Package 'enetLTS'

October 13, 2022

Type Package

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

Version 1.1.0

Author Fatma Sevinc Kurnaz and Irene Hoffmann and Peter Filzmoser

Maintainer Fatma Sevinc Kurnaz <fatmasevinckurnaz@gmail.com>

Description
Fully robust versions of the elastic net estimator are introduced for linear and binary and multinomial regression, in particular high dimensional data. The algorithm searches for outlier free subsets on which the classical elastic net estimators can be applied. A reweighting step is added to improve the statistical efficiency of the proposed estimators. Selecting appropriate tuning parameters for elastic net penalties are done via cross-validation.

Imports ggplot2, glmnet, grid, reshape, parallel, cvTools, stats, robustbase, robustHD

License GPL (>= 3)

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2022-05-21 23:00:14 UTC

R topics documented:

`coef.enetLTS` *coefficients from the* `enetLTS` *object*

Description

Extracts model coefficients from object returned by regression model.

Usage

```
  ## S3 method for class 'enetLTS'
coef(object,vers,zeros,...)
```

Arguments

`object`	fitted `enetLTS` model object.
`vers`	a character string specifying for which fit to make predictions. Possible values are `reweighted` (the default) for predicting values from the reweighted fit, `raw` for predicting values from the raw fit.
`zeros`	a logical indicating whether to give nonzero coefficients indices. (`TRUE`, the default) or to omit them (`FALSE`).
`...`	additional arguments from the `enetLTS` object if needed.

Value

a numeric vector (or a list object for family="multinomial") containing the requested coefficients.

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER
Maintainer: Fatma Sevinc KURNAZ <fatmasevinckurnaz@gmail.com>;<fskurnaz@yildiz.edu.tr>

See Also

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

Examples

```
## for gaussian

set.seed(86)
n <- 100; p <- 25                              # number of observations and variables
beta <- rep(0,p); beta[1:6] <- 1               # 10% nonzero coefficients
sigma <- 0.5                                   # controls signal-to-noise ratio
x <- matrix(rnorm(n*p, sigma),nrow=n)
e <- rnorm(n,0,1)                              # error terms
eps <- 0.1                                     # contamination level
m <- ceiling(eps*n)                            # observations to be contaminated
eout <- e; eout[1:m] <- eout[1:m] + 10         # vertical outliers
yout <- c(x %*% beta + sigma * eout)         # response
xout <- x; xout[1:m,] <- xout[1:m,] + 10       # bad leverage points


fit1 <- enetLTS(xout,yout)
coef(fit1)
coef(fit1,vers="raw")
coef(fit1,vers="reweighted",zeros=FALSE)



## for binomial

eps <-0.05                                       # %10 contamination to only class 0
m <- ceiling(eps*n)
y <- sample(0:1,n,replace=TRUE)
xout <- x
xout[y==0,][1:m,] <- xout[1:m,] + 10;            # class 0
yout <- y                                      # wrong classification for vertical outliers


fit2 <- enetLTS(xout,yout,family="binomial")
coef(fit2)
coef(fit2,vers="reweighted")
coef(fit2,vers="raw",zeros=FALSE)



## for multinomial

n <- 120; p <- 15
NC <- 3                # number of groups
X <- matrix(rnorm(n * p), n, p)
betas <- matrix(1:NC, ncol=NC, nrow=p, byrow=TRUE)
betas[(p-5):p,]=0; betas <- rbind(rep(0,NC),betas)
lv <- cbind(1,X)%*%betas
probs <- exp(lv)/apply(exp(lv),1,sum)
y <- apply(probs,1,function(prob){sample(1:NC, 1, TRUE, prob)})
xout <- X
eps <-0.05                          # %10 contamination to only class 0
```

```
m <- ceiling(eps*n)
xout[1:m,] <- xout[1:m,] + 10       # bad leverage points
yout <- y


fit3    <- enetLTS(xout,yout,family="multinomial")
coef(fit3)
coef(fit3,vers="reweighted")
coef(fit3,vers="raw",zeros=FALSE)
```

---

cv.enetLTS *Cross-validation for the* enetLTS *object*

Description

Does k-fold cross-validation for enetLTS, produces a plot, and returns optimal values for alpha and lambda. Combine the cross-validation functions internally used in the algorithm enetLTS.

Usage

```
cv.enetLTS(index=NULL,family,xx,yy,alphas,lambdas,nfold,repl,ncores,plot=TRUE)
```

Arguments

index	A user supplied index. The default is NULL in the algorithm enetLTS.
family	a description of the error distribution and link function to be used in the model. "gaussian" and "binomial" and "multinomial" options are available.
xx	matrix xx as in enetLTS.
yy	response yy as in enetLTS.
alphas	a user supplied alpha sequence for the elastic net penalty, which is the mixing proportion of the ridge and lasso penalties and takes value in [0,1]. Here $\alpha = 1$ is the lasso penalty, and $\alpha = 0$ the ridge penalty.
lambdas	a user supplied lambda sequence for the strength of the elastic net penalty.
nfold	a user supplied numeric value for fold number of k-fold cross-validation which used in varied functions of the algorithm. The default is 5-fold cross-validation.
repl	a user supplied posiitive number for more stable results, repeat the k-fold CV repl times and take the average of the corresponding evaluation measure. The default is 5.
ncores	a positive integer giving the number of processor cores to be used for parallel computing. The default is 4.
plot	a logical indicating if produces a plot for k-fold cross-validation based on alpha and lambda combinations. The default is TRUE.

Value

produces a plot, and returns optimal values for `alpha` and `lambda`

Note

This is an internal function. But, it is also available for direct usage to obtain optimal values of alpha and lambda for user supplied index set.

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER
Maintainer: Fatma Sevinc KURNAZ <fskurnaz@gmail.com>;<fskurnaz@yildiz.edu.tr>

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

Description

Compute fully robust versions of the elastic net estimator, which allows for sparse model estimates, for linear regression and binary and multinomial logistic regression.

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")
  )
```

Arguments

xx	a numeric matrix containing the predictor variables.
yy	response variable. Quantitative for family="gaussian". For family="binomial" should be a factor with two levels which is coded as 0 and 1. For family="multinomial" should be a factor with the number of categories (NC) which is coded as 1,2,...,NC.
family	a description of the error distribution and link function to be used in the model. "gaussian", "binomial" and family="multinomial" options are available.
alphas	a user supplied alpha sequence for the elastic net penalty, which is the mixing proportion of the ridge and lasso penalties and takes value in [0,1]. $\alpha = 1$ is the lasso penalty, and $\alpha = 0$ the ridge penalty. If not provided a sequence, default is 41 equally spaced values.
lambdas	a user supplied lambda sequence for the strength of the elastic net penalty. If not provided a sequence, default is chosen with steps of size -0.025 lambda0 with $0 \leq \lambda \leq lambda0$ for linear regression and -0.025 lambda00 with $0 \leq \lambda \leq lambda00$ for binary logistic regression. lambda0 is determined based on the Pearson correlation between y and the jth predictor variable x_j on winsorized data for linear regression. In lambda00 for logistic regression, the Pearson correlation is replaced by a robustified point-biserial correlation. Default is chosen with steps of size -0.05 from 0.95 to 0.05 for multinomial logistic regression.
lambdaw	a user supplied lambda sequence for reweighting step. If not provided, default is computed by using k-fold cross-validation via cv.glmnet function.
intercept	a logical indicating whether a constant term should be included in the model (the default is TRUE).
scal	a logical value indicating whether scale the predictors by their arithmetic means and standard deviations. For family="gaussian", it also indicates if mean-center the response variable or not. The default is TRUE. Note that scaling is performed on the subsamples rather than the full data set.
hsize	a user supplied numeric value giving the percentage of the residuals for which the elastic net penalized sum of squares for linear regression or for which the elastic net penalized sum of deviances for binary and multinomial logistic regression should be minimized. The default is 0.75.
nsamp	a numeric vector giving the number of subsamples to be used in the beginning of the algorithm, which gives the number of initial subsamples to be used. The default is to first perform C-steps on 500 initial subsamples, and then to keep the s1 subsamples with the lowest value (or highest value based on which model is used - family="gaussian" or family="binomial" or family="multinomial") of the objective function for additional C-steps until convergence.
nCsteps	a positive integer giving the number of C-steps to perform on determined s1 subsamples. The default is 20.
nfold	a user supplied numeric value for fold number of k-fold cross-validation which used in varied functions of the algorithm. The default is 5-fold cross-validation.
repl	a user supplied positive number for more stable results, repeat the k-fold CV repl times and take the average of the corresponding evaluation measure. The default is 1.

ncores	a positive integer giving the number of processor cores to be used for parallel computing (the default is 1 for no parallelization). If this is set to NA, all available processor cores are used. For prediction error estimation, parallel computing is implemented on the R level using package **parallel**.
tol	a small numeric value for convergence. The default is -1e6.
seed	optional initial seed for the random number generator (see .Random.seed) when determine initial subsets at thebeginning of the algorithm. The default is NULL.
del	The default is 0.0125.
crit.plot	a logical value indicating if produces a plot for k-fold cross-validation based on alpha and lambda combinations. The default is TRUE.
typegrouped	This argument is available for only family="multinomial" in the last fit based on the best subset. TRUE means "grouped" and FALSE means "ungrouped". If "TRUE" then a grouped lasso penalty is used on the multinomial coefficients for a variable. The default is FALSE.
type.response	type of prediction required. type="link" gives the linear predictors. type="response" gives the fitted probabilities for family="multinomial" and family="binomial" and gives the fitted values for family="gaussian". type="class" is available only for family="binomial" and family="multinomial", and produces the class label corresponding to the maximum probability.

Details

The idea of repeatedly applying the non-robust classical elastic net estimators to data subsets only is used for linear and logistic regression. The algorithm starts with 500 elemental subsets only for one combination of α and λ, and takes the warm start strategy for subsequent combinations. This idea saves the computation time. To choose the elastic net penalties, k-fold cross-validation is used and the replication option is provided for more stable results. Robustness has been achieved by using trimming idea, therefore a reweighting step is introduced in order to improve the efficiency. The outliers are identified according to current model. For family="gaussian", standardized residuals are used. For family="binomial", the Pearson residuals which are approximately standard normally distributed is used. Then the weights are defined by the binary weight function using del=0.0125, which allows to be flagged as outliers of the 2.5% of the observations in the normal model. For family="multinomial", group-wise scaled robust distances are used. The the binary weights defined using the constant $c_2=5$. Therefore, binary weight function produces a clear distinction between the "good observations" and "outliers".

Value

objective	a numeric vector giving the respective values of the enetLTS objective function, i.e., the elastic net penalized sums of the h smallest squared residuals from the raw fits for family="gaussian" and the elastic net penalized sums of the h deviances from the raw fits for family="binomial".
raw.rmse	root mean squared error for raw fit, which is available for only family="gaussian".
rmse	root mean squared error for reweighted fit, which is available for only family ="gaussian".
raw.mae	mean absolute error for raw fit.

`mae`	mean absolute error for reweighted fit.
`best`	an integer vector containing the respective best subsets of h observations found and used for computing the raw estimates.
`raw.wt`	an integer vector containing binary weights that indicate outliers from the respective raw fits, i.e., the weights used for the reweighted fits.
`wt`	an integer vector containing binary weights that indicate outliers from the respective reweighted fits, i.e., the weights are 1 for observations with reasonably small reweighted residuals and 0 for observations with large reweighted residuals.
`raw.coefficients`	a numeric vector containing the respective coefficient estimates from the raw fit.
`coefficients`	a numeric vector containing the respective coefficient estimates from the reweighted fit.
`raw.fitted.values`	a numeric vector containing the respective fitted values of the response from the raw fits.
`fitted.values`	a numeric vector containing the respective fitted values of the response from the reweighted fits.
`raw.residuals`	a numeric vector containing the respective residuals for `family="gaussian"` and respective deviances for `family="binomial"` and `family="multinomial"` from the raw fits.
`residuals`	a numeric vector containing the respective residuals for `family="gaussian"` and respective deviances for `family="binomial"` and `family="multinomial"` from the reweighted fits.
`alpha`	an optimal elastic net mixing parameter value obtained with k-fold cross-validation.
`lambda`	an optimal value for the strength of the elastic net penalty obtained with k-fold cross-validation.
`lambdaw`	an optimal value for the strength of the elastic net penalty re-obtained with k-fold cross-validation for reweighted fit.
`num.nonzerocoef`	the number of the nonzero coefficients in the model.
`n`	the number of observations.
`p`	the number of variables.
`h`	the number of observations used to compute the raw estimates.
`classnames`	class names for logistic model, which is available for only `family="binomial"` and `family="multinomial"`.
`classize`	class sizes for logisitic model, which is available for only `family="binomial"` and `family="multinomial"`.
`inputs`	all inputs used in the function `enetLTS.R`.
`call`	the matched function call.

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER

References

Kurnaz, F.S., Hoffmann, I. and Filzmoser, P. (2017) Robust and sparse estimation methods for high dimensional linear and logistic regression. *Chemometrics and Intelligent Laboratory Systems.*

See Also

print, predict, coef, nonzeroCoef.enetLTS, plot, plotCoef.enetLTS, plotResid.enetLTS, plotDiagnostic.enetLTS, residuals, fitted, weights

Examples

```
## for gaussian

set.seed(86)
n <- 100; p <- 25                              # number of observations and variables
beta <- rep(0,p); beta[1:6] <- 1               # 10% nonzero coefficients
sigma <- 0.5                                   # controls signal-to-noise ratio
x <- matrix(rnorm(n*p, sigma),nrow=n)
e <- rnorm(n,0,1)                              # error terms
eps <- 0.1                                     # contamination level
m <- ceiling(eps*n)                            # observations to be contaminated
eout <- e; eout[1:m] <- eout[1:m] + 10         # vertical outliers
yout <- c(x %*% beta + sigma * eout)         # response
xout <- x; xout[1:m,] <- xout[1:m,] + 10       # bad leverage points


# determine user supplied alpha and lambda sequences
# alphas=seq(0,1,length=11)
# l0 <- robustHD::lambda0(xout,yout)          # use lambda0 function from robustHD package
# lambdas <- seq(l0,0,by=-0.1*l0)
# fit <- enetLTS(xout,yout,alphas=alphas,lambdas=lambdas)


## for binomial

eps <-0.05                                       # %10 contamination to only class 0
m <- ceiling(eps*n)
y <- sample(0:1,n,replace=TRUE)
xout <- x
xout[y==0,][1:m,] <- xout[1:m,] + 10;            # class 0
yout <- y                                      # wrong classification for vertical outliers


# determine user supplied alpha and lambda sequences
# alphas=seq(0,1,length=11)
# l00 <- lambda00(xout,yout,normalize=TRUE,intercept=TRUE)
# lambdas <-  seq(l00,0,by=-0.01*l00)
# fit <- enetLTS(xout,yout,family="binomial",alphas=alphas,lambdas=lambdas)


## for multinomial
```

```
n <- 120; p <- 15
NC <- 3           # number of groups
X <- matrix(rnorm(n * p), n, p)
betas <- matrix(1:NC, ncol=NC, nrow=p, byrow=TRUE)
betas[(p-5):p,]=0; betas <- rbind(rep(0,NC),betas)
lv <- cbind(1,X) %*% betas
probs <- exp(lv)/apply(exp(lv),1,sum)
y <- apply(probs,1,function(prob){sample(1:NC, 1, TRUE, prob)})
xout <- X
eps <-0.05                          # %10 contamination to only class 0
m <- ceiling(eps*n)
xout[1:m,] <- xout[1:m,] + 10       # bad leverage points
yout <- y


# determine user supplied alpha and lambda sequences
alphas=seq(0,1,length=11)
lambdas <- seq(from=0.95,to=0.05,by=-0.05)
fit <- enetLTS(xout,yout,family="multinomial",alphas=alphas,lambdas=lambdas)
```

`fitted.enetLTS` *the fitted values from the* `"enetLTS"` *object.*

Description

A numeric vector which extract fitted values from the current model.

Usage

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

Arguments

`object`	the model fit from which to extract fitted values.
`vers`	a character string specifying for which fit to make predictions. Possible values are `"reweighted"` (the default) for predicting values from the reweighted fit, `"raw"` for predicting values from the raw fit, or `"both"` for predicting values from both fits.
`type`	type of prediction required. `type="response"` gives the fitted probabilities for `"multinomial"` and `"binomial"` and gives the fitted values for `"gaussian"`. `type="class"` is available only for `"multinomial"` and `"binomial"` model, and produces the class label corresponding to the maximum probability.
`...`	additional arguments from the `enetLTS` object if needed.

Value

A numeric vector containing the requested fitted values.

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER
Maintainer: Fatma Sevinc KURNAZ <fskurnaz@gmail.com>;<fskurnaz@yildiz.edu.tr>

See Also

enetLTS, predict.enetLTS, residuals.enetLTS

Examples

```
## for gaussian

set.seed(86)
n <- 100; p <- 25                              # number of observations and variables
beta <- rep(0,p); beta[1:6] <- 1               # 10% nonzero coefficients
sigma <- 0.5                                   # controls signal-to-noise ratio
x <- matrix(rnorm(n*p, sigma),nrow=n)
e <- rnorm(n,0,1)                              # error terms
eps <- 0.1                                     # contamination level
m <- ceiling(eps*n)                            # observations to be contaminated
eout <- e; eout[1:m] <- eout[1:m] + 10         # vertical outliers
yout <- c(x %*% beta + sigma * eout)         # response
xout <- x; xout[1:m,] <- xout[1:m,] + 10       # bad leverage points


fit1 <- enetLTS(xout,yout,crit.plot=FALSE)
fitted(fit1)
fitted(fit1,vers="raw")
fitted(fit1,vers="both")
fitted(fit1,vers="reweighted",type="response")


## for binomial
eps <-0.05                                       # %10 contamination to only class 0
m <- ceiling(eps*n)
y <- sample(0:1,n,replace=TRUE)
xout <- x
xout[y==0,][1:m,] <- xout[1:m,] + 10;            # class 0
yout <- y                                      # wrong classification for vertical outliers



fit2 <- enetLTS(xout,yout,family="binomial")
fitted(fit2)
fitted(fit2,vers="raw")
fitted(fit2,vers="both",type="class")
fitted(fit2,vers="both")
fitted(fit2,vers="reweighted",type="class")
```

```
## for multinomial

n <- 120; p <- 15
NC <- 3
X <- matrix(rnorm(n * p), n, p)
betas <- matrix(1:NC, ncol=NC, nrow=p, byrow=TRUE)
betas[(p-5):p,]=0; betas <- rbind(rep(0,NC),betas)
lv <- cbind(1,X) %*% betas
probs <- exp(lv)/apply(exp(lv),1,sum)
y <- apply(probs,1,function(prob){sample(1:NC, 1, TRUE, prob)})
xout <- X
eps <-0.05                          # %10 contamination to only class 0
m <- ceiling(eps*n)
xout[1:m,] <- xout[1:m,] + 10       # bad leverage points
yout <- y


fit3 <- enetLTS(xout,yout,family="multinomial")
fitted(fit3)
fitted(fit3,vers="raw")
fitted(fit3,vers="both",type="class")
fitted(fit3,vers="both")
fitted(fit3,vers="reweighted",type="class")
```

`lambda00` *Upper limit of the penalty parameter for* `family="binomial"`

Description

Use bivariate winsorization to estimate the smallest value of the upper limit for the penalty parameter.

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 containing the predictor variables.
`y`	a numeric vector containing the response variable.
`normalize`	a logical indicating whether the winsorized predictor variables should be normalized or not (the default is `TRUE`).
`intercept`	a logical indicating whether a constant term should be included in the model (the default is `TRUE`).
`const`	numeric; tuning constant to be used in univariate winsorization (the default is 2).

prob	numeric; probability for the quantile of the χ^2 distribution to be used in bivariate winsorization (the default is 0.95).
tol	a small positive numeric value used to determine singularity issues in the computation of correlation estimates for bivariate winsorization.
eps	a small positive numeric value used to determine whether the robust scale estimate of a variable is too small (an effective zero).
...	additional arguments if needed.

Details

The estimation procedure is done with similar approach as in Alfons et al. (2013). But the Pearson correlation between y and the jth predictor variable xj on winsorized data is replaced to a robustified point-biserial correlation for logistic regression.

Value

A robust estimate of the smallest value of the penalty parameter for enetLTS regression (for `family="binomial"`).

Note

For linear regression, we take exactly same procedure as in Alfons et al., which is based on the Pearson correlation between y and the jth predictor variable xj on winsorized data. See Alfons et al. (2013).

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER
Maintainer: Fatma Sevinc KURNAZ <fatmasevinckurnaz@gmail.com>;<fskurnaz@yildiz.edu.tr>

References

Kurnaz, F.S., Hoffmann, I. and Filzmoser, P. (2017) Robust and sparse estimation methods for high dimensional linear and logistic regression. *Chemometrics and Intelligent Laboratory Systems.*

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`, `sparseLTS`, `lambda0`

Examples

```
set.seed(86)
n <- 100; p <- 25                          # number of observations and variables
beta <- rep(0,p); beta[1:6] <- 1           # 10% nonzero coefficients
sigma <- 0.5                               # controls signal-to-noise ratio
x <- matrix(rnorm(n*p, sigma),nrow=n)
e <- rnorm(n,0,1)                          # error terms
eps <-0.05                                 # %10 contamination to only class 0
```

```
m <- ceiling(eps*n)
y <- sample(0:1,n,replace=TRUE)
xout <- x
xout[y==0,][1:m,] <- xout[1:m,] + 10;         # class 0
yout <- y                                    # wrong classification for vertical outliers

# compute smallest value of the upper limit for the penalty parameter
l00 <- lambda00(xout,yout)
```

`nonzeroCoef.enetLTS` *nonzero coefficients indices from the* `"enetLTS"` *object*

Description

A numeric vector which gives the indices of nonzero coefficients from the current model.

Usage

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

Arguments

`object`	the model fit from which to extract nonzero coefficients indices.
`vers`	a character string denoting which model to use. Possible values are `"reweighted"` (the default) for plots from the reweighted fit, and `"raw"` for plots from the raw fit.

Value

A numeric vector (or a list object for family="multinomial") containing the request.

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER
Maintainer: Fatma Sevinc KURNAZ <fatmasevinckurnaz@gmail.com>;<fskurnaz@yildiz.edu.tr>

See Also

`enetLTS`, `predict.enetLTS`, `coef.enetLTS`

Examples

```
## for gaussian

set.seed(86)
n <- 100; p <- 25                            # number of observations and variables
beta <- rep(0,p); beta[1:6] <- 1             # 10% nonzero coefficients
sigma <- 0.5                                 # controls signal-to-noise ratio
x <- matrix(rnorm(n*p, sigma),nrow=n)
```

```
e <- rnorm(n,0,1)                              # error terms
eps <- 0.1                                     # contamination level
m <- ceiling(eps*n)                            # observations to be contaminated
eout <- e; eout[1:m] <- eout[1:m] + 10         # vertical outliers
yout <- c(x %*% beta + sigma * eout)         # response
xout <- x; xout[1:m,] <- xout[1:m,] + 10       # bad leverage points


fit1 <- enetLTS(xout,yout)
nonzeroCoef.enetLTS(fit1)
nonzeroCoef.enetLTS(fit1,vers="raw")


## for binomial

eps <-0.05                                       # %10 contamination to only class 0
m <- ceiling(eps*n)
y <- sample(0:1,n,replace=TRUE)
xout <- x
xout[y==0,][1:m,] <- xout[1:m,] + 10;           # class 0
yout <- y                                      # wrong classification for vertical outliers


fit2 <- enetLTS(xout,yout,family="binomial")
nonzeroCoef.enetLTS(fit2)
nonzeroCoef.enetLTS(fit2,vers="raw")



## for multinomial

n <- 120; p <- 15
NC <- 3
X <- matrix(rnorm(n * p), n, p)
betas <- matrix(1:NC, ncol=NC, nrow=p, byrow=TRUE)
betas[(p-5):p,]=0; betas <- rbind(rep(0,NC),betas)
lv <- cbind(1,X) %*% betas
probs <- exp(lv)/apply(exp(lv),1,sum)
y <- apply(probs,1,function(prob){sample(1:NC, 1, TRUE, prob)})
xout <- X
eps <-0.05                          # %10 contamination to only class 0
m <- ceiling(eps*n)
xout[1:m,] <- xout[1:m,] + 10       # bad leverage points
yout <- y


fit3    <- enetLTS(xout,yout,family="multinomial")
nonzeroCoef.enetLTS(fit3)
nonzeroCoef.enetLTS(fit3,vers="raw")
```

plot.enetLTS *plots from the* "enetLTS" *object*

Description

Produce plots for the coefficients, residuals, and diagnostics of the current model.

Usage

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

Arguments

x	object of class enetLTS, the model fit to be plotted.
method	a character string specifying the type of plot. Possible values are "coefficients" to plot the coefficients via plotCoef.enetLTS, "resid" to plot the residuals via plotResid.enetLTS, or "diagnostic" for diagnostic plot via plotDiagnostic.enetLTS.
vers	a character string denoting which model to use for the plots. Possible values are "reweighted" (the default) for plots from the reweighted fit, and "raw" for plots from the raw fit.
...	additional arguments from the enetLTS object if needed.

Value

An object of class "ggplot" (see ggplot).

Note

For method, the choices are:

method="coefficients" - coefficients vs indices.

method="resid" - residuals vs indices. (for both family="binomial" and family="gaussian").

- additionally, residuals vs fitted values (for only family="gaussian").

method="diagnostics" - fitted values vs indices.

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER
Maintainer: Fatma Sevinc KURNAZ <fatmasevinckurnaz@gmail.com>;<fskurnaz@yildiz.edu.tr>

References

Kurnaz, F.S., Hoffmann, I. and Filzmoser, P. (2017) Robust and sparse estimation methods for high dimensional linear and logistic regression. *Chemometrics and Intelligent Laboratory Systems*.

See Also

ggplot, enetLTS, coef.enetLTS, predict.enetLTS, residuals.enetLTS, fitted.enetLTS

Examples

```
## for gaussian

set.seed(86)
n <- 100; p <- 25                              # number of observations and variables
beta <- rep(0,p); beta[1:6] <- 1               # 10% nonzero coefficients
sigma <- 0.5                                   # controls signal-to-noise ratio
x <- matrix(rnorm(n*p, sigma),nrow=n)
e <- rnorm(n,0,1)                              # error terms
eps <- 0.1                                     # contamination level
m <- ceiling(eps*n)                            # observations to be contaminated
eout <- e; eout[1:m] <- eout[1:m] + 10         # vertical outliers
yout <- c(x %*% beta + sigma * eout)           # response
xout <- x; xout[1:m,] <- xout[1:m,] + 10       # bad leverage points


fit1 <- enetLTS(xout,yout,crit.plot=FALSE)
plot(fit1)
plot(fit1,method="resid",vers="raw")
plot(fit1,method="coefficients",vers="reweighted")
plot(fit1,method="diagnostic")

## for binomial
eps <-0.05                                      # %10 contamination to only class 0
m <- ceiling(eps*n)
y <- sample(0:1,n,replace=TRUE)
xout <- x
xout[y==0,][1:m,] <- xout[1:m,] + 10;           # class 0
yout <- y                                     # wrong classification for vertical outliers


fit2 <- enetLTS(xout,yout,family="binomial",crit.plot=FALSE)
plot(fit2)
plot(fit2,method="resid",vers="raw")
plot(fit2,method="coefficients",vers="reweighted")
plot(fit2,method="diagnostic")


## for multinomial

n <- 120; p <- 15
NC <- 3
X <- matrix(rnorm(n * p), n, p)
betas <- matrix(1:NC, ncol=NC, nrow=p, byrow=TRUE)
betas[(p-5):p,]=0; betas <- rbind(rep(0,NC),betas)
lv <- cbind(1,X) %*% betas
probs <- exp(lv)/apply(exp(lv),1,sum)
y <- apply(probs,1,function(prob){sample(1:NC, 1, TRUE, prob)})
```

```
xout <- X
eps <-0.05                          # %10 contamination to only class 0
m <- ceiling(eps*n)
xout[1:m,] <- xout[1:m,] + 10       # bad leverage points
yout <- y


fit3 <- enetLTS(xout,yout,family="multinomial")
plotCoef.enetLTS(fit3)
plotCoef.enetLTS(fit3,vers="raw")
```

`plotCoef.enetLTS` *coefficients plots from the* `"enetLTS"` *object*

Description

Produce plots for the coefficients of the current model.

Usage

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

Arguments

`object`	the model fit to be plotted.
`vers`	a character string denoting which model to use for the plots. Possible values are `"reweighted"` (the default) for plots from the reweighted fit, and `"raw"` for plots from the raw fit.
`colors`	optional parameter, list object with list names `bars`, `errorbars`, `background`, `abline`, `scores`, `cutoffs`, `badouts`, `modouts`, each containing a string referring to a color.
`...`	additional arguments from the `enetLTS` object if needed.

Value

An object of class `"ggplot"` (see `ggplot`).

Note

gives the matplot of

- coefficients vs indices.

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER
Maintainer: Fatma Sevinc KURNAZ <fatmasevinckurnaz@gmail.com>;<fskurnaz@yildiz.edu.tr>

References

Kurnaz, F.S., Hoffmann, I. and Filzmoser, P. (2017) Robust and sparse estimation methods for high dimensional linear and logistic regression. *Chemometrics and Intelligent Laboratory Systems*.

See Also

ggplot, enetLTS, coef.enetLTS, predict.enetLTS

Examples

```
## for gaussian

set.seed(86)
n <- 100; p <- 25                              # number of observations and variables
beta <- rep(0,p); beta[1:6] <- 1               # 10% nonzero coefficients
sigma <- 0.5                                   # controls signal-to-noise ratio
x <- matrix(rnorm(n*p, sigma),nrow=n)
e <- rnorm(n,0,1)                              # error terms
eps <- 0.1                                     # contamination level
m <- ceiling(eps*n)                            # observations to be contaminated
eout <- e; eout[1:m] <- eout[1:m] + 10         # vertical outliers
yout <- c(x %*% beta + sigma * eout)         # response
xout <- x; xout[1:m,] <- xout[1:m,] + 10       # bad leverage points


fit1 <- enetLTS(xout,yout,crit.plot=FALSE)
plotCoef.enetLTS(fit1)
plotCoef.enetLTS(fit1,vers="raw")

## for binomial
eps <-0.05                                      # %10 contamination to only class 0
m <- ceiling(eps*n)
y <- sample(0:1,n,replace=TRUE)
xout <- x
xout[y==0,][1:m,] <- xout[1:m,] + 10;           # class 0
yout <- y                                     # wrong classification for vertical outliers



fit2 <- enetLTS(xout,yout,family="binomial")
plotCoef.enetLTS(fit2)
plotCoef.enetLTS(fit2,vers="raw")


## for multinomial

n <- 120; p <- 15
NC <- 3
X <- matrix(rnorm(n * p), n, p)
betas <- matrix(1:NC, ncol=NC, nrow=p, byrow=TRUE)
betas[(p-5):p,]=0; betas <- rbind(rep(0,NC),betas)
lv <- cbind(1,X) %*% betas
```

```
probs <- exp(lv)/apply(exp(lv),1,sum)
y <- apply(probs,1,function(prob){sample(1:NC, 1, TRUE, prob)})
xout <- X
eps <-0.05                          # %10 contamination to only class 0
m <- ceiling(eps*n)
xout[1:m,] <- xout[1:m,] + 10       # bad leverage points
yout <- y


fit3 <- enetLTS(xout,yout,family="multinomial")
plotCoef.enetLTS(fit3)
plotCoef.enetLTS(fit3,vers="raw")
```

`plotDiagnostic.enetLTS` *diagnostics plots from the* `"enetLTS"` *object*

Description

Produce plots for the diagnostics of the current model.

Usage

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

Arguments

`object`	the model fit to be plotted.
`vers`	a character string denoting which model to use for the plots. Possible values are `"reweighted"` (the default) for plots from the reweighted fit, and `"raw"` for plots from the raw fit.
`...`	additional arguments from the `enetLTS` object if needed.

Value

An object of class `"ggplot"` (see `ggplot`).

Note

gives the plot of

- First two components of estimated scores for multinomial logistic regression (for `family="multinomial"`)

- y vs fitted values/link function. (for for both `family="binomial"` and `family="gaussian"`).

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER
Maintainer: Fatma Sevinc KURNAZ <fatmasevinckurnaz@gmail.com>; <fskurnaz@yildiz.edu.tr>

References

Kurnaz, F.S., Hoffmann, I. and Filzmoser, P. (2017) Robust and sparse estimation methods for high dimensional linear and logistic regression. *Chemometrics and Intelligent Laboratory Systems*.

See Also

ggplot, enetLTS, coef.enetLTS, predict.enetLTS, residuals.enetLTS, fitted.enetLTS

Examples

```
## for gaussian

set.seed(86)
n <- 100; p <- 25                              # number of observations and variables
beta <- rep(0,p); beta[1:6] <- 1               # 10% nonzero coefficients
sigma <- 0.5                                   # controls signal-to-noise ratio
x <- matrix(rnorm(n*p, sigma),nrow=n)
e <- rnorm(n,0,1)                              # error terms
eps <- 0.1                                     # contamination level
m <- ceiling(eps*n)                            # observations to be contaminated
eout <- e; eout[1:m] <- eout[1:m] + 10         # vertical outliers
yout <- c(x %*% beta + sigma * eout)         # response
xout <- x; xout[1:m,] <- xout[1:m,] + 10       # bad leverage points


fit1 <- enetLTS(xout,yout,crit.plot=FALSE)
plotDiagnostic.enetLTS(fit1)
plotDiagnostic.enetLTS(fit1,vers="raw")

## for binomial

eps <-0.05                                       # %10 contamination to only class 0
m <- ceiling(eps*n)
y <- sample(0:1,n,replace=TRUE)
xout <- x
xout[y==0,][1:m,] <- xout[1:m,] + 10;            # class 0
yout <- y                                      # wrong classification for vertical outliers


fit2 <- enetLTS(xout,yout,family="binomial",crit.plot=FALSE)
plotDiagnostic.enetLTS(fit2)
plotDiagnostic.enetLTS(fit2,vers="raw")


## for multinomial

n <- 120; p <- 15
NC <- 3
X <- matrix(rnorm(n * p), n, p)
betas <- matrix(1:NC, ncol=NC, nrow=p, byrow=TRUE)
betas[(p-5):p,]=0; betas <- rbind(rep(0,NC),betas)
lv <- cbind(1,X) %*% betas
```

```
probs <- exp(lv)/apply(exp(lv),1,sum)
y <- apply(probs,1,function(prob){sample(1:NC, 1, TRUE, prob)})
xout <- X
eps <-0.05                          # %10 contamination to only class 0
m <- ceiling(eps*n)
xout[1:m,] <- xout[1:m,] + 10       # bad leverage points
yout <- y


fit3 <- enetLTS(xout,yout,family="multinomial",crit.plot=FALSE)
plotDiagnostic.enetLTS(fit3)
plotDiagnostic.enetLTS(fit3,vers="raw")
```

`plotResid.enetLTS` *residuals plots from the* `"enetLTS"` *object*

Description

Produce plots for the residuals of the current model. Residuals corresponds to deviances for `family="multinomial"` and `family="binomial"`.

Usage

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

Arguments

`object`	the model fit to be plotted.
`vers`	a character string denoting which model to use for the plots. Possible values are `"reweighted"` (the default) for plots from the reweighted fit, and `"raw"` for plots from the raw fit.
`...`	additional arguments from the `enetLTS` object if needed.

Value

An object of class `"ggplot"` (see `ggplot`).

Note

gives the plot of - residuals vs indices. (for `family="gaussian"`).

- deviances vs indices. (for both `family="multinomial"` and `family="binomial"`).

- additionally, residuals vs fitted values/link function (for `family="binomial"` and `family="gaussian"`).

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER
Maintainer: Fatma Sevinc KURNAZ <fatmasevincskurnaz@gmail.com>;<fskurnaz@yildiz.edu.tr>

References

Kurnaz, F.S., Hoffmann, I. and Filzmoser, P. (2017) Robust and sparse estimation methods for high dimensional linear and logistic regression. *Chemometrics and Intelligent Laboratory Systems*.

See Also

ggplot, enetLTS, predict.enetLTS, residuals.enetLTS, fitted.enetLTS

Examples

```
## for gaussian

set.seed(86)
n <- 100; p <- 25                              # number of observations and variables
beta <- rep(0,p); beta[1:6] <- 1               # 10% nonzero coefficients
sigma <- 0.5                                   # controls signal-to-noise ratio
x <- matrix(rnorm(n*p, sigma),nrow=n)
e <- rnorm(n,0,1)                              # error terms
eps <- 0.1                                     # contamination level
m <- ceiling(eps*n)                            # observations to be contaminated
eout <- e; eout[1:m] <- eout[1:m] + 10         # vertical outliers
yout <- c(x %*% beta + sigma * eout)         # response
xout <- x; xout[1:m,] <- xout[1:m,] + 10       # bad leverage points


fit1 <- enetLTS(xout,yout)
plotResid.enetLTS(fit1)
plotResid.enetLTS(fit1,vers="raw")

## for binomial

eps <-0.05                                       # %10 contamination to only class 0
m <- ceiling(eps*n)
y <- sample(0:1,n,replace=TRUE)
xout <- x
xout[y==0,][1:m,] <- xout[1:m,] + 10;            # class 0
yout <- y                                      # wrong classification for vertical outliers


fit2 <- enetLTS(xout,yout,family="binomial",crit.plot=FALSE)
plotResid.enetLTS(fit2)
plotResid.enetLTS(fit2,vers="raw")


## for multinomial

n <- 120; p <- 15
NC <- 3
X <- matrix(rnorm(n * p), n, p)
betas <- matrix(1:NC, ncol=NC, nrow=p, byrow=TRUE)
betas[(p-5):p,]=0; betas <- rbind(rep(0,NC),betas)
lv <- cbind(1,X) %*% betas
```

```
probs <- exp(lv)/apply(exp(lv),1,sum)
y <- apply(probs,1,function(prob){sample(1:NC, 1, TRUE, prob)})
xout <- X
eps <-0.05                          # %10 contamination to only class 0
m <- ceiling(eps*n)
xout[1:m,] <- xout[1:m,] + 10       # bad leverage points
yout <- y


fit3 <- enetLTS(xout,yout,family="multinomial")
plotResid.enetLTS(fit3)
plotResid.enetLTS(fit3,vers="raw")
```

`predict.enetLTS` *make predictions from the* `"enetLTS"` *object.*

Description

Similar to other predict methods, this function predicts fitted values, logits, coefficients and 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`	the model fit from which to make predictions.
`newX`	new values for the predictor matrix `X`. Must be a matrix; can be sparse as in `Matrix` package. This argument is not used for `type=c("coefficients","nonzero")`.
`vers`	a character string denoting which fit to use for the predictions. Possible values are `"reweighted"` (the default) for predicting values from the reweighted fit, `"raw"` for predicting values from the raw fit.
`type`	type of prediction required. `type="link"` gives the link function. `type="response"` gives the fitted probabilities for `"binomial"` and gives the fitted values for `"gaussian"`. `type="coefficients"` computes the coefficients from the fitted model. `type="nonzero"` returns a list of the indices of the nonzero coefficients. `type="class"` is available only for `"binomial"` model, and produces the class label corresponding to the maximum probability.
`...`	additional arguments from the `enetLTS` object if needed.

Details

The newdata argument defaults to the matrix of predictors used to fit the model such that the fitted values are computed.

coef.enetLTS(...) is equivalent to predict.enetLTS(object,newX,type="coefficients",...), where newX argument is the matrix as in enetLTS.

Value

The requested predicted values are returned.

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER
Maintainer: Fatma Sevinc KURNAZ <fatmasevinckurnaz@gmail.com>;<fskurnaz@yildiz.edu.tr>

See Also

enetLTS, coef.enetLTS, nonzeroCoef.enetLTS

Examples

```
## for gaussian

set.seed(86)
n <- 100; p <- 25                              # number of observations and variables
beta <- rep(0,p); beta[1:6] <- 1               # 10% nonzero coefficients
sigma <- 0.5                                   # controls signal-to-noise ratio
x <- matrix(rnorm(n*p, sigma),nrow=n)
e <- rnorm(n,0,1)                              # error terms
eps <- 0.1                                     # contamination level
m <- ceiling(eps*n)                            # observations to be contaminated
eout <- e; eout[1:m] <- eout[1:m] + 10         # vertical outliers
yout <- c(x %*% beta + sigma * eout)         # response
xout <- x; xout[1:m,] <- xout[1:m,] + 10       # bad leverage points


fit1 <- enetLTS(xout,yout)
predict(fit1,newX=xout)
predict(fit1,newX=xout,type="coefficients")
predict(fit1,newX=xout,type="nonzero",vers="raw")
# provide new X matrix
newX <- matrix(rnorm(n*p, sigma),nrow=n)
predict(fit1,newX=newX,type="response")
predict(fit1,newX=newX,type="coefficients")
predict(fit1,newX=newX,type="nonzero")

## for binomial

eps <-0.05                                     # %10 contamination to only class 0
m <- ceiling(eps*n)
y <- sample(0:1,n,replace=TRUE)
```

```
xout <- x
xout[y==0,][1:m,] <- xout[1:m,] + 10;          # class 0
yout <- y                                    # wrong classification for vertical outliers


fit2 <- enetLTS(xout,yout,family="binomial")
predict(fit2,newX=xout)
predict(fit2,newX=xout,type="coefficients")
predict(fit2,newX=xout,type="nonzero",vers="raw")
predict(fit2,newX=newX,type="response")
predict(fit2,newX=newX,type="class")
predict(fit2,newX=newX,type="coefficients",vers="raw")
predict(fit2,newX=newX,type="nonzero")



## for multinomial

n <- 120; p <- 15
NC <- 3
X <- matrix(rnorm(n * p), n, p)
betas <- matrix(1:NC, ncol=NC, nrow=p, byrow=TRUE)
betas[(p-5):p,]=0; betas <- rbind(rep(0,NC),betas)
lv <- cbind(1,X) %*% betas
probs <- exp(lv)/apply(exp(lv),1,sum)
y <- apply(probs,1,function(prob){sample(1:NC, 1, TRUE, prob)})
xout <- X
eps <-0.05                          # %10 contamination to only class 0
m <- ceiling(eps*n)
xout[1:m,] <- xout[1:m,] + 10       # bad leverage points
yout <- y


fit3    <- enetLTS(xout,yout,family="multinomial")
predict(fit3,newX=xout)
predict(fit3,newX=xout,type="coefficients")
predict(fit3,newX=xout,type="nonzero",vers="raw")
predict(fit3,newX=xout,type="response")
predict(fit3,newX=xout,type="class")
predict(fit3,newX=xout,type="coefficients",vers="raw")
predict(fit3,newX=xout,type="nonzero")
```

`print.enetLTS` *print from the* "enetLTS" *object*

Description

Print a summary of the enetLTS object.

Usage

```
## S3 method for class 'enetLTS'
print(x,vers=c("reweighted","raw"),...)
```

Arguments

x	fitted enetLTS object
vers	a character string specifying for which fit to make predictions. Possible values are "reweighted" (the default) for predicting values from the reweighted fit, "raw" for predicting values from the raw fit.
...	additional arguments from the enetLTS object if needed.

Details

The call that produced the enetLTS object is printed, followed by the coefficients, the number of nonzero coefficients and penalty parameters.

Value

The produced object, the coefficients, the number of nonzero coefficients and penalty parameters are returned.

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER
Maintainer: Fatma Sevinc KURNAZ <fatmasevinckurnaz@gmail.com>;<fskurnaz@yildiz.edu.tr>

See Also

enetLTS, predict.enetLTS, coef.enetLTS

Examples

```
## for gaussian

set.seed(86)
n <- 100; p <- 25                              # number of observations and variables
beta <- rep(0,p); beta[1:6] <- 1               # 10% nonzero coefficients
sigma <- 0.5                                   # controls signal-to-noise ratio
x <- matrix(rnorm(n*p, sigma),nrow=n)
e <- rnorm(n,0,1)                              # error terms
eps <- 0.1                                     # contamination level
m <- ceiling(eps*n)                            # observations to be contaminated
eout <- e; eout[1:m] <- eout[1:m] + 10         # vertical outliers
yout <- c(x %*% beta + sigma * eout)         # response
xout <- x; xout[1:m,] <- xout[1:m,] + 10       # bad leverage points


fit1 <- enetLTS(xout,yout)
print(fit1)
```

```
print(fit1,vers="raw")


## for binomial

eps <-0.05                                      # %10 contamination to only class 0
m <- ceiling(eps*n)
y <- sample(0:1,n,replace=TRUE)
xout <- x
xout[y==0,][1:m,] <- xout[1:m,] + 10;           # class 0
yout <- y                                     # wrong classification for vertical outliers



fit2 <- enetLTS(xout,yout,family="binomial")
print(fit2)
print(fit2,vers="raw")


## for multinomial

n <- 120; p <- 15
NC <- 3
X <- matrix(rnorm(n * p), n, p)
betas <- matrix(1:NC, ncol=NC, nrow=p, byrow=TRUE)
betas[(p-5):p,]=0; betas <- rbind(rep(0,NC),betas)
lv <- cbind(1,X) %*% betas
probs <- exp(lv)/apply(exp(lv),1,sum)
y <- apply(probs,1,function(prob){sample(1:NC, 1, TRUE, prob)})
xout <- X
eps <-0.05                          # %10 contamination to only class 0
m <- ceiling(eps*n)
xout[1:m,] <- xout[1:m,] + 10       # bad leverage points
yout <- y


fit3    <- enetLTS(xout,yout,family="multinomial")
print(fit3)
print(fit3,vers="raw")
```

`residuals.enetLTS` *the residuals from the* "enetLTS" *object*

Description

A numeric vector which returns residuals from the enetLTS object. Residuals correspond to deviances if `family="multinomial"` and `family="binomial"`.

Usage

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

Arguments

object	the model fit from which to extract residuals.
vers	a character string specifying for which estimator to extract outlier weights. Possible values are "reweighted" (the default) for weights indicating outliers from the reweighted fit, "raw" for weights indicating outliers from the raw fit, or "both" for the outlier weights from both estimators.
...	additional arguments from the enetLTS object.

Value

A numeric vector containing the requested residuals.

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER
Maintainer: Fatma Sevinc KURNAZ <fatmasevinckurnaz@gmail.com>;<fskurnaz@yildiz.edu.tr>

See Also

enetLTS, fitted.enetLTS, predict.enetLTS, coef.enetLTS

Examples

```
## for gaussian

set.seed(86)
n <- 100; p <- 25                              # number of observations and variables
beta <- rep(0,p); beta[1:6] <- 1               # 10% nonzero coefficients
sigma <- 0.5                                   # controls signal-to-noise ratio
x <- matrix(rnorm(n*p, sigma),nrow=n)
e <- rnorm(n,0,1)                              # error terms
eps <- 0.1                                     # contamination level
m <- ceiling(eps*n)                            # observations to be contaminated
eout <- e; eout[1:m] <- eout[1:m] + 10         # vertical outliers
yout <- c(x %*% beta + sigma * eout)         # response
xout <- x; xout[1:m,] <- xout[1:m,] + 10       # bad leverage points


fit1 <- enetLTS(xout,yout)
residuals(fit1)
residuals(fit1,vers="raw")
residuals(fit1,vers="both")


## for binomial
```

```
eps <-0.05                                    # %10 contamination to only class 0
m <- ceiling(eps*n)
y <- sample(0:1,n,replace=TRUE)
xout <- x
xout[y==0,][1:m,] <- xout[1:m,] + 10;         # class 0
yout <- y                                    # wrong classification for vertical outliers



fit2 <- enetLTS(xout,yout,family="binomial")
residuals(fit2)
residuals(fit2,vers="raw")
residuals(fit2,vers="both")



## for multinomial

n <- 120; p <- 15
NC <- 3
X <- matrix(rnorm(n * p), n, p)
betas <- matrix(1:NC, ncol=NC, nrow=p, byrow=TRUE)
betas[(p-5):p,]=0; betas <- rbind(rep(0,NC),betas)
lv <- cbind(1,X) %*% betas
probs <- exp(lv)/apply(exp(lv),1,sum)
y <- apply(probs,1,function(prob){sample(1:NC, 1, TRUE, prob)})
xout <- X
eps <-0.05                          # %10 contamination to only class 0
m <- ceiling(eps*n)
xout[1:m,] <- xout[1:m,] + 10       # bad leverage points
yout <- y


fit3    <- enetLTS(xout,yout,family="multinomial")
residuals(fit3)
residuals(fit3,vers="raw")
residuals(fit3,vers="both")
```

`weights.enetLTS` *binary weights from the* "enetLTS" *object*

Description

Extract binary weights that indicate outliers from the current model.

Usage

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

Arguments

object	the model fit from which to extract outlier weights.
vers	a character string specifying for which estimator to extract outlier weights. Possible values are "reweighted" (the default) for weights indicating outliers from the reweighted fit, "raw" for weights indicating outliers from the raw fit, or "both" for the outlier weights from both estimators.
index	a logical indicating whether the indices of the weight vector should be included or not (the default is FALSE).
...	additional arguments from the enetLTS object if needed.

Value

A numeric vector containing the requested outlier weights.

Note

The weights are 1 for observations with reasonably small residuals and 0 for observations with large residuals. Here, residuals represent standardized residuals for family="gaussian", Pearson residuals for family="binomial" and group-wise scaled robust distances family="multinomial".

Use weights with or without index is available.

Author(s)

Fatma Sevinc KURNAZ, Irene HOFFMANN, Peter FILZMOSER
Maintainer: Fatma Sevinc KURNAZ <fatmasevinckurnaz@gmail.com>;<fskurnaz@yildiz.edu.tr>

See Also

enetLTS

Examples

```
## for gaussian

set.seed(86)
n <- 100; p <- 25                              # number of observations and variables
beta <- rep(0,p); beta[1:6] <- 1               # 10% nonzero coefficients
sigma <- 0.5                                   # controls signal-to-noise ratio
x <- matrix(rnorm(n*p, sigma),nrow=n)
e <- rnorm(n,0,1)                              # error terms
eps <- 0.1                                     # contamination level
m <- ceiling(eps*n)                            # observations to be contaminated
eout <- e; eout[1:m] <- eout[1:m] + 10         # vertical outliers
yout <- c(x %*% beta + sigma * eout)         # response
xout <- x; xout[1:m,] <- xout[1:m,] + 10       # bad leverage points


fit1 <- enetLTS(xout,yout)
weights(fit1)
```

```
weights(fit1,vers="raw",index=TRUE)
weights(fit1,vers="both",index=TRUE)


## for binomial

eps <-0.05                                      # %10 contamination to only class 0
m <- ceiling(eps*n)
y <- sample(0:1,n,replace=TRUE)
xout <- x
xout[y==0,][1:m,] <- xout[1:m,] + 10;           # class 0
yout <- y                                     # wrong classification for vertical outliers


fit2 <- enetLTS(xout,yout,family="binomial")
weights(fit2)
weights(fit2,vers="raw",index=TRUE)
weights(fit2,vers="both",index=TRUE)


## for multinomial

n <- 120; p <- 15
NC <- 3
X <- matrix(rnorm(n * p), n, p)
betas <- matrix(1:NC, ncol=NC, nrow=p, byrow=TRUE)
betas[(p-5):p,]=0; betas <- rbind(rep(0,NC),betas)
lv <- cbind(1,X) %*% betas
probs <- exp(lv)/apply(exp(lv),1,sum)
y <- apply(probs,1,function(prob){sample(1:NC, 1, TRUE, prob)})
xout <- X
eps <-0.05                          # %10 contamination to only class 0
m <- ceiling(eps*n)
xout[1:m,] <- xout[1:m,] + 10       # bad leverage points
yout <- y


fit3    <- enetLTS(xout,yout,family="multinomial")
weights(fit3)
weights(fit3,vers="raw",index=TRUE)
weights(fit3,vers="both",index=TRUE)
```

Index