Package ‘hdme’

May 20, 2019

Type     Package
Title    High-Dimensional Regression with Measurement Error
Version  0.3.1
Encoding UTF-8
Author   Oystein Sorensen
Maintainer Oystein Sorensen <oystein.sorensen.1985@gmail.com>
Description Penalized regression for generalized linear models for measurement error problems (aka. errors-in-variables). The package contains a version of the lasso (L1-penalization) which corrects for measurement error (Sorensen et al. (2015) <doi:10.5705/ss.2013.180>). It also contains an implementation of the Generalized Matrix Uncertainty Selector, which is a version the (Generalized) Dantzig Selector for the case of measurement error (Sorensen et al. (2018) <doi:10.1080/10618600.2018.1425626>.)
License GPL-3
LazyData TRUE
RoxygenNote 6.1.1
Imports glmnet (> 2.0-13), ggplot2 (> = 2.2.1), Rdpack, Rcpp (> = 0.12.15), Rglpk (> = 0.6-1),
URL https://github.com/osorensen/hdme
RdMacros Rdpack
Suggests knitr, rmarkdown, testthat, flare, dplyr, tidyR, covr
VignetteBuilder knitr
LinkingTo Rcpp, RcppArmadillo
NeedsCompilation yes
Repository CRAN
Date/Publication 2019-05-20 16:50:03 UTC
coef.corrected_lasso

R topics documented:

coef.corrected_lasso ........................................ 2
coef.gds .......................................................... 3
coef.gmus .......................................................... 3
coef.gmu_lasso ..................................................... 4
corrected_lasso .................................................. 4
cv_corrected_lasso ............................................. 6
gds ................................................................. 7
gmus ................................................................. 9
gmu_lasso .......................................................... 10
mus ................................................................. 12
plot.corrected_lasso ........................................... 13
plot.cv_corrected_lasso ......................................... 14
plot.gds ........................................................... 14
plot.gmus .......................................................... 15
plot.gmu_lasso .................................................... 16
print.corrected_lasso .......................................... 17
print.cv_corrected_lasso ........................................ 17
print.gds .......................................................... 18
print.gmus ........................................................ 18
print.gmu_lasso ................................................... 19

Index 20

coef.corrected_lasso Extract Coefficients of a Corrected Lasso object

Description

Default coef method for a corrected_lasso object.

Usage

## S3 method for class 'corrected_lasso'
coef(object, ...)

Arguments

object Fitted model object returned by corrected_lasso.

... Other arguments (not used).
 coef.gds      Extract Coefficients of a Generalized Dantzig Selector Object

Description

Default coef method for a gds object.

Usage

```r
## S3 method for class 'gds'
coef(object, all = FALSE, ...)
```

Arguments

- `object`: Fitted model object returned by `gds`.
- `all`: Logical indicating whether to show all coefficient estimates, or only non-zeros.
- `...`: Other arguments (not used).

 coef.gmus      Extract Coefficients of a GMUS object

Description

Default coef method for a gmus object.

Usage

```r
## S3 method for class 'gmus'
coef(object, all = FALSE, ...)
```

Arguments

- `object`: Fitted model object returned by `gmus`.
- `all`: Logical indicating whether to show all coefficient estimates, or only non-zeros. Only used when delta is a single value.
- `...`: Other arguments (not used).
coef.gmu_lasso | Extract Coefficients of a GMU Lasso object

Description

Default coef method for a gmu_lasso object.

Usage

```r
## S3 method for class 'gmu_lasso'
coef(object, all = FALSE, ...)
```

Arguments

- `object` Fitted model object returned by `gmu_lasso`.
- `all` Logical indicating whether to show all coefficient estimates, or only non-zeros.
- `...` Other arguments (not used).

corrected_lasso | Corrected Lasso

Description

Lasso (L1-regularization) for generalized linear models with measurement error.

Usage

```r
corrected_lasso(W, y, sigmaUU, family = c("gaussian", "binomial", "poisson"), radii = NULL, no_radii = NULL, alpha = 0.1, maxits = 5000)
```

Arguments

- `W` Design matrix, measured with error. Must be a numeric matrix.
- `y` Vector of responses.
- `sigmaUU` Covariance matrix of the measurement error.
- `family` Response type. Character string of length 1. Possible values are "gaussian", "binomial" and "poisson".
- `radii` Vector containing the set of radii of the l1-ball onto which the solution is projected. If not provided, the algorithm will select an evenly spaced vector of 20 radii.
- `no_radii` Length of vector radii, i.e., the number of regularization parameters to fit the corrected lasso for.
corrected_lasso

alpha Step size of the projected gradient descent algorithm. Default is 0.1.
maxits Maximum number of iterations of the project gradient descent algorithm for each radius. Default is 5000.

Details
Corrected version of the lasso for generalized linear models. The method does require an estimate of the measurement error covariance matrix.

Value
An object of class "corrected_lasso".

References

Examples
```r
# Example with linear regression
# Number of samples
n <- 100
# Number of covariates
p <- 50
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
# Measurement error covariance matrix
# (typically estimated by replicate measurements)
sigmaU <- diag(x = 0.2, nrow = p, ncol = p)
# Measurement matrix (this is the one we observe)
W <- X + rnorm(n, sd = diag(sigmaU))
# Coefficient
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Response
y <- X %*% beta + rnorm(n, sd = 1)
# Run the corrected lasso
fit <- corrected_lasso(W, y, sigmaU, family = "gaussian")
coef(fit)
plot(fit)
plot(fit, type = "path")

# Binomial, logistic regression
# Number of samples
n <- 1000
# Number of covariates
p <- 50
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
```
cv_corrected_lasso

Description
Cross-validated Corrected lasso

Usage
cv_corrected_lasso(W, y, sigmaUU, n_folds = 10, family = "gaussian",
                   radii = NULL, no_radii = 100, alpha = 0.1, maxits = 5000)

Arguments

W         Design matrix, measured with error.
y         Vector of the continuous response value.
sigmaUU   Covariance matrix of the measurement error.
n_folds   Number of folds to use in cross-validation. Default is 100.
family    Only "gaussian" is implemented at the moment.
radii     Optional vector containing the set of radii of the l1-ball onto which the solution
           is projected.
no_radii  Length of vector radii, i.e., the number of regularization parameters to fit the
           corrected lasso for.
alpha     Optional step size of the projected gradient descent algorithm. Default is 0.1.
maxits    Optional maximum number of iterations of the project gradient descent algo-
           rithm for each radius. Default is 5000.

Details
Corrected version of the lasso for the case of linear regression, estimated using cross-validation.
The method does require an estimate of the measurement error covariance matrix.

Value
An object of class "cv_corrected_lasso".
References


Examples

```r
# Gaussian
set.seed(100)
n <- 100; p <- 50 # Problem dimensions
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
# Measurement error covariance matrix
# (typically estimated by replicate measurements)
sigmaUU <- diag(x = 0.2, nrow = p, ncol = p)
# Measurement matrix (this is the one we observe)
W <- X + rnorm(n, sd = diag(sigmaUU))
# Coefficient
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Response
y <- X %*% beta + rnorm(n, sd = 1)
# Run the corrected lasso
cvfit <- cv_corrected_lasso(W, y, sigmaUU, no_radii = 5, n_folds = 3)
plot(cvfit)
print(cvfit)
# Run the standard lasso using the radius found by cross-validation
fit <- corrected_lasso(W, y, sigmaUU, family = "gaussian",
radii = cvfit$radius_min)
coef(fit)
plot(fit)
```

---

gds

**Generalized Dantzig Selector**

Description

Generalized Dantzig Selector

Usage

```r
gds(X, y, lambda = NULL, family = "gaussian", weights = NULL)
```

Arguments

- **X**: Design matrix.
- **y**: Vector of the continuous response value.
- **lambda**: Regularization parameter. Only a single value is supported.
family Use "gaussian" for linear regression, "binomial" for logistic regression and "poisson" for Poisson regression.

weights A vector of weights for each row of X.

Value

Intercept and coefficients at the values of lambda specified.

References


Examples

```r
# Example with logistic regression
n <- 1000 # Number of samples
p <- 10 # Number of covariates
X <- matrix(rnorm(n * p), nrow = n) # True (latent) variables # Design matrix
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5)) # True regression coefficients
y <- rbinom(n, 1, (1 + exp(-X %*% beta))^(-1)) # Binomially distributed response
fit <- gds(X, y, family = "binomial")
print(fit)
plot(fit)
coef(fit)

# Try with more penalization
fit <- gds(X, y, family = "binomial", lambda = 0.1)
coef(fit)
coef(fit, all = TRUE)

# Case weighting
# Assume we wish to put more emphasis on predicting the positive cases correctly
# In this case we give the 1s three times the weight of the zeros.
weights <- (y == 0) * 1 + (y == 1) * 3
fit_w <- gds(X, y, family = "binomial", weights = weights, lambda = 0.1)

# Next we test this on a new dataset, generated with the same parameters
X_new <- matrix(rnorm(n * p), nrow = n)
y_new <- rbinom(n, 1, (1 + exp(-X_new %*% beta))^(-1))
# We use a 50 % threshold as classification rule
# Unweighted classification
classification <- ((1 + exp(- fit$intercept - X_new %*% fit$beta))^(-1) > 0.5) * 1
# Weighted classification
classification_w <- ((1 + exp(- fit_w$intercept - X_new %*% fit_w$beta))^(-1) > 0.5) * 1

# As expected, the weighted classification predicts many more 1s than 0s, since
# these are heavily up-weighted
```
```
table(classification, classification_w)

# Here we compare the performance of the weighted and unweighted models.
# The weighted model gets most of the 1s right, while the unweighted model
# gets the highest overall performance.
table(classification, y_new)
table(classification_w, y_new)
```

---

**gmus**

*Generalized Matrix Uncertainty Selector*

**Description**

Generalized Matrix Uncertainty Selector

**Usage**

```
gmus(w, y, lambda = NULL, delta = NULL, family = "gaussian",
    weights = NULL)
```

**Arguments**

- `w` Design matrix, measured with error. Must be a numeric matrix.
- `y` Vector of responses.
- `lambda` Regularization parameter.
- `delta` Additional regularization parameter, bounding the measurement error.
- `family` "gaussian" for linear regression, "binomial" for logistic regression or "poisson" for Poisson regression. Defaults go "gaussian".
- `weights` A vector of weights for each row of `X`.

**Value**

An object of class "gmus".

**References**


Examples

```r
# Example with linear regression
set.seed(1)
n <- 100  # Number of samples
p <- 50  # Number of covariates
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
# Measurement matrix (this is the one we observe)
W <- X + matrix(rnorm(n*p, sd = 1), nrow = n, ncol = p)
# Coefficient vector
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Response
y <- X %*% beta + rnorm(n, sd = 1)
# Run the MU Selector
fit1 <- gmu(W, y)
# Draw an elbow plot to select delta
plot(fit1)
coef(fit1)

# Now, according to the "elbow rule", choose
# the final delta where the curve has an "elbow".
# In this case, the elbow is at about delta = 0.08,
# so we use this to compute the final estimate:
fit2 <- gmu(W, y, delta = 0.08)
# Plot the coefficients
plot(fit2)
coef(fit2)
coef(fit2, all = TRUE)
```

---

gmu_lasso

**Generalized Matrix Uncertainty Lasso**

**Description**

Generalized Matrix Uncertainty Lasso

**Usage**

```r
gmu_lasso(W, y, lambda = NULL, delta = NULL, family = "binomial",
active_set = TRUE, maxit = 1000)
```

**Arguments**

- `W` Design matrix, measured with error. Must be a numeric matrix.
- `y` Vector of responses.
- `lambda` Regularization parameter. If not set, lambda.min from glmnet::cv.glmnet is used.
delta  Additional regularization parameter, bounding the measurement error.
family  Character string. Currently "binomial" and "poisson" are supported.
active_set  Logical. Whether or not to use an active set strategy to speed up coordinate descent algorithm.
maxit  Maximum number of iterations of iterative reweighing algorithm.

Value

An object of class "gmu_lasso".

References


Examples

```r
set.seed(1)
# Number of samples
n <- 200
# Number of covariates
p <- 100
# Number of nonzero features
s <- 10
# True coefficient vector
beta <- c(rep(1, s), rep(0, p-s))
# Standard deviation of measurement error
sdu <- 0.2
# True data, not observed
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
# Measured data, with error
W <- X + sdu * matrix(rnorm(n * p), nrow = n, ncol = p)
# Binomial response
y <- rbinom(n, 1, (1 + exp(-X%*%beta))**(-1))
# Run the GMU Lasso
fit <- gmu_lasso(W, y, delta = NULL)
print(fit)
plot(fit)
coef(fit)
# Get an elbow plot, in order to choose delta.
plot(fit)
```
mus

Matrix Uncertainty Selector

Description

Matrix Uncertainty Selector for linear regression.

Usage

\[ \text{mus}(W, y, \lambda = \text{NULL}, \delta = \text{NULL}) \]

Arguments

- \( W \): Design matrix, measured with error. Must be a numeric matrix.
- \( y \): Vector of responses.
- \( \lambda \): Regularization parameter.
- \( \delta \): Additional regularization parameter, bounding the measurement error.

Details

This function is just a wrapper for \( \text{gmus}(W, y, \lambda, \delta, \text{family = "gaussian"}) \).

Value

An object of class "gmus".

References


Examples

```r
# Example with Gaussian response
set.seed(1)
# Number of samples
n <- 100
# Number of covariates
p <- 50
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
# Measurement matrix (this is the one we observe)
W <- X + matrix(rnorm(n*p, sd = 1), nrow = n, ncol = p)
```
# Coefficient vector
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Response
y <- X %*% beta + rnorm(n, sd = 1)
# Run the MU Selector
fit1 <- mus(W, y)
# Draw an elbow plot to select delta
plot(fit1)
coef(fit1)

# Now, according to the "elbow rule", choose the final delta where the curve has an "elbow".
# In this case, the elbow is at about delta = 0.08, so we use this to compute the final estimate:
fit2 <- mus(W, y, delta = 0.08)
plot(fit2) # Plot the coefficients
coef(fit2)
coef(fit2, all = TRUE)

---

**plot.corrected_lasso**

**plot.corrected_laso**

**Description**

Plot the output of corrected_lasso

**Usage**

```r
## S3 method for class 'corrected_lasso'
plot(x, type = "nonzero", ...)
```

**Arguments**

- `x`  
  Object of class corrected_lasso, returned from calling corrected_lasso()
- `type`  
  Type of plot. Either "nonzero" or "path". Ignored if length(x$radii) == 1, in case of which all coefficient estimates are plotted at the given regularization parameter.
- `...`  
  Other arguments to plot (not used)

**Examples**

```r
# Example with linear regression
n <- 100  # Number of samples
p <- 50  # Number of covariates
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
# Measurement error covariance matrix
# (typically estimated by replicate measurements)
sigmaUU <- diag(x = 0.2, nrow = p, ncol = p)
# Measurement matrix (this is the one we observe)
```
W <- X + rnorm(n, sd = diag(sigmaUU))
# Coefficient
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Response
y <- X %*% beta + rnorm(n, sd = 1)
# Run the corrected lasso
fit <- corrected_lasso(W, y, sigmaUU, family = "gaussian")
plot(fit)

plot.cv_corrected_lasso

Description
Plot the output of cv_corrected_lasso

Usage
## S3 method for class 'cv_corrected_lasso'
plot(x, ...)

Arguments
x The object to be plotted, returned from cv_corrected_lasso
...
Other arguments to plot (not used).

plot.gds

Description
Plot the number of nonzero coefficients at the given lambda.

Usage
## S3 method for class 'gds'
plot(x, ...)

Arguments
x An object of class gds
...
Other arguments to plot (not used).
**Examples**

```r
set.seed(1)
# Example with logistic regression
# Number of samples
n <- 1000
# Number of covariates
p <- 10
# True (latent) variables (Design matrix)
X <- matrix(rnorm(n * p), nrow = n)
# True regression coefficients
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Binomially distributed response
y <- rbinom(n, 1, (1 + exp(-X %*% beta))^(-1))
# Fit the generalized Dantzig Selector
gds <- gds(X, y, family = "binomial")
# Plot the estimated coefficients at the chosen lambda
plot(gds)
```

**Description**

Plot the number of nonzero coefficients along a range of delta values if delta has length larger than 1, or the estimated coefficients if delta has length 1.

**Usage**

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

**Arguments**

- `x` An object of class `gmus`
- `...` Other arguments to `plot` (not used).

**Examples**

```r
# Example with linear regression
set.seed(1)
# Number of samples
n <- 100
# Number of covariates
p <- 50
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
# Measurement matrix (this is the one we observe)
W <- X + matrix(rnorm(n*p, sd = 0.4), nrow = n, ncol = p)
```
# Coefficient vector
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Response
y <- X %*% beta + rnorm(n, sd = 1)
# Run the MU Selector
mus1 <- mus(W, y)
# Draw an elbow plot to select delta
plot(mus1)

# Now, according to the "elbow rule", choose the final
# delta where the curve has an "elbow".
# In this case, the elbow is at about delta = 0.08, so
# we use this to compute the final estimate:
mus2 <- mus(W, y, delta = 0.08)
# Plot the coefficients
plot(mus2)

plot.gmu_lasso  
"Plot the estimates returned by gmu_lasso"

Description
Plot the number of nonzero coefficients along a range of delta values if delta has length larger than
1, or the estimated coefficients of delta has length 1.

Usage
```r
## S3 method for class 'gmu_lasso'
plot(x, ...)
```

Arguments
- **x**  
  An object of class gmu_lasso
- **...**  
  Other arguments to plot (not used).

Examples
```r
set.seed(1)
n <- 200
p <- 50
s <- 10
beta <- c(rep(1,s),rep(0,p-s))
sdU <- 0.2
X <- matrix(rnorm(n*p),nrow = n,ncol = p)
W <- X + sdU * matrix(rnorm(n * p), nrow = n, ncol = p)
y <- rbinom(n, 1, (1 + exp(-X%*%beta))**(-1))
```
print.corrected_lasso

```r
gmu_lasso <- gmu_lasso(W, y)
plot(gmu_lasso)
```

---

### Print a Corrected Lasso object

**Description**

Default print method for a `corrected_lasso` object.

**Usage**

```r
## S3 method for class 'corrected_lasso'
print(x, ...)
```

**Arguments**

- `x` Fitted model object returned by `corrected_lasso`.
- `...` Other arguments (not used).

---

### Print a Cross-Validated Corrected Lasso object

**Description**

Default print method for a `cv_corrected_lasso` object.

**Usage**

```r
## S3 method for class 'cv_corrected_lasso'
print(x, ...)
```

**Arguments**

- `x` Fitted model object returned by `cv_corrected_lasso`.
- `...` Other arguments (not used).
print.gds  
*Print a Generalized Dantzig Selector Object*

**Description**

Default print method for a gds object.

**Usage**

```r
## S3 method for class 'gds'
print(x, ...)
```

**Arguments**

- `x`  
  Fitted model object returned by `gds`.
- `...`  
  Other arguments (not used).

---

print.gmus  
*Print a GMUS object*

**Description**

Default print method for a gmus object.

**Usage**

```r
## S3 method for class 'gmus'
print(x, ...)
```

**Arguments**

- `x`  
  Fitted model object returned by `gmus`.
- `...`  
  Other arguments (not used).
print.gmu_lasso

Print a GMU Lasso object

Description

Default print method for a gmu_lasso object.

Usage

```r
## S3 method for class 'gmu_lasso'
print(x, ...)
```

Arguments

- `x` Fitted model object returned by `gmu_lasso`.
- `...` Other arguments (not used).
Index

coef.corrected_lasso, 2
coef.gds, 3
coef.gmu_lasso, 4
coef.gmus, 3
corrected_lasso, 2, 4, 17
cv_corrected_lasso, 6, 17
gds, 3, 7, 18
gmu_lasso, 4, 10, 19
gmus, 3, 9, 18

mus, 12

plot.corrected_lasso, 13
plot.cv_corrected_lasso, 14
plot.gds, 14
plot.gmu_lasso, 16
plot.gmus, 15
print.corrected_lasso, 17
print.cv_corrected_lasso, 17
print.gds, 18
print.gmu_lasso, 19
print.gmus, 18