Package 'cxreg'

July 5, 2026

Title Complex-Valued Lasso and Complex-Valued Graphical Lasso

Version 1.1.2

Description Implements 'glmnet'-style complex-valued lasso (CLASSO) and complex-valued graphical lasso (CGLASSO) via a pathwise coordinate descent algorithm for complex-valued parameters, using an isomorphism between complex numbers and 2x2 orthogonal matrices. Also provides a full inference pipeline for high-dimensional sparse spectral precision matrices, including data-driven bandwidth selection, one-step debiasing, asymptotic variance estimation, entry-wise confidence regions, and FDR-controlled hypothesis testing. Supporting tools for cross-validation, simulation, coefficient extraction, and plotting are included. See Deb, Kuceyeski, and Basu (2024) <doi:10.48550/arXiv.2401.11128> and Deb, Kim, and Basu (2026) <doi:10.48550/arXiv.2606.07986>.

License MIT + file LICENSE

Depends R (>= 3.5.0)

Imports fields, foreach, gdata, grDevices, Matrix, methods, mvtnorm, stats, utils

Suggests devtools, doParallel, knitr, rmarkdown, testthat (>= 3.0.0), tinytex, tools, xfun

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation yes

Author Younghoon Kim [aut, cre],
Navonil Deb [aut],
Sumanta Basu [aut]

Maintainer Younghoon Kim <ykim124@ua.edu>

Repository CRAN

Date/Publication 2026-07-05 20:30:17 UTC

Contents

`cxreg-package` *Complex-valued Lasso, graphical Lasso, and spectral inference*

Description

This package fits a complex-valued Lasso for regression using coordinate descent. The algorithm is extremely fast and exploits sparsity in the input `x` matrix where it exists. A variety of predictions can be made from the fitted models.

Details

This package also provides fitting for a complex-valued graphical Lasso (CGLASSO) using coordinate descent. The function is built upon `classo` with covariate updates, just as the regular real-valued coordinate descent algorithm for graphical Lasso.

Beyond estimation, the package implements a full inference pipeline for high-dimensional sparse spectral precision matrices:

- `select_m`: data-driven bandwidth selection via generalised cross-validation (GCV) on the diagonal periodogram.
- `decglasso`: one-step debiasing (desparsification) of the CGLASSO estimator.
- `var.cov`: asymptotic variance and pseudovariance estimation (plug-in or HAC) with real/imaginary decomposition.
- `spec.test`: entry-wise Z-statistics, Mahalanobis chi-squared statistics, and confidence ellipse areas.
- `spec.fdr`: FDR-controlled multiple testing for spectral precision matrix support recovery.

Package:	cxreg
Type:	Package
Version:	1.1.0
Date:	2026-06-01
License:	MIT + file LICENSE

Author(s)

Younghoon Kim, Navonil Deb, Sumanta Basu
Maintainer: Younghoon Kim ykim124@ua.edu

References

Deb, N., Kuceyeski, A., Basu, S. (2024) *Regularized Estimation of Sparse Spectral Precision Matrices*, https://arxiv.org/abs/2401.11128.

Deb, N., Kim, Y., Basu, S. (2026) *Inference for High-Dimensional Sparse Spectral Precision Matrices*, https://arxiv.org/abs/2606.07986.

Examples

```
## --- classo: complex-valued lasso regression ---
set.seed(1234)
n <- 100; p <- 20
x <- array(rnorm(n*p), c(n,p)) + (1+1i) * array(rnorm(n*p), c(n,p))
for (j in 1:p) x[,j] <- x[,j] / sqrt(mean(Mod(x[,j])^2))
e <- rnorm(n) + (1+1i) * rnorm(n)
b <- c(1, -1, rep(0, p-2)) + (1+1i) * c(-0.5, 2, rep(0, p-2))
y <- x %*% b + e
fit <- classo(x, y)
predict(fit, newx = x[1:5, ], s = c(0.01, 0.005))
predict(fit, type = "coef")
```

```
plot(fit, xvar = "lambda")



## --- cglasso: complex-valued graphical lasso ---
library(mvtnorm)
p <- 30; n <- 500
C <- diag(0.7, p)
C[row(C) == col(C) + 1] <- 0.3
C[row(C) == col(C) - 1] <- 0.3
Sigma <- solve(C)
set.seed(1010)
X_t <- rmvnorm(n = n, mean = rep(0, p), sigma = Sigma)
m <- floor(sqrt(n)); j <- 1
d_j <- dft.j(X_t, j, m)
f_j_hat <- t(d_j) %*% Conj(d_j) / (2*m + 1)
fit <- cglasso(S = f_j_hat, m = m, type = "I")



## --- Inference pipeline: debiasing, variance, testing ---
library(mvtnorm)
p <- 10; n <- 200
set.seed(42)
X <- rmvnorm(n, mean = rep(0, p), sigma = diag(p))

## Bandwidth selection
bw_sel <- select_m(X)
m <- bw_sel$m_opt

## Spectral density estimate and cglasso fit
j <- floor(n / 4)
dft  <- dft.all(X)
fhat <- fhat_at(dft, j, m)
fit  <- cglasso(S = fhat, m = m)

## Debiasing
res  <- decglasso(object = fit, fhat = fhat)

## Variance estimation
vc <- var.cov(Theta = res$Theta_tilde, X = X, j = j, m = m, type = "plug-in")

## Confidence regions and test statistics (H0: Theta = 0)
st <- spec.test(Est = res$Theta_tilde, varcov = vc, m = m, alpha = 0.05)

## FDR-controlled support recovery
fdr_res <- spec.fdr(Chi_sq = st$Chi_sq, alpha = 0.05, diag = FALSE)
fdr_res$tau
fdr_res$Decision
```

`buildPredmat` *Build Prediction Matrix*

Description

These are not intended for use by users. Constructs a prediction matrix for cross-validation folds using the coefficient paths in `outlist`. Inspired by internal functions in the **glmnet** package.

Usage

```
buildPredmat(outlist, lambda, x, foldid, alignment, ...)

## Default S3 method:
buildPredmat(outlist, lambda, x, foldid, alignment, ...)
```

Arguments

`outlist`	A list of fitted `classo` models, one per fold.
`lambda`	A vector of penalty values from the master fit.
`x`	The full predictor matrix (complex-valued).
`foldid`	Integer vector of fold assignments, length `nrow(x)`.
`alignment`	Alignment method: `"lambda"` (default) aligns fold predictions to the master lambda sequence; `"fraction"` aligns by fraction of path progress.
`...`	Other arguments passed to `predict`.

Value

A complex matrix of predicted values, `nrow(x)` by `length(lambda)`. Rows correspond to observations held out in each fold; all other entries are `NA`.

`cglasso` *fit a complex-valued graphical lasso*

Description

Fit a complex-valued graphical lasso for spectral precision matrix (inverse spectral density matrix) via a complex variable-wise coordinate descent algorithm for classo with covariates update.

Usage

```
cglasso(
  S,
  D = NULL,
  type = c("I", "II"),
  m,
  lambda = NULL,
  nlambda = 50,
  lambda.min.ratio = 0.01,
  W.init = NULL,
  stopping_rule = TRUE,
  stop_criterion = c("EBIC", "AIC", "RMSE"),
  maxit = 500,
  thresh = 1e-04,
  trace.it = 0,
  ...
)
```

Arguments

S	nvar x nvar-dimensional symmetric spectral density (or spectral coherence) matrix. S is considered as being computed by average smoothed periodogram (the bandwidth is computed by using the given nobs).
D	The nvar x nvar-dimensional diagonal matrix to produce the partial spectral coherence matrix from S. Default is NULL. If D is not specified, the function automatically takes the inverse square root of the diagonals of S.
type	A logical flag to choose the formulation to solve. Default is I. If type is I, the algorithm solves CGLASSO-I in the reference, $$D^{-1/2}\left(\arg\min_{\Theta}\operatorname{Tr}\left[\hat{R}\hat{\Theta}\right]-\log\det\Theta+\sum_{i\neq j}|\Theta_{ij}|\right)D^{-1/2}$$ for the given D. If type is II, the algorithm solves CGLASSO-II in the reference. It is for each iterative classo with covariate update, the squared-root of scale matrix $D^{-1/2}$ is multiplied. Please refer to Algorithm 2 in the reference for the details.
m	Window size. This quantity is need to compute the Fourier frequency, extended BIC, and bandwidth for the average smoothed periodogram.
lambda	A user supplied lambda sequence. Typical usage is to have the program compute its own lambda sequence based on nlambda and lambda.min.ratio. Supplying a value of lambda overrides this. WARNING: use with care. Avoid supplying a single value for lambda
nlambda	The number of lambda values - default is 50.
lambda.min.ratio	Smallest value for lambda, as a fraction of lambda.max, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). ???

W.init	Logical flag whether the initially estimated spectral density matrix is given. Default is NULL.
stopping_rule	Logical flag if the algorithm is terminated by stopping rule. If the algorithm is early terminated ,not all estimates for initially designated lambdas are explored.
stop_criterion	Stopping criterion for early termination. Default is EBIC (Extended BIC). Alternatively, AIC (AIC) and RMSE (root mean squared error between two consecutive estimates) can be used.
maxit	Maximum number of iterations of both outer and inner loops. Default 500.
thresh	Convergence threshold for coordinate descent. Default is 1e-4.
trace.it	If trace.it=1, then a progress bar is displayed;
...	Other arguments that can be passed to cglasso useful for big models that take a long time to fit.

Details

The sequence of models implied by lambda is fit by coordinate descent.

Value

An object with class "cglassofit" and "cglasso".

stop_arr Sequence of values of information criterion for a fixed lambda.

stop_criterion Stopping criterion used.

min_index The index for lambda that minimizes the value of the information criterion.

lambda_grid Sequence of lambdas used.

Theta_list Estimated inverse spectral matrix for each fixed lambda. It is provided in the list.

type Type of the formulation used, either CGLASSO-I or CGLASSO-II.

D Used scale diagonal matrix.

Author(s)

Navonil Deb, Younghoon Kim, Sumanta Basu
Maintainer: Younghoon Kim <ykim124@ua.edu>

References

Deb, N., Kim, Y., Basu, S. (2026) *Inference for High-Dimensional Sparse Spectral Precision Matrices*, https://arxiv.org/abs/2606.07986.

Examples

```
p <- 30
n <- 500
C <- diag(0.7, p)
C[row(C) == col(C) + 1] <- 0.3
C[row(C) == col(C) - 1] <- 0.3
Sigma <- solve(C)
```

```
set.seed(1010)
m <- floor(sqrt(n)); j <- 1
X_t <- mvtnorm::rmvnorm(n = n, mean = rep(0, p), sigma = Sigma)
d_j <- dft.j(X_t,j,m)
f_j_hat <- t(d_j) %*% Conj(d_j) / (2*m+1)
fit <- cglasso(S=f_j_hat, m=floor(sqrt(n)))
```

`cglasso.path` *fit a complex-valued graphical Lasso for a path of lambda values*

Description

Fit a complex-valued graphical Lasso formulation for a path of lambda values. `cglasso.path` solves the penalized Gaussian maximum likelihood problem for a path of lambda values.

Usage

```
cglasso.path(
  S,
  D,
  type,
  m,
  lambda,
  nlambda,
  lambda.min.ratio,
  W.init,
  stopping_rule,
  stop_criterion,
  maxit,
  thresh,
  trace.it,
  ...
)
```

Arguments

`S`	p x p-dimensional symmetric spectral density (or spectral coherence) matrix. S is considered as being computed by average smoothed periodogram (the bandwidth is computed by using the given nobs).
`D`	The p x p-dimensional diagonal matrix to produce the partial spectral coherence matrix from S. Default is `NULL`. If D is not specified, the function automatically takes the inverse square root of the diagonals of S.
`type`	Logical flag to choose the formulation to solve. Default is `I`. If type is `I`, the algorithm solves CGLASSO-I in the reference,

$$D^{-1/2}\Big(\arg\min_{\Theta} \mathrm{Tr}[\hat{R}\hat{\Theta}] - \log\det\Theta + \sum_{i\neq j}|\Theta_{ij}|\Big)D^{-1/2}$$

for the given D. If type is `II`, the algorithm solves CGLASSO-II in the reference: for each iterative classo with covariate update, the scale matrix D is multiplied. Please see the reference for details.

`m` Window size. This quantity is need to compute the Fourier frequency, extended BIC, and bandwidth for the average smoothed periodogram.

`lambda` A user supplied `lambda` sequence. Typical usage is to have the program compute its own `lambda` sequence based on `nlambda` and `lambda.min.ratio`. Supplying a value of `lambda` overrides this. WARNING: use with care. Avoid supplying a single value for `lambda`

`nlambda` The number of `lambda` values - default is 50.

`lambda.min.ratio`
Smallest value for `lambda`, as a fraction of `lambda.max`, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). ???

`W.init` Logical flag whether the initially estimated spectral density matrix is given. Default is `NULL`.

`stopping_rule` Logical flag if the algorithm is terminated by stopping rule. If the algorithm is early terminated, not all estimates for initially designated lambdas are explored.

`stop_criterion` Stopping criterion for early termination. Default is `EBIC` (Extended BIC). Alternatively, `AIC` (AIC) and `RMSE` (root mean squared error between two consecutive estimates) can be used.

`maxit` Maximum number of iterations of both outer and inner loops. Default 500.

`thresh` Convergence threshold for coordinate descent. Default is 1e-4.

`trace.it` If `trace.it=1`, then a progress bar is displayed; useful for big models that take a long time to fit.

`...` Other arguments that can be passed to `cglasso`

Value

An object with class `"cglassofit"` and `"cglasso"`.

stop_arr Sequence of values of the information criterion for each lambda.

stop_criterion Stopping criterion used.

min_index The index for the lambda that minimizes the information criterion.

lambda_grid Sequence of lambdas used.

Theta_list Estimated spectral precision matrix for each fixed lambda, provided as a list.

type Type of the formulation used, either CGLASSO-I or CGLASSO-II.

D Scale diagonal matrix used.

`cglasso_example` *Example data for cglasso*

Description

A synthetic dataset used to illustrate the complex-valued graphical Lasso (`cglasso`). The data consist of a smoothed periodogram matrix at a single Fourier frequency, computed from a simulated multivariate time series with a known sparse spectral precision matrix structure.

Usage

```
data(cglasso_example)
```

Format

A list with the following components:

`f_hat` A $p \times p$ complex-valued Hermitian matrix. The smoothed periodogram (spectral density) estimate at Fourier frequency index `j`, computed via `fhat_at` with half-bandwidth `m` = `floor(sqrt(n))`.

`n` A positive integer. The number of time series observations used to generate the data. The half-bandwidth is `m` = `floor(sqrt(n))`.

`p` A positive integer. The number of variables (dimension of the time series).

`j` A positive integer. The Fourier frequency index at which `f_hat` was evaluated.

Details

The dataset is generated from a $p = 30$-dimensional VMA(3) process with $n = 500$ observations. The half-bandwidth `m` is chosen as `floor(sqrt(n))` and the smoothed periodogram `f_hat` is evaluated at frequency index `j` = 1.

References

Deb, N., Kim, Y., Basu, S. (2026) *Inference for High-Dimensional Sparse Spectral Precision Matrices*, https://arxiv.org/abs/2606.07986.

See Also

`cglasso`, `fhat_at`, `dft.all`

`classo` *fit a complex-valued lasso*

Description

Fit a complex-valued lasso formulation via complex update coordinate descent algorithm. By defining a field isomophism between complex values and its 2 by 2 representation, it enables to update each coordinate of the solution as a regular coordinate descent algorithm.

Usage

```
classo(
  x,
  y,
  weights = NULL,
  lambda = NULL,
  nlambda = 100,
  lambda.min.ratio = ifelse(nobs < nvars, 0.01, 1e-04),
  standardize = TRUE,
  intercept = FALSE,
  maxit = 10000,
  thresh = 1e-07,
  trace.it = 0,
  ...
)
```

Arguments

`x`	input matrix, of dimension nobs x nvars; each row is an observation vector. Requirement: `nvars >1`; in other words, `x` should have 2 or more columns.
`y`	response variable.
`weights`	observation weights. Default is 1 for each observation.
`lambda`	A user supplied `lambda` sequence. Typical usage is to have the program compute its own `lambda` sequence based on `nlambda` and `lambda.min.ratio`. Supplying a value of `lambda` overrides this. WARNING: use with care. Avoid supplying a single value for `lambda` (for predictions after CV use `predict()` instead). Supply instead a decreasing sequence of `lambda` values. `classo` relies on its warms starts for speed, and its often faster to fit a whole path than compute a single fit.
`nlambda`	The number of `lambda` values - default is 100.
`lambda.min.ratio`	Smallest value for `lambda`, as a fraction of `lambda.max`, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). The default depends on the sample size `nobs` relative to the number of variables `nvars`. If `nobs > nvars`, the default is `0.0001`, close to zero. If `nobs < nvars`, the default is `0.01`. A very small value of `lambda.min.ratio` will lead to a saturated fit in the `nobs < nvars` case.

standardize	Logical flag for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is standardize = TRUE.
intercept	Should intercept(s) set to zero (default=FALSE) or be fitted (TRUE).
maxit	Maximum number of iterations of outer loop. Default 10,000.
thresh	Convergence threshold for coordinate descent. Each inner coordinate-descent loop continues until the maximum change in the objective after any coefficient update is less than thresh times the null deviance. Defaults value is 1e-7.
trace.it	If trace.it=1, then a progress bar is displayed; useful for big models that take a long time to fit.
...	Other arguments that can be passed to classo

Details

The sequence of models implied by lambda is fit by coordinate descent.

Value

An object with class "classofit" and "classo".

a0 Intercept sequence of length length(lambda).

beta A nvars x length(lambda) matrix of coefficients, stored in sparse matrix format.

df The number of nonzero coefficients for each value of lambda.

dim Dimension of coefficient matrix.

lambda The actual sequence of lambda values used. When alpha=0, the largest lambda reported does not quite give the zero coefficients reported (lambda=inf would in principle). Instead, the largest lambda for alpha=0.001 is used, and the sequence of lambda values is derived from this.

dev.ratio The fraction of (null) deviance explained. The deviance calculations incorporate weights if present in the model. The deviance is defined to be 2*(loglike_sat - loglike), where loglike_sat is the log-likelihood for the saturated model (a model with a free parameter per observation). Hence dev.ratio=1-dev/nulldev.

nulldev Null deviance (per observation). This is defined to be 2*(loglike_sat -loglike(Null)). The null model refers to the intercept model.

call The call that produced this object.

nobs Number of observations.

Author(s)

Navonil Deb, Younghoon Kim, Sumanta Basu
Maintainer: Younghoon Kim <yk748@cornell.edu>

References

Deb, N., Kuceyeski, A., Basu, S. (2024) *Regularized Estimation of Sparse Spectral Precision Matrices*, https://arxiv.org/abs/2401.11128.

Examples

```
set.seed(1010)
n <- 1000
p <- 200
x <- array(rnorm(n*p), c(n,p)) + (1+1i) * array(rnorm(n*p), c(n,p))
for (j in 1:p) x[,j] <- x[,j] / sqrt(mean(Mod(x[,j])^2))
e <- rnorm(n) + (1+1i) * rnorm(n)
b <- c(1, -1, rep(0, p-2)) + (1+1i) * c(-0.5, 2, rep(0, p-2))
y <- x %*% b + e
fit.test <- classo(x, y)
```

---

classo.control *Internal classo parameters*

Description

View and/or change the factory default parameters in classo

Usage

```
classo.control(
  fdev = NULL,
  devmax = NULL,
  eps = NULL,
  big = NULL,
  mnlam = NULL,
  pmin = NULL,
  exmx = NULL,
  prec = NULL,
  mxit = NULL,
  itrace = NULL,
  epsnr = NULL,
  mxitnr = NULL,
  factory = FALSE
)
```

Arguments

fdev	minimum fractional change in deviance for stopping path; factory default = 1.0e-5
devmax	maximum fraction of explained deviance for stopping path; factory default = 0.999
eps	minimum value of lambda.min.ratio (see classo); factory default= 1.0e-6
big	large floating point number; factory default = 9.9e35. Inf in definition of upper.limit is set to big

mnlam	minimum number of path points (lambda values) allowed; factory default = 5
pmin	minimum probability for any class. factory default = 1.0e-9. Note that this implies a pmax of 1-pmin.
exmx	maximum allowed exponent. factory default = 250.0
prec	convergence threshold for multi response bounds adjustment solution. factory default = 1.0e-10
mxit	maximum iterations for multiresponse bounds adjustment solution. factory default = 100
itrace	If 1 then progress bar is displayed when running classo and cv.classo. factory default = 0
epsnr	convergence threshold for classo.fit. factory default = 1.0e-6
mxitnr	maximum iterations for the IRLS loop in classo.fit. factory default = 25
factory	If TRUE, reset all the parameters to the factory default; default is FALSE

Details

If called with no arguments, classo.control() returns a list with the current settings of these parameters. Any arguments included in the call sets those parameters to the new values, and then silently returns. The values set are persistent for the duration of the R session.

Value

A list with named elements as in the argument list

See Also

classo

Examples

```
classo.control(fdev = 0)  # continue along path even though not much changes
classo.control()           # view current settings
classo.control(factory = TRUE)  # reset all parameters to their default
```

---

classo.path *fit a complex-valued lasso for a path of lambda values*

Description

Fit a complex-valued lasso formulation for a path of lambda values. classo.path solves the Lasso problem for a path of lambda values.

Usage

```
classo.path(
  x,
  y,
  weights = NULL,
  standardize = FALSE,
  lambda = NULL,
  nlambda = 100,
  lambda.min.ratio = ifelse(nobs < nvars, 0.01, 1e-04),
  intercept = FALSE,
  thresh = 1e-10,
  maxit = 1e+05,
  trace.it = 0,
  ...
)
```

Arguments

x	Complex-valued input matrix, of dimension nobs by nvar; each row is an observation vector.
y	Complex-valued response variable, nobs dimensional vector.
weights	Observation weights. Default is 1 for each observation.
standardize	Logical flag for x variable standardize beforehand; i.e. for n and p by nobs and nvar, $$\|X_j\| = \sqrt{n} \text{ for all } j = 1, \ldots, p$$ is satisfied for the input x. Default is FALSE.
lambda	A user supplied lambda sequence. Default is NULL.
nlambda	The number of lambda values. Default is 100.
lambda.min.ratio	If nobs < nvars, the default is 0.01.
intercept	Should intercept be set to zero (default=FALSE) or fitted (TRUE)? This default is reversed from glmnet package.
thresh	Convergence threshold for coordinate descent. Each inner coordinate-descent loop continues until the maximum change in the objective after any coefficient update is less than thresh times the null deviance. Default value is 1e-10.
maxit	Maximum number of iterations of outer loop. Default 10,000.
trace.it	Controls how much information is printed to screen. Default is trace.it = 0 (no information printed). If trace.it = 1, a progress bar is displayed. If trace.it=2, some information about the fitting procedure is printed to the console as the model is being fitted.
...	Other arguments that can be passed to classo

Value

An object with class "classofit" and "classo".

a0 Intercept sequence of length length(lambda).

beta A nvars x length(lambda) matrix of coefficients, stored in sparse matrix format.

df The number of nonzero coefficients for each value of lambda.

dim Dimension of coefficient matrix.

lambda The actual sequence of lambda values used. When alpha=0, the largest lambda reported does not quite give the zero coefficients reported (lambda=inf would in principle). Instead, the largest lambda for alpha=0.001 is used, and the sequence of lambda values is derived from this.

dev.ratio The fraction of (null) deviance explained. The deviance calculations incorporate weights if present in the model. The deviance is defined to be 2*(loglike_sat - loglike), where loglike_sat is the log-likelihood for the saturated model (a model with a free parameter per observation). Hence dev.ratio=1-dev/nulldev.

nulldev Null deviance (per observation). This is defined to be 2*(loglike_sat -loglike(Null)). The null model refers to the intercept model.

call The call that produced this object.

nobs Number of observations.

---

classo_example *Example data for classo*

Description

A synthetic dataset used to illustrate the complex-valued Lasso (classo). The data consist of a complex-valued design matrix and response vector generated from a sparse linear model.

Usage

```
data(classo_example)
```

Format

A list with the following components:

x A complex-valued matrix of dimension $n \times p$ (1000×200). Each column has been normalised so that $\|x_j\|/\sqrt{n} = 1$.

y A complex-valued vector of length n. The response $y = X\beta + \varepsilon$ where ε is complex Gaussian noise.

beta A complex-valued vector of length p. The true sparse coefficient vector, with nonzero entries at positions 1 and 2.

n A positive integer. The number of observations.

p A positive integer. The number of variables.

Details

The dataset is generated with $n = 1000$ observations and $p = 200$ variables. The true coefficient vector β has two nonzero entries. Each column of x is normalised to have unit complex magnitude. The response y is $X\beta + \varepsilon$ with complex Gaussian noise ε.

References

Deb, N., Kuceyeski, A., Basu, S. (2024) *Regularized Estimation of Sparse Spectral Precision Matrices*, https://arxiv.org/abs/2401.11128.

See Also

classo, cv.classo

---

coef.classo *Extract coefficients from a classo object*

Description

Similar to other predict methods, this function predicts fitted values, coefficients and more from a fitted "classo" object.

Usage

```
## S3 method for class 'classo'
coef(object, s = NULL, exact = FALSE, ...)

## S3 method for class 'classo'
predict(
  object,
  newx,
  s = NULL,
  type = c("link", "response", "coefficients", "nonzero"),
  exact = FALSE,
  ...
)
```

Arguments

object	Fitted "classo" model object.
s	Value(s) of the penalty parameter lambda at which predictions are required. Default is the entire sequence used to create the model.
exact	Relevant only when predictions are made at values of s different from those used in fitting. If exact = FALSE (default), linear interpolation is used. If exact = TRUE, the model is refit at the new s values; the original x and y (and weights if used) must be supplied as additional named arguments.

`...`	Additional arguments, e.g. `x=` and `y=` when `exact = TRUE`.
`newx`	Matrix of new values for `x` at which predictions are to be made. Must be a complex-valued matrix. Not used for `type = c("coefficients", "nonzero")`.
`type`	Type of prediction required. `"link"` and `"response"` give the fitted values $X\hat{\beta}$. `"coefficients"` returns the coefficient matrix at the requested values of `s`. `"nonzero"` returns a list of indices of nonzero coefficients for each value of `s`.

Details

`coef(...)` is equivalent to `predict(type="coefficients",...)`. `"link"` and `"response"` are treated identically and both return the fitted complex-valued predictions $\hat{y} = X\hat{\beta}$.

Value

The object returned depends on `type`. `"coefficients"` returns a matrix; `"nonzero"` returns a list of index vectors; `"link"` and `"response"` return a complex matrix of predictions.

Author(s)

Younghoon Kim, Navonil Deb, and Sumanta Basu
Maintainer: Younghoon Kim ykim124@ua.edu

References

Deb, N., Kuceyeski, A. and Basu, S. (2024) *Regularized estimation of sparse spectral precision matrices*, https://arxiv.org/abs/2401.11128.

See Also

`classo`, and `print`, and `coef` methods, and `cv.classo`.

`coef.cv.classo` *Extract coefficients from a cv.classo object*

Description

A convenience wrapper that extracts coefficients from the `classo.fit` stored in a `"cv.classo"` object at a chosen lambda value.

Usage

```
## S3 method for class 'cv.classo'
coef(object, s = c("lambda.1se", "lambda.min"), ...)
```

Arguments

object	Fitted "cv.classo" object.
s	Value(s) of the penalty parameter lambda. Default is "lambda.1se". Alternatively "lambda.min" can be used, or a numeric value (or vector) of lambda directly.
...	Additional arguments passed to coef.classo.

Value

A complex-valued coefficient matrix; see predict.classo.

See Also

classo, coef.classo, cv.classo.

---

cv.classo *Cross-validation for classo*

Description

Does k-fold cross-validation for classo, produces a plot, and returns a value for lambda

Usage

```
cv.classo(
  x,
  y,
  weights = NULL,
  lambda = NULL,
  nfolds = 10,
  foldid = NULL,
  alignment = c("lambda", "fraction"),
  keep = FALSE,
  parallel = FALSE,
  trace.it = 0,
  ...
)
```

Arguments

x	x matrix as in classo.
y	response y as in classo.
weights	Observation weights; defaults to 1 per observation

lambda	Optional user-supplied lambda sequence; default is NULL, and classo chooses its own sequence. Note that this is done for the full model (master sequence), and separately for each fold. The fits are then aligned using the master sequence (see the alignment argument for additional details). Adapting lambda for each fold leads to better convergence. When lambda is supplied, the same sequence is used everywhere.
nfolds	number of folds - default is 10. Although nfolds can be as large as the sample size (leave-one-out CV), it is not recommended for large dataset. Smallest value allowable is nfolds=3
foldid	an optional vector of values between 1 and nfolds identifying what fold each observation is in. If supplied, nfolds can be missing.
alignment	This is an experimental argument, designed to fix the problems users were having with CV, with possible values "lambda" (the default) else "fraction". With "lambda" the lambda values from the master fit (on all the data) are used to line up the predictions from each of the folds. In some cases this can give strange values, since the effective lambda values in each fold could be quite different. With "fraction" we line up the predictions in each fold according to the fraction of progress along the regularization. If in the call a lambda argument is also provided, alignment="fraction" is ignored (with a warning).
keep	If keep=TRUE, a *prevalidated* array is returned containing fitted values for each observation and each value of lambda. This means these fits are computed with this observation and the rest of its fold omitted. The foldid vector is also returned. Default is keep=FALSE.
parallel	If TRUE, use parallel foreach to fit each fold. Must register a parallel backend before calling, e.g. via doParallel::registerDoParallel() or doMC::registerDoMC(). Limited progress tracing is available when parallel = TRUE.
trace.it	If trace.it=1, then progress bars are displayed; useful for big models that take a long time to fit. Limited tracing if parallel=TRUE
...	Other arguments that can be passed to classo

Details

The function runs classo nfolds+1 times; the first to get the lambda sequence, and then the remainder to compute the fit with each of the folds omitted. The error is accumulated, and the average error and standard deviation over the folds is computed.

Note that the results of cv.classo are random, since the folds are selected at random. Users can reduce this randomness by running cv.classo many times, and averaging the error curves.

Value

an object of class "cv.classo" is returned, which is a list with the ingredients of the cross-validation fit.

lambda the values of lambda used in the fits.

cvm The mean cross-validated error - a vector of length length(lambda).

cvsd estimate of standard error of cvm.

cvup upper curve = cvm+cvsd.

cvlo lower curve = cvm-cvsd.

nzero number of non-zero coefficients at each lambda.

name a text string indicating type of measure for plotting purposes.

classo.fit a fitted classo object for the full data.

lambda.min value of lambda that gives minimum cvm.

lambda.1se largest value of lambda such that error is within 1 standard error of the minimum.

fit.preval if keep=TRUE, this is the array of pre-validated fits. Some entries can be NA, if that and subsequent values of lambda are not reached for that fold.

foldid if keep=TRUE, the fold assignments used.

index a one column matrix with the indices of lambda.min and lambda.1se in the sequence of coefficients, fits etc.

Author(s)

Navonil Deb, Younghoon Kim, Sumanta Basu
Maintainer: Younghoon Kim <ykim124@ua.edu>

See Also

classo and plot and coef methods for "cv.classo".

Examples

```
set.seed(1010)
n <- 1000
p <- 200
x <- array(rnorm(n*p), c(n,p)) + (1+1i) * array(rnorm(n*p), c(n,p))
for (j in 1:p) x[,j] <- x[,j] / sqrt(mean(Mod(x[,j])^2))
e <- rnorm(n) + (1+1i) * rnorm(n)
b <- c(1, -1, rep(0, p-2)) + (1+1i) * c(-0.5, 2, rep(0, p-2))
y <- x %*% b + e
cv.test <- cv.classo(x, y)
```

---

cxreg *Simulated data for the cxreg vignette*

Description

Simple simulated data, used to demonstrate the features of cxreg

Format

Data objects used to demonstrate features in the cxreg vignette

Details

These datasets are artificial, and are used to test out some of the features of cxreg.

Examples

```
data(classo_example)
x <- classo_example$x
y <- classo_example$y
classo(x, y)

data(cglasso_example)
f_hat <- cglasso_example$f_hat
m     <- floor(sqrt(cglasso_example$n))
cglasso(S = f_hat, m = m, type = "I")
cglasso(S = f_hat, m = m, type = "II")
```

---

decglasso *Debias the cglasso estimator*

Description

Computes the one-step debiased (desparsified) estimator of the spectral precision matrix from a fitted `cglasso` object. The debiasing correction follows the formula

$$\tilde{\Theta} = 2\hat{\Theta} - \hat{\Theta}\hat{f}\hat{\Theta},$$

where $\hat{\Theta}$ is the `cglasso` estimate and $\hat{f}$ is the smoothed periodogram at the target frequency.

Usage

```
decglasso(object, fhat, index = NULL)
```

Arguments

`object`	A fitted object of class `"cglassofit"` returned by `cglasso`, or a $p \times p$ complex-valued matrix giving $\hat{\Theta}$ directly.
`fhat`	A $p \times p$ complex-valued matrix. The smoothed spectral density (periodogram) estimate at the target Fourier frequency, typically the output of `fhat_at()`.
`index`	Integer. When `object` is a `"cglassofit"` object, `index` selects which element of `object$Theta_list` to use as $\hat{\Theta}$. Defaults to `object$min_index`, i.e. the lambda chosen by the stopping criterion.

Value

A list of class `"decglasso"` with the following components:

`Theta_tilde` The $p \times p$ complex-valued debiased spectral precision matrix estimate.

`Theta_hat` The $p \times p$ complex-valued `cglasso` estimate used as input to the debiasing step.

`fhat` The smoothed spectral density matrix supplied via `fhat`.

Author(s)

Navonil Deb, Younghoon Kim, Sumanta Basu
Maintainer: Younghoon Kim <ykim124@ua.edu>

References

Deb, N., Kim Y., Basu, S. (2026) *Inference for High-Dimensional Sparse Spectral Precision Matrices*, https://arxiv.org/abs/2606.07986.

See Also

cglasso, cglasso.path

Examples

```
library(mvtnorm)
p <- 30
n <- 500
C <- diag(0.7, p)
C[row(C) == col(C) + 1] <- 0.3
C[row(C) == col(C) - 1] <- 0.3
Sigma <- solve(C)
set.seed(1010)
m <- floor(sqrt(n)); j <- 1
X_t <- rmvnorm(n = n, mean = rep(0, p), sigma = Sigma)
dft  <- dft.all(X_t)
fhat <- fhat_at(dft, j, m)
fit  <- cglasso(S = fhat, m = m)

res <- decglasso(object = fit, fhat = fhat)
Theta_tilde <- res$Theta_tilde
```

---

dft.all *Discrete Fourier Transform of matrix X*

Description

Computes the (normalized) discrete Fourier transform (DFT) of a matrix `X` row-wise using `mvfft` over all Fourier frequencies.

Usage

```
dft.all(X)
```

Arguments

`X`	A numeric matrix of size $nobs \times nvar$, where DFT is applied across the rows (time points).

Value

A complex-valued matrix of dimension $nobs \times nvar$ representing all DFT components of the original matrix.

`dft.j` *Discrete Fourier Transform of matrix X at a fixed frequency omega_j*

Description

Computes the (normalized) discrete Fourier transform (DFT) of a matrix `X` row-wise using `mvfft`, and extracts a window of frequencies centered at a target index.

Usage

```
dft.j(X, j, m)
```

Arguments

`X`	A numeric matrix of size $nobs \times nvar$, where DFT is applied across the rows (time points).
`j`	An integer index in $1, \ldots, nobs$ around which the frequency window is centered.
`m`	A non-negative integer specifying the window half-width. The function returns `2m + 1` DFT components centered around `j`.

Value

A complex-valued matrix of dimension $(2m + 1) \times nvar$ representing selected DFT components of the original matrix.

`family.classo` *Extract the family from a classo object*

Description

Returns the model family for objects fitted by `classo` or `cv.classo`. All `classo` models fit a complex-valued Gaussian likelihood, so the family is always `"gaussian"`.

Usage

```
## S3 method for class 'classo'
family(object, ...)

## S3 method for class 'classofit'
family(object, ...)

## S3 method for class 'cv.classo'
family(object, ...)
```

Arguments

`object`	A fitted model object of class `"classo"`, `"classofit"`, or `"cv.classo"`.
`...`	Not used.

Value

A character string: `"gaussian"`.

See Also

`classo`, `cv.classo`

---

fhat_at *Smoothed periodogram at a fixed frequency*

Description

Extracts a window of `2m + 1` DFT components centered at index `j` from a pre-computed full DFT matrix and returns their averaged outer product, i.e. the smoothed periodogram (spectral density) estimate at frequency $\omega_j = 2\pi j/n$:

$$\hat{f}(\omega_j) = \frac{1}{2m+1} \sum_{k=j-m}^{j+m} d_k d_k^*,$$

where d_k is the k-th row of the DFT matrix (as a column vector) and $*$ denotes the conjugate transpose.

Usage

```
fhat_at(dft, j, m)
```

Arguments

`dft`	A complex-valued matrix of dimension $nobs \times nvar$, typically the output of `dft.all`.
`j`	An integer index in $1, \ldots, nobs$ at which the smoothed periodogram is evaluated.
`m`	A non-negative integer specifying the window half-width. Frequency indices outside $[1, nobs]$ are wrapped circularly.

Value

A $nvar \times nvar$ complex-valued Hermitian matrix giving the smoothed periodogram estimate at frequency ω_j.

Author(s)

Navonil Deb, Younghoon Kim, Sumanta Basu
Maintainer: Younghoon Kim <ykim124@ua.edu>

References

Deb, N., Kim Y., Basu, S. (2026) *Inference for High-Dimensional Sparse Spectral Precision Matrices*, https://arxiv.org/abs/2606.07986.

See Also

dft.all, dft.j

Examples

```
p <- 30
n <- 500
set.seed(1010)
X_t <- matrix(rnorm(n * p), n, p)
dft  <- dft.all(X_t)
m    <- floor(sqrt(n)); j <- 1
fhat <- fhat_at(dft, j, m)
```

---

plot.cglasso *Plot heatmap from a "cglasso" object*

Description

Produces a heatmap of the estimated spectral precision matrix for a fitted "cglasso" object. An inverse spectral matrix heatmap is produced.

Usage

```
## S3 method for class 'cglasso'
plot(
  x,
  index = 1,
  type = c("mod", "real", "imaginary", "both"),
  label = FALSE,
  ...
)
```

Arguments

x	Fitted "cglasso" model, i.e. an object of class "cglassofit" returned by cglasso.
index	Integer index selecting which element of x$Theta_list to plot (i.e. which lambda value). Must be in $1, \ldots,$ length(x$Theta_list). Default is 1.

type	Which component to plot: "real" for the real part, "imaginary" for the imaginary part, "mod" for the modulus (absolute value), or "both" for real and imaginary side by side. Default is "mod".
label	If TRUE, label the axes with variable names.
...	Other graphical parameters passed to image.plot.

Value

No return value; called for its side effect of producing a plot.

Author(s)

Navonil Deb, Younghoon Kim, Sumanta Basu
Maintainer: Younghoon Kim <ykim124@ua.edu>

See Also

cglasso

---

plot.classo *plot coefficients from a "classo" object*

Description

Produces a coefficient profile plot of the coefficient paths for a fitted "classo" object.

Usage

```
## S3 method for class 'classo'
plot(x, xvar = c("norm", "lambda", "dev"), label = FALSE, ...)
```

Arguments

x	fitted "classo" model
xvar	What is on the X-axis. "norm" plots against the L1-norm of the coefficients, "lambda" against the log-lambda sequence, and "dev" against the percent deviance explained.
label	If TRUE, label the curves with variable sequence numbers.
...	Other graphical parameters to plot

Details

A coefficient profile plot is produced.

Value

No return value, called for side effects (produces a plot).

Author(s)

Navonil Deb, Younghoon Kim, Sumanta Basu
Maintainer: Younghoon Kim <ykim124@ua.edu>

See Also

`classo`

`plot.cv.classo` *plot the cross-validation curve produced by cv.classo*

Description

Plots the cross-validation curve, and upper and lower standard deviation curves, as a function of the `lambda` values used.

Usage

```
## S3 method for class 'cv.classo'
plot(x, sign.lambda = 1, ...)
```

Arguments

`x`	fitted "`cv.classo`" object
`sign.lambda`	Either plot against `log(lambda)` (default) or its negative if `sign.lambda=-1`.
`...`	Other graphical parameters to plot.

Details

A plot is produced, and nothing is returned.

Value

No return value, called for side effects (produces a plot).

Author(s)

Navonil Deb, Younghoon Kim, Sumanta Basu
Maintainer: Younghoon Kim <ykim124@ua.edu>

See Also

`classo` and `plot` methods for "`cv.classo`".

Examples

```
set.seed(1010)
n <- 1000
p <- 200
x <- array(rnorm(n*p), c(n,p)) + (1+1i) * array(rnorm(n*p), c(n,p))
for (j in 1:p) x[,j] <- x[,j] / sqrt(mean(Mod(x[,j])^2))
e <- rnorm(n) + (1+1i) * rnorm(n)
b <- c(1, -1, rep(0, p-2)) + (1+1i) * c(-0.5, 2, rep(0, p-2))
y <- x %*% b + e
cv.test <- cv.classo(x, y)
plot(cv.test)
```

---

predict.cv.classo *Make predictions from a "cv.classo" object*

Description

This function makes predictions from a cross-validated classo model, using the stored "classo.fit" object and a chosen lambda value.

Usage

```
## S3 method for class 'cv.classo'
predict(object, newx, s = c("lambda.1se", "lambda.min"), ...)
```

Arguments

object	Fitted "cv.classo" object.
newx	Matrix of new values for x at which predictions are to be made. Must be a complex-valued matrix.
s	Value(s) of the penalty parameter lambda at which predictions are required. Default is "lambda.1se". Alternatively "lambda.min" can be used, or a numeric value (or vector) of lambda directly.
...	Additional arguments passed to the predict method for "classo" objects.

Value

The object returned by predict.classo for the chosen s: a complex-valued matrix of predicted values, or a coefficient matrix, depending on type.

Author(s)

Younghoon Kim, Navonil Deb, Sumanta Basu
Maintainer: Younghoon Kim ykim124@ua.edu

See Also

classo, predict.classo, coef.cv.classo, cv.classo.

print.classo *print a classo object*

Description

Print a summary of the classo path at each step along the path.

Usage

```
## S3 method for class 'classo'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	fitted classo object
digits	significant digits in printout
...	additional print arguments

Details

The call that produced the object x is printed, followed by a three-column matrix with columns Df, %Dev and Lambda. The Df column is the number of nonzero coefficients (Df is a reasonable name only for lasso fits). %Dev is the percent deviance explained (relative to the null deviance).

Value

The matrix above is silently returned

See Also

classo, predict and coef methods.

print.cv.classo *print a cross-validated classo object*

Description

Print a summary of the results of cross-validation for a classo model.

Usage

```
## S3 method for class 'cv.classo'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	fitted 'cv.classo' object
digits	significant digits in printout
...	additional print arguments

Details

A two-row table is printed showing the optimal lambda values selected by the minimum CV error (lambda.min) and the 1-standard-error rule (lambda.1se), along with the corresponding CV error, its standard error, and the number of nonzero coefficients.

Value

The matrix above is silently returned

See Also

classo, predict and coef methods.

---

select_m *Select the optimal bandwidth* m *for spectral density estimation*

Description

Searches over a grid of half-bandwidths m and selects the one that minimises a generalised cross-validation (GCV) criterion derived from the gamma deviance of the periodogram, as proposed by Ombao et al. (2001). For each candidate m, the smoothed diagonal periodogram $\hat{f}_{jj}(\omega_j)$ is compared to the raw diagonal periodogram $I_{jj}(\omega_j)$ via the average gamma deviance

$$D(m) = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} w_j \left[-\log\left(\frac{I_{jj}}{f_{jj}}\right) + \frac{I_{jj} - f_{jj}}{f_{jj}} \right],$$

and the GCV score is $\mathrm{GCV}(m) = D(m)/(1 - 1/(2m+1))^2$.

Usage

```
select_m(
  X,
  m_grid = NULL,
  freq_idx = NULL,
  q_weights = NULL,
  lambda_fun = NULL,
  eps = 1e-10,
  verbose = TRUE
)
```

Arguments

X	A numeric matrix of size $nobs \times nvar$ (time points by variables).
m_grid	An integer vector of candidate half-bandwidths to search over. Default is a data-driven grid of multiples of $\sqrt{nobs}$, filtered to satisfy $m \geq 1$ and $2m+1 < nobs$.
freq_idx	An integer vector of frequency indices (0-based) over which the GCV criterion is averaged. Default is `0:(nobs-1)`, i.e. all Fourier frequencies. The DC component (`j = 0`) and, for even `n`, the Nyquist frequency (`j = nobs/2`) are automatically down-weighted to 0.5 via `q_weights`.
q_weights	A numeric vector of non-negative weights of the same length as `freq_idx`, controlling the relative contribution of each frequency to the GCV criterion. Default weights are 1 for all frequencies, except 0.5 for the DC component (`j = 0`) and for the Nyquist frequency (`j = nobs/2`) when `nobs` is even. Weights are internally normalised to have mean 1.
lambda_fun	A function of the form `function(m, p)` returning the regularisation parameter λ to pass to `cglasso` for a given half-bandwidth `m` and number of variables `p`. Default is `function(m, p) sqrt(log(p) / (2*m + 1))`.
eps	A small positive number used as a lower bound when clamping diagonal periodogram values to avoid `log(0)`. Default `1e-10`.
verbose	Logical. If `TRUE` (default), prints a progress line for each candidate `m`.

Value

A list with the following components:

`m_opt` The optimal half-bandwidth minimising the GCV score.

`bw_opt` The corresponding full bandwidth `2*m_opt + 1`.

`lambda_opt` The λ value returned by `lambda_fun` at `m_opt`.

`gcv_min` The minimum GCV score.

`gcv_table` A `data.frame` with columns `m`, `bw`, `lambda`, `deviance`, `gcv`, and `gcv_denom`, one row per candidate in `m_grid`.

`all` A list of full output from `gcv_theta_one_m` for each candidate `m`.

Author(s)

Navonil Deb, Younghoon Kim, Sumanta Basu
Maintainer: Younghoon Kim <ykim124@ua.edu>

References

Ombao, H. C., Raz, J. A., Strawderman, R. L., Von Sachs, R. (2001). *A simple generalised cross-validation method of span selection for periodogram smoothing.* Biometrika, **88**(4), 1186–1192. doi:10.1093/biomet/88.4.1186.

See Also

`cglasso`, `fhat_at`, `dft.all`

Examples

```
p <- 10
n <- 200
set.seed(42)
X <- matrix(rnorm(n * p), n, p)
res <- select_m(X)
res$m_opt
res$gcv_table
```

`spec.fdr` *FDR-controlled hypothesis testing for spectral precision matrix entries*

Description

Applies an asymptotic FDR control procedure to the matrix of chi-squared test statistics produced by `spec.test`, returning the rejection threshold, the binary decision matrix, and (optionally) the empirical FDR and power when the true support is known.

Usage

```
spec.fdr(Chi_sq, alpha, diag = FALSE, Truth = NULL, grid_size = 100)
```

Arguments

`Chi_sq`	A $nvar \times nvar$ matrix of chi-squared test statistics, typically the `Chi_sq` component of a `spec.test` result evaluated at `Truth = 0`.
`alpha`	A number in $(0, 1)$. Target FDR level.
`diag`	Logical. If `FALSE` (default), diagonal entries are excluded from the multiple testing procedure. If `TRUE`, they are included.
`Truth`	An optional $nvar \times nvar$ logical or binary matrix indicating the true support (`TRUE`/`1` for non-zero entries). When supplied, the empirical FDR and power are computed. Default `NULL` (only the threshold and decision matrix are returned).
`grid_size`	Integer. Number of points in the threshold grid. Default `100`.

Details

The threshold $\hat{\tau}$ is the infimum over a grid of values $t \in (0, 2\log q]$ of

$$\frac{q\, e^{-t/2}}{\max(1,\, |\{T_{ab} \geq t\}|)} \leq \alpha,$$

where q is the number of upper-triangular entries tested and $e^{-t/2}$ approximates the tail probability of a $\chi^2(2)$ distribution. Entries with $T_{ab} \geq \hat{\tau}$ are rejected.

Value

A list of class `"specfdr"` with the following components:

`Decision` A $nvar \times nvar$ binary matrix with 1 for rejected entries.

`tau` The selected threshold $\hat{\tau}$, or `NA` if no threshold satisfies the FDR constraint.

`FDR` Empirical FDR (only present when `Truth` is supplied).

`Power` Empirical power (only present when `Truth` is supplied).

Author(s)

Navonil Deb, Younghoon Kim, Sumanta Basu
Maintainer: Younghoon Kim <ykim124@ua.edu>

References

Deb, N., Kim Y., Basu, S. (2026) *Inference for High-Dimensional Sparse Spectral Precision Matrices*, https://arxiv.org/abs/2606.07986.

Examples

```
library(mvtnorm)
p <- 10; n <- 200
set.seed(42)
X    <- rmvnorm(n, mean = rep(0, p), sigma = diag(p))
m    <- floor(sqrt(n)); j <- 1; alpha <- 0.05
dft  <- dft.all(X)
fhat <- fhat_at(dft, j, m)
fit  <- cglasso(S = fhat, m = m)
res  <- decglasso(object = fit, fhat = fhat)
vc   <- var.cov(Theta = res$Theta_tilde, X = X, j = j, m = m,
                type = "plug-in")
st   <- spec.test(Est = res$Theta_tilde, varcov = vc, m = m, alpha = alpha)

# FDR-controlled test (no truth known):
fdr_res <- spec.fdr(Chi_sq = st$Chi_sq, alpha = alpha, diag = FALSE)
fdr_res$tau
fdr_res$Decision
```

`spec.test` *Confidence regions and test statistics for spectral precision matrix entries*

Description

Computes entry-wise Z-statistics, half-widths of marginal confidence intervals, Mahalanobis chi-squared statistics, and joint elliptical confidence region areas for the real and imaginary parts of the debiased spectral precision matrix estimator $\tilde{\Theta}$.

Usage

```
spec.test(Est, varcov, m, alpha, Truth = NULL, eps = 1e-08)
```

Arguments

`Est`	A $nvar \times nvar$ complex-valued matrix. The debiased spectral precision matrix estimate $\tilde{\Theta}$, typically the `Theta_tilde` component of a `decglasso` result.
`varcov`	A `"varcov"` object returned by `var.cov`, supplying the estimated `Vre`, `Vim`, and `Cov` matrices.
`m`	A positive integer. The half-bandwidth used to compute $\hat{f}$; the full bandwidth $\mathrm{bw} = 2m + 1$ is used to scale the asymptotic covariance.
`alpha`	A number in $(0, 1)$. Nominal significance level for confidence intervals and confidence regions.
`Truth`	A $nvar \times nvar$ complex-valued matrix giving the true (or hypothesised) value of Θ. Default is `matrix(0 + 0i, nvar, nvar)`, which corresponds to testing $H_0 : \Theta_{ab} = 0$ for all entries.
`eps`	A small positive number used to regularise near-zero variances. Default `1e-8`.

Details

For each entry (a, b), the joint asymptotic distribution of $(\mathrm{Re}\,\tilde{\Theta}_{ab},\ \mathrm{Im}\,\tilde{\Theta}_{ab})$ is bivariate normal with covariance Σ_{ab}/bw, where Σ_{ab} is the 2×2 matrix built from the `Vre`, `Vim`, and `Cov` components of `varcov`. Diagonal entries are treated as real-valued (one degree of freedom); off-diagonal entries use a 2-df chi-squared statistic. The default null hypothesis is $H_0 : \Theta_{ab} = 0$ (i.e. `Truth = 0`), which corresponds to hypothesis testing. A non-zero `Truth` can be supplied for coverage evaluation in simulation studies.

Value

A list of class `"spectest"` with the following $nvar \times nvar$ matrices:

`Chi_sq` Mahalanobis chi-squared test statistics. For off-diagonal entries these are asymptotically $\chi^2(2)$ under the null; for diagonal entries $\chi^2(1)$.

`area` Area of the joint $(1 - \alpha)$ confidence ellipse for $(\mathrm{Re}\,\tilde{\Theta}_{ab},\ \mathrm{Im}\,\tilde{\Theta}_{ab})$.

`Z_re` Marginal Z-statistics for the real parts.

`wing_re` Half-widths of marginal $(1 - \alpha)$ confidence intervals for the real parts.

`Z_im` Marginal Z-statistics for the imaginary parts.

`wing_im` Half-widths of marginal $(1 - \alpha)$ confidence intervals for the imaginary parts.

Author(s)

Navonil Deb, Younghoon Kim, Sumanta Basu
Maintainer: Younghoon Kim <ykim124@ua.edu>

References

Deb, N., Kim Y., Basu, S. (2026) *Inference for High-Dimensional Sparse Spectral Precision Matrices*, https://arxiv.org/abs/2606.07986.

Examples

```
library(mvtnorm)
p <- 10; n <- 200
set.seed(42)
X    <- rmvnorm(n, mean = rep(0, p), sigma = diag(p))
m    <- floor(sqrt(n)); j <- 1; alpha <- 0.05
dft  <- dft.all(X)
fhat <- fhat_at(dft, j, m)
fit  <- cglasso(S = fhat, m = m)
res  <- decglasso(object = fit, fhat = fhat)
vc   <- var.cov(Theta = res$Theta_tilde, X = X, j = j, m = m,
                type = "plug-in")

# Hypothesis test (H0: Theta = 0):
st <- spec.test(Est = res$Theta_tilde, varcov = vc, m = m, alpha = alpha)
st$Chi_sq[1:3, 1:3]
```

`var.cov` *Variance-covariance estimation for the debiased spectral precision matrix*

Description

Estimates the asymptotic variance and pseudovariance of the debiased (desparsified) `cglasso` estimator $\tilde{\Theta}$ at a target Fourier frequency ω_j, then decomposes them into the variance of the real part, variance of the imaginary part, and their cross-covariance. Two estimators are available: a plug-in estimator based on the theoretical sandwich formula, and a heteroscedasticity- and autocorrelation-consistent (HAC) estimator.

Usage

```
var.cov(Theta, X, j, m, type = c("plug-in", "HAC"))
```

Arguments

`Theta`	A $nvar \times nvar$ complex-valued matrix. The debiased spectral precision matrix estimate $\tilde{\Theta}$ at frequency ω_j, typically the `Theta_tilde` component returned by `decglasso`.
`X`	A numeric matrix of size $nobs \times nvar$ (time points by variables). The DFT is computed internally via `dft.all`.
`j`	An integer frequency index (1-based) identifying the target Fourier frequency $\omega_j = 2\pi j/nobs$.

`m` A positive integer. The half-bandwidth used in the smoothed periodogram estimator; must satisfy $2m + 1 < nobs$.

`type` Character string selecting the variance estimator. Either `"plug-in"` (default) for the sandwich plug-in estimator, or `"HAC"` for the heteroscedasticity- and autocorrelation-consistent estimator.

Details

For a $p \times p$ complex-valued estimator $\tilde{\Theta}$, the asymptotic distribution involves both the variance matrix $V^{(ab)} = \mathrm{Var}(\tilde{\Theta}_{ab})$ and the pseudovariance matrix $PV^{(ab)} = \mathrm{PVar}(\tilde{\Theta}_{ab})$. From these, the variances and covariance of the real and imaginary parts are recovered via

$$\mathrm{Var}(\mathrm{Re}\,\tilde{\Theta}_{ab}) = \frac{1}{2}\mathrm{Re}(V + PV),$$

$$\mathrm{Var}(\mathrm{Im}\,\tilde{\Theta}_{ab}) = \frac{1}{2}\mathrm{Re}(V - PV),$$

$$\mathrm{Cov}(\mathrm{Re}\,\tilde{\Theta}_{ab},\, \mathrm{Im}\,\tilde{\Theta}_{ab}) = \frac{1}{2}\mathrm{Im}(PV).$$

Value

A list of class `"varcov"` with the following components:

`Vre` A $nvar \times nvar$ real matrix. Estimated variance of the real part of $\tilde{\Theta}$.

`Vim` A $nvar \times nvar$ real matrix. Estimated variance of the imaginary part of $\tilde{\Theta}$.

`Cov` A $nvar \times nvar$ real matrix. Estimated covariance between the real and imaginary parts of $\tilde{\Theta}$.

`V` A $nvar \times nvar$ complex matrix. The raw variance matrix before decomposition.

`PV` A $nvar \times nvar$ complex matrix. The raw pseudovariance matrix before decomposition.

`type` Character string indicating which estimator was used.

Author(s)

Navonil Deb, Younghoon Kim, Sumanta Basu
Maintainer: Younghoon Kim <ykim124@ua.edu>

References

Deb, N., Kim Y., Basu, S. (2026) *Inference for High-Dimensional Sparse Spectral Precision Matrices*, https://arxiv.org/abs/2606.07986.

Examples

```
library(mvtnorm)
p <- 10
n <- 200
set.seed(42)
X <- rmvnorm(n, mean = rep(0, p), sigma = diag(p))
m <- floor(sqrt(n)); j <- 1
dft  <- dft.all(X)
```

```
fhat <- fhat_at(dft, j, m)
fit  <- cglasso(S = fhat, m = m)
res  <- decglasso(object = fit, fhat = fhat)

# Plug-in variance estimator:
vc_plug <- var.cov(Theta = res$Theta_tilde, X = X, j = j, m = m,
                   type = "plug-in")

# HAC variance estimator:
vc_hac  <- var.cov(Theta = res$Theta_tilde, X = X, j = j, m = m,
                   type = "HAC")
```

Index