Package ‘cglasso’

February 12, 2024

Version 2.0.7
Date 2024-02-10
Type Package
Title Conditional Graphical LASSO for Gaussian Graphical Models with Censored and Missing Values
Depends R (>= 3.6.0), igraph
Description Conditional graphical lasso estimator is an extension of the graphical lasso proposed to estimate the conditional dependence structure of a set of p response variables given q predictors. This package provides suitable extensions developed to study datasets with censored and/or missing values. Standard conditional graphical lasso is available as a special case. Furthermore, the package provides an integrated set of core routines for visualization, analysis, and simulation of datasets with censored and/or missing values drawn from a Gaussian graphical model. Details about the implemented models can be found in Augugliaro et al. (2023) <doi:10.18637/jss.v105.i01>, Augugliaro et al. (2020b) <doi:10.1007/s11222-020-09945-7>, Augugliaro et al. (2020a) <doi:10.1093/biostatistics/kxy043>, Yin et al. (2001) <doi:10.1214/11-AOAS494> and Stadler et al. (2012) <doi:10.1007/s11222-010-9219-7>.

Imports methods, MASS
License GPL (>= 2)
LazyLoad yes
NeedsCompilation yes
Author Luigi Augugliaro [aut, cre] (<https://orcid.org/0000-0002-4603-7541>), Gianluca Sottile [aut] (<https://orcid.org/0000-0001-9347-7251>), Ernst C. Wit [aut] (<https://orcid.org/0000-0002-3671-9610>), Veronica Vinciotti [aut] (<https://orcid.org/0000-0002-2625-7977>)
Maintainer Luigi Augugliaro <luigi.augugliaro@unipa.it>
Repository CRAN
Date/Publication 2024-02-12 08:40:02 UTC

R topics documented:

cglasso-package .......................................................... 2
Description

Conditional graphical lasso (cglasso) estimator (Yin and other, 2011) is an extension of the graphical lasso (Yuan and other, 2007) proposed to estimate the conditional dependence structure of a set of p response variables given q predictors. This package provides suitable extensions developed to study datasets with censored and/or missing values (Augugliaro and other, 2020a and Augugliaro and other, 2020b). Standard conditional graphical lasso is available as a special case. Furthermore, the cglasso package provides an integrated set of core routines for visualization, analysis, and simulation of datasets with censored and/or missing values drawn from a Gaussian graphical model.

Details

Package: cglasso
Type: Package
Version: 2.0.6
Date: 2023-01-17
License: GPL (>=2)

Author(s)

Luigi Augugliaro [aut, cre],
Gianluca Sottile [aut]
Ernst C. Wit [aut]
Veronica Vinciotti [aut]
Maintainer: Luigi Augugliaro <luigi.augugliaro@unipa.it>

References


Description

‘AIC’ computes the ‘Akaike Information Criterion’.

Usage

## S3 method for class 'cglasso'
AIC(object, k = 2, mle, ...)

Arguments

object  an R object inheriting class ‘cglasso’, that is, the output of the model-fitting functions cglasso and cggm.
k  the penalty parameter to be used; the default k = 2 is the classical AIC.
mle  logical. TRUE if the measure of goodness-of-fit should be computed using the maximum likelihood estimates. Default depends on the class of the argument object: mle = FALSE for objects of class cglasso and mle = TRUE for objects of class cggm.
...  further arguments passed to cggm.

Details

‘AIC’ computes the following measure of goodness-of-fit (Ibrahim and other, 2008):

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

where \(k\) is the penalty parameter and df represents the number of unique non-zero parameters in the fitted model.

The values of the Q-function function are computed using QFun. By default, for an object of class cglasso these values are computed using the penalized estimates whereas maximum likelihood estimates are used if the object is of class cggm (see argument ‘mle’ in QFun).

The Akaike Information Criterion (AIC) is returned by letting \(k = 2\) (default value of the function AIC) whereas the ‘Bayesian Information Criterion’ (BIC) is returned by letting \(k = \log(n)\), where \(n\) is the sample size.

Function AIC can be passed to the functions select_cglasso and summary.cglasso to select and print the optimal fitted model, respectively.

The function plot.GoF can be used to graphically evaluate the behaviour of the fitted models in terms of goodness-of-fit.
Value

`AIC` return an R object of S3 class “GoF”, i.e., a named list containing the following components:

- `value_gof`: a matrix storing the values of the measure of goodness-of-fit used to evaluate the fitted models.
- `df`: a matrix storing the number of the estimated non-zero parameters.
- `dfB`: a matrix storing the number of estimated non-zero regression coefficients.
- `dfTht`: a matrix storing the number of estimated non-zero partial correlation coefficients.
- `value`: a matrix storing the values of the Q-function.
- `n`: the sample size.
- `p`: the number of response variables.
- `q`: the number of columns of the design matrix $X$ used to fit the model.
- `lambda`: the $\lambda$-values used to fit the model.
- `nlambda`: the number of $\lambda$-values used.
- `rho`: the $\rho$-values used to fit the model.
- `nrho`: the number of $\rho$-values used.
- `type`: a description of the computed measure of goodness-of-fit.
- `model`: a description of the fitted model passed through the argument `object`.

Author(s)

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

References


See Also

`BIC.cglasso`, `cglasso`, `cggm`, `QFun`, `plot.GoF` and `summary.cglasso`

Examples

```r
set.seed(123)
# Y ~ N(b0 + XB, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
```
```r
q <- 2L
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
AIC(out)
out.mle <- cggm(out, lambda.id = 3L, rho.id = 3L)
AIC(out.mle)
```

### Description

‘BIC’ computes the Bayesian Information Criterion.

### Usage

```r
## S3 method for class 'cglasso'
BIC(object, g = 0, type, mle, ...)
```

### Arguments

- **object**: an R object inheriting class ‘cglasso’, that is, the output of the model-fitting functions `cglasso` and `cggm`.
- **g**: a value belonging to the interval [0, 1]. Classical BIC is returned by letting g = 0 (default value), whereas extended BIC corresponds to the case g = 0.5.
- **type**: character; if g is not zero then the measure proposed in Foygel and other (2010) is returned by setting type = "FD", otherwise (type = "CC") returns the measure proposed in Chen and other (2008, 2012). See section ‘Details’ for more details.
- **mle**: logical. TRUE if the measure of goodness-of-fit should be computed using the maximum likelihood estimates. Default depends on the class of the argument object: mle = FALSE for objects of class cglasso and mle = TRUE for objects of class cggm.
- **...**: further arguments passed to the model-fitting function `cggm`.

### Details

‘BIC’ computes the Bayesian Information Criterion (BIC) for models fitted by `cglasso` or `cggm`. As proposed in Ibrahim and other (2008), BIC computes the measure of goodness-of-fit by replacing the log-likelihood function with the Q-function, that is, the function maximized in the M-Step of the EM-algorithm. The values of the Q-function are computed using QFun. By default, for an object
of class cglasso these values are computed using the penalized estimates whereas, if the object has class cggm, maximum likelihood estimates are used (see argument ‘mle’ in QFun).

By default, BIC computes the standard BIC measure ($\gamma = 0$):

$$-2 \text{Q-function} + \log(n) \text{ df},$$

where $n$ is the sample size and df represents the number of unique non-zero parameters in the fitted model.

If $\gamma \neq 0$, the default depends on the number of predictors ($q$).

If $q = 0$, BIC computes the measure of goodness-of-fit proposed in Foygel and other (2010) (type = "FD"):

$$eBIC = -2 \text{QFun} + \log(n + 4 \gamma \log p) \text{ df},$$

where $\gamma$ is a value belonging to the interval $[0, 1]$ and indexing the measure of goodness-of-fit.

If $q \neq 0$, BIC computes the measure of goodness-of-fit proposed in Chen and other (2008, 2012) (type = "CC"):

$$eBIC = -2 \text{QFun} + \log(n + 2 \gamma \log q) \text{ df},$$

BIC can be passed to the functions `select_cglasso` and `summary.cglasso` to select and print the optimal fitted model, respectively.

The function `plot.GoF` can be used to graphically evaluate the behaviour of the fitted models in terms of goodness-of-fit.

**Value**

‘BIC’ returns an R object of S3 class “GoF”, i.e. a named list containing the following components:

- `value_gof` a matrix storing the values of the measure of goodness-of-fit used to evaluate the fitted models.
- `df` a matrix storing the number of the estimated non-zero parameters.
- `dfB` a matrix storing the number of estimated non-zero regression coefficients.
- `dfThet` a matrix storing the number of estimated non-zero partial correlation coefficients.
- `value` a matrix storing the values of the Q-function.
- `n` the sample size.
- `p` the number of response variables.
- `q` the number of columns of the design matrix $X$ used to fit the model.
- `lambda` the $\lambda$-values used to fit the model.
- `nlambda` the number of $\lambda$-values used.
- `rho` the $\rho$-values used to fit the model.
- `nrho` the number of $\rho$-values used.
- `type` a description of the computed measure of goodness-of-fit.
- `model` a description of the fitted model passed through the argument object.
Author(s)

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

References


See Also

cglasso, cggm, AIC.cglasso, QFun, plot.GoF and summary.cglasso

Examples

```
set.seed(123)

# Y ~ N(0, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
BIC(out) # standard BIC measure
BIC(out, mle = TRUE, g = 0.5, type = "FD") # eBIC proposed in Foygel and other (2010)

# Y ~ N(b0 + XB, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
BIC(out) # standard BIC measure
BIC(out, mle = TRUE, g = 0.5, type = "FD") # eBIC proposed in Foygel and other (2010)
BIC(out, mle = TRUE, g = 0.5, type = "CC") # eBIC proposed in Chen and other (2008, 2010)
```
Description

`cggm` is used to perform post-hoc maximum likelihood refitting of a selected conditional graphical lasso model with censored and/or missing values.

Usage

```r
cggm(object, GoF = AIC, lambda.id, rho.id, tp.min = 1.0E-6, ntp = 100L,
    maxit.em = 1.0E+4, thr.em = 1.0E-3, maxit.bcd = 1.0E+5, thr.bcd = 1.0E-4,
    trace = 0L, ...)
```

Arguments

- **object**: an R object of S3 class `cglasso`, that is, the output of the function `cglasso`.
- **GoF**: a valid goodness-of-fit function, such as `AIC.cglasso` or `BIC.cglasso`, or an R object of class `GoF`.
- **lambda.id**: an optional integer used to identify a λ-value stored in `object`. See section 'Details' for more details.
- **rho.id**: an optional integer used to identify a ρ-value stored in `object`. See section 'Details' for more details.
- **tp.min**: the smallest λ and ρ value. See section 'Details' for more details.
- **ntp**: integer; number of λ- and ρ-values used to compute the coefficients path. See section 'Details' for more details.
- **maxit.em**: maximum number of iterations of the EM algorithm. Default is 1.0E+4.
- **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+5.
- **thr.bcd**: threshold for the convergence of the glasso algorithm. Default is 1.0E-4.
- **trace**: integer for printing information out as iterations proceed: trace = 0 no information is printed out on screen; trace = 1 minimal information is printed; trace = 2 detailed information is printed on screen.

Details

The model-fitting function `cggm` is used to obtain the maximum likelihood estimates of a censored Gaussian graphical model whose structure was found by penalised cglasso inference. That is, given a fitted cglasso model, identified along the path via a goodness-of-fit function (passed by `GoF`) or via the optional arguments `lambda.id` and `rho.id`, `cggm` computes the maximum likelihood estimates of the parameters with zero constraints associated to the given structure.
Maximum likelihood estimates are computed using cglasso as workhorse function, that is, cgmm fits a sequence of cglasso models, with length equal to ntp, reducing $\lambda$ and $\rho$ until they are equal to the value $\min(tp.min, \lambda.min, \rho.min)$, where lambda.min and rho.min are the smallest $\lambda$- and $\rho$-values stored in object. Maximum likelihood estimates are obtained from the last fitted cglasso model.

The model-fitting function cgmm returns an R object inheriting the S3 class ‘cglasso’, for which all printing and plotting functions, designed for a cglasso object, can be used (see section ‘See Also’). Function ShowStructure can be used to show the structure of the package.

NOTE: if the user tries a problem when the sample size is not large enough, then increase the number of fitted cglasso models (argument ntp).

Value

cgmm returns an object of S3 class “cgmm”, i.e., a list containing the following components:

- `call` the call that produced this object.
- `Yipt` an array storing the ‘working response matrix’.
- `B` the maximum likelihood estimate of the regression coefficient matrix.
- `mu` the fitted expected values.
- `R` the ‘working residuals’ matrix.
- `S` the ‘working empirical covariance matrix’.
- `Sgm` the maximum likelihood estimate of the covariance matrix.
- `Tht` the maximum likelihood estimate of the precision matrix.
- `dfB` the number of estimated non-zero regression coefficients. Only for internal purpose.
- `dfTht` the number of estimated non-zero (off-diagonal) partial correlation coefficients. Only for internal purpose.
- `InfoStructure` a named list whose elements are used to store the information about the estimated networks. Only for internal purpose.
- `nit` the number of EM steps.
- `Z` the ‘datacgmm’ object used to compute the censored graphical lasso estimator.
- `nlambda` Only for internal purpose.
- `lambda` the $\lambda$-value of the selected cglasso model.
- `nrho` Only for internal purpose.
- `rho` the $\rho$-value of the selected 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.
- `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 information on screen.
nobs the sample size
nresp the number of response variables used to fit the model.
npred the number of predictors used to fit the model.

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

See Also
cglasso, coef.cglasso, fitted.cglasso, residuals.cglasso, predict.cggm, impute, AIC.cglasso, BIC.cglasso, summary.cglasso, to_graph, plot.cglasso2igraph and ShowStructure.

Examples
set.seed(123)
# Y ~ N(XB, Sigma) and
# 1. probability of left/right censored values equal to 0.05
# 2. probability of missing-at-random values equal to 0.05
n <- 100L
p <- 3L
q <- 2L
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.5, probna = 0.05)
out <- cglasso(. ~ ., data = Z)
# MLE of the censored Gaussian graphical model identified by 'BIC'
out.mle <- cggm(out, GoF = BIC)
out.mle

# accessor functions
coef(out.mle, drop = TRUE)
fitted(out.mle, drop = TRUE)
residuals(out.mle, type = "working", drop = TRUE)
impute(out.mle, type = "both")

# goodness-of-fit functions
AIC(out.mle)
BIC(out.mle)
summary(out.mle)

# network analysis
out.graph <- plot(out.mle)
out.graph
Conditional Graphical Lasso Estimator

Description

‘cglasso’ fits the conditional graphical lasso model to datasets with censored and/or missing values.

Usage
cglasso(formula, data, subset, contrasts = NULL, diagonal = FALSE, weights.B = NULL, weights.Tht = NULL, nlambdas, lambda.min.ratio, lambda, nrho, rho.min.ratio, rho, maxit.em = 1.0E+4, thr.em = 1.0E-3, maxit.bcd = 1.0E+5, thr.bcd = 1.0E-4, trace = 0)

Arguments

formula

an object of class ‘formula’ (or one that can be coerced to that class): a symbolic description of the model to be fitted.

data

an R object of S3 class ‘datacggm’, that is, the output of the function datacggm. See section ‘Description’ for more details.

subset

an optional vector specifying a subset of observations to be used in the fitting process.

contrasts

an optional list. See the contrasts.arg of model.matrix.default.

diagonal

logical. Should diagonal entries of the concentration matrix be penalized? Default is ‘diagonal = FALSE’.

weights.B

an optional q × p dimensional matrix of non-negative weights used to penalize the regression coefficients (intercepts are unpenalized). This matrix can be also used to specify the unpenalized regression coefficients (’weights.B[i, j] = 0’) or the structural zeros (’weights.B[i, j] = +Inf’). By default, all weights are set equal to 1, meaning that the penalized regression coefficients are unweighted.

weights.Tht

an optional symmetric matrix of non-negative weights used to penalize the partial regression coefficients. This matrix can be used to specify the unpenalized partial correlation coefficients (’weights.Tht[i, j] = 0’) or the structural zeros in the precision matrix (’weights.Tht[i, j] = +Inf’). By default, the off-diagonal entries of the matrix ’weights.Tht’ are set equal to 1, meaning that the partial correlation coefficients are unweighted, whereas the diagonal entries are equal to 0, that is the diagonal entries of the precision matrix are unpenalized.

nlambdas

integer. The number of λ-values used to penalize the regression coefficients. By default ‘nlambdas = 10’.

lambda.min.ration

the smallest λ-value is defined as a fraction of ‘lambda.max’ (i.e., the smallest λ-value for which all the estimated regression coefficients are equal to zero). The default depends on the sample size ‘n’ relative to the number of predictors ‘q’. If ‘q < n’, default is ‘1.0E-6’ otherwise the value ‘1.0E-2’ is used as default.
A very small value of 'lambda.min.ratio' will lead to a saturated fitted model in the 'q < n' case.

**lambda**

an optional user-supplied decreasing sequence of λ-values. By default cglasso computes a sequence of λ-values using nlambda and lambda.min.ratio. If lambda is supplied then nlambda and lambda.min.ratio are overwritten. WARNING: use with care and avoid supplying a single λ-value; supply instead a decreasing sequence.

**nrho**

integer. The number of ρ-values used to penalize the partial correlation coefficients. By default 'nrho = 10'.

**rho.min.ratio**

the smallest ρ-value is defined as a fraction of 'rho.max' (i.e., the smallest ρ-value for which all the estimated partial correlation coefficients are equal to zero). The default depends on the sample size 'n' relative to the number of response variables 'p'. If 'p < n', the default is '1.0E-6' 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**

an optional user supplied decreasing sequence of ρ-values. By default cglasso computes a sequence of ρ-values using nrho and rho.min.ratio. If rho is supplied then nrho and rho.min.ratio are overwritten. WARNING: use with care and avoid supplying a single ρ-value; supply instead a decreasing sequence.

**maxit.em**

maximum number of iterations of the EM algorithm. Default is 1.0E+4.

**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+5.

**thr.bcd**

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

**trace**

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

**Details**

cglasso is the main model-fitting function and can be used to fit a broad range of extensions of the glasso estimator (Friedman and other, 2008). It is specifically proposed to study datasets with censored and/or missing response values. To help the user, the cglasso function has been designed to automatically select the most suitable extension by using the information stored in the 'datacggm' object passed through Z.

Below we sum up the available extensions:

- if only left/right-censored are observed in the response matrix Y (without missing values) and no predictor matrix is stored in Z, then cglasso computes the censored glasso estimator (Augugliaro and other, 2020a);
- if only left/right-censored are observed in the response matrix Y (without missing values) and a predictor matrix X is stored in Z, then cglasso computes the conditional censored glasso estimator (Augugliaro and other, 2020b);
- if only missing values are stored in the response matrix Y (without censored values), then cglasso computes the missglasso estimator (Stadler and other, 2012);
- starting with version 2.0.0, cglasso can also handle datasets with both missing and censored response values.
See section ‘Examples’ for some example.

The model-fitting function cglasso returns an R object of S3 class ‘cglasso’ for which there are available a set of accessor functions, a set of functions designed to evaluate the goodness-of-fit of the fitted models and, finally, a set of functions developed to analyze the selected network. The function `ShowStructure` can be used to show the structure of the package.

The accessor functions `coef.cglasso`, `fitted.cglasso`, `residuals.cglasso`, `predict.cglasso` and `impute` can be used to extract various useful features of the object fitted by cglasso.

For an R object returned by cglasso, the functions `AIC.cglasso` and `BIC.cglasso` can be used to evaluate the goodness-of-fit of the fitted models. Usually, these functions are used together with the function `summary.cglasso`, which gives more information about the sequence of fitted models. The plotting function `plot.GoF` can be used to graphically identify the optimal pair of the tuning parameters.

Given a pair of the tuning paremeters, the functions `to_graph` and `plot.cglasso2igraph` can be used to analyze and show the selected network. Finally, the function `cggm` can be used to produce post-hoc maximum likelihood refitting of the selected graphical model.

Value

cglasso returns an object of S3 class “cglasso”, i.e., a named list containing the following components:

call the call that produced this object.

Yipt an array of dimension ‘n x p x nlambda x nrho’ storing the ‘working response matrices’, that is, Yipt[, , i, j] is used as response matrix to compute the multilasso estimator (Augugliaro and other, 2020b). The accessor function ‘impute’ can be used to extract the desidered imputed matrix.

B an array of dimension ‘(q + 1) x p x nlambda x nrho’ storing the penalized estimate of the regression coefficient matrices. The accessor function ‘coef.cglasso’ can be used to extract the desired estimates.

mu an array of dimension ‘n x p x nlambda x nrho’ storing the fitted values. The accessor function ‘fitted.cglasso’ can be used to extract the desired fitted values.

R an array of dimension ‘n x p x nlambda x nrho’ storing the ‘working residuals’, that is, R[, , i, j] is defined as the difference between Yipt[, , i, j] and mu[, , i, j]. The accessor function ‘residuals.cglasso’ can be used to extract the desidered residual matrix.

S an array of dimension ‘p x p x nlambda x nrho’ storing the ‘working empirical covariance matrices’, that is, ‘S[, , i, j]’ is used to compute the glasso estimator.

Sgm an array of dimension ‘p x p x nlambda x nrho’ storing the estimated covariance matrices. The accessor function ‘coef’ can be used to extract the desired estimates.

Tht an array of dimension ‘p x p x nlambda x nrho’ storing the estimated precision matrices. The accessor function ‘coef’ can be used to extract the desired estimates.
dfB a matrix of dimension \((p + 1) \times n\lambda x n\rho\) storing the number of estimated non-zero regression coefficients. Only for internal purpose.

dfTht a matrix of dimension \(n\lambda x n\rho\) storing the number of estimated non-zero (off-diagonal) partial correlation coefficients. Only for internal purpose.

InfoStructure a named list whose elements contain information about the estimated networks. Only for internal purpose.

nit an array of dimension \(2 \times n\lambda x n\rho\) storing the number of EM steps.

Z the ‘dataCGGM’ object used to compute the censored graphical lasso estimator.

diagonal the flag used to specify if the diagonal entries of the precision matrix are penalized.

weights.B the matrix of non-negative weights used for the regression coefficients.

weights.Tht the matrix of non-negative weights used for the precision matrix.

n\lambda the number of \(\lambda\)-values used.

lambda.min.ratio the value used to compute the smallest \(\lambda\)-value.

lambda the sequence of \(\lambda\)-values used to fit the model.

n\rho the number of \(\rho\)-values used.

rho.min.ratio the value used to compute the smallest \(\rho\)-value.

rho the sequence of \(\rho\)-values used to fit the model.

model a description of the fitted 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.

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 information on screen.

nobs the sample size

nresp the number of response variables used to fit the model.

npred the number of predictors used to fit the model.

Author(s)

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


See Also
dataCggm, coef.cglasso, fitted.cglasso, residuals.cglasso, predict.cglasso, impute, AIC.cglasso, BIC.cglasso, summary.cglasso, select.cglasso, plot.GoF, to_graph, plot.cglasso2igraph, cggm and ShowStructure.

Examples

```r
set.seed(123)

# Model 1: censored glasso estimator (Augugliaro *and other*, 2020a)
# Y ~ N(0, Sigma) and probability of left/right censored values equal to 0.05
n <- 1000L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)

# Model 2: conditional censored glasso estimator (Augugliaro *and other*, 2020b)
# Y ~ N(b0 + XB, Sigma) and probability of left/right censored values equal to 0.05
n <- 1000L
p <- 3L
q <- 2L
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)

# Model 3: missglasso estimator (Stadler *and other*, 2012)
```


```r
# Y ~ N(b0 + XB, Sigma) and probability of missing-at-random values equal to 0.05
n <- 1000L
p <- 3L
q <- 2L
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probna = 0.05)
out <- cglasso(. ~ ., data = Z)
out

# Model 4: mixed estimator
# Y ~ N(b0 + XB, Sigma) and
# 1. probability of left/right censored values equal to 0.05
# 2. probability of missing-at-random values equal to 0.05
n <- 1000L
p <- 3L
q <- 2L
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, X = X, b0 = b0, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05, probna = 0.05)
out <- cglasso(. ~ ., data = Z)
out
```

---

**coef**

Extract Model Coefficients

**Description**

The accessor function `coef` extracts model coefficients from an R object inheriting class ’cglasso’.

**Usage**

```r
## S3 method for class 'cglasso'
coef(object, type = c("all", "B", "Sigma", "Theta"), lambda.id, rho.id, drop = TRUE, ...)```

**Arguments**

- `object`: an R object inheriting class ‘cglasso’, that is, the output of the model-fitting functions `cglasso` and `cggm`.
- `type`: a description of the desired estimates.
- `lambda.id`: an optional vector of integers used to specify the λ-values.
rho.id: an optional vector of integers used to specify the $\rho$-values.
drop: logical. Dimensions of the required objects can only be dropped if their extent is one.
...

Value

Coefficients extracted from 'object' are returned. By default, a named list storing all the estimated parameters is returned.

Author(s)

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

See Also

Model-fitting functions cglasso, cggm and the accessor functions fitted.cglasso, residuals.cglasso, predict.cglasso and impute.

Examples

```r
set.seed(123)
# Y ~ N(0, Sigma) and probability of left/right censored values equal to 0.05
n <- 1000L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
coef(out, type = "Theta", rho.id = 1:4)
coef(out, type = "Theta", rho.id = 3, drop = TRUE)

# Y ~ N(b0 + XB, Sigma) and probability of left/right censored values equal to 0.05
n <- 1000L
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
coef(out, type = "B", lambda.id = 3, rho.id = 1:4)
coef(out, type = "B", lambda.id = 3, rho.id = 3, drop = TRUE)
```
Description

Retrieve column means and column variances of a “datacggm” object.

Usage

ColMeans(x)
ColVars(x)

Arguments

x an object of class ‘datacggm’.

Details

For an R object x of class ‘datacggm’, ColMeans (ColVars) retrieves the column means (variances) of the matrices obtained by getMatrix(x, "Y") and getMatrix(x, "X"). For the response variables, marginal means and variances are estimated using a EM-algorithm under the assumption that the p response variables are marginally normally distributed (see also Details section in datacggm). For the numeric predictor variables, marginal means and variances are computed by mean and var, whereas, for categorical data, ColMeans (ColVars) retrieves the statistical mode and the Gini-Simpson Index, respectively.

Value

ColMeans (ColVars) returns a named list with the columns means (variances).

Author(s)

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

See Also

datacggm, rcggm, qqcnorm and hist.datacggm.

Examples

```r
set.seed(123)
n <- 1000L
p <- 3L
b0 <- rep(0, p)
Z <- rcggm(n = n, b0 = b0, probl = 0.05, probr = 0.05)
ColMeans(Z)
ColVars(Z)
n <- 1000L
```
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
Z <- rcggm(n = n, b0 = b0, X = X, B = B, probl = 0.05, probr = 0.05)
ColMeans(Z)
ColVars(Z)

datacggm

Create a Dataset from a Conditional Gaussian Graphical Model with Censored and/or Missing Values

Description

The datacggm function is used to create a dataset from a conditional Gaussian graphical model with censored and/or missing values.

Usage

datacggm(Y, lo = -Inf, up = +Inf, X = NULL, control = list(maxit = 1.0E+4, thr = 1.0E-4))

Arguments

Y

a \((n \times p)\)-dimensional matrix; each row is an observation from a conditional Gaussian graphical model with censoring vectors \(\text{lo}\) and \(\text{up}\). Missing-at-random values are recorded as 'NA'.

lo

the lower censoring vector; \(\text{lo}[j]\) is used to specify the lower censoring value for the random variable \(Y_j\).

up

the upper censoring vector; \(\text{up}[j]\) is used to specify the upper censoring value for the random variable \(Y_j\).

X

an optional \((n \times q)\)-dimensional data frame of predictors. If missing (default), a dataset from a Gaussian graphical model is returned otherwise a dataset from a conditional Gaussian graphical model is returned.

control

a named list used to pass the arguments to the EM algorithm (see below for more details). The components are:

- maxit: maximum number of iterations. Default is \(1.0E+4\).
- thr: threshold for the convergence. Default value is \(1.0E-4\).

Details

The function 'datacggm' returns an \(R\) object of class 'datacggm', that is a named list containing the elements needed to fit a conditional graphical LASSO (cgl-ass) model to datasets with censored and/or missing values.
A set of specific method functions are developed to describe data with censored/missing values. For example, the method function `print.datacggm` prints out the left and right-censored values using the following rules: a right-censored value is labeled adding the symbol ‘+’ at the end of the value, whereas the symbol ‘-’ is used for the left-censored values (see examples below). The summary statistics can be obtained using the method function `summary.datacggm`. The matrices $Y$ and $X$ are extracted from a datacggm object using the function `getMatrix`.

For each column of the matrix ‘Y’, mean and variance are estimated using a standard EM-algorithm based on the assumption of a Gaussian distribution. ‘maxit’ and ‘thr’ are used to set the number of iterations and the threshold for convergence, respectively. Marginal means and variances can be extracted using the accessor functions ‘ColMeans’ and ‘ColVars’, respectively. Furthermore, the plotting functions `hist.datacggm` and `qqcnorm` can be used to inspect the marginal distribution of each column of the matrix ‘Y’.

The status indicator matrix, denoted by $R$, can be extracted by using the function `event`. The entries of this matrix specify the status of an observation using the following code:

- ‘$R[i, j] = 0$’ means that the $y_{ij}$ is inside the open interval $(lo[j], up[j])$;
- ‘$R[i, j] = -1$’ means that the $y_{ij}$ is a left-censored value;
- ‘$R[i, j] = +1$’ means that the $y_{ij}$ is a right-censored value;
- ‘$R[i, j] = +9$’ means that the $y_{ij}$ is a missing value.

See below for the other functions related to an object of class ‘datacggm’.

**Value**

‘datacggm’ returns an R object of S3 class “datacggm”, that is, a nested named list containing the following components:

- **Y** the $(n \times p)$-dimensional matrix $Y$.
- **X** the $(n \times q)$-dimensional data frame $X$.

**Info**

- lo: the lower censoring vector;
- up: the upper censoring vector;
- $R$: the status indicator matrix encoding the censored/missing values (mainly for internal purposes);
- order: an integer vector used for the ordering of the matrices $Y$ and $X$ (for internal purposes only);
- Pattern: a matrix encoding the information about the the patterns of censored/missing values (for internal purposes only);
- $ym$: the estimated marginal means of the random variables $Y_j$;
- $yv$: the estimated marginal variances of the random variables $Y_j$;
- n: the sample size;
- p: the number of response variables;
- q: the number of columns of the data frame $X$.

**Author(s)**

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


See Also

Related to the R objects of class “datacggm” there are the accessor functions, `rowNames`, `colNames`, `getMatrix`, `ColMeans`, `ColVars`, `upper`, `lower`, `event`, `qqcnorm` and the method functions `is.datacggm`, `dim.datacggm`, `summary.datacggm` and `hist.datacggm`. The function `rcggm` can be used to simulate a dataset from a conditional Gaussian graphical model whereas `cglasso` is the model fitting function devoted to the l1-penalized censored Gaussian graphical model.

Examples

```r
set.seed(123)

# a dataset from a right-censored Gaussian graphical model
def <- 100L
p <- 3L
Y <- matrix(rnorm(def * p), def, p)
up <- 1
Y[Y >= up] <- up
Z <- datacggm(Y = Y, up = up)
Z

# a dataset from a conditional censored Gaussian graphical model
def <- 100L
p <- 3L
q <- 2
Y <- matrix(rnorm(def * p), def, p)
up <- 1
lo <- -1
Y[Y >= up] <- up
Y[Y <= lo] <- lo
X <- matrix(rnorm(def * q), def, q)
Z <- datacggm(Y = Y, lo = lo, up = up, X = X)
Z

# a dataset from a conditional censored Gaussian graphical model
# and with missing-at-random values
def <- 100L
p <- 3L
q <- 2
Y <- matrix(rnorm(def * p), def, p)
NA.id <- matrix(rbinom(def * p, 1L, 0.01), def, p)
```
Y[NA.id == 1L] <- NA
up <- 1
lo <- -1
Y[Y >= up] <- up
Y[Y <= lo] <- lo
X <- matrix(rnorm(n * q), n, q)
Z <- datacggm(Y = Y, lo = lo, up = up, X = X)
Z

---

**dim.datacggm**

*Dimensions of a “datacggm” Object*

**Description**

Retrieve the dimension of an R object of class “datacggm”, that is, the sample size \( n \), the number of response variables \( p \) and the number of predictors \( q \).

**Usage**

```r
## S3 method for class 'datacggm'
dim(x)
```

**Arguments**

- `x` an R object of class ‘datacggm’.

**Details**

For an R object of class ‘datacggm’, `dim` retrieves a list with elements named `Y` and `X`. The first component is the “dim” attribute of the matrix `Y` whereas the second component is the “dim” attribute of the matrix `X`. If `X` is missing, the second component is `NULL`.

**Value**

For an object of class ‘datacggm’, `dim` retrieves a named list with components `Y` and `X` which are the “dim” attribute of the two matrices, respectively. If `X` is missing then the last component is `NULL`.

**Author(s)**

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

**See Also**

datacggm, rcgmm,nobs, nresp and npred.
Examples

```r
set.seed(123)
# a dataset from a censored Gaussian graphical model
n <- 100L
p <- 3L
b0 <- rep(0, p)
Z <- rcggm(n = n, b0 = b0, probl = 0.05, probr = 0.05)
dim(Z)

# a dataset from a conditional censored Gaussian graphical model
n <- 100L
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
Z <- rcggm(n = n, b0 = b0, X = X, B = B, probl = 0.05, probr = 0.05)
dim(Z)
```

---

dimnames.datacggm

**Dimnames of a “datacggm” Object**

***Description***

Retrieve or set the dimnames of a “datacggm” object.

***Usage***

```r
## S3 method for class 'datacggm'
dimnames(x)

## S3 replacement method for class 'datacggm'
dimnames(x) <- value
```

***Arguments***

- `x` an object of class 'datacggm'.
- `value` a nested list with names 'X' and 'Y'. See below for mode details.

***Details***

For an R object 'x' of class ‘datacggm’, dimnames retrieves or sets the dimnames attribute for the matrices Y and X (see 'datacggm' for more details). When setting the dimnames attribute, value$Y can be NULL (which is not stored) or a list of length two. In the last case, value$Y is passed to the method function dimnames for setting the dimnames attributes of the matrix retrieved by getMatrix(x, “Y”). In the same way, value$X can be used for setting the dimnames attributes of the matrix retrieved by getMatrix(x, “X”).
event

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

See Also
datacggm, rcggm, getMatrix, rowNames and colNames.

Examples

```r
set.seed(123)
# a dataset from a censored Gaussian graphical model
n <- 100L
p <- 3L
b0 <- rep(0, p)
Z <- rcggm(n = n, b0 = b0, probl = 0.05, probr = 0.05)
dimnames(Z)

dimnames(Z) <- list(Y = list(paste0("i", seq_len(n)), paste("Y", 1:p, sep = ":")))
dimnames(Z)

# the same as
# dimnames(Z)$Y <- list(paste0("i", seq_len(n)), paste("Y", 1:p, sep = ":"))
# dimnames(Z)

# a dataset from a conditional censored Gaussian graphical model
n <- 100L
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
Z <- rcggm(n = n, b0 = b0, X = X, B = B, probl = 0.05, probr = 0.05)
dimnames(Z)

dimnames(Z) <- list(Y = list(NULL, paste("Y", 1:p, sep = ":")),
                   X = list(NULL, paste("X", 1:q, sep = ":")))
dimnames(Z)

# the same as
# dimnames(Z)$Y <- list(NULL, paste("Y", 1:p, sep = ":"))
# dimnames(Z)$X <- list(NULL, paste("X", 1:q, sep = ":"))
# dimnames(Z)
```

---

**event**

**Status Indicator Matrix from a ‘datacggm’ Object**

Description

The ‘event’ function retrieves the status indicator matrix from an object of class ‘datacggm’.
Usage

```
event(x, ordered = FALSE)
```

Arguments

- `x`: an object of class 'datacggm'.
- `ordered`: logical value used to specify if the rows of the status indicator matrix should be ordered according to the patterns of censored/missing values. Default `ordered = FALSE`.

Details

The `event` function is used to retrieve the status indicator matrix, denoted by \( R \), from an object of class 'datacggm'. The entries of the matrix are used to specify the status of the response variable:

- ‘\( R[i, j] = 0 \)’ means that \( y_{ij} \) is inside the open interval \((\text{lo}[j], \text{up}[j])\);
- ‘\( R[i, j] = -1 \)’ means that \( y_{ij} \) is a left-censored value;
- ‘\( R[i, j] = +1 \)’ means that \( y_{ij} \) is a right-censored value;
- ‘\( R[i, j] = +9 \)’ means that \( y_{ij} \) is a missing value.

Value

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

Author(s)

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

References


See Also

datacggm and rcggm.
Examples

set.seed(123)

# Y ~ N(b0 + XB, Sigma) and
# 1. probability of left/right censored values equal to 0.05
# 2. probability of missing-at-random euqals to 0.05
n <- 100L
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05,
         probr = 0.05, probna = 0.05)

# status indicator matrix
event(Z)

# in this case the status indicator matrix is returned with
# rows ordered according to the patterns of missing data
event(Z, ordered = TRUE)

Example

Simulated data for the cglasso vignette

Description

Simulated data, used to demonstrate the features of datacggm class. It contains two simulated datasets that mimic the structural characteristics of the two real datasets MKMEP and MM.

1. MKMEP.Sim: is a reponse matrix containing 50 samples of 10 genes;
2. MM.Sim: is a list containing two components;
   • Y: is the response matrix containing 50 measurements of 10 miRNAs
   • X: is the predictor matrix containing 50 measurements of 5 variables (four numerical and one categorical)

In both datasets, the response matrices are right-censored with an upper limit of detection fixed to 40.

Usage

data("Example")

See Also
cglasso, to_graph, and the method functions summary, coef, plot, AIC.cglasso, BIC.cglasso, MKMEP and MM.
Examples

data("Example")

MM.Sim <- datacggm(Y = MM.Sim$Y, up = 40, X = MM.Sim$X)
ColMeans(MM.Sim)
ColVars(MM.Sim)
summary(MM.Sim)

MKMEP.Sim <- datacggm(Y = MKMEP.Sim, up = 40)
ColMeans(MKMEP.Sim)
ColVars(MKMEP.Sim)
summary(MKMEP.Sim)

---

fitted Extract Model Fitted Values

Description

The accessor function fitted extracts model fitted values from an R object inheriting class ‘cglasso’.

Usage

## S3 method for class 'cglasso'
fitted(object, lambda.id, rho.id, drop = TRUE, ...)

Arguments

object an R object inheriting class ‘cglasso’, that is, the output of the model-fitting functions cglasso or cggm.
lambda.id a vector of integers used to specify the \( \lambda \)-values.
rho.id a vector of integers used to specify the \( \rho \)-values.
drop logical. Dimensions can only be dropped if their extent is one.
... further arguments passed to or from other methods.

Value

Fitted values extracted from ‘object’ are returned.

Author(s)

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

See Also

Model-fitting functions cglasso, cggm and the accessor functions coef.cglasso, residuals.cglasso, predict.cglasso and impute.
Examples

```r
set.seed(123)
# Y ~ N(0, Sigma) and probability of left/right censored values equal to 0.05
n <- 1000L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
fitted(out, rho.id = 3L, drop = TRUE)

# Y ~ N(b0 + XB, Sigma) and probability of left/right censored values equal to 0.05
n <- 1000L
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
fitted(out, lambda.id = 3L, rho.id = 3L, drop = TRUE)
```

---

**getGraph**

**Retrieve Graphs from a 'cglasso2igraph' Object**

Description

‘getGraph’ retrieves graphs from an R object of class ‘cglasso2igraph’.

Usage

```r
getGraph(x, type = c("Gyy", "Gxy", "both"))
```

Arguments

- **x** an object of class ‘cglasso2igraph’ (see also to_graph).
- **type** a description of the required graph. Default is ‘Gyy’.

Value

getGraph retrieves an R object of class ‘igraph’ representing the graph required by the argument type.

Author(s)

Luigi Augugliaro (<luigi.augugliaro@unipa.it>)
See Also
to_graph, is.cglasso2igraph and plot.cglasso2igraph.

Examples

```r
set.seed(123)
# Y ~ N(0, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
out.graph <- to_graph(out)
getGraph(out.graph)

# Y ~ N(b0 + XB, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
q <- 2L
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
out.graph <- to_graph(out, lambda.id = 3, rho.id = 3, weighted = TRUE)
getGraph(out.graph, type = "Gyy")
getGraph(out.graph, type = "Gxy")
getGraph(out.graph, type = "both")
```

getMatrix

### Retrieve Matrices ‘Y’ and ‘X’ from a ‘datacggm’ Object

**Description**

‘getMatrix’ retrieves matrices ‘Y’ and/or ‘X’ from an object of class ‘datacggm’.

**Usage**

```r
getMatrix(x, name = c("Y", "X", "both"), ordered = FALSE)
```

**Arguments**

- **x**: an object of class ‘datacggm’.
- **name**: the name of the required matrix.
getMatrix

ordered  logical value used to specify if the required matrix should be retrieved with rows ordered according to the patterns of censored values. See below for some example.

Value

getMatrix retrieves the matrix specified by ‘name’ and with row ordering specified by ‘ordered’. A named list returned if name is “both”.

Author(s)

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

References


See Also

datacggm, rcggm.

Examples

```r
set.seed(123)

# a dataset from a conditional censored Gaussian graphical model
n <- 100L
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
Z <- rcggm(n = n, b0 = b0, X = X, B = B, probl = 0.05, probr = 0.05)
getMatrix(Z, name = "Y")

# in the following examples 'Y' and 'X' is returned with rows ordered
# according to the patterns of censored data
getMatrix(Z, name = "Y", ordered = TRUE)
getMatrix(Z, name = "X")
getMatrix(Z, name = "X", ordered = TRUE)
getMatrix(Z, name = "both")
getMatrix(Z, name = "both", ordered = TRUE)
```
Description

Creates histograms with a normal density as background and two segments corresponding to left
and/or right censored values (if any) in a conditional censored Gaussian graphical model.

Usage

```r
## S3 method for class 'datacggm'
hist(x, breaks = "Sturges", include.lowest = TRUE, right = TRUE, nclass = NULL,
which, max.hist = 1L, save.hist = FALSE, grdev = pdf, grdev.arg,
polygon.col = adjustcolor("grey", alpha.f = 0.25), polygon.border = NA,
segments.lwd = 4L, segments.lty = 2L, segments.col = "gray40",
points.pch = c(4L, 1L), points.cex = 1.8, points.col = rep("black", 2L),
legend = TRUE, ...)
```

Arguments

- `x` an object of class `datacggm`.
- `breaks` one of:
  - a vector giving the breakpoints between histogram cells,
  - a function to compute the vector of breakpoints,
  - a single number giving the number of cells for the histogram,
  - a character string naming an algorithm to compute the number of cells,
  - a function to compute the number of cells.
  See `hist` for more details.
- `include.lowest` logical; if `TRUE`, an `x[i]` equal to the breaks value will be included in the first
  (or last, for `right = FALSE`) bar. This will be ignored (with a warning) unless
  `breaks` is a vector.
- `right` logical; if `TRUE`, the histogram cells are right-closed (left open) intervals.
- `nclass` numeric (integer). For S(-PLUS) compatibility only, `nclass` is equivalent to
  `breaks` for a scalar or character argument.
- `which` a vector of integers used to specify the response variables for which the his-
  togram is required.
- `max.hist` the maximum number of histograms drawn in a single figure.
- `save.hist` a logical variable or a string specifying the path of the directory where plots
  will be saved. Letting `save.hist = TRUE`, the required plots will be saved as
  external files inside the current working directory. User can save these files
  on a specific working directory passing the absolute path through the argument
  `save.hist`. On exit, working directory will be always set the previous one.
- `grdev` the graphics device used to save the required histograms on external files. See
  `device` for more details.
additional parameters passed to the graphics device specified by ‘grdev’.

the colour for filling the area underlying the Gaussian distribution.

the colour of the border of the Gaussian distribution.

the line width used in plotting the segments associated to the symbols specified by ‘points.pch’.

the line type used in plotting the segments associated to the symbols specified by ‘points.pch’.

the colour used in plotting the segments associated to the symbols specified by ‘points.pch’.

densities, plotted on the y-axis, are computed in such a way that the sum of all densities plus the proportion of censored values is equal to one.

To evaluate the distributional assumption underlying the censored Gaussian distribution, on each histogram the area underlying the Gaussian density function is also shown in the background (marginal parameters are estimated as described in ‘datacggm’ and ‘ColMeans’). Furthermore, on each plot, the proportions of left/right censored values are graphically compared with the corresponding Gaussian tail probabilities. If the assumption about the censoring mechanism is satisfied, then the proportion of censored values and the tail probability are approximately equals to each other.

Gianluca Sottile (<gianluca.sottile@unipa.it>)

See Also
datacggm, rcggm, ColMeans, ColVars and qqcnorm.

Examples

set.seed(123)

# a dataset from a right-censored Gaussian graphical model
impute(n <- 1000L, p <- 10L, 
Y <- matrix(rnorm(n * p), n, p), 
up <- 1, 
Y[Y >= up] <- up, 
Z <- datacggm(Y = Y, up = up), 
hist(Z, max.hist = 4L)

impute

Imputation of Missing and Censored Values

Description

Imputes multivariate missing and censored values.

Usage

impute(object, type = c("mar", "censored", "both"), lambda.new, rho.new)

Arguments

object

an \texttt{R} object inheriting class ‘\texttt{cglasso}’, that is, the output of the model-fitting functions \texttt{cglasso} and \texttt{cggm}.

type

a description of the imputation required (see section ‘Description’). By default only the missing-at-random values are imputed.

lambda.new

value of the tuning parameter \(\lambda\) at which the imputations are required.

rho.new

value of the tuning parameter \(\rho\) at which the imputations are required.

Details

The \texttt{impute} function returns the response matrix with the imputed missing-at-random values (‘type = "mar"’), the imputed censored values (‘type = "censored"’), or both (‘type = "both"’).

If ‘type = "mar"’ then, for each row, the missing response values are replaced with the expected values of a multivariate normal distribution conditioned on the observed response values.

If ‘type = "censored"’ then, for each row, the censored response values are imputed using the expected values of a multivariate truncated normal distribution conditioned on the observed response values.

If ‘type = "both"’ then missing-at-random and censored values are imputed. In this case the returned matrix corresponds to the working response matrix computed during the E-Step.

Value

The \texttt{impute} function returns a response matrix with the required imputed data.

Author(s)

Luigi Augugliaro (luigi.augugliaro@unipa.it)
See Also

Model-fitting functions cglasso, cggm and the accessor functions coef.cglasso, fitted.cglasso, residuals.cglasso and predict.cglasso.

Examples

```r
set.seed(123)
# Y ~ N(0, Sigma) and
# 1. probability of left/right censored values equal to 0.05
# 2. probability of missing-at-random values equal to 0.05
n <- 100L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05, probna = 0.05)

out <- cglasso(. ~ ., data = Z)
rho.new <- mean(out$rho)

# imputing missing values
Y.impute <- impute(out, type = "mar", rho.new = rho.new)

# imputing censored values
Y.impute <- impute(out, type = "censored", rho.new = rho.new)

# imputing missing and censored values
Y.impute <- impute(out, type = "both", rho.new = rho.new)

# Y ~ N(XB, Sigma) and
# 1. probability of left/right censored values equal to 0.05
# 2. probability of missing-at-random values equal to 0.05
n <- 100L
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05, probna = 0.05)

out <- cglasso(. ~ ., data = Z)
lambda.new <- mean(out$lambda)
rho.new <- mean(out$rho)

# imputing missing values
Y.impute <- impute(out, type = "mar", lambda.new = lambda.new, rho.new = rho.new)

# imputing censored values
Y.impute <- impute(out, type = "censored", lambda.new = lambda.new, rho.new = rho.new)
```

# imputing missing and censored values
Y.impute <- impute(out, type = "both", lambda.new = lambda.new, rho.new = rho.new)

is.cglasso2igraph

Is an Object of Class ‘cglasso2igraph’?

Description

is.cglasso2igraph tests if its argument is an object of class ‘cglasso2igraph’.

Usage

is.cglasso2igraph(x)

Arguments

x
  object to be tested.

Author(s)

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

See Also

to_graph.

Examples

set.seed(123)
n <- 100L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
out.graph <- to_graph(out)
is.cglasso2igraph(out.graph)
is.datacggm

Is an Object of Class ‘datacggm’?

Description

‘is.datacggm’ tests if its argument is an object of class ‘datacggm’.

Usage

is.datacggm(x)

Arguments

x object to be tested.

Author(s)

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

References


See Also

datacggm and rcggm

Examples

set.seed(123)
n <- 100L
p <- 3L
b0 <- rep(0, p)
Z <- rcggm(n = n, b0 = b0, probl = 0.05, probr = 0.05)
is.datacggm(Z) # TRUE
Description

Functions ‘lower’ and ‘upper’ retrieve the lower and upper censoring values from an object of class ‘datacggm’.

Usage

\[
\text{lower}(x) \\
\text{upper}(x)
\]

Arguments

\(x\)  \\
an object of class ‘datacggm’.

Details

For an \(R\) object \(x\) of class ‘datacggm’, \text{lower} (\text{upper}) retrieves the lower (upper) censoring values of the response variables.

Author(s)

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

See Also

datacggm

Examples

\begin{verbatim}
set.seed(123)
n <- 100L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))

Z <- rcggm(n = n, Sigma = Sigma, probr = 0.05)
lower(Z)
upper(Z)

Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05)
lower(Z)
upper(Z)

Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)
lower(Z)
upper(Z)
\end{verbatim}
Description

In a study about the formation of blood cells, Psaila *and others* (2016) have recently identified three distinct sub-populations 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.

Multiplex RT-qPCR has been 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.

‘MKMEP’ is a matrix containing a subset of the data available from Psaila *and others* (2016).

Usage

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

References


See Also

cglaso, to_graph, and the method functions summary, coef, plot, AIC.cglasso and BIC.cglasso.

Examples

```r
data("MKMEP")
is.matrix(MKMEP)
Z <- datacggm(Y = MKMEP, up = 40)
str(Z)
```
The Rule of miRNA in Multiple Myeloma

Description

MicroRNAs (miRNAs) are endogenous small non-coding RNAs, approximately 22 nucleotides in length, that play a regulatory role in gene expression by mediating mRNA cleavage or translation expression. Several studies have shown that a deregulation of the miRNAs can cause a disruption in the gene regulation mechanisms of the cell and that this might even lead to cancerous phenotypes.

Gutierrez and others (2010) investigated the expression level of a set of 141 miRNAs, measured by RT-qPCR technology on a sample of 64 patients with multiple myeloma (MM). Patients were selected to represent the most relevant recurrent genetic abnormalities in MM. RT-qPCR data are typically right-censored and, in this study, the upper limit of detection was fixed to 40 cycles.

Formally, \( \mathbb{M} \) is a list with containing a subset of the data studied in Gutierrez and others (2010). Its components are:

- \( Y \): the response matrix containing the cycle-threshold of the measured 49 miRNAs;
- \( X \): the predictor matrix containing the expression levels of 15 endogenous internal reference genes, called housekeeping, and a factor encoding the cytogenetic abnormalities.

Usage

\[
\text{data("}\mathbb{M}\text{")}
\]

References


See Also

cglasso, to_graph, and the method functions summary, coef, plot, AIC.cglasso and BIC.cglasso.
Examples

```r
data("MM")
Z <- datacggm(Y = MM$Y, X = MM$X, up = 40)
print(Z, width = 80L)
```

### `nobs + nresp + npred`

Extract the number of observations, response variables and predictors from an object of class `datacggm`.

#### Description

Extract the number of observations, response variables and predictors from an object of class `datacggm`.

#### Usage

```r
## S3 method for class 'datacggm'
nobs(object, ...)
## S3 method for class 'datacggm'
nresp(object, ...)
## S3 method for class 'datacggm'
npred(object, ...)
```

#### Arguments

- `object`: an R object of class `datacggm`.
- `...`: further arguments to be passed to methods.

#### Author(s)

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

#### See Also

- `datacggm`, `rcggm` and `dim.datacggm`.

#### Examples

```r
set.seed(123)
n <- 100
p <- 3
q <- 2
b0 <- rep(1, p)
X <- matrix(rnorm(n * q), n, q)
B <- matrix(rnorm(q * p), q, p)
Sigma <- outer(1:p, 1:p, function(i, j) 0.3*abs(i - j))
probl <- 0.05
probr <- 0.05
```
plot.cggm <- 0.05
Z <- rcggm(n = n, b0 = b0, Sigma = Sigma, probl = probl, probr = probr,
        probna = probna)
nobs(Z)
nresp(Z)
npred(Z)

Z <- rcggm(b0 = b0, X = X, B = B, Sigma = Sigma, probl = probl, probr = probr,
        probna = probna)
nobs(Z)
nresp(Z)
npred(Z)

---

plot.cggm

Plot Method for a 'cggm' Object

Description

The plot.cggm produces graphs from an R object of class 'cggm'.

Usage

## S3 method for class 'cggm'
plot(x, type, weighted = FALSE, simplify = TRUE, ...)

Arguments

x
  an R object of class 'cggm', that is, the output of the function cggm.

type
  a description of the required graph. Default depends on the type of fitted model,
  that is, type = both if a conditional censored graphical lasso estimator is fitted,
  otherwise type is equal to Gyy (see getGraph) for more details.

weighted
  logical. Should weighted graphs be created? Default is FALSE.

simplify
  logical. Should isolated vertices be removed from the graph? Default is TRUE,
  i.e., isolated vertices are removed.

... additional graphical arguments passed to the functions plot.igraph.

Author(s)

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

See Also

cggm, to_graph, getGraph and plot.cglasso2igraph.
Examples

```r
set.seed(123)
# Y ~ N(XB, Sigma) and
# 1. probability of left/right censored values equal to 0.05
# 2. probability of missing-at-random values equal to 0.05
n <- 100L
p <- 3L
q <- 2L
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.5,
probr = 0.05)
out <- cglasso(. ~ ., data = Z)
out.mle <- cggm(out, lambda.id = 3L, rho.id = 3L)
plot(out.mle, type = "Gyy")
plot(out.mle, type = "Gxy")
plot(out.mle, type = "both")
```

---

**plot.cglasso**

*Plot Method for ‘cglasso’ Object*

**Description**

‘The `plot.cglasso` function produces plots to study the coefficient paths of fitted cglasso models.

**Usage**

```r
## S3 method for class 'cglasso'
plot(x, what = c("Theta", "diag(Theta)", "b0", "B"),
penalty = ifelse(x$nrho >= x$nlambda, "rho", "lambda"), given = NULL,
GoF = AIC, add.labels, matplot.arg1, matplot.arg2, labels.arg, abline.arg,
mtext.arg, save.plot, grdev = pdf, grdev.arg, digits = 4L, ...)
```

**Arguments**

- `x` an R object of class ‘cglasso’, that is, the output of the fitting function cglasso.
- `what` a character or a formula specifying the required plot. If what is a character, then it is used to specify the conditional coefficient path. Allowed descriptors are: ‘Theta’, the off-diagonal entries of the precision matrix, ‘diag(Theta)’, the diagonal entries of the precision matrix, ‘b0’, the intercepts of the conditional models, or equivalently the expected values of the response variables, and ‘B’, the regression coefficients. Optionally, ‘what’ can be a model formula (see sections Description and Examples for more details).
penalty

optional character argument used to specify the tuning parameter needed to plot the conditional coefficient path. Allowed descriptors are ‘rho’ and ‘lambda’. This argument can be omitted if what is a model formula.

given

an optional vector of integers identifying the conditioning values of the second tuning parameter to be used for the coefficient profile plots of the parameters identified by the input ‘what’ across their corresponding tuning parameter. This argument is required only if ‘what’ is a character, otherwise it can be omitted.

GoF

a valid goodness-of-fit function, such as AIC.cglasso and BIC.cglasso, or an R object of class ‘GoF’.

add.labels

logical value. Should labels be added to the plot?

matplot.arg1

a named list with the graphical parameters used to plot the paths of the estimates identified by ‘GoF’.

matplot.arg2

a named list with the graphical parameters used to plot the paths of the remaining estimates.

labels.arg

a named list with the graphical parameters used to plot the labels.

abline.arg

a named list with the graphical parameters used to plot the line identifying the optimal tuning parameter value.

mtext.arg

a named list with the graphical parameters used to plot the text reported on the third axis.

save.plot

a logical variable or a string specifying the path of the directory where plots will be saved. Letting ‘save.plot = TRUE’, the required plots will be saved as external files inside the current working directory. User can save these files on a specific working directory passing the absolute path through the argument save.plot. On exit, working directory will be always set the previous one.

grdev

the graphics device used to save the required histograms on external files. See ‘device’ for more details.

grdev.arg

additional parameters passed to the graphics device specified by ‘grdev’.

digits

number of digits used to print the value of the second tuning parameter identified by ‘given’.

... further arguments passed to the chosen goodness-of-fit function, such as ‘AIC’ or ‘BIC’.

Details

The function plot.cglasso produces the coefficient profile plot. The output depends both on the type of fitted model and on the setting of the three main arguments, what, penalty and given. Below we give more details.

If the model fitting function cglasso is used to fit an L1-penalized censored Gaussian graphical model (see first part in Section Example), then the user can specify only the main argument ‘what’, whereas ‘penalty’ and ‘given’ can be omitted. If ‘penalty’ is specified, then it must be equal to “rho”. The main argument ‘what’ is used to specify the estimator that is plotted on the y-axis. In this case, it must be equal to one of the following descriptors:

- “Theta”: the path of the estimated partial correlation coefficients is returned;
- “diag(Theta)”: the path of the diagonal entries of the estimated precision matrix is returned;
• “b0”: the path of the estimated expected values of the response variables is returned;

If an l1-penalized conditional censored Gaussian graphical model is fitted (see second part in Section Example), then all the main arguments can be used. In this case:

1. ‘what’ must be equal to one of the following descriptors:
   • “Theta”: the path of the off-diagonal entries of the estimated precision matrix is returned;
   • “diag(Theta)”: the path of the estimated partial correlation coefficients is returned;
   • “b0”: the path of the estimated intercepts is returned;
   • “B”: the path of the estimated regression coefficients is returned for each response variable;

2. ‘penalty’ is used to specify the tuning parameter that is reported on the x-axis and it must be equal to one of the following descriptors:
   • “rho”: the tuning parameter used to penalize the precision matrix;
   • “lambda”: the tuning parameter used to penalize the regression coefficient matrix;

3. ‘given’ is an optional vector of integers used to specify the values of the second tuning parameter. For instance, letting what = “Theta”, penalty = “rho” and given = 1 then plot.cglasso returns the plot of the estimated partial correlation coefficients versus the ρ-values and given the first value of the used λ-values. If given is left unspecified then a sequence of conditional profile plot is returned, a plot for each value of the second tuning parameter.

Optionally, the user can specify a conditional profile plot using a model formula with the following template:

‘what ~ penalty | given’

For instance, the previous plot can be specified using the following model formula

‘Theta ~ rho | 1’

In this case, the arguments penalty and given must be left unspecified whereas what is used for the model formula (see also examples below).

The optional argument GoF can be used to identify the non-zero estimates by the output of a goodness-of-fit function, such as AIC.cglasso and BIC.cglasso. In this case, a vertical red dashed line is used the identify the optimal value of the tuning parameter reported on the x-axis, whereas the path of the identified non-zero estimates is drawn using a solid black line; the remaining paths are drawn using gray dashed lines. This option is disabled if we let ‘GoF = NULL’, when all the paths are drawn using solid black lines.

Author(s)

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

References


See Also

cglasso, AIC.cglasso and BIC.cglasso.

Examples

```r
set.seed(123)

# Model 1: censored glasso estimator (Augugliaro *and other*, 2020a)
# Y ~ N(0, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 10L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
# coefficient profile plot of the off-diagonal entries of the precision matrix
plot(out, what = "Theta")
# the same as
#plot(out, what = "Theta", penalty = "rho")

# the output of a goodness-of-fit function can be used to identify the
# non-zero estimates
plot(out, what = "Theta", GoF = BIC)

# letting 'GoF = NULL' the previous option is disabled
plot(out, what = "Theta", GoF = NULL)

# coefficient profile plot of the diagonal entries of the precision matrix
plot(out, what = "diag(Theta)")

# coefficient profile plot of the expected values of the response variables
plot(out, what = "b0")
```

# Model 2: conditional censored glasso estimator (Augugliaro *and other*, 2020b)
# Y ~ N(b0 + XB, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 10L
q <- 5L
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rccgm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)

# conditional profile plot of the estimator partial correlation coefficients versus
# the used values of the tuning parameter rho and given the first lambda-value
plot(out, what = "Theta", penalty = "rho", given = 1L)
out$lambda[1L]

lambda.id <- c(2, 4)
plot(out, what = "Theta", penalty = "rho", given = lambda.id)
out$lambda[lambda.id]

# in this case a sequence of ten conditional profile plots is returned, that is a
# plot for each lambda-value.
plot(out, what = "Theta", penalty = "rho")

# optionally, the user can specify the conditional profile plots using the model
# formula
plot(out, what = Theta ~ rho | lambda.id)
lambda.id

plot(out, what = Theta ~ rho)

# the output of a goodness-of-fit function can be used to identify the
# non-zero estimates
plot(out, what = Theta ~ rho | 10, GoF = BIC)

# letting 'GoF = NULL' the previous option is disabled
plot(out, what = Theta ~ rho | 10, GoF = NULL)

---

**plot.cglasso2igraph**  
*Plot Method for a cglasso2igraph Object*

#### Description

`plot.cglasso2igraph` produces graphs from an R object of class `cglasso2igraph`.

#### Usage

```r
## S3 method for class 'cglasso2igraph'
plot(x, type, ...)
```
Arguments

x

an R object of class `cglasso2igraph`, that is, the output of the function `to_graph`.

type

da description of the required graph. Default is ‘both’ in a conditional glasso estimator or ‘Gyy’ in a glasso estimator (see `getGraph`).

... additional graphical arguments passed to the functions `plot.igraph`.

Author(s)

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

See Also

`to_graph` and `getGraph`.

Examples

```r
set.seed(123)
# Y ~ N(0, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
out.graph <- to_graph(out)
plot(out.graph, type = "Gyy")

out.graph <- to_graph(out, weighted = TRUE)
plot(out.graph, type = "Gyy")
plot(out.graph, type = "both")

# Y ~ N(b0 + XB, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
q <- 2L
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
out.graph <- to_graph(out, lambda.id = 3, rho.id = 3, weighted = TRUE)
plot(out.graph, type = "Gyy")
plot(out.graph, type = "Gxy")
plot(out.graph, type = "both")
```
**Description**

'The `plot.GoF`' function produces plots to study the sequence of fitted models.

**Usage**

```r
## S3 method for class 'GoF'
plot(x, add.line = TRUE, arg.line = list(lty = 2L, lwd = 2L, col = "red"),
     add.text = FALSE, arg.text = list(side = 3L), arg.points = list(pch = 2L),
     ...)
```

**Arguments**

- `x`: an R object of class 'GoF', that is, the output of a goodness-of-fit function such as `AIC.cglasso` or `BIC.cglasso`.
- `add.line`: logical; if `add.line = TRUE` then a line is added to identify the optimal value of the tuning parameter.
- `arg.line`: a named list of graphical parameters passed to the function `abline` (see also `par`).
- `add.text`: logical; if `add.text = TRUE` then a text is added to the line used to identify the optimal value of the tuning parameter.
- `arg.text`: a list of further parameters passed to the function `mtext` (only if `add.text = TRUE`).
- `arg.points`: a named list of graphical parameters passed to the function `points`.
- `...`: additional graphical arguments passed to the functions `plot`, `contour` or `filled.contour`.

**Details**

`plot.GoF` is the plotting method function of an R object of class 'GoF', that is, the output of a goodness-of-fit function (see `AIC.cglasso`, or `BIC.cglasso`). This function produces a plot aimed both to evaluate the sequence of fitted models in terms of goodness-of-fit and to identify the optimal values of the tuning parameters.

If a tuning parameter is held fixed, then `plot.GoF` produces a plot showing the chosen measure of goodness-of-fit as a function of the remaining tuning parameter. In this case, the optimal value is identified by a vertical dashed line. The degrees-of-freedom of the selected fitted model are also shown.

If the cglasso model is fitted using both a sequence of $\rho$ and $\lambda$ values, then `plot.GoF` produces a contour plot and a triangle is used to identify the optimal pair of the two tuning parameters.

**Author(s)**

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

See Also
cglasso, AIC.cglasso, BIC.cglasso, summary.cglasso and select_cglasso.

Examples

```r
set.seed(123)
n <- 1000L
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z, nlambda = 1L)
plot(AIC(out))
plot(BIC(out))
out <- cglasso(. ~ ., data = Z, nrho = 1L)
plot(AIC(out))
plot(BIC(out))
out <- cglasso(. ~ ., data = Z)
plot(AIC(out))
plot(BIC(out))
```

**predict**

*Predict Method for cglasso and cggm Fits*

**Description**

Obtains predictions from an R object inheriting class ‘cglasso’.

**Usage**

```r
## S3 method for class 'cglasso'
predict(object, type = c("B", "mu", "Sigma", "Theta"), X.new, lambda.new, rho.new, 
          ...)

## S3 method for class 'cggm'
predict(object, X.new, ...)
```

**Arguments**

- `object` an R object inheriting class ‘cglasso’, that is, the output of the model-fitting function ‘cglasso’ or ‘cggm’.
predict

- **type**: a description of prediction required.
- **X.new**: matrix of new values for X at which predictions are to be made. This argument is used only if `type = "mu"`.
- **lambda.new**: value of the tuning parameter λ at which predictions are required.
- **rho.new**: value of the tuning parameter ρ at which predictions are required.
- **...**: further arguments passed to or from other methods.

**Details**

If object has S3 class `cglasso`, then for a new pair of the tuning parameters λ and ρ, the predict function can be used to predict the estimate of the regression coefficient matrix ("type = "B""), the estimate of the covariance matrix ("type = "Sigma""), or the estimate of the precision matrix ("type = "Theta""). If X.new is missing and `type = "mu"`, then the predict function returns the predicted values using the matrix of predictors X, otherwise the predicted fitted values are computed using the matrix X.new.

For a new pair of the tuning parameters λ and ρ, the predicted values are computed using a bilinear interpolation.

If the object has S3 class `cggm`, then the predict function returns only the predicted fitted values using the argument X.new.

**Value**

The matrix of predicted values.

**Author(s)**

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

**See Also**

Model-fitting function `cglasso` and the other accessor functions `coef.cglasso`, `fitted.cglasso`, `residuals.cglasso` and `impute`.

**Examples**

```r
set.seed(123)
# Y ~ N(0, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
rho.new <- mean(out$rho)
Theta.pred <- predict(out, rho.new = rho.new, type = "Theta")
Theta.pred

# Y ~ N(b0 + XB, Sigma) and probability of left/right censored values equal to 0.05
```

n <- 100L
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)

out <- cglasso(. ~ ., data = Z)
rho.new <- mean(out$rho)
lambda.new <- mean(out$lambda)
Theta.pred <- predict(out, lambda.new = lambda.new, rho.new = rho.new, type = "Theta")
Theta.pred

QFun

Extract Q-Function

Description

‘QFun’ extracts the values of the Q-function from an R object inheriting class ‘cglasso’.

Usage

QFun(object, mle, verbose = FALSE, ...)

Arguments

object an R object inheriting class ‘cglasso’, that is, the output of the model-fitting functions cglasso and cggm.
mle logical. Should Q-values be computed using the maximum likelihood estimates? Default depends on the class of the argument object: mle = FALSE for objects with class cglasso and mle = TRUE for objects with class cggm.
verbose logical for printing out a progress bar on the R console. Default is verbose = FALSE.
... further arguments passed to cggm.

Details

‘QFun’ returns the value of the Q-function, i.e., the value of the function maximised in the M-step of the EM 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 the ‘working’ empirical covariance matrix computed during the E-step.

QFun is used as a workhorse function to compute the measures of goodness-of-fit returned by the functions AIC.cglasso and BIC.cglasso.

The function ‘print.QFun’ is used to improve the readability of the results.
Value

‘QFun’ returns an R object of S3 class “QFun”, i.e., a named list containing the following components:

- `value`: a matrix with the values of the Q-function.
- `df`: a matrix with the number of estimated non-zero parameters.
- `dfB`: a matrix with the number of estimated non-zero regression coefficients.
- `dfTht`: a matrix with the number of estimated non-zero partial correlation coefficients.
- `n`: the sample size.
- `p`: the number of response variables.
- `q`: the number of columns of the design matrix X used to fit the model.
- `lambda`: the λ-values used to fit the model.
- `nlambda`: the number of λ-values.
- `rho`: the ρ-values used to fit the model.
- `nrho`: the number of ρ-values.
- `model`: a description of the fitted model passed through the argument object.

Author(s)

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

See Also

AIC.cglasso, BIC.cglasso, cglasso, cggm, summary.cglasso, select_cglasso and to_graph.

Examples

```r
set.seed(123)
# Y ~ N(b0+ XB, Sigma) and
# 1. probability of left/right censored values equal to 0.05
# 2. probability of missing-at-random values equal to 0.05
n <- 100L
p <- 3L
q <- 2L
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1:p, 1:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.5,
           probna = 0.05)
out <- cglasso(. ~ ., data = Z)
QFun(out)

out.mle <- cggm(out, lambda.id = 3L, rho.id = 3L)
QFun(out.mle)
```
**Description**

Creates a quantile-quantile plot for a censored Gaussian distribution.

**Usage**

```r
qucnorm(x, which, max.plot = 1L, save.plot = FALSE, grdev = pdf, grdev.arg,
main = "Censored Normal Q-Q Plot", xlab = "Theoretical Quantiles",
ylab = "Sample Quantiles", plot.it = TRUE, plot.pch = c(2L, 20L),
plot.col = c(2L, 1L), plot.cex = c(2L, 1L), abline = FALSE,
line.col = "gray50", line.lwd = 1L, line.lty = 2L, ...)
```

**Arguments**

- `x`: an object of class `datacggm`.
- `which`: a vector of integers used to specify the response variables for which the histogram is required.
- `max.plot`: the maximum number of plots drawn in a single figure.
- `save.plot`: a logical variable or a string specifying the path of the directory where plots will be saved. Letting `save.plot = TRUE`, the required plots will be saved as external files inside the current working directory. User can save these files on a specific working directory passing the absolute path through the argument `save.plot`. On exit, working directory will be always set the previous one.
- `grdev`: the graphics device used to save the required histograms on external files. See `device` for more details.
- `grdev.arg`: additional parameters passed to the graphics device specified by `grdev`.
- `main`, `xlab`, `ylab`: plot labels.
- `plot.it`: logical. Should the result be plotted?
- `plot.pch`: a pair of graphical parameters. The first entry specifies the symbol used for plotting the points associated to the censoring values `lo` and `up` (by default a triangle), whereas the second entry specifies the symbol used for the remaining points (by default a black point).
- `plot.col`: a pair of graphical parameters. The first entry specifies the colour used for plotting the points associated to the censoring values `lo` and `up` (by default a red triangle), whereas the second entry specifies the colour used for the remaining points (by default a black point).
- `plot.cex`: a pair of graphical parameters. The first entry specifies the size of the symbol used for plotting the points associated to the censoring values `lo` and `up`, whereas the second entry specifies the size of the symbol used for the remaining points.
- `abline`: logical. Should the line $y = x$ be plotted?
line.col  graphical parameter. If ‘abline = TRUE’, then this argument specifies the colour of the line $y = x$.
line.lwd  graphical parameter. If ‘abline = TRUE’, then this argument specifies the line width of the line $y = x$.
line.lty  graphical parameter. If ‘abline = TRUE’, then this argument specifies the line type of the line $y = x$.

...  additional graphical parameter passed to ‘plot’.

Details

‘qqcnorm’ produces a censored normal QQ plot, that is, a graphical method for comparing the empirical distribution of a given response variable (specified by the argument ‘which’) to the censored Gaussian distribution, which is defined as:

\[
\begin{align*}
\Phi((lo - \mu)/\sigma) & \quad \text{if } y \leq lo \\
\phi((y - \mu)/\sigma)/\sigma & \quad \text{if } lo < y < up \\
1 - \Phi((up - \mu)/\sigma) & \quad \text{if } y \geq up
\end{align*}
\]

where $\phi$ and $\Phi$ are the probability density function and the cumulative distribution of the standard normal distribution, respectively, whereas $lo$ and $up$ are the lower and upper censoring values, respectively.

The comparison is done by plotting the empirical quantiles ($y$-coordinate) against the theoretical quantiles ($x$-coordinate) of the censored Gaussian distribution, which are defined as follows:

\[
\begin{align*}
lo & \quad \text{if } p \leq \Phi((lo - \mu)/\sigma) \\
\mu + \sigma \Phi^{-1}(p) & \quad \text{if } \Phi((lo - \mu)/\sigma) < p < 1 - \Phi((up - \mu)/\sigma) \\
up & \quad \text{if } p \geq 1 - \Phi((up - \mu)/\sigma)
\end{align*}
\]

where $p \in (0, 1)$. If the two distributions are similar, the points will approximately lie on the line $y = x$. If the distributions are linearly related, the points will approximately lie on a line, but not necessarily on the line $y = x$. In order to evaluate if the proportions of left/right-censored values are similar to the Gaussian tail probabilities, points corresponding to the censored values are plotted using a specific symbol (see argument ‘plot.pch’), colour (see argument ‘plot.col’) and size (see argument ‘plot.cex’).

Finally, maximum likelihood estimates of the marginal parameters $\mu$ and $\sigma$ are computed as described in ‘datacggm’ and can be extracted from an R of class ‘datacggm’ by using the functions ‘ColMeans’ and ‘ColVars’, respectively.

Value

A named list is silently returned. Each element of the list contains a two-columns matrix; first columns (named ‘x’) contains the theoretical quantiles whereas second columns (named ‘y’) contains the empirical quantiles.

Author(s)

Gianluca Sottile (<gianluca.sottile@unipa.it>)
References


See Also

datacggm, rcggm, ColMeans, ColVars and hist.datacggm.

Examples

```r
set.seed(123)

# a dataset from a right-censored Gaussian graphical model
n <- 1000L
p <- 10L
Y <- matrix(rnorm(n * p), n, p)
up <- 1
Y[Y >= up] <- up
Z <- datacggm(Y = Y, up = up)
qqcnorm(Z, max.plot = 4L)

# a dataset from a conditional censored Gaussian graphical model
n <- 1000L
p <- 10L
q <- 2
Y <- matrix(rnorm(n * p), n, p)
up <- 1
lo <- -1
Y[Y >= up] <- up
Y[Y <= lo] <- lo
X <- matrix(rnorm(n * q), n, q)
Z <- datacggm(Y = Y, lo = lo, up = up, X = X)
qqcnorm(Z, max.plot = 4L)
```

---

**rcggm**

**Simulate Data from a Conditional Gaussian Graphical Model with Censored and/or Missing Values**

**Description**

‘rcggm’ function is used to produce one or more samples from a conditional Gaussian graphical model with censored and/or missing values.
Usage

`rcggm(n, p, b0, X, B, Sigma, probl, probr, probna, ...)`

Arguments

- **n**: the number of samples required (optional, see below for a description).
- **p**: the number of response variables (optional, see below for a description).
- **b0**: a vector of length `p` used to specify the intercepts. Default is a zero vector of length `p` (optional, see below for a description).
- **X**: a matrix of dimension `n` × `q` used to model the expected values of the response variables (optional, see below for a description).
- **B**: a matrix of dimension `q` × `p` used to specify the regression coefficients. If `X` is missing then `B` is set equal to NULL (optional, see below for a description).
- **Sigma**: a positive-definite symmetric matrix specifying the covariance matrix of the response variables. Default is the identity matrix (optional, see below for a description).
- **probl**: a vector giving the probabilities that the response variables are left-censored.
- **probr**: a vector giving the probabilities that the response variables are right-censored.
- **probna**: the probability that a response value is missing-at-random. By default ‘probna’ is set equal to zero.
- **...**: further arguments passed to the function ‘mvnorm’.

Details

The `rcggm` function simulates a dataset from a conditional Gaussian graphical model with censored or missing values and returns an object of class ‘datacggm’. Censoring values are implicitly specified using arguments `probl` and `probr`, that is, `lo` and `up` are computed in such a way that the average probabilities of left and right censoring are equal to `probl` and `probr`, respectively. Missing-at-random values are simulated using a Bernoulli distribution with probability `probna`.

The dataset is simulated through the following steps:

1. lower and upper censoring values (`lo` and `up`) are computed according to the arguments `probl` and `probr`;
2. The function `mvnorm` is used to simulate one or more samples from the multivariate Gaussian distribution specified by the arguments `b0`, `X`, `B` and `Sigma`;
3. The response values that are outside of the interval `[lo, up]` are replaced with the corresponding censoring values;
4. if `probna` is greater than zero, then missing-at-random values are simulated using a Bernoulli distribution with probability `probna`.

<table>
<thead>
<tr>
<th>Model</th>
<th>n</th>
<th>p</th>
<th>b0</th>
<th>X</th>
<th>B</th>
<th>Sigma</th>
<th>Gaussian distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>$Y \sim N(0, I)$</td>
</tr>
<tr>
<td>2</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>$Y \sim N(0, \Sigma)$</td>
</tr>
<tr>
<td>3</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$Y \sim N(b0, I)$</td>
</tr>
</tbody>
</table>
The previous table sums up the default setting of the multivariate Gaussian distribution used in step 2 (specified arguments are marked by the symbol ‘x’). See also below for some examples.

Value
rccgm returns an object of class ‘datacggm’. See datacggm for further details.

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

References


See Also
datacggm.

Examples
set.seed(123)

n <- 100
p <- 3
q <- 2
b0 <- rep(1, p)
X <- matrix(rnorm(n * q), n, q)
B <- matrix(rnorm(q * p), q, p)
Sigma <- outer(1:p, 1:p, function(i, j) 0.3^abs(i - j))
probl <- 0.05
probr <- 0.05
probna <- 0.05

# Model 1: Y ~ N(0, I)
Z <- rccgm(n = n, p = p, probl = probl, probr = probr, probna = probna)
summary(Z)
# Model 2: Y ~ N(0, Sigma)
Z <- rcggm(n = n, Sigma = Sigma, probl = probl, probr = probr, probna = probna)
summary(Z)

# Model 3: Y ~ N(b0, I)
Z <- rcggm(n = n, b0 = b0, probl = probl, probr = probr, probna = probna)
summary(Z)

# Model 4: Y ~ N(b0, Sigma)
Z <- rcggm(n = n, b0 = b0, Sigma = Sigma, probl = probl, probr = probr, probna = probna)
summary(Z)

# Model 5: Y ~ N(XB, I)
Z <- rcggm(X = X, B = B, probl = probl, probr = probr, probna = probna)
summary(Z)

# Model 6: Y ~ N(XB, Sigma)
Z <- rcggm(X = X, B = B, Sigma = Sigma, probl = probl, probr = probr, probna = probna)
summary(Z)

# Model 7: Y ~ N(b0 + XB, I)
Z <- rcggm(b0 = b0, X = X, B = B, probl = probl, probr = probr, probna = probna)
summary(Z)

# Model 8: Y ~ N(b0 + XB, Sigma)
Z <- rcggm(b0 = b0, X = X, B = B, Sigma = Sigma, probl = probl, probr = probr, probna = probna)
summary(Z)

---

**residuals**

*Extract Model Residuals*

**Description**

Extracts model residuals from an R object inheriting class ‘cglasso’.

**Usage**

```r
## S3 method for class 'cglasso'
residuals(object, type = c("observed", "working"), lambda.id, rho.id, drop = TRUE, ...)
```

**Arguments**

- `object`: an R object inheriting class ‘cglasso’, that is, the output of the model-fitting functions `cglasso` and `cggm`.
- `type`: a description of the desired residuals (see section ‘Details’).
lambda.id a vector of integers used to specify the $\lambda$-values.

rho.id a vector of integers used to specify the $\rho$-values.

drop logical. Dimensions can only be dropped if their extent is one.

... further arguments passed to or from other methods.

Details

The accessor function `residuals` returns an array storing the `observed` or `working` residuals, depending on the argument type.

The `observed` residuals are defined as the difference between the observed response values and the corresponding fitted expected values. For missing and censored response values, the `observed` residuals are set equal to `NA`.

The `working` residuals are obtained as a byproduct of the EM-algorithm used to fit the model. In the E-step, the algorithm computes the `working` response matrix, that is, the response matrix with missing values replaced by the conditional expected values of the multivariate normal distribution and censored values replaced by the conditional expected values of the multivariate truncated normal distribution. The `working` residuals are defined as the difference between `working` responses and fitted expected values.

Value

The accessor function `residuals` returns an array storing the desired residuals.

Author(s)

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

See Also

Model-fitting functions `cglasso`, `cggm` and the accessor functions `coef.cglasso`, `fitted.cglasso`, `predict.cglasso` and `impute`.

Examples

```r
set.seed(123)
# Y ~ N(0, Sigma) and probability of left/right censored values equal to 0.05
n <- 1000L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
residuals(out, type = "observed", rho.id = 3L, drop = TRUE)
residuals(out, type = "working", rho.id = 3L, drop = TRUE)

# Y ~ N(b0 + XB, Sigma) and probability of left/right censored values equal to 0.05
n <- 1000L
p <- 3L
q <- 2
```
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
residuals(out, type = "observed", lambda.id = 3L, rho.id = 3L, drop = TRUE)
residuals(out, type = "working", lambda.id = 3L, rho.id = 3L, drop = TRUE)

---

rowNames + colNames  
Row and Column Names of a “datacggm” Object

Description

Retrieve or set the row or column names of a “datacggm” object.

Usage

rowNames(x)
rowNames(x) <- value
colNames(x)
colNames(x) <- value

Arguments

x  an R object of class ‘datacggm’.
value  a named list with elements ‘X’ and ‘Y’.

Details

For an R object of class ‘datacggm’, rowNames (colNames) retrieves or set rownames (colnames) for matrices Y and X. See below for some examples.

Author(s)

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

See Also

datacggm, rcggm and the method function dimnames.
Examples

set.seed(123)
# a dataset from a censored Gaussian graphical model
n <- 100L
p <- 3L
Z <- rcggm(n = n, p = p, probl = 0.05, probr = 0.05)

rowNames(Z)
rowNames(Z) <- list(Y = paste0("i", seq_len(n)))
# the same as rowNames(Z)$Y <- paste0("i", seq_len(n))
rowNames(Z)

colNames(Z)
colNames(Z) <- list(Y = paste("Y", 1:p, sep = ":"))
# the same as colNames(Z)$Y <- paste("Y", 1:p, sep = ":")
colNames(Z)

# a dataset from a conditional censored Gaussian graphical model
n <- 100L
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
Z <- rcggm(n = n, b0 = b0, X = X, B = B, probl = 0.05, probr = 0.05)

rowNames(Z)$Y <- paste0("i", seq_len(n))
rowNames(Z)$X <- paste0("i", seq_len(n))
dimnames(Z)

colNames(Z)$Y <- paste("Y", 1:p, sep = ":")
colNames(Z)$X <- paste("X", 1:q, sep = ":")
dimnames(Z)

select_cglasso

Model Selection for the Conditional Graphical Lasso Estimator

Description

‘select_cglasso’ returns the optimal fitted model selected by a chosen measure of goodness-of-fit.

Usage

select_cglasso(object, GoF = AIC, ...)

Arguments

object an R object of class cglasso.
select.cglasso

GoF a valid goodness-of-fit function, such as \texttt{AIC.cglasso} or \texttt{BIC.cglasso}, or an \texttt{R} object of class ‘GoF’.

... further arguments passed to the chosen goodness-of-fit function (argument ‘GoF’).

Details

The function \texttt{select.cglasso} evaluates the goodness-of-fit of the models fitted by cglasso and extracts the selected model.

Model evaluation can be made in two ways. The easiest way is to use a valid goodness-of-fit function, such as \texttt{AIC.cglasso} or \texttt{BIC.cglasso}. In this case, further arguments are passed to these functions by ‘...’. The second way consists on passing the output of a goodness-of-fit function, that is, an \texttt{R} object of class ‘GoF’. Usually, this approach is preferable when the computation of the chosen goodness-of-fit measure is time-consuming, such as when the sample size is small relative to the number of parameters and the AIC or BIC functions are used to evaluate a long sequence of fitted models. In these cases, we suggest the computation of several measures of goodness-of-fit in a preliminary step and then the use of the \texttt{select.cglasso} function in a subsequent step to select the optimal fitted model.

Value

‘select.cglasso’ returns the optimal fitted model.

Author(s)

Gianluca Sottile (<gianluca.sottile@unipa.it>)

See Also

\texttt{cglasso}, \texttt{AIC.cglasso} and \texttt{BIC.cglasso}.

Examples

\begin{verbatim}
set.seed(123)
# Y ~ N(0, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcgmm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)

out <- cglasso(. ~ ., data = Z)
select.cglasso(out) # models selection by AIC
select.cglasso(out, GoF = BIC) # models selection by BIC
select.cglasso(out, GoF = BIC, mle = TRUE, g = 0.5) # models selection by eBIC

# Y ~ N(b0 + XB, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
\end{verbatim}
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)

out <- cglasso(. ~ ., data = Z, lambda = 0.01)
select_cglasso(out)  # models selection by AIC
select_cglasso(out, GoF = BIC)  # models selection by BIC
select_cglasso(out, GoF = BIC, mle = TRUE, g = 0.5)  # models selection by eBIC

out <- cglasso(. ~ ., data = Z, rho = 0.01)
select_cglasso(out)  # models selection by AIC
select_cglasso(out, GoF = BIC)  # models selection by BIC
select_cglasso(out, GoF = BIC, mle = TRUE, g = 0.5)  # models selection by eBIC

out <- cglasso(. ~ ., data = Z)
select_cglasso(out)  # models selection by AIC
select_cglasso(out, GoF = BIC)  # models selection by BIC
select_cglasso(out, GoF = BIC, mle = TRUE, g = 0.5)  # models selection by eBIC

--
ShowStructure

Description

ShowStructure gives information about the package structure.

Usage

ShowStructure(module = c("ALL", "DM", "MF", "MS", "NA"), description = TRUE, plot = TRUE)

Arguments

module a description of the module for which the description is required.
plot logical. If TRUE a graph showing the package structure is plotted.
description logical. If TRUE a description of the available function is printed on screen.

Value

ShowStructure silently returns a named list with components:
description a named list containing the description of the available functions.
graph an \texttt{R} object of class ‘igraph’ describing the package structure.

Author(s)

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

‘sammary’ produces a summary of the models fitted using cglasso or cggm.

Usage

## S3 method for class 'cglasso'
summary(object, GoF = AIC, print.all = TRUE, digits = 3L, ...)

Arguments

object an R object inheriting class cglasso, that is, the output of the model-fitting functions cglasso and cggm.

GoF a valid goodness-of-fit function, such as AIC.cglasso or BIC.cglasso, or an R object of class 'GoF'.

print.all logical. Should all summary statistics be printed?

digits the minimum number of significant digits to be used: see 'print.default'.

... the penalty parameter passed to the goodness-of-fit function (argument ‘GoF’) or further arguments passed to 'print.data.frame' and 'print.listof'.

Details

The function summary.cglasso computes the summary statistics needed to evaluate the goodness-of-fit of the models fitted by cglasso or cggm.

Model evaluation can be made in two ways. The easiest way is to use a valid goodness-of-fit function, such as AIC.cglasso or BIC.cglasso. In this case, further arguments can be passed to these functions by the argument ‘...’. The second way consists in passing the output of a goodness-of-fit function, that is, an R object of class ‘GoF’. Usually, this approach is preferable when the computation of the chosen goodness-of-fit measure is time-consuming, such as when the sample size is small relative to the number of parameters and the AIC or BIC functions are used to evaluate a sequence of fitted models. In these cases, we suggest the computation of several measures of goodness-of-fit in a preliminary step and then the use of summary.cglasso in a subsequent step to evaluate the sequence of fitted models.

To improve the readability of the results, the output is divided into two sections.

The first section is structured in a sequence of tables showing, for each combination of the two tuning parameters ‘lambda’ and ‘rho’, the number of estimated non-zero regression coefficients (‘df.B’), the number of estimated non-zero partial correlation coefficients (‘df.Tht’), the degrees-of-freedom (‘df’), and the value of the goodness-of-fit measure used to evaluate the fitted models. To help the user with the identification of the optimal fitted model, the last column of each table reports the ranking of the models (where the optimal model is marked with the symbol ‘<-’).

The second section reports the summary statistics of the selected optimal model.
Value

A named list with the following elements is silently returned:

- **table**: data.frame containing the summary statistics used to evaluate the sequence of fitted models.
- **lambda.id**: position of the optimal $\lambda$-value identified by the chosen goodness-of-fit function.
- **rho.id**: position of the optimal $\rho$-value identified by the chosen goodness-of-fit function.

Author(s)

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

See Also

cglasso, cggm, AIC.cglasso and BIC.cglasso.

Examples

```r
set.seed(123)

# Y ~ N(0, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)

out <- cglasso(. ~ ., data = Z)
summary(out) # models evaluation by AIC
summary(out, GoF = BIC) # models evaluation by BIC
summary(out, GoF = BIC, mle = TRUE, g = 0.5) # models evaluation by eBIC

# Y ~ N(b0 + XB, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
q <- 2
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)

out <- cglasso(. ~ ., data = Z, lambda = 0.01)
summary(out) # models evaluation by AIC
summary(out, GoF = BIC) # models evaluation by BIC
summary(out, GoF = BIC, mle = TRUE, g = 0.5) # models evaluation by eBIC

out <- cglasso(. ~ ., data = Z, rho = 0.01)
summary(out) # models evaluation by AIC
summary(out, GoF = BIC) # models evaluation by BIC
```
summary.datacggm

```r
summary(out, GoF = BIC, mle = TRUE, g = 0.5)  # models evaluation by eBIC
out <- cglasso(. ~ ., data = Z)
summary(out)  # models evaluation by AIC
summary(out, GoF = BIC)  # models evaluation by BIC
summary(out, GoF = BIC, mle = TRUE, g = 0.5)  # models evaluation by eBIC
```

---

**Description**

The function `summary.datacggm` produces summaries of an object of class `datacggm`.

**Usage**

```r
## S3 method for class 'datacggm'
summary(object, n, quantile.type = 7L, digits = 3L, quote = FALSE, ...)
```

**Arguments**

- `object`: an object of class `datacggm` for which a summary is desired.
- `n`: maximum number of rows printed on video. By default all rows are printed.
- `quantile.type`: an integer between 1 and 9 selecting one of the nine quantile algorithms (see also `quantile` function).
- `digits`: integer, used for number formatting with `format`.
- `quote`: logical, indicating whether or not strings should be printed with surrounding quotes.
- `...`: further arguments passed to `print.listof`.

**Details**

The function `summary.datacggm` extends the results given by `summary.matrix()` taking into account the specific structure of an object of class `datacggm`. Summary statistics are printed out in two sections: the first section reports the summary statistics for the response matrix ‘Y’, whereas the second section is devoted to summarising the matrix of predictors ‘X’.

For each response variable, the first section reports the standard summary statistics computed using only the observed values, that is the entries of the response matrix whose status indicators are equals to zero (see `event` for more details), together with the information about the lower and upper censoring values (columns named `Lower` and `Upper`) and the percentage of missing-at-random, left-censored and right-censored response values (columns named `NA%`, `LC%` and `RC%`, respectively).

The second section is divided into two subsections reporting the main summary statistics computed for numeric and categorical predictors, respectively.

The two sections are printed out in such a way that the readability of the summary statistics is improved in a high-dimensional setting, that is, when the number of predictors and response variables exceeds the sample size.
Value

`summary.datacggm` returns a named list with components

- **Y**: a matrix with class ‘table’ obtained by computing the summary measures on the response variables.
- **X.numeric**: a matrix with class ‘table’ obtained by computing the summary measures on the numeric predictors.
- **X.categorical**: a list containing the summary measures on the categorical predictors.

Author(s)

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

References


See Also

datacggm, rcggm and event.

Examples

```r
set.seed(123)

# case 1: Y ~ N(0, Sigma)
# 1. probability of left/right censored values equal to 0.05
# 2. probability of missing-at-random equals to 0.05
n <- 50L
p <- 100L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05, probna = 0.05)
summary(Z, max = n * p)

# case 2: Y ~ N(b0 + XB, Sigma) and
# 1. probability of left/right censored values equal to 0.05
# 2. probability of missing-at-random equals to 0.05
n <- 50L
p <- 100L
q <- 100L
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
```
rho <- 0.3
Sigma <- outer(1:p, 1:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05, probna = 0.05)
summary(Z)
summary(Z, max = n * p)

---

### to_graph

**Create Graphs from cglasso or cggm Objects**

#### Description

`to_graph` returns a named list of graphs using the results of an R object of class `cglasso` or `cggm`.

#### Usage

```r
to_graph(object, GoF = AIC, lambda.id, rho.id, weighted = FALSE, simplify = TRUE, ...)
```

#### Arguments

- **object**: an R object inheriting class `cglasso`, that is, the output of the model-fitting functions `cglasso` and `cggm`.
- **GoF**: a valid goodness-of-fit function, such as `AIC.cglasso` or `BIC.cglasso`, or an R object of class `GoF`.
- **lambda.id**: optional. If `object` has class `cglasso`, this argument is an integer used to identify a specific fitted cglasso model, otherwise (i.e. if the object has class `cggm`) this argument can be omitted. See section ‘Details’ for more details.
- **rho.id**: optional. If `object` has class `cglasso`, this argument is an integer used to identify a specific fitted cglasso model, otherwise (i.e. if object has class `cggm`) this argument can be omitted. See section ‘Details’ for more details.
- **weighted**: logical. Should weighted graphs be created? Default is `FALSE`.
- **simplify**: logical. Should isolated vertices be removed from the graph? Default is `TRUE`, i.e., isolated vertices are removed.
- **...**: further arguments passed to the chosen goodness-of-fit function (argument ‘GoF’).

#### Details

`to_graph` returns a named list of graphs using the results of an R object of class `cglasso` or `cggm`.

If `object` has class `cglasso`, then the goodness-of-fit function passed through the argument `GoF` is used to identify the adjacency matrix (`object$InfoStructure$Adj_yy`) describing the undirected edges among the $p$ response variables. If the model is fitted using $q$ predictors, then the matrix describing the effects of the predictors onto the response variables (see `object$InfoStructure$Adj_xy`)
is also returned. Finally, these matrices are used to return an undirected and directed graph. Optionally, the user can identify a specific fitted model using the arguments lambda.id and rho.id. If object has class ‘cggm’, then GoF, lambda.id and rho.id can be omitted.

If argument weighted is set equal to ‘TRUE’, then the estimated precision matrix and, if available, the estimated regression coefficient matrix are used to return weighted graphs. In this case, edges associated with positive estimates are shown using a solid line. Otherwise, a dashed line is used.

Value
‘to_graph’ returns an R object of S3 class “cglasso2igraph”, i.e., a named list containing the following components:

- **Gyy**: an undirected graph representing the conditional dependence structure among the \( p \) response variables.
- **Gxy**: a directed graph representing the effects of the \( q \) predictors onto the \( p \) response variables.

Each component is an R object of class `igraph`.

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

See Also
cglasso, cggm and plot.cglasso2igraph. For more details about the object of class ‘igraph’, the interested reader is referred to the package `igraph`.

Examples
```
set.seed(123)
# Y ~ N(0, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, Sigma = Sigma, probl = 0.05, probr = 0.05)
out <- cglasso(. ~ ., data = Z)
out.graph <- to_graph(out)
out.graph

# Y ~ N(b0 + XB, Sigma) and probability of left/right censored values equal to 0.05
n <- 100L
p <- 3L
q <- 2L
b0 <- runif(p)
B <- matrix(runif(q * p), nrow = q, ncol = p)
X <- matrix(rnorm(n * q), nrow = n, ncol = q)
rho <- 0.3
Sigma <- outer(1L:p, 1L:p, function(i, j) rho^abs(i - j))
Z <- rcggm(n = n, b0 = b0, X = X, B = B, Sigma = Sigma, probl = 0.05, probr = 0.05)
```
out <- cglasso(~., data = Z)
out.graph <- to_graph(out, lambda.id = 3, rho.id = 3, weighted = TRUE)
out.graph
Index

* array
  ColMeans + ColMeans, 19
dim.datacggm, 23
dimnames.datacggm, 24
lower + upper, 38
rowNames + colNames, 61
* classes
  datacggm, 20
event, 25
getGraph, 29
getMatrix, 30
is.cglasso2igraph, 36
is.datacggm, 37
nobs + nresp + npred, 41
* datagen
  rcggm, 56
* datasets
  Example, 27
  MKMEP, 39
  MM, 40
* distribution
  rcggm, 56
* graphs
  plot.cggm, 42
  plot.cglasso, 43
  plot.cglasso2igraph, 47
  plot.GoF, 49
to_graph, 69
* hplot
  hist.datacggm, 32
  qqcnorm, 54
* manip
  ColMeans + ColMeans, 19
dimnames.datacggm, 24
lower + upper, 38
rowNames + colNames, 61
* methods
  datacggm, 20
event, 25
getGraph, 29
getMatrix, 30
is.cglasso2igraph, 36
is.datacggm, 37
nobs + nresp + npred, 41
summary.datacggm, 67
* models
  AIC.cglasso, 4
  BIC.cglasso, 6
cglasso-package, 2
  coef, 17
  fitted, 28
  impute, 34
  predict, 50
  QFun, 52
  residuals, 59
  select_cglasso, 62
  summary.cglasso, 65
* multivariate
  cglasso-package, 2
  rcggm, 56
* package
  cglasso-package, 2
  ShowStructure, 64
* regression
  cggm, 9
cglasso, 12
  coef, 17
  fitted, 28
  impute, 34
  predict, 50
  residuals, 59
  select_cglasso, 62
  summary.cglasso, 65

abline, 49
AIC(AIC.cglasso), 4
AIC.cglasso, 4, 8, 9, 11, 14, 16, 27, 39, 40,
  44–46, 49, 50, 52, 53, 63, 65, 66, 69

72
INDEX

BIC (BIC.cglasso), 6
BIC.cglasso, 5, 6, 9, 11, 14, 16, 27, 39, 40, 44–46, 49, 50, 52, 53, 63, 65, 66, 69

cggm, 4–6, 8, 9, 14, 16–18, 28, 34, 35, 42, 52, 53, 59, 60, 65, 66, 69, 70
cglasso, 4–6, 8, 9, 11, 12, 17, 18, 22, 27, 28, 34, 35, 39, 40, 43, 44, 46, 50–53, 59, 60, 63, 65, 66, 69, 70
cglasso-package, 2
col, 14, 17, 27, 39, 40
colMeans, 11, 14, 16, 28, 35, 51, 60
ColMeans, 21, 22, 23, 33, 55, 56
ColMeans (ColMeans + ColMeans), 19
ColMeans + ColMeans, 19
colNames, 22, 25
colNames (rowNames + colNames), 61
colNames <- rowNames + colNames, 61
ColVars, 21, 22, 23, 33, 55, 56
ColVars (ColMeans + ColMeans), 19
contour, 49
datacggm, 12, 13, 16, 19, 20, 23–26, 30, 31, 33, 37, 38, 41, 55, 56, 58, 61, 68
device, 32, 44, 54
dim (dim.datacggm), 23
dim.datacggm, 22, 23, 41
dimnames, 24, 61
dimnames (dimnames.datacggm), 24
dimnames.datacggm, 24
dimnames <- datacggm
dimnames.datacggm, 24
dimnames.datacggm, 24
event, 21, 22, 25, 67, 68
Example, 27
filled.contour, 49
fitted, 28
fitted.cglasso, 11, 14, 16, 18, 35, 51, 60
format, 67

getGraph, 29, 42, 48
getMatrix, 21, 22, 25, 30
hist, 32
hist (hist.datacggm), 32
hist.datacggm, 19, 21, 22, 32, 56
igraph, 70
impute, 11, 14, 16, 18, 28, 34, 51, 60
is.cglasso2igraph, 30, 36
is.datacggm, 22, 37
lower, 22
lower (lower + upper), 38
lower + upper, 38
mean, 19
MKMEP, 27, 39
MKMEP.Sim (Example), 27
MM, 27, 40
MM.Sim (Example), 27
model.matrix.default, 12
mtext, 49
mvnorm, 57
nobs, 23
nobs (nobs + nresp + npred), 41
nobs + nresp + npred, 41
npred, 23
nresp (nobs + nresp + npred), 41
nresp, 23
nresp (nobs + nresp + npred), 41
par, 49
plot, 27, 39, 40, 49
plot.cglasso, 42
plot.cglasso, 43
plot.cglasso2igraph, 11, 14, 16, 30, 42, 47, 70
plot.GoF, 4, 5, 7, 8, 14, 16, 49
plot.histogram, 33
plot.igraph, 42, 48
points, 49
predict, 50
predict.cgmm, 11
predict.cglasso, 14, 16, 18, 28, 35, 60
print.data.frame, 65
print.default, 65
print.listof, 65, 67
QFun, 4–8, 52
qqcnorm, 19, 21, 22, 33, 54
quantile, 67
rcggm, 19, 22, 23, 25, 26, 31, 33, 37, 41, 56, 56, 61, 68
residuals, 59
residuals.cglasso, 11, 14, 16, 18, 28, 35, 51
rowNames, 22, 25
rowNames (rowNames + colNames), 61
rowNames + colNames, 61
rowNames<- (rowNames + colNames), 61

select_cglasso, 4, 7, 16, 50, 53, 62
ShowStructure, 10, 11, 14, 16, 64
summary, 27, 39, 40
summary.cglasso, 4, 5, 7, 8, 11, 14, 16, 50, 53, 65
summary.datacggm, 21, 22, 67
summary.matrix, 67

to_graph, 11, 14, 16, 27, 29, 30, 36, 39, 40, 42, 48, 53, 69

upper, 22
upper (lower + upper), 38

var, 19