Package 'pense'

June 8, 2019

Type Package

Title Penalized Elastic Net S/MM-Estimator of Regression

Version 1.2.5

Date 2019-06-07

Copyright See the file COPYRIGHTS for copyright details on some of the functions and algorithms used.

Encoding UTF-8

Biarch true

URL https://github.com/dakep/pense-rpkg

BugReports https://github.com/dakep/pense-rpkg/issues

Description Robust penalized elastic net S and MM estimator for linear regression. The method is described in detail in Cohen Freue, G. V., Kepplinger, D., Salibian-Barrera, M., and Smucler, E. (2017) <https://gcohenfr.github.io/pdfs/PENSE_manuscript.pdf>.

Depends R (>= 3.1.0), Matrix

Imports Rcpp, robustbase, parallel, methods

LinkingTo Rcpp, RcppArmadillo

Suggests testthat, lars

License GPL (>= 2)

NeedsCompilation yes

RoxygenNote 6.1.1

Author David Kepplinger [aut, cre],
Matias Salibian-Barrera [aut],
Gabriela Cohen Freue [aut],
Derek Cho [ctb]

Maintainer David Kepplinger <david.kepplinger@gmail.com>

Repository CRAN

Date/Publication 2019-06-08 14:30:06 UTC

R topics documented:

coef.elnetfit . . . 2
coef.pense . . . 4
elnet . . . 5
elnet_cv . . . 8
enpy . . . 10
en_options_aug_lars . . . 12
initest_options . . . 13
mscale . . . 14
mstep_options . . . 15
pense . . . 16
pensem . . . 19
pense_options . . . 22
plot.cv_elnetfit . . . 23
plot.elnetfit . . . 24
plot.pense . . . 25
predict.elnetfit . . . 26
predict.pense . . . 28
prinsens . . . 29
residuals.elnetfit . . . 30
residuals.pense . . . 32

Index **34**

`coef.elnetfit` *Extract Model Coefficients*

Description

Extract Model Coefficients

Usage

```
## S3 method for class 'elnetfit'
coef(object, lambda, exact = FALSE, sparse = FALSE,
  ...)
```

Arguments

`object`	object of type `elnetfit` to extract coefficients from.
`lambda`	the value of the penalty parameter. Default is to use the optimal lambda (`object$lambda_opt`).
`exact`	if the lambda is not part of the lambda grid, should the estimates be obtained by linear interpolation between the nearest lambda values (default) or computed exactly.
`sparse`	return a sparse vector or a dense (base R) numeric vector
`...`	currently not used.

Value

if `sparse = FALSE` a numeric vector of size $p + 1$. Otherwise a sparse matrix with one column and $p + 1$ rows.

Examples

```
# Generate data with highly correlated groups of variables
set.seed(12345)
n <- 100
p <- 20
x <- 1 + matrix(rnorm(n * p), ncol = p)
x[, 2] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 3] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 5] <- x[, 4] + rnorm(n, sd = 0.01)
x[, 6] <- x[, 4] + rnorm(n, sd = 0.01)

y <- drop(x %*% c(rep(c(2, 5), each = 3), numeric(p - 6)) + rnorm(n))

# Compute the classical EN and select the optimal lambda by CV
set.seed(1234)
est_en_cv <- elnet_cv(
    x, y,
    alpha = 0.5,
    correction = TRUE
)

# For cross-validated EN fits, the `coef`, `predict`, and `residuals` methods
# return/use the estimated coefficients at the "optimal" lambda
coef(est_en_cv) # Extract coefficients
predict(est_en_cv) # Extract fitted values
predict(est_en_cv, newdata = x) # Predict values
residuals(est_en_cv) # Extract residuals

# We can also request the coefficient at another lambda. By default,
# this will interpolate between the solutions at the two surrounding
# lambda values.
coef(est_en_cv, lambda = 6)

# If needed, the solution at the given lambda can also be computed exactly.
coef(est_en_cv, lambda = 6, exact = TRUE)


# If we compute the EN estimator without choosing an optimal lambda,
# the lambda parameter needs to be specified in the call to coef
est_en <- elnet(
    x, y,
    alpha = 0.5
)

# Without specifying lambda, the `coef` method would raise an error.
coef(est_en, lambda = 6)
```

coef.pense *Extract Model Coefficients*

Description

Extract Model Coefficients

Usage

```
## S3 method for class 'pense'
coef(object, lambda, exact = FALSE, sparse = FALSE,
  correction = TRUE, ...)
```

Arguments

object	a PENSE or PENSEM estimate to extract coefficients from.
lambda	the value of the penalty parameter. Default is to use the optimal lambda (object$lambda_opt).
exact	if lambda is not part of the lambda grid, should the estimates be obtained by linear interpolation between the nearest lambda values (default) or computed exactly.
sparse	return a sparse vector or a dense (base R) numeric vector
correction	should a correction factor be applied to the EN estimate? See elnet for details on the applied correction.
...	currently not used.

Value

if sparse = FALSE a numeric vector of size $p + 1$. Otherwise a sparse matrix with one column and $p + 1$ rows.

Examples

```
# Generate data with highly correlated groups of variables and some outliers
set.seed(12345)
n <- 50
n_out <- 3
p <- 20
x <- 1 + matrix(rnorm(n * p), ncol = p)
x[, 2] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 3] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 5] <- x[, 4] + rnorm(n, sd = 0.01)
x[, 6] <- x[, 4] + rnorm(n, sd = 0.01)

y <- x %*% c(rep(c(2, 5), each = 3), numeric(p - 6)) + rnorm(n)

y[seq_len(n_out)] <- rnorm(n_out, -100, sd = 3)
```

```
# Compute the PENSE estimator
set.seed(1234)
est_en <- pense(x, y, alpha = 0.5, warm_reset = 1, cv_k = 3)

# From the fitted model we can extract the coefficients, fitted values, and
# residuals at the "optimal" lambda as chosen by CV.
coef(est_en) # Extract coefficients
predict(est_en) # Extract fitted values
predict(est_en, newdata = x) # Predict values
residuals(est_en) # Extract residuals

# We can also request the coefficients/predicitons/residuals at another lambda.
# If the requested lambda is not in the original lambda grid, the methods
# will approximate the coefficient vector by linear interpolation of
# the solutions at the surrounding lambda values.
coef(est_en, lambda = 5)

# If the exact solution is needed, this can be requested
coef(est_en, lambda = 5, exact = TRUE)
residuals(est_en, lambda = 5, exact = TRUE)
```

`elnet` *Elastic Net Estimator for Regression*

Description

Estimate the elastic net regression coefficients.

Usage

```
elnet(x, y, alpha, nlambda = 100, lambda, weights, intercept = TRUE,
  options = en_options_aug_lars(), lambda_min_ratio, xtest,
  correction = TRUE)
```

Arguments

`x`	data matrix with predictors
`y`	response vector
`alpha`	controls the balance between the L1 and the L2 penalty. `alpha = 0` is the ridge (L2) penalty, `alpha = 1` is the lasso.
`nlambda`	size of the lambda grid if `lambda` is not specified.
`lambda`	a grid of decreasing lambda values.
`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If non-NULL, weighted EN is used with weights `weights`. See also 'Details'.
`intercept`	should an intercept be estimated?
`options`	additional options for the EN algorithm. See `en_options` for details.

`lambda_min_ratio`
: if the lambda grid should be automatically defined, the ratio of the smallest to the largest lambda value in the grid. The default is `1e-6` if $n < p$, and `1e-5 * 10^floor(log10(p / n))` otherwise.

`xtest`
: data matrix with predictors used for prediction. This is useful for testing the prediction performance on an independent test set.

`correction`
: should the "corrected" EN estimator be returned. If `TRUE`, as by default, the corrected estimator as defined in Zou & Hastie (2005) is returned.

Details

This solves the minimization problem

$$\frac{1}{2N} RSS + \lambda \left(\frac{(1-\alpha)}{2} \|\beta\|_2^2 + \alpha \|\beta\|_1 \right)$$

If weights are supplied, the minimization problem becomes

$$\frac{1}{2N} \sum_{i=1}^{n} w_i r_i^2 + \lambda \left(\frac{(1-\alpha)}{2} \|\beta\|_2^2 + \alpha \|\beta\|_1 \right)$$

Value

`lambda`
: vector of lambda values.

`status`
: integer specifying the exit status of the EN algorithm.

`message`
: explanation of the exit status.

`coefficients`
: matrix of regression coefficients. Each column corresponds to the estimate for the lambda value at the same index.

`residuals`
: matrix of residuals. Each column corresponds to the residuals for the lambda value at the same index.

`predictions`
: if `xtest` was given, matrix of predicted values. Each column corresponds to the predictions for the lambda value at the same index.

Algorithms

Currently this function can compute the elastic net estimator using either augmented LARS or the Dual Augmented Lagrangian (DAL) algorithm (Tomioka 2011). Augmented LARS performs LASSO via the LARS algorithm (or OLS if `alpha = 0`) on the data matrix augmented with the L2 penalty term. The time complexity of this algorithm increases fast with an increasing number of predictors. The algorithm currently can not leverage a previous or an approximate solution to speed up computations. However, it is always guaranteed to find the solution.

DAL is an iterative algorithm directly minimizing the Elastic Net objective. The algorithm can take an approximate solution to the problem to speed up convergence. In the case of very small lambda values and a bad starting point, DAL may not converge to the solution and hence give wrong results. This would be indicated in the returned status code. Time complexity of this algorithm is dominated by the number of observations.

DAL is much faster for a small number of observations (< 200) and a large number of predictors, especially if an approximate solution is available.

References

Tomioka, R., Suzuki, T. and Sugiyama, M. (2011). Super-Linear Convergence of Dual Augmented Lagrangian Algorithm for Sparse Learning. *Journal of Machine Learning Research* **12**(May):1537-1586.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*. Series B (Statistical Methodology), **67**(2):301-320.

See Also

elnet_cv for automatic selection of the penalty parameter based on the cross-validated prediction error.

Examples

```
# Generate some dummy data
set.seed(12345)
n <- 30
p <- 15
x <- 1 + matrix(rnorm(n * p), ncol = p)
y <- x %*% c(2:5, numeric(p - 4)) + rnorm(n)

x_test <- matrix(rnorm(10 * n * p), ncol = p)
y_test <- drop(x_test %*% c(2:5, numeric(p - 4)) + rnorm(n))

# Compute the classical EN with predictions for x_test
set.seed(1234)
est <- elnet(
    x, y,
    alpha = 0.6,
    nlambda = 100,
    xtest = x_test
)

# Plot the RMSPE computed from the given test set
rmspe_test <- sqrt(colMeans((y_test - est$predictions)^2))
plot(est$lambda, rmspe_test, log = "x")

##
## For large data sets, the DAL algorithm is much faster
##
set.seed(12345)
n <- 100
p <- 1500
x <- 1 + matrix(rnorm(n * p), ncol = p)
y <- x %*% c(2:5, numeric(p - 4)) + rnorm(n)

x_test <- matrix(rnorm(10 * n * p), ncol = p)
y_test <- drop(x_test %*% c(2:5, numeric(p - 4)) + rnorm(n))

# The DAL algorithm takes ~1.5 seconds to compute the solution path
set.seed(1234)
```

```
system.time(
    est_dal <- elnet(
        x, y,
        alpha = 0.6,
        nlambda = 100,
        options = en_options_dal(),
        xtest = x_test
    )
)


# In comparison, the augmented LARS algorithm can take several minutes
set.seed(1234)
system.time(
    est_auglars <- elnet(
        x, y,
        alpha = 0.6,
        nlambda = 100,
        options = en_options_aug_lars(),
        xtest = x_test
    )
)
```

`elnet_cv` *Cross-validate Elastic Net*

Description

Perform k-fold cross-validation for `elnet`.

Usage

```
elnet_cv(x, y, alpha, nlambda = 100, lambda, weights, intercept = TRUE,
  cv_k = 10, cv_measure, ncores = getOption("mc.cores", 1L),
  cl = NULL, options = en_options_aug_lars(), lambda_min_ratio, ...)
```

Arguments

`x`	data matrix with predictors
`y`	response vector
`alpha`	controls the balance between the L1 and the L2 penalty. `alpha = 0` is the ridge (L2) penalty, `alpha = 1` is the lasso.
`nlambda`	size of the lambda grid if `lambda` is not specified.
`lambda`	a grid of decreasing lambda values.
`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If non-NULL, weighted EN is used with weights `weights`. See also 'Details'.

intercept	should an intercept be estimated?
cv_k	number of cross validation folds.
cv_measure	function (name) which takes a matrix of residuals and returns a performance measure for each column.
ncores, cl	the number of processor cores or an actual parallel cluster. At most cv_k + 1 nodes will be used.
options	additional options for the EN algorithm. See en_options for details.
lambda_min_ratio	if the lambda grid should be automatically defined, the ratio of the smallest to the largest lambda value in the grid. The default is 1e-6 if $n < p$, and 1e-5 * 10^floor(log10(p / n)) otherwise.
...	additional arguments passed on to elnet.

Value

lambda	vector of lambda values.
status	integer specifying the exit status of the EN algorithm.
message	explanation of the exit status.
coefficients	matrix of regression coefficients. Each column corresponds to the estimate for the lambda value at the same index.
residuals	matrix of residuals. Each column corresponds to the residuals for the lambda value at the same index.
cvres	data frame with lambda, average cross-validated performance and the estimated standard deviation.

See Also

elnet to compute only the solution path, without selecting the optimal penalty parameter using CV.

Examples

```
# Generate some dummy data
set.seed(12345)
n <- 30
p <- 15
x <- 1 + matrix(rnorm(n * p), ncol = p)
y <- x %*% c(2:5, numeric(p - 4)) + rnorm(n)

x_test <- matrix(rnorm(10 * n * p), ncol = p)
y_test <- drop(x_test %*% c(2:5, numeric(p - 4)) + rnorm(n))

# Compute the classical EN and select lambda based on CV
set.seed(1234)
est <- elnet_cv(
    x, y,
    alpha = 0.6,
    nlambda = 100
```

```
)

# The optimal lambda according to CV is
est$lambda_opt
plot(est)

# and the RMSPE at this lambda is
sqrt(mean((y_test - predict(est, x_test))^2))
```

---

enpy *PY (Pena-Yohai) initial estimates for EN S-estimators*

Description

Computes the PY initial estimates for the EN S-estimator with different strategies for computing the principal sensitivity components.

Usage

```
enpy(x, y, alpha, lambda, delta, cc, options = initest_options(),
  en_options = en_options_aug_lars())
```

Arguments

`x`	data matrix with predictors.
`y`	response vector.
`alpha, lambda`	EN penalty parameters (NOT adjusted for the number of observations in `x`).
`delta`	desired breakdown point of the resulting estimator.
`cc`	tuning constant for the S-estimator. Default is to chosen based on the breakdown point `delta`. Should never have to be changed.
`options`	additional options for the initial estimator. See `initest_options` for details.
`en_options`	additional options for the EN algorithm. See `en_options` for details.

Details

Two different methods to calculate the sensitivity components are implemented:

`"rr"` Approximate the PSCs by using the residuals from the elastic net fit and the hat matrix from the ridge regression. This method only works if `alpha` < 1 or `ncol(x)` < `nrow(x)`.

`"exact"` Calculate the PSCs from the difference between the residuals and leave-one-out residuals from elastic net.

Value

`coeff`	A numeric matrix with one initial coefficient per column
`objF`	A vector of values of the objective function for the respective coefficient

References

Pena, D., and Yohai, V.J. (1999). A Fast Procedure for Outlier Diagnostics in Large Regression Problems. *Journal of the American Statistical Association*, **94**(446), 434-445. http://doi.org/10.2307/2670164

Examples

```
set.seed(12345)
n <- 50
p <- 15
beta <- c(2:5, numeric(p - 4))
x <- 1 + matrix(rnorm(n * p), ncol = p)
y <- x %*% beta  + rnorm(n)

# add outliers to y
y[1:5] <- rnorm(5, -100, 0.5)

x_test <- matrix(rnorm(10 * n * p), ncol = p)
y_test <- drop(x_test %*% beta + rnorm(n))

# Compute the PY using the "exact" solutions. Here, the LOO residuals
# are computed using the EN estimator directly. For this method,
# the DAL algorithm is usually faster since it naturally supports
# "warm" starts from a nearby solution vector.
init_exact <- enpy(
    x, y,
    alpha = 0.8,
    lambda = 0.1,
    delta = 0.25,
    options = initest_options(
        psc_method = "exact"
    ),
    en_options = en_options_dal()
)

# Compute the PY using the approximate ("rr") solutions. Here, the LOO residuals
# are approximated by ridge regression, where all the LOO residuals can be
# computed at once using linear algebra.
init_approx <- enpy(
    x, y,
    alpha = 0.8,
    lambda = 0.1,
    delta = 0.25,
    options = initest_options(
        psc_method = "rr"
    ),
    en_options = en_options_aug_lars()
)

# How do they objective functions of the initial estimates compare?
min(init_exact$objF)
min(init_approx$objF)
```

```
# The "exact" method finds slightly estimates with slightly lower objective.

# How do the initial estimates perform on an external test set?
rmspe_exact <- colMeans((y_test - cbind(1, x_test) %*% init_exact$coeff)^2)
rmspe_approx <- colMeans((y_test - cbind(1, x_test) %*% init_approx$coeff)^2)

min(rmspe_exact)
min(rmspe_approx)
# The "exact" method also has better predictive power
```

`en_options_aug_lars` *Additional Options for the EN Algorithms*

Description

Specify additional options for the augmented LARS and the DAL algorithm to solve the EN problem.

Usage

```
en_options_aug_lars(use_gram = c("auto", "yes", "no"), eps = 1e-12)

en_options_dal(maxit = 100, eps = 1e-08, eta_mult = 2,
  eta_start_numerator = 0.01, eta_start, preconditioner = c("approx",
  "none", "diagonal"), verbosity = 0)
```

Arguments

`use_gram`	should the Gram matrix be pre-computed.
`eps`	numeric tolerance for convergence.
`maxit`	maximum number of iterations allowed.
`eta_mult`	multiplier to increase eta at each iteration.
`eta_start_numerator`	if `eta_start` is missing, it is defined by `eta_start = eta_start_numerator / lambda`.
`eta_start`	the start value for eta.
`preconditioner`	preconditioner for the numerical solver. If none, a standard solver will be used, otherwise the faster preconditioned conjugate gradient is used.
`verbosity`	verbosity of the algorithm.

Value

a checked options list.

See Also

Other specifying additional options: `initest_options`, `mstep_options`, `pense_options`

initest_options *Additional Options for the Initial Estimator*

Description

Specify additional options for the initial estimator based on the Pena-Yohai estimator.

Usage

```
initest_options(keep_solutions = 5, psc_method = c("exact", "rr"),
  maxit = 10, maxit_pense_refinement = 5, eps = 1e-06,
  psc_keep = 0.5, resid_keep_method = c("proportion", "threshold"),
  resid_keep_prop = 0.6, resid_keep_thresh = 2, mscale_eps = 1e-08,
  mscale_maxit = 200)
```

Arguments

keep_solutions	how many initial estimates should be kept to perform full PENSE iterations?
psc_method	The method to use for computing the principal sensitivity components. See details for the possible choices.
maxit	maximum number of refinement iterations.
maxit_pense_refinement	maximum number of PENSE iterations to refine initial estimator.
eps	numeric tolerance for convergence.
psc_keep	proportion of observations to keep based on the PSC scores.
resid_keep_method	How to clean the data based on large residuals. If "proportion", observations with the smallest resid_keep_prop residuals will be retained. If "threshold", all observations with scaled residuals smaller than the threshold resid_keep_thresh will be retained.
resid_keep_prop, resid_keep_thresh	proportion or threshold for observations to keep based on their residual.
mscale_eps, mscale_maxit	maximum number of iterations and numeric tolerance for the M-scale.

Details

Two different methods to calculate the sensitivity components are implemented:

"rr" Approximate the PSCs by using the residuals from the elastic net fit and the hat matrix from the ridge regression. This method only works if alpha < 1 or ncol(x) < nrow(x).

"exact" Calculate the PSCs from the difference between the residuals and leave-one-out residuals from elastic net.

Value

a checked options list.

References

Pena, D., & Yohai, V.. (1999). A Fast Procedure for Outlier Diagnostics in Large Regression Problems. *Journal of the American Statistical Association*, 94(446), 434-445. http://doi.org/10.2307/2670164

See Also

Other specifying additional options: `en_options_aug_lars`, `mstep_options`, `pense_options`

---

mscale *Robust M-estimate of Scale*

Description

Compute the M-estimate of scale with MAD as initial estimate.

Usage

```
mscale(x, delta = 0.5, rho = c("bisquare", "huber"), cc, eps = 1e-08,
  maxit = 200)
```

Arguments

`x`	numeric vector of observations.
`delta`	target value of the M-estimation equation.
`rho`	rho function to use in the M-estimation equation.
`cc`	non-negative constant for the chosen rho function. If missing, it will be chosen such that the expected value of the rho function under the normal model is equal to `delta`.
`eps`	threshold for convergence.
`maxit`	maximum number of iterations.

Details

This solves the M-estimation equation given by

$$\sum_{i=1}^{n} \rho(x_i / s_n; cc) = n\delta$$

Value

the M-scale as a numeric vector of length one.

Note

NA values in x are removed before calculating the M-scale.

Examples

```
## Estimate the M-scale of a vector of values
set.seed(1234)
x <- rnorm(100)
mscale(x)
mscale(x, delta = 0.25) # For a breakdown point of 25%
mscale(x, rho = "huber") # Using Huber's rho function
```

---

mstep_options *Additional Options for the Penalized EN MM-estimator*

Description

Additional Options for the Penalized EN MM-estimator

Usage

```
mstep_options(cc = 3.44, maxit = 1000, eps = 1e-06,
  adjust_bdp = FALSE, verbosity = 0, en_correction = TRUE)
```

Arguments

cc	tuning constant for the M-estimator.
maxit	maximum number of iterations allowed.
eps	numeric tolerance for convergence.
adjust_bdp	should the breakdown point be adjusted based on the effective degrees of freedom?
verbosity	verbosity of the algorithm.
en_correction	should the corrected EN estimator be used to choose the optimal lambda with CV. If TRUE, as by default, the estimator is "bias corrected".

Value

a checked options list.

See Also

Other specifying additional options: en_options_aug_lars, initest_options, pense_options

pense *Penalized Elastic Net S-estimators for Regression*

Description

Computes the highly robust Penalized Elastic Net S-estimators (PENSE) for linear regression models.

Usage

```
pense(x, y, alpha = 0.5, nlambda = 50, lambda, lambda_min_ratio,
  standardize = TRUE, initial = c("warm", "cold"), warm_reset = 10,
  cv_k = 5, cv_objective, ncores = getOption("mc.cores", 1L),
  cl = NULL, options = pense_options(),
  init_options = initest_options(), en_options = en_options_aug_lars())
```

Arguments

x	design matrix with predictors.
y	response vector.
alpha	elastic net mixing parameter with $0 \leq \alpha \leq 1$. `alpha = 1` is the LASSO penalty, and `alpha = 0` the Ridge penalty.
nlambda	if `lambda` is not given or `NULL` (default), a grid of `nlambda` lambda values is generated based on the data.
lambda	a single value or a grid of values for the regularization parameter lambda. Assumed to be on the same scale as the data and adjusted for S-estimation. If missing a grid of lambda values is automatically generated (see parameter `nlambda`). If given and `standardize = TRUE`, the lambda values will be adjusted accordingly.
lambda_min_ratio	If the grid should be chosen automatically, the ratio of the smallest lambda to the (computed) largest lambda.
standardize	should the data be standardized robustly? Estimates are returned on the original scale. Defaults to `TRUE`.
initial	how to initialize the estimator at a new lambda in the grid. The default, `"warm"`, computes a cold initial estimator at several lambda values and uses the PENSE coefficient to warm-start the estimator at the next larger lambda value. At the largest value in the lambda grid, PENSE will be initialized with the 0-vector. `"cold"` computes the full initial estimator at every lambda value.
warm_reset	if `initial = "warm"` (default), how many cold initial estimates be computed?
cv_k	number of cross-validation segments to use to choose the optimal lambda from the grid. If only a single value of lambda is given, cross-validation can still done to estimate the prediction performance at this particular lambda.

cv_objective	a function (name) to compute the CV performance. By default, the robust tau-scale is used.
ncores, cl	the number of processor cores or an actual parallel cluster to use to estimate the optimal value of lambda. See makeCluster on how to create a cluster manually.
options	additional options for the PENSE algorithm. See pense_options for details.
init_options	additional options for computing the cold initial estimates. Ignored if initial = "warm" and warm_reset = 0. See initest_options for details.
en_options	additional options for the EN algorithm. See elnet and en_options for details.

Details

The PENSE estimate minimizes the robust M-scale of the residuals penalized by the L1 and L2 norm of the regression coefficients (elastic net penalty). The level of penalization is chosen to minimize the cv_k-fold cross-validated prediction error (using a robust measure).

Value

An object of class "pense" with elements

lambda	grid of regularization parameter values for which an estimate is available.
lambda_opt	the optimal value of the regularization parameter according to CV.
coefficients	a sparse matrix of coefficients for each lambda in the grid.
residuals	a matrix of residuals for each lambda in the grid.
cv_lambda_grid	a data frame with CV prediction errors and several statistics of the solutions.
scale	the estimated scales each lambda in the grid.
objective	value of the objective function at each lambda in the grid.
adjusted	necessary information to compute the corrected EN estimates.
call	the call that produced this object.
...	values of the given arguments.

Initial Estimate

By default (initial == "warm"), the method does not compute a full initial estimate at each lambda value in the grid, but only at warm_reset of the lambda values. At the remaining lambda values, the estimate at the previous lambda value is used to initialize the estimator (the lambda grid is first traversed in descending and then in ascending direction). If warm_reset is 1, only the 0-vector is used to initialize PENSE at the largest penalty value. No further initial estimates are computed.

If initial == "cold", a full initial estimate is computed at each lambda value. This is equal to setting warm_reset to length(lambda).

See Also

To improve the S-estimate with an M-step, see pensem.

Examples

```
##
## A very simple example on artificial data
##

# Generate some dummy data
set.seed(12345)
n <- 30
p <- 15
x <- 1 + matrix(rnorm(n * p), ncol = p)
y <- x %*% c(2:5, numeric(p - 4)) + rnorm(n)

x_test <- matrix(rnorm(10 * n * p), ncol = p)
y_test <- x_test %*% c(2:5, numeric(p - 4)) + rnorm(n)

# Compute the S-estimator with an EN penalty for 30 lambda values
# (Note: In real applications, warm_reset should be at least 5)
set.seed(1234)
est <- pense(
    x, y,
    alpha = 0.6,
    nlambda = 20,
    warm_reset = 1L,
    cv_k = 3
)

# We can plot the CV prediction error curve
plot(est)

# What is the RMSPE on test data
(rmspe <- sqrt(mean((y_test - predict(est, newdata = x_test))^2)))

##
## What happens if we replace 5 observations in the dummy data
## with outliers?
##
y_out <- y
y_out[1:3] <- rnorm(3, -500)

# Compute the S-estimator again
# (Note: In real applications, warm_reset should be at least 5)
set.seed(12345)
est_out <- pense(
    x, y_out,
    alpha = 0.6,
    nlambda = 20,
    warm_reset = 1L,
    cv_k = 3
)

# How does the RMSPE compare?
rmspe_out <- sqrt(mean((y_test - predict(est_out, newdata = x_test))^2))
```

```
c(rmspe = rmspe, rmspe_out = rmspe_out)
```

---

pensem *Perform an M-step after the EN S-Estimator*

Description

Compute the PENSEM estimate, an efficient and robust elastic net estimator for linear regression.

Usage

```
pensem(x, ...)

## Default S3 method:
pensem(x, y, alpha = 0.5, nlambda = 50, lambda,
  lambda_s, lambda_min_ratio, standardize = TRUE, initial = c("warm",
  "cold"), warm_reset = 10, cv_k = 5, cv_objective,
  ncores = getOption("mc.cores", 1L), cl = NULL,
  s_options = pense_options(), mm_options = mstep_options(),
  init_options = initest_options(), en_options = en_options_aug_lars(),
  ...)

## S3 method for class 'pense'
pensem(x, alpha, scale, nlambda = 50, lambda,
  lambda_min_ratio, standardize, cv_k = 5, cv_objective,
  ncores = getOption("mc.cores", 1L), cl = NULL,
  mm_options = mstep_options(), en_options, x_train, y_train, ...)
```

Arguments

x	either a numeric data matrix or a fitted PENSE estimate obtained from pense.
...	currently ignored.
y	numeric response vector.
alpha	elastic net mixing parameter with $0 \leq \alpha \leq 1$. alpha = 1 is the LASSO penalty, and alpha = 0 the Ridge penalty. If a pense object is supplied as first argument,
nlambda	if lambda is not given, a grid of nlambda lambda values is generated based on the data.
lambda	a single value or a grid of values for the regularization parameter of the M-step. Assumed to be on the same scale as the data. If missing, a grid of lambda values is automatically generated (see parameter nlambda). If supplied and standardize = TRUE, the lambda values will be adjusted accordingly.
lambda_s	regularization parameter for the *S-estimator*. If missing, a grid of lambda values is chosen automatically. If standardize = TRUE, the lambda values will be adjusted accordingly.

`lambda_min_ratio`
: If the grid should be chosen automatically, the ratio of the smallest lambda to the (computed) largest lambda.

`standardize`
: should the data be standardized robustly? Estimates are returned on the original scale. Defaults to `TRUE`.

`initial`
: how to initialize the estimator at a new lambda in the grid. The default, `"warm"`, computes a cold initial estimator at several lambda values and uses the PENSE coefficient to warm-start the estimator at the next larger lambda value. At the largest value in the lambda grid, PENSE will be initialized with the 0-vector. `"cold"` computes the full initial estimator at every lambda value.

`warm_reset`
: if `initial = "warm"` (default), how many cold initial estimates be computed?

`cv_k`
: perform k-fold CV to choose the optimal lambda for prediction.

`cv_objective`
: a function (name) to compute the CV performance. By default, the robust tau-scale is used.

`ncores, cl`
: use multiple cores or the supplied cluster for the cross-validation. See `pense` for more details.

`s_options`
: additional options for the PENSE algorithm. See `pense_options` for details.

`mm_options`
: additional options for the M-step.

`init_options`
: additional options for computing the cold initial estimates. Ignored if `initial = "warm"` and `warm_reset = 0`. See `initest_options` for details.

`en_options`
: additional options for the EN algorithm. See `elnet` and `en_options` for details.

`scale`
: initial scale estimate for the M step. By default the S-scale from the initial estimator (`x`) is used.

`x_train, y_train`
: override arguments provided to the original call to `pense`.

Details

Performs an M-step using the S-estimator at the optimal penalty parameter as returned from `pense` as the initial estimate. For "fat" datasets, the initial scale as returned by the S-estimate is adjusted according to Maronna & Yohai (2010).

Value

An object of class `"pensem"`. All elements as an object of class `pense` as well as the following:

`init_scale`
: the initial scale estimate used in the M step.

`sest`
: the PENSE estimate used to initialize the M step.

`bdp`
: breakdown point of the MM-estimator.

References

Maronna, R. and Yohai, V. (2010). Correcting MM estimates for "fat" data sets. *Computational Statistics & Data Analysis*, **54**:31683173.

See Also

pense to compute only the S-estimator.

Examples

```
##
## A very simple example on artificial data
##

# Generate some dummy data
set.seed(12345)
n <- 30
p <- 15
x <- 1 + matrix(rnorm(n * p), ncol = p)
y <- x %*% c(2:5, numeric(p - 4)) + rnorm(n)

x_test <- 1 + matrix(rnorm(10 * n * p), ncol = p)
y_test <- x_test %*% c(2:5, numeric(p - 4)) + rnorm(n)

# Compute the MM-estimator with an EN penalty for 30 lambda values
# (Note: In real applications, warm_reset should be at least 5)
set.seed(1234)
est_mm <- pensem(
    x, y,
    alpha = 0.7,
    nlambda = 20,
    warm_reset = 1L,
    cv_k = 3
)

# We can plot the CV prediction error curve
plot(est_mm)

# What is the RMSPE on test data
(rmspe <- sqrt(mean((y_test - predict(est_mm, newdata = x_test))^2)))

##
## This is the same as computing first the S-estimator and adding the
## M-step afterwards
##
set.seed(1234)
est_s <- pense(
    x, y,
    alpha = 0.7,
    nlambda = 20,
    warm_reset = 1L,
    cv_k = 3
)

est_mm_2 <- pensem(
    est_s,
    nlambda = 20,
```

```
    cv_k = 3
)

## The initial S-estimate is the same used in both `pensem` calls
## because the seed which governs the CV to select the optimal lambda was the
## same
sum(abs(est_s$coefficients - est_mm$sest$coefficients))
## Therefore, the MM-estimate at each lambda is also the same
sum(abs(est_mm_2$coefficients - est_mm$coefficients))
```

`pense_options` *Additional Options for the Penalized EN S-estimator*

Description

Additional Options for the Penalized EN S-estimator

Usage

```
pense_options(delta = 0.25, maxit = 1000, eps = 1e-06,
  mscale_eps = 1e-08, mscale_maxit = 200, verbosity = 0, cc,
  en_correction = TRUE)
```

Arguments

`delta`	desired breakdown point of the resulting estimator.
`maxit`	maximum number of iterations allowed.
`eps`	numeric tolerance for convergence.
`mscale_eps, mscale_maxit`	maximum number of iterations and numeric tolerance for the M-scale.
`verbosity`	verbosity of the algorithm.
`cc`	tuning constant for the S-estimator. Default is to chosen based on the breakdown point `delta`. Should never have to be changed.
`en_correction`	should the corrected EN estimator be used to choose the optimal lambda with CV. If `TRUE`, as by default, the estimator is "bias corrected".

Value

a checked options list.

See Also

Other specifying additional options: `en_options_aug_lars`, `initest_options`, `mstep_options`

plot.cv_elnetfit *Plot Method for Cross-Validated Elastic Net Models*

Description

Plot the cross-validation error or the coefficient path for a fitted elastic net regression model.

Usage

```
## S3 method for class 'cv_elnetfit'
plot(x, what = c("cv", "coef.path"), ...)
```

Arguments

x	a fitted, cross-validated EN model from elnet_cv.
what	plot either the cross-validated prediction error ("cv"; default) or the coefficient paths.
...	currently ignored.

Examples

```
# Generate data with highly correlated groups of variables
set.seed(12345)
n <- 100
p <- 20
x <- 1 + matrix(rnorm(n * p), ncol = p)
x[, 2] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 3] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 5] <- x[, 4] + rnorm(n, sd = 0.01)
x[, 6] <- x[, 4] + rnorm(n, sd = 0.01)

y <- x %*% c(rep(c(2, 5), each = 3), numeric(p - 6)) + rnorm(n)

# Compute the classical EN and select the optimal lambda by CV
set.seed(1234)
est_en <- elnet_cv(
    x, y,
    alpha = 0.5
)

# By default, `plot` shows the CV prediction error
plot(est_en)
# We can also plot the coefficient paths
plot(est_en, what = "coef.path")

# Compute the LASSO solution
set.seed(1234)
est_lasso <- elnet_cv(
    x, y,
```

```
    alpha = 1
)

plot(est_lasso)
plot(est_lasso, what = "coef.path")
```

`plot.elnetfit` *Plot Method for Fitted Elastic Net Models*

Description

Plot the coefficient path for a fitted elastic net regression model.

Usage

```
## S3 method for class 'elnetfit'
plot(x, ...)
```

Arguments

`x`	a fitted EN model from `elnet`.
`...`	currently ignored.

Examples

```
# Generate data with highly correlated groups of variables
set.seed(12345)
n <- 100
p <- 20
x <- 1 + matrix(rnorm(n * p), ncol = p)
x[, 2] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 3] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 5] <- x[, 4] + rnorm(n, sd = 0.01)
x[, 6] <- x[, 4] + rnorm(n, sd = 0.01)

y <- x %*% c(rep(c(2, 5), each = 3), numeric(p - 6)) + rnorm(n)

# Compute the classical EN
est_en <- elnet(
    x, y,
    alpha = 0.5
)

# The `plot` method for the `elnet` output shows the coefficient paths
plot(est_en)
# --> from the paths it can be seen that the variables 1,2,3 and 4,5,6 are
# grouped together

# Compute the LASSO solution
```

```
est_lasso <- elnet(
    x, y,
    alpha = 1
)

# The LASSO solution, on the other hand, can not unveil the grouping
# structure
plot(est_lasso)

# Compute the Ridge solution
est_ridge <- elnet(
    x, y,
    alpha = 0
)

# The Ridge also shows the grouping structure very nicely
plot(est_ridge)
```

`plot.pense` *Plot Method for Fitted Penalized Elastic Net S/MM-Estimates of Regression*

Description

Plot the cross-validation error or the coefficient path for a fitted PENSE estimate.

Usage

```
## S3 method for class 'pense'
plot(x, what = c("cv", "coef.path"), ...)
```

Arguments

`x`	a PENSE or PENSEM estimate from `pense` or `pensem`.
`what`	plot either the cross-validated prediction error (`"cv"`; default) or the coefficient paths.
`...`	currently ignored.

Examples

```
# Generate data with highly correlated groups of variables and some outliers
set.seed(12345)
n <- 50
n_out <- 3
p <- 20
x <- 1 + matrix(rnorm(n * p), ncol = p)
x[, 2] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 3] <- x[, 1] + rnorm(n, sd = 0.01)
```

```
x[, 5] <- x[, 4] + rnorm(n, sd = 0.01)
x[, 6] <- x[, 4] + rnorm(n, sd = 0.01)

y <- x %*% c(rep(c(2, 5), each = 3), numeric(p - 6)) + rnorm(n)

y[seq_len(n_out)] <- rnorm(n_out, -100, sd = 3)

# Compute the PENSE estimator
set.seed(1234)
est_en <- pense(x, y, alpha = 0.5, warm_reset = 1, cv_k = 3, nlambda = 25)

# The `plot` method by default shows the CV prediction error
plot(est_en)

# We can also plot the coefficient paths which shows that variables 1-3 and
# 4-6 appear to be grouped
plot(est_en, "coef.path")

# Compute the LASSO solution
set.seed(1234)
est_lasso <- pense(x, y, alpha = 1, warm_reset = 1, cv_k = 3, nlambda = 25,
                   init_options = initest_options(psc_method = "exact"))
plot(est_lasso)
plot(est_lasso, "coef.path")
# The coefficient path from the LASSO does not show the grouping anymore.
```

`predict.elnetfit` *Predict Method for the classical Elastic Net Estimator*

Description

Predict Method for the classical Elastic Net Estimator

Usage

```
## S3 method for class 'elnetfit'
predict(object, newdata, lambda, exact = FALSE,
  correction = TRUE, ...)
```

Arguments

`object`	an object of type `elnetfit` to use for prediction.
`newdata`	an optional design matrix
`lambda`	the value of the penalty parameter. Default is to use the optimal lambda `lambda_opt`.
`exact`	if the lambda is not part of the lambda grid, should the estimates be obtained by linear interpolation between the nearest lambda values (default) or computed exactly.
`correction`	should a correction factor be applied to the EN estimate?
`...`	currently ignored.

Value

a numeric vector of predicted values for the given lambda.

Examples

```
# Generate data with highly correlated groups of variables
set.seed(12345)
n <- 100
p <- 20
x <- 1 + matrix(rnorm(n * p), ncol = p)
x[, 2] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 3] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 5] <- x[, 4] + rnorm(n, sd = 0.01)
x[, 6] <- x[, 4] + rnorm(n, sd = 0.01)

y <- drop(x %*% c(rep(c(2, 5), each = 3), numeric(p - 6)) + rnorm(n))

# Compute the classical EN and select the optimal lambda by CV
set.seed(1234)
est_en_cv <- elnet_cv(
    x, y,
    alpha = 0.5,
    correction = TRUE
)

# For cross-validated EN fits, the `coef`, `predict`, and `residuals` methods
# return/use the estimated coefficients at the "optimal" lambda
coef(est_en_cv) # Extract coefficients
predict(est_en_cv) # Extract fitted values
predict(est_en_cv, newdata = x) # Predict values
residuals(est_en_cv) # Extract residuals

# We can also request the coefficient at another lambda. By default,
# this will interpolate between the solutions at the two surrounding
# lambda values.
coef(est_en_cv, lambda = 6)

# If needed, the solution at the given lambda can also be computed exactly.
coef(est_en_cv, lambda = 6, exact = TRUE)


# If we compute the EN estimator without choosing an optimal lambda,
# the lambda parameter needs to be specified in the call to coef
est_en <- elnet(
    x, y,
    alpha = 0.5
)

# Without specifying lambda, the `coef` method would raise an error.
coef(est_en, lambda = 6)
```

predict.pense *Predict Method for Penalized Elastic Net S- and MM-estimators*

Description

Predict Method for Penalized Elastic Net S- and MM-estimators

Usage

```
## S3 method for class 'pense'
predict(object, newdata, lambda, exact = FALSE,
  correction = TRUE, ...)
```

Arguments

object	an object of type pense or pensem to use for prediction.
newdata	an optional design matrix
lambda	the value of the penalty parameter. Default is to use the optimal lambda lambda_opt.
exact	if the lambda is not part of the lambda grid, should the estimates be obtained by linear interpolation between the nearest lambda values (default) or computed exactly.
correction	should a correction factor be applied to the PENSE(M) estimate?
...	currently ignored.

Value

a numeric vector of predicted values for the given lambda.

Examples

```
# Generate data with highly correlated groups of variables and some outliers
set.seed(12345)
n <- 50
n_out <- 3
p <- 20
x <- 1 + matrix(rnorm(n * p), ncol = p)
x[, 2] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 3] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 5] <- x[, 4] + rnorm(n, sd = 0.01)
x[, 6] <- x[, 4] + rnorm(n, sd = 0.01)

y <- x %*% c(rep(c(2, 5), each = 3), numeric(p - 6)) + rnorm(n)

y[seq_len(n_out)] <- rnorm(n_out, -100, sd = 3)

# Compute the PENSE estimator
set.seed(1234)
```

```
est_en <- pense(x, y, alpha = 0.5, warm_reset = 1, cv_k = 3)

# From the fitted model we can extract the coefficients, fitted values, and
# residuals at the "optimal" lambda as chosen by CV.
coef(est_en) # Extract coefficients
predict(est_en) # Extract fitted values
predict(est_en, newdata = x) # Predict values
residuals(est_en) # Extract residuals

# We can also request the coefficients/predicitons/residuals at another lambda.
# If the requested lambda is not in the original lambda grid, the methods
# will approximate the coefficient vector by linear interpolation of
# the solutions at the surrounding lambda values.
coef(est_en, lambda = 5)

# If the exact solution is needed, this can be requested
coef(est_en, lambda = 5, exact = TRUE)
residuals(est_en, lambda = 5, exact = TRUE)
```

`prinsens` *Principal Sensitivity Components*

Description

Compute the principal sensitivity components (PSC) for regression.

Usage

```
prinsens(x, y, method = c("ols", "en"), intercept = TRUE, alpha,
  lambda, en_options = en_options_aug_lars())
```

Arguments

`x`	data matrix with predictors
`y`	response vector
`method`	use ordinary least squares (`"ols"`) or elastic net (`"en"`) to compute the PSCs.
`intercept`	Should an intercept be added or not.
`alpha, lambda`	The values for the parameters controlling the penalization for elastic net.
`en_options`	additional options for the EN algorithm. See `en_options` for details.

Value

A numeric matrix with as many rows as `x` and as many columns as PSCs found (at most the number of columns in `x` plus one for the intercept). Each column is a PSC.

References

Pena, D., and Yohai, V.J. (1999). A Fast Procedure for Outlier Diagnostics in Large Regression Problems. *Journal of the American Statistical Association*, **94**(446), 434-445. http://doi.org/10.2307/2670164

Examples

```
set.seed(12345)
n <- 50
n_out <- 6
p <- 500
beta <- c(2:5, numeric(p - 4))
x <- 1 + matrix(rnorm(n * p), ncol = p)
y <- x %*% beta  + rnorm(n)

# add outliers to y
y[seq_len(n_out)] <- rnorm(n_out, -100, 0.5)

# Compute Principal Sensitivity Components
pscs <- prinsens(
    x, y,
    alpha = 0.8,
    method = "en",
    lambda = 10
)

# The 6 outlying observations are reflected in the 15th PSC
colors <- rep.int(c(2, 1), times = c(n_out, n - n_out))
ord <- order(abs(pscs[, 15L]))
plot(abs(pscs[ord, 15L]), col = colors[ord], xlab = "Sorted Index",
     ylab = "Absolute PSC score")
```

`residuals.elnetfit` *Extract Residuals from a Fitted Elastic-Net Estimator*

Description

Extract Residuals from a Fitted Elastic-Net Estimator

Usage

```
## S3 method for class 'elnetfit'
residuals(object, lambda, exact = FALSE, ...)
```

Arguments

object	a elnet estimate to extract the residuals from.
lambda	the value of the penalty parameter. Default is to use the optimal lambda object$lambda_opt if the given estimator was cross-validated or the smallest lambda if not.
exact	if the lambda is not part of the lambda grid, should the estimates be obtained by linear interpolation between the nearest lambda values (default) or computed exactly.
...	currently ignored.

Value

a numeric vector of residuals for the given lambda.

Examples

```
# Generate data with highly correlated groups of variables
set.seed(12345)
n <- 100
p <- 20
x <- 1 + matrix(rnorm(n * p), ncol = p)
x[, 2] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 3] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 5] <- x[, 4] + rnorm(n, sd = 0.01)
x[, 6] <- x[, 4] + rnorm(n, sd = 0.01)

y <- drop(x %*% c(rep(c(2, 5), each = 3), numeric(p - 6)) + rnorm(n))

# Compute the classical EN and select the optimal lambda by CV
set.seed(1234)
est_en_cv <- elnet_cv(
    x, y,
    alpha = 0.5,
    correction = TRUE
)

# For cross-validated EN fits, the `coef`, `predict`, and `residuals` methods
# return/use the estimated coefficients at the "optimal" lambda
coef(est_en_cv) # Extract coefficients
predict(est_en_cv) # Extract fitted values
predict(est_en_cv, newdata = x) # Predict values
residuals(est_en_cv) # Extract residuals

# We can also request the coefficient at another lambda. By default,
# this will interpolate between the solutions at the two surrounding
# lambda values.
coef(est_en_cv, lambda = 6)

# If needed, the solution at the given lambda can also be computed exactly.
coef(est_en_cv, lambda = 6, exact = TRUE)
```

```
# If we compute the EN estimator without choosing an optimal lambda,
# the lambda parameter needs to be specified in the call to coef
est_en <- elnet(
    x, y,
    alpha = 0.5
)

# Without specifying lambda, the `coef` method would raise an error.
coef(est_en, lambda = 6)
```

`residuals.pense` *Extract Residuals from a Fitted Penalized Elastic-Net S/MM-estimator*

Description

Extract Residuals from a Fitted Penalized Elastic-Net S/MM-estimator

Usage

```
## S3 method for class 'pense'
residuals(object, lambda, exact = FALSE,
  correction = TRUE, ...)
```

Arguments

`object`	a PENSE or PENSEM estimate to extract the residuals from.
`lambda`	the value of the penalty parameter. Default is to use the optimal lambda `object$lambda_opt`.
`exact`	if the lambda is not part of the lambda grid, should the estimates be obtained by linear interpolation between the nearest lambda values (default) or computed exactly.
`correction`	should a correction factor be applied to the EN estimate? See `elnet` for details on the applied correction.
`...`	currently ignored.

Value

a numeric vector of residuals for the given lambda.

Examples

```
# Generate data with highly correlated groups of variables and some outliers
set.seed(12345)
n <- 50
n_out <- 3
p <- 20
x <- 1 + matrix(rnorm(n * p), ncol = p)
x[, 2] <- x[, 1] + rnorm(n, sd = 0.01)
```

```
x[, 3] <- x[, 1] + rnorm(n, sd = 0.01)
x[, 5] <- x[, 4] + rnorm(n, sd = 0.01)
x[, 6] <- x[, 4] + rnorm(n, sd = 0.01)

y <- x %*% c(rep(c(2, 5), each = 3), numeric(p - 6)) + rnorm(n)

y[seq_len(n_out)] <- rnorm(n_out, -100, sd = 3)

# Compute the PENSE estimator
set.seed(1234)
est_en <- pense(x, y, alpha = 0.5, warm_reset = 1, cv_k = 3)

# From the fitted model we can extract the coefficients, fitted values, and
# residuals at the "optimal" lambda as chosen by CV.
coef(est_en) # Extract coefficients
predict(est_en) # Extract fitted values
predict(est_en, newdata = x) # Predict values
residuals(est_en) # Extract residuals

# We can also request the coefficients/predicitons/residuals at another lambda.
# If the requested lambda is not in the original lambda grid, the methods
# will approximate the coefficient vector by linear interpolation of
# the solutions at the surrounding lambda values.
coef(est_en, lambda = 5)

# If the exact solution is needed, this can be requested
coef(est_en, lambda = 5, exact = TRUE)
residuals(est_en, lambda = 5, exact = TRUE)
```

Index

coef.elnetfit, 2
coef.pense, 4

elnet, *4*, 5, *8, 9*, *17*, *20*, *24*, *31, 32*
elnet_cv, *7*, 8, *23*
en_options, *5*, *9, 10*, *17*, *20*, *29*
en_options *(*en_options_aug_lars*)*, 12
en_options_aug_lars, 12, *14, 15*, *22*
en_options_dal *(*en_options_aug_lars*)*, 12
enpy, 10

initest_options, *10*, *12*, 13, *15*, *17*, *20*, *22*

makeCluster, *17*
mscale, 14
mstep_options, *12*, *14*, 15, *22*

pense, 16, *19–21*, *25*
pense_options, *12*, *14, 15*, *17*, *20*, 22
pensem, *17*, 19, *25*
plot.cv_elnetfit, 23
plot.elnetfit, 24
plot.pense, 25
predict.elnetfit, 26
predict.pense, 28
prinsens, 29

residuals.elnetfit, 30
residuals.pense, 32