Package: biglasso (via r-universe)

October 24, 2026

Title Extending Lasso Model Fitting to Big Data

Version 1.7.2

Description Extend lasso and elastic-net model fitting for large data sets that cannot be loaded into memory. Designed to be more memory- and computation-efficient than existing lasso-fitting packages like 'glmnet' and 'ncvreg', thus allowing the user to analyze big data with limited RAM <doi:10.32614/RJ-2021-001>.

License GPL-3

URL https://pbreheny.github.io/biglasso/, https://github.com/pbreheny/biglasso

BugReports https://github.com/pbreheny/biglasso/issues

Depends R (>= 3.2.0), bigmemory (>= 4.5.0), Matrix, ncvreg

Imports Rcpp (>= 0.12.1), methods

Suggests glmnet, knitr, parallel, rmarkdown, survival, tinytest

LinkingTo Rcpp, RcppArmadillo (>= 0.8.600), bigmemory, BH

VignetteBuilder knitr

Encoding UTF-8

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Yaohui Zeng [aut], Chuyi Wang [aut], Tabitha Peter [aut], Patrick Breheny [aut, cre] (ORCID: <https://orcid.org/0000-0002-0650-1119>)

Maintainer Patrick Breheny <patrick-breheny@uiowa.edu>

Repository https://cran.r-universe.dev

Date/Publication 2026-08-25 11:01:05 UTC

RemoteUrl https://github.com/cran/biglasso

RemoteRef HEAD

RemoteSha 2acaa762b680b9c88fa5614a7ba3e0cb9d81b74a

Contents

biglasso 2
biglasso_fit 7
biglasso_path 10
colon 12
cv.biglasso 13
plot.biglasso 16
plot.cv.biglasso 16
plot.mbiglasso 17
predict.biglasso 18
predict.cv.biglasso 20
setupX 22
summary.cv.biglasso 23

Index **25**

`biglasso` *Fit lasso penalized regression path for big data*

Description

Extend lasso model fitting to big data that cannot be loaded into memory. Fit solution paths for linear, logistic or Cox regression models penalized by lasso, ridge, or elastic-net over a grid of values for the regularization parameter lambda.

Usage

```
biglasso(
  X,
  y,
  row.idx = 1:nrow(X),
  penalty = c("lasso", "ridge", "enet"),
  family = c("gaussian", "binomial", "cox", "mgaussian"),
  alg.logistic = c("Newton", "MM"),
  screen = c("Adaptive", "SSR", "Hybrid", "None"),
  safe.thresh = 0,
  update.thresh = 1,
  ncores = 1,
  alpha = 1,
  lambda.min = ifelse(nrow(X) > ncol(X), 0.001, 0.05),
  nlambda = 100,
  lambda.log.scale = TRUE,
  lambda,
  eps = 1e-07,
  max.iter = 1000,
  dfmax = ncol(X) + 1,
  penalty.factor = rep(1, ncol(X)),
```

```
  warn = TRUE,
  output.time = FALSE,
  return.time = TRUE,
  verbose = FALSE
)
```

Arguments

X	The design matrix, without an intercept. It must be a double type bigmemory::big.matrix() object. The function standardizes the data and includes an intercept internally by default during the model fitting.
y	The response vector for family="gaussian" or family="binomial". For family="cox", y should be a two-column matrix with columns 'time' and 'status'. The latter is a binary variable, with '1' indicating death, and '0' indicating right censored. For family="mgaussin", y should be a n*m matrix where n is the sample size and m is the number of responses.
row.idx	The integer vector of row indices of X that used for fitting the model. 1:nrow(X) by default.
penalty	The penalty to be applied to the model. Either "lasso" (the default), "ridge", or "enet" (elastic net).
family	Either "gaussian", "binomial", "cox" or "mgaussian" depending on the response.
alg.logistic	The algorithm used in logistic regression. If "Newton" then the exact hessian is used (default); if "MM" then a majorization-minimization algorithm is used to set an upper-bound on the hessian matrix. This can be faster, particularly in data-larger-than-RAM case.
screen	The feature screening rule used at each lambda that discards features to speed up computation: "SSR" (default if penalty="ridge" or penalty="enet") is the sequential strong rule; "Hybrid", which combines the strong rule with a safe rule. "Adaptive" (default for penalty="lasso" without penalty.factor), which is a variant of Hybrid that avoids recalculating the safe rule when it would not be efficient to do so. Valid values are family-dependent: "Hybrid" is only available for family="gaussian"; for family="cox", "Hybrid" is not available but three additional cox-specific safe-rule variants are: "scox", "sscox", and "safe". **Note that:** 1. For linear regression with elastic net penalty, both "SSR" and "Hybrid" are applicable since version 1.3-0 2. Only "SSR" is applicable to elastic-net-penalized logistic regression or cox regression 3. Active set cycling is incorporated into all of these screening rules
safe.thresh	the threshold value between 0 and 1 that controls when to stop safe test. For example, 0.01 means to stop safe test at next lambda iteration if the number of features rejected by safe test at current lambda iteration is not larger than 1% of the total number of features. So 1 means to always turn off safe test, whereas 0 (default) means to turn off safe test if the number of features rejected by safe test is 0 at current lambda.

`update.thresh` the non negative threshold value that controls how often to update the reference of safe rules for "Adaptive" methods. Smaller value means updating more often.

`ncores` The number of OpenMP threads used for parallel computing.

`alpha` The elastic-net mixing parameter that controls the relative contribution from the lasso (l1) and the ridge (l2) penalty. The penalty is defined as

$$\alpha||\beta||_1 + (1-\alpha)/2||\beta||_2^2.$$

`alpha=1` is the lasso penalty, `alpha=0` the ridge penalty, `alpha` in between 0 and 1 is the elastic-net ("enet") penalty.

`lambda.min` The smallest value for lambda, as a fraction of lambda.max. Default is 0.001 if the number of observations is larger than the number of covariates and 0.05 otherwise.

`nlambda` The number of lambda values. Default is 100.

`lambda.log.scale`
Whether compute the grid values of lambda on log scale (default) or linear scale.

`lambda` A user-specified sequence of lambda values. By default, a sequence of values of length `nlambda` is computed, equally spaced on the log scale.

`eps` Convergence threshold for inner coordinate descent. The algorithm iterates until the maximum change in the objective after any coefficient update is less than `eps` times the null deviance. Default value is `1e-7`.

`max.iter` Maximum number of iterations. Default is 1000.

`dfmax` Upper bound for the number of nonzero coefficients. Default is no upper bound. However, for large data sets, computational burden may be heavy for models with a large number of nonzero coefficients.

`penalty.factor` A multiplicative factor for the penalty applied to each coefficient. If supplied, `penalty.factor` must be a numeric vector of length equal to the number of columns of `X`. The purpose of `penalty.factor` is to apply differential penalization if some coefficients are thought to be more likely than others to be in the model. Currently, the package doesn't allow unpenalized coefficients (`penalty.factor` cannot be 0). `penalty.factor` is only supported for "SSR" screen.

`warn` Return warning messages for failures to converge and model saturation? Default is TRUE.

`output.time` Whether to print out the start and end time of the model fitting. Default is FALSE.

`return.time` Whether to return the computing time of the model fitting. Default is TRUE.

`verbose` Whether to output the timing of each lambda iteration. Default is FALSE.

Details

The objective function for linear regression or multiple responses linear regression (`family = "gaussian"` or `family = "mgaussian"`) is

$$\frac{1}{2n}\text{RSS} + \lambda \cdot \text{penalty},$$

where for family = "mgaussian"), a group-lasso type penalty is applied. For logistic regression (family = "binomial") it is

$$-\frac{1}{n}\ell + \lambda \cdot \text{penalty},$$

for cox regression, Breslow approximation for ties is applied.

Several advanced feature screening rules are implemented. For lasso-penalized linear regression, all the options of screen are applicable. Our proposal adaptive rule - "Adaptive" - achieves highest speedup so it's the recommended one, especially for ultrahigh-dimensional large-scale data sets. For cox regression and/or the elastic net penalty, only "SSR" is applicable for now. More efficient rules are under development.

Value

An object with S3 class "biglasso" for "gaussian", "binomial", "cox" families, or an object with S3 class "mbiglasso" for "mgaussian" family, with following variables.

beta The fitted matrix of coefficients, store in sparse matrix representation. The number of rows is equal to the number of coefficients, whereas the number of columns is equal to nlambda. For "mgaussian" family with m responses, it is a list of m such matrices.

iter A vector of length nlambda containing the number of iterations until convergence at each value of lambda.

lambda The sequence of regularization parameter values in the path.

penalty Same as above.

family Same as above.

alpha Same as above.

loss A vector containing either the residual sum of squares (for "gaussian", "mgaussian") or negative log-likelihood (for "binomial", "cox") of the fitted model at each value of lambda.

penalty.factor Same as above.

n The number of observations used in the model fitting. It's equal to length(row.idx).

center The sample mean vector of the variables, i.e., column mean of the sub-matrix of X used for model fitting.

scale The sample standard deviation of the variables, i.e., column standard deviation of the sub-matrix of X used for model fitting.

y The response vector used in the model fitting. Depending on row.idx, it could be a subset of the raw input of the response vector y.

screen Same as above.

col.idx The indices of features that have 'scale' value greater than 1e-6. Features with 'scale' less than 1e-6 are removed from model fitting.

rejections The number of features rejected at each value of lambda.

safe_rejections The number of features rejected by safe rules at each value of lambda.

References

Zeng Y and Breheny P. (2021) The biglasso Package: A Memory- and Computation- Efficient Solver for Lasso Model Fitting with Big Data in R. *R Journal*, 12: 6-19. doi:10.32614/RJ2021001

See Also

`setupX()`, `cv.biglasso()`, `plot.biglasso()`, `ncvreg::ncvreg()`

Examples

```
## Linear regression
data(colon)
X <- colon$X
y <- colon$y
x_bm <- as.big.matrix(X)
# lasso, default
fit_lasso <- biglasso(x_bm, y, family = "gaussian")
plot(fit_lasso, log.l = TRUE, main = "lasso")
# elastic net
fit_enet <- biglasso(
  x_bm,
  y,
  penalty = "enet",
  alpha = 0.5,
  family = "gaussian"
)
plot(fit_enet, log.l = TRUE, main = "elastic net")

## Logistic regression
fit_bin_lasso <- biglasso(x_bm, y, penalty = "lasso", family = "binomial")
plot(fit_bin_lasso, log.l = TRUE, main = "lasso")
# elastic net
fit_bin_enet <- biglasso(
  x_bm,
  y,
  penalty = "enet",
  alpha = 0.5,
  family = "binomial"
)
plot(fit_bin_enet, log.l = TRUE, main = "elastic net")

## Cox regression
set.seed(10101)
n <- 1000
p <- 30
nzc <- p / 3
x <- matrix(rnorm(n * p), n, p)
beta <- rnorm(nzc)
fx <- x[, seq(nzc)] %*% beta / 3
hx <- exp(fx)
ty <- rexp(n, hx)
tcens <- quantile(ty, 0.7)
```

```
y <- cbind(time = pmin(ty, tcens), status = ty < tcens)
x_bm <- as.big.matrix(x)
fit <- biglasso(x_bm, y, family = "cox")
plot(fit, main = "cox")

## Multiple responses linear regression
set.seed(10101)
n <- 300
p <- 300
m <- 5
s <- 10
b <- 1
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(seq(from = -b, to = b, length.out = s * m), s, m)
y <- x[, 1:s] %*% beta + matrix(rnorm(n * m, 0, 1), n, m)
x_bm <- as.big.matrix(x)
fit <- biglasso(x_bm, y, family = "mgaussian")
plot(fit, main = "mgaussian")
```

`biglasso_fit` *Direct interface to biglasso fitting, no preprocessing*

Description

This function is intended for users who know exactly what they're doing and want complete control over the fitting process. It

- does NOT add an intercept
- does NOT standardize the design matrix
- does NOT set up a path for lambda (the lasso tuning parameter) all of the above are critical steps in data analysis. However, a direct API has been provided for use in situations where the lasso fitting process is an internal component of a more complicated algorithm and standardization must be handled externally.

Usage

```
biglasso_fit(
  X,
  y,
  r,
  init = rep(0, ncol(X)),
  xtx,
  penalty = "lasso",
  lambda,
  alpha = 1,
  gamma,
  ncores = 1,
```

```
  max.iter = 1000,
  eps = 1e-05,
  dfmax = ncol(X) + 1,
  penalty.factor = rep(1, ncol(X)),
  warn = TRUE,
  output.time = FALSE,
  return.time = TRUE
)
```

Arguments

X	The design matrix, without an intercept. It must be a double type bigmemory::big.matrix() object.
y	The response vector
r	Residuals (length n vector) corresponding to init. WARNING: If you supply an incorrect value of r, the solution will be incorrect.
init	Initial values for beta. Default: zero (length p vector)
xtx	X scales: the jth element should equal crossprod(X[,j])/n. In particular, if X is standardized, one should pass xtx = rep(1, p). WARNING: If you supply an incorrect value of xtx, the solution will be incorrect. (length p vector)
penalty	String specifying which penalty to use. Default is 'lasso', Other options are 'SCAD' and 'MCP' (the latter are non-convex)
lambda	A single value for the lasso tuning parameter.
alpha	The elastic-net mixing parameter that controls the relative contribution from the lasso (l1) and the ridge (l2) penalty. The penalty is defined as: \deqn{ \alpha||\beta||_1 + (1-\alpha)/2||\beta||_2^2.} alpha=1 is the lasso penalty, alpha=0 the ridge penalty, alpha in between 0 and 1 is the elastic-net ("enet") penalty.
gamma	Tuning parameter value for nonconvex penalty. Defaults are 3.7 for penalty = 'SCAD' and 3 for penalty = 'MCP'
ncores	The number of OpenMP threads used for parallel computing.
max.iter	Maximum number of iterations. Default is 1000.
eps	Convergence threshold for inner coordinate descent. The algorithm iterates until the maximum change in the objective after any coefficient update is less than eps times the null deviance. Default value is 1e-7.
dfmax	Upper bound for the number of nonzero coefficients. Default is no upper bound. However, for large data sets, computational burden may be heavy for models with a large number of nonzero coefficients.
penalty.factor	A multiplicative factor for the penalty applied to each coefficient. If supplied, penalty.factor must be a numeric vector of length equal to the number of columns of X.
warn	Return warning messages for failures to converge and model saturation? Default is TRUE.

`output.time` Whether to print out the start and end time of the model fitting. Default is FALSE.

`return.time` Whether to return the computing time of the model fitting. Default is TRUE.

Details

Note:

- Hybrid safe-strong rules are turned off for `biglasso_fit()`, as these rely on standardization
- Currently, the function only works with linear regression (`family = 'gaussian'`).

Value

An object with S3 class `"biglasso"` with following variables.

beta The vector of estimated coefficients

iter A vector of length `nlambda` containing the number of iterations until convergence

resid Vector of residuals calculated from estimated coefficients.

lambda The sequence of regularization parameter values in the path.

alpha Same as in `biglasso()`

loss A vector containing either the residual sum of squares of the fitted model at each value of lambda.

penalty.factor Same as in `biglasso()`.

n The number of observations used in the model fitting.

y The response vector used in the model fitting.

Examples

```
data(Prostate)
X <- cbind(1, Prostate$X)
xtx <- apply(X, 2, crossprod) / nrow(X)
y <- Prostate$y
X.bm <- as.big.matrix(X)
init <- rep(0, ncol(X))
fit <- biglasso_fit(
  X = X.bm, y = y, r = y, init = init, xtx = xtx,
  lambda = 0.1, penalty.factor = c(0, rep(1, ncol(X) - 1)), max.iter = 10000
)
fit$beta

fit <- biglasso_fit(
  X = X.bm, y = y, r = y, init = init, xtx = xtx, penalty = "MCP",
  lambda = 0.1, penalty.factor = c(0, rep(1, ncol(X) - 1)), max.iter = 10000
)
fit$beta
```

`biglasso_path` *Direct interface to biglasso fitting, no preprocessing, path version*

Description

This function is intended for users who know exactly what they're doing and want complete control over the fitting process. It

- does NOT add an intercept
- does NOT standardize the design matrix both of the above are critical steps in data analysis. However, a direct API has been provided for use in situations where the lasso fitting process is an internal component of a more complicated algorithm and standardization must be handled externally.

Usage

```
biglasso_path(
  X,
  y,
  r,
  init = rep(0, ncol(X)),
  xtx,
  penalty = "lasso",
  lambda,
  alpha = 1,
  gamma,
  ncores = 1,
  max.iter = 1000,
  eps = 1e-05,
  dfmax = ncol(X) + 1,
  penalty.factor = rep(1, ncol(X)),
  warn = TRUE,
  output.time = FALSE,
  return.time = TRUE
)
```

Arguments

`X`	The design matrix, without an intercept. It must be a double type `bigmemory::big.matrix()` object.
`y`	The response vector
`r`	Residuals (length n vector) corresponding to `init`. WARNING: If you supply an incorrect value of `r`, the solution will be incorrect.
`init`	Initial values for beta. Default: zero (length p vector)

xtx	X scales: the jth element should equal crossprod(X[,j])/n. In particular, if X is standardized, one should pass xtx = rep(1, p). WARNING: If you supply an incorrect value of xtx, the solution will be incorrect. (length p vector)
penalty	String specifying which penalty to use. Default is 'lasso', Other options are 'SCAD' and 'MCP' (the latter are non-convex)
lambda	A vector of numeric values the lasso tuning parameter.
alpha	The elastic-net mixing parameter that controls the relative contribution from the lasso (l1) and the ridge (l2) penalty. The penalty is defined as: \deqn{ \alpha||\beta||_1 + (1-\alpha)/2||\beta||_2^2.} alpha=1 is the lasso penalty, alpha=0 the ridge penalty, alpha in between 0 and 1 is the elastic-net ("enet") penalty.
gamma	Tuning parameter value for nonconvex penalty. Defaults are 3.7 for penalty = 'SCAD' and 3 for penalty = 'MCP'
ncores	The number of OpenMP threads used for parallel computing.
max.iter	Maximum number of iterations. Default is 1000.
eps	Convergence threshold for inner coordinate descent. The algorithm iterates until the maximum change in the objective after any coefficient update is less than eps times the null deviance. Default value is 1e-7.
dfmax	Upper bound for the number of nonzero coefficients. Default is no upper bound. However, for large data sets, computational burden may be heavy for models with a large number of nonzero coefficients.
penalty.factor	A multiplicative factor for the penalty applied to each coefficient. If supplied, penalty.factor must be a numeric vector of length equal to the number of columns of X.
warn	Return warning messages for failures to converge and model saturation? Default is TRUE.
output.time	Whether to print out the start and end time of the model fitting. Default is FALSE.
return.time	Whether to return the computing time of the model fitting. Default is TRUE.

Details

biglasso_path() works identically to biglasso_fit() except it offers the additional option of fitting models across a path of tuning parameter values.

Note:

- Hybrid safe-strong rules are turned off for biglasso_fit(), as these rely on standardization
- Currently, the function only works with linear regression (family = 'gaussian').

Value

An object with S3 class "biglasso" with following variables.

beta A sparse matrix where rows are estimates a given coefficient across all values of lambda

iter A vector of length `nlambda` containing the number of iterations until convergence

resid Vector of residuals calculated from estimated coefficients.

lambda The sequence of regularization parameter values in the path.

alpha Same as in `biglasso()`

loss A vector containing either the residual sum of squares of the fitted model at each value of lambda.

penalty.factor Same as in `biglasso()`.

n The number of observations used in the model fitting.

y The response vector used in the model fitting.

Examples

```
data(Prostate)
X <- cbind(1, Prostate$X)
xtx <- apply(X, 2, crossprod) / nrow(X)
y <- Prostate$y
X.bm <- as.big.matrix(X)
init <- rep(0, ncol(X))
fit <- biglasso_path(
  X = X.bm, y = y, r = y, init = init, xtx = xtx,
  lambda = c(0.5, 0.1, 0.05, 0.01, 0.001),
  penalty.factor = c(0, rep(1, ncol(X) - 1)), max.iter = 2000
)
fit$beta

fit <- biglasso_path(
  X = X.bm, y = y, r = y, init = init, xtx = xtx,
  lambda = c(0.5, 0.1, 0.05, 0.01, 0.001), penalty = "MCP",
  penalty.factor = c(0, rep(1, ncol(X) - 1)), max.iter = 2000
)
fit$beta
```

---

colon *Gene expression data from colon-cancer patients*

Description

The data file contains gene expression data of 62 samples (40 tumor samples, 22 normal samples) from colon-cancer patients analyzed with an Affymetrix oligonucleotide Hum6000 array.

Usage

```
data(colon)
```

Format

A list of 2 variables:

X A 62-by-2000 matrix that records the gene expression data. Used as design matrix.

y A binary vector of length 62 recording the sample status: 1 = tumor; 0 = normal. Used as response vector.

Source

The raw data can be found on Bioconductor: https://bioconductor.org/packages/release/data/experiment/html/colonCA.html.

References

Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, and Levine AJ (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci.* 96: 6745–6750 doi:10.1073/pnas.96.12.6745

Examples

```
data(colon)
X <- colon$X
y <- colon$y
str(X)
dim(X)
X.bm <- as.big.matrix(X, backingfile = "") # convert to big.matrix object
str(X.bm)
dim(X.bm)
```

---

cv.biglasso *Cross-validation for biglasso*

Description

Perform k-fold cross validation for penalized regression models over a grid of values for the regularization parameter lambda.

Usage

```
cv.biglasso(
  X,
  y,
  row.idx = 1:nrow(X),
  family = c("gaussian", "binomial", "cox", "mgaussian"),
  eval.metric = c("default", "MAPE", "auc", "class"),
  ncores = parallel::detectCores(),
  ...,
  nfolds = 5,
```

```
  seed,
  cv.ind,
  trace = FALSE,
  grouped = TRUE
)
```

Arguments

X	The design matrix, without an intercept, as in biglasso().
y	The response vector, as in biglasso.
row.idx	The integer vector of row indices of X that used for fitting the model. as in biglasso.
family	Either "gaussian", "binomial", "cox" or "mgaussian" depending on the response. "cox" and "mgaussian" are not supported yet.
eval.metric	The evaluation metric for the cross-validated error and for choosing optimal lambda. "default" for linear regression is MSE (mean squared error), for logistic regression is binomial deviance. "MAPE", for linear regression only, is the Mean Absolute Percentage Error. "auc", for binary classification, is the area under the receiver operating characteristic curve (ROC). "class", for binary classification, gives the misclassification error.
ncores	The number of cores to use when fitting the cross-validation folds in parallel. Two subtleties worth noting: • X must be file-backed in order to take advantage of this option (fold workers are separate processes that reattach X from its backing file). • ncores is also passed to biglasso() for the initial, full-data fit ().
...	Additional arguments to biglasso.
nfolds	The number of cross-validation folds. Default is 5.
seed	The seed of the random number generator in order to obtain reproducible results.
cv.ind	Which fold each observation belongs to. By default the observations are randomly assigned by cv.biglasso.
trace	If set to TRUE, cv.biglasso will inform the user of its progress by announcing the beginning of each CV fold. Default is FALSE.
grouped	Whether to calculate CV standard error (cvse) over CV folds (TRUE), or over all cross-validated predictions. Ignored when eval.metric is 'auc'.

Details

The function calls biglasso nfolds times, each time leaving out 1/nfolds of the data. The cross-validation error is based on the residual sum of squares when family="gaussian" and the binomial deviance when family="binomial".

The S3 class object cv.biglasso inherits class ncvreg::cv.ncvreg(). So S3 functions such as "summary", "plot" can be directly applied to the cv.biglasso object.

Value

An object with S3 class "cv.biglasso" which inherits from class "cv.ncvreg". The following variables are contained in the class:

cve The error for each value of lambda, averaged across the cross-validation folds.

cvse The estimated standard error associated with each value of for cve.

lambda The sequence of regularization parameter values along which the cross-validation error was calculated.

fit The fitted biglasso object for the whole data.

min The index of lambda corresponding to lambda.min.

lambda.min The value of lambda with the minimum cross-validation error.

lambda.1se The largest value of lambda for which the cross-validation error is at most one standard error larger than the minimum cross-validation error.

null.dev The deviance for the intercept-only model.

pe If family="binomial", the cross-validation prediction error for each value of lambda.

cv.ind Same as above.

See Also

biglasso(), plot.cv.biglasso(), summary.cv.biglasso(), setupX()

Examples

```
## Not run:
## cv.biglasso
data(colon)
X <- colon$X
y <- colon$y
X.bm <- as.big.matrix(X)

## logistic regression
cvfit <- cv.biglasso(X.bm, y, family = "binomial", seed = 1234, ncores = 2)
par(mfrow = c(2, 2))
plot(cvfit, type = "all")
summary(cvfit)

## End(Not run)
```

plot.biglasso *Plot coefficients from a "biglasso" object*

Description

Produce a plot of the coefficient paths for a fitted `biglasso()` object.

Usage

```
## S3 method for class 'biglasso'
plot(x, alpha = 1, log.l = TRUE, ...)
```

Arguments

`x`	Fitted `biglasso()` model.
`alpha`	Controls alpha-blending, helpful when the number of covariates is large. Default is alpha=1.
`log.l`	Should horizontal axis be on the log scale? Default is TRUE.
`...`	Other graphical parameters to `plot()`

Value

Invisibly returns `NULL`. Called for its side effect of producing a plot.

See Also

`biglasso()`, `cv.biglasso()`

plot.cv.biglasso *Plots the cross-validation curve from a "cv.biglasso" object*

Description

Plot the cross-validation curve from a `cv.biglasso()` object, along with standard error bars.

Usage

```
## S3 method for class 'cv.biglasso'
plot(
  x,
  log.l = TRUE,
  type = c("cve", "rsq", "scale", "snr", "pred", "all"),
  selected = TRUE,
  vertical.line = TRUE,
  col = "red",
  ...
)
```

Arguments

x	A "cv.biglasso" object.
log.l	Should horizontal axis be on the log scale? Default is TRUE.
type	What to plot on the vertical axis. cve plots the cross-validation error (deviance); rsq plots an estimate of the fraction of the deviance explained by the model (R-squared); snr plots an estimate of the signal-to-noise ratio; scale plots, for family="gaussian", an estimate of the scale parameter (standard deviation); pred plots, for family="binomial", the estimated prediction error; all produces all of the above.
selected	If TRUE (the default), places an axis on top of the plot denoting the number of variables in the model (i.e., that have a nonzero regression coefficient) at that value of lambda.
vertical.line	If TRUE (the default), draws a vertical line at the value where cross-validaton error is minimized.
col	Controls the color of the dots (CV estimates).
...	Other graphical parameters to plot

Details

Error bars representing approximate 68\ estimates at value of lambda. For rsq and snr, these confidence intervals are quite crude, especially near.

Value

Invisibly returns NULL. Called for its side effect of producing a plot.

See Also

biglasso(), cv.biglasso()

---

plot.mbiglasso *Plot coefficients from a "mbiglasso" object*

Description

Produce a plot of the coefficient paths for a fitted multiple responses mbiglasso object.

Usage

```
## S3 method for class 'mbiglasso'
plot(x, alpha = 1, log.l = TRUE, norm.beta = TRUE, ...)
```

Arguments

x	Fitted mbiglasso model.
alpha	Controls alpha-blending, helpful when the number of covariates is large. Default is alpha=1.
log.l	Should horizontal axis be on the log scale? Default is TRUE.
norm.beta	Should the vertical axis be the l2 norm of coefficients for each variable? Default is TRUE. If False, the vertical axis is the coefficients.
...	Other graphical parameters to plot()

Value

Invisibly returns NULL. Called for its side effect of producing a plot.

See Also

biglasso()

---

predict.biglasso *Model predictions based on a fitted* biglasso *object*

Description

Extract predictions (fitted reponse, coefficients, etc.) from a fitted biglasso() object.

Usage

```
## S3 method for class 'biglasso'
predict(
  object,
  X,
  row.idx = 1:nrow(X),
  type = c("link", "response", "class", "coefficients", "vars", "nvars"),
  lambda = NULL,
  which = 1:length(object$lambda),
  ...
)

## S3 method for class 'mbiglasso'
predict(
  object,
  X,
  row.idx = 1:nrow(X),
  type = c("link", "response", "coefficients", "vars", "nvars"),
  lambda = NULL,
  which = 1:length(object$lambda),
  k = 1,
```

```
  ...
)

## S3 method for class 'biglasso'
coef(object, lambda = NULL, which = 1:length(object$lambda), drop = TRUE, ...)

## S3 method for class 'mbiglasso'
coef(
  object,
  lambda = NULL,
  which = 1:length(object$lambda),
  intercept = TRUE,
  ...
)
```

Arguments

`object`	A fitted `"biglasso"` model object.
`X`	Matrix of values at which predictions are to be made. It must be a `bigmemory::big.matrix()` object. Not used for `type="coefficients"`.
`row.idx`	Similar to that in `biglasso()`, it's a vector of the row indices of `X` that used for the prediction. `1:nrow(X)` by default.
`type`	Type of prediction: • `"link"` returns the linear predictors • `"response"` gives the fitted values • `"class"` returns the binomial outcome with the highest probability • `"coefficients"` returns the coefficients • `"vars"` returns a list containing the indices and names of the nonzero variables at each value of `lambda` • `"nvars"` returns the number of nonzero coefficients at each value of `lambda`
`lambda`	Values of the regularization parameter `lambda` at which predictions are requested. Linear interpolation is used for values of `lambda` not in the sequence of lambda values in the fitted models.
`which`	Indices of the penalty parameter `lambda` at which predictions are required. By default, all indices are returned. If `lambda` is specified, this will override `which`.
`...`	Not used.
`k`	Index of the response to predict in multiple responses regression (`family="mgaussian"`).
`drop`	If coefficients for a single value of `lambda` are to be returned, reduce dimensions to a vector? Setting `drop=FALSE` returns a 1-column matrix.
`intercept`	Whether the intercept should be included in the returned coefficients. For `family="mgaussian"` only.

Value

The object returned depends on `type`.

See Also

biglasso(), cv.biglasso()

Examples

```
## Logistic regression
data(colon)
x <- colon$X
y <- colon$y
x_bm <- as.big.matrix(x, backingfile = "")
fit <- biglasso(x_bm, y, penalty = "lasso", family = "binomial")
coef <- coef(fit, lambda = 0.05, drop = TRUE)
coef[which(coef != 0)]
predict(fit, x_bm, type = "link", lambda = 0.05)[1:10]
predict(fit, x_bm, type = "response", lambda = 0.05)[1:10]
predict(fit, x_bm, type = "class", lambda = 0.1)[1:10]
predict(fit, type = "vars", lambda = c(0.05, 0.1))
predict(fit, type = "nvars", lambda = c(0.05, 0.1))
```

---

predict.cv.biglasso *Model predictions based on a fitted* cv.biglasso() *object*

Description

Extract predictions from a fitted cv.biglasso() object.

Usage

```
## S3 method for class 'cv.biglasso'
predict(
  object,
  X,
  row.idx = 1:nrow(X),
  type = c("link", "response", "class", "coefficients", "vars", "nvars"),
  lambda = object$lambda.min,
  which = object$min,
  ...
)

## S3 method for class 'cv.biglasso'
coef(object, lambda = object$lambda.min, which = object$min, ...)
```

Arguments

object	A fitted "cv.biglasso" model object.
X	Matrix of values at which predictions are to be made. It must be a bigmemory::big.matrix() object. Not used for type="coefficients".

row.idx	Similar to that in biglasso(), it's a vector of the row indices of X that used for the prediction. 1:nrow(X) by default.
type	Type of prediction: • "link" returns the linear predictors • "response" gives the fitted values • "class" returns the binomial outcome with the highest probability • "coefficients" returns the coefficients • "vars" returns a list containing the indices and names of the nonzero variables at each value of lambda • "nvars" returns the number of nonzero coefficients at each value of lambda
lambda	Values of the regularization parameter lambda at which predictions are requested. The default value is the one corresponding to the minimum cross-validation error. Accepted values are also the strings "lambda.min" (lambda of minimum cross-validation error) and "lambda.1se" (Largest value of lambda for which the cross-validation error was at most one standard error larger than the minimum.).
which	Indices of the penalty parameter lambda at which predictions are requested. The default value is the index of lambda corresponding to lambda.min. Note: this is overridden if lambda is specified.
...	Not used.

Value

The object returned depends on type.

See Also

biglasso(), cv.biglasso()

Examples

```
## Not run:
## predict.cv.biglasso
data(colon)
X <- colon$X
y <- colon$y
X.bm <- as.big.matrix(X, backingfile = "")
fit <- biglasso(X.bm, y, penalty = "lasso", family = "binomial")
cvfit <- cv.biglasso(
  X.bm,
  y,
  penalty = "lasso",
  family = "binomial",
  seed = 1234,
  ncores = 2
)
coef <- coef(cvfit)
coef[which(coef != 0)]
predict(cvfit, X.bm, type = "response")
```

```
predict(cvfit, X.bm, type = "link")
predict(cvfit, X.bm, type = "class")
predict(cvfit, X.bm, lambda = "lambda.1se")

## End(Not run)
```

---

setupX *Set up design matrix X by reading data from big data file*

Description

Set up the design matrix X as a `big.matrix` object based on external massive data file stored on disk that cannot be fullly loaded into memory. The data file must be a well-formated ASCII-file, and contains only one single type. Current version only supports `double` type. This function reads the massive data, and creates a `big.matrix` object. By default, the resulting `big.matrix` is file-backed, and can be shared across processors or nodes of a cluster.

Usage

```
setupX(
  filename,
  dir = getwd(),
  sep = ",",
  backingfile = paste0(unlist(strsplit(filename, split = "\\."))[1], ".bin"),
 descriptorfile = paste0(unlist(strsplit(filename, split = "\\."))[1], ".desc"),
  type = "double",
  ...
)
```

Arguments

`filename`	The name of the data file. For example, "dat.txt".
`dir`	The directory used to store the binary and descriptor files associated with the `big.matrix`. The default is current working directory.
`sep`	The field separator character. For example, "," for comma-delimited files (the default); "\t" for tab-delimited files.
`backingfile`	The binary file associated with the file-backed `big.matrix`. By default, its name is the same as `filename` with the extension replaced by ".bin".
`descriptorfile`	The descriptor file used for the description of the file-backed `big.matrix`. By default, its name is the same as `filename` with the extension replaced by ".desc".
`type`	The data type. Only "double" is supported for now.
`...`	Additional arguments that can be passed into function `bigmemory::read.big.matrix()`.

Details

For a data set, this function needs to be called only one time to set up the `big.matrix` object with two backing files (.bin, .desc) created in current working directory. Once set up, the data can be "loaded" into any (new) R session by calling `attach.big.matrix(discriptorfile)`.

This function is a simple wrapper of `bigmemory::read.big.matrix()`. See bigmemory for more details.

Value

A `big.matrix` object corresponding to a file-backed `bigmemory::big.matrix()` and ready to be used as the design matrix `X` in `biglasso()` and `cv.biglasso()`.

See Also

`biglasso()`, `ncvreg::cv.ncvreg()`

---

summary.cv.biglasso *Summarizing inferences based on cross-validation*

Description

Summary method for `cv.biglasso` objects.

Usage

```
## S3 method for class 'cv.biglasso'
summary(object, ...)

## S3 method for class 'summary.cv.biglasso'
print(x, digits, ...)
```

Arguments

`object`	A `cv.biglasso` object.
`...`	Further arguments passed to or from other methods.
`x`	A `"summary.cv.biglasso"` object.
`digits`	Number of digits past the decimal point to print out. Can be a vector specifying different display digits for each of the five non-integer printed values.

Value

`summary.cv.biglasso` produces an object with S3 class `"summary.cv.biglasso"`. The class has its own print method and contains the following list elements:

penalty The penalty used by `biglasso`.

model Either `"linear"` or `"logistic"`, depending on the `family` option in `biglasso`.

n Number of observations

p Number of regression coefficients (not including the intercept).

min The index of `lambda` with the smallest cross-validation error.

lambda The sequence of `lambda` values used by `cv.biglasso`.

cve Cross-validation error (deviance).

r.squared Proportion of variance explained by the model, as estimated by cross-validation.

snr Signal to noise ratio, as estimated by cross-validation.

sigma For linear regression models, the scale parameter estimate.

pe For logistic regression models, the prediction error (misclassification error).

See Also

`biglasso()`, `cv.biglasso()`, `plot.cv.biglasso()`

Index

∗ **datasets**
colon, 12

biglasso, 2
biglasso(), *14–21*, *23*, *24*
biglasso_fit, 7
biglasso_fit(), *11*
biglasso_path, 10
bigmemory::big.matrix(), *3*, *8*, *10*, *19*, *20*, *23*
bigmemory::read.big.matrix(), *22*, *23*

coef.biglasso *(*predict.biglasso*)*, 18
coef.cv.biglasso *(*predict.cv.biglasso*)*, 20
coef.mbiglasso *(*predict.biglasso*)*, 18
colon, 12
cv.biglasso, 13
cv.biglasso(), *6*, *16*, *17*, *20*, *21*, *23*, *24*

ncvreg::cv.ncvreg(), *14*, *23*
ncvreg::ncvreg(), *6*

plot(), *16*, *18*
plot.biglasso, 16
plot.biglasso(), *6*
plot.cv.biglasso, 16
plot.cv.biglasso(), *15*, *24*
plot.mbiglasso, 17
predict.biglasso, 18
predict.cv.biglasso, 20
predict.mbiglasso *(*predict.biglasso*)*, 18
print.summary.cv.biglasso *(*summary.cv.biglasso*)*, 23

setupX, 22
setupX(), *6*, *15*
summary.cv.biglasso, 23
summary.cv.biglasso(), *15*