Package 'cxreg'

September 17, 2025

Title Complex-Valued Lasso and Complex-Valued Graphical Lasso

Version 1.0.0

Description Implements 'glmnet'-style complex-valued lasso and complex-valued graphical lasso using a 'pathwise' coordinate descent algorithm for complex-valued parameters. The package provides supporting tools for estimation, simulation, and prediction. See Deb (2024) <doi:10.48550/ArXiv.2401.11128> for the algorithm description.

License MIT + file LICENSE

Depends R (>= 3.5.0)

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

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation yes

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

Maintainer Younghoon Kim <yk748@cornell.edu>

Repository CRAN

Date/Publication 2025-09-17 10:50:02 UTC

Contents

`cxreg-package` *Complex-valued Lasso and graphical Lasso paths*

Description

This package fits 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 complex-valued graphical Lasso using coordinate descent. The function is built upon classo with covariate updates, just as the regular real-valued coordinate descent algorithm for graphical Lasso.

Package:	cxreg
Type:	Package
Version:	1.0.0
Date:	2025-07-01
License:	MIT + file LICENSE

Very simple to use. Accepts `x,y` data for penalized regression models, and produces the regularization paths over a grid of values for the tuning parameters `lambda`. Similarly, accepts `S,n` data for penalized Gaussian likelihood, and produce the regularization paths over a grid of values for the tuning parameter `lambda`.

Author(s)

Younghoon Kim, Navonil Deb, 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(1234)
x <- array(rnorm(100*20), c(100,20)) + (1+1i) * array(rnorm(100*20), c(100,20))
for (j in 1:20) x[,j] <- x[,j] / sqrt(mean(Mod(x[,j])^2))
e <- rnorm(100) + (1+1i) * rnorm(100)
b <- c(1, -1, rep(0, 18)) + (1+1i) * c(-0.5, 2, rep(0, 18))
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")



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.X(X_t,j,m)
f_j_hat <- t(d_j) %*% Conj(d_j) / (2*m+1)
fit <- cglasso(S=f_j_hat, nobs=n,type="I")
plot(fit$Theta_list,index=fit$min_index,type="mod",label=FALSE)
```

`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, ...)
```

Arguments

outlist	A list of fitted models.
lambda	A vector of penalty values.
x	A predictor matrix.
foldid	Fold identifiers.
alignment	Alignment method (not used).
...	Other arguments

Value

The return value depends on the method; for the default method, see buildPredmat.default.

---

buildPredmat.default *Build Prediction Matrix (Default Method)*

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

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

Arguments

outlist	A list of fitted models.
lambda	A vector of penalty values.
x	A predictor matrix.
foldid	Fold identifiers.
alignment	Alignment method (not used).
...	Other arguments

Value

A numeric matrix of predicted values with dimensions nrow(x) by length(lambda). Rows correspond to observations and columns correspond to penalty values.

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"),
  nobs,
  lambda = NULL,
  nlambda = 50,
  lambda.min.ratio = ifelse(nobs < p, 0.01, 1e-04),
  W.init = NULL,
  stopping_rule = TRUE,
  stop_criterion = c("EBIC", "AIC", "RMSE"),
  maxit = 500,
  thresh = 1e-04,
  trace.it = 0,
  ...
)
```

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 with spectral densities as the diagonal entries. Default is NULL. If D is not provided, diagonals of S are chosen.
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, $$D^{-1/2}\left(\arg\min_{\Theta} \mathrm{Tr}\left[\hat{R}\hat{\Theta}\right] - \log\det\Theta + \sum_{i\neq j}|\Theta_{ij}|\right)D^{-1/2}$$ 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 the equation (5.2) in the reference for the details.

nobs	Number of observations used in computation of the spectral density matrix S. 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). The default depends on the sample size nobs relative to the number of variables nvars. If nobs > p, the default is 0.0001, close to zero. If nobs < p, the default is 0.01.
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 CGALSSO-I or CGLASSO-II.
scale	Whether the spectral density matrix (covariance) or spectral coherence (coherence) is given.
D	Used scale diagonal matrix.

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

```
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.X(X_t,j,m)
f_j_hat <- t(d_j) %*% Conj(d_j) / (2*m+1)
fit <- cglasso(S=f_j_hat, nobs=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,
  nobs,
  lambda,
  nlambda,
  lambda.min.ratio,
  W.init,
  stopping_rule,
  stop_criterion,
  maxit = maxit,
  thresh = thresh,
  trace.it = 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 with spectral densities as the diagonal entries. Default is NULL. If D is not provided, diagonals of S are chosen.
type	Logical flag to choose the formulation to solve. Default is I. If type is I, the algorithm solves CGLASSO-I in the reference, $$D^{-1/2}(\arg\min_{\Theta} Tr[\hat{R}\hat{\Theta}] - \log\det\Theta + \sum_{i\neq j}|\Theta_{ij}|)D^{-1/2}$$ 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 scale matrix D is multiplied. Please see the reference for the details of the iterative updates.
nobs	Number of observations used in computation of the spectral density matrix S. 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). The default depends on the sample size nobs relative to the number of variables nvar. If nobs > p, the default is 0.0001, close to zero. If nobs < p, the default is 0.01.
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".

a0	Intercept sequence of length length(lambda).
beta	A nvar 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	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=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.
npasses	Total passes over the data summed over all lambda values.
jerr	Error flag, for warnings and errors (largely for internal debugging).
call	The call that produced this object.
family	Family used for the model.
nobs	Number of observations.

#' @return 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_hat	Estimated inverse spectral matrix for each fixed lambda. It is provided in the list.
type	Type of the formulation used, either CGALSSO-I or CGLASSO-II.
scale	Whether the spectral density matrix (covariance) or spectral coherence (coherence) is given.
D	Used scale diagonal matrix.

cglasso_example *Example Data for CGLASSO*

Description

A synthetic dataset used to illustrate complex-valued graphical Lasso.

Usage

```
data(cglasso_example)
```

Format

A list with components:

S Spectral density matrix

n Sample size

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`	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=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.
`npasses`	Total passes over the data summed over all lambda values.
`jerr`	Error flag, for warnings and errors (largely for internal debugging).
`call`	The call that produced this object.
`family`	Family used for the model.
`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 = 1e-05,
  devmax = 0.999,
  eps = 1e-06,
  big = 9.9e+35,
  mnlam = 5,
  pmin = 1e-09,
  exmx = 250,
  prec = 1e-10,
  mxit = 100,
  itrace = 0,
  epsnr = 1e-06,
  mxitnr = 25,
  factory = TRUE
)
```

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 the 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 (FALSE)? 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.

`npasses` Total passes over the data summed over all lambda values.

`jerr` Error flag, for warnings and errors (largely for internal debugging).

`call` The call that produced this object.

`family` Family used for the model.

`nobs` Number of observations.

classo_example *Example Data for CLASSO*

Description

A toy example demonstrating complex-valued lasso.

Usage

```
data(classo_example)
```

Format

A list with components:

X Design matrix

Y Response vector

coef.classo *Extract coefficients from a classo object*

Description

Similar to other predict methods, this functions 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("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	This argument is relevant only when predictions are made at values of s (lambda) *different* from those used in the fitting of the original model. If exact=FALSE (default), then the predict function uses linear interpolation to make predictions for values of s (lambda) that do not coincide with those used in the fitting algorithm. While this is often a good approximation, it can sometimes be a bit coarse. With exact=TRUE, these different values of s are merged (and sorted) with object$lambda, and the model is refit before predictions are made. In this case, it is required to supply the original data x= and y= as additional named arguments to predict() or coef(). The workhorse predict.classo() needs to update the model, and so needs the data used to create it. The same is true of weights if these were used in the original call. Failure to do so will result in an error. type="nonzero")
...	This is the mechanism for passing arguments like x= when exact=TRUE; seeexact argument.
newx	Matrix of new values for x at which predictions are to be made. Must be a matrix. This #' argument is not used for type=c("coefficients","nonzero")
type	Type of prediction required. Type "link" givee the fitted values for "gaussian". Type "response" is equivalent to type "link". Type "coefficients" computes the coefficients at the requested values for s. Type "nonzero" returns a list of the indices of the nonzero coefficients for each value of s.

Details

This function actually calls NextMethod(). coef(...) is equivalent to predict(type="coefficients",...)

Value

The object returned depends on type.

Author(s)

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

References

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

See Also

classo, and print, and coef methods, and 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 parallel before hand, such as doMC or others. Currently it is unavailable.

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 <yk748@cornell.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$f_hat
n <- cglasso_example$n
cglasso(S=f_hat,type="I",nobs=n)
cglasso(S=f_hat,type="II",nobs=n)
```

dft.X — *Discrete Fourier Transform of matrix X*

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.X(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) × nvar` representing selected DFT components of the original matrix.

fixm — *Fixes indices that fall outside the valid range* `1:n` *using circular (modulo) wrapping.*

Description

Fixes indices that fall outside the valid range `1:n` using circular (modulo) wrapping.

Usage

```
fixm(v, n)
```

Arguments

`v`	An integer vector of indices.
`n`	A positive integer giving the length of the domain; valid indices are `1` to `nobs`.

Value

An integer vector of the same length as `v`, with all values wrapped into the range `1` to `n`.

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

Description

Produces plot of the estimated inverse spectral matrix for a fitted `"cglasso"` object. A inverse spectral matrix profile plot is produced.

Usage

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

Arguments

`x`	fitted `"cglasso"` model
`index`	For which inverse spectral matrix profile is the plot to be drawn? Default is 1. The index must be provided within the length of the sequence of lambdas.
`type`	Whether the plot is for real or imaginary part, or both, or in modulus (mod; absolute scale). Default is `mod`.
`label`	If `TRUE`, label the axes with variable names.
`...`	Other graphical parameters to plot.

Value

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

Author(s)

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

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 matrix
s	Value(s) of the penalty parameter lambda at which predictions are required. Default is the value s="lambda.1se" stored on the CV object. Alternatively s="lambda.min" can be used. If s is numeric, it is taken as the value(s) of lambda to be used. (For historical reasons we use the symbol 's' rather than 'lambda' to reference this parameter)
...	Not used. Other arguments to predict.

Details

This function makes it easier to use the results of cross-validation to make a prediction.

Value

The object returned depends on the ... argument which is passed on to the predict method for classo objects.

Author(s)

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

See Also

classo, and print, and coef methods, and 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 summary of the cross-validated fit is produced, slightly different for a 'cv.relaxed' object than for a 'cv.classo' object. Note that a 'cv.relaxed' object inherits from class 'cv.classo', so by directly invoking `print.cv.classo(object)` will print the summary as if `relax=TRUE` had not been used.

Value

The matrix above is silently returned

See Also

`classo`, `predict` and `coef` methods.

Index