Package ‘ncvreg’

July 22, 2025

Title Regularization Paths for SCAD and MCP Penalized Regression Models

Version 3.15.0

Date 2025-02-11

Suggests ashr, knitr, parallel, rmarkdown, survival, tinytest

VignetteBuilder knitr

Description Fits regularization paths for linear regression, GLM, and Cox regression models using lasso or nonconvex penalties, in particular the minimax concave penalty (MCP) and smoothly clipped absolute deviation (SCAD) penalty, with options for additional L2 penalties (the ``elastic net'' idea). Utilities for carrying out cross-validation as well as post-fitting visualization, summarization, inference, and prediction are also provided. For more information, see Breheny and Huang (2011) <doi:10.1214/10-AOAS388> or visit the ncvreg homepage <https://pbreheny.github.io/ncvreg/>.

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

License GPL-3

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

LazyData TRUE

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Author Patrick Breheny [aut, cre] (ORCID: <https://orcid.org/0000-0002-0650-1119>), Ryan Miller [aut] (ORCID: <https://orcid.org/0000-0003-0446-9992>), Logan Harris [aut]

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

Repository CRAN

Date/Publication 2025-02-11 23:10:02 UTC

Contents

assign_fold 2
AUC.cv.ncvsurv 3
boot_ncvreg 4
cv.ncvreg 7
Heart 9
local_mfdr 11
logLik.ncvreg 12
Lung 13
mfdr 14
ncvfit 15
ncvreg 18
ncvsurv 21
perm.ncvreg 25
permres 27
plot.cv.ncvreg 28
plot.mfdr 30
plot.ncvreg 31
plot.ncvsurv.func 33
predict.cv.ncvreg 34
predict.ncvsurv 36
Prostate 38
residuals.ncvreg 39
std 39
summary.cv.ncvreg 41
summary.ncvreg 42

Index **45**

`assign_fold` *Assign folds for cross-validation*

Description

If y has only two unique values, fold assignments are chosen so that the balance between outcomes is the same in each fold. This is useful for logistic regression and time-to-event data (to balance the fraction of observations that are censored).

Usage

```
assign_fold(y, folds, seed)
```

Arguments

`y`	Either (i) the vector of outcomes or (ii) a vector such as `1:n`
`folds`	Number of folds
`seed`	(optional) set a seed for reproducibility

Value

A vector of integers indicating fold assignments

See Also

[cv.ncvreg()]

Examples

```
assign_fold(rnorm(11), 2)
assign_fold(1:41, 7)
assign_fold(1:41, 7) |> table()
data(Heart)
assign_fold(Heart$y, 7) |> head()
assign_fold(Heart$y, 7) |> table(Heart$y)
```

AUC.cv.ncvsurv *AUC for cv.ncvsurv objects*

Description

Calculates the cross-validated AUC (concordance) from a `cv.ncvsurv` object.

Usage

```
## S3 method for class 'cv.ncvsurv'
AUC(obj, ...)
```

Arguments

`obj`	A `cv.ncvsurv` object. You must run `cv.ncvsurv()` with the option `returnY=TRUE` in order for `AUC()` to work.
`...`	For S3 method compatibility; not used

Details

The area under the curve (AUC), or equivalently, the concordance statistic (C), is calculated according to the procedure described in van Houwelingen and Putter (2011). The function calls `survival::concordancefit()`, except cross-validated linear predictors are used to guard against overfitting. Thus, the values returned by `AUC.cv.ncvsurv()` will be lower than those you would obtain with `concordancefit()` if you fit the full (unpenalized) model.

Author(s)

Patrick Breheny, Brandon Butcher, and Lawrence Hunsicker

References

van Houwelingen H, Putter H (2011). Dynamic Prediction in Clinical Survival Analysis. CRC Press.

See Also

cv.ncvsurv(), survival::concordancefit()

Examples

```
data(Lung)
X <- Lung$X
y <- Lung$y

cvfit <- cv.ncvsurv(X, y, returnY=TRUE)
head(AUC(cvfit))
lam <- cvfit$lambda
plot(lam, AUC(cvfit), xlim=rev(range(lam)), lwd=3, type='l',
     las=1, xlab=expression(lambda), ylab='AUC')
```

boot_ncvreg *Hybrid Bootstrap Confidence Intervals*

Description

Performs a hybrid bootstrapping approach to construct quantile based confidence intervals around the original lasso/MCP/SCAD estimator. Specifically, a traditional pairs bootstrap is performed with 1 adjustment: if the bootstrap sample for a given covariate is zero, a random sample from the full conditional posterior is used as the bootstrap sample instead. This avoids the creation of intervals with endpoints exactly equal to zero.

Usage

```
boot_ncvreg(
  X,
  y,
  fit,
  lambda,
  sigma2,
  cluster,
  seed,
  nboot = 1000,
  penalty = "lasso",
  level = 0.95,
  gamma = switch(penalty, SCAD = 3.7, 3),
  alpha = 1,
  returnCV = FALSE,
  return_boot = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`X` The design matrix, without an intercept. `boot_ncvreg` standardizes the data and includes an intercept by default.

`y` The response vector.

`fit` (optional) An object of class `ncvreg` or `cv.ncvreg`. An object of class `ncvreg` provides data, penalty choices, and `lambda` sequence to `boot_ncvreg`. An object of class `cv.ncvreg` can in addition can provide information for selecting `lambda` and estimating `sigma2`. If provided, `y` should not be provided and `X` should only be provided if `fit` does not contain `X`.

`lambda` (optional) The value of lambda to provide interval estimates for. If left missing will be selected using CV. If user wants to set the lambda sequence used to select `lambda` via cross validation, they should call `cv.ncvreg` separately and pass the resulting object to `fit`.

`sigma2` (optional) The variance to use for the Hybrid sampling. If left missing will be set using the estimator suggested by Reid et. al. (2016) using CV.

`cluster` Bootstrapping and `cv.ncvreg` (if applicable) can be run in parallel across a cluster using the **parallel** package. The cluster must be set up in advance using the `parallel::makeCluster()` function from that package. The cluster must then be passed to `boot_ncvreg`.

`seed` You may set the seed of the random number generator in order to obtain reproducible results. This is set for the overall process. If the user wishes to set a seed specifically for `cv.ncvreg` they should call it separately then pass the fitted object as an argument to `fit`.

`nboot` The number of bootstrap replications to use.

`penalty` The penalty to be applied to the model. Either "lasso" (the default), "MCP", or "SCAD".

`level` The confidence level required.

`gamma` The tuning parameter of the MCP/SCAD penalty (see `ncvreg` for details). Default is 3 for MCP and 3.7 for SCAD. Ignored if fit is provided.

`alpha` Tuning parameter for the Elastc net estimator which controls the relative contributions from the lasso/MCP/SCAD penalty and the ridge, or L2 penalty. `alpha=1` is equivalent to lasso/MCP/SCAD penalty, while `alpha=0` would be equivalent to ridge regression. However, `alpha=0` is not supported; `alpha` may be arbitrarily small, but not exactly 0. Ignored if fit is provided.

`returnCV` If `TRUE`, the `cv.ncvreg` fit will be returned (if applicable).

`return_boot` If `TRUE`, the bootstrap draws will be returned.

`verbose` If `FALSE`, non-essential messages are suppressed.

`...` named arguments to be passed to `ncvreg` and `cv.ncvreg`.

Details

The resulting intervals WILL NOT have exact nominal coverage for all covariates. They are instead constructed in a way that overall coverage will be approximately equal to nominal so long as the true distribution of betas is Laplace and the covariates are independent. That said, in practice, average coverage is fairly robust to these assumptions.

Note: Draws from the full conditional posterior are approximations for MCP/SCAD or when `alpha` is not 1.

Value

A list with:

confidence_intervals A `data.frame` with the original point estimates along with lower and upper bounds of Hybrid CIs.

lambda The value of `lambda` the `confidence_intervals` were constructed at.

sigma2 The value of `sigma2` used for the Hybrid bootstrap sampling.

penalty The penalty the intervals correspond to.

alpha The tuning parameter for the Enet estimator used.

level The confidence level the intervals correspond to.

If a penalty other than "lasso" is used,

gamma The tuning parameter for MCP/SCAD penalty.

If `returnCV` is `TRUE` and a `cv.ncvreg` object was fit or supplied

cv.ncvreg The `cv.ncvreg` fit used to estimate `lambda` and `sigma2` (if applicable).

If `return_boot` is `TRUE`

boot_draws A `data.frame` of the Hybrid bootstrap draws are returned.

Examples

```
data(Prostate)
X <- Prostate$X
y <- Prostate$y
boot_ncvreg(X, y, level = 0.8)
```

cv.ncvreg *Cross-validation for ncvreg/ncvsurv*

Description

Performs k-fold cross validation for MCP- or SCAD-penalized regression models over a grid of values for the regularization parameter lambda.

Usage

```
cv.ncvreg(
  X,
  y,
  ...,
  cluster,
  nfolds = 10,
  fold,
  returnY = FALSE,
  trace = FALSE
)

cv.ncvsurv(
  X,
  y,
  ...,
  cluster,
  nfolds = 10,
  fold,
  se = c("quick", "bootstrap"),
  returnY = FALSE,
  trace = FALSE
)
```

Arguments

`X`	The design matrix, without an intercept, as in `ncvreg()` or `ncvsurv()`.
`y`	The response, as in `ncvreg()` or `ncvsurv()`.
`...`	Additional arguments to `ncvreg()` or `ncvsurv()`.
`cluster`	`cv.ncvreg()` and `cv.ncvsurv()` can be run in parallel across a cluster using the **parallel** package. The cluster must be set up in advance using the `parallel::makeCluster()` function from that package. The cluster must then be passed to `cv.ncvreg()` or `cv.ncvsurv()` (see example).
`nfolds`	The number of cross-validation folds. Default is 10.
`fold`	Which fold each observation belongs to. By default the observations are randomly assigned.

returnY Should cv.ncvreg()/cv.ncvsurv() return the linear predictors from the cross-validation folds? Default is FALSE; if TRUE, this will return a matrix in which the element for row i, column j is the fitted value for observation i from the fold in which observation i was excluded from the fit, at the jth value of lambda. NOTE: For cv.ncvsurv(), the rows of Y are ordered by time on study, and therefore will not correspond to the original order of observations pased to cv.ncvsurv().

trace If set to TRUE, inform the user of progress by announcing the beginning of each CV fold. Default is FALSE.

se For cv.ncvsurv(), the method by which the cross-valiation standard error (CVSE) is calculated. The 'quick' approach is based on a rough approximation, but can be calculated more or less instantly. The 'bootstrap' approach is more accurate, but requires additional computing time.

Details

The function calls ncvreg/ncvsurv nfolds times, each time leaving out 1/nfolds of the data. The cross-validation error is based on the deviance; see here for more details.

For family="binomial" models, the cross-validation fold assignments are balanced across the 0/1 outcomes, so that each fold has the same proportion of 0/1 outcomes (or as close to the same proportion as it is possible to achieve if cases do not divide evenly).

For Cox models, cv.ncvsurv() uses the approach of calculating the full Cox partial likelihood using the cross-validated set of linear predictors. Other approaches to cross-validation for the Cox regression model have been proposed in the literature; the strengths and weaknesses of the various methods for penalized regression in the Cox model are the subject of current research. A simple approximation to the standard error is provided, although an option to bootstrap the standard error (se='bootstrap') is also available.

Value

An object with S3 class cv.ncvreg or cv.ncvsurv containing:

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.

fold The fold assignments for cross-validation for each observation; note that for cv.ncvsurv(), these are in terms of the ordered observations, not the original observations.

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

fit The fitted ncvreg() or ncvsurv() 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.

null.dev The deviance for the intercept-only model. If you have supplied your own lambda sequence, this quantity may not be meaningful.

Bias The estimated bias of the minimum cross-validation error, as in Tibshirani and Tibshirani (2009) doi:10.1214/08AOAS224

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

Y If returnY=TRUE, the matrix of cross-validated fitted values (see above).

Author(s)

Patrick Breheny; Grant Brown helped with the parallelization support

References

Breheny P and Huang J. (2011) Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, **5**: 232-253. doi:10.1214/10AOAS388

See Also

`ncvreg()`, `plot.cv.ncvreg()`, `summary.cv.ncvreg()`

Examples

```
data(Prostate)

cvfit <- cv.ncvreg(Prostate$X, Prostate$y)
plot(cvfit)
summary(cvfit)

fit <- cvfit$fit
plot(fit)
beta <- fit$beta[,cvfit$min]

## requires loading the parallel package
## Not run:
library(parallel)
X <- Prostate$X
y <- Prostate$y
cl <- makeCluster(4)
cvfit <- cv.ncvreg(X, y, cluster=cl, nfolds=length(y))
## End(Not run)

# Survival
data(Lung)
X <- Lung$X
y <- Lung$y

cvfit <- cv.ncvsurv(X, y)
summary(cvfit)
plot(cvfit)
plot(cvfit, type="rsq")
```

Heart *Risk factors associated with heart disease*

Description

Data from a subset of the Coronary Risk-Factor Study baseline survey, carried out in rural South Africa.

Usage

```
Heart
```

Format

A list of two objects: `y` and `X`

y Coronary heart disease at baseline; 1=Yes 0=No

X A matrix with 462 observations (rows) and 9 predictor variables (columns). The remainder of this list describes the columns of `X`

sbp Systolic blood pressure

tobacco Cumulative tobacco consumption, in kg

ldl Low-density lipoprotein cholesterol

adiposity Adipose tissue concentration

famhist Family history of heart disease (1=Present, 0=Absent)

typea Score on test designed to measure type-A behavior

obesity Obesity

alcohol Current consumption of alcohol

age Age of subject

Source

https://web.stanford.edu/~hastie/ElemStatLearn/

References

- Hastie T, Tibshirani R, and Friedman J. (2001). *The Elements of Statistical Learning*. Springer.
- Rousseauw J, et al. (1983). Coronary risk factor screening in three rural communities. *South African Medical Journal*, **64**: 430-436.

local_mfdr *Estimate local mFDR for all features*

Description

`local_mfdr()` is called by `summary.ncvreg()`, which typically offers a more convenient interface to users. If, however, you are working with local mfdrs programmatically rather than interactively, you probably want to use `local_mfdr()`, which skips the sorting, filtering, and print formatting of `summary.ncvreg()`.

Usage

```
local_mfdr(
  fit,
  lambda,
  X = NULL,
  y = NULL,
  method = c("ashr", "kernel"),
  sigma,
  ...
)
```

Arguments

`fit`	A fitted `ncvreg` or `ncvsurv` object.
`lambda`	The value of lambda at which inference should be carried out.
`X, y`	The design matrix and response used to fit the model; in most cases, it is not necessary to provide `X` and `y` as they are returned by `ncvreg`, but see the `returnX` argument in `ncvreg()`.
`method`	What method should be used to calculate the local fdr? Options are `ashr` (which tends to be more accurate) and `kernel` (which requires no additional packages). The default is to use `ashr` if the package is installed.
`sigma`	For linear regression models, users can supply an estimate of the residual standard deviation. The default is to use RSS / DF, where degrees of freedom are approximated using the number of nonzero coefficients.
`...`	Additional arguments to `ashr::ash()` if using `method='ashr'`.

Value

If all features are penalized, then the object returns a data frame with one row per feature and four columns:

- `Estimate`: The coefficient estimate from the penalized regression fit
- `z`: A test statistic that approximately follows a standard normal distribution under the null hypothesis that the feature is marginally independent of the outcome

- mfdr: The estimated marginal local false discovery rate
- Selected: Features with nonzero coefficient estimates are given an asterisk

If some features are penalized and others are not, then a list is returned with two elements: pen.vars, which consists of the data frame described above, and unpen.vars, a data frame with four columns: Estimate, SE, Statistic, and p.value. The standard errors and p-values are based on a classical lm/glm/coxph model using the effect of the penalized features as an offset.

See Also

summary.ncvreg()

Examples

```
# Linear regression
data(Prostate)
fit <- ncvreg(Prostate$X, Prostate$y)
local_mfdr(fit, 0.1)

fit <- ncvreg(Prostate$X, Prostate$y, penalty.factor=rep(0:1, each=4))
local_mfdr(fit, 0.1)

# Logistic regression
data(Heart)
X <- Heart$X
y <- Heart$y
fit <- ncvreg(X, y, family='binomial')
local_mfdr(fit, 0.1)

# Cox regression
data(Lung)
X <- Lung$X
y <- Lung$y
fit <- ncvsurv(X, y)
local_mfdr(fit, 0.1)
```

logLik.ncvreg *Extract Log-Likelihood*

Description

Extract the log-likelihood of an ncvreg or ncvsurv object.

Usage

```
## S3 method for class 'ncvreg'
logLik(object, REML = FALSE, ...)

## S3 method for class 'ncvsurv'
logLik(object, ...)
```

Arguments

`object`	An `ncvreg` or `ncvsurv` object, as obtained from `ncvreg()` or `ncvsurv()`
`REML`	As in `logLik.lm()`
`...`	For S3 compatibility

See Also

`logLik()`

`Lung` *VA lung cancer data set*

Description

Data from a randomised trial of two treatment regimens for lung cancer. This is a standard survival analysis data set from the classic textbook by Kalbfleisch and Prentice.

Usage

```
Lung
```

Format

A list of two objects: `y` and `X`

y A two column matrix (`Surv` object) containing the follow-up time (in days) and an indicator variable for whether the patient died while on the study or not.

X A matrix with 137 observations (rows) and 9 predictor variables (columns). The remainder of this list describes the columns of `X`

trt Treatment indicator (1=control group, 2=treatment group)

karno Karnofsky performance score (0=bad, 100=good)

diagtime Time from diagnosis to randomization (months)

age Age (years, at baseline)

prior Prior therapy (0=no, 1=yes)

squamous Indicator for whether the cancer type is squamous cell carcinoma (0=no, 1=yes)

small Indicator for whether the cancer type is small cell lung cancer (0=no, 1=yes)

adeno Indicator for whether the cancer type is adenocarcinoma (0=no, 1=yes)

large Indicator for whether the cancer type is large cell carcinoma (0=no, 1=yes)

Source

`https://cran.r-project.org/package=survival`

References

- Kalbfleisch D and Prentice RL (1980), *The Statistical Analysis of Failure Time Data.* Wiley, New York.

See Also

`ncvsurv()`

`mfdr` *Marginal false discovery rates*

Description

Estimates the marginal false discovery rate (mFDR) of a penalized regression model.

Usage

```
mfdr(fit, X)
```

Arguments

`fit`	An `ncvreg` or `ncvsurv` object.
`X`	The model matrix corresponding to `fit`. This is not necessary for linear regression, but in logistic and Cox regression, the mFDR depends on X. It is not necessary to supply `X` if it is already contained in `fit`; i.e., if `ncvreg`/`ncvsurv` was run with `returnX=TRUE`.

Details

The function estimates the marginal false discovery rate (mFDR) for a penalized regression model. The estimate tends to be accurate in most settings, but will be slightly conservative if predictors are highly correlated. For an alternative way of estimating the mFDR, typically more accurate in highly correlated cases, see `perm.ncvreg()`.

Value

An object with S3 class `mfdr` inheriting from `data.frame`, containing:

EF The number of variables selected at each value of `lambda`, averaged over the permutation fits.

S The actual number of selected variables for the non-permuted data.

mFDR The estimated marginal false discovery rate (`EF/S`).

Author(s)

Patrick Breheny and Ryan Miller

See Also

ncvreg(), ncvsurv(), plot.mfdr(), perm.ncvreg()

Examples

```
# Linear regression --------------------------------
data(Prostate)
fit <- ncvreg(Prostate$X, Prostate$y)

obj <- mfdr(fit)
obj[1:10,]

# Comparison with perm.ncvreg
op <- par(mfrow=c(2,2))
plot(obj)
plot(obj, type="EF")
pmfit <- perm.ncvreg(Prostate$X, Prostate$y)
plot(pmfit)
plot(pmfit, type="EF")
par(op)

# Logistic regression ------------------------------
data(Heart)
fit <- ncvreg(Heart$X, Heart$y, family="binomial")
obj <- mfdr(fit)
head(obj)
op <- par(mfrow=c(1,2))
plot(obj)
plot(obj, type="EF")
par(op)

# Cox regression -----------------------------------
data(Lung)
fit <- ncvsurv(Lung$X, Lung$y)
obj <- mfdr(fit)
head(obj)
op <- par(mfrow=c(1,2))
plot(obj)
plot(obj, type="EF")
par(op)
```

ncvfit *Direct interface for nonconvex penalized regression (non-pathwise)*

Description

This function is intended for users who know exactly what they're doing and want complete control over the fitting process: no standardization is applied, no intercept is included, no path is fit. All of these things are best practices for data analysis, so if you are choosing not to do them, you are on your own – there is no guarantee that your results will be meaningful. Some things in particular that you should pay attention to:

- If your model has an intercept, it is up to you to (un)penalize it properly, typically by settings its corresponding element of penalty.factor to zero.
- You should provide initial values for the coefficients; in nonconvex optimization, initial values are very important in determining which local solution an algorithm converges to.

Usage

```
ncvfit(
  X,
  y,
  init = rep(0, ncol(X)),
  r,
  xtx,
  penalty = c("MCP", "SCAD", "lasso"),
  gamma = switch(penalty, SCAD = 3.7, 3),
  alpha = 1,
  lambda,
  eps = 1e-05,
  max.iter = 1000,
  penalty.factor = rep(1, ncol(X)),
  warn = TRUE
)
```

Arguments

X Design matrix; no intercept will be added, no standardization will occur (n x p matrix)

y Response vector (length n vector)

init Initial values for beta. Default: zero (length p vector)

r Residuals corresponding to init; these will be calculated if not supplied, but if they have already been calculated elsewhere, it is more efficient to pass them as an argument. WARNING: If you supply an incorrect value of r, the solution will be incorrect. (length n vector)

xtx X scales: the jth element should equal crossprod(X[,j])/n. These will be calculated if not supplied, but if they have already been calculated elsewhere, it is more efficient to pass them as an argument. 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 Penalty function to be applied, either "MCP" (default), "SCAD", or "lasso")

gamma Tuning parameter of the MCP/SCAD penalty, as in ncvreg(); default is 3 for MCP and 3.7 for SCAD.

alpha Tuning paramter controlling the ridge component of penalty, as in ncvreg(); default is 1 (meaning no ridge penalty) `lambda` Regularization parameter value at which to estimate beta; must be scalar – for pathwise optimization, see `ncvreg()`

`eps` Convergence threshhold. The algorithm iterates until the RMSD for the change in linear predictors for each coefficient is less than eps. Default is 1e-4.

`max.iter` Maximum number of allowed iterations; if this number is reached, algorithm will terminate prior to convergence. Default: 1000.

`penalty.factor` Multiplicative factor for the penalty applied to each coefficient, as in `ncvreg()`. In particular, note that if you include an intercept, you probably want to set its entry to zero here.

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

Details

At the moment, this function only works for least-squares loss functions. Additional functionality for other loss functions (logistic, Cox) is in development.

Value

A list containing:

- `beta`: The estimated regression coefficients
- `iter`: The number of iterations required to solve for ‘beta
- `loss`: The loss (residual sum of squares) at convergence
- `resid`: The residuals at convergence
- `lambda`: See above
- `penalty`: See above
- `gamma`: See above
- `alpha`: See above
- `penalty.factor`: See above
- `n`: Sample size

Examples

```
data(Prostate)
X <- cbind(1, Prostate$X)
y <- Prostate$y
fit <- ncvfit(X, y, lambda=0.1, penalty.factor=c(0, rep(1, ncol(X)-1)))
fit$beta
# Compare with:
coef(ncvreg(X, y), 0.1)
# The unstandardized version makes little sense here, as it fails to account
# for differences in the scales of the predictors.
```

`ncvreg` *Fit an MCP- or SCAD-penalized regression path*

Description

Fit coefficients paths for MCP- or SCAD-penalized regression models over a grid of values for the regularization parameter lambda. Fits linear and logistic regression models, with option for an additional L2 penalty.

Usage

```
ncvreg(
  X,
  y,
  family = c("gaussian", "binomial", "poisson"),
  penalty = c("MCP", "SCAD", "lasso"),
  gamma = switch(penalty, SCAD = 3.7, 3),
  alpha = 1,
  lambda.min = ifelse(n > p, 0.001, 0.05),
  nlambda = 100,
  lambda,
  eps = 1e-04,
  max.iter = 10000,
  convex = TRUE,
  dfmax = p + 1,
  penalty.factor = rep(1, ncol(X)),
  warn = TRUE,
  returnX,
  ...
)
```

Arguments

`X` The design matrix, without an intercept. `ncvreg` standardizes the data and includes an intercept by default.

`y` The response vector.

`family` Either "gaussian", "binomial", or "poisson", depending on the response.

`penalty` The penalty to be applied to the model. Either "MCP" (the default), "SCAD", or "lasso".

`gamma` The tuning parameter of the MCP/SCAD penalty (see details). Default is 3 for MCP and 3.7 for SCAD.

`alpha` Tuning parameter for the Mnet estimator which controls the relative contributions from the MCP/SCAD penalty and the ridge, or L2 penalty. `alpha=1` is equivalent to MCP/SCAD penalty, while `alpha=0` would be equivalent to ridge regression. However, `alpha=0` is not supported; `alpha` may be arbitrarily small, but not exactly 0.

`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 .05 otherwise.
`nlambda`	The number of lambda values. Default is 100.
`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 threshhold. The algorithm iterates until the RMSD for the change in linear predictors for each coefficient is less than `eps`. Default is `1e-4`.
`max.iter`	Maximum number of iterations (total across entire path). Default is 10000.
`convex`	Calculate index for which objective function ceases to be locally convex? Default is TRUE.
`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. In particular, `penalty.factor` can be 0, in which case the coefficient is always in the model without shrinkage.
`warn`	Return warning messages for failures to converge and model saturation? Default is TRUE.
`returnX`	Return the standardized design matrix along with the fit? By default, this option is turned on if X is under 100 MB, but turned off for larger matrices to preserve memory. Note that certain methods, such as `summary.ncvreg()` require access to the design matrix and may not be able to run if `returnX=FALSE`.
`...`	Not used.

Details

The sequence of models indexed by the regularization parameter `lambda` is fit using a coordinate descent algorithm. For logistic regression models, some care is taken to avoid model saturation; the algorithm may exit early in this setting. The objective function is defined to be

$$Q(\beta|X, y) = \frac{1}{n}L(\beta|X, y) + P_\lambda(\beta),$$

where the loss function L is the deviance (-2 times the log likelihood) for the specified outcome distribution (gaussian/binomial/poisson). See here for more details.

This algorithm is stable, very efficient, and generally converges quite rapidly to the solution. For GLMs, adaptive rescaling is used.

Value

An object with S3 class `"ncvreg"` containing:

beta The fitted matrix of coefficients. The number of rows is equal to the number of coefficients, and the number of columns is equal to nlambda.

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, family, gamma, alpha, penalty.factor Same as above.

convex.min The last index for which the objective function is locally convex. The smallest value of lambda for which the objective function is convex is therefore lambda[convex.min], with corresponding coefficients beta[,convex.min].

loss A vector containing the deviance (i.e., the loss) at each value of lambda. Note that for gaussian models, the loss is simply the residual sum of squares.

n Sample size.

Additionally, if returnX=TRUE, the object will also contain

X The standardized design matrix.

y The response, centered if family='gaussian'.

References

Breheny P and Huang J. (2011) Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, **5**: 232-253. doi:10.1214/10AOAS388

See Also

plot.ncvreg(), cv.ncvreg()

Examples

```
# Linear regression --------------------------------------------------
data(Prostate)
X <- Prostate$X
y <- Prostate$y

op <- par(mfrow=c(2,2))
fit <- ncvreg(X, y)
plot(fit, main=expression(paste(gamma,"=",3)))
fit <- ncvreg(X, y, gamma=10)
plot(fit, main=expression(paste(gamma,"=",10)))
fit <- ncvreg(X, y, gamma=1.5)
plot(fit, main=expression(paste(gamma,"=",1.5)))
fit <- ncvreg(X, y, penalty="SCAD")
plot(fit, main=expression(paste("SCAD, ",gamma,"=",3)))
par(op)

op <- par(mfrow=c(2,2))
fit <- ncvreg(X, y)
plot(fit, main=expression(paste(alpha,"=",1)))
```

```
fit <- ncvreg(X, y, alpha=0.9)
plot(fit, main=expression(paste(alpha,"=",0.9)))
fit <- ncvreg(X, y, alpha=0.5)
plot(fit, main=expression(paste(alpha,"=",0.5)))
fit <- ncvreg(X, y, alpha=0.1)
plot(fit, main=expression(paste(alpha,"=",0.1)))
par(op)

op <- par(mfrow=c(2,2))
fit <- ncvreg(X, y)
plot(mfdr(fit))             # Independence approximation
plot(mfdr(fit), type="EF")  # Independence approximation
perm.fit <- perm.ncvreg(X, y)
plot(perm.fit)
plot(perm.fit, type="EF")
par(op)

# Logistic regression ------------------------------------------------
data(Heart)
X <- Heart$X
y <- Heart$y

op <- par(mfrow=c(2,2))
fit <- ncvreg(X, y, family="binomial")
plot(fit, main=expression(paste(gamma,"=",3)))
fit <- ncvreg(X, y, family="binomial", gamma=10)
plot(fit, main=expression(paste(gamma,"=",10)))
fit <- ncvreg(X, y, family="binomial", gamma=1.5)
plot(fit, main=expression(paste(gamma,"=",1.5)))
fit <- ncvreg(X, y, family="binomial", penalty="SCAD")
plot(fit, main=expression(paste("SCAD, ",gamma,"=",3)))
par(op)

op <- par(mfrow=c(2,2))
fit <- ncvreg(X, y, family="binomial")
plot(fit, main=expression(paste(alpha,"=",1)))
fit <- ncvreg(X, y, family="binomial", alpha=0.9)
plot(fit, main=expression(paste(alpha,"=",0.9)))
fit <- ncvreg(X, y, family="binomial", alpha=0.5)
plot(fit, main=expression(paste(alpha,"=",0.5)))
fit <- ncvreg(X, y, family="binomial", alpha=0.1)
plot(fit, main=expression(paste(alpha,"=",0.1)))
par(op)
```

ncvsurv *Fit an MCP- or SCAD-penalized survival model*

Description

Fit coefficients paths for MCP- or SCAD-penalized Cox regression models over a grid of values for the regularization parameter lambda, with option for an additional L2 penalty.

Usage

```
ncvsurv(
  X,
  y,
  penalty = c("MCP", "SCAD", "lasso"),
  gamma = switch(penalty, SCAD = 3.7, 3),
  alpha = 1,
  lambda.min = ifelse(n > p, 0.001, 0.05),
  nlambda = 100,
  lambda,
  eps = 1e-04,
  max.iter = 10000,
  convex = TRUE,
  dfmax = p,
  penalty.factor = rep(1, ncol(X)),
  warn = TRUE,
  returnX,
  ...
)
```

Arguments

`X` The design matrix of predictor values. `ncvsurv` standardizes the data prior to fitting.

`y` The time-to-event outcome, as a two-column matrix or `survival::Surv()` object. The first column should be time on study (follow up time); the second column should be a binary variable with 1 indicating that the event has occurred and 0 indicating (right) censoring.

`penalty` The penalty to be applied to the model. Either "MCP" (the default), "SCAD", or "lasso".

`gamma` The tuning parameter of the MCP/SCAD penalty (see details). Default is 3 for MCP and 3.7 for SCAD.

`alpha` Tuning parameter for the Mnet estimator which controls the relative contributions from the MCP/SCAD penalty and the ridge, or L2 penalty. `alpha=1` is equivalent to MCP/SCAD penalty, while `alpha=0` would be equivalent to ridge regression. However, `alpha=0` is not supported; `alpha` may be arbitrarily small, but not exactly 0.

`lambda.min` The smallest value for lambda, as a fraction of lambda.max. Default is .001 if the number of observations is larger than the number of covariates and .05 otherwise.

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

`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 threshhold. The algorithm iterates until the RMSD for the change in linear predictors for any coefficient is less than `eps`. Default is `1e-4`.

`max.iter` Maximum number of iterations (total across entire path). Default is 1000.

`convex` Calculate index for which objective function ceases to be locally convex? Default is TRUE.

`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. In particular, `penalty.factor` can be 0, in which case the coefficient is always in the model without any penalization/shrinkage.

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

`returnX` Return the standardized design matrix along with the fit? By default, this option is turned on if X is under 100 MB, but turned off for larger matrices to preserve memory. Note that certain methods, such as `summary.ncvreg()`, require access to the design matrix and may not be able to run if `returnX=FALSE`.

`...` Not used.

Details

The sequence of models indexed by the regularization parameter `lambda` is fit using a coordinate descent algorithm. In order to accomplish this, the second derivative (Hessian) of the Cox partial log-likelihood is diagonalized (see references for details). The objective function is defined to be

$$Q(\beta|X,y) = \frac{1}{n}L(\beta|X,y) + P_\lambda(\beta),$$

where the loss function L is the deviance (-2 times the partial log-likelihood) from the Cox regression mode. See here for more details.

Presently, ties are not handled by `ncvsurv` in a particularly sophisticated manner. This will be improved upon in a future release of **ncvreg**.

Value

An object with S3 class `ncvsurv` containing:

beta The fitted matrix of coefficients. The number of rows is equal to the number of coefficients, and the number of columns is equal to `nlambda`.

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, gamma, penalty.factor, alpha, model Same as above.

convex.min The last index for which the objective function is locally convex. The smallest value of lambda for which the objective function is convex is therefore `lambda[convex.min]`, with corresponding coefficients `beta[,convex.min]`.

loss The deviance of the fitted model at each value of `lambda`.

n The number of instances.

For Cox models, the following objects are also returned (and are necessary to estimate baseline survival conditonal on the estimated regression coefficients), all of which are ordered by time on study. I.e., the ith row of `W` does not correspond to the ith row of `X`):

W Matrix of `exp(beta)` values for each subject over all `lambda` values.

time Times on study.

fail Failure event indicator.

Additionally, if `returnX=TRUE`, the object will also contain

X The standardized design matrix.

References

- Breheny P and Huang J. (2011) Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, **5**: 232-253. doi:10.1214/10AOAS388
- Simon N, Friedman JH, Hastie T, and Tibshirani R. (2011) Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent. *Journal of Statistical Software*, **39**: 1-13. doi:10.18637/jss.v039.i05

See Also

`plot.ncvreg()`, `cv.ncvsurv()`

Examples

```
data(Lung)
X <- Lung$X
y <- Lung$y

op <- par(mfrow=c(2,2))
fit <- ncvsurv(X, y)
plot(fit, main=expression(paste(gamma,"=",3)))
fit <- ncvsurv(X, y, gamma=10)
plot(fit, main=expression(paste(gamma,"=",10)))
fit <- ncvsurv(X, y, gamma=1.5)
plot(fit, main=expression(paste(gamma,"=",1.5)))
fit <- ncvsurv(X, y, penalty="SCAD")
plot(fit, main=expression(paste("SCAD, ",gamma,"=",3)))
par(op)

fit <- ncvsurv(X,y)
```

```
ll <- log(fit$lambda)
op <- par(mfrow=c(2,1))
plot(ll, BIC(fit), type="l", xlim=rev(range(ll)))
lam <- fit$lambda[which.min(BIC(fit))]
b <- coef(fit, lambda=lam)
b[b!=0]
plot(fit)
abline(v=lam)
par(op)

S <- predict(fit, X, type='survival', lambda=lam)
plot(S, xlim=c(0,200))
```

perm.ncvreg *Permutation fitting for ncvreg*

Description

Fits multiple penalized regression models in which the outcome is randomly permuted, thereby allowing estimation of the marginal false discovery rate.

Usage

```
perm.ncvreg(
  X,
  y,
  ...,
  permute = c("outcome", "residuals"),
  N = 10,
  seed,
  trace = FALSE
)
```

Arguments

`X`	The design matrix, without an intercept, as in `ncvreg`.
`y`	The response vector, as in `ncvreg`.
`...`	Additional arguments to `ncvreg`.
`permute`	What to permute. If `'outcome'`, the response vector, `y`, is permuted. If `'residuals'`, the residuals are permuted. This is only available for linear regression (i.e., for `family='gaussian'`). Note that permuting the residuals may take a long time, as the residuals differ for each value of `lambda`, so separate permutations are required at every value of `lambda`. See also `permres()`.
`N`	The number of permutation replications. Default is 10.
`seed`	You may set the seed of the random number generator in order to obtain reproducible results.
`trace`	If set to TRUE, perm.ncvreg will inform the user of its progress by announcing the beginning of each permutation fit. Default is FALSE.

Details

The function fits a penalized regression model to the actual data, then repeats the process N times with a permuted version of the response vector. This allows estimation of the expected number of variables included by chance for each value of lambda. The ratio of this expected quantity to the number of selected variables using the actual (non-permuted) response is called the marginal false discovery rate (mFDR).

Value

An object with S3 class "perm.ncvreg" containing:

EF The number of variables selected at each value of lambda, averaged over the permutation fits.

S The actual number of selected variables for the non-permuted data.

mFDR The estimated marginal false discovery rate (EF/S).

fit The fitted ncvreg object for the original (non-permuted) data.

loss The loss/deviance for each value of lambda, averaged over the permutation fits. This is an estimate of the explanatory power of the model under null conditions, and can be used to adjust the loss of the fitted model in a manner akin to the idea of an adjusted R-squared in classical regression.

Author(s)

Patrick Breheny patrick-breheny@uiowa.edu

See Also

ncvreg(), plot.mfdr(), mfdr()

Examples

```
# Linear regression --------------------------------------------------
data(Prostate)
pmfit <- perm.ncvreg(Prostate$X, Prostate$y)

op <- par(mfcol=c(2,2))
plot(pmfit)
plot(pmfit, type="EF")
plot(pmfit$fit)
lam <- pmfit$fit$lambda

pmfit.r <- perm.ncvreg(Prostate$X, Prostate$y, permute='residuals')
plot(pmfit.r, col="red")              # Permuting residuals is
lines(lam, pmfit$mFDR, col="gray60")  # less conservative
par(op)

# Logistic regression ------------------------------------------------
data(Heart)
pmfit <- perm.ncvreg(Heart$X, Heart$y, family="binomial")
```

```
op <- par(mfcol=c(2,2))
plot(pmfit)
plot(pmfit, type="EF")
plot(pmfit$fit)
par(op)
```

permres *Permute residuals for a fitted ncvreg model*

Description

Fits multiple penalized regression models in which the residuals are randomly permuted, thereby allowing estimation of the marginal false discovery rate.

Usage

```
permres(fit, ...)

## S3 method for class 'ncvreg'
permres(fit, lambda, N = 10, seed, trace = FALSE, ...)
```

Arguments

`fit`	A fitted ncvreg model, as produced by `ncvreg()`. To use with `permres`, the model must be fit using the `returnX=TRUE` option.
`...`	Not used.
`lambda`	The regularization parameter to use for estimating residuals. Unlike `perm.ncvreg()`, `permres()` calculates EF and mFDR for a specific `lambda` value, not an entire path. As a result, it runs much faster.
`N`	The number of permutation replications. Default is 10.
`seed`	You may set the seed of the random number generator in order to obtain reproducible results.
`trace`	If set to TRUE, perm.ncvreg will inform the user of its progress by announcing the beginning of each permutation fit. Default is FALSE.

Details

The function fits a penalized regression model to the actual data, then repeats the process `N` times with a permuted version of the response vector. This allows estimation of the expected number of variables included by chance for each value of `lambda`. The ratio of this expected quantity to the number of selected variables using the actual (non-permuted) response is called the marginal false discovery rate (mFDR).

Value

A list with the following components:

`EF` The number of variables selected at each value of `lambda`, averaged over the permutation fits.

`S` The actual number of selected variables for the non-permuted data.

`mFDR` The estimated marginal false discovery rate (`EF/S`).

`loss` The loss/deviance, averaged over the permutation fits. This is an estimate of the explanatory power of the model under null conditions, and can be used to adjust the loss of the fitted model in a manner akin to the idea of an adjusted R-squared in classical regression.

Author(s)

Patrick Breheny patrick-breheny@uiowa.edu

See Also

`ncvreg()`, ‘`mfdr()`, `perm.ncvreg()`

Examples

```
data(Prostate)
fit <- ncvreg(Prostate$X, Prostate$y, N=50)
permres(fit, lambda=0.15)
```

`plot.cv.ncvreg` *Plots the cross-validation curve from a cv.ncvreg object*

Description

Plots the cross-validation curve from a `cv.ncvreg` or `cv.ncvsurv` object, along with standard error bars.

Usage

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

Arguments

`x` A `cv.ncvreg` or `cv.ncvsurv` 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% confidence intervals are plotted along with the estimates across values of `lambda`. For `rsq` and `snr` applied to models other than linear regression, the Cox-Snell R-squared is used.

Author(s)

Patrick Breheny

References

Breheny P and Huang J. (2011) Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, **5**: 232-253. doi:10.1214/10AOAS388

See Also

`ncvreg()`, `cv.ncvreg()`

Examples

```
# Linear regression --------------------------------------------------
data(Prostate)
cvfit <- cv.ncvreg(Prostate$X, Prostate$y)
plot(cvfit)
```

```
op <- par(mfrow=c(2,2))
plot(cvfit, type="all")
par(op)

# Logistic regression -------------------------------------------------
data(Heart)
cvfit <- cv.ncvreg(Heart$X, Heart$y, family="binomial")
plot(cvfit)
op <- par(mfrow=c(2,2))
plot(cvfit, type="all")
par(op)

# Cox regression ------------------------------------------------------
data(Lung)
cvfit <- cv.ncvsurv(Lung$X, Lung$y)
op <- par(mfrow=c(1,2))
plot(cvfit)
plot(cvfit, type="rsq")
par(op)
```

plot.mfdr *Plot marginal false discovery rate curves*

Description

Plot marginal false discovery rate curves from an mfdr or perm.ncvreg object.

Usage

```
## S3 method for class 'mfdr'
plot(
  x,
  type = c("mFDR", "EF"),
  log.l = FALSE,
  selected = TRUE,
  legend = TRUE,
  ...
)
```

Arguments

x A perm.ncvreg or mfdr object.

type What to plot on the vertical axis. mFDR plots the marginal false discovery rate; EF plots the expected number of false discoveries along with the actual number of variables included in the model.

log.l Should horizontal axis be on the log scale? Default is FALSE.

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.

`legend` For `type="EF"` plots, draw a legend to indicate which line is for the actual selections and which line is for the expected number of false discoveries? Default is `TRUE`.

`...` Other graphical parameters to pass to `plot()`

Author(s)

Patrick Breheny

References

Breheny P (2019). Marginal false discovery rates for penalized regression models. Biostatistics, 20: 299-314.

See Also

`mfdr()`, `perm.ncvreg()`

Examples

```
data(Prostate)
fit <- ncvreg(Prostate$X, Prostate$y)

obj <- mfdr(fit)
obj[1:10,]

# Some plotting options
plot(obj)
plot(obj, type="EF")
plot(obj, log=TRUE)


# Comparison with perm.ncvreg
op <- par(mfrow=c(2,2))
plot(obj)
plot(obj, type="EF")
pmfit <- perm.ncvreg(Prostate$X, Prostate$y)
plot(pmfit)
plot(pmfit, type="EF")
par(op)
```

`plot.ncvreg` *Plot coefficients from a ncvreg object*

Description

Produces a plot of the coefficient paths for a fitted `ncvreg` object.

Usage

```
## S3 method for class 'ncvreg'
plot(x, alpha = 1, log.l = FALSE, shade = TRUE, col, ...)
```

Arguments

`x` Fitted "ncvreg" model.

`alpha` Controls alpha-blending, helpful when the number of features is large. Default is alpha=1.

`log.l` Should horizontal axis be on the log scale? Default is FALSE.

`shade` Should nonconvex region be shaded? Default is TRUE.

`col` Vector of colors for coefficient lines. By default, evenly spaced colors are selected automatically.

`...` Other graphical parameters to `plot()`

Author(s)

Patrick Breheny

References

Breheny P and Huang J. (2011) Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, **5**: 232-253. doi:10.1214/10AOAS388

See Also

`ncvreg()`

Examples

```
data(Prostate)
fit <- ncvreg(Prostate$X, Prostate$y)
plot(fit)
plot(fit, col="black")
plot(fit, log=TRUE)
fit <- ncvreg(Prostate$X, Prostate$y, penalty.factor=rep(c(1, 1, 1, Inf), 2))
plot(fit, col=c('red', 'black', 'green'))  # Recycled among nonzero paths
```

plot.ncvsurv.func *Plot survival curve for ncvsurv model*

Description

Plot survival curve for a model that has been fit using `ncvsurv()` followed by a prediction of the survival function using `predict.ncvsurv()`.

Usage

```
## S3 method for class 'ncvsurv.func'
plot(x, alpha = 1, ...)
```

Arguments

`x` A `ncvsurv.func` object, which is returned by `predict.ncvsurv()` if `type='survival'` is specified. See examples.

`alpha` Controls alpha-blending (i.e., transparency). Useful if many overlapping lines are present.

`...` Other graphical parameters to pass to `plot`

Author(s)

Patrick Breheny

See Also

`ncvsurv()`, `predict.ncvsurv()`

Examples

```
data(Lung)
X <- Lung$X
y <- Lung$y

fit <- ncvsurv(X, y)

# A single survival curve
S <- predict(fit, X[1,], type='survival', lambda=.15)
plot(S, xlim=c(0,200))

# Lots of survival curves
S <- predict(fit, X, type='survival', lambda=.08)
plot(S, xlim=c(0,200), alpha=0.3)
```

predict.cv.ncvreg *Model predictions based on a fitted ncvreg object.*

Description

Similar to other predict methods, this function returns predictions from a fitted ncvreg object.

Usage

```
## S3 method for class 'cv.ncvreg'
predict(
  object,
  X,
  type = c("link", "response", "class", "coefficients", "vars", "nvars"),
  which = object$min,
  ...
)

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

## S3 method for class 'cv.ncvsurv'
predict(
  object,
  X,
 type = c("link", "response", "survival", "median", "coefficients", "vars", "nvars"),
  which = object$min,
  ...
)

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

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

Arguments

object	Fitted ncvreg model object.

`X` Matrix of values at which predictions are to be made. Not used for `type="coefficients"` or for some of the `type` settings in `predict`.

`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`.

`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.

`lambda` Values of the regularization parameter `lambda` at which predictions are requested. For values of `lambda` not in the sequence of fitted models, linear interpolation is used.

`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.

Value

The object returned depends on type.

Author(s)

Patrick Breheny

References

Breheny P and Huang J. (2011) Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, **5**: 232-253. doi:10.1214/10AOAS388

See Also

`ncvreg()`

Examples

```
data(Heart)

fit <- ncvreg(Heart$X, Heart$y, family="binomial")
coef(fit, lambda=0.05)
head(predict(fit, Heart$X, type="link", lambda=0.05))
head(predict(fit, Heart$X, type="response", lambda=0.05))
head(predict(fit, Heart$X, type="class", lambda=0.05))
predict(fit, type="vars", lambda=c(0.05, 0.01))
predict(fit, type="nvars", lambda=c(0.05, 0.01))
```

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

Description

Similar to other predict methods, this function returns predictions from a fitted ncvsurv object.

Usage

```
## S3 method for class 'ncvsurv'
predict(
  object,
  X,
 type = c("link", "response", "survival", "hazard", "median", "coefficients", "vars",
    "nvars"),
  lambda,
  which = 1:length(object$lambda),
  ...
)
```

Arguments

object Fitted "ncvsurv" model object.

X Matrix of values at which predictions are to be made. Not used for type="coefficients" or for some of the type settings in predict.

type Type of prediction:

- link returns the linear predictors
- response gives the risk (i.e., exp(link))
- survival returns the estimated survival function
- median estimates median survival times The other options are all identical to their ncvreg() counterparts:
- 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. For values of lambda not in the sequence of fitted models, linear interpolation is used.

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.

Details

Estimation of baseline survival function conditional on the estimated values of beta is carried out according to the method described in Chapter 4.3 of Kalbfleish and Prentice. In particular, it agrees exactly the results returned by survfit.coxph(..., type='kalbfleisch-prentice') in the survival package.

Value

The object returned depends on type.

Author(s)

Patrick Breheny patrick-breheny@uiowa.edu

References

- Breheny P and Huang J. (2011) Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, **5**: 232-253. doi:10.1214/10AOAS388
- Kalbfleish JD and Prentice RL (2002). *The Statistical Analysis of Failure Time Data*, 2nd edition. Wiley.

See Also

ncvsurv()

Examples

```
data(Lung)
X <- Lung$X
y <- Lung$y

fit <- ncvsurv(X,y)
coef(fit, lambda=0.05)
head(predict(fit, X, type="link", lambda=0.05))
head(predict(fit, X, type="response", lambda=0.05))

# Survival function
S <- predict(fit, X[1,], type="survival", lambda=0.05)
S(100)
S <- predict(fit, X, type="survival", lambda=0.05)
plot(S, xlim=c(0,200))

# Medians
predict(fit, X[1,], type="median", lambda=0.05)
M <- predict(fit, X, type="median")
M[1:10, 1:10]

# Nonzero coefficients
predict(fit, type="vars", lambda=c(0.1, 0.01))
predict(fit, type="nvars", lambda=c(0.1, 0.01))
```

`Prostate` *Factors associated with prostate specific antigen*

Description

Data from a study by by Stamey et al. (1989) to examine the association between prostate specific antigen (PSA) and several clinical measures that are potentially associated with PSA in men who were about to receive a radical prostatectomy.

Usage

```
Prostate
```

Format

A list of two objects: `y` and `X`

y Log PSA

X A matrix with 97 instances (rows) and 8 predictor variables (columns). The remainder of this list describes the columns of `X`

lcavol Log cancer volume

lweight Log prostate weight

age The man's age (years)

lbph Log of the amount of benign hyperplasia

svi Seminal vesicle invasion (1=Yes, 0=No)

lcp Log of capsular penetration

gleason Gleason score

pgg45 Percent of Gleason scores 4 or 5

Source

`https://web.stanford.edu/~hastie/ElemStatLearn/`

References

- Hastie T, Tibshirani R, and Friedman J. (2001). *The Elements of Statistical Learning*. Springer.
- Stamey T, et al. (1989). Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. II. Radical prostatectomy treated patients. *Journal of Urology*, **16**: 1076-1083.

residuals.ncvreg *Extract residuals from a ncvreg or ncvsurv fit*

Description

Currently, only deviance residuals are supported.

Usage

```
## S3 method for class 'ncvreg'
residuals(object, lambda, which = 1:length(object$lambda), drop = TRUE, ...)
```

Arguments

object	Object of class ncvreg or ncvsurv.
lambda	Values of the regularization parameter at which residuals are requested (numeric vector). For values of lambda not in the sequence of fitted models, linear interpolation is used.
which	Index of the penalty parameter at which residuals are requested (default = all indices). If lambda is specified, this take precedence over which.
drop	By default, if a single value of lambda is supplied, a vector of residuals is returned (logical; default=TRUE). Set drop=FALSE if you wish to have the function always return a matrix (see drop()).
...	Not used.

Examples

```
data(Prostate)
X <- Prostate$X
y <- Prostate$y
fit <- ncvreg(X, y)
residuals(fit)[1:5, 1:5]
head(residuals(fit, lambda=0.1))
```

std *Standardizes a design matrix*

Description

Accepts a design matrix and returns a standardized version of that matrix (i.e., each column will have mean 0 and mean sum of squares equal to 1).

Usage

```
std(X, Xnew)
```

Arguments

X	A matrix (or object that can be coerced to a matrix, such as a data frame or numeric vector).
Xnew	Optional. If supplied, X must be the output of std() and Xnew is to be standardized in the same way. See examples for why this might be useful.

Details

This function centers and scales each column of X so that

$$\sum_{i=1}^{n} x_{ij} = 0$$

and

$$n^{-1} \sum_{i=1}^{n} x_{ij}^2 = 1$$

for all j. This is usually not necessary to call directly, as **ncvreg** internally standardizes the design matrix, but inspection of the standardized design matrix can sometimes be useful. This differs from the base R function scale() in two ways:

1. scale() uses the sample standard deviation sqrt(sum(x^2)/(n-1)), while std() uses the root-mean-square standard deviation sqrt(mean(sum(x^2)) without the $n/(n-1)$ correction
2. std is faster.

Value

The standardized design matrix, with the following attribues:

center, scale	mean and standard deviation used to scale the columns
nonsingular	A vector indicating which columns of the original design matrix were able to be standardized (constant columns cannot be standardized to have a standard deviation of 1)

Examples

```
data(Prostate)
S <- std(Prostate$X)
apply(S, 2, sum)
apply(S, 2, function(x) mean(x^2))

# Standardizing new observations
X1 <- Prostate$X[1:90,]
X2 <- Prostate$X[91:97,]
S <- std(X1)
head(std(S, X2))
# Useful if you fit to a standardized X, but then get new obs:
y <- Prostate$y[1:90]
fit <- ncvreg(S, y)
predict(fit, std(S, X2), lambda=0.1)
# Same as
predict(ncvreg(X1, y), X2, lambda=0.1)
```

`summary.cv.ncvreg` *Summarizing cross-validation-based inference*

Description

Summary method for `cv.ncvreg` objects

Usage

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

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

Arguments

`object`	A `cv.ncvreg` or `cv.ncvsurv` object.
`...`	Further arguments passed to or from other methods.
`x`	A `summary.cv.ncvreg` 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

An object with S3 class `summary.cv.ncvreg`. The class has its own print method and contains the following list elements:

penalty The penalty used by `ncvreg`.

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

n Number of instances

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.ncvreg`.

cve Cross-validation error (deviance).

r.squared Proportion of variance explained by the model, as estimated by cross-validation. For models outside of linear regression, the Cox-Snell approach to defining R-squared is used.

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).

Author(s)

Patrick Breheny

References

Breheny P and Huang J. (2011) Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, **5**: 232-253. doi:10.1214/10AOAS388

See Also

`ncvreg()`, `cv.ncvreg()`, `plot.cv.ncvreg()`

Examples

```
# Linear regression ---------------------------------------------------
data(Prostate)
cvfit <- cv.ncvreg(Prostate$X, Prostate$y)
summary(cvfit)

# Logistic regression -------------------------------------------------
data(Heart)
cvfit <- cv.ncvreg(Heart$X, Heart$y, family="binomial")
summary(cvfit)

# Cox regression ------------------------------------------------------
data(Lung)
cvfit <- cv.ncvsurv(Lung$X, Lung$y)
summary(cvfit)
```

`summary.ncvreg` *Summary method for ncvreg objects*

Description

Inferential summaries for `ncvreg` and `ncvsurv` objects based on local marginal false discovery rates.

Usage

```
## S3 method for class 'ncvreg'
summary(object, lambda, which, number, cutoff, sort = TRUE, sigma, ...)

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

Arguments

`object` An `ncvreg` or `ncvsurv` object.

`lambda` The regularization parameter value at which inference should be reported.

which — Alternatively, lambda may be specified by index; which=10 means: report inference for the 10th value of lambda along the regularization path. If both lambda and which are specified, lambda takes precedence.

number — By default, summary will provide an inferential summary for each variable that has been selected (i.e., each variable with a nonzero coefficient). Specifying number=5, for example, means that the summary table will include the 5 features with the lowest mfdr values, regardless of whether they were selected. To see all features, number=Inf.

cutoff — Alternatively, specifying for example cutoff=0.3 will report inference for all features with mfdr under 30%. If both number and cutoff are specified, the intersection between both sets of features is reported.

sort — Should the results be sorted by mfdr? (default: TRUE)

sigma — For linear regression models, users can supply an estimate of the residual standard deviation. The default is to use RSS / DF, where degrees of freedom are approximated using the number of nonzero coefficients.

... — Further arguments; in particular, if you have set returnX=FALSE, you will need to supply X and y in order to calculate local mFDRs.

x — A summary.ncvreg 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

An object with S3 class summary.ncvreg. The class has its own print method and contains the following list elements:

penalty — The penalty used by ncvreg or ncvsurv

model — Either "linear", "logistic", or "Cox".

n — Number of instances.

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

lambda — The lambda value at which inference is being reported.

nvars — The number of nonzero coefficients (again, not including the intercept) at that value of lambda.

table — A table containing estimates, normalized test statistics (z), and an estimate of the local mfdr for each coefficient. The mfdr may be loosely interpreted, in an empirical Bayes sense, as the probability that the given feature is null.

unpen.table — If there are any unpenalized coefficients, a separate inferential summary is given for them. Currently, this is based on lm/glm/coxph using the penalized coefficients to provide an offset. This is useful and more or less accurate, but not ideal; we hope to improve the inferential methods for unpenalized variables in the future.

Author(s)

Patrick Breheny patrick-breheny@uiowa.edu

See Also

ncvreg(), cv.ncvreg(), plot.cv.ncvreg(), local_mfdr()

Examples

```
# Linear regression --------------------------------------------------
data(Prostate)
fit <- ncvreg(Prostate$X, Prostate$y)
summary(fit, lambda=0.08)

# Logistic regression ------------------------------------------------
data(Heart)
fit <- ncvreg(Heart$X, Heart$y, family=”binomial”)
summary(fit, lambda=0.05)

# Cox regression -----------------------------------------------------
data(Lung)
fit <- ncvsurv(Lung$X, Lung$y)
summary(fit, lambda=0.1)

# Options ------------------------------------------------------------
fit <- ncvreg(Heart$X, Heart$y, family=”binomial”)
summary(fit, lambda=0.08, number=3)
summary(fit, lambda=0.08, number=Inf)
summary(fit, lambda=0.08, cutoff=0.5)
summary(fit, lambda=0.08, number=3, cutoff=0.5)
summary(fit, lambda=0.08, number=5, cutoff=0.1)
summary(fit, lambda=0.08, number=Inf, sort=FALSE)
summary(fit, lambda=0.08, number=3, cutoff=0.5, sort=FALSE)

# If X and y are not returned with the fit, they must be supplied
fit <- ncvreg(Heart$X, Heart$y, family=”binomial”, returnX=FALSE)
summary(fit, X=Heart$X, y=Heart$y, lambda=0.08)
```

Index

∗ **datasets**
Heart, 9
Lung, 13
Prostate, 38

ashr::ash(), *11*
assign_fold, 2
AUC (AUC.cv.ncvsurv), 3
AUC.cv.ncvsurv, 3

boot_ncvreg, 4

coef.cv.ncvreg (predict.cv.ncvreg), 34
coef.ncvreg (predict.cv.ncvreg), 34
coef.ncvsurv (predict.ncvsurv), 36
cv.ncvreg, 7
cv.ncvreg(), *20*, *29*, *42*, *44*
cv.ncvsurv (cv.ncvreg), 7
cv.ncvsurv(), *3, 4*, *24*

drop(), *39*

Heart, 9

local_mfdr, 11
local_mfdr(), *44*
logLik(), *13*
logLik.lm(), *13*
logLik.ncvreg, 12
logLik.ncvsurv (logLik.ncvreg), 12
Lung, 13

mfdr, 14
mfdr(), *26*, *28*, *31*

ncvfit, 15
ncvreg, 18
ncvreg(), *7–9*, *11*, *13*, *15–17*, *26–29*, *32*, *35, 36*, *42*, *44*
ncvsurv, 21
ncvsurv(), *7, 8*, *13–15*, *33*, *37*

parallel::makeCluster(), *5*, *7*
perm.ncvreg, 25
perm.ncvreg(), *14, 15*, *27, 28*, *31*
permres, 27
permres(), *25*
plot(), *29*, *31, 32*
plot.cv.ncvreg, 28
plot.cv.ncvreg(), *9*, *42*, *44*
plot.mfdr, 30
plot.mfdr(), *15*, *26*
plot.ncvreg, 31
plot.ncvreg(), *20*, *24*
plot.ncvsurv.func, 33
predict.cv.ncvreg, 34
predict.cv.ncvsurv (predict.cv.ncvreg), 34
predict.ncvreg (predict.cv.ncvreg), 34
predict.ncvsurv, 36
predict.ncvsurv(), *33*
print.summary.cv.ncvreg (summary.cv.ncvreg), 41
print.summary.ncvreg (summary.ncvreg), 42
Prostate, 38
prostate (Prostate), 38

residuals.ncvreg, 39

scale(), *40*
std, 39
summary.cv.ncvreg, 41
summary.cv.ncvreg(), *9*
summary.ncvreg, 42
summary.ncvreg(), *11, 12*, *19*, *23*
survival::concordancefit(), *3, 4*
survival::Surv(), *22*