) associated to the sequence of mglasso estimates. Formally, for a given value of the tuning parameter let $\hat{\Theta}^\rho$ be the mglasso estimate of the precision matrix, then '`mle.mglasso`' computes the maximum likelihood estimate by the following EM-like algorithm:

Step	Description
1.	let $\hat{\Theta}^\rho$ be the mglasso estimate of the precision matrix;
2.	**E-step** use the moments of the conditional normal distribution to impute the missing values;
3.	**M-step** let X' the completed matrix and S' the corresponding empirical variance matrix, then: let $\hat{\mu}_h = \sum_{i=1}^n x'_{ih}/n$ estimate $\bar{\Theta}$ maximixing the function: $\log det\bar{\Theta} - tr\{S'\bar{\Theta}\}$, where $\bar{\theta}_{hk} = 0$ if $\hat{\theta}^\rho_{hk} = 0$ otherwise it is estimated;
4.	repeat steps 2. and 3. until a convergence criterion is met.

If '`object`' has class '`cglasso`', the method function '`mle.cglasso`' computes the maximum likelihood estimates of the parameters of the censored Gaussian graphical models (cggm) associated to the sequence of cglasso estimates. Formally, for a given value of the tuning parameter let $\hat{\Theta}^\rho$ be the cglasso estimate of the precision matrix, then '`mle.cglasso`' computes the maximum likelihood estimate by the following EM-like algorithm:

Step	Description
1.	let $\hat{\Theta}^\rho$ be the cglasso estimate of the precision matrix;
2.	**E-step** use the moments of the truncated normal distribution to compute the current estimates of the marginal means, denoted by $\bar{x}'$, and to complete the empirical covariance matrix S';
3.	**M-step** let $\hat{\mu} = \bar{x}'$; estimate $\bar{\Theta}$ maximixing the Q-function: $\log det\bar{\Theta} - tr\{S'\bar{\Theta}\}$, where $\bar{\theta}_{hk} = 0$ if $\hat{\theta}^\rho_{hk} = 0$ otherwise it is estimated;
4.	repeat steps 2. and 3. until a convergence criterion is met.

Value

If '`object`' has class '`glasso`', then '`mle`' returns and object with S3 class '`ggm`', which inherits the class '`glasso`'. An object with class '`ggm`' is a list containing the following components:

`call`	the call that produced this object.
`X`	the matrix used to compute the covariance matrix.

S	the covariance matrix used to fit the ggm model.
nrho	the number of fitted ggm model.
rho	the p-dimensional vector reporting the values of the tuning parameter used to fit the glasso model.
maxit	the maximum number of iterations of the glasso algorithm.
thr	the threshold for the convergence of the glasso algorithm.
Sgm	an array of dimension $(p \times p \times \texttt{nrho})$. `Sgm[,,k]` is the estimate of the covariance matrix of the kth ggm model.
Tht	an array of dimension $(p \times p \times \texttt{nrho})$. `Tht[,,k]` is the estimate of the precision matrix of the kth ggm model.
Adj	an array of dimension $(p \times p \times \texttt{nrho})$. `Adj[,,k]` is the adjacency matrix associated to `Tht[,,k]`, i.e. `Adj[i,j,k]` $= 1$ iff `Tht[i,j,k]` $\neq 0$ and `0` otherwise.
df	the p-dimensional vector reporting the number of non-zero partial correlation coefficients.
R2	the p-dimensional vector reporting the values of the measure R^2. See section **Details**, in `glasso`.
ncomp	the p-dimensional vector reporting the number of connected components (for internal purposes only).
Ck	the $(p \times \texttt{nrho})$-dimensional matrix encoding the connected components (for internal purposes only).
pk	the $(p \times \texttt{nrho})$-dimensional matrix reporting the number of vertices per connected component (for internal purposes only).
nit	the p-dimensional vector reporting the number of iterations.
conv	a description of the error that has occurred.
trace	the integer used for printing out information.

If '`object`' has class '`mglasso`', then '`mle`' returns and object with S3 class '`mggm`', which inherits the class '`mglasso`'. An object with class '`mggm`' is a list containing the following components:

call	the call that produced this object.
X	the object with S3 class '`datacggm`' used to fit the cggm model.
nrho	the number of fitted cggm model.
rho	the p-dimensional vector reporting the values of the tuning parameter used to fit the cglasso model.
maxit_em	maximum number of iterations of the EM algorithm.
thr_em	threshold for the convergence of the EM algorithm.
maxit_bcd	maximum number of iterations of the glasso algorithm.
thr_bcd	threshold for the convergence of the glasso algorithm.
Xipt	an array of dimension $n \times p \times \texttt{nrho}$. `Xipt[,,k]` is the matrix where the censored vaules are replaced using the conditional expected vaules computed in the E-step of the algorithm describted in section **Details**.

`S` an array of dimension $p \times p \times$ `nrho`. `S[,,k]` is the matrix S' used to fit the cggm model (see the section **Details**).

`mu` a matrix of dimension $p \times$ `nrho`. The kth column is the estimate of the expected values of the kth cggm model.

`Sgm` an array of dimension $p \times p \times$ `nrho`. `Sgm[,,k]` is the estimate of the covariance matrix of the kth cggm model.

`Tht` an array of dimension $p \times p \times$ `nrho`. `Tht[,,k]` is the estimate of the precision matrix of the kth cggm model.

`Adj` an array of dimension $p \times p \times$ `nrho`. `Adj[,,k]` is the adjacency matrix associated to `Tht[,,k]`, i.e. `Adj[i,j,k]` $= 1$ iff `Tht[i,j,k]` $\neq 0$ and `0` otherwise.

`df` the p-dimensional vector reporting the number of non-zero partial correlation coefficients.

`R2` the p-dimensional vector reporting the values of the measure R^2. See section **Details**, in `cglasso`.

`ncomp` the p-dimensional vector reporting the number of connected components (for internal purposes only).

`Ck` the $(p \times$ `nrho`$)$-dimensional matrix encoding the connected components (for internal purposes only).

`pk` the $(p \times$ `nrho`$)$-dimensional matrix reporting the number of vertices per connected component (for internal purposes only).

`nit` the $($`nrho` $\times 2)$-dimensional matrix reporting the number of iterations.

`conv` a description of the error that has occurred.

`subrout` the name of the Fortran subroutine where the error has occurred (for internal debug only).

`trace` the integer used for printing out information.

If '`object`' has class '`cglasso`', then '`mle`' returns and object with S3 class '`cggm`', which inherits the class '`cglasso`'. An object with class '`cggm`' is a list containing the following components:

`call` the call that produced this object.

`X` the object with S3 class '`datacggm`' used to fit the cggm model.

`nrho` the number of fitted cggm model.

`rho` the p-dimensional vector reporting the values of the tuning parameter used to fit the cglasso model.

`maxit_em` maximum number of iterations of the EM algorithm.

`thr_em` threshold for the convergence of the EM algorithm.

`maxit_bcd` maximum number of iterations of the glasso algorithm.

`thr_bcd` threshold for the convergence of the glasso algorithm.

`Xipt` an array of dimension $n \times p \times$ `nrho`. `Xipt[,,k]` is the matrix where the censored vaules are replaced using the conditional expected vaules computed in the E-step of the algorithm describted in section **Details**.

`S` an array of dimension $p \times p \times$ `nrho`. `S[,,k]` is the matrix S' used to fit the cggm model (see the section **Details**).

`mu`	a matrix of dimension $p \times$ `nrho`. The kth column is the estimate of the expected values of the kth cggm model.
`Sgm`	an array of dimension $p \times p \times$ `nrho`. `Sgm[,,k]` is the estimate of the covariance matrix of the kth cggm model.
`Tht`	an array of dimension $p \times p \times$ `nrho`. `Tht[,,k]` is the estimate of the precision matrix of the kth cggm model.
`Adj`	an array of dimension $p \times p \times$ `nrho`. `Adj[,,k]` is the adjacency matrix associated to `Tht[,,k]`, i.e. `Adj[i,j,k]` $= 1$ iff `Tht[i,j,k]` $\neq 0$ and `0` otherwise.
`df`	the p-dimensional vector reporting the number of non-zero partial correlation coefficients.
`R2`	the p-dimensional vector reporting the values of the measure R^2. See section **Details**, in `cglasso`.
`ncomp`	the p-dimensional vector reporting the number of connected components (for internal purposes only).
`Ck`	the $(p \times$ `nrho`$)$-dimensional matrix encoding the connected components (for internal purposes only).
`pk`	the $(p \times$ `nrho`$)$-dimensional matrix reporting the number of vertices per connected component (for internal purposes only).
`nit`	the $($`nrho` $\times 2)$-dimensional matrix reporting the number of iterations.
`conv`	a description of the error that has occurred.
`subrout`	the name of the Fortran subroutine where the error has occurred (for internal debug only).
`trace`	the integer used for printing out information.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

References

Augugliaro, L., Abbruzzo, A., and Vinciotti, V. (2018). ℓ_1-Penalized gaussian graphical model. *Biostatistics* (to appear).

Stadler N., and Buhlmann P. (2012) <DOI:10.1007/s11222-010-9219-7>. Missing values: sparse inverse covariance estimation and an extension to sparse regression. *Statistics and Computing* **22**, 219–235.

See Also

`glasso`, `mglasso`, `cglasso`, `to_graph`, and the method functions `summary`, `coef`, `plot`, `aic`, `bic` and `ebic`.

Examples

```
library("cglasso")
set.seed(123)

#################
# cglasso model #
#################
n <- 100L
p <- 5L
mu <- rep.int(0L, times = p)
X <- rdatacggm(n = n, mu = mu, probr = 0.05)
out <- cglasso(X = X)
out_mle <- mle(out)
out_mle

inherits(out_mle, "cglasso")
class(out_mle)

# inheriting method functions from 'cglasso': some examples
coef(out_mle, nrho = 10L, print.info = TRUE)
to_graph(out_mle, nrho = 10L, weighted = TRUE)
out_aic <- aic(out_mle)
out_aic
plot(out_mle, typeplot = "graph", gof = out_aic)

#################
# mglasso model #
#################
R <- event(X)
X <- as.matrix(X)
X[R == 1L] <- NA
out <- mglasso(X = X)
out_mle <- mle(out)
out_mle

inherits(out_mle, "mglasso")
class(out_mle)

# inheriting method functions from 'mglasso': some examples
coef(out_mle, nrho = 10L, print.info = TRUE)
to_graph(out_mle, nrho = 10L, weighted = TRUE)
out_aic <- aic(out_mle)
out_aic
plot(out_mle, typeplot = "graph", gof = out_aic)

################
# glasso model #
################
X <- MASS::mvrnorm(n = n, mu = mu, Sigma = diag(p))
out <- glasso(X)
out_mle <- mle(out)
out_mle
```

```
inherits(out_mle, "glasso")
class(out_mle)

# inheriting method functions from 'glasso': some examples
coef(out_mle, nrho = 10L, print.info = TRUE)
to_graph(out_mle, nrho = 10L, weighted = TRUE)
out_aic <- aic(out_mle)
out_aic
plot(out_mle, typeplot = "graph", gof = out_aic)
```

---

plot *Plot for* 'glasso' *and* 'cglasso' *Object*

Description

The method functions 'plot.glasso' and 'plot.mglasso' produce plots to study the sequence of models estimated by glasso, mglasso or cglasso.

Usage

```
## S3 method for class 'glasso'
plot(x, typeplot = c("path", "graph"),
     gof, diag = FALSE, nrho, weighted = FALSE,
     isolated = FALSE, ...)

## S3 method for class 'mglasso'
plot(x, typeplot = c("path", "graph"),
     gof, diag = FALSE, nrho, weighted = FALSE,
     isolated = FALSE, ...)
```

Arguments

x	a fitted model object.
typeplot	a string specifying the produced plot.
gof	an object with class 'gof'. See examples below.
diag	flag specifying whether the diagonal elements of the concentration matrix are plotted.
nrho	availables only if 'typeplot = graph': integer specifying the model used to procude the 'igraph' object. This argument is overwritten if 'gof' is available. See examples below.
weighted	argument availables only if 'typeplot = graph'. Flag specifying whether to create a weighted graph.
isolated	flag specifying whether the isolated vertices are removed from the graph. Default is FALSE, i.e. the isolated vertices are removed.
...	additional arguments passed to the method function 'plot.default'.

Details

The plot produced by the method functions 'plot.glasso' and 'plot.mglasso' depends on the argument 'typeplot'.

If 'typeplot = path', the regularization paths are produced; in this case, if an object with class 'gof' is passed by the argument 'gof', then a vertical dashed line is added to identify the optimal ρ-value.

If 'typeplot = graph', the method functions 'plot.glasso' and 'plot.mglasso' produce the undirected graph associated to the model specified by the argument 'nrho'. If an object with class 'gof' is passed by 'gof', the undirected graph of the model selected by the function 'aic', 'bic' or 'ebic' is produced.

Value

If typeplot = "graph" then the igraph object is returned (see example below).

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

See Also

glasso, mglasso and cglasso.

Examples

```
library("cglasso")
set.seed(123)

#################
# cglasso model #
#################
n <- 100L
p <- 5L
mu <- rep.int(0L, times = p)
X <- rdatacggm(n = n, mu = mu, probr = 0.05)
out <- cglasso(X = X, nrho = 100L)
out_aic <- aic(out)

# plotting the regularization paths + 'gof' object
plot(out, typeplot = "path")
plot(out, typeplot = "path", gof = out_aic)

# plotting the graph associated to the fitted model
# specified by 'nrho'
out_graph <- plot(out, typeplot = "graph", nrho = 10L)
out_graph
# plotting the graph associated to the fitted model
# specified by 'gof'
out_graph <- plot(out, typeplot = "graph", gof = out_aic)
out_graph
```

```
#################
# mglasso model #
#################
R <- event(X)
X <- as.matrix(X)
X[R == 1L] <- NA
out <- mglasso(X = X, nrho = 100L)
out_aic <- aic(out)

# plotting the regularization paths + 'gof' object
plot(out, typeplot = "path")
plot(out, typeplot = "path", gof = out_aic)

# plotting the graph associated to the fitted model
# specified by 'nrho'
out_graph <- plot(out, typeplot = "graph", nrho = 10L)
out_graph
# plotting the graph associated to the fitted model
# specified by 'gof'
out_graph <- plot(out, typeplot = "graph", gof = out_aic)
out_graph

#################
# glasso model #
#################
X <- MASS::mvrnorm(n = n, mu = mu, Sigma = diag(p))
out <- glasso(X, nrho = 100L)
out_aic <- aic(out)

# plotting the regularization paths + 'gof' object
plot(out, typeplot = "path")
plot(out, typeplot = "path", gof = out_aic)

# plotting the graph associated to the fitted model
# specified by 'nrho'
out_graph <- plot(out, typeplot = "graph", nrho = 10L)
out_graph
# plotting the graph associated to the fitted model
# specified by 'gof'
out_graph <- plot(out, typeplot = "graph", gof = out_aic)
out_graph
```

`rdatacggm` *Simulate from a Censored Gaussian Graphical Model*

Description

'`rdatacggm`' function is used to produce one or more samples from the specified censored Gaussian graphical model.

Usage

```
rdatacggm(n, mu, Sigma, probl, probr, lo, up, ...)
```

Arguments

n	the number of samples required.
mu	a vector giving the means of the variables. By default all the expected vaules are equal to zero.
Sigma	a positive-definite symmetric matrix specifying the covariance matrix of the variables. By default the identity matrix is used as covariance matrix.
probl	a vector giving the probabilities that the random variables are left-censored.
probr	a vector giving the probabilities that the random variables are right-censored.
lo	a vector giving the left-censoring values.
up	a vector giving the right-censoring values.
...	futher arguments passed to the function 'mvrnorm'.

Details

'rdatacggm' function simulates a dataset from a censored Gaussian graphical model and returns an object with class 'datacggm'.

The dataset is simulated in two steps:

1. in the first step the arguments n, mu, Sigma and ... are passed to the function mvrnorm to simulate one or more samples from the specified multivariate Gaussian distribution.
2. in the second step, the values that are below or upper the censoring values are replaced.

The user can specify the censoring values in two equivalent ways. The simplest way is to use the arguments lo and up; a warning is produced if a full-observed dataset is simulated (see the last example). Alternatively, the censoring values can be implicitly specified using the arguments probl and probr. The jth lower censoring value, denoted by l_j, is such that:

$$\texttt{probl[j]} = \Pr\{X_j \leq l_j\}.$$

In the same way, the jth upper censoring value, denoted by u_j, is such that:

$$\texttt{probr[j]} = \Pr\{X_j \geq u_j\}.$$

Value

rdatacggm returns an object with class 'datacggm'. See datacggm for more details.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

References

Augugliaro, L., Abbruzzo, A. and Vinciotti, V. (2018). ℓ_1-Penalized gaussian graphical model. *Biostatistics* (to appear).

See Also

datacggm, event, cglasso and the method function summary.datacggm.

Examples

```
set.seed(123)

n <- 1000L
p <- 3L
mu <- rep(1L, p)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))

# simulating a dataset from a left-censored Gaussian graphical model
X <- rdatacggm(n = n, mu = mu, Sigma = diag(p), probl = 0.05)
# the same: X <- rdatacggm(n = n, mu = mu, Sigma = Sigma, probl = 0.05, probr = 0)
# the same: X <- rdatacggm(n = n, mu = mu, Sigma = Sigma, probl = 0.05, up = +Inf)
summary(X)

# simulating a dataset from a right-censored Gaussian graphical model
X <- rdatacggm(n = n, mu = mu, Sigma = diag(p), probr = 0.05)
# the same: X <- rdatacggm(n = n, mu = mu, Sigma = Sigma, probr = 0.05, probl = 0)
# the same: X <- rdatacggm(n = n, mu = mu, Sigma = Sigma, probr = 0.05, lo = -Inf)
summary(X)

# simulating a dataset from a censored Gaussian graphical model
X <- rdatacggm(n = n, mu = mu, Sigma = Sigma, probr = 0.05, probl = 0.05)
summary(X)

# simulating a full observed dataset: a warning is produced
X <- rdatacggm(n = n, mu = mu, Sigma = Sigma, lo = -10, up = 10)
summary(X)
```

---

scale.datacggm *Scaling and Centering of "datacggm" Objects*

Description

The method function scale.datacggm centers and/or scales the columns of a numeric matrix storaged in a 'datacggm' object.

Usage

```
## S3 method for class 'datacggm'
scale(x, center = TRUE, scale = TRUE)
```

Arguments

x	an object of class 'datacggm'.
center	either a logical value or numeric-alike vector of length equal to the number of columns of x, where 'numeric-alike' means that as.numeric(.) will be applied successfully if is.numeric(.) is not true.
scale	either a logical value or a numeric-alike vector of length equal to the number of columns of x.

Details

The value of center determines how column centering is performed. If center is a numeric-alike vector with length equal to the number of columns of x, then each column of x has the corresponding value from center subtracted from it. If center is TRUE then centering is done by subtracting the column means (omitting censoring values) of x$X from their corresponding columns, and if center is FALSE, no centering is done. The same is done for x$lo and x$up.

The value of scale determines how column scaling is performed (after centering). If scale is a numeric-alike vector with length equal to the number of columns of x, then each column of x$X is divided by the corresponding value from scale. If scale is TRUE then scaling is done by dividing the (centered) columns of x$X by their standard deviations if center is TRUE, and the root mean square otherwise. If scale is FALSE, no scaling is done. The same is done for x$lo and x$up.

The root-mean-square for a (possibly centered) column is defined as $\sqrt{\sum(x^2)/(n-1)}$, where x is a vector of observed values and n is the number of observed values. In the case center = TRUE, this is the same as the standard deviation, but in general it is not. (To scale by the standard deviations without centering, use scale(x,center = FALSE,scale = apply(x,2,sd,na.rm = TRUE)).)

Value

The method function 'scale.datacggm' returns an object of class datacggm. The numeric centering and scalings used (if any) are returned as attributes "scaled:center" and "scaled:scale".

See Also

datacggm.

Examples

```
set.seed(123)

n <- 100L
p <- 3L
mu <- rep(1L, p)
X <- rdatacggm(n = n, mu = mu, probr = 0.05, probl = 0.05)
centered.X <- scale(X)
centered.X
attr(centered.X, "scaled:center")
attr(centered.X, "scaled:scale")
```

summary *Summary Method*

Description

'summary' produces a summary of the sequence of fitted models.

Usage

```
## S3 method for class 'glasso'
summary(object, ..., gof = c("BIC", "AIC", "eBIC"), par.gof, digits = 4L)
```

Arguments

object	an object of class 'glasso', 'ggm', 'cglasso' or 'cggm'.
gof	string specifying the measure of goodness-of-fit used to evaluate the fitted models. Default is 'BIC'.
par.gof	the parameter of the measure of goodness-of-fit used to evaluate the fitted models.
digits	the minimum number of significant digits to be used: see 'print.default'.
...	further arguments passed to the method function 'print.data.frame'.

Details

The method function summary.glasso gives information about the sequence of fitted models. The output is divided in two sections.

First section shows the call that produced object followed by a data.frame reporting the values of the tuning parameter used to fit the model (rho), the number of non-zero estimates (df), the values of the pseudo R-squared (R2) describted in glasso, mglasso and cglasso, the values of the measure of goodness-of-fit used to evaluate the fitted models and the ranking of the fitted models (Rank). The model with the lowest measure of goodness-of-fit is pointed out by an arrow.

Second section shows the details of the selected model plus the number of connected components and the number of vertices per component.

Value

The function 'summary.glasso' computes and returns a list of summary statistics with the following elements:

table	a data.frame containing the summary statistics used to evaluate the sequence of fitted models.
which.min	the number of the model with the lowest measure of goodness-of-fit.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

See Also

link{glasso}, mglasso, cglasso, mle, aic bic and ebic.

Examples

```
library("cglasso")
set.seed(123)

#################
# cglasso model #
#################
n <- 100L
p <- 5L
mu <- rep.int(0L, times = p)
X <- rdatacggm(n = n, mu = mu, probr = 0.05)
out <- cglasso(X = X)
summary(out, gof = "AIC")
summary(out, gof = "BIC")
summary(out, gof = "eBIC")

##############
# cggm model #
##############
out_mle <- mle(out)
summary(out_mle, gof = "AIC")
summary(out_mle, gof = "BIC")
summary(out_mle, gof = "eBIC")

#################
# cglasso model #
#################
R <- event(X)
X <- as.matrix(X)
X[R == 1L] <- NA
out <- mglasso(X = X)
summary(out, gof = "AIC")
summary(out, gof = "BIC")
summary(out, gof = "eBIC")

##############
# mggm model #
##############
out_mle <- mle(out)
summary(out_mle, gof = "AIC")
summary(out_mle, gof = "BIC")
summary(out_mle, gof = "eBIC")

#################
# glasso model #
#################
X <- MASS::mvrnorm(n = n, mu = mu, Sigma = diag(p))
out <- glasso(X)
```

```
summary(out, gof = "AIC")
summary(out, gof = "BIC")
summary(out, gof = "eBIC")

#############
# ggm model #
#############
out_mle <- mle(out)
summary(out_mle, gof = "AIC")
summary(out_mle, gof = "BIC")
summary(out_mle, gof = "eBIC")
```

---

summary.datacggm *Summarizing Objects of Class* 'cggmdata'

Description

'summary.datacggm' function is used to produce summaries of an object of class 'datacggm'.

Usage

```
## S3 method for class 'datacggm'
summary(object, ..., digits = max(3L, getOption("digits") - 3L))
```

Arguments

object	an object of class 'datacggm'.
digits	integer used for number formatting with 'format()'.
...	further arguments passed to 'format()'.

Details

The method function 'summary.datacggm' extends the results given by 'summary.matrix()' taking into account the censoring values. For each variable, the mean and the quartiles are computing using only the observed values; the lower and upper censoring values (denoted by 'Lower' and 'Upper') are also reported. The number of observed and censured values are computed and showed in the second part of the output (see example below).

Value

'summary.dataccgm' returns a matrix with class 'table', obtained by computing the summary measures to each variable and collating the results.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

References

Augugliaro, L., Abbruzzo, A. and Vinciotti, V. (2018). ℓ_1-Penalized gaussian graphical model. *Biostatistics* (to appear).

See Also

datacggm and rdatacggm.

Examples

```
set.seed(123)
library("cglasso")

n <- 1000L
p <- 3L
mu <- rep(1L, p)
rho <- 0.3
Sigma <- outer(1:p, 1:p, function(i, j) rho^abs(i - j))
X <- rdatacggm(n = n, mu = mu, Sigma = Sigma, probr = 0.05, probl = 0.05)
summary(X)
```

---

to_graph *Create Undirected Graphs*

Description

'to_graph' function is used to create an undirected graph from a fitted model object.

Usage

```
to_graph(object, nrho = 1L, weighted = FALSE, isolated = FALSE)
```

Arguments

object	a fitted model object.
nrho	integer specifying the model used to create the undireced graph. Default is 1L.
weighted	flag specifying whether to create a weighted graph from the adjacency matrix. Default is FALSE.
isolated	flag specifying whether the isolated vertices are removed from the graph. Default is FALSE, i.e. the isolated vertices are removed.

Details

The adjacency matrix 'object$Adj[,,nrho]' is passed to graph_from_adjacency_matrix, available in the R package **igraph**, to create the undirected graph estimated by the glasso, mglasso, cglasso model or using the function mle. If 'weighted = TRUE' then the precision matrix 'object$Tht[,,nrho]' is used to create a weighted undirected graph. In this case, an edge associated to a negative partial correlation coefficient is plotted using a dashed line (see examples below).

Value

'to_graph' returns an igraph object.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)

References

Augugliaro, L., Abbruzzo, A. and Vinciotti, V. (2018). ℓ_1-Penalized gaussian graphical model. *Biostatistics* (to appear).

See Also

glasso, codemglasso and codecglasso. For more details about the object of class 'igraph', the interested reader is referred to the package **igraph**.

Examples

```
library("cglasso")
set.seed(123)

#################
# cglasso model #
#################
n <- 100L
p <- 10L
mu <- rep.int(0L, times = p)
X <- rdatacggm(n = n, mu = mu, probr = 0.05)
out <- cglasso(X)
out

# creating the undirected graph associated to the fitted
# model number 3
graph_cglasso <- to_graph(out, nrho = 3L)
graph_cglasso
V(graph_cglasso)
E(graph_cglasso)
plot(graph_cglasso, layout = layout_in_circle(graph_cglasso))

# creating the weighted graph associated to the fitted
# model number 3
graph_cglasso <- to_graph(out, nrho = 3L, weighted = TRUE)
graph_cglasso
E(graph_cglasso)
E(graph_cglasso)$weight
plot(graph_cglasso, layout = layout_in_circle(graph_cglasso))

#################
# mglasso model #
#################
R <- event(X)
```

```
X <- as.matrix(X)
X[R == 1L] <- NA
out <- mglasso(X)
out

# creating the undirected graph associated to the fitted
# model number 3
graph_mglasso <- to_graph(out, nrho = 3L)
graph_mglasso
V(graph_mglasso)
E(graph_mglasso)
plot(graph_mglasso, layout = layout_in_circle(graph_mglasso))

# creating the weighted graph associated to the fitted
# model number 3
graph_mglasso <- to_graph(out, nrho = 3L, weighted = TRUE)
graph_mglasso
E(graph_mglasso)
E(graph_mglasso)$weight
plot(graph_mglasso, layout = layout_in_circle(graph_mglasso))

##################
# glasso model #
##################
n <- 100L
p <- 10L
X <- matrix(rnorm(n * p), nrow = n, ncol = p)
out <- glasso(X)
out

# creating the undirected graph associated to the fitted
# model number 3
graph_cglasso <- to_graph(out, nrho = 3L)
graph_cglasso
V(graph_cglasso)
E(graph_cglasso)
plot(graph_cglasso, layout = layout_in_circle(graph_cglasso))

# creating the weighted graph associated to the fitted
# model number 3
graph_cglasso <- to_graph(out, nrho = 3L, weighted = TRUE)
graph_cglasso
E(graph_cglasso)
E(graph_cglasso)$weight
plot(graph_cglasso, layout = layout_in_circle(graph_cglasso))
```

Index