Package 'L2E'

October 12, 2022

Type Package

Title Robust Structured Regression via the L2 Criterion

Version 2.0

Description An implementation of a computational framework for performing robust structured regression with the L2 criterion
from Chi and Chi (2021+). Improvements using the majorization-minimization (MM) principle from Liu, Chi, and
Lange (2022+) added in Version 2.0.

Maintainer Jocelyn Chi <jocetchi@gmail.com>

Depends R (>= 3.5.0), osqp

Imports isotone, cobs, ncvreg, Matrix, signal, robustbase

Suggests knitr, rmarkdown, ggplot2, latex2exp

VignetteBuilder knitr

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Xiaoqian Liu [aut, ctb],
Jocelyn Chi [aut, cre],
Lisa Lin [ctb],
Kenneth Lange [aut],
Eric Chi [aut]

Repository CRAN

Date/Publication 2022-09-08 21:13:00 UTC

R topics documented:

---

bank *Bank data*

Description

Data from an Italian bank on 1,949 customers. The response y is the amount of money made over a year. The 13 covariates are possible macroservices the customers can sign up for.

Format

A data frame with 1949 rows and 14 variables

References

Marco Riani, Andrea Cerioli, and Anthony C. Atkinson (2014). Monitoring robust regression. Electronic Journal of Statistics, Volume 8, 646-677.

`CV_L2E_sparse_dist` *Cross validation for L2E sparse regression with distance penalization*

Description

`CV_L2E_sparse_dist` performs k-fold cross-validation for robust sparse regression under the L2 criterion with distance penalty

Usage

```
CV_L2E_sparse_dist(
  y,
  X,
  beta0,
  tau0,
  kSeq,
  rhoSeq,
  nfolds = 5,
  seed = 1234,
  method = "median",
  max_iter = 100,
  tol = 1e-04,
  trace = TRUE
)
```

Arguments

`y`	Response vector
`X`	Design matrix
`beta0`	Initial vector of regression coefficients, can be omitted
`tau0`	Initial precision estimate, can be omitted
`kSeq`	A sequence of tuning parameter k, the number of nonzero entries in the estimated coefficients
`rhoSeq`	A sequence of tuning parameter rho, can be omitted
`nfolds`	The number of cross-validation folds. Default is 5.
`seed`	Users can set the seed of the random number generator to obtain reproducible results.

method	Median or mean to compute the objective
max_iter	Maximum number of iterations
tol	Relative tolerance
trace	Whether to trace the progress of the cross-validation

Value

Returns a list object containing the mean and standard error of the cross-validation error (vectors) – CVE and CVSE – for each value of k, the index of the k value with the minimum CVE and the k value itself (scalars), the index of the k value with the 1SE CVE and the k value itself (scalars), the sequence of rho and k used in the regression (vectors), and a vector listing which fold each element of y was assigned to

Examples

```
## Completes in 15 seconds

set.seed(12345)
n <- 100
tau <- 1
f <- matrix(c(rep(2,5), rep(0,45)), ncol = 1)
X <- X0 <- matrix(rnorm(n*50), nrow = n)
y <- y0 <- X0 %*% f + (1/tau)*rnorm(n)

## Clean Data
k <- c(6,5,4)
# (not run)
# cv <- CV_L2E_sparse_dist(y=y, X=X, kSeq=k, nfolds=2, seed=1234)
# (k_min <- cv$k.min) ## selected number of nonzero entries

# sol <- L2E_sparse_dist(y=y, X=X, kSeq=k_min)
# r <- y - X %*% sol$Beta
# ix <- which(abs(r) > 3/sol$Tau)
# l2e_fit <- X %*% sol$Beta

# plot(y, l2e_fit, ylab='Predicted values', pch=16, cex=0.8)
# points(y[ix], l2e_fit[ix], pch=16, col='blue', cex=0.8)

## Contaminated Data
i <- 1:5
y[i] <- 2 + y0[i]
X[i,] <- 2 + X0[i,]

# (not run)
# cv <- CV_L2E_sparse_dist(y=y, X=X, kSeq=k, nfolds=2, seed=1234)
# (k_min <- cv$k.min) ## selected number of nonzero entries

# sol <- L2E_sparse_dist(y=y, X=X, kSeq=k_min)
# r <- y - X %*% sol$Beta
# ix <- which(abs(r) > 3/sol$Tau)
# l2e_fit <- X %*% sol$Beta
```

```
# plot(y, l2e_fit, ylab='Predicted values', pch=16, cex=0.8)
# points(y[ix], l2e_fit[ix], pch=16, col='blue', cex=0.8)
```

`CV_L2E_sparse_ncv` *Cross validation for L2E sparse regression with existing penalization methods*

Description

`CV_L2E_sparse_ncv` performs k-fold cross-validation for robust sparse regression under the L2 criterion. Available penalties include lasso, MCP and SCAD.

Usage

```
CV_L2E_sparse_ncv(
  y,
  X,
  beta0,
  tau0,
  lambdaSeq,
  penalty = "MCP",
  nfolds = 5,
  seed = 1234,
  method = "median",
  max_iter = 100,
  tol = 1e-04,
  trace = TRUE
)
```

Arguments

`y`	Response vector
`X`	Design matrix
`beta0`	Initial vector of regression coefficients, can be omitted
`tau0`	Initial precision estimate, can be omitted
`lambdaSeq`	A decreasing sequence of tuning parameter lambda, can be omitted
`penalty`	Available penalties include lasso, MCP and SCAD.
`nfolds`	The number of cross-validation folds. Default is 5.
`seed`	Users can set the seed of the random number generator to obtain reproducible results.
`method`	Median or mean to compute the objective
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance
`trace`	Whether to trace the progress of the cross-validation

Value

Returns a list object containing the mean and standard error of the cross-validation error – CVE and CVSE – for each value of k (vectors), the index of the lambda with the minimum CVE and the lambda value itself (scalars), the index of the lambda value with the 1SE CVE and the lambda value itself (scalars), the sequence of lambda used in the regression (vector), and a vector listing which fold each element of y was assigned to

Examples

```
## Completes in 20 seconds

set.seed(12345)
n <- 100
tau <- 1
f <- matrix(c(rep(2,5), rep(0,45)), ncol = 1)
X <- X0 <- matrix(rnorm(n*50), nrow = n)
y <- y0 <- X0 %*% f + (1/tau)*rnorm(n)

## Clean Data
lambda <- 10^seq(-1, -2, length.out=20)
# (not run)
# cv <- CV_L2E_sparse_ncv(y=y, X=X, lambdaSeq=lambda, penalty="SCAD", seed=1234, nfolds=2)
# (lambda_min <- cv$lambda.min)

# sol <- L2E_sparse_ncv(y=y, X=X, lambdaSeq=lambda_min, penalty="SCAD")
# r <- y - X %*% sol$Beta
# ix <- which(abs(r) > 3/sol$Tau)
# l2e_fit <- X %*% sol$Beta

# plot(y, l2e_fit, ylab='Predicted values', pch=16, cex=0.8)
# points(y[ix], l2e_fit[ix], pch=16, col='blue', cex=0.8)

## Contaminated Data
i <- 1:5
y[i] <- 2 + y0[i]
X[i,] <- 2 + X0[i,]

# (not run)
# cv <- CV_L2E_sparse_ncv(y=y, X=X, lambdaSeq=lambda, penalty="SCAD", seed=1234, nfolds=2)
# (lambda_min <- cv$lambda.min)

# sol <- L2E_sparse_ncv(y=y, X=X, lambdaSeq=lambda_min, penalty="SCAD")
# r <- y - X %*% sol$Beta
# ix <- which(abs(r) > 3/sol$Tau)
# l2e_fit <- X %*% sol$Beta

# plot(y, l2e_fit, ylab='Predicted values', pch=16, cex=0.8)
# points(y[ix], l2e_fit[ix], pch=16, col='blue', cex=0.8)
```

`CV_L2E_TF_dist` *Cross validation for L2E trend filtering regression with distance penalization*

Description

`CV_L2E_TF_dist` performs k-fold cross-validation for robust trend filtering regression under the L2 criterion with distance penalty

Usage

```
CV_L2E_TF_dist(
  y,
  X,
  beta0,
  tau0,
  D,
  kSeq,
  rhoSeq,
  nfolds = 5,
  seed = 1234,
  method = "median",
  max_iter = 100,
  tol = 1e-04,
  trace = TRUE
)
```

Arguments

`y`	Response vector
`X`	Design matrix. Default is the identity matrix.
`beta0`	Initial vector of regression coefficients, can be omitted
`tau0`	Initial precision estimate, can be omitted
`D`	The fusion matrix
`kSeq`	A sequence of tuning parameter k, the number of nonzero entries in Dbeta
`rhoSeq`	A sequence of tuning parameter rho, can be omitted
`nfolds`	The number of cross-validation folds. Default is 5.
`seed`	Users can set the seed of the random number generator to obtain reproducible results.
`method`	Median or mean to calculate the objective value
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance
`trace`	Whether to trace the progress of the cross-validation

Value

Returns a list object containing the mean and standard error of the cross-validation error – CVE and CVSE – for each value of k (vectors), the index of the k value with the minimum CVE and the k value itself (scalars), the index of the k value with the 1SE CVE and the k value itself (scalars), the sequence of rho and k used in the regression (vectors), and a vector listing which fold each element of y was assigned to

Examples

```
## Completes in 20 seconds

set.seed(12345)
n <- 100
x <- 1:n
f <- matrix(rep(c(-2,5,0,-10), each=n/4), ncol=1)
y <- y0 <- f + rnorm(length(f))

## Clean Data
plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

D <- myGetDkn(1, n)
k <- c(4,3,2)
rho <- 10^8
# (not run)
# cv <- CV_L2E_TF_dist(y=y0, D=D, kSeq=k, rhoSeq=rho, nfolds=2, seed=1234)
# (k_min <- cv$k.min)

# sol <- L2E_TF_dist(y=y0, D=D, kSeq=k_min, rhoSeq=rho)

# plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
# lines(x, f, lwd=3)
# lines(x, sol$Beta, col='blue', lwd=3)

## Contaminated Data
ix <- sample(1:n, 10)
y[ix] <- y0[ix] + 2

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

# (not run)
# cv <- CV_L2E_TF_dist(y=y, D=D, kSeq=k, rhoSeq=rho, nfolds=2, seed=1234)
# (k_min <- cv$k.min)

# sol <- L2E_TF_dist(y=y, D=D, kSeq=k_min, rhoSeq=rho)

# plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
# lines(x, f, lwd=3)
# lines(x, sol$Beta, col='blue', lwd=3)
```

`CV_L2E_TF_lasso` *Cross validation for L2E trend filtering regression with Lasso penalization*

Description

`CV_L2E_TF_lasso` performs k-fold cross-validation for robust trend filtering regression under the L2 criterion with the Lasso penalty

Usage

```
CV_L2E_TF_lasso(
  y,
  X,
  beta0,
  tau0,
  D,
  lambdaSeq,
  nfolds = 5,
  seed = 1234,
  method = "median",
  max_iter = 100,
  tol = 1e-04,
  trace = TRUE
)
```

Arguments

`y`	Response vector
`X`	Design matrix. Default is the identity matrix.
`beta0`	Initial vector of regression coefficients, can be omitted
`tau0`	Initial precision estimate, can be omitted
`D`	The fusion matrix
`lambdaSeq`	A decreasing sequence of tuning parameter lambda, can be omitted
`nfolds`	The number of cross-validation folds. Default is 5.
`seed`	Users can set the seed of the random number generator to obtain reproducible results.
`method`	Median or mean to calculate the objective value
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance
`trace`	Whether to trace the progress of the cross-validation

Value

Returns a list object containing the mean and standard error of the cross-validation error – CVE and CVSE – for each value of k (vectors), the index of the lambda with the minimum CVE and the lambda value itself (scalars), the index of the lambda value with the 1SE CVE and the lambda value itself (scalars), the sequence of lambda used in the regression (vector), and a vector listing which fold each element of y was assigned to

Examples

```
## Completes in 30 seconds

set.seed(12345)
n <- 100
x <- 1:n
f <- matrix(rep(c(-2,5,0,-10), each=n/4), ncol=1)
y <- y0 <- f + rnorm(length(f))

## Clean Data
plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

D <- myGetDkn(1, n)
lambda <- 10^seq(-1, -2, length.out=20)
# (not run)
# cv <- CV_L2E_TF_lasso(y=y0, D=D, lambdaSeq=lambda, nfolds=2, seed=1234)
# (lambda_min <- cv$lambda.min)

# sol <- L2E_TF_lasso(y=y0, D=D, lambdaSeq=lambda_min)

# plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
# lines(x, f, lwd=3)
# lines(x, sol$Beta, col='blue', lwd=3)

## Contaminated Data
ix <- sample(1:n, 10)
y[ix] <- y0[ix] + 2

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

# (not run)
# cv <- CV_L2E_TF_lasso(y=y, D=D, lambdaSeq=lambda, nfolds=2, seed=1234)
# (lambda_min <- cv$lambda.min)

# sol <- L2E_TF_lasso(y=y, D=D, lambdaSeq=lambda_min)

# plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
# lines(x, f, lwd=3)
# lines(x, sol$Beta, col='blue', lwd=3)
```

L2E *L2E*

Description

The L2E package is an R implementation of a user-friendly computational framework for performing a wide variety of robust structured regression methods via the L2 criterion.

Details

Please refer to the vignette for examples of how to use this package.

L2E_convex *L2E convex regression*

Description

`L2E_convex` performs convex regression under the L2 criterion. Available methods include proximal gradient descent (PG) and majorization-minimization (MM).

Usage

```
L2E_convex(
  y,
  beta,
  tau,
  method = "MM",
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

`y`	Response vector
`beta`	Initial vector of regression coefficients
`tau`	Initial precision estimate
`method`	Available methods include PG and MM. MM by default.
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance
`Show.Time`	Report the computing time

Value

Returns a list object containing the estimates for beta (vector) and tau (scalar), the number of outer block descent iterations until convergence (scalar), and the number of inner iterations per outer iteration for updating beta (vector) and tau or eta (vector)

Examples

```
set.seed(12345)
n <- 200
tau <- 1
x <- seq(-2, 2, length.out=n)
f <- x^4 + x
y <- f + (1/tau) * rnorm(n)

## Clean Data
plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

tau <- 1
b <- y
## Least Squares method
cvx <- fitted(cobs::conreg(y, convex=TRUE))
## MM method
sol_mm <- L2E_convex(y, b, tau)
## PG method
sol_pg <- L2E_convex(y, b, tau, method='PG')

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
lines(x, cvx, col='blue', lwd=3) ## LS
lines(x, sol_mm$beta, col='red', lwd=3) ## MM
lines(x, sol_pg$beta, col='dark green', lwd=3) ## PG

## Contaminated Data
ix <- 0:9
y[45 + ix] <- 14 + rnorm(10)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

tau <- 1
b <- y
cvx <- fitted(cobs::conreg(y, convex=TRUE))
sol_mm <- L2E_convex(y, b, tau)
sol_pg <- L2E_convex(y, b, tau, method='PG')

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
lines(x, cvx, col='blue', lwd=3) ## LS
lines(x, sol_mm$beta, col='red', lwd=3) ## MM
lines(x, sol_pg$beta, col='dark green', lwd=3) ## PG
```

`L2E_isotonic` *L2E isotonic regression*

Description

`L2E_isotonic` performs isotonic regression under the L2 criterion. Available methods include proximal gradient descent (PG) and majorization-minimization (MM).

Usage

```
L2E_isotonic(
  y,
  beta,
  tau,
  method = "MM",
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

`y`	Response vector
`beta`	Initial vector of regression coefficients
`tau`	Initial precision estimate
`method`	Available methods include PG and MM. MM by default.
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance
`Show.Time`	Report the computing time

Value

Returns a list object containing the estimates for beta (vector) and tau (scalar), the number of outer block descent iterations until convergence (scalar), and the number of inner iterations per outer iteration for updating beta (vector) and tau or eta (vector)

Examples

```
set.seed(12345)
n <- 200
tau <- 1
x <- seq(-2.5, 2.5, length.out=n)
f <- x^3
y <- f + (1/tau) * rnorm(n)

## Clean Data
```

```
plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

tau <- 1
b <- y
## Least Squares method
iso <- isotone::gpava(1:n, y)$x
## MM method
sol_mm <- L2E_isotonic(y, b, tau)
## PG method
sol_pg <- L2E_isotonic(y, b, tau, method='PG')

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
lines(x, iso, col='blue', lwd=3) ## LS
lines(x, sol_mm$beta, col='red', lwd=3) ## MM
lines(x, sol_pg$beta, col='dark green', lwd=3) ## PG

## Contaminated Data
ix <- 0:9
y[45 + ix] <- 14 + rnorm(10)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

tau <- 1
b <- y
iso <- isotone::gpava(1:n, y)$x
sol_mm <- L2E_isotonic(y, b, tau)
sol_pg <- L2E_isotonic(y, b, tau, method='PG')

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
lines(x, iso, col='blue', lwd=3) ## LS
lines(x, sol_mm$beta, col='red', lwd=3) ## MM
lines(x, sol_pg$beta, col='dark green', lwd=3) ## PG
```

`L2E_multivariate` *L2E multivariate regression*

Description

`L2E_multivariate` performs multivariate regression under the L2 criterion. Available methods include proximal gradient descent (PG) and majorization-minimization (MM).

Usage

```
L2E_multivariate(
  y,
```

```
  X,
  beta,
  tau,
  method = "MM",
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

y	Response vector
X	Design matrix
beta	Initial vector of regression coefficients
tau	Initial precision estimate
method	Available methods include PG and MM. MM by default.
max_iter	Maximum number of iterations
tol	Relative tolerance
Show.Time	Report the computing time

Value

Returns a list object containing the estimates for beta (vector) and tau (scalar), the number of outer block descent iterations until convergence (scalar), and the number of inner iterations per outer iteration for updating beta (vector) and tau or eta (vector)

Examples

```
# Bank data example
y <- bank$y
X <- as.matrix(bank[,1:13])
X0 <- as.matrix(cbind(rep(1,length(y)), X))

tau <- 1/mad(y)
b <- matrix(0, 14, 1)

# MM method
sol_mm <- L2E_multivariate(y, X0, b, tau)
r_mm <- y - X0 %*% sol_mm$beta
ix_mm <- which(abs(r_mm) > 3/sol_mm$tau)
l2e_fit_mm <- X0 %*% sol_mm$beta

# PG method
sol_pg <- L2E_multivariate(y, X0, b, tau, method="PG")
r_pg <- y - X0 %*% sol_pg$beta
ix_pg <- which(abs(r_pg) > 3/sol_pg$tau)
l2e_fit_pg <- X0 %*% sol_pg$beta

plot(y, l2e_fit_mm, ylab='Predicted values', main='MM', pch=16, cex=0.8) # MM
```

```
points(y[ix_mm], l2e_fit_mm[ix_mm], pch=16, col='blue', cex=0.8) # MM
plot(y, l2e_fit_pg, ylab='Predicted values', main='PG', pch=16, cex=0.8) # PG
points(y[ix_pg], l2e_fit_pg[ix_pg], pch=16, col='blue', cex=0.8) # PG
```

`l2e_regression` *L2E multivariate regression - PG*

Description

`l2e_regression` performs L2E multivariate regression via block coordinate descent with proximal gradient for updating both beta and tau.

Usage

```
l2e_regression(y, X, b, tau, max_iter = 100, tol = 1e-04, Show.Time = TRUE)
```

Arguments

`y`	Response vector
`X`	Design matrix
`b`	Initial vector of regression coefficients
`tau`	Initial precision estimate
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance
`Show.Time`	Report the computing time

Value

Returns a list object containing the estimates for beta (vector) and tau (scalar), the number of outer block descent iterations until convergence (scalar), and the number of inner iterations per outer iteration for updating beta and tau (vectors)

Examples

```
# Bank data example
y <- bank$y
X <- as.matrix(bank[,1:13])
X0 <- as.matrix(cbind(rep(1,length(y)), X))
tauinit <- 1/mad(y)
binit <- matrix(0, 14, 1)

sol <- l2e_regression(y, X0, binit, tauinit)
r <- y - X0 %*% sol$beta
ix <- which(abs(r) > 3/sol$tau)
l2e_fit <- X0 %*% sol$beta
```

```
plot(y, l2e_fit, ylab='Predicted values', pch=16, cex=0.8)
points(y[ix], l2e_fit[ix], pch=16, col='blue', cex=0.8)
```

`l2e_regression_convex` *L2E convex regression - PG*

Description

`l2e_regression_convex` performs L2E convex regression via block coordinate descent with proximal gradient for updating both beta and tau.

Usage

```
l2e_regression_convex(y, b, tau, max_iter = 100, tol = 1e-04, Show.Time = TRUE)
```

Arguments

`y`	Response vector
`b`	Initial vector of regression coefficients
`tau`	Initial precision estimate
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance
`Show.Time`	Report the computing time

Value

Returns a list object containing the estimates for beta (vector) and tau (scalar), the number of outer block descent iterations until convergence (scalar), and the number of inner iterations per outer iteration for updating beta and tau (vectors)

Examples

```
set.seed(12345)
n <- 200
tau <- 1
x <- seq(-2, 2, length.out=n)
f <- x^4 + x
y <- f + (1/tau) * rnorm(n)

## Clean data example
plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

tau <- 1
b <- y
sol <- l2e_regression_convex(y, b, tau)
```

```
plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
cvx <- fitted(cobs::conreg(y, convex=TRUE))
lines(x, cvx, col='blue', lwd=3)
lines(x, sol$beta, col='dark green', lwd=3)

## Contaminated data example
ix <- 0:9
y[45 + ix] <- 14 + rnorm(10)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

tau <- 1
b <- y
sol <- l2e_regression_convex(y, b, tau)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
cvx <- fitted(cobs::conreg(y, convex=TRUE))
lines(x, cvx, col='blue', lwd=3)
lines(x, sol$beta, col='dark green', lwd=3)
```

`l2e_regression_convex_MM` *L2E convex regression - MM*

Description

`l2e_regression_convex_MM` performs L2E convex regression via block coordinate descent with MM for updating beta and modified Newton for updating tau.

Usage

```
l2e_regression_convex_MM(
  y,
  beta,
  tau,
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

`y`	response
`beta`	initial vector of regression coefficients

tau	initial precision estimate
max_iter	maximum number of iterations
tol	relative tolerance
Show.Time	Report the computing time

Value

Returns a list object containing the estimates for beta (vector) and tau (scalar), the number of outer block descent iterations until convergence (scalar), and the number of inner iterations per outer iteration for updating beta and eta (vectors)

Examples

```
set.seed(12345)
n <- 200
tau <- 1
x <- seq(-2, 2, length.out=n)
f <- x^4 + x
y <- f + (1/tau)*rnorm(n)

## Clean
plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

tau <- 1
b <- y
sol <- l2e_regression_convex_MM(y, b, tau)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
cvx <- fitted(cobs::conreg(y, convex=TRUE))
lines(x, cvx, col='blue', lwd=3)
lines(x, sol$beta, col='red', lwd=3)

## Contaminated
ix <- 0:9
y[45 + ix] <- 14 + rnorm(10)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

tau <- 1
b <- y
sol <- l2e_regression_convex_MM(y, b, tau)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
cvx <- fitted(cobs::conreg(y, convex=TRUE))
lines(x, cvx, col='blue', lwd=3)
lines(x, sol$beta, col='red', lwd=3)
```

`l2e_regression_isotonic` *L2E isotonic regression - PG*

Description

`l2e_regression_isotonic` performs L2E isotonic regression via block coordinate descent with proximal gradient for updating both beta and tau.

Usage

```
l2e_regression_isotonic(
  y,
  b,
  tau,
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

`y`	Response vector
`b`	Initial vector of regression coefficients
`tau`	Initial precision estimate
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance
`Show.Time`	Report the computing time

Value

Returns a list object containing the estimates for beta (vector) and tau (scalar), the number of outer block descent iterations until convergence (scalar), and the number of inner iterations per outer iteration for updating beta and tau (vectors)

Examples

```
set.seed(12345)
n <- 200
tau <- 1
x <- seq(-2.5, 2.5, length.out=n)
f <- x^3
y <- f + (1/tau)*rnorm(n)

# Clean Data
plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
```

```
lines(x, f, lwd=3)

tau <- 1
b <- y
sol <- l2e_regression_isotonic(y, b, tau)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
iso <- isotone::gpava(1:n, y)$x
lines(x, iso, col='blue', lwd=3)
lines(x, sol$beta, col='dark green', lwd=3)

# Contaminated Data
ix <- 0:9
y[45 + ix] <- 14 + rnorm(10)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

tau <- 1
b <- y
sol <- l2e_regression_isotonic(y, b, tau)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
iso <- isotone::gpava(1:n, y)$x
lines(x, iso, col='blue', lwd=3)
lines(x, sol$beta, col='dark green', lwd=3)
```

`l2e_regression_isotonic_MM` *L2E isotonic regression - MM*

Description

`l2e_regression_isotonic_MM` performs L2E isotonic regression via block coordinate descent with MM for updating beta and modified Newton for updating tau.

Usage

```
l2e_regression_isotonic_MM(
  y,
  beta,
  tau,
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

y	Response vector
beta	Initial vector of regression coefficients
tau	Initial precision estimate
max_iter	Maximum number of iterations
tol	Relative tolerance
Show.Time	Report the computing time

Value

Returns a list object containing the estimates for beta (vector) and tau (scalar), the number of outer block descent iterations until convergence (scalar), and the number of inner iterations per outer iteration for updating beta and eta (vectors)

Examples

```
set.seed(12345)
n <- 200
tau <- 1
x <- seq(-2.5, 2.5, length.out=n)
f <- x^3
y <- f + (1/tau)*rnorm(n)

## Clean
plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

tau <- 1
b <- y
sol <- l2e_regression_isotonic_MM(y, b, tau)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
iso <- isotone::gpava(1:n, y)$x
lines(x, iso, col='blue', lwd=3)
lines(x,sol$beta,col='red',lwd=3)

## Contaminated
ix <- 0:9
y[45 + ix] <- 14 + rnorm(10)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

tau <- 1
b <- y
sol <- l2e_regression_isotonic_MM(y,b,tau)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
```

```
iso <- isotone::gpava(1:n, y)$x
lines(x, iso, col='blue', lwd=3)
lines(x,sol$beta,col='red',lwd=3)
```

`l2e_regression_MM` *L2E multivariate regression - MM*

Description

`l2e_regression_MM` performs L2E multivariate regression via block coordinate descent with MM for updating beta and modified Newton for updating tau.

Usage

```
l2e_regression_MM(
  y,
  X,
  beta,
  tau,
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

`y`	response
`X`	Design matrix
`beta`	initial vector of regression coefficients
`tau`	initial precision estimate
`max_iter`	maximum number of iterations
`tol`	relative tolerance
`Show.Time`	Report the computing time

Value

Returns a list object containing the estimates for beta (vector) and tau (scalar), the number of outer block descent iterations until convergence (scalar), and the number of inner iterations per outer iteration for updating beta and eta (vectors)

Examples

```
# Bank data example
y <- bank$y
X <- as.matrix(bank[,1:13])
X0 <- as.matrix(cbind(rep(1,length(y)), X))
tau <- 1/mad(y)
b <- matrix(0, 14, 1)

sol <- l2e_regression_MM(y, X0, b, tau)
r <- y - X0 %*% sol$beta
ix <- which(abs(r) > 3/sol$tau)
l2e_fit <- X0 %*% sol$beta

plot(y, l2e_fit, ylab='Predicted values', pch=16, cex=0.8)
points(y[ix], l2e_fit[ix], pch=16, col='blue', cex=0.8)
```

`l2e_regression_sparse_dist` *L2E sparse regression with distance penalization*

Description

`l2e_regression_sparse_dist` performs robust sparse regression under the L2 criterion with the distance penalty

Usage

```
l2e_regression_sparse_dist(
  y,
  X,
  beta,
  tau,
  k,
  rho = 1,
  stepsize = 0.9,
  sigma = 0.5,
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

`y`	Response vector
`X`	Design matrix
`beta`	Initial vector of regression coefficients

tau	Initial precision estimate
k	The number of nonzero entries in the estimated coefficients
rho	The parameter in the proximal distance algorithm
stepsize	The stepsize parameter for the MM algorithm (0, 1)
sigma	The halving parameter sigma (0, 1)
max_iter	Maximum number of iterations
tol	Relative tolerance
Show.Time	Report the computing time

`l2e_regression_sparse_ncv` *L2E sparse regression with existing penalization methods*

Description

`l2e_regression_sparse_ncv` performs robust sparse regression under the L2 criterion. Available penalties include lasso, MCP and SCAD.

Usage

```
l2e_regression_sparse_ncv(
  y,
  X,
  beta,
  tau,
  lambda,
  penalty,
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

y	Response vector
X	Design matrix
beta	Initial vector of regression coefficients
tau	Initial precision estimate
lambda	Tuning parameter
penalty	Available penalties include lasso, MCP and SCAD.
max_iter	Maximum number of iterations
tol	Relative tolerance
Show.Time	Report the computing time

`l2e_regression_TF_dist` *L2E trend filtering regression with distance penalization*

Description

`l2e_regression_TF_dist` performs robust trend filtering regression under the L2 criterion with distance penalty

Usage

```
l2e_regression_TF_dist(
  y,
  X,
  beta,
  tau,
  D,
  k,
  rho = 1,
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

`y`	Response vector
`X`	Design matrix
`beta`	Initial vector of regression coefficients
`tau`	Initial precision estimate
`D`	The fusion matrix
`k`	The number of nonzero entries in D*beta
`rho`	The parameter in the proximal distance algorithm
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance
`Show.Time`	Report the computing time

`l2e_regression_TF_lasso` *L2E trend filtering regression with Lasso penalization*

Description

`l2e_regression_TF_lasso` performs robust trend filtering regression under the L2 criterion with Lasso penalty

Usage

```
l2e_regression_TF_lasso(
  y,
  X,
  beta,
  tau,
  D,
  lambda = 1,
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

`y`	Response vector
`X`	Design matrix
`beta`	Initial vector of regression coefficients
`tau`	Initial precision estimate
`D`	The fusion matrix
`lambda`	The tuning parameter
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance
`Show.Time`	Report the computing time

`L2E_sparse_dist` *Solution path of L2E sparse regression with distance penalization*

Description

`L2E_sparse_dist` computes the solution path of the robust sparse regression under the L2 criterion with distance penalty

Usage

```
L2E_sparse_dist(
  y,
  X,
  beta0,
  tau0,
  kSeq,
  rhoSeq,
  stepsize = 0.9,
  sigma = 0.5,
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

`y`	Response vector
`X`	Design matrix
`beta0`	Initial vector of regression coefficients, can be omitted
`tau0`	Initial precision estimate, can be omitted
`kSeq`	A sequence of tuning parameter k, the number of nonzero entries in the estimated coefficients
`rhoSeq`	An increasing sequence of tuning parameter rho, can be omitted
`stepsize`	The stepsize parameter for the MM algorithm (0, 1)
`sigma`	The halving parameter sigma (0, 1)
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance
`Show.Time`	Report the computing time

Value

Returns a list object containing the estimates for beta (matrix) and tau (vector) for each value of the tuning parameter k, the path of estimates for beta (list of matrices) and tau (matrix) for each value of rho, the run time (vector) for each k, and the sequence of rho and k used in the regression (vectors)

Examples

```
set.seed(12345)
n <- 100
tau <- 1
f <- matrix(c(rep(2,5), rep(0,45)), ncol = 1)
X <- X0 <- matrix(rnorm(n*50), nrow = n)
y <- y0 <- X0 %*% f + (1/tau)*rnorm(n)

## Clean Data
k <- 5
sol <- L2E_sparse_dist(y=y, X=X, kSeq=k)
r <- y - X %*% sol$Beta
ix <- which(abs(r) > 3/sol$Tau)
l2e_fit <- X %*% sol$Beta

plot(y, l2e_fit, ylab='Predicted values', pch=16, cex=0.8)
points(y[ix], l2e_fit[ix], pch=16, col='blue', cex=0.8)

## Contaminated Data
i <- 1:5
y[i] <- 2 + y0[i]
X[i,] <- 2 + X0[i,]

sol <- L2E_sparse_dist(y=y, X=X, kSeq=k)
r <- y - X %*% sol$Beta
ix <- which(abs(r) > 3/sol$Tau)
l2e_fit <- X %*% sol$Beta

plot(y, l2e_fit, ylab='Predicted values', pch=16, cex=0.8)
points(y[ix], l2e_fit[ix], pch=16, col='blue', cex=0.8)
```

`L2E_sparse_ncv` *Solution path of L2E sparse regression with existing penalization methods*

Description

`L2E_sparse_ncv` computes the solution path of robust sparse regression under the L2 criterion. Available penalties include lasso, MCP and SCAD.

Usage

```
L2E_sparse_ncv(
  y,
  X,
  b,
  tau,
  lambdaSeq,
```

```
  penalty = "MCP",
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

y	Response vector
X	Design matrix
b	Initial vector of regression coefficients, can be omitted
tau	Initial precision estimate, can be omitted
lambdaSeq	A decreasing sequence of values for the tuning parameter lambda, can be omitted
penalty	Available penalties include lasso, MCP and SCAD.
max_iter	Maximum number of iterations
tol	Relative tolerance
Show.Time	Report the computing time

Value

Returns a list object containing the estimates for beta (matrix) and tau (vector) for each value of the tuning parameter lambda, the run time (vector) for each lambda, and the sequence of lambda used in the regression (vector)

Examples

```
set.seed(12345)
n <- 100
tau <- 1
f <- matrix(c(rep(2,5), rep(0,45)), ncol = 1)
X <- X0 <- matrix(rnorm(n*50), nrow = n)
y <- y0 <- X0 %*% f + (1/tau)*rnorm(n)

## Clean Data
lambda <- 10^(-1)
sol <- L2E_sparse_ncv(y=y, X=X, lambdaSeq=lambda, penalty="SCAD")
r <- y - X %*% sol$Beta
ix <- which(abs(r) > 3/sol$Tau)
l2e_fit <- X %*% sol$Beta

plot(y, l2e_fit, ylab='Predicted values', pch=16, cex=0.8)
points(y[ix], l2e_fit[ix], pch=16, col='blue', cex=0.8)

## Contaminated Data
i <- 1:5
y[i] <- 2 + y0[i]
X[i,] <- 2 + X0[i,]
```

```
sol <- L2E_sparse_ncv(y=y, X=X, lambdaSeq=lambda, penalty="SCAD")
r <- y - X %*% sol$Beta
ix <- which(abs(r) > 3/sol$Tau)
l2e_fit <- X %*% sol$Beta

plot(y, l2e_fit, ylab='Predicted values', pch=16, cex=0.8)
points(y[ix], l2e_fit[ix], pch=16, col='blue', cex=0.8)
```

`L2E_TF_dist` *Solution path of the L2E trend filtering regression with distance penalization*

Description

`L2E_TF_dist` computes the solution path of the robust trend filtering regression under the L2 criterion with distance penalty

Usage

```
L2E_TF_dist(
  y,
  X,
  beta0,
  tau0,
  D,
  kSeq,
  rhoSeq,
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

`y`	Response vector
`X`	Design matrix. Default is the identity matrix.
`beta0`	Initial vector of regression coefficients, can be omitted
`tau0`	Initial precision estimate, can be omitted
`D`	The fusion matrix
`kSeq`	A sequence of tuning parameter k, the number of nonzero entries in D*beta
`rhoSeq`	An increasing sequence of tuning parameter rho, can be omitted
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance
`Show.Time`	Report the computing time

Value

Returns a list object containing the estimates for beta (matrix) and tau (vector) for each value of the tuning parameter k, the path of estimates for beta (list of matrices) and tau (matrix) for each value of rho, the run time (vector) for each k, and the sequence of rho and k used in the regression (vectors)

Examples

```
## Completes in 15 seconds

set.seed(12345)
n <- 100
x <- 1:n
f <- matrix(rep(c(-2,5,0,-10), each=n/4), ncol=1)
y <- y0 <- f + rnorm(length(f))

## Clean Data
plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

D <- myGetDkn(1, n)
k <- c(4,3,2)
rho <- 10^8
# (not run)
# sol <- L2E_TF_dist(y=y, D=D, kSeq=k, rhoSeq=rho)

# plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
# lines(x, f, lwd=3)
# lines(x, sol$Beta[,3], col='blue', lwd=3) ## k=2
# lines(x, sol$Beta[,2], col='red', lwd=3) ## k=3
# lines(x, sol$Beta[,1], col='dark green', lwd=3) ## k=4

## Contaminated Data
ix <- sample(1:n, 10)
y[ix] <- y0[ix] + 2

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

# (not run)
# sol <- L2E_TF_dist(y=y, D=D, kSeq=k, rhoSeq=rho)

# plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
# lines(x, f, lwd=3)
# lines(x, sol$Beta[,3], col='blue', lwd=3) ## k=2
# lines(x, sol$Beta[,2], col='red', lwd=3) ## k=3
# lines(x, sol$Beta[,1], col='dark green', lwd=3) ## k=4
```

L2E_TF_lasso *Solution path of the L2E trend filtering regression with Lasso*

Description

L2E_TF_lasso computes the solution path of the robust trend filtering regression under the L2 criterion with Lasso penalty

Usage

```
L2E_TF_lasso(
  y,
  X,
  beta0,
  tau0,
  D,
  lambdaSeq,
  max_iter = 100,
  tol = 1e-04,
  Show.Time = TRUE
)
```

Arguments

y	Response vector
X	Design matrix. Default is the identity matrix.
beta0	Initial vector of regression coefficients, can be omitted
tau0	Initial precision estimate, can be omitted
D	The fusion matrix
lambdaSeq	A decreasing sequence of values for the tuning parameter lambda, can be omitted
max_iter	Maximum number of iterations
tol	Relative tolerance
Show.Time	Report the computing time

Value

Returns a list object containing the estimates for beta (matrix) and tau (vector) for each value of the tuning parameter lambda, the run time (vector) for each lambda, and the sequence of lambda used in the regression (vector)

Examples

```
## Completes in 10 seconds

set.seed(12345)
n <- 100
x <- 1:n
f <- matrix(rep(c(-2,5,0,-10), each=n/4), ncol=1)
y <- y0 <- f + rnorm(length(f))
```

```
## Clean Data
plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

D <- myGetDkn(1, n)
lambda <- 10^seq(-1, -2, length.out=20)
sol <- L2E_TF_lasso(y=y, D=D, lambdaSeq=lambda)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
lines(x, sol$Beta[,1], col='blue', lwd=3) ## 1st lambda

## Contaminated Data
ix <- sample(1:n, 10)
y[ix] <- y0[ix] + 2

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)

sol <- L2E_TF_lasso(y=y, D=D, lambdaSeq=lambda)

plot(x, y, pch=16, cex.lab=1.5, cex.axis=1.5, cex.sub=1.5, col='gray')
lines(x, f, lwd=3)
lines(x, sol$Beta[,1], col='blue', lwd=3) ## 1st lambda
```

`myGetDkn` *Compute kth order differencing matrix*

Description

`myGetDkn` computes the kth order differencing matrix for use in trend filtering regression

Usage

```
myGetDkn(k, n)
```

Arguments

`k`	Order of the differencing matrix
`n`	Number of time points

Value

Returns a Matrix object as the kth order differencing matrix

objective *Objective function of the L2E regression - eta*

Description

`objective` computes the objective of the L2E regression in terms of eta

Usage

```
objective(eta, r, method = "mean")
```

Arguments

`eta`	The current estimate of eta
`r`	Vector of residuals
`method`	Mean or median

Value

Returns the output of the objective function (scalar)

objective_tau *Objective function of the L2E regression - tau*

Description

`objective_tau` computes the objective of the L2E regression in terms of tau

Usage

```
objective_tau(tau, r, method = "mean")
```

Arguments

`tau`	The current estimate of tau
`r`	Vector of residuals
`method`	Mean or median

Value

Returns the output of the objective function (scalar)

update_beta_convex *Beta update in L2E convex regression - PG*

Description

update_beta_convex updates beta for L2E convex regression using PG

Usage

```
update_beta_convex(y, b, tau, max_iter = 100, tol = 1e-04)
```

Arguments

y	Response vector
b	Current estimate for beta
tau	Current estimate for tau
max_iter	Maximum number of iterations
tol	Relative tolerance

Value

Returns a list object containing the new estimate for beta (vector) and the number of iterations (scalar) the update step utilized

update_beta_isotonic *Beta update in L2E isotonic regression - PG*

Description

update_beta_isotonic updates beta for L2E isotonic regression using PG

Usage

```
update_beta_isotonic(y, b, tau, max_iter = 100, tol = 1e-04)
```

Arguments

y	Response vector
b	Current estimate for beta
tau	Current estimate for tau
max_iter	Maximum number of iterations
tol	Relative tolerance

Value

Returns a list object containing the new estimate for beta (vector) and the number of iterations (scalar) the update step utilized

---

`update_beta_MM_convex` *Beta update in L2E convex regression - MM*

Description

`update_beta_MM_convex` updates beta for L2E convex regression using MM

Usage

```
update_beta_MM_convex(y, beta, tau, max_iter = 100, tol = 1e-04)
```

Arguments

`y`	Response
`beta`	Initial vector of regression coefficients
`tau`	Precision estimate
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance

Value

Returns a list object containing the new estimate for beta (vector) and the number of iterations (scalar) the update step utilized

---

`update_beta_MM_isotonic`
Beta update in L2E isotonic regression - MM

Description

`update_beta_MM_isotonic` updates beta for L2E isotonic regression using MM

Usage

```
update_beta_MM_isotonic(y, beta, tau, max_iter = 100, tol = 1e-04)
```

Arguments

y	response
beta	initial vector of regression coefficients
tau	precision estimate
max_iter	maximum number of iterations
tol	relative tolerance

Value

Returns a list object containing the new estimate for beta (vector) and the number of iterations (scalar) the update step utilized

---

update_beta_MM_ls *Beta update in L2E multivariate regression - MM*

Description

update_beta_MM_ls updates beta for L2E multivariate regression using MM

Usage

```
update_beta_MM_ls(y, X, beta, tau, max_iter = 100, tol = 1e-04)
```

Arguments

y	Response
X	Design matrix
beta	Initial vector of regression coefficients
tau	Precision estimate
max_iter	Maximum number of iterations
tol	Relative tolerance

Value

Returns a list object containing the new estimate for beta (vector) and the number of iterations (scalar) the update step utilized

`update_beta_MM_sparse` *Beta update in L2E sparse regression - MM*

Description

`update_beta_MM_sparse` updates beta for L2E sparse regression using the distance penalty

Usage

```
update_beta_MM_sparse(
  y,
  X,
  beta,
  tau,
  k,
  rho,
  stepsize = 0.9,
  sigma = 0.5,
  max_iter = 100,
  tol = 1e-04
)
```

Arguments

`y`	Response vector
`X`	Design matrix
`beta`	Initial vector of regression coefficients
`tau`	Initial precision estimate
`k`	The number of nonzero entries in the estimated coefficients
`rho`	The parameter in the proximal distance algorithm
`stepsize`	The stepsize parameter for the MM algorithm (0, 1)
`sigma`	The halving parameter sigma (0, 1)
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance

Value

Returns a list object containing the new estimate for beta (vector) and the number of iterations (scalar) the update step utilized

update_beta_MM_TF *Beta update in L2E trend filtering regression - MM*

Description

update_beta_MM_TF updates beta in L2E trend filtering regression using the distance penalty

Usage

```
update_beta_MM_TF(y, X, beta, tau, D, k, rho, max_iter = 100, tol = 1e-04)
```

Arguments

y	Response vector
X	Design matrix
beta	Initial vector of regression coefficients
tau	Initial precision estimate
D	The fusion matrix
k	The number of nonzero entries in D*beta
rho	The parameter in the proximal distance algorithm
max_iter	Maximum number of iterations
tol	Relative tolerance

Value

Returns a list object containing the new estimate for beta (vector) and the number of iterations (scalar) the update step utilized

update_beta_qr *Beta update in L2E multivariate regression - PG*

Description

update_beta_qr updates beta for L2E multivariate regression via a QR solve

Usage

```
update_beta_qr(y, X, QRF, tau, b, max_iter = 100, tol = 1e-04)
```

Arguments

y	Response vector
X	Design matrix
QRF	QR factorization object for X (obtained via 'QRF=qr(X)')
tau	Current estimate for tau
b	Current estimate for beta
max_iter	Maximum number of iterations
tol	Relative tolerance

Value

Returns a list object containing the new estimate for beta (vector) and the number of iterations (scalar) the update step utilized

---

update_beta_sparse_ncv

Beta update in L2E sparse regression - NCV

Description

update_beta_sparse_ncv updates beta for L2E sparse regression using existing penalization methods

Usage

```
update_beta_sparse_ncv(
  y,
  X,
  beta,
  tau,
  lambda,
  penalty,
  max_iter = 100,
  tol = 1e-04
)
```

Arguments

y	Response vector
X	Design matrix
beta	Initial vector of regression coefficients
tau	Initial precision estimate
lambda	Tuning parameter
penalty	Available penalties include lasso, MCP and SCAD.
max_iter	Maximum number of iterations
tol	Relative tolerance

Value

Returns a list object containing the new estimate for beta (vector) and the number of iterations (scalar) the update step utilized

---

`update_beta_TF_lasso` *Beta update in L2E trend filtering regression using Lasso*

Description

`update_beta_TF_lasso` updates beta in L2E trend filtering regression using the Lasso penalty

Usage

```
update_beta_TF_lasso(y, X, beta, tau, D, lambda, max_iter = 100, tol = 1e-04)
```

Arguments

`y`	Response vector
`X`	Design matrix
`beta`	Initial vector of regression coefficients
`tau`	Initial precision estimate
`D`	The fusion matrix
`lambda`	The tuning parameter
`max_iter`	Maximum number of iterations
`tol`	Relative tolerance

Value

Returns a list object containing the new estimate for beta (vector) and the number of iterations (scalar) the update step utilized

---

`update_eta_bktk` *Eta update using Newton's method with backtracking*

Description

`update_eta_bktk` updates the precision parameter tau = e^eta for L2E regression using Newton's method

Usage

```
update_eta_bktk(r, eta, max_iter = 100, tol = 1e-10)
```

Arguments

r	Vector of residual
eta	Initial estimate of eta
max_iter	Maximum number of iterations
tol	Relative tolerance

Value

Returns a list object containing the new estimate for eta (scalar), the number of iterations (scalar) the update step utilized, the eta and objective function solution paths (vectors), and the first and second derivatives calculated via Newton's method (vectors)

---

update_tau_R *Tau update function*

Description

update_tau_R updates the precision parameter tau

Usage

```
update_tau_R(r, tau, sd_y, max_iter = 100, tol = 1e-10)
```

Arguments

r	Residual vector
tau	Current estimate for tau
sd_y	Standard deviation of y
max_iter	Maximum number of iterations
tol	Relative tolerance

Value

Returns a list object containing the new estimate for tau (scalar) and the number of iterations (scalar) the update step utilized

Index