Package: enetLTS (via r-universe)

November 22, 2026

Type Package

Title Robust and Sparse Methods for High Dimensional Linear, Binary and Multinomial Regression

Version 1.2.0

Description Fully robust versions of the elastic net estimator for linear, binary and multinomial regression, in particular for high dimensional data. The algorithm searches for outlier-free subsets on which the classical elastic net estimator can be applied. A reweighting step is added to improve the statistical efficiency of the estimators. The tuning parameters of the elastic net penalty are selected by cross-validation. The methods are described in Kurnaz, Hoffmann and Filzmoser (2018) <doi:10.1016/j.chemolab.2017.11.017> and in Kurnaz and Filzmoser (2023) <doi:10.1007/s10618-023-00936-6>.

Depends R (>= 3.6.0)

Imports ggplot2 (>= 3.4.0), glmnet, parallel, stats, robustbase, robustHD, grDevices

Suggests knitr, rmarkdown

VignetteBuilder knitr

License GPL (>= 3)

URL https://github.com/fatmasevinck/enetLTS

BugReports https://github.com/fatmasevinck/enetLTS/issues

Encoding UTF-8

Language en-US

LazyData true

NeedsCompilation no

Author Fatma Sevinc Kurnaz [aut, cre], Irene Hoffmann [aut], Peter Filzmoser [aut]

Maintainer Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

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

Date/Publication 2026-11-22 00:28:35 UTC

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

RemoteRef HEAD

RemoteSha dee75c48cd5cdbba07bd56119cf8de737901583c

Contents

assess.enetLTS . . . 2
BinomialExample . . . 4
coef.enetLTS . . . 5
cv.enetLTS . . . 7
enetLTS . . . 8
fitted.enetLTS . . . 13
GaussianExample . . . 15
lambda00 . . . 16
MultinomialExample . . . 17
nonzeroCoef.enetLTS . . . 18
outliers.enetLTS . . . 20
plot.enetLTS . . . 21
plotCoef.enetLTS . . . 23
plotDiagnostic.enetLTS . . . 25
plotResid.enetLTS . . . 26
predict.enetLTS . . . 28
print.enetLTS . . . 30
residuals.enetLTS . . . 32
weights.enetLTS . . . 34

Index **36**

`assess.enetLTS` *Assess an* `"enetLTS"` *fit on test data*

Description

Performance measures, confusion matrix and ROC curve of a fitted `"enetLTS"` object on (new) data.

Usage

```
assess.enetLTS(object, newX, newy, vers = c("both", "reweighted", "raw"))

confusion.enetLTS(object, newX, newy, vers = c("reweighted", "raw"))

roc.enetLTS(object, newX, newy, vers = c("reweighted", "raw"), plot = FALSE, ...)

## S3 method for class 'confusion.enetLTS'
print(x, ...)
```

Arguments

object	a fitted object of class "enetLTS".
newX	a numeric matrix with the predictors of the test data.
newy	the response of the test data: numeric for "gaussian", class labels (as in the training data) for "binomial" and "multinomial".
vers	which fit to assess: "reweighted" (the default of confusion.enetLTS and roc.enetLTS) or "raw". assess.enetLTS also accepts "both" (its default), which reports the two fits side by side.
plot	logical; if TRUE the ROC curve is drawn. The default is FALSE.
x	an object of class "confusion.enetLTS".
...	not used.

Details

The test data should be free of outliers, since they stand for the data on which the model is used.

assess.enetLTS computes the mean squared error mse, its root rmse and the mean absolute error mae (gaussian); the deviance deviance, the misclassification rate class, auc, mse and mae of the predicted probabilities (binomial); and the same without auc (multinomial; mse and mae are summed over the classes). As in glmnet::assess.glmnet, deviance is twice the mean negative log-likelihood. For "binomial" the second class of object$classnames is the positive class, and an observation is classified as positive if its probability is larger than 0.5.

confusion.enetLTS and roc.enetLTS are available for the classification families (roc.enetLTS for "binomial" only).

Value

assess.enetLTS: a list with one element per measure, in the layout of glmnet::assess.glmnet. Every element is a named vector with one value per fit (reweighted, raw) and has the attribute "measure" with the name of the measure.

confusion.enetLTS: an object of class "confusion.enetLTS", a list with the contingency table table (predicted by true class) and the proportion of correct classifications correct.

roc.enetLTS: a data frame with the columns FPR and TPR, with the AUC as attribute "auc", invisibly.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

References

Hastie, T., Qian, J. and Tay, K. (2026). An Introduction to glmnet. Vignette of the R package **glmnet**. https://glmnet.stanford.edu/articles/glmnet.html

See Also

enetLTS, predict.enetLTS, outliers.enetLTS

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                              # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))            # six nonzero coefficients
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
xt <- matrix(rnorm(200 * p), 200, p)           # clean test data
yt <- drop(xt %*% beta) + rnorm(200)
m <- 10                                        # contaminated observations
y[1:m] <- y[1:m] + 10                          # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                      # bad leverage points

## logistic regression
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
xbt <- matrix(rnorm(200 * p), 200, p)          # clean test data
ybt <- rbinom(200, 1, plogis(drop(xbt %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                      # observations of class 0 ...
xb[i0, ] <- xb[i0, ] + 10                      # ... become bad leverage points


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
assess.enetLTS(fit, xt, yt)
assess.enetLTS(fitb, xbt, ybt)
confusion.enetLTS(fitb, xbt, ybt)
roc.enetLTS(fitb, xbt, ybt, plot = TRUE)
```

`BinomialExample` *Simulated high-dimensional logistic regression data with outliers*

Description

Simulated data for `family = "binomial"` with $n = 100$ observations and $p = 200$ predictors ($p > n$), following the simulation design of Kurnaz, Hoffmann and Filzmoser (2018).

Usage

```
data("BinomialExample")
```

Format

A list with components

`x` training predictors, a 100×200 matrix.

`y` training response, a 0/1 vector of length 100.

`outliers` indices of the contaminated training observations (1, 3, 7, 13).

`xtest`, `ytest` clean test data with 200 observations.

`beta` the true slope vector (the true intercept is 1).

Details

The first 20 predictors are informative ($\beta_j = 1$) and correlated with $\mathrm{cor}(x_j, x_k) = 0.9^{|j-k|}$; the other 180 are uninformative and correlated with $0.5^{|j-k|}$. The response is $y = 1$ if $1 + x^\top \beta + \varepsilon > 0$ with $\varepsilon \sim N(0, 1)$, and $y = 0$ otherwise. The first 10% of the observations of class 0 (4 observations) are contaminated: their informative predictors are drawn from $N(20, 1)$ while their label stays 0, so they are bad leverage points. Values are rounded to three decimals. The generating code is in the `data-raw` directory of the package repository on GitHub.

References

Kurnaz, F.S., Hoffmann, I. and Filzmoser, P. (2018). Robust and sparse estimation methods for high-dimensional linear and logistic regression. *Chemometrics and Intelligent Laboratory Systems*, **172**, 211–222. doi:10.1016/j.chemolab.2017.11.017

Examples

```
data("BinomialExample")
table(BinomialExample$y)
BinomialExample$outliers
```

---

coef.enetLTS *Coefficients of an* "enetLTS" *fit*

Description

Extracts the estimated coefficients of the reweighted or the raw fit.

Usage

```
## S3 method for class 'enetLTS'
coef(object, vers = c("reweighted", "raw"), zeros = TRUE, ...)
```

Arguments

`object`	a fitted object of class `"enetLTS"`.
`vers`	a character string: `"reweighted"` (the default) for the reweighted fit or `"raw"` for the raw fit.
`zeros`	logical; if `TRUE` (the default) all coefficients are returned, if `FALSE` only the nonzero ones (for the multinomial family: the rows that are nonzero in at least one class).
`...`	not used.

Value

For `family = "gaussian"` and `"binomial"`, a named numeric vector: the intercept `"(Intercept)"` (if `intercept = TRUE`) followed by the slopes, named by the column names of `xx` (`x1`, `x2`, ... if `xx` has no column names). For `family = "multinomial"`, a matrix with one row per coefficient (named as above) and one column per class.

Note

In versions before 1.2.0, `zeros = FALSE` returned the indices of the nonzero coefficients; these are now given by `nonzeroCoef.enetLTS`.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

See Also

`enetLTS`, `predict.enetLTS`, `nonzeroCoef.enetLTS`, `print.enetLTS`

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                              # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))            # six nonzero coefficients
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
m <- 10                                        # contaminated observations
y[1:m] <- y[1:m] + 10                          # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                      # bad leverage points

## logistic regression
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                      # observations of class 0 ...
xb[i0, ] <- xb[i0, ] + 10                      # ... become bad leverage points

## multinomial regression
set.seed(86)
n <- 120; p <- 15; K <- 3                      # observations, variables, classes
Bm <- matrix(0, p, K)
Bm[1:2, 1] <- 3; Bm[3:4, 2] <- 3; Bm[5:6, 3] <- 3
xm <- matrix(rnorm(n * p), n, p)
pr <- exp(xm %*% Bm); pr <- pr / rowSums(pr)
ym <- apply(pr, 1, function(q) sample(K, 1, prob = q))
mm <- 10
```

```
xm[1:mm, 1:2] <- xm[1:mm, 1:2] + 5             # far out in the predictors of class 1 ...
ym[1:mm] <- 3                                   # ... but labeled as class 3


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
fitm <- enetLTS(xm, ym, family = "multinomial")
coef(fit, zeros = FALSE)
coef(fit, vers = "raw", zeros = FALSE)
coef(fitb, zeros = FALSE)
coef(fitm, zeros = FALSE)
```

`cv.enetLTS` *Cross-validation over the* (α, λ) *grid*

Description

Repeated k-fold cross-validation over a grid of α and λ values, as used inside `enetLTS`. Returns the selected pair and, optionally, draws a heat map of the criterion.

Usage

```
cv.enetLTS(index = NULL, family = c("gaussian", "binomial", "multinomial"),
          xx, yy, alphas, lambdas, nfold = 5, repl = 1, ncores = NA, plot = TRUE)
```

Arguments

`index`	`NULL` (the default) or the subsets found by the C-steps: a list with one element per value of `alphas`, each a matrix with h rows (observation numbers) and one column per value of `lambdas`. If given, the cross-validation for a pair (α_i, λ_j) uses only the observations `index[[i]][, j]`.
`family`	`"gaussian"`, `"binomial"` or `"multinomial"`.
`xx`	a numeric matrix of predictors.
`yy`	the response: numeric (gaussian), 0/1 (binomial), or class labels or an indicator matrix (multinomial).
`alphas`	the values of the mixing parameter $\alpha \in [0, 1]$ ($\alpha = 1$: lasso, $\alpha = 0$: ridge).
`lambdas`	the values of the penalty parameter. If missing, the default grid of `enetLTS` for the family is used.
`nfold`	the number of folds. The default is 5.
`repl`	the number of repetitions of the cross-validation; the criterion is averaged over the repetitions. The default is 1.
`ncores`	the number of processor cores; `NA` (the default) uses all available cores.
`plot`	logical; if `TRUE` (the default) a heat map of the criterion over the grid is drawn.

Details

The folds are random for the gaussian family and stratified by class for the binomial and multinomial families. The criterion is the root mean squared prediction error (gaussian) or the mean negative log-likelihood (binomial, multinomial) of the test folds. If `index = NULL`, the observations are not trimmed beforehand, so the largest 10% of the test-fold losses are left out of the mean (labels `RTMSPE`, `TMNLL`). For the multinomial family the largest 10% are always left out, within every class.

Value

A list with components

`indexbest`	the subset of the selected pair (`NULL` if `index = NULL`).
`evalCrit`	the matrix of the criterion, rows for `alphas` and columns for `lambdas`.
`minevalCrit`	the smallest value of the criterion.
`lambdaopt`	the selected λ (rounded to 4 decimals).
`alphaopt`	the selected α (rounded to 4 decimals).

Note

This function is called by `enetLTS`; it is exported so that the tuning parameters can also be chosen for a given set of subsets.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

See Also

`enetLTS`

Examples

```
data("GaussianExample")
cv <- cv.enetLTS(family = "gaussian", xx = GaussianExample$x,
                 yy = GaussianExample$y, alphas = c(0.5, 1), ncores = 1)
cv[c("alphaopt", "lambdaopt")]
```

`enetLTS` *Robust and Sparse Methods for High Dimensional Linear, Binary and Multinomial Regression*

Description

Computes the robust elastic net estimator enetLTS for linear regression (`family = "gaussian"`), binary logistic regression (`"binomial"`) and multinomial logistic regression (`"multinomial"`). The estimator searches for an outlier-free subset of h observations on which the classical elastic net is fitted, and a reweighting step improves its efficiency.

Usage

```
enetLTS(xx, yy, family = c("gaussian", "binomial", "multinomial"),
        alphas = seq(0, 1, length = 41), lambdas = NULL, lambdaw = NULL,
        intercept = TRUE, scal = TRUE, hsize = 0.75, nsamp = c(500, 10),
        nCsteps = 20, nfold = 5, repl = 1, ncores = 1, tol = -1e6,
        seed = NULL, del = 0.0125, crit.plot = FALSE, typegrouped = FALSE,
        type.response = c("link", "response", "class"),
        hgrid = c(0.65, 0.75, 0.85, 0.95), screen = TRUE, crossfit = NULL)
```

Arguments

xx	a numeric matrix of predictors, $n \times p$; $p > n$ is allowed.
yy	the response. Numeric for `family = "gaussian"`; a numeric 0/1 vector (numeric, or a factor with levels 0 and 1) for `"binomial"`; for `"multinomial"` a factor (or a vector of class labels) with $K \geq 3$ classes, or an $n \times K$ indicator matrix.
family	`"gaussian"` (the default), `"binomial"` or `"multinomial"`.
alphas	the grid of the elastic net mixing parameter $\alpha \in [0, 1]$ ($\alpha = 1$: lasso, $\alpha = 0$: ridge). The default is 41 equally spaced values.
lambdas	the grid of the penalty parameter λ. By default the grid runs from λ_0 down to 0 in steps of $0.025\lambda_0$, where λ_0 is a robust estimate of the smallest penalty that sets all slopes to zero: `lambda0` for `"gaussian"` and `lambda00` for `"binomial"`. For `"multinomial"` the default is 20 values, equally spaced on the log scale, from the λ_{max} of **glmnet** ($\alpha = 1$, robustly standardized data) down to 1% of it.
lambdaw	the penalty of the reweighting step. By default it is chosen by `cv.glmnet` on the observations with raw weight 1.
intercept	logical; whether the model has an intercept (for `"multinomial"`: class intercepts). The default is `TRUE`.
scal	logical; if `TRUE` (the default), the predictors are centered and scaled by the means and standard deviations of the current subset in every C-step, and for `"gaussian"` the response is centered as well.
hsize	the proportion of observations in the subset, $h = \lfloor (n+1) \cdot$ `hsize` $\rfloor$. The default is 0.75. *Experimental:* with `hsize = "optimal"` the model is fitted for every value in `hgrid`, and the value with the smallest cross-fitted loss on all observations (ignoring the worst 25%) is chosen, taking the largest value within 10% of the best criterion. This takes about `length(hgrid)` times as long as a single fit. The result then contains `hsize.search` with the criterion of every candidate, and a warning is given if the chosen value is below 0.70.
nsamp	a vector of two integers. Two C-steps are carried out on each of `nsamp[1]` elemental subsets (default 500), and the `nsamp[2]` best of them (default 10) are iterated further.
nCsteps	the largest number of C-steps in one iteration; the C-steps stop earlier when the criterion no longer improves. The default is 20.
nfold	the number of folds of the cross-validation. The default is 5.
repl	the number of repetitions of the cross-validation. The default is 1.

`ncores` the number of processor cores (package **parallel**); `NA` uses all available cores. The default is 1.

`tol` not used by the current algorithm; kept for compatibility with earlier versions.

`seed` an optional seed for the random number generator, set before the elemental subsets are drawn (and, for `"multinomial"`, before the screening step). The default `NULL` does not set a seed.

`del` the tail probability of the reweighting step. For `"gaussian"` and `"binomial"` an observation gets weight 0 if its standardized or Pearson residual is larger than `qnorm(1 - del)` in absolute value; with the default 0.0125 this flags about 2.5% of the observations under the model. For `"multinomial"` with `crossfit = TRUE` the cut-off is calibrated at `2 * del` (see `crossfit`).

`crit.plot` logical; if `TRUE` a heat map of the cross-validation criterion over the (α, λ) grid is drawn. The default is `FALSE`.

`typegrouped` logical; `"multinomial"` only. If `TRUE`, the grouped lasso penalty of **glmnet** (`type.multinomial = "grouped"`) is used in the fits on the best subset and in the reweighting step. The default is `FALSE`.

`type.response` the scale of `fitted.values` and `raw.fitted.values` in the result: `"link"` (the default) for the linear predictors, `"response"` for the fitted values (gaussian) or probabilities, `"class"` for class labels (not for `"gaussian"`).

`hgrid` the candidate values of `hsize` (between 0.5 and 1), used only if `hsize = "optimal"`.

`screen` logical; `"multinomial"` only. If `TRUE` (the default), observations that are outlying in the predictors (leverage points) are found beforehand without using the class labels (spatial-sign principal components and robust distances of their scores) and are left out of the fit; they get weight zero in `raw.wt` and `wt` and are listed in `screened`. This protects against contamination that falls mostly into one class, for which the class-wise robust distances of the reweighting step are unreliable.

`crossfit` logical or `NULL` (the default); `"gaussian"` and `"multinomial"` only. `NULL` means `TRUE` for `"multinomial"` with `scal = TRUE` and `FALSE` otherwise. If `TRUE`, the observations that are kept are chosen by *cross-fitted* residuals: an observation is never judged by a model that has seen it. The ordinary trimming keeps the observations that the current fit describes best; with many predictors this partly selects observations that noise variables happen to explain, which raises the false positive rate and the prediction error even on clean data. After the trimming, α and λ are chosen again on the reweighted observations (common folds), within the range of `alphas`. For `"multinomial"` the cross-fitted loss is $-\log \hat{p}(y_i)$, and the reweighting step keeps an observation if its cross-fitted probability of the observed class, from a model fitted on all screened observations, is at least a cut-off c. The cut-off is calibrated: c is the largest value for which, if these probabilities were correct, the expected share of observations with $\hat{p}(y_i) < c$ is at most `2 * del`. An earlier rule, a robust median/MAD bound on the cross-fitted loss within every class, is used with `options(enetLTS.wrule = "mad")`. The final model contains class intercepts. Requires `scal = TRUE`. For `"binomial"` it is ignored with a warning.

Details

The estimator applies the classical elastic net (**glmnet**) to subsets of the data only. For one pair (α, λ) the search starts from `nsamp[1]` elemental subsets and improves them by C-steps; the C-steps keep the h observations with the smallest squared residuals (`"gaussian"`) or negative log-likelihood contributions (`"binomial"`, within each class). For `"multinomial"` they keep, within each class k, the $\lfloor n_k \cdot$`hsize`$+1 \rfloor$ observations with the smallest robust (MCD) distance in the space of the linear predictors. For `"binomial"` the start subsets are ranked by the Bianco-Yohai criterion (Croux and Haesbroeck, 2003), as in Kurnaz et al. (2018). The other grid points are computed with warm starts. The pair (α, λ) is chosen by `repl` times repeated `nfold`-fold cross-validation on the best subsets (see `cv.enetLTS`), and the raw fit is the elastic net on the best subset of the chosen pair.

In the reweighting step the observations with large residuals of the raw fit get weight 0 (see `del` and `weights.enetLTS`), and the elastic net with penalty `lambdaw` is fitted to the observations with weight 1; this is the reweighted fit. The weights `wt` are then computed from the reweighted fit with the same rule. With `crossfit = TRUE` the raw weights come from the cross-fitted trimming instead, and for `"multinomial"` `wt` then equals `raw.wt`.

Value

An object of class `"enetLTS"`, a list with components

`objective` the penalized criterion of the raw fit on the final subset H (the h retained observations), on the scale on which the C-steps fit and penalize: $\frac{1}{h}\sum_{i \in H} \ell_i + \lambda P_\alpha(s_H \circ \beta)$, where ℓ_i is the squared residual divided by two (`"gaussian"`) or the negative log-likelihood contribution (`"binomial"`, `"multinomial"`), β the raw coefficients without the intercept, s_H the standard deviations of the columns of `xx` on H (one if `scal = FALSE` and `family = "multinomial"`), and (α, λ) those of the raw fit. It is reported only and is not used by the algorithm. The search for `"binomial"` ranks subsets by the Bianco-Yohai criterion without penalty, so for that family the value is not what the C-steps optimize.

`raw.rmse`, `rmse` `"gaussian"` only: root mean squared residual of the raw and the reweighted fit on all observations.

`raw.mae`, `mae` mean absolute residual of the raw and the reweighted fit on all observations; for `"binomial"` and `"multinomial"` the mean absolute difference between the 0/1 indicators and the fitted probabilities.

`best` the indices of the best subset of h observations, used for the raw fit.

`raw.wt` integer 0/1 weights from the raw fit; the observations with weight 1 are used for the reweighted fit.

`wt` integer 0/1 weights from the reweighted fit (0 = flagged as outlier).

`raw.coefficients`, `coefficients` the coefficients of the raw and the reweighted fit, intercept first (a matrix with one column per class for `"multinomial"`). Use `coef.enetLTS` to get them with names.

`raw.fitted.values`, `fitted.values` the fitted values of the raw and the reweighted fit, on the scale given by `type.response`.

`raw.residuals`, `residuals` the residuals (`"gaussian"`) or the negative log-likelihood contributions (`"binomial"`, `"multinomial"`) of the raw and the reweighted fit.

alpha	the selected mixing parameter of the reweighted fit.
alpha.raw	"gaussian" and "multinomial": the mixing parameter of the raw fit. It equals alpha unless crossfit selects α again for the reweighted fit.
lambda	the selected penalty of the raw fit.
lambdaw	the penalty of the reweighted fit.
num.nonzerocoef	the number of nonzero coefficients of the reweighted fit.
n, p	the number of observations and of predictors.
h	the size of the subsets.
classnames, classize	"binomial" and "multinomial": the class labels and the class sizes.
screened	"multinomial" only: the indices of the observations removed by the screening step (screen = TRUE).
hsize.search	only if hsize = "optimal": the criterion of every candidate value.
inputs	a list with the data and the settings used by the fit.
call	the matched call.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

References

Kurnaz, F.S. and Filzmoser, P. (2023). enetLTS: Robust and sparse methods for high dimensional linear, binary, and multinomial regression. *Journal of Open Source Software*, **8**(82), 4773. doi:10.21105/joss.04773

Kurnaz, F.S. and Filzmoser, P. (2023). Robust and sparse multinomial regression in high dimensions. *Data Mining and Knowledge Discovery*, **37**(4), 1609–1629. doi:10.1007/s10618023009366

Kurnaz, F.S., Hoffmann, I. and Filzmoser, P. (2018). Robust and sparse estimation methods for high-dimensional linear and logistic regression. *Chemometrics and Intelligent Laboratory Systems*, **172**, 211–222. doi:10.1016/j.chemolab.2017.11.017

Croux, C. and Haesbroeck, G. (2003). Implementing the Bianco and Yohai estimator for logistic regression. *Computational Statistics & Data Analysis*, **44**, 273–295.

See Also

print.enetLTS, coef.enetLTS, predict.enetLTS, outliers.enetLTS, assess.enetLTS, plot.enetLTS, cv.enetLTS, GaussianExample, BinomialExample, MultinomialExample

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                          # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))        # six nonzero coefficients
```

```
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
m <- 10                                          # contaminated observations
y[1:m] <- y[1:m] + 10                            # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                        # bad leverage points

## logistic regression
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                        # observations of class 0 ...
xb[i0, ] <- xb[i0, ] + 10                        # ... become bad leverage points

## multinomial regression
set.seed(86)
n <- 120; p <- 15; K <- 3                        # observations, variables, classes
Bm <- matrix(0, p, K)
Bm[1:2, 1] <- 3; Bm[3:4, 2] <- 3; Bm[5:6, 3] <- 3
xm <- matrix(rnorm(n * p), n, p)
pr <- exp(xm %*% Bm); pr <- pr / rowSums(pr)
ym <- apply(pr, 1, function(q) sample(K, 1, prob = q))
mm <- 10
xm[1:mm, 1:2] <- xm[1:mm, 1:2] + 5               # far out in the predictors of class 1 ...
ym[1:mm] <- 3                                     # ... but labeled as class 3


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
fitm <- enetLTS(xm, ym, family = "multinomial")
fit
which(fit$wt == 0)
fitb
outliers.enetLTS(fitb)
fitm
fitm$screened
```

`fitted.enetLTS` *Fitted values of an* "enetLTS" *fit*

Description

Fitted values of the reweighted and/or the raw fit for all observations of xx.

Usage

```
## S3 method for class 'enetLTS'
```

```
fitted(object, vers = c("reweighted", "raw", "both"),
       type = c("response", "class"), ...)
```

Arguments

object	a fitted object of class "enetLTS".
vers	a character string: "reweighted" (the default) for the reweighted fit, "raw" for the raw fit, or "both" for both fits.
type	"response" (the default) gives the fitted values for family = "gaussian" and the fitted probabilities for "binomial" and "multinomial"; "class" gives the class with the largest fitted probability (not available for "gaussian").
...	not used.

Value

A list with the component fitted.values (vers = "reweighted"), raw.fitted.values (vers = "raw") or both (vers = "both"). Each is a one-column matrix (gaussian, binomial), a matrix with one column per class (multinomial, type = "response") or a vector of class labels (type = "class").

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

See Also

enetLTS, predict.enetLTS, residuals.enetLTS

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                              # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))            # six nonzero coefficients
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
m <- 10                                        # contaminated observations
y[1:m] <- y[1:m] + 10                          # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                      # bad leverage points

## logistic regression
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                      # observations of class 0 ...
xb[i0, ] <- xb[i0, ] + 10                      # ... become bad leverage points
```

```
## multinomial regression
set.seed(86)
n <- 120; p <- 15; K <- 3                      # observations, variables, classes
Bm <- matrix(0, p, K)
Bm[1:2, 1] <- 3; Bm[3:4, 2] <- 3; Bm[5:6, 3] <- 3
xm <- matrix(rnorm(n * p), n, p)
pr <- exp(xm %*% Bm); pr <- pr / rowSums(pr)
ym <- apply(pr, 1, function(q) sample(K, 1, prob = q))
mm <- 10
xm[1:mm, 1:2] <- xm[1:mm, 1:2] + 5             # far out in the predictors of class 1 ...
ym[1:mm] <- 3                                   # ... but labeled as class 3


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
fitm <- enetLTS(xm, ym, family = "multinomial")
head(fitted(fit)$fitted.values)
table(fitted(fitb, type = "class")$fitted.values, yb)
head(fitted(fitm, vers = "raw")$raw.fitted.values)
```

`GaussianExample` *Simulated high-dimensional linear regression data with outliers*

Description

Simulated data for `family = "gaussian"` with $n = 100$ observations and $p = 200$ predictors ($p > n$), following the simulation design of Kurnaz, Hoffmann and Filzmoser (2018).

Usage

```
data("GaussianExample")
```

Format

A list with components

`x` training predictors, a 100×200 matrix.

`y` training response, a numeric vector of length 100.

`outliers` indices of the contaminated training observations (1 to 10).

`xtest`, `ytest` clean test data with 200 observations.

`beta` the true slope vector (the true intercept is 1).

Details

The predictors form three independent blocks: two informative blocks of 10 predictors each with $\mathrm{cor}(x_j, x_k) = 0.9^{|j-k|}$, and 180 uninformative predictors with $0.2^{|j-k|}$. The response is $y = 1 + x^\top \beta + \varepsilon$ with $\varepsilon \sim N(0,1)$, $\beta_j = 1$ for the 20 informative predictors and 0 otherwise. The first 10 observations are contaminated: their informative predictors are drawn from $N(20, 1)$ and their errors from $N(20\,\hat{\sigma}_y, 1)$, where $\hat{\sigma}_y$ is the standard deviation of the clean response, so they are both leverage points and vertical outliers. Values are rounded to three decimals. The generating code is in the `data-raw` directory of the package repository on GitHub.

References

Kurnaz, F.S., Hoffmann, I. and Filzmoser, P. (2018). Robust and sparse estimation methods for high-dimensional linear and logistic regression. *Chemometrics and Intelligent Laboratory Systems*, **172**, 211–222. doi:10.1016/j.chemolab.2017.11.017

Examples

```
data("GaussianExample")
dim(GaussianExample$x)
GaussianExample$outliers
```

`lambda00` *Robust upper end of the penalty grid for logistic regression*

Description

Robust estimate of the smallest value of the penalty parameter that sets all slopes of the (lasso-penalized) logistic regression to zero. `enetLTS` uses it as the upper end of the default `lambdas` grid for `family = "binomial"`.

Usage

```
lambda00(x, y, normalize = TRUE, intercept = TRUE, const = 2, prob = 0.95,
         tol = .Machine$double.eps^0.5, eps = .Machine$double.eps, ...)
```

Arguments

`x`	a numeric matrix of predictors.
`y`	a numeric 0/1 response vector.
`normalize`	logical; whether the winsorized predictors are normalized. The default is `TRUE`.
`intercept`	logical; whether the model has an intercept. The default is `TRUE`.
`const`	tuning constant of the univariate winsorization. The default is 2.
`prob`	probability of the χ^2 quantile used in the bivariate winsorization. The default is 0.95.
`tol`	a small positive value used to detect singularity in the computation of the correlations for the bivariate winsorization.

eps	a small positive value below which a robust scale estimate is treated as zero.
...	further arguments passed to robStandardize.

Details

The procedure follows lambda0 (Alfons et al., 2013), which is used for family = "gaussian": the predictors are robustly standardized and the correlations between the response and each predictor are computed on winsorized data. For the 0/1 response of logistic regression the Pearson correlation is replaced by a robustified point-biserial correlation (Kurnaz et al., 2018).

Value

A single positive number.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

References

Kurnaz, F.S., Hoffmann, I. and Filzmoser, P. (2018). Robust and sparse estimation methods for high-dimensional linear and logistic regression. *Chemometrics and Intelligent Laboratory Systems*, **172**, 211–222. doi:10.1016/j.chemolab.2017.11.017

Alfons, A., Croux, C. and Gelper, S. (2013). Sparse least trimmed squares regression for analyzing high-dimensional large data sets. *The Annals of Applied Statistics*, **7**(1), 226–248.

See Also

enetLTS, lambda0, sparseLTS

Examples

```
data("BinomialExample")
lambda00(BinomialExample$x, BinomialExample$y)
```

MultinomialExample *Simulated high-dimensional multinomial regression data with outliers*

Description

Simulated data for family = "multinomial" with $n = 150$ observations, $p = 200$ predictors ($p > n$) and $K = 3$ classes.

Usage

```
data("MultinomialExample")
```

Format

A list with components

`x` training predictors, a 150×200 matrix.

`y` training response, a factor with levels 1, 2, 3.

`outliers` indices of the contaminated training observations (1 to 15).

`xtest`, `ytest` clean test data with 300 observations.

`beta` the true 200×3 matrix of slopes (the true intercepts are 0).

Details

The predictors are independent standard normal. Each class is driven by its own predictors: $\beta_{1,1} = \beta_{2,1} = 3$ (class 1), $\beta_{3,2} = \beta_{4,2} = 3$ (class 2), $\beta_{5,3} = \beta_{6,3} = 3$ (class 3); the class probabilities follow the multinomial logistic model. The first 15 observations are contaminated: predictors 1 and 2 are shifted by 5 (far out in the direction of class 1) and the label is set to 3, so they are bad leverage points. Values are rounded to three decimals. The generating code is in the `data-raw` directory of the package repository on GitHub.

Examples

```
data("MultinomialExample")
table(MultinomialExample$y)
```

`nonzeroCoef.enetLTS` *Indices of the nonzero coefficients of an* `"enetLTS"` *fit*

Description

Returns the positions of the nonzero coefficients of the reweighted or the raw fit.

Usage

```
nonzeroCoef.enetLTS(object, vers = c("reweighted", "raw"))
```

Arguments

`object`	a fitted object of class `"enetLTS"`.
`vers`	a character string: `"reweighted"` (the default) for the reweighted fit or `"raw"` for the raw fit.

Value

For `family = "gaussian"` and `"binomial"`, an integer vector with the positions of the nonzero entries of `coef(object, vers = vers)`. With `intercept = TRUE` position 1 is the intercept, so the slope of predictor j has position $j + 1$. For `family = "multinomial"`, a list with one such vector per class (a data frame if all classes have the same number of nonzero coefficients).

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

See Also

enetLTS, coef.enetLTS, predict.enetLTS

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                              # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))            # six nonzero coefficients
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
m <- 10                                        # contaminated observations
y[1:m] <- y[1:m] + 10                          # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                      # bad leverage points

## logistic regression
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                      # observations of class 0 ...
xb[i0, ] <- xb[i0, ] + 10                      # ... become bad leverage points

## multinomial regression
set.seed(86)
n <- 120; p <- 15; K <- 3                      # observations, variables, classes
Bm <- matrix(0, p, K)
Bm[1:2, 1] <- 3; Bm[3:4, 2] <- 3; Bm[5:6, 3] <- 3
xm <- matrix(rnorm(n * p), n, p)
pr <- exp(xm %*% Bm); pr <- pr / rowSums(pr)
ym <- apply(pr, 1, function(q) sample(K, 1, prob = q))
mm <- 10
xm[1:mm, 1:2] <- xm[1:mm, 1:2] + 5             # far out in the predictors of class 1 ...
ym[1:mm] <- 3                                   # ... but labeled as class 3


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
fitm <- enetLTS(xm, ym, family = "multinomial")
nonzeroCoef.enetLTS(fit)
nonzeroCoef.enetLTS(fitb, vers = "raw")
nonzeroCoef.enetLTS(fitm)
```

`outliers.enetLTS` *Observations flagged as outliers by an* `"enetLTS"` *fit*

Description

Returns, for every training observation, whether it has weight zero, its loss, the observed and the predicted response and a type label. The print method shows only the flagged observations.

Usage

```
outliers.enetLTS(object, vers = c("reweighted", "raw"))

## S3 method for class 'outliers.enetLTS'
print(x, ...)
```

Arguments

`object`	a fitted object of class `"enetLTS"`.
`vers`	`"reweighted"` (the default; flagged means weight 0 in `wt`) or `"raw"` (weight 0 in `raw.wt`).
`x`	an object of class `"outliers.enetLTS"`.
`...`	not used.

Value

A data frame of class `"outliers.enetLTS"` with one row per observation and the columns `index`, `flagged`, `loss` (absolute residual for `"gaussian"`, $-\log \hat{p}_{i,y_i}$ otherwise), `observed`, `predicted` and `type`.

For `"gaussian"`, `type` is `"outlier"` for flagged observations. For the classification families it is `"leverage point"` if a flagged observation is predicted in its observed class, `"possible mislabel / outlier"` if it is predicted in another class, and `"leverage point (screened)"` or `"bad leverage point (screened)"` (predicted in the observed class or not) if it was removed by the screening step of the multinomial family. Observations that are not flagged are `"regular"`.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

See Also

`enetLTS`, `weights.enetLTS`, `plotResid.enetLTS`

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                              # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))            # six nonzero coefficients
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
m <- 10                                        # contaminated observations
y[1:m] <- y[1:m] + 10                          # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                      # bad leverage points

## logistic regression
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                      # observations of class 0 ...
xb[i0, ] <- xb[i0, ] + 10                      # ... become bad leverage points

## multinomial regression
set.seed(86)
n <- 120; p <- 15; K <- 3                      # observations, variables, classes
Bm <- matrix(0, p, K)
Bm[1:2, 1] <- 3; Bm[3:4, 2] <- 3; Bm[5:6, 3] <- 3
xm <- matrix(rnorm(n * p), n, p)
pr <- exp(xm %*% Bm); pr <- pr / rowSums(pr)
ym <- apply(pr, 1, function(q) sample(K, 1, prob = q))
mm <- 10
xm[1:mm, 1:2] <- xm[1:mm, 1:2] + 5             # far out in the predictors of class 1 ...
ym[1:mm] <- 3                                   # ... but labeled as class 3


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
fitm <- enetLTS(xm, ym, family = "multinomial")
out <- outliers.enetLTS(fit)
out
## compare with the contaminated observations 1, ..., m
table(flagged = out$flagged, contaminated = out$index %in% 1:m)
outliers.enetLTS(fitb)
outliers.enetLTS(fitm)
```

---

plot.enetLTS *Plots of an* "enetLTS" *fit*

Description

Draws the coefficient, residual and diagnostic plots of a fitted "enetLTS" object.

Usage

```
## S3 method for class 'enetLTS'
plot(x, method = c("coefficients", "resid", "diagnostic"),
     vers = c("reweighted", "raw"), ...)
```

Arguments

x	a fitted object of class "enetLTS".
method	the plot to draw: "coefficients" (plotCoef.enetLTS), "resid" (plotResid.enetLTS) or "diagnostic" (plotDiagnostic.enetLTS). If method is not given, all three are drawn; in an interactive session R asks before each new page.
vers	a character string: "reweighted" (the default) for the reweighted fit or "raw" for the raw fit.
...	further arguments passed to the plot function, for example zoom for method = "resid" and method = "diagnostic".

Value

The object x, invisibly. The plots are drawn as a side effect.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

See Also

enetLTS, plotCoef.enetLTS, plotResid.enetLTS, plotDiagnostic.enetLTS

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                              # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))            # six nonzero coefficients
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
m <- 10                                        # contaminated observations
y[1:m] <- y[1:m] + 10                          # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                      # bad leverage points

## logistic regression
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                      # observations of class 0 ...
xb[i0, ] <- xb[i0, ] + 10                      # ... become bad leverage points
```

```
## multinomial regression
set.seed(86)
n <- 120; p <- 15; K <- 3                      # observations, variables, classes
Bm <- matrix(0, p, K)
Bm[1:2, 1] <- 3; Bm[3:4, 2] <- 3; Bm[5:6, 3] <- 3
xm <- matrix(rnorm(n * p), n, p)
pr <- exp(xm %*% Bm); pr <- pr / rowSums(pr)
ym <- apply(pr, 1, function(q) sample(K, 1, prob = q))
mm <- 10
xm[1:mm, 1:2] <- xm[1:mm, 1:2] + 5             # far out in the predictors of class 1 ...
ym[1:mm] <- 3                                   # ... but labeled as class 3


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
fitm <- enetLTS(xm, ym, family = "multinomial")
plot(fit, method = "resid")
plot(fitb, method = "diagnostic", vers = "raw")
plot(fitm, method = "coefficients")
```

`plotCoef.enetLTS` *Coefficient plot of an* `"enetLTS"` *fit*

Description

Needle plot of the estimated slopes against the predictor index.

Usage

```
plotCoef.enetLTS(object, vers = c("reweighted", "raw"), colors = NULL, ...)
```

Arguments

`object`	a fitted object of class `"enetLTS"`.
`vers`	a character string: `"reweighted"` (the default) for the reweighted fit or `"raw"` for the raw fit.
`colors`	not used; kept for compatibility with earlier versions.
`...`	not used.

Details

Every slope is drawn as a vertical segment from zero, and the nonzero slopes are marked by a point. The intercept (one per class for the multinomial family) is given in the subtitle, together with the number of nonzero slopes. For the multinomial family there is one panel per class.

Value

The "ggplot" object, invisibly; the plot is drawn as a side effect.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

See Also

enetLTS, coef.enetLTS, plot.enetLTS

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                              # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))            # six nonzero coefficients
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
m <- 10                                        # contaminated observations
y[1:m] <- y[1:m] + 10                          # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                      # bad leverage points

## logistic regression
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                      # observations of class 0 ...
xb[i0, ] <- xb[i0, ] + 10                      # ... become bad leverage points

## multinomial regression
set.seed(86)
n <- 120; p <- 15; K <- 3                      # observations, variables, classes
Bm <- matrix(0, p, K)
Bm[1:2, 1] <- 3; Bm[3:4, 2] <- 3; Bm[5:6, 3] <- 3
xm <- matrix(rnorm(n * p), n, p)
pr <- exp(xm %*% Bm); pr <- pr / rowSums(pr)
ym <- apply(pr, 1, function(q) sample(K, 1, prob = q))
mm <- 10
xm[1:mm, 1:2] <- xm[1:mm, 1:2] + 5             # far out in the predictors of class 1 ...
ym[1:mm] <- 3                                   # ... but labeled as class 3


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
fitm <- enetLTS(xm, ym, family = "multinomial")
plotCoef.enetLTS(fit)
plotCoef.enetLTS(fitb, vers = "raw")
```

```
plotCoef.enetLTS(fitm)
```

`plotDiagnostic.enetLTS` *Diagnostic plot of an* `"enetLTS"` *fit*

Description

Plots the response against the linear predictor (gaussian, binomial) or the first two principal components of the linear predictors (multinomial).

Usage

```
plotDiagnostic.enetLTS(object, vers = c("reweighted", "raw"), zoom = TRUE, ...)
```

Arguments

`object`	a fitted object of class `"enetLTS"`.
`vers`	a character string: `"reweighted"` (the default) for the reweighted fit or `"raw"` for the raw fit.
`zoom`	logical; if `TRUE` (the default) and the flagged observations lie far from the regular ones, a second panel shows only the range of the regular observations (and of the flagged observations close to them), so that the fit and the cut-off lines remain visible.
`...`	not used.

Details

For `family = "gaussian"` the plot shows y against $X\hat{\beta}$ with the line $y = X\hat{\beta}$. For `family = "binomial"` it shows the 0/1 response against $X\hat{\beta}$ with the fitted logistic curve. For `family = "multinomial"` it shows the scores of the first two principal components of the K linear predictors.

Observations are marked as flagged if they have weight 0 in `wt` (`vers = "reweighted"`) or if they are not in the best subset `best` of the raw fit (`vers = "raw"`). For the gaussian family regular observations are blue crosses and flagged ones red triangles. For the binomial and multinomial families the class is shown by color and shape, and flagged observations are drawn filled, regular ones open.

Value

The `"ggplot"` object, invisibly; the plot is drawn as a side effect.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

See Also

enetLTS, plotResid.enetLTS, outliers.enetLTS

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                              # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))            # six nonzero coefficients
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
m <- 10                                        # contaminated observations
y[1:m] <- y[1:m] + 10                          # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                      # bad leverage points

## logistic regression
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                      # observations of class 0 ...
xb[i0, ] <- xb[i0, ] + 10                      # ... become bad leverage points

## multinomial regression
set.seed(86)
n <- 120; p <- 15; K <- 3                      # observations, variables, classes
Bm <- matrix(0, p, K)
Bm[1:2, 1] <- 3; Bm[3:4, 2] <- 3; Bm[5:6, 3] <- 3
xm <- matrix(rnorm(n * p), n, p)
pr <- exp(xm %*% Bm); pr <- pr / rowSums(pr)
ym <- apply(pr, 1, function(q) sample(K, 1, prob = q))
mm <- 10
xm[1:mm, 1:2] <- xm[1:mm, 1:2] + 5             # far out in the predictors of class 1 ...
ym[1:mm] <- 3                                   # ... but labeled as class 3


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
fitm <- enetLTS(xm, ym, family = "multinomial")
plotDiagnostic.enetLTS(fit)
plotDiagnostic.enetLTS(fitb)
plotDiagnostic.enetLTS(fitm, zoom = FALSE)
```

---

plotResid.enetLTS *Residual plots of an* "enetLTS" *fit*

Description

Plots the residuals of the reweighted or the raw fit against the observation index and, for the gaussian and binomial families, against the linear predictor.

Usage

```
plotResid.enetLTS(object, vers = c("reweighted", "raw"), zoom = TRUE, ...)
```

Arguments

object	a fitted object of class "enetLTS".
vers	a character string: "reweighted" (the default) for the reweighted fit or "raw" for the raw fit.
zoom	logical; if TRUE (the default) and the flagged observations lie far from the regular ones, a second panel shows only the range of the regular observations (and of the flagged observations close to them), so that the fit and the cut-off lines remain visible.
...	not used.

Details

For family = "gaussian" the residuals are standardized as in the reweighting step (location and scale from the observations with raw weight 1 for the reweighted fit, from the best subset for the raw fit), and the horizontal lines at $\pm$qnorm(1 - del) are the cut-offs of the reweighting step. For family = "binomial" and "multinomial" the residuals are the negative log-likelihood contributions (see residuals.enetLTS). The multinomial family has only the index plot.

Observations are marked as flagged if they have weight 0 in wt (vers = "reweighted") or if they are not in the best subset best of the raw fit (vers = "raw"). For the gaussian family regular observations are blue crosses and flagged ones red triangles. For the binomial and multinomial families the class is shown by color and shape, and flagged observations are drawn filled, regular ones open.

Value

A list of the drawn "ggplot" objects (two for the gaussian and binomial families, one for the multinomial family), invisibly.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

See Also

enetLTS, residuals.enetLTS, plotDiagnostic.enetLTS, outliers.enetLTS

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                              # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))            # six nonzero coefficients
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
m <- 10                                        # contaminated observations
y[1:m] <- y[1:m] + 10                          # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                      # bad leverage points

## logistic regression
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                      # observations of class 0 ...
xb[i0, ] <- xb[i0, ] + 10                      # ... become bad leverage points

## multinomial regression
set.seed(86)
n <- 120; p <- 15; K <- 3                      # observations, variables, classes
Bm <- matrix(0, p, K)
Bm[1:2, 1] <- 3; Bm[3:4, 2] <- 3; Bm[5:6, 3] <- 3
xm <- matrix(rnorm(n * p), n, p)
pr <- exp(xm %*% Bm); pr <- pr / rowSums(pr)
ym <- apply(pr, 1, function(q) sample(K, 1, prob = q))
mm <- 10
xm[1:mm, 1:2] <- xm[1:mm, 1:2] + 5             # far out in the predictors of class 1 ...
ym[1:mm] <- 3                                   # ... but labeled as class 3


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
fitm <- enetLTS(xm, ym, family = "multinomial")
plotResid.enetLTS(fit)
plotResid.enetLTS(fit, vers = "raw", zoom = FALSE)
plotResid.enetLTS(fitb)
plotResid.enetLTS(fitm)
```

`predict.enetLTS` *Predictions from an* `"enetLTS"` *fit*

Description

Linear predictors, fitted values or probabilities, class labels, coefficients or the indices of the nonzero coefficients from a fitted `"enetLTS"` object.

Usage

```
## S3 method for class 'enetLTS'
predict(object, newX, vers = c("reweighted", "raw"),
        type = c("link", "response", "coefficients", "nonzero", "class"), ...)
```

Arguments

object	a fitted object of class "enetLTS".
newX	a numeric matrix (or an object that as.matrix converts to one) with the same predictors, in the same order, as xx in enetLTS. Not needed for type = "coefficients" and type = "nonzero".
vers	a character string: "reweighted" (the default) for the reweighted fit or "raw" for the raw fit.
type	"link" (the default) gives the linear predictors; "response" the fitted values (gaussian) or the probabilities (binomial, multinomial); "class" the class with the largest probability (binomial, multinomial); "coefficients" the coefficients; "nonzero" the indices of the nonzero coefficients, as nonzeroCoef.enetLTS.
...	not used.

Value

A list with one component. Its name is link, response, class, coefficients or nonzeroCoef, with the prefix raw. for vers = "raw" (for example raw.response). For the multinomial family, link and response are matrices with one column per class.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

See Also

enetLTS, coef.enetLTS, fitted.enetLTS, assess.enetLTS

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                          # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))        # six nonzero coefficients
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
xt <- matrix(rnorm(200 * p), 200, p)       # clean test data
yt <- drop(xt %*% beta) + rnorm(200)
m <- 10                                    # contaminated observations
y[1:m] <- y[1:m] + 10                      # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                  # bad leverage points

## logistic regression
```

```
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
xbt <- matrix(rnorm(200 * p), 200, p)          # clean test data
ybt <- rbinom(200, 1, plogis(drop(xbt %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                      # observations of class 0 ...
xb[i0, ] <- xb[i0, ] + 10                       # ... become bad leverage points

## multinomial regression
set.seed(86)
n <- 120; p <- 15; K <- 3                       # observations, variables, classes
Bm <- matrix(0, p, K)
Bm[1:2, 1] <- 3; Bm[3:4, 2] <- 3; Bm[5:6, 3] <- 3
xm <- matrix(rnorm(n * p), n, p)
pr <- exp(xm %*% Bm); pr <- pr / rowSums(pr)
ym <- apply(pr, 1, function(q) sample(K, 1, prob = q))
xmt <- matrix(rnorm(200 * p), 200, p)         # clean test data
pr <- exp(xmt %*% Bm); pr <- pr / rowSums(pr)
ymt <- apply(pr, 1, function(q) sample(K, 1, prob = q))
mm <- 10
xm[1:mm, 1:2] <- xm[1:mm, 1:2] + 5             # far out in the predictors of class 1 ...
ym[1:mm] <- 3                                   # ... but labeled as class 3


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
fitm <- enetLTS(xm, ym, family = "multinomial")
yhat <- predict(fit, xt, type = "response")$response
mean((yt - yhat)^2)
pb <- predict(fitb, xbt, type = "class")$class
table(predicted = pb, observed = ybt)
head(predict(fitm, xmt, type = "response")$response)
predict(fitm, xmt, vers = "raw", type = "class")$raw.class[1:10]
```

`print.enetLTS` *Print an* "enetLTS" *fit*

Description

Prints a short summary of a fitted "enetLTS" object.

Usage

```
## S3 method for class 'enetLTS'
print(x, vers = c("reweighted", "raw"),
      digits = max(3, getOption("digits") - 3), maxprint = 20, ...)
```

Arguments

x	a fitted object of class "enetLTS".
vers	a character string: "reweighted" (the default) for the reweighted fit or "raw" for the raw fit.
digits	the number of digits for the coefficients.
maxprint	the largest number of nonzero coefficients (for the multinomial family: rows) that are printed; all are available with coef.enetLTS.
...	not used.

Details

The family and the call are printed, followed by the nonzero coefficients with the variable names (x1, x2, ... if xx has no column names), the number of nonzero slopes, the selected alpha, lambda and lambdaw, and the number of observations flagged as outliers. For the multinomial family the rows of the coefficient matrix that are nonzero in at least one class are printed.

Value

The object x, invisibly.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

See Also

enetLTS, coef.enetLTS, outliers.enetLTS

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                              # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))            # six nonzero coefficients
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
m <- 10                                        # contaminated observations
y[1:m] <- y[1:m] + 10                          # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                      # bad leverage points

## logistic regression
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                      # observations of class 0 ...
```

```
xb[i0, ] <- xb[i0, ] + 10                      # ... become bad leverage points

## multinomial regression
set.seed(86)
n <- 120; p <- 15; K <- 3                      # observations, variables, classes
Bm <- matrix(0, p, K)
Bm[1:2, 1] <- 3; Bm[3:4, 2] <- 3; Bm[5:6, 3] <- 3
xm <- matrix(rnorm(n * p), n, p)
pr <- exp(xm %*% Bm); pr <- pr / rowSums(pr)
ym <- apply(pr, 1, function(q) sample(K, 1, prob = q))
mm <- 10
xm[1:mm, 1:2] <- xm[1:mm, 1:2] + 5             # far out in the predictors of class 1 ...
ym[1:mm] <- 3                                   # ... but labeled as class 3


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
fitm <- enetLTS(xm, ym, family = "multinomial")
fit
print(fit, vers = "raw")
fitb
fitm
```

`residuals.enetLTS` *Residuals of an* `"enetLTS"` *fit*

Description

Residuals of the reweighted and/or the raw fit for all observations of `xx`.

Usage

```
## S3 method for class 'enetLTS'
residuals(object, vers = c("reweighted", "raw", "both"), ...)
```

Arguments

`object`	a fitted object of class `"enetLTS"`.
`vers`	a character string: `"reweighted"` (the default) for the reweighted fit, `"raw"` for the raw fit, or `"both"` for both fits.
`...`	not used.

Details

For `family = "gaussian"` the residuals are $y_i - \hat{y}_i$. For `family = "binomial"` and `"multinomial"` they are the negative log-likelihood contributions of the observations, $-y_i\eta_i + \log(1 + e^{\eta_i})$ and $-\log \hat{p}_{i,y_i}$, respectively (η_i is the linear predictor and $\hat{p}_{i,y_i}$ the fitted probability of the observed class).

Value

A list with the component `residuals` (`vers = "reweighted"`), `raw.residuals` (`vers = "raw"`) or both (`vers = "both"`); each is a numeric vector.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

See Also

`enetLTS`, `fitted.enetLTS`, `plotResid.enetLTS`, `outliers.enetLTS`

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                              # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))            # six nonzero coefficients
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
m <- 10                                        # contaminated observations
y[1:m] <- y[1:m] + 10                          # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                      # bad leverage points

## logistic regression
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                      # observations of class 0 ...
xb[i0, ] <- xb[i0, ] + 10                      # ... become bad leverage points

## multinomial regression
set.seed(86)
n <- 120; p <- 15; K <- 3                      # observations, variables, classes
Bm <- matrix(0, p, K)
Bm[1:2, 1] <- 3; Bm[3:4, 2] <- 3; Bm[5:6, 3] <- 3
xm <- matrix(rnorm(n * p), n, p)
pr <- exp(xm %*% Bm); pr <- pr / rowSums(pr)
ym <- apply(pr, 1, function(q) sample(K, 1, prob = q))
mm <- 10
xm[1:mm, 1:2] <- xm[1:mm, 1:2] + 5             # far out in the predictors of class 1 ...
ym[1:mm] <- 3                                   # ... but labeled as class 3


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
fitm <- enetLTS(xm, ym, family = "multinomial")
head(residuals(fit)$residuals)
```

```
str(residuals(fitb, vers = "both"))
summary(residuals(fitm)$residuals)
```

`weights.enetLTS` *Binary outlier weights of an* `"enetLTS"` *fit*

Description

Extracts the 0/1 weights that mark the outliers of the reweighted and/or the raw fit.

Usage

```
## S3 method for class 'enetLTS'
weights(object, vers = c("reweighted", "raw", "both"), index = FALSE, ...)
```

Arguments

`object`	a fitted object of class `"enetLTS"`.
`vers`	a character string: `"reweighted"` (the default) for the reweighted fit, `"raw"` for the raw fit, or `"both"` for both fits.
`index`	logical; if `TRUE`, the weights are named by the observation numbers. The default is `FALSE`.
`...`	not used.

Details

An observation has weight 1 if it is regular and 0 if it is flagged as an outlier. The raw weights `raw.wt` come from the raw fit and define the observations used for the reweighted fit; the weights `wt` come from the reweighted fit. With `del` the argument of `enetLTS`, the rules are:

gaussian standardized residuals (location and scale estimated from the observations with raw weight 1, with a consistency factor) larger than `qnorm(1 - del)` in absolute value get weight 0.

binomial Pearson residuals larger than `qnorm(1 - del)` in absolute value get weight 0.

multinomial with `crossfit = TRUE` (the default) an observation gets weight 0 if its cross-fitted probability of the observed class is below a calibrated cut-off (see `crossfit` in `enetLTS`); with `crossfit = FALSE` it gets weight 0 if it is outlying in the space of the linear predictors (class-wise robust distances) or if the fitted probability of its observed class is below 1/6. Observations removed by `screen = TRUE` have weight 0 in both vectors.

Value

A list with the component `wt` (`vers = "reweighted"`), `raw.wt` (`vers = "raw"`) or both (`vers = "both"`); each is an integer vector of length n.

Author(s)

Fatma Sevinc Kurnaz, Irene Hoffmann, Peter Filzmoser
Maintainer: Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

See Also

enetLTS, outliers.enetLTS

Examples

```
## linear regression
set.seed(86)
n <- 100; p <- 25                              # observations and variables
beta <- c(rep(1, 6), rep(0, p - 6))            # six nonzero coefficients
x <- matrix(rnorm(n * p), n, p)
y <- drop(x %*% beta) + rnorm(n)
m <- 10                                        # contaminated observations
y[1:m] <- y[1:m] + 10                          # vertical outliers
x[1:m, ] <- x[1:m, ] + 10                      # bad leverage points

## logistic regression
set.seed(86)
n <- 100; p <- 25
beta <- c(rep(1, 6), rep(0, p - 6))
xb <- matrix(rnorm(n * p), n, p)
yb <- rbinom(n, 1, plogis(drop(xb %*% beta)))
mb <- 5
i0 <- which(yb == 0)[1:mb]                      # observations of class 0 ...
xb[i0, ] <- xb[i0, ] + 10                      # ... become bad leverage points

## multinomial regression
set.seed(86)
n <- 120; p <- 15; K <- 3                      # observations, variables, classes
Bm <- matrix(0, p, K)
Bm[1:2, 1] <- 3; Bm[3:4, 2] <- 3; Bm[5:6, 3] <- 3
xm <- matrix(rnorm(n * p), n, p)
pr <- exp(xm %*% Bm); pr <- pr / rowSums(pr)
ym <- apply(pr, 1, function(q) sample(K, 1, prob = q))
mm <- 10
xm[1:mm, 1:2] <- xm[1:mm, 1:2] + 5             # far out in the predictors of class 1 ...
ym[1:mm] <- 3                                   # ... but labeled as class 3


fit  <- enetLTS(x, y)
fitb <- enetLTS(xb, yb, family = "binomial")
fitm <- enetLTS(xm, ym, family = "multinomial")
which(weights(fit)$wt == 0)
which(weights(fitb)$wt == 0)
w <- weights(fitm, vers = "both")
table(raw = w$raw.wt, reweighted = w$wt)
```

Index

∗ **classif**
  assess.enetLTS, 2
  enetLTS, 8
∗ **datasets**
  BinomialExample, 4
  GaussianExample, 15
  MultinomialExample, 17
∗ **hplot**
  plot.enetLTS, 21
  plotCoef.enetLTS, 23
  plotDiagnostic.enetLTS, 25
  plotResid.enetLTS, 26
∗ **models**
  assess.enetLTS, 2
  cv.enetLTS, 7
  print.enetLTS, 30
∗ **regression**
  coef.enetLTS, 5
  cv.enetLTS, 7
  enetLTS, 8
  fitted.enetLTS, 13
  lambda00, 16
  nonzeroCoef.enetLTS, 18
  outliers.enetLTS, 20
  predict.enetLTS, 28
  print.enetLTS, 30
  residuals.enetLTS, 32
  weights.enetLTS, 34
∗ **robust**
  enetLTS, 8
  lambda00, 16
  outliers.enetLTS, 20

assess.enetLTS, 2, *12*, *29*

BinomialExample, 4, *12*

coef.enetLTS, 5, *11, 12*, *19*, *24*, *29*, *31*
confusion.enetLTS *(*assess.enetLTS*)*, 2
cv.enetLTS, 7, *11, 12*
cv.glmnet, *9*

enetLTS, *3*, *6, 7*, 8, *8*, *14*, *16, 17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33–35*
enetLTS-package *(*enetLTS*)*, 8

fitted.enetLTS, 13, *29*, *33*

GaussianExample, *12*, 15

lambda0, *9*, *17*
lambda00, *9*, 16

MultinomialExample, *12*, 17

nonzeroCoef.enetLTS, *6*, 18, *29*

outliers.enetLTS, *3*, *12*, 20, *26, 27*, *31*, *33*, *35*

plot.enetLTS, *12*, 21, *24*
plotCoef.enetLTS, *22*, 23
plotDiagnostic.enetLTS, *22*, 25, *27*
plotResid.enetLTS, *20*, *22*, 26, *26*, *33*
predict.enetLTS, *3*, *6*, *12*, *14*, *19*, 28
print.confusion.enetLTS *(*assess.enetLTS*)*, 2
print.enetLTS, *6*, *12*, 30
print.outliers.enetLTS *(*outliers.enetLTS*)*, 20

residuals.enetLTS, *14*, *27*, 32
robStandardize, *17*
roc.enetLTS *(*assess.enetLTS*)*, 2

sparseLTS, *17*

weights.enetLTS, *11*, *20*, 34